Graz, University of Technology

Institute for Theoretical Computer Science (IGI),

Graz University of Technology

A-8010 Graz, Austria

Master's Thesis

Novel Methods for Probabilistic Inference and Learning in Spiking Neural Networks

Submitted by:

Stefan Habenschuss

Supervisor:

O.Univ.-Prof. Dipl.-Ing. Dr.rer.nat. Wolfgang Maass

Graz University of Technology, Austria

Co-Supervisor:

Dr. Shih-Chii Liu

University and ETH Zurich, Switzerland

October 14, 2010

TECHNISCHE UNIVERSITÄT GRAZ

INSTITUT FÜR GRUNDLAGEN DER INFORMATIONSVERARBEITUNG (IGI),

TECHNISCHE UNIVERSITÄT GRAZ

A-8010 GRAZ

MASTERARBEIT

Neue Methoden für probabilistische Inferenz und Lernen in Spikenden Neuronalen Netzen

Vorgelegt von:

Stefan HABENSCHUSS

Betreuer:

O.Univ.-Prof. Dipl.-Ing. Dr.rer.nat. Wolfgang Maass

Technische Universität Graz, Österreich

Zweitbetreuerin:

Dr. Shih-Chii Liu

Universität und ETH Zürich, Schweiz

14. Oktober 2010

Abstract

Spiking neural networks constitute the third generation of artificial neural networks. They differ from traditional neural network models by respecting biological networks in greater detail and mapping their behavior more accurately. One of the main research goals in the field of spiking neural networks is to elucidate connections between structure and function. Recent research in cognitive psychology and neuroscience points to an instrumental role of probabilistic reasoning and learning in human and animal behavior. The objective of this thesis was to investigate the ability of certain architectures of spiking neural networks to perform probabilistic inference and learning, either exact or approximate. To this end, a recently proposed architecture which has established a link between the Expectation-Maximization (EM) algorithm and neural mechanisms was extended and generalized. A rigorous theory was developed which relates activity and plasticity in a special class of spiking neural networks to an online approximation of EM. This was achieved by mapping a probabilistic mixture model with component distributions from exponential families to a neural architecture. Inference and learning were demonstrated to correspond to neural integration with lateral inhibition and Hebbian plasticity, or more precisely Spike-Timing Dependent Plasticity (STDP). The method proposed in this thesis is consistent with biological data in important aspects and complements recent discoveries in the

area, by providing a theoretical explanation of the emergence of a particular function in certain classes of spiking neural networks.

Kurzfassung

Spikende neuronale Netze stellen die dritte Generation von neuronalen Netzen dar. Sie unterscheiden von herkömmlichen Modellen neuronaler Netze durch eine höhere Detailtreue in der Repräsentation von biologischen Netzwerken. Ein besonders interessanter Aspekt bei der Untersuchung von spikenden neuronalen Netzen besteht im Aufzeigen von Zusammenhängen zwischen Struktur und Funktion. Aktuelle Forschungsergebnisse aus kognitiver Psychologie und den Neurowissenschaften deuten darauf hin, dass probabilistische Inferenz und Lernen eine entscheidende Rolle im Verhalten von Menschen und Tieren spielen. Das Ziel dieser Arbeit war die Untersuchung bestimmter Architekturen von spikenden neuronalen Netzen hinsichtlich ihrer Fähigkeit, probabilistische Inferenz und Lernen, entweder exakt oder approximativ, zu implementieren. Zu diesem Zweck wurde eine kürzlich vorgeschlagene Architektur erweitert und verallgemeinert, in welcher neuronale Mechanismen in Verbindung mit dem Expectation-Maximization (EM) Algorithmus gebracht wurden. Es wurde eine Theorie für eine bestimmte Klasse von spikenden neuronalen Netzen entwickelt, welche neuronale Aktivität und Plastizität mit einer online Approximation des EM Algorithms in Verbindung setzt. Dies wurde durch die Projektion einer probabilistischen Mischverteilung mit Komponentenverteilungen aus Exponentialfamilien auf eine neuronale Architektur erreicht. Es konnte gezeigt werden, dass Inferenz und Lernen in solchen Modellen neuronaler Integration und lateraler Inhibition, beziehungsweise Hebb'scher Plastizität oder genauer, Spike-Timing Dependent Plasticity (STDP), entspricht. Die in dieser Arbeit präsentierte Methode ist mit biologischen Daten in wesentlichen Aspekten konsistent und ergänzt jüngste Entwicklungen dieses Gebiets durch eine neuen theoretische Grundlage, die erklärt wie eine konkrete Funktion in bestimmten Klassen von spikenden neuronalen Netzen entstehen kann.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....	....
(Place, Date)	(Signature)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

