SPATIOTEMPORAL PATTERN DETECTION IN MULTI-CELL RECORDINGS USING UNSUPERVISED LEARNING

by

Hassan Nikoo

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2015

To my beloved parrents,

Shahrzad and Hamid,

for their endless love, support and encouragement.

Table of Contents

List of Tables

List of Figures

Abstract

Detection of spatiotemporal patterns have many applications in areas such as computer vision and data mining. Specifically, the analysis and mining of biological data with high dimensionality (e.g. multi-cell recordings, fMRI) are heavily dependent on detection of these patterns. In this thesis, we propose two unsupervised learning algorithms for obtaining filters that capture temporal patterns. In particular, we are interested in applying our methods for detection of regularities in multi-cell recordings of neurons. We propose two approaches: convolutional restricted Boltzmann machine (RBM) and convolutional denoising auto-encoder. The experimental results demonstrate that the proposed methods are able to detect temporal patterns in artificial data and multi-cell recordings from rat's brain. Moreover, we propose a Monte Carlo method for quantitatively evaluating the convolutional RBM by estimating the log-likelihood of data under the model distribution. The experimental results on test dataset of handwritten digits (MNIST), demonstrate that the convolutional RBM can learn a good generative model with small number of parameters.

List of Abbreviations Used

AIS Annealed Importance Sampling. 39

CD Contrastive Divergence. 21

CDAE Convolutional Denoising Auto-encoder. 32

CRBM Convolutional Restricted Boltzmann Machine. 19

DBN Deep Belief Network. 18

RBM restricted Boltzmann machine. 18

SM Score Matching. 44

Acknowledgements

This is a great opportunity for me to publicly thank those who have been influential during my studies at Dalhousie University. I am profoundly indebted to my dear supervisor, Professor Thomas Trappenberg, because of his calmness, endless support and deep insights. Whenever I was disappointed of ideas, he inspired for more endeavor. I thank my mom and dad and my little brother for their unconditional support and love even from oversees. I would like to thank my great friend Paul for his endless efforts and patience in answering my questions. I thank Paul for his close cooperation and helpful discussions. I would like to thank my dear friends, Hossein and Farzaneh for their supports and warm friendship. I was fortunate to be a member of HAL lab at Dalhousie University. I kindly acknowledge my fellow students at lab for their help, idea and support. Finally, I warmly thank Dr. Dirk Arnold and Dr. Sageev Oore for reading this thesis and their insightful feedbacks.

Chapter 1

Introduction

1.1 Motivation

The goal of machine learning is to build systems that can learn from complex data and make accurate predictions for previously unseen data. In the past few decades, machine learning has been successful in tackling many real world problems. For example it has been successfully applied in optical character recognition [56, 22, 14, 12], face detection [71, 13, 59, 72], and speech recognition [26, 24, 61, 21]. Moreover it has been a powerful tool in solving the challenging problems in computer vision [35, 73], natural language understanding [5], autonomous car driving [52, 47], data mining of biological data [67, 15], medical imaging [30], and web search/information retrieval [62].

Many real-world machine learning applications require a good feature representation to be successful. A good feature representation is a representation that has captured regularities or patterns in the data. In machine learning two paradigms exist in general for modeling the regularities for obtaining good representations: the engineering, or design paradigm, and the learning paradigm. The design paradigm is based on the knowledge of human experts about the structure of regularities and patterns. The design paradigm is the classical approach of machine learning for describing content of data. On the other hand, the learning paradigm is concerned with devising algorithms that can "learn" to capture the regularities and ideally learn a good representation of data. Indeed, the algorithm learns the feature representation of the input.

The most challenging machine learning tasks are defined over high dimensional temporal data types such as video sequences, acoustic data, etc. These tasks require modeling the temporal regularities of data to represent the content. For example video classification is based on modeling regularities of moving objects in the frames in order to extract features that could be used to classify content of video. The

prospect of unsupervised feature learning has recently gained much attention for facing the challenges of high dimensional data.

The unsupervised feature learning algorithms can learn regularities and patterns from unlabeled data in order to transform them to better representations. The interesting aspect is that it is possible to build hierarchical representations. This approach is called "deep unsupervised feature learning". The deep methods learn structures of data by capturing simple concepts or regularities first and then successfully build up more complex concepts by composing the simpler ones together. At each level of hierarchy the algorithm learns a more abstract representation which captures regularities of that level. In recent years, this approach has gained significant interest as a way of building hierarchical representations [28, 6, 53, 8, 36]. Moreover, applying the deep feature learning methods to machine learning tasks with high dimensional temporal data has improved the performance of state of the art methods [48, 73, 26, 49].

Many machine learning tasks and data mining applications work with biological temporal data with high dimensionality such as fMRI, and multi-cell recordings of neurons. Specifically, the problem of detecting the spatiotemporal patterns from these data is very challenging and currently depends on the design paradigm. The challenges of this applications can be tackled by unsupervised learning algorithms. These approaches typically learn a group of filters or pattern detectors to obtain better representations. These filters capture the motifs or regularities in the input. Hence, the task of detecting spatiotemporal patterns from biological data can be tackled by learning such filters. In the recent years, such approaches have been successfully applied on biological data. For example, recently an unsupervised method has been used for learning temporal representation of fMRI to classify memories of brain [18]. Moreover, another similar approach has been proposed for identifying brain networks and temporal activations of the brain regions [30].

In this thesis, we are interested in solving the problem of detection of spatiotemporal patterns; specifically, the problem of detection of patterns in multi-cell recordings which contain simultaneous records of many neurons' spike. The multi-cell recordings are obtained by inserting micro-electrodes into animals' brains and recording temporal firing of multiple neurons. The spatiotemporal patterns in these data encapsulate

the perceived information by brain. For example, it has been shown that firing patterns are directly correlated with location of an animal in a given environment [64] or the past experiences of the animal [19, 31, 16]. Moreover, these patterns have been shown to be packets of information that are being transfered from one brain region to another [42]. Solving the task based on the design paradigm is unpractical because the structure of patterns are not known. However, there are extensive works on modeling and analyzing structure of patterns from multi-cell recordings by using algorithms such principle component analysis [9] , Clustered factor analysis [10] and many latent variable models [32, 11, 44, 58]. We tackle this task by proposing approaches that can learn to detect the patterns. Developing such a system can fundamentally help the neuroscience research studies of multi-cell recordings.

1.2 Proposed Methods

This work presents two methodologies for discovering spatiotemporal patterns in multi-cell recordings. The first method is a heuristic search which is our baseline algorithm. The other method is based on unsupervised learning. This approach consists of two stages. The first stage is an unsupervised learning process. We propose two algorithms to be used in this stage. The learning algorithms provide a group of filters which have captured the patterns in the data. The next stage of our methodology is to use the learned filters for discovering the patterns. Figure 1.1 shows the schematic overview of our system. The input of the system is a binary temporal dataset and the outputs are the discovered patterns. In the following, we will give a brief explanation of the heuristic search and the unsupervised learning methods.

The proposed search algorithm is in principle a brute force search which can discover temporal motifs that occur with sufficient frequency. What this search method does, is roughly the following. The algorithm randomly selects a part of data as a potential pattern and computes the frequency of its occurrences. If the frequency is high enough, the procedure continues by searching for other parts of the possible pattern. The method does not assume anything about structure of patterns which makes it a generic search. However, it is extremely vulnerable when the patterns have variations. We use this algorithm as a baseline algorithm for the spike pattern detection task and we compare its accuracy with our other proposed methods.

The first unsupervised algorithm is based on a generative probabilistic model called restricted Boltzmann machine (RBM). The RBM is a bipartite stochastic network with two layers, where a set of latent (or hidden) binary random variables are densely connected to a set of input (or visible) variables. However, there are no connections between hidden variables or between visible variables. The joint distribution is described by an energy function which is simply a sum of products between the hidden, visible units, and the weights between the two layers. The model is trained by maximizing the likelihood of observed inputs under model distribution. Roughly speaking, after training, the RBM represents its input data using latent variables. The learned weights are pattern detectors and latent variables represent presence of patterns. This model has been basis of representation learning for many applications: human motion learning [66], object recognition [54, 7, 1], information retrieval [55] in which it has helped classification tasks by learning good pattern detectors to transform corresponding data to better representations.

Although RBM has shown promise even for learning pattern detectors from temporal data, dimensionality of the input space is fairly limited. Since our pattern recognition task is concerned with high dimensional temporal datasets, we use a shared parameter model of RBM called convolutional RBM (CRBM). The challenge of these data is that spatiotemporal motifs can occur at any location of input space. To address this issue the convolutional RBM has the property that its pattern detectors share their filters among input space. In this fashion, a feature detector which captures useful information in one part of an input can pick up the same information elsewhere. This model has been used for many high dimensional temporal data classification tasks such as speech recognition [40] and video classification [60]. We will use this model to learn the desired filters to capture the temporal patterns.

The other proposed unsupervised learning method, is based on denoising auto-encoder (DAE). The auto-encoder is basically a neural network which reconstruct its input. In other words, the output of neural network is the input itself. Denosing auto-encoder is a stochastic version where the input is stochastically corrupted, but the uncorrupted input is still used as target for the reconstruction. We use the shared parameter version of DAE known as convolutional DAE for the temporal pattern recognition. The reason is that spatiotemporal patterns occur at arbitrary location

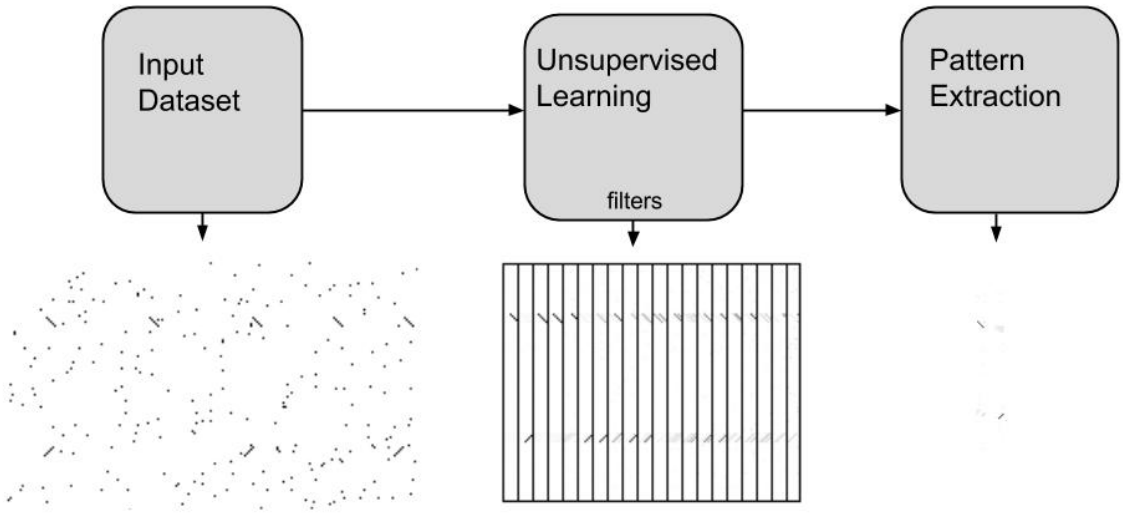


Figure 1.1: The schematic overview of proposed spatiotemporal pattern detection system.

of input space and convolutional DAE can learn pattern detectors that deal with this issue. This model has been used for applications that deal with temporal data such as video classification [20], speech recognition [33], sequence classification [2].

In this work, we use various techniques for a better training of models. The performance of our system is extremely dependent on the learned filters. Therefore, many regulations are added to the learning process to obtain best filters. The sparsity regularizations are incorporated to the training procedures to learn distinct and clear filters. Moreover, an over-fitting prevention technique is proposed for convolutional RBM which uses the free-energy of model. Furthermore, a quantitative analysis is performed for obtaining a better insight to the learned models. For this analysis, the annealed importance sampling is used to estimate the log-likelihood of data under CRBM distribution. This estimation for the training set and validation gives an insight into the learned model.

In this thesis, the proposed approaches are evaluated on artificial datasets as well as empirical multi-cell recordings. An algorithm is proposed for generating artificial data that has similar properties of empirical multi-cell recording while it has known patterns. The experiments shows that all proposed methods are able to detect the patterns of artificial data. Moreover, the performance of proposed approaches against changes in the data such as noise, variations of patterns and scale of input

space, are evaluated. The results suggest that convolutional RBM model has a superior performance over other approaches. Furthermore, the experiments using CRBM on empirical data suggest that this model is able to detect shared patterns in two multi-cell recordings.

1.3 Summary of Contributions

The main contributions of this thesis are as follows:

- We present a generic heuristic search algorithm for detecting spatiotemporal patterns. By applying the method on artificial data, we successfully extract the inserted patterns.

- We propose an algorithm to use the filters of learning models and extract occurring spatiotemporal patterns. This algorithm is able to select the best detected patterns without any manual inspection.

- This thesis proposes a technique for prevention of over-fitting of convolutional RBM. For this technique, the free energy of model is used as an indicator of over-fitting. This method played an important role for a better training CRBM.

- We use a Monte Carlo method, namely Annealed importance Sampling (AIS), for convolutional RBM to estimate the log-likelihood of data under model distribution. We develop the necessary formulations to apply the AIS on CRBM. By this method, we are able to do a quantitative analysis of trained models by comparison of log-likelihoods.

- We show that training criterion of convolutional DAE is equivalent to score matching of an energy-based model. The free-energy of this energy model can be used as a over-fitting indicator.

- We present an algorithm for generating synthetic multi-cell recording which has the similar properties of empirical data and has known injected spatiotemporal patterns. The generated data of our algorithm are used to evaluate the proposed detection system.

- We apply the proposed method on a empirical multi-cell recording and we show that there are common patterns between the spontaneous activities and stimuli-driven activities in rat's brain.

1.4 Organization of Thesis

The organization of the thesis is as follows. Chapter 2 will present some Neuroscience background, a brief description of what firing patterns represent and the goal of applying our proposed algorithms on the data. In Chapter 3, we will discuss the heuristic search algorithm. Chapter 4 presents the convolutional restricted Boltzmann machine and discusses the training algorithm and necessary monitoring procedures, also its results on MNIST dataset. Chapter 5 presents the convolutional denoising auto-encoder and compares the learned filters of this model with different regularizations. Chapter 6 discusses the Monte Carlo method for estimating the log-likelihood of convolutional RBM and compares the results of different models. Chapter 7 presents our proposed algorithm for using the learned filters of unsupervised learning and extracting the patterns of spike data. In the Chapter 8, the experiments on the our synthetic multi-cell recoding and empirical collected data are discussed. Finally Chapter 9 provides a summary and the conclusion.

Chapter 2

Background

2.1 Introduction

This chapter presents a brief background about Neuroscience and the importance of pattern discovery on neuronal multi-cell recordings. We describe briefly some neuroscience studies and the discoveries by cell recording from brain. Then, open questions and possible research applications of pattern detection system are discussed.

2.2 Neuroscience Background

The neural network of the brain consists of billions of neurons. Neurons process and transmit information through electro-chemical signals. A short-lasting raise and fall of electrical potential of a cell is called action potential which is known as spike or fire of a neuron. They play a major role in cell-to-cell communication of signals. Moreover, there are different ways of classifying neurons. One way of classification is based on location and shape of the cell where some of them are: Pyramidal cells, Basket cells, Spindle cells and etc. Moreover, there are different ways to categorize regions of the brain but a major one is: frontal lobe, parietal lobe, occipital lobe, temporal lobe, cerebellum and brainstem.

The classical approach of learning about neurons is recording neuronal activities of animals. The typical method is that we insert a micro-electrode system into animal's brain to measure when neurons generate action potentials. These systems have single-unit and multi-unit recorder devices. The simultaneous recording of multi-cells have provided researchers with rich information about the firing rate, frequency, patterns, etc. There are many informative studies by cell recordings that we will discuss to give a brief understanding of what we can learn by the recording of neurons.

Multi-cell Recording and Memory Replay

Many important studies using cell recording are made in the Hippocampus. The Hippocampus is located in the medial temporal lobe of the brain and is considered to play important roles in the consolidation of information from short-term memory to long-term memory. The study by John O'Keefe et al. was conducted by recording firing patterns of multiple pyramidal cells in the Hippocampus of rats. The striking results of this research showed that there is a strong correlation between the spatial location of rat in a given environment and the sequence of firing pattern in the Hippocampus [51]. The recorded pyramidal cells in the Hippocampus for this reason are called the place cells in the brain. The fact that the patterns of spikes are in direct correlation with the external stimuli drew attention for more studies.

Another important study by recording was by Skaggs and McNaughton [64] which showed that the patterns of activity of neurons of rats during exploration of environment are replayed again during sleep. In other words, the study suggested that temporally sequenced memories for previous experiences are reactivated again during sleep. Moreover, Foster and Wilson [19] showed that sequential replay also occurs during "awake periods" immediately after environmental exploration of rats. A large number of studies in the recent years have been done in this regard and have suggested that patterns of spikes in resting or sleep replays the firing patterns seen during prior behaviors [16, 31]. In addition, it was not only patterns of behaviors were replayed. The replays have been shown in the visual cortex in the absence of behavioral experience [34, 31]. In other words, the patterns of spikes were replayed after a visual stimulus were presented to animal.

All in all, it has been argued that the replay of sequences might play a major role in memory and learning mechanism [19]. Therefore, understanding more about the sequences and the firing patterns of neurons can help researchers understand the learning procedures and memory mechanism of brain. The replay of sequences of firing were not the only discovery by multi-cell recordings. In the following we discuss another discovery which shows that firing patterns also play a role in transferring information from one region of the brain to the others.

Multi-cell Recording and Packet-like Patterns

In recent years, Luczak et al. [41, 43] have been studying the neuronal population spiking patterns of rats in auditory cortex. Their studies have suggested that the repetition of patterns evoked by external stimuli, during spontaneous activities (spikes when rat is sleeping or resting) do not necessarily imply replay of the stimuli. In these studies, they record neurons in the auditory cortex of rats. Therefore, auditory stimulation was used as external input in order to record evoked responses of neurons. The results have shown that there are common structures conserved among various sensory stimuli-driven activities and spontaneous activities. They argued that due to circuitry of connections of the neurons, organization of activities (patterns) have "limited vocabulary". Therefore it is speculated that spiking patterns have limited structures that we might be able to discover.

In [42], Luzak et al. made another breakthrough by showing that population activity of neurons can be viewed as packets. They showed that packet-like structure is an ever-present feature of spontaneous and stimulus evoked cortex activities. This means that spikes of a group of neurons is roughly speaking a package of information which has a certain structure. They suggest that the packets reflect an opening of a gate that allows some part of the brain transmit a representation of external information to other brain regions. In other words, different regions of the brain communicate with packets of informations with certain structures. This implies that learning more about structure of patterns can teach us more about communication of information between regions of the brain.

2.3 Statistical Analysis of Empirical Multi-cell Recording

In the following, we describe a statistical analysis of our empirical data to demonstrate the basic known structure in the data that we will work with. The multi-cell recordings of this thesis are taken from [41]. The data are from experiments on multiple rats. The recording device of these experiments use silicon micro-electrode to collect data from auditory cortex. The system recorded simultaneously from populations of $40 - 100$ neurons. During the experiments rats were under urethane anesthesia while presenting 500 ms long auditory stimulus (tone or natural sound snipets) every

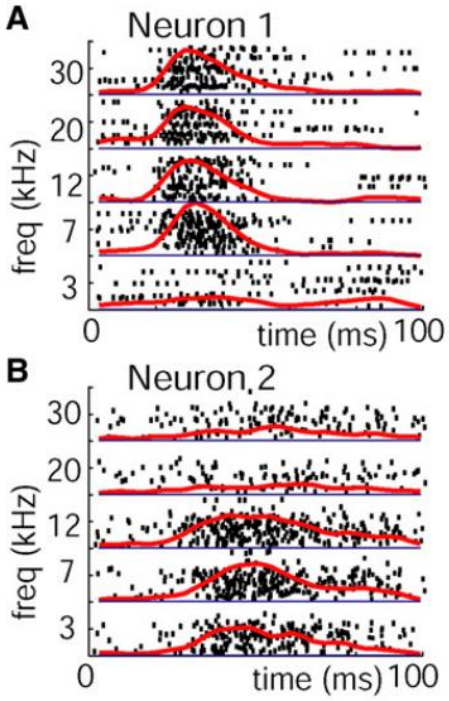


Figure 2.1: Plot of spikes of representative neurons in response to a pure tone with various frequencies. The solid line is the average time of spikes and is called perievent time histograms (PETHs). The plot is taken from [41] with kind permission of author.

2 second.

The following analysis can be carried out to demonstrate the sequential structure of firing patterns (full description is in [41]). First, let's look at the individual activity of neurons in response to external input. Figure 2.1 demonstrate raster plot of spikes of two selected neurons in response to various tone frequencies. Vertical axis shows frequencies and for each frequency there are multiple trials. The solid line is called perievent time histograms (PETHs) which is basically the average number of spikes over different trials. It can be observed that PETH of each individual neuron has a stereotypical pattern. In other words, the response time of each neuron to stimulus is different than others. This stereotyped PETH suggest that responses of neurons at population level might have a sequential organization. Figure 2.2 visualizes the sequential organization. Figure 2.2 shows a gray-scale bar denoting the mean PETHs over all tones and trials within onset 100 ms response. These gray-scale bars are sorted by mean response (spike) time of neurons(MSL) depicted by red dots.

