IMPACT OF ATTENTION ON PERCEPTION IN COGNITIVE DYNAMIC SYSTEMS

IMPACT OF ATTENTION ON PERCEPTION
IN COGNITIVE DYNAMIC SYSTEMS

By ASHKAN AMIRI, M.SC., M.ENG, and B.SC.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the
Requirements for the Degree Doctor of Philosophy

Contents

1 Introduction **1**
- 1.1 Perception-action Cycle . . . 2
- 1.2 Perceptual Attention . . . 3
- 1.3 Hierarchical Structure . . . 4
- 1.4 Convolutional Feature Extraction . . . 5
- 1.5 Organization of the Thesis . . . 6
- 1.6 Contributions . . . 8

2 Attention: A Fundamental Cognitive Function **9**

3 Bayesian Estimation of Stochastic Processes **13**
- 3.1 Bayes Rule . . . 13
- 3.2 Estimation of a Time-varying Stochastic Process . . . 15
- 3.3 Recursive Estimation Under Linearity and Gaussianity Assumptions 16

4 Improving Performance of a Single-layered Perceptor Under the Influence of Attention **20**
- 4.1 Sparse Coding: General Overview . . . 20
  - 4.1.1 Brief History . . . 20
  - 4.1.2 Application to Problems in Engineering . . . 22
  - 4.1.3 Mathematical Formulation of Sparse Coding . . . 22
  - 4.1.4 Does Regularization Result in Well-posedness? . . . 24
  - 4.1.5 Challenges Faced in Sparse Coding . . . 25
  - 4.1.6 Conditions for Well-posedness of a Sparse Coding Problem . 27
- 4.2 Mitigating Ill-posedness by Introducing Additional Information . . . 28
- 4.3 Algorithmic Implementation of Perceptual Attention . . . 29
  - 4.3.1 How Information Filtering Leads to Attention . . . 32
  - 4.3.2 Cognit: the Name of the New Algorithm . . . 34
- 4.4 Simulations: Part I . . . 34

4.4.1 Objective I: Inferring Original Representations 34
4.4.2 Objective II: Discovering Original Dictionary from Observables 38
4.4.3 Summarizing the Results From Part I 39
4.5 Application to Complex-valued Problems 41
4.5.1 Sparse Coding in the Complex Domain 42
4.5.2 Information Filtering for a Cognit in the Complex Domain . 43
4.6 Simulations: Part II - Applying the Cognit to Real-life Data 45
4.6.1 Description of the Data 45
4.6.2 Objective I: Learning Features from the Observables 47
4.6.3 Objective II: Testing the Stability of Representations 51
4.6.4 Objective III: Classifying the Representations 52
4.7 Summary 53

5 Model Identification Through Expectation Maximization Algorithm 54
5.1 Model Identification 54
5.2 Outline of EM Algorithm 55
5.2.1 Step 1 - Expectation 57
5.2.2 Step 2 - Maximization 58
5.3 Results of EM Algorithm 59

6 Improving Performance of a Hierarchical Perceptor Under the Influence of Attention 61
6.1 Overview of Convolutional Networks 62
6.2 Approach 1: CSCN and the Influence of Perceptual Attention 63
6.2.1 CSCN 63
6.2.2 CSCN - Perceptual Attention 64
6.3 Approach 2: CN and the Influence of Perceptual Attention 64
6.3.1 CN 64
6.3.2 CN - Perceptual Attention 64
6.4 Computer Experiment 65
6.4.1 Preparation of Training and Testing Examples 66
6.4.2 Training Phase 67
6.4.3 Testing Phase 68
6.4.4 Structural Configurations of the Perceptors 68
6.5 Results of the Experiment 69
6.5.1 Learning Rate 69
6.5.2 Target Detection 71

6.5.3 Pseudo Target-tracking 73
6.5.4 Sensitivity to Measurement Noise 73
6.5.5 Summarizing Remarks on the Experiment 75
6.6 Adjusting Sparseness Regularization Parameter 76

7 Concluding Remarks **80**

Appendix A Recursive Estimation Under Linearity and Gaussianity Assumptions - Long Version **83**

Appendix B Algorithmic Steps of EM Algorithm **88**
B.1 Step 1 - Expectation 88
B.2 Step 2 - Maximization 89

Appendix C Wirtinger Calculus for Complex-valued Problems **92**
C.1 Differentiability of a Function in the Complex Domain 93
C.2 Wirtinger Calculus 94
C.3 Optimization of Real-valued Cost Functions 95

Appendix D Brief Overview of Hierarchical Perceptors **97**
D.1 Multilayer Perceptron 97
D.2 Random Decision Forests 99
D.3 Deep Belief Networks 102

Appendix E Preprocessing Radar Data: Imbalance Correction of Amplitude and Phase **105**

Appendix F Radar Target Detection **107**

Bibliography **108**

List of Figures

1.1 Perception-action cycle. 2

2.1 Reciprocal information feedback through local-PAC. 11

3.1 The information-flow diagram for recursive estimation of a stochastic process. 17

4.1 Simple Example. 26
4.2 Block diagram of algorithms: (a) sparse coding. (b) sparse coding under the influence of perceptual attention. 31
4.3 Information-flow diagram of algorithms: (a) sparse coding. (b) sparse coding under the influence of perceptual attention. 32
4.4 A sample observable from a generated sequence. 35
4.5 Simulation results for objective I. 36
4.6 Simulation results for objective I (continued). 37
4.7 Values of the average conformity measures vs. the number of training epochs. 39
4.8 Dictionary elements: (a) original, (b) learned by sparse coding, (c) learned by the cognit. 40
4.9 Amplitudes of the radar returns across the measured range-bins. . . 46
4.10 Amplitude spectra of two observables: (a) target-plus-clutter (b) clutter. 47
4.11 Performing one-step transitions in time to simulate the arrival of new observables for a particular range-bin. 47
4.12 Amplitude spectra of observables from two range-bins: (a) target-plus-clutter (b) clutter. 48
4.15 Sparse representations of successive observables inferred by the cognit: (a) target-plus-clutter (b) clutter. 49

4.13 The dictionary elements learned by the cognit; the solid line is the real part and the dashed line is the imaginary part of the feature. . . 50
4.14 Amplitude spectra of denoised observables for: (a) target-plus-clutter (b) clutter. 51
4.16 The MSD of representations for the cognit and the sparse coding algorithm: (a) target-plus-clutter (b) clutter. 52

5.1 (Left) Trace of the estimated covariance matrix for measurement noise, denoted by $\hat{R}$, computed over successive iterations of EM algorithm. (Right) Trace of the estimated covariance matrix for process noise, denoted by $\hat{Q}$, computed over successive iterations of EM algorithm. 59
5.2 Absolute values of covariance matrices: (Left) Estimated covariance matrix for the measurement noise, $\hat{R}$. (Right) Estimated covariance matrix for the process noise, $\hat{Q}$. 60
5.3 (Left) Raw signal received from radar. (Right) Filtered signal under the influence of perceptual attention. 60

6.1 A convolutional network with two convolutional layers, two subsampling layers, and a classification layer, aimed at processing 2-dimensional input data. 62
6.2 Selecting samples from 5 neighbouring range-bins and with a frame-size of $N = 64$ samples. Time index and bin index are selected randomly. Range-bins are selected in a way that presence of the target in all selected range-bins is uniformly distributed over training examples. 66
6.3 Successive transitions of input frames over recorded samples of radar returns. 67
6.4 M^{th} frame of samples, according to which classification errors are computed in Step 2 of the experiment. 68
6.5 Mean squared error of classifiers for CN and CSCN. 69
6.6 Track of the target estimated over 50 successive frames of samples. . 73
6.7 Effect of perceptual attention on performance of CSCN in the presence of noise. 74
6.8 Effect of perceptual attention on performance of CN in the presence of noise. 75
6.9 Mean activity ratios for the 1st and 2nd convolutional layers of CSCN, generated using 100 randomly selected inputs. 77

6.10 Mean activity ratios for the 1st and 2nd convolutional layers of CSCN, their corresponding fits. 78
6.11 Evolution of estimated λ values over consecutive estimation cycles. . 79

D.1 Structure of a perceptron. 97
D.2 Structure of a multilayer perceptron. 100
D.3 Structure of a simple decision tree. 101
D.4 Network of a RBM. 103

List of Tables

4.1 The classification results using SVM-RBF. 53

6.1 Approach 1: Results on target detection. 71
6.2 Approach 2: Results on target detection. 71
6.3 Results for a classic approach. 72

Dedications

To my parents, ***Asad*** *and* ***Monir****, to whom I owe everything that I have accomplished in my life.*

To my sisters and brother, ***Azadeh****,* ***Anahid****, and* ***Ardavan****, for their unconditional love and support.*

And to my wife, ***Marjan****, who fills my heart with love and happiness and makes my life colourful.*

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Dr. Simon Haykin for supporting me selflessly in every way possible; it was both an honour and a delight to work with him and to learn from him. I would also like to thank Dr. Bartosz Protas, Dr. Sue Becker, and Dr. Jim Reilly for all the priceless advice they gave me over the past four years. I could never stop learning from them even if I wanted to. Last but by no means least, I am indebted to Dr. Terrence J. Sejnowski for the assessment of my dissertation and for his extremely kind words about my work.

Abstract

The proposed aim of this thesis, inspired by the human brain, is to improve on the performance of a perceptual processing algorithm, referred to as a "perceptor". This is done by trying to bridge the gap between neuroscience and engineering. To this end, we build on localized perception-action cycle in cognitive neuroscience by categorizing it under the umbrella of perceptual attention, which lends itself to increase gradually the contrast between relevant information and irrelevant information. Stated in another way, irrelevant information is filtered away while relevant information about the environment is enhanced from one cycle to the next. Accordingly, we propose to improve on the performance of a perceptor by modifying it to operate under the influence of perceptual attention. For this purpose, we first start with a single-layered perceptor and investigate the impact of perceptual attention on its performance through two computer experiments: The first experiment uses simulated (real-valued) data that are generated to purposely make the problem challenging. The second experiment uses real-life radar data that are complex-valued, hence the proposal to introduce Wirtinger calculus into derivation of our proposed method. We then take one step further and extend our proposed method to the case where a perceptor is hierarchical. In this context, every constitutive component of a hierarchical perceptor is modified to operate under the influence of perceptual attention. Then, another experiment is carried out to demonstrate the positive impact of perceptual attention on the performance of that hierarchical perceptor, just described.

Chapter 1

Introduction

The literature on cognitive neuroscience has been the source of inspiration for many machine learning algorithms especially in the context of perceptual problems. This extent of attention to neuroscience seems justifiable, since there exists yet no algorithm that can perform in a way that is even close to the perceptual processing power of human brain. Whether we should follow neuroscience closely in order to design better algorithms for perception, or not, is debatable. What is clear, however, is that if a cognitive system can be implemented biologically through evolution, then it can also be implemented through engineering; when or how is yet unknown, but the feasibility of it is undeniable.

Our work focuses on the same area of research, i.e., designing algorithms for the perceptual processing of data with the ultimate goal of being used in a cognitive dynamic system (Haykin, 2012). From the very beginning, we have committed ourselves to learn from the brain by studying the literature on cognitive neuroscience. Building on ideas inspired by the literature, we have then tried to take a step toward bridging the gap between neuroscience and engineering. A description of our efforts and the results we have achieved are presented in this thesis. Throughout the thesis, we simply refer to a perceptual processing algorithm as a "**perceptor**."

The work presented herein builds on four main pillars, namely,

i) **Perception-action cycle**, which is a well-known principle in cognitive neuroscience and engineering.

ii) **Perceptual attention** for an improved processing of sensory information, the functionality of which in our work is sustained by the operation of successive perception-action cycles.

iii) **Hierarchical structure** for a layer-by-layer processing of sensory information.

iv) **Convolutional feature extraction**, which has shown to be an effective method for solving perceptual problems.

The choice of these four pillars is motivated by neurobiological evidence, details of which are provided in what follows:

1.1 Perception-action Cycle

One of the important principles in cognitive neuroscience is the perception-action cycle (PAC) (Fuster, 1989), which has been given different names by different authors (von Weizsäcker, 1950; Neisser, 1976; Arbib, 1981). Through PAC, an organism perceives the environment and acts accordingly. Then, by observing the consequences of its actions on the environment, it can learn and adapt itself iteratively from one cycle to the next. Hence, a *global* feedback loop is formed that embodies both the organism and the environment; this loop allows the organism to become more fit for survival through learning from continuous interactions with the environment.

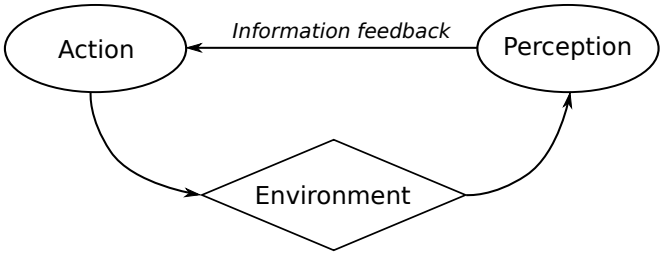


Figure 1.1: Perception-action cycle.

PAC is a known principle in engineering as well, going back to the pioneering work of Wiener (1948) on cybernetics, an area which has been explored thoroughly in the course of decades. In addition, PAC is a core feature in the more recent paradigm of cognitive dynamic systems (Haykin, 2012). In this regard, Haykin, Xue, and Setoodeh (2012) showed how by building on PAC, a radar system can improve its performance in tracking a target.

It is noteworthy that PAC could be of a global kind by embodying the environment, or it could be of a *local* kind within the perceptual or executive part of a

cognitive dynamic system. In the case of local-PAC, actions may be in the form of regulatory signals that influence the internal parts of the system, thereby performing internal control.

1.2 Perceptual Attention

A plausible theory involving local-PAC in the human brain is feedback signals from the frontal cortex that reenter sensory cortices to improve sensory analysis (Fuster, 2008). Consequently, the competition among affected neurons will be biased in favour of those neurons that represent the *relevant* part of sensory information that is to be attended. In direct contrast, the activity of other neurons that represent the *irrelevant* part of the sensory information will be suppressed (Desimone and Duncan, 1995; Miller and Cohen, 2001; Posner, 2011). We may therefore say that local-PAC facilitates selective processing of sensory information.

According to Neisser (1976), selective processing of sensory information is realized through interactions of two types of processes:

a) *preattentive processes* that segregate sensory information through rapid parallel processing, followed by

b) *focal attention* that leads to a slower and more detailed analysis.

Similar descriptions for such interactions have been provided by other authors using the interplay of bottom-up information processing and top-down control (Fuster, 2003; Posner, 2011).

In light of the descriptions provided in (Neisser, 1976; Fuster, 2003; Posner, 2011), local-PAC can be categorized under the umbrella of *attention*. In our research, we concern ourselves with perception of the environment. Thus, we may go on and say it in a more specific manner that local-PAC is part and parcel of *perceptual attention*, and thus propose how to improve the performance of a perceptor under the influence of perceptual attention.

In the evolution of a perceptual task, perceptual attention gradually increases the contrast between relevant information and irrelevant information; that is, in a cyclic manner, irrelevant information is gradually filtered out, while more relevant information is gained to perceive the environment better. Building on this idea, in (Amiri and Haykin, 2014), we proposed how to improve sparse coding under the influence of perceptual attention using an information filter. It is noteworthy that in information filtering, first introduced by Fraser (1967), iterations from one cycle

to the next result in the algorithmic emergence of a feedback loop, which takes us back to local-PAC.

The information filter, rooted in *Fisher information*, is a special class of filtering derived from the Bayesian paradigm (Bar-Shalom, Li, and Kirubarajan, 2001), meaning that in our approach, perceptual attention is introduced in a Bayesian context. The Bayesian paradigm has been previously used in several studies for modelling attention (Lee and Mumford, 2003; Rao, 2005; Torralba, Oliva, Castelhano, and Henderson, 2006; Chikkerur, Serre, Tan, and Poggio, 2010). In addition, various studies have provided results on how Bayesian principles can be used to explain perception, inference, and decision making in the brain (Knill and Richards, 1996; Rao, Olshausen, and Lewicki, 2002; Körding and Wolpert, 2006). In this regard, our motivation for using the Bayesian approach for perceptual attention is very much in line with those studies just mentioned.

1.3 Hierarchical Structure

An important finding about the brain is the hierarchical structure of the cerebral cortex, the influence of which on the field of machine learning goes back to the advent of multilayer perceptrons (MLPs) (Rosenblatt, 1961). In the context of hierarchy, a popular theory in cognitive neuroscience is that memories that are more concrete, explicit, and shorter in length, are stored at lower stages of the hierarchical structure; on the other hand, memories that are more generalized, abstract, and longer in length are stored at upper stages of the structure (Fuster, 2003).

The way in which sensory information is processed in such a hierarchy is generally described as follows (Fuster, 2003; Granger, 2006): The sensory information is first processed in primary sensory-processing areas of the cortex, as a result of which primitive features and hidden structures within the sensory information are extracted. Then, in the next stage, relative relationships of those features are analyzed. Consequently, sequences of features are assembled that represent broader and more general categories of knowledge compared to the features from the previous stage. This process is then repeated, stage after stage, to form longer sequences of features.

In machine learning, hierarchical processing of information has received a great deal of attention, serving as the foundation of many methods, including, MLPs, convolutional nets, deep learning, and decision forests; details of these methods are sufficiently described in Chapter 6 and Appendix D.

A justification for the popularity of hierarchical approaches may be expressed

as follows:

> *Shallow* structures can be very inefficient for solving complex problems in terms of the number of computational units they require (Håstad and Goldmann, 1991). Whereas, the same problems can be handled by *deep* structures with fewer computational units (Bengio, Lamblin, Popovici, and Larochelle, 2007), and therefore using a smaller set of training examples (Bengio and LeCun, 2007).

As a result, deep structures that process data in a hierarchical manner have been the framework of several algorithmic procedures (LeCun and Bengio, 1995; Lee, Ekanadham, and Ng, 2007; Bengio, 2009; Hinton, 2010). In the same way, hierarchy serves as the backbone of our approach.

1.4 Convolutional Feature Extraction

Vision is an extremely challenging perceptual problem. Nevertheless, the human brain has an astonishing ability to accomplish visual perception in a manner that seems effortless, which from early on, has made the visual cortex a very engaging topic in the literature on neuroscience. In particular, studies on the visual cortex entered a new phase after the analysis of evoked potentials of neurons gained an increasing popularity in studies of functional anatomy in animals like the rabbit, cat, and monkey (Talbot and Marshall, 1941; Thompson, Woolsey, and Talbot, 1950; Mountcastle, 1957; Powell and Mountcastle, 1959).

Applying a similar technique, Hubel and Wiesel (1962) investigated receptive fields[1] of neurons and their functional architecture in the cat's visual cortex. This revealing study resulted in discovery of neurons with local receptive fields and selective responses; these neurons could only be evoked when strict requirements for shape, direction, and location of stimulus on the retina were satisfied altogether.[2]

In the context of artificial neural networks, connecting neurons to local receptive fields that are considerably smaller than the whole span of input has an important

[1] According to Hubel and Wiesel (1962): "The receptive field of a cell in the visual system may be defined as the region of the retina (or visual field) over which one can influence the firing of that cell."

[2] Hubel and Wiesel won the "Nobel Prize in Physiology or Medicine" in 1981 for their valuable discoveries in the 1960s and 1970s concerning the visual cortex.

outcome. To elaborate, local receptive fields allow us to investigate local dependencies in input data in a more efficient manner, whereby local features may be extracted from the data; in the context of visual data, these local features include edges, corners, and endpoints.

It is noteworthy that primitive features, just described, may be present in any part of the input. Thus, primitive feature detectors of one receptive field are very likely to be useful for detecting primitive features at other local receptive fields as well. Consequently, it is possible to use a fixed set of feature detectors across different parts of the input in order to produce *maps* of detected features (Rumelhart, Hinton, and Williams, 1986). Using the same technique, LeCun and Bengio (1995) proposed a convolutional method for feature extraction by sliding a set of feature detectors with small receptive fields across the span of input. Accordingly, in the first layer, maps of primitive features are generated. Then, these maps serve as inputs to the layer above, where more complex features are extracted (i.e., *features of features*).

Convolutional feature extraction has been used in many studies and has shown to be a very effective approach for solving perceptual problems (Ranzato, Huang, Boureau, and LeCun, 2007; Jarrett, Kavukcuoglu, Ranzato, and LeCun, 2009; Lee, Grosse, Ranganath, and Ng, 2009; Taylor, Fergus, LeCun, and Bregler, 2010; Cireşan, Meier, Masci, Gambardella, and Schmidhuber, 2011; Krizhevsky, Sutskever, and Hinton, 2012), hence explaining our motivation for choosing the convolutional approach.

1.5 Organization of the Thesis

To improve exposition of this thesis, contents are presented in two portions, namely,

i) thesis *chapters*, which describe general ideas and motivations behind this thesis as well as procedures that are carried out, and,

ii) *appendices*, which provide details of engineering tools, methods, and background material that are important for the accomplishment of this thesis.

Accordingly, thesis chapters are organized as follows:

- **Chapter 2** lends itself to *attention*, a cognitive function that is fundamental to cognition. In this chapter, relevant literature on this topic are presented. Furthermore, motivations for choosing a Bayesian paradigm as the mathematical tool for mimicking perceptual attention are explained in this chapter.

- In **Chapter 3**, fundamentals of the Bayesian paradigm, which is basic to our algorithmic implementation of perceptual attention, are explained. Furthermore, the way in which the Bayesian paradigm can be applied for the estimation of a stochastic process of interest is explained thoroughly.

- **Chapter 4** attempts to improve the performance of a single-layered perceptor under the influence of perceptual attention. The perceptor used in this chapter is based on sparse coding algorithm (Olshausen and Field, 1996), where the algorithmic implementation of perceptual attention is carried out in a Bayesian context through the use of an information filter. To support the material of this chapter, two computer experiments are presented. The first experiment uses simulated (real-valued) data that are generated to purposely make the problem challenging. The second experiment uses real-life radar data that are complex-valued, hence the proposal to introduce Wirtinger calculus into derivation of our new algorithm.

- **Chapter 5** concerns itself with the problem of model identification for the dynamics of a stochastic process, which is essential to our algorithmic implementation of perceptual attention. In this regard, *expectation maximization* approach is adopted for the identification of model parameters, and the outline of this approach is elaborately described.

- In **Chapter 6**, we concern ourselves with hierarchical perceptors and investigate how we may improve on the performance of a hierarchical perceptor under the influence of perceptual attention. In this regard, performances of two different hierarchical perceptors are evaluated in a computer experiment. The first perceptor is a convolutional hierarchy that builds on sparse coding, and the second one is a convolutional hierarchy composed of perceptrons. Here again, the experiment is carried out using real-life radar data, except with a more challenging set-up for the experiment compared to the one chosen for the experiment in Chapter 4. Then, in light of descriptions provided in Chapter 4, the two perceptors are modified to function under the influence of perceptual attention and their performances are re-evaluated accordingly. Thereby, the impact of perceptual attention on the performance of each one of the two perceptors is examined.

  In addition, performances of the two perceptors are compared with a classic approach, which is based on radar target detection using Doppler spectrum amplitude; it also involves the use of a particle filter.

- Finally, concluding remarks are provided in **Chapter 7**.

Furthermore, appendices of this thesis are structured in the following manner:

- **Appendix A** describes a complete procedure for the derivation of an optimal Bayesian filter under linearity and Gaussianity assumptions; this appendix is provided as a supplement to Chapter 3.
- **Appendix B** explains the derivation of algorithmic steps for the expectation maximization algorithm, which relates data to model; this appendix builds on the material presented in Chapter 5.
- In **Appendix C** Wirtinger calculus is sufficiently explained.
- **Appendix D** provides an overview of hierarchical perceptors.
- In relation to radar data, which are used in the experiments of this thesis, background material are presented in **Appendices E** and **F** on the two topics of "preprocessing radar data" and "radar target detection," respectively.

Due to the demanding nature of computations in this thesis, a parallel programming approach was taken and computer experiments were carried out on SHARCNET cluster.

1.6 Contributions

Contributions of this thesis may be summarized as follows:

i) Proposing an algorithm for improved sparse coding under the influence of perceptual attention.

ii) Introducing Wirtinger calculus into derivation of sparse coding algorithm for the first time, hence, simplifying the application of this algorithm to complex-valued problems.

iii) Extending the use of the algorithm, described in i), to hierarchical perceptors, thereby improving on performances of those perceptors.

iv) Illustrating the impact of perceptual attention on reducing the sensitivity of a perceptor in the face of uncertainties.

v) Proposing a method to estimate a proper set of regularization parameters for ℓ^1-regularized optimization within a multilayer structure.

Chapter 2

Attention: A Fundamental Cognitive Function

The literature of neuroscience has been dominated for a long time by the findings of *localizationists*; this way of thinking presents a modular view of the cortex with different cognitive functions being localized at different cortical regions (Gall, 1825; Broca, 1861; Brodmann, 1905). In the more recent years, however, there has been a greater tendency to explain the cognitive functions with respect to a network model of distributed cortical systems (Lashley, 1950; Hayek, 1952; Edelman and Mountcastle, 1978; Rumelhart and McClelland, 1986). According to this model, specialized areas and modules for sensory and motion are placed at lower hierarchical stages of neural processing; furthermore, cognitive functions exist on vast territories of the cortex while including those specialized areas and modules (Fuster, 1997, 2003).

In the *holistic* view, just described, cognitive functions can engage different parts of the cortex according to their degree of association with a cognitive task of interest (Eacott and Gaffan, 1992; Tomita, Ohbayashi, Nakahara, Hasegawa, and Miyashita, 1999; Ranganath, Cohen, Dam, and D'Esposito, 2004). Hence, under the command of cognitive functions, activities of cortical circuits are moderated and the flow of information among those circuits is controlled (Fuster, 1997, 2003).

Neural circuits are highly interconnected through *feedforward* (bottom-up), *feedback* (top-down), and *lateral* synaptic connections. In a network with a highly complex topographical structure such as that of the cortex, local activity of a small group of neurons may spread to other neurons that are in immediate contact with that small group. Then, through sequential activation of other neighbouring groups, an avalanche of activities may occur, causing the network to burst into an explosion

of firings of neurons. Such an outburst is certainly undesirable, since it paralyzes the network and destroys its ability to work in a purposeful manner.

To support successful operation of the network, it is therefore necessary to manage activities of neurons in both *exclusive* and *inclusive* manners. That is to say, that part of neurons, the activities of which are irrelevant to a cognitive task of interest, should be suppressed. On the other hand, activities of those neurons that are relevant to the task of interest, should be facilitated, as long as they serve the accomplishment of that task. The process just described is the basic role of *attention* in cognition, which makes it possible to selectively allocate neural resources for the processing of information.

Exclusionary and inclusive mechanisms of attention are carried out through the inhibition and excitation of neurons, respectively, the end result of which is a "biased competition" in the activity of neurons (Desimone and Duncan, 1995). In this regard, it is presumed that prefrontal cortex has an essential role in exerting attentional control signals, in a top-down manner, on other information processing parts of the brain (Fuster, 2003; Posner, 2011). A prevalent view is that these control signals, which are sent as feedback to lower stages of the cortex, facilitate the processing of sensory and motor information at those lower stages; for instance, feedback signals seem to mitigate ambiguities in sensory information (Mumford, 1994; Knill and Richards, 1996), or in a similar manner, they appear to improve the performance of a motor action on the environment (Fuster, 2014).[1]

It is remarkable that feedback signals of attention play an important role for an organism to adapt to its environment; this statement might not seem very surprising, since "adaptation" and "feedback" have a long history of coming along together. In this context, we may refer again to PAC (Figure 1.1), which is the simplest form of adaptation through feedback. In primitive organisms, such as sea anemone (Uexküll, 1926), it is presumed that feedback comes only from the environment, or in other words, PAC is global. Thus, actions that are performed by the organism on the environment may be considered as immediate responses to sensory information.

In higher animals, however, the nervous system is sufficiently complex, hence

[1] It is important to note here that attentional control signals should not necessarily come from the top; in fact, it is also possible that lower structures apply those control signals locally (Fuster, 2014). A plausible example of this case is the simple "winner-take-all" mechanism, where a strongly activated neuron wins its way to the top of the competition after suppressing its neighbours via lateral inhibition (Coultrip, Granger, and Lynch, 1992).

allowing the presence of numerous internal feedback loops. As illustrated in Figure 2.1, internal feedback loops make it possible for different parts of the system to exchange information and perform internal actions. These internal actions may in turn provide the opportunity of exhibiting more sophisticated adaptive behaviours, compared to the case of a global PAC.

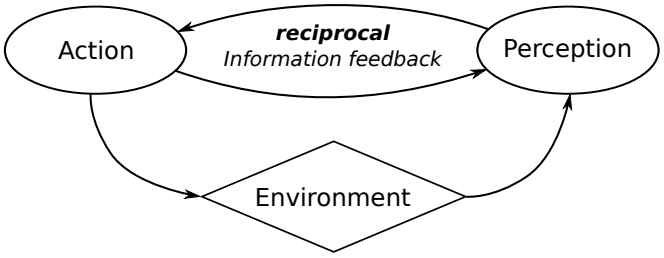


Figure 2.1: Reciprocal information feedback through local-PAC.

It is important to note that attention builds on *memory*, which is learned and updated continuously through observations and interactions with the environment. In higher animals, memory is capable of capturing advanced relations that exist in the environment. For instance, relations between a sequence of events that occur separately over time may be learned through association. By learning those relations, an animal may then gain a better understanding about the ways in which the environment changes. Thus, knowing about these changes in prior, it may well improve its chances of survival through preadaptation; this proactive behaviour is considered to be an important outcome of the interplay of memory and attention (Bar, 2004; Fuster, 2014).[2]

Looking up to the fundamental role of attention in adaptive behaviour, many studies have tried to propose methods for mimicking the functionality of attention. On this very topic, probabilistic approaches are highly popular, where *latent* processes that we wish to uncover by processing sensory information may be described as stochastic processes. Consequently, we may apply solid mathematical foundations of probability theory to estimate a model for those processes in a robust and reliable manner (Rao et al., 2002). In this context, Bayesian paradigm has been proposed by many studies to be an appropriate framework for explaining the role of top-down influences on perception, reasoning, and decision making (Knill and Richards, 1996; Friston, 2003; Körding and Wolpert, 2006; Doya, 2007; Chikkerur

[2]In a similar manner, Haykin and Fuster (2014) describe pre-adaptation as an indispensable feature of a cognitive dynamic system (Haykin, 2012).

et al., 2010).

Moreover, the Bayesian paradigm has been repeatedly adopted to model interactive cortical computations in visual perception. For example, in a Bayesian context, Rao and Ballard (1999) describe a model of visual processing in which feedback connections from a higher stage of visual processing in the cortex to a lower stage carry predictions of lower stage neural activities. Likewise, Lee and Mumford (2003) use the Bayesian paradigm to integrate top-down contextual priors and bottom-up observations to model concurrent probabilistic inference along the visual hierarchy. In a similar context, George and Hawkins (2005) propose a Bayesian model to explain invariant visual pattern recognition in the visual cortex, which builds on temporal coherence criterion to perceive object concepts and movement patterns. Last but not least, the Bayesian paradigm has also been applied in different studies for improving algorithmic processing of visual information (Hinton, Dayan, Frey, and Neal, 1995; Lewicki and Sejnowski, 1997; Fergus, Perona, and Zisserman, 2003; Karklin and Lewicki, 2005).

In this thesis, we share the same motivations with the studies, just described, and we plan to mimic the top-down influence of attention using the Bayesian paradigm. Accordingly, the next chapter lends itself completely to explaining the Bayesian paradigm and the way in which it may be used to estimate a stochastic process of interest.