....	....
(Ort, Datum)	(Unterschrift)

Contents

List of Figures

List of Tables

Acknowledgements

During the project which led to the writing of this thesis, I was given the opportunity to spend six exciting and insightful months at the Institute of Neuroinformatics in Zurich. For making that possible, I would like to express my deep gratitude to my supervisors, Wolfgang Maass and Shih-Chii Liu. Without their guidance and support, this thesis would not have been possible.

It is a pleasure to thank my colleague and friend Helmut Puhr for the countless inspiring discussions, in which many ideas were born which have made it into this thesis.

I owe my deepest gratitude to my parents, Gabriela and Wolfgang for their unconditional support and patience throughout my time as a student.

Last but not least, I would like to thank my family and friends for distracting me when I most needed it, and for reminding me of other facets of life than work.

Stefan Habenschuss
Graz, Austria, October 14, 2010

Chapter 1

Introduction

What is the secret behind human cognition? Is there a physical substrate for memory, abstract thought or planning? Shall we contemplate that neurons and synapses are responsible for complex perception and behavior? Will an understanding of the biophysical processes, which take place in the brain, shed light on the miracle of human intelligence?

Questions like these have engaged ancient philosophers and modern scientists alike for quite some time (Morton [1997]; Purves [2008]; Carlson and Braun [2002]). The 20th century witnessed many new approaches to these issues. Perhaps most important, the concept of artificial neural networks was born and has rapidly occupied a key position in fields like artificial intelligence and machine learning over the past decades (Haykin [1994]; Bishop [2006]). Just before the turn of the century, models of spiking neural networks entered the stage: the third generation of neural networks (Maass and Bishop [2001]). In contrast to their predecessors, they capture one of the most significant peculiarities of communication in biological neurons. When a biological neuron transmits information, it sends out spikes, small pulses of activity traveling from one neuron to another. In between neurons, the pulses pass through synapses which act as gates and are able to modulate the strength and impact of a spike.

Models of spiking neural networks have reached a previously unmatched level of detail in describing the intricate processes which underlie communication and computation in real biological networks. Many central questions in the study of spiking neural networks are about function (Maass and Bishop [2001]). How powerful is a network of neurons and synapses in computational terms? Are there certain structures and architectures which can be associated with certain functions? Eventually, one would like to know whether and how high-level functions such as abstract thought and language comprehension may be embedded in neural circuitry. For the time being, the answers to these high-level questions remain out of reach. There are, however, some intermediate-level features of human intelligence which seem promising to study. Pattern learning and recognition as well as reasoning in the face of uncertainty are examples of features which offer themselves to analysis as they have been studied extensively in non-spiking networks (Bishop [2006]).