This simple analysis demonstrates the global structure of firing pattern evoked by

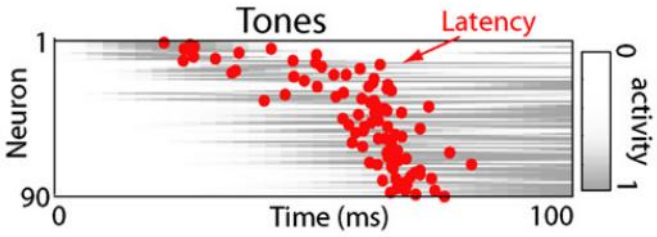


Figure 2.2: Mean activity of 90 simultaneously recorded neurons to tone stimuli. Gray bars represent Neuron's PETH normalized between 0 and 1. Red dots denote each neurons mean spike time (latency of neuron's response to stimuli also known as MSL)in the 100 ms after tone onset. Neurons are ordered vertically by MSL to illustrate sequential spread of activity. The plot is taken from [41] with kind permission of author.

stimuli. The same analysis for activities not evoked by stimuli (spontaneous spikes) also shows the similar global sequential structure. Further analysis described in the paper suggested that there is a correlation between the order of neurons spiking in the sequence of stimulus-locked spikes and spontaneous spikes. The paper concludes that the structure of activities have a limited vocabulary which means there are limited patterns in the activity of a population of neurons.

2.4 Open Questions

Given the described studies and analysis, the importance of recognition of firing patterns in the brain can be realized. The firing patterns of Hippocampus have a direct correlation with environment representation in the brain and they are replayed for memory consolidation. The structure of these patterns are limited and they represent information packages in the brain. Therefore, discovering the new structures in these data can help us learn more about the brain. The researches raise the question of what other structures and organizations are in the cortical activities? What kind of messages, are transfered between the regions of the brain? Answering these questions needs data mining tools that can discover new patterns in these data.

We can define the data as sequence of vectors that contain activity of neurons over time. In other words, the multi-cell recording of N_v neurons over N_T times as a binary matrix of $N_v \times N_T$. Figure 2.3 shows a portion of recorded data provided by [41] that is recorded from auditory cortex of rats.

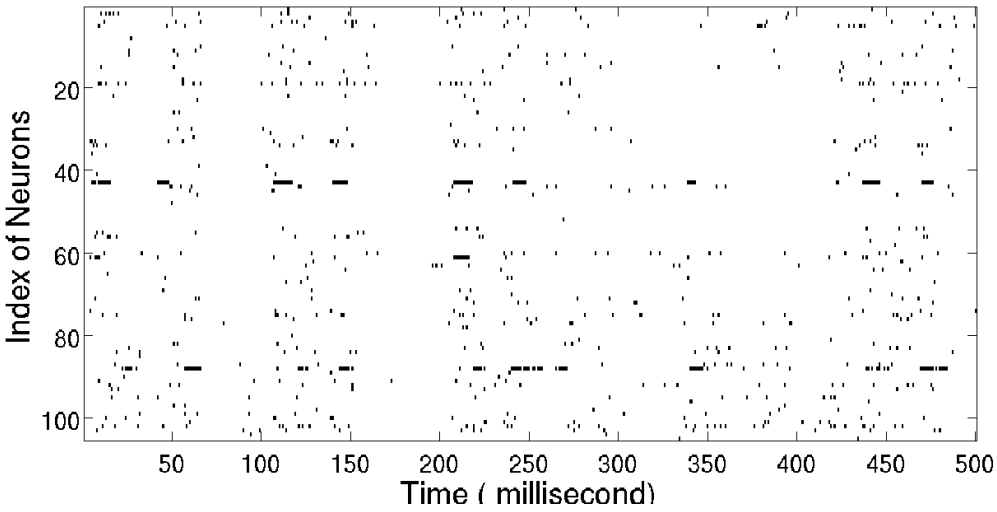


Figure 2.3: The multi-cell recording from auditory cortex of rats. There are 90 neurons and time measurements are in milliseconds.

In many of the studies described in this chapter, the procedure of discovering sequences were carried out by human experts. For example, the sequences that many papers showed the replay in the brain were discovered by manually reordering recorded neurons in a way that they appear in a sequence. The experts have tried many combination of orders to find the right order in which some neurons had a particular sequence. In the case of described statistical analysis, the sequential spike of neurons were emerged by ordering them by their latency in response. However, there might be many subsets of neurons that have sequential order in response but they have different frequency so that simple analysis can not reveal them.

The goal of our proposed pattern discovery system is to detect possible existing structures in the multi-cell recordings. This system can be used in general to detect any pattern in the data. In this thesis, for the acquired data, we ask the system to discover new patterns in the stimulus evoked activities and spontaneous activities. Then we compare the detected motifs to recognize the shared structures between these two spike trains.

Chapter 3

Heuristic Search

3.1 Introduction

This chapter presents a heuristic search for discovering spatiotemporal patterns. The algorithm is fundamentally a brute force approach for detecting recursing temporal motifs. The advantage of this method is that it is pretty simple and does not need any presumption about structure of patterns. It works very fast and does not need any pre-processing of data. The method does not go through entire search space and it takes into account the fact that temporal patterns tend to occur repetitively over time. In this chapter we demonstrate the algorithm on simple artificial data and evaluate the result visually. In the chapter 8, we will discuss the result of the method on more complicated data.

3.2 Algorithm

Let us begin by an intuitive view of what the algorithm is doing. At the core, it is searching for spikes of neurons which happen with particular orders (e.g. sequentially) or spikes that occur together with a probability higher than random coincidences. There are infinite number of combination of spikes that could occur in the data (patterns can appear in any forms). Therefore, it is impossible to systematically go through all possible combinations and search for their occurrences. This algorithm randomly selects combination of spikes that exist in the data and searches for their repetitions. If the frequency is sufficient to be a pattern, then it looks for other possible parts.

At each iteration of search, the algorithm randomly selects n spikes within the onset I time steps. Figure 3.1 illustrates the procedure for an example spike dataset. The randomly selected spikes within interval I are marked by circles ($n = 3$). Then algorithm searches for time windows in which exactly the same spikes occur (in the

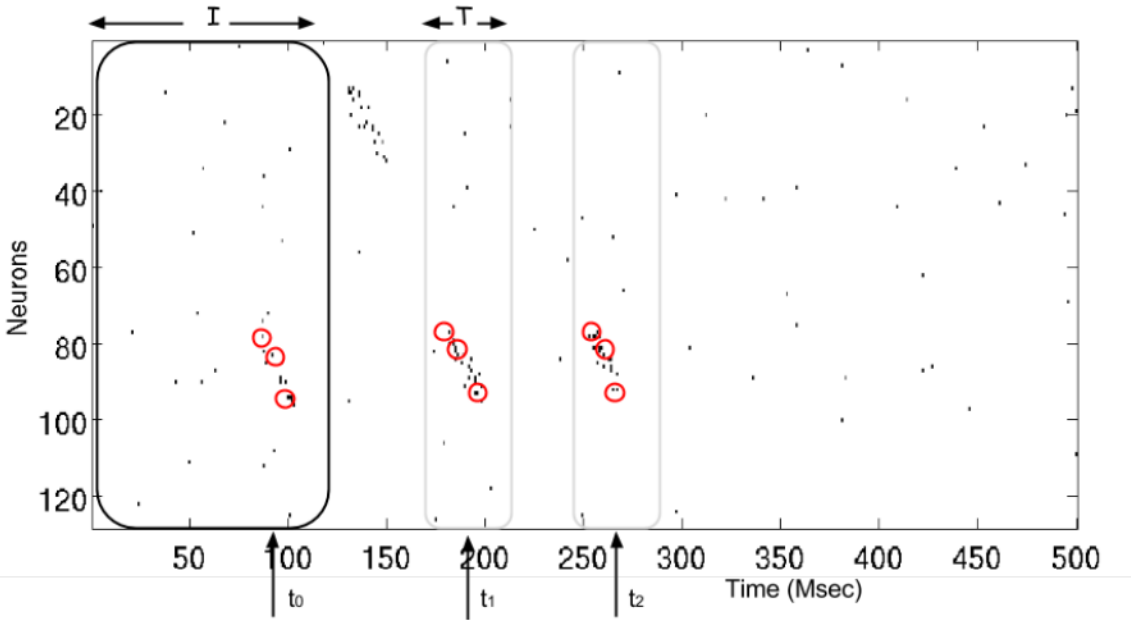


Figure 3.1: An illustration of heuristic search algorithm on an example dataset. The horizontal axis is time and vertical is index of neurons. Each dot represents a spike at a given time. The ω is the set of $\{t_0, t_1, t_2\}$.

figure, circles spot occurrences). Let us define the randomly selected combination of spikes as γ spikes. The number of occurrence of γ determines frequency of repetitions. This count gives us an intuition about how likely this combination is. If the likelihood of occurrence is more than a *threshold*, then γ is a potential distinct pattern. If this combination is a distinct pattern, then the algorithm should find entire pattern. The likelihood of other possible spikes around this combination gives us a better clue about the structure of pattern. Hence, at this stage algorithm finds all the time steps that the combination occurs. Let us name these time steps ω ($\omega = \{t_i : i = 1, .., m\}$ where t_i are time steps that γ spikes occur). In algorithm 1, the size of ω set is compared with a *threshold* which is an integer number bigger than 1. The algorithm looks at all possible spikes within T window around ω time steps (Figure 3.1). The boxes in Figure 3.1 demonstrate the T window time around occurred spikes with circle marks.

The map of probability of all possible spikes within T window describes the overall structure a pattern that is occurring over time. If the probability for some of spikes (including γ spikes) is dramatically higher than others in this map, then those spikes are part of a pattern. We define matrix of probability for all possible spikes within time window T as probability map ($Pmap$). We call this map a probability map

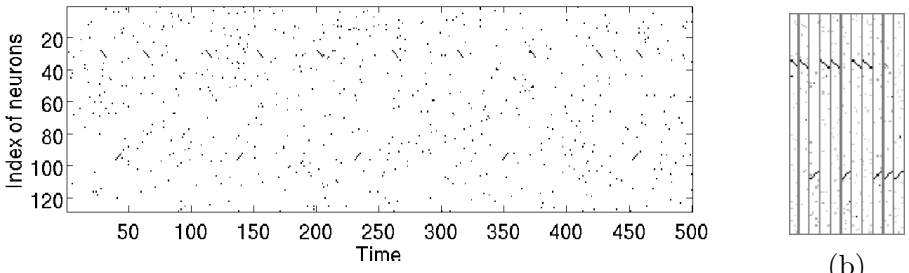
Algorithm 1: The Heuristic Search algorithm for temporal spike patterns.

while *TRUE* **do**
  $\gamma \leftarrow$ randomly select n spikes in $X(1 : N_v, I)$;
  $\omega \leftarrow$ time steps of occurrence of γ;
  if $\|\omega\| > threshold$ **then**
    $Pmap \leftarrow \frac{1}{\|\omega\|} \sum_{t \in \omega} X(1 : N_v, t - T/2 : t + T/2)$;
    if $Pmap$ is a new pattern save;

which means it contains the probability of spike of each neuron at a given time for a window of time. This map is matrix that for each neuron there is a row and for each time, there is a column. The map shows us the probability of each spike at a given time. The procedure for computing $Pmap$ is as follows. Suppose the multi-cell recording X is a binary $N_v \times N_t$ array where N_v is number of neurons and N_t total number of time measurements. Having obtained the ω set, the map is computed by $Pmap = \frac{1}{\|\omega\|} \sum_{t \in \omega} X(1 : N_v, t - T/2 : t + T/2)$ where $\|\omega\|$ is the size of set ω. The matrices $X(1 : N_v, t - T/2 : t + T/2)$ for two values of $t \in \omega$ are shown by boxes in Figure 3.1. Note that matrix X is binary and by summing $X(1 : N_v, t-T/2 : t+T/2)$ matrices, the number of occurred spikes is being counted. The $Pmap$ is a matrix of size $N_v \times T$.

The algorithm 1 shows the pseudo code of proposed heuristic search. After obtaining $Pmap$ at each iteration, algorithm compares the current map with previously saved maps to make sure that it is not saving repetitive results. The comparison is done by computing cross-correlation between the maps. An important point is that the $Pmap$ determines likelihood of spikes within a predefined window of time. If some spikes in this map have a high probability, then this indicates that their neurons have a tendency to spike very often together with a particular structure. For example in the figure 3.1 the selected neurons happened to have systematic sequential spikes. Therefore, in their corresponding $Pmap$, they have high probability of spike.

We expect this algorithm to work by randomly selecting spikes because it is exponentially unlikely that a random combination occur with high frequency in data. It is obvious that there is no guarantee to find all the patterns with this algorithm. However with limited number of neurons it is conceivable that after waiting long enough



(a) (b)

Figure 3.2: (a) the raster plot of artificial data with two sequential pattern with different frequencies. (b) the returned *Pmap*s of heuristic search on artificial data.

we could expect to locate many patterns. If one two selected spikes are not part of pattern, then the frequency of its occurrence would be very low and consequently we would not continue with algorithm.

The hyper parameters of algorithm such as thresholds and size of intervals should be determined through trial and error. In the chapter 8, the actual numbers in the implementation will be discussed. This algorithm is vulnerable against jitter of spikes in the patterns or in other words variation of patterns over time. The reason is that if the relative timing of γ spikes starts to vary over time, then the algorithm would miss many occured time steps and therefore, the obtained *Pmap* can not capture accurately the true structure of patterns.

3.3 Results on artificial data

In order to initially test the algorithm, we generated a simple binary temporal data. The data is a 128×10000 matrix. The I parameter were selected to be 5000 time steps and the T is 5. There are two sequential patterns occurring independently with different frequencies and at different location of input space (5 sequential spike in a row). Figure 3.2a shows the generated data. After 100 iteration (we count iteration if the random spikes have high enough frequency) in this data, the returned *Pmap*s show the sequential patterns that were embedded in the data. Figure 3.2b depicts the returned maps.

3.4 Summary

We presented a heuristic search for spike pattern recognition. This method finds patterns of spikes over time and selects those that occur more repetitively. The approach is very simple and do not presume any structure for the expected patterns. There are parameters that should be tweaked around to get better results by algorithm.

Chapter 4

Convolutional Restricted Boltzmann Machine

4.1 Introduction

Discovering spike patterns from multi-cell recording dataset is an extremely hard task since there is no information about the structure of patterns. Therefore, it is vary hard to design any filter to extract patterns from an input data since structure of patterns are not known. Learning pattern detector filters from data is the main task of unsupervised feature learning algorithms. The main objective of these approaches is to learn filters that transform input data to a more abstract representations. Virtually any type of data has multiple levels of abstraction. For example, the visual world can be described in many levels: pixel intensities, edges, object parts, objects and beyond. The prospect of learning models that simultaneously represent these levels of abstraction has recently generated much interest in the field of deep learning algorithms [28, 6, 53, 8, 36]. Ideally, a feature learning method learns multiple pattern detectors which transform input to a more abstract representation. The idea is to obtain an invariant representation from a highly variable space(e.g. pixel intensities). Building a hierarchy upon such feature learning model yields the desired multiple levels of abstraction. For our particular application, we are not interested in learning deep hierarchy from spiking datasets. Rather we are interested in training such learning models to obtain pattern detectors. In other words, our goal is to obtain spike pattern detectors in a purely unsupervised manner.

In this chapter, we build upon belief networks because we are interested in learning a generative model of our dataset which can be trained in a purely unsupervised manner. The deep model of belief networks has been successfully used for unsupervised feature learning and has been incorporated into many machine learning tasks. The deep belief network (DBN) was introduced by Hinton et al. [27] and is based on restricted Boltzmann machine(RBM). RBMs can be stacked on top of each other to

form a hierarchical representation. This deep model has proven to be computationally efficient (being able to apply in practical applications) for large datasets of high dimensional data such as images, human motion (e.g. [66, 28]).

The DBNs were successful in controlled domains, however scaling up to realistic input sizes(e.g. 200×200 pixel size images) is a challenging task. Therefore, a convolutional structure has been introduced into these models to mitigate lack of success in larger domains [39]. The convolutional structure is a shared parameter scheme which enables hidden nodes of RBM to share their parameters. This technique enjoys the benefits of having less parameters and yet being able to capture patterns that replicate at different locations of input space. The patterns in the input space can appear at any arbitrary location: object parts in images appear at any place. The Convolutional RBM (CRBM) addresses this issue by learning feature detectors that are shared among all locations of input. In this fashion a feature detector which captures useful information in one part of an input can pick up the same information elsewhere. Thus, the model can represent large input spaces using only a small number of feature detectors. This factor is essential for our multi-cell recording pattern recognition since the patterns in our dataset, can appear at any location or can occur at any time. That's why we need a model that learn filters that can detect pattern at any location.

This chapter introduces the convolutional restricted Boltzmann machine. We discuss the necessary background about the model, training algorithm and the main monitoring procedures during training. In addition, we present our proposed techniques for preventing the model from over-fitting which is crucial for our application. We evaluate our techniques on the hand written digits (MNIST) as a standard dataset. In addition, we discuss the crucial role of sparsity regularizations. These techniques are essential when we train our model on the multi-cell recording and play a vital role for learning distinct pattern detectors.

4.2 Restricted Boltzmann Machine

We begin by describing the restricted Boltzmann machine (RBM), then we present the convolutional version in the following section. The RBM is a generative bipartite stochastic network (Markov random field) with two layers, where a set of latent

(or hidden) binary random variables are densely connected to a set of input (or visible) variables. The visible variables are observed and hidden variables are feature detectors of the network. In general this model could be designed for binary observed random variables or real-valued visible variables. In this thesis we work with binary datasets therefore we define only the binary model. Figure 4.1 shows the connections of visible and hidden variables of RBM and the corresponding weights between their connections. The general form of these random fields are called Boltzmann machines [29] where there are connections among visible and hidden units. The restriction of RBMs allows for more efficient training algorithm. The joint distribution is described by an energy function (as in standard log-linear models), which is simply a sum of products between the hidden, visible units, and the weights between the two layers. The joint probability distribution over this network is defined as follows:

$$P(v,h) = \frac{1}{Z} exp(-E(v,h)) \tag{4.1}$$

Where h and v are hidden and visible variables represented as vectors of size n and m. The Z is the partition function which is simply sum over all possible configurations and it is defined as:

$$Z = \sum_{v,h} exp(-E(v,h)) \tag{4.2}$$

Energy function for binary visible variables is defined as:

$$E(v,h) = -\sum_{i,j} v_i W_{ij} h_j - \sum_j b_j h_j - \sum_i a_i v_i \tag{4.3}$$

Where v_i , h_j are binary states of visible unit i and hidden unit j, the b_j,a_i are their corresponding bias and $W_{i,j}$ is the weight matrix. Training this model is basically maximizing the log-likelihood of observed variables. In other words, the parameters of the network (weights and bias) are updated so that energy of observed instances are minimized (maximize the probability). On the other hand, training procedure maximizes the energy of all other unobserved possible instances and consequently they would have lower probability. In order to obtain the log-likelihood function, we need the probability of the visible layer ($p(v)$). The $p(v)$ is determined by marginalizing out h:

$$p(v) = \frac{1}{Z} \sum_h exp(-E(v,h)) \tag{4.4}$$

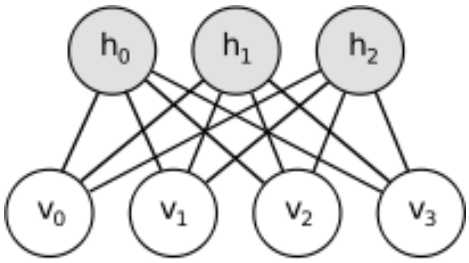


Figure 4.1: A restricted Boltzmann machine with 3 hidden nodes and 4 visible nodes. There is a weight w_{ij} for each connect between hidden node i and visible node j.

Therefore the log-likelihood function J is defined over the training set as the following:

$$J = \sum_{i \in D} \log(P(v^{(i)})) = \sum_{i \in D} \log\big(\sum_{h} exp(-E(v^{(i)}, h)\big) \tag{4.5}$$

Where D is the training dataset. The parameters of network namely W, a, b should be obtained such that function J is maximized. In other words:

$$argmax_{W,a,b} J = argmax_{W,a,b} \sum_{i \in D} \log(P(v^{(i)})) \tag{4.6}$$

This process is typically based on stochastic gradient ascent algorithm which needs derivative of objective function. The derivative with respect to the parameter W is derived as follows:

$$\frac{\partial J}{\partial W_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{4.7}$$

where the angle brackets are used to denote expectations under the distribution specified by the subscript that follows. The data distribution is $p(h|v)$ where v is clamped to training examples(e.g. images) and the model distribution is $p(v, h)$.

Computing the exact log-likelihood gradient is not practical since expectation terms are intractable due to intractability of partition function Z. There is an approximation algorithm called Contrastive Divergence (CD) [28] which has been proved to work in practical applications. The CD algorithm computes unbiased samples of expectations terms in a procedure called Gibbs sampling. Gibbs sampling requires conditional independence of variables. Because of special structure of RBM , the hidden and visible variables are conditionally independent. Given training data v, we can obtain an unbiased sample of h using the following conditional distribution:

$$p(h_j = 1|v) = \sigma\big(b_j + \sum_i v_i W_{ij}\big) \tag{4.8}$$

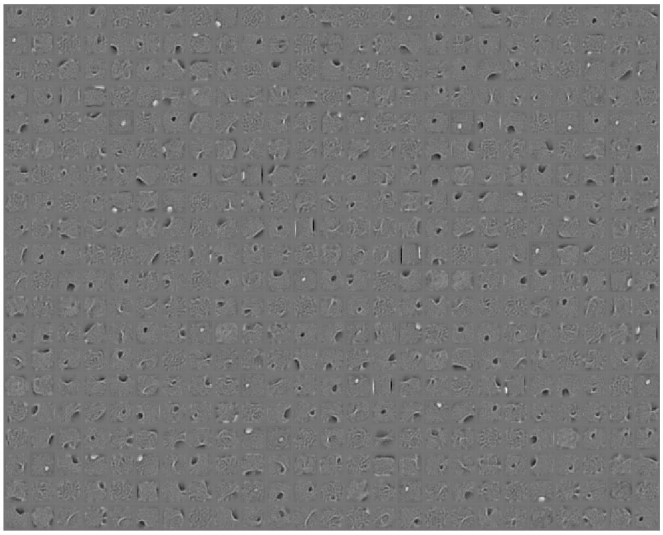


Figure 4.2: The 500 filters learned by RBM model on MNIST.