Chapter 3

Bayesian Estimation of Stochastic Processes

In previous chapter, we described our motivation for the use of probabilistic approaches, through which, latent processes that lie behind sensory information may be thought of as stochastic processes. We further explained that the way of thinking, just described, allows us to apply tools from probability theory in order to model those latent processes. This chapter is therefore devoted to Bayesian paradigm, which is our probabilistic method of choice for algorithmic implementation of perceptual attention.

Accordingly, fundamentals of the Bayesian paradigm are explained and the way in which it may be applied for the estimation of a stochastic process of interest is described. The material presented in this chapter are to be used later in Chapter 4, where algorithmic implementation of perceptual attention is presented.

3.1 Bayes Rule

Bayes rule is a significantly important tool in probability theory, which is used for mathematical manipulation of a conditional probability density function (pdf). Having two stochastic processes, y and z, the pdf of the process z given y, denoted by $p(z|y)$, may be defined using the Bayes rule as follows:

$$p(z|y) = \frac{p(y|z)p(z)}{p(y)}, \tag{3.1}$$

where $p(z|y)$ is described as the *posterior* distribution of z, $p(z)$ is the $prior$, and $p(y|z)$ is known as the *likelihood* function. $p(y)$ is described as the *evidence* and it

can be derived through the following marginal integral:

$$p(y) = \int_{z \in \mathbb{R}^m} p(y, z) \mathrm{d}z, \tag{3.2}$$

with $p(y, z)$ being the *joint* pdf of y and z. Since y is given, $p(y)$ simply plays the role of a *normalizing factor*, which scales the numerator of equation (3.1) in order for the right-hand side of the equation to be a valid pdf. Hence, it is also correct to express (3.1) as follows:

$$p(z|y) \propto p(y|z)p(z). \tag{3.3}$$

Derivation of Bayes rule is straightforward. By definition, joint pdf of y and z may be written as follows:

$$p(y, z) = p(y|z)p(z), \tag{3.4}$$

and in a similar manner,

$$p(y, z) = p(z|y)p(y). \tag{3.5}$$

Thus, setting equation (3.4) equal to (3.5), we may write

$$p(z|y)p(y) = p(y|z)p(z), \tag{3.6}$$

which obviously leads to the formula provided in (3.1).

Accordingly, the Bayes rule provides a basis to estimate the pdf of a process of interest z that is not directly measured from the environment, based on a model that relates an observable process y to z. This model is often described as the *measurement equation* and it has the following general form:

$$y = h(z, \omega), \tag{3.7}$$

where $h(\cdot)$ is the measurement function and ω is a stochastic process that represents imperfections that may exist in the model. The hidden stochastic process z is also known as the *state variable*.

In practical applications, it is common to represent the general form in (3.7) using a simpler expression,

$$y = h(z) + \omega, \tag{3.8}$$

where uncertainties are modelled as an additive stochastic process ω, also known as the *measurement noise*. This model may also take a linear form

$$y = Hz + \omega, \tag{3.9}$$

which greatly simplifies matters for the estimation of z based on the observation y. It is noteworthy that by defining ω as the term responsible for uncertainties, y may be simply treated as a given observation and not a stochastic process.

3.2 Estimation of a Time-varying Stochastic Process

Assuming we have a stream of observations discretely acquired at equal time-steps, we may rewrite (3.7) to include time in the equation

$$y_k = h(z_k, \omega_k), \tag{3.10}$$

where k is the index for k^{th} time-step. To proceed, we assume that the measurement noise sequence ω_k is white. In addition, we assume that z is a Markov process, i.e., future evolution of its state can be fully described by the information that is available in its current state (Haykin, 2001).

Evolution of the process z is described by the *process equation*, the general form of which is expressed as follows:

$$z_{k+1} = f(z_k, \gamma_k), \tag{3.11}$$

where $f(\cdot)$ is the *transition function* and γ is the *process noise*. It is noteworthy that γ and ω are assumed to be mutually independent. The two equations (3.10) and (3.11) collectively form the *state-space model*, which describes dynamics of a stochastic process and the way in which it is related to observations over time.

The estimation of a time-varying stochastic process may therefore be carried out using Bayesian paradigm in two steps, namely, *prediction*, and *innovation*, which repeat recursively over successive time-steps. Figure 3.1 displays the information-flow diagram of this recursive process.

Step 1 - Prediction

In the prediction step, we start with the assumption that the posterior distribution of the process is available at time-step k; to simplify notation, we denote the posterior by $p_k(z)$, which is defined by

$$p_k(z) \triangleq p(z_k|Y_k), \tag{3.12}$$

where Y_k denotes the sequence of *observables* available up to time-step k, i.e.,

$$Y_k \triangleq \{y_i, i \leq k\}. \tag{3.13}$$

Then, the posterior $p_k(z)$ is used for predicting the distribution of the process for upcoming time-step, $k+1$. For this purpose, the *predicted* distribution, denoted by

$p_{k+1|k}(z)$ is calculated with respect to the process equation of the state-space model described in equation (3.11) using the *total probability theorem*:

$$p_{k+1|k}(z) \triangleq p(z_{k+1}|Y_k) \tag{3.14}$$

$$= \int_{z_k \in \mathbb{R}^m} p(z_{k+1}|z_k, Y_k) p(z_k|Y_k) \mathrm{d}z_k \tag{3.15}$$

$$= \int_{z_k \in \mathbb{R}^m} p(z_{k+1}|z_k) p(z_k|Y_k) \mathrm{d}z_k. \tag{3.16}$$

Equation (3.16) is known as the *Chapman-Kolmogorov equation* (Bar-Shalom et al., 2001).

Step 2 - Innovation

In the innovation step, a new observable is acquired. This new observable y_{k+1} is then used to form the likelihood function of the process, $p(y_{k+1}|z_{k+1})$, based on the measurement equation described in equation (3.10). As for the prior distribution of the process, the pdf $p_{k+1|k}(z)$, which is available from Step 1, is used. Accordingly, the likelihood function is conditioned by the prior using Bayes rule and the posterior distribution $p_{k+1}(z)$ is derived. This may be equivalently written as follows:

$$p_{k+1}(z) \triangleq p(z_{k+1}|Y_{k+1}) \tag{3.17}$$

$$= p(z_{k+1}|Y_k, y_{k+1}) \tag{3.18}$$

$$= \frac{1}{c} p(y_{k+1}|z_{k+1}, Y_k) p(z_{k+1}|Y_k) \tag{3.19}$$

$$= \frac{1}{c} p(y_{k+1}|z_{k+1}) p_{k+1|k}(z), \tag{3.20}$$

where c is a normalizing factor.

3.3 Recursive Estimation Under Linearity and Gaussianity Assumptions

For a linear state-space model, defined by

$$\begin{cases} z_{k+1} &= F z_k + \gamma_k \\ y_k &= H z_k + \omega_k \end{cases}, \tag{3.21}$$

and white zero-mean Gaussian distributions for ω and γ, the integral in Chapman-Kolmogorov equation, described in (3.16), may be solved in closed form for two reasons:

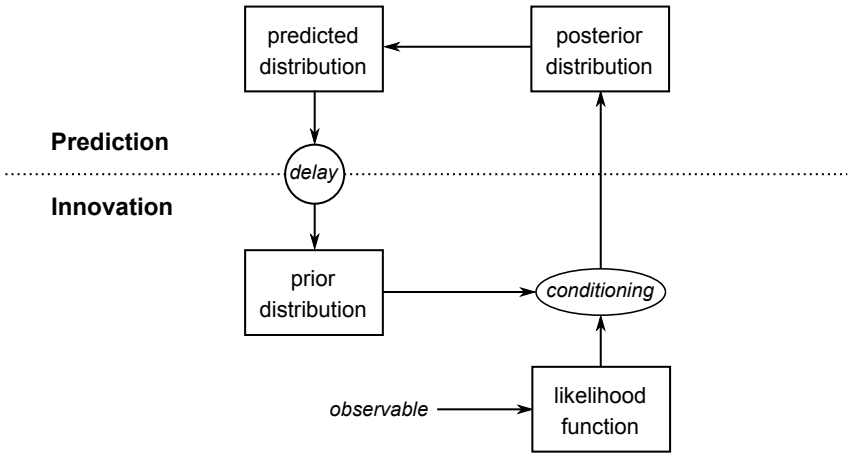


Figure 3.1: The information-flow diagram for recursive estimation of a stochastic process.

i) A closed-form solution may be derived for equation (3.16) when both $p(z_{k+1}|z_k)$ and $p(z_k|Y_k)$ are Gaussian.

ii) Gaussianity of probability distributions remains unaffected by linear operations that are performed over successive time-steps, thus maintaining the point raised in i).

Kalman Filter

A Gaussian process is sufficiently described using first- and second-order moments. Hence, to update probability distributions from one time-step to the next, it is only necessary to derive the formulas required to update their first- and second-order moments. On this very topic, Bar-Shalom et al. (2001) describes a simple procedure to derive those formulas (a lengthier version of this procedure is provided in Appendix A of this thesis). The end result that is achieved is the famous Kalman filter (Kalman, 1960), the steps of which are described in Algorithm 1.

In Algorithm 1, μ_k and $\mu_{k+1|k}$ are expected values, respectively defined by,

$$\mu_k = \mathbb{E}[z_k|Y_k], \text{ and } \mu_{k+1|k} = \mathbb{E}[z_{k+1}|Y_k],$$

and Σ_k and $\Sigma_{k+1|k}$ are covariance matrices, defined in a similar manner by,

$$\Sigma_k = \text{Cov}[z_k|Y_k], \text{ and } \Sigma_{k+1|k} = \text{Cov}[z_{k+1}|Y_k].$$

In addition, R is the covariance matrix of ω, Q is the covariance matrix of γ, and S and G are *residual covariance* and *filter gain*, respectively.

Algorithm 1 Recursive Estimation of a Stochastic Process Using Kalman Filter

set μ_0 and Σ_0 to proper initial values.
Repeat for $k = 1, 2, 3, ...$

prediction:

$\mu_{k+1|k} = F\mu_k$
$\Sigma_{k+1|k} = F\Sigma_k F^{\mathrm{T}} + Q$

innovation:

$S = R + H\Sigma_{k|k-1}H^{\mathrm{T}}$
$G_k = \Sigma_{k|k-1}H^{\mathrm{T}}S^{-1}$
$\mu_k = \mu_{k|k-1} + G_k(y_k - H\mu_{k|k-1})$
$\Sigma_k = \Sigma_{k|k-1} - G_k S G_k^{\mathrm{T}}$

Information Filter: An Alternative Derivation

In recursive estimation of a stochastic process, it is possible to use an alternative approach, where the algorithmic steps are derived with respect to the inverse of the covariance matrix, also known as the *information matrix*[1], and the *information vector* ψ, defined by:

$$\psi = \Sigma^{-1}\mu \tag{3.22}$$

This leads to a dual approach for Kalman filter, known as information filter. The correspondence between key elements in the formulation of the Kalman filter and the information filter can be summarized as follows (Haykin, 2013):

Kalman filter		**Information filter**
z : state variable	$\longleftrightarrow$	ψ : information vector
Σ : covariance matrix	$\longleftrightarrow$	Σ^{-1} : information matrix

Instead of passing the covariance matrix and expected value of a stochastic process from one cycle to the next, the information filter works with the information

[1] Under linearity and Gaussianity assumptions, information matrix equals the Fisher information matrix (Van Trees, 1968).

matrix and information vector. Consequently, the notion of state variable is no longer observed in the information filtering algorithm, and *information propagation* becomes the main issue of interest.

In addition, the information filter has an advantage over the Kalman filter; it allows us to start information filtering without an initial estimate using a *diffuse* prior (i.e., zero information). Thus, at the beginning, Σ^{-1} and ψ can be simply set to zero. Algorithm 2 describes the algorithmic steps of an information filter.

Algorithm 2 Recursive Estimation Using Information Filter

$\Sigma_0^{-1} \leftarrow 0$
$\psi_0 \leftarrow 0$
Repeat for $k = 1, 2, 3, ...$

prediction:

$B = F^{-\mathrm{T}}\Sigma_k^{-1}F^{-1}$
$A_k = F^{-\mathrm{T}} - B(B + Q^{-1})^{-1}F^{-\mathrm{T}}$
$\Sigma_{k+1|k}^{-1} = A_k\Sigma_k^{-1}F^{-1}$
$\psi_{k+1|k} = A_k\psi_k$

innovation:

$\Sigma_k^{-1} = \Sigma_{k|k-1}^{-1} + H^{\mathrm{T}}R^{-1}H$
$\psi_k = \psi_{k|k-1} + H^{\mathrm{T}}R^{-1}y_k$

Chapter 4

Improving Performance of a Single-layered Perceptor Under the Influence of Attention

In light of the material presented in Chapters 1 to 3, in this chapter, we make an attempt to improve the performance of a single-layered perceptor under the influence of perceptual attention. The perceptor that is used is based on sparse coding algorithm, which has established itself as a useful tool for the representation of natural data in the neuroscience as well as signal-processing literature. The algorithmic implementation of perceptual attention is carried out in a Bayesian context through the use of an information filter, described in the previous chapter.

In order to support the material of this chapter, two computer experiments are presented. In the first experiment, simulated (real-valued) data are used, which are generated to purposely make the problem challenging. The second experiment, on the other hand, uses real-life radar data that are complex-valued. Hence, we propose to introduce Wirtinger calculus into derivation of our new algorithm. Details on Wirtinger calculus are described in Appendix C.

4.1 Sparse Coding: General Overview

4.1.1 Brief History

Sparse coding is a well-known principle in the literature on neuroscience. Early discussions on sparse coding may be traced back to the principle of *"economy of impulses"* first proposed by Barlow (1961) as a form of redundancy reduction; the

work which in its own way was inspired by the classic paper of Shannon (1951) on the redundancy of written English. Economy of impulses was also described in the context of energy consumption: Due to the huge population of neurons, the brain requires a large supply of energy, which by itself can be a big challenge; hence, Levy and Baxter (1996) suggested that the brain economizes energy expenditure by decreasing average firing of neurons.

In an earlier work, Barlow (1972) points out to a similar phenomenon; he argues that in general, activity of neurons in higher layers of the cortex is less than activity of the ones in earlier stages. Furthermore, he suggests that when the probability of a neuron being active is low, its activation provides more information than a neuron with higher activity ratio, and therefore makes a bigger contribution to the perception of the environment. Hence, it is assumed that sparse firing of neurons, i.e., being oftentimes inactive, facilitates the perceptual mechanisms of the brain (e.g., categorizing sensory data into objects), since it provides better separation between categorical representations (Barlow, 2001).

Based on the idea of sparseness, just described, Olshausen and Field (1996) proposed an algorithm for learning features from data in an unsupervised manner. This algorithm is widely known as sparse coding and it works in a two-step fashion: First, input data are processed with respect to a dictionary in order to infer sparse representations corresponding to the data. Then, based on those inferred representations, dictionary elements are updated (e.g., using a single gradient step) with the goal of minimizing reconstruction error over the whole data. These two steps are repeated until convergence is achieved.

The sparse-coding algorithm learns features from data that are statistically independent. Otherwise stated, the learned features capture independent sources that constitute the data. In a similar context, it has also been shown that by maximizing independence of features that are learned from the data, features like the ones learned by the sparse-coding algorithm will emerge (Bell and Sejnowski, 1997; Olshausen and Field, 1997; Hyvärinen and Hoyer, 2000).

Another interesting outcome of sparse coding, when applied to natural image patches, is that receptive fields that emerge as a result of learning are very similar to receptive fields of simple-cells in primary visual cortex (V1). This interesting result has encouraged Olshausen and Field (2004) to suggest that sparse coding can be a strategy used by primary sensory areas in the cortex.

4.1.2 Application to Problems in Engineering

An important point to note here is that sparse coding does not confine to neuroscience. In fact, it is also a well-studied topic in the engineering literature on signal-processing. Various studies with different goals have adopted sparse coding to learn and extract features from data of interest. These goals include denoising, feature extraction, and classification (Hyvärinen, 1999; Lustig, Donoho, and Pauly, 2007; Ranzato et al., 2007; Raina, Battle, Lee, Packer, and Ng, 2007).

Sparse coding has been applied both for static observables (e.g., images) and time-varying observables (e.g., speech, video). Most of the works on time-varying observables operate on a finite sequence of observables with a preset time-duration. In this thesis, however, we are focused on stream of observables with infinite horizon.

4.1.3 Mathematical Formulation of Sparse Coding

Sparse coding is basically a linear inverse problem, where an *observable*, $y \in \mathbb{R}^n$, is represented using a linear superposition of a group of *feature vectors*, $W_i \in \mathbb{R}^n$, as shown in equation (4.1):

$$y = \sum_i z_i W_i, \tag{4.1}$$

or equally:

$$y = Wz, \tag{4.2}$$

with $W \in \mathbb{R}^{n \times m}$ being a *dictionary* that contains non-orthogonal feature vectors, and $z \in \mathbb{R}^m$ being the representation of y in a m-dimensional transform domain ($m > n$). The system of equations for this problem is underdetermined and it does not therefore provide sufficient information that can lead to finding a unique solution. Accordingly, there exists a subspace $C \subset \mathbb{R}^m$ where all the points belonging to that subspace satisfy equation (4.2).

For that reason, the problem does not meet the requirements of being *well-posed* with respect to Hadamard's definition (Hadamard, 1902), and it is regarded as an *ill-posed* problem.[1] Since we are interested in finding a unique solution, additional information must be introduced into the system, which can be done through regularization of the solution (Tikhonov, 1963).

[1] As described in (Haykin, 2009), a problem is well-posed in Hadamard's sense if and only if: there exists a solution for the problem, and that solution can be *uniquely* and *stably* defined.

Sparse coding aims to find representations using only a small number of features from the dictionary. As a result, we may compromise the insufficiency of information by introducing a regularization term involving the ℓ^0-norm. By definition, the ℓ^0-norm of a vector provides the number of nonzero elements in that vector. Accordingly, an optimization problem can be formulated as follows:

$$\hat{z} = \arg\min_{z} \|z\|_0 \quad \text{subject to:} \quad y = Wz. \tag{4.3}$$

This formulation is not convex; hence, it requires a combinatorial approach for minimizing the ℓ^0-norm. However, it is possible to relax this non-convex problem by using the ℓ^1-norm instead of ℓ^0-norm, where $\|z\|_1 = \sum_{i=1}^{m} |z_i|$ with m denoting the dimensionality of z. This way, we will have a convex problem and we can apply convex optimization methods (Boyd and Vandenberghe, 2004) to find the solution.

In real-life problems, observables are acquired with overriding measurement noise. Moreover, the features that are stacked in the dictionary may be imperfect for describing all possible observables. Therefore, it is logical to relax the equality constraint in equation (4.3) and consider a proper error margin. Consequently, other formulations may be introduced for the problem (Tropp and Wright, 2010).

Regularization is in fact the same as assuming a prior distribution for the solution. It can be easily shown (Abdallah and Plumbley, 2006) that the problem can also be formulated using a probabilistic approach by modifying equation (4.2) to:

$$y = Wz + e, \tag{4.4}$$

where $e \in \mathbb{R}^n$ is assumed to be an additive zero-mean Gaussian noise with its elements being independent and identically distributed (*i.i.d.*) with variance σ^2.

In order to impose sparsity on the solution, it is required to select a heavy-tailed distribution that is highly-peaked at zero as the prior distribution of sparse representations. In this regard, a very popular choice is the Laplacian distribution. Assuming that the elements of a representation z are *i.i.d.*, the prior will be:

$$p(z) = \prod_{j=1}^{m} \frac{\lambda}{2} \exp\left(-\lambda |z_j|\right), \tag{4.5}$$

where λ denotes a regularization parameter. Then, after deriving the *likelihood* function of z given the observable y, the sparse representation is found by calculating *maximum a posteriori* of z, also known as the MAP estimate, using Bayes rule:

$$\hat{z} = \arg\max_{z} p(z|y, W) = \arg\max_{z} \frac{p(y|z, W)p(z)}{p(y|W)}. \tag{4.6}$$

Instead of trying to maximize the posterior distribution, the solution to equation (4.6) can be derived by minimizing the negative of the logarithm of the posterior. Therefore, under the probabilistic assumptions just described, the final formulation of the problem is:

$$\hat{z} = \arg\max_{z} \ \frac{1}{2\sigma^2}\|y - Wz\|_2^2 + \lambda\|z\|_1, \tag{4.7}$$

where the parameter λ regularizes the sparseness of the solution; larger values of λ produce more sparse representations.

4.1.4 Does Regularization Result in Well-posedness?

Although the problem just described was formulated in a relatively straightforward manner, the major question is whether regularization of the solution is sufficient for well-posedness of sparse coding or not. The response is positive given that some important conditions are met. However, before elaborating on those conditions, we describe the impact of regularization on the problem using a different approach.

In this regard, we start by doing algebraic expansion on the likelihood function and rewrite it in another form, namely:

$$p(y|z,W) = \tfrac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left[-\tfrac{(y-Wz)^T(y-Wz)}{2\sigma^2}\right] \tag{4.8}$$

$$= \tfrac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left[-\tfrac{1}{2\sigma^2}\left(y^Ty - y^TWz - z^TW^Ty + z^TW^TWz\right)\right] \tag{4.9}$$

$$= \tfrac{1}{c} \exp\left\{-\tfrac{1}{2}\left[z-(W^TW)^{-1}W^Ty\right]^T \ \tfrac{W^TW}{\sigma^2} \ \left[z-(W^TW)^{-1}W^Ty\right]\right\} \tag{4.10}$$

$$= \tfrac{1}{c} \exp\left[-\tfrac{1}{2}\left(z-\mu\right)^T \ \Sigma^{-1} \ \left(z-\mu\right)\right], \tag{4.11}$$

where c is a normalizing factor to equalize both sides of the equation, μ is the pseudo-inverse solution for equation (4.2), and Σ^{-1} is the Fisher information matrix (Van Trees, 1968), which is defined by:

$$\Sigma^{-1} = \frac{W^TW}{\sigma^2}. \tag{4.12}$$

Since the columns of W are non-orthogonal, the matrix product W^TW is ill-conditioned being close to singular. Thus, it is not accurately invertible and as a result, the exact value of μ may not be computable. Fortunately, we are not interested in the pseudo-inverse solution. Rather, we may look to the Fisher information

matrix for properties that are insightful to understand the sparse coding problem better.

Performing eigenvalue decomposition on Σ^{-1} shows that out of the whole set of eigenvalues, some have absolutely insignificant values. Hence, in the transform domain that is described by the set of eigenvectors of this matrix (with μ as the origin), no considerable information is available along the eigenvectors that are associated with those insignificant eigenvalues. This can be equivalently said in another way:

> ***The uncertainty that exists along the eigenvectors with insignificant eigenvalues is extremely high.***

Simple Example

To illustrate the concept just described, we assume the simple example of a bivariate representation space. In this example, the dictionary consists of two non-orthogonal feature vectors with a small angle between them. As a result, one of the two eigenvalues of the Fisher information matrix will be very small. This leads to having a surface for the likelihood function of z, the contours of which are similar to those in Figure 4.1(a), hence expressing the amount of uncertainty that exists (or the available information) along each eigenvector.

By imposing the sparsity constraint through the Laplacian prior, additional information will be introduced into the sparse coding problem. In particular, it is then possible to select the point with the highest probability as the final solution. Contours of the Laplacian prior and the resulting posterior distribution are respectively pictured in Figures 4.1(b) and 4.1(c).

4.1.5 Challenges Faced in Sparse Coding

Figure 4.1(d) clearly shows that an estimated solution may be different from the optimal representation that we wish to find. The estimation error for this simple example may be regarded as negligible; however, practical applications are absolutely not comparable, since the number of feature vectors in the dictionary are significantly larger. Hence, in practical applications, errors in calculated representations may show themselves not just as small amplitude differences but as *inaccurately selected sets of features* to represent observables. This means that the representations may use *irrelevant features* to interpret the observables, which can make the performance of any information processing unit that receives these flawed representations become unreliable.

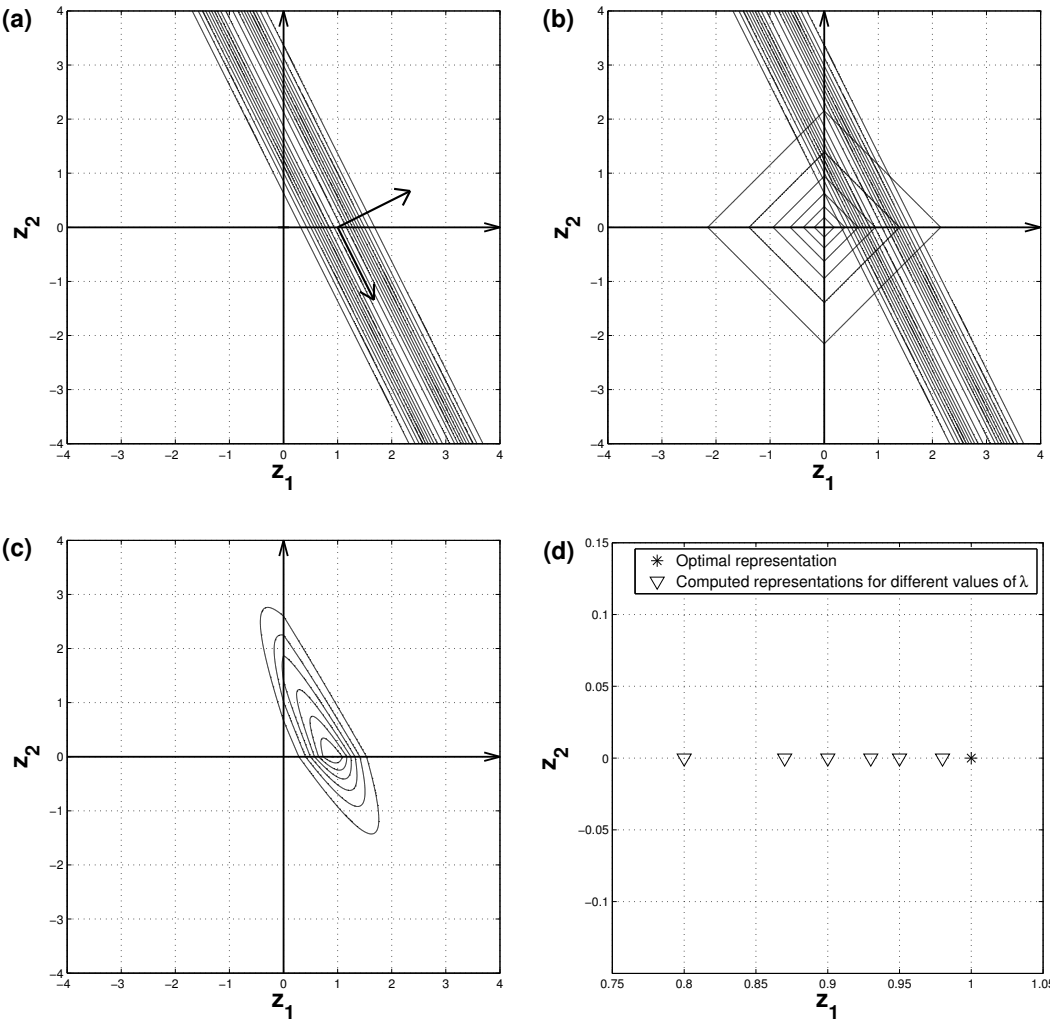


Figure 4.1: (a) The likelihood of z along with the eigenvectors of the Fisher information matrix placed on top of the the contours. (b) The Laplacian prior. (c) The posterior distribution. (d) Calculated solutions for different values of λ.

Moreover, different choices of the parameter λ in the prior result in different estimated values, making it hard to decide which one is a better solution (see, Tropp, 2006). The situation may get even worse, where different initial values for the optimization process may result in different solutions, leading to *instability* of the solution.

The points just described clearly show that along with regularization, there are

other factors that need to be considered in order to guarantee well-posedness of a sparse coding problem.

4.1.6 Conditions for Well-posedness of a Sparse Coding Problem

To describe the conditions for well-posedness, we consider the case of a s-sparse problem, meaning that any observable can be well-represented by a weighted summation of at most s different feature vectors from the dictionary; the meaning is equivalent to saying:

$$\|z\|_0 \leq s. \tag{4.13}$$

Then, we assume that there is a sparse representation z^o that is the *unique sparsest representation* behind the generation of a noiseless observable y^o (i.e., $y^o = Wz^o$); then, considering the unavoidable uncertainty in observables, we may write:

$$y = Wz^o + e, \tag{4.14}$$

with e being the uncertainty with a covariance matrix that is defined by:

$$\mathbb{E}\{ee^T\} = \sigma^2 I_{n\times n}, \tag{4.15}$$

where $\mathbb{E}\{\cdot\}$ provides the expected valued and I is an identity matrix. Given y as an observable, the algorithm calculates a representation $\hat{z}$. Hence, the problem is to define *the conditions that guarantee $\hat{z}$ being equal or reasonably close to z^o*; in (Donoho, Elad, and Temlyakov, 2006), z^o is referred to as the *ideal representation* for the noiseless observable y^o.

According to Donoho and Elad (2003), ability of the sparse coding algorithm to infer z^o from the observable y, is very much influenced by the structure of the dictionary W. This structure may be assessed in a very simple manner using the Gram matrix, which is defined by:

$$G = W^T W. \tag{4.16}$$

Except for the scaling factor $\frac{1}{\sigma^2}$, G is the Fisher information matrix (equation (4.12)). Apparently, its diagonal elements are equal to one, since the columns of W are normalized to unit ℓ^2-norm. Based on the off-diagonal entries of G, a measure known as *mutual coherence*, denoted by M, is introduced, which equals the maximum

value among the absolute values of those entries. Then, if the problem satisfies the following constraint:

$$s < \frac{1 + \frac{1}{M}}{2}, \tag{4.17}$$

it has been shown by Donoho et al. (2006) that $\hat{z}$ approximates z^o with a deviation constrained as follows:

$$\|\hat{z} - z^o\|_2^2 \leq \frac{4n\sigma^2}{1 - M(2s-1)}. \tag{4.18}$$

It is noteworthy that equation (4.17) is for the ℓ^0-norm solution; for the ℓ^1-norm, the constraints will be twice as strict, where in equations (4.17) and (4.18), s is replaced with $2s$ (Donoho et al., 2006).

Therefore, basically, well-posedness of a sparse coding problem depends on three important factors, namely:

i) sparseness of the process, s,

ii) the mutual coherence of the dictionary, M, and

iii) the level of uncertainty in observables, σ^2.

In case equation (4.17) does not hold, and/or the uncertainty of observables is high, the information available is insufficient for solving the problem.[2] As a result, there is no way to guarantee that $\hat{z}$ equals or is reasonably close to z^o; even worse than that, we may not be sure that $\hat{z}$ represents y using the same set of features present in z^o, or in other words, using *relevant features*.

The point just made is of great importance in perceptual problems and classification tasks, since a representation that includes irrelevant features is in fact providing misleading information about the source that generated an observable. Although such a representation may still be good for denoising the observable, it will not be effective for perceiving the environment.

4.2 Mitigating Ill-posedness by Introducing Additional Information

In practical applications of real-life data, we do not have prior knowledge of features in the observables. As a result, we are required to learn the features statistically,

[2]Here, we mean the information available from the likelihood function added to the information introduced by regularization.

which makes the algorithm susceptible to learning imperfect features. Furthermore, the statistically learned features may result in having improper structural properties for the dictionary. Added to the points just described will be the presence of uncertainties in the observables, which collectively may reduce the ability of the sparse coding algorithm to approximate ideal representations closely.

Consequently, sparse coding may turn into an ill-posed problem that is incapable of extracting relevant features from the observables. This leads us to consider the *instability* of representations, where a sequence of noisy observables generated from the same set of features, may result in the calculation of highly different representations.