This thesis is about associating a certain class of spiking neural networks with a particular function, namely probabilistic learning, or more precisely density estimation, and inference. Ideally, learning in the sense of density estimation refers to an agent's ability to build an internal model of the world while observing it. An ideal model should comprise a full-fledged description containing raw sensory inputs, intermediate concepts like objects which summarize sensory inputs, and high-level concepts like abstract categories. Furthermore, it should capture dependencies and causal relationships among sensory inputs, objects and concepts, and represent the basic rules, according to which elements in the environment, including the agent itself, interact and evolve in time. Inference in an ideal model would allow to optimally exploit the acquired knowledge in any given situation. This includes interpreting current sensory inputs, drawing conclusions about the state of the world, or even predicting the future (MacKay [2003]; Bishop [2006]). Clearly, understanding how networks of spiking neurons could implement probabilistic models which can deal with real-world complexity is a long-term objective. At the moment, this objective can only serve as an inspiration to more modest endeavors.

Recent work by Deneve [2008a] or Nessler et al. [2010] indicates the complexity of probabilistic models which are currently within reach for spiking neural networks[1]. In the latter publication, the authors consider a mixture model of categorical variables and the mapping of this model to the neural domain. Both learning and inference are successfully associated with standard neural mechanisms. Due to its attractive simplicity, the model proposed by Nessler et al. [2010] will serve as a starting point for the original part of this thesis. Being a relatively concise model, it has a number of weaknesses which limit its value from a technical perspective and diminish its plausibility in a biological context. These weaknesses naturally motivate the adjustments and extensions proposed in this work.

The first weakness is about how the input must be delivered to the model. The authors must assume that there exists a so-called population coding of input variables. The inputs must be provided in groups and in each group there can be only one active input at a time. The second weakness concerns how an input spike influences the state of an internal model neuron. The temporal evolution of this influence is called excitatory post-synaptic potential (EPSP, see Purves [2008]). In biology, EPSPs decay smoothly over time, whereas the proposed model must assume that this influence stops abruptly after a fixed window of constant influence. Therefore, the second weakness is about the EPSP shape. In this work, these two shortcomings will be addressed by considering mixture models with component distributions which differ from the original approach.

The remainder of this thesis is structured as follows. Chapter 2 introduces spiking neural networks. Chapter 3 gives an introduction to probabilistic models. Chapter 4 covers existing work of immediate relevance to this thesis. Chapter 5 presents a generalization of the model from Nessler et al. [2010] which deals with the issue of assuming input population coding. In Chapter 6, it is shown how more realistic EPSP shapes can be incorporated in the model. In addition, two further extensions of the method are presented. Finally, Chapter 7 gives a conclusion and a brief outlook. Appendix A contains a convergence proof of the proposed method.

[1]At least when mechanisms for both inference and learning should be implemented by the same network.

Chapter 2

Spiking Neural Networks

Neural networks are a fascinating, and deeply enigmatic field of study. In their relentless pursuit of knowledge and understanding, modern biology, neuroscience and cognitive psychology have left little doubt that the miraculous process by which we humans perceive, think and act, comes about through the collaboration of myriads of highly specialized brain cells, most importantly neurons (see for example Carlson and Braun [2002]). Densely connected via electrical or chemical synapses, organized in areas, local networks and columns and embedded in supporting tissue, they are thought to lend us spirit and reason. Picturing such an intricate and elusive process at work is clearly beyond our imaginative capabilities, albeit the inspiring words once conceived by Charles S. Sherrington, a pioneer of modern neuroscience, to express his wonder and excitement at the human brain (Sherrington [1941]):

> *"The great topmost sheet of the mass, that where hardly a light had twinkled or moved, becomes now a sparkling field of rhythmic flashing points with trains of traveling sparks hurrying hither and thither. The brain is waking and with it the mind is returning. It is as if the Milky Way entered upon some cosmic dance. Swiftly the head mass becomes an enchanted loom where millions of flashing shuttles weave a dissolving pattern, always a meaningful pattern though never an abiding one; a shifting harmony of*

sub-patterns. ”

Today, nearly sixty years after Sherrington’s groundbreaking spadework, the neuroscientific community may look back on quite a series of important discoveries which have greatly shaped, and sometimes permanently changed our understanding of the most complex of organs we possess. Many fields of study have grown a close relation to neuroscience, sharing an unresting and longanimous eagerness to unravel the brain’s mystery. While western philosophers have been pondering on the mind and its relation to the body ever since the dawn of ancient Greek philosophy (Morton [1997]), and psychologists have long provided a phenomenological perspective of our mental functions and behaviors, more recently the formulation of the Turing machine (Turing [1937]) and the advent of computers in the twentieth century have added yet another delicate twist to this longstanding endeavor[1].

