machine learning for modeling dynamic stochastic systems : application to adaptive control on deep-brain stimulation

Rémi Souriau

université
PARIS-SACLAY

Machine Learning for Modeling Dynamic Stochastic Systems: Application to Adaptive Control on Deep-Brain Stimulation

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication - STIC
Spécialité de doctorat: Traitement du Signal et des Images
Unité de recherche: Université Paris-Saclay, Univ Evry, IBISC, 91020, Evry-Courcouronnes, France.
Référent: Université d'Evry val d'Essonne

Thèse présentée et soutenue en visioconférence totale le 28 janvier 2021, par

Rémi SOURIAU

Composition du jury:

Florence Forbes Directeur de Recherche, INRIA Rhones	Présidente
Aurelia Fraysse MCU-HDR, Université Paris-Saclay	Examinatrice
Ali Mansour Professeur, ENSTA Bretagne	Rapporteur
Bertrand Rivet MCU-HDR, Grenoble-INP	Rapporteur
Vincent Vigneron MCU-HDR, Université Paris-Saclay	Directeur
Jean Lerbet Professeur, Université Paris-Saclay	Codirecteur
Eric Desailly Docteur, Fondation Ellen Poidatz	Invité
Hsin Chen Professeur, National Tsing Hua University	Invité

Thèse de doctorat

NNT: 2021UPASG004

To my family

Acknowledgements

This thesis has been realized in the Informatique, BioInformatique, Systèmes Complexes (IBISC) laboratory from the University of Evry Val-D'Essonne (UEVE). It was funded by the doctoral school Sciences et technologies de l'information et de la communication (STIC) of the Université Paris-Saclay. This work was carried out within the IBISC/SIAM team under the direction of Pr. Vincent Vigneron. This thesis has also being supervised by Pr. Jean Lerbet from the Laboratoire de Mathématiques et Modélisation d'Évry (LAMME) and Pr. Hsin Chen, director of the Neuro-Engineering Laboratory (NEL) from the National Tsing Hua University (國立清華大學) (NTHU) in Taiwan.

First, I would like to thank my PhD director: Professor Vincent Vigneron for the trust he has placed in me, for the regular support to my work especially when I was doubting (sometimes) about my work. Pr. Vigneron helps me to have more self-confidence. I also would like to thank my co-supervisor: Professor Jean Lerbet for every advise he gave me. Thank you to Doctor Jonathan Kobold, Professor Dominique Fourer and every member of the IBISC laboratory who participate from near and far to my project.

Second, I would like to thank Professor 陳新 (Hsin Chen) for his participation to the project and for welcoming me to his laboratory in Taiwan. This was an extraordinary experience I am very proud of and happy. Thanks also to 劉子顥 (Zi-Hau Liu) for welcoming me in the NEL and make me discover this wonderful country. Thanks to the Ministry of Science and Technology (MOST) for the financial support and the program to discover Taiwan's culture and many people. A special thanks also to Ramesh Perumal for his help in this work, for providing me animal experiment data related to the Parkinson's Disease.

Third, I would like to thank Dr. Eric Desailly and Dr. Omar Antonio Galarraga Castillo from the foundation Ellen Poidatz for the data on walk prediction application and the regular support.

Finally, I would like to thank the thesis jury who have agreed to participate to the thesis defense: Ali Mansour, Bertrand Rivet, Aurelia Fraysse and Florence Forbes.

List of contributions

First, Chapter 2 presents a review of this overview (summarized in Fig. 2.17). Second, the use of Continuous Restricted Boltzmann Machine to model time-series is studied in Chapter 3. Classifiers are proposed to learn and detect signals in an unsupervised way. These models were tested and evaluated on intracranial electroencephalography (iEEG) data from Parkisonian rats in Chapter 5. Results on iEEG data are compared with supervised approaches.

A major part of this thesis is dedicated to the study of diffusion networks which are capable to model stochastic differential equations. Multiple improvements on the neuron architecture, the graph and the learning procedures are presented in Chapter 4. Tests on toy data (in Chapter 4) and iEEG signals (in Chapter 5) have been performed to challenge their learning capacity.

List of publications

- Probit latent variables estimation for a Gaussian Process classifier. Application to the detection of High-Voltage Spindles. The 14th International Conference LVA/ICA, 2018.
- Boltzmann Machines for signals decomposition. Application to Parkinson's disease control. The 27th colloquium of GRETSI, communication paper, 2019.
- High-Voltage Spindles detection from EEG signals using recursive synchrosqueezing transform. The 27th colloquium of GRETSI, communication paper, 2019.
- A smart closed-loop deep-brain stimulation system. 6th Congress of the European Academy of Neurology - 1st Virtual Congress, 2020.
- A review on generative Boltzmann networks applied to dynamic systems. Mechanical Systems and Signal Processing, 2021.
- Continuous Restricted Boltzmann Machine for Unsupervised Signal Decomposition. Submitted in Neural Network, 2021.
- Signal modelization with Stochastic Differential Equations: the Diffusion Network. Under writing, 2021(?).

Contents

List of Figures

List of Tables

List of notations

- $\mathbb{R}$: real space
- $x(t)$: continuous time signal at instant t.
- $x[k]$: $k-$th discrete time signal sample.
- $[\![a, b]\!]$ set of integer between a and b.
- $\odot$: Hadamard product.
- $\boldsymbol{x}$: vector (bold text).
- ^{T}: transpose operator.
- $E[X] = \langle X \rangle$: expectation of the random variable X.
- v_i: $i-$th component of vector $\boldsymbol{v}$.
- W_{ij}: coefficient (row i, column j) of the matrix W.
- $\text{sgn}(\cdot)$: sign function.
- $\tanh(\cdot)$: hyperbolic tangent.
- $\sim$: "equivalent to".
- $\mathcal{N}(\mu, \Sigma)$: Gaussian law with mean vector μ and covariance matrix σ.
- $\mathcal{U}(D)$: Uniform law on D.
- $|\cdot|$: absolute value
- $\lfloor \cdot \rfloor$: floor function.
- $\langle \cdot, \cdot \rangle$: scalar product.
- $\mathrm{d}f$: differential operator.
- $\partial f / \partial x$: partial derivation.
- A^{-1}: Matrix A inverse.

Chapter 1

Introduction

1.1 Dynamic systems

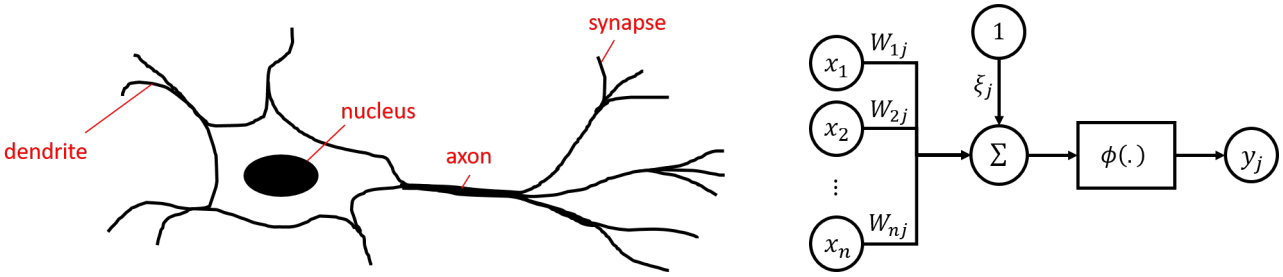
Modelling dynamic systems is a recurring problem in many field of studies. Generally speaking, a system is characterized by a set of variables. These variables are governed by laws and dependence relationships. A system is said dynamic if its variables are function of the time. From the movement of planets in space to the movement of electrons in an atom, through the propagation of a disease or the evolution of funded markets, dynamic systems are present in almost all real systems and are still widely studied today.

There are many different approaches to handle dynamic systems today. First, mathematical models based on the observation and experimental validation. This approach allows to write the evolution of variables over time as a *differential equation*. These models are very present in mechanics or electronics.

In automatic, the use of sensors allows to receive a constant flow of data, called *signals*, that must be processed and analyzed to make decisions. Numerous signal processing tools and techniques have been studied to interpret temporal signals. Fourier transform is, for example, one of the widely studied processing tools that converts temporal *data representation* into a frequency representation. Filtering is also one of the tools studied in signal processing to select useful information in the signals and remove the rest. There are also several proposed operators to analyze signals such as the convolution or the inter-correlation function. Finally, different prediction models have also been proposed in signal processing such as autoregressive models or the Kalman filter.

In dynamics, variables are linked by *causal relationships*. An event that takes place in the past has an influence on the events of the future. For example, a technical inspection not performed on a car increases the risk of having a car crash. Statistical models such as Bayesian networks or Markov chains model temporal information through a network where network nodes are variables of the system and links model causal relationships between variables. Past events have an influence on the future. This influence can be short-term but also long-term. This means an event can have consequences after a longer or shorter duration. In the example of the car, the risk of a car crash increases after the absence of a technical inspection. The more time passes, the greater the risk. To model a long-term causality relationship, it is necessary to be able to store *memory*. Memory can be modeled using additional variables that will synthesize past information.

The past recent years have been marked by the emergence of a large amount of database in many field. The creation of many database paves the way to new applications. Unfortunately, these data are becoming more and more complex and hard to interpret. In addition, many of the tools defined above are often outdated due to the complex properties of the data (non-linearity, large dimension, non-stationary, random variables, etc.). In order to answer the need for temporal data modeling, artificial neural networks are a promising solution particularly studied today.



(a) Biological neuron. (b) Artificial neuron.

Figure 1.1: Neuron structure. **(a)** is a simplified scheme of a biological neuron. The dendrite receive the information from other neurons, the nucleus process the information from input and the state of the nucleus is transmitted to other neurons thanks to the axon and synapses. **(b)** is the structure of the artificial neuron.

1.2 Bioinspired Neural Networks

Models of interest in this thesis are machine learning models. Machine learning refers to methods used to estimate a model from a set of data $\mathcal{D}$. This model is characterized by a configurable function f with a set of parameters $\mathbf{w} = (w_1, w_2, \ldots, w_p)^T$. Learning the model consists in finding the best set of parameters $\widehat{\mathbf{w}}$ allowing the function f to meet the needs of the application. To achieve this objective, a loss function $\mathcal{L}^{\mathbf{w}}$ is defined to ensure that:

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{L}^{\mathbf{w}}(\mathcal{D}) \tag{1.1}$$

The main difficulties in (1.1) are the choice of the loss function, the learning algorithm and how to design the network architecture. There is a large number of learning models today and many families of models sharing common properties have emerged from the literature.

An Artificial Neural Networks (ANN) is a bioinspired graphical learning model where the processing units (neurons) are organized and connected each other. A biological neuron scheme is given in Fig. 1.1a. The dendrites are connected to other neurons and receive the information from those neurons. The nucleus of the neuron process the information and update or not it state. The state of the neuron is then transmitted via the axon to other neurons thanks to the synapse connections. Fig. 1.1b gives the general structure of ANNs. First, inputs are weighted and added with a bias. Then the linear combination passes into a nonlinear function $\phi(.)$ which computes the neuron's state. This property allows the model to capture nonlinear features. Alternative structures have been proposed in the literature (see chapter 2).

An ANN is composed of visible neurons (or units) which contains the data. The success of ANNs lies into the use of hidden (or latent) neurons in addition to visible units. Hidden neurons allow the model to learn a new representation (called *latent* representation) of the data able to capture nonlinear and complex features. The latent variables have become a popular concept in machine learning. They have been highlighted in models such as Kalman Filter (KF) [44] or the Hidden Markov Model (HMM) [86, 21]. But contrary to ANN, latent variables in a KF and a HMM have a physical meaning that can be interpreted. There is no physical interpretation of the latent representation learned by an ANN. This is why they are named nonlinear nonparametric models.

ANNs can be categorized into multiple model families. Feedforward neural networks consitute a very popular class of supervised neural networks for classification or regression. These networks are based on the forward propagation of information from input neurons to output neurons and the back-propagation of the prediction error to learn the network. Boltzmann machines, also name *associative memories* are stochastic neural networks that can be used for unsupervised problem. This thesis is dedicated to the study of Boltzmann Machine.

Fig. 1.2 provides a scheme to locate the models studied in this thesis.

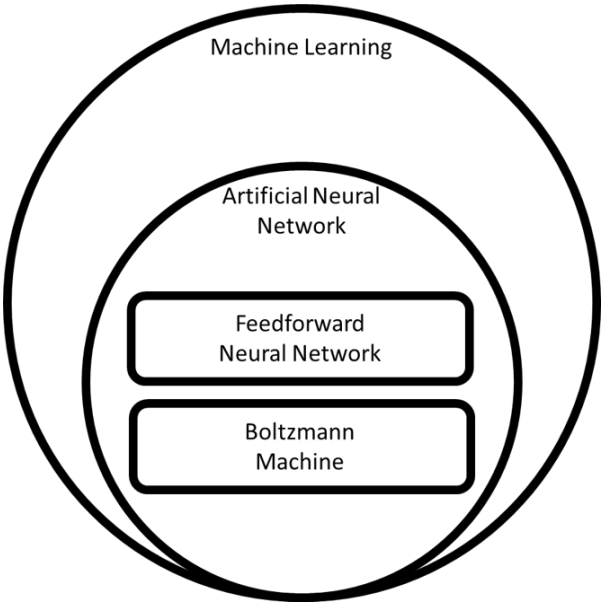


Figure 1.2: Scheme of contextualization.

1.3 Organization of the manuscript

The content of the thesis is organized as follows.

The following chapter is a review on generative neural network. It focuses on time series modeling and its applications. The aim of the review is to provide an overview of research on the Boltzmann machines, published in [102]. It provides answers to the following questions. What models inspired the Boltzmann Machine ? What are the extensions of the Boltzmann machine and what are the performances of these models ? This chapter introduces also the diffusion network which is closely related to Boltzmann Machine.

The third chapter presents some applications of the continuous restricted Boltzmann machine to model a time window. The convergence of the learning algorithm is analysed to understand the latent representation of the Boltzmann machine and a theorem is proposed to dimension the model. It is based on frequency decomposition theory. Finally, we proposed new unsupervised methods for signal detection using continuous restricted Boltzmann machines. Both methods are evaluated on toy data.

The fourth chapter is an original contribution to the stochastic dynamic modeling of diffusion networks inspired by Movellan [72, 73]. From the analyses of the weaknesses of the Movellan's model, several modifications to the model and the learning algorithm have been proposed. The performances on the multidimensional toy data with the diffusion network have been evaluated for two main tasks: signal prediction and channel reconstruction.

Chapter 5 presents first results of the application of the Boltzmann machines and diffusion network for the deep stimulation of Parkinsonian rat with the objective to control their symptoms. The animal experiment context is an important aspect for the validation of the thesis works. This application is the second major part of this thesis after the work on the generative neural networks. After the introduction of the experiments, the studied models have been evaluated again with EEG signals from the rat brain. These evaluations with data from a real problem aim to strengthen our confidence in the proposed models.

Finally the chapter 6 concludes this thesis and provides some highlights of the future works.

Chapter 2

Literature

Models of interest in this thesis are the continuous Restricted Boltzmann Machine and the Diffusion Network. In this chapter, a background on feedforward neural networks will first be given to introduce some basics about neural networks. Then, family of Boltzmann machines will be presented.

2.1 Background

2.1.1 Feedforward neural networks: a short presentation

FeedForward Neural Network (FFNN) have met a large success these recent years. This family of neural network is composed of one layer of input units (called the input layer), one output layer and one or more hidden layers. Neurons from the layer l take as input the state of the previous layer $l-1$. Then, the output of the layer l is propagated to the layer $l+1$. Fig. 2.1 is an example of feedforward neural network with two hidden layers. G. Cybenko [15] showed in 1989 a feedforward neural network with only one hidden layer sis capable to approximate any continuous function of compact subsets of $\mathbb{R}^n$. This theorem is known as *universal approximation theorem*. Note that the universal approximation theorem provides no information about the size of the network (the number of hidden units).

Each neuron has the structure given in Fig. 1.1b: inputs of a neuron are weighted with coefficient learned beforehand and all are summed with a bias (learned too). The result of the linear combination pass into an activation function (see Fig. 2.2 for three examples). This function plays a key role because it is responsible of the nonlinear behavior of the network and its learning capacities. Without the activation function, a FFNN is equivalent to a linear regression model. The first proposed neural network named perceptron was proposed with the sign activation function $\phi(X) = \text{sign}(X)$. Then more functions have been proposed like the sïgmoid function:

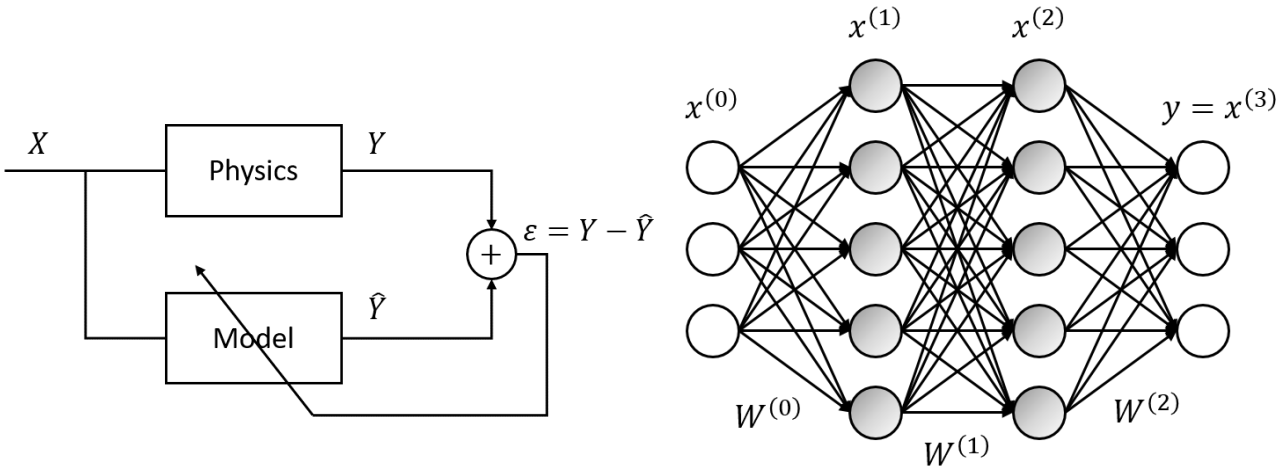
$$\phi(X) = \frac{1}{1 + \exp(-X)} \tag{2.1}$$

The sïgmoid function is strictly increasing and bounded between 0 and 1 (see Fig. 2.2a). There are many other functions used in the literature like the hyperbolic tangent (see Fig. 2.2b) or the Rectified Linear Unit (ReLU) function (see Fig. 2.2c):

$$\phi(X) = \max(0, X), \tag{2.2}$$

and many other extensions available in the literature (SeLU [47], elu or CeLU [2], etc.). The activation function of output neurons depends on the application. For example in the case of a regression problem the activation function of the output is the identity function. In the case of $n-$ary classification problem, the *softmax* function is used:

$$y_i = \frac{\exp(z_i)}{\sum_{j=1}^{\dim y} \exp(z_j)}, i = 1, \dots \dim y \tag{2.3}$$



(a) Principle of a FFNN.

(b) Classic graph of a FFNN.

Figure 2.1: Feedforward neural network. **(a)** is a scheme summarizing the principle of the model. FFNN is a supervised model: the error of prediction ϵ is used to learn the parameters of the model. **(b)** is a example of graph of the FFNN. The model is composed of three units in the input layer, two hidden layers with five hidden neurons each and one output layer with three units.

where y_i is the state of the $i-$th neurons of the output layer and z_i is the result of the linear combination of neuron i. The *softmax* function allows to get values of neurons bounded between zero and one and neuron outputs sum to 1.

Fig. 2.1 represents a FFNN (also named a Multi Layer Perceptron (MLP). This network is one of the most classic and common neural network. All the neurons from a layer are connected to the next one. Researchers proposed many graphical networks in the literature to learn various problems. The choice of the graph depends on many factors like the nature of data (image, time-series, etc.) or the targeted application (prediction, classification or control). Some structures derived from FFNN like the convolutional neural network (presented in next sections), the long short-term memory [36] or the generative adversarial network which has got a large interest the past recent years [26].

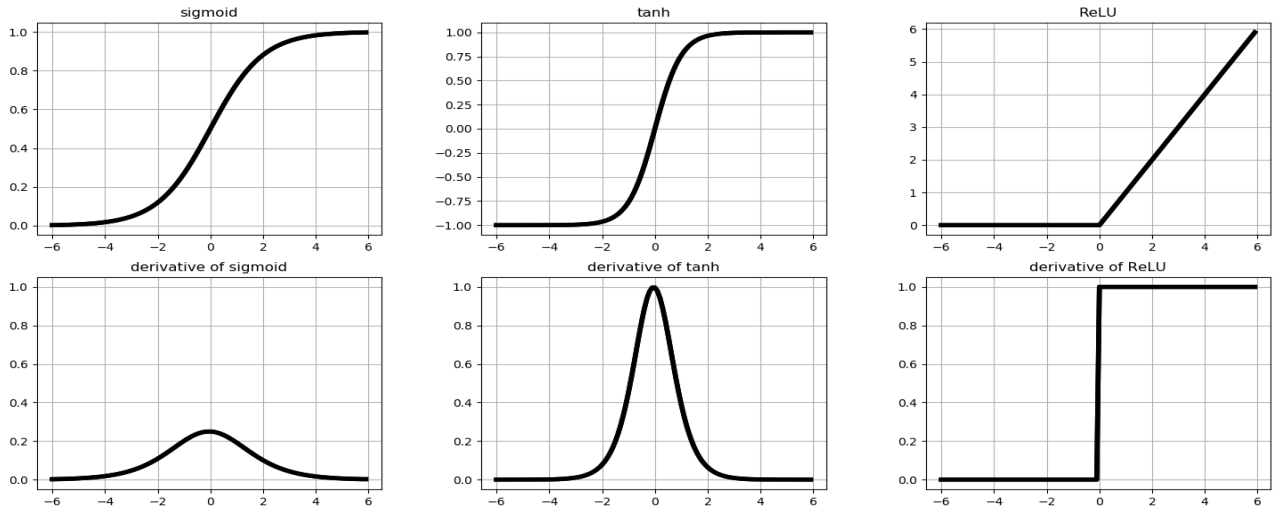
Let be the training database $\mathcal{D} = \{X^{(n)}, Y^{(n)}\}_{n\in[\![1,N]\!]}$ of N observations. $X^{(n)}$ is the $n-$th input observation vector and $Y^{(n)}$ is its associated output. λ is the set of parameters of the model. The training procedure of the model f is based on back-propagation. First, the input data is propagated from the input layer to the output layer. Second, the error of prediction is estimated between the result of the model $\widehat{Y^{(n)}} = f(X^{(n)};\lambda)$ and the expected result $Y^{(n)}$. The error of prediction is adapted to the nature of the output. For example the *mean square error* can be used with output units defined on $\mathbb{R}$

$$err_{\text{mse}} = \frac{1}{2}\sum_{n=1}^{N}\left\|\widehat{Y^{(n)}} - Y^{(n)}\right\|^2 \tag{2.4}$$

or the cross entropy for a binary output

$$err_{\text{ce}} = -\sum_{n=1}^{N}\left(Y^{(n)}\log(\widehat{Y^{(n)}}) + (1-Y^{(n)})\log(1-\widehat{Y^{(n)}})\right) \tag{2.5}$$

Once the error is computed, the error is back-propagated to compute the gradient of each parameter. Consider the case where the activation function of the output layer is linear and the error is the mean square error. The number of hidden layer is denoted L and the size of the $\ell-$th hidden layer is equal to d_ℓ.



(a) Sigmoid. (b) Tangent hyperbolic. (c) ReLU.

Figure 2.2: Activation functions. Upper row: the sigmoid **(a)**, the tangent hyperbolic **(b)** and the ReLU **(c)**. Lower row: their derivative.

d_o is the size of the output layer. The error measure can be decomposed into the sum of the error for each observation in training set:

$$err = \sum_{n=1}^{N} err^{(n)} \tag{2.6}$$

Now consider the evaluation of the derivative of $err^{(n)}$ with respect to the coefficient $W_{ji}^{(\ell)}$ where ℓ refers to the $\ell-$th layer of the FFNN. As presented in Fig. 1.1b, a linear combination is computed at the neuron level:

$$z_j^{(\ell)} = \sum_{i=1}^{d_\ell} W_{ji}^{(\ell)} x_i^{(\ell)} \tag{2.7}$$

and the result is passed into the activation function:

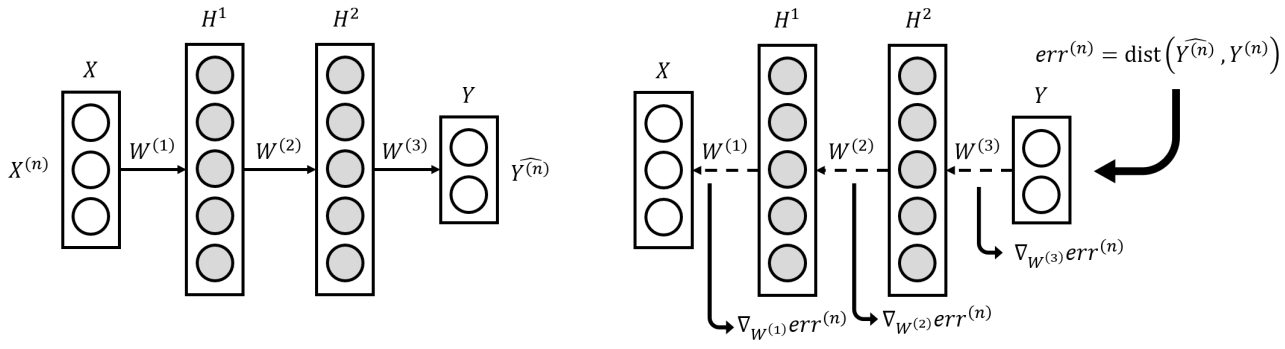
$$x_j^{(\ell+1)} = \phi\left(z_j^{(\ell)}\right). \tag{2.8}$$

In (Eq. 2.7), the bias can be included in the transfer matrix by adding a column index $i = 0$ and set $x_0^{(\ell)} = 1$. The chain rule for partial derivative is applied to compute the derivative of the error for each parameters:

$$\frac{\partial err^{(n)}}{\partial W_{ji}^{(\ell)}} = \frac{\partial err^{(n)}}{\partial z_j^{(\ell)}} \times \frac{\partial z_j^{(\ell)}}{\partial W_{ji}^{(\ell)}} \tag{2.9}$$

Let $\delta_j^{(n,\ell)} = \frac{\partial err^{(n)}}{\partial z_j^{(\ell)}}$ the "back-propagation error" to neuron j from the $\ell-$th layers. Then, (Eq. 2.9) can be written:

$$\frac{\partial err^{(n)}}{\partial W_{ji}^{(\ell)}} = \delta_j^{(n,\ell)} \times x_i^{(\ell)} \tag{2.10}$$



(a) Propagation.

(b) Back-propagation.

Figure 2.3: Illustration of the back-propagation algorithm. First the input is propagated through the hidden layers to the output layer. Then, the error of prediction is back-propagated to compute the gradient of each transfer matrix.

The coefficient $\delta_j^{(\ell)}$ from layer ℓ can be decomposed using the estimation error from the next layer:

$$\delta_j^{(\ell)} = \phi'(z_j^{(\ell)}) \sum_{i=1}^{d_{\ell+1}} \delta_i^{(\ell+1)} W_{ji}^{(\ell)} \tag{2.11}$$

where ϕ' is the derivative of the activation function. So, to compute the derivative of each parameter, the back-propagation error from the output layer to the layer of interest has to be computed first. And then, it is possible to compute the gradient for all parameters. Fig. 2.3 summarizes the back-propagation algorithm.

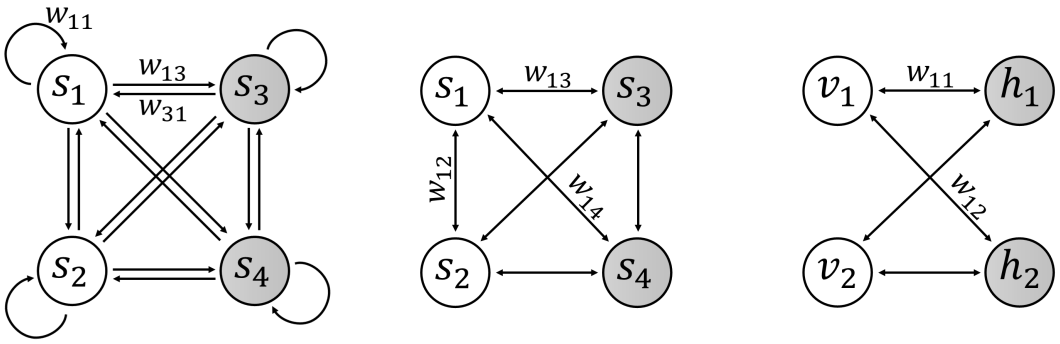
The main drawback of the back-propagation algorithm is the presence of the term $\phi'(z_j^{(\ell)})$ in (Eq. 2.11). The more hidden layers in the neural network the more the back-propagation error is multiplied by a number of derivatives the of activation function. Back in Fig. 2.2, it is easy to note that the back-propagation error will decrease for each layer. For example with the sïgmoid function, the error is multiplied by a coefficient bounded between 0 and 0.25. The error becomes then negligible after some hidden layers and it becomes very hard to learn the first layers. The choice of the activation function is then very important to reduce the decrease of the back-propagation error. In practice, ReLU function is widely used in deep learning due to its derivative (see Fig. 2.2c).

For a given learning task, building a network requires to choose the network *topology*, *i.e.* the number of inputs, the number of layers, the connectivity, the activation function, etc.

2.1.2 Generative Boltzmann Networks

Another family of ANN widely studied in the literature is the family based on Boltzmann machine, sometimes called recurrent neural networks. But the recurrent neural network refers also to an extension of dynamic FFNN. In this thesis, the family based on Boltzmann machine will be named the family of Generative Boltzmann Network (GBN). Contrary to FFNNs, which are *discriminative* models, GBNs are *generative* models. A discriminative model focus on the modeling of the conditional expectancy $E[output|input]$ and a generative model focus on the modeling on the joint probability $\Pr(data)$. GBNs are *unsupervised* models. There is no need of "output" data to learn a GBN. These properties are needed in particular in applications where output data are not available.

In a GBN, the state of each neuron is updated according to the previous state of the network until the global state of the network becomes stable. This state is called the *equilibrium* state. A full graph of a GBN is given in Fig. 2.4a. The coefficient W_{ij} refers to the weight of the link from the neuron i to the



(a) Full graph (b) Symmetric graph (c) Restricted graph

Figure 2.4: Graphs with 4 neurons. White neurons are visible neurons and gray neurons are hidden neurons. Double direction arrow means $W_{ij} = W_{ji}$, else $W_{ij} \neq W_{ji}$. In **(a)** and **(b)**, the link W_{ij}. In **(c)**, the coefficient W_{ij} refers to the weight of the symmetric link between the $i-$th visible unit v_i and the $j-$th hidden unit h_j.

neuron j. In a network with n neurons, let note $W = (W_{ij})_{1 \leq i,j \leq n}$ the transfer matrix between all the neurons of the network and let note $\boldsymbol{\xi} = (\xi_i)^T_{1 \leq i \leq n}$ the bias vector (not represented in Fig. 2.4). There are two kinds of neurons: visible neurons and hidden neurons. The visible neurons are the observations and the hidden neurons are the *latent* information. The hidden neurons are trained to capture and synthesize the data features. The latent information refers to the data features unknown by the user.

Further sections in this chapter are dedicated to the presentation of the GBN's family.

2.1.3 The father of Boltzmann Machine: Hopfield Network

In 1982 and 1984, Hopfield presented the first two deterministic and generative neural networks, respectively, the binary Hopfield Network (bHN) [37] and the continuous Hopfield Network (cHN) [38]. His work on deterministic networks later inspired many research works on probabilistic networks. Both bHN and cHN have symmetric connections between neurons ($W_{ij} = W_{ji}$) and no feedback link ($W_{ii} = 0$). Fig. 2.4b gives the graph of a Hopfield Network (HN) and Fig. 2.5 gives the two neuron structures of bHN and cHN. HNs has been used in different research fields like image processing for noise reduction in [80] or more recently for super-resolution images [58] or in economics in [81].

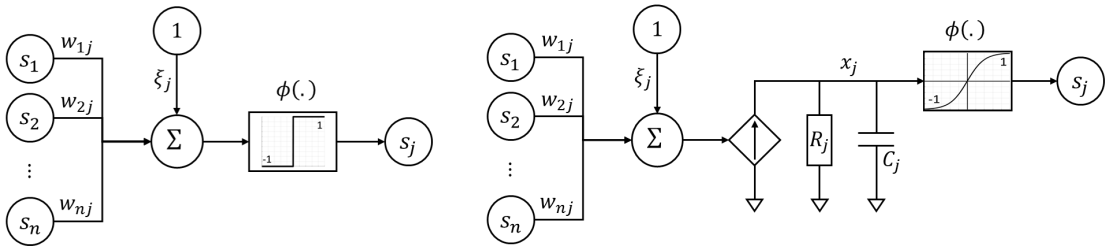
The bHN employs binary neurons ($s_j = +1$, or, $s_j = -1$). The weighted inputs and the bias pass into the activation function called the *sign* function to provide a binary output:

$$s_j = \text{sgn}\left(\xi_j + \sum_{i=1}^{n} W_{ij} s_i\right). \tag{2.12}$$

The energy function of the bHN is:

$$E_{bHN}(\boldsymbol{s}) = -\sum_{\substack{1 \leq i,j \leq n \\ i \neq j}} W_{ij} s_i s_j - \sum_{i=1}^{n} \xi_i s_i. \tag{2.13}$$

The main limitation of the binary behavior of these neurons is its capacity of data representation. For instance with n neurons, a bHN can encode 2^n different states. Continuous neurons increase significantly



(a) bHN (b) cHN

Figure 2.5: HNs' neuron structure. $\phi(.)$ is the activation function: the *sign* function in **(a)** and the *sigmoïd* function in **2.5b**. In **(b)** R_j and C_j are, respectively, the resistor and capacitor of neuron j. There is no reason that R_j and C_j are the same for all neurons. See section 2.4 for more details.

this capacity of representation with only few neurons. In addition, data are barely binary. The benefit of transitioning from a binary network to a continuous network is then double.

In 1984, Hopfield proposed as a generalization of bHN the cHN to get closer to the biological neuron behavior. He proposed a new architecture for working with continuous neurons varying continuously over time. The structure of the neuron j is given in Fig. 2.5b. The activation function is now sigmoïd function. The state space of the network became a $n-$dimensional hypercube bounded with the variation range of the activation function. The weighted input is converted into a current and passes an electronic RC filter. X_j is the input voltage of $\phi(.)$. The variation of $X_j(t)$ (see Fig. 2.5b) is given by the following differential equation:

$$C_j \frac{\mathrm{d}X_j(t)}{\mathrm{d}t} = -\frac{X_j(t)}{R_j} + \xi_j + \sum_{i=1}^{n} W_{ij}\phi(X_i(t)). \tag{2.14}$$

The state of the cHN tends to an *equilibrium* state characterized by the energy function:

$$E_{cHN}(\boldsymbol{s}) = -\sum_{\substack{1\leq i,j\leq n \\ i\neq j}} W_{ij}s_i s_j - \sum_{i=1}^{n}\xi_i s_i + \sum_{i=1}^{n}\frac{1}{R_i}\int_0^{s_i}\phi^{-1}(s')\mathrm{d}s'. \tag{2.15}$$

The energy function in (Eq. 2.15) introduced by Hopfield is a Lyapunov function. A Lyapunov function is a scalar, positive definite function. The gradient vector of the function is negative around the solution. E_{cHN} is the sum of E_{bHN} in (Eq. 2.13) and an integral term due to the continuous behavior of neurons.

The deterministic nature of artificial neurons limits their functioning. First, the evolution of the state $\boldsymbol{s}$ tends to decrease the energy, until the the equilibrium state corresponding to the lowest possible energy is reached. As many local minima are present in the energy landscape, the state of the network converges to a local minima. The second limitation is the risk of overfitting during the training step: a training procedure tends to decrease as much as possible the energy function over a training database. Not stopping the training procedure of these models will result a over specialization of these models to fit exclusively with the training database and not anymore with the validation set.

2.2 Boltzmann Machines

2.2.1 Description

Introduced by Fahlman and Hinton in 1983 [23], the Boltzmann Machine (BM) is the *stochastic* extension of the bHN. A stochastic model is a model involving random processes. These are in many cases where stochastic models are more adapted than *deterministic* model to fit with real world application where

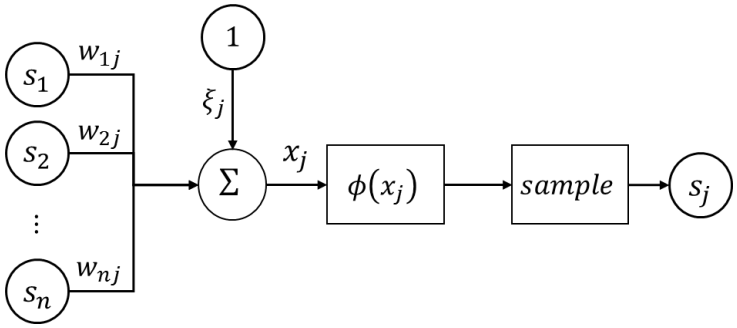


Figure 2.6: Structure of the neuron j of a BM. $\phi(X_j) = \Pr(s_j = 1|\boldsymbol{s})$

.

random processes are present. Each neuron can be seen as a Bernoulli random variable. Like in the bHN, the BM is composed of visible and hidden units. The neuron structure in a BM is depicted in Fig. 2.6 where s_j is the state of the neuron j ('0' or '1').

For each neuron, the result of the linear combination of the inputs is thresholded by an activation function $\phi(.)$ such as the sigmoïd function:

$$\phi(X_j) = \frac{1}{1 + \exp(-X_j)}. \tag{2.16}$$

As this function is bounded between 0 and 1, the result of $\phi(X_j)$ can be interpreted as the probability $\Pr(s_j = 1|\boldsymbol{s})$, *i.e.* the probability that $s_j = 1$ given the previous state of the network. The 'sample' step in Fig. 2.6 consists in sampling an uniform random variable $u_j \sim \mathcal{U}_{[0,1]}$ and the state s_j depending on the inequality:

$$s_j = \begin{cases} 1, & \text{if } u_j < \Pr(s_j = 1|\boldsymbol{s}), \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

The sampling step in (Eq. 2.17) is at the origin of the stochastic behavior of the network (to be compared with (Eq. 2.12)). At a given time, even if the expected state of a neuron is fixed to '0' or '1' by the state of the network itself, there is always a possibility that the state of the neuron will change. The probability of overfitting is then reduced.

The BM is an energy based model. The energy of a BM is similar to the energy function of the bHN (see (Eq. 2.13)): $E_{BM}(\boldsymbol{s}) = E_{cHN}(\boldsymbol{s})$. The associated joint probability $P_{BM}(\boldsymbol{s})$ is defined as:

$$P_{BM}(\boldsymbol{s}) = \frac{1}{Z} \exp\left(-E_{BM}(\boldsymbol{s})\right), \tag{2.18}$$

where Z is a marginalization constant so that P_{BM} is a probability distribution function, *i.e.*

$$Z = \sum_{\boldsymbol{s} \in [\![\mathbf{0},\mathbf{1}]\!]^{\boldsymbol{n}}} \exp\left(-E_{BM}(\boldsymbol{s})\right) \tag{2.19}$$

Z is also called the *partition function*.

2.2.2 Restricted Boltzmann Machine

Fig. 2.4b represents the architecture of a BM, each neurons being updated independently. For a large network, update neurons sequentially can be time consuming. Smolensky proposed in 1986 the Harmonium model [98], known, today, as the Restricted Boltzmann Machine (RBM). The graph structure of the RBM in Fig. 2.4c is a bi-partite BM with two layers: the visible layer $\boldsymbol{v} \in [\![0,1]\!]^n$ and the hidden layer $\boldsymbol{h} \in [\![0,1]\!]^m$. Neurons from the same layer are not connected to each other. For a given state of one layer, neurons from

the second layer are each other conditionally independent and can be updated at once. Removing links between neurons reduces the complexity of the model. To balance this restriction on the network structure, neurons or layers can be added (see section 2.3.1). We note $W \in \mathbb{R}^{(n \times m)}$ the *transfer* matrix between the two layers and $\boldsymbol{\xi^v}$ and $\boldsymbol{\xi^h}$ the bias vectors of, respectively, the visible layer and the hidden layer. The energy expression of a RBM and of a BM are the same but thanks to the absence of intra-layer links, the energy function of the BM can be simplified into (Eq. 2.20):

$$E_{RBM}(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^T W \boldsymbol{h} - \boldsymbol{v}^T \boldsymbol{\xi^v} - \boldsymbol{h}^T \boldsymbol{\xi^h}. \tag{2.20}$$

Like for BM, a joint probability distribution function is defined for the RBM and the marginalized probability distribution over visible units as:

$$P_{RBM}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E_{RBM}(\boldsymbol{v}, \boldsymbol{h})\right), \tag{2.21}$$

where Z is the normalization constant $Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\left(-E_{RBM}(\boldsymbol{v}, \boldsymbol{h})\right)$. Hidden units are not accessible in the training database. Compute the energy function in (Eq. 2.20) requires to sample the hidden units. The free energy $F_{RBM}(\boldsymbol{v})$ is an energy function where hidden state are marginalized [64]. The *free energy* is defined as:

$$P_{RBM}(\boldsymbol{v}) = \frac{1}{Z} \sum\nolimits_{\boldsymbol{h}} \exp\left(-E_{RBM}(\boldsymbol{v}, \boldsymbol{h})\right) = \frac{1}{Z} \exp\left(-F_{RBM}(\boldsymbol{v})\right), \tag{2.22}$$

where

$$F_{RBM}(\boldsymbol{v}) = -\boldsymbol{v}^T \boldsymbol{\xi^v} - \sum_{j=1}^{m} \log\left(1 + \exp(\boldsymbol{v}^T W(:, j) + \xi_j^h)\right), \tag{2.23}$$

with $W(:, j)$ the $j-$th column vector of W. Training a RBM is performed in minimizing the energy function (Eq. 2.20) or the free energy function (Eq. 2.23). In the first case, hidden neurons are involved into the learning algorithm. In the second case, hidden neurons are marginalized. Different learning procedures of BMs has been explored in the literature.

2.2.3 Learning procedures for Boltzmann Machines

A training set $\mathcal{D} = \{\boldsymbol{v^k}\}_{1 \leq k \leq N}$ composed of N observations is used to find the best set of parameters $\mathcal{P} = \{W, \boldsymbol{\xi}\}$, $\boldsymbol{\xi}$ regrouping visible and hidden bias vectors. Let $P^0(\boldsymbol{v})$ be the probability distribution over the training set, $P^\infty(\boldsymbol{v})$ be the probability distribution of the data corresponding to the equilibrium state and $P^q(\boldsymbol{v})$ be the probability distribution after q steps of reconstruction (a reconstruction step corresponds to a *two-way sampling* between the visible and the hidden layers: $\boldsymbol{h}$ sampled from $\boldsymbol{v}$ and $\boldsymbol{v}$ sampled from $\boldsymbol{h}$). Finally, $\boldsymbol{s} = (\boldsymbol{v}, \boldsymbol{h})^T$ is the set of all units of the model.

Different methods to train BMs have been proposed in the literature. In this section, the minimization of the Contrastive Divergence (CD) (see [29, 4]) is presented as well as other methods.

Training Boltzmann Machines using the Contrastive Divergence

An intuitive idea to train the BM consists in maximizing the joint log-likelihood:

$$\mathcal{L}^{ML}(\mathcal{P}|\mathcal{D}) = \sum_{k=1}^{N} \log P_{RBM}(\boldsymbol{v^k}). \tag{2.24}$$

where "ML" refers to Maximum Likelihood.