Where $\sigma(x)$ is the logistic function $1/\big(1 + exp(-x)\big)$. Given sampled hidden states h, a "reconstruction" of visible variables can be obtained using the following conditional distribution:

$$p(v_i = 1|h) = \sigma\big(a_i + \sum_j h_j W_{ij}\big) \tag{4.9}$$

Algorithm 2 explains the procedure of training RBM. The algorithm begins by sampling hidden values given the training samples. Then it needs to estimate the expectation term $\langle v^T h \rangle_{data}$ which means multiplication of values of hidden nodes given the value of visible nodes. Thus algorithm computes the probability of h given data points v in the mini-batch. Then it samples h based on the obtained probability. The procedure continues by estimating the expectation term $\langle v^T h \rangle_{model}$ which requires sampling values of both h given v and vice versa. Estimating this distribution can be done by continuing sampling procedure. The algorithm has a hyper parameter k which determines how many samples the algorithm draws from model distribution. However, in practice just one sampling step is shown to be practical [28].

4.2.1 Training RBM on hand written digit dataset (MNIST)

We trained an RBM model on MNIST dataset. The dataset contains 60000 images of hand written digits where the size is 28×28 and images are binary. We trained

Algorithm 2: Contrastive Divergence Algorithm(CDk) for training RBM.

while *NOT Converged* **do**
  for *All Mini-batches of training set D* **do**
    Sampling $h \sim p(h_j = 1|v) = \sigma\big(b_j + \sum_i v_i W_{ij}\big)$;
    Sampling $v' \sim p(v_i = 1|h) = \sigma\big(a_i + \sum_j h_j W_{ij}\big)$;
    Sampling $h' \sim p(h_j = 1|v') = \sigma\big(b_j + \sum_i v'_i W_{ij}\big)$;
    for $CDk = 2$ *to* k **do**
      Sampling $v' \sim p(v_i = 1|h') = \sigma\big(a_i + \sum_j h_j W_{ij}\big)$;
      Sampling $h' \sim p(h'_j = 1|v') = \sigma\big(b_j + \sum_i v'_i W_{ij}\big)$;
    Update all parameters W, a, b using (ϵ is the learning rate.):
    $W_{i,j} \leftarrow W_{i,j} + \epsilon\big(\langle v_i h_j\rangle - \langle v'_i h'_j\rangle\big)$
    $a_i \leftarrow a_i + \epsilon\big(\langle v_i\rangle - \langle v'_i\rangle\big)$
    $b_j \leftarrow b_j + \epsilon\big(\langle h_j\rangle - \langle h'_j\rangle\big)$;

an RBM with 500 hidden nodes. The k parameter of contrastive divergence was 25 and we trained the model with a sparsity regularization explained in the section 4.3. Figure 4.2 shows the weights after training. Each hidden node has 784 weights connected to input layer. We transformed this vector to a 28×28 matrix to show it in the figure. As it is shown in Fig. 4.2 the network has learned edge detectors similar. Training this unsupervised learning algorithm on image dataset results in filters that can capture low-level patterns(edges) in images [4].

4.2.2 Sharing Parameters

The main problem of restricted Boltzmann machine is that it is not scalable to large input spaces (e.g. big image sizes). The reason is that the pattern detectors of this network (hidden nodes) are connected to the entire input space. Thus, it has huge number of parameters to be learned. In addition, if patterns with the same structure are at different locations of input space, RBM learns different pattern detectors for each location. Figure 4.2 shows the filters of RBM on MNIST. It demonstrates the fact that there many similar edge detectors (with the same orientation) but at different locations. A share parameter scheme should be introduced to address this

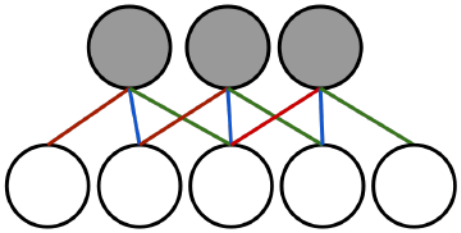


Figure 4.3: A shared parameter scheme for RBM. The hidden layer or feature map layer has hidden nodes that are connected to a few visible nodes and all share the same weights (indicated with the same color).

issue. In this scheme, pattern detectors (hidden nodes) share their weights among input locations. Figure 4.3 illustrates a shared parameter structure for RBM. The figure shows that hidden nodes are connected to few visible nodes and all of them share the same weights (3 weights are shared and depicted with 3 different colors). With the help of this scheme, the feature detection ability of model would be location invariant. Moreover, the model would have much fewer parameters. This is very important when it comes to training large networks. In the following section we present the shared parameter RBM known as Convolutional RBM.

4.3 Convolutional Restricted Boltzmann Machine

The convolutional restricted Boltzmann machine (CRBM) were proposed by Lee et al. [39]. We begin explaining the model by introducing the necessary notions.

4.3.1 Notations

In this thesis we use the following notations. The size of input space is $N_v \times N_v$, even though it is not a requirement for having squared input or even two dimensional input. All of the variables are binary-valued, while noting that it is possible to generalize model to real-valued visible units. We use $*$ to denote convolution [1], and $\bullet$ to denote element-wise product followed by summation, i.e. $A \bullet B = trA^T B$. Also the tilde on top of an array ($\widetilde{A}$) is the operation of flipping the array horizontally and vertically.

[1]The convolution of an $m \times m$ array with an $n \times n$ array may result in an $m-n+1 \times m-n+1$ array or an $m+n-1 \times m+n-1$ array. We let the reader to determine the case by context

4.3.2 Algorithm

The CRBM is the shared parameter RBM. In this model we have two layers: Visible layer V and hidden layer H (corresponding to the figure 4.4). The input layer is $N_v \times N_v$ of binary random variable. The hidden layer H has K groups. Each group is $N_h \times N_h$ binary random variables. Each hidden group has a weight matrix with size $N_w \times N_w$ where $(N_w = N_v - N_h + 1)$. It should be noted that the weights are shared among hidden units within each group. In addition, each hidden group has a bias b_k and all visible units share a single bias c. We define the joint probability distribution as:

$$P(v,h) = \frac{1}{Z} exp(-E(v,h)) \tag{4.10}$$

$$E(v,h) = -\sum_{k=1}^{K}\sum_{i,j=1}^{N_h}\sum_{r,s=1}^{N_w} h_{ij}^{k} W_{r,s}^{k} v_{i+r-1,j+s-1} - \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^{k} - c\sum_{i,j} v_{i,j} \tag{4.11}$$

using the introduced notations, we can re-write as:

$$E(v,h) = -\sum_{k=1}^{K} h^k \bullet (\widetilde{W}^k * v) - \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^{k} - c\sum_{i,j} v_{i,j} \tag{4.12}$$

Similar to the RBM, we need to maximize the log-likelihood of observed samples to minimize the energy of observed instances while maximizing the energy of all possible unobserved instances. The gradient of the log-likelihood with respect to weights has the following form:

$$\frac{\partial J}{\partial W^k} = \langle v * h^k \rangle_{data} - \langle v * h^k \rangle_{model} \tag{4.13}$$

To estimate the expectation terms, we use the contrastive divergence algorithm [27]. The CRBM version of algorithm is presented in Algorithm 3. The procedure performs Gibbs sampling which requires the conditional distributions for hidden and visible variables:

$$P(h_{i,j}^{k} = 1|v) = \sigma((\widetilde{W}^k * v)_{i,j} + b_k) \tag{4.14}$$

$$P(v_{i,j} = 1|h) = \sigma((\sum_{k} W^k * h^k)_{i,j} + c) \tag{4.15}$$

where the σ is the logistic function. Similar to RBM, it is needed to sample hidden nodes given training examples to estimate data expectation and to draw samples of

Algorithm 3: The Contrastive Divergence Algorithm for Convolutional RBM.

while *NOT Converged* **do**
  for *All v in all Mini-batches* **do**
    for all k sample $h^k \sim P(h^k_{i,j} = 1|v) = \sigma((\widetilde{W}^k * v)_{i,j} + b_k)$;
    for all hidden group k sample and sum $v' \sim$
    $P(v_{i,j} = 1|h) = \sigma((\sum_k W^k * h^k)_{i,j} + c)$;
    for all hidden group k sample $h'^k \sim P(h^k_{i,j} = 1|v) = \sigma((\widetilde{W}^k * v)_{i,j} + b_k)$;
    for $CDk = 2$ *to* k' **do**
      for all hidden group k sample and sum $v' \sim$
      $P(v_{i,j} = 1|h) = \sigma((\sum_k W^k * h^k)_{i,j} + c)$;
      for all hidden group k sample $h'^k \sim$
      $P(h^k_{i,j} = 1|v) = \sigma((\widetilde{W}^k * v)_{i,j} + b_k)$;
    Update all parameters W, a, b using :
    for *all* k **do**
      $W^k_{i,j} \leftarrow W^k_{i,j} + \epsilon\big(\langle v_i h^k_j\rangle - \langle v'_i h'^k_j\rangle\big)$
      $a_i \leftarrow a_i + \epsilon\big(\langle v_i\rangle - \langle v'_i\rangle\big)$
      $b^k_j \leftarrow b^k_j + \epsilon\big(\langle h^k_j\rangle - \langle h'^k_j\rangle\big)$;

hidden and visible given each other (h' and v') to estimate the model expectation term. The k' parameter of algorithm determines the number of sampling steps that we should take in order to estimate the model expectation. In other words, in order to get better estimation of model distribution, algorithm could keep drawing samples v' and h' (going up and downs in the network).

4.4 Monitoring learning procedure

In order to increase the performance of learning procedure, hyper parameters have been introduced for update rule of the CD algorithm. Moreover, monitoring procedures have been proposed for having insight into the learning process. Specifically, we propose a technique to prevent the model from learning trivial filters caused by over-fitting. In the following subsections we will discuss these techniques in details.

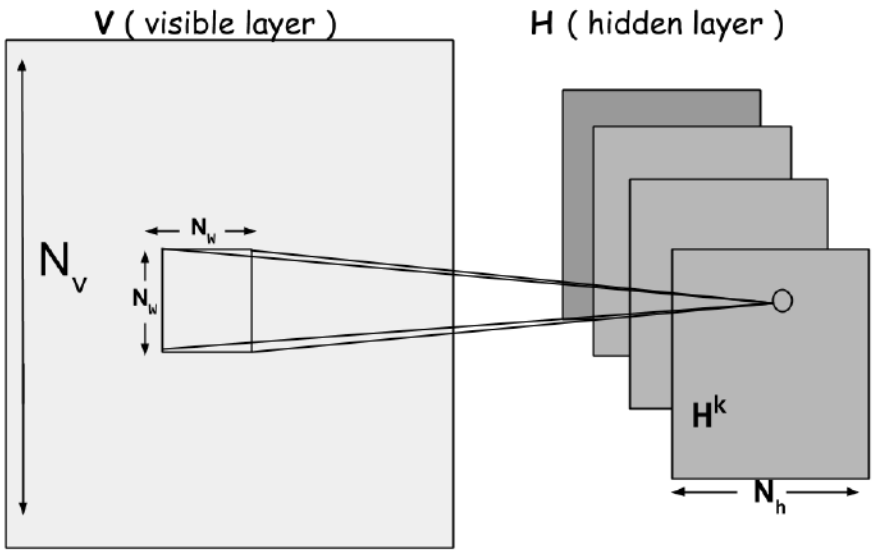


Figure 4.4: The convolutional restricted Boltzmann machine architecture.

4.4.1 Momentum

Momentum is a method for gradient descent optimization to avoid local minimums. This method improve procedure to escape the complex error surface of log-likelihood function of RBM and CRBM model. Given the gradient and momentum term m, the update term of parameters at time t would have the following form:

$$\Delta\omega(t) = m\Delta\omega(t-1) + \frac{\partial J}{\partial W} \tag{4.16}$$

Where ω is derivative or the update term for our parameters and $0 < m < 1$ is a hyper parameter which should be determined through trial and error. Every step we add fraction m of previous update terms to current one. During the training process, this fraction should become close to 1 while learning rate should reduce. If we combine a huge learning rate and momentum the update steps would overshoot the global minimum. The reason for having a momentum term is that gradient at most points of the error surface do not direct to the global minimum and successive gradients oscillate from one side to another. The momentum term helps the learning process by damping the oscillations.

4.4.2 Weight Decay

The wight decay is a regularization method widely being used for artificial neural networks to regularize optimization process and help generalization performance. In this method optimization objective is penalized for having large parameters by adding the sum of squared of parameter to the objective function. In this way, by minimizing the objective the parameter of network itself is constrained to be minimized as well. Therefore, large values of weight matrix is penalized to decrease when there is no gradient in that direction. In the case of CRBM we use this technique while minimizing the negative log-likelihood of data. Thus, given the gradient term, the update term $\Delta\omega$ (the term that we use to update our parameter W) is modified to be:

$$\Delta\omega = \alpha\frac{\partial J}{\partial W} + \alpha\lambda W \tag{4.17}$$

Where α is the learning rate and λ controls the effect of weight decay regularization. We used this technique to help the CRBM learn the strongest signal in the visible layer and avoid noise in the data.

4.4.3 Sparsity Regularization

The convolutional RBM model is over-complete in that the size of the representation is much larger than the size of the input. In other words, since the size of input and hidden groups are roughly the same (filters are typically very small), this model is over-complete by a factor of K (number of hidden groups). In general, this issue runs the risk of learning trivial solutions, for instance feature detectors of images could be representing single pixels [38]. Moreover, they might learn multiple filters with the same captured pattern. To address these issues, it is needed to force the representation to be "sparse", meaning only a tiny fraction of the hidden units should be active in relation to a given stimulus.

In this thesis, we use the sparsity regularization used by lee et al. [38]. This approach, regularizes the objective function (log-likelihood) to encourage the hidden groups to have their mean activity be close to a small constant. Specifically, it has been suggested by [38] that the following simple update rule (followed by each

contrastive divergence update) works well in practice:

$$b_k \propto \rho - \frac{1}{N_h^2} \sum_{i,j} p(h_{i,j}^k = 1|v) \tag{4.18}$$

where ρ is a target sparsity, and each example v is treated as a mini-batch. We select the learning rate such that we could keep the average activity of each hidden group for entire training set to be constant while allowing individual variations.

4.4.4 Free Energy

Preventing machine learning methods from over-fitting is one of the main challenges of subject. In order to prevent learners from simply memorizing the training examples, monitoring the generalization error of the model while the learning process is progressing is essential. The most typical monitoring statistic for RBMs is the reconstruction error of training and validation set. The reconstruction of an instance of visible variables can be done by going up and down in the network. In other words, first samples of hidden states are drawn using equation 4.14 (going up) then the samples of visible are drawn given the hidden samples using equation 4.15 (going down). The mean squared error of the reconstruction and the true sample is an indication of progress of learning. Typically at the beginning there should be fluctuations for error but generally it goes down as the learning progress. The average reconstruction error of network on a validation set roughly indicates the progress of generalization. However, this error is not a perfect measure since it is not the objective function that model is trying to minimize. Thus, it is necessary to have good insight during the training. We measure the free energy of Convolutional RBM as an indicator of learning progress. The negative log of unnormalized probability of data under the model distribution is called the free energy. Since the measure is unnormalized, it can not directly measure the likelihood of data. However the ratio of average free energy of training and validation examples is a good indicator that whether model has a good generalization because the normalizing constant is canceled out. The subtraction of free energy of training and validation set should be near 1 and dramatic divergence of them indicates over-fitting [25]. The general form of free-energy of data point x

for CRBM is defined as:

$$F(x) = -\log \sum_{h} e^{-E(x,h)} \tag{4.19}$$

By substituting the energy function of convolutional RBM, we can derive the following equation:

$$F(x) = -c \sum_{i,j} x_{i,j} - \sum_{k=1}^{K} \log(1 + exp(\sum_{i,j} (\widetilde{W^k} * x)_{i,j}) + b_k) \tag{4.20}$$

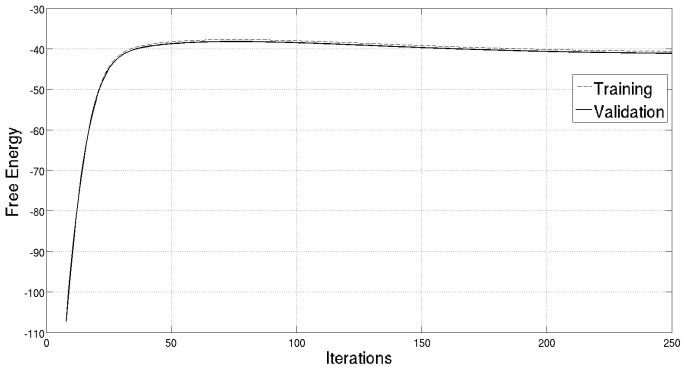
The proof is given in the appendix A. During the training process we compare free energy of the two sets and when the difference between these two values change substantially, we stop the training.

4.4.5 Training CRBM on MNIST

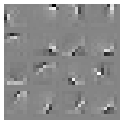
We tested our implementation on MNIST dataset. In order to train a better model, we whitened the images of MNIST. The whitening technique of [35] were used which is a standard method in deep learning field. We trained a convolutional RBM with 16 filters of 8×8. The number of contrastive divergence was 1 and we used the explained sparsity regularization. The momentum term was 0.9 and weight decay was 0.01. Figure 4.5a depicts the free energy of model for validation and training set. We stop the training when the two numbers started to diverge. Figure 4.5b shows the learned filters of network. It shows that model were able to learn Gabor edge filters with different orientations.

4.5 Summary

In this chapter, we presented the convolutional restricted Boltzmann machine as an unsupervised learning method. The training algorithm and monitoring procedures were discussed. Moreover, the implemented sparsity regularizations and over-fitting prevention techniques were explained. Finally, we discussed the results of training this model on MNIST dataset.



(a)



(b)

Figure 4.5: (a) The free energy of convolutional RBM during training on MNIST. (b) The filters learned by CRBM on MNIST.

Chapter 5

Convolutional Auto-encoders

5.1 Introduction

We discussed in the previous chapter that the main purpose of unsupervised feature learning is to extract useful features from unlabeled data. These algorithms aim to detect and remove data redundancies by preserving essence of data and learning to transform it into robust and discriminative representations. Feature learning methods have been widely used in scientific and industrial applications [23, 2, 46, 54]. Most unsupervised models are based on encoder-decoder paradigm [46]. In this paradigm, first the input is transformed into feature space (encoder) which is typically low-dimensional and then expanded into original space (decoder). Example models of such paradigm are restricted Boltzmann machines [27], auto-encoders [54] and energy based models [37]. There are connections between the auto-encoder and RBMs that auto-encoder training approximates RBM training by Contrastive Divergence [3]. Because training an auto-encoder seems easier than training an RBM, they have been used as building blocks to train deep networks.

In this chapter we build upon convolutional denoising auto-encoders (CDAE) as our feature learning algorithm. We are interested in auto-encoders since training this model is easier than RBM. Moreover, RBM need probabilistic inference and approximations for learning while auto-encoders do not need any approximation. We use the convolutional version of auto-encoder as we did for RBM, to address the issue of variability of location of patterns in input space. Also to reduce the number of learn-able parameters in our model. We discussed in the previous chapter that shared parameter scheme help model to learn translation invariant representations. These benefits are important for multi-cell recording dataset with huge input spaces and repetition of patterns.

The issue of learning repetitive pattern detectors or filters with trivial solutions are very important challenges for unsupervised feature learners in general. Sparsity

regularizations are typically the remedy to address these issue. These regularizations encourages encoder layer to transform input to a representation with sparse active variables. Consequently, such measures encourage model to learn distinct filters that transform input to more robust and discriminative representation. The role of these measures are vital for the multi-cell recording dataset since we need clear and distinct pattern detectors.

This chapter presents the convolutional auto-encoder model. We use a winner-take all algorithm for the sparsity regularization which helps our learned filters be clear. Moreover, we explain the denoising auto-encoder. This model adds a simple modification to the original model which improves its robustness against noise in the data. The robustness against noise is a very important factor for multi-cell recordings since these data sets are extremely noisy. We test our implementation on MNIST as a baseline dataset. The results demonstrate the effects of regularizations and denoising scheme.

5.2 The Basic Auto-encoder

We begin by describing the auto-encoder and continue by the convolutional version in the following section. The auto-encoder is basically a neural network which reconstructs its input. In other words, the output of neural network is the input itself. The model encodes input X to a new representation H and it reconstructs from this representation. Figure 5.1 shows the neural network of auto-encoder. The network can have multiple hidden layers that transform its input to new representations. We continue by assuming there is only one layer. The hidden layer is obtained by

$$H = f(WX + b) \tag{5.1}$$

where f is could be a linear or nonlinear function and W and b are weights and bias. The reconstruction is given by

$$Z = g(W^T H + a) \tag{5.2}$$

where g could be the same as f. The objective function is typically the mean squared error of input and reconstruction. To train the model, the objective function is minimized by stochastic gradient descent. The relationship of this model and principal

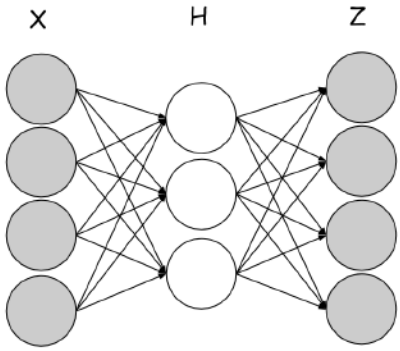


Figure 5.1: The Auto-encoder neural network.

component analysis is very interesting. If there is one linear hidden layer with k hidden nodes and mean squared criterion is used then the model projects input in the span of k principal components of data [3]. Bengio [3] uses the formulation that generalizes the mean squared error criterion to the minimization of the negative log-likelihood of the reconstruction, given the encoding representation H:

$$RE = -\log(p(X|H)) \tag{5.3}$$

If the input data is binary (which is the case for our multi-cell recording), then the criterion becomes the cross-entropy between input and the reconstruction:

$$-\log(p(X|H)) = -\sum_i X_i \log(g_i(H)) + (1 - X_i)\log(1 - g_i(H)) \tag{5.4}$$

5.3 Convolutional Auto-encoder

The convolutional version uses the same basic paradigm of auto-encoder. However, the nodes in the feature space share their weights among the input locations. In other words, instead of having one group of feature detectors, there are multiple groups of detectors where nodes inside each group share the same filter. The feature nodes inside each group have different receptive field. To put it in formal notation, there are K filters which project input x to kth representation h^k:

$$h^k = f(W^k * x + b^k) \tag{5.5}$$

where $*$ denotes convolution and f could be a linear or non-linear function (typically sigmoid). W^k is the weight or filter and b^k is the bias term. The hidden representation is projected back to original space with the following:

$$z = g(\sum_{k=1}^{K} \widetilde{W}^k * h^k + c) \tag{5.6}$$

Where the tilde operation is to denote flipping the array horizontal and vertically. The criterion for training the model is typically the squared error but we used the cross-entropy since the data is binary.