The continuing scientific diversification has given birth to fields like computational neuroscience, artificial intelligence and machine learning. Each has its own focus, pursuing a slightly different goal. Yet, there is one particularly intriguing question of interest to all related disciplines: If the brain’s function, nature’s most ingenious piece of art, can be fully understood in terms of neurons and synapses, can we someday build an artificial brain just as easily as we can build today planes, cameras and microphones, inspired by birds, human sight and hearing?

In fact there are three separate questions hidden in this formulation:

Conceptual question: Can the brain’s function be fully understood as a network of neurons and synapses? This is a highly controversial topic (see for example Churchland and Sejnowski [1994]). In a more careful formulation, the question should probably run: Is there some level of detail of description which suffices for a complete account of the brain’s function? While many researchers seem to agree on that, our knowledge is too limited for a certain answer. Some go as far as to

[1]It is remarkable that the dawn of computers initiated by Turing’s work, was motivated by an attempt to “build a brain”, as Turing himself expressed his intentions (the interested reader is referred to Hodges [2007]).

speculate that effects on a level as small as quantum mechanics might be responsible for what we perceive as higher brain function, including consciousness and will (Koch and Hepp [2006]). On the other hand, experience from man copying nature might suggest that an overwhelming body of details can be omitted in an abstraction step as long as a few underlying principles are implemented correctly. Even if we can motivate ourselves with such an assumption, it only moves the problem towards identifying which parts of the entire structure and process are central to its function. Which pieces of the puzzle can be ascribed to deeper principles of intelligence, and which parts are implementation details which we might be able to neglect as they only reflect evolutionary design choices due to physical or other constraints that do not apply in a technical engineering setting?

Measurement question: Can all functionally relevant facts be reverse engineered on the required level of detail? This issue strongly depends on technological progress in neuroimaging (see Filler [2009] or Monchi et al. [2008]). Both structural and functional imaging are fast developing branches, making a prognosis difficult. The emerging field of optogenetics (see Miesenbock [2009]) as well as clever combinations of existing techniques like Magnetic Resonance Imaging (MRI) and Positron Emission Tomography (PET) (see Judenhofer et al. [2008]) hold great promise for the future. Still, the temporal and spatial resolution required to obtain a thorough picture of the human brain remain out of reach.

Engineering question: Having gathered all required knowledge, can an artificially intelligent system be built, fast and power-efficient enough to meet our needs? While Moore's law (Moore [1965]) is thought to continue to hold for the near future, power efficiency is more and more emerging as a real obstacle in this matter (see for example Markram [2006]). Current processor technology is tailored to optimize performance versus space requirements. Even supercomputers based on current high-end architectures like Blue Gene/P claiming an unmatched performance/power consumption trade-off, spend 2.9 Mio. Watt/Petaflops (see I.B.M.B.G. Team [2008]), as opposed to approximately 1 Watt/Petaflops in the human brain[2].

Identifying the principles of human intelligence and measuring the brain's activity at the required level of detail go hand in hand. In neuroscience, there is a constant demand for modeling observed phenomena and ascribing functional meaning to them. Conversely, modelers need biological data for inspiration, extraction of principles and testing of predictions. There is also a growing interest in collaboration between engineering and modeling labs, providing exclusive mutual benefits for both sides (see for example, Meier [2005]). In this spirit, the following sections will briefly introduce some basic concepts of Spiking Neural Networks, intending to provide a peek at all three perspectives, namely modeling, neuroscience and engineering.