Working directly on (Eq. 2.24) is difficult due to the presence of the constant Z. The minimization of the Kullback-Leibler (KL) divergence noted G [49] between $P^0(\boldsymbol{v})$ and $P^\infty(\boldsymbol{v})$ is equivalent to the maximization of the log-likelihood and bypass the need to calculate Z. $P^0(\boldsymbol{v})$ is the distribution function

of the visible units over the training set and $P^{\infty}(\boldsymbol{v})$ is the distribution function of the visible units after the model converges to the equilibrium state. The derivation of G with respect to the parameters leads to the following update rule for each parameter λ_i [34]:

$$\frac{\partial G}{\partial \lambda_i} = \left\langle \frac{\partial E_{BM}(\boldsymbol{s})}{\partial \lambda_i} \right\rangle_0 - \left\langle \frac{\partial E_{BM}(\boldsymbol{s})}{\partial \lambda_i} \right\rangle_{\infty}, \tag{2.25}$$

where $\langle . \rangle_0$ is the expectation over the training set at the initial time and $\langle . \rangle_{\infty}$ is the expectation value at the equilibrium state.

More specifically, the update law of the weight W_{ij} between neurons i and j and the update law bias of neuron i are:

$$\begin{cases} \Delta W_{ij} \propto (< s_i s_j >_0 - < s_i s_j >_{\infty}) \\ \Delta \xi_i \propto (< s_i >_0 - < s_i >_{\infty}) \end{cases} \tag{2.26}$$

Updating the parameters requires to compute at each iteration the equilibrium distribution which means a large number of calculus. To reduce the loss function, the minimization of the KL-divergence is replaced by the Minimization of the Contrastive Divergence (MCD) [29] which consists in minimizing the contrast D between two KL-divergences:

$$D = KL(P^0(\boldsymbol{v}), P^{\infty}(\boldsymbol{v})) - KL(P^q(\boldsymbol{v}), P^{\infty}(\boldsymbol{v})). \tag{2.27}$$

where: $P^q(\boldsymbol{v})$ is the distribution function of the visible units after q steps of Gibbs sampling.

The only case where D is zero is the case when $P^0(\boldsymbol{v}) = P^q(\boldsymbol{v})$, *i.e.* when after q steps of reconstruction the probability distribution does not change, then the RBM is already stable and $P^0(\boldsymbol{v}) = P^{\infty}(\boldsymbol{v})$. Update rules become:

$$\frac{\partial D}{\partial \lambda_i} = \left\langle \frac{\partial E_{BM}(\boldsymbol{s})}{\partial \lambda_i} \right\rangle_0 - \left\langle \frac{\partial E_{BM}(\boldsymbol{s})}{\partial \lambda_i} \right\rangle_q, \tag{2.28}$$

with $\langle . \rangle_q$ the expectation after q reconstruction steps. In the case of RBMs with binary values, note that $\langle s_i \rangle_q$ refers to the probability of $s_i = 1$ after q iterations (even, if, in practice, $q = 1$). Markov Chain Monte Carlo (MCMC) method is used with the sampling rule (Eqs. 2.16-2.17), to estimate $\langle s_i \rangle_q$. The update equations for the transfer matrix coefficient W_{ij} and the bias ξ_i at each iteration can be deduced from (Eq. 2.28):

$$\begin{cases} \Delta W_{ij} \sim \langle s_i s_j \rangle_0 - \langle s_i s_j \rangle_1 , \\ \Delta \xi_i \sim \langle s_i \rangle_0 - \langle s_i \rangle_1 . \end{cases} \tag{2.29}$$

To conclude, minimizing the KL-divergence can be replaced by the MCD rule to speed up the training. Note that if the energy function is used with explicit values of neurons like in (Eq. 2.29), the learning algorithm requires to sample the hidden units to update the parameters of the model.

Carreira-Perpinan [6], MacKay [62], Williams [114] and Yuille [119] studied properties of the CD and demonstrated MCD rule converges to the optimal solution with a small bias.

Training based on free energy

To avoid the approximation of the CD by sampling hidden units, different training methods using the free energy ((Eq. 2.23)) have been proposed to get a better approximation than the log-likelihood in (Eq. 2.24). The main advantage is the free energy does not require to be minimized with respect to the hidden variables. Marlin *et al.* in [64] review various scores based on free energy to train BMs.

Younes [118] and Tieleman [106] proposed to simulate the Markov chain with only one step to estimate the probability distribution $P_{BM}(\boldsymbol{s})$ and to update the parameters with a small learning rate to maximize $\mathcal{L}^{ML}(\mathcal{P}|\mathcal{D})$ given (Eq. 2.24). The Constrastive Divergence [29] can also be used to learn the parameters of

a BM by replacing the energy function in (Eq. 2.28) with the free energy function. The Ratio Matching proposed by Hyvärinen [43] consists in minimizing the Euclidean distance between the conditional probability of each component of $v_i^k = 1$ and $v_i^k = 0$ given $\boldsymbol{v^{k\backslash i}}$, the visible layer without the $i-$th component:

$$\mathcal{L}^{RM}(\mathcal{P}|\mathcal{D}) = \sum_{k=1}^{N}\sum_{i=1}^{n}\sum_{\epsilon\in[\![0,1]\!]} P^{\infty}(\boldsymbol{v^k})\left(P^0(v_i^k = \epsilon|\boldsymbol{v^{k\backslash i}}) - P^{\infty}(v_i^k = \epsilon|\boldsymbol{v^{k\backslash i}})\right)^2. \tag{2.30}$$

The Generalized Score Matching proposed by Lyu in [61] and improved by Marlin in [64] consists in maximizing the difference of the inverses of the conditional probabilities of a visible unit v_i^k given $\boldsymbol{v^{k\backslash i}}$ known over the training set. The function to maximize is given by:

$$\mathcal{L}^{GSM}(\mathcal{P}|\mathcal{D}) = \sum_{k=1}^{N}\sum_{i=1}^{n} P^{\infty}(\boldsymbol{v^k})\left(\frac{1}{P^0(v_i^k|\boldsymbol{v^{k\backslash i}}))} - \frac{1}{P^{\infty}(v_i^k|\boldsymbol{v^{k\backslash i}}))}\right)^2. \tag{2.31}$$

Marlin [64] concludes in his paper on the *inductive principles* for RBM that the stochastic maximum likelihood and the CD are well suited in many situations (applications, computation time, ...).

Research on training algorithm for RBM remains active today. In Montavon's paper [71], the Wasserstein distance [109] is used instead of the KL divergence to train Wasserstein Restricted Boltzmann Machine (WRBM). Kuleshov proposed to use variational inference techniques to train RBM [48]. Finally, Fisher introduced in [24] the Boltzmann Encoded Adversarial Machine (BEAM) which consists in adding an adversarial term in the loss function.

2.3 Extensions of Boltzmann Machines

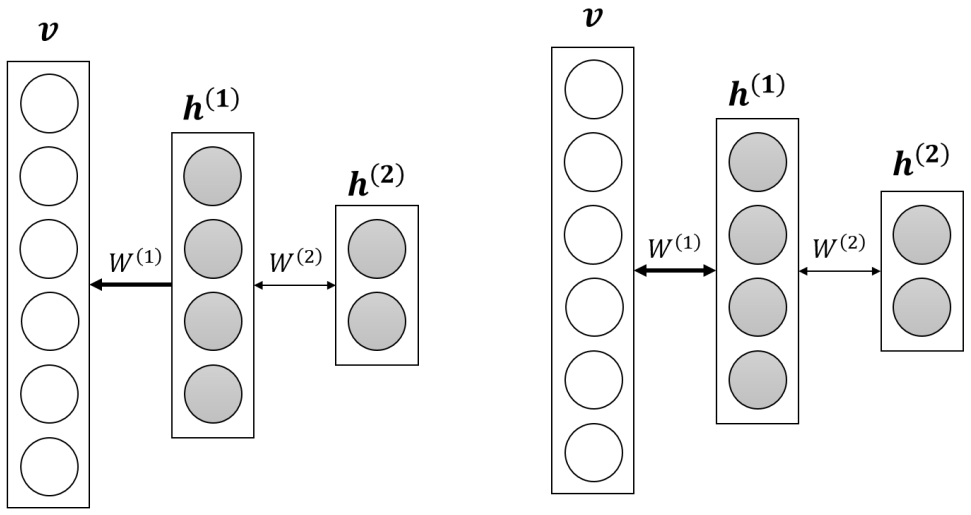
One of the major strength of the RBM lies in its flexible architecture which makes it suitable for many different applications. There exists a lot of versions of the RBM, for instance some papers propose to modify the original BM with a ReLU activation function [74] or to fuzzified the parameters of the model as in the Fuzzy Restricted Boltzmann Machine (FRBM) [7]. In this section some well-known extensions of the RBM are presented.

2.3.1 Multi-layers Boltzmann Machine

In practice, RBM are preferred to fully connected BM to model complex systems because of RBM update rule. However, the absence of inter-layer links in the RBM reduces the capacity of the model to capture complex features. A simple solution to overcome this limit is to add additional hidden layers to capture high-order information. Two RBMs with multiple hidden layers have been proposed in the literature: the Deep Belief Network (DBN) and the Deep Boltzmann Machine (DBM).

Deep Belief Network

In FFNNs, the information is propagated from the input layer to the output layer and the error of prediction output is back-propagated [28] to correct the parameters of the network by minimizing the loss function. Back-propagation algorithm proposed by Sejnowski and Lecun [54] has however some well-known limitations: the training requires labeled data, it is hard to learn parameters (see Sec. 2.1.1). The DBN introduced in 2006 by Hinton [35, 30] has been proposed to pretrain deep neural network. The DBN can be seen as stacked RBMs (see Fig. 2.7a), each RBMs being trained independently. Let $\boldsymbol{h^{(i)}}$ be the $i-$th hidden layer. The first step consists in learning $W^{(1)}$ between the visible and the first hidden layer $\boldsymbol{h^{(1)}}$. The observations from the training set are sampled into the first hidden layer and will constitute a new training set for the next RBM. In Fig. 2.7a for instance, once the parameters between ($\boldsymbol{v}$,$\boldsymbol{h^{(1)}}$) are trained, the second RBM ($\boldsymbol{h^{(1)}}$,$\boldsymbol{h^{(2)}}$) is trained but $\boldsymbol{h^{(1)}}$ is updated by ignoring $\boldsymbol{v}$ and is computed only with respect



(a) DBN (b) DBM

Figure 2.7: Graphs of Deep Belief Network and Deep Boltzmann Machines. In both figures, all links between neurons are represented with one arrow for the visibility. In **(a)**, links between the two last layers are undirected and structures between other layers are directed graph.

to $\boldsymbol{h^{(2)}}$. A large number of applications used DBN or its extension *e.g.* for face recognition [41], audio classification task [56], machine health monitoring systems[121, 45], schizophrenia prediction [84], for detecting faults in axial coupling systems [112] or for time series forecasting [50].

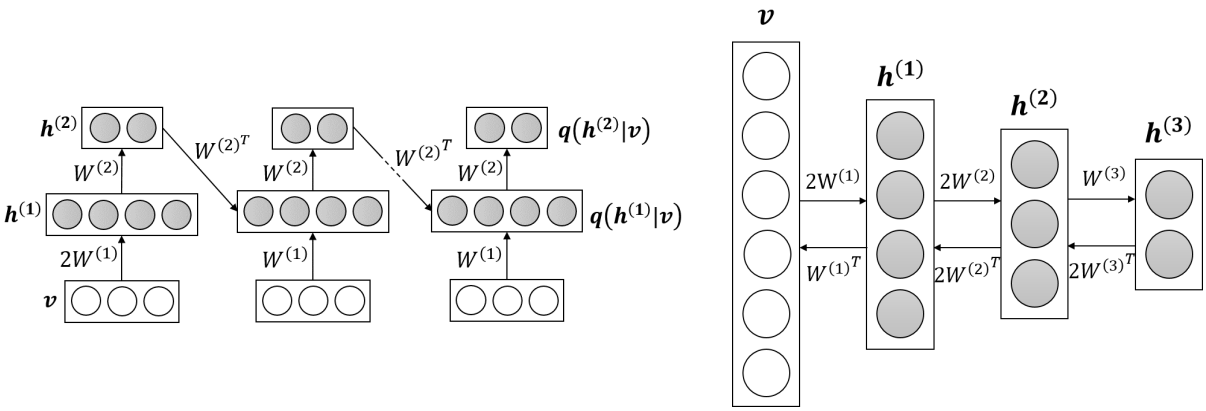
Deep Boltzmann Machines

Unlike DBN, the DBM [93] is an undirected graph. In the Fig. 2.7b, $\boldsymbol{v}$ and $\boldsymbol{h^{(2)}}$ are updated in function of $\boldsymbol{h^{(1)}}$. $\boldsymbol{v}$ and $\boldsymbol{h^{(2)}}$ allow to update $\boldsymbol{h^{(1)}}$ according to (Eq. 2.32) the following equation:

$$\begin{cases} \Pr(\boldsymbol{v}=1|\boldsymbol{h^{(1)}}) = \phi\left(W^{(1)}\boldsymbol{h^{(1)}} + \boldsymbol{\xi^{v}}\right), \\ \Pr(\boldsymbol{h^{(2)}}=1|\boldsymbol{h^{(1)}}) = \phi\left({W^{(2)}}^{T}\boldsymbol{h^{(1)}} + \boldsymbol{\xi^{h^{(2)}}}\right), \\ \Pr(\boldsymbol{h^{(1)}}=1|\boldsymbol{v},\boldsymbol{h^{(2)}}) = \phi\left({W^{(1)}}^{T}\boldsymbol{v} + W^{(2)}\boldsymbol{h^{(2)}} + \boldsymbol{\xi^{h^{(1)}}}\right). \end{cases} \tag{2.32}$$

The structure of the DBM in Fig. 2.7b can be seen as a RBM whose visible and second hidden layers are concatenated into one layer. The visible and the hidden layers $\boldsymbol{h^{(i)}}$ (i being an even number) are updated at once and all hidden layer $\boldsymbol{h^{(j)}}$ (j an odd number) are updated at once. The feedback information from deepest layers when updating lower layers allows the DBM to be more robust than the DBN [95]. For a given visible layer, the *mean field inference* allows to update hidden layers [94, sect. 4.2].

This method consists in estimating the probability of activation of each hidden neuron given the visible layer only. In the case of two hidden layers $\boldsymbol{h^{(1)}}$ and $\boldsymbol{h^{(2)}}$, the mean field inference allows us to compute the probability of $\boldsymbol{h^{(2)}}$ given the visible layer and to update $\boldsymbol{h^{(1)}}$ according to $\boldsymbol{v}$ and $\Pr(\boldsymbol{h^{(2)}}|\boldsymbol{v})$ (see Fig. 2.8a). The DBM training algorithm presented in [27, Chap. 20] for a 3-layer model requires to apply mean field inference for each iteration due to the change of weight matrices. A greedy layerwise pretraining of DBM has been proposed by Salakhudinov and Larochelle in [95, 94] to simplify the training of the DBM. Like DBN, each RBM is trained independently but weight values are doubled to compute layers which are connected with two layers. Fig. 2.8b illustrates one step of training.



(a) Mean field inference. (b) Greedy layerwise pretraining.

Figure 2.8: Methods to train DBM. In **(a)**, each iteration consists in updating $\boldsymbol{h^{(1)}}$ with $\boldsymbol{v}$ and $\boldsymbol{h^{(2)}}$ from the previous iteration. $\boldsymbol{h^{(2)}}$ is then updated. For the first iteration, $\boldsymbol{h^{(2)}}$ value is 0 but the transfer matrix W_1 is doubled. The number of iterations has to be sufficiently large to stabilize the estimation probability $q(\boldsymbol{h^{(2)}}|\boldsymbol{v})$. Note that $q(\boldsymbol{h^{(2)}}|\boldsymbol{v})$ is not binary but represents the expectation of the state $\boldsymbol{h^{(2)}}$. In **(b)**, each RBM is trained independently. For the first RBM, $\boldsymbol{h^{(1)}}$ is updated with two times the weight matrix and the bias. For the other RBM each layer has been updated with two times the weight matrix except in the last RBM where the last layer is updated normally. Once a RBM is trained, the training set is converted into the hidden layer with two times the weigth matrix for the next RBM.

2.3.2 Convolutional Boltzmann Machine

Deep RBM architectures ignore the geometric data structure and connect all visible units to each hidden unit. RBMs can become quickly huge and hard to train efficiently in applications like image processing. Lee *et al.* [55] proposed the convolutional Restricted Boltzmann Machine (conv-RBM), using as depicted in Fig. 2.9, a set of K convolutional filters for sampling K hidden representations of the input image $\boldsymbol{v}$ ("$*$" is put for the convolution operator):

$$\Pr(h_{ij}^{(k)} = 1|v) = \phi\left(\xi_k^h + (W^{(k)^T} * v)_{i,j}\right), \tag{2.33}$$

$$\Pr(v_{ij} = 1|\{h^{(k)}\}_{1\leq k\leq K}) = \phi\left(\xi^v + \sum_{k=1}^{K}(W^{(k)} * h^{(k)})_{i,j}\right). \tag{2.34}$$

In order to stack conv-RBM to form a convolutional DBN, the dimensions of the hidden representation are reduced using a *probabilistic max-pooling* operator. The probabilistic max-pooling consists in converting blocks B_α in the hidden representation $h^{(k)}$ into a single pixel $p_\alpha^{(k)}$. The pixel $p_\alpha^{(k)}$ is equal to one if one hidden units in the block B_α is equal to one. A constraint on hidden units in B_α is enforced: at most one hidden unit can be equal to one in B_α. The structure of a conv-RBM is given in Fig. 2.9 and the energy function of the max-pooling conv-RBM is:

$$E_{conv-RBM}(I,h) = -\sum_{1\leq i,j\leq N_H}\sum_{k=1}^{K}\left(h_{ij}^{(k)}(W^{(k)^T} * v)_{i,j} + \xi_k^h h_{ij}^{(k)}\right) - \xi^v \sum_{1\leq i,j\leq N_V} v_{ij}, \tag{2.35}$$

$$\text{subj. to} \sum_{(i,j)\in B_\alpha} h_{ij}^{(k)} \leq 1, \forall k, \alpha$$

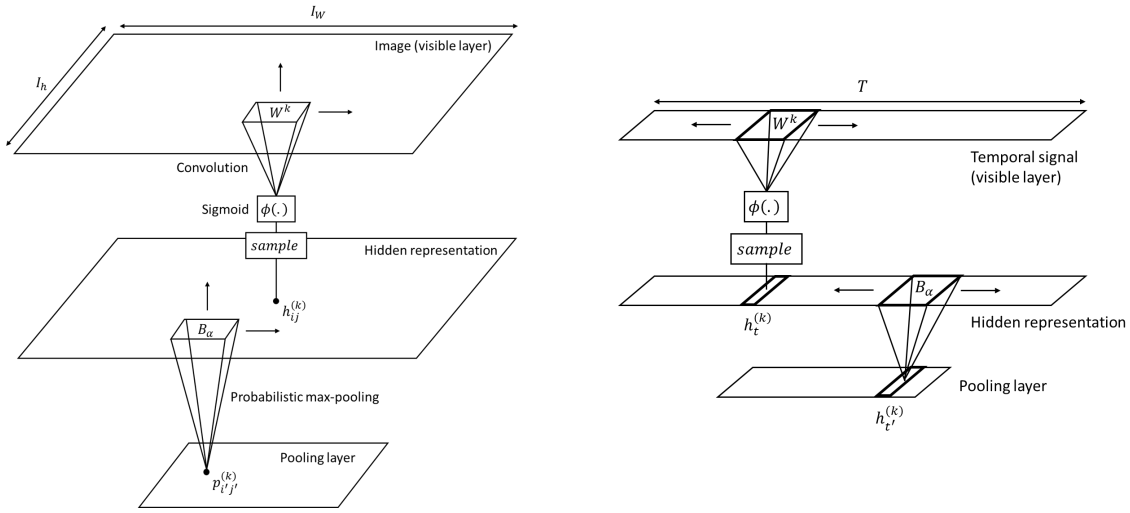


Figure 2.9: Max-pooling convolutional RBM. The figure on the left is the original model designed for image processing. The figure on the right is a 1-D convolutional RBM adapted for temporal signal processing.

In practice, deep architectures are preferred to capture more complex features. conv-RBMs are stacked to form a convolutional Deep Belief Network (cDBN). conv-RBM has been introduced for instance to detect pedestrian [78, 111], but this model has also been used to model time series for environmental sound classification [92] and in [97] for rolling bearing fault analysis. A sound signal being seen as a 1D image (see Fig 2.9), the model learns dynamic dependencies between sampled values.

2.3.3 Boltzmann Machine with real value visible units

Neurons in BMs are binary. However, variables of real systems generally have a continuous or discrete behaviour. This is the case, for example, in image processing [105] where the pixels of an image vary between 0 and 256 in 8 bits coding. To keep the continuous behavior of the visible units, several extensions of the BM have been proposed to tolerate continuous visible units.

Gaussian-Bernoulli Restricted Boltzmann Machines

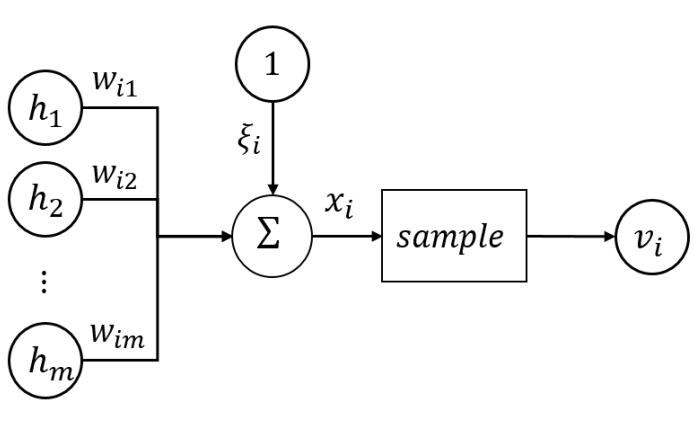
In 2006, Hinton proposed the GBRBM [33] to reduce the dimensionality of the RBM using continuous visible units. Hidden units keep on the binary behaviour but visible units are random variables following a conditioned Gaussian distribution (hence the name) with a variance σ_{ii}^2 (see Fig. 2.10a), *i.e.*:

$$\Pr(v_i|\boldsymbol{h}) \sim \mathcal{N}\left(W(i,:)\boldsymbol{h} + \xi_i^v, \sigma_{ii}^2\right). \tag{2.36}$$

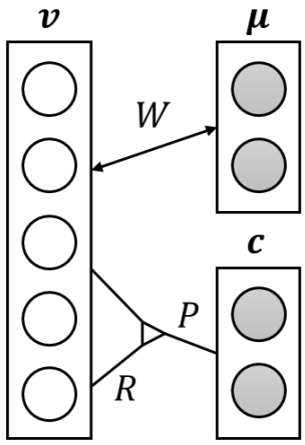
The energy is now defined as:

$$E_{GBRBM}(\boldsymbol{v},\boldsymbol{h}) = \sum_{i=1}^{n} \frac{(v_i - \xi_i^v)^2}{2\sigma_{ii}^2} - \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{v_i}{\sigma_{ii}^2} h_j W_{ij} - \sum_{j=1}^{m} h_j \xi_j^h. \tag{2.37}$$

An improved training procedure of the GBRBM is proposed in [11]. The GBRBM is now systematically combined with new extensions of the RBM to handle continuous data. For example, [12] proposed a DBM with Gaussian visible units for facial reconstruction. The main limitation of the GBRBM is that the model does not learn the covariance between visible units, and so GBRBM gives unsatisfactory results for modeling natural images as in [32] where neighboring pixels may be strongly correlated. A recent paper reconsiders the GBRBM as a mixture of Gaussian model for modeling natural image statistics [65].



(a) GBRBM, neuron.



(b) mcRBM, graph model.

Figure 2.10: **(a)** Structure of visible units in a Gaussian-Bernoulli Restricted Boltzmann Machine (GBRBM). The state is sampled from the Gaussian distribution with mean X_i and variance σ_{ii}^2. **(b)** the visible layer $\boldsymbol{v}$ and the mean hidden layer $\boldsymbol{\mu}$ form a GBRBM. R is a matrix whose columns filter $\boldsymbol{v}$. Each column is associated to a hidden neurons of $\boldsymbol{c}$. P is a transfer matrix between c and filtered visible neurons $(\boldsymbol{v}^T R)^2$ (P is a diagonal matrix with non-positive entries).

Mean and covariance Restricted Boltzmann Machines

The mean and covariance Restricted Boltzmann Machine (mcRBM) proposed by Hinton and Ranzato [32] has two groups of hidden units: mean units and covariance units. Mean units capture information about the mean intensity of each visible neurons and covariance units model dependencies between visible units. The BM with visible and mean units forms the GBRBM and the BM with the visible and covariance units forms the covariance Restricted Boltzmann Machine (covRBM). The energy function of a mcRBM is defined as the sum of the energy of a GBRBM between the visible units $\boldsymbol{v}$ and the mean units $\boldsymbol{\mu}$ (see (Eq. 2.37)) and the energy of a covRBM between visible units $\boldsymbol{v}$ and covariance units $\boldsymbol{c}$:

$$E_{mcRBM}(\boldsymbol{v}, \boldsymbol{\mu}, \boldsymbol{c}) = E_{GBRBM}(\boldsymbol{v}, \boldsymbol{\mu}) + E_{covRBM}(\boldsymbol{v}, \boldsymbol{c}). \tag{2.38}$$

The covariance term aims to capture correlations between visible neurons. An intuitive idea is to define a tensor W where each coefficient W_{ijk} associates two visible neurons, v_i and v_j and one hidden neurons h_k. In the case of natural images whose dimensions are those of the visible layer, the number of parameters shall increase in an exponential way with the number of hidden neurons.

To avoid a large number of parameters, the tensor is factorized in two matrices R and P. The product $\boldsymbol{v}^T R$ is a row vector where each component is a projection of $\boldsymbol{v}$ filtered by a column of R. Each coefficients of the row vector are squared to avoid divergence of parameters during the learning [16]. The second factor matrix P gives the projection to sample covariance hidden units. The energy of the covariance hidden units is given by:

$$E_{covRBM}(\boldsymbol{v}, \boldsymbol{c}) = -\boldsymbol{c}^T \boldsymbol{\xi}^{\boldsymbol{c}} - \left(\boldsymbol{v}^T R\right) \odot \left(\boldsymbol{v}^T R\right) P\boldsymbol{c}. \tag{2.39}$$

with $\odot$ the Hadamar product and $\boldsymbol{\xi}^{\boldsymbol{c}}$ the bias vector of the covariance layer. The structure of a mcRBM is given in Fig. 2.10b. In practice, the Hybrid Monte Carlo (HMC) sampling algorithm [77, sect.5] is used to teach the mcRBM to prevent sampling of visible units which requires to inverse a square matrix of dimension $\dim(\boldsymbol{v})^2$. Like the conv-RBM, mcRBM has been proposed for image processing but it has also been used to model time series. In [16], Dahl used mcRBM for the speech recognition task. The difference between the conv-RBM and the mcRBM lies in the modelling approach of those models. On one hand the

conv-RBM models local correlations – we have an assumption on the geometry (spatial and/or temporal) of data –, on the other hand the mcRBM, considers only global correlations.

Laws to update neurons are:

$$\begin{cases} \Pr(\boldsymbol{v}|\boldsymbol{\mu},\boldsymbol{c}) & \sim \mathcal{N}\left(\Sigma\left(\sum_{j=1}^{m} W_{ij}\mu_j\right),\Sigma\right) \\ \Pr(\mu_j=1|\boldsymbol{v}) & = \phi\left(\sum_{i=1}^{n} W_{ij}v_i + \xi_j^{\mu}\right) \\ \Pr(c_k=1|\boldsymbol{v}) & = \phi\left(\frac{1}{2}\sum_{f=1}^{F} P_{fk}\left(\sum_{i=1}^{n} R_{if}v_i\right)^2 + \xi_k^{c}\right) \end{cases} \tag{2.40}$$

where the covariance matrix $\Sigma = \left(R\,\mathrm{diag}(P\boldsymbol{c})R^T\right)^{-1}$ and F is the number of rows of the matrix P and the number of columns of R.

Mean Product of Student t-distributions

The Mean Product of Student t-distributions (mPoT) proposed in [69] extends the Product of Student t-distributions (PoT) [113] in the same way the mcRBM extends the covRBM. Like the mcRBM, the mPoT has one visible layer and two hidden layers: the binary latent vector $\boldsymbol{\mu}$ modelling the means of visible units and the continuous hidden units $\boldsymbol{c}$ following a Gamma distribution to model the dependencies between the visible units. The BM between $\boldsymbol{v}$ and $\boldsymbol{\mu}$ is a GBRBM and the BM between $\boldsymbol{v}$ and $\boldsymbol{c}$ is a PoT. The energy of the mPoT is given by the following sum:

$$E_{mPoT}(\boldsymbol{v},\boldsymbol{\mu},\boldsymbol{c}) = E_{GBRBM}(\boldsymbol{v},\boldsymbol{\mu}) + E_{PoT}(\boldsymbol{v},\boldsymbol{c}) \tag{2.41}$$

And the energy of the PoT model is:

$$E_{PoT}(\boldsymbol{v},\boldsymbol{c}) = \sum_{i=1}^{\dim \boldsymbol{c}} \left[c_i\left(1+\frac{1}{2}(\boldsymbol{R_i}^T\boldsymbol{v})^2\right) + (1-\gamma)\log c_i\right] \tag{2.42}$$

where $\boldsymbol{R_i}$ is a filter vector associated to the value of the $i-$th hidden unit c_i.

Spike and slab Restricted Boltzmann Machines

Courville proposed in 2011 the spike and slab Restricted Boltzmann Machine (ssRBM) [13], an original approach to model correlations between the visible neurons. The ssRBM is a bipartite graph with visible neurons $\boldsymbol{v}$, each one following a Gaussian distribution, and hidden units. Each hidden unit is composed of a binary variable called *spike* and a continuous vector called *slab*. Denote the spike vector $\boldsymbol{h} \in [\![0,1]\!]^m$, whose component h_i is associated to a *slab* vector of dimension k $\boldsymbol{s^{(i)}} \in \mathbb{R}^k$. The visible layer is sampled by means of each slab vector for which the associated spike value is equal to one. The associated slab vector $\boldsymbol{s^{(i)}}$ gives the intensity of the $i-$th component. The energy function is given by:

$$E_{ssRBM}(\boldsymbol{v},\{h_i,\boldsymbol{s^{(i)}}\}_{1\leq i\leq m}) = \frac{1}{2}\boldsymbol{v}^T\Lambda\boldsymbol{v} - \sum_{i=1}^{m}\left(\boldsymbol{v}^T W^{(i)}\boldsymbol{s^{(i)}}h_i + \frac{1}{2}\boldsymbol{s^{(i)}}^T\alpha^{(i)}\boldsymbol{s^{(i)}} + \xi_i^h h_i\right). \tag{2.43}$$

$W^{(i)}$ is the $i-$th weight matrix $(n\times k)$ between $\boldsymbol{v}$ and $\boldsymbol{s^{(i)}}$. $\alpha^{(i)}$ and Λ are both diagonal matrices which prevent $\boldsymbol{s^{(i)}}$ and $\boldsymbol{v}$ from having large values. Courville showed that ssRBM provides better results than mcRBM in image classification tasks [13]. However, according to [27, chap. 20], the risk with the ssRBM is to obtain a non-positive definite covariance matrix which can be avoided with heuristic rules. Courville also proposed some extensions to the ssRBM [14] to provide results in classification task. Goodfellow [25] added an additional term to the ssRBM to make the partition function Z tractable at the loss of loosing the generative property. In [117], the ssRBM has been mixed with the DBM and the conv-RBM to form a Contrastive spike and slab Convolutional Deep Boltzmann Machine (CssCDBM) for image classification.

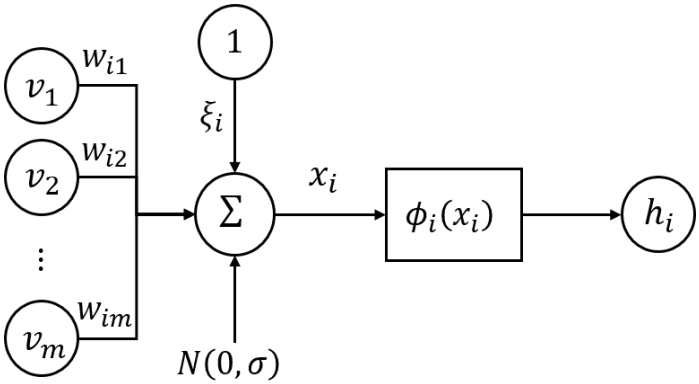


Figure 2.11: Structure of the hidden neuron i of a DN-RBM. The expression of the activation function $\phi_i(x_i)$ is given in (Eq. 2.44).

Continuous Restricted Boltzmann Machines

Another approach of continuous neuron has been proposed by Chen and Murray: the continuous Restricted Boltzmann Machine (DN-RBM) in [8], a RBM using the neuron structure in Fig. 2.11. Chen and Murray proposed the abbreviation "CRBM" for continuous RBM. Different models based on RBM using the prefix 'c' are already mentioned previously (conditional, convolutional, covariance [69] and now continuous). The continuous RBM has been introduced for the first time as a special case of the Diffusion Network (presented in Sec. 2.4). To avoid any confusion with the previous models, the name DN-RBM is used.

In the DN-RBM, the activation function is unique for each neuron. The expression of the activation function is given by:

$$s_j = \phi_j(X_j) = \theta_L + (\theta_H - \theta_L)\frac{1}{1+\exp(-a_j X_j)} \tag{2.44}$$

where θ_L and θ_H are respectively the lower and the upper bounds of the function. a_j is a slope parameter of $\phi_j(.)$. The influences of the parameter a_j are in the behavior of neurons and in the noise regularization. An illustration of the role of a_j is given in Fig. 2.12.

The energy function of the DN-RBM is very similar to the energy of the cHN [39]:

$$E_{DN-RBM}(\boldsymbol{s}=\{\boldsymbol{v},\boldsymbol{h}\}) = -\boldsymbol{v}^T W \boldsymbol{h} - \boldsymbol{v}^T\boldsymbol{\xi}^{\boldsymbol{v}} - \boldsymbol{h}^T\boldsymbol{\xi}^{\boldsymbol{h}} + \sum\nolimits_i \frac{1}{a_i}\int_0^{s_i}\phi^{-1}(s')\mathrm{d}s', \tag{2.45}$$

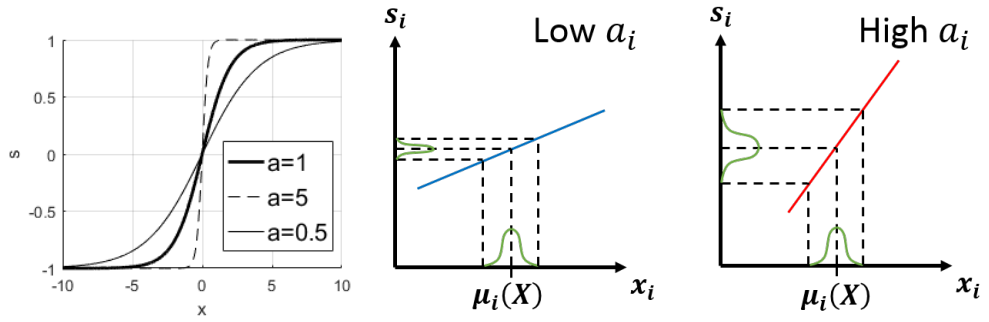
with $\phi^{-1}(.)$ the inverse of the activation for a coefficient slope $a_i = 1$. As for BM, we can use the MCD rule to train parameters of the DN-RBM. (Eq. 2.29) does not change for the DN-RBM and the update law for the coefficient a_i reads :

$$\Delta a_i \propto \frac{1}{a_i^2}\int_{\langle s_i\rangle_1}^{\langle s_i\rangle_0}\phi^{-1}(s')\mathrm{d}s'. \tag{2.46}$$

Like BMs, the DN-RBM is used to model the stochastic equilibrium of the network. The neuron structure can be used in a large number of RBM extension (previously presented) and the continuous behavior for the hidden units allows us to capture more information than with binary units. Like for binary RBMs, it is possible to associate extension of the RBM to the DN-RBM in many applications such as wind speed forecasting in [40] or evaluation of sound quality in [42] where author train a DBN using the structure of DN-RBM.

2.3.4 Dynamic extensions of Boltzmann Machine

Modelling causal relationships is essential for time series forecasting. Prediction models are usually discriminative. Different approaches using modified FFNN have been proposed to model temporal data like



(a) Behaviour of neuron (b) Regularization of noise

Figure 2.12: Influence of the activation parameters a_i. **(a)** displays the activation between -1 and 1 for three different values of a. The bigger the parameter a_i, the more the neuron will have a binary behavior. The smaller a_i, the more the neuron will have a linear behavior. In **(b)**, the two schemes describe how the noise influences the state of the neuron. At a given time, $X_i(t)$ is a Gaussian random variable centered on $\mu_i(X(t)) = \xi_i + \sum_j W_{ij}s_j$ with a variance σ^2. In the left figure, a_i has a low value, the slope of the function is almost horizontal. The dispersion of the noise is "compressed" and the state of the neuron is almost deterministic. If a_i is high like in the second schema, the slope of the curve is more vertical and the dispersion of the state increases.

the Elman Networks [83], autoencoders [51], Recurrent Neural Network (RNN) [67] or Long-Short Term Memory (LSTM) [36].

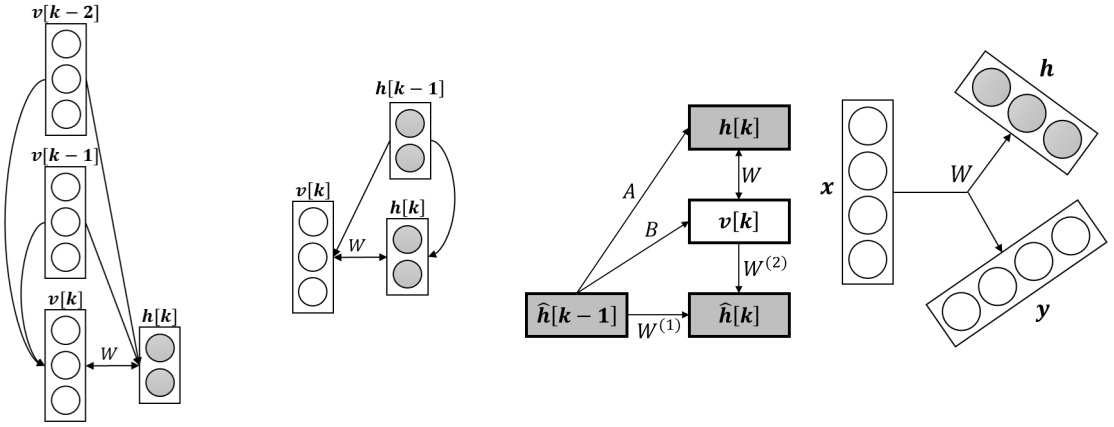
In BMs, the state of the network is updated from the visible layer of the training set which, in this particular case, is $\mathcal{D} = \{\boldsymbol{v}[\boldsymbol{k}]\}_{1\leq k\leq N}$, until the equilibrium state is reached. All $\boldsymbol{v}[\boldsymbol{k}]$ are treated independently regardless the previous state. Models previously presented in this paper can be used to handle temporal data by using a short time-window to the visible units as illustrated in [50] which make use of extensions of BMs but alternative structures have also been proposed mixing generative and discriminative properties of the RBM to take into account past observations, *e.g.* the Discriminative Restricted Boltzmann Machine (DRBM) and the Hybrid Discriminative Restricted Boltzmann Machine (HDRBM) for classification tasks [53, 52].

Conditional Restricted Boltzmann Machine

The conditional Restricted Boltzmann Machine (CRBM) [70] is an extension of the vector autoregressive (VAR) model [60] using RBM structure (see Fig. 2.13a). A linear combination of the past observations are added to the bias values of the visible and the hidden layers (see (Eq. 2.47)). The energy of a CRBM and a classical RBM are similar. Let define the following iterative schemes:

$$\begin{cases} \boldsymbol{\xi^v}[k] = \boldsymbol{\xi^v}[0] + \sum_{i=1}^{p} B^{(i)}\boldsymbol{v}[\boldsymbol{k}-\boldsymbol{i}], \\ \boldsymbol{\xi^h}[k] = \boldsymbol{\xi^h}[0] + \sum_{i=1}^{p} C^{(i)}\boldsymbol{v}[\boldsymbol{k}-\boldsymbol{i}]. \end{cases} \tag{2.47}$$

$\boldsymbol{\xi^v}[0]$ and $\boldsymbol{\xi^h}[0]$ are fixed biases of, respectively, the visible and hidden units, $B^{(i)}$ and $C^{(i)}$ are transfer matrices between the observation $\boldsymbol{v}[k-i]$ and, respectively, the visible and the hidden layers. p is the past observation order. Unlike VAR model, there is no proposed criterion for choosing p. The CRBM can also be used in other applications such as modelling the dependencies of a Multiple Input Multiple Output (MIMO) system as in [115], intra/inter-gender voice conversion [116] or missing label classification [59].



(a) CRBM (b) TRBM (c) RNN-RBM (d) GRBM

Figure 2.13: Architectures of RBMs for sequential dynamic systems. One direction arrows materialize the dependencies. The probability of interest is the conditional probability of the new state given past states. In **(c)**, the prediction of $\hat{\boldsymbol{h}}[k]$ is a RNN.

Temporal Restricted Boltzmann Machines

A similar structure of CRBM has been proposed in [103] named Temporal Restricted Boltzmann Machine (TRBM) which models the current state of the network conditionally on past hidden units. This structure reminds the HMM. (Eq. 2.48) of the TRBM is slightly modified with respect to (Eq. 2.47) to describe CRBM

$$\boldsymbol{\xi}^{\boldsymbol{h}}[k] = \boldsymbol{\xi}^{\boldsymbol{h}}[0] + \sum_{i=1}^{p} A^i \boldsymbol{h}[k-i]. \tag{2.48}$$

Past hidden units are required to update present hidden units. Because hidden units are not available, training a TRBM using CD algorithm requires to sample hidden units from a sequence of observations. A large number of applications has been proposed using TRBM and its extensions like in [22] with an Input-Output TRBM used for 2D facial expression transfer. A widely used extension of TRBM is the Recurrent Temporal Restricted Boltzmann Machine (RTRBM) [104] (see Fig. 2.13b) where visible and hidden units at time k ($\boldsymbol{v}[k], \boldsymbol{h}[k]$) depends conditionally on the previous hidden state $\boldsymbol{h}[k-1]$. The hidden units in a TRBM memorize past observations. In the literature, we can find a large number of TRBM-based architectures for different applications. For instance Nakashika used the RTRBM for voice conversion in [76, 75]. Mittelman introduced the Structured Recurrent Temporal Restricted Boltzmann Machines (SRTRBM) and the Spike-and-slab SRTRBM to learn time series signals [68] for different examples (motion capture video or weather modeling). Introduced by Boulanger-Lewandowski, the Recurrent Neural Network - Restricted Boltzmann Machine (RNN-RBM) [5] (see Fig. 2.13c) is a combination of a RTRBM and a RNN. The expectation of hidden units (or *mean field*) $\hat{\boldsymbol{h}}[k]$ is propagated through the RNN structure:

$$\hat{\boldsymbol{h}}[k] = \phi\left(W^{(1)}\hat{\boldsymbol{h}}[k-1] + W^{(2)}\hat{\boldsymbol{v}}[k] + \boldsymbol{\xi}^{\hat{\boldsymbol{h}}}\right), \tag{2.49}$$

and the biases in the RTRBM are defined as:

$$\begin{cases} \boldsymbol{\xi}^{\boldsymbol{v}}[k] = \boldsymbol{\xi}^{\boldsymbol{v}}[0] + A\hat{\boldsymbol{h}}[k], \\ \boldsymbol{\xi}^{\boldsymbol{h}}[k] = \boldsymbol{\xi}^{\boldsymbol{h}}[0] + B\hat{\boldsymbol{h}}[k]. \end{cases} \tag{2.50}$$

The RNN-RBM is a popular extension of the TRBM. It provides better results than RTRBM for human motion modelling or polyphonic music modelling [5]. A similar approach replacing the RNN by a LSTM has been proposed in [110] to model long term dependencies in music generation.