For sparse coding to be well-posed, there has to be sufficient information. To this end, increasing the amount of available information by some practical means is required to mitigate the ill-posedness of sparse coding and thereby improve the *stability* of inferred representations. This requirement is realized by bringing into play perceptual attention through the use of an information filter. Thereby, the information that could exist in successive observables accumulates iteratively. In light of increasing information-level from one filtering cycle to the next, relevant features will be favoured and irrelevant features will be suppressed. Hence, the inferred representations become gradually closer estimates of ideal representations and with it, they become more reliable.

4.3 Algorithmic Implementation of Perceptual Attention

As indicated previously in Chapter 2, we may use the Bayesian approach to improve sparse coding under the influence of perceptual attention. Thus, for implementation of the new algorithm, the first idea that comes to mind is to follow the diagram depicted in Figure 3.1. This can be done using the posterior distribution of the sparse representation that is provided in equation (4.6). Then, according to a suitable state-space model for the problem at hand, a predicted distribution will be derived for the sparse representation, which will be used as the prior for the next cycle. However, there are two major challenges that make this idea undesirable, as described next.

First, the posterior distribution of the representation provided by the sparse coding algorithm is non-Gaussian and performing filtering cycles using a non-Gaussian distribution requires approximation methods. Second, considering the discussion presented in Section 4.2, the sparse coding algorithm may represent an observable using a combination of relevant and irrelevant features. Therefore, if we define a

predicted distribution based on a flawed representation, the information content of the observable will not be preserved and invalid information will be imposed on the next filtering cycle. In effect, the estimation performed in the next cycle will be flawed as well; as we go through such filtering cycles, errors would tend to accumulate successively and the algorithm diverges.

For this particular reason, the filtering process that is performed in the new algorithm is carried out separately without involving the hard decisions made by the sparse coding algorithm. Accordingly, it is proposed that the new algorithm consists of a filtering block followed by a sparse coding unit, as depicted in Figure 4.2(b).

As observables are received from one cycle to the next, the information they contain will be accumulated in the posterior distribution of the filter. In addition, sparse representations are calculated at each cycle using the posterior of the filter. This is done by imposing sparsity on the posterior; to elaborate, the posterior of the filter is conditioned by a Laplacian distribution using Bayes rule. Finally, the MAP estimate of the resulting distribution is calculated and sparse representations are inferred.

Figure 4.3(b) displays the information-flow diagram of the new algorithm. In this figure, $\hat{z}_k$ is the estimated value for sparse code z at k^{th} cycle, and $p(z_k)$ is the Laplacian distribution applied at cycle k for the regularization of the solution. Moreover, the pdfs $p(y_k|z_k)$, $p(z_k|Y_k)$, and $p(z_{k+1}|Y_k)$, are the likelihood function, posterior distribution, and predicted distribution, respectively, with Y_k being the sequence of observables available at cycle k.

Details of the sparse coding algorithm are already described in Section 4.1. Thus, by deriving the algorithmic steps of the filtering block shown in Figure 4.2(b), implementation of the new algorithm will be accomplished. In this regard, the filtering block is implemented using the simplified state-space model:

$$\begin{cases} z_{k+1} = z_k + \gamma_k \\ \\ y_k = W z_k + \omega_k \end{cases} . \tag{4.19}$$

This model basically means that we are dealing with a slowly changing environment. In such an environment, different features that are present in observables emerge and disappear slowly.[3] It follows therefore that features from cycle k represented by z_k are expected to be present at cycle $k+1$ with some degree of uncertainty

[3]Changes in the environment are governed by meaningful physical dynamics, which make the changes have trackable trajectories. Likewise, it is reasonable to expect such behaviour from features that are latent in our observations of the environment. In other words, it is expected that those features emerge and disappear

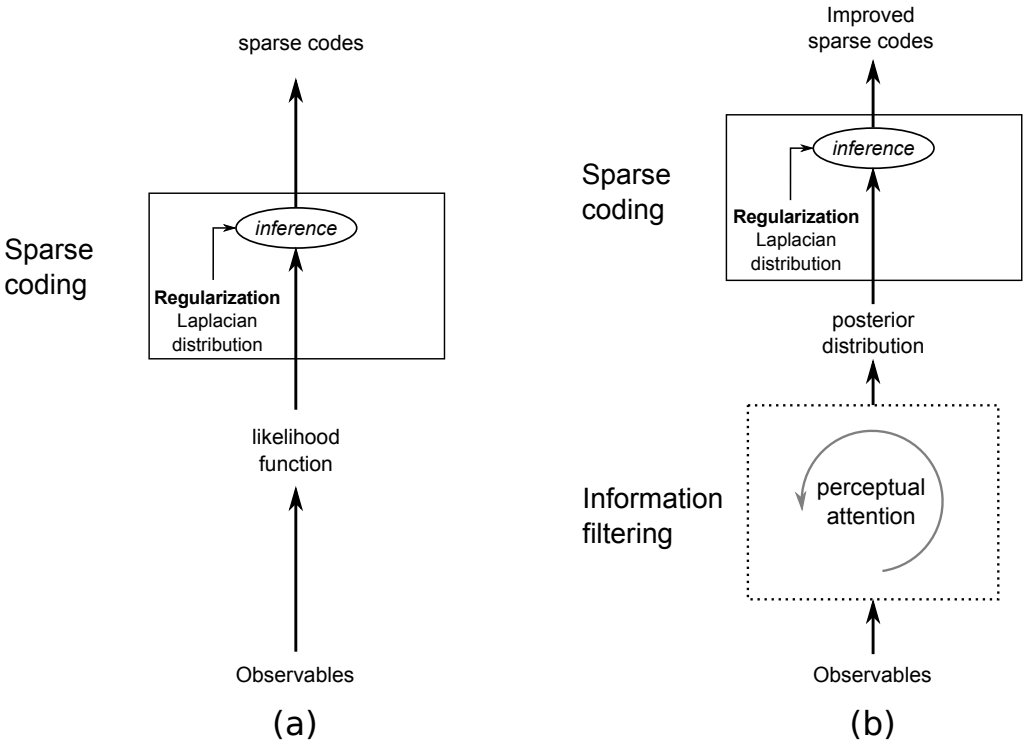


Figure 4.2: Block diagram of algorithms: (a) sparse coding. (b) sparse coding under the influence of perceptual attention.

(e.g., the process noise). It goes without saying that the measurement equation in equation (4.19) is the same linear model used in the sparse coding algorithm (equation (4.4)).

In equation (4.19), y denotes an observable. To simplify matters, the process noise denoted by γ is assumed to be a white, additive, zero-mean, Gaussian process with a covariance matrix Q. Moreover, the measurement noise denoted by ω is

with trackable trajectories; whereas, an emerging feature remains present over a sequence of successive observables for a reasonable duration of time before it gradually disappears. This kind of behaviour has been the main idea behind many studies on spatio-temporal analysis of data, e.g. spatial invariance learning (Földiák, 1991; Einhäuser, Kayser, König, and Körding, 2002; Hyvärinen, Hurri, and Väyrynen, 2003; Karklin and Lewicki, 2005), independent feature subspaces (Hyvärinen and Hoyer, 2000), and slow feature analysis (Wiskott and Sejnowski, 2002; Berkes and Wiskott, 2005).

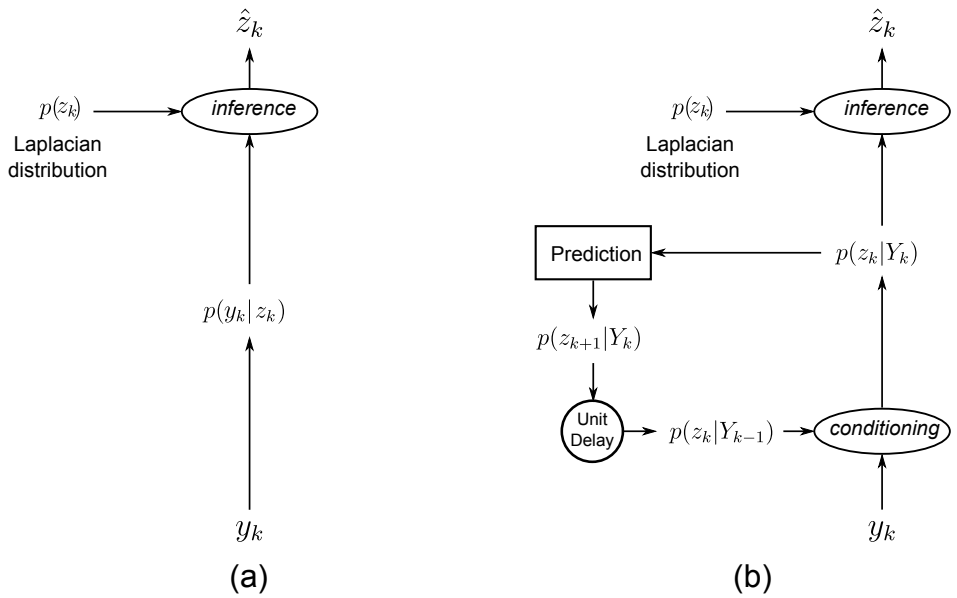


Figure 4.3: Information-flow diagram of algorithms: (a) sparse coding. (b) sparse coding under the influence of perceptual attention.

assumed to be a white, additive, zero-mean, Gaussian process, whose elements are *i.i.d.* with variance σ_ω^2.

Due to the Gaussian assumption for both γ and ω, representations, denoted by z, are normally (Gaussian) distributed as well. Hence, both optimal derivations of a Bayesian filter, i.e., Kalman filter and information filter, may be used for our problem. In spite of that, our preference for the method of filtering lies with an information filter. Our motivation is simple: Algorithmic steps of an information filter may be combined with the sparse coding algorithm in a more computationally efficient manner, where Σ^{-1} and ψ are directly used by the sparse coding algorithm without any need for matrix inversions.

Based on descriptions provided in Section 3.3, algorithmic steps of an information filter for the state-space model of our problem, described in (4.19), are presented in Algorithm 3.

4.3.1 How Information Filtering Leads to Attention

Before describing how attention is actually performed in our algorithm, it is important to note that attention requires a decision-making process; that is, in order to attend to the relevant part of sensory information and suppress the irrelevant part,

Algorithm 3 Information Filtering Block in the New Algorithm

$\Sigma_0^{-1} \leftarrow 0$
$\psi_0 \leftarrow 0$
Repeat for $k = 1, 2, 3, ...$
prediction:
$A_k = I_{m\times m} - \Sigma_k^{-1}(\Sigma_k^{-1} + Q^{-1})^{-1}$
$\Sigma_{k+1|k}^{-1} = A_k \Sigma_k^{-1}$
$\psi_{k+1|k} = A_k \psi_k$
innovation:
$\Sigma_k^{-1} = \Sigma_{k|k-1}^{-1} + \frac{1}{\sigma_\omega^2} W^{\mathrm{T}} W$
$\psi_k = \psi_{k|k-1} + \frac{1}{\sigma_\omega^2} W^{\mathrm{T}} y_k$

a decision must be made to distinguish between these two parts, just described. In this context, the decision-making process is intrinsically performed during the filtering cycles, based on the state-space model that embodies the measurement noise and the process equation.

The measurement noise, responsible for irrelevant information, is assumed to be white; values of the noise are therefore independent over successive filtering cycles. On the other hand, features that are present in those successive cycles are not independent; thereby, they are related to each other over time in accordance with the process equation of the state-space model. Consequently, the information filtering algorithm is enabled to make predictions about new observables. In those predictions, the extent to which each feature in the dictionary is expected to be present in the new observables is probabilistically described. The probabilistic descriptions may then influence the extraction of features from the observables in the following manner:

- In the estimation of sparse representations, competition between the features will be biased in favour of the ones that have higher probabilities of being present in the new observables. Hence, the higher the predicted probability is for the presence of a particular feature, the stronger the bias will be toward the selection of that feature in the representation of the pertinent new observable. In a neurobiological context (Reynolds and Chelazzi, 2004), the effect of such a bias is similar to that of reducing a neuron's contrast-response threshold in the visual cortex for an attended location in the receptive field, the result of which is to facilitate response of the neuron to a stimulus that would otherwise be too weak to excite that particular neuron.

- In a similar manner, features that are unlikely to be present in the observables (i.e., features with high probability of not being present in the observables) are more likely to be suppressed by the algorithm and therefore not used in the representations. The effect of this kind of inhibition is also similar to that of response-suppression of neurons in the visual cortex (Reynolds and Chelazzi, 2004).

Benefiting from the *excitatory* and *inhibitory* effects just described, the algorithm is equipped with a process that mimics attention. To regulate the influence of attention on the operation of the algorithm, the process noise comes into play. To be specific, the level of process noise directly defines the confidence that the algorithm has in predicting observables in the future; when the process noise is relatively large, it will relax the influence of attention. On the other hand, when the process noise is relatively small, the influence of attention becomes stronger.

4.3.2 Cognit: the Name of the New Algorithm

To ease the reference to our new algorithm (Figure 4.2(b)), we have chosen the terminology *cognit* introduced by Fuster (2003). Cognits are described as networks of knowledge in the cortex, which represent any information about "*the world, the self, and the relations between them*;" the statement justifies the proposed terminology rather neatly.

4.4 Simulations: Part I

In order to evaluate the performance of the cognit, a simulation with two objectives will be presented. The objectives are described in the subsections that follow. With data generation being under our control for this specific experiment, we use the terminologies "original representations" and "original dictionary".

4.4.1 Objective I: Inferring Original Representations

The first objective of the experiment is to illustrate how the cognit could extract features from observables to infer the original representations, and compare it to the sparse coding algorithm acting alone.

For the experiment, feature vectors were chosen to be Gabor wavelets generated with random parameters, with a dictionary consisting of 50 features. To produce ob-

servables, an initial s-sparse random representation vector, z_0, was first generated, where s was randomly selected from the set: $\{3, 4, 5\}$.[4]

Initializing the state-space model described in equation (4.19) with z_0, original representations, z_k^o, and noisy observables, y_k, were then generated recursively for a duration of 50 cycles with a relatively low signal-to-noise ratio (SNR) of about 6dB. The process noise, γ, and measurement noise, ω, were both chosen to be white, zero-mean, Gaussian processes.[5] Figure 4.4 shows a sample observable from the generated sequence.

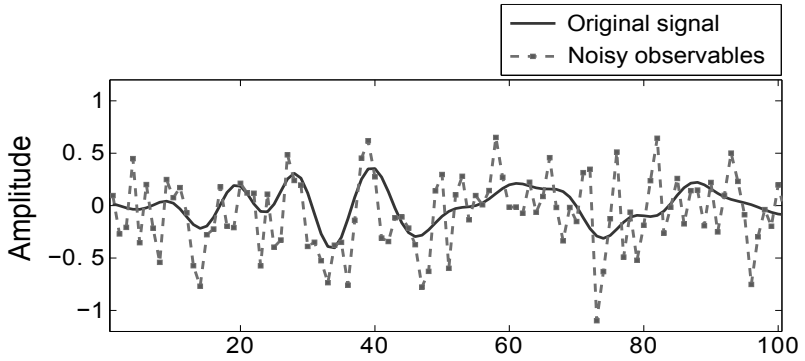


Figure 4.4: A sample observable from a generated sequence.

As explained in Section 4.2, a sparse coding problem experienced in practice is prone to be ill-posed; in other words, the conditions required for having a well-posed problem may be violated in practical applications (equations (4.17) and (4.18)). To add realism to the simulation, we purposely assigned values to the mutual coherence of the generated dictionaries, M,[6] the sparseness of the problem, s, and the SNR of observables, which would make the experiment ill-posed.

The cognit and the sparse coding algorithm were then initialized with dictionaries and noise process models that were exactly the same as the ones used in generating the observables. In addition, the sparsity prior that was used in both algorithms

[4]The representation was first sampled from a Laplacian distribution with $\lambda = 1$. Then, only s number of elements with highest absolute values were kept active in the representation and the rest of the elements were set to zero.

[5]Elements of γ and ω were *i.i.d.*, respectively with variances $\sigma_\gamma^2 = 0.05$ and $\sigma_\omega^2 = 0.1$; similar results were achieved using other values for σ_γ^2 and σ_ω^2.

[6]In the experiment, the average value for the mutual coherence of generated dictionaries was $M = 0.79$, which is highly undesirable for sparse coding applications.

was a Laplacian distribution with $\lambda = 1$. To assess performance of the two algorithms, 10,000 Monte Carlo runs were performed; on each run, a new randomly generated dictionary[7] was used and new observables were produced accordingly. Moreover, both algorithms were supplied with the same set of observables.

Figures 4.5(a) and 4.5(b), respectively display ability of the algorithms to find the original representations, and to denoise the observables. According to the results presented therein, the cognit showed a better performance than the sparse coding algorithm in both of the tasks just mentioned. The criterion used for these graphs was the mean squared error (MSE) values at each filtering cycle, averaged over the Monte Carlo runs.

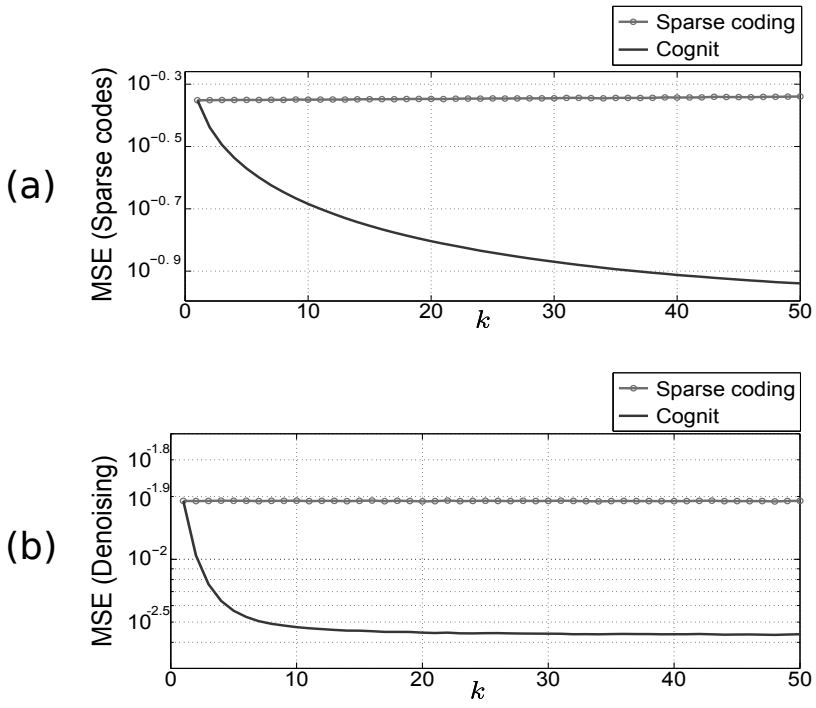


Figure 4.5: Simulation results for objective I. Performance of the algorithms in: (a) finding the original representations, (b) denoising the observables.

Figure 4.6(a) illustrates absolute values of original representations in a Monte Carlo run, and Figures 4.6(b) and 4.6(c) display absolute values of representations that were computed by the sparse coding algorithm and the cognit, respectively. The

[7] Figure 4.8(a), to be projected later, displays a sample dictionary.

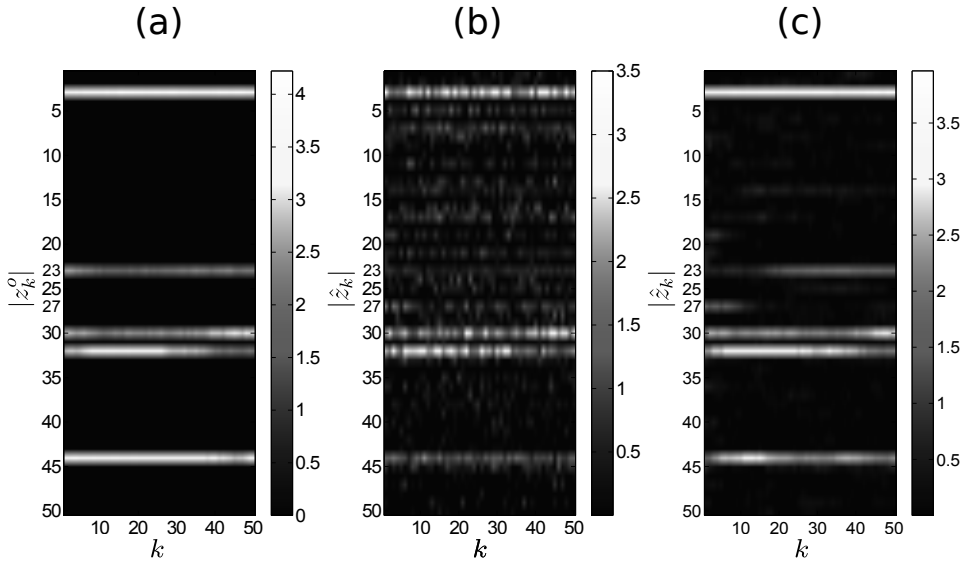


Figure 4.6: Simulation results for objective I. Absolute values of representations across successive cycles: (a) original values, (b) the sparse coding algorithm, (c) the cognit.

y-axis in each figure shows the elements of a representation vector and the x-axis displays the corresponding filtering cycles. It can therefore be seen that as a result of information feedback that flows through local-PAC, the stability of representations as well as their sparseness are much better for the cognit than they are for the sparse coding algorithm acting on its own.

Furthermore, Figure 4.6(a) shows that feature vector number 27 is not present in the original representations. However, that feature vector was wrongly detected by the cognit in the beginning, as displayed in Figure 4.6(c). Nevertheless, this irrelevant feature was gradually suppressed after a number of cycles. On the other hand, feature vector number 23 that is present in the original representations was too faint to be detected in the beginning but it gradually emerged during the subsequent cycles. Such a desirable behaviour, attributed to the cognit, is a consequence of the accumulation of information from one incoming observable to the next through information filtering.

4.4.2 Objective II: Discovering Original Dictionary from Observables

The second objective of the experiment is to demonstrate how the cognit could learn features from the observables to discover the original dictionary, and compare it to the sparse coding algorithm acting alone.

To be specific, given a particular dictionary, sufficient training data were generated in a manner similar to that in the previous subsection using the same models for the noise processes. The data were then supplied to the sparse coding algorithm and the cognit, and features were learned accordingly; as for the learning rule, we used the one proposed in (Olshausen and Field, 1996), defined by,

$$\Delta W = \big(y - Wz\big)z^T, \tag{4.20}$$

$$W(t+1) = W(t) + \eta < \Delta W >_p, \tag{4.21}$$

where $< \cdot >_p$ is the average over a batch containing p number of examples, and η is a proper learning rate.

The process of learning the two dictionaries occupied 10,000 *epochs*; in each epoch, the two dictionaries were trained using a mini-batch that contained 100 training examples, where each example was a sequence of observables with the duration of 20 cycles.

Dictionaries are often initialized with random values; thus, during the primary stages of dictionary learning, it is more efficient for the cognit to learn features without applying information feedback. Once a desired level of progress in learning is achieved, local-PAC is initiated; in this experiment, we used the same learning strategy and activated the feedback channel of the cognit after 1000 training epochs.

During training, the learned dictionaries were compared to the original dictionary. The comparison was done by evaluating how successful each of the algorithms was in discovering the original features in the observables. For this purpose, the learned features were first paired with the ones in the original dictionary by finding the best match for each one of them. Then, an *average conformity* measure was defined by:

$$C = \frac{1}{m}\sum_{i=1}^{m} | < W_i^l, W_i^o > |, \tag{4.22}$$

where W^o denotes the original dictionary, W^l denotes the learned dictionary, and $< W_i^l, W_i^o >$ is the inner product of the ith pair of matched features. Figure 4.7 shows values of the average conformity measures for the two algorithms

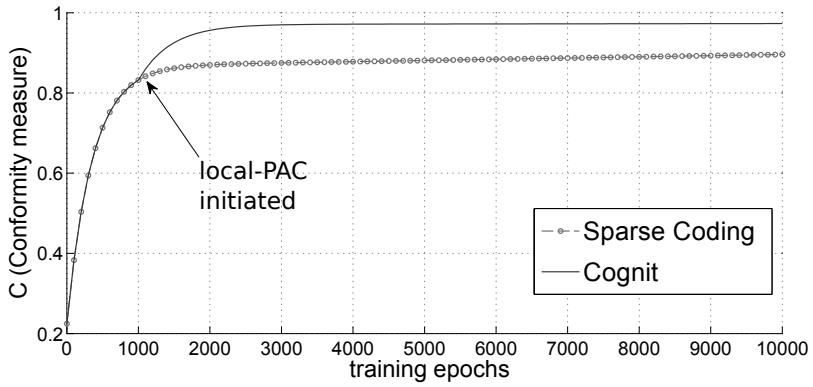


Figure 4.7: Values of the average conformity measures vs. the number of training epochs.

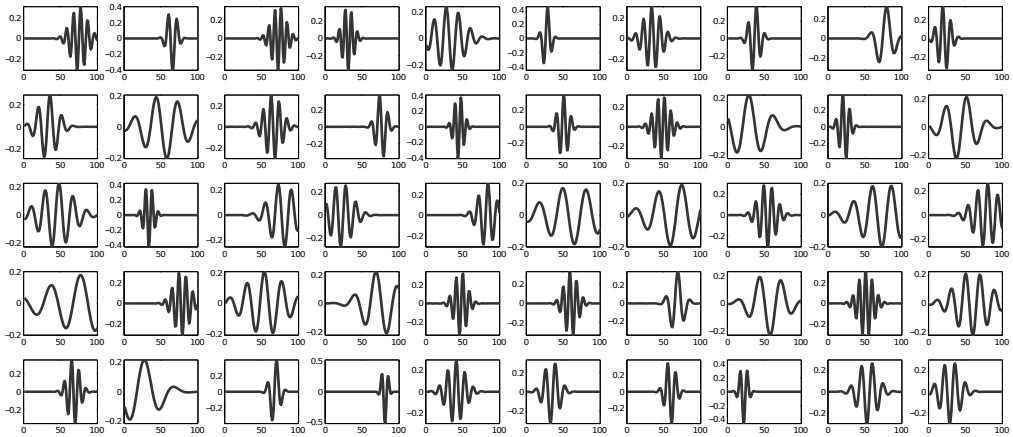
with respect to the number of training epochs; the values are averaged over 200 Monte Carlo runs, where on each run, a new randomly generated dictionary was used and new observables were produced accordingly.

At the end of the training process, the results showed an improved learning ability for the cognit (97.1%) compared to the sparse coding algorithm (89.6%). Moreover, the cognit was able to learn more than 80 percent of the features with above 99% conformity with the original features. On the other hand, the corresponding results for the sparse coding algorithm were inferior, where only about 35 percent of the features reached 99% conformity. Figure 4.8(a) presents a sample original dictionary, and Figures 4.8(b) and 4.8(c) display the dictionaries that were learned respectively by the sparse coding algorithm and the cognit on one of the Monte Carlo runs.

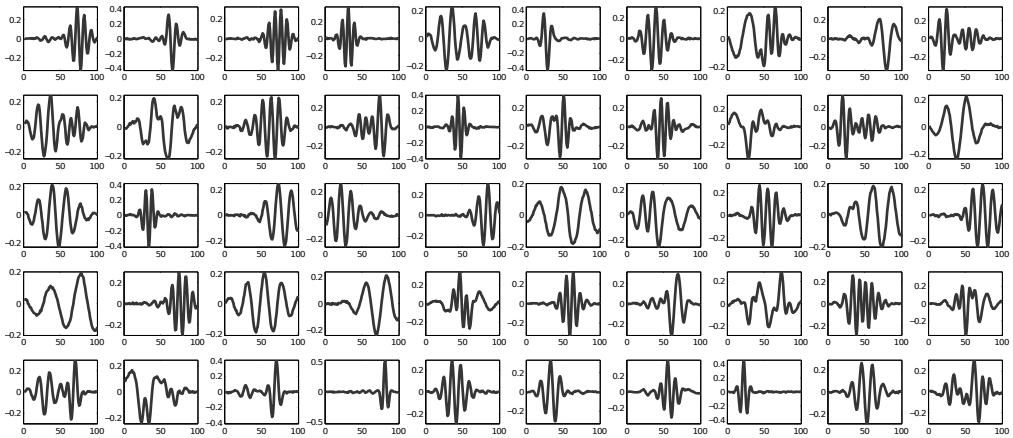
The results just presented confirm the positive effect of local-PAC on performance of the cognit. The ability of the cognit to accumulate information over successive observables results in inferring representations that are closer to the original representations. Therefore, the quality of learning will improve at each cycle due to having more reliable representations. Then, in a reinforcing manner, a better learned dictionary will improve the quality of representations in subsequent cycles.

4.4.3 Summarizing the Results From Part I

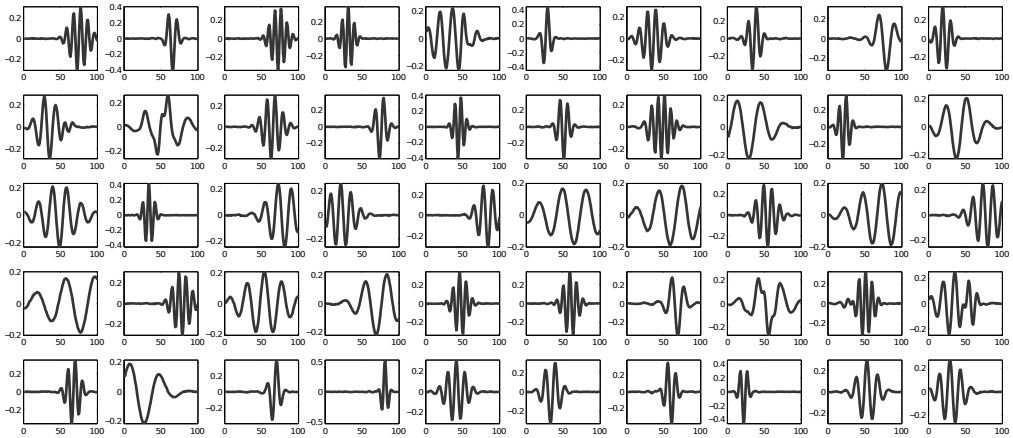
The iterative employment of local-PAC in the cognit has a successively purifying effect as we proceed from one filtering cycle of the information filter to the next. Due to the presence of perceptual attention in the cognit that is facilitated through



(a)



(b)



(c)

Figure 4.8: Dictionary elements: (a) original, (b) learned by sparse coding, (c) learned by the cognit.

local-PAC, the results show an improved ability of the cognit to infer the original representations and denoise the incoming observables, compared to the sparse coding algorithm.

In addition, it has been shown that initiating local-PAC results in a boost in the learning ability of the cognit, enabling it to outperform the sparse coding algorithm. This can be attributed to the supporting role of perceptual attention in the operation of the cognit, which improves the extraction of information from observables by focusing on relevant information and suppressing irrelevant information. Thus, at each cycle of the information filter, the dictionary of the cognit is updated using a refined version of information, which improves the quality of learning.

4.5 Application to Complex-valued Problems

The sparse coding algorithm is typically applied to real-valued observables. In many practical applications, however, we are required to work in the complex domain (e.g., radar, medical imaging, communication systems, etc.). To be able to use sparse coding in such applications or other similar problems, it is necessary to rewrite the formulation of the sparse coding algorithm with respect to complex-valued variables of the problem. On this topic, it is of interest to note that Cadieu and Olshausen (2012) provided an implementation of sparse coding with both dictionary and sparse representations being complex-valued. Their algorithm was applied to real-valued image sequences from natural movies in order learn representations of form and motion.