2.1 Motivation for Investigating Spikes

Artificial neural networks (ANNs) are mathematical models which emulate the behavior of biological neural networks in an abstracted fashion. The history of ANNs can be traced back to the introduction of the perceptron, mapping analog inputs to binary outputs through a weighted sum and a threshold gate (Rosenblatt [1958]). A step towards more sophisticated models was to allow for analog output values by replacing the threshold gate by a sigmoid activation function. The resulting class of ANNs, in the following referred to as traditional ANNs, continues to be of great importance in countless applications up to the present day (Haykin [1994]; Bishop [2006]). Functionally, traditional ANNs can be understood as defining a mapping between a number of analog input and output ports. The network computes its output as a function of the input and a set of parameters, called weights. The input-output mapping can be seen as an atomic step, updating all neurons in a pre-specified order that propagates information from input to output in a

[2]Based on an estimated 10+ Petaflops (Narayan [2009]) and 20 Watt power consumption (Drubach [1999]) in the human brain.

single update wave. Traditionally, time is not explicitly incorporated in the model. Instead, each invocation of the network on a different input pattern can be considered a single discrete time step. Unfortunately, ANNs are quite at odds with neuroscientific findings regarding the way neurons communicate in biological networks.

While traditional ANNs have been successfully employed as a tool in a number of difficult machine learning problems and as such have become a field of study in their own right, they fail to account for many observations made in real neurons and synapses, most importantly:

- Biological neurons communicate via spikes, short pulses of activity of identical magnitude. The influence of one neuron's spike on another neuron is characteristically time-dependent (Purves [2008]). The significance of synchronicity in firing patterns of biological neuron ensembles is a natural consequence. ANNs cannot capture these effects since they assume that communication takes place via analog messages, as opposed to spikes.

- Recent research has consolidated that exact spike timings cannot be dismissed as a technical artifact of spiking transmission. Conflicting with the assumption behind ANNs that exact timing can be ignored as an epiphenomenon, information in observed spike patterns seems to be encoded not only in the frequency of spiking but also the exact spike timing relative to other events (see for example, Petersen et al. [2001])

- The plasticity of synapses, that is their ability to adjust their transmission strength, has been found to be highly sensitive to the exact timing of spikes (see Section 2.3.2).

The shortcomings of traditional ANNs to model these phenomena (among many others) render them quite impractical in the attempt of understanding neural communication at any reasonable level. This has led many researches to consider new types of neural network models which explicitly incorporate continuous time and biologically more realistic communication protocols to overcome the aforementioned limitations.

Most notably, Spiking Neural Networks, sometimes also called Pulsed Neural Networks, have emerged as a significant improvement over traditional ANNs regarding proximity to biological findings and the ability to produce useful testable predictions (see Maass [1997] for a good introduction; a more thorough treatment is given in Maass and Bishop [2001]). Apart from operating closer to biological reality, the use of a spiking communication protocol has also been shown to bring unique benefits from a computational perspective (see Maass [2004]).

2.2 Single Neuron Modeling

A common systematic approach to understanding a complex system is to take it apart and analyze each part individually. The human brain, with its estimated 50-100 billion neurons and 1000 trillion synapses (Murre and Sturdy [1995]) is undoubtedly the most complex organized system we can ever hope to have access to. It is hardly surprising that single neuron (and synapse) models play such an important role in the neuroscientific community, as they must provide the solid foundation upon which more complex network models, involving an increasing number of elements, can build.

In the following, the two most important models will be briefly discussed. A more detailed account of spiking neuron models can be found in (Gerstner and Kistler [2002]).

2.2.1 Leaky integrate-and-fire (LIF)

The Leaky integrate-and-fire neuron model consists of three basic elements:

Integration: The membrane potential, a variable which captures the state of the neuron, accumulates incoming spikes by summing up their weighted contributions.