Gated Restricted Boltzmann Machine

Initially proposed for video compression and denoising using the temporal structure by Memisevic [66], the Gated Restricted Boltzmann Machine (GRBM) is a discriminative extension of the RBM (see Fig. 2.13d for a graphical representation). It is composed of two visible layers and one hidden layer: the input layer $\boldsymbol{x}$, the output layer $\boldsymbol{y}$ and the hidden layer $\boldsymbol{h}$. The three layers are connected with a 3D tensor W of dimensions $n_x \times n_y \times n_h$. The energy function is given by:

$$E_{GRBM}(\boldsymbol{y}, \boldsymbol{h}; \boldsymbol{x}) = -\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_h} W_{ijk} x_i y_j h_k - \sum_{k=1}^{n_h} \xi_k^h h_k - \sum_{j=1}^{n_y} \xi_j^y y_j. \tag{2.51}$$

Here, the function to maximize during the training stage is the conditional probability $\Pr(\boldsymbol{y}|\boldsymbol{x})$:

$$\Pr(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{h}} \Pr(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}) = \sum_{\boldsymbol{h}} \frac{1}{Z(\boldsymbol{x})} \exp(-E_{GRBM}(\boldsymbol{y}, \boldsymbol{h}; \boldsymbol{x})), \tag{2.52}$$

with $Z(\boldsymbol{x})$ the marginal function. Inference test allows to estimate $\boldsymbol{h}$ and $\boldsymbol{y}$ as functions of $\boldsymbol{x}$. The tensor W in (Eq. 2.51) captures the correlations between the input and the output layers. Setting $\boldsymbol{y} = \boldsymbol{x}$ allows the GRBM to capture dependencies between the input components of $\boldsymbol{x}$ [87] as for an autoencoder. For large inputs, the number of parameters may quickly become too large to use this model for real time applications. Models using factors like mcRBM (see Fig. 2.10b) are preferred to GRBM to model the correlations between the visible neurons. Thereby the GRBM can be used to estimate observation $\hat{\boldsymbol{x}}[k+1]$ as function of past $\hat{\boldsymbol{x}}[k]$ [51]. Fig. 2.13d gives the graph representation of the GRBM.

These RBM-based approaches for modelling dynamic systems are used with sequential data. The next section introduces the Diffusion Network for the modelling of dynamic continuous systems.

2.4 Diffusion Network basis

The Diffusion Network (DN) has been introduced by Movellan in [72, 73]. The DN is an artificial neural network based on a Stochastic Differential Equation (SDE) (see [79]) to model time dependencies between neurons of the network. A SDE is a complex mathematical tool much used in mechanics, or financial engineering. The use of DN requires the understanding of mathematical terms not normally used in neural network. A preliminary introduction to mathematics notions is previously needed.

2.4.1 Definitions

Definition 2.4.1 (Stochastic process)**.** A continuous/discrete time stochastic process is of Random Value (RV) $(X(t))_{t\geq 0}$ $(t \in \mathbb{R})$ / $(X[n])_{n\geq 0}$ $(n \in \mathbb{N})$ defined on the same probability space. A continuous process $(X(t))_{t\geq 0}$ is said to be Gaussian if $\forall n \in \mathbb{N}$, $0 \leq t_0 < t_1 < \cdots < t_n$: $(X(t_0), \ldots, X(t_n))$ is a Gaussian vector. A white noise is an example of stochastic process.

Definition 2.4.2 (Filtration)**.** The studied phenomena depends on (continuous) time. What is known at time t is gathered in the stack σ-field $\mathcal{F}(t)$. $\mathcal{F}(t)$ can be interpreted as the "information gathered at time t". A filtration is an increasing sequence of sub σ-fields of $\mathcal{F}$, *i.e.* $\mathcal{F}(s) \subset \mathcal{F}(t)$, if $s \leq t$ (see Fig. 2.14).

Definition 2.4.3 (Brownian Motion)**.** The Brownian motion or Wiener process $(B(t))_{t\geq 0}$ is a continuous stochastic process for which $B(0) = 0$ and $\forall 0 \leq s \leq t$, $B(t) - B(s) \sim \mathcal{N}(0, t-s)$. Fig. 2.14 is an example of 1−D Brownian motion. For $s \leq t$, $\mathrm{E}\left[B(t)|\mathcal{F}(s)\right] = B(s)$ and $\mathrm{Var}\left[B(t)|\mathcal{F}(s)\right] = t - s$.

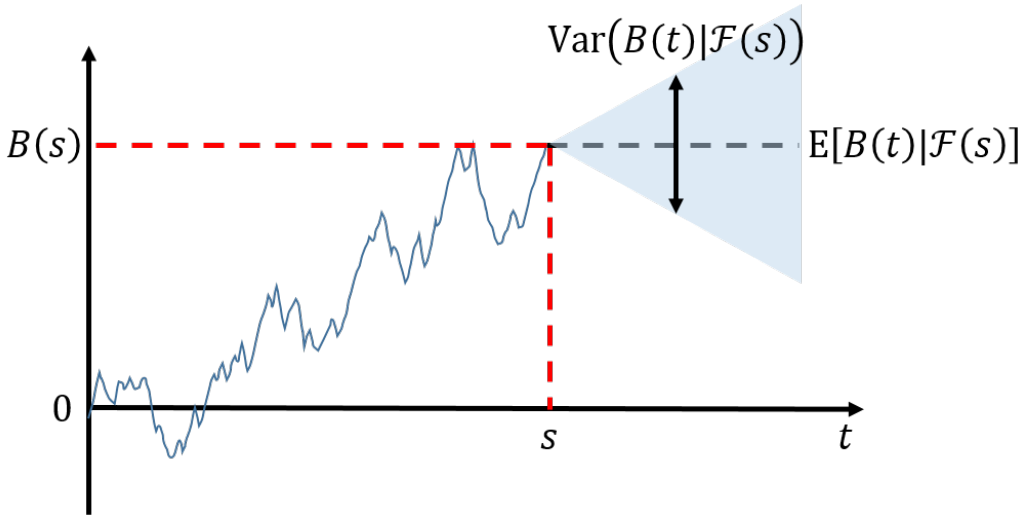


Figure 2.14: A Brownian motion of dimension 1. If the signal $B(u)$ is known until time s, the value of $B(t)$ with $t \geq s$ is a Gaussian noise centered in $B(s)$ and with the variance $\mathrm{Var}\left[(B(t)|\mathcal{F}(s))\right]$ proportional to the distance $t-s$.

Definition 2.4.4 (Wiener integral)**.** Let ξ be the set of piece-wise constant functions:

$$\xi = \left\{ f : \mathbb{R}^+ \to \mathbb{R}, f(s) := \sum_{i=0}^{n-1} a_i 1_{[t_i, t_{i+1}[}(s), n \in \mathbb{N} \right\} \tag{2.53}$$

For $f \in \xi$, the Wiener integral is:

$$I(f) = \int_0^{+\infty} f(s)\mathrm{d}B(s) = \sum_{i=0}^{n-1} a_i \left(B(t_{i+1}) - B(t_i)\right) \tag{2.54}$$

Definition 2.4.5 (Itô stochastic integral)**.** The Itô stochastic integral extends the Wiener integral for stochastic integrands. Let $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ be a filtered probability space with a Brownian motion B on it (*i.e.* B is an $\mathbb{F}-$Brownian motion).

$$\psi = \left\{ \begin{array}{c} (\theta(t))_{t\in\mathbb{R}^+} \text{ adapted, } \theta_t \text{ is } \mathcal{F}(t)-\text{measurable } s.t. \\ \exists n \in \mathbb{N}, \theta(t) = \sum_{i=0}^{n-1} \theta^i 1_{]t_i, t_{i+1}]}(t) \text{ and } s.t. \ \forall i \in [\![0, n-1]\!], \theta^i \in L^2(\mathbb{P}) \end{array} \right\}$$

For $\theta \in \psi$, it is natural to define $\int_0^{+\infty} \theta_s \mathrm{d}B_s$ as:

$$I(\theta) = \int_0^{+\infty} \theta_s \mathrm{d}B_s = \sum_{i=0}^{n-1} \theta^i (B_{t_{i+1}} - B_{t_i})$$

Remark. Adaptness of $(\theta(t))_{t\geq 0}$ implies that θ^i is $\mathcal{F}(t)-$measurable. Note that $\xi \subset \psi$. For $\theta \in \psi$, it is natural to define $\int_0^t \theta(s)\mathrm{d}B(s)$ as:

$$\int_0^t \theta(s)\mathrm{d}B(s) = \sum_{i=0}^{n-1} \theta^i (B(t_{i+1}) - B(t_i)). \tag{2.55}$$

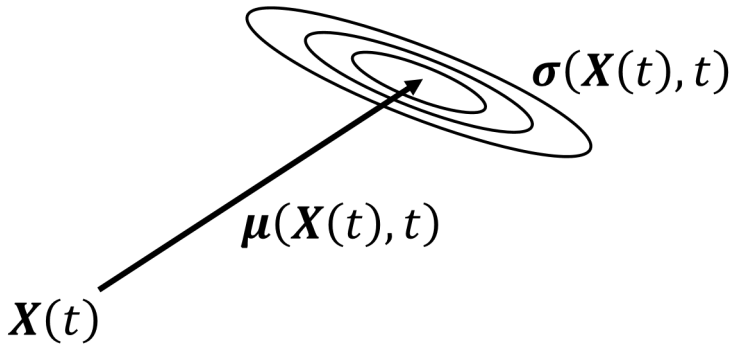


Figure 2.15: SDE principle. The differential $\mathrm{d}\boldsymbol{X}(t)$ is the sum of a deterministic part called the drift term $\boldsymbol{\mu}(\boldsymbol{X}(t),t)$ and a stochastic part the called diffusion term $\boldsymbol{\sigma}(\boldsymbol{X}(t),t)$.

2.4.2 Stochastic Differential Equation

An Ordinary Differential Equation (ODE) describes the temporal evolution of a signal. However, an ODE is not adapted to model the dynamic of a stochastic process. To model the uncertainty part of a stochastic process with a differential equation, an additional noisy term has to be added. An ODE with an additional noisy term is called SDE. Let $\boldsymbol{X}(t) \in \mathbb{R}^d$ a stochastic process. A SDE is written in the following form:

$$\boldsymbol{X}(t) = \boldsymbol{X}(0) + \int_0^t \mu(\boldsymbol{X}(s),s)\mathrm{d}s + \int_0^t \sigma(\boldsymbol{X}(s),s)\mathrm{d}\boldsymbol{B}(s) \tag{2.56}$$

where $(\boldsymbol{B}(s))_{s\geq 0}$ is a standard Brownian motion. (Eq. 2.56) can be also written in short differential form as:

$$\mathrm{d}\boldsymbol{X}(t) = \mu(\boldsymbol{X}(t),t)\mathrm{d}t + \sigma(\boldsymbol{X}(t),t)\mathrm{d}\boldsymbol{B}(t) \tag{2.57}$$

where

- $\mu : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d$ is the the *drift term*.
- $\sigma : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ is the *diffusion term*.
- $\boldsymbol{B}$ an $\mathbb{R}^d-$Brownian motion.

The drift term is the deterministic part of the equation and the diffusion term is the stochastic part of the equation. Fig. 2.15 is a scheme illustrating the SDE model.

Theorem 2.4.1 (Existence and Uniqueness)**.** Let μ and σ be continuous functions in space *s.t.* $\exists K > 0$ satisfying:

- $\forall t \in [0,T], \forall x,y \in \mathbb{R}^n, |\mu(t,x) - \mu(t,y)| + |\sigma(t,x) - \sigma(t,y)| \leq K|x-y|$ (Lipschitz property in space, uniformly in time).
- $|\mu(t,x)|^2 + |\sigma(t,x)|^2 \leq K(1+|x|^2)$.
- $X(0)$ can be random, independent of $\boldsymbol{B}$ and in $L^2(\mathbb{P})$.

Then, (Eq. 2.56) admits an unique solution which satisfies: $\mathrm{E}\left[\sup_{t\in[0,T]} |X(t)|^2\right] < +\infty$.

2.4.3 The Girsanov's theorem

Theorem 2.4.2 (Change of probability)**.** Let $\mathbb{P}$ and $\mathbb{Q}$ two probability spaces. And, let $(B(t))_{t\geq 0}$ be a $d-$dimensional Brownian motion on $(\Omega, \mathcal{F}(t), \mathbb{P})$ and $(\mathcal{F}(t))_{t\geq 0}$ its canonical filtration ($\mathcal{F}(t) = \sigma(B(s), s \leq t)$), $(\theta(s))_{s\geq 0}$ is a $d-$dimensional and adapted process and $L(t)$ well defined:

$$\left.\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\right|_{\mathcal{F}(t)} = L(t) = \exp\left(\int_0^t \theta(s)\mathrm{d}B(s) - \frac{1}{2}\int_0^t |\theta(s)|^2 \mathrm{d}s\right) \tag{2.58}$$

Then,

$$D(t) = B(t) - \int_0^t \theta(s)\mathrm{d}s \tag{2.59}$$

is a standard $\mathbb{Q}-$Brownian motion.

2.4.4 Application to the SDE

Let $X = (\boldsymbol{X}(t))_{t\geq 0}$ a $d-$dimensional process (also named "path") for which (Eq. 2.57) can be written. The drift and the diffusion are two parametric functions and λ is the set of parameters. Let be $\mathbb{Q}^\lambda$ and R, respectively, the probability measure of the model for a given set of parameter and the probability measure of a reference path. The reference is induced by a DN with no drift (only the diffusion term $\sigma(\boldsymbol{X}(t), t)$). In (Eq. 2.57), the Brownian motion $(B(t))_{t\geq 0}$ is a $\mathbb{Q}^\lambda-$Brownian motion. Let denote:

$$\widetilde{B}(t) = B(t) + \int_0^t \frac{\mu(\boldsymbol{X}(s), s)}{\sigma(\boldsymbol{X}(s), s)}\mathrm{d}s \tag{2.60}$$

the $R-$Brownian motion. Indeed, from (Eq. 2.60) and (Eq. 2.57):

$$\mathrm{d}\boldsymbol{X}(t) = \sigma(\boldsymbol{X}(t), t)\,\mathrm{d}\widetilde{B}(t) \tag{2.61}$$

In (Eq. 2.60), θ_s can be identified from (Eq. 2.59) and, according to the Girsanov's theorem:

$$\left.\frac{\mathrm{d}\mathbb{Q}^\lambda}{\mathrm{d}R}\right|_{\mathcal{F}(t)} = \exp\left(\int_0^t \frac{\mu(\boldsymbol{X}(s), s)}{\sigma(\boldsymbol{X}(s), s)}\mathrm{d}\widetilde{B}(s) - \frac{1}{2}\int_0^t \left|\frac{\mu(\boldsymbol{X}(s), s)}{\sigma(\boldsymbol{X}(s), s)})\right|^2 \mathrm{d}s\right) \tag{2.62}$$

(Eq. 2.61) is injected in the previous equation:

$$\left.\frac{\mathrm{d}\mathbb{Q}^\lambda}{\mathrm{d}R}\right|_{\mathcal{F}(t)} = L^\lambda(\boldsymbol{X}) = \exp\left(\int_0^t \frac{\mu(\boldsymbol{X}(s), s)}{\sigma(\boldsymbol{X}(s), s)^2}\mathrm{d}\boldsymbol{X}(s) - \frac{1}{2}\int_0^t \left|\frac{\mu(\boldsymbol{X}(s), s)}{\sigma(\boldsymbol{X}(s), s)}\right|^2 \mathrm{d}s\right) \tag{2.63}$$

$L^\lambda(\boldsymbol{X})$ can be interpreted as the likelihood of the path $\boldsymbol{X}$ generated with the SDE in (Eq. 2.57) compared to the likelihood of the reference model (without drift). For a fixed $\boldsymbol{X}$ used as a training set, $L^\lambda(\boldsymbol{X})$ is the likelihood function of the set of parameters λ.

2.4.5 Back to the Diffusion Network

The DN is a fully connected neural network with non-symmetrical links as represented in Fig. 2.4a. This model extends the cHN by adding a noise generator inside the neuron structure (see Fig. 2.16). The signal

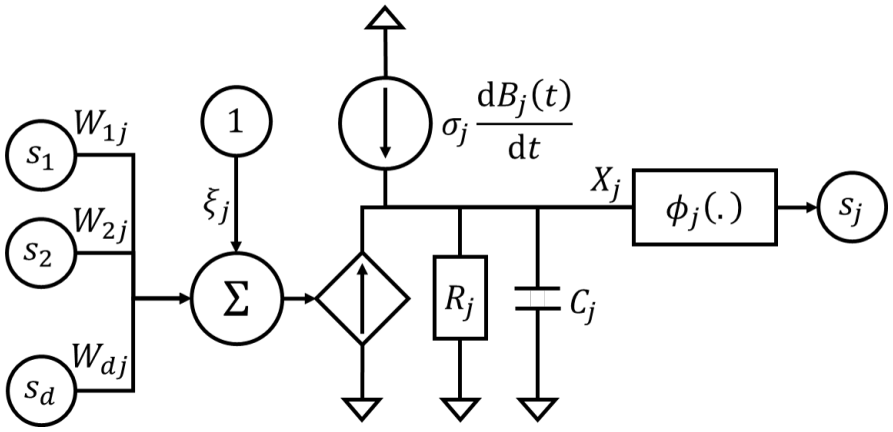


Figure 2.16: Electronic structure of the neuron j in a DN.

of interest is the input of the activation function $\boldsymbol{X}(t) = (X_j(t))_{1\leq j\leq d} \in \mathbb{R}^d$ where d is the size of the network.

$\boldsymbol{X}(t)$ is a stochastic process which can be written under the following SDE system:

$$\mathrm{d}X_j(t) = \mu_j\left(\boldsymbol{X}(t), t\right)\mathrm{d}t + \sigma_j \mathrm{d}B_j(t),\ \forall j = 1, 2, \ldots, d \tag{2.64}$$

(Eq. 2.65) gives the expression of the drift for the $j-$th neuron.

$$\mu_j(\boldsymbol{X}(t), t) = \kappa_j \left(-\rho_j X_j(t) + \xi_j + \sum_{i=1}^{d} W_{ij} S_i(t)\right) \tag{2.65}$$

In (Eq. 2.65), ρ_j and κ_j are, respectively, the inverse resistor and the inverse capacitance of the neuron i (Fig. 2.16). W is the transfer matrix between neurons (see Fig. 2.4a). $S_i(t) = \phi_i(X_i(t))$ is the output of neuron i and ϕ_i is the $i-$th activation function of the neuron. The activation function of a DN is a sigmoïd function:

$$S_j = \phi_j(X_j) = \theta_L + (\theta_H - \theta_L)\frac{1}{1 + \exp(-a_j X_j)} \tag{2.66}$$

The state of a neuron S_j is bounded between θ_L and θ_H which are respectively the lower and the upper bounds of the sigmoïd function. a_j is a slope parameter of $\phi_j(.)$. The parameter a_j regularizes the behavior and the noise of the neuron j. An illustration of the role of a_j is given in Fig. 2.12.

The sampling rule of a sequence with a sampling period $\Delta t = t_{k+1} - t_k$ is given by:

$$X_j(t_{k+1}) = X_j(t_k) + \mu_j(\boldsymbol{X}(t_k), t_k)\Delta t + \sigma_j Z_j(t_k)\sqrt{\Delta t} \tag{2.67}$$

where $Z_j(t)$ is a standard Gaussian noise.

Training a DN consists in finding the SDE which gives the best description of the data. Movellan in [73] proposed a training procedure based on Monte Carlo Expectation Maximization (MCEM) algorithm using (Eq. 2.63) as the loss function. A new training procedure of the DN will be proposed in the chapter 4.

2.5 Discussion

Table 2.1 lists the different models mentioned in this review and offers a comparison between them. To quantify the impact of those models a mark for each model related to the number of documents (all type) found on SCOPUS has been proposed. Marks are based on the number of documents we got when we enter the name of the model (all field). Some specific rules have been added to avoid bias for the DN

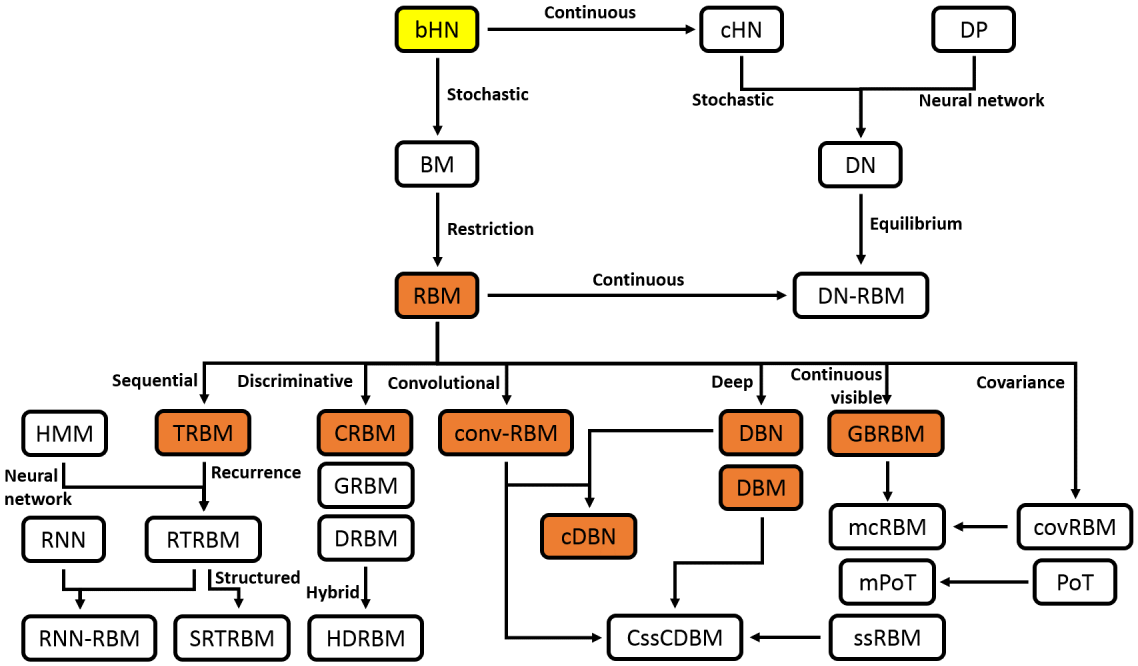


Figure 2.17: Hierarchy of generative Boltzmann network. Each cell represents a model and directed links give the kinship relating one model to an extension. The tree starts from the yellow cell with the bHN proposed in 1982. Orange cells are models that have impacted significantly the research onto this family. The RBM is the original model. The GBRBM is used in almost all applicative papers because of the continuous behavior of the visible units. The DBN and the DBM are two deep versions of the RBM. The other orange cell models presented in the paper have impacted the research on RBM's extensions.

and the DN-RBM. The exact number of documents is not a relevant indicator due to the introduced bias. Another bias lies in the fact that each author does not call a model with the same name (for example, the Gaussian-Bernoulli RBM has also been called the Gaussian RBM or the Continuous RBM). Also some different models can have the same name or acronym (example: CRBM with 'c' for continuous, covariance, conditional or convolutional). Finally, the diffusion network is a very popular expression used in many different fields. The mark in Table 2.1 is put for each model before and after 2010 to provide an idea of a global evolution. Fig. 2.17 summarizes the hierarchy of models presented in the section.

Model Name	Date	Visible units behavior	Hidden units behavior	(G) / (D)	Mark <2010	Mark ≥ 2010	Sources
Hidden Markov Model	1966	Discrete	Discrete	D			[86]
binary Hopfield Network	1982	Binary	Binary	G	⋆⋆⋆	⋆⋆	[37]
continuous Hopfield Network	1984	Continuous	Continuous	G	⋆⋆⋆⋆	⋆⋆⋆⋆	[38]
Boltzmann Machine	1983	Binary	Binary	G	⋆⋆⋆⋆⋆	⋆⋆⋆⋆⋆	[1]
Restricted Boltzmann Machine	1986	Binary	Binary	G	⋆⋆	⋆⋆⋆⋆⋆	[31]
Deep Belief Network	2006	Binary	Binary	G	⋆⋆	⋆⋆⋆⋆⋆	[35]
Deep Boltzmann Machine	2009	Binary	Binary	G	⋆⋆⋆	⋆⋆⋆⋆⋆	[93]
convolutional RBM	2009	Binary	Binary	G	⋆	⋆⋆⋆⋆⋆	[55]
convolutional DBN	2009	Binary	Binary	G	⋆⋆	⋆⋆⋆⋆⋆	[55]
Gaussian-Bernoulli RBM	2006	Continuous	Binary	G	⋆⋆	⋆⋆⋆	[12]
covariance RBM	2010	Continuous	Binary	G	⋆	⋆⋆⋆	[87]
mean and covariance RBM	2010	Continuous	Binary	G	⋆	⋆⋆	[32]
Product of Student t-distributions	2003	Continuous	Continuous	G	⋆⋆	⋆⋆	[113]
mean Product of Student t-distributions	2010	Continuous	B and C	G		⋆⋆	[69]
spike and slab RBM	2011	Continuous	B and C	G		⋆⋆	[13]
Contrastive spike and slab Convolutional DBM	2018	Continuous	B and C	G		⋆	[117]
Conditional RBM	2007	Binary	Binary	G	⋆	⋆⋆⋆	[96]
Gated RBM	2007	Binary	Binary	G	⋆	⋆⋆⋆	[66]
Discriminative RBM	2008	Binary	Binary	D	⋆	⋆⋆⋆⋆	[52]
Hybrid Discriminative RBM	2008	Binary	Binary	D and G	⋆	⋆⋆⋆	[52]
Temporal RBM	2007	Binary	Binary	D	⋆	⋆⋆⋆⋆	[103]
Recurrent Temporal RBM	2009	Binary	Binary	D	⋆	⋆⋆⋆	[104]
Structured Recurrent Temporal RBM	2014	Binary	Binary	D	⋆	⋆⋆	[68]
Recurrent Neural Network - RBM	2012	Binary	Binary	D	⋆	⋆⋆⋆⋆	[5]
Diffusion Network	1993	Continuous	Continuous	G	⋆	⋆⋆	[73]
Diffusion Network - RBM	2002	Continuous	Continuous	G	⋆	⋆	[9]

Table 2.1: Table of comparison of the different models.
Column **(G) / (D)** refers to Generative model of Discriminative model.
Marks are based on the number of document on SCOPUS when we enter the name of the model (some specific rule has been added to avoid bias). The exact number of documents is not a relevant indicator due to the introduced bias. We choose to give a mark in function of the order of magnitude of the number of documents.
WARNING: the column of visible units behaviour is not significantly because Gaussian visible units of the GBRBM are almost always used in every BM extension.

Chapter 3

DN-RBM for learning signal representation

This chapter is dedicated to the use of DN-RBM on a time-window of signals. The aim is to understand how the model converges and how the DN-RBM can be used to detect non stationary signals. First, the convergence of the model using a time-window is studied to better understand the role of the hidden state. In the second section, the influence of the hidden layer size is studied. Finally, an algorithm of detection is proposed to detect non stationary signals and results on experiments with toy data are given.

3.1 Convergence of the training procedure

Let consider a DN-RBM trained on short-time window of signal to model it. Suppose a time-window $\boldsymbol{v} = (v(t_1), \ldots, v(t_m))^T$ where m is the window length (and the size of the visible layer). In a DN-RBM composed of n hidden neurons bounded between $\theta_H = +\theta$ and $\theta_L = -\theta$. Let h_i, the $i-$th component of the hidden layer defined as $h_i = \phi_i(x_i)$ where $x_i \sim \mathcal{N}(z_i, \sigma))$ with:

$$z_i = \xi_i^h + \sum_{j=1}^{m} W_{ij} v(t_j) \tag{3.1}$$

Let $\boldsymbol{w_i}$ the $i-$th line vector of the transfer matrix W. $\boldsymbol{w_i}$ can be seen as a temporal vector $W_{ij} = \boldsymbol{w_i}(t_j)$. For a signal without offset the hidden bias vector $\boldsymbol{\xi^h} = (\xi_1^h, \ldots, \xi_n^h)^T$ (where n is the number of hidden units) tends to zeros during the training and z_i becomes:

$$z_i = \sum_{j=1}^{m} \boldsymbol{w_i}(t_j) v(t_j) = \Gamma_{\boldsymbol{w_i v}}(0) \tag{3.2}$$

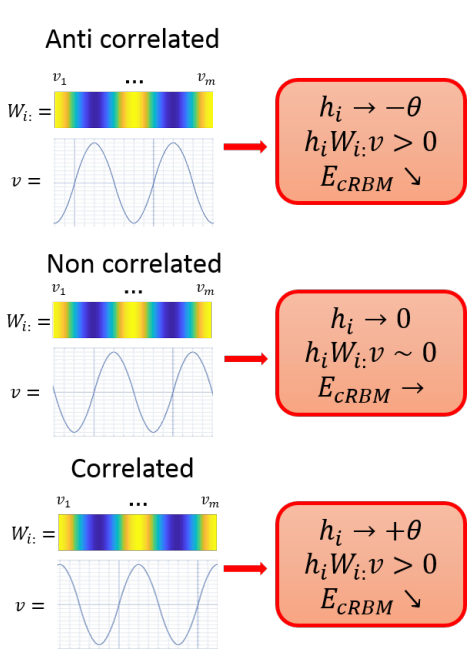
where "(0)" in $\Gamma_{\boldsymbol{w_i v}}(0)$ is put to emphasize there are no time-shift between $\boldsymbol{v}$ and $\boldsymbol{w_i}$. The sum over i of the correlation function $\Gamma_{\boldsymbol{w_i v}}$ is also present in the energy function:

$$-\boldsymbol{h}^T W \boldsymbol{v} = -\sum_{i=1}^{n} h_i z_i = -\sum_{i=1}^{n} h_i \Gamma_{\boldsymbol{w_i v}}(0) \tag{3.3}$$

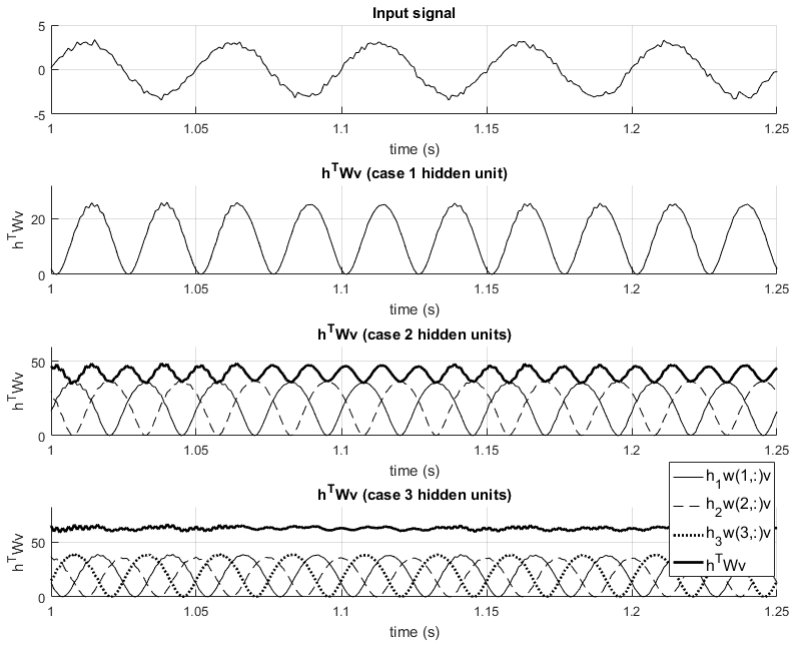
The learning step consists in maximizing the likelihood of data for the probability function. This is equivalent to minimize the energy function (Eq. 2.45) for every observation. Because the hidden units are bounded, the bigger z_i, the lower is the energy. z_i is a scalar product and the Cauchy–Schwartz inequality reads:

$$|z_i| = |\boldsymbol{w_i v}| \leq \|\boldsymbol{w_i}\| \times \|\boldsymbol{v}\| \tag{3.4}$$

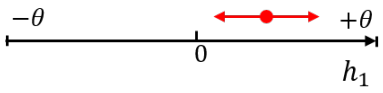
The equality in (Eq. 3.4) is observed when the line vector $\boldsymbol{v}$ and the $i-$th line of the transfer matrix $\boldsymbol{w_i}$ are co-linear, *i.e.* $\boldsymbol{w_i}^T = \alpha \boldsymbol{v}$. The non-linear behavior of neurons ensures that α will not diverge. If $\alpha \to \infty$,



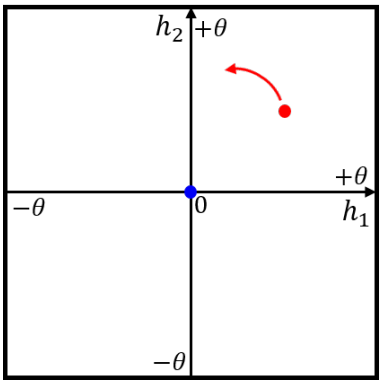
(a) Scheme of energy variation.



(b) $\boldsymbol{h}^T W \boldsymbol{v}$ variation.(Toy example)



(c) Hidden state with one neuron.



(d) Hidden state with two neurons.

Figure 3.1: A discover of this work is the role of the phase between lines of the transfer matrix and the visible layer. **(a)** is a scheme to explain how the energy varies in function of the phase between the hidden state and the signal. The correlation between each the signal and line i of W gives how $\boldsymbol{h}$ and the energy varies. **(b)** shows a sinus function and the variation of the energy term $\boldsymbol{h}^T W \boldsymbol{v}$ for three well-trained DN-RBMs with respectively 1, 2 and 3 hidden unit(s). Both hidden units capture the same frequency but with a delay which ensures an energy level as stable as possible regardless of the phase of the visible layer. **(c)** and **(d)** are two diagrams explaining the movement of hidden neuron(s) when the observation window is shifted. In the case of one hidden neurons **(c)**, the hidden neuron state (red point) oscillates around 0. The point 0 may correspond to a signal that is not correlated to the transfer matrix or the absence of a signal. In the case of two hidden neurons **(d)** the hidden revolve around the gravity center (0 in the figure). The distance to the gravity center gives the intensity of the learned component. The angle gives the phase of the learned component. At least two hidden units are needed to rebuild the visible layer from the hidden representation.

then the neuron's behavior would tend to be binary and the model would not be able to reconstruct the signal.

The learning step of the DN-RBM in (Eq. 2.29) and (Eq. 2.46) captures the frequencies of the signal. The transfer matrix will capture the different signal components observable in the training set. The phase ψ between the visible layer and vectors $\boldsymbol{w_i}$ changes when the visible layer is shifted in time: $\boldsymbol{w_i}$ and $\boldsymbol{v}$ can be correlated ($\psi = 0$), non-correlated ($|\psi| = \pi/2$) or anti-correlated ($|\psi| = \pi$). Fig. 3.1a shows the three configurations for a signal containing a single frequency. The value of the hidden unit is the projection value of the window of observation onto the $i-$th line of the transfer matrix. This property allows to keep a negative energy component for cases $\psi = 0$ and $|\psi| = \pi$. However, when $|\psi| = \pi/2$ the energy associated to the $i-$th line of the transfer increase and tends to zero.

For a given time-window $\boldsymbol{v}$, each hidden units $h_i = \phi_i(z_i)$ gives the non linear transform of a Gaussian noise centered around the correlation between the visible layer and the $i-$th line of the transfer matrix $\Gamma_{\boldsymbol{w_i v}}(0)$. The product $h_i \Gamma_{\boldsymbol{w_i v}}(0)$ tends to remain positive and to decrease the energy according to (Eq. 3.3). In the case $|\psi| = \pi/2$, the correlation $\Gamma_{\boldsymbol{w_i v}}(0)$ tends to zero. When the observation window is shifted in time, the energy component of the $i-$th hidden unit $h_i \boldsymbol{w_i v}$ oscillates with a frequency twice than the captured frequency in $\boldsymbol{w_i}$. Fig. 3.1b plots the evolution in time of the energy term $\boldsymbol{h}^T W \boldsymbol{v}$ for a DN-RBM trained with a sinusoidal signal. The upper plot is the original signal, the last three ones are the $\boldsymbol{h}^T W \boldsymbol{v}$ terms in time with, resp. $1 \leq n \leq 3$ hidden units. Note that for a DN-RBM with two or three hidden units all hidden neurons capture the same frequency but with a delay of $\pi/2$ and $\pi/3$ for the DN-RBM with resp. two and then three hidden units. The delay introduced between the hidden units allows them to keep the energy as stable as possible.

In a well trained DN-RBM, the weights of the transfer matrix capture the *main signal components*. The frequency information is then included in the lines of the transfer matrix and the hidden state gives the correlation information between the observed signal and the transfer matrix. At least two hidden units specialized on the same frequency are required to be able to reconstruct the learned frequency with the corresponding phase and intensity (see Fig. 3.1c and Fig. 3.1d). In the case of signal with multiple frequencies, this is not impossible that a hidden units will be specialized on more than one frequency.

3.2 Sizing the network

In neural network community, there is no criterion to size a network: choice of the number of hidden layers and the then number of units in each hidden layer. Contrary to the VAR model (for example) where the Akaike Information Criterion (AIC) is used to choose the size of the model. The researchers have to choose the size of neural networks according to the specific score of the application. In the application of signal decomposition, the root mean square error is chosen to the check the performance of the DN-RBM.

Let the toy data be given in Fig. 3.3a. The signal duration is 30 second, the first 15 second of signal is the training set and the last 15 second is the validation set. DN-RBMs has been trained with 2 to 20 hidden units and the Root Mean Square Error (RMSE) over the validation set has been computed. Results are plotted in Fig. 3.3b. In particular, the results of three models, with respectively, two, five and ten hidden units is detailed in figures 3.4-3.6. Once the model is trained, a time-window from the validation set is selected and reconstructed using one step of Gibbs sampling. In figures 3.4-3.6.a, the plot on the left gives the temporal representation of v^0 (in blue), the original signal and v^1 (in red) the reconstructed signal. The plot on the right is the Fourier transform of both signals. The transfer matrix of each model is given in figures 3.4-3.6.b on the left. The right plot gives the Fourier transform of each rows of the transfer matrix.

A similar experience is performed with a new toy signal where the intensity of each frequency of the toy data is reversed. The component with the higher frequency have the most energy. See figures 3.7-3.10 for the results with the second toy database. Both experiences confirm that DN-RBMs learn the frequency information of the signal. With not enough hidden units, the model will focus on the learning of the

frequencies with the highest intensities. The more the model have hidden neurons, the more the model learns the frequencies with lower intensity.

In figures 3.4-3.9, it is easy to notice that frequencies learned by hidden units are phase-shifted. As explain in Sec. 3.1, the different phases between hidden units ensure a stable energy regardless the phase of the visible layer.

Let a signal s be defined as the linear combination of two signals c_1 and c_2:

$$s = a_1 c_1 + a_2 c_2 \tag{3.5}$$

where $a_1 > a_2 > 0$ and w_i is another signal defined as:

$$w_i = c_i \tag{3.6}$$

According to previous observations, a DN-RBM without a sufficient number of hidden neurons focuses only on few features of the training database. If w_i is a row of the transfer matrix that learns only one component, what is the priority feature learned from s? ($w_i = c_1$ or $w_i = c_2$). According to the demonstration in Sec. 3.1, the scalar product between w_i and s is present in the energy function:

$$-\boldsymbol{h}^T W \boldsymbol{v} = -\sum_{i=1}^{n} h_i \langle s, w_i \rangle_T \tag{3.7}$$

where $\langle ., . \rangle_T$ is a scalar product in the time-window of length T. In an energy model, the learning step encourages the lowest possible energy. In other words, according to the previous demonstration in Sec. 3.1, the learning step of a DN-RBM tends to get the biggest possible scalar product between time-window of the training signal and rows of the transfer matrix. According to Cauchy-Schwartz inequality:

$$\langle c_1, c_2 \rangle_T \leq ||c_1||_T . ||c_2||_T \tag{3.8}$$

From (Eq. 3.8):

$$\begin{aligned}
-(a_1 - a_2)\langle c_1, c_2 \rangle_T &\geq -(a_1 - a_2)||c_1||_T . ||c_2||_T \\
a_1 . ||c_1||_T^2 - a_2 . ||c_2||_T^2 - (a_1 - a_2)\langle c_1, c_2 \rangle_T &\geq a_1 . ||c_1||_T^2 - a_2 . ||c_2||_T^2 - (a_1 - a_2)||c_1||_T . ||c_2||_T \\
\langle s, c_1 \rangle_T - \langle s, c_2 \rangle_T &\geq a_1 . ||c_1||_T^2 - a_2 . ||c_2||_T^2 - (a_1 - a_2)||c_1||_T . ||c_2||_T \\
\langle s, c_1 \rangle_T - \langle s, c_2 \rangle_T &\geq (a_1 . ||c_1||_T + a_2 . ||c_2||_T) \times (||c_1||_T - ||c_2||_T).
\end{aligned} \tag{3.9}$$

In (Eq. 3.9), $(a_1 . ||c_1||_T + a_2 . ||c_2||_T) > 0$. Then, $||c_1||_T > ||c_2||_T$ is a sufficient condition for having $\langle s, c_1 \rangle_T > \langle s, c_2 \rangle_T$. Now, let suppose:

$$\begin{cases}
\forall t \in \mathbb{R}, c_1(t) = \cos(2\pi f_1 t) \\
\forall t \in \mathbb{R}, c_2(t) = \cos(2\pi f_2 t) \\
\forall T \in \mathbb{R}, \langle s, w_i \rangle_T = \int_0^T s(t) w_i(t) \mathrm{d}t
\end{cases} \tag{3.10}$$

The norm of c_i is:

$$||c_i||_T^2 = \int_0^T \cos(2\pi f_i t)^2 \mathrm{d}t = \frac{T}{2}(1 + \mathrm{sinc}(4\pi f_i T)) \tag{3.11}$$

where sinc is the cardinal sinus. Consider the ratio $x_i = T f_i = T/T_i$ between the time-window length of observation and the period of c_i. Fig. 3.2 gives $\mathrm{sinc}(4\pi x_i)$ function of x_i. The upper bound of the cardinal sinus decreases and tends to 0 when x_i increase. For T at least superior to $\max(T_1, T_2)$, the cardinal sinus is negligible next to one and the norm of both sinus becomes equivalent: $||c_1||_T \sim ||c_2||_T$. In (Eq. 3.9),

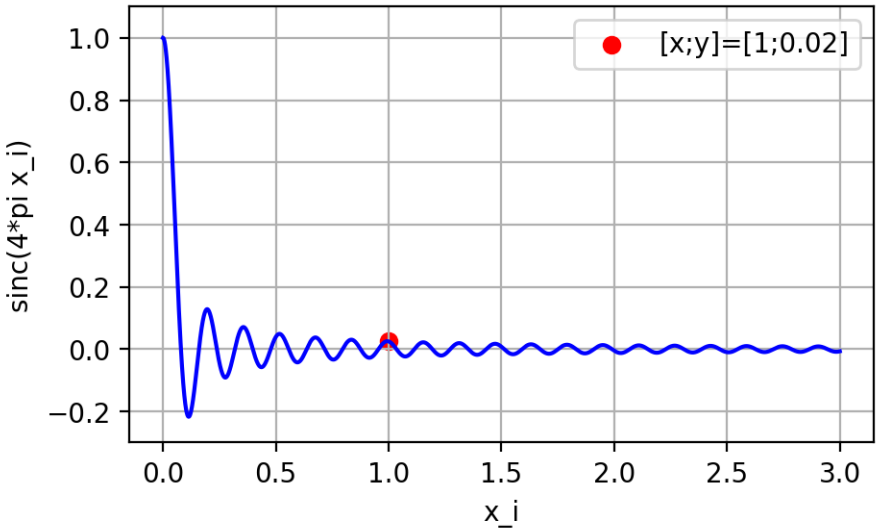


Figure 3.2: Cardinal sinus function of $x_i = Tf_i$.

the parenthesis $(||c_1||_T - ||c_2||_T)$ tends quickly to zero while $(a_1.||c_1||_T + a_2.||c_2||_T)$ is positive and remain bounded. With $T > \max(T_1, T_2)$, $\langle s, c_1 \rangle_T$ is likely greater than $\langle s, c_2 \rangle_T$. Fig. 3.2 shows that the cardinal sinus becomes quickly negligible when the ratio between T and f_i^{-1} increases.