In many complex-valued problems, like the one described in (Cadieu and Olshausen, 2012), derivation of formulas is done by mapping complex-valued variables to the real domain. However, this will complicate both the derivation process and the algorithmic steps. The reason is that mapping of variables, replaces each complex-valued variable with two of the real-valued kind. To avoid this problem and preserve simplicity of the sparse coding algorithm, we suggest the use of *Wirtinger calculus* (Wirtinger, 1927), well described in Appendix C.

Using Wirtinger calculus, formulation of a problem in the complex domain is simplified and most of the tools for real-valued signals can be extended to complex-valued problems (Adali and Haykin, 2010). In this section, the formulation of sparse coding is presented for complex-valued applications using Wirtinger calculus. Likewise, our proposed algorithm that benefits from perceptual attention is extended to applications in the complex domain.

4.5.1 Sparse Coding in the Complex Domain

Sparse coding in the complex domain is basically similar to the real-valued case. The major difference is that the parameters of the system are in the complex domain. Thus, probabilistic modelling of the algorithm should be done using complex-valued probability distributions and learning rules should be derived accordingly.

The sparsity requirement demands that an observable is represented by only a small number of features out of the whole dictionary. Thus, for a complex-valued representation, denoted by z, the sparsity requirement is only concerned with amplitudes of the coefficients in a representation. The amplitude of a coefficient, denoted by $|z_j|$, represents the extent to which a complex-valued feature, W_j, exists in an observable. Thus, sparseness, per se, does not impose any constraint on the phase value of that coefficient, $\angle z_j$.

The point just mentioned simplifies the choice of a prior distribution that is suitable for complex-valued sparse coding. In this regard, a circular-symmetric distribution that is heavy-tailed and highly-peaked at $z = 0$ may be a suitable choice. Assuming the statistical independence of amplitude and phase for each coefficient, z_j, the prior distribution of a coefficient is written as:

$$p(z_j) = p(|z_j|)\, p(\angle z_j), \tag{4.23}$$

where, by using a circular-symmetry property for $p(z_j)$, values of the phase will have a uniform distribution in $[0, 2\pi)$. Therefore, a zero-order Gamma distribution could be a good candidate for the prior. Thus, assuming that the coefficients of z are *i.i.d.*, $p(z)$ will be:

$$p(z) = \prod_{j=1}^{m} \frac{\lambda^2}{2\pi} \exp\left(-\lambda |z_j|\right), \tag{4.24}$$

where λ denotes a regularization parameter, and m denotes the dimensionality of z.

Depending on the application of interest, the observables that are supplied to a complex sparse coding algorithm may be real- or complex-valued. In this thesis, we concern ourselves with complex-valued observables. Hence, the formulation of the problem will be:

$$y = W z^* + e, \tag{4.25}$$

where z^* is the complex conjugate of a sparse representation, y is a complex-valued observable, and e is the error. Using z^* instead of z has no computational privilege and we have only used it to distinguish the formulation from a sparse coding algorithm in the real domain.

In our generative model, the probability distribution of error, $p(e)$, is represented by a circular-symmetric multivariate Gaussian distribution, namely,

$$p(e) = \frac{1}{(\pi\sigma^2)^n} \exp -\frac{e^H e}{\sigma^2}, \tag{4.26}$$

where, e^H is the Hermitian transpose of e, and σ^2 is the variance of complex-valued error elements, e_i. The elements of e are assumed to be *i.i.d.*

Accordingly, the cost function is defined by:

$$C = \frac{1}{\sigma^2}\big(y - Wz^*\big)^H\big(y - Wz^*\big) + \lambda\sum_{j=1}^{m}|z_j|, \tag{4.27}$$

and a representation may be inferred by minimizing the cost function, for example, using a gradient based algorithm. In this regard, the direction of descent for equation (4.27) will be:

$$-\frac{\partial C}{\partial z^*} = \frac{1}{\sigma^2}W^T\big(y - Wz^*\big)^* - \frac{\lambda}{2}\,\text{sign}(z), \tag{4.28}$$

where, $\text{sign}(z_i) = \dfrac{z_i}{|z_i|}$. In addition, the complex-valued dictionary is updated in a manner similar to the way in which it was done for a real-valued dictionary in equation (4.20). Thus, using the following rule:

$$\Delta W = \big(y - Wz^*\big)z^T, \tag{4.29}$$

$$W(t+1) = W(t) + \eta < \Delta W >_p, \tag{4.30}$$

where $< \cdot >_p$ is the average over a batch containing p number of examples, and η is a proper learning rate.

4.5.2 Information Filtering for a Cognit in the Complex Domain

In the case of complex-valued representations and observables, the state-space model of the cognit can be defined in a similar way that it is done in equation (4.19) with only the measurement equation slightly modified. Thus, we have:

$$\begin{cases} z_{k+1} = z_k + \gamma_k \\ \\ y_k = Wz_k^* + \omega_k \end{cases}. \tag{4.31}$$

To be able to formulate the information filter, we first need to define the probability distribution of the process noise, γ. To this end, we assume γ to be an additive white noise that has a zero-mean circular-symmetric complex Gaussian distribution with covariance matrix Q, where Q is defined by:

$$Q = \mathbb{E}\{\gamma\gamma^H\}. \tag{4.32}$$

Then, filtering steps may be derived in a manner similar to that in Section 4.3. In this case, the measurement noise, ω, has a distribution like the one described in equation (4.26). Accordingly, both noise models that are used in the state-space model of the cognit are white circular-symmetric complex Gaussian distributions. For this reason, when we derive the likelihood function of z (or similarly z^*), we shall see that the *relation* matrix, C, equals zero, where C is defined by:

$$C = \mathbb{E}\{zz^T\}. \tag{4.33}$$

For details on complex multivariate Gaussian distributions, see (Picinbono, 1996).

Consequently, the information filter of the cognit may be derived either around z or z^*, in a m-dimensional space, and mapping to $\mathbb{C}^{2m}$ is therefore not required. Hence, algorithmic steps of the filter will be generally the same as Algorithm 3 with only two minor modifications. Algorithm 4 describes the steps needed for the information filter; the steps are derived with respect to z.

Algorithm 4 Information Filtering Block for Complex-valued Observables

$\Sigma_0^{-1} \leftarrow 0$
$\psi_0 \leftarrow 0$
Repeat for $k = 1, 2, 3, ...$

prediction:

$A_k = I_{m\times m} - \Sigma_k^{-1}(\Sigma_k^{-1} + Q^{-1})^{-1}$
$\Sigma_{k+1|k}^{-1} = A_k\Sigma_k^{-1}$
$\psi_{k+1|k} = A_k\psi_k$

innovation:

$\Sigma_k^{-1} = \Sigma_{k|k-1}^{-1} + \frac{1}{\sigma_w^2}W^{\mathrm{T}}W^*$
$\psi_k = \psi_{k|k-1} + \frac{1}{\sigma_w^2}W^{\mathrm{T}}y_k^*$

4.6 Simulations: Part II - Applying the Cognit to Real-life Data

In the second part of computer experiments for this chapter, we applied the cognit and the sparse coding algorithm to real-life data from the Dartmouth dataset (available at http://soma.ece.mcmaster.ca/ipix/dartmouth/datasets.html) collected by the McMaster IPIX radar. The data were acquired for a radar experiment involving a small spherical target made up of foam, about one meter in diameter, anchored in water in close proximity to Dartmouth on the East Coast of Canada.[8] The dataset consists of 14 files with each file containing over two minutes of recording from 14 range-bins with a sampling rate of 1 kHz.

Before describing the details of this experiment, it is noteworthy that by illuminating the environment with an electromagnetic wave, radar provides visual information about the environment. Thereby, objects are located and categorized through analysis of echo signals (radar returns) by considering factors such as power of echo signals, time delay between signal transmission and its received echo, angle of arrival of the echo, and the Doppler shift.

In biology, visualizing the environment through the analysis of echos is known as *echolocation*, a term coined by Griffin (1944). Some examples of echolocating animals include horseshoe bats (Neuweiler, 1984), wandering shrews (Buchler, 1976), and dolphins (Au, 1993). In addition, advanced form of human echolocation has been studied in subjects with impaired vision (Rojas, Hermosilla, Montero, and Espi, 2009).

With this background material, there are three objectives in conducting this second experiment, as described in what follows. Before that, however, the data used in this experiment will be presented next.

4.6.1 Description of the Data

The real-life data used in this experiment are characterized by amplitude and phase, with the latter resulting in a continually varying Doppler shift that characterizes

[8]The Dartmouth dataset collected by the IPIX radar is somewhat rare and with the dataset being available online, many papers from around the world have been published using it for a variety of radar applications. For example, see the following cited papers: Anastassopoulos, Lampropoulos, Drosopoulos, and Rey (1999); Conte, De Maio, and Galdi (2004); Hu, wen Tung, and Gao (2006); Gao, Hu, Tung, Cao, Sarshar, and Roychowdhury (2006); Sangston, Gini, and Greco (2010).

stochastic motion of the target, sometimes visible on the ocean surface and at other times buried in water below the surface. Figure 4.9 exemplifies a small portion of the whole dataset, where amplitudes of the radar returns are shown across the 14 range-bins for a duration of 10 seconds.

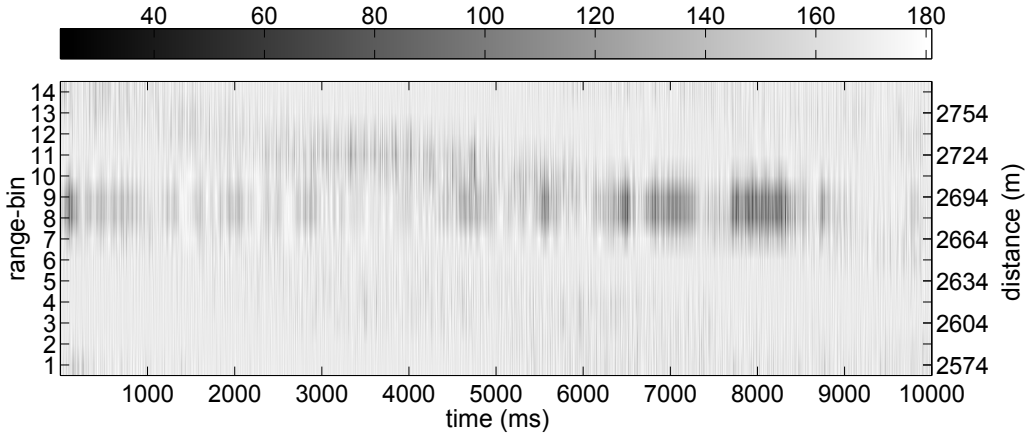


Figure 4.9: Amplitudes of the radar returns across the measured range-bins.

In Figure 4.9, the target is anchored in range-bin number 8, and occasionally, fluctuations cause it to slightly show its presence in bin number 7, 9, and 10; the rest of the bins belong to sea-clutter. As it may be observed, presence of the target may only be discerned visually every once in a while, and often there is no clear difference among the range-bins.

For this experiment, a window size of 128 samples (or equally 128 ms) was selected as the dimensionality of an observable y. Accordingly, the Fourier amplitude spectra of two observables acquired from the bin containing target-plus-clutter and a bin containing only clutter (e.g., bin number 4) are displayed respectively in Figures 4.10(a) and 4.10(b) using the discrete Fourier transform.

When the target is visible, like it is in Figure 4.10(a), there is high likelihood that the amplitude of its spectrum is larger than that of clutter. However, during the times that the target disappears under water, which happens very often in this dataset, the amplitude spectrum for the bin containing target-plus-clutter is essentially similar to the bin containing only clutter (i.e., Figure 4.10(b)).

Motion of the target is restricted due the target being anchored. Hence, as shown in Figure 4.10(a), the Doppler shift that is caused by motion of the target is smaller than that caused by the ocean waves. Nevertheless, there is no visible separation between the amplitude spectrum of the target and that of clutter.

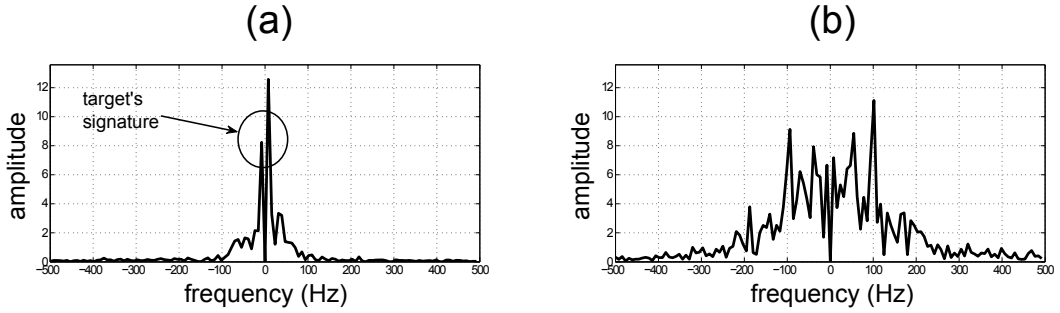


Figure 4.10: Amplitude spectra of two observables: (a) target-plus-clutter (b) clutter.

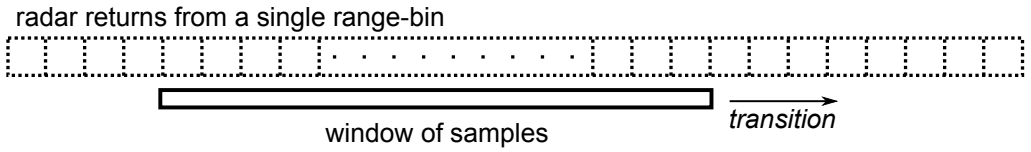


Figure 4.11: Performing one-step transitions in time to simulate the arrival of new observables for a particular range-bin.

To simulate the arrival of new observables at every time-step, the window of samples would be shifted one step forward in time to include a new sample in the selected window; thus, for every range-bin, a new observable could be defined at each time-step, individually, as illustrated in Figure 4.11.

Consequently, amplitude spectra of observables from two range-bins containing target-plus-clutter and containing only clutter are respectively illustrated in Figures 4.12(a) and 4.12(b); the observables used for computing these two spectra are generated from the radar returns that were previously presented in Figure 4.9. In order to provide better visibility across different frequencies, the logarithm of amplitudes has been used in Figures 4.12(a) and 4.12(b). In addition, the observables were preprocessed individually to be of zero-mean.

4.6.2 Objective I: Learning Features from the Observables

The first objective of the experiment is to learn the features that characterize the observables, using the cognit and the sparse coding algorithm side by side.

To be specific, we employed the cognit and the sparse coding algorithm to learn complex-valued features from the Dartmouth dataset; the dictionary size of both

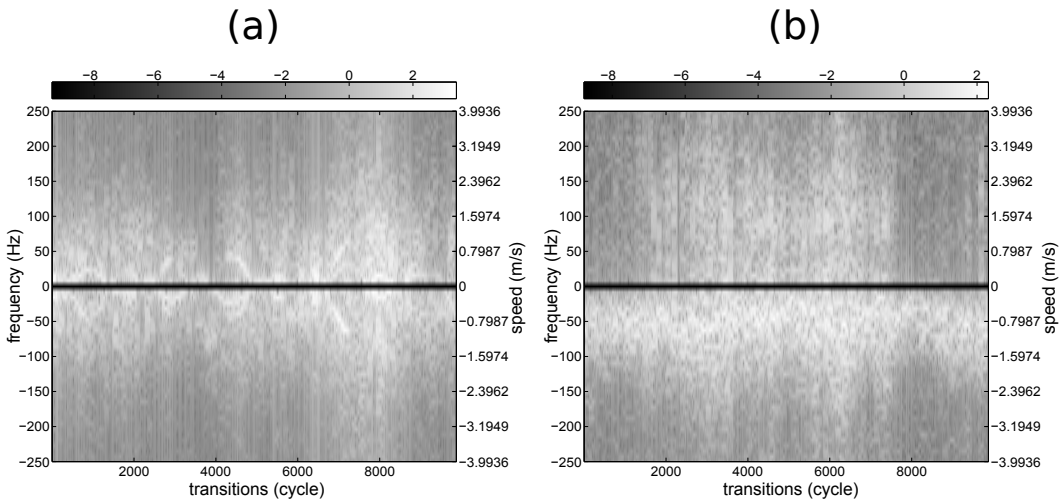


Figure 4.12: Amplitude spectra of observables from two range-bins: (a) target-plus-clutter (b) clutter.

algorithms was set to 100. The process of learning the two dictionaries occupied 10,000 epochs; in each epoch, the two dictionaries were trained using a mini-batch that contained 100 training examples. Half of the examples in every mini-batch were randomly selected from the bins containing target-plus-clutter and the other half were chosen randomly from the bins containing only clutter. Moreover, each training example consisted of 20 successive observables produced with one-step transitions in time, as previously described in Figure 4.12. Observables in a training example were preprocessed to be of zero-mean and unit variance. As for the learning rule, we used the formulas described in equations (4.29) and (4.30).

The cognit used the following state-space model:[9]

$$\begin{cases} z_{k+1} = z_k + \gamma_k \\ \\ y_k = W z_k^* + \omega_k \end{cases} . \tag{4.34}$$

The sparse coding algorithm was also designed with respect to the measurement equation described in equation (4.34).

The dictionary learned by the cognit is displayed in Figure 4.13; the elements of this dictionary resemble complex-valued wavelets and reflect underlying features

[9]The process noise, γ, and measurement noise, ω, were both chosen to be white, zero-mean, circular-symmetric, complex Gaussian processes. An approach for the identification of state-space model parameters is described in the next chapter.

of the radar returns. For discrimination between the physical sources of the learned features, the following points are addressed:

1) Due to the lower frequency content of target's signature, the features with a strong low-frequency component are attributed to echoes from the target.

2) By the same token, those features with a low-frequency envelope and high-frequency component may represent the returns of target-plus-clutter.

3) With higher certainty, the high-frequency features belong to clutter.

Note that the dictionary learned by the sparse coding algorithm follows a similar set of interpretations to those just described; hence, there is no practical benefit gained in repeating them.

Equipped with its own learned dictionary, the cognit denoises the observables. The sharpening effect of the denoising process on the amplitude spectra of observables is illustrated in Figures 4.14(a) and 4.14(b), which represent denoised versions of Figures 4.12(a) and 4.12(b), respectively.

Continuing on, a portion of sparse representations that are inferred by the cognit from successive observables of two range-bins are correspondingly displayed in Figures 4.15(a) and 4.15(b), which pertain to target-plus-clutter and clutter, respectively. The two new figures reveal that in general, representations of the bin containing target-plus-clutter show a higher number of active elements and greater amplitudes compared to the representations for clutter only.

It is noteworthy that similar results can be achieved by the sparse coding algorithm using its own learned dictionary; hence, repeating them is unnecessary.

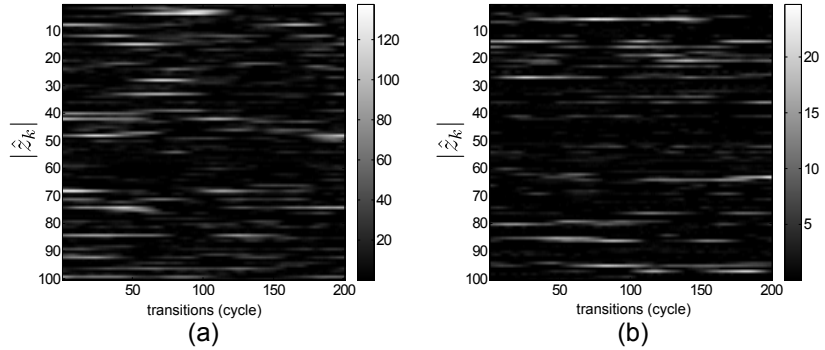


Figure 4.15: Sparse representations of successive observables inferred by the cognit: (a) target-plus-clutter (b) clutter.

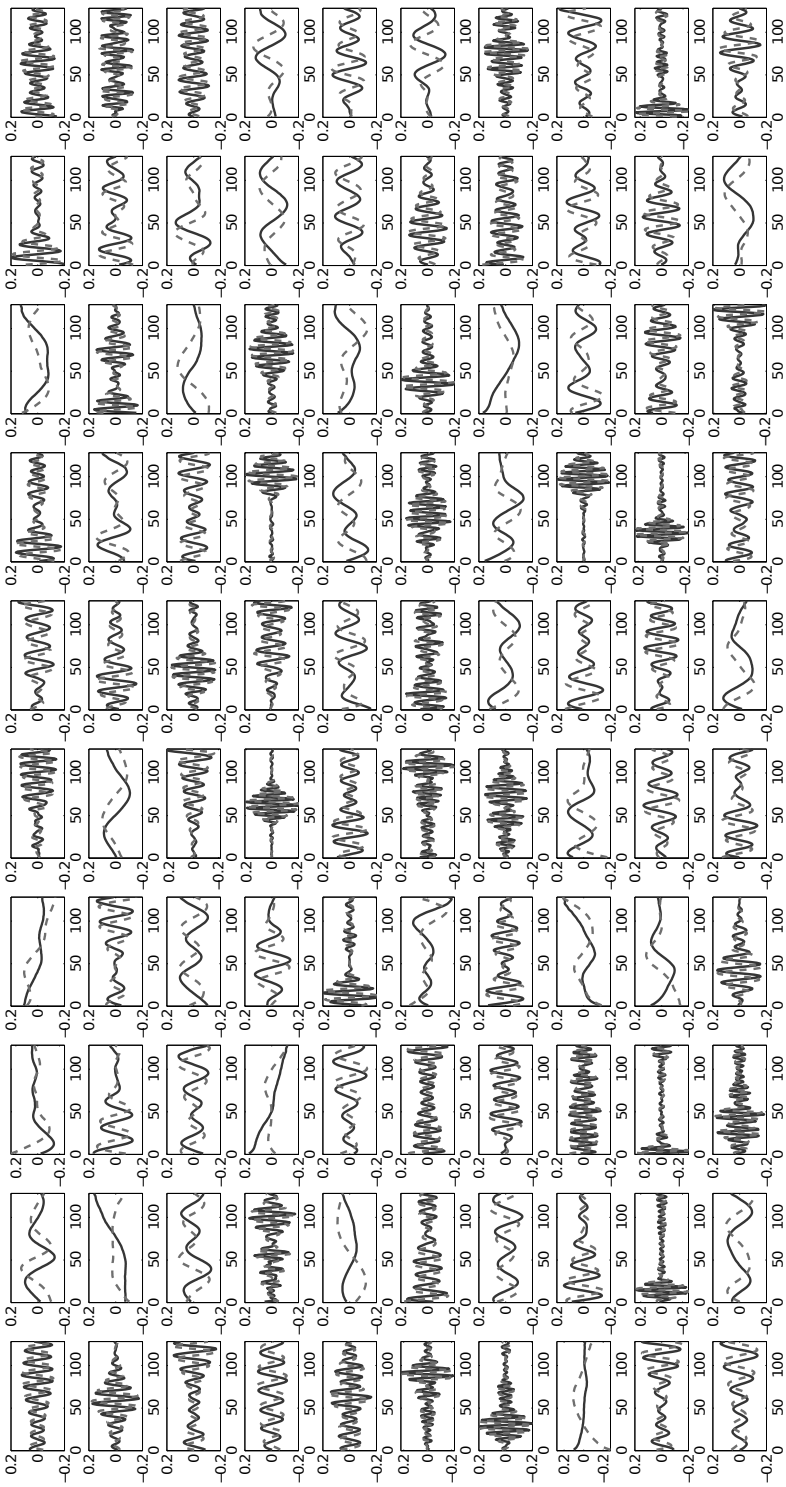


Figure 4.13: The dictionary elements learned by the cognit; the solid line is the real part and the dashed line is the imaginary part of the feature.

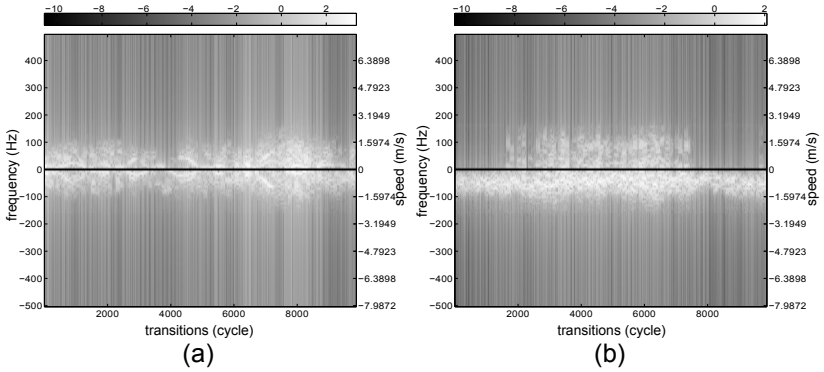


Figure 4.14: Amplitude spectra of denoised observables for: (a) target-plus-clutter (b) clutter.

4.6.3 Objective II: Testing the Stability of Representations

The second objective of this experiment is to test the stability of representations inferred by the cognit over time, compared to those representations inferred by the sparse coding algorithm.

To elaborate, our objective for adding local-PAC to sparse coding as the means of introducing perceptual attention, is to represent observables using relevant features while suppressing irrelevant features. Consequently, we may achieve a better stability of representations over time, where the emergence and disappearance of features in the observables result in comparably "smooth" variations in inferred representations.

The criterion, used for assessing stability of representations of the cognit and stability of representations of the sparse coding algorithm, was based on smoothness of the variations in those representations. To elaborate, we may say that the smoother the variations are in the representations, the smaller the power of high-frequency elements will be in the spectral density of those representations.

To this end, the short-time Fourier transform of representations was computed using a window-size of 128 samples with zero overlap between successive windows. Accordingly, the power spectrum of each window was calculated, and the mean spectral density (MSD) of representations was then computed using the power spectra of the whole set of windows. Figures 4.16(a) and 4.16(b) provide a comparison between the MSD of representations for the cognit and the sparse coding algorithm; the representations belong respectively to the bin containing target-plus-clutter and a bin with only clutter.

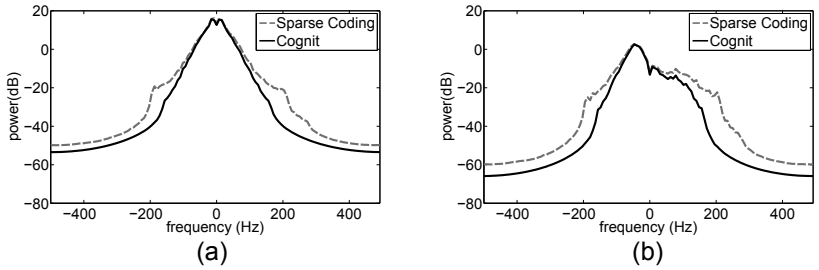


Figure 4.16: The MSD of representations for the cognit and the sparse coding algorithm: (a) target-plus-clutter (b) clutter.

As displayed in Figures 4.16(a) and 4.16(b), the MSDs of representations for the cognit show a lower power level for high-frequency elements when compared to the MSDs of representations for the sparse coding algorithm. Based on this comparison, we may go on to say that the variations in the representations happen more smoothly in the case of the cognit, which indicates that representations inferred by the cognit have better stability over successive time-steps. This is indeed a consequence of information feedback that flows through local-PAC of the cognit.

4.6.4 Objective III: Classifying the Representations

The final objective of the experiment is to use classification for the purpose of illustrating how the representations inferred by the cognit are superior to those inferred by the sparse coding algorithm.

To this end, a support vector machine (SVM) with radial-basis function (RBF) kernel (Chang and Lin, 2011) was used to classify the representations in two groups: target-plus-clutter, and clutter.

The SVM was first applied directly to the radar returns. Next, it was applied to the representations inferred by the sparse coding algorithm, using its own learned dictionary; the second result is denoted by *Sparse coding (I)*. For a different line of attack, the SVM was applied to the representations of the sparse coding algorithm, this time equipping it with the dictionary learned by the cognit; the third result is denoted by *Sparse coding (II)*. Finally, the SVM was applied to the representations inferred by the cognit. The classification results of the four categories just described are provided in Table 4.1.

Table 4.1: The classification results using SVM-RBF.

Radar returns	Sparse coding (I)	Sparse coding (II)	Cognit
83.6%	85.9%	88.1%	89.4%

Examining the results of Table 4.1, we make the following observations:

1) There is gain to be made in using sparse representations over raw data.

2) The dictionary learned by the cognit is superior to the one learned by the sparse coding algorithm. Hence, the features learned in the dictionary of the cognit have a better ability to represent underlying features of the observables; the improvement owes itself entirely to perceptual attention in the cognit.

3) The classification result for the cognit shows further improvement over all preceding results, since not only the cognit has learned a better dictionary, it has also benefited from perceptual attention in interpreting the observables. Simply put, the representations provided by the cognit convey more relevant information (or less irrelevant information) about the two original sources that generated the observables.

4.7 Summary

In this chapter, we tried to mimic functionality of perceptual attention through the use of an information filter rooted in Bayesian filtering paradigm. Accordingly, we managed to improve the performance of a single-layered perceptor under the influence of perceptual attention, the positive impact of which was supported by two challenging computer experiments.

The results, presented in this chapter, showed that under the influence of perceptual attention, a perceptor gains an improved ability to extract relevant features from sensory information while filtering away irrelevant features. Consequently, the perceptor may learn more efficiently and therefore perceive the environment better.

Chapter 5

Model Identification Through Expectation Maximization Algorithm

In Chapter 4, we described a method to improve the performance of a single-layered perceptor under the influence of perceptual attention, where state-space model (refer to equation (4.19)) was an indispensable part of that method. In this regard, a procedure is required to identify model parameters of the state-space model, hence the use of EM algorithm for model identification. This chapter is therefore devoted to describing the outline of EM algorithm. In relation to the experiment presented in Section 4.6, some of the results for model identification through EM algorithm are also provided in this chapter.

5.1 Model Identification

To be able to apply Bayesian filtering paradigm for the estimation of a stochastic process, dynamics of that stochastic process must be available, for instance in the form of a state-space model like the one described in the following equation:

$$\begin{cases} z_{k+1} &= f(z_k) + \gamma_k \\ y_k &= h(z_k) + \omega_k \end{cases} . \tag{5.1}$$

Accordingly, it is essential to have knowledge about model parameters, namely, $f(\cdot)$, $g(\cdot)$, and the pdfs for both ω and γ. However, this important piece of knowledge is not always available and a complementary process is therefore required to estimate those model parameters from sequences of observables, either prior to or alongside the filtering process. Such complementary processes are commonly known as *model identification*.

On this very topic, several approaches have adopted a *maximum likelihood* approach for model identification using an iterative scheme, e.g., using second-order methods like Gauss-Newton and Newton-Raphson in order to solve nonlinear equations that arise from differentiating a log-likelihood function (Dhrymes, Klein, and Steiglitz, 1970; Gupta and Mehra, 1974; Goodrich and Caines, 1979; Jones, 1980). However, there are disadvantages associated with such second-order methods like the need for computing the inverse of the Hessian matrix at each iteration, or the fact that successive iterations may not necessarily increase the log-likelihood function due complex nature of the maximization process.

On the other hand, a simple yet effective method for the iterative computation of maximum-likelihood is the expectation maximization (EM) algorithm, first introduced by Dempster, Laird, and Rubin (1977). Apart from simplicity of the steps involved, EM has the advantage of continuously increasing the likelihood and for an exponential family of distributions, it assuredly converges to a stationary point (Wu, 1983), which makes EM a desirable approach.