Leakage: The membrane potential is subject to a constant leakage. In the absence of input it slowly decays to zero.

Firing: Each time an incoming spike causes the membrane potential to exceed a fixed threshold, an output spike is emitted and the membrane potential is reset to zero.

The model is based on a simple electrical RC-circuit with an additional threshold mechanism. Formally, integration and leakage can be expressed in a single equation:

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -u(t) + RI(t). \tag{2.1}$$

u denotes the membrane potential, τ_m is the time constant of decay, and $I(t)$ collects the contribution of all weighted inputs at a specific time. The simplicity of the model often makes it the first choice in network simulations. However, it should be noted that the LIF model is inferior to many alternatives if biological realism is of importance (Izhikevich [2004]).

Since the LIF model is a linear filter during sub-threshold operation, one can study the effect of single spikes on the membrane potential in terms of the impulse response of the filter. It is easy to verify that the impulse response of a LIF neuron is an exponentially decaying function of time. This also corresponds to the observation of decaying excitatory post-synaptic potentials (EPSPs) in biological neurons (Purves [2008]).

2.2.2 Spike Response Model (SRM)

The Spike Response Model (Gerstner [2001]) is a generalization of the Leaky integrate-and-fire model. In contrast to the differential equation governing the membrane potential in Eq. 2.1, here the influence of input $I(t)$ (generated by spikes or injected current) on the membrane potential is explicitly characterized by a linear filtering operation,

$$u(t) = \eta(t - \hat{t}) + \int_0^\infty \kappa(t - \hat{t}, s) I(t - s) ds.$$

The kernel $\kappa(t - \hat{t}, s)$ captures the influence of a single input pulse (for example a spike) on $u(t)$. Note that the kernel allows for a dependency on the last output spike time $\hat{t}$. Using this mechanism the model can account for neural refractoriness by dampening or completely erasing input effects after an output spike was emitted, thereby enforcing refractory periods between output spikes. The $\eta(t-\hat{t})$ kernel incorporates a complete description of the action-potential which is triggered each time when the membrane potential exceeds the variable threshold $\theta(t-\hat{t})$ which may also depend on the last spike-time.

The SRM is a very attractive model as it provides a great deal of flexibility through its adjustable functions while having a very simple form. The free parameters, namely the functions $\kappa(t-\hat{t}, s)$, $\eta(t-\hat{t})$ and $\theta(t-\hat{t})$, can be fit to experimental data, for example to achieve optimal prediction performance of spike-timings (see Jolivet et al. [2006]). The model compares well with other models in benchmark tests of prediction performance (see for example, Jolivet et al. [2008]).

2.3 Plasticity and Learning

One of the most fascinating aspects of the brain is its ability to adapt, to learn, to improve. Despite extensive research, the mechanisms by which new memories are formed and retained in the human brain are still poorly understood. Today, it is widely believed that the main neural substrate responsible for learning and memory is synaptic long-term plasticity, the ability of synapses to change their connection strength in response to local and global signal triggers (see for example Martin et al. [2000]).

The two most influential theories of synaptic long-term plasticity, Hebbian learning and Spike-Timing Dependent Plasticity (STDP), will be introduced briefly.

2.3.1 Hebbian Learning

Formulated by Donald Hebb in 1949, the Hebbian theory of learning states that the connection from neuron A to neuron B is reinforced if neuron A "repeatedly or persistently takes part in firing" neuron B (Hebb [1949]). In a crudely simplified version, Hebb's postulate is often quoted as "Cells that fire together, wire together".