The DN-RBM tends to learn first frequencies with the higher intensity to reduce as much as possible the energy of the model. In both toy examples, the models succeed to learn all the five frequencies with ten hidden units. A larger size of hidden layer is then not required anymore because all the information is learned. In Fig. 3.3b the RMSE stops to decrease significantly after reaching a hidden layer with ten neurons. This observation is coherent with results on Figures 3.4-3.6.

According to previous demonstration and results on toy data, the following lemma is proposed:

Lemma 1. If $s(t)$ is a signal composed of n frequencies:

$$s(t) = \sum_{i=1}^{n} a_n \cos(2\pi f_n t), \tag{3.12}$$

then, at most $2 \times n$ hidden units are sufficient to learn each frequency of the signal with a DN-RBM.

Additional experiences are proposed in Appendix B to evaluate the DN-RBM.

3.3 Signal detection using DN-RBM

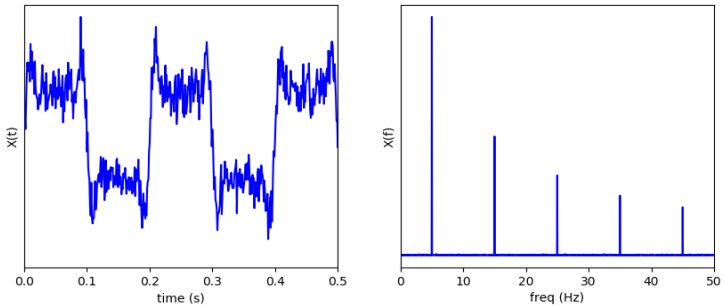
Now, let us focus on the application of signal detection by considering two examples. First, a single channel signal and second, a multiple channels signal.

3.3.1 Single channel example

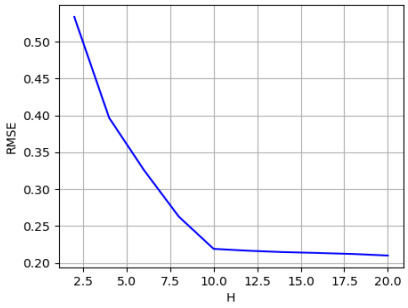
In this example, a single channel signal $X(t)$ is considered (see Fig. 3.11). $X(t)$ is a non stationary signal with two states:

$$\begin{cases} \text{State 1: } X(t) = z(t) \\ \text{State 2: } X(t) = s(t) + z(t) \end{cases} \tag{3.13}$$

where $z(t) \sim \sigma\mathcal{N}(0, 1)$ with $\sigma = 0.3$ is a noisy component and $s(t)$ is a sinusoïdal signal weighted by a Hanning window. The state 2 appears randomly and the goal of this application is to detect the presence of



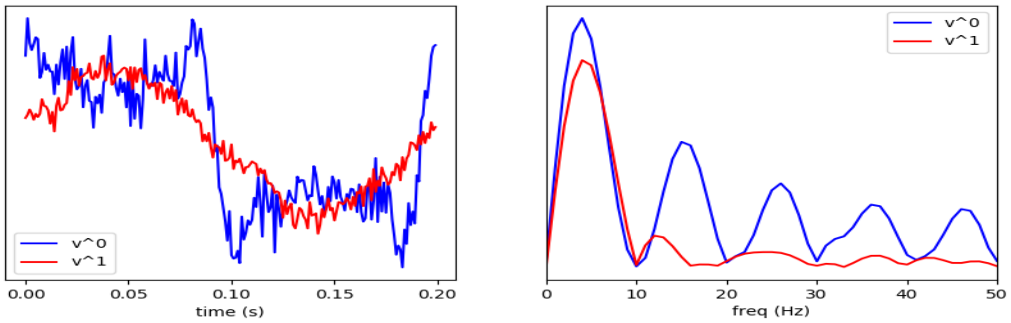
(a) Toy data, first database.



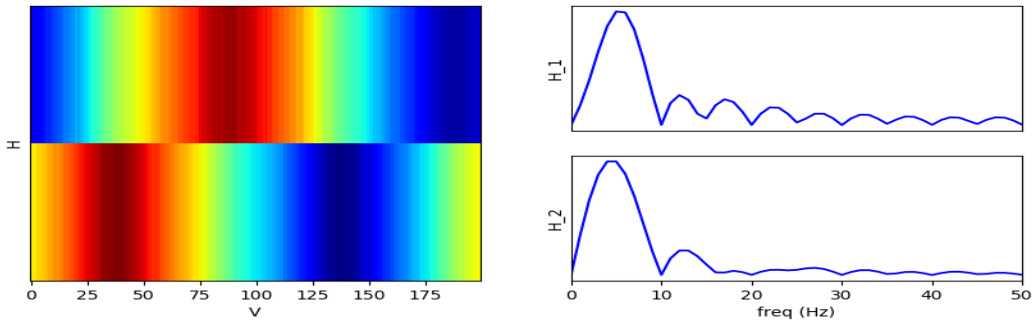
(b) RMSE in function of the hidden layer size

Figure 3.3: Toy data. In **(a)**, the left plot is the temporal signal and the right plot is the Fourier transform of the signal. **(b)** is the result of multiple testing: RMSE in function of the hidden layer size.

$s(t)$. In this experiment, the frequency of $s(t)$ is 25 Hz. A DN-RBM with 2 hidden neurons and 200 visible neurons is trained using a session of 30s of signal. The sampling frequency is 1kHz which corresponds to an observation window of 200ms for the visible layer. Fig. 3.11 gives the result for a test signal. The model manages to find the frequency. The hidden state and the energy level are both promising entries for a classifier to detect the state 2.

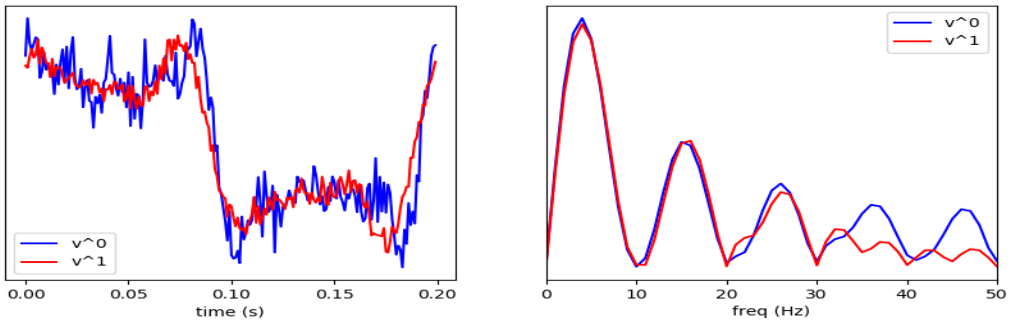


(a) Reconstruction.

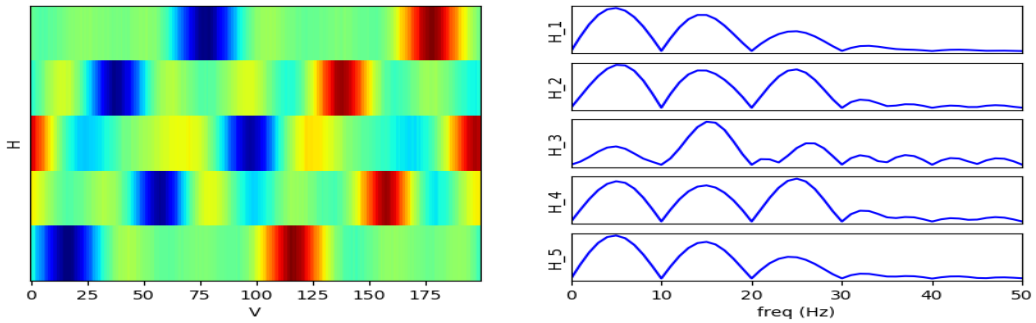


(b) Transfer matrix.

Figure 3.4: Result of training with two hidden units.

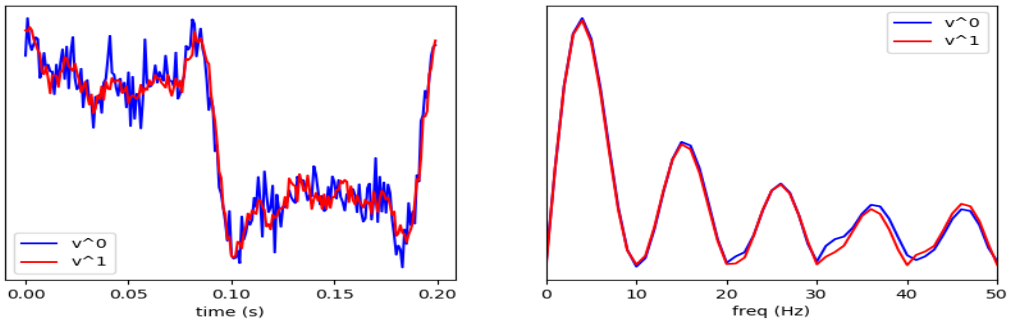


(a) Reconstruction.

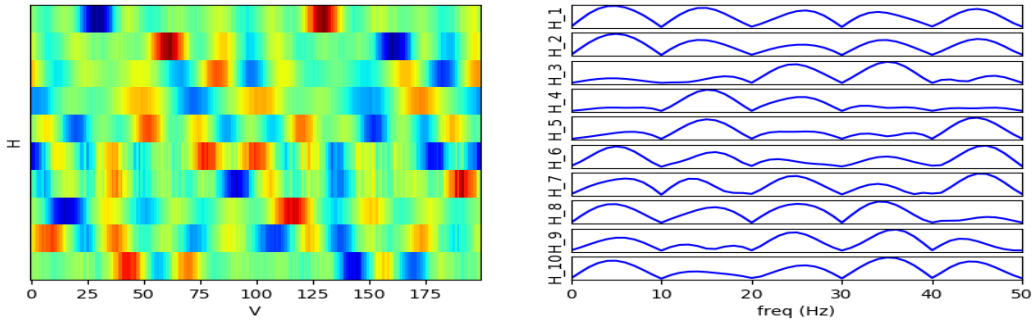


(b) Transfer matrix.

Figure 3.5: Result of training with five hidden units.

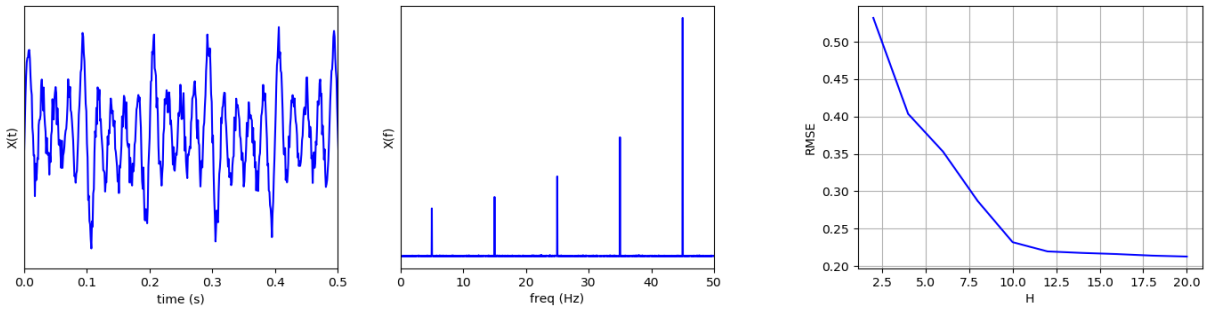


(a) Reconstruction.



(b) Transfer matrix.

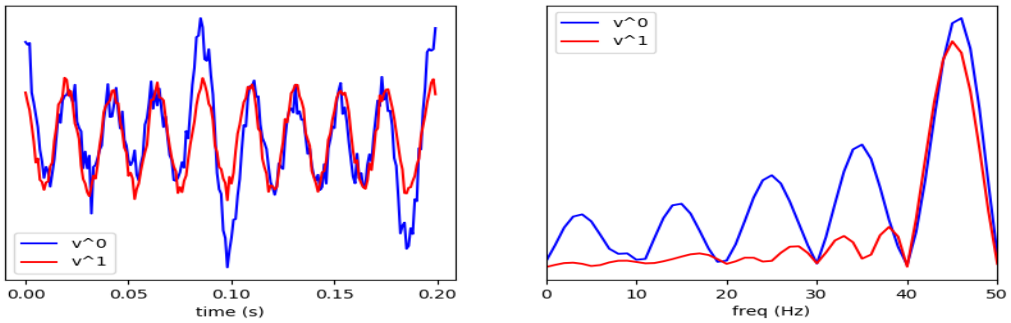
Figure 3.6: Result of training with ten hidden units.



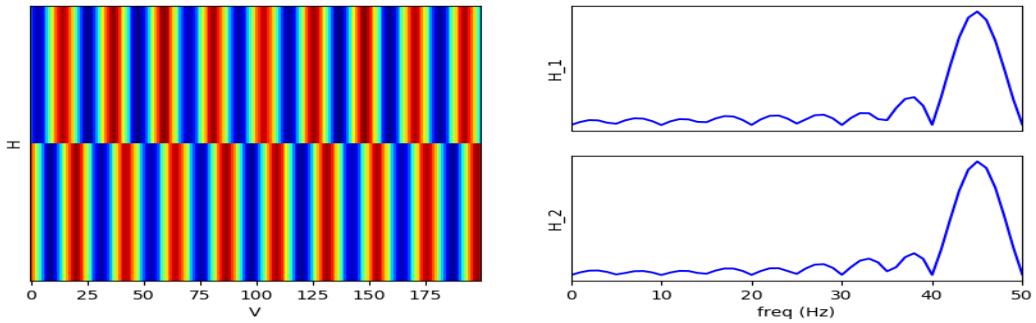
(a) Toy data, second database.

(b) RMSE in function of the hidden layer size

Figure 3.7: Toy data. In **(a)**, the left plot is the temporal signal and the right plot is the Fourier transform of the signal. **(b)** is the result of multiple testing: RMSE in function of the hidden layer size.

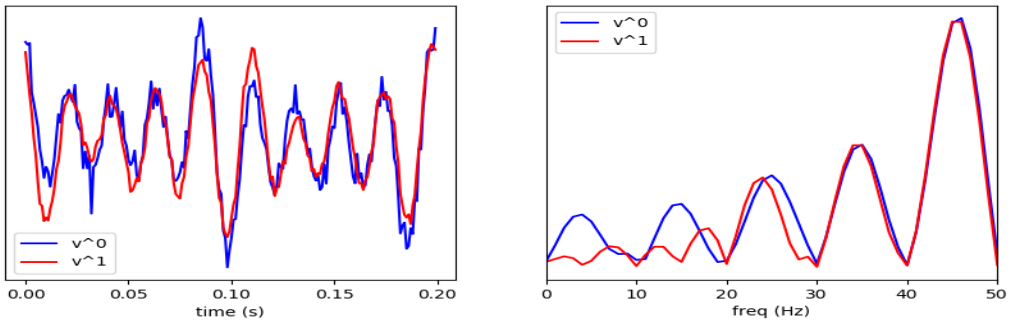


(a) Reconstruction.

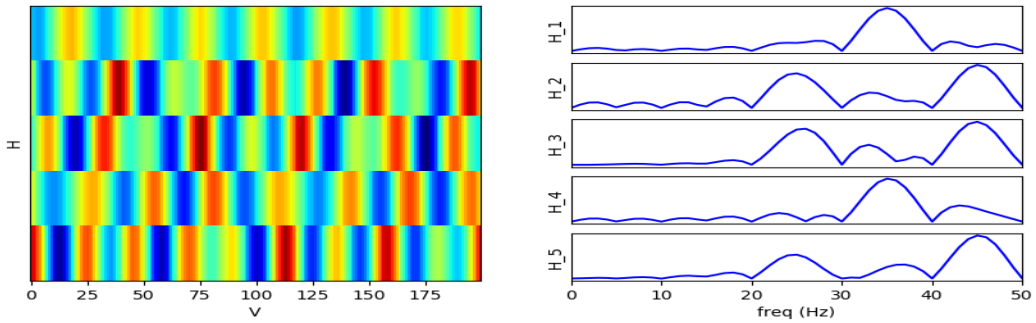


(b) Transfer matrix.

Figure 3.8: Result of training with two hidden units.

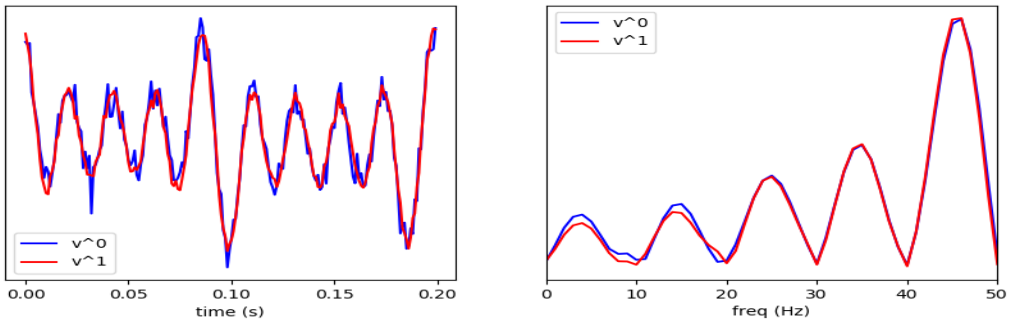


(a) Reconstruction.

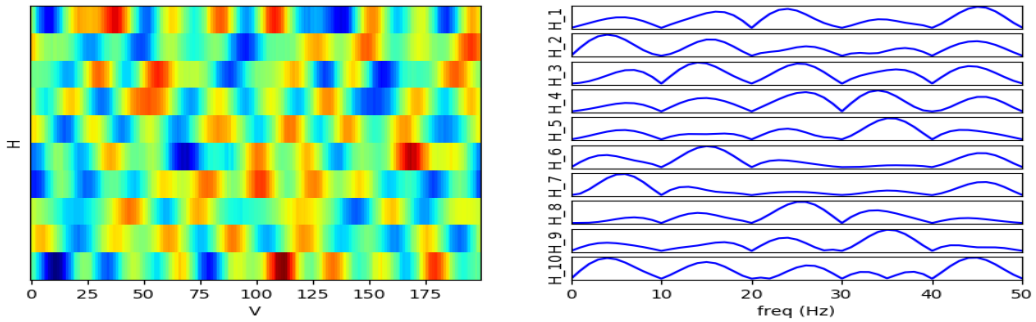


(b) Transfer matrix.

Figure 3.9: Result of training with five hidden units.

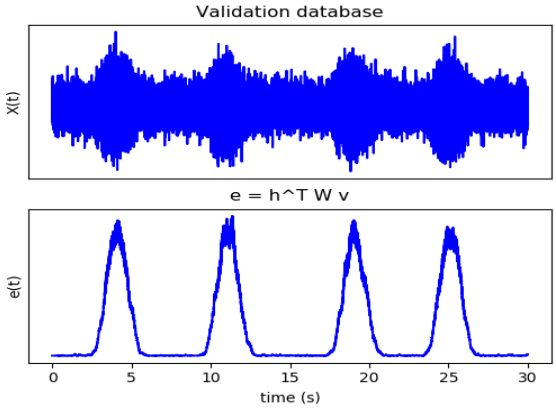


(a) Reconstruction.

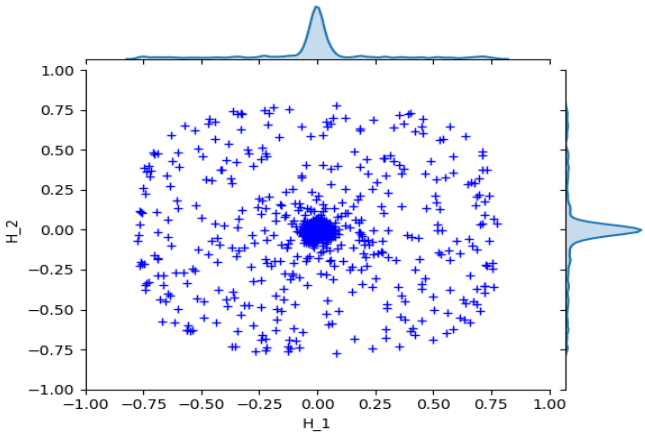


(b) Transfer matrix.

Figure 3.10: Result of training with ten hidden units.



(a) Validation set + $\boldsymbol{h}^T W \boldsymbol{v}$ in time.



(b) Hidden state and marginal distribution.

Figure 3.11: Detection of sinus in toy data. In **(a)**, the validation set and the energy term $\boldsymbol{h}^T W \boldsymbol{v}$ in time are given. In **(b)**, 1000 points randomly selected of hidden state (for visibility) are plotted in the hidden representation space. Marginal distribution over the two dimensions are also given.

Bayes classifier based on energy level

The use of the energy level to detect the state 2 has been proposed in [101]. This methods is based on the computation of the energy terms $e = \boldsymbol{h}^T W \boldsymbol{v}$ and the application of a Otsu's threshold to detect the signal. In this section a Bayes classifier is proposed. As explain in Sec. 3.1, the presence of the learned signal in the visible unit tends to decrease the energy regardless the phase of the signal. Fig. 3.12 is an illustration of signals with the energy term $e(t)$. $e(t)$ is computed using the time-window $\boldsymbol{v(t)} = (X(t), X(t-T), X(t-2T), \ldots, X(t-pT))^T$ where $X(t)$ is the learned signal. p is the order of past observation ($p+1$ points) and T is the sampling period (or lag value) allowing to decrease the number of the visible units while keeping a time-window "large" enough to capture the desired frequencies. The signal $X(t)$ in Fig. 3.12 are real data presented in Chapter 5. The presence of the sinus components are characterized by an increase of energy. During the phase without signal, the energy level decreases and is centered on zero. The idea of the classifier is to model the distribution of energy as two Gaussian distributions. The distribution centered on zero with low variance (no signal) and a Gaussian with a larger standard deviation (see Fig. 3.12c for an example). Let the classifier with the two following states:

$$\begin{cases} \text{State 1: } H_0 := \{e \sim \mathcal{N}_1(\mu_0, \Sigma_0)\} \\ \text{State 2: } H_1 := \{e \sim \mathcal{N}_1(\mu_1, \Sigma_1)\} \end{cases} \tag{3.14}$$

where $\mathcal{N}_1$ is a Gaussian law of dimension 1. The proposed algorithm given in Algo. 1 is an iterative algorithm which uses the Bayes classifier to classify the energy level. At each iteration, the energy level is classified using the current level of classification. The distribution of both class is then estimated. The Bayes classifier is then applied using equation:

$$B^{(i)} = \frac{\Pr(\boldsymbol{h}^{(i)} \in H_0)}{\Pr(\boldsymbol{h}^{(i)} \in H_1)} = \frac{\Phi^c(\boldsymbol{h^{(i)}}; \boldsymbol{\mu_0}, \Sigma_0)}{\Phi^c(\boldsymbol{h^{(i)}}; \boldsymbol{\mu_1}, \Sigma_1)} \underset{H_1}{\overset{H_0}{\gtrless}} \frac{\Pr(H_1)}{\Pr(H_0)} \tag{3.15}$$

to find the new level for the next iteration. $\boldsymbol{h}^{(i)}$ refers to the $i-$th latent representation of the training database and $\Phi^c(.; \boldsymbol{\mu}, \Sigma)$ is the cumulative distribution function of Gaussian $\mathcal{N}(\mu, \Sigma)$. The result gives then the energy level to separate the two classes.

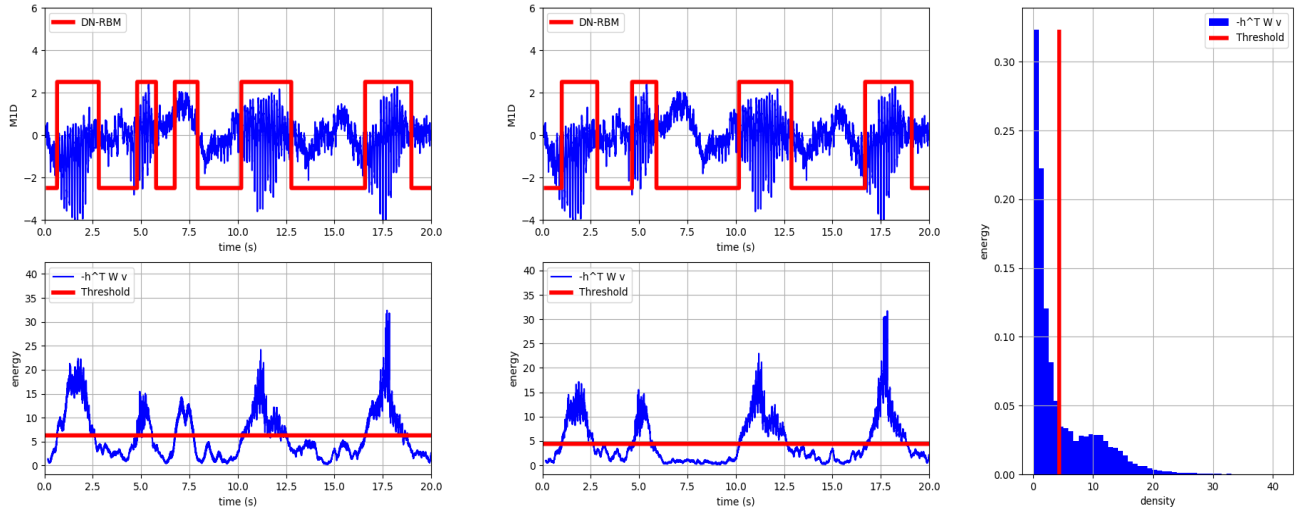
The binary classification based on the energy level is weaken by unsupervised properties of the DN-RBM. The DN-RBM can learn undesirable components of the signal and model can make false detections. In Fig. 3.12a, the presence of non stationary offset has be learned by the DN-RBM. The energy term $e(t)$ increases then in the case of non stationary offset (see the energy level at 7.5 second). To avoid the false detection due to offset, each visible layer is standardized according to:

$$\forall i \in [\![1, \dim v]\!], v_i^* = \frac{v_i}{\sum_{j=1}^{\dim v} v_j} \tag{3.16}$$

Results of the classifier with the new dataset v^* is given in Fig. 3.12b. The DN-RBM succeeds to learn only the useful components.

Bayes classifier based on hidden units

A Bayes classifier using the latent representation can be used to detect the sinus function. From the observation of latent representation in Fig. 3.11b and in Fig. 3.13 it appear clearly it is possible to detect the signal from the hidden state. The hidden representation is modeled by a mixture of a Gaussian distribution and an uniform distribution. As explain in Fig. 3.1, in the case without signals, hidden units will tend to zero. When signals appear, the hidden state starts to turns around the null point. The phenomena can also be applied on DN-RBM with more than two hidden units. The dimension of the distribution of both states



(a) Without standardization. (b) With standardization. (c) Histogram of energy level.

Figure 3.12: Detection of signal with the Bayes classifier (energy level). Energy level in time is also plotted in both case. In case **(a)** (without standardization), the DN-RBM learns the non-stationary offset. The model then detects both sinus signal and the offset (see at 7.5 second). To reduce the number of false detection, the standardization (see **(b)**) automatically remove the offset before the training of the DN-RBM. Signals are intracranial ElectroEncephaloGraphy (iEEG) signals presented in Chapter 5. The figure **(c)** gives the histogram of the energy level of the training set (case with standardization).

is equal to the number of hidden units. The points from the Gaussian distribution are associated to the state 1 (Eq. 3.13) and the points from the second distribution are associated to the state 2. According to observation on toy data (see Fig. 3.11b) and real data (see Fig. 3.13), the second distribution is modeled as an uniform distribution in a ball $\mathcal{R}^n$ (n being the size of the hidden layer) of radius R. Let us denote the null hypothesis and the alternative one:

$$\begin{cases} \text{State 1: } H_0 := \{\boldsymbol{h} \sim \mathcal{N}_n(\boldsymbol{\mu}, \Sigma)\} \\ \text{State 2: } H_1 := \{\boldsymbol{h} \sim \mathcal{U}[\mathcal{R}^n]\} \end{cases} \tag{3.17}$$

Algo. 2 is proposed to learn the classification rule.

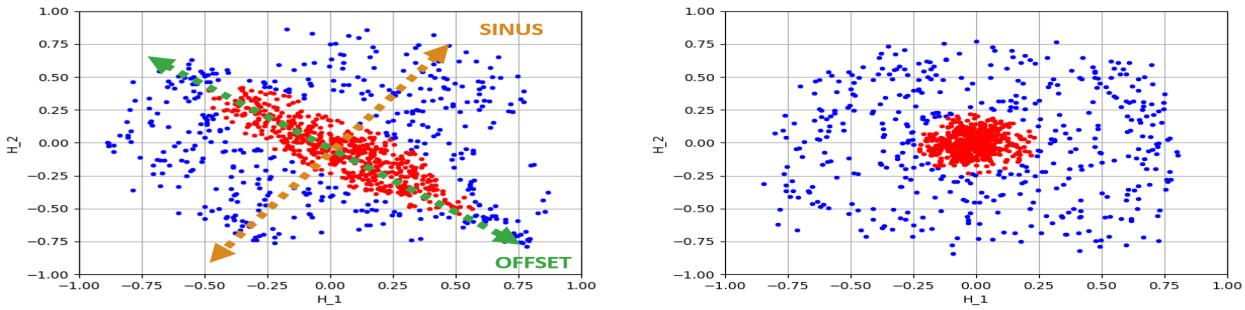
The initialization of label (H_0 or H_1 for each point) is performed by considering (arbitrary) that the 10% quantile closest points to the gravity center $\boldsymbol{\mu}$ over the training set coincide to H_0. The other points are attributed to H_1. First, the radius R (bounding the uniform distribution) is estimated: $\widehat{R} = \max \text{dist}(X, \boldsymbol{\mu})$ where the "dist" function is the Euclidean distance. After initialization, the estimated covariance matrix $\widehat{\Sigma}$ and the proportion $\pi_0 = \Pr(H_0)/\Pr(H_1)$ are sequentially computed. Then, new labels by computing the Bayes factor $B^{(i)}$ of each hidden state $\boldsymbol{h^{(i)}}$ are estimated:

$$\begin{aligned} B^{(i)} &= \frac{\Pr(\boldsymbol{h}^{(i)} \in H_0)}{\Pr(\boldsymbol{h}^{(i)} \in H_1)} = \frac{\Phi^c(\boldsymbol{h^{(i)}}; \boldsymbol{\mu}, \Sigma)}{1/V} \underset{H_1}{\overset{H_0}{\gtrless}} \Pr(H_0) \\ &= V \times \Phi^c(\boldsymbol{h^{(i)}}; \boldsymbol{\mu}, \Sigma) \underset{H_1}{\overset{H_0}{\gtrless}} \pi_0 \end{aligned} \tag{3.18}$$

where $\Phi^c(.; \boldsymbol{\mu}, \Sigma)$ is the cumulative distribution function of Gaussian $\mathcal{N}(\mu, \Sigma)$. The new labels of the hidden

Algorithm 1 Train based classifier on energy level

Require: e : training set.
Require: idx : index vector which covers the full variation of the energy level
  $2\text{level}_0 = (\max(e) + \min(e))/5$
  for epoch $\leftarrow 0$ **to** epochs **do**
    $e_{g1} = e \ s.t. \ e < \text{level}_{\text{epoch}}$: group without signal
    $e_{g2} = e \ s.t. \ e > \text{level}_{\text{epoch}}$: group with signal
    Estimate μ_0, μ_1, Σ_0 and Σ_1
    $\pi_0 = \Pr(H_1)/\Pr(H_0) \approx 1$ (π_0 is fixed to 1 to avoid data hazard and bug)
    label $= \frac{\Phi^c(idx;\mu_0,\Sigma_0)}{\Phi^c(idx;\mu_1,\Sigma_1)} > \pi_0$, $\Phi^c(.;\mu,\Sigma)$ is the cumulative function of Gaussian distribution of mean μ and variance Σ^2
    Deduce $\text{level}_{\text{epoch}+1}$ by dichotomy according to (Eq. 3.15).
  end for
  return $\text{level}_{\text{epochs}}$



(a) Hidden state from data without standardization.

(b) Hidden state from data with standardization.

Figure 3.13: Hidden state from DN-RBM trained with two hidden units with data introduced in Chapter 5. Each point is the latent representation of a multi-channels short-time window (200ms). States with and without sinus signal detected by the Bayes classifier are, respectively, in blue and red. In **(a)**, the DN-RBM learned the apparition of non-stationary offset (see Fig. 3.12a). Observations in real time show the offset has been captured according to the green direction in the latent space (dimension 2). The sinus function is captured according to the orange dotted line. When the signal appears, the hidden state alternates between states on the top right and on the down left and will pass into the area associated to the learned Gaussian distribution. The two states (Eq. 3.13) are inseparable with two hidden units. In **(b)**, however, the standardization allows to remove the offset and only the frequencies of the sinus function has been learned. The two states can be easily separated with two hidden units.

state are given by $B^{(i)}$ for all $i \in [\![1, N]\!]$ where N is the size of the database. The algorithm allows to detect the signal and can be adapted for every size of the hidden layer.

Algorithm 2 Train based classifier on hidden unit

Require: h, n (h: training set, n: dimension of h).
  $\boldsymbol{\mu} \leftarrow \text{mean}(h)$
  $d \leftarrow \text{dist}(h, \boldsymbol{\mu})$
  $\widehat{R} \leftarrow \max(d)$: radius of the ball.
  $V \leftarrow \pi^{\frac{n}{2}} \frac{\widehat{R}^n}{\Gamma\left(\frac{n}{2}+1\right)}$
  $h_{H_0} =$ the 10% closest point from $\boldsymbol{\mu}$.
  for *epoch* $\leftarrow 0$ **to** *epochs* **do**
    $h_{H_0} \leftarrow X_{H_0} - \boldsymbol{\mu}$: centered point.
    $\Sigma \leftarrow \frac{1}{\#h_{H_0}} h_{H_0}^T h_{H_0}$: covariance matrix.
    $\pi_0 \leftarrow \frac{\#h_{H_0}}{\#h - \#h_{H_0}}$
    $label \leftarrow V \times \Phi^c(\boldsymbol{h}; \boldsymbol{\mu}, \Sigma) > \pi_0$
    $h_{H_0} \leftarrow h(label == 1)$
  end for
  return V, $\boldsymbol{\mu}$, Σ, π_0

3.3.2 Multiple channels signal detection

Consider now that DN-RBM input is a time-window of a multi-channel signals. For a given time-window, signals for each channels are concatenated sequentially to get the visible layer (see Fig. 3.14b). To reduce the dimension of the model, observation windows are down sampled. The toy example here is a six channels signal with the two states presented in (Eq. 3.13). But here, some channels contain only noise. A DN-RBM was trained with $n = 2$ which gives the result in Fig. 3.14. Columns of W associated to signal channels capture the frequency and the columns associated to channels without sinus remain null.

The DN-RBM is then able to detect the signal and adapt the detection to each channels.

3.4 Discussion: comparison with the Discrete Fourier Transform

The classical Discrete Fourier Transform (DFT) of a signal $s[t]$ defined for $t \in [\![0, T]\!]$ is given by:

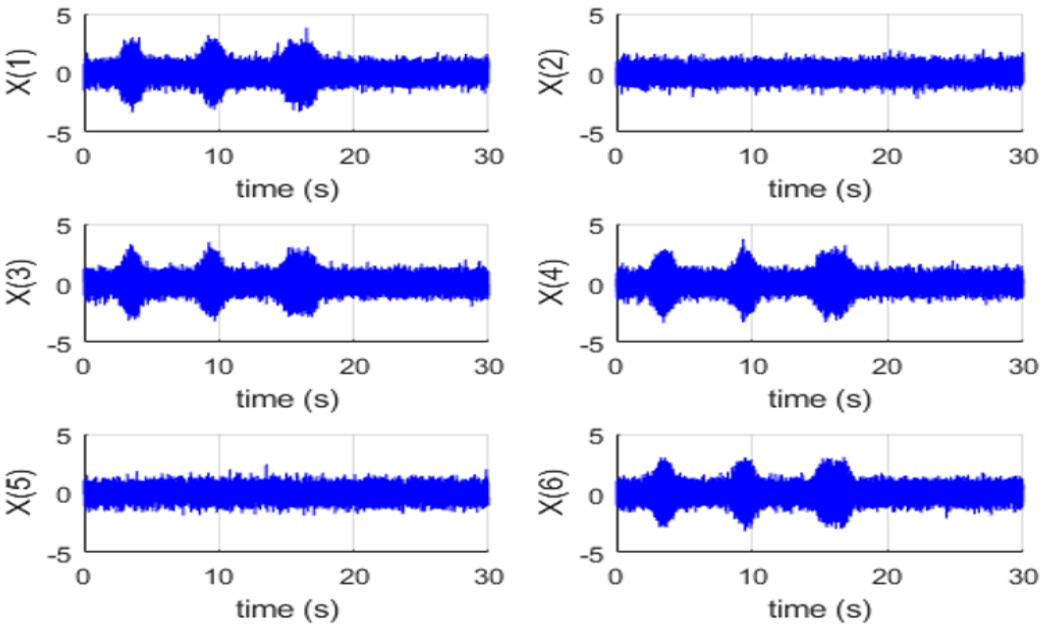
$$S[k] = \frac{1}{T} \left(\langle s, \cos \rangle_k + i \langle s, \sin \rangle_k \right) \tag{3.19}$$

$\langle s, \cos \rangle_k$ is a scalar product between the signal and a sinus signal of frequency k. The result of the scalar product can also be founded into the Fourier series of the signal:

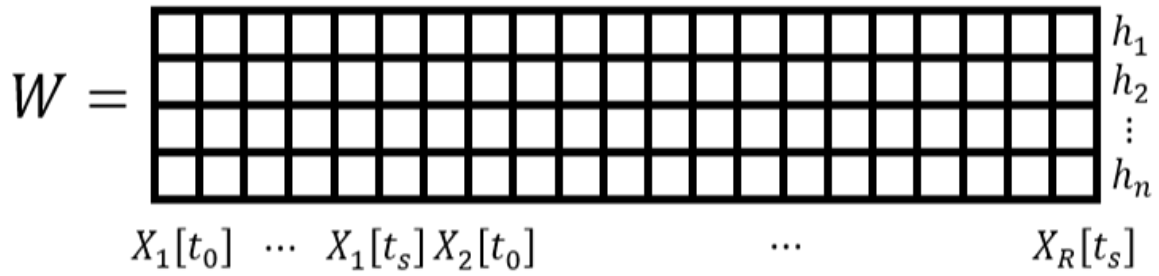
$$s[t] = \sum_k a_k \cos(2\pi f_k t) + b_k \sin(2\pi f_k t) \tag{3.20}$$

where f_k is the $k-$th frequency presents in the signal s, a_k is the scalar product between s and the cosinus of frequency f_k and b_k is the scalar product between s and the sinus of frequency f_k.

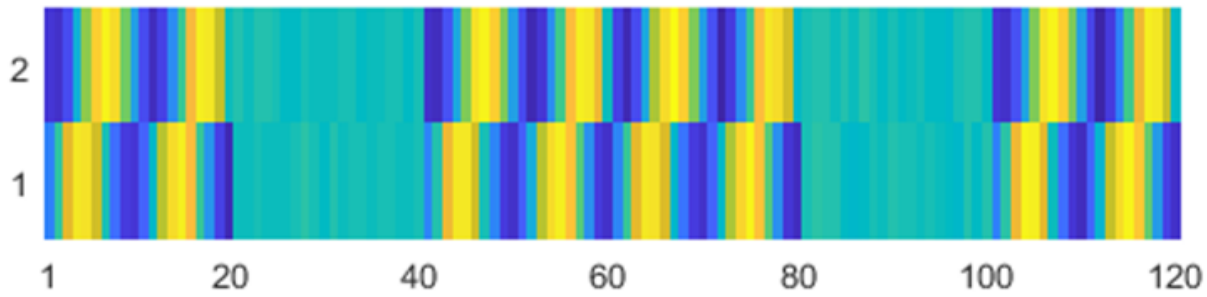
In a DN-RBM, the hidden state gives the image of the scalar product between the signal and the learned components in the transfer matrix W. The imaginary part in the DFT is a second scalar product with the same frequency as the real part but the sinus signal has a phase of $\pi/2$ compared to the real part. The same phenomenon is observed in the DN-RBM: to minimize the energy for each observation from the training set, a DN-RBM with multiple hidden neurons captures the same frequency information under different phase.



(a) Multiple channel toy signal.



(b) Structure of the transfer matrix.



(c) Learned W

Figure 3.14: Detection on multiple channels. A DN-RBM is trained with the data depicted in **(a)** as input. The signal has six channels and channel 2 and 5 have no signal. **(b)** gives the data organization in the transfer matrix W. **(c)** is the learned transfer matrix W. Only the visible units associated to channels with the signal learned the frequency information.

Finally sampling the visible units consists in computing the image of the linear combination of the different learned component times the hidden states:

$$v = f\left(\sum_i h_i \times W_i\right) \tag{3.21}$$

The cos in (Eq. 3.20) is replaced by the transfer matrix. The DN-RBM acts like the Fourier transform but by learning its own decomposition.

Many research on signal processing propose the use of dictionary learning to decompose a signal with a sparse representation [63, 107]. The DN-RBM can be seen as a non linear and stochastic dictionary learning methods where the transfer matrix is learned and the hidden representation is bounded. The properties of the DN-RBM offer a variety of advantages and possible improvement approaches are currently available. DN-RBM is an unsupervised generative model that can learn the optimal frequencies to detect and can be used as a classifier or predictor. DN-RBM can extract uncorrelated components but separability depends on the data. A model architecture study is needed to help hidden units to separate components in function of the application.

Chapter 4

Diffusion Network

This chapter presents an original contribution for the research on DN. First, a new training procedure based on gradient descent is proposed. The equation is rewritten to simplify the learning of parameters and several approximations are proposed. Finally, the model has been validated on toy data: reconstruction of missing channels, prediction with a vector DN and prediction using multiple DNs from an unique observation.

4.1 Training the Diffusion Network

4.1.1 Module

The DN consists in modelling the variation of a $d-$dimensional stochastic process (or path) $\boldsymbol{X} = (\boldsymbol{X}(t))_{t \leq T}$ as a system of stochastic differential equation (see (Eq. 2.64)). Let λ be the set of parameters and $\mathcal{L}^{\lambda}(\boldsymbol{X})$ (according the Girsanov's theorem) the likelihood function of the set of parameters λ. Then

$$\mathcal{L}^{\lambda}(\boldsymbol{X}) = \exp\left[\sum_{j=1}^{d} \frac{1}{\sigma_j^2}\left(\int_0^T \mu_j\left(\boldsymbol{X}(t), t\right) \mathrm{d}X_j(t) - \frac{1}{2}\int_0^T \mu_j\left(\boldsymbol{X}(t), t\right)^2 \mathrm{d}t\right)\right]. \tag{4.1}$$

Training a DN consists in finding the SDE which gives the best description of the data. Let us give a fixed path $\boldsymbol{X}$ as the training set. The derivation of the log-likelihood for each parameter λ_i is given by:

$$\frac{\partial \log \mathcal{L}^{\lambda}(\boldsymbol{X})}{\partial \lambda_i} = \frac{1}{\sigma_j^2}\sum_{j=1}^{d}\left(\int_0^T \frac{\partial \mu_j(\boldsymbol{X}(t), t)}{\partial \lambda_i}\mathrm{d}X_j(t) - \int_0^T \frac{\mu_j(\boldsymbol{X}(t), t)}{\partial \lambda_i}\mu_j(\boldsymbol{X}(t), t)\mathrm{d}t\right). \tag{4.2}$$

The λ_i are solutions of the stochastic equations if $\forall i, \frac{\partial \log \mathcal{L}^{\lambda}(\boldsymbol{X})}{\partial \lambda_i} = 0$. In the literature, Movellan *et al.* proposed to maximize the log-likelihood using MCEM algorithm [73]. In this study, the gradient descent algorithm is proposed to train the DN. But first, the drift function is rewritten to simplify its expression:

$$\begin{aligned}\mu_j(\boldsymbol{X}(t), t) &= \kappa_j\left(-\rho_j X_j(t) + \xi_j + \sum_{i=1}^{d} W_{ij} S_i(t)\right) \\ &= -\tau_j X_j(t) + \psi_j + \sum_{i=1}^{d} \omega_{ij}\phi(X_i(t)).\end{aligned} \tag{4.3}$$

Parameters τ_j, ψ_j and ω_{ij} are, respectively, the product of ρ_j, ξ_j and W_{ij} with κ_j. The writing of the drift function in (Eq. 4.3) allows to highlight two terms in the SDE. The first term is a linear product between the signal $X_j(t)$ and the constant τ_j. The RC filter inside the neuron structure (see Fig. 2.16) is responsible

of this term: $\frac{1}{R_j C_j} = \kappa_j \rho_j = \tau_j$. The second term quantifies the influence of the network on each channels. It is the product of the transfer matrix and the signal values filtered by ϕ. $\phi(.)$ is the activation function of the model. Initial works on DN proposed the use of sigmoïd function but a different activation has been tested during this thesis.