5.2 Outline of EM Algorithm

In the Bayesian filtering paradigm, the state of a stochastic process is estimated by finding the point that maximizes the probability of the posterior distribution of that stochastic process, i.e., the MAP estimate;[1] for a stochastic process z, the MAP estimate is often denoted by z^{MAP}. Hence, for time-step k, where the posterior distribution is available given the sequence of observables Y_k, we may derive z^{MAP} as follows:

$$z_k^{\text{MAP}} = \arg\max_z \; p(z_k|Y_k, M^0),$$

where M^0 is a set of model parameters used to initialize the filtering process,

$$M^0 = \{f^0(\cdot),\; h^0(\cdot),\; R^0,\; Q^0, \mu_0^0, P_0^0\}.$$

The EM algorithm builds on a similar approach but from a different direction. In other words, the algorithm tries to find the model parameter M^* that maximizes the likelihood of observables,

$$M^* = \arg\max_M \; p(Y|M),$$

[1] When the posterior distribution of a stochastic process is Gaussian, the expected value of that process and the MAP estimate are actually the same.

where Y is the set of all possible observables. The same result may also be achieved by maximizing the log-likelihood function, defined by

$$\begin{aligned}\ell(M) &= \log p(Y|M) && (5.2)\\ &= \log \int_Z p(Y, Z|M)\mathrm{d}Z && (5.3)\\ &= \log \int_Z \Delta(Z)\frac{p(Y, Z|M)}{\Delta(Z)}\mathrm{d}Z, && (5.4)\end{aligned}$$

where Z is the set of all possible states and $\Delta(Z)$ is any pdf that describes Z . Using Jensen's inequality, we may now continue to write,

$$\begin{aligned}\ell(M) &\geq \int_Z \Delta(Z) \log \frac{p(Y, Z|M)}{\Delta(Z)}\mathrm{d}Z\\ &= \int_Z \Delta(Z) \log p(Y, Z|M)\mathrm{d}Z \qquad (5.5)\\ &\quad - \int_Z \Delta(Z) \log \Delta(Z)\mathrm{d}Z,\end{aligned}$$

which clearly shows that the right-hand side of the equation is the negative of Kullback–Leibler divergence (relative entropy). Alternatively, as pointed out in (Haykin, 2001), the right-hand side of equation (5.6) may be described in terms of *free energy* (Friston, 2010). Here, free energy $F(\Delta, M)$ is defined by:

$$\begin{aligned}F(\Delta, M) &= -\int_Z \Delta(Z) \log p(Y, Z|M)\mathrm{d}Z + \int_Z \Delta(Z) \log \Delta(Z)\mathrm{d}Z && (5.6)\\ &= \mathbb{E}_\Delta[-\log p(Y, Z|M)] - \mathcal{H}_\Delta(Z), && (5.7)\end{aligned}$$

where $\mathbb{E}_\Delta[\cdot]$ is the expectation operator over $\Delta(Z)$, and $\mathcal{H}_\Delta(Z)$ is the entropy of $\Delta(Z)$. Accordingly, we may rewrite (5.6) as follows:

$$\ell(M) \geq -F(\Delta, M). \qquad (5.8)$$

The right-hand side of inequality (5.8) plays the role of a lower-bound for the log-likelihood function. Hence, maximizing the lower-bound will have the same effect as maximizing the log-likelihood function itself, which is also the same as minimizing the free energy. Since the lower-bound is a function of Δ and M, the maximization process is carried out by iteratively alternating between two steps, namely, *expectation*, and *maximization*.

5.2.1 Step 1 - Expectation

Assuming that we are in i^{th} iteration of the algorithm, the expectation step is expressed as follows:[2]

$$\begin{aligned}
\Delta^{i+1} &= \arg\max_{\Delta} \ \left[-F(\Delta, M^i)\right] \\
&= \arg\max_{\Delta} \ \left[\int_Z \Delta(Z) \log \frac{p(Y, Z|M^i)}{\Delta(Z)} \mathrm{d}Z\right] \\
&= \arg\max_{\Delta} \ \left[\int_Z \Delta(Z) \log \frac{p(Z|Y, M^i)p(Y|M^i)}{\Delta(Z)} \mathrm{d}Z\right] \\
&= \arg\max_{\Delta} \ \left[\int_Z \Delta(Z) \log \frac{p(Z|Y, M^i)}{\Delta(Z)} \mathrm{d}Z + \log\ p(Y|M^i)\right].
\end{aligned}$$

Considering the fact that $\log\ p(Y|M^i)$ does not influence the maximization process, it can be quickly inferred from the equations above that the maximum value is achieved at $\Delta(Z) = p(Z|Y, M^i)$. Hence, solution of the expectation step is directly computed through a procedure referred to as *smoothing*, which is described next.

Bayesian Smoothing

Bayesian smoothing of a hidden stochastic process z builds on the same principles used in Bayesian filtering, presented earlier in Section 3.2. Assuming we have a sequence of N observables, denoted by Y_N (refer to equation (3.13)), smoothing is accomplished by finding the pdf $p(z_k|Y_N)$ for all k, where $k < N$.

This problem may therefore be solved using two approaches, namely,

i) **Two-filter smoothing** (Fraser and Potter, 1969), where smoothing is performed by combining the results of two separate filtering processes, one from $k = 0$ to $k = N$, and the other backward in time, from $k = N$ to $k = 0$.

ii) **Forward-backward smoothing** (Rauch, Striebel, and Tung, 1965), which is more computationally efficient. This approach is formulated as follows:

$$p(z_k|Y_N) = \int p(z_k, z_{k+1}|YN)\mathrm{d}z_{k+1} \tag{5.9}$$

$$= \int p(z_{k+1}|Y_N)p(z_k|z_{k+1}, Y_N)\mathrm{d}z_{k+1}. \tag{5.10}$$

[2]It is noteworthy that at iteration $i = 0$, the EM algorithm starts with M^0, described earlier in Section 5.2.

It is important to note that z is assumed to be a Markov process. Hence, given z_{k+1} and Y_k, z_k is independent of any observable y_t where $k < t$ given. Consequently, we may write,

$$p(z_k|Y_N) = \int p(z_{k+1}|Y_N)p(z_k|z_{k+1}, Y_k)\mathrm{d}z_{k+1}. \tag{5.11}$$

Then using Bayes rule, equation (5.11) may be expressed as follows:

$$p(z_k|Y_N) = \int p(z_{k+1}|Y_N)\frac{p(z_{k+1}|z_k)p(z_k|Y_k)}{p(z_{k+1}|Y_k)}\mathrm{d}z_{k+1} \tag{5.12}$$

$$= p(z_k|Y_k)\int \frac{p(z_{k+1}|Y_N)p(z_{k+1}|z_k)}{p(z_{k+1}|Y_k)}\mathrm{d}z_{k+1}. \tag{5.13}$$

Equation (5.11) clearly shows that in order to perform smoothing, two steps should be completed: First, posterior distribution $p(z_k|Y_k)$ and predicted distribution $p(z_{k+1}|Y_k)$ should be computed from $k = 0$ to $k = N$; this step is referred to as *forward filtering pass*. Then, $p(z_k|Y_N)$ may be iteratively computed backwards, from $k = N$ to $k = 0$, using equation (5.13).

5.2.2 Step 2 - Maximization

Having $\Delta^{i+1}(Z)$ available from the expectation step, model parameters may now be updated to maximize $\ell(M)$. Hence, we have

$$\begin{aligned} M^{i+1} &= \arg\max_{M} \ \left[-F(\Delta^{i+1}, M)\right] \\ &= \arg\max_{M} \ \left[\int_Z \Delta^{i+1}(Z)\log p(Y, Z|M)\mathrm{d}Z + c\right], \end{aligned}$$

where c is a constant representing the entropy of $\Delta^{i+1}(Z)$ and therefore has no effect on the value of M^{i+1}. Accordingly, we may write

$$M^{i+1} = \arg\max_{M} \ \left[\int_Z p(Z|Y, M^i)\log p(Y, Z|M)\mathrm{d}Z\right].$$

EM algorithm is then iteratively performed by updating values of Δ and M from one iteration to the next. This process is continued until convergence is reached.

Note: Algorithmic steps of EM algorithm, under linearity and Gaussianity assumptions, are described in Appendix B.

5.3 Results of EM Algorithm

In the experiment that was presented in Section 4.6, EM algorithm was used for model identification, the outline of which is already described. We now present some of the results for model identification that are computed through the use of EM algorithm; these results correspond to the experiment in Section 4.6.

Figure 5.1 trace of the covariance matrices for measurement noise and process noise, where tr$[\cdot]$ denotes trace of a matrix, and $\hat{R}$ and $\hat{Q}$ are covariance matrices that are estimated for the measurement noise and process noise, respectively. As shown in the figure, the two matrices, $\hat{R}$ and $\hat{Q}$, are first initialized with large values for their diagonal elements. After successive iterations of the EM algorithm, they converge to their final values. Figure 5.2 displays absolute values of $\hat{R}$ and $\hat{Q}$ that are estimated using the EM algorithm.[3]

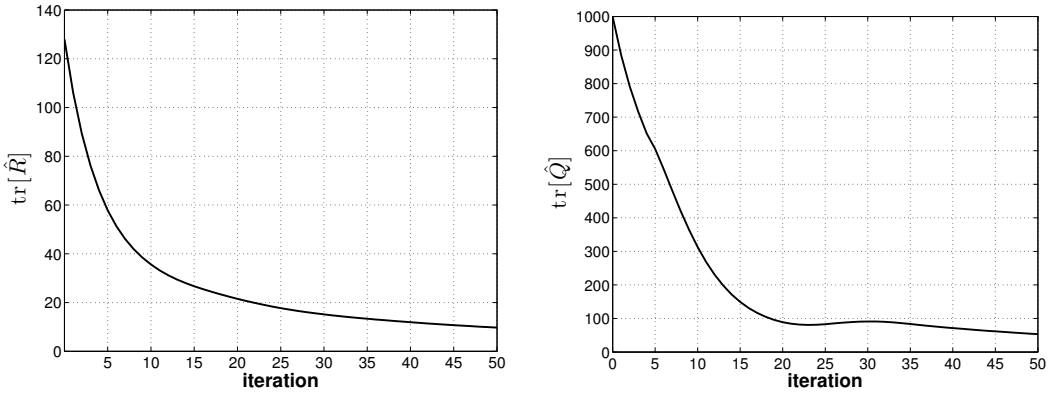


Figure 5.1: (Left) Trace of the estimated covariance matrix for measurement noise, denoted by $\hat{R}$, computed over successive iterations of EM algorithm. (Right) Trace of the estimated covariance matrix for process noise, denoted by $\hat{Q}$, computed over successive iterations of EM algorithm.

Figure 5.3 illustrates the impact of perceptual attention on sensory information. As shown in the figure, the purifying impact of perceptual attention on sensory information results in gradually increasing the contrast between relevant information and irrelevant information. Consequently, irrelevant information is filtered away and relevant information is preserved.

[3] $\hat{R}$ and $\hat{Q}$ are complex-valued. Hence, for the purpose of visualization, absolute values of their elements have been used.

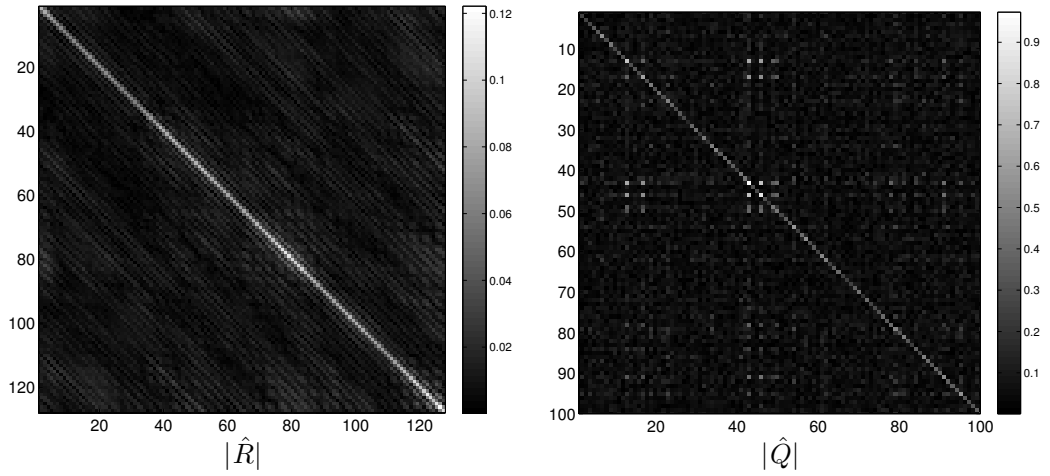


Figure 5.2: Absolute values of covariance matrices: (Left) Estimated covariance matrix for the measurement noise, $\hat{R}$. (Right) Estimated covariance matrix for the process noise, $\hat{Q}$.

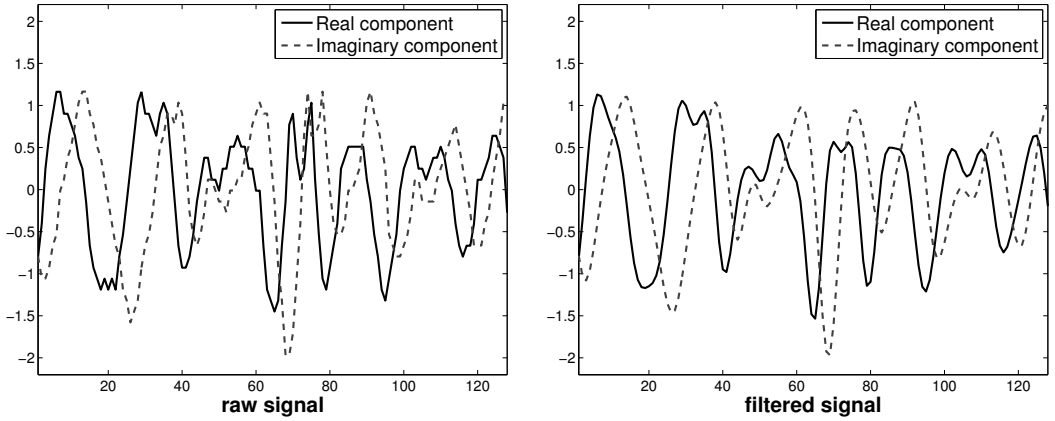


Figure 5.3: (Left) Raw signal received from radar. (Right) Filtered signal under the influence of perceptual attention.

Chapter 6

Improving Performance of a Hierarchical Perceptor Under the Influence of Attention

In Chapter 4, we described a method to improve the performance of a single-layered perceptor under the influence of perceptual attention. In this chapter, we take one step further and consider the case where perceptor is hierarchical (refer to Section 1.3). The goal is therefore to investigate how we may improve the performance of a hierarchical perceptor by benefitting from the positive impact of perceptual attention. The hierarchical framework that we choose is of a convolutional type, a description of which is presented next in Section 6.1. Accordingly, we consider two different approaches in this chapter for making a convolutional hierarchy. These two approaches are described as follows:

1. In the first hierarchy, where learning is *unsupervised*, sparse coding is applied in a hierarchical manner to learn and extract features from data. We denote this hierarchical perceptor by convolutional sparse coding network, simply referred to as **CSCN**, hereafter.

2. The second convolutional hierarchy learns features from data in a *supervised* manner. Computational units of this hierarchy are perceptrons. Hence, the second hierarchical perceptor is a standard convolutional network. Hereafter, the second hierarchy is referred to as **CN**.

Based on the two approaches, just described, an experiment is then carried out in this chapter, once again using real-life radar data that was described earlier in Section 4.6; except, the set-up of the experiment is more challenging than the one

presented in that section. The goal of this experiment is to evaluate the impact of attention on performances of these two approaches.

6.1 Overview of Convolutional Networks

Following the descriptions provided previously in Section 1.4, we briefly sketch the standard configuration of convolutional networks (LeCun and Bengio, 1995). As displayed in Figure 6.1, using an *encoder* (feature detector) with a kernel-size that is basically much smaller than dimensionality of input data, a patch from the input data is encoded to a N_1-dimensional vector. Then, by moving receptive field of the encoder over the input data, just like the way in which traditional convolution is done, other patches from the input data are encoded as well. As a result, N_1 feature-maps are computed that provide low-level features of the input data.

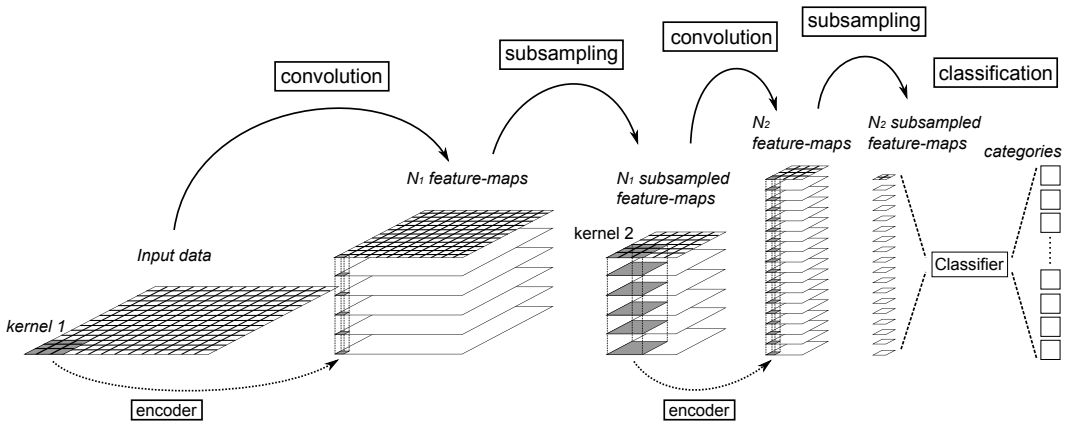


Figure 6.1: A convolutional network with two convolutional layers, two subsampling layers, and a classification layer, aimed at processing 2-dimensional input data.

The size of the feature-maps are then reduced through subsampling, hence making the network invariant to slight displacements in input patterns. Then, the subsampled feature-maps are used by the next convolutional layer and N_2 new feature-maps are generated accordingly. These new feature-maps capture relative relationships that exist among low-level features of the first layer. Therefore, the features extracted by the second layer are more abstract compared to those extracted by the first layer.

The most common encoder that is used in a convolutional network is a perceptron. Thus, the network as a whole may be described as a sophisticated multilayer perceptron, the elements of which may be trained using backpropagation. However, using perceptron as the encoder necessitates supervised training of the network and therefore requires abundance of labelled data for supervised learning. As a result, not having enough labelled data becomes a challenge and impairs the performance of the network. To this end, other approaches have been proposed to realize unsupervised training of the network, which is then followed by a supervised fine-tuning procedure (Ranzato, Huang, Boureau, and Lecun, 2007; Jarrett et al., 2009; Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, and Cun, 2010).

In this chapter, we focus on two approaches for making a convolutional hierarchy, namely, CSCN and CN; these two approaches are then modified to function under the influence of perceptual attention. Details follow in the next two sections.

6.2 Approach 1: CSCN and the Influence of Perceptual Attention

6.2.1 CSCN

As described earlier in this chapter, hierarchical structure of CSCN is based on a standard convolutional network that was described in previous section. The only difference between the two is that unlike a standard convolutional network, where encoders are perceptrons, the encoders in convolutional layers of CSCN are sparse coding units.[1]

The way sparse coding units function in CSCN may be understood better by referring back to Figure 6.1, which was depicted earlier. To elaborate, each unit processes a particular chunk from feature maps of previous layer. Then, it encodes that chunk to sparse representations. Representations from all sparse coding units will therefore collectively form a new set of feature maps, ready to be processed by the next layer.

[1] In order to perform ℓ^1-regularized optimization tasks that are required in CSCN, fast iterative shrinkage algorithm (FISTA) proposed by Beck and Teboulle (2009) is adopted.

6.2.2 CSCN - Perceptual Attention

To add the influence of perceptual attention to CSCN, its structure needs to be modified. In this regard, we refer back to Chapter 4, where we proposed an algorithm for improved sparse coding, denoted by cognit. Thus, for CSCN to function under the influence of perceptual attention, it is modified simply by replacing each one of its sparse-coding units with a cognit. The end result will be a hierarchical perceptor with multiple feedback loops, which carry out local perception-action cycles in a distributed manner.

To simplify reference, we denote this new hierarchical perceptor by **CSCN-PA**, where PA stands for perceptual attention.

6.3 Approach 2: CN and the Influence of Perceptual Attention

6.3.1 CN

Structure of CN is based on the configuration of a standard convolutional network, which was described earlier in Section 6.1.

6.3.2 CN - Perceptual Attention

For CN to function under the influence of perceptual attention, its structure should be modified, which is described next. To simplify reference, the end result will be referred to as **CN-PA**.

In CN, the encoder in a convolutional layer is a perceptron, where we denote the activities in its receptive field by y. Needless to say, measurement noise, denoted by ω, overrides the activities in that receptive field. Thus, for the input-output relationship of that perceptron, we may write:

$$o = \phi\left\{W(y+\omega)+b\right\} \tag{6.1}$$

where o is the vector of output activations, W represents connection wights, b denotes bias elements stacked as a vector, and $\phi\{\cdot\}$ is an element-wise activation function (e.g., a stack of sigmoid functions working separately on elements of a vector).

We now define a simple state-space model similar to the one used for cognit in Section 4.3, which is expressed as follows:

$$\begin{cases} z_{k+1} = z_k + \gamma_k \\ \\ z_k = W(y_k + \omega_k) + b \end{cases} \tag{6.2}$$

where z is the state variable of interest, the expected value of which is used for computing output activities of the perceptrons (i.e., $o = \phi\{\mathbb{E}[z]\}$); γ is the process noise, and the index k denotes time. To simplify equation (6.2), we define two new variables $x = Wy + b$ and $\zeta = W\omega$. Thus, we may write:

$$\begin{cases} z_{k+1} = z_k + \gamma_k & \text{process equation} \\ \\ z_k = x_k + \zeta_k & \text{measurement equation} \end{cases} \tag{6.3}$$

Here, we are interested in the expected value of z to compute activities of perceptrons. As a result, Kalman filter will be an appropriate choice for carrying out filtering cycles in CN-PA, and therefore, it is adopted for the algorithmic implementation of perceptual attention. Accordingly, to derive CN-PA, the structure of CN is modified by replacing its simple encoders, i.e., perceptrons, by new encoders that are utilized with Kalman filter, just described. The end result of this modification will again be a hierarchical perceptor with multiple feedback loops, which carry out local perception-action cycles in a distributed manner.

6.4 Computer Experiment

The experiment is carried out with two main objectives in mind, namely, how accurately the location of target is estimated,

i) at every point in time (i.e., target detection), and

ii) over successive radar measurements (i.e., target tracking).

With these two objectives in mind, the experiment is carried out in three steps:

1) During Step 1, the two hierarchical perceptors from Approaches 1 and 2, namely, CSCN and CN, are trained and tested with radar data to detect and track the floating target that is buried in sea clutter.

2) In Step 2, the impact of perceptual attention is put to the test. Accordingly, CSCN-PA and CN-PA from Approaches 1 and 2, which function under the influence of perceptual attention, are trained and tested again for the tasks

of target detection and tracking, similar to the way in which it is was done in previous step. The impact of perceptual attention is then evaluated by comparing the results of Step 2 to those achieved in Step 1.

3) In the third step of the experiment, radar returns are degraded with additive measurement noise. Correspondingly, performances of all four hierarchical perceptors, trained in Steps 1 and 2, are evaluated at different levels of noise power. Thereby, the impact of perceptual attention on reducing sensitivity of perceptors to measurement noise is examined.

To provide a measure of comparison for the results achieved in Steps 1 and 2, a classic approach to target detection and tracking is also applied to the data of this experiment. The classic approach is based on target detection using Doppler spectrum amplitude, and target tracking using a particle filter.

6.4.1 Preparation of Training and Testing Examples

To perform the experiment, it is required to divide the Dartmouth dataset into training and testing sets. To this end, the first 90,000 samples recorded in files of the dataset are assigned to the training set, and the next 40,000 samples in those files are assigned to the testing set.

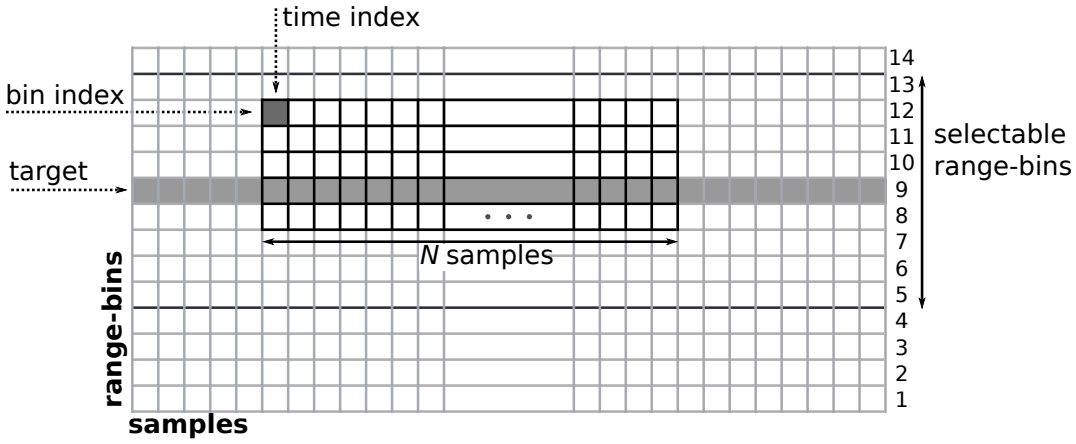


Figure 6.2: Selecting samples from 5 neighbouring range-bins and with a frame-size of $N = 64$ samples. Time index and bin index are selected randomly. Range-bins are selected in a way that presence of the target in all selected range-bins is uniformly distributed over training examples.

In Step 1, training and testing examples are randomly selected using a fixed frame-size, hence picking samples for a desired time-duration from a number of neighbouring range-bins. To avoid overfitting, those range-bins are selected in a way that they all have the same probability of containing the target. To this end, the samples in our experiment were selected from 5 neighbouring range-bins and for a duration of $N = 64$ samples, as pictured in Figure 6.2.

In Step 2 of the experiment, examples are selected in a manner that is essentially similar to the one just described for Step 1. Except, in Step 2, we are trying to take advantage of dependency of successive input frames through perceptual attention. Thus, instead of N samples, $N + M - 1$ successive samples are selected from neighbouring range-bins, where M is the number of successive transitions of input frame in time; in our experiment, M was set to 50. Figure 6.3 displays how successive input frames are selected and processed.

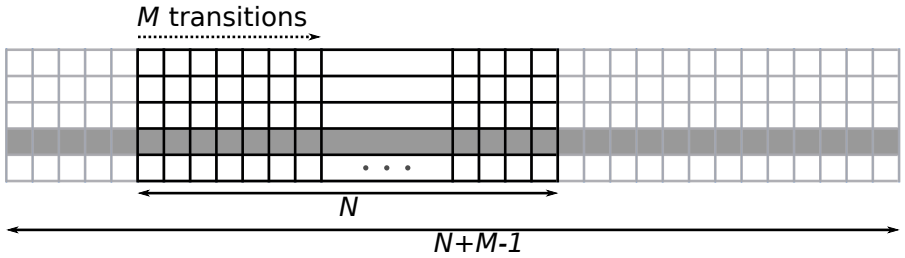


Figure 6.3: Successive transitions of input frames over recorded samples of radar returns.

6.4.2 Training Phase

In Step 1 of the training phase, both CSCN and CN are trained using mini-batches (50 examples). Training examples of those mini-batches are randomly selected from the training set and are used only once during a complete run over the whole training set, which hereafter is denoted by a training *epoch*. Hence, each epoch involves $1800 = 90\text{k}/50$ mini-batches. Then, CSCN and CN are trained for sufficient number of epochs until convergence is achieved for them both.

In Step 2, each training example consists of $M = 50$ successive frames of samples (Figure 6.3). Nevertheless, training process of CSCN-PA and CN-PA is fundamentally the same as that of CSCN and CN in Step 1, in that, weight updates are computed and applied only once. The weight update happens at frame of 50.

Another important matter in Step 2 is setting parameters of the state-space models used by CSCN-PA, CN-PA; these model parameters are the covariance matrices of the process noise and measurement noise. In this regard, an expectation maximization (EM) approach, described earlier in Chapter 5, is adopted to identify model parameters.

6.4.3 Testing Phase

In the testing phase, performances of the perceptors trained in Steps 1 and 2 of the training phase are evaluated, where classification error is used as the performance measure. Testing examples are selected in a manner similar to the way in which it was done in the training phase (Figures 6.2 and 6.3). When testing performances of CSCN-PA and CN-PA, which require M successive input frames (transitions), only those classification errors that correspond to M^{th} frame are recorded for the evaluation of performance (Figure 6.4). The state-space models of these two perceptors are initiated in the testing phase with the model parameters that were previously identified in the training phase.

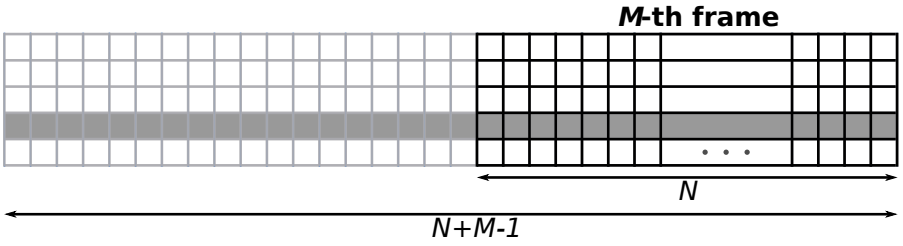


Figure 6.4: M^{th} frame of samples, according to which classification errors are computed in Step 2 of the experiment.

6.4.4 Structural Configurations of the Perceptors

We set up all four perceptors using similar configurations composed of an input layer, denoted by I, three convolutional layers, denoted by C, and a linear classifier as the final layer that identifies which one of the five range-bins contains the target. Each convolutional layer uses a kernel size of 1×22 samples to extract features from the inputs it receives. Dictionaries that are used to set up 1st, 2nd, and 3rd convolutional layers of CSCN and CSCN-PA consist of 12, 24, and 36 feature vectors, respectively. In a similar manner, CN and CN-PA are set up with 12, 24, and 36 feature detectors in their convolutional layers.

Structural configurations of the perceptors may be alternatively summarized as follows:

$$I: \begin{bmatrix} 5 \\ \times \\ 64 \end{bmatrix} \xrightarrow[1 \times 22]{C,\ 12} \begin{bmatrix} 5 \\ \times \\ 43 \\ \times \\ 12 \end{bmatrix} \xrightarrow[1 \times 22]{C,\ 24} \begin{bmatrix} 5 \\ \times \\ 22 \\ \times \\ 24 \end{bmatrix} \xrightarrow[1 \times 22]{C,\ 36} \begin{bmatrix} 5 \\ \times \\ 1 \\ \times \\ 36 \end{bmatrix} \xrightarrow{\text{Classifier}} \begin{bmatrix} 5 \end{bmatrix}$$

6.5 Results of the Experiment

6.5.1 Learning Rate

Figure 6.5 illustrates mean squared error (MSE) of classifiers pertaining to CN and CSCN. By the error of classifier we mean the difference between classifier's actual output and its corresponding desired output. Mean values are calculated by averaging squared errors over training examples of each mini-batch.[2]

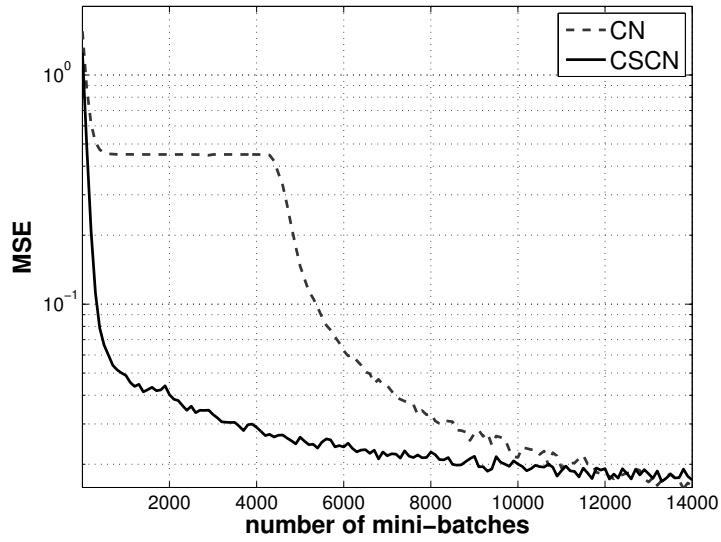


Figure 6.5: Mean squared error of classifiers for CN and CSCN.