5.4 Sparsity Regularization

One of the key challenges of these models is the issue of learning sparse representation. This means that model should learn a representation where tiny number of feature detectors are active. Encouraging the model to learn such representation forces the filters to be distinct and informative. In this thesis, we regularize training algorithm by a winner take all method developed in [45].

In this method, we explicitly inhibit all the weak response from input. In other words, after transforming the input to feature map, within each feature map the node with strongest value is selected as the response of input and all other nodes in that representation are set to zero. To put it differently, after computing each h^k from input, the h^k_{ij} with maximum value in this map is kept and other values are set to zero. The method during the training, directly forces the representation to be sparse. The reconstruction is done with the new sparse representation. In this fashion, only variables with maximum value in h^k would have gradient to update the filter. Hence each filter learn to reconstruct strongest signal in data and consequently algorithm would learn distinct filters. To put it differently, after the selection of strongest hidden nodes, the reconstruction error is only back-propagated through these hidden units. Hence the regularizer prevents the model from the use of an overly large number of hidden units within each feature map, when reconstructing the input. Therefore, we encourage explicitly to learn sparse representation and ideally learning sharper filters.

5.5 Denoising Auto-encoder

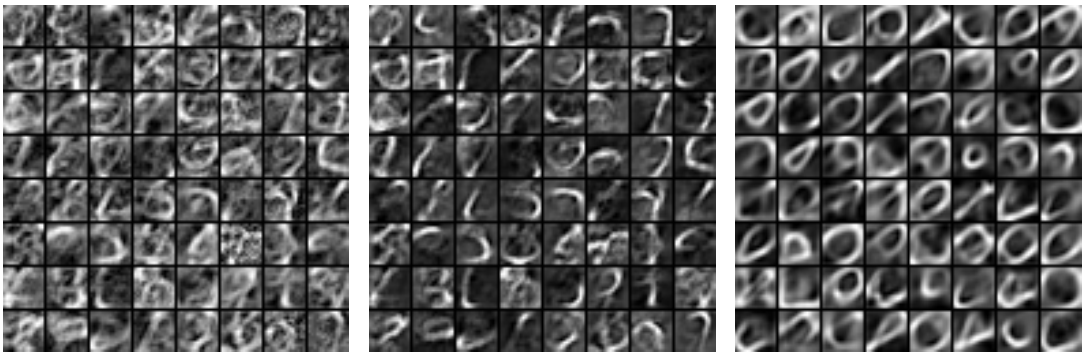
Denosing auto-encoder [69] is a stochastic version of the auto-encoder where the input is stochastically corrupted, but the uncorrupted input is still used as target for the reconstruction. This model is trying to do two main jobs: to encode input with keeping relevant information and also to undo the effect of corruption process. The model do the latter by capturing the statistical dependencies in the input dimensions [3]. In [70] the corruption process set half of the input dimensions to zero and the network tries to predict missing values given received information which resulted in learning very robust representation. To put it formally, the training criterion of denosing auto-encoder is the following:

$$-\log(p(X|H(\widetilde{X}))) \tag{5.7}$$

Where $H(\widetilde{X})$ is the hidden representation of stochastically corrupted data point $\widetilde{X}$. The main argument for having such scheme is that a good representation learning algorithm should learn representations that have robustness to partial destruction of the input , i.e., partially destroyed inputs should yield almost the same representations [69]. Learning such representation is a perfect attribute for the model that is going to be trained on multi-cell recording. We incorporate this technique for our convolutional auto-encoder to learn better filters and better model of our dataset. The multi-cell recording dataset is extremely noisy and consequently spike patterns are extremely noisy. Learning a model which learns clear and distinct spike pattern detectors is possible by a denoising scheme.

5.6 Training the Models on MNIST

We trained different models on the MNIST handwritten digit in order to demonstrate the effects of the sparsity regularization and denoising scheme. We trained convolutional auto-encoder with 64, 16×16 filters on MNIST. Figure 5.2a shows the learned filters without any regularization. The picture vividly shows that filters are noisy and smeared. We then add the winner take all sparsity constraint to the model and train it again. Figure 5.2b depicts the filters that model with sparsity regularization learns. The figure shows that filters are more distinct than previous model. The effect of denoising scheme is shown by figure 5.2c.



(a) (b) (c)

Figure 5.2: The filters learned by convolutional auto-encoder on MNIST: (a) with no regularization (b) with sparsity regularization (c) denoising CAE with sparsity.

5.7 Summary

We presented convolutional auto-encoder as a feature learning algorithm. The winner take all sparsity method were explained as the regularization for learning distinct filters. Moreover, denoising scheme for obtaining robustness against noise were introduced to the convolutional AE. We tested our implementations on MNIST dataset and visualized the learned filters.

Chapter 6

The Quantitative Analysis

6.1 Introduction

Measuring the performance of an unsupervised feature learning algorithm is typically based on improvement of these algorithms on classification error. For example stacked auto-encoders and deep belief networks have been used as pre-training step of a deep neural network and has improved the classification error significantly (e.g. [28, 17]). In other words, the unsupervised feature learning has helped to improve a supervised learning task. More improvement of classification error indicates a better unsupervised learning algorithm and therefore a better learned model. We can not use such indicator for our learning algorithms, since our task is not classification. In addition, what typically is reported as an indicator of good learned model for a generative stochastic feature learning like RBM is the generated samples of model. By starting from a random initial state of visible units and Gibbs chain sampling of model's variables (going up and down in the network), many examples would be generated after long enough updates. These samples should look like the training examples. For instance, by training on MNIST, the generated samples should look like digits. This method also can not help us for our task since generating spike looking data would not give us any insight to what the model has learned. However, an important indicator for generative models is the likelihood of data under model distribution.

The log-likelihood of unseen data under the trained model distribution, gives a rich insight to what model has learned. In other words, by comparing the likelihood of training data and validation set under the model distribution, it is possible to assess how good is the learned model. It gives us confidence that the unsupervised learning model has a good generalization over the data generating distribution. An unfortunate limitation of restricted Boltzmann machine is that the probability of data under the model is not known due to computationally intractable normalizing

constant, known as the partition function. Therefore, a good estimation of partition function would provide the ability to estimate the likelihood of data under RBM distribution. A good estimate of the partition function would help for controlling model complexity, which plays an important role for making RBMs generalize well.

This chapter presents an estimation algorithm for partition function of convolutional restricted Boltzmann machine based on a Monte Carlo method known as annealed importance sampling(AIS). Ruslan and Murray [63] demonstrated that by using AIS and taking advantage of bipartite structure of RBM, in principle one can obtain an efficient estimate of the partition function and further could approximate log-probability of data under the model distribution. We adapted the formulation of their work for convolutional RBM and compare our results on MNIST with reported estimates in [63].

Moreover, denoising auto-encoders are competitive alternative for RBMs for unsupervised learning and have been used for pre-training of deep networks. The fact that training this model is easier than RBM, justifies further research to develop better algorithms for analyzing training of this model. Vincent et al. [68] show that DAE training criterion is equivalent to training an energy-based model. This yields several useful insights to training procedure. It defines a proper probabilistic model for the DAE model, which makes it in principle possible to compute free-energy. The free energy can give us insight to prevent model from over-fitting. In this chapter, we adapt the formulation of [68] and generalize it to convolutional DAE. We evaluate our work on the MNIST dataset.

6.2 Annealed Importance Sampling

The goal in this section is to introduce an algorithm to estimate the ratio of partition function of two distributions. Suppose that there are two distributions defined over space X with probability density function $p_A = p_A^*(x)/Z_A$ and $p_B = p_B^*(x)/Z_B$. The p_A is an intractable distribution and p_B is its approximation. To estimate the ratio of their normalizing constant, we could do simple importance sampling. Suppose that $p_A(x) \neq 0$ whenever $p_B(x) \neq 0$, also assume that we can draw independent samples from p_A, the unbiased estimate of ratio of partition functions can be obtained using

the following Monte Carlo approximation:

$$\frac{Z_B}{Z_A} \approx \frac{1}{M}\sum_{i=1}^{M}\frac{p_B^*(x^{(i)})}{p_A^*(x^{(i)})} \equiv \frac{1}{M}\sum_{i=1}^{M}\omega^{(i)} = \hat{r}_{IS} \tag{6.1}$$

Where $x^{(i)} \sim p_A$. If p_A is not a good approximation of p_B or in other words, two distributions are not close enough, the estimator $\hat{r}_{IS}$ will be very poor especially if the space X is very high dimensional. Therefore, we should incorporate a better method for our RBM with high dimensional space. The Annealed Importance Sampling (AIS) has been widely used for estimating partition functions of energy-based models (e.g. [50, 65]). The AIS method is the following. There are conditions which are fully described in [63] for this method to work. However we mention three of them: (1) we should be able to define a sequence of intermediate distributions $p_0, p_1, ...p_K$, with $p_0 = p_A$ and $p_K = p_B$. (2) we should be able to define a Markov Chain transition operator $T_k(x; x')$ to draw sample x' given x. (3) we should be able to compute unnormalized probability of intermediate distributions. A typical choice for the intermediate distributions are:

$$p_k(x) \propto p_A^*(x)^{1-\beta_k} p_B^*(x)^{\beta_k} \tag{6.2}$$

with $0 < \beta_0 < \beta_1 < ... < \beta_K = 1$ chosen by user. After defining the intermediate distributions, the following procedure can be done for estimating ratio of partition function (taken from [63]):

Annealed Importance sampling run:

1. Generate $x_1, x_2, ...x_K$ as follows:
   - Sample x_1 from $p_A = p_0$
   - Sample x_2 given x_1 from T_1
   - ...
   - Sample x_K given x_{K-1} from T_{K-1}
2. Set

$$\omega^{(i)} = \frac{p_1^*(x_1)}{p_0^*(x_1)}\frac{p_2^*(x_2)}{p_1^*(x_2)}...\frac{p_K^*(x_K)}{p_{K-1}^*(x_{K-1})}$$

It is important that there is no need for commuting normalizing constants of intermediate distributions. After performing M runs of AIS, the estimate of ratio would be obtained by:

$$\frac{Z_B}{Z_A} \approx \frac{1}{M}\sum_{i=1}^{M} \omega^{(i)} = \hat{r}_{AIS} \tag{6.3}$$

If we could define p_A such that its partition function be tractable, then it is possible to get a good estimate of the target partition function of p_B.

6.3 AIS for Restricted Boltzmann Machine

salakhutdinov and Murray proposed the following algorithm for AIS on RBM [63]. In order to estimate the partition function of a target RBM with parameters $\theta_B = \{W^B, b^B, a^B\}$, RBM is defined with zero weights $\theta_A = \{0, b^A, a^A\}$. These RBMs have distributions p_A, p_B over visible variables space $\nu \in \{0,1\}^D$. The intermediate distributions according to AIS generic choice are:

$$p_k(v) = \frac{p_k^*(v)}{Z_k} = \frac{1}{Z_k}\sum_h exp(-E_k(v,h)) \tag{6.4}$$

with the energy function (given in Eq. 4.3):

$$E_k(v,h) = (1-\beta_k)E(v,h^A;\theta_A) + \beta_k E(v,h^B;\theta_B) \tag{6.5}$$

with $0 < \beta_0 < \beta_1 < ... < \beta_K = 1$. Therefore, for $i = 0$ there is $p_0 = p_A$ and for $i = K$ there is $p_K = p_B$. The Markov chain transition operator necessary for AIS is proposed to be defined as the following. Using equations 6.4, 6.5, it is easy to derive the conditional distributions of RBM which could be used as Gibbs sampler for visible variables of intermediate distributions:

$$p(h_j^A = 1|v) = \sigma((1-\beta_k)b_j^A) \tag{6.6}$$

$$p(h_j^B = 1|v) = \sigma(\beta_k \sum_i W_{ij}^B v_i + b_j^B) \tag{6.7}$$

$$p(v_i' = 1|h) = \sigma((1-\beta_k)a_i^A + \beta_k(\sum_j W_{ij}^B h_j^B + a_i^B)) \tag{6.8}$$

Therefore, given v, v' can be sampled using these conditional distributions. The next condition of AIS was being able to compute unnormalized probability of intermediates. According to equation 6.4, it is needed to marginalize h in order to get the $p_k^*(v)$. Due to special structure of RBM, summing over h has a linear cost in terms of number of hidden nodes:

$$p_k^*(v) = \sum_{h^A,h^B} e^{(1-\beta_k)E(v,h^A)+\beta_k E(v,h^B)} \tag{6.9}$$

$$= e^{(1-\beta_k)\sum_i a_i^A v_i} \prod_j (1 + e^{(1-\beta_k)b_j^A})$$

$$\times e^{\beta_k \sum_i a_i^B v_i} \prod_j (1 + e^{\beta_k(\sum_i W_{ij}^B v_i + b_j^B))}$$

After performing AIS run using the described equations, the partition function of RBM A is needed to finalize the calculation of normalizing constant of RBM B. Based on Eq. 4.2 and 4.3, the partition function of A is computed by:

$$Z_A = \prod_j (1 + e^{b_j}) \prod_i (1 + e^{a_i}) \tag{6.10}$$

Therefore, by using Eq. 6.3 and Z_A, the Z_B is computable.

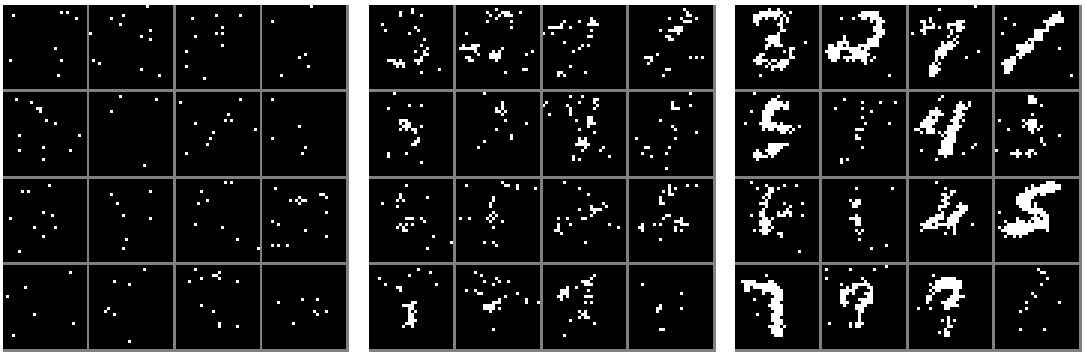
6.4 AIS for Convolutional RBM

We adapted the formulation of AIS from RBM to the convolutional RBM. The intermediate distributions are the same as RBM with the difference that energy functions have the form of CRBM. The transition operator based on Eq. 4.14, 4.15 and 6.4 have the following form:

$$p((h_A^k)_{ij} = 1|v) = \sigma((1 - \beta_l)b_k^A))_{ij} \tag{6.11}$$

$$p((h_B^k)_{ij} = 1|v) = \sigma(\beta_l \sum_k W_B^k * v + b_k^B)_{ij} \tag{6.12}$$

$$p(v'_{ij} = 1|h) = \sigma((1 - \beta_l)c^A + \beta_l(\sum_k W_B^k * h^B + c^B))_{ij} \tag{6.13}$$



(a) (b) (c)

Figure 6.1: Generated samples during AIS run : (a) samples at $k = 0$ (b) samples at $k = 10000$ (c) samples at $k = 14000$.

with $0 < \beta_0 < \beta_1 < ... < \beta_l = 1$ for the AIS run. The unnormalized intermediate probability distributions $p_k^*(v)$ have the following form and the proof is given in the appendix B.

$$p_l^*(v) = \sum_{h^A, h^B} e^{(1-\beta_l)E(v,h^A)+\beta_l E(v,h^B)} \tag{6.14}$$

$$= e^{(1-\beta_l)c^A \sum_{ij} v_{ij}} \prod_k (1 + e^{(1-\beta_l)b_k^A})$$

$$\times e^{\beta_l c^B \sum_{ij} v_{ij}} \prod_k (1 + e^{\beta_l \sum_{ij} (\widetilde{W_B^k} * v + b_k^B)_{ij}})$$

Finally, the partition function of CRBM A is needed to estimate target constant Z_B:

$$Z_A = (\prod_k (1 + e^{N_h^2 b_k})) \times (1 + e^c)^{N_v^2} \tag{6.15}$$

6.5 Estimating the Log-likelihood

In section 4.4.4, it was explained that the free energy is the log of unnormalized probability of data. In other words, the unnormalized log-likelihood of data under model distribution. Hence, the F of data point x for convolutional RBM is:

$$F(x) = -c \sum_{i,j} x_{i,j} - \sum_{k=1}^{K} \log(1 + exp(\sum_{i,j} (\widetilde{W^k} * x)_{i,j}) + b_k) \tag{6.16}$$

Table 6.1: Comparison of log-likelihood of CRBM with RBM on MNIST

The Model	LogZ	LogZ $\pm\sigma$	Avg. Train Log-Prob	Avg. Test Log-Prob
RBM500-CD1[63]	350.15	0.1	-122.86	-125.53
RBM500-CD3[63]	280.09	1.9	-102.81	-105.50
RBM500-CD25[63]	451.28	0.09	-83.1	-86.34
CRBM32x21x21CD1	176.56	3.4	-128.04	-129.1
CRBM32x21x21CD25	193.1	1.1	-105.2	-107.09

After AIS run and estimating the Z of model, the log-likelihood of example x is obtained by:

$$\log p(x) = F(x) - \log Z \tag{6.17}$$

We implemented the AIS for convolutional RBM and evaluated on MNIST dataset. In the experiments the CRBM has 32, 8×8 filters. The contrastive divergences with $k = 1$, and $k = 25$ were performed to compare their log-likelihoods. It is expected that RBM with more contrastive divergence steps, learn a model with higher likelihood of data. We used the same experimental setup as suggested in [63] in our implementation. We used 500 β_k spaced uniformly from 0 to 0.5, 4000 β_k spaced uniformly from 0.5 to 0.9, and 10,000 β_k spaced uniformly from 0.9 to 1.0, with a total of $14,500$ intermediate distributions. Figure 6.1 shows the generated samples of AIS run at different stage of run. The picture shows that algorithm converges (Model successfully mixes) at last iterations.

Table 6.1 reports the estimated partition functions and average log-likelihood and compares with reported numbers of [63]. The RBM500-CD25 is the RBM with 500 hidden nodes and 25 contrastive divergence iterations. The average log-likelihood on test set of MNIST for CRBM model with much less parameters than RBM500 is -107.09 which is relatively close enough to RBM's number. To clarify this point, the RBM500 model has $784 \times 500 = 392000$ parameters while CRBM32x21x21 has $32 \times 8 \times 8 = 2048$ parameters (without considering biases). Thus the CRBM can learn a good model in comparison with RBM with a huge number of parameters. Moreover, the average log-likelihood of training and validation are very close which demonstrate good generalization of models.

6.6 Denoising Auto-encoders as Energy-based Model

Having a good insight into the learning process of denoising auto-encoder yields better training and model selection. Vincent et al. [68] have shown a connection between training criterion of DAE and score matching (SM) of an energy-based model with a particular energy function. Obtaining this energy model yields several benefits. The main benefit for our application is that we could avoid over-fitting by monitoring the free energy of model. We explain this theory and generalize it for convolutional DAE. In this theory for an energy based model with the following probability distribution:

$$p(x;\theta) = \frac{1}{Z(\theta)} exp(-E(x;\theta)) \tag{6.18}$$

The score is the derivate of log density with respect to data point: $\psi(x;\theta) = \frac{\partial \log p(x;\theta)}{\partial x}$. The core principle of score matching is learning parameter θ so that score $\psi(x;\theta)$ best matches the corresponding score of true distribution $\frac{\partial \log q(x)}{\partial x}$. Thus the objective function is expected value of squared error of two vectors under true distribution:

$$J_{SM} = E_{q(x)} \left[\frac{1}{2} \left\| \psi(x;\theta) - \frac{\partial \log q(x)}{\partial x} \right\|^2 \right] \tag{6.19}$$

In the framework of [68],the following necessary conditions should hold in order to prove that the training denoising auto-encoder with squared error is equivalent to the score matching. First condition is that the input x should be corrupted by Gaussian additive noise ($\widetilde{x} = x + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$). Second, the objective should be mean squared error of input and reconstruction of corrupted input. Therefore the DAE should minimize the following objective function:

$$J_{DAE} = E_{q(x,\widetilde{x})} \left[\frac{1}{2} \| W^T sigmoid(W\widetilde{x} + b) + c - x \|^2 \right] \tag{6.20}$$

Where $q(x, \widetilde{x})$ is the joint distribution of x and corrupted $\widetilde{x}$ (equations are in the Appendix C). Given these necessary conditions and some other conditions discussed in [68], it is shown in [68] that minimizing the J_{DAE} is equivalent of score matching of a energy based model with the following energy function:

$$E(x; W, b, c) = -\frac{1}{\sigma^2} (c^T x - \frac{1}{2} \|x\|^2 + \sum_j softplus(\langle W_j, x \rangle + b_j)) \tag{6.21}$$

where the function $softplus(x) = \log(1 + e^x)$. In the following, We generalize this framework for convolutional denoising auto-encoder discussed in previous chapter. The corresponding energy function of convolutional DAE is the following:

$$E(x; W, b, c) = -\frac{1}{\sigma^2}(\sum_{ij}(cx + \sum_k softplus(W^k * x + b_k))_{ij} - \frac{1}{2}\|x\|^2) \tag{6.22}$$

Where c is a constant bias term and x is a matrix. The proof for obtaining this equation is given in the Appendix C. The energy function can be used to monitor the training of our model. We compute the difference of energy function under training and validation set as a measure of over-fitting. The free energy is equal to the energy function for this model:

$$F(x) = -\frac{1}{\sigma^2}(\sum_{ij}(cx + \sum_k softplus(W^k * x + b_k))_{ij} - \frac{1}{2}\|x\|^2) \tag{6.23}$$

6.7 Summary

We presented a Monte Carlo approximation algorithm for convolutional RBM to estimate log-likelihood of data under the model distribution. The log-likelihood of data under model distribution is an indicator or measurement of goodness of a learned generative model. We tested the AIS algorithm for convolutional RBM on MNIST data set. The results suggest that CRBM model is able to learn a good generative model of dataset with dramatically fewer parameters than RBM. Moreover, Denoising auto-encoders were discussed with their corresponding energy based model. We discussed our generalization of DAE for convolutional version.