In real life problem, continuous time data are not available, only sampled signals are. The two integrals inside the density $\mathcal{L}^\lambda(\boldsymbol{X})$ (Eq. 2.58) are Itô integrals (cf. Sec. 2.4.1). Sampled signal $X(t)$ is a piece-wise constant function. The integral of the drift term is equal to the discrete sum :

$$\begin{cases} \int_0^T \mu_j^2(\boldsymbol{X}(t), t)\mathrm{d}t = \sum_{k=0}^{s-1} \mu_j(\boldsymbol{X}(t_k), t_k)^2 \Delta t, \\ \int_0^T \mu_j(\boldsymbol{X}(t), t)\mathrm{d}X_j(t) = \sum_{k=0}^{s-1} \mu_j(\boldsymbol{X}(t_k), t_k) \left(X_j(t_{k+1}) - X_j(t_k)\right), \end{cases} \tag{4.4}$$

with $0 = t_0 < t_1 < \ldots < t_s = T$ and $t_{k+1} = t_k + \Delta t$, Δt being the sampling period. The log likelihood defined in (Eq. 4.1) becomes:

$$\begin{aligned} \log \mathcal{L}^\lambda(\boldsymbol{X}) &= \sum_{j=1}^{d} \sum_{k=0}^{s-1} \left(\frac{1}{\sigma_j^2} \mu_j(\boldsymbol{X}(t_k), t_k) \left(X_j(t_{k+1}) - X_j(t_k)\right) - \frac{\Delta t}{2\sigma_j^2} \mu_j(\boldsymbol{X}(t_k), t_k)^2 \right) \\ &= \sum_{k=0}^{s-1} \sum_{j=1}^{d} \left(\frac{1}{\sigma_j^2} \mu_j(\boldsymbol{X}(t_k), t_k) \left(X_j(t_{k+1}) - X_j(t_k)\right) - \frac{\Delta t}{2\sigma_j^2} \mu_j(\boldsymbol{X}(t_k), t_k)^2 \right) \\ &= \sum_{k=0}^{s-1} M(\boldsymbol{X}(t_{k+1}), \boldsymbol{X}(t_k), t_k) \end{aligned} \tag{4.5}$$

where $M(\boldsymbol{X}(t_{k+1}), \boldsymbol{X}(t_k), t_k)$ is a term named *module* for now equal to the contribution at the instant t_k of the log density $\log \mathcal{L}^\lambda(\boldsymbol{X})$. Maximizing $\log \mathcal{L}^\lambda(\boldsymbol{X})$ is equivalent to maximizing each module $M(\boldsymbol{X}(t_{k+1}), \boldsymbol{X}(t_k), t_k)$ independently. Writing the loss function as a sum of module is useful, for the learning procedure, to use mini-batch gradient descent algorithm (See Appendix A).

4.1.2 RC filter and noise power estimation

The drift function μ_j (see (Eq. 4.3)) is composed of two terms. The first term is a linear term $-\tau_j X_j(t)$. If the other terms of the SDE in the DN are ignored, (Eq. 2.64) leads to a classic linear first order differential equation:

$$\frac{\mathrm{d}X_j(t)}{\mathrm{d}t} = -\tau_j X_j(t) \tag{4.6}$$

This equation has an evident solution:

$$X_j(t) = X_j(0) \exp(-\tau_j t), \text{for } t \geq 0. \tag{4.7}$$

For $\tau_j > 0$, the first term forces the sampled signal to converge toward zero more or less faster depending of $|\tau_j|$. Sampling the signal with the DN diverges if $\tau_j < 0$. The first term of the μ_j allows the model to remain stable.

The second term $(\sum_i \omega_{ij} \phi(X_i(t)))$ is the nonlinear combination of the network which competes with the first term by adding a disturbance caused by the global state of the network to the signal.

In practice, the learning algorithm using Girsanov's theorem fails to find the parameter τ_j. Various procedures to learn the parameter τ_j have been tested without success. The training step encourages τ_j to be very small and the model fails to perform good results in the proposed applications. The proposed solution in this thesis consists in estimating τ_j directly from the signal $X_j(t)$. The structure of the neurons

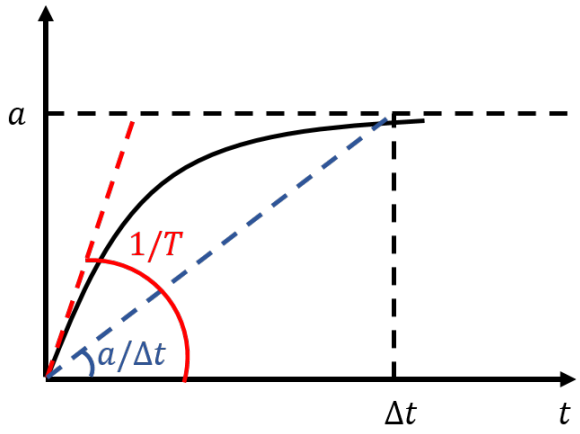


Figure 4.1: Step answer of the RC filter. T is the time constant of the filter. The choice of $\tau = 1/T$ is adapted to make the system able to converge close to a after the time Δt.

is composed of a RC filter (see Fig. 2.16). The parameter τ_j is the inverse time constant of the filter. An estimation based on the step answer of the filter is then proposed. Let $(\boldsymbol{X}(t_k))_{k\in[\![0,s]\!]}$ be the training sequence and Δt be the sampling period. The RC filter in the neuron's structure is a first order system, the step order is given in Fig. 4.1 where a is the command (see Fig. 4.1) or the step input. The initial slope of the answer of the RC filter is equal to τ_j and the command a is the mean absolute speed of the signal over the training set:

$$a = \frac{1}{s}\sum_{k=0}^{s-1}\left|\frac{X_j(t_{k+1}) - X_j(t_k)}{\Delta t}\right| \tag{4.8}$$

The estimation of τ_j is then given by:

$$\widehat{\tau}_j = \alpha \times a = \frac{\alpha}{s}\sum_{k=0}^{s-1}\left|\frac{X_j(t_{k+1}) - X_j(t_k)}{\Delta t}\right| \tag{4.9}$$

where α is a hyper-parameter which measures how fast the RC filter converges. For example, choosing $\alpha = 5$ allows the filter to converge to 99% towards the command in one time step.

In addition to the parameter τ_j, each neuron has a constant noise power σ_j^2. The noise power gives the variance of $\mathrm{d}X_j(t)$. The standard deviation σ_j can easily be estimated for each neuron j:

$$\widehat{\sigma_j} = \sqrt{\frac{1}{s}\sum_{k=0}^{s-1}\left(X_j(t_{k+1}) - X_j(t_k)\right)^2} \tag{4.10}$$

4.1.3 Hidden units

Hidden neurons have various names: observators (or software sensor), latent variables, hidden states, ... Hidden neurons are not strictly necessary in DN [73]. They are expected to learn the hidden structure of the sequential data distribution itself.

The use of hidden units allows a synthesis of the data to define latent characteristics of the system. However, using hidden units requires an additional step during the learning phase: sampling the *hidden path* before to compute the loss function. The maximization of the log-likelihood $\log \mathcal{L}^{\lambda}(\boldsymbol{X})$ is then based on the visible path and the sampled hidden path. The hidden path is sampled at each epoch with the (Eq. 2.67). Note that there is no rule for choosing the noise power value σ_j and the RC filter parameter τ_j for each hidden neuron.

Algorithm 3 Training of a DN

Require: $epochs, bs, \eta$: epoch number, batch size and learning rate.
  Estimate the noise power of each neuron σ_j. (Eq. 4.10)
  Estimate the RC filter of each neuron τ_j. (Eq. 4.9)
  for $epoch \leftarrow 0$ **to** $epochs$ **do**
    Select randomly a mini batch t_k of size bs.
    Sample hidden neurons (if any).
    $loss \leftarrow 0$
    for $k \leftarrow 0$ **to** bs **do**
      $loss \leftarrow loss - M(\boldsymbol{X}(t_{k+1}), \boldsymbol{X}(t_k), t_k)$
    end for
    Compute gradient for each parameters.
    Update parameters.
    Validation.
  end for

4.1.4 Constraint on the transfer matrix

The DN proposed by Movellan in [73] is a full connected network (see Fig. 2.4a), *i.e.* all neurons are connected to each other. The link between a neuron and itself is problematic during the learning. Experiments on real signals show the diagonal of the transfer matrix (or link from a neuron to itself) tends to increase significantly in comparison of the other coefficients if parameters τ_j for all j are fixed to large value. This phenomenon decreases the performance of the model. No mathematical demonstration has been found to explain this problem. The signal $X_j(t)$ in the $j-$th equation of the system of SDEs appears in the first two terms:

$$\mu_j(X(t), t) = -\tau_j X_j(t) + \omega_{jj}\phi(X_j(t)) + ... \tag{4.11}$$

A large and positive value of ω_{jj} counterbalances the linear term $-\tau_j X_j(t)$. The bounded nature of the activation function $\phi(.)$ prevents the signal $X_j(t)$ to diverge in case of large and positive ω_{jj}. This phenomenon is observed when the training of the DN is not stopped just after the convergence of the loss function. A solution to avoid the problem of convergence is to apply a null constraint on the diagonal of the transfer matrix. This is equivalent to consider a graph without link self connections.

The learning algorithm of a DN is given in Algo. 3.

4.1.5 Learning the activation function

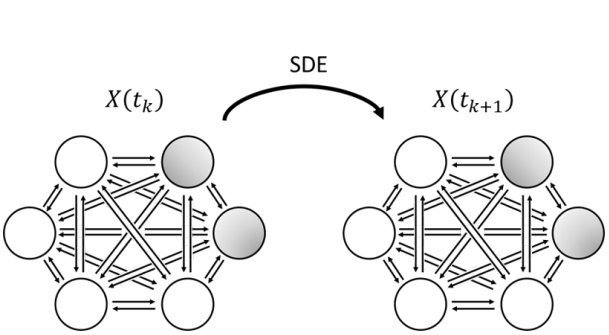
In the drift function (see (Eq. 4.3)), signals are filtered by the activation function ϕ. The filter ϕ is arbitrary chosen by the user. A parameter v_i is added to adapt the activation function for each neuron. The drift function becomes:

$$\forall j \in [\![1, d]\!], \mu_j(X(t), t) = -\tau_j X_j(t) + \psi_j + \sum_{i=1}^{d} \omega_{ij}\phi(v_i X_i(t)). \tag{4.12}$$

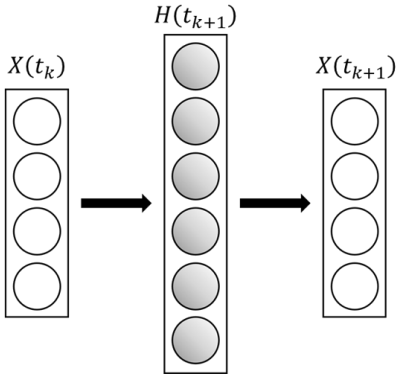
In the next section, DNs will be evaluated using or not the adaptive parameter of the activation function $\boldsymbol{V} = (v_1, \ldots, v_d)^T$.

4.2 Evaluation of DNs: applications on toy data

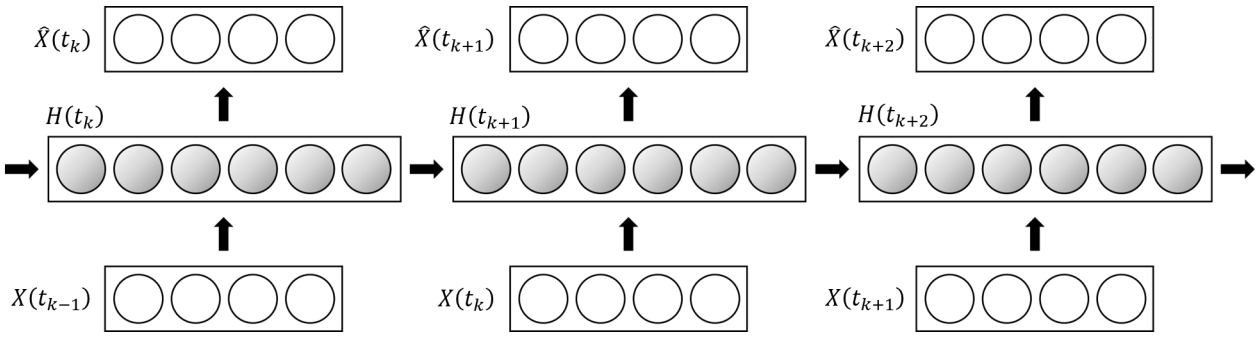
This section is dedicated to the evaluation of DNs and the comparison with other models. Two applications are proposed to evaluate the performance of the DN: the reconstruction of missing channels and the



(a) DN graph.



(b) MLP graph.



(c) RNN graph.

Figure 4.2: Graphs for the evaluation of DNs and comparison. Grey neurons are hidden neurons (optional for the DN). The goal for each model is to estimate the next state of the data.

prediction of time-window signal presenting a vector of observation. Results from DNs are compared with results from MLP and the RNN (see Fig. 4.2). For each models, the same input/output data is used.

4.2.1 Toy model

Toy data are simulated by a DN. The proposed DN is composed of four neurons as depicted in Fig. 4.3a. The channel X_1 is first sampled. Then, the other channels are generated by the DN according to the SDEs. Each neuron of the DN has the parameter $\tau_j = 100$ and $\sigma_j = 10^{-2}$ $\forall j$. The activation function is linear. The first channel X_1 is a non stationary sinus composed of two frequencies: 10Hz and 20Hz. The sinus signal $s(t)$ appears only during 1.5 second and is then filtered by an Hanning window:

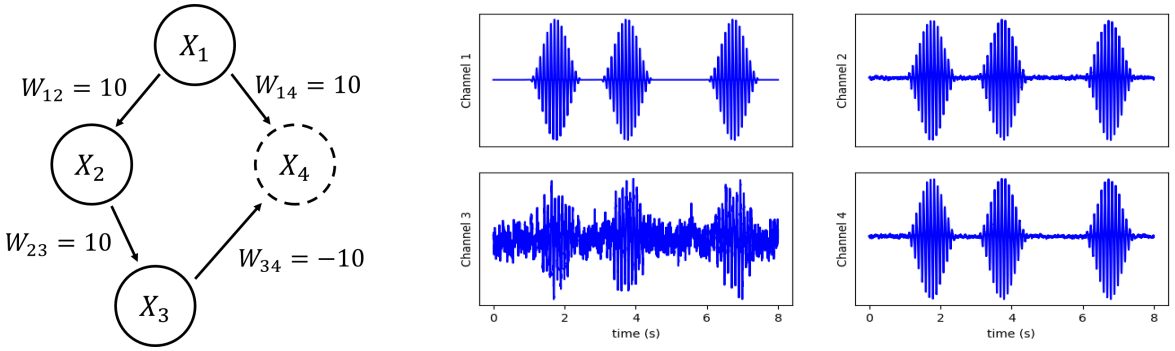
$$s(t) = 0.5\,\mathrm{hann}(t) \times [\sin(2\pi 10t) + 0.5\sin(2\pi 20t)] \tag{4.13}$$

The sampling frequency is 1kHz and the validation set is given in Fig. 4.3b.

The procedure of generation of the toy signals ensures the presence of a diffusion between the channels and the global signal form is close to the process presented in Chapter 5.

4.2.2 Application 1: Missing channels reconstruction

The DN is a generative model. This application consists in evaluating the capacity of the model to generate (or reconstruct) a missing neuron signal according to the other neurons. After the training procedure, the information of one (or more) neuron(s) is removed. The objective is to reconstruct the missing neurons according to the other and past observations of the network.



(a) Graph structure. (b) Validation session

Figure 4.3: Toy model. (a) is the graph structure of the toy data. (b) is an example of time-series sampled using the model.

Let $(\boldsymbol{X}(t_k))_{k\in[\![0,s]\!]} \in \mathbb{R}^p$ be a stochastic process. At each time t_k, the input vector $\boldsymbol{X}(t_k)$ is used to predict the next sample $\boldsymbol{X}(t_{k+1})$. Once the missing neurons in $\boldsymbol{X}(t_{k+1})$ are retained for the next prediction, the true value of the other neurons is taken for the prediction. The result with a DN is given in Fig. 4.4a. The DN is trained without hidden unit and the activation function is learned. The fourth channel only is removed and reconstructed according to the three firsts.

Four models are compared: DNs (with and without learning the activation function), a MLP and a RNN. For each model, the RMSE is evaluated as:

$$\text{RMSE}_{app1} = \frac{1}{\#\text{missing channels}} \sum_{j\in\text{missing data}} \sqrt{\frac{1}{T}\sum_{k=0}^{s-1}\left(\widehat{X_j}(t_k) - X_j(t_k)\right)^2} \tag{4.14}$$

where "#missing channels" is equal to the number of missing channels (= 1 in this case of study). Results are given in Table 4.1. The hyper-parameter value for the DN are given in Table 4.1, such as the number of hidden units, the learning parameters etc. The number of parameters refers to the number of parameters trained with the gradient descent. The estimation of τ_j and σ_j for all j in the DN's case is not included in this counting.

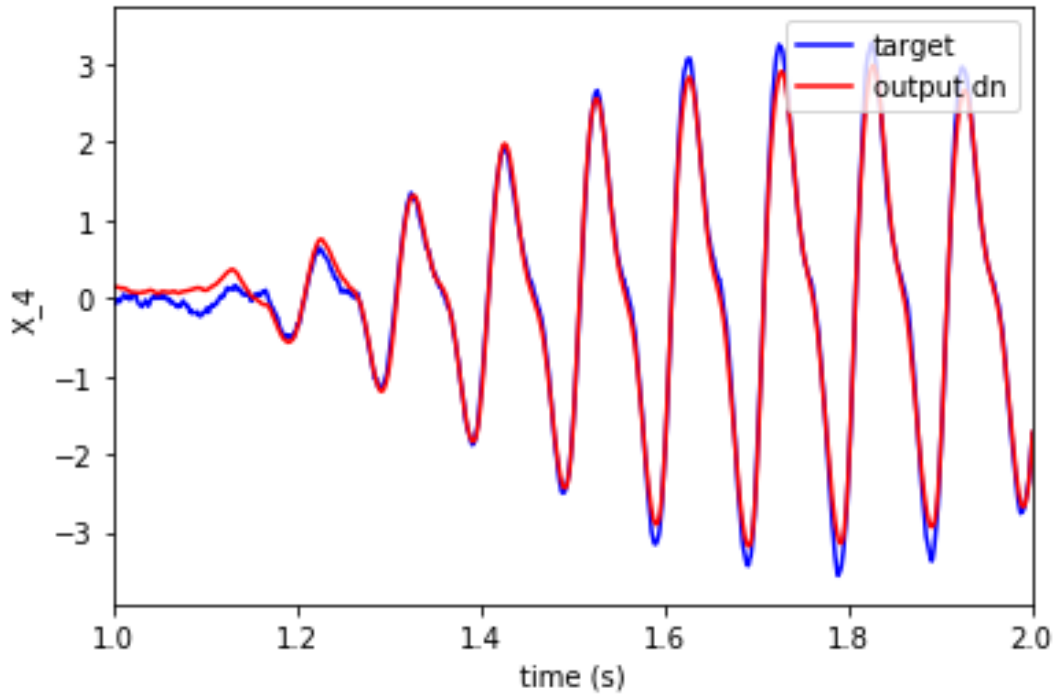
The DNs is very good for missing channel reconstruction. Technically, the result with the MLP is the best but note that there is no hidden unit with the DN and less parameters to learn. Finally, the result with the DN with the learning of the activation function is better than the case without the learning of the activation function.

4.2.3 Application 2: Time prediction

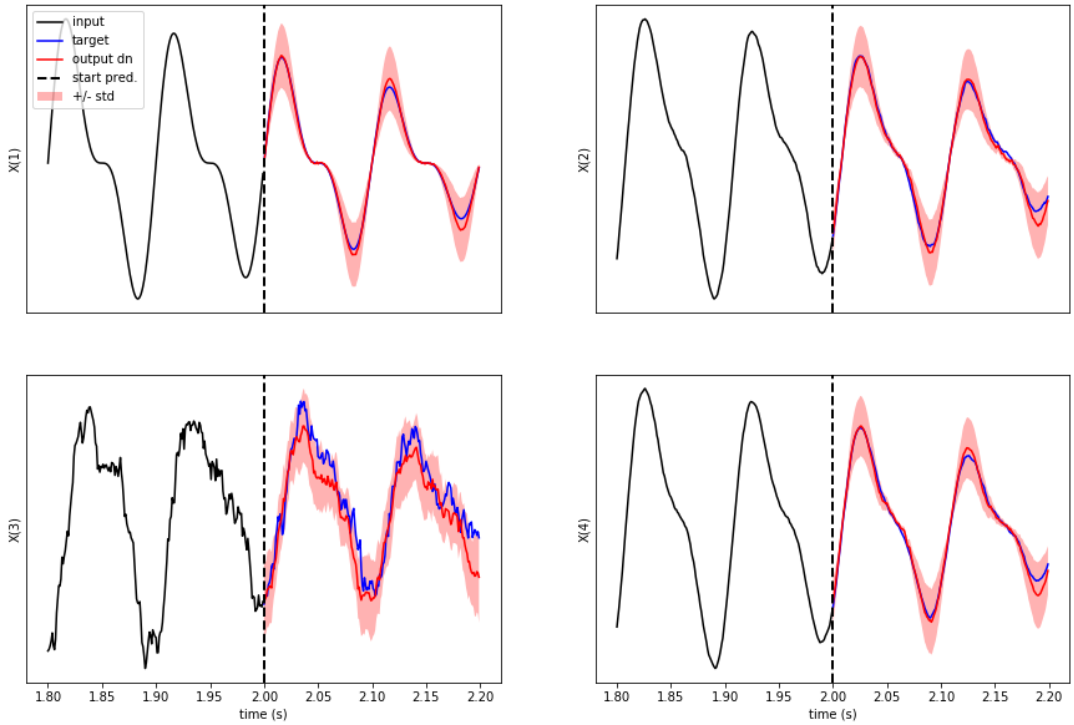
The sampling rule given in (Eq. 2.67) allows to predict $X(t_{k+1})$ according to previous samples $X(t_k)$. The multidimensional signal $(X(t_k))_{k\in[\![0,s]\!]} \in \mathbb{R}^4$ is reorganized into the following database:

$$\left(\widetilde{X}(t_k)\right)_{k\in[\![0,\lfloor s/N_{\text{sequence}}\rfloor]\!]} \in \mathbb{R}^{4\times N_{\text{sequence}}}$$

where $\widetilde{X}(t_k)$ is a sequence of observations. The sampling period of this new database is equal to $\widetilde{\Delta t} = N_{\text{sequence}} \times \Delta t$. Graphs used for prediction for both models are the same as those presented in Fig. 4.3b where X becomes $\widetilde{X}$. Fig. 4.4b gives the result for the DN with learning of the activation function. The black signal is the initial state of the network. The blue signal is the expected state after sampling and the red signal is the result obtain with the model. The red area is computed using the noise power of the DN.



(a) Missing channel reconstruction.



(b) Prediction of time-window.

Figure 4.4: Tests on toy data with the DN. **(a)** The fourth channel is reconstructed using a DN. The blue signal is the true signal and the red signal is the signal sampled using the DN according to the state of the other neurons. **(b)** Prediction of time-window, toy data. The black signal is the initial state of the network. The blue signal is the expected state after sampling and the red signal is the result obtain with the model. The red area is computed using the noise power of the DN.

Table 4.1: Toy model results: missing channels reconstruction.

	DN (1)	**DN** (2)	**MLP**	**RNN**
α **(DN only)**	5.0	5.0	-	-
Number of hidden units	0	0	5	5
Number of parameters	16	20	49	74
Activation function (hidden units only)	tanh	tanh (learned)	elu linear	elu linear
RMSE	5.011E-01	4.174E-01	2.983E-01	6.034E-01

Table 4.2: Toy model results: prediction of time-window.

	DN (1)	**DN** (2)	**MLP**	**RNN**
α **(DN only)**	1.0	1.0	-	-
Number of hidden units	0	0	10	10
Number of parameters	640,000	640,800	16,810	16,910
Activation function (hidden units only)	tanh	tanh (learned)	elu linear	elu linear
RMSE	4.083E-01	3.535E-01	4.442E-01	3.430E-01

Results are given in Table 4.1. At time t_k, $\widetilde{X}(t_k)$ is used to predict the next sample $\widetilde{X}(t_{k+1})$. The RMSE used to evaluate each models is defined as:

$$\text{RMSE}_{app2} = \sqrt{\frac{1}{\lfloor s/N_{\text{sequence}} \rfloor \times d \times N_{\text{sequence}}} \sum_{k=1}^{\lfloor s/N_{\text{sequence}} \rfloor} \sum_{j=1}^{d \times N_{\text{sequence}}} \left(\widehat{\widetilde{X}_j}(t_k) - \widetilde{X}_j(t_k) \right)^2} \tag{4.15}$$

In this experience, the RNN got the best results. Results between all models are close but, the number of parameters of the DN is much more larger than for the other models. According on Fig. 4.4b, DNs predicts successfully a time-window of the toy model.

4.3 Discussion: limit and path of improvement

The learning procedure of the DN has been revisited in this chapter. Parameters of the model has been rewritten to simplify the learning procedure and estimations for the RC filter. An estimation of the noise power of each neuron has been proposed. The comparison with the other model with the toy data shows the DN is functioning. The choice of the neuron structure and the number of hidden units for the other models is not optimal. With more investigation, it is possible to get better results. But the idea was to keep an equivalent number of neurons between models to remain comparable. The DN is able to handle the different applications without hidden units.

The DN, however, retains some defaults that deserve more investigations. First, the learning procedure using gradient descent algorithm and Girsanov's theorem fails to learn the RC filter parameters enabling a good performance on applications. An estimation based on the mean absolute speed of signals has been proposed. This estimation requires an additional hyper-parameter α which has a major influence on the result. Second, results on toy data show that adding hidden neurons in the DN does not improve significantly the performances of the model. In addition, the hidden units are not available in the training database, there is no rule to estimate the RC filter and the noise power of hidden units. The estimation proposed in the tests on toy data consisted in taking the mean over visible units.

Despite these defaults, notice that unlike other families of ANNs, DNs are largely unknown. And, in the same way that many models originally proposed for feedforward neural network have inspired researchers to

propose extensions to BMs, it stays possible to apply again these different approaches for DNs. Additional tests on DNs will be proposed on Chapter 5 on real data.

Chapter 5

Application: iEEG data analysis for the control of the deep-brain stimulation

This works is originated from a French-Taiwan collaboration between the UEVE, the University of Paris-Saclay and the NTHU.

5.1 Introduction

The Parkinson's Disease (PD) is a progressive degenerative disease described for the first time by James Parkinson in 1817 [82]. The causes of the disease remain unknown today. Around 0,2 % of the population are affected by the disease. This ratio increases to 1 % for people above 60 y.o. [108]. The number of people living with PD is expected to double in 25 years [19]. This increase is partly the result of the ageing increase of the population.

PD is very well known for it symptoms: tremor, slowness of movements or absence of postural reflexes. Symptoms of the PD are attributed to the degeneration of dopaminergic neurons in the *BAsal GAnglia* (GABA). The GABA circuitry is a neural loop between the cortex and different brain regions. It is playing a key role of regularization in the control of body movements. The state of neurons from the *Substantia Nigra pars compacta* (SNpc) activates the dopaminergic receptor D1 and inhibits the dopaminergic receptor D2 of the striatum. For patients with PD, the degeneration of the SNpc would be responsible of the abnormal activity of neurons in the GABA and, therefore, the symptoms. Fig. 5.1 (according to [57, Chapter 19]) gives the GABA biological pathway in both patient's state (without and with the Parkinson disease). The state of the patients continues to deteriorate in time. Four phases has been identified to characterize the progress of the disease.

1. **The first expression of the symptoms** is not strong enough yet to diagnose the disease with certainty. There is no specific test for diagnosing PD.

2. **The Parkinson's "honeymoon" period** is the name of the period where medicament to relieve patient from symptoms of the PD remain efficient. The medicament's treatment aims to add a substance (Levodopa [88]) which mimics the role of the dopamine or compensate for dopamine deficiency with an exogenous input.

3. **The motor complications** start when medicament's treatments are no longer efficient. It is the ON/OFF phase when the symptoms appear and disappear.

4. **The advanced phase** manifests itself in falls, loss of balance, vegetative disorders and intellectual difficulties.

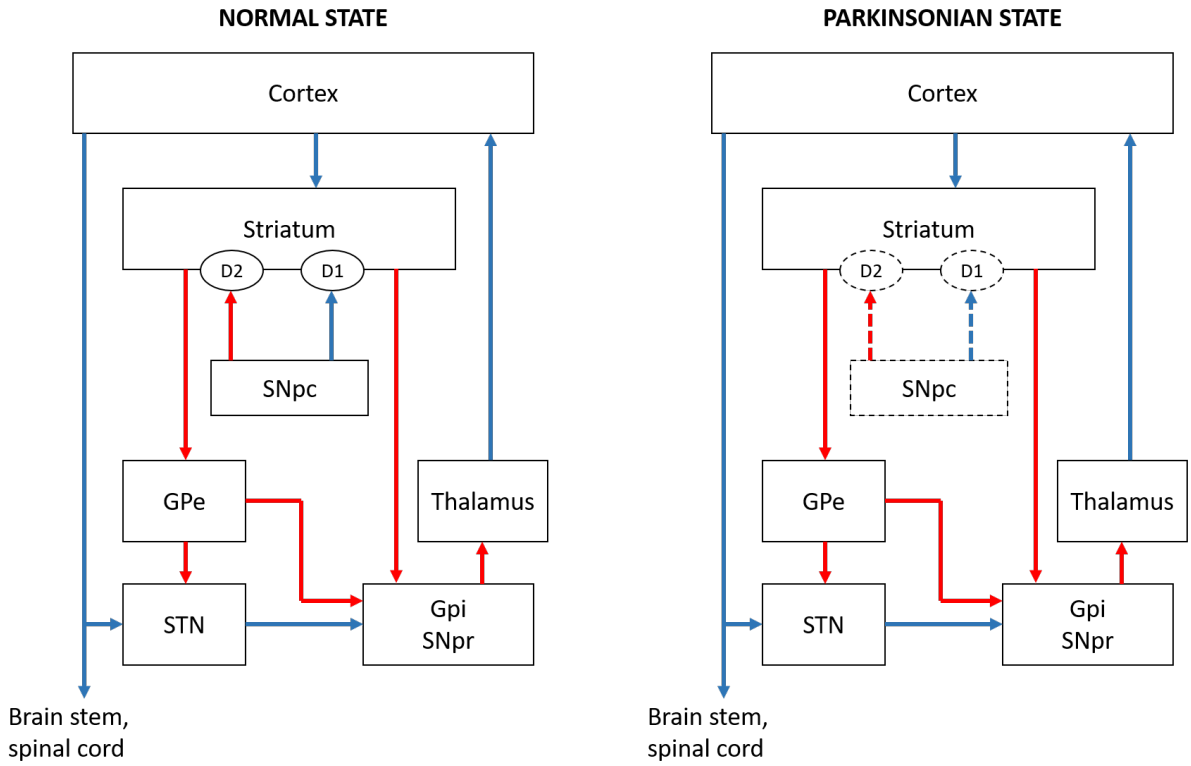
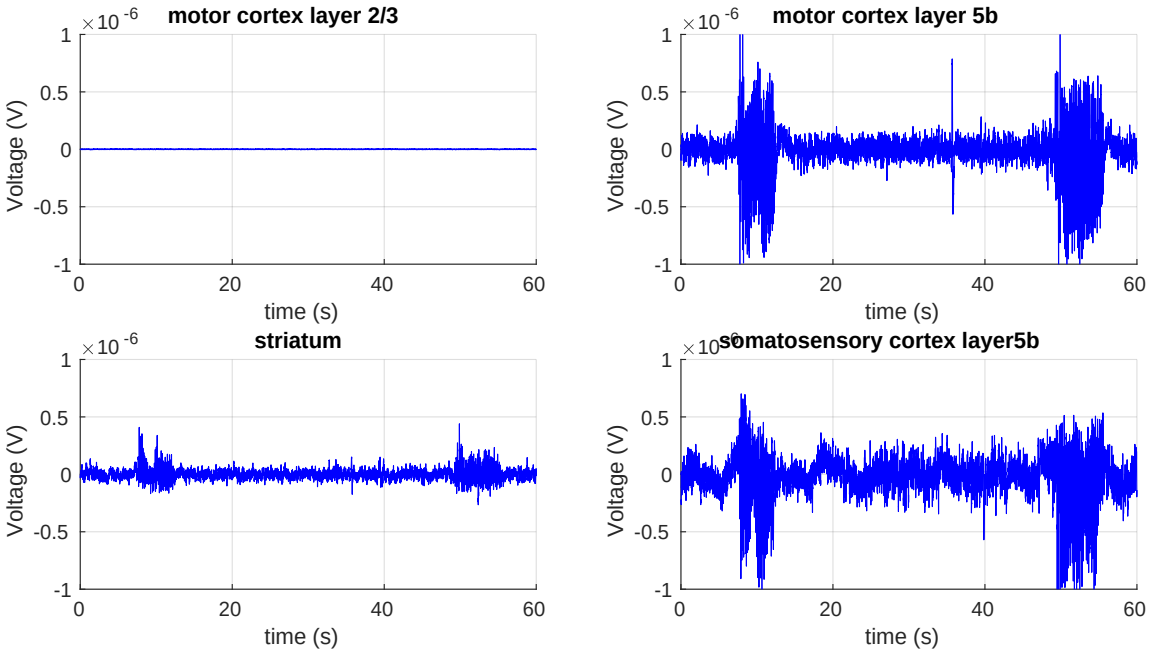


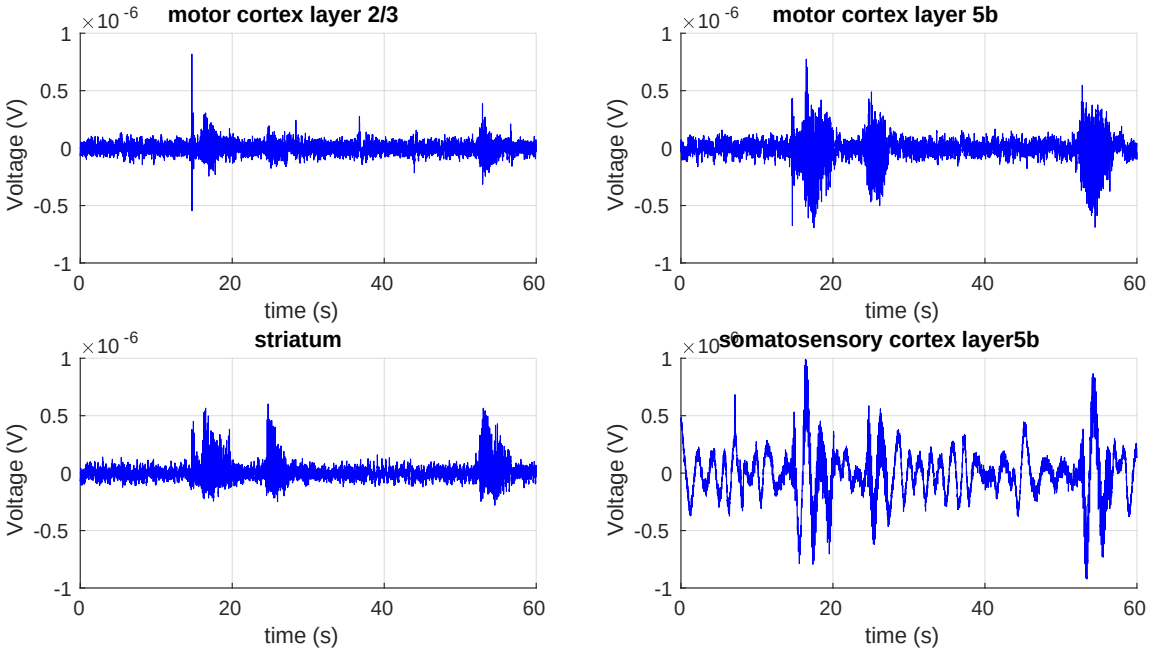
Figure 5.1: GABA circuitry in two cases: the normal state in the left and the Parkinsonian state on the right. Researches on brain functioning have identified different areas (or regions) in the brain. Each area has specific functions and the different brain regions interact with other areas. There are some type of interactions such as activation and inhibition links. An "activated" brain area will active and inhibit (or deactivate) brain regions with, respectively, activation and inhibition links. Red color is put for inhibition links and blue color for activation links. Notice that different pathways are proposed in the literature (with more or less details). Dashed lined in the Parkinsonian state correspond to the dysfunction parts in the pathway. Every parts of the GABA circuitry and its interactions are necessary to ensure the good functionning of the body movement. The degeneration of the SNpc leads to a global dysfunction of the GABA pathways. The research on the brain functioning remains active today and our understanding continues to evolve.

After the third phase, it has been shown that the Deep Brain Stimulation (DBS) is an efficient technique to relieve patients from the symptoms of the PD [3]. DBS necessitates the implantation by surgery of stimulation electrodes connected to a stimulator placed in the subclavicular cavity. This will allow continuous electrical stimulation in the *SubThalamic Nucleus* (STN). Since DBS is not a zero-risk procedure, this treatment is only considered if there are disabling symptoms despite optimized pharmacotherapy. Biological mechanisms are not widely understood today and a permanent application of the DBS leads to several side effects like psychiatric ones. The second drawback of the DBS concerns the battery life duration of the system. A surgery procedure is required to change battery of the patient every 5-6 years.

A low energy closed-loop system to control the DBS is therefore needed and the research on the subject is very active [90]. Dejean in [17] shows the arrival of the PD's symptoms can be predicted by detecting the presence of High-Voltage Spindles (HVS) in brain Local Field Potentials (LFPs). The HVS is a synchronous spike-and-wave patterns oscillating in the 5-13 Hz frequency band. Suppressing HVS signals has been found useful for delaying the progress of PD and deleting symptoms. The control of the DBS is a promising trail to reduce side effects induced by the DBS. But the HVS detection is a challenging problem for two reasons.



(a) PD rat recording session: example one.



(b) PD rat recording session: example two.

Figure 5.2: Two recording sessions from different PD rats. The high amplitude corresponds to HVS. We note in **(a)** that the channel M1U does not work. In **(b)**, the frequency information from the channel SD is out of range of the expected frequency for the HVS. This figure highlights difficulties we can meet to detect the HVS and the need to adapt the model for each patient.

Notation	Region name
M1D	Layer 5b of the primary motor cortex
M1U	Layer 2/3 of the primary motor cortex
M2D	Layer 5b of the secondary motor cortex
M2U	Layer 2/3 of the secondary motor cortex
SD	Layer 5b of the primary somatosensory cortex
SU	Layer 2/3 of the primary somatosensory cortex
STRI	Dorsal region of striatum
THAL	Ventrolateral thalamus

Table 5.1: List of brain regions where LFPs signals were recorded.

First, symptoms of the PD appear some milliseconds after the first HVS. Second, many factors like the progression of the disease, the location of probes inside the brain makes recorded signals very different from a patient to another. A fast and robust model capable to learn automatically from the data on real time is then required.

If detecting a signal on a frequency range seems apparently simple, there are different constraints making the problem more difficult. First, the recorded signals can be very different (HVS's frequencies and form) between two PD rats or two recording channels. The diffusion of signals inside the brain is not exactly the same for all people, some channels may provide no useful information or have unexpected frequencies. Second, the recorded signals can also change in time. As part of a progressive disease, HVS can appear more or less frequently according to the disease progress. Moreover, the electronic probes may move inside the brain or deteriorate over time. In the worst case a probe may disconnect. Finally, we know that symptoms appears only a few milliseconds after the HVS. A fast adaptive system capable to predict the venue of the HVS from multi-channel signals is expected.

Previous works on iEEG data have been proposed by Dr. Ramesh Perumal. The detection of HVS is performed on a single recording channel using a method based on the Cosine Wavelet Transform (CWT) [10]. In this chapter, new unsupervised algorithms are proposed to detect HVS on multiple recording channels. In parallel to the work presented in this chapter, an analysis presented in [100] proposes another detection method based on the synchrosqueezing transform for a time-frequency analysis in real time. This study has been proposed to detect HVS using M1D channel only.

5.2 HVS database and its construction

Fig. 5.2 gives two typical recording sessions of PD rats. Data extracted from PD rats are used to test different systems. The PD rat models were induced in 3-4 months old Sprague-Dawley rats by unilateral injection (coordinates: AP -4.4 mm, ML +1.2 mm, V -7.8 mm relatively to bregma) of 6-OHDA in the medial forebrain bundle at the rate of 0.5μl/min using an integrated electrophysiology instrument suitable for DBS procedure. Four weeks following the unilateral injection of 6-OHDA, the lesioned group of rats were determined as successful PD models through amphetamine-induced rotational behaviour (Amp, 3mg/kg, ip) by measuring the rotational speed of the lesioned rats as 6 turns per minute.The rats were unilaterally implanted bipolar stimulation electrode into the ipsilateral STN with their initial coordinates at AP -3.6 mm and L +2.5 mm. The electrode was lowered slowly along the dorsal ventral axis of brain and then advanced ventrally to the STN to obtain the electrophysiological signal with a strikingly silent structure. The LFPss were recorded from eight or four different brain regions depending on PD rat. The different brain regions are listed in Table 5.1. The sampling frequency was 1 kHz and the recording duration of one session was 60 seconds (60,000 samples). Several sessions have been recorded on PD rats.

The collected data were carried out by researchers from the NeuroEngineeringLab (NEL) in Taïwan. Results of experiments were saved in different formats (mat, csv) and organized differently according to

Algorithm 4 Ground truth: definition procedure

Require: $X \in \mathbb{R}^{(K \times R)}$: raw data with R channels and K time samples.
  for $r \leftarrow 0$ **to** R **do**
    $X[:, r] = X[:, r] / \text{std}(X[:, r])$ (Preprocessing)
    $X_{\text{cwt}} = |\text{cwt}(X[:, r])|$ (Cosine wavelet transform: $N_{\text{cwt}} \times K$)
    $E_{[5Hz;13Hz]} = \sum_{f \in [5Hz;13Hz]} X_{\text{cwt}}[:, :]$ (Sum of CWT between 5 Hz and 13 Hz)
    $\text{thesh} = \text{OTSU}(E_{[5Hz;13Hz]})$ (Threshold per channel)
    $Y[:, r] = X[:, r] > \text{thesh}$
  end for
  $Y = \frac{1}{R} \sum_{r=1}^{R} Y[:, r] > \frac{3}{4}$ (Decision for all channels)
  $Y = \text{medianFilt}(Y, 1sec)$ (Denoising)
  return X, Y

the researchers. The first step consists in standardizing all the received data. Each PD rat has a different number of recording sessions. The second step was to observe the sessions for each rat in order to select two sessions per rats: one for training, the other to validate the model. The session selection is an important step to ensure that the data contains HVSs because some recording sessions do not have any HVS signals or very few. Table 5.2 summarizes the organization of the database after the standardization and the selection of the training and validation sets. Columns "Training_Session" "and Validation_Session" are respectively, the chosen number of both training and validation sessions.