[2]For better visualization, MSE curves are smoothened using the expression: $\text{MSE}_k = 0.99\,\text{MSE}_{k-1} + 0.01\,\text{MSE}_k^o$, where MSE_k^o is the observed MSE value for k^{th} mini-batch.

Figure 6.5 clearly shows that CN has a slower learning rate than that of CSCN. It is noteworthy that learning in CN is based on backpropagation, which is a global learning method, in that, it trains a perceptor as a single system. Thus, as a perceptor grows *deep*, i.e., the number of layers increases, learning rate that is achieved through backpropagation degrades and becomes slower. It is then required to expose the perceptor to more training examples until the learning curve leaves *plateau* and starts decreasing once again, just as it is pictured in Figure 6.5. It is noteworthy that the extreme case of this phenomenon is known as the "vanishing gradient" problem, which is especially well-known in the context of recurrent neural networks.[3]

To mitigate the problem, just described, different strategies have been applied in the literature on machine learning, which may accelerate the learning rate of a perceptor, if adjusted properly. Some of these methods include, adding a *momentum* term to weight updates (Nesterov, 1983; Rumelhart et al., 1986; Møller, 1993), adapting learning rates individually for each parameter in the structure (Jacobs, 1988; Riedmiller and Braun, 1993; Schaul, Zhang, and LeCun, 2013), and the application of second order methods (Becker and LeCun, 1989). These methods have shown to be very helpful in improving the convergence rate of the backpropagation algorithm. Yet, they cannot eliminate the negative impact of increased structural depth on the learning rate, which makes the task of training a deep neural structure using backpropagation, a challenging one, with no simple recipe.

Coming back to Figure 6.5, the learning curve of CSCN shows an asymptotic decrease until convergence is reached, without emergence of a plateau in between. This is due to localized learning processes in CSCN, through which, constituting components of CSCN are trained in parallel, in a greedy manner, hence accelerating the training process. Consequently, we may go on to say that localized learning is vital for the training of deep structures, if we are interested in fast convergence.

[3]Backpropagation through time is a standard method for training recurrent neural networks. In this method, a recurrent network is first unfolded in time to form a feedforward network. Then, layers of that network are trained using backpropagation algorithm. When connection weights are small, which often they are, the error signal becomes smaller as it propagates back layer after layer, and it therefore becomes less effective for the modification of weights in earlier layers, hence the terminology: the vanishing gradient problem (Haykin, 2009). Note that another case, which is exactly the opposite, may also occur when connection weights are relatively large. Then, instead of vanishing, the error signal explodes during backpropagation and deletes everything that was learned before in connection weights of previous layers.

The point, just addressed, is consistent with the results achieved by (Hinton and Salakhutdinov, 2006; Ranzato, Poultney, Chopra, and LeCun, 2006; Bengio, 2009). In those studies, constituting layers of a deep perceptor are first trained locally, in a greedy manner. Then, the whole structure is fine-tuned using backpropagation.

6.5.2 Target Detection

In this part, results that were achieved for the task of target detection are presented. The results pertain to Approaches 1 and 2, as well as to the classic approach, which were all described earlier in this chapter. The goal has been to correctly identify the range-bin that contained the target.

Approach 1: CSCN and CSCN-PA

Table 6.1 displays the results of Approach 1, where performances of CSCN and CSCN-PA were evaluated. The results show an improvement for CSCN-PA, the operation of which is supported by the influence of perceptual attention, over CSCN which acts on its own and therefore without the influence of perceptual attention. This improvement clearly shows the positive impact of perceptual attention.

Table 6.1: Approach 1: Results on target detection.

Method	Classification error
CSCN	13.6±1.3%
CSCN-PA	11.4±1.1%

Approach 2: CN and CN-PA

The results of Approach 2 are described in Table 6.2. In a similar manner, the results show that the influence of perceptual attention improved the performance of CN-PA over that of CN, which operates on its own. This once again manifests the positive impact of perceptual attention.

Table 6.2: Approach 2: Results on target detection.

Method	Classification error
CN	10.2±0.7%
CN-PA	7.6±0.7%

Comparing the results shown in Tables 6.1 and 6.2, we readily see that in Approach 2, where the training of encoders is supervised, a higher performance level is reached compared to the one achieved through Approach 1. It is noteworthy that in Approach 1, encoders of the the two perceptors, namely, CSCN and CSCN-PA, are self-organized feeding only on sensory information. Thus, those encoders may not be trained by taking advantage of the labels of training data.

Nevertheless, the intended purpose of this experiment is not to compare the performances of Approaches 1 and 2 against one another. On the contrary, our intention is to demonstrate the positive impact of attention on the performance of a perceptor, the validity of which is supported by the results of two different approaches.

Classic Approach

To provide a measure of comparison for the results achieved by Approaches 1 and 2, we applied a classic approach to the radar data of this experiment. This classic approach consists of a target detector (TD) that computes amplitude spectrum of radar returns and then performs threshold check to detect target's location. In order to take advantage of temporal dependency of radar returns, a particle filter (PF) (Gordon, Salmond, and Smith, 1993), may also be used in combination with TD, where PF is supplied with outputs of TD. Accordingly, a strategy of detection while tracking is pursued, where output of PF is used to define the range-bin that contains the target. The results are shown in Table 6.3.

Table 6.3: Results for a classic approach.

Method	Classification error
TD	85.9%
TD + PF	69.1%

As it is readily seen in Table 6.3, TD has dramatically failed in detecting correct location of the target. Since we use a frame-size of 64 samples, which is relatively small, computed amplitude spectra for the samples taken from each range-bin do not provide enough information to distinguish between target and clutter. This is due to passive motion of the floating target, causing its Doppler frequency to be close to that of clutter. In addition, reflections from sea-waves in this particular example are almost in the same range of amplitude as those from the target. As a result, checking amplitude versus a prescribed threshold, as it is done in TD, may not lead to acceptable results, especially when a small frame of samples is picked.

Table 6.3 shows that by taking advantage of temporal dependency of observables, as it is realized through PF, performance of the classic approach is improved to some limited extent. However, the performance level that is achieved through this classic approach is still nowhere near the ones pertaining to Approaches 1 and 2.

6.5.3 Pseudo Target-tracking

Figure 6.6 depicts tracks of the target estimated by the approaches used in this report. The tracks in the top row of Figure 6.6 are generated by processing successive frames independently. On the other hand, the second row displays how precision of those tracks are improved in CN-PA and CSCN-PA due to the influence of perceptual attention. It is noteworthy that in these images, the target is anchored in range-bin number 3.

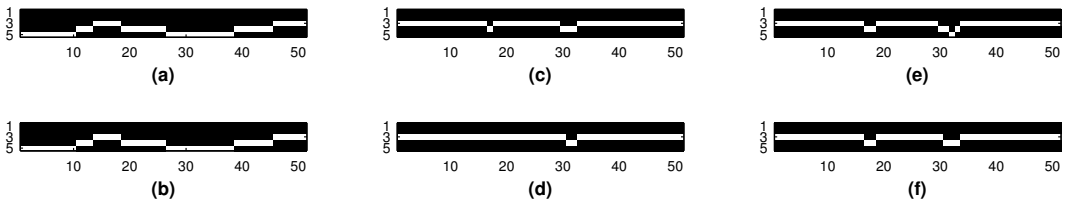


Figure 6.6: Track of the target estimated over 50 successive frames of samples: (a) TD, (b) TD+PF, (c) CN, (d) CN-PA, (e) CSCN, (f) CSCN-PA. The target is anchored in bin number 3.

6.5.4 Sensitivity to Measurement Noise

In practical applications, sensory information is often acquired with overriding uncertainties, e.g., measurement noise. Hence, a desirable property of a percpetor is to withstand degrading effect of uncertainties. For this purpose, an experiment is performed with measurement noise playing the role of uncertainties in observables. Accordingly, performances of the perceptors pertaining to Approaches 1 and 2 are evaluated in the face of different levels of overriding noise.[4] Thereby, sensitivity

[4]Noisy data are synthesized by adding complex-valued circular-symmetric Gaussian noise to radar samples, with the noise being independently and identically distributed.

of each perceptor is assessed and the impact of perceptual attention on sensitivity is examined.

Approach 1: CSCN and CSCN-PA

Figure 6.7 shows degradations in performance of CSCN in response to induced measurement noise. In addition, the figure displays how the degradations are mitigated in CSCN-PA due to the influence of perceptual attention. Accordingly, we may go on to say that the perceptor has become less sensitive to degrading effect of uncertainties caused by the measurement noise, which owes itself completely to the influence of perceptual attention.

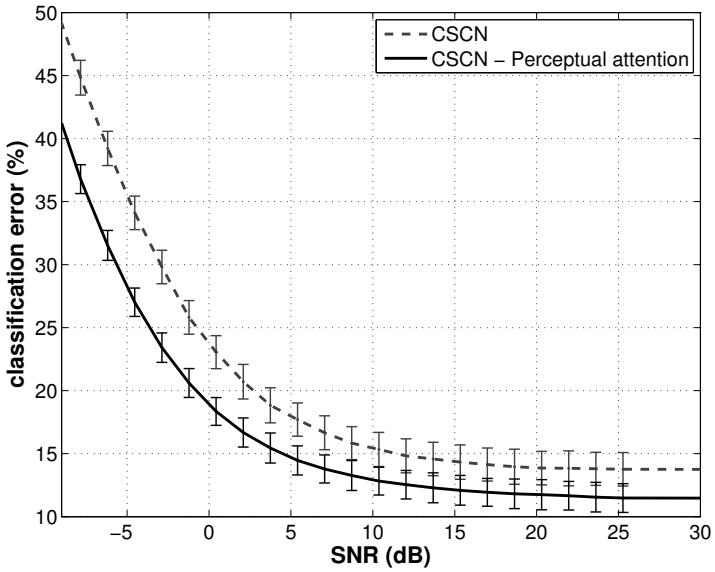


Figure 6.7: Effect of perceptual attention on performance of CSCN in the presence of noise.

Approach 2: CN and CN-PA

In a similar manner, Figure 6.8 illustrates degradations in performance of CN with respect to SNR of induced measurement noise. It also displays how those degradations vary in CN-PA, which functions under the influence of perceptual attention. It is readily seen from Figure 6.8 that perceptual attention reduces sensitivity of CN-PA to measurement noise.

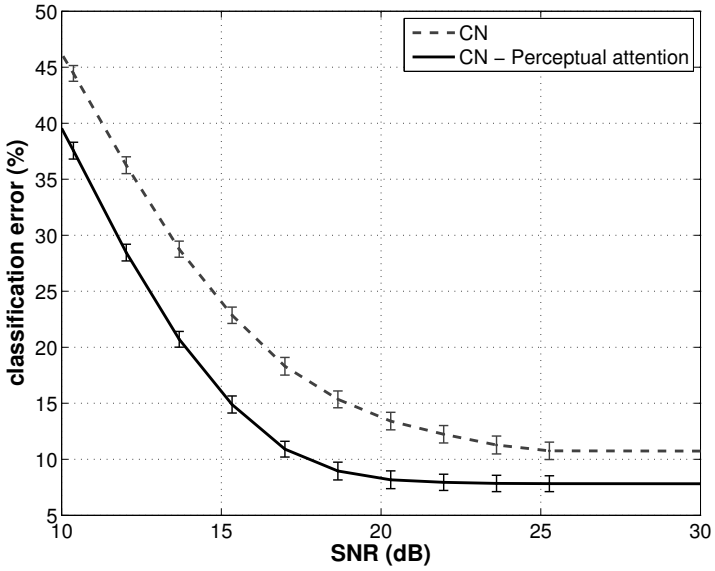


Figure 6.8: Effect of perceptual attention on performance of CN in the presence of noise.

6.5.5 Summarizing Remarks on the Experiment

When dealing with time-varying data, dependencies between consecutive frames of input data can be a valuable source of information for solving a perceptual problem of interest (Földiák, 1991; Becker, 1993). By taking advantage of temporal dependencies, we can interpret each observable, at a particular point in time, in light of the information we have acquired from previous observations. As we go forward in time, information accumulates gradually, increasing the contrast between relevant and irrelevant information. Thus, relevant information in input data can be attended better, and by the same token, irrelevant information can be disregarded in a more efficient manner. The procedure, just described, is the essence of perceptual attention.

In this experiment, we chose two different approaches for making a convolutional structure, resulting in two different hierarchical perceptors. Then, we modified those perceptors to have them operate under the influence of perceptual attention. The results of the experiment confirmed the positive impact of perceptual attention on performances of those perceptors, where both perceptors showed an improved classification accuracy. In addition, perceptual attention mitigated their sensitivity to degrading effect of uncertainties that was induced through measure-

ment noise.

6.6 Adjusting Sparseness Regularization Parameter

In sparse-coding algorithm, sparse representations may be computed through the optimization of the following cost function:

$$\hat{z} = \arg\max_{z} \ \|y - Wz\|_2^2 + \lambda\|z\|_1, \tag{6.4}$$

where y is an input vector, W is a dictionary of features, and z is the sparse representation of y with respect to W. The parameter λ regularizes the sparseness of the solution, where larger values of λ produce more sparse representations. Accordingly, the choice of the parameter λ plays an important role in the estimation of sparse representations and therefore in learning features from input data. Thereby, to ensure efficient performance of the sparse-coding algorithm, and consequently that of CSCN, it is essential to choose λ properly.

A naive approach for picking a value for λ would be to run CSCN for a set of values and the select the one that corresponds to best generalization ability for CSCN. This method might be possible for a single layer, but with the addition of each extra layer, computational cost increases exponentially. Hence, this method is practically unsuitable for a hierarchical structure and a simpler strategy with reasonable computational cost is required. In this regard, we have developed an approach to estimate values of λ at different layers of CSCN in an adaptive manner. The approach is motivated by observations made from experimental results of our computer simulations.

To proceed, we first define an *activity-ratio* function, denoted by $act(\cdot)$, as follows:

$$act(z) = \frac{\|z\|_0}{m}, \tag{6.5}$$

where m is the dimensionality of vector z, and $\|\cdot\|_0$ is the ℓ^0-norm, which computes the number of non-zero elements in a vector argument. The output of $act(\cdot)$ ranges between 0 and 1. This output depends on the value chosen for λ, namely: For large values of λ, activity-ratio of estimated representations will be close (or equal) to zero and for small values of λ, it will be close (or equal) to one.

Then, at each convolutional layer of CSCN, we randomly pick a small subset of inputs that are processed by sparse coding units of that layer; we compute activity-ratios of that subset for a range of λ values, from reasonably small to reasonably

large. If we average those activity ratios over the selected subset of inputs and plot the outcome versus the values of λ, the resulting curve will look similar to those displayed in Figure 6.9. To improve visualization, the logarithm of λ has been used in the horizontal axes of the plots in that figure.

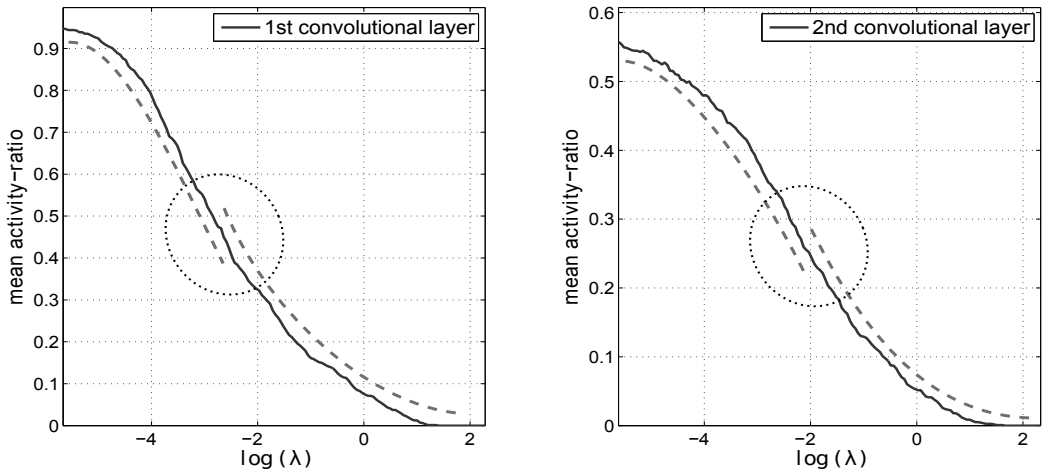


Figure 6.9: Mean activity ratios for the 1st and 2nd convolutional layers of CSCN, generated using 100 randomly selected inputs. Dashed lines are aimed at improving visibility of the change in curvature of the graphs; as the value of λ increases, the graphs switch from being concave to being convex. Dotted circles show the region of interest where the change in curvature happens.

Figure 6.9 shows clearly that as the value of λ increases, a change happens in the curvature of the graphs. As a result, the graphs change after some point from being concave to being convex, which hereafter is referred to as *permutation point*. Through experience, we have noticed that if, by adaptation, we keep the values of λ in vicinity of permutation points (with some bias toward the right side of the λ-axis), then sparse coding units and therefore CSCN will function properly and produce fine results.

Consequently, during each adaptation cycle, which happens concurrently with the processing of each training mini-batch, we first perform curve-fitting using a proper function. The function we have used is a hyperbolic tangent function, namely:

$$f(x) = \alpha \tanh(\beta x + \gamma),$$

where α, β, and γ, are parameters of interest, and x is the function argument, which corresponds to $\log(\lambda)$. Accordingly, Figure 6.10 shows the curves fitted to mean

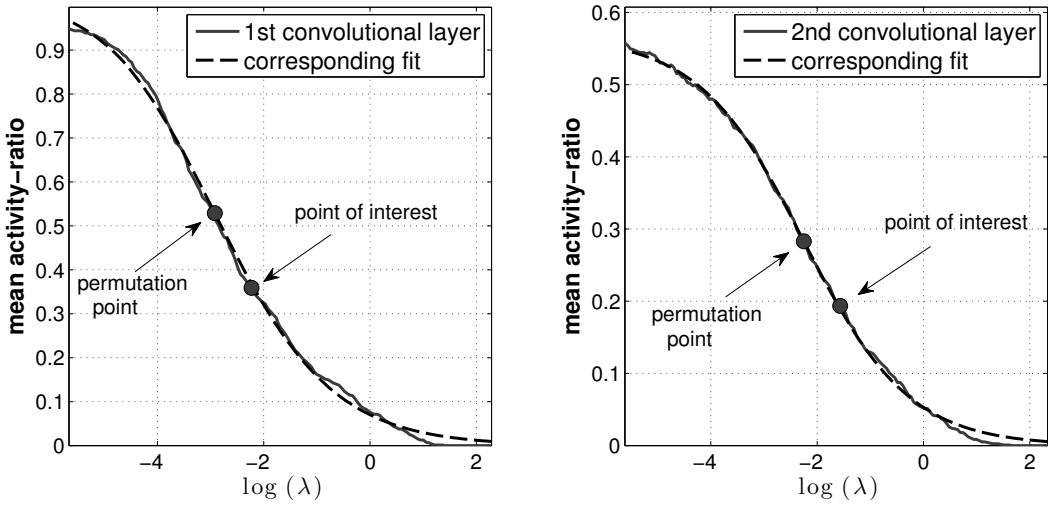


Figure 6.10: Mean activity ratios for the 1st and 2nd convolutional layers of CSCN, their corresponding fits.

activity ratios (previously shown in Figure 6.9), as well as their corresponding permutation points. The permutation point, denoted by x_p, is computed by setting the second derivative of a fit equal to zero, that is:

$$\text{find} \quad x_p, \quad \text{such that:} \quad \left.\frac{\partial^2 f}{\partial x^2}\right|_{x_p} = 0.$$

Hence, our point of interest, denoted by $\hat{x}$, is calculated by adding a bias term to x_p as follows:

$$\hat{x} = x_p + \log(b),$$

where $\log(b)$ is the additive bias term.[5] We may now go on and describe $\hat{\lambda}$ as the observed value for lambda at an adaptation cycle, defined by,

$$\begin{aligned}\hat{\lambda} &= \exp(\hat{x}) \\ &= b\exp(x_p).\end{aligned}$$

Finally, we adapt λ at each cycle by calculating a *moving average*, using the following expression:

$$\lambda_k = 0.99\lambda_{k-1} + 0.01\hat{\lambda}_k,$$

[5]For the experiments presented in this report, $\log(b)$ was computed using $b = 2$.

where index k denotes k-th adaptation cycle, and λ_k is the estimate of λ at cycle k. This procedure is carried out separately at each convolutional layer of CSCN.[6]

Figure 6.11 shows the smooth evolution of λ values from initial settings to final values. After convergence is achieved, adaptation process may be stopped.

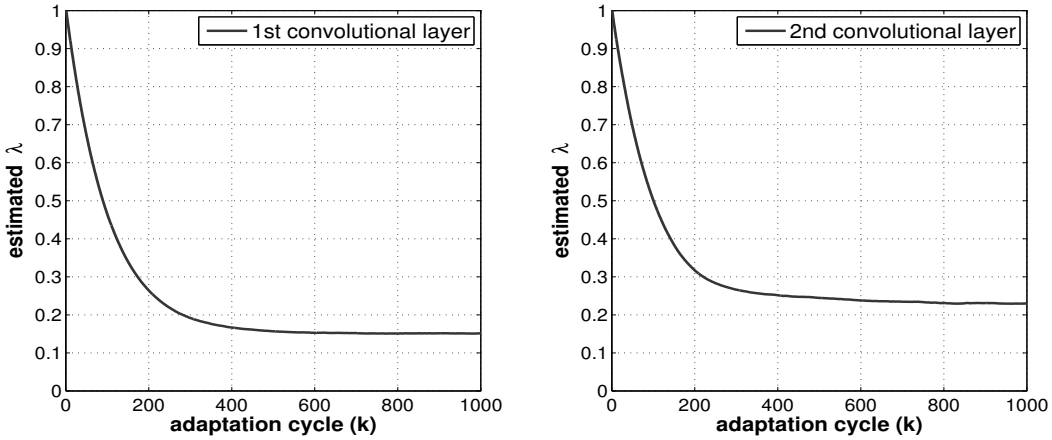


Figure 6.11: Evolution of estimated λ values over consecutive estimation cycles.

Note: The procedure that was described in this section was followed in CSCN-PA as well.

[6]To avoid bias in the estimation of λ, the logarithmic window of λ values, through which activity-ratios are calculated, is also adapted gradually. This particular adaptation aims to keep the $\log(\lambda)$-window roughly centred around $\log(\lambda_k)$.

Chapter 7

Concluding Remarks

In this thesis, inspired by the fundamental role of perceptual attention in cognition, we proposed a method to improve information processing power of a perceptor and reduce its sensitivity to uncertainties that override sensory information. Then, we supported our method by three computer experiments. In this regard, we reiterate on the contributions of this thesis by summarizing them as follows:

i) Proposing an algorithm for improved sparse coding under the influence of perceptual attention through the use of an information filter, which was supported by two challenging computer experiments, one using simulated data and the other using real-life radar data.

ii) Introducing Wirtinger calculus in the derivation of sparse coding algorithm for the first time, hence, simplifying the application of this algorithm to complex-valued problems.

iii) Extending the use of the algorithm, described in i), to hierarchical perceptors, which was supported by an extensive computer experiment for two different perceptors with the same convolutional architecture.

iv) Illustrating the impact of perceptual attention on reducing the sensitivity of a perceptor in the face of uncertainties, with measurement noise playing the role of uncertainties.

v) Proposing a method to estimate a proper set of regularization parameters for ℓ^1-regularized optimization in a multilayer structure.

The impact of this work should be considered in the context of "visualizing" the environment, which is the essence of perception. It is noteworthy that perception is

an ill-posed inverse problem (Haykin, 2012); i.e., the information required to complete a perceptual task *uniquely* and *stably* is often not available all at the same time. Rather, the information is acquired more or less in a piece-by-piece manner. As a result, it is required to preserve available information while trying to complement it over time (e.g., through additional sensory measurements); this is where perceptual attention plays an essential role in cognition.

Perceptual attention may also be described from another perspective, which is allocating proper computational resource to the processing of relevant information, while filtering away irrelevant information. In this regard, attentional control signals should "selectively" include and exclude information until perception is accomplished; as described in Chapter 2, it is suggested that this inclusive/exclusionary process in the brain builds on local-PAC.

Influenced by the profound role of attention in cognition, local-PAC does also have a special place in the paradigm of cognitive dynamic systems for both perception and acting on the environment (Haykin, 2012). In this thesis, we were focused on the perception of the environment in a cognitive dynamic system (CDS); we demonstrated that through iterative operation of local-PAC, the contrast between relevant and irrelevant parts of information is increased, resulting in an improved perception.

It is noteworthy that the same concept may be employed in the executive part of a CDS, which acts on the environment. Then, through local feedback loops, *executive* attention may be realized. As a result, the performance of actions on the environment may be improved, hence a better adaptation of the CDS to the environment. This is in fact a future prospect of the work presented in this thesis.

Another important point is that the work presented in this thesis was implemented under linearity assumption for the state-space model; this model was used in the algorithmic implementation of perceptual attention. In this regard, a future prospect may be to use nonlinear models to capture dynamics of a stochastic process of interest. Accordingly, the implementation of perceptual attention may be extended to problems with more complex dynamics.

It should be noted that under nonlinearity assumption for the state-space model, finding closed form solutions for the integral in Chapman-Kolmogorov equation, described in (3.16), is not possible. To overcome this problem, it is therefore required to use methods that find approximate solutions for that equation. Some of these methods include: Extended Kalman Filter (Bar-Shalom et al., 2001), unscented Kalman filter (Julier and Uhlmann, 1997), cubature Kalman filter (Arasaratnam and Haykin, 2009), Gaussian sum approximation (Sorenson and Alspach, 1971; Alspach and Sorenson, 1972) and particle filter (Gordon et al., 1993).

Computational procedures required by the experiments of this thesis were demanding and therefore were carried out using parallel programming techniques on SHARCNET cluster.

Appendix A

Recursive Estimation Under Linearity and Gaussianity Assumptions - Long Version

We reference the linear state-space model in equation 3.21, with white zero-mean Gaussian distributions for ω and γ. To derive the two steps of prediction and innovation under the assumptions just described, we first assume that at time-step k, posterior distribution of the stochastic process z is available, with μ_k and Σ_k being its expected value and covariance matrix, respectively. Thus, we may write,

$$p(z_k|Y_k) = \frac{1}{c_k^{(1)}} \exp\left[-\frac{1}{2}(z_k - \mu_k)^{\mathrm{T}}\Sigma_k^{-1}(z_k - \mu_k)\right], \tag{A.1}$$

where $c_k^{(1)}$ is a normalizing factor and T is the transposition operator. In addition, based on the process equation in (3.21), we may express $p(z_{k+1}|z_k)$ as follows:

$$p(z_{k+1}|z_k) = \frac{1}{c_k^{(2)}} \exp\left[-\frac{1}{2}(z_{k+1} - Fz_k)^{\mathrm{T}}Q^{-1}(Fz_{k+1} - z_k)\right], \tag{A.2}$$

where Q is the covariance matrix for the system noise γ and $c_k^{(2)}$ is again a normalizing factor.