Chapter 7

Spike Pattern Recognition

7.1 Introduction

This chapter presents the pattern extraction stage of the methodology. In the previous chapters we discussed two proposed unsupervised learning algorithms for the first stage of system. The purpose of first stage was learning distinct filters. The obtained filters are able transform input data into new representations. The extraction stage of our methodology incorporates new representation of data and extract patterns. There are some considerations associated with this stage. Firstly, the learned filters do not necessarily reflect distinct patterns. There might be filters that reflect mixture of two patterns due to coincidence of occurrences. Moreover, there might be filters that reflect noise or smeared patterns. In addition, it is a possibility that a filter only reflect small part of actual pattern due to bad design. Therefore, it is crucial that the extraction stage to address these issues.

7.2 Model Design for Spike Data

The multi-cell recoding is a binary two dimensional array of neurons spike record. The system is searching for motifs or repeated structural activity of neurons over time. Therefore, the filter size of learning algorithms should be designed such that filters transform data into a feature space where occurrence of patterns over time could be revealed. In other words, filters size should be such that new representation be a one dimensional time representation of patterns. The input space of convolutional RBM and convolutional DAE is $N_v \times N_v$. In order to fit the data for model, the data with typically long time dimensions is sliced into many instances of $N_v \times N_v$ matrices. In our system, the filter size is designed to be $N_v \times N_f$. Therefore, each pattern detector's receptive field contains activity of all the neurons (N_v) over N_f time steps. Given this filter design, the hidden representation of input would be an ($H^k = 1 \times N_v - N_f + 1$)

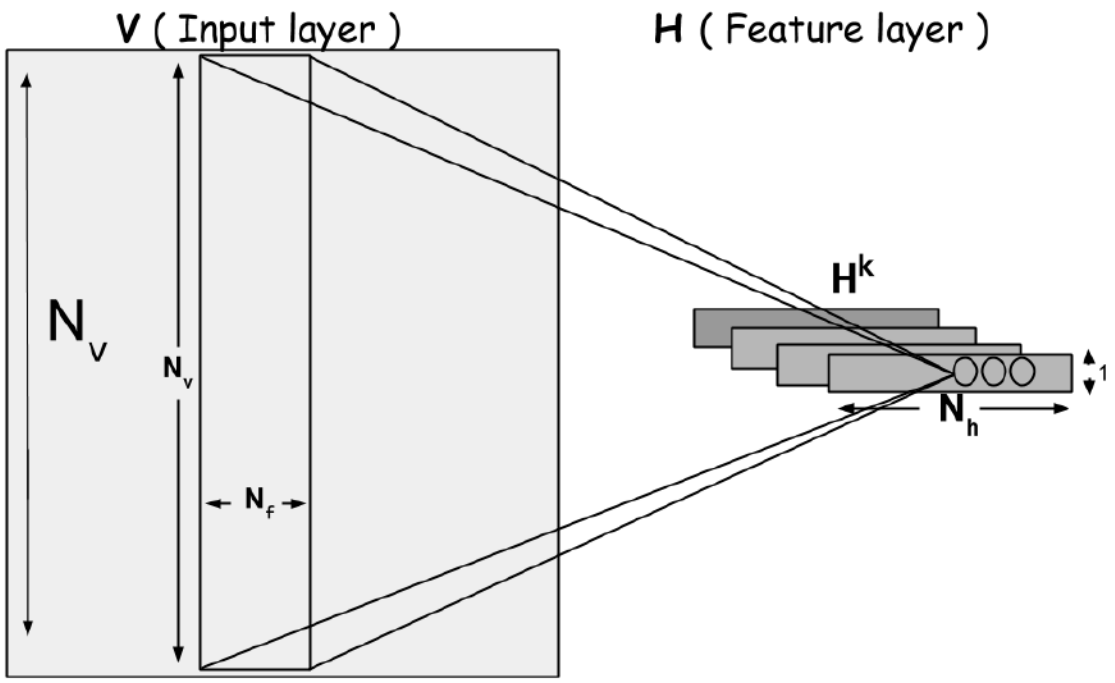


Figure 7.1: Design of architecture of the proposed unsupervised learning algorithms for learning spatiotemporal filters. Both of the convolutional RBM and convolutional DAE have the same filter design.

array. Ideally these array would represent presence of patterns captured by a filter. The Figure 7.1 demonstrates the designed model for unsupervised learning. This filter design helps us in the extraction stage to search for time steps that have maximum probability of presence of a pattern. It should be mentioned that in our design, the height of filter is the same as height of input because we expect filter to look at all neurons at the same time and pick up those spikes that have structural organization.

7.3 Extraction of Patterns

At this stage after training CRBM and CDAE on data, the learned filters are used to extract the patterns. The filters transform data into hidden representation in order to work with a simpler representation. The core principle of extraction is to create a map of structure of patterns. In other words, creating a probability map which describes probability of spike of neurons in a pattern-like activity. Let's visually illustrate the point. The first step of extraction is to find time steps that filters indicate there are

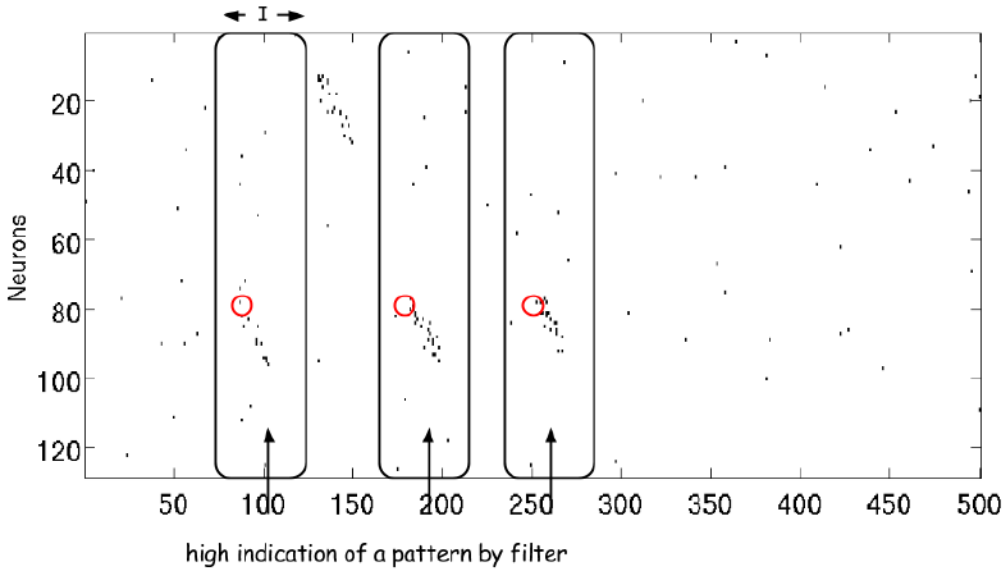


Figure 7.2: Illustration of computing probability of spike around time steps with high indication of pattern by a filter.

high chance of presence for a pattern. Figure 7.2 shows these time steps by uparrow. Within interval I around the time steps, the repetition of spikes is counted over the indicated time steps (for example spike with circle mark in Figure 7.2 repeated 3 times). By normalizing this count by number of time steps, the probability of that spike within interval I is computed. If this probability is computed for all possible spikes in the interval I, the matrix that contains probabilities is called probability map within interval I. This probability map indicates the likelihood of spike of neurons when a filter suggests occurrence of a pattern. This map describes the structure of occurred patterns. In our system, the new representation of data is used to detect time steps that there is high indication for presence of patterns.

Obtaining new representation of data for each method is different. The CRBM is a probabilistic model and the values of hidden representations are samples of distribution in equation 4.14. This equation is the conditional probability of hidden space h^k given data input v ($p(h^k|v)$). In our system, the probability distribution $p(h|v)$ is considered as the new representation. Please note that input v at this stage could the whole multi-cell recording with arbitrary time dimension, so v could have size of $N_v \times T$ where T is total number of time measurements (the new representation would be $1 \times T - N_f + 1$ array). For CAE model, the feature space h^k of given

input v is computed by equation 5.5. Each hidden variable h_j^k in the new representation determines the likelihood for presence of filter W^k at time step j. There are strong indication for presence of a pattern at time steps that likelihoods are high in h. From signal processing perspective, local peaks in the signal h determine high chance of pattern existence. Therefore, we define operation $max_l\{h^k\}$ to return set l which contains indexes that have maximum value in the vector h^k (in other words $l = \{t_i : h^k(t_i)$ is a local maximum $\}$). Given these indexes, we construct the probability map for τ window around l time points. To put it formally, the map of kth filter is $MAP_\tau^k = \frac{1}{\|l\|}\sum_{i\in l} x(1 : N_v, i - \tau/2 : i + \tau/2)$ where l is returned by max operation and x is input data. The matrix $x(1 : N_v, i - \tau/2 : i + \tau/2)$ are illustrated by the boxes in Figure 7.2.

We mentioned previously that filters might capture a small part of patterns. To address this issue, we set $\tau \gg N_f$ so that probability maps be over a large time window. The N_f of each filter determines the maximum number of time steps of a captured pattern. If the filters capture a pattern partially, it would be possible to reveal other parts by choosing a big τ. We also discussed the problem of learning filters that contain smeared patterns or noise in the data. To mitigate this problem, the following analysis is carried out to distinguish smeared patterns from clear ones.

If there were no pattern in the activity of neurons then they would be spiking independently and randomly with a constant probability. At this condition, the probability of spike of neuron at any time is estimated to be mean of data. We know that this is not true and neurons are not spiking randomly and there are moments where their activity have a structural form. If the learned filters are supposed to capture such structural activities, then their corresponding MAPs should be as far as possible form randomly independent neurons map. In other words, the probabilities in the MAP should be as far as possible from mean of data. We use Kullback-leibler divergence in order to quantify the distance of a MAP and mean of data (chance).

Kullback-leibler divergence is defined as a measure of distance between two probability distributions. The KL divergence of two distribution p and q is defined as follows:

$$D_{KL}(p||q) = -\sum_x p(x)\log q(x) + \sum_x p(x)\log p(x) = H(p,q) - H(p) \qquad (7.1)$$

Where $H(p,q)$ is cross-entropy and $H(p)$ is entropy. This is a measure of information lost when q is used to approximate p. The p and q for our application are defined as follows. We divide MAP by its sum and treat it as p distribution. This is the distribution of probability of spike of neurons within interval τ. The q is based on the MAP of chance for τ interval over all time steps of data which is computed by $MAP_{\tau}^{chance} = \frac{1}{|T|}\sum_{i}^{T} x(:, i-\tau/2 : i+\tau/2)$. Similar to computing MAP but instead of summing over only l time steps, it is summed over all time steps. The chance MAP is divided by its sum to create the q distribution. The KL divergence of this two distributions determines how much information is lost, if the pattern MAP is approximated with chance (randomly spiking neurons). If the MAP is a sharp and distinct pattern, KL should be big number. It is very easy to rank the obtained MAPs of system by the KL distance from chance. By ranking the MAPs, the best ones whose distance is greater than a threshold can be selected. If a MAP contain smeared pattern or noise its distance would be high enough to be separated.

7.4 Summary

We presented a two stage pattern recognition paradigm for extracting spike patterns. The architecture of unsupervised learning for the application were explained with detailed reasons of choices. It was explained that how the learned filters are used to create a map which describes the pattern in the data. Our system uses KL divergence for distinguishing the best detected patterns.

Chapter 8

Experimental Results

8.1 Introduction

Evaluating the proposed algorithms on the multi-cell recordings is not possible since existing patterns in the empirical data are not known and the purpose of our research is to discover those patterns. Hence, we generate synthetic datasets which have basic properties of actual multi-cell recordings and known patterns embedded at known times. Using artificial input for evaluation has two benefits. First, it is possible to visually inspect that algorithms have indeed detected the injected patterns. Second, it is possible to manipulate the parameters of data and measure accuracy of algorithms. The manipulation of the data are typically intended to make the problem harder. For example, the noise in the data or the variability of structure of patterns can be manipulated. By comparing accuracies, the most reliable detection approach can be selected among the heuristic search algorithm and unsupervised algorithms. In this chapter, first algorithm for generation of artificial data is discussed with the results of methods on the data. Then, the empirical multi-cell recordings and the corresponding experiment will be discussed.

8.2 Generating Synthetic Data

The important factor for the artificial data is that it should resemble observed recordings of the cortex. We know from empirical datasets that the timing of successive spikes are highly irregular. The interpretation of this regularity has led to divergent views of cortical recordings. On the one hand, it has been argued that irregularity arise from stochastic forces and inter-spike intervals convey little information. Alternatively, it has been postulated that timing of spikes, inter-spike intervals and their patterns can convey information. We obviously are on the later side and looking for the patterns. A synthetic cortical recording should have the property of irregular

spike timing and at the same time inter-spike interval should convey information. We incorporate a two stage algorithm for generating data with such property.

In the first stage, the interpretation that inter-spike intervals reflect a random process is assumed. Therefore, the generation of each spike is driven by continuous signal $r(t)$ which will be referred to as instantaneous spike rate. It follows from the assumption that generation of each spike is independent of all others. The spike train can be described by a particular kind of random process called a Poisson process. In the next stage, a pattern embedding algorithm is used in order to have inter-spike interval of some of neurons conveying pattern like activity. The embedding stage changes the instantaneous spike rate of selected neurons at random time points so that they exhibit sequential organization. The core principle of pattern embedding stage is that at some random time steps, a few neurons will fire in a row. However, they are not precise and sometimes they fire with jitter in time.

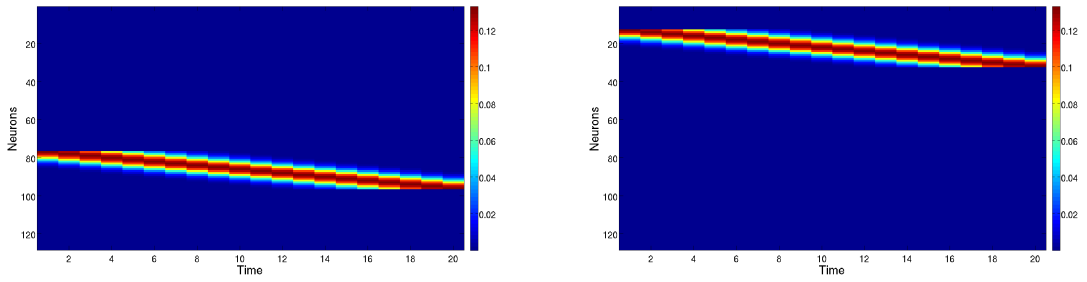
Let us begin by explaining Poisson random process for generating spike train. We assume that underlying instantaneous firing rate is a constant r. Imagine that we are given a long interval $(0, T)$ and we place a single spike in that interval at random. Then we pick a sub-interval $(t1, t2)$ of length $\Delta t = t2 - t1$. The probability that the spike occurred during the sub-interval equals $\Delta t/T$. Now let's place k spikes in the $(0, T)$ interval and find the probability that n of them fall in the $(t1; t2)$ sub-interval. The answer is given by the binomial formula:

$$P\{n\_spikes\_during\Delta t\} = \frac{k!}{(k-n)!n!} p^n q^{k-n} \tag{8.1}$$

where $p = \Delta t/T$ and $q = 1 - p$. If the mean firing rate (ratio $r = k/T$) is constant while k and T are increased, then it can be shown that as $k \to \infty$, the probability that n spike will be in an interval of length Δt equals:

$$P\{n\_spikes\_during\Delta t\} = e^{-r\Delta t \frac{(r\Delta t)^n}{n!}} \tag{8.2}$$

This is the formula for the Poisson probability density function. Therefore by uniformly random placing of spikes in $(0, T)$, the inter-spike interval would be Poisson distributed. Given the above proof, by the following algorithm, a Poisson spike train is generated. The continuous interval $(0, T)$ is divided to small intervals of δt. For each small interval with index i, a sequence of random numbers $x[i]$, uniformly distributed



(a) (b)

Figure 8.1: The Firing rate map R for two groups of neurons at time step $t_0 = 0$ and Gaussian with std $\sigma = 3$. (a) group of neurons with indexes $[76 - 92]$ (b) group of neurons with indexes $[12 - 32]$

between 0 and 1 are generated. If $x[i] < r$, a spike is generated ($X[i] = 1$), otherwise no spike ($X[i] = 0$). This process is performed for multiple neurons independently. Hence, for generating Poisson spike train of N_v neurons for T small intervals, random numbers $x[i, j]$ where $0 < i < N_v$ and $0 < j < T$, are generated. Then if $x[i, j] < r$, a spike is generated, otherwise no spike.

According to [41], the cortical recordings contain sequential structures. Therefore a sequential organization is assumed as the basic pattern to be inserted into the Poisson process. The pattern embedding stage modifies the Poisson spike generation in the following way. The algorithm first selects its target neurons by choosing n_p groups with n_s neurons among all N_v. Let's call the selected groups as γ neurons. Then a set of exponentially distributed random time steps between 0 and T (for each n_p group of neurons separately), are generated. Let us define the generated times as Γ time steps. The algorithm changes the firing rate of γ neurons at Γ time points, such that they spike in a sequence. In other words, the generated numbers x would not be simply compared with r but rather for Γ, they are compared with firing rate map R.

The firing rate R is created in the following way. Since the observations in the multi-cell recordings demonstrate that timing of spikes within the neuronal organizations are not precise [42], the algorithm do not change the firing rate of neurons at precisely the Γ times. It increases the rate of time steps before and after Γ with Gaussian form centered at Γ times. Suppose we want to generate a spike train with 128 neurons and 500 time steps. We select 2 groups with 20 neurons (neurons with

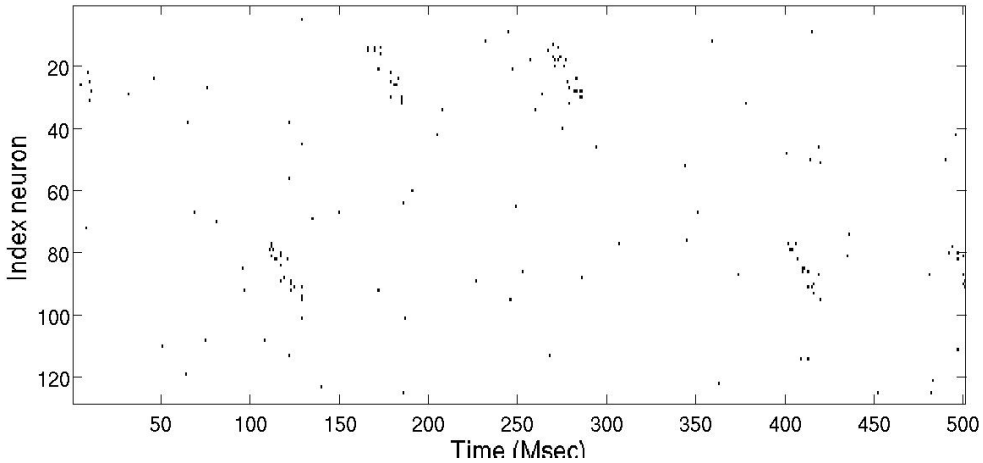


Figure 8.2: The generated data with 128 neurons and 500 time steps. Each dot is a spike. The group of neurons with index $12-32$ and $76-96$ have a sequential spiking pattern at some random time steps.

indexes $[12-32]$ and $[76-96]$). Let's imagine that randomly generated time steps in Γ are $\{t_0, t_1, t_3\}$. For the all time steps t in the interval $(t_0, t_0+\tau)$, the firing rate is a Gaussian with mean t_0 and std σ. The rate of neurons in each group is shifted in time in a way that they form a sequence. For example the neuron with index 12 in the first group has $R(12,t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-t_0)^2}{2\sigma^2}}$, the index 13 has $R(13,t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-(t_0+1))^2}{2\sigma^2}}$ and so on to the last neuron. The center of Gaussian is shifted 1 time step forward for the next neuron in the sequence. The same story goes for the other group $(76-96)$. The rate of all other neurons in the interval $(t_0, t_0+\tau)$ remains as our constant r. The figure 8.1 shows the R for our both groups of neurons separately with $t_0 = 0$, $\tau = 20$, $\sigma = 3$ and $r = 0.01$. For other times in the Γ, the same procedure takes place. The σ parameter determines the jitter of spikes within sequences. We vary this parameter in our experiments to measure robustness of the our algorithms against jitter of spikes.