5.3 Ground truth

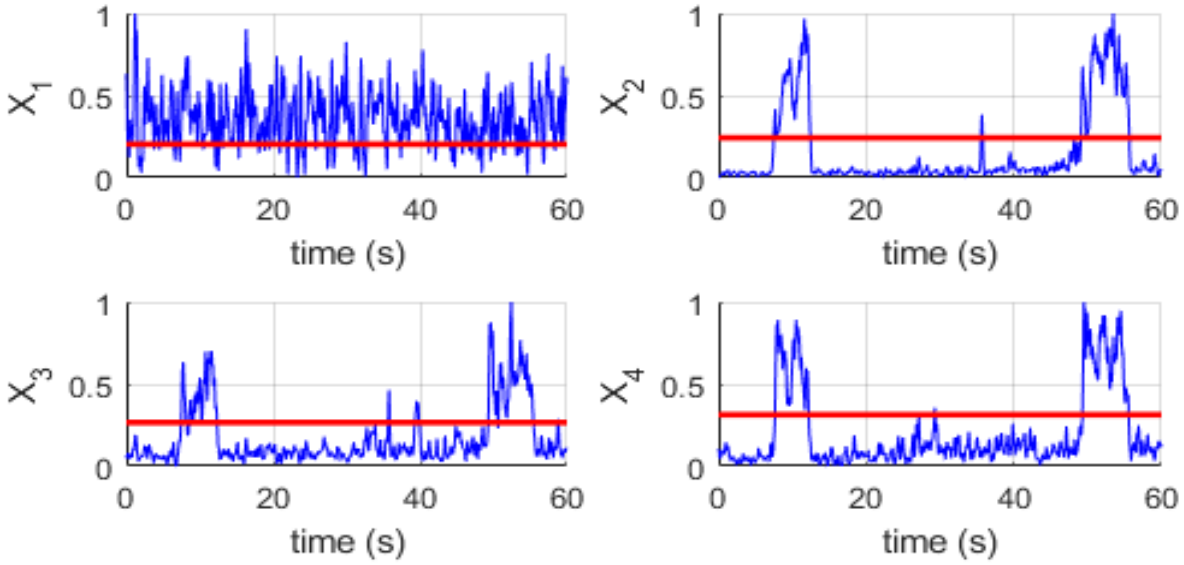
To evaluate the different explored models in this thesis, a reference output is defined to compare all results of each model with the reference. For each rat, we consider two 1 minute sessions. One session is used to train the model and to learn the ground truth. The second session is used to evaluate the model. Algo. 4 summarizes the different steps to learn the ground truth.

The preprocessing step is a standardization step. The variances of the LFPs signals are estimated and set to one by standardization, channel per channel. The frequencies characteristics of the HVS are identified by the CWT. The computation of the sum of the CWT coefficients between 5 Hz and 13 Hz gives the energy of each channel in time and in the frequency range of the HVS signal. Automatic thresholding from the shape of the histogram of the signal is applied on each channel to detect the HVS. Otsu's method provides a threshold $\theta(i)$ for $i-$th channel. The HVS is detected on the channel i at time t if $X_i(t) > \theta(i)$.

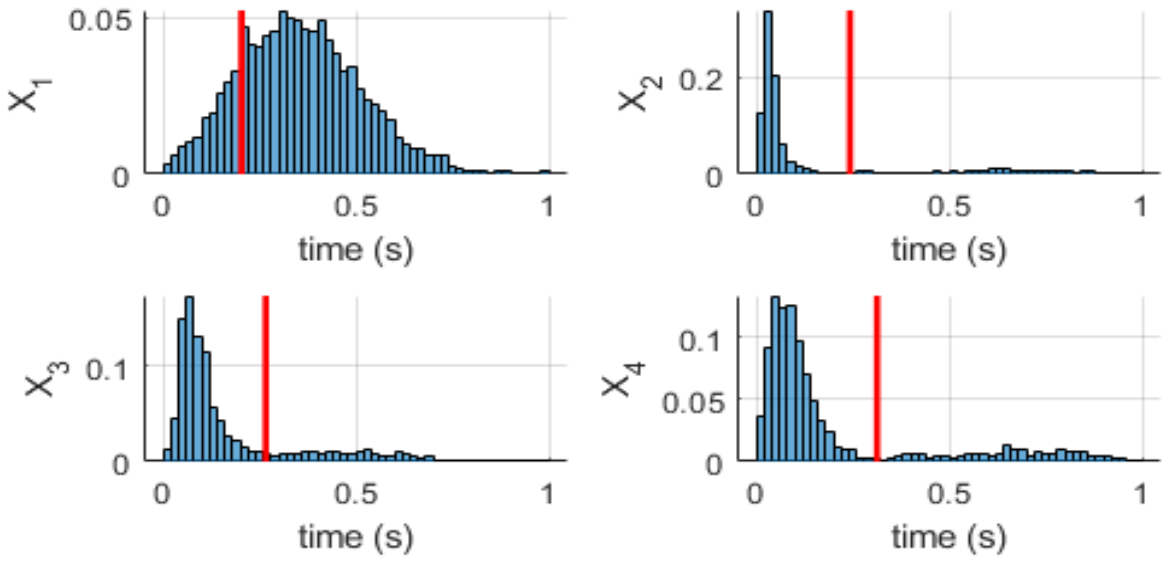
Fig. 5.3 gives the evolution of this energy through time and the histogram of it over the training set. As it is illustrated in Fig. 5.2 (see Sec. 5.1), some channels may not capture the signals for many possible reasons. The HVS is detected in the decision step if the HVS is detected in at least $\frac{3}{4}$ of all channels. The choice of $\frac{3}{4}$ is for taking into account the fact the HVS will may not appear on all channels. It also helps to be more robust against possible false detection caused by a single channel.

After the detection step, some false detections and/or misses may remain during the transition with and without HVS. The duration of an HVS is rarely less than 2 seconds. Then, false detections are deleted by applying a median filter of 1 second length. Sometimes, during the state "with HVS", the model based on Otsu' thresholding fails to detect the HVS. To remove this source of mistakes, a window of 200ms is automatically treated as a part of the HVS after the detection using Otsu' thresholding.

The ground truth makes it possible to define an output vector to compare models. However, ground truth extraction on low quality data may still be (slightly) noisy. For some PD rats in particular, the definition of the ground truth provides mitigate results.



(a) Sum of CWT's coefficients between 5-13 Hz through time.



(b) Histograms of the sum of CWT coefficients in 5-13 Hz band.

Figure 5.3: Result of the threshold per channel. In **(a)**, the presence of HVS is usually identified by the peak in the sum of CWT's coefficient between 5-13Hz only on the last three channels. The channel X_1 did not capture any information of the HVS. **(b)** gives histograms of channels, the distribution of CWT's coefficient between 5-13Hz over the session was modelized as a mixture of two Gaussian distributions. The first Gaussian corresponds to a heavy tailed distribution of low values. It corresponds to the state without HVS. The second Gaussian on the right with large value and large dispersion corresponding to the distribution during the HVS state. Otsu's method is used to separate the two cases. The vertical red line gives the threshold learned with Otsu's method for each channel ($\theta(i)$). The channel X_1 did not capture any information, it can be due to a dysfunction of the recording probes (see Fig. 5.2a as an example).

Record ID	Number of channels	Channel name	Session duration (ms)	Training session	Validation session	Note
20160106 N9_LES	4	motor cortex layer 2/3 motor cortex layer 5b striatum somatosensory cortex layer 5b	60000	7	28	
20160113 N9_LES	4	same as previous	60000	15	19	
20160113 N9_LES night	4	same as previous	60000	12	14	
20160407 N14_LES	4	same as previous	60000	27	28	
20160420 N14_LES	4	same as previous	60000	1	29	
20160421 N12_LES	4	same as previous	60000	24	25	
20160504 N14_LES	4	same as previous	60000	10	25	
20160505 N12_LES	4	same as previous	60000	22	23	
P-R-4 20131205	8	Layer 5b of the primary motor cortex Layer 2/3 of the primary motor cortex Layer 5b of the secondary motor cortex Layer 2/3 of the secondary motor cortex Layer 5b of the primary somatosensory cortex Layer 2/3 of the primary somatosensory cortex Dorsal region of striatum ventrolateral thalamus	60000	1	2	
20140120 P-R-6 BASELINE	4	Layer 5b of the primary motor cortex Layer 2/3 of the primary motor cortex Layer 5b of the primary somatosensory cortex Layer 2/3 of the primary somatosensory cortex	60000	12	13	
P-R-4 20131205	8	Layer 5b of the primary motor cortex Layer 2/3 of the primary motor cortex Layer 5b of the secondary motor cortex Layer 2/3 of the secondary motor cortex Layer 5b of the primary somatosensory cortex Layer 2/3 of the primary somatosensory cortex Dorsal region of striatum ventrolateral thalamus	60000	1	2	
20160113 N9_LES night	4 ($\rightarrow$3)	Layer 5b of the primary motor cortex Layer 2/3 of the primary motor cortex Dorsal region of striatum ventrolateral thalamus	60000	4	6	Channels 1 and 2 are identical.
?	4	Intact site M1 deep layer (5b) Intact site M1 up layer Lesion site M1 deep layer (5b) Lesion site M1 up layer	59999	33	34	
?	4	same as previous	59999	19	20	
?	4	same as previous	59999	26	30	

Table 5.2: Data organization.

Let H_0 be the hypothesis "no HVS" and H_1 the hypothesis "presence of HVS" and h the decision attached to an observation window. The following evaluation tools are considered:

- The **sensitivity**: $Se = \Pr(h \in \widehat{H_1}|H_1)$.
- The **specificity**: $Sp = \Pr(h \in \widehat{H_0}|H_0)$.
- The **mean delay of detection**.

The mean delay of detection is estimated after the application of a median filter on the model output to eliminate the noisy detection (detection of less than 1 second).

Remark. Notice that the aim of the application is not to have a model with 100% sensitivity and 100% specificity, but to be able to detect the HVS the sooner as possible and if possible before the ground truth. The sensitivity and the specificity are useful to make sure the studied model detects efficiently the HVS. But detecting the HVS before the ground truth will decrease the sensitivity. Note also that the ground truth defined in previous sections is not perfect. The evaluation has to be viewed with hindsight and requires a balance between the sensitivity, the specificity and the mean delay of detection.

5.4 Application with machine learning methods

Three models has been studied in this thesis and compared with the groundtruth. The first one is the combination of the DN-RBM with the Bayes classifier presented in Sec. 3.3. The second one is a supervised model based on the use of an autoencoder and a Gaussian process classifier [99]. These methods were presented in [101] as the combination of the DN-RBM with the Otsu threshold method. The next section presents the main principles of the autoencoder and the Bayes classifier.

The autoencoder and the DN-RBM are two neural networks with the same objective: to reduce the size of the data before the classification step. To detect the presence of HVS, the short-time window signal has to be large enough to capture the desired frequencies. The sampling frequency is equal to 1 kHz and the fundamental frequency of a HVS is 5Hz or higher. A window of length 200ms ensures to have at least one period of HVS and is used as input of models.

5.4.1 The Autoencoder and the Gaussian Process classifier

Autoencoder

The autoencoder is a feedforward neural network which computes a compressed representation of the data. The model is composed of an encoder part (or transfer matrix) and a decoder part (see Fig. 5.4). The encoder computes the latent representation of the input data and the decoder reconstructs the input data according to the latent representation, *i.e.* $x = f(x)$ where f is the neural network model. Once the model is trained, only the encoder module is retained to compress the data.

Gaussian Process classifier

A Bayesian Network (BN) is a directed acyclic graph. Each nodes is a Random Variable (RV). A link from node a to node b means the state of b depends on the state of a, a is then a parent of b. In a BN, it is impossible to have two parent nodes of each other. A Gaussian Process (GP) is a Bayesian Network with continuous valued nodes. Let $X \in \mathbb{R}^d$ be the parent of $Y \in [\![0, 1]\!]$. Let note $\mathcal{D} = \{\boldsymbol{X}^{(n)}, \boldsymbol{Y}^{(n)}\}_{n\in[1,N]}$ the training set with $X = \{\boldsymbol{X}^{(n)}\}_{n\in[1,N]}$ being independent randomly selected input observations and $Y = \{\boldsymbol{Y}^{(n)}\}_{n\in[1,N]}$ the associated output decision respectively. The GP classifier focus on modeling the *posterior* probabilities by defining the latent variables $\boldsymbol{f}_n = f(\boldsymbol{X}^{(n)})$.

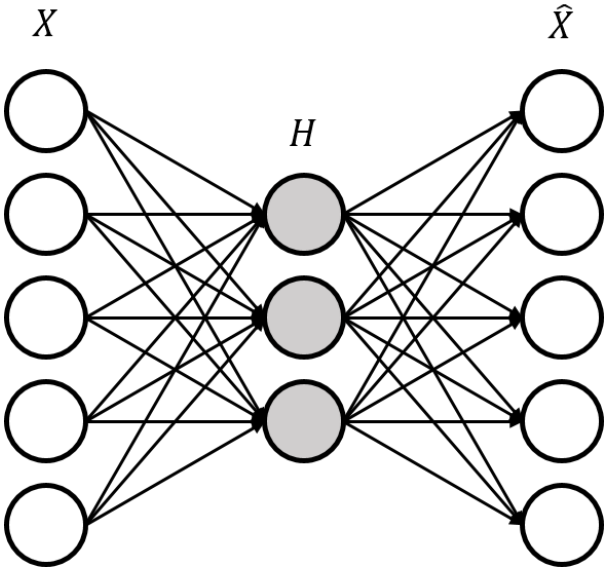


Figure 5.4: Autoencoder. The passage from the input layer to the hidden layer is the encoder part and the second passage from the hidden layer to the output layer is the decoder part. Back-propagation algorithm is used to learn this model. Autoencoders with multiple hidden layers are also possible.

The model used here is the *probit* model: $\Pr(Y = 1|X) = \Phi(f(X))$ where Φ denotes the cumulative density function of the standard normal distribution, *i.e.* $\phi(x) \to 0$ for $x \to -\infty$ and $\phi(x) \to 1$ for $x \to +\infty$. The likelihood of the probit model with independent observations, given by $f = \{f(\boldsymbol{X}^{(n)})\}_{n\in[1,N]}$ is:

$$p(Y|f) = \prod_{n=1}^{N} p(\boldsymbol{Y}^{(n)}|\boldsymbol{f}_n) = \prod_{n=1}^{N} \Phi(\boldsymbol{Y}^{(n)}\boldsymbol{f}_n). \tag{5.1}$$

In a GP, f is a stochastic process which associates a zero mean normal random value to an input $\boldsymbol{X}^{(n)}$. For X from the training set $\mathcal{D}$, $p(f|X,\Theta) \sim \mathcal{N}(0, \mathbf{C}_N)$ where Θ is a set of hyper-parameters and C_N is a covariance matrix modelized by a *squared exponential* and a Gaussian noise [89], *i.e.* :

$$C_N(\boldsymbol{X}^{(i)}, \boldsymbol{X}^{(j)}) = \theta_0^2 \exp\left(-\frac{1}{2}\sum_{n=1}^{\dim(\boldsymbol{X}^{(i)})} \frac{(X_n^{(i)} - X_n^{(j)})^2}{\lambda_n^2}\right) + \theta_1^2 \delta_{(\boldsymbol{X}^{(i)},\boldsymbol{X}^{(j)})}. \tag{5.2}$$

$X_n^{(i)}$ is the $n-$th component of $\boldsymbol{X}^{(i)}$ and $\delta_{(\cdot)}$ is the Kronecker delta. The set of hyper-parameters Θ is composed of $\left\{\theta_1, \theta_2, \{\lambda_n\}_{n\in[1,N]}\right\}$. Baye's posterior probability rule of the latent variable f with Θ known ($p(f|\mathcal{D},\Theta)$) can be written:

$$p(f|\mathcal{D},\Theta) = \frac{p(Y|f)p(f|X,\Theta)}{p(\mathcal{D}|\Theta)} = \frac{\mathcal{N}(f|0,C_N)}{p(\mathcal{D}|\Theta)} \prod_{n=1}^{N} \Phi(\boldsymbol{Y}^{(n)}\boldsymbol{f}_n). \tag{5.3}$$

The marginalization of (Eq. 5.3) for a new observation $\boldsymbol{X}^{(N+1)}$ gives:

$$\Pr(\boldsymbol{f}_{N+1}|\mathcal{D},\Theta,\boldsymbol{X}^{(N+1)}) = \int \Pr(\boldsymbol{f}_{N+1}|f,X,\Theta,\boldsymbol{X}^{(N+1)}) \Pr(f|\mathcal{D},\Theta)\mathrm{d}f, \tag{5.4}$$

and the expectation of the (Eq. 5.4) gives:

$$\Pr(Y^{(N+1)}|\mathcal{D},\Theta,\boldsymbol{X}^{(N+1)}) = \int \Pr(Y^{(N+1)}|\boldsymbol{f}_{N+1}) \Pr(\boldsymbol{f}_{N+1}|\mathcal{D},\Theta,\boldsymbol{X}^{(N+1)})\mathrm{d}\boldsymbol{f}_{N+1}. \tag{5.5}$$

The posterior probability $q(f|\mathcal{D},\Theta)$ is modelized as $\mathcal{N}(\boldsymbol{m},A)$ to compute $\Pr(Y^{(N+1)}=1|\mathcal{D},\Theta,\boldsymbol{X}^{(N+1)})$. Then, for a new observation $N+1$, we can show that the posterior probability of $\boldsymbol{f}_{N+1}$ is $q(\boldsymbol{f}_{N+1}|\mathcal{D},\Theta,\boldsymbol{X}^{(N+1)}) \sim \mathcal{N}(\mu,\sigma)$ with:

$$\begin{cases} \mu &= \boldsymbol{k}^T C_N^{-1}\boldsymbol{m}, \\ \sigma^2 &= \kappa - \boldsymbol{k}^T(C_N^{-1} - C_N^{-1}AC_N^{-1})\boldsymbol{k}. \end{cases} \tag{5.6}$$

where $\boldsymbol{k} = \left(C_N(\boldsymbol{X}^{(1)},\boldsymbol{X}^{(N+1)}),\ldots,C_N(\boldsymbol{X}^{(n)},\boldsymbol{X}^{(N+1)})\right)^T$ is the covariance vector between each observation of the training set and the new observation $\boldsymbol{X}^{(N+1)}$ and $\kappa = C_N(\boldsymbol{X}^{(N+1)},\boldsymbol{X}^{(N+1)}) = \theta_0^2+\theta_1^2$ is the variance of $\boldsymbol{X}^{(N+1)}$.
With the approximation of $\Pr(f|\mathcal{D},\Theta)$, (Eq. 5.5) becomes:

$$\Pr(Y^{(N+1)}=1|\mathcal{D},\Theta,\boldsymbol{X}^{(N+1)}) = \Phi\left(\frac{\mu}{\sqrt{1+\sigma^2}}\right) \tag{5.7}$$

Training a GP consists in finding $\{\Theta,\boldsymbol{m},A\}$. We can learn Θ by computing the log-likelihood of the posterior probability $\log q(Y|X,\Theta)$ (Eq. 5.8) (see [89, Chapter 5]) and its gradient as a function of Θ:

$$\log q(Y|X,\Theta) = -\frac{1}{2}f^T C_N^{-1} f + \log p(Y|f) - \frac{1}{2}\log\det\left(I + W^{\frac{1}{2}}C_N W^{\frac{1}{2}}\right) \tag{5.8}$$

with $W = -\Delta_f \log p(Y|f)$ (The Hessian of the log likelihood given f), the unormalized log likelihood $\log p(f|\mathcal{D},\Theta)$ is maximized:

$$\begin{aligned} \log p(f|\mathcal{D},\Theta) &= \log p(Y|f) + \log p(f|X) \\ &= \log p(Y|f) - \frac{1}{2}f^T C_N^{-1} f - \frac{1}{2}\log\det(C_N) - \frac{N}{2}\log 2\pi. \end{aligned} \tag{5.9}$$

For a given Θ, we can find $\boldsymbol{m} = \arg\max_f \log p(f|\mathcal{D},\Theta)$ by using the Newton's method. (Eq. 5.10) and (Eq. 5.11) give the gradient and the Hessian of $\log p(f|\mathcal{D},\Theta)$, respectively.

$$\nabla_f \log p(f|\mathcal{D},\Theta) = \nabla_f \log p(Y|f) - C_N^{-1} f. \tag{5.10}$$

$$\Delta_f \log p(f|\mathcal{D},\Theta) = \Delta_f \log p(Y|f) - C_N^{-1}. \tag{5.11}$$

The maximization of $\log p(f|\mathcal{D},\Theta)$ makes use of the first and second order partial derivation of $\log p(Y|f)$ in function of f_i:

$$\frac{\partial}{\partial f_i}\log p(Y|f) = \frac{Y_i\phi(f_i)}{\Phi(Y_i f_i)}. \tag{5.12}$$

$$\frac{\partial^2}{\partial f_i^2}\log p(Y|f) = -\frac{\phi(f_i)^2}{\Phi(Y_i f_i)^2} - \frac{Y_i f_i \phi(f_i)}{\Phi(Y_i f_i)}. \tag{5.13}$$

Where $\phi(.)$ is the density function of the standard normal distribution. Learning $\boldsymbol{m}$ allows to compute (Eq. 5.8) and their gradient in function of Θ. We implement a gradient descent search of the optimum Θ^* that leads to the following iterative algorithm:

$$\Theta^{(k+1)} = \Theta^{(k)} - \alpha_k \nabla_\Theta \log q(Y|X,\Theta)). \tag{5.14}$$

But (Eq. 5.14) requires to inverse the $N \times N$ matrix C_N at each iteration which can be time consuming for a large number of observations. Two solutions are possible to decrease the learning time. First, to decrease the size of the training database N and to select randomly the training samples. Second, compute an approximation of the inverse matrix C_n^{-1}

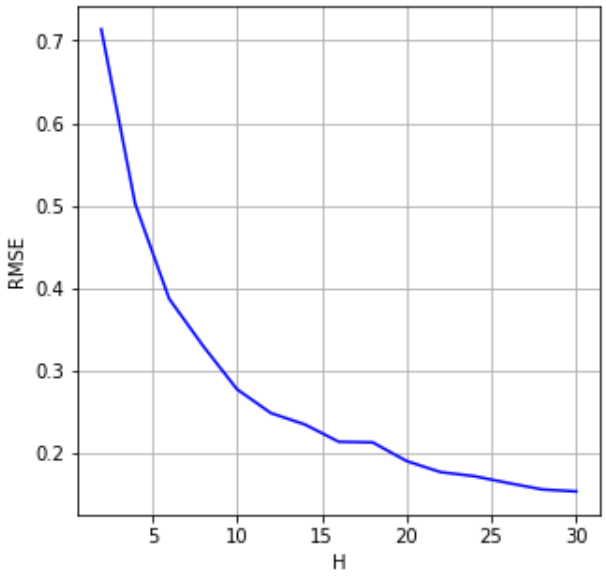


Figure 5.5: Root mean square error of reconstruction over the testing set of iEEG data as a function of H (size of the hidden layer). DN-RBMs has hidden units (from 2 to 30).

Once $\boldsymbol{m}$ and Θ found, we can compute $A = \left(C_N^{-1} + W\right)^{-1}$.
Finally, the GP classifier is learned by identifying the covariance matrix between observations C_N as a function of the hyper-parameters Θ, the mean vector $\boldsymbol{m}$ is learned at each iteration and the covariance matrix A is deduced from C_N and $\boldsymbol{m}$.

Once the learning is done, the prediction step computes the covariance vector $\boldsymbol{k}$ between the new observation $\boldsymbol{X}^{(N+1)}$ and the training set X and then estimates the probability $\Pr(Y^{(N+1)} = 1|\mathcal{D}, \Theta, \boldsymbol{X}^{(N+1)})$. If $\Pr(Y^{(N+1)} = 1|\mathcal{D}, \Theta, \boldsymbol{X}^{(N+1)}) > 0.5$ then $Y^{(N+1)} = \pm 1$.

Learning the model consists in two steps: learning the hyper-parameters Θ and learning the parameters of $q(\boldsymbol{f}_{N+1}|\mathcal{D}, \Theta, \boldsymbol{X}^{(N+1)})$. The prediction of $Y^{(N+1)}$ is based on the posterior probability: the $N+1$ input is compared to the N first inputs and the model compute correlation between the new observation and the previous ones to compute the output.

5.4.2 Sizing models

The three following classifier are trained with different sizes of hidden layer:

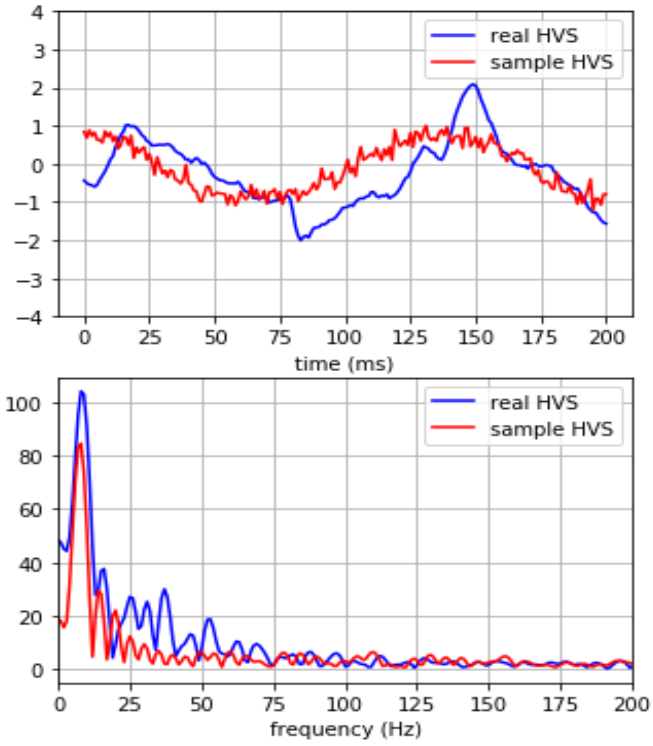
(a) DN-RBM+Bayes (hidden state),

(b) autoencoder+GP classifier and

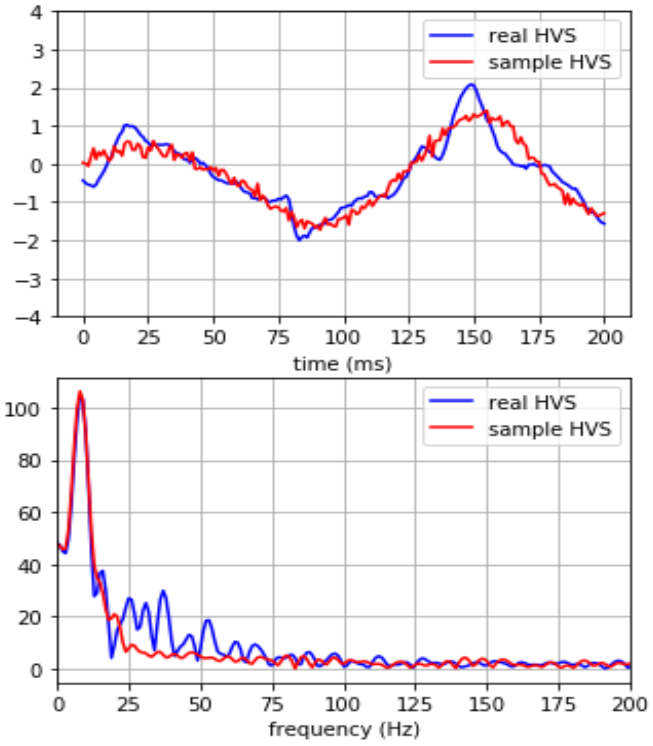
(c) DN-RBM+Bayes (energy level).

Previous experiences to size both networks have been proposed in [99, 101]. New experiences have been proposed for this thesis in the following sections. The dimension of the DN-RBM is chosen in function of the results of the classifier. However, the study proposed in Sec. 3.2 has also been performed for iEEG data to size a DN-RBM independently. Results are given in Fig. 5.5 from the RMSE of the reconstruction in function of visible units and Fig. 5.6 for four cased of size of hidden layer.

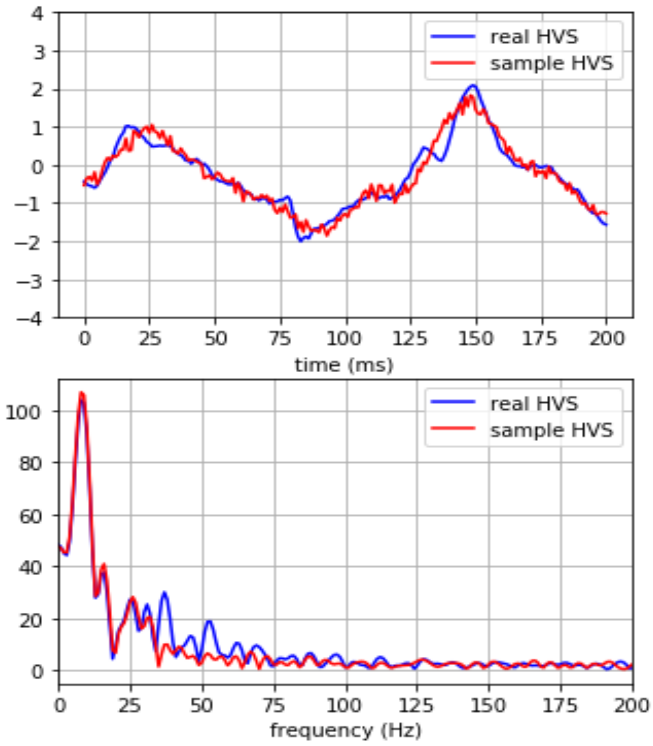
Let R be the number of recording channels for a given PD rat. The period of the HVS is close to 200ms for all the tested PD rats. To be able to detect the HVS, each model has been trained using a time-window of length 200ms as input. In the case of DN-RBM approaches, the time-window is downsampled to reduce the size of the network. The sampling period is fixed to 20ms which is sufficient to detect frequencies up to



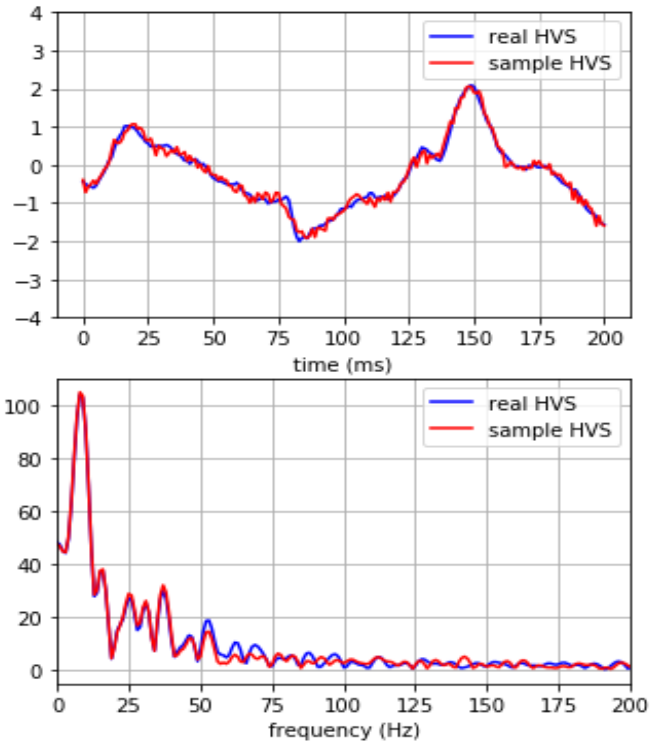
(a) $\dim H = 2$.



(b) $\dim H = 8$.



(c) $\dim H = 12$.



(d) $\dim H = 30$.

Figure 5.6: Reconstruction of the visible layer. Model has been trained using channel M1D of iEEG data for different dimensions of the hidden layer. The selected window used for the reconstruction contains an HVS. On the 4 figures, the upper plot gives the temporal signals and the lower plots are the Fourier transforms.

25Hz and then (in most cases) the three first harmonics of the HVS signal. The size of the visible layer of a DN-RBM is then equal to $10 \times R$, with R the number of channels.

DN-RBM + Bayes classifier (hidden state)

Different models have been tested for different value of n, the size of the hidden layer. The noise power σ is fixed to 10^{-2}. First, the average reconstruction error is evaluated to estimate how well the trained DN-RBM is able to reconstruct data:

$$\text{MSE} = \|\langle \boldsymbol{v} \rangle_0 - \langle \boldsymbol{v} \rangle_1\|^2 \tag{5.15}$$

Since the computation of the expected visible layer for all sample of the database is time consuming, the average over the set of tests is computed:

$$\text{MSE} \approx \frac{1}{N} \sum_{n=1}^{N} \left\| \boldsymbol{v}^{(n)} - \widehat{\boldsymbol{v}^{(n)}} \right\|^2 \tag{5.16}$$

where $\widehat{\boldsymbol{v}^{(n)}}$ is the reconstructed visible layer of the observation $\boldsymbol{v}^{(n)}$ after one reconstruction step. The results of the second classifier are then evaluated using the sensitivity Se, the specificity Sp and the mean detection time. An overview of the mean results and their uncertainty is given in Fig. 5.7a with box-plots. According to the multiple tests, the case $n = 4$ is promising. Table 5.3 gives the results for each PD rats with $n = 4$.

Autoencoder + Gaussian Process classifier

For the model autoencoder + GP classifier, only the influence of hidden layer size n is studied. The size of the training set remains an important choice because the correlation matrix $\boldsymbol{C}_N$ dimension depends on the number of observation used for the training procedure.

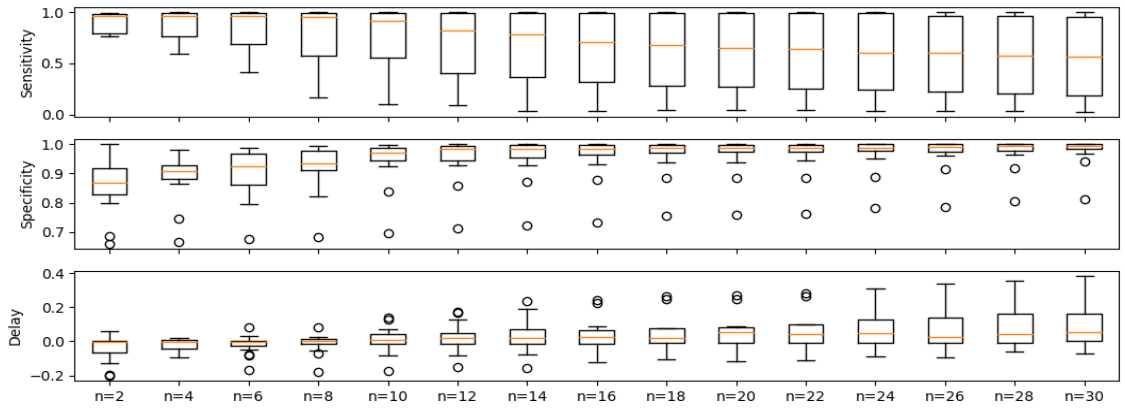
Previous study in [99] shows the more observation are in the training set, the more the model will be efficient. The number of observations is fixed to 200 and the sensitivity, the specificity and the mean delay of detection are given for different sizes n in Fig. 5.7c. Table 5.3 gives detailed results in the case where $n = 10$.

DN-RBM + Bayes classifier (energy level)

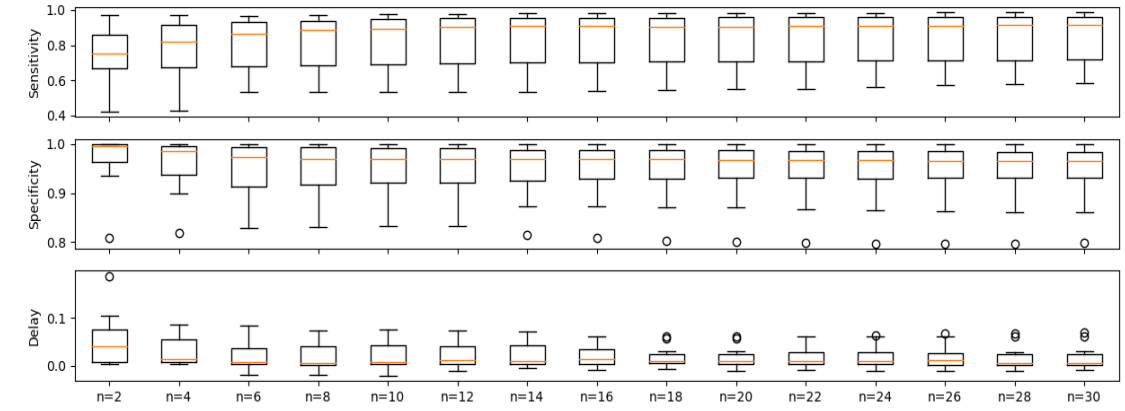
Results of the model DN-RBM + Bayes classifier based on the energy level is based on the computation and classification of the energy term $-h^T W v$ (see Sec. 3.3.1). To avoid false detection due to non stationary offset, an additional step of standardisation of each visible layer is applied independently aforehead the use of DN-RBM (see (Eq. 3.16)). Previous work proposed in [101] uses Otsu's threshold to classify both state. In this thesis, the Bayes classifier has been tested. The noise power is fixed to 10^{-2}. The results are given in Fig. 5.7b in function of the size of the hidden layer. Table 5.3 gives detailed results in the case where $n = 6$.

5.4.3 Discussion

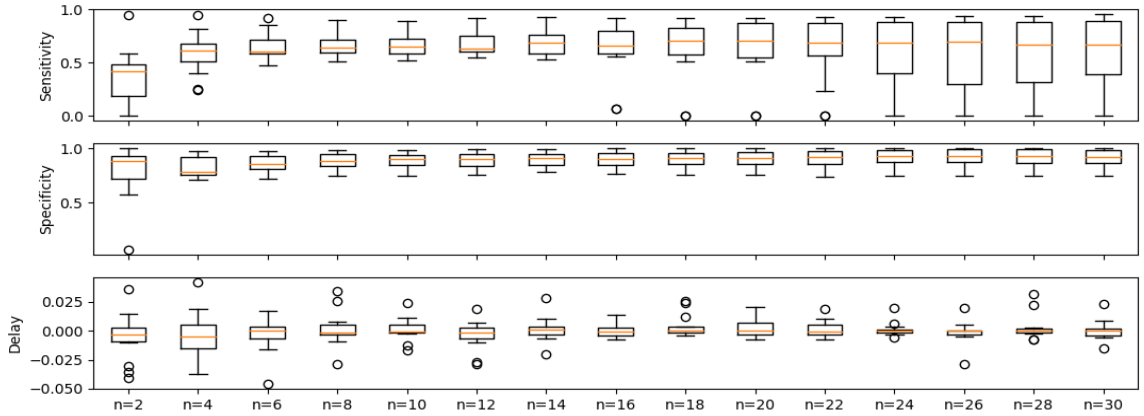
Each of the three proposed models detects successfully HVS signals. Results are better on the sensitivity for the unsupervised models. For other evaluation criteria, the results of the GP classifier is equivalent to other models. In the case of DN-RBM, there is a trade-off between sensitivity and specificity. In the case of the classification based on the hidden states, the more the number of hidden neurons, the more the sensitivity of the model decreases. The trend is reversed in the case of the energy-based classification but is much less pronounced. Fig. 5.8 shows the results obtained for the classifier with hidden neurons with different sizes



(a) DN-RBM + Bayes (hidden unit).



(b) DN-RBM + Bayes (energy function).



(c) Autoencoder + GP classifier.

Figure 5.7: Multiple tests for the detection of HVS. DN-RBMs have been trained for each PD rats and for different size of hidden layer (n) ten times. After computing the mean over the ten models, a box plot is proposed for each for each hidden layer.

Data			Sensitivity			Specificity			Delay (ms)		
Record	C	N	m1	m2	m3	m1	m2	m3	m1	m2	m3
01	8	12	**0.99**	0.77	0.92	0.90	0.89	**0.99**	**-47**	+18	+63
02	4	26	0.60	0.61	**0.63**	0.87	0.82	**0.91**	+23	+8	**+7**
03	8	9	**0.90**	0.55	0.72	0.93	0.94	**0.99**	**+16**	+18	+83
04	3	15	0.57	**0.58**	0.53	**0.87**	0.82	0.84	+4	-11	**-20**
05	4	10	**0.99**	0.77	0.87	0.67	0.77	**0.93**	**-57**	+0	+4
06	4	5	**1.00**	0.62	0.96	0.98	0.95	**0.99**	**-52**	-2	+7
07	4	4	**0.98**	0.72	0.82	0.93	0.93	**1.00**	**-108**	-2	+18
08	4	22	**0.78**	0.66	0.63	0.88	0.79	**0.91**	**+5**	+10	+49
09	4	10	**0.97**	0.86	0.82	0.92	0.89	**0.97**	**-25**	-1	+6
10	4	16	**0.74**	0.59	0.53	0.90	0.83	**0.91**	+12	**+0**	+25
11	4	4	**0.98**	0.66	0.94	0.90	0.97	**0.98**	**+8**	+10	+65
12	4	2	0.76	0.90	0.90	0.98	0.85	**1.00**	**+1**	+4	+5
13	4	6	**0.97**	0.65	0.89	0.90	0.95	**0.97**	**-2**	+11	+4
14	4	9	**1.00**	0.58	0.97	0.75	**0.92**	0.83	**-23**	+3	+2
15	4	1	**0.99**	0.64	0.96	0.97	0.98	**1.00**	**-64**	+1	+10
mean			**0.88**	0.68	0.81	0.89	0.89	**0.95**	**-21**	+4	+22

Table 5.3: Results with DN-RBM + Bayes classifier (hidden units) **(m1)**, the autoencoder + GP classifier **(m2)** and DN-RBM + Bayes classifier (energy level) **(m3)**. C is the number of channels and N the number of HVS detected (by the ground truth). For each rat, ten models are trained. Then, the mean over models of the sensitivity, the specificity and the mean delay of detection compared with the ground truth method are given in the table. A negative value on the delay means the model detect the HVS before the ground truth.

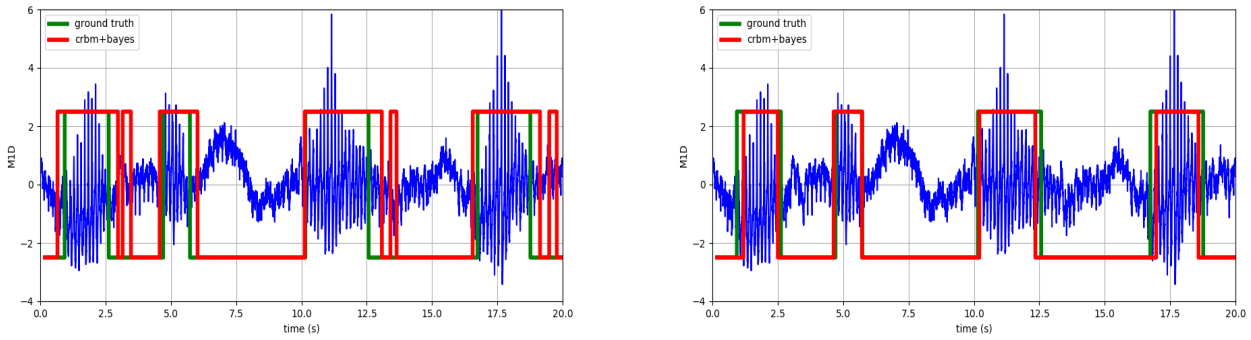
of the hidden layer. A small number of latent units causes some false detections but allows detection of the HVS signal. According to Table 5.3 the classifier with the hidden state allows to obtain a better sensitivity and the classifier using energy level provides a better specificity. The classification based on the hidden units detects the HVS signals earlier. In addition, it is necessary to recall that the energy-based classification required an additional standardization step on the visible layer to not detect undesirable components. In practice, adding an additional step will slow down the detection of HVS.