Step 1 - Prediction

For the prediction step, equation (3.16) should be solved, that is,

$$p(z_{k+1}|Y_k) = \frac{1}{c_k^{(1)} c_k^{(2)}} \int \exp\left[-\frac{1}{2}(z_{k+1} - Fz_k)^{\mathrm{T}} Q^{-1}(z_{k+1} - Fz_k)\right] \times \exp\left[-\frac{1}{2}(z_k - \mu_k)^{\mathrm{T}} \Sigma_k^{-1}(z_k - \mu_k)\right] \mathrm{d}z_k \tag{A.3}$$

$$= \frac{1}{c_k^{'}} \int \exp\left[-\frac{1}{2}\Phi(z_k)\right] \mathrm{d}z_k, \tag{A.4}$$

where $c_k^{'} = c_k^{(1)} c_k^{(2)}$

$$\Phi(z_k) = (z_{k+1} - Fz_k)^{\mathrm{T}} Q^{-1}(z_{k+1} - Fz_k) + (z_k - \mu_k)^{\mathrm{T}} \Sigma_k^{-1}(z_k - \mu_k) \tag{A.5}$$

$$= z_k^{\mathrm{T}} F^{\mathrm{T}} Q^{-1} F z_k + z_k^{\mathrm{T}} \Sigma_k^{-1} z_k - (F^{\mathrm{T}} Q^{-1} z_{k+1} + \Sigma_k^{-1}\mu_k)^{\mathrm{T}} z_k - z_k^{\mathrm{T}}(F^{\mathrm{T}} Q^{-1} z_{k+1} + \Sigma_k^{-1}\mu_k) + \mu_k^{\mathrm{T}} \Sigma_k^{-1} \mu_k + z_{k+1}^{\mathrm{T}} Q^{-1} z_{k+1}. \tag{A.6}$$

We now define

$$A = F^{\mathrm{T}} Q^{-1} F + \Sigma_k^{-1},$$
$$B = F^{\mathrm{T}} Q^{-1} z_{k+1} + \Sigma_k^{-1}\mu_k,$$

and

$$C = \mu_k^{\mathrm{T}} \Sigma_k^{-1} \mu_k + z_{k+1}^{\mathrm{T}} Q^{-1} z_{k+1}.$$

To proceed with the solution of the integral in (A.4), we express $\phi(z_k)$ in terms of A, B, and C. Thus, we may write

$$p(z_{k+1}|Y_k) = \frac{1}{c_k^{'}} \int \exp\left[-\frac{1}{2}\Phi(z_k)\right] \mathrm{d}z_k \tag{A.7}$$

$$= \frac{1}{c_k^{'}} \exp\left(-\frac{C}{2}\right) \int \exp\left[-\frac{1}{2}(z_k^{\mathrm{T}} A z_k - B^{\mathrm{T}} z_k - z_k^{\mathrm{T}} B)\right] \mathrm{d}z_k \tag{A.8}$$

$$= \frac{1}{c_k^{'}} \exp\left(-\frac{C}{2}\right) (2\pi)^{\frac{n}{2}} |A|^{-\frac{1}{2}} \exp\left[\frac{1}{2} B^{\mathrm{T}} A^{-1} B\right] \tag{A.9}$$

$$= \frac{1}{c_k^{'}} (2\pi)^{\frac{n}{2}} |A|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(C - B^{\mathrm{T}} A^{-1} B)\right] \tag{A.10}$$

$$= \frac{1}{c_k^{''}} \exp\left[-\frac{1}{2}(C - B^{\mathrm{T}} A^{-1} B)\right], \tag{A.11}$$

where

$$c_k^{''} = \frac{1}{c_k^{'}} (2\pi)^{\frac{n}{2}} |A|^{-\frac{1}{2}}. \tag{A.12}$$

We now continue to write,

$$\begin{aligned} C - B^{\mathrm{T}}A^{-1}B =& \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k + z_{k+1}^{\mathrm{T}}Q^{-1}z_{k+1} \\ & - (z_{k+1}^{\mathrm{T}}Q^{\text{-T}}F + \mu_k^{\mathrm{T}}\Sigma_k^{\text{-T}})A^{-1}(F^{\mathrm{T}}Q^{-1}z_{k+1} + \Sigma_k^{-1}\mu_k), \end{aligned} \tag{A.13}$$

where $Q^{\text{-T}}$ and $\Sigma_k^{\text{-T}}$ are transposed versions of $Q^{\text{-1}}$ and $\Sigma_k^{\text{-1}}$, respectively. Both Q and Σ_k are covariance matrices and therefore symmetric. Hence, $Q^{\text{-T}} = Q^{\text{-1}}$ and $\Sigma_k^{\text{-T}} = \Sigma_k^{\text{-1}}$. Accordingly, we may go on to write

$$\begin{aligned} C - B^{\mathrm{T}}A^{-1}B =& z_{k+1}^{\mathrm{T}}Q^{-1}z_{k+1} - z_{k+1}^{\mathrm{T}}Q^{-1}FA^{-1}F^{\mathrm{T}}Q^{-1}z_{k+1} \\ & + \mu_k^{\mathrm{T}}\Sigma_k^{-1}\mu_k - \mu_k^{\mathrm{T}}\Sigma_k^{-1}A^{-1}\Sigma_k^{-1}\mu_k \\ & - \mu_k^{\mathrm{T}}\Sigma_k^{-1}A^{-1}F^{\mathrm{T}}Q^{-1}z_{k+1} - z_{k+1}^{\mathrm{T}}Q^{-1}FA^{-1}\Sigma_k^{-1}\mu_k, \end{aligned} \tag{A.14}$$

which may be further simplified as follows:

$$\begin{aligned} C - B^{\mathrm{T}}A^{-1}B =& z_{k+1}^{\mathrm{T}}\left[Q^{-1} - Q^{-1}FA^{-1}F^{\mathrm{T}}Q^{-1}\right]z_{k+1} \\ & + \mu_k^{\mathrm{T}}\left[\Sigma_k^{-1} - \Sigma_k^{-1}A^{-1}\Sigma_k^{-1}\right]\mu_k \\ & - (\mu_k^{\mathrm{T}}\Sigma_k^{-1}A^{-1}F^{\mathrm{T}}Q^{-1}z_{k+1}) \\ & - (\mu_k^{\mathrm{T}}\Sigma_k^{-1}A^{-1}F^{\mathrm{T}}Q^{-1}z_{k+1})^{\mathrm{T}}, \end{aligned} \tag{A.15}$$

The first line of equation (A.20) may be modified in the following manner:

$$\begin{aligned} Q^{-1} - Q^{-1}FA^{-1}F^{\mathrm{T}}Q^{-1} =& Q^{-1} - Q^{-1}(F^{\text{-T}}AF^{\text{-1}})^{-1}Q^{-1} \\ =& Q^{-1} - Q^{-1}(Q^{\text{-1}} + F^{\text{-T}}\Sigma_k^{-1}F^{\text{-1}})^{-1}Q^{-1} \\ =& Q^{-1} - Q^{-1}\left[Q^{\text{-1}} + (F\Sigma_k F^{\mathrm{T}})^{-1}\right]^{-1}Q^{-1} \\ =& Q^{-1} - Q^{-1}\left[Q - Q(Q + F\Sigma_k F^{\mathrm{T}})^{-1}Q\right]Q^{-1} \qquad \text{(A.16)} \\ =& (Q + F\Sigma_k F^{\mathrm{T}})^{-1}, \qquad \text{(A.17)} \end{aligned}$$

where to derive equation (A.16) *matrix inversion lemma* (Haykin, 2013) is applied. Similarly, the second line of equation (A.20) may be modified as follows:

$$\begin{aligned} \Sigma_k^{-1} - \Sigma_k^{-1}A^{-1}\Sigma_k^{-1} =& \Sigma_k^{-1} - \Sigma_k^{-1}(F^{\mathrm{T}}Q^{-1}F + \Sigma_k^{-1})^{-1}\Sigma_k^{-1} \\ =& \Sigma_k^{-1} - \Sigma_k^{-1}\left[\Sigma_k - \Sigma_k(F^{\text{-1}}QF^{\text{-T}} + \Sigma_k)^{-1}\Sigma_k\right]\Sigma_k^{-1} \qquad \text{(A.18)} \\ =& (F^{\text{-1}}QF^{\text{-T}} + \Sigma_k)^{-1} \\ =& F^{\mathrm{T}}F^{\text{-T}}(F^{\text{-1}}QF^{\text{-T}} + \Sigma_k)^{-1}F^{\text{-1}}F \\ =& F^{\mathrm{T}}(FF^{\text{-1}}QF^{\text{-T}} + F\Sigma_k F^{\mathrm{T}})^{-1}F \\ =& F^{\mathrm{T}}(Q + F\Sigma_k F^{\mathrm{T}})^{-1}F. \qquad \text{(A.19)} \end{aligned}$$

Equation (A.16) is also derived using the inversion lemma.

The same process may be carried out for the third line of equation (A.20), and in the end, we may simplify it as follows:

$$C - B^{\mathrm{T}}A^{-1}B = (z_{k+1} - F\mu_k)^{\mathrm{T}}(Q + F\Sigma_k F^{\mathrm{T}})^{-1}(z_{k+1} - F\mu_k).$$

Thus, we may go on to write

$$p(z_{k+1}|Y_k) = \frac{1}{c_k''} \exp\left[-\frac{1}{2}(z_{k+1} - \mu_{k+1|k})^{\mathrm{T}}\Sigma_{k+1|k}^{-1}(z_{k+1} - \mu_{k+1|k})\right], \qquad \text{(A.20)}$$

where $\mu_{k+1|k}$ and $\Sigma_{k+1|k}$ are respectively defined by:

$$\begin{aligned}\mu_{k+1|k} &= \mathbb{E}[z_{k+1}|Y_k] \\ &= F\mu_k,\end{aligned}$$

and

$$\begin{aligned}\Sigma_{k+1|k} &= \mathrm{Cov}[z_{k+1}|Y_k] \\ &= Q + F\Sigma_k F^{\mathrm{T}}.\end{aligned}$$

which are the formulas for the prediction step in Kalman filter.

Step 2 - Innovation

The innovation step is carried out through the use of Bayes rule. Hence, we may write,

$$p(z_{k+1}|Y_{k+1}) = \frac{1}{c_{k+1}} p(y_{k+1}|z_{k+1})p(z_{k+1}|Y_k) \qquad \text{(A.21)}$$

where c_{k+1} is again a normalizing factor. Substituting the right side of the equation with actual pdfs, we will have

$$\begin{aligned}p(z_{k+1}|Y_{k+1}) =& \frac{1}{c_{k+1}} \exp\left[-\frac{1}{2}(y_{k+1} - Hz_{k+1})^{\mathrm{T}}R^{-1}(y_{k+1} - Hz_{k+1})\right] \\ &\times \exp\left[-\frac{1}{2}(z_{k+1} - \mu_{k+1|k})^{\mathrm{T}}\Sigma_{k+1|k}^{-1}(z_{k+1} - \mu_{k+1|k})\right] \\ &= \frac{1}{c_{k+1}} \exp\left[-\frac{1}{2}A\right]. \qquad \text{(A.22)}\end{aligned}$$

with A being defined by,

$$\begin{aligned} A =&(y_{k+1} - Hz_{k+1})^{\mathrm{T}} R^{-1}(y_{k+1} - Hz_{k+1}) \\ &+ (z_{k+1} - \mu_{k+1|k})^{\mathrm{T}} \Sigma_{k+1|k}^{-1}(z_{k+1} - \mu_{k+1|k}) \qquad \text{(A.23)} \\ =&z_{k+1}^{\mathrm{T}}(\Sigma_{k+1|k}^{-1} + H^{\mathrm{T}} R^{-1} H) z_{k+1} \\ &- z_{k+1}^{\mathrm{T}}(H^{\mathrm{T}} R^{-1} y_{k+1} + \Sigma_{k+1|k}^{-1}\mu_{k+1|k}) \\ &- (H^{\mathrm{T}} R^{-1} y_{k+1} + \Sigma_{k+1|k}^{-1}\mu_{k+1|k})^{\mathrm{T}} z_{k+1} \\ &+ (y_{k+1}^{\mathrm{T}} R^{-1} y_{k+1} + \mu_{k+1|k}^{\mathrm{T}} \Sigma_{k+1|k}^{-1}\mu_{k+1|k}). \qquad \text{(A.24)} \end{aligned}$$

Based on the first line of equation (A.24), we now proceed by defining $\Sigma_{k+1|k+1}^{-1}$ and apply matrix inversion lemma as follows:

$$\begin{aligned} \Sigma_{k+1|k+1} =&(\Sigma_{k+1|k}^{-1} + H^{\mathrm{T}} R^{-1} H)^{-1} \\ =&\Sigma_{k+1|k} - \Sigma_{k+1|k} H^{\mathrm{T}} (R + H\Sigma_{k+1|k} H^{\mathrm{T}})^{-1} H \Sigma_{k+1|k} \end{aligned}$$

which is exactly the innovation rule of Kalman filter for updating the covariance matrix. Based on this equation, we now are in the position to define residual covariance matrix S and filter gain G. Hence, we may write,

$$S = R + H\Sigma_{k+1|k} H^{\mathrm{T}} \qquad \text{(A.25)}$$

$$G = \Sigma_{k+1|k} H^{\mathrm{T}} S^{-1}, \qquad \text{(A.26)}$$

where,

$$\Sigma_{k+1|k+1} = \Sigma_{k+1|k} - GS^{-1}G^{\mathrm{T}}.$$

Then, if we rewrite the second line of equation (A.24) in the following manner

$$z_{k+1}^{\mathrm{T}} \Sigma_{k+1|k+1}^{-1} \Sigma_{k+1|k+1}(H^{\mathrm{T}} R^{-1} y_{k+1} + \Sigma_{k+1|k}^{-1}\mu_{k+1|k}) \qquad \text{(A.27)}$$

we may express $\mu_{k+1|k+1}$ as follows:

$$\mu_{k+1|k+1} = \Sigma_{k+1|k+1}(H^{\mathrm{T}} R^{-1} y_{k+1} + \Sigma_{k+1|k}^{-1}\mu_{k+1|k}) \qquad \text{(A.28)}$$

$$= \mu_{k+1|k} + G(y_{k+1} - H\mu_{k+1|k}) \qquad \text{(A.29)}$$

which is the desired innovation rule for updating the expected value.

Appendix B

Algorithmic Steps of EM Algorithm

In this section, we present algorithmic steps of EM algorithm under linearity and Gaussianity assumptions. These steps should be iteratively repeated until convergence is reached.

B.1 Step 1 - Expectation

As described earlier in Subsection 5.2.1, the expectation step of the EM algorithm is accomplished through Bayesian smoothing. Similar to the case of Bayesian filtering, the solution to Bayesian smoothing may be optimally derived under linearity and Gaussianity assumptions (Bar-Shalom et al., 2001), the end result of which is also known as Kalman smoother. Algorithm 5 presents the steps of a Kalman filter followed by smoother, for a sequence of N observables.

Hence, completing the steps of Algorithm 5 result in the accomplishment of the expectation step.

Algorithm 5 Kalman Filter Followed by Smoother

set μ_0 and Σ_0 to proper initial values.
Repeat for $k = 1, 2, 3, ..., N$

prediction:

$\mu_{k+1|k} = F\mu_{k|k}$
$\Sigma_{k+1|k} = F\Sigma_{k|k}F^{\mathrm{T}} + Q$

innovation:

$S = R + H\Sigma_{k|k-1}H^{\mathrm{T}}$
$G_k = \Sigma_{k|k-1}H^{\mathrm{T}}S^{-1}$
$\mu_{k|k} = \mu_{k|k-1} + G_k(y_k - H\mu_{k|k-1})$
$\Sigma_{k|k} = \Sigma_{k|k-1} - G_k S G_k^{\mathrm{T}}$

Repeat for $k = N, N-1, N-2, ..., 0$

smoothing:

$J_{k-1} = \Sigma_{k-1}F^{\mathrm{T}}\Sigma_{k|k-1}^{-1}$
$\mu_{k-1|N} = \mu_k + J_{k-1}(\mu_{k|N} - F\mu_{k-1})$
$\Sigma_{k-1|N} = \Sigma_{k-1} + J_{k-1}(\Sigma_{k|N} - \Sigma_{k|k-1})J_{k-1}^{\mathrm{T}}$

B.2 Step 2 - Maximization

In the maximization step, we start with the function $\log p(Y, Z|M)$, which is expressed as follows:

$$\begin{aligned}\log p(Y, Z|M) = &-\frac{1}{2}\log|\Sigma_0| - \frac{1}{2}(z_0 - \mu_0)^{\mathrm{T}}\Sigma_0^{-1}(z_0 - \mu_0) \\ &- \frac{N}{2}\log|Q| - \frac{1}{2}\sum_{k=1}^{N}(z_k - Fz_{k-1})^{\mathrm{T}}Q^{-1}(z_k - Fz_{k-1}) \\ &- \frac{N}{2}\log|R| - \frac{1}{2}\sum_{k=1}^{N}(y_k - Hz_k)^{\mathrm{T}}R^{-1}(y_k - Hz_k)\end{aligned} \tag{B.1}$$

We now assume that we are in iteration i of the EM algorithm. Hence, $p(Z|Y, M^i)$ is defined by,

$$p(Z|Y, M^i) = \prod_{k=0}^{N}\frac{1}{c_k^i}\exp\left[-\frac{1}{2}(z_k - \mu_{k|N}^i)^{\mathrm{T}}(\Sigma_{k|N}^i)^{-1}(z_k - \mu_{k|N}^i)\right] \tag{B.2}$$

where c_k^i is a proper normalizing factor, and $\mu_{k|N}^i$ and $\Sigma_{k|N}^i$ are the smoothing results available at i^{th} iteration of the EM algorithm.

To proceed, it is necessary to derive the expectation of the equation described in (B.1) over the pdf presented in equation (B.2), which may be expressed as follows:

$$\mathbb{E}\left[\log p(Y,Z|M)\right]_{p(Z|Y,M^i)} = \int_Z p(Z|Y,M^i)\log p(Y,Z|M)\mathrm{d}Z. \tag{B.3}$$

The matrices Σ_0, R, and Q are covariance matrices; hence, they are assumed to be positive definite. In addition, $p(Z|Y,M^i)$ is Gaussian. As a result, solving equation (B.3) is relatively straightforward, with the end result, as described in (Shumway and Stoffer, 1982), being defined by :

$$\begin{aligned}\mathbb{E}\left[\log p(Y,Z|M)\right] = &-\frac{1}{2}\log|\Sigma_0| - \frac{1}{2}\,\text{tr}\left\{\Sigma_0^{-1}\left[\Sigma_{0|N}^i + (\mu_0-\mu_{0|N}^i)(\mu_0-\mu_{0|N}^i)^{\text{T}}\right]\right\}\\ &-\frac{N}{2}\log|Q| - \frac{1}{2}\,\text{tr}\left\{Q^{-1}\left(F\mathcal{A}F^{\text{T}} - \mathcal{B}F^{\text{T}} - F\mathcal{B}^{\text{T}} + \mathcal{C}\right)\right\}\\ &-\frac{N}{2}\log|R|\\ &-\frac{1}{2}\,\text{tr}\left\{R^{-1}\sum_{k=1}^{N}\left[(y_k - H\mu_{k|N}^i)(y_k - H\mu_{k|N}^i)^{\text{T}} + H\Sigma_{k|N}^i H^{\text{T}}\right]\right\},\end{aligned} \tag{B.4}$$

where $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are respectively defined by,

$$\mathcal{A} = \sum_{k=1}^{N}\left[\Sigma_{k-1|N}^i + \mu_{k-1|N}^i(\mu_{k-1|N}^i)^{\text{T}}\right],$$

$$\mathcal{B} = \sum_{k=1}^{N}\left[\Sigma_{k,k-1|N}^i + \mu_{k|N}^i(\mu_{k-1|N}^i)^{\text{T}}\right],$$

and,

$$\mathcal{C} = \sum_{k=1}^{N}\left[\Sigma_{k|N}^i + \mu_{k|N}^i(\mu_{k|N}^i)^{\text{T}}\right].$$

It is important to note for the computation of $\mathcal{B}$, it is necessary to calculate covariance of z_k and z_{k-1} for all $0 \le k \le N$, defined by

$$\Sigma_{k,k-1|N}^i = \text{Cov}\left[z_k, z_{k-1}|Y_N\right].$$

The term, just described, roots from the second line of equation (B.1), the solution for which at $k = N$ is expressed as follows:

$$\Sigma_{N,N-1|N} = (I - G_k H) F \Sigma_{N-1|N-1},$$

where I is an identity matrix of proper dimensions, and G is the the filter gain described in Algorithm 5. For other time-steps, the solution is defined by

$$\Sigma_{k-1,k-2|N} = \Sigma_{k-1|k-1} J_{k-2}^{\mathrm{T}} + J_{k-1}(\Sigma_{k,k-1|N} - F\Sigma_{k-1|k-1}) J_{k-2}^{\mathrm{T}},$$

with k going backward in time form N to 0, and J being the smoother gain in Algorithm 5.

Now that the solution to equation (B.3) is available, the maximization step may be completed simply by minimizing equation B.4 with respect to model parameters μ_0, Σ_0, R, and Q (and possibly F and H). For more details, refer to (Shumway and Stoffer, 1982).

Appendix C

Wirtinger Calculus for Complex-valued Problems

In many practical applications such as radar, medical imaging, communication systems, etc., we are required to work with complex-valued data, where due to the nature of the problem, complex representations are most convenient. In order to consider all the information available in observables (e.g., the correlation between the real and imaginary parts), these applications demand the processing of data to be carried out in the complex domain.

When dealing with complex variables, it is common to use mappings from the complex domain, $\mathbb{C}^N$, to the real domain, $\mathbb{R}^{2N}$, where computations are performed. However, this complicates the computation process since it doubles the number of variables in the problem. Moreover, transformations to and from the computation space may be required repeatedly. It is therefore beneficial to find a proper method for carrying out computations directly in the complex domain. This can greatly simplify the complexity of algorithms; moreover, it is more realistic with respect to underlying physical characteristics of the problem.

In this appendix, preliminaries of complex calculus are first explained. Then, for a computationally efficient method to handle complex-valued problems, Wirtinger calculus (Wirtinger, 1927) is introduced.

Using Wirtinger calculus, a problem of interest is formulated in the complex domain in a manner similar to that in the real domain. As a result, formulation of the problem is simplified and computational cost is reduced. Moreover, it becomes possible to simply extend most of the tools that are developed for real-valued signals to complex-valued problems. For more details, refer to (Adali and Haykin, 2010; Ablowitz and Fokas, 2003).

C.1 Differentiability of a Function in the Complex Domain

Considering a function $f(z) : \mathbb{C} \rightarrow \mathbb{C}$ with the format:

$$f(z) = u(x, y) + \mathrm{j}v(x, y),$$

where $\mathrm{j} = \sqrt{-1}$ and $z = x + \mathrm{j}y$, the derivative of $f(z)$ at the point z_0 can be written as follows:

$$f'(z_0) = \lim_{\Delta z \rightarrow 0} \frac{f(z_0 + \Delta z) - f(z_0)}{\Delta z}.$$

In the case of real-valued functions, a bounded limit for the equation just described will be sufficient for differentiability of f at a point z_0. However, for complex-valued functions, another requirement should be fulfilled due to the additional dimensionality of complex-valued numbers; this demands the result of the limit to be independent of the direction of approach of Δz toward zero.

We can evaluate differentiability of f at z_0 by deriving $f'(z_0)$ using two different directions of approach, namely: $\Delta x \rightarrow 0$ while $\Delta y = 0$, and $\Delta y \rightarrow 0$ while $\Delta x = 0$. This will respectively provide the following two first-order partial-derivatives:

$$f'(z_0) = \frac{\partial u}{\partial x} + \mathrm{j}\frac{\partial v}{\partial x}, \tag{C.1}$$

$$f'(z_0) = \frac{\partial v}{\partial y} - \mathrm{j}\frac{\partial u}{\partial y}. \tag{C.2}$$

In order to be differentiable, equations (C.1) and (C.2) must provide the same values at z_0, resulting in the well-known Cauchy-Riemann equations:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \quad \text{and} \quad \frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}. \tag{C.3}$$

The two equations in (C.3) are the necessary conditions for differentiability of $f(z)$; if and only if $f(z)$ remains differentiable at every point of a given region, then it will be *analytic* within that region.

The Cauchy-Riemann equations are very strong requirements for differentiability of complex-valued functions, and they impose a rigid constraint on the structure of real and imaginary parts of a function f (i.e., u and v). For instance, for a real-valued cost function where $v(x, y) = 0$, the Cauchy-Riemann equations will not hold and as a result, the cost function will not be differentiable.

C.2 Wirtinger Calculus

Wirtinger calculus provides a solution to the problem of non-analyticity by relaxing the constraints of the Cauchy-Riemann equations and introducing a more flexible requirement: *real differentiability*. A function $f(z) = u(x, y) + \mathrm{j}v(x, y)$ is real differentiable, when both $u(x, y)$ and $v(x, y)$ are smooth and differentiable functions of real variables x and y. As shown in (Remmert, 1991), we can write the variables x and y in terms of complex variables z and z^*, which result in:

$$x = \frac{z + z^*}{2} \quad \text{and} \quad y = \frac{z - z^*}{2\mathrm{j}}, \tag{C.4}$$

where the asterisk $*$ denotes complex conjugation. Then by treating z and z^* as independent variables, we can apply the chain rule of calculus and derive the following equations:

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial z} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial z} = \frac{1}{2}\frac{\partial f}{\partial x} + \frac{1}{2\mathrm{j}}\frac{\partial f}{\partial y}, \tag{C.5}$$

$$\frac{\partial f}{\partial z^*} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial z^*} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial z^*} = \frac{1}{2}\frac{\partial f}{\partial x} - \frac{1}{2\mathrm{j}}\frac{\partial f}{\partial y}, \tag{C.6}$$

which result in the two generalized complex partial-derivative formulas:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial f}{\partial x} - \mathrm{j}\frac{\partial f}{\partial y}\right), \tag{C.7}$$

$$\frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\frac{\partial f}{\partial x} + \mathrm{j}\frac{\partial f}{\partial y}\right). \tag{C.8}$$

Wirtinger calculus, on the other hand, provides a simpler solution by considering a function $f(z) : \mathbb{C} \to \mathbb{C}$ as a function of two real variables x and y, hence, $f(x, y) : \mathbb{R}^2 \to \mathbb{C}$. Since z and z^* can be written as follows:

$$z = x + \mathrm{j}y \quad \text{and} \quad z^* = x - \mathrm{j}y, \tag{C.9}$$

$f(x, y)$ can be rewritten as a bivariate function, $f(z, z^*)$. Thus, Wirtinger calculus can be applied using the *trick* of treating z and z^* as independent variables; and the partial-derivatives, $\frac{\partial f}{\partial z}$ and $\frac{\partial f}{\partial z^*}$, can be derived in the same manner as it is done for the real-valued case. Thus, when taking the partial-derivative with respect to z, we

consider z^* as a constant and derive $\frac{\partial f}{\partial z}$. Likewise, when deriving $\frac{\partial f}{\partial z^*}$, we consider z as a constant and take the partial-derivative with respect to z^*.

The essence of this trick may be shown using a simple example: supposing we have a function $f(z) = |z|$, which is equivalent to $f(x, y) = \sqrt{x^2 + y^2}$, the generalized complex partial-derivatives of $f(z)$ will be:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{2x}{2\sqrt{x^2+y^2}} - \mathrm{j}\frac{2y}{2\sqrt{x^2+y^2}}\right) = \frac{x - \mathrm{j}y}{2\sqrt{x^2+y^2}} = \frac{z^*}{2|z|}, \tag{C.10}$$

$$\frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\frac{2x}{2\sqrt{x^2+y^2}} + \mathrm{j}\frac{2y}{2\sqrt{x^2+y^2}}\right) = \frac{x + \mathrm{j}y}{2\sqrt{x^2+y^2}} = \frac{z}{2|z|}. \tag{C.11}$$

By rewriting $f(z)$ with respect to z and z^*, $f(z, z^*) = \sqrt{zz^*}$, we can directly derive the partial-derivatives using Wirtinger calculus:

$$\frac{\partial f}{\partial z} = \frac{z^*}{2\sqrt{zz^*}} = \frac{z^*}{2|z|}, \tag{C.12}$$

$$\frac{\partial f}{\partial z^*} = \frac{z}{2\sqrt{zz^*}} = \frac{z}{2|z|}. \tag{C.13}$$

An important point that can be seen from the example just presented is that when a function $f(z)$ is real-valued, its partial-derivatives will have the following property:

$$\left(\frac{\partial f}{\partial z}\right)^* = \frac{\partial f}{\partial z^*}. \tag{C.14}$$

C.3 Optimization of Real-valued Cost Functions

As mentioned earlier, real-valued functions that have a complex-valued argument are not analytic in the complex domain. Although Wirtinger calculus provides the solution to this problem, attention must be given when dealing with the optimization of real-valued cost functions. This is due to the fact that the direction of the steepest descent toward *minima* is not the negative of the gradient with respect to z, i.e., $-\nabla_z f$. In fact, it is the negative of the gradient with respect to z^*, which is $-\nabla_{z^*} f$.

Therefore, the update rule of optimization variable in a simple gradient descent algorithm will be:

$$z(t+1) = z(t) - \mu \nabla_{z^*} f, \tag{C.15}$$

where, μ is a proper step size, and $\nabla_{z^*} f$ is derived as follows:

$$\nabla_{z^*} f = 2\frac{\partial f}{\partial z^*}. \tag{C.16}$$

For more details on optimization in the complex domain, refer to (Adali and Haykin, 2010; Kreutz-Delgado, 2009).

Appendix D

Brief Overview of Hierarchical Perceptors

D.1 Multilayer Perceptron

Perceptron, first proposed by Rosenblatt (1958), is the most well-known neural component in the literature on neural networks, which in a very efficient manner, allows us to distinguish between a number of linearly separable classes of patterns using affine hyperplanes. The idea of the perceptron builds on the nonlinear model of a neuron introduced by (McCulloch and Pitts, 1943) that consists of a linear combiner followed by a hard limiter (e.g., sigmoid function).

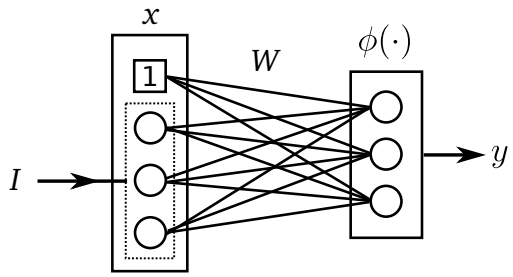


Figure D.1: Structure of a perceptron.

Figure D.1 exemplifies a perceptron, where I is a vector of inputs, W is a matrix of connection weights, $\phi(\cdot)$ is the hard limiter, also known as the activation function, and y represents activities of output neurons. In Figure D.1 bias terms of output neurons are recorded in the first row of the matrix W; these bias terms

are required for having affine hyperplanes that do not necessarily pass through the origin. Accordingly, a unit term is added on top of the input vector I, thus forming vector x, which is used in numerical operations of the perceptron. In this regard, feedforward operation of a perceptron may be expressed as follows:

$$y = \phi(Wx). \tag{D.1}$$

It is very common to train perceptrons in a supervised manner. Thus, we first require a training set T, composed of a set of inputs X and their corresponding desired outputs D, which may be expressed as follows:

$$T : \begin{cases} X = \{\ x^i\ \} \\ D = \{\ d^i\ \} \end{cases}, \quad i = 1, \cdots, m. \tag{D.2}$$

where $\{x^i, d^i\}$ corresponds to i^{th} training example in the training set. The goal will then be to modify connection weights in matrix W so that the set of output activities of neurons Y, produced in response to X, becomes closer to the desired set D. To this end, a cost function may be defined and W may be updated using a gradient based method to minimize that cost function. A natural choice that comes to mind for this cost function is to use the euclidean norm in the following manner:

$$C(W) = \frac{1}{2m} \sum_{i=1}^{m} ||y^i - d^i||_2^2, \tag{D.3}$$

which actually works well for regression problems. In classification problems, however, a better choice for the cost function may be made, which makes the learning of weights more efficient. This choice has roots in a description of neurons as on-off switches and therefore the adoption of Bernoulli distribution to model their firings. For a stochastic binary process s, Bernoulli distribution is defined by,

$$p(s; \alpha) = \alpha^s (1 - \alpha)^{1-s}, \tag{D.4}$$

where $0 < \alpha < 1$, and $s \in \{0, 1\}$. Accordingly, this new cost function that is to be minimized may be expressed as follows:

$$C(W) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[d_j^i \log y_j^i + (1 - d_j^i) \log(1 - y_j^i) \right], \tag{D.5}$$

where y_j identifies j^{th} element of vector y, and n is the length of that vector.

An important challenge that exists in the training of a perceptron, and in general, any other machine learning algorithm, is the possibility of overfitting. In other words, it is possible to fit the algorithm in a way that performs very well on the training set without understanding the fact that the algorithm may in fact be losing its generalization ability, where by generalization ability, we mean the ability of an algorithm to make correct decisions when exposed to new data that are not used in the training process.

To overcome this challenge, a very popular and effective method is to add a regularization term to the cost function, for example a quadratic term like the one proposed by Tikhonov (1963). As a result, the new cost function $C_{\text{reg}}(W)$ may be defined by

$$C_{\text{reg}}(W) = C(W) + \frac{\lambda}{2m} \sum_{i=2}^{p} \sum_{j=1}^{n} W_{ij}^2, \tag{D.6}$$

where λ is the regularization parameter, p is the length of vector x, and $C(W)$ is either of the two cost functions described in (D.3) and (D.3).[1]

It is possible to expand on the idea of a perceptron by stacking a number of them, one after another, where outputs of one perceptron is supplied as inputs to the next. Accordingly, a multilayer perceptron is formed (Rosenblatt, 1961), which consists of input and output layers, and *hidden* layers that exist in between. As simple as this idea may seem, it was only after mid-1980s, with the development of backpropagation algorithm for training (Rumelhart and McClelland, 1986), that multilayer perceptrons gained an increasing popularity. Until this day, they are still the most commonly-used hierarchical structures in machine learning applications. Figure D.2 display a multilayer perceptron.

Generally speaking, the method of backpropagation builds on the same ideas described earlier for the training of a simple perceptron. In fact, the same cost functions may be used for the computation of update rules for connection weights. To derive these update rules, the chain rule must be applied.

D.2 Random Decision Forests

A random decision forest (Breiman, 2001), is formed by a group of randomly-trained decision trees. Consequently, the trees that constitute such a random forest

[1] It is noteworthy that connection weights in W that pertain to bias terms should not be regularized.

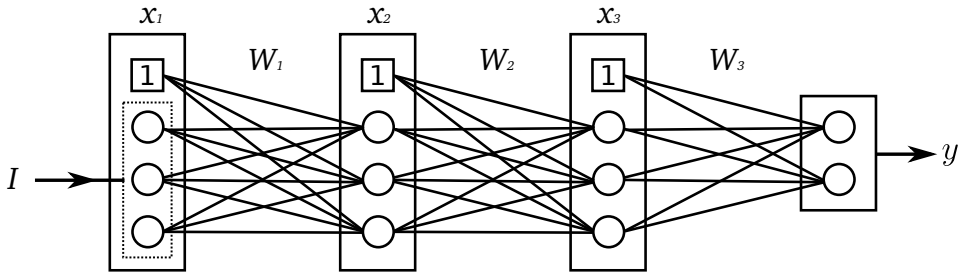


Figure D.2: Structure of a multilayer perceptron.

are different from each other. The random difference between constituting trees improves generalization ability of the forest compared to that of each one of the trees individually.

A tree is a simple graph with a hierarchical structure, which may not contain any loops. Flow of information in its graph is unidirectional, which starts from the *root node*. Information then propagates over *edges*, passing through *internal nodes* layer after layer, until a *terminal node* is reached; a terminal node is also known as a *leaf*.