An important attribute of a realistic synthetic spike recording is that neurons involved in spike patterns are not located near each other. During the recording procedure from animals, when the electrodes are placed in the cortex, the label or index of recorded neurons are arbitrary. Therefore, if a group of neurons were involved in pattern like activity, their index would not necessarily be close to each other. Having such property is critical for our synthetic datasets. We randomly permute

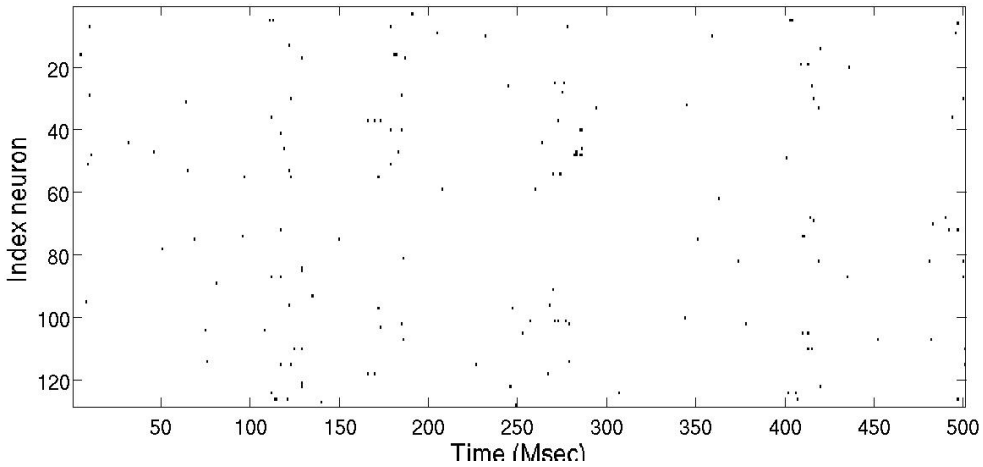


Figure 8.3: The reordered version of generated data with 128 neurons and 500 time steps.

the neurons in the generated data to simulate the situation in the actual multi-cell recordings. In other words, the index of neurons in the data are randomly reordered.

The dataset that we generated for evaluation of algorithms has the following parameters. The number of neurons $N_v = 128$ and $T = 100000$ time steps. The instantaneous firing rate (r) of neurons or the probability of independent spike is 0.01. We have the same selected neurons as in the example for sequential spiking ($12 - 32$ and $76 - 96$). The σ of Gaussian is in most of experiments 3. Figure 8.2 shows the 500 time steps of generated spikes before reordering of neurons. Figure 8.3 shows the permuted version of Figure 8.2.

8.3 Experiments on Synthetic Data

We evaluate our proposed methods by extracting patterns of synthetic dataset. The synthetic dataset contains sequential spiking patterns occurring at random times. The main expected result from our proposed methods is that they find those times and detect which neurons are involved in the pattern. Previous chapters explained how the proposed models work. It is expected that methods roughly find the Γ time steps and estimate the R map that we used to generate our spikes. It was discussed that the detection phase of our unsupervised algorithms first determine the time steps of high indication for a pattern, therefore these high indication time steps should be

very close to Γ times. The probability map of all possible spikes around these high indication should be a good estimate of R.

Measuring the accuracy of detected patterns is not straightforward. However, we define a measure to quantify accuracy of our methods. The accuracy of detection of patterns is defined as the distance between the obtained MAPs (or $Pmap$s) and the firing rate map R. Defining the distance between the two maps is not also easy. It should be considered that the window size of MAPs could be bigger than R. In other words, it is not possible to align these two maps so that spikes involved in patterns coincide. To address these issues, the normalized cross-correlation of two matrices is defined as the measure. What the cross-correlation do is sliding one map over the other and computes the integration of their products for each position. When the maps match, the value of operation is maximized. This is because when the patterns of maps are aligned, they make a large contribution to the integral. The normalized cross-correlation of two matrices F, T at point (u, v) is defined as:

$$ncc(u,v) = \frac{\sum_{x,y}(F(x,y) - \bar{F})(T(x-u, y-u) - \bar{T})}{\sigma_F \sigma_T} \tag{8.3}$$

where $\bar{f}$ is the mean and σ_f is the standard deviation of matrix f. The maximum of ncc is considered as the accuracy measure. In the following, we will discuss results of the proposed methods on the artificial data and their corresponding accuracy.

8.3.1 Heuristic Search

In oder to perform heuristic search on synthetic dataset, we used the following parameters. In the 500 onset time steps of data ($I = 500$), the algorithm selects $n = 3$ spikes as potential patterns (number of γ spikes). The threshold frequency of repetition of these γ spikes is the expected number of spikes during T time steps. The expected number of spikes in the dataset x is $m \times T$ where m is mean of data ($m = mean(x(:))$) and T is number of times. In other words, if the frequency of repetition of γ spikes were greater than $m \times T$, then they would be considered as a potential distinct pattern (please look at algorithm 1). The window size τ for the computing the $Pmap$s was 20 time steps.

We tested the heuristic search on the synthetic dataset to detect the inserted patterns. As it was explained in the chapter 3, the search algorithm returns its patterns

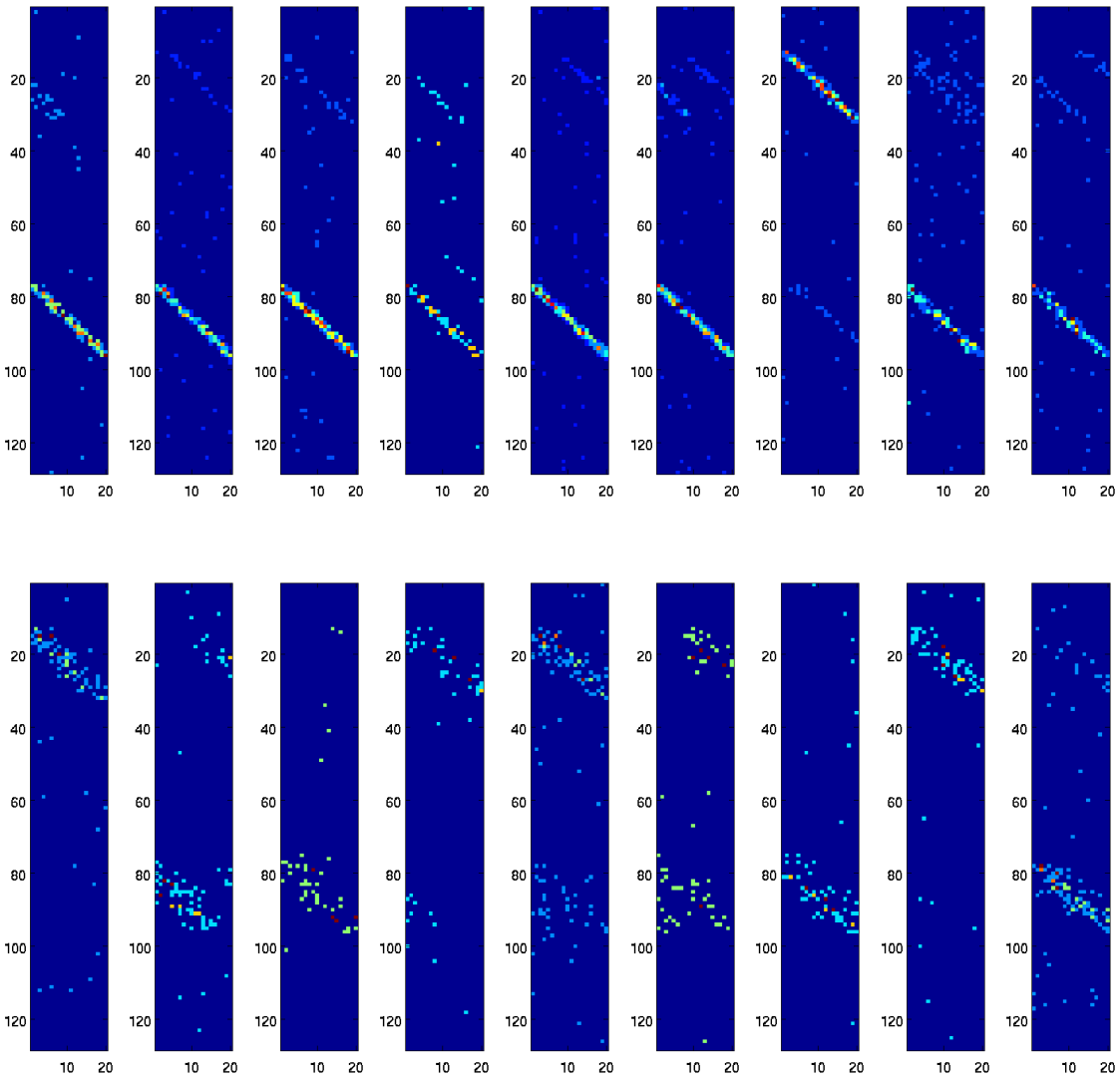


Figure 8.4: Top row are the obtained *Pmaps* of heuristic search from synthetic dataset with jitter $\sigma = 1$. The down row are the *Pmaps* from dataset with jitter $\sigma = 3$. The order of neurons were rearranged to the pre-shuffling arrangement to visualize the detected patterns.

as probability maps ($Pmap$) which describe the probability of spikes in a pattern-like activity. If the search algorithm detect the embedded patterns, the returned maps would be very similar to the firing map R. In order to visually demonstrate execution of the search on the data, Figure 8.4 shows the obtained maps from searching two datasets. The first data is generated with standard deviation of Gaussian (jitter parameter) $\sigma = 1$ in which patterns are a sharp sequence. After searching for patterns, the obtained $Pmaps$ are shown in the top row of Figure 8.4. This visual depiction of $Pmap$s shows that indeed algorithm successfully detected sequential patterns. The other data were generated with $\sigma = 3$ so that the patterns have variations over time. Detection of such transient regularities is much harder for our search algorithm. This can be observed in the obtained maps in the second row of Figure 8.4 which show the results from latter dataset. As explained in the chapter 3 the heuristic search is extremely vulnerable to jitter of spikes. If neurons involved in pattern do not spike at a precise timing, then the computed probabilities in $Pmap$ would be very small due to lack of detection of all occurrences of patterns. The obtained map should be a good estimate of firing rate R, however, if the heuristic search miss time steps then the $Pmap$s would be very weak estimate. To quantify the accuracy of a map in Figure 8.4, we compute the cross-correlation between the firing rate R and the selected $Pmap$ after sorting by KL. The accuracy of the selected map from second row of Figure 8.4 is 0.65. Obviously, with only one result, it is not possible to evaluate the method. Therefore, we will measure the accuracy of method in section 8.3.3 for different training sets with different parameters in order to have a better insight of performance.

8.3.2 Unsupervised Learning

In order to apply convolutional RBM on the synthetic spike data, we have tried many filter designs and many hyper parameters and the following parameters were selected in the end. The input layer were designed to be a 128×128 array and therefore, the generated data were sliced into 80 mini-batches of 100 examples of 128×128 array ($100000/128 \approx 80 \times 100$). There were two set of synthetic data generated, one for training and another for validation. The CRBM had 20 filters of size 128×5. The learning rate were initially 0.01 and gradually decreased during the iterations. The

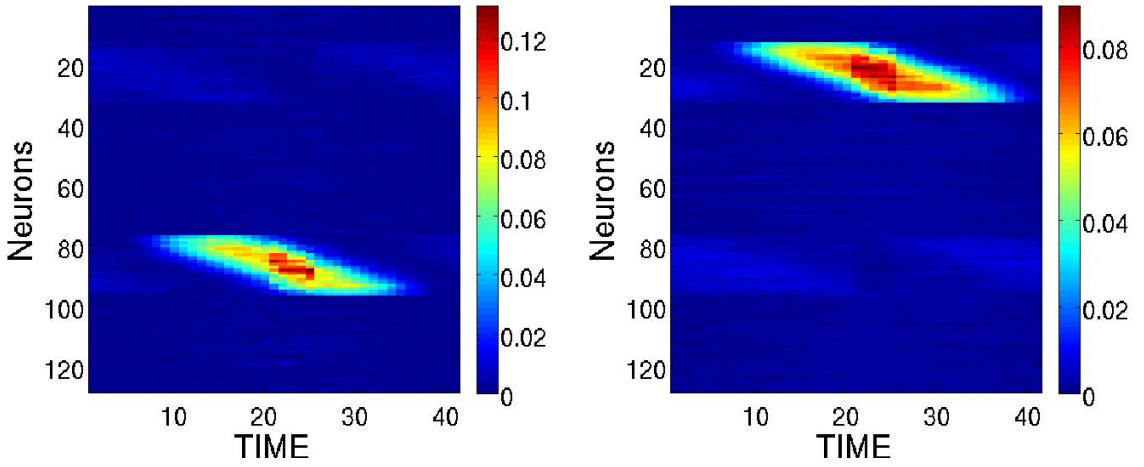


Figure 8.5: The MAPs obtained by convolutional RBM on the synthetic dataset. (a) and (b) are the best selected maps.

momentum were initially 0.5 and were gradually increased to 0.9. The weight decay parameter λ were 0.001. The maximum number of iterations were 2000 however the learning usually stops when the free energy of validation and training set are diverging. After the training phase of our methodology, the detection stage, generated MAPs for all filters with window size parameter $\tau = 40$. The maps were sorted by their KL divergence from chance in order to select the best ones.

Figure 8.5a and 8.5b depict the best MAPs learned by CRBM from generated data with $\sigma = 3$ and after sorting by KL in order to select the best two maps. The order of neurons in the depicted MAPs were restored to the original arrangement before randomization. In other words, we reorder neurons to the arrangement that sequential organizations are visually detectable. The figures show that neurons with index 12-32 and 76-96 have a sequential probability to spike as we expected. By computing the normalized cross-correlation of these two maps and their corresponding R, the accuracy of them are 0.86 and 0.81. This accuracy means that two maps have more than 80 percent overlap with the actual firing rate and therefore our method was able to detect structure of embedded patterns. Figure 8.6 depicts the learned filters of the model. The area with red color in the filters have higher values which means filter detect a pattern if it exist there and the areas with blue color indicate smaller values and filter do not care about that region. In this visual inspection, it can be

Table 8.1: Comparison of log-likelihood of two CRBM models on Synthetic dataset

Model	LogZ	LogZ $\pm\sigma$	Avg. Train Log-Prob	Avg. Test Log-Prob
CRBM20x128x5	520.2	0.08	-188.32	-189.4
CRBM20x128x15	821.1	1.3	-245.02	-248.9

revealed that filters of the model are able to detect the sequential patterns that are occurring in the data. Figure 8.7 has the rest of obtained maps of model that are sorted by KL divergence. It can be observed from this sorting that the last maps are less accurate in terms of estimating the actual pattern of data. They contain mostly noisy activities in the data.

In order to demonstrate a quantitative measurement of the learned CRBM model the log-likelihood of synthetic dataset under the model distribution is estimated. For training the model two dataset is generated with the same parameters, training set and validation set. The log-likelihood of training data and validation set are estimated to demonstrate that the learn models have a good generalization. Table 8.1 reports the estimated numbers. The average log-likelihood of CRBM with $20 \times 128 \times 5$ filters for the training set was -188.32 and validation set was -189.4 which shows a good generalization because the numbers are very close. Moreover, the table compares estimated numbers of CRBM with $20 \times 128 \times 5$ filters and CRBM with $20 \times 128 \times 15$ which has a bigger N_f and therefore bigger filters. The CRBM with $20 \times 128 \times 15$ filters has -245.02 and -248.9 average log-likelihood. The numbers in comparison to the first model, can be interpreted in this way that CRBM with smaller filters have a better generalization. We speculate that the reason for this better generalization is that training a model with smaller filters is easier since there are fewer learn-able parameters. However, it is not good to design very small filters (e.g. $N_f = 1$ or $N_f = 2$) since it would not be able to capture the patterns with jitter.

Moreover, the convolutional DAE is also evaluated by training on the synthetic dataset. The architecture of the model and hyper parameters are selected the same as CRBM. The training objective function is cross-entropy of reconstruction and input since the data is binary. After training the model on the dataset and obtaining filters, the detection phase create MAPs with $\tau = 40$. Figure 8.8 shows the MAPs obtained

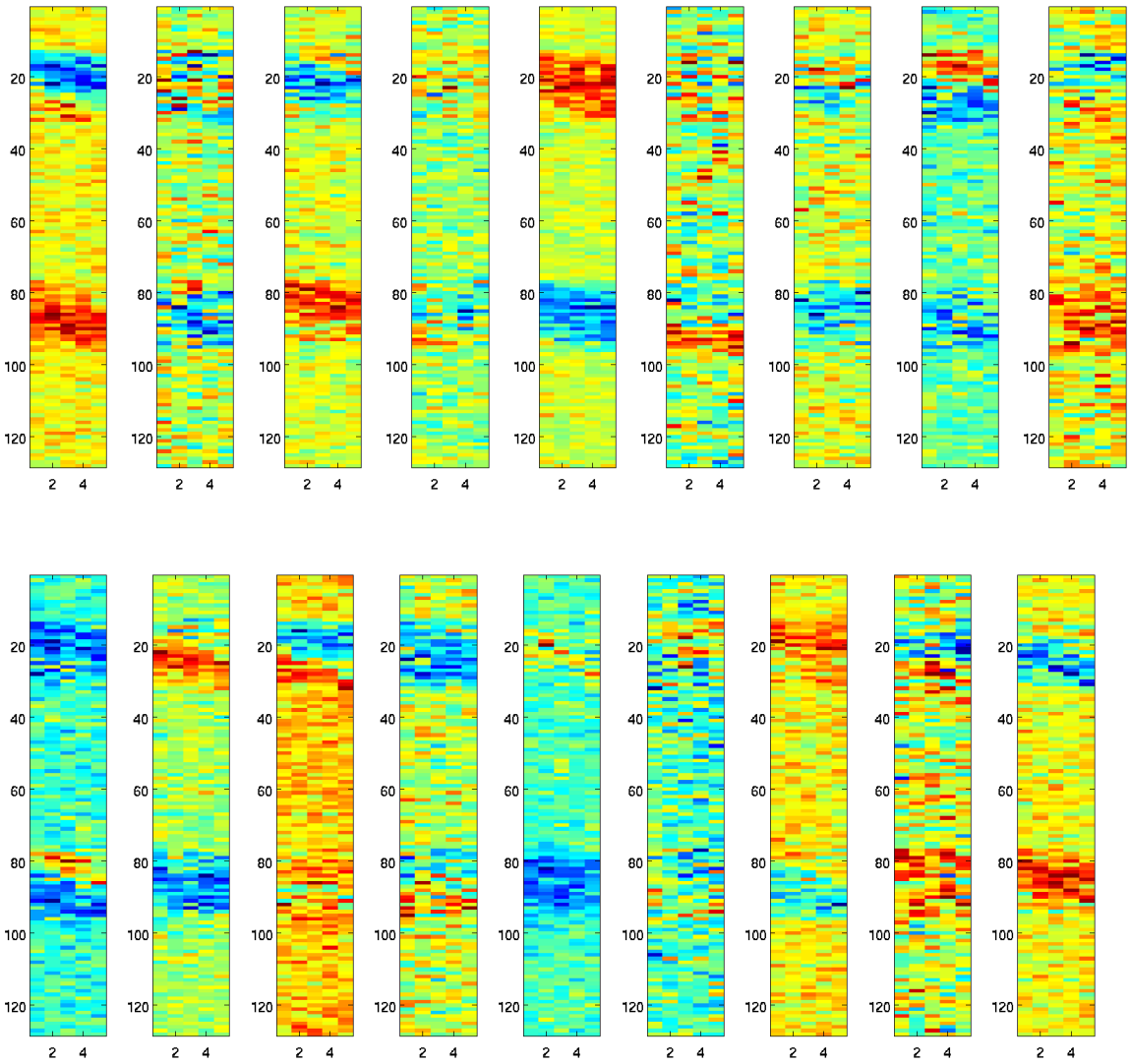


Figure 8.6: The obtained filters by convolutional RBM from synthetic data with $\sigma = 3$. The neurons of filters are rearranged to the pre-shuffling order to demonstrate visually what the filters have captured.