5.5 Evaluation of the Diffusion Network on iEEG data

In this section, the use of DN is proposed for the two applications presented in Chapter 4. Previous studies provide solution to detect the HVS based on the analysis of time-window. The main issue of interest in this study is to see what the DN could bring to improve previous approaches. First, the missing channels reconstruction application to generate signals in event of malfunction of some electronic probes for example. Second, the prediction of time-window to evaluate the capacity of the model to predict. As for the toy data, ten models (DNs, RNNs and MLPs) has been trained. The RMSE is given for each model (see (Eq. 4.14) and (Eq. 4.15)). Tests has been realized on the PD rat P-R-4 20131205 (see Table 5.2).

5.5.1 Application 1: Missing channels reconstruction

Results are separated in two parts. First, each recording channel has been reconstructed in function of the other ones. Table 5.4 summarizes the results for the eight channels and Fig. 5.9 gives four examples of reconstruction of signal. The blue signal is the expected signal and the red signal is sampled using the DN. The DN succeeded to reconstruct each channel efficiently. The DN with the learning of the activation



(a) Detection of HVS with 4 hidden units.

(b) Detection of HVS with 30 hidden units.

Figure 5.8: Detection of HVS using the DN-RBM + Bayes (hidden state). The two figures give the results on the validation set for two sizes of hidden layer. This figure highlights the compromise between the sensitivity and the specificity by modifying the size of the hidden layer. Adding hidden units will reduce the number of false detection but will slow down the detection of HVS.

function (DN 2 in tables) provides best results for all channels. Note that results can change bit by repeating the experiment (due to the noise of the DN or the random selection of mini-batch during the learning stage). The performance depends on the removed channels too. Table 5.4 gives results for each architecture. For channel M1D, results are very close between the DN and the MLP, by repeating the experiment, the MLP can be better for this channel only.

The second part of results concerns the reconstruction of the first channel (M1D) if more than one channels is removed. Table 5.5 gives quantitative results for one to five missing channels (on eight in total) and Fig. 5.10 is the result for the DN only. For each case, both DNs succeed to reconstruct the channel M1D. The efficiently decreases each time a new channel is removed. The DN with the learned of the activation function provides best results.

5.5.2 Application 2: Prediction of horizon using a vector DN

For this application, vector DNs using a time-window of 200ms has been tested on iEEG data and compare with MLP and RNN. For each PD rat, each model has been trained using the same database. Results with the DN are given in Fig. 5.10. Four examples have been proposed on four different PD rats on the sequence with HVS signal. DNs learn frequency components with varying degrees of success in function of PD rats.

The RMSE of the prediction is computed for the PD rat P-R-4 20131205 for each model:

- DN (without learning the activation function): RMSE=7.832E-01,
- DN (with learning the activation function): RMSE=7.325E-01,
- MLP: RMSE=8.134E-01,
- RNN: RMSE=8.074E-01.

According to Fig. 5.10, DNs successfully predict a time-window from a sequence of observation with the same length. Results depends on the PD rat and channel. On Fig. 5.11a for example, the prediction is correct visually for channels M1U, M1D and STRI but there is a phase shift between the expected signal and the predicted signal for the channel SD. The comparison with other models show the DN provides better results. The main drawback of the DN in comparison with the other model is the number of parameters, to predict a time-window of length N since a sequence of the same length is required. To reduce the size of the model, other graph architecture could be considered.

Table 5.4: Missing Channel Reconstruction.

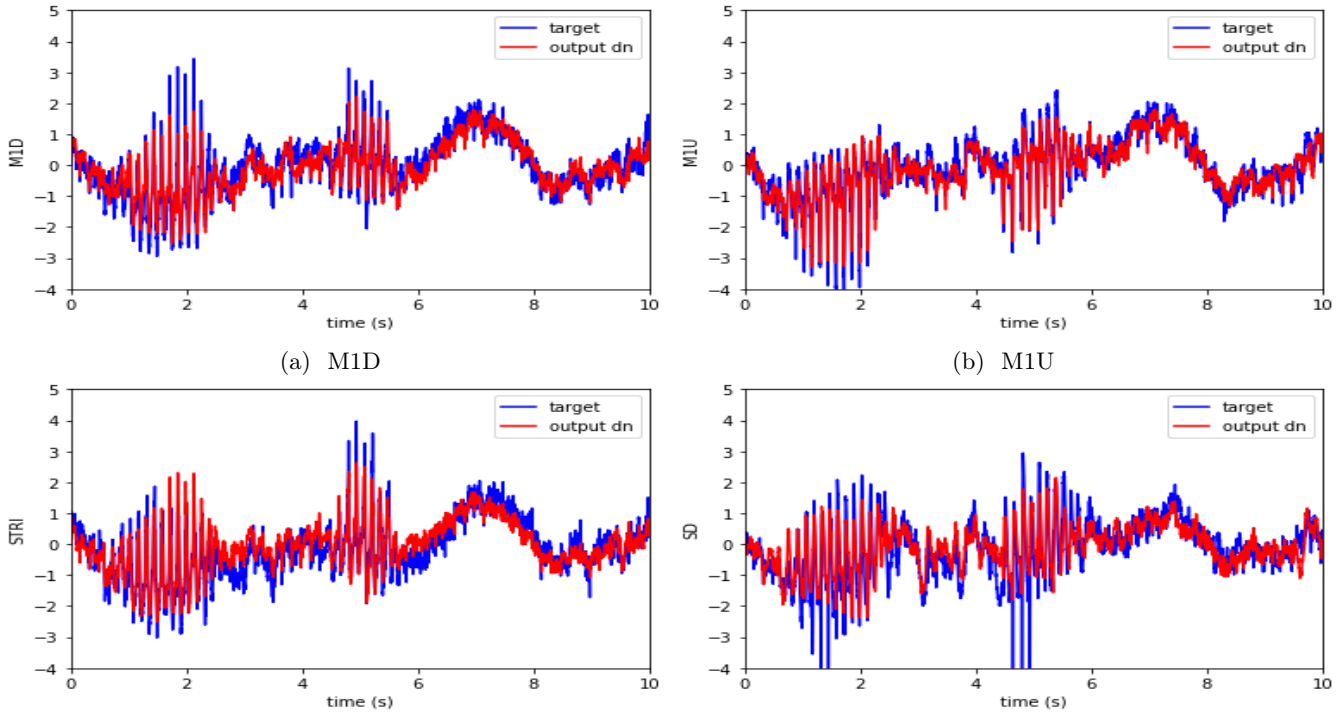
Missing channel[a]	RMSE (DN 1)	RMSE (DN 2)	RMSE (MLP)	RMSE (RNN)
M1D	5.688E-01	**4.980E-01**	4.985E-01	6.085E-01
M1U	4.773E-01	**3.953E-01**	4.173E-01	4.489E-01
M2D	8.187E-01	**7.480E-01**	9.124E-01	9.241E-01
M2U	5.795E-01	**4.692E-01**	5.113E-01	6.710E-01
SD	7.090E-01	**6.799E-01**	9.124E-01	1.035E+00
SU	9.937E-01	**9.671E-01**	1.320E+00	1.269E+00
STRI	4.969E-01	**4.524E-01**	4.905E-01	6.787E-01
THAL	6.376E-01	**5.755E-01**	6.162E-01	8.818E-01

[a] The missing channel.
Bold text give the best result for each tests.

Table 5.5: M1D Reconstruction.

# **Missing channel**[a]	RMSE (DN 1)	RMSE (DN 2)	RMSE (MLP)	RMSE (RNN)
1	5.688E-01	**4.980E-01**	4.985E-01	6.085E-01
2	5.272E-01	**4.581E-01**	4.609E-01	5.186E-01
3	7.163E-01	**6.258E-01**	8.102E-01	7.524E-01
4	1.013E+00	**7.834E-01**	3.182E+00	1.156E+00
5	1.281E+00	**9.991E-01**	2.874E+00	1.214E+00

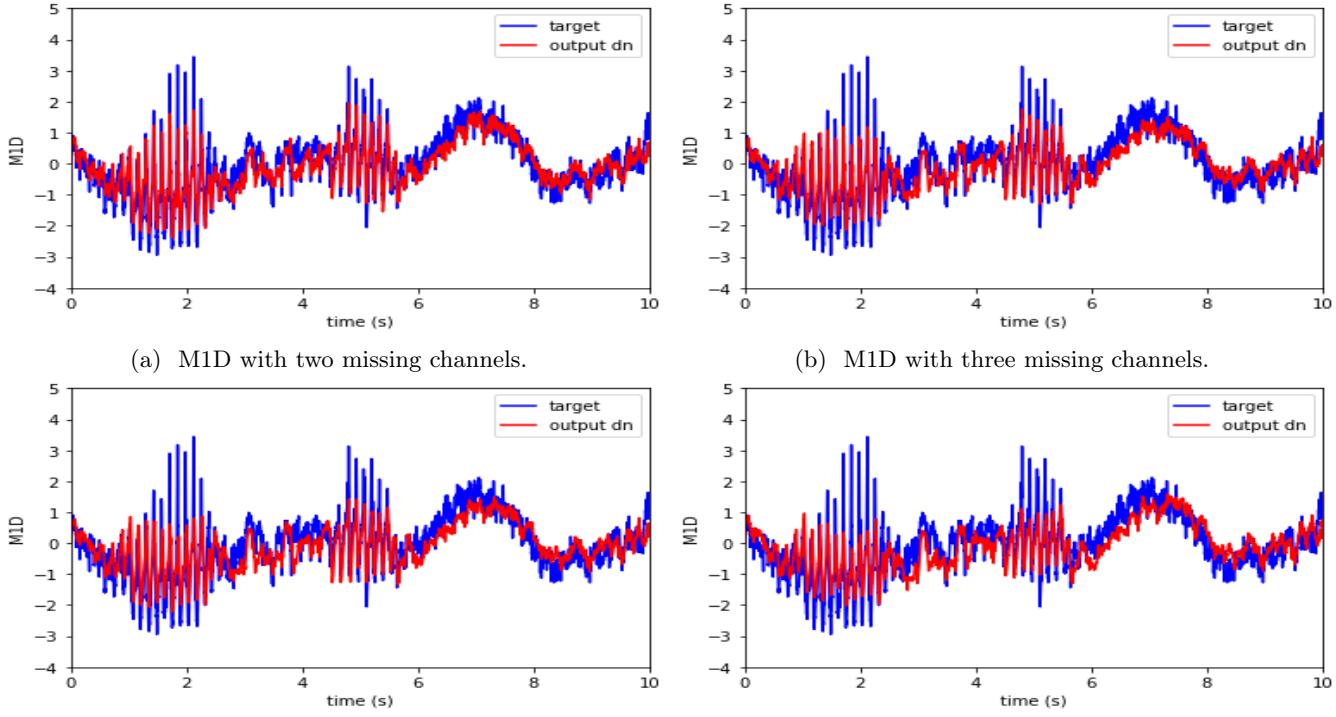
[a] Channels are removed in the order of channels presented in Table 5.1.
Bold text give the best result for tests.



(a) M1D (b) M1U

(c) STRI (d) SU

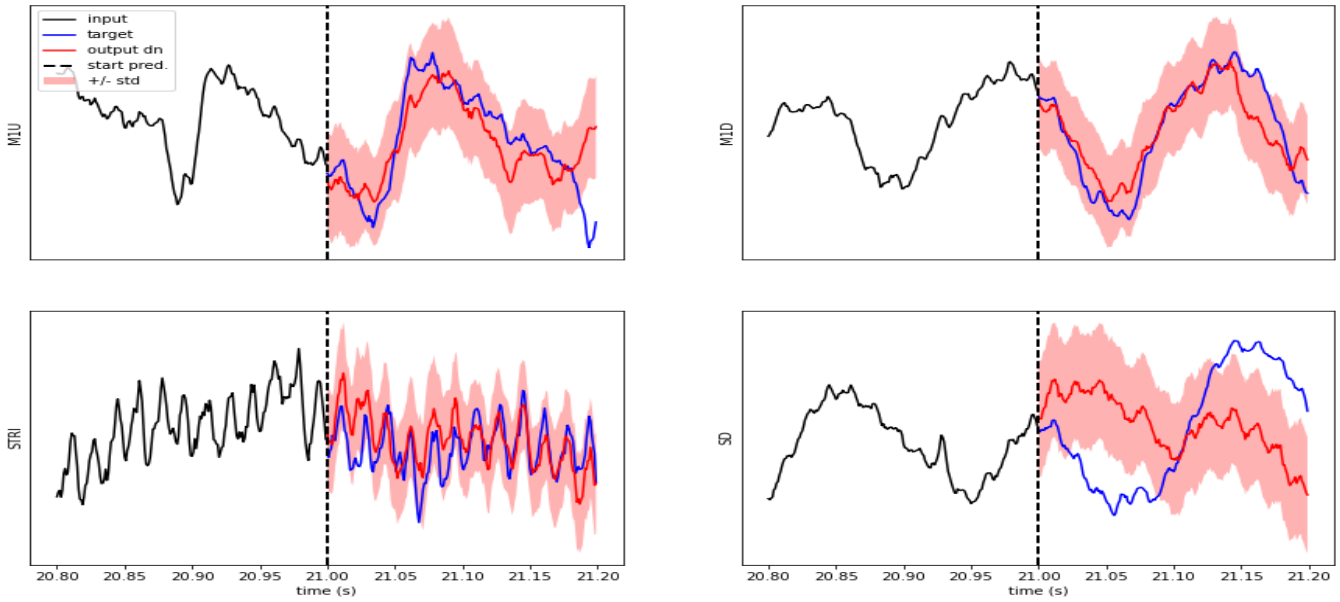
Figure 5.9: Reconstruction of missing channel. Observed signals are in blue and signals predicted by the DN in red.



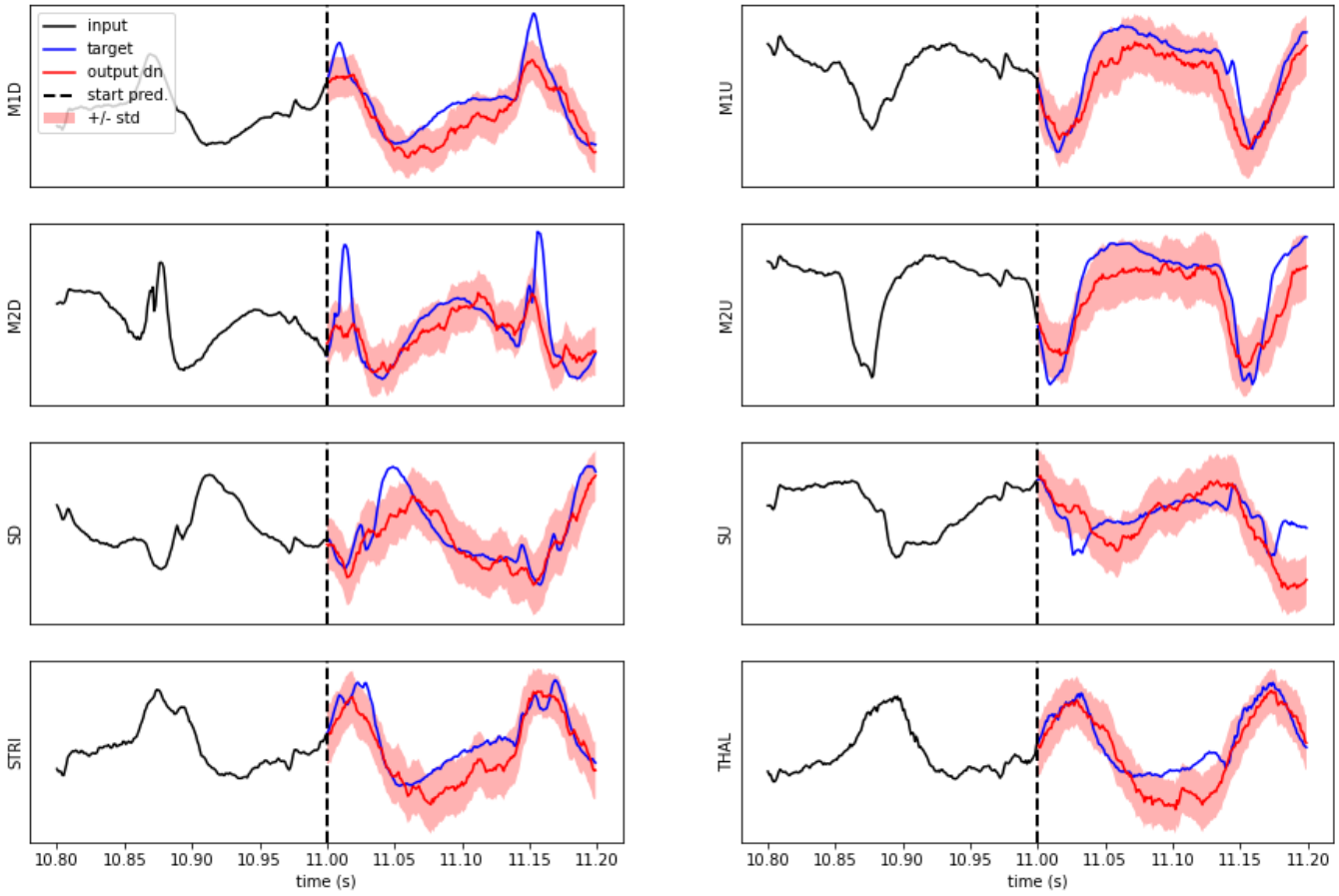
(a) M1D with two missing channels. (b) M1D with three missing channels.

(c) M1D with four missing channels. (d) M1D with five missing channels.

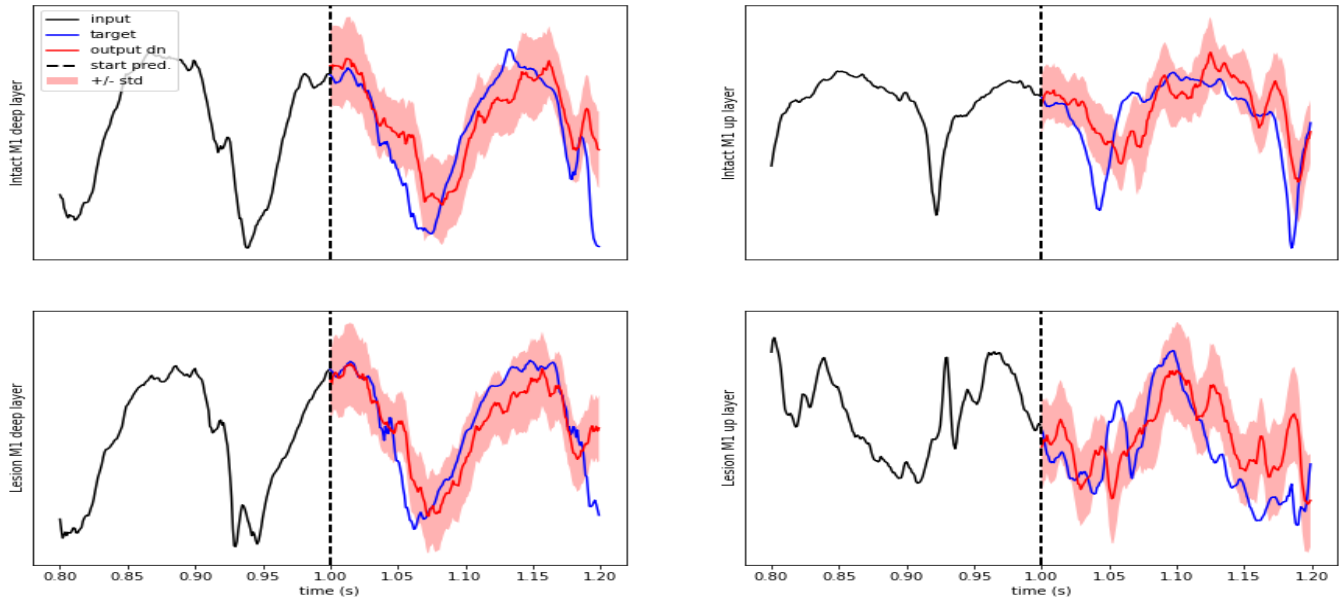
Figure 5.10: Reconstruction of missing channel for more than one missing channels. The blue color is put for real signals and the red color is put the signals predicted by the DN.



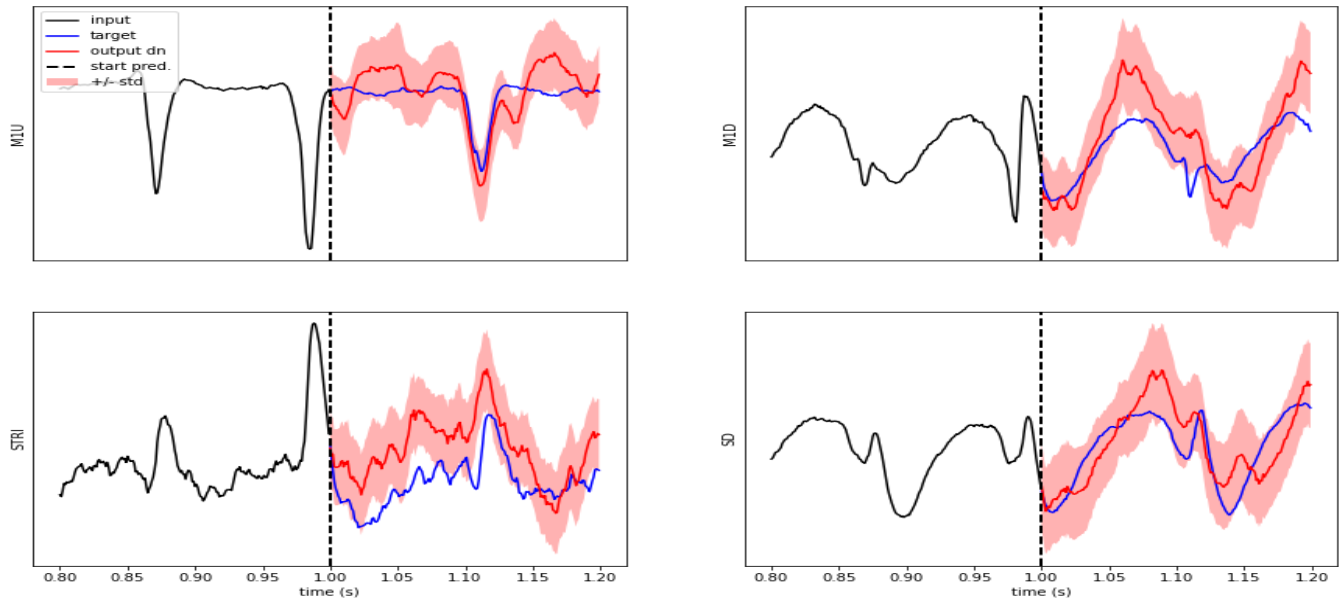
(a) Vector DN, example 1. (4 channels)



(b) Vector DN, example 2. (8 channels)



(c) Vector DN, example 3. (4 channels)



(d) Vector DN, example 4. (4 channels)

Figure 5.10: Vector DN, testing on iEEG data. The networks are composed of a time-window of 200ms with all the channels. The black signal represent the network state "$X(t_k)$". The targeted prediction "$X(t_{k+1})$" is the blue signal and the red signal gives the sampled signal. The red area is the diffusion term of each neuron. Each figures come from different rats on a sequence with HVS. Results depends on the PD rats.

Chapter 6

Conclusion

In this thesis, the use of artificial neural networks has been studied for applications involving time series data. Artificial neural networks are learning models inspired by the functioning of biological neural networks. The purpose of learning models is to allow the computer to learn by itself the rules governing data from a system using a training database. This solution is especially effective to deal with problems in many applications previously unreachable to human understanding. Today, research on artificial neural networks is very active, and is focused on different families of neural networks. Each family is characterized by common properties shared by the networks. The family based on Boltzmann machines, presented in this thesis, is issued from the family of *unsupervised* models. The output data are not necessary to learn these models, the Boltzmann machines identifying and learning by themselves data features. Boltzmann machines are also *stochastic*. Stochastic models are models involving random phenomena to be much closer to reality. Finally, Boltzmann machines are *generative* models. Unlike discriminative models that are limited to learn a conditional probability distribution $\Pr(Y|X)$, Boltzmann machines learn a joint probability distribution $\Pr(Y, X)$. This opens the door to Boltzmann machines to a wide range of applications unreachable for discriminative models such as the reconstruction of missing data for example. These properties explain the increasing interest of the neural networks community for Boltzmann machines in recent decades.

6.1 Boltzmann machines

The emergence of the Boltzmann family of machines began in 1982 thanks to Hopfield's work on binary and deterministic networks followed by deterministic and continuous networks. His work led Hinton to propose the famous RBM that has generated great interest in the neural network community. Numerous extensions were subsequently proposed to address the different constraints of data often encountered in applications. Thus, different directions to extend the scope of Boltzmann machines were proposed: networks to work with continuous visible units, deep networks, and networks to process time series, etc. The main strength of the Boltzmann machine family lies in its great flexibility: it is easy to transform the classic Boltzmann machine to fit to any problem. Many extensions of the Boltzmann machine have been often inspired by previous models. The bibliographic review proposed at the beginning of the thesis shows how research on Boltzmann machines was developed and gives main paths of improvement:

- **How the neurons interact each other?** Researchers have proposed different structures to take account priors of the data. For example mcRBM and GRBM encourage the network to capture correlations between variables. Some researchers proposed to adapt an existing model like Lee *et al.* in [55] with the conv-RBM. A mixture of models has also been studied, like the CssCDBM or the RNN-RBM.
- **How the information is processed inside a neuron?** Some modifications of the neuron structure have been proposed to better fit in with the data behavior. For example the GBRBM and the DN-RBM

have been proposed to deal with continuous data. Neuron structure has also been studied to improve the performance of the network (see for example [74]).

- **How the model is trained?** The motivation of this path of improvement is to reduce the bias between the data distribution $P_{data}(\boldsymbol{x})$ and the estimated data distribution $P_{model}(\boldsymbol{x})$ by modifying the cost function. This question has been addressed by changing the metrics between $P_{data}(\boldsymbol{x})$ and $P_{model}(\boldsymbol{x})$ as in [43, 61, 71] or by adding a penalization term [24].

The more neurons, the more computational efforts are needed: massive networks should not be the only way to reduce the modelling mistake. The choice of the dimension remains today an unsolved issue. There is neither theorem nor criterion (equivalent to the AIC for auto-regressive models or Extrem Learning Machine (ELM) methods for feedforward neural networks [18]) to help decide which is the optimal number of hidden neurons and hidden layers. Besides being energy consuming, a large dimension network requires much time to learn. In many papers, authors focus on the comparison of the performance between models but barely compare computational efforts between models. The issue of computational efforts can have a major impact in particular in real-time system for example but it is strongly depending on the data, the application and the used hardware.

The main part of this thesis consists in studying the use of generative models to model dynamic systems. DN-RBM are used to model a short-time window and are studied to understand how the models converge. By rewriting the DN-RBM energy function, we were able to interpret the hidden states of these models. The weights of the neurons in a DN-RBM learn the different frequency components contained in the learned signals. The states of the hidden units provide the information of the correlation between the observation window and the weight of the neurons. Based on these properties, we proposed two algorithms for automatic detection of non-stationary signals based on the energy of the model and the hidden states. Each method was able to learn the signals and detect them on toy data and iEEG data.

Such results are very promising. The DN-RBM learns by itself a signal representation from training samples. The only limitation of such an approach is that DN-RBM is an unsupervised model: there is no control about which features the model learns. A single hidden unit can be specialized on many features present in the database which can be a problem if some features are useful and the others are not. A few more investigations would be necessary to separate the components of interest according to the application.

6.2 Diffusion Network

Proposed by Movellan, the DN is the stochastic extension of the cHN. But contrary to RBMs, DNs did not draw the same attention of the neural network community. In addition being an extension of the cHN, the DN generalizes the Markov model family among which HMM, Kalman-Bucy filters, BM or RNN are special cases. *E.g.* a HMM can be seen as a DN with discrete-valued hidden states and discrete-time dynamics; a Kalman filter is a DN driven by linear dynamics; a RNN is a zero-noise DN. Fig. 6.1 summarizes the different relations between the DN and other models. In this thesis, the DN has been re-explored in order to propose new algorithms and to test it in different situations.

First, the SDE has been rewritten to reduce the number of parameters. The Girsanov's theorem is used to learn the transfer matrix, the bias vector and the noise power (σ_j) as well as the filter RC parameters (τ_j) are estimated directly from the data. The missing data reconstruction and the prediction application are tested on toy data and iEEG data. The DN provides interesting results. Properties of the model are well suited to handle times series data. The comparison with the other models has to be taken with care. Indeed, the choice of the dimension of the comparison models is purely arbitrary; there is no doubt that the comparison models can provide better results with dimensions chosen optimally. The idea was to keep an equivalent number of neurons because the DNs tested do not use hidden neurons.

More studies are needed to evaluate the performance of DNs and understand the limits of the models. Two major issues of the DN require more investigations. First, tests with hidden units do not improve

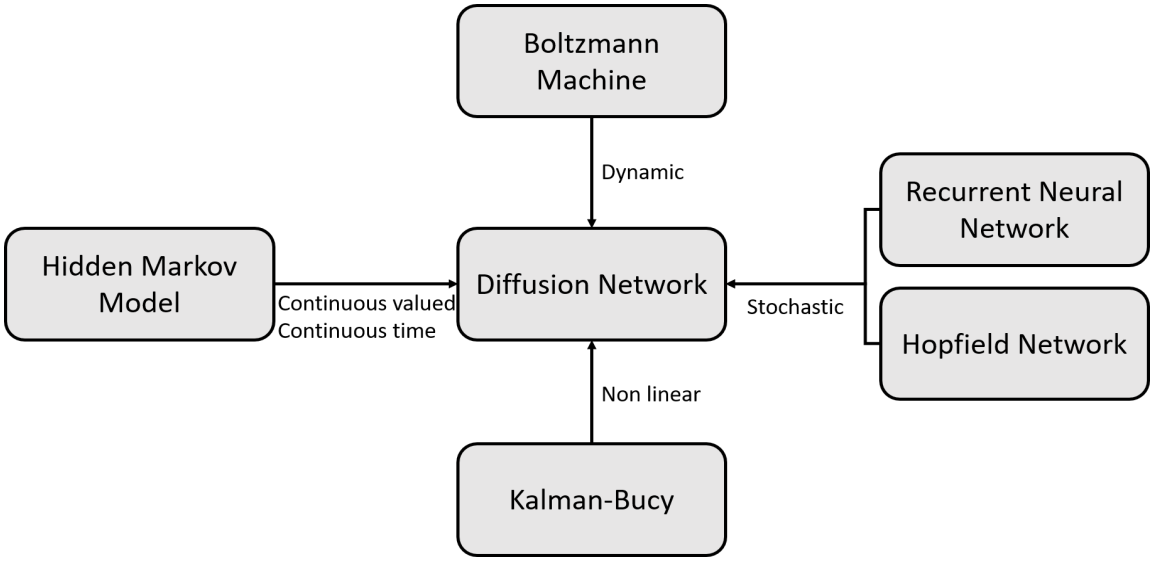


Figure 6.1: Relation between the Diffusion network and other models.

results of the proposed applications. According to our observations, the DN is able to learn hidden units: by sampling hidden units in parallel to visible units, the hidden signal shares common properties with the visible signal but evaluation tools are needed. Moreover, the estimation of the noise power and the RC filter parameters are performed by using observable data. There is no rule for the choice of the noise power and the RC filter parameters of hidden neurons. Using additional latent units does not seem to be a solution. One possible direction of improvement could be finding a new neuron's architecture using a latent memory like in LSTM. The second limit of the DN is the estimation of the RC filter parameters. The neuron structure consists in a linear filter RC and a nonlinear perturbation caused by the state of the network. The role of the parameter is to compete with the perturbation to prevent the neuron j to diverge too much. In other words, the more τ_j is large the more the neurons will converge towards zero. The training τ_j using Girsanov's theorem tends to have a too small parameter (in absolute value). An estimation based on the mean variation has been proposed to compute τ_j. This estimation is also depending on a hyper-parameter fixed by the user. The reason for the failure to learn τ_j using Girsanov's theorem the parameter is not yet widely understood.

Despite these limitations, the results of DNs on toy data and iEEG data are encouraging. This study shows once again how flexible the neural networks are. In addition to being able to modify the structure of neurons or the organization of the graph, it is also possible to change the function to model. For example, a feedforward learns a function ($y = f(x)$), the Boltzmann machines learn a joint probability distribution ($\Pr(x, y)$) and a DN learns a SDE ($\mathrm{d}X(t) = \mu(X(t), t)\mathrm{d}t + \sigma\mathrm{d}B(t)$).

6.3 Medical applications

In this thesis, we proposed the use of the DN-RBM to detect the presence of HVS. The DN-RBM can learn by itself frequency informations. The transfer matrix stores features present in the training database and the hidden state gives the correlation between the signal and the learned components. Once the model trained, two classifiers based on the hidden state and the energy level have been studied and both allow to efficiently detect the HVS. Results have been compared to a ground truth based on the CWT and a third learning model. The third model is an autoencoder followed by a GP classifier, the GP is a supervised model. The proposed approaches succeed to detect the HVS and the next step of the study consists of tests on closed-loop systems. We have to verify if the detection still works in the closed-loop systems and we have to ensure models will work in real-time.

Appendix A

Gradient Descent methods for neural networks

The key in artificial intelligence is the learning procedure of parameters. Note λ the *vector of parameters* of the studied model. $\mathcal{L}^{\lambda}$, the loss function to minimize and the training set $\mathcal{D}$. During the training step, the aim is to solve the following equation:

$$\widehat{\lambda} = \arg\min_{\lambda} \mathcal{L}^{\lambda}(\mathcal{D}_{train}). \tag{A.1}$$

There are different options to solve the equation. First, it is possible to write an explicit expression of $\widehat{\lambda}$. Then, the algorithm expectation maximization can be used to find a solution. In the case where the first option is not possible, the gradient descent method is a solution widely used in machine learning.

A.1 Batch Gradient descent

For a fixed λ, the gradient $\nabla_{\lambda}\mathcal{L}^{\lambda}(\mathcal{D})$ gives the direction vector to move λ in order to decrease the loss function. Batch gradient algorithm (or Vanilla Gradient Descent) consists in updating λ from a random initialization using the equation:

$$\lambda^{k+1} = \lambda^{k} - \eta\nabla_{\lambda}\mathcal{L}^{\lambda^{k}}(\mathcal{D}) \tag{A.2}$$

where k is the iteration of the learning procedure and η is the learning rate. The full training set is used to update the parameter at once, this method is called the *batch gradient descent* (see Algo. 5).

A.2 Mini-batch gradient descent

To converge faster, the *stochastic gradient descent* algorithm updates parameters by computing the gradient of the loss function using only one sample randomly selected from the training set. During the learning step, the loss function converge faster but present many fluctuations as in Fig. A.1 due to the random selection of sample. One drawback of the stochastic gradient descent is the risk to not converge towards the best solution.

The *mini-batch gradient descent* algorithm (see Algo. 6) is a compromise between the batch gradient descent and the stochastic gradient descent. A small set of data (of size bs) is randomly selected at each update. During one epoch, the vector of parameters is updated as many times as possible to cover the training set according to (Eq. A.3).

$$\lambda^{i+1} = \lambda^{i} - \eta\nabla_{\lambda}\mathcal{L}^{\lambda^{k}}(X_{(i-1)\times bs:i\times bs}, Y_{(i-1)\times bs:i\times bs}) \tag{A.3}$$

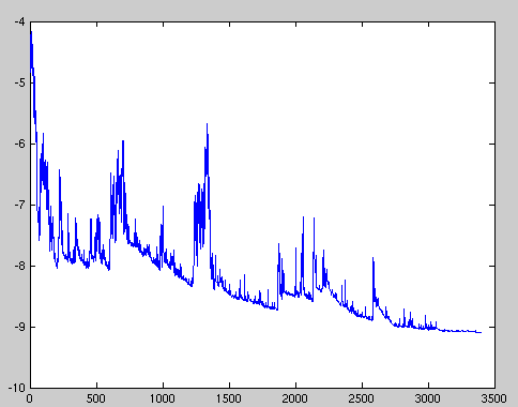


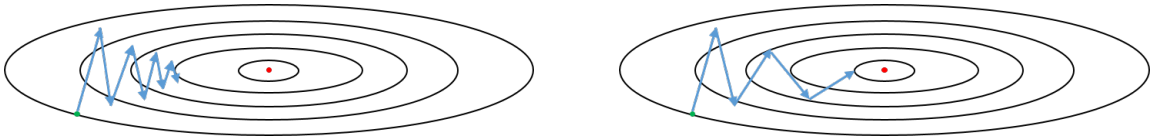
Figure A.1: Stochastic gradient descent fluctuation. Figure from [91].

Algorithm 5 Batch Gradient Descent

Require: The training and the validation database $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$,
Require: The learning rate η and the number of epochs K,
Require: Initial state of parameters λ^0 randomly initialized.
  for $k \leftarrow 0$ **to** $K-1$ **do**
    Compute $\nabla_\lambda \mathcal{L}^{\lambda^k}(\mathcal{D}_{train})$.
    Update λ^{k+1} according to (Eq. A.2).
    Display $\mathcal{L}^{\lambda^{k+1}}(\mathcal{D}_{train})$ and $\mathcal{L}^{\lambda^{k+1}}(\mathcal{D}_{valid})$ for control.
  end for
  return λ^K

Algorithm 6 Mini-batch Gradient Descent

Require: The training and the validation database $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$,
Require: The learning rate η and the number of epochs K,
Require: Initial state of parameters λ^0 randomly initialized.
  for $k \leftarrow 0$ **to** $K-1$ **do**
    Note $\lambda^{temp} = \lambda^k$
    Shuffle the order of the training set using the permutation operator.
    for $i \leftarrow 1$ **to** $\lfloor N_{train}/bs \rfloor$ **do**
      Compute $\nabla_\lambda \mathcal{L}^{\lambda^{temp}}(X_{(i-1)\times bs:i\times bs}, Y_{(i-1)\times bs:i\times bs})$.
      Update λ^{temp} according to (Eq. A.3).
    end for
    Set $\lambda^{k+1} = \lambda^{temp}$
    Display $\mathcal{L}^{\lambda^{k+1}}(\mathcal{D}_{train})$ and $\mathcal{L}^{\lambda^{k+1}}(\mathcal{D}_{valid})$ for control.
  end for
  return λ^K



(a) Without momentum. (b) With momentum.

Figure A.2: Momentum role. The scheme is an example of a two-dimensional map of the loss function where the red point is the optimal solution, the green point is the initial state of the parameters and the blue arrow represents the update of λ. The momentum allows to keep in memory direction of last iterations and to adapt the learning rate for each dimension.

A.3 Gradient descent with momentum

Many challenges have been addressed to the gradient descent algorithm for neural networks and are reviewed in [91]. The main issue of the gradient descent rely on the choose of the learning rate η. A small learning rate provides accurate result but with slow convergence while a high η is faster but can never converge. In addition, not all parameters have the same influence on the loss function and require the learning rate to be adjusted. Different approaches exist to improve the gradient descent algorithm. For example: heuristic rules can be added in function of the evolution of the loss function.

Very used in gradient descent extension, the *momentum* [85] consists in keeping in memory gradient of previous iterations. This method encourages a direction to update λ and helps the model to converge faster. The gradient descent with momentum use a momentum vector v to update λ as in (Eq. A.4):

$$\begin{cases} v_{k+1} = \gamma v_k + (1-\gamma)\nabla_\lambda \mathcal{L}(\lambda^k; \mathcal{D}) \\ \lambda^{k+1} = \lambda^k - v_{k+1} \end{cases} \tag{A.4}$$

The vector momentum takes into consideration last updates of parameters. The hyper-parameters γ allows to control how fast the momentum "forgets" last updates and the weigth of the last gradient to update parameters.

A.4 Gradient descent with adaptive learning rate: example with RMSprop

The impact of parameter variation on the loss function can be very different. Some parameters can require larger updates while the other ones need smaller updates. Many algorithms have been proposed to adapt the learning rate to each parameters like, for example, ADAGRAD [20], ADADELTA [120] or ADAM [46]. In this thesis, the RMSPROP has been widely used to train the models. This algorithm adapts the learning rate for each parameters independently with an equation of the form:

$$\lambda_i^{k+1} = \lambda_i^k - \frac{\eta}{\sqrt{G_i^k + \epsilon}} \nabla_\lambda \mathcal{L}^{\lambda^k}(\mathcal{D}) \tag{A.5}$$

where G_i^k is function of previous gradient and provides larger updates for parameters associated with infrequent features. The expression of G_i^k depends on the chosen algorithm. In the case of RMSPROP [91], G_i^k is equal to the momentum of past square gradient:

$$G^k = \gamma G^{k-1} + (1-\gamma)\nabla_\lambda (\mathcal{L}^{\lambda^k})^2(\mathcal{D}) \tag{A.6}$$

In (Eq. A.5), ϵ is a smoothing term that avoids division by zero.

In practice, these algorithms are proposed in python programming packages for deep learning: Keras or Pytorch.

```
model = net(hyperparameters)
opt = optim.RMSprop(model.parameters(), lr=1e-3, alpha=0.9, eps=1e-8)
```

(a) Initialization of the model and the optimizer.

```
Yhat = model.forward(X) ## Estimate the output with forward
loss = lossFunc(Y,Yhat) ## Compute the loss function
loss.backward() ## Compute the gradient of all parameters
opt.step() ## Update the parameters
opt.zero_grad() ## Free the memory for next epoch
```

(b) Update the parameters.

Figure A.3: Gradient descent with Pytorch. **(a)** gives the notation for the initialization. "model" is a class which contains at least the parameters and the function forward. "opt" is the optimizer (algorithm) class used to train the model. "lr" refers to η, "alpha" refers to γ and "epsilon" is equal to ϵ in (Eq. A.5). **(b)** gives the main lines of the gradient descent algorithm. In the case of unsupervised model (without output), the loss is computed directly from the input X only. Note that in python functions and variables "object_name" inside a class "class_name" can be accessed by "class_name.object_name".

A.5 Programming gradient descent with Pytorch

The different models tested were coded on python during the thesis. Python is a programming language offering several packages dedicated to machine learning and neural networks. Among these packages, Pytorch was chosen to code the models to be tested. Many tools to build deep networks easily are available. The programmer first defines a class (named module in pytorch) by initializing the full set of parameters. Pytorch recognizes the parameter of a model. Each time a variable is computed using the parameters, the program keeps in memory the calculation. Then the function *forward* which computes the output of the network is added in the class. Once the model is initialized, the programmer introduce an optimizer object. The optimizer is a class referring to a learning algorithm, it takes as argument, the parameters of the model and the hyper-parameters relative to the algorithm.

During the training procedure, the loss is computed and gradient of all parameters is computed using simply the function *backward*. This function allows the programmer to not compute the gradient by himself thanks to the program which keeps in memory the chain the calculation from the parameters to the loss function. Finally, the function *step* of the optimizer updates the parameters. Fig. A.3 gives few lines as example for coding the gradient descent.

Appendix B

DN-RBM for the modelling of time-window: other experiments

In Chapter 3, tests on toy signals have been performed to evaluate the capacity of the DN-RBM to learn a signal decomposition. In this appendix, three additional experiments are proposed. These experiences are not presented in Chapter 3 to not overloaded it.