A decision tree tackles a complex problem using *divide and conquer* strategy. In other words, a complex problem is first broken down into a set of less complicated problems. These problems are then handled by nodes of the tree, which run primitive tests on information they receive; these primitive tests are also denoted by *split functions* or *weak learners*. For instance, having a vector of input data $x \in \mathbb{R}^n$, a binary split function $h(\cdot)$ for j^{th} node of the tree is defined by

$$h(x, W_j) : \mathbb{R}^n \rightarrow \{0, 1\}, \tag{D.7}$$

where W_j is the split parameter associated with node j.[2] Depending on the output value being 0 or 1 (False or True), x is then sent over to one of the child nodes, where another split function is applied. This process continues until we reach a terminal node, where the final decision of the tree is readily available. Figure D.3 illustrates a simple decision tree with binary split functions.

The process that was described so far was related to testing, or in other words, making decisions about a single testing example. In order to train the tree, however,

[2] In equation (D.7), it is assumed that the split function operates on all elements of vector x. In practical applications, however, it is possible for each split function to process a different subset of elements in x.

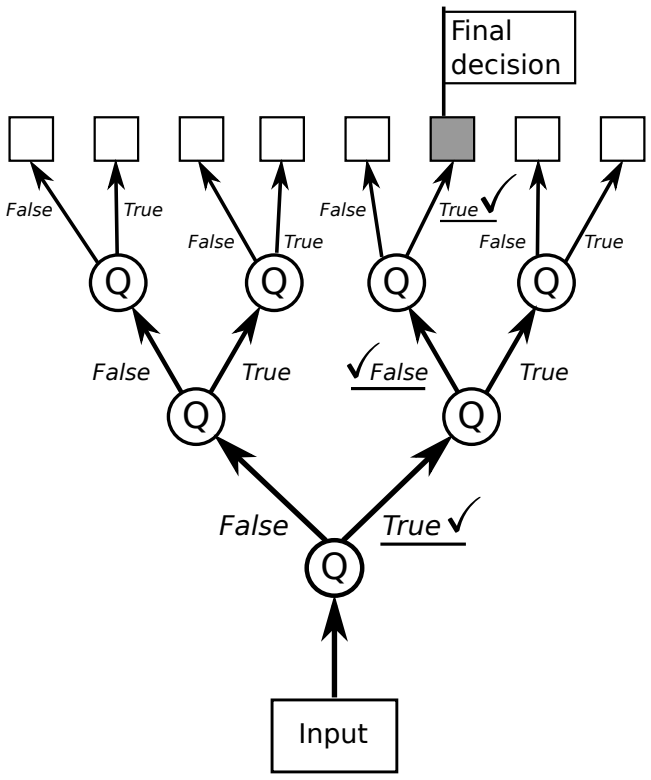


Figure D.3: Structure of a simple decision tree; Q stands for the question that is asked by split function of a node.

the whole set of training examples should be sent into the tree. Split parameters of each node are then learned in a greedy manner, starting from node 0 (the root node). The learning of split parameters is carried out by minimizing a desired energy function with respect to those parameters over the training set.

Alternatively, the same goal may be accomplished by maximizing an information-gain function. For this purpose, we assume that out of a training set T, the subset of training examples that are delivered to j^{th} node is denoted by T_j. Based on result of the test performed at this node, T_j is then once again split into two subsets, namely, T_j^0 and T_j^1. Hence, the information $\mathcal{I}$ gained after running the test at node j is defined by

$$\mathcal{I}(T_j, W) = \mathcal{H}(T_j) - \sum_{i \in \{0,1\}} \frac{|T_j^i|}{|T_j|} \mathcal{H}(T_j^i), \tag{D.8}$$

where $\mathcal{H}(\cdot)$ is the entropy and $|\cdot|$ indicates the cardinal number of a set. Hence, we may find split parameters of node j as follows:

$$W_j = \underset{W \in \mathcal{W}}{\arg\max}\ \mathcal{I}(T_j, W), \tag{D.9}$$

where $\mathcal{W}$ is the space of all possible split parameters.

After the learning of split parameters is completed,[3] each terminal node of the tree ends up with a subset of training examples. Each subset is then used to learn a *prediction model* for its pertinent terminal node. Prediction models of terminal nodes are required for interpreting previously unseen data. In this context, a popular choice for terminal node *predictors* is the MAP estimate.

Having described the testing and training of a decision tree, we now proceed to explain how we may group decision trees to form a forest. In so doing, the first requirement is to train the trees in a randomized manner. Two of the most common approaches for randomized training include:

i) **Bagging**, where each tree is trained with a different subset of training data that is randomly selected.

ii) **Randomized node optimization**, where split parameters of each node, which are learned through optimization, are restricted to remain within a subset $\mathcal{W}_{\text{sub}}$ that is randomly selected from the space of all possible split parameters $\mathcal{W}$.

Once all trees of a forest are trained properly, the forest can provide a unified decision about an input by combining the end results of all trees (e.g., using a simple averaging operation); this process is known as fusion . For details on random decision forests, refer to (Criminisi and Shotton, 2013).

D.3 Deep Belief Networks

A deep belief net is a hierarchical structure that is made by stacking a number of restricted Boltzmann machines (RBMs), a neural network first introduced by Smolensky (1986). Thus, a neural structure with multiple hidden layers may be formed, where each hidden layer, i.e., the output layer of one RBM, serves as the input (visible) layer of another. Figure D.4 depicts a RBM, where h represents the hidden layer and v represents the visible layer.

[3] Training of a tree is completed either when a prescribed height is reached for the tree, or when information gained by adding a new level to the tree is less than a desired threshold.

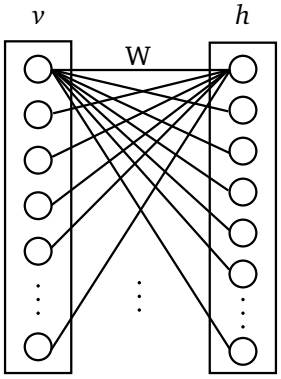


Figure D.4: Network of a RBM: The network is symmetric and fully connected, with no recurrent $h - h$ and $v - v$ connections, where v and h are visible and hidden layers of the RBM, respectively, and W represents connection wights of the network.

RBM is a bipartite graph; hence, given the activities of visible neurons, activities in the hidden layer are mutually independent. The same argument is also valid for the activities of visible neurons when activations of hidden neurons are given. Thus, assuming we have n visible and m hidden neurons in layers v and h, respectively, conditional probability distributions of visible and hidden layers may in the same order be expressed as follows:

$$p(v|h) = \prod_{i=1}^{n} p(v_i|h),$$

$$p(h|v) = \prod_{j=1}^{m} p(h_j|v),$$

where individual activation probabilities are defined by:

$$p(v_i = 1|h) = \text{sigm}\left(\sum_{j=1}^{m} W_{ij}h_j + b_i\right),$$

$$p(h_j = 1|v) = \text{sigm}\left(\sum_{i=1}^{n} W_{ij}v_i + c_j\right).$$

In these equations, sigm$(\cdot)$ is the sigmoid function, and b and c are bias terms for visible and hidden layers, respectively.

In order to define the joint probability distribution $p(v, h)$, an energy function $E(v, h)$ may be formed with respect to activities in both layers, which is defined by:

$$E(v, h) = -h^T W v - b^T v - c^T h.$$

Hence, we may write:

$$p(v, h) = \frac{1}{z}\, e^{-E(v,h)},$$

where z is a partition function computed by the sum of $e^{-E(v,h)}$ over all possible pairs of (v, h).

Accordingly, a RBM may be trained by computing connection weights that maximize the joint distribution of visible neurons over a dataset of interest **D**. This may be alternatively written as follows:

$$W^* = \arg\max_W \prod_{v \in \mathbf{D}} p(v),$$

where $p(v)$ is the marginal distribution for a visible neuron that is computed from $p(v, h)$. In this regard, Hinton, Osindero, and Teh (2006) proposed a greedy layer-wise algorithm for the training of RBMs in a deep belief net, which has shown to be a very efficient method.

Appendix E

Preprocessing Radar Data: Imbalance Correction of Amplitude and Phase

As a result of possible imperfections in different components of a radar system, the In-Phase and Quadrature components of received signals, denoted by I and Q, respectively, may be imbalanced both in phase ϕ, and amplitude A. Hence, it is necessary to correct those imbalances before using the received signals for detection and tracking purposes.

First, we consider the ideal I and Q components of the signal as follows:

$$\begin{cases} I = A\cos(\phi) \\ \\ Q = A\sin(\phi) \end{cases} \tag{E.1}$$

Now we suppose that the received signal is imbalanced in the following manner:

$$\begin{cases} I = A\cos(\phi) \\ \\ Q = A(1+\epsilon)\sin(\phi+\beta) \end{cases} \tag{E.2}$$

where β denotes phase imbalance ($0 \leq \beta \leq \frac{\pi}{2}$), and ϵ denotes amplitude imbalance ($0 \leq \epsilon < 1$). Thus, we may write,

$$\begin{aligned} I \times Q &= A^2\,(1+\epsilon)\,cos(\phi)\sin(\phi+\beta) \\ &= \frac{A^2}{2}\,(1+\epsilon)\,[\sin(2\,\phi+\beta)+\sin(\beta)] \end{aligned} \tag{E.3}$$

By considering ϕ as a random variable that has a uniform distribution in $[0, 2\pi)$, we may calculate the expected value of the right side of Eq. E.3. Assuming we have a large set of measurements corresponding to I and Q, denoted by $\{I, Q\}$, we can approximate the $\mathbb{E}[I \times Q]$ by the mean value of this large set. Thus, we may go on to write,

$$\begin{aligned}\mathbb{E}[I \times Q] &= \mathbb{E}\Big[\frac{A^2}{2}\,(1+\epsilon)\; cos(\phi)\sin(\phi+\beta)\Big] \\ &= \frac{A^2}{2}\,(1+\epsilon)\;\sin(\beta) \qquad \text{(E.4)} \\ &\approx < I \times Q >_{\{I\times Q\}} \qquad \text{(E.5)}\end{aligned}$$

where $< \cdot >$ denotes averaging operation. In a similar manner, we are able to calculate the expected values of $I \times I$ and $Q \times Q$, which may be expressed as follows:

$$\begin{aligned}\mathbb{E}[I \times I] &= \frac{A^2}{2} \qquad \text{(E.6)} \\ &\approx < I \times I >_{\{I\times I\}} \qquad \text{(E.7)}\end{aligned}$$

$$\begin{aligned}\mathbb{E}[Q \times Q] &= \frac{A^2}{2}\,(1+\epsilon)^2 \qquad \text{(E.8)} \\ &\approx < Q \times Q >_{\{Q\times Q\}} \qquad \text{(E.9)}\end{aligned}$$

Using the Equations E.5, E.7, and E.9, it is possible to calculate the values for phase imbalance β, and amplitude imbalance ϵ.

Appendix F

Radar Target Detection

In a typical problem of radar target detection, the traditional procedure is based on binary *hypothesis testing*. To elaborate, we define two hypotheses:

i) H_0, which describes the hypothesis of no target being present, and

ii) H_1, which identifies the presence of a target.

Then, we decide which one of the two hypotheses is true.

In the context of hypothesis testing, a popular procedure is to look to the Bayesian paradigm for guidance, whereby a likelihood ratio is defined as the ratio of two conditional pdfs, namely,

i) $p_{r|H_0}(R|H_0)$, pertaining to noise acting alone, and

ii) $p_{r|H_1}(R|H_1)$, pertaining to the presence of a target in additive noise.

where R is a received signal. To simplify matters, the hypothesis test is traditionally based on the log-likelihood ratio with no information loss (Van Trees, 1968). This ratio is then compared against a prescribed threshold; if the threshold is exceeded, the decision is then made in favour of H_1 for the target plus noise. Otherwise, the decision is made in favour of H_0 for the noise acting alone. This process may be expressed as follows:

$$\log \Lambda(R) = \log \frac{p_{r|H_1}(R|H_1)}{p_{r|H_0}(R|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \eta \tag{F.1}$$

where η is a properly chosen threshold. Naturally, the detection procedure summarized in this appendix is entirely different from the detection procedure based on pattern-recognition that is presented in this thesis.

Bibliography

Abdallah, S. and M. Plumbley (2006). Unsupervised analysis of polyphonic music by sparse coding. *Neural Networks, IEEE Transactions on 17*(1), 179–196. 23

Ablowitz, M. and A. Fokas (2003). *Complex variables: introduction and applications*, Volume 35. Cambridge University Press. 92

Adali, T. and S. Haykin (2010). *Adaptive signal processing: next generation solutions*. Wiley-IEEE Press. 41, 92, 96

Alspach, D. L. and H. W. Sorenson (1972). Nonlinear bayesian estimation using gaussian sum approximations. *Automatic Control, IEEE Transactions on 17*(4), 439–448. 81

Amiri, A. and S. Haykin (2014). Improved sparse coding under the influence of perceptual attention. *Neural Computation 26*(2), 377–420. 3

Anastassopoulos, V., G. A. Lampropoulos, A. Drosopoulos, and M. Rey (1999). High resolution radar clutter statistics. *Aerospace and Electronic Systems, IEEE Transactions on 35*(1), 43–60. 45

Arasaratnam, I. and S. Haykin (2009). Cubature kalman filters. *Automatic Control, IEEE Transactions on 54*(6), 1254–1269. 81

Arbib, M. A. (1981). Perceptual structures and distributed motor control. *Comprehensive Physiology*. 2

Au, W. W. (1993). *The sonar of dolphins*. Springer New York. 45

Bar, M. (2004). Visual objects in context. *Nature Reviews Neuroscience 5*(8), 617–629. 11

Bar-Shalom, Y., X. R. Li, and T. Kirubarajan (2001). *Estimation with Applications to Tracking and Navigation: theory algorithms and software*. Wiley-Interscience. 4, 16, 17, 81, 88

Barlow, H. (1961). The coding of sensory messages. *Current problems in animal behaviour*, 331–360. 20

Barlow, H. (1972). Single units and sensation: a neuron doctrine for perceptual psychology. *Perception 1*(4), 371–394. 21

Barlow, H. (2001). Redundancy reduction revisited. *Network: Computation in Neural Systems 12*(3), 241–253. 21

Beck, A. and M. Teboulle (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences 2*(1), 183–202. 63

Becker, S. (1993). Learning to categorize objects using temporal coherence. *Advances in neural information processing systems*, 361–361. 75

Becker, S. and Y. LeCun (1989). Improving the convergence of backpropagation learning with second order methods pages 29-37. *In Proceedings of the 1988 Connectionist Models Summer School*. 70

Bell, A. and T. Sejnowski (1997). The "independent components" of natural scenes are edge filters. *Vision research 37*(23), 3327–3338. 21

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends in Machine Learning 2*(1), 1–127. 5, 71

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems 19*, 153. 5

Bengio, Y. and Y. LeCun (2007). Scaling learning algorithms towards ai. *Large-Scale Kernel Machines 34*. 5

Berkes, P. and L. Wiskott (2005). Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision 5*(6). 31

Boyd, S. P. and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press. 23

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32. 99

Broca, P. (1861). Remarques sur le siège de la faculté du langage articulé, suivies d'une observation d'aphémie (perte de la parole). *Bulletin de la Société Anatomique 6*, 330–357. 9

Brodmann, K. (1905). Beitraege zur histologischen lokalisation der grosshirnrinde. *Psychol Neurol, Leipzig 4:176-226*. 9

Buchler, E. (1976). The use of echolocation by the wandering shrew (sorex vagrans). *Animal Behaviour 24*(4), 858 – 873. 45

Cadieu, C. F. and B. A. Olshausen (2012). Learning intermediate-level representations of form and motion from natural movies. *Neural computation 24*(4), 827–866. 41

Chang, C.-C. and C.-J. Lin (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST) 2*(3), 27. 52

Chikkerur, S., T. Serre, C. Tan, and T. Poggio (2010). What and where: A bayesian inference theory of attention. *Vision research 50*(22), 2233–2247. 4, 11

Cireşan, D. C., U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber (2011). Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two*, pp. 1237–1242. AAAI Press. 6

Conte, E., A. De Maio, and C. Galdi (2004). Statistical analysis of real clutter at different range resolutions. *Aerospace and Electronic Systems, IEEE Transactions on 40*(3), 903–918. 45

Coultrip, R., R. Granger, and G. Lynch (1992). A cortical model of winner-take-all competition via lateral inhibition. *Neural networks 5*(1), 47–54. 10

Criminisi, A. and J. Shotton (2013). *Decision forests for computer vision and medical image analysis*. Springer. 102

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society 39*(1), 1–38. 55

Desimone, R. and J. Duncan (1995). Neural mechanisms of selective visual attention. *Annual review of neuroscience 18*(1), 193–222. 3, 10

Dhrymes, P. J., L. R. Klein, and K. Steiglitz (1970). Estimation of distributed lags. *International Economic Review*, 235–250. 55

Donoho, D. and M. Elad (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proceedings of the National Academy of Sciences 100*(5), 2197. 27

Donoho, D., M. Elad, and V. Temlyakov (2006, jan.). Stable recovery of sparse overcomplete representations in the presence of noise. *Information Theory, IEEE Transactions on 52*(1), 6 – 18. 27, 28

Doya, K. (2007). *Bayesian brain: Probabilistic approaches to neural coding*. MIT Press. 11

Eacott, M. and D. Gaffan (1992). Inferotemporal-frontal disconnection: The uncinate fascicle and visual associative learning in monkeys. *European Journal of Neuroscience 4*(12), 1320–1332. 9

Edelman, G. and V. Mountcastle (1978). *The mindful brain: Cortical organization and the group-selective theory of higher brain function.* Massachusetts Inst of Technology Pr. 9

Einhäuser, W., C. Kayser, P. König, and K. Körding (2002). Learning the invariance properties of complex cells from their responses to natural stimuli. *European Journal of Neuroscience 15*(3), 475–486. 31

Fergus, R., P. Perona, and A. Zisserman (2003). Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, Volume 2, pp. II–264. IEEE. 12

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation 3*(2), 194–200. 31, 75

Fraser, D. and J. Potter (1969). The optimum linear smoother as a combination of two optimum linear filters. *Automatic Control, IEEE Transactions on 14*(4), 387–390. 57

Fraser, D. C. (1967). *A new technique for the optimal smoothing of data*. Ph. D. thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics. 3

Friston, K. (2003). Learning and inference in the brain. *Neural Networks 16*(9), 1325–1352. 11

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience 11*(2), 127–138. 56

Fuster, J. (1989). *The prefrontal cortex* (Second ed.). Raven Press, New York. 2

Fuster, J. (1997). Network memory. *Trends in Neurosciences 20*(10), 451–459. 9

Fuster, J. (2003). *Cortex and mind: unifying cognition*. Oxford University Press. New York. 3, 4, 9, 10, 34

Fuster, J. (2008). *The prefrontal cortex* (Fourth ed.). Academic Press. 3

Fuster, J. M. (2014). The prefrontal cortex makes the brain a preadaptive system. 10, 11

Gall, F. J. (1825). *Sur les fonctions du cerveau*, Volume 5. Chez l'auteur, Boucher, Bossange père, Béchet jeune, JB Baillière. 9

Gao, J., J. Hu, W.-W. Tung, Y. Cao, N. Sarshar, and V. P. Roychowdhury (2006, Jan). Assessment of long-range correlation in time series: How to avoid pitfalls. *Phys. Rev. E 73*, 016117. 45

George, D. and J. Hawkins (2005). A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, Volume 3, pp. 1812–1817. IEEE. 12

Goodrich, R. and P. E. Caines (1979). Linear system identification from nonstationary cross-sectional data. *Automatic Control, IEEE Transactions on 24*(3), 403–411. 55

Gordon, N. J., D. J. Salmond, and A. F. Smith (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, Volume 140, pp. 107–113. IET. 72, 81

Granger, R. (2006). Engines of the brain: The computational instruction set of human cognition. *AI Magazine 27*(2), 15. 4

Griffin, D. R. (1944). Echolocation by blind men, bats and radar. *Science 100*(2609), pp. 589–590. 45

Gupta, N. K. and R. K. Mehra (1974). Computational aspects of maximum likelihood estimation and reduction in sensitivity function calculations. *Automatic Control, IEEE Transactions on 19*(6), 774–783. 55

Hadamard, J. (1902). Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin 13*(49-52), 28. 22

Håstad, J. and M. Goldmann (1991). On the power of small-depth threshold circuits. *Computational Complexity 1*(2), 113–129. 5

Hayek, F. A. (1952). *The sensory order*. University of Chicago Press, Chicago. 9

Haykin, S. (2001). *Kalman filtering and neural networks*. Wiley Online Library. 15, 56

Haykin, S. (2009). *Neural networks and learning machines*. Prentice Hall. 22, 70

Haykin, S. (2012). *Cognitive Dynamic Systems*. Cambridge University Press. 1, 2, 11, 81

Haykin, S. (2013). *Adaptive Filter Theory* (Fifth ed.). Pearson. 18, 85

Haykin, S. and J. M. Fuster (2014). On cognitive dynamic systems: Cognitive neuroscience and engineering learning from each other. *Proceedings of the IEEE 102*(4), 608–628. 11

Haykin, S., Y. Xue, and P. Setoodeh (2012). Cognitive radar: Step toward bridging the gap between neuroscience and engineering. *Proceedings of the IEEE 100*(11), 3102–3130. 2

Hinton, G. E. (2010). Learning to represent visual input. *Philosophical Transactions of the Royal Society B: Biological Sciences 365*(1537), 177–184. 5

Hinton, G. E., P. Dayan, B. J. Frey, and R. M. Neal (1995). The" wake-sleep" algorithm for unsupervised neural networks. *SCIENCE-NEW YORK THEN WASHINGTON-*, 1158–1158. 12

Hinton, G. E., S. Osindero, and Y.-W. Teh (2006). A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554. 104

Hinton, G. E. and R. R. Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. *Science 313*(5786), 504–507. 71

Hu, J., W. wen Tung, and J. Gao (2006). Detection of low observable targets within sea clutter by structure function based multifractal analysis. *Antennas and Propagation, IEEE Transactions on 54*(1), 136–143. 45

Hubel, D. H. and T. N. Wiesel (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology 160*(1), 106. 5

Hyvärinen, A. (1999). Sparse code shrinkage: Denoising of nongaussian data by maximum likelihood estimation. *Neural computation 11*(7), 1739–1768. 22

Hyvärinen, A. and P. Hoyer (2000). Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation 12*(7), 1705–1720. 21, 31

Hyvärinen, A., J. Hurri, and J. Väyrynen (2003). Bubbles: a unifying framework for low-level statistical properties of natural image sequences. *JOSA A 20*(7), 1237–1252. 31

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural networks 1*(4), 295–307. 70

Jarrett, K., K. Kavukcuoglu, M. Ranzato, and Y. LeCun (2009). What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153. IEEE. 6, 63

Jones, R. H. (1980). Maximum likelihood fitting of arma models to time series with missing observations. *Technometrics 22*(3), 389–395. 55

Julier, S. J. and J. K. Uhlmann (1997). A new extension of the kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, Volume 3, pp. 3–2. Orlando, FL. 81

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering 82*(1), 35–45. 17

Karklin, Y. and M. Lewicki (2005). A hierarchical Bayesian model for learning nonlinear statistical regularities in nonstationary natural signals. *Neural Computation 17*(2), 397–423. 12, 31

Kavukcuoglu, K., P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun (2010). Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, pp. 1090–1098. 63

Knill, D. C. and W. Richards (1996). *Perception as Bayesian inference*. Cambridge University Press. 4, 10, 11

Körding, K. and D. Wolpert (2006). Bayesian decision theory in sensorimotor control. *Trends in cognitive sciences 10*(7), 319–326. 4, 11

Kreutz-Delgado, K. (2009). The complex gradient operator and the cr-calculus. *arXiv preprint arXiv:0906.4835*. 96

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*, Volume 1, pp. 4. 6

Lashley, K. (1950). In search of the engram. In *Symposia of the society for experimental biology*, Volume 4, pp. 454–482. 9

LeCun, Y. and Y. Bengio (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*. 5, 6, 62

Lee, H., C. Ekanadham, and A. Ng (2007). Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems*, pp. 873–880. 5

Lee, H., R. Grosse, R. Ranganath, and A. Y. Ng (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 609–616. ACM. 6

Lee, T. S. and D. Mumford (2003). Hierarchical bayesian inference in the visual cortex. *JOSA A 20*(7), 1434–1448. 4, 12

Levy, W. and R. Baxter (1996). Energy efficient neural codes. *Neural Computation 8*(3), 531–543. 21

Lewicki, M. S. and T. J. Sejnowski (1997). Bayesian unsupervised learning of higher order structure. *Advances in neural information processing systems*, 529–535. 12

Lustig, M., D. Donoho, and J. M. Pauly (2007). Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic resonance in medicine 58*(6), 1182–1195. 22

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*(4), 115–133. 97

Miller, E. K. and J. D. Cohen (2001). An integrative theory of prefrontal cortex function. *Annual review of neuroscience 24*(1), 167–202. 3

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks 6*(4), 525–533. 70

Mountcastle, V. B. (1957). Modality and topographic properties of single neurons of cat's somatic sensory cortex. *J. Neurophysiol 20*(4), 408–434. 5

Mumford, D. (1994). Pattern theory: a unifying perspective. In *First European congress of mathematics*, pp. 187–224. Springer. 10

Neisser, U. (1976). *Cognition and reality: principles and implications of cognitive psychology*. WH Freeman. 2, 3

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $o(1/k^2)$. In *Soviet Mathematics Doklady*, Volume 27, pp. 372–376. 70

Neuweiler, G. (1984). Foraging, echolocation and audition in bats. *Naturwissenschaften 71*(9), 446–455. 45

Olshausen, B. A. and D. J. Field (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature 381*(6583), 607–609. 7, 21, 38

Olshausen, B. A. and D. J. Field (1997). Sparse coding with an overcomplete basis set: A strategy employed by vi? *Vision research 37*(23), 3311–3326. 21

Olshausen, B. A. and D. J. Field (2004). Sparse coding of sensory inputs. *Current opinion in neurobiology 14*(4), 481–487. 21

Picinbono, B. (1996, oct). Second-order complex random vectors and normal distributions. *Signal Processing, IEEE Transactions on 44*(10), 2637 –2640. 44

Posner, M. I. (2011). *Cognitive neuroscience of attention* (Second ed.). The Guilford Press. 3, 10

Powell, T. and V. B. Mountcastle (1959). Some aspects of the functional organization of the cortex of the postcentral gyrus of the monkey: a correlation of findings obtained in a single unit analysis with cytoarchitecture. *Bulletin of the Johns Hopkins Hospital 105*, 133–162. 5

Raina, R., A. Battle, H. Lee, B. Packer, and A. Y. Ng (2007). Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pp. 759–766. ACM. 22

Ranganath, C., M. Cohen, C. Dam, and M. D'Esposito (2004). Inferior temporal, prefrontal, and hippocampal contributions to visual working memory maintenance and associative memory retrieval. *The Journal of Neuroscience 24*(16), 3917–3925. 9

Ranzato, M., F. J. Huang, Y. L. Boureau, and Y. LeCun (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07.*, pp. 1–8. 6, 22

Ranzato, M., F. J. Huang, Y.-L. Boureau, and Y. Lecun (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8. IEEE. 63

Ranzato, M., C. Poultney, S. Chopra, and Y. LeCun (2006). Efficient learning of sparse representations with an energy-based model. *Advances in neural information processing systems 19*. 71

Rao, R. and D. Ballard (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience 2*, 79–87. 12

Rao, R., B. A. Olshausen, and M. Lewicki (2002). *Probabilistic models of the brain: Perception and neural function*. The MIT Press. 4, 11

Rao, R. P. (2005). Bayesian inference and attentional modulation in the visual cortex. *Neuroreport 16*(16), 1843–1848. 4

Rauch, H. E., C. Striebel, and F. Tung (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA journal 3*(8), 1445–1450. 57

Remmert, R. (1991). *Theory of complex functions*, Volume 122. Springer. 94

Reynolds, J. H. and L. Chelazzi (2004). Attentional modulation of visual processing. *Annual Review of Neuroscience 27*, 611–647. 33, 34

Riedmiller, M. and H. Braun (1993). A direct adaptive method for faster back-propagation learning: the rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pp. 586–591 vol.1. 70

Rojas, J. A. M., J. A. Hermosilla, R. S. Montero, and P. L. L. Espi (2009). Physical analysis of several organic signals for human echolocation: oral vacuum pulses. *Acta Acustica united with Acustica 95*(2), 325–330. 45

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*(6), 386. 97

Rosenblatt, F. (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Spartan, Washington DC. 4, 99

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. *Cambridge, MA*, 318–362. 6, 70

Rumelhart, D. E. and J. L. McClelland (1986). *Parallel distributed processing: Psychological and biological models*, Volume 2. The MIT press. 9, 99

Sangston, K., F. Gini, and M. Greco (2010). New results on coherent radar target detection in heavy-tailed compound-gaussian clutter. In *Radar Conference, 2010 IEEE*, pp. 779–784. 45

Schaul, T., S. Zhang, and Y. LeCun (2013). No more pesky learning rates. In *International Conference on Machine Learning (ICML)*. 70

Shannon, C. (1951). Prediction and entropy of printed english. *Bell System Technical Journal 30*(1), 50–64. 21

Shumway, R. H. and D. S. Stoffer (1982). An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis 3*(4), 253–264. 90, 91

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. 102

Sorenson, H. W. and D. L. Alspach (1971). Recursive bayesian estimation using gaussian sums. *Automatica 7*(4), 465–479. 81

Talbot, S. and W. Marshall (1941). Physiological studies on neural mechanisms of localization and discrimination. *Amer. J. Ophtalmology 24*, 1225–1263. 5

Taylor, G. W., R. Fergus, Y. LeCun, and C. Bregler (2010). Convolutional learning of spatio-temporal features. In *Computer Vision–ECCV 2010*, pp. 140–153. Springer. 6

Thompson, J., C. Woolsey, and S. Talbot (1950). Visual areas i and ii of cerebral cortex of rabbit. *Journal of Neurophysiology 13*, 177–188. 5

Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. In *Soviet Math. Dokl.*, Volume 5, pp. 1035. 22, 99

Tomita, H., M. Ohbayashi, K. Nakahara, I. Hasegawa, and Y. Miyashita (1999). Top-down signal from prefrontal cortex in executive control of memory retrieval. *Nature 401*(6754), 699–703. 9

Torralba, A., A. Oliva, M. S. Castelhano, and J. M. Henderson (2006). Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. *Psychological review 113*(4), 766. 4

Tropp, J. (2006, march). Just relax: convex programming methods for identifying sparse signals in noise. *Information Theory, IEEE Transactions on 52*(3), 1030 –1051. 26

Tropp, J. and S. Wright (2010). Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE 98*(6), 948–958. 23

Uexküll, J. v. (1926). Theoretical biology. *Kegan, Paul, Trench, Tubner, London*. 10

Van Trees, H. (1968). Detection, estimation, and modulation theory, Part I. *Massachusetts institute of technology, John Wiley and Sons*. 18, 24, 107

von Weizsäcker, V. (1950). *Der Gestaltkreis*. Stuttgart, Thieme. 2

Wiener, N. (1948). *Cybernetics: or the Control and Communication in the Animal and the Machine*. John Wiley. 2

Wirtinger, W. (1927). Zur formalen theorie der funktionen von mehr komplexen veränderlichen. *Mathematische Annalen 97*(1), 357–375. 41, 92

Wiskott, L. and T. Sejnowski (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural computation 14*(4), 715–770. 31

Wu, C. J. (1983). On the convergence properties of the em algorithm. *The Annals of statistics*, 95–103. 55