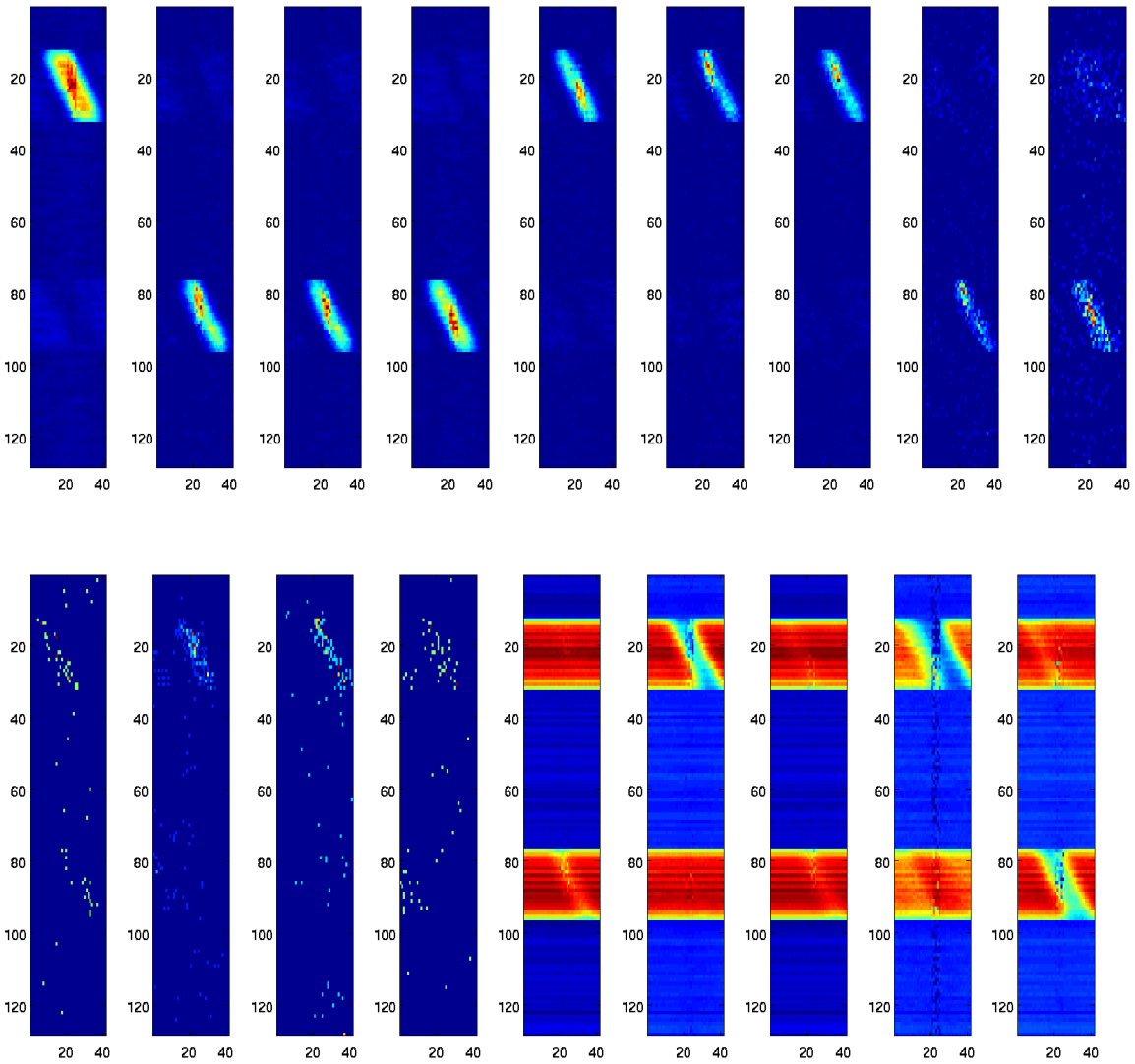


Figure 8.7: The obtained *MAP*s of convolutional RBM and sorted by KL divergence from top left to bottom right. The Neurons of each map are reordered to the pre-shuffling arrangement to demonstrate visually the obtained results.

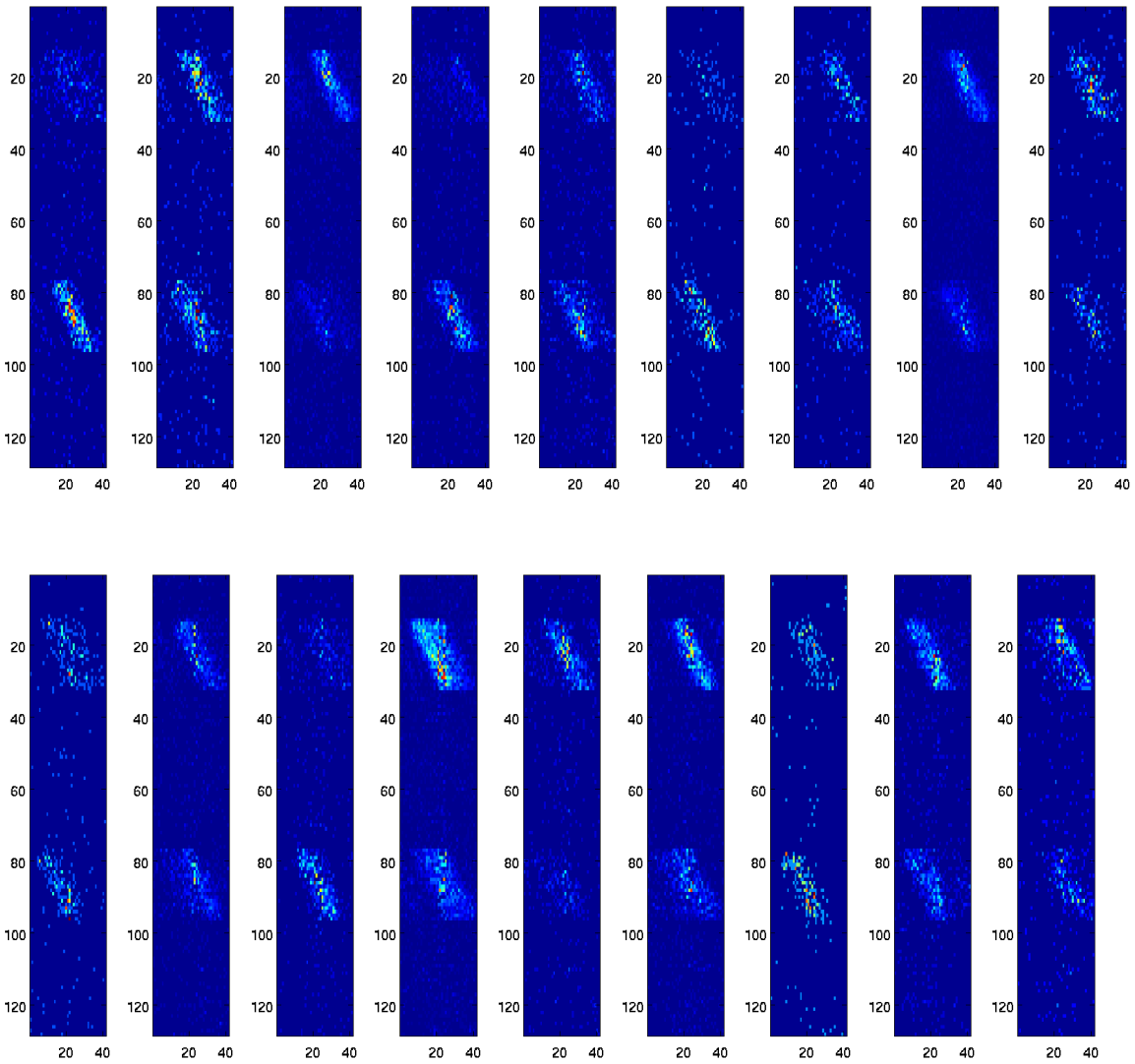


Figure 8.8: The obtained MAPs by convolutional denoising auto-encoder. The Neurons of each map are reordered to the pre-shuffling arrangement to demonstrate visually the obtained results.

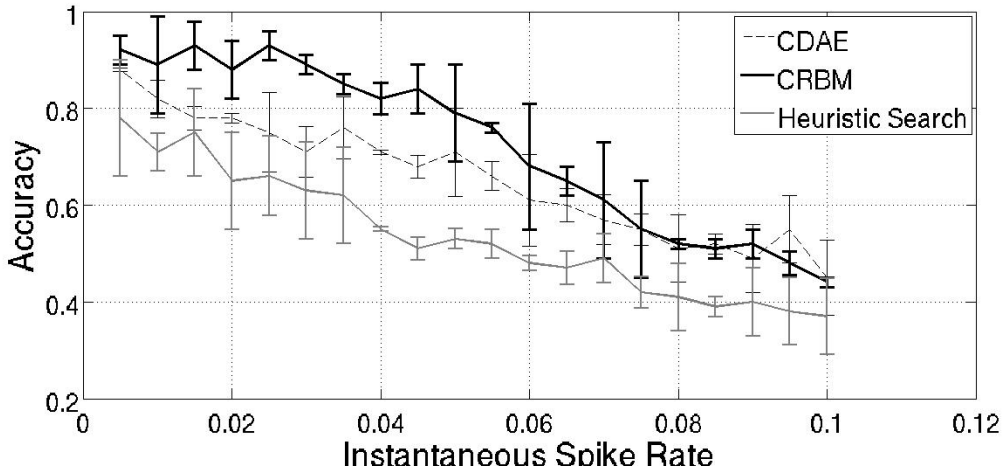


Figure 8.9: The plot of performance against increasing instantaneous spike rate.

by CDAE to visualize the detected patterns. The normalized cross-correlation between the best selected map and the firing rate map R is 0.62. The 60 percent overlap of two maps shows that it has captured the pattern however the accuracy is lower than the CRBM model. We will compare the two methods more systematically in the following section.

8.3.3 Evaluation of Algorithms against Data Manipulation

In this section we will discuss the evaluation of methods on different generated datasets. The main purpose is the evaluation of our algorithms on generated datasets with different parameters. The proposed algorithms should be able to detect patterns of multi-cell recordings even in highly hard conditions. For example, if the records are highly noisy or when the patterns are highly variable. Therefore, the parameters of synthetic dataset are manipulated to measure the accuracy of each method against the changes. In the following, we describe the effect of these changes.

Effect of Noise

One of the main difficulties in our task is the noise. The probability of instantaneous spike (r) determines the amount of noisy activities. The proposed algorithms should be able to deal with highly noisy activity of neurons. The instantaneous spike rate of empirical multi-cell recordings is roughly 0.05. Therefore, it is important that our

methods be able to recognize patterns of synthetic dataset with such spike rate. Figure 8.9 shows the result of the corresponding experiment which is the plot of accuracy versus noise of data. The accuracy in the plot is the cross-correlation between the best selected MAP of each method. We obtain the maps, then we sort them by KL divergence to select the best one. Then we compute the normalized cross-correlation of best MAP with the R of all patterns. The maximum of normalized cross-correlations, determines the accuracy of method. The accuracy is between 0 and 1 and indicates percentage of match of two maps. It is shown in the figure that as we increase the noise, the performance of all algorithms decrease. The accuracy of heuristic search rapidly drops with more noise in the data. The performance drops since finding the right combination that fall into the patterns becomes much more harder when the activity of neurons become more uncorrelated. The CRBM is the best method and demonstrates a robust accuracy against noise. The accuracy of CRBM for firing rate of 0.05 is around 0.83 which is relatively acceptable.

We speculate that the reason for such performance is that CRBM is a stochastic generative model. During the training, some group of hidden nodes learn to model noise and other hidden nodes learn patterns in the data. In other words, during the training, algorithm decreases the energy of observed noise in addition to observed patterns in the data which maximize the log-likelihood. Hence, some of the filters learn the noise of data and other filters become pattern detectors. When the MAPs of filters are sorted by KL divergence, the noise detectors can be marginalized. This ability of RBM for handling noise in the data even better than denoising auto-encoder has been reported on background noise of MNIST as well [57]. The denoising auto-encoder is more vulnerable to noise if our inserted noise to the model and the actual noise of data are not the same.

Effect of Jitter

The next important factor in this application is jitter of individual spikes within patterns. The parameter that determines jitter is the σ of Gaussian in the firing rate R. We did a performance analysis on Jitter by increasing the σ. By this increment, the Gaussian of firing rate becomes wider and therefore the neurons will spike with less precise timing in the sequence. Increasing this parameter causes the sequential

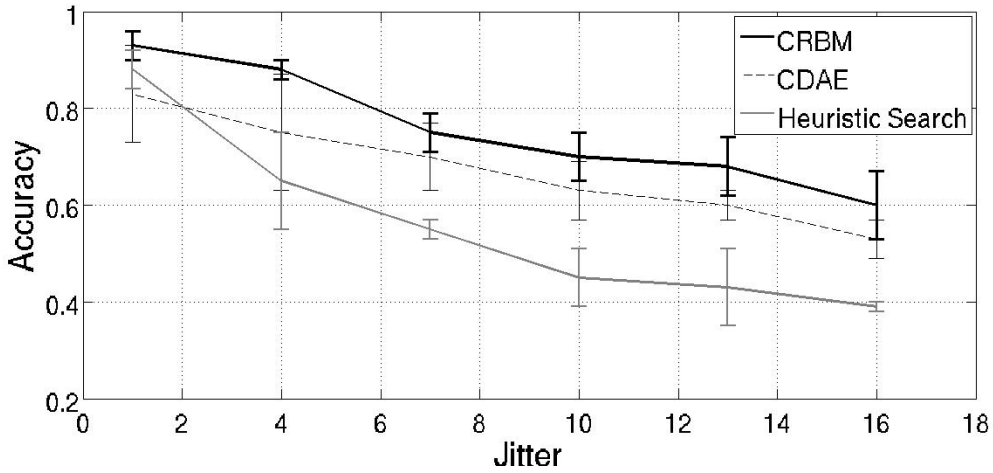


Figure 8.10: The plot of performance against increasing jitter of spikes.

activities, spread over time and correlations between spikes become weaker. Therefore it is much harder to detect the sequences over time. Figure 8.10 shows the performance of all methods against jitter. The plot demonstrates that heuristic search dramatically lose its performance as we expected to happen. The reason is that heuristic search looks for precise occurrence of a combination of spikes over time. Increase in the jitter means that probability of precise occurrence of that combination becomes very low and the obtained $Pmap$ would be a bad estimate of R.

The CRBM method demonstrates the best performance against jitter over CDAE. We again surmise that this superior performance is because CRBM is a generative probabilistic model. As we train it on data, we are maximizing the likelihood of all observed jitters of spikes in the patterns, while decreasing the likelihood of all possible unobserved jitters. This gives a representation power to the model and allow various hidden nodes learn the jitters. In other words, the model learns the probable variations of its patterns. Therefore, the learned filters of CRBM can detect patterns with variations over their occurrences.

Effect of Number of Neurons

An important aspect of our task is scalability of proposed algorithms. As the number of recorded neurons in multi-cell recordings increases, the problem of pattern detection becomes harder. In other words, increase in the number of recorded neurons and

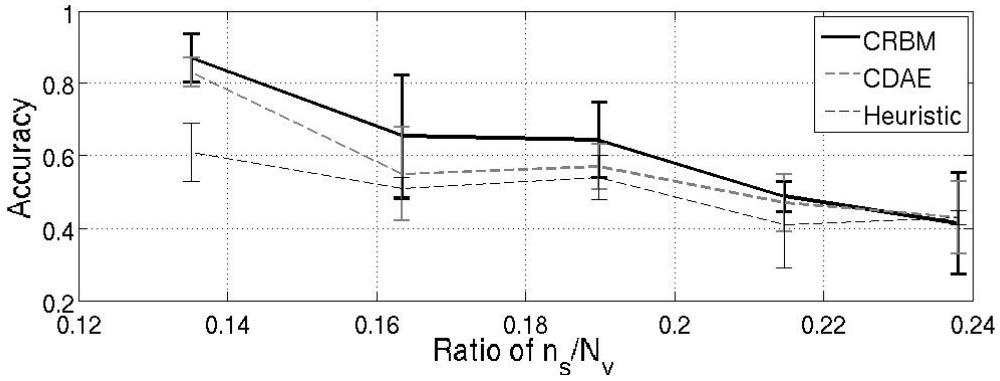


Figure 8.11: The plot of performance against increasing ratio of number of pattern neurons over total number of neurons.

consequently increase in the possible number of neurons involved in the patterns, makes the recognition problem much harder. We evaluate the ability of our proposed algorithms against this challenge. Figure 8.11 depicts the performance of algorithms against increase in ratio of $\frac{n_s}{N_v}$ where N_v is total number of neurons and n_s is number of neurons in the selected groups for sequential activity. In other words, the dimension of patterns that are occurring over time is increasing. In this way, the filter size of unsupervised learning also need to be increased which mean more parameters to be learned. Therefore, training the algorithms would be harder as well. The N_v is multiplied by 2 and n_s by 2.2 to increase the ratio.

Figure 8.11 demonstrate that performance of algorithm drops by this increase. However the CRBM model shows a relatively more stable performance than other two methods. The reason of drop for heuristic method is obvious. If there are more neurons, there would be more possible correlated spikes in the data. Hence, the likelihood of randomly picking the spikes of patterns becomes lower. We speculate the reason of the drop for unsupervised learning algorithms is bad design. The data with larger patterns and input size need more careful choose of filter and hyper parameters. For every data, different designs should be tested and compared by a quantitative analysis such as the analysis of estimating log-likelihood to select the best matching filter design.

8.4 Empirical Multi-cell Recordings

The experimental multi-cell recordings are provided by [41]. The data contains records of spikes of 105 neurons in the auditory cortex of rats. The entire dataset contains 40 minutes record of spikes. The sampling time resolution is 3.2 ms which means that recorder device in the brain captures the spikes every 3.2 ms. The rats were presented by multiple auditory stimulus from natural sounds to tones with different frequencies. The external inputs were exposed to animals every 2 seconds for a duration of 500 ms. The reason for these stimulus is to capture the patterns of activity of neurons when animal is exposed to external input. The data is first pre-processed to create stimulus-driven dataset and spontaneous-activities data and then the CRBM model is applied to discover patterns in these two datasets.

8.4.1 Preprocessing

In the preprocessing stage, the onset 100 ms activity of neurons in response to stimuli are concatenated to form the stimulus-driven dataset. In other words, the onset 100 ms activities at the beginning of presence of stimuluses every 2 second, are taken out. The spontaneous activity dataset is the spiking activity of neurons without any stimulus. Hence, the last 100 ms activity of neurons before next stimuli, are taken out. The external input is only present for 500 ms and 2 sec later, next stimuli is present, therefore the 500 ms of activities before next one could be considered as spontaneous activity. The spontaneous dataset contains record of 105 neurons for 186170 ms and stimulus-locked dataset contains 114208 ms.

8.4.2 Training CRBM

The convolutional RBM is selected for discovering the patterns of empirical datasets because this method had the best performance on the synthetic dataset. The following parameters are selected for this experiment. The model has 20 filters of 105×5. The hyper parameters are the same as CRBM for synthetic dataset. Figure 8.12 shows 8 obtained filters after training of model on stimulus-driven dataset. After training the model, the corresponding MAP of each filter with $\tau = 40$ are computed. The obtained maps are then sorted by KL divergence from chance (mean of data) and

16 MAP are selected as the best ones. In order to reveal the possible sequences in the obtained maps, the order of neurons are rearranged by the mean spike time. The mean spike time is computed for each neuron at the map by averaging times multiplied by probability of spike. In other words, for each neuron n, all time steps t in interval $(0, 40)$ are multiplied by $MAP(n, t)$ to form the array ϑ or simply $(\vartheta(t) = t \times MAP(n, t))$. Then the mean of this array is calculated for mean spike time $MST = \frac{1}{40}\sum_t \vartheta(t)$. All neurons are then sorted by their MST in the obtained maps. The comparison of the MAPs from spontaneous dataset and stimulus dataset can show the common structure of two datasets.

8.4.3 Discussion

We compare the selected MAPs from both datasets by comparing the normalized cross-correlation (NCC) between the maps. After reordering the neurons by mean spike time, the sequential pattern in the maps are expected to be detectable. Hence, the max of NCC between each pair of maps from the datasets is a good measure of similarity of sequential activities. The histogram of obtained similarity between the maps is in Figure 8.13. The figure shows that majority of maps have over 70 percent correlation which shows that they share common sequences. In other words, since overwhelming majority of maps have a high percentage of matching, there are common patterns that exist between stimulus-driven activities and spontaneous spikes.

8.5 Summary

In this chapter, the proposed spike pattern detection methods were evaluated on artificial datasets. The noise, jitter and number neurons of dataset were manipulated to test the performance of algorithms. The CRBM model demonstrate the most stable accuracy against changes in the parameters. Moreover, the CRBM were applied on the empirical multi-cell recording of rats. We showed that there are common structures in spontaneous and stimuli driven dataset.

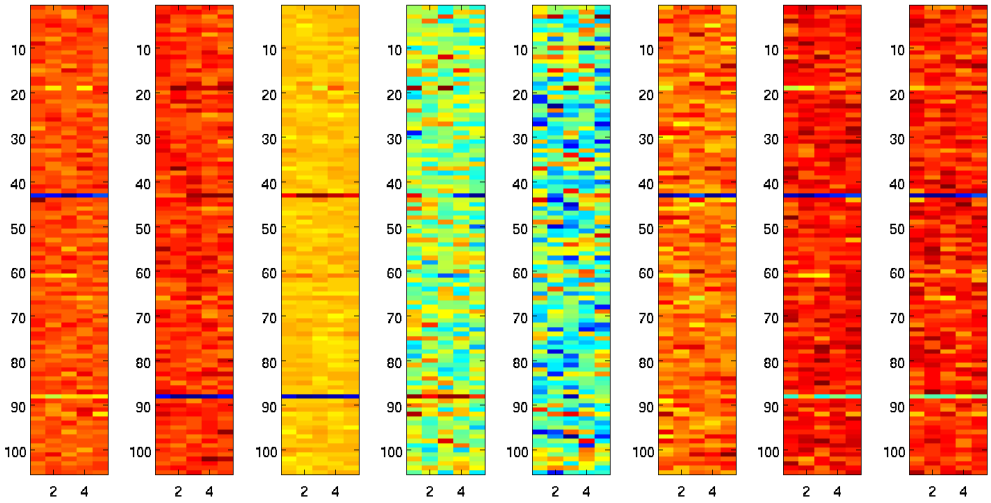


Figure 8.12: The obtained filter by CRBM on Multi-cell recording of stimulus-driven activities dataset. The order of neurons are NOT rearranged by means spike time.