B.1 Learning a chirp

In previous experiments, the DN-RBM has been tested to learn non-stationary signals with fix frequencies. Now, the DN-RBM is used to learn a chirp. A chirp is a pseudo-periodical signal with a continuously increasing frequency:

$$s(t) = \sin(2\pi(af \times t)t) \tag{B.1}$$

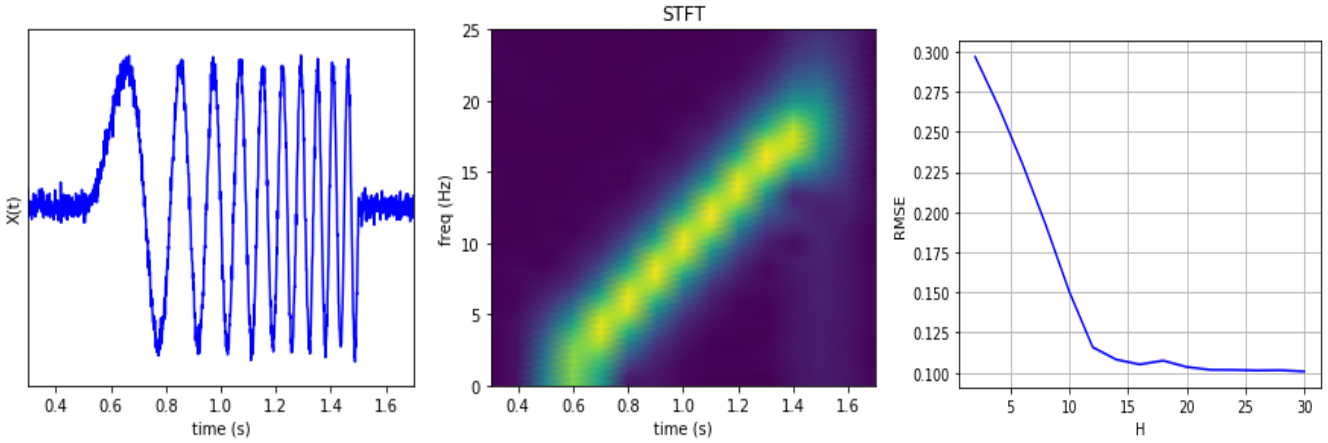
At time t, the instantaneous frequency of s is equal to $af \times t$ (a is a coefficient fixed in this application to control how the instantaneous frequency increase). The toy data is a non-stationary signal with chirp appearing during 1 second. Fig. B.1 gives a sequence of the toy data with a chirp ($a = 10$). DN-RBMs are trained on time-window of 300 ms. Results of reconstruction and the learned transfer matrix are given for DN-RBMs with, two, five and sixteen hidden units. Figs. B.2-B.4 gives results for the three models. The reconstruction of the visible layer is given for two instants of the chirp (low frequencies and high frequencies). The DN-RBM is able to learn a chirp. Lines of the transfer matrix learn multiple frequencies and the model is able to reconstruct the chirp. Fig. B.1b gives the RMSE in function of the size of the hidden layer. All the frequencies are learned with 15 and more hidden units.

B.2 Signal-to-Noise Ratio

The impact of the Signal-to-Noise Ratio (SNR) is studied in this section. In previous tests, the capacity of the DN-RBM to model frequencies features has been studied regardless the noise level. Let $x(t)$ the signal to learn defined as:

$$\begin{cases} x(t) & = s(t) + n(t) \\ s(t) & = \sin(2\pi ft) \\ n(t) & \sim \mathcal{N}(0, \sigma^2) \end{cases} \tag{B.2}$$

where the frequency f is fixed to 5 Hertz and the SNR is equal to $0.5\sigma^{-2}$. DN-RBMs with two hidden units are trained for different values of σ^{-2}. Results are given in Fig. B.5 for different values of the SNR. Results show the DN-RBM can learn the sinus even in the case of low SNR. The DN-RBM is still able to learn frequencies even if the frequencies in question are not visible by human (see Figs. B.5c - B.5d). However,



(a) Non-stationary chirp in toy data .

(b) RMSE function of the size of the hidden layer .

Figure B.1: Non-stationary chirp in toy data and RMSE. In **(a)**, the left figure in a zoom on a time-window with the chirp signal. The right plot is the same signal using Short-Term Fourier Transform representation. The figure in **(b)** give the RMSE in function of the size of the hidden layer.

there is a level of the SNR for which the DN-RBM is no longer able to learn the frequency anymore (see Figs. B.5e - B.5f).

B.3 The lower and the upper bounds

The last study concerns the influence of the activation function. As presented in the Sec. 2.3.3, the activation function of neurons is a sïgmoid function:

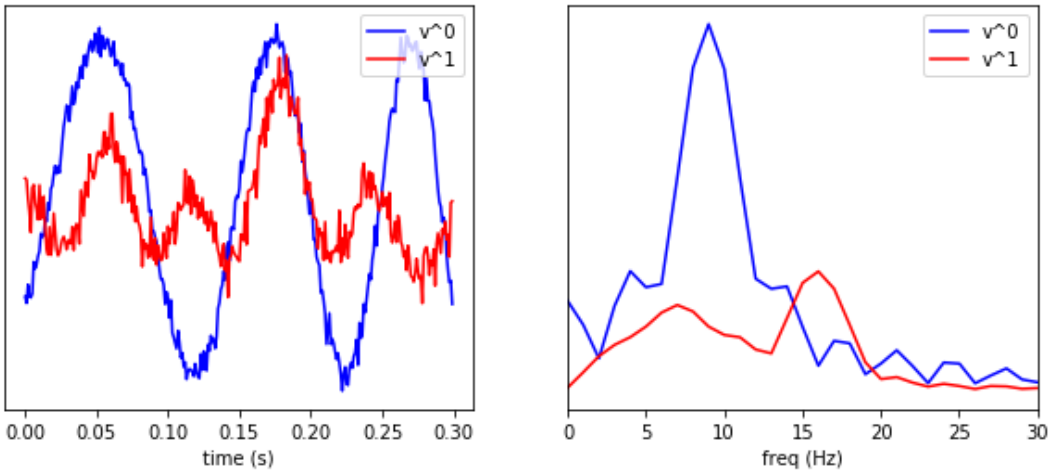
$$\phi_i(x) = \theta_L + (\theta_H - \theta_L)\frac{1}{1+\exp\left(-a_i x\right)} \tag{B.3}$$

where θ_L and θ_H are, respectively, the lower and upper bounds of the neuron (see Fig. 2.12). For hidden units, the lower and the upper bound are fixed to: $\theta_H = -\theta_L = 1$. For the visible units, the choice of the upper and the lower bound have an impact on the results. The first point is the update rule of the activation function parameter a_i. If s_i is the neuron state of the $i-$th neuron, then the update rule of the activation function parameter is given by (see also (Eq. 2.46)):

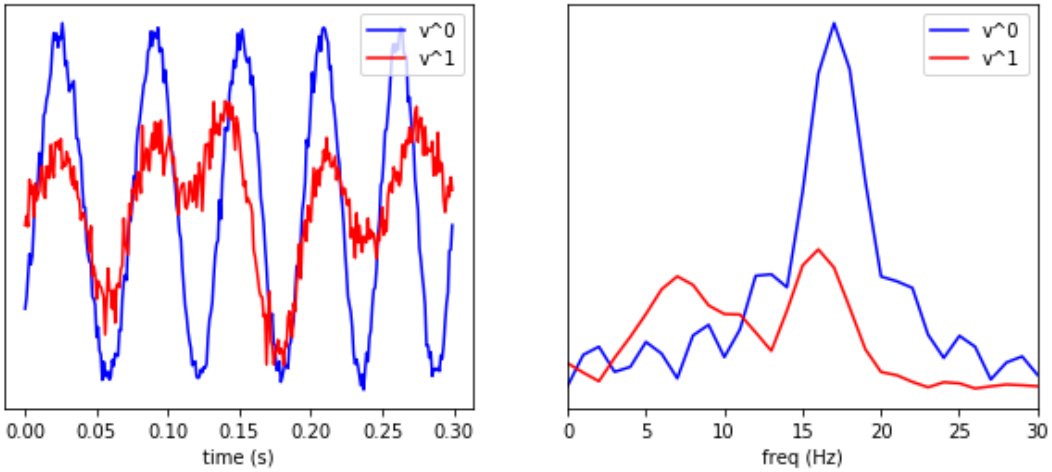
$$\begin{aligned}\Delta a_i \propto \frac{1}{a_i^2}\int_{\langle s_i\rangle_1}^{\langle s_i\rangle_0}\phi^{-1}(s')\mathrm{d}s' = &\frac{1}{a_i^2}\left((\langle s_i\rangle_0 - \theta_L)\log(\langle s_i\rangle_0 - \theta_L) + (\theta_H - (\langle s_i\rangle_0)\log(\theta_H - (\langle s_i\rangle_0)\right)\\ &- \frac{1}{a_i^2}\left((\langle s_i\rangle_1 - \theta_L)\log(\langle s_i\rangle_1 - \theta_L) + (\theta_H - (\langle s_i\rangle_1)\log(\theta_H - (\langle s_i\rangle_1)\right).\end{aligned} \tag{B.4}$$

To be able to compute the gradient of a_i, the training database must be bounded between θ_L and θ_H. And then, the choice of θ_L and θ_H will have an influence on parameters a_i for all i.

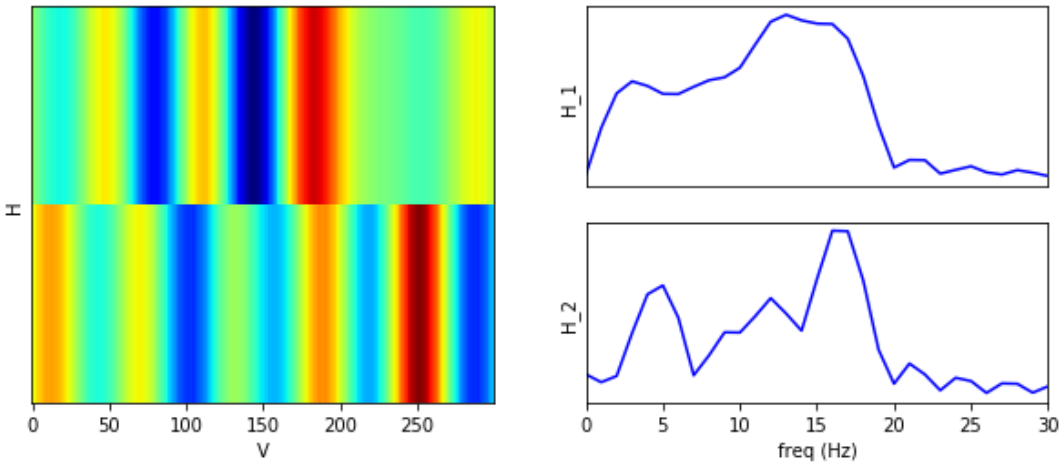
For all the previous tests (toy data and iEEG data), bounds of visible units have been fixed to: $\theta_H = -\theta_L = 15$ to avoid the saturation of data. To illustrate the influence of the choice of bounds of visible units, Figs. B.5g-B.5h, from the previous section, give results with a SNR equal to 0.32 (same as in Figs. B.5e-B.5f) but with $\theta_H = -\theta_L = 5$. In this case, the DN-RBM success to learn the frequency of $s(t)$ according to Fig. B.5h. Unfortunately, there is no rule for the choice of the bounds.



(a) Reconstruction 1.

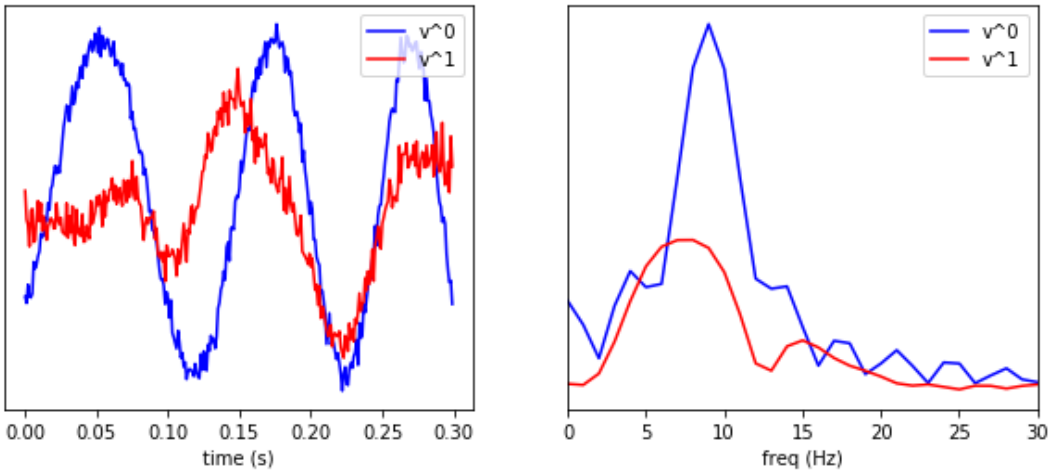


(b) Reconstruction 2.

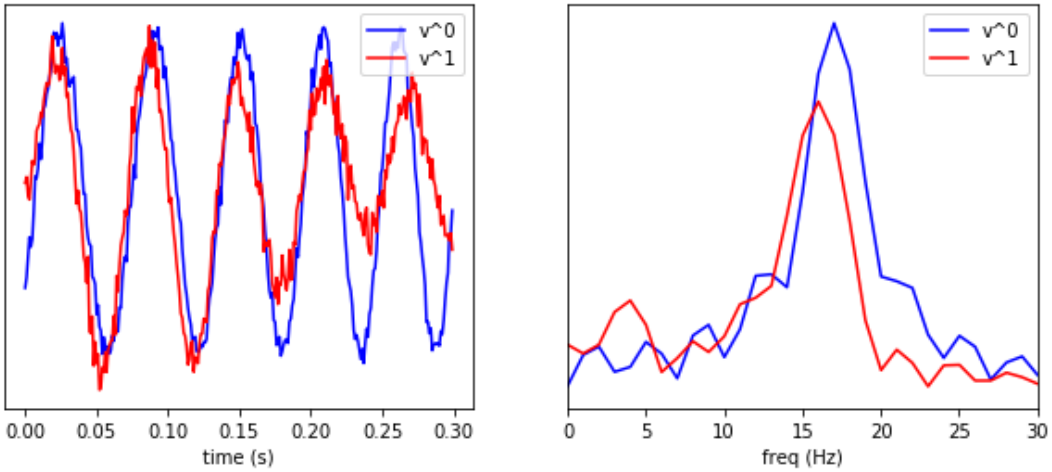


(c) Transfer matrix.

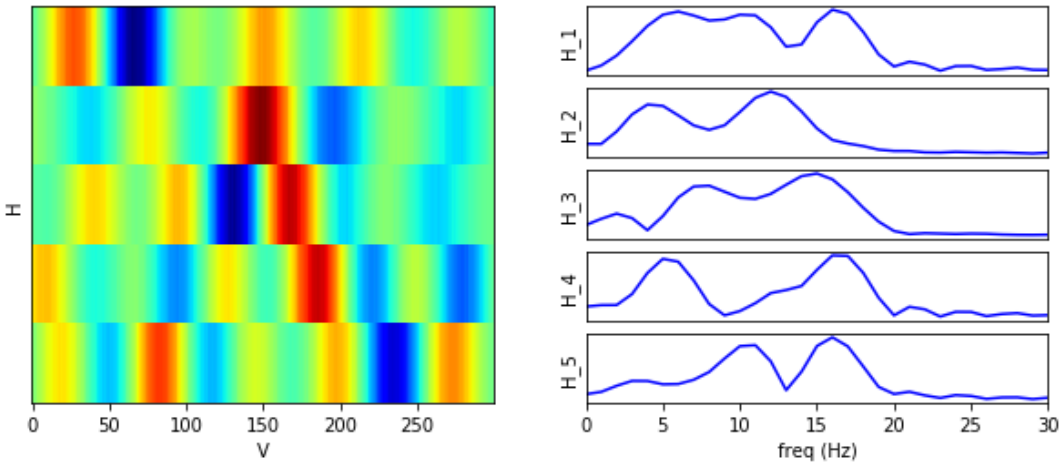
Figure B.2: Learning chirp with two hidden units. **(a)** and **(b)** are two reconstructions of visible layers at different moments of the chirp (the chirp duration is 1 second and the time-window of the visible layer is 300 millisecond). **(c)** gives the learned transfer matrix and the Fourier transform of each line.



(a) Reconstruction 1.

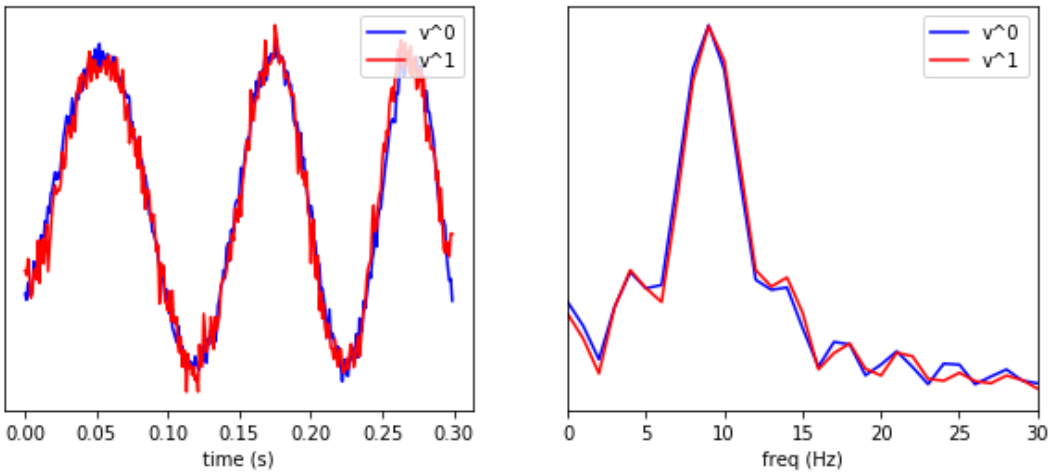


(b) Reconstruction 2.

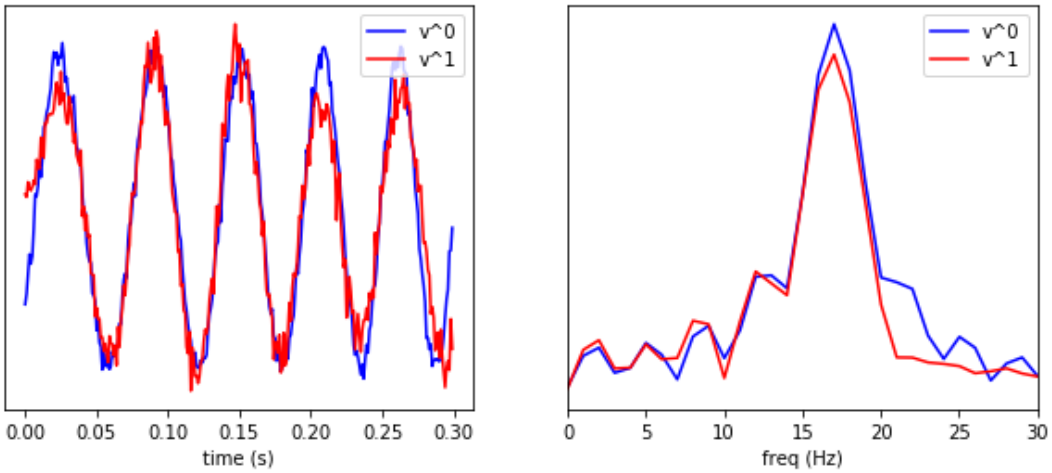


(c) Transfer matrix.

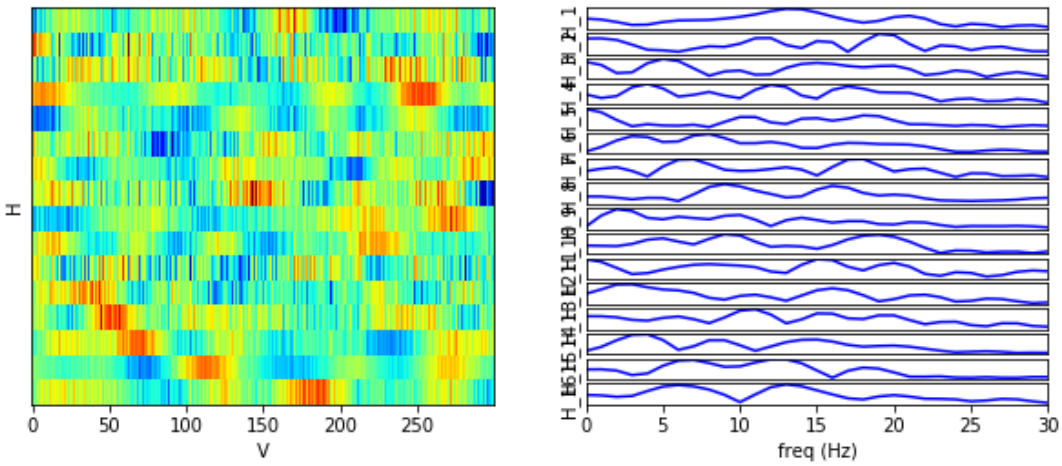
Figure B.3: Learning chirp with five hidden units. **(a)** and **(b)** are two reconstructions of visible layers at different moments of the chirp (the chirp duration is 1 second and the time-window of the visible layer is 300 millisecond). **(c)** gives the learned transfer matrix and the Fourier transform of each line.



(a) Reconstruction 1.

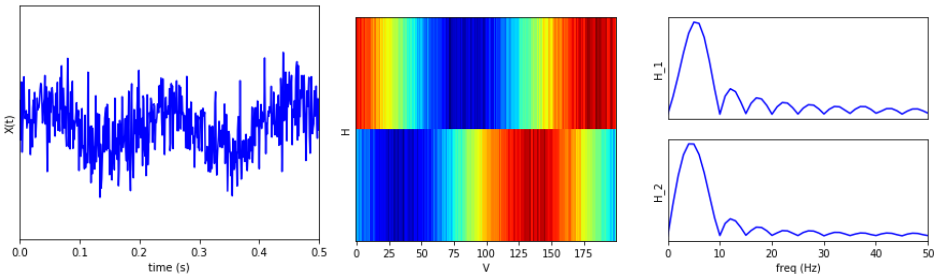


(b) Reconstruction 2.



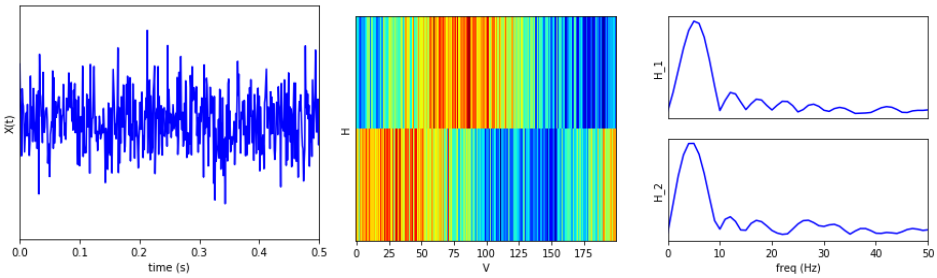
(c) Transfer matrix.

Figure B.4: Learning chirp with sixteen hidden units. **(a)** and **(b)** are two reconstructions of visible layers at different moments of the chirp (the chirp duration is 1 second and the time-window of the visible layer is 300 millisecond). **(c)** gives the learned transfer matrix and the Fourier transform of each line.



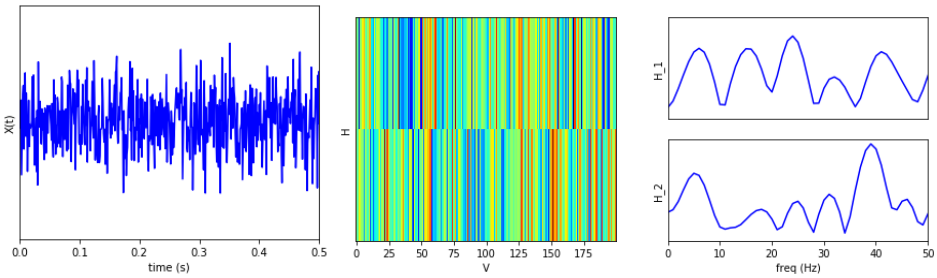
(a) SNR = 0.5, training signal.

(b) SNR = 0.5, transfer matrix.



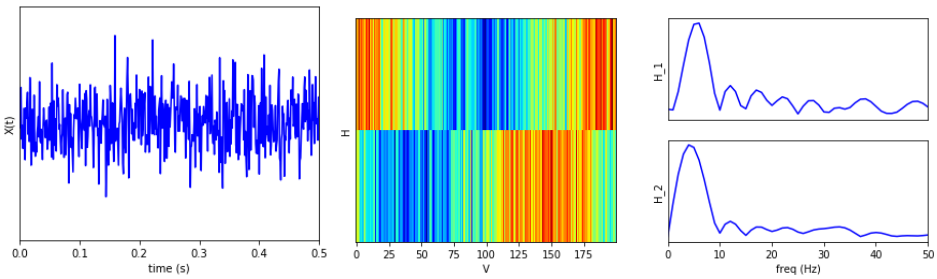
(c) SNR = 0.02, training signal.

(d) SNR = 0.02, transfer matrix.



(e) SNR = 0.005, training signal.

(f) SNR = 0.005, transfer matrix.



(g) SNR = 0.005, training signal (2).

(h) SNR = 0.005, transfer matrix (2).

Figure B.5: Influence of the SNR. Each line is organized as bellow. The left plot is a zoom of the training set. The plot at the middle is the learned transfer matrix and the right plot are the Fourier transform of each row of the transfer matrix. On the last line (Figs. B.5g-B.5h), the experience from the previous lines has been applied a second time but with another set of hyper-parameters (see Sec. B.3). The lower and the upper bounds of visible units have been changed.

Bibliography

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, Inc., New York, NY, USA, 1989. ISBN 0-471-92146-7.

[2] J. Barron. Continuously differentiable exponential linear units, 2017.

[3] A. Benabid, S. Chabardes, J. Mitrofanis, and P. Pollak. Deep brain stimulation of the subthalamic nucleus for the treatment of parkinson's disease. *The Lancet Neurology*, 8(1):67–81, 2009.

[4] Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural computation*, 21(6):1601–1621, 2009.

[5] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[6] M. Carreira-Perpinan and G. Hinton. On contrastive divergence learning. In *Conference of Artificial Intelligence and Statistics*, volume 10, pages 33–40, 2005.

[7] C. Chen, C. Zhang, L. Chen, and M. Gan. Fuzzy restricted boltzmann machine for the enhancement of deep learning. *IEEE Transactions on Fuzzy Systems*, 23(6):2163–2173, 2015.

[8] H. Chen and A. Murray. Continuous restricted boltzmann machine with an implementable training algorithm. *IEE Proceedings-Vision, Image and Signal Processing*, 150(3):153–158, 2003.

[9] H. Chen, P. Fleury, and A. Murray. Continuous-valued probabilistic behavior in a vlsi generative model. *IEEE Transactions on Neural Networks*, 17(3):755–770, 2006.

[10] Y.-C. Chen, C.-C. Chang, R. Perumal, S.-R. Yeh, Y.-C. Chang, and H. Chen. Optimization and implementation of wavelet-based algorithms for detecting high-voltage spindles in neuron signals. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5):1–16, 2019.

[11] K. Cho, A. Ilin, and T. Raiko. Improved learning of gaussian-bernoulli restricted boltzmann machines. In *International conference on artificial neural networks*, pages 10–17, 2011.

[12] K. Cho, T. Raiko, and A. Ilin. Gaussian-bernoulli deep boltzmann machine. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Aug 2013. doi: 10.1109/IJCNN.2013.6706831.

[13] A. Courville, J. Bergstra, and Y. Bengio. A spike and slab restricted boltzmann machine. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 233–241, 2011.

[14] A. Courville, G. Desjardins, J. Bergstra, and Y. Bengio. The spike-and-slab rbm and extensions to discrete and sparse data distributions. *IEEE transactions on pattern analysis and machine intelligence*, 36(9):1874–1887, 2014.

[15] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[16] G. Dahl, A. Mohamed, G. Hinton, and R. M.A. Phone recognition with the mean-covariance restricted boltzmann machine. In *Advances in neural information processing systems*, pages 469–477, 2010.

[17] C. Dejean, C. Gross, B. Bioulac, and T. Boraud. Dynamic changes in the cortex-basal ganglia network after dopamine depletion in the rat. *Journal of neurophysiology*, 100(1):385–396, 2008.

[18] S. Ding, X. Xu, and R. Nie. Extreme learning machine and its applications. *Neural Computing and Applications*, 25(3-4):549–556, 2014.

[19] E. Dorsey, R. Constantinescu, J. Thompson, K. Biglan, R. Holloway, K. Kieburtz, F. Marshall, B. Ravina, G. Schifitto, A. Siderowf, et al. Projected number of people with parkinson disease in the most populous nations, 2005 through 2030. *Neurology*, 68(5):384–386, 2007.

[20] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[21] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[22] M. D.Z., G. W.T., L. Sigal, I. M., and R. Fergus. Facial expression transfer with input-output temporal restricted boltzmann machines. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1629–1637. Curran Associates, Inc., 2011. URL `http://papers.nips.cc/paper/4368-facial-expression-transfer-with-input-output-temporal-restricted-boltzmann-machines.pdf`.

[23] S. Fahlman, G. Hinton, and T. Sejnowski. Massively parallel architectures for AI: NETL, thistle, and boltzmann machines. In *National Conference on Artificial Intelligence*, pages 109–113, 01 1983.

[24] C. Fisher, A. Smith, and J. Walsh. Boltzmann encoded adversarial machines, 2018.

[25] I. Goodfellow, A. Courville, and Y. Bengio. Spike-and-slab sparse coding for unsupervised feature discovery. *arXiv preprint arXiv:1201.3382*, 2012.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[27] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[28] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[29] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[30] G. Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[31] G. Hinton. A practical guide to training restricted boltzmann machines. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700, pages 599–619. Springer, 2012. URL `http://dblp.uni-trier.de/db/series/lncs/lncs7700.html#Hinton12`.

[32] G. Hinton and R. M.A. Modeling pixel means and covariances using factorized third-order boltzmann machines. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2551–2558. IEEE, 2010.

[33] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[34] G. Hinton and T. Sejnowski. Learning and releaming in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986.

[35] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[37] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[38] J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[39] J. Hopfield and D. Tank. Computing with neural circuits: A model. *Science*, 233(4764):625–633, 1986.

[40] Y. Hu, J. Liu, J. You, and P. Chan. Continuous rbm based deep neural network for wind speed forecasting in hong kong. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, page 368. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[41] G. Huang, H. Lee, and E. Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2518–2525, 2012.

[42] H. Huang, R. Li, M. Yang, T. Lim, and W. Ding. Evaluation of vehicle interior sound quality using a continuous restricted boltzmann machine-based DBN. *Mechanical Systems and Signal Processing*, 84:245–267, 2017.

[43] A. Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5): 2499–2512, 2007.

[44] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[45] S. Khan and T. Yairi. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107:241–265, 2018.

[46] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks, 2017.

[48] V. Kuleshov and S. Ermon. Neural variational inference and learning in undirected graphical models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6734–6743. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7250-neural-variational-inference-and-learning-in-undirected-graphical-models.pdf`.

[49] S. Kullback and R. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22 (1):79–86, 1951.

[50] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi. Time series forecasting using a deep belief network with restricted boltzmann machines. *Neurocomputing*, 137:47–56, 2014.

[51] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[52] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM, 2008.

[53] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio. Learning algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, 13:643–669, Mar 2012.

[54] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[55] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553453. URL `http://doi.acm.org/10.1145/1553374.1553453`.

[56] H. Lee, P. Pham, Y. Largman, and A. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[57] T. Lehner, B. Miller, and M. State. *Genomics, Circuits, and Pathways in Clinical Neuropsychiatry*. Academic Press, 2016.

[58] X. Li. A spatial–temporal hopfield neural network approach for super-resolution land cover mapping with multi-temporal different resolution remotely sensed images. *ISPRSjournal of photogrammetry and remote sensing*, 93:76–87, 01 2014.

[59] X. Li, F. Zhao, and Y. Guo. Conditional restricted boltzmann machines for multi-label learning with incomplete labels. In *Artificial Intelligence and Statistics*, pages 635–643, 2015.

[60] H. Lütkepohl. *Vector Autoregressive Models*, pages 1645–1647. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[61] S. Lyu. Interpretation and generalization of score matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 359–366. AUAI Press, 2009.

[62] D. MacKay. Failures of the one-step learning algorithm. In *Available electronically at http://www. inference. phy. cam. ac. uk/mackay/abstracts/gbm. html*, 09 2001.

[63] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[64] B. Marlin, K. Swersky, B. Chen, and N. Freitas. Inductive principles for restricted boltzmann machine learning. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 509–516, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/marlin10a.html`.

[65] J. Melchior, N. Wang, and L. Wiskott. Gaussian-binary restricted boltzmann machines for modeling natural image statistics. *PloS one*, 12(2):e0171015, 2017.

[66] R. Memisevic and G. Hinton. Unsupervised learning of image transformations. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[67] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048. ISCA, 2010.

[68] R. Mittelman, B. Kuipers, S. Savarese, and H. Lee. Structured recurrent temporal restricted boltzmann machines. In *International Conference on Machine Learning*, pages 1647–1655, 2014.

[69] V. Mnih, G. Hinton, and M. Ranzato. Generating more realistic images using gated MRF's. In *Advances in Neural Information Processing Systems*, pages 2002–2010, 2010.

[70] V. Mnih, H. Larochelle, and G. Hinton. Conditional restricted boltzmann machines for structured output prediction. *arXiv preprint arXiv:1202.3748*, 2012.

[71] G. Montavon, K.-R. Müller, and M. Cuturi. Wasserstein training of restricted boltzmann machines. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3718–3726. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6248-wasserstein-training-of-restricted-boltzmann-machines.pdf`.

[72] J. Movellan and J. McClelland. Learning continuous probability distributions with symmetric diffusion networks. *Cognitive Science*, 17(4):463–496, 1993.

[73] J. Movellan, P. Mineiro, and R. Williams. A monte carlo em approach for partially observable diffusion processes: Theory and applications to neural networks. *Neural computation*, 14(7):1507–1544, 2002.

[74] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, pages 807–814, 2010.

[75] T. Nakashika, T. Takiguchi, and Y. Ariki. High-order sequence modeling using speaker-dependent recurrent temporal restricted boltzmann machines for voice conversion. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2278–2282, 01 2014.

[76] T. Nakashika, T. Takiguchi, and Y. Ariki. Voice conversion using rnn pre-trained by recurrent temporal restricted boltzmann machines. *Transactions on Audio, Speech and Language Processing*, 23(3): 580–587, 2015.

[77] R. M. Neal. *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993.

[78] M. Norouzi, M. Ranjbar, and G. Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *Conference on Computer Vision and Pattern Recognition*, pages 2735–2742. IEEE, 2009.

[79] B. Øksendal. Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer, 2003.

[80] J. Paik and A. Katsaggelos. Image restoration using a modified hopfield network. *IEEE Transactions on image processing*, 1(1):49–63, 1992.

[81] J. Park, Y. Kim, I. Eom, and K. Lee. Economic load dispatch for piecewise quadratic cost function using hopfield neural network. *IEEE transactions on power systems*, 8(3):1030–1038, 1993.

[82] J. Parkinson. An essay on the shaking palsy. *The Journal of neuropsychiatry and clinical neurosciences*, 14(2):223–236, 2002.

[83] D. T. Pham and X. Liu. Training of elman networks and dynamic system modelling. *International Journal of Systems Science*, 27(2):221–226, 1996.

[84] W. H. Pinaya, A. Gadelha, O. Doyle, C. Noto, A. Zugman, Q. Cordeiro, A. Jackowski, R. Bressan, and J. Sato. Using deep belief network modelling to characterize differences in brain morphometry in schizophrenia. *Scientific reports*, 6:38897, 2016.

[85] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1): 145–151, 1999.

[86] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[87] M. Ranzato, A. Krizhevsky, and G. Hinton. Factored 3-way restricted boltzmann machines for modeling natural images. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 621–628, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/ranzato10a.html`.

[88] O. Rascol, P. Payoux, F. Ory, J. Ferreira, C. Brefel-Courbon, and J.-L. Montastruc. Limitations of current parkinson's disease therapy. *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society*, 53(S3):S3–S15, 2003.

[89] C. Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[90] B. Rosin, M. Slovik, R. Mitelman, M. Rivlin-Etzion, S. Haber, Z. Israel, E. Vaadia, and H. Bergman. Closed-loop deep brain stimulation is superior in ameliorating parkinsonism. *Neuron*, 72(2):370–384, 2011.

[91] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[92] H. Sailor, D. Agrawal, and H. Patil. Unsupervised filterbank learning using convolutional restricted boltzmann machine for environmental sound classification. In *INTERSPEECH*, pages 3107–3111, 2017.

[93] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL `http://proceedings.mlr.press/v5/salakhutdinov09a.html`.

[94] R. Salakhutdinov and G. Hinton. An efficient learning procedure for deep boltzmann machines. *Neural Computation*, 24(8):1967–2006, 2012.

[95] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010.

[96] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[97] H. Shao, H. Jiang, H. Zhang, W. Duan, T. Liang, and S. Wu. Rolling bearing fault feature learning using improved convolutional deep belief network with compressed sensing. *Mechanical Systems and Signal Processing*, 100:743–765, 2018.

[98] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel distributed processing: Explorations in the microstructure of cognition*, pages 194–281–. MIT Press, Cambridge, MA, 1986.

[99] R. Souriau, V. Vigneron, J. Lerbet, and H. Chen. Probit Latent Variables Estimation for a Gaussian Process Classifier: Application to the Detection of High-Voltage Spindles. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 514–523. Springer, 2018.

[100] R. Souriau, D. Fourer, H. Chen, J. Lerbet, H. Maaref, and V. Vigneron. High-Voltage Spindles detection from EEG signals using recursive synchrosqueezing transform. In *Groupement de Recherche en Traitement du Signal et des Images*, August 2019.

[101] R. Souriau, V. Vigneron, J. Lerbet, and H. Chen. Boltzmann Machines for signals decomposition. Application to Parkinson's Disease control. In *Groupement de Recherche en Traitement du Signal et des Images*, August 2019.

[102] R. Souriau, J. Lerbet, C. Hsin, and V. V. A review on generative boltzmann networks applied to dynamic systems. *Mechanical Systems and Signal Processing*, 147:107072, 2021.

[103] I. Sutskever and G. Hinton. Learning multilevel distributed representations for high-dimensional sequences. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 548–555, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR. URL `http://proceedings.mlr.press/v2/sutskever07a.html`.

[104] I. Sutskever, G. Hinton, and G. Taylor. The recurrent temporal restricted boltzmann machine. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1601–1608. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3567-the-recurrent-temporal-restricted-boltzmann-machine.pdf`.

[105] Y. Teh and G. Hinton. Rate-coded restricted boltzmann machines for face recognition. In *Advances in neural information processing systems*, pages 908–914, 2001.

[106] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.

[107] I. Tosic and P. Frossard. Dictionary learning: What is the right representation for my signal? *IEEE Signal Processing Magazine*, 28(ARTICLE):27–38, 2011.

[108] O. Tysnes and A. Storstein. Epidemiology of parkinson's disease. *Journal of Neural Transmission*, 124(8):901–905, 2017.

[109] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[110] R. Vohra, K. Goel, and J. Sahoo. Modeling temporal dependencies in data using a dbn-lstm. In *International Conference on Data Science and Advanced Analytics*, pages 1–4. IEEE, 2015.

[111] L. Wang. Three-dimensional convolutional restricted boltzmann machine for human behavior recognition from rgb-d video. *EURASIP Journal on Image and Video Processing*, 2018(1):120, Nov 2018. ISSN 1687-5281. doi: 10.1186/s13640-018-0365-8. URL `https://doi.org/10.1186/s13640-018-0365-8`.

[112] S. Wang, J. Xiang, Y. Zhong, and H. Tang. A data indicator-based deep belief networks to detect multiple faults in axial piston pumps. *Mechanical Systems and Signal Processing*, 112:154–170, 2018.

[113] M. Welling, S. Osindero, and G. Hinton. Learning sparse topographic representations with products of student-t distributions. In *Advances in neural information processing systems*, pages 1383–1390, 2003.

[114] C. Williams and F. Agakov. An analysis of contrastive divergence learning in gaussian boltzmann machines. *Institute for Adaptive and Neural Computation*, 2002.

[115] M. Wong, B. Farooq, and G. Bilodeau. Discriminative conditional restricted boltzmann machine for discrete choice and latent variable modelling. *Journal of choice modelling*, 29:152–168, 2018.

[116] Z. Wu, E. Chng, and H. Li. Conditional restricted boltzmann machine for voice conversion. In *China Summit and International Conference on Signal and Information Processing*, pages 104–108. IEEE, 2013.

[117] B. Xiaojun and W. Haibo. Contractive slab and spike convolutional deep boltzmann machine. *Neurocomputing*, 290:208–228, 2018.

[118] L. Younes. Parametric inference for imperfectly observed gibbsian fields. *Probability theory and related fields*, 82(4):625–645, 1989.

[119] A. Yuille. The convergence of contrastive divergences. In *Advances in neural information processing systems*, pages 1593–1600, 2005.

[120] M. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[121] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Apprentissage Automatique pour la Modélisation de Systèmes Stochastiques Dynamiques : Application au Contrôle Adaptatif sur la Stimulation Cérébrale Profonde

Mots clés: Réseaux de neuronaux, Processus de Diffusion, Equation Différentielles Stochastique, Maladie de Parkinson

Résumé: Ces dernières années ont été marquées par l'émergence d'un grand nombre de base données dans de nombreux domaines comme la médecine par exemple. La création de ces bases données a ouvert la voie à de nouvelles applications. Les propriétés des données sont parfois complexes (non linéarité, dynamique, grande dimension ou encore absence d'étiquette) et nécessite des modèles d'apprentissage performants. Parmi les modèles d'apprentissage existant, les réseaux de neurones artificiels ont connu un large succès ces dernières décennies. Le succès de ces modèles repose sur la non linéarité des neurones, l'utilisation de variables latentes et leur grande flexibilité leur permettant de s'adapter à de nombreux problèmes. Les machines de Boltzmann présentées dans cette thèse sont une famille de réseaux de neurones non supervisés. Introduite par Hinton dans les années 80, cette famille de modèle a connu un grand intérêt dans le début du 21e siècle et de nouvelles extensions sont proposées régulièrement.
Cette thèse est découpée en deux parties. Une partie exploratoire sur la famille des machines de Boltzmann et une partie applicative. L'application étudiée est l'apprentissage non supervisé des signaux électroencéphalogramme intracrânien chez les rats Parkinsonien pour le contrôle des symptômes de la maladie de Parkinson. Les machines de Boltzmann ont donné naissance aux réseaux de diffusion. Il s'agit de modèles non supervisés qui reposent sur l'apprentissage d'une équation différentielle stochastique pour des données dynamiques et stochastiques. Ce réseau fait l'objet d'un développement particulier dans cette thèse et un nouvel algorithme d'apprentissage est proposé. Son utilisation est ensuite testée sur des données jouet ainsi que sur des données réelles.

Title: Machine Learning for Modeling Dynamic Stochastic Systems: Application to Adaptive Control on Deep-Brain Stimulation

Keywords: Neural networks, Diffusion Process, Stochastic Differential Equation, Parkinson's Disease

Abstract: The past recent years have been marked by the emergence of a large amount of database in many fields like health. The creation of many databases paves the way to new applications. Properties of data are sometimes complex (non linearity, dynamic, high dimensions) and require to perform machine learning models. Belong existing machine learning models, artificial neural network got a large success since the last decades. The success of these models lies on the non linearity behavior of neurons, the use of latent units and the flexibility of these models to adapt to many different problems. Boltzmann machines presented in this thesis are a family of generative neural networks. Introduced by Hinton in the 80's, this family have got a large interest at the beginning of the 21st century and new extensions are regularly proposed.
This thesis is divided into two parts. A first part exploring Boltzmann machines and their applications. In this thesis the unsupervised learning of intracranial electroencephalogram signals on rats with Parkinson's disease for the control of the symptoms is studied. Boltzmann machines gave birth to Diffusion networks which are also generative model based on the learning of a stochastic differential equation for dynamic and stochastic data. This model is studied again in this thesis and a new training algorithm is proposed. Its use is tested on toy data as well as on real database.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France