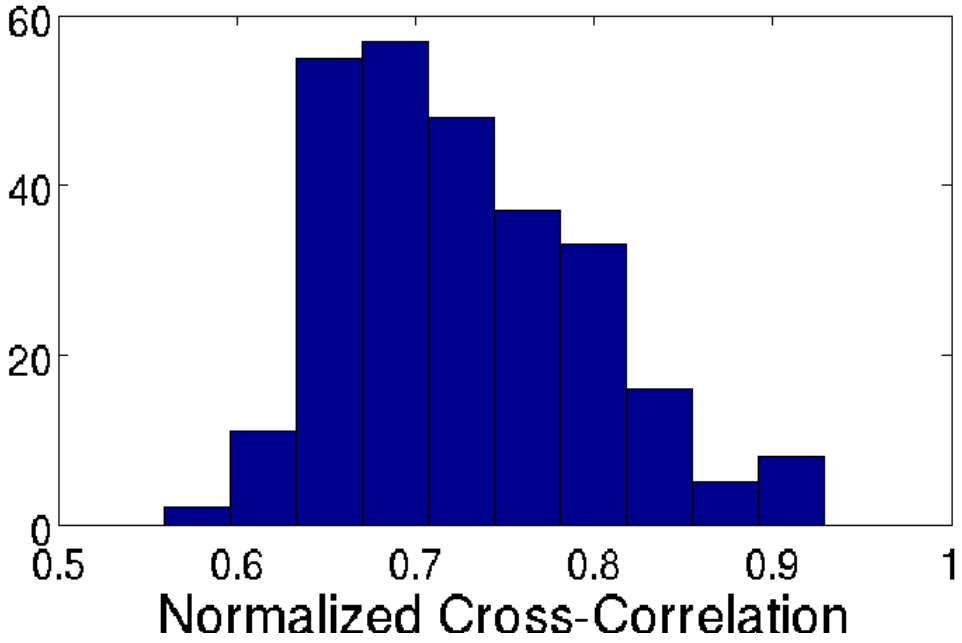


Figure 8.13: The histogram of Normalized Cross-Correlation between the selected MAPs from spontaneous dataset and stimulus-locked dataset. The NCC were computed between each pair of selected MAPs.

Chapter 9

Conclusion and Future Studies

9.1 Conclusion

This thesis presented a search algorithm that was able to detect patterns of temporal binary data. The method was very simple and it was experimentally shown that it is able to detect spatiotemporal of patterns with low variation over time. In other words, if patterns occur with a fixed form over time, this algorithm is able to detect it. Moreover, we presented two unsupervised learning approaches for detection of temporal patterns. Initially, we evaluated our methods on image dataset and it was demonstrated that sparsity regularization help learning distinct filters. Furthermore, the log-likelihood estimation of convolutional RBM revealed that this method is able to learn a good generative model with small number of parameters.

The proposed unsupervised methods were successful in detection of patterns from artificial temporal data even with high variability of structures. We showed that convolutional RBM has a superior performance over other unsupervised approach in terms of accuracy of detection. Moreover, it was experimentally shown that the convolutional RBM exhibit a relatively better performance against extreme variability of patterns in addition to extreme noise. All in all, the good accuracy of CRBM, encouraged us to use this model for empirical brain records. By applying this method on activities evoked by stimulus to the animal and also activities without being provoked by any external source, it was shown that there are common structural patterns in these activities. The method was able to detect the sequential patterns of empirical multi-cell recordings.

9.2 Future Studies

There are many open challenges for this task. So far we have tested our algorithms on limited datasets and we have shown a basic demonstration for empirical multi-cell

recordings. The main future study will be more experimenst on various datasets. Datasets of recordings from different brain regions and with different experimental setups. It is crucial to demonstrate that the proposed methods actually work in practice for discovering spike patterns. In addition, another important issue is that after training the unsupervised methods on multi-cell recordings, it is not easy to interpret the learned filters and the obtained maps. Further researches are required for devising new ways of evaluating obtained filters in order to discover the structures. Moreover, It is reasonable to assume existence of hierarchal structures in patterns of the multi-cell recordings. Hence, we would need further researches about using unsupervised hierarchical representation learning for detecting patterns that have hierarchical structure. In other words, more researches should be done in the area of deep unsupervised learning methods to discover hierarchical patterns of multi-cell recordings.

Appendix A

The Free Energy of Convolutional Restricted Boltzmann Machine

A.1 Preliminary

Before proving the formula of free energy of CRBM, we need to prove the following statement:

$$\sum_{x}\prod_{j} f(x_j) = \prod_{j}\sum_{x_j} f(x_j) \tag{A.1}$$

where x is a binary valued n dimensional array and f is a function. The left hand side sum is over all possible values of array x. We start from the left side by setting only the first index x_0 to its possible values:

$$\begin{aligned}\sum_{x,x_0=0}\left(f(0)\prod_{j=1}^{n} f(x_j)\right) + \sum_{x,x_0=1}\left(f(1)\prod_{j=1}^{n} f(x_j)\right) \\ = \sum_{x}\left(f(0)+f(1)\right)\prod_{j=1}^{n} f(x_j)\end{aligned} \tag{A.2}$$

If we continue with this process for all other indexes of x, we will get:

$$\left(f(0)+f(1)\right)\left(f(0)+f(1)\right)...\left(f(0)+f(1)\right) = \prod_{j}\sum_{x_j} f(x_j) \tag{A.3}$$

Therefore, we reached to the right hand side and the statement is proved. A special case of function could be $f(t) = e^t$. Then the equation A.1 will become:

$$\sum_{x}\prod_{j} f(x_j) = \prod_{j}\left(1+e^{x_j}\right) \tag{A.4}$$

A.2 The Free Energy

Let's obtain the equation of the free energy of CRBM. By definition we know that:

$$F(v) = -\log\sum_{h} e^{-E(v,h)} \tag{A.5}$$

and the energy is:

$$E(v,h) = -\sum_{k=1}^{K} h^k \bullet (\widetilde{W}^k * v) - \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^k - c \sum_{i,j} v_{i,j} \tag{A.6}$$

Therefore $F(v)$ is:

$$F(v) = -\log \sum_{h} exp\left(\sum_{k=1}^{K} h^k \bullet (\widetilde{W}^k * v) + \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^k + c \sum_{i,j} v_{i,j}\right) \tag{A.7}$$

we can take out the $c\sum_{i,j} v_{i,j}$ which is not dependent on the h and use the definition of operation $\bullet$:

$$\begin{aligned} F(v) &= -c \sum_{i,j} v_{i,j} - \log \sum_{h} exp\left(\sum_{k=1}^{K} \sum_{ij} \left((\widetilde{W}^k * v)_{ij} + b_k\right) h_{ij}^k\right) \\ &= -c \sum_{i,j} v_{i,j} - \log \sum_{h} \prod_{k=1}^{K} exp\left(\sum_{ij} \left((\widetilde{W}^k * v)_{ij} + b_k\right) h_{ij}^k\right) \end{aligned} \tag{A.8}$$

Using the proved statement A.1 and the special case A.4, we can get:

$$F(v) = -c \sum_{i,j} v_{i,j} - \log \prod_{k=1}^{K} \left(1 + exp\left(\sum_{ij} \left((\widetilde{W}^k * v)_{ij} + b_k\right)\right)\right) \tag{A.9}$$

By applying the log, we will get the final equation:

$$F(v) = -c \sum_{i,j} v_{i,j} - \sum_{k=1}^{K} \log \left(1 + exp\left(\sum_{ij} \left((\widetilde{W}^k * v)_{ij} + b_k\right)\right)\right) \tag{A.10}$$

Appendix B

The Annealed Importance Sampling for CRBM

B.1 Intermediate Distribution

In this section, we show how the equation of intermediate distribution equation are derived for the AIS algorithm of CRBM. The definition of intermediate distribution is :

$$p_l^*(v) = \sum_{h^A,h^B} e^{(1-\beta_l)E(v,h^A)+\beta_l E(v,h^B)} \tag{B.1}$$

where $0 < \beta_0 < \beta_1 < ... < \beta_l < .. < \beta_L = 1$. The energy function of A is:

$$E_A(v,h) = \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^k + c^A \sum_{i,j} v_{i,j} \tag{B.2}$$

and energy of B is:

$$E_B(v,h) = \sum_{k=1}^{K} h^k \bullet (\widetilde{W}^k * v) + \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^k + c^B \sum_{i,j} v_{i,j} \tag{B.3}$$

We can take out the terms that are not dependent on h:

$$p_l^*(v) = exp(c^A \sum_{i,j} v_{i,j}) \sum_{h^A} exp\left((1-\beta_l) \sum_{k=1}^{K} b_{A,k} \sum_{i,j} h_{i,j}^{A,k}\right) \tag{B.4}$$

$$\times exp(c^B \sum_{i,j} v_{i,j}) \sum_{h^B} exp\left(\beta_l \sum_{k=1}^{K} b_{A,k} \sum_{i,j} h_{i,j}^{B,k}\right)$$

Because hidden vectors are binary, by using the same technique in the section A.1 we can get:

$$p_l^*(v) = exp(c^A \sum_{i,j} v_{i,j}) \prod_k 1 + exp\left((1-\beta_l) b_{A,k}\right) \tag{B.5}$$

$$\times exp(c^B \sum_{i,j} v_{i,j}) \prod_k 1 + exp(\beta_l b_{B,k})$$

B.2 Partition Function

In this section, we show the how the partition function of convolutional RBM with zero weight matrix can be computed. By definition we have:

$$Z = \sum_{v} \sum_{h} exp\left(- \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^{k} - c \sum_{i,j} v_{i,j} \right) \tag{B.6}$$

by using the same technique in the section A.1, we marginalize out h:

$$Z = \prod_{k} (1 + e^{N_h^2 b_k}) \sum_{v} exp(c \sum_{i,j} v_{i,j}) \tag{B.7}$$

where N_h is the number of hidden nodes in each hidden group. We marginalize v to get the partition function:

$$Z = \left(\prod_{k} (1 + e^{N_h^2 b_k}) \right) \times (1 + e^{c})^{N_v^2} \tag{B.8}$$

Appendix C

The Energy-based Model for Convolutional Denoising Auto-encoder

In this section, we explain how the energy based model of convolutional auto-encoder can be obtained. First we bring the proof for denoising auto-encoder from [68] then we expand it to convolutional denoising aut-encoder. The following necessary conditions should be hold for a basic denoising auto-encoder, in order to be proven that training criterion is equivalent to the score matching. First, the real valued input x should be corrupted by Gaussian additive noise ($\dot{x} = x + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$). Second, the objective should be mean squared error of input and reconstruction of corrupted input. Therefore the DAE should minimize the following objective function:

$$J_{DAE} = E_{q(x,\dot{x})}\left[\frac{1}{2}\|W^T sigmoid(W\dot{x} + b) + c - x\|^2\right] \tag{C.1}$$

where the $q(x, \dot{x})$ can be found in table C.1. For convolutional denoising auto-encoder, the objective function is:

$$J_{CDAE} = E_{q(x,\dot{x})}\left[\frac{1}{2}\left\|\sum_k \widetilde{W}^k * sigmoid(W^k * \dot{x} + b_k) + c - x\right\|^2\right] \tag{C.2}$$

In [68], it is considered that matching the score of Parzen density estimate $q_\sigma(\tilde{x})$ which is based on our corrupted example of the training set (Please see Table C.1), is equivalent to score matching of our dataset which has the objective in equation 6.19. Therefore the regular score matching objective J_{SM} is equivalent to score matching of Parzen density estimator J_{ESM}:

$$J_{SM} \equiv J_{ESM} = E_{q_\sigma(\dot{x})}\left[\frac{1}{2}\left\|\psi(\dot{x};\theta) - \frac{\partial \log q_\sigma(\dot{x})}{\partial \dot{x}}\right\|^2\right] \tag{C.3}$$

The J_{ESM} itself is equivalent to denoising score matching objective [68]:

$$J_{DSM} = E_{q_\sigma(x,\dot{x})}\left[\frac{1}{2}\left\|\psi(\dot{x};\theta) - \frac{\partial \log q_\sigma(\dot{x}|x)}{\partial \dot{x}}\right\|^2\right] \tag{C.4}$$

Table C.1: The notations and symbols.

$q(x)$	Unknown *true* pdf. $x \in R^d$
$D_n = \{x^{(1)}, ..., x^{(n)}\}$	Training Set: i.i.d. sample from q
$q_0(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(\|x - x^{(i)}\|)$	Empirical pdf associated with D_n
$q_\sigma(\dot{x}\|x) = \frac{1}{(2\pi)^{d/2}\sigma^d} e^{-\frac{1}{2\sigma^2}\|\dot{x}-x\|^2}$	noise model
$q_\sigma(\dot{x}, x) = q_\sigma(\dot{x}\|x)q_0(x)$	Joint pdf
$q_\sigma(\dot{x}) = \frac{1}{n}\sum_{i=1}^{n} q_\sigma(\dot{x}\|x^{(i)})$	Parzen density estimate based on D_n
$softplus(x) = \log(1 + e^x)$	softplus function
$softplus'(x) = sigmoid(x) = \frac{1}{1+e^{-x}}$	derivative of softplus

where:

$$\frac{\partial \log q_\sigma(\dot{x}|x)}{\partial \dot{x}} = \frac{1}{\sigma^2}(x - \dot{x}) \tag{C.5}$$

Given the above equivalences, we prove that training convolutional denoising auto-encoder with objective J_{DAE} is equivalent to score matching of following energy-based model:

$$p(x;\theta) = \frac{1}{Z(\theta)} exp(-E(x;\theta)), \tag{C.6}$$

$$E(x; W, b, c) = -\frac{1}{\sigma^2}(\sum_{ij}(cx + \sum_k softplus(W^k * x + b_k))_{ij} - \frac{1}{2}\|x\|^2)$$

The score of this model is:

$$\psi(x;\theta) = \frac{\partial \log p(x)}{\partial x} = -\frac{\partial E}{\partial x} \tag{C.7}$$

By taking the derivative:

$$\frac{-\partial E}{\partial x_{u,v}} = \frac{1}{\sigma^2}\left((c + \sum_k softplus'(W^k * x + b_k)\frac{\partial(W^k * x + b_k)}{\partial x} - x)_{u,v}\right) \tag{C.8}$$

$$= \frac{1}{\sigma^2}\left((c + \sum_k \widetilde{W}^k * sigmoid(W^k * x + b_k) - x)_{u,v}\right)$$

Which can be rewritten as:

$$\psi(x;\theta) = \frac{-\partial E}{\partial x} = \frac{1}{\sigma^2}\left((c + \sum_k \widetilde{W}^k * sigmoid(W^k * x + b_k) - x)\right) \tag{C.9}$$

Substituting this score in equation C.4:

$$J_{DSM} = E_{q_\sigma(x,\dot{x})}\left[\frac{1}{2}\left\|\psi(\dot{x};\theta) - \frac{\partial \log q_\sigma(\dot{x}|x)}{\partial \dot{x}}\right\|^2\right] \tag{C.10}$$

$$= E_{q_\sigma(x,\dot{x})}\left[\frac{1}{2}\left\|\frac{1}{\sigma^2}\left((c + \sum_k \widetilde{W}^k * sigmoid(W^k\dot{x} + b_k) - x\right) - \frac{1}{\sigma^2}(x - \dot{x})\right\|^2\right]$$

$$= E_{q_\sigma(x,\dot{x})}\left[\frac{1}{2\sigma^4}\left\|\sum_k \widetilde{W}^k * sigmoid(W^k\dot{x} + b_k) + c - x\right\|^2\right] = \frac{1}{2\sigma^4}J_{CDAE}$$

Thus we shown that training convolutional denoising auto-encoder with objective J_{CDAE} is equivalent to score matching objective J_{DSM}. Putting all together:

$$J_{SM} \equiv J_{ESM} \equiv J_{DSM} \equiv J_{CDAE} \tag{C.11}$$

Bibliography

[1] Amr Ahmed, Kai Yu, Wei Xu, Yihong Gong, and Eric Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *Computer Vision–ECCV 2008*, pages 69–82. Springer, 2008.

[2] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Spatio-temporal convolutional sparse auto-encoder for sequence classification. In *BMVC*, pages 1–12, 2012.

[3] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[7] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5), 2007.

[8] Y-lan Boureau, Yann L Cun, et al. Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, pages 1185–1192, 2008.

[9] Stacey L Brown, Joby Joseph, and Mark Stopfer. Encoding a temporally structured stimulus with a temporally structured neural representation. *Nature neuroscience*, 8(11):1568–1576, 2005.

[10] Lars Buesing, Timothy A Machado, John P Cunningham, and Liam Paninski. Clustered factor analysis of multineuronal spike data. In *Advances in Neural Information Processing Systems*, pages 3500–3508, 2014.

[11] M Yu Byron, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In *Advances in neural information processing systems*, pages 1881–1888, 2009.

[12] Binu P Chacko, VR Vimal Krishnan, G Raju, and P Babu Anto. Handwritten character recognition using wavelet energy and extreme learning machine. *International Journal of Machine Learning and Cybernetics*, 3(2):149–161, 2012.

[13] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. *arXiv preprint arXiv:1506.03607*, 2015.

[14] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew Y Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 440–445. IEEE, 2011.

[15] Christos Davatzikos, Kosha Ruparel, Yong Fan, DG Shen, M Acharyya, JW Loughead, RC Gur, and Daniel D Langleben. Classifying spatial patterns of brain activity with machine learning methods: application to lie detection. *Neuroimage*, 28(3):663–668, 2005.

[16] Kamran Diba and György Buzsáki. Forward and reverse hippocampal place-cell sequences during ripples. *Nature neuroscience*, 10(10):1241–1242, 2007.

[17] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on artificial intelligence and statistics*, pages 153–160, 2009.

[18] Orhan Firat, Emre Aksan, Ilke Oztekin, and Fatos T Yarman Vural. Learning deep temporal representations for brain decoding. *arXiv preprint arXiv:1412.7522*, 2014.

[19] David J Foster and Matthew A Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680–683, 2006.

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

[21] Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013.

[22] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552, 2009.

[23] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[24] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deepspeech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[25] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[26] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

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

[28] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[29] Geoffrey E Hinton and Terrance J Sejnowski. Learning and relearning in boltzmann machines. *MIT Press, Cambridge, Mass*, 1:282–317, 1986.

[30] R Devon Hjelm, Vince D Calhoun, Ruslan Salakhutdinov, Elena A Allen, Tulay Adali, and Sergey M Plis. Restricted boltzmann machines for neuroimaging: an application in identifying intrinsic networks. *NeuroImage*, 96:245–260, 2014.

[31] Daoyun Ji and Matthew A Wilson. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature neuroscience*, 10(1):100–107, 2007.

[32] Lauren M Jones, Alfredo Fontanini, Brian F Sadacca, Paul Miller, and Donald B Katz. Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proceedings of the National Academy of Sciences*, 104(47):18772–18777, 2007.

[33] Mike Kayser and Victor Zhong. Denoising convolutional autoencoders for noisy speech recognition.

[34] Tal Kenet, Dmitri Bibitchkov, Misha Tsodyks, Amiram Grinvald, and Amos Arieli. Spontaneously emerging cortical representations of visual attributes. *Nature*, 425(6961):954–956, 2003.

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

[36] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

[37] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1:0, 2006.

[38] Honglak Lee. *Unsupervised feature learning via sparse hierarchical representations*. Stanford University, 2010.

[39] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[40] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[41] Artur Luczak, Peter Barthó, and Kenneth D Harris. Spontaneous events outline the realm of possible sensory responses in neocortical populations. *Neuron*, 62(3):413–425, 2009.

[42] Artur Luczak, Peter Bartho, and Kenneth D Harris. Gating of sensory input by spontaneous cortical activity. *The Journal of Neuroscience*, 33(4):1684–1695, 2013.

[43] Artur Luczak, Peter Barthó, Stephan L Marguet, György Buzsáki, and Kenneth D Harris. Sequential structure of neocortical spontaneous activity in vivo. *Proceedings of the National Academy of Sciences*, 104(1):347–352, 2007.

[44] Jakob H Macke, Lars Buesing, John P Cunningham, M Yu Byron, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In *Advances in neural information processing systems*, pages 1350–1358, 2011.

[45] Alireza Makhzani and Brendan Frey. k-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[46] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59. Springer, 2011.

[47] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 593–600. ACM, 2005.

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

[49] A-R Mohamed, Tara N Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E Hinton, and Michael A Picheny. Deep belief networks using discriminative features for phone recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5060–5063. IEEE, 2011.

[50] Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[51] John O'keefe and Lynn Nadel. *The hippocampus as a cognitive map*, volume 3. Clarendon Press Oxford, 1978.

[52] Dean Pomerleau and Todd Jochem. Rapidly adapting machine vision for automated vehicle steering. *IEEE Intelligent Systems*, (2):19–27, 1996.

[53] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

[54] M Ranzato, Fu Jie Huang, Y-L Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[55] Marc'Aurelio Ranzato and Martin Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799. ACM, 2008.

[56] Sheikh Faisal Rashid, Marc-Peter Schambach, Jörg Rottland, and Stephan von der Nüll. Low resolution arabic recognition with multidimensional recurrent neural networks. In *Proceedings of the 4th International Workshop on Multilingual OCR*, page 6. ACM, 2013.

[57] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011.

[58] Yasser Roudi, Benjamin Dunn, and John Hertz. Multi-neuronal activity and functional connectivity in cell assemblies. *Current opinion in neurobiology*, 32:38–44, 2015.

[59] Aruni RoyChowdhury, Tsung-Yu Lin, Subhransu Maji, and Erik Learned-Miller. Face identification with bilinear cnns. *arXiv preprint arXiv:1506.01342*, 2015.

[60] Devendra Singh Sachan, Umesh Tekwani, and Amit Sethi. Sports video classification from multimodal information using deep neural networks. In *2013 AAAI Fall Symposium Series*, 2013.

[61] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[62] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[63] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879. ACM, 2008.

[64] William E Skaggs and Bruce L McNaughton. Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science*, 271(5257):1870–1873, 1996.

[65] John Skilling. Nested sampling. *Bayesian inference and maximum entropy methods in science and engineering*, 735:395–405, 2004.

[66] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2006.

[67] George Tzanis, Christos Berberidis, and Ioannis P Vlahavas. Biological data mining., 2005.

[68] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

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

[70] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[71] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[72] Cha Zhang and Zhengyou Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 1036–1041. IEEE, 2014.

[73] Will Zou, Shenghuo Zhu, Kai Yu, and Andrew Y Ng. Deep learning of invariant features via simulated fixations in video. In *Advances in neural information processing systems*, pages 3212–3220, 2012.