Formally it predicts, that if x and y denote pre- and post-synaptic activity, respectively, then the weight change will be proportional to the product of these activities:

$$\Delta w = \eta x y.$$

Note that pure Hebbian learning is intrinsically unstable, since it does not include a mechanism for synaptic depression, leading to a positive feedback loop between post-synaptic activity and synaptic efficacy. Most later learning theories are based on Hebbian learning and incorporate mechanisms to cope with the stability issue, most notably BCM theory (Bienenstock et al. [1988]) and Oja's rule (Oja [1982]). Thanks to its beauty and simplicity, Hebbian learning and its close relatives constitute the most popular and influential body of theories of synaptic plasticity to date.

2.3.2 Spike-Timing Dependent Plasticity

Rate-based plasticity models like Hebbian theory and its descendants dominated the community until researchers began to look into more subtle effects that contribute to synaptic plasticity (see for instance, Abbott and Nelson [2000]). One of their findings was that sign and magnitude of synaptic change depends strongly on the exact timing between pre- and post-synaptic spikes. Specifically, they found that a pre-synaptic spike shortly before a post-synaptic spike induced the greatest positive change in synaptic efficacy, while the sign of change quickly reversed when a pre-synaptic spike was emitted after a post-synaptic spike (Bi and Poo [2001]). In short, a synapse is potentiated if it persistently contributes to post-synaptic firing (note the similarity to Hebb's original phrase). If pre-synaptic spikes occur too late with respect to post-synaptic firing, the synapse is depressed. The resulting dependency of weight change on the spike timing has a characteristic asymmetric form. Apart from underlining the importance of exact spike timings in the nervous system, these findings have fueled the debate on the importance of causality and its detection in neural networks and its alleged role in the formation of episodic memories (see for example, Neves et al. [2008]).

More recent research has suggested that STDP appears in very different forms in cortex, such that the originally findings cover only one of many observed spike-timing dependencies of long-term plasticity. In order to avoid misunderstandings, the original asymmetric time dependence which was found first is sometimes referred to as standard STDP .

In addition to different STDP curves, many surrounding mechanisms have been identified which can modulate plasticity (see Dan and Poo [2006] for an excellent summary). Many aspects of plasticity remain severely understudied and have not entered an appropriate modeling-prediction-testing cycle yet.

2.4 Populations of Neurons

Perception, reasoning, motor control – is it conceivable that the most sophisticated cognitive functions come about by plugging together a bunch of neurons with synapses? While the story is certainly more complex than that, there is hope that investigating populations of neurons and elaborating theories of how neurons can interact to produce meaningful behavior might shed some light on the way the human brain achieves its unmatched performance at a great variety of tasks.

A good starting point for analyzing ensembles of neurons are structural peculiarities observed in brain tissue. For example, one striking feature of neocortex, the most recently evolved part of the brain in mammals, is its layered organization which is briefly discussed in section 2.4.1. Another common theme in neocortical circuits is lateral inhibition, which has been postulated to perform small-scale tasks like noise suppression, decorrelation of firing and many more. One theory which states that local microcircuits with lateral inhibition are able to compute a Winner-Take-All response, “selecting“ only the strongest from a set of neurons while suppressing all others, is discussed in section 2.4.2. For more aspects of neuron populations, the interested reader is referred to the corresponding chapter in Maass and Bishop [2001].

2.4.1 Cortical layers and columns

The cortex is the outermost sheet of neural tissue in the mammalian brain and is thought to mediate higher level cognitive functions like perception, planning and conscious thought. The neocortex, the most recently evolved part of cortex forming almost 80% of the human brain, is organized in six layers with different structural properties (see Jones and Peters [1990]). The six layers are primarily distinguished by prevalence of certain cell types and connection profiles.

In addition to its layered structure, a very prominent feature of cortex is a strong bias towards dense local connectivity and sparse long-distance connections (Purves [2008]). Generally, connections seem to prefer either vertical (perpendicularly across layers) or horizontal (within layers) directions. The vertical connectivity in cortex appears stereotypical and is believed to underlie the formation of small densely connected modules, called columns. Neighboring columns tend to be stronger connected than columns which lie further apart, in accordance within the observation of stronger local connectivity. The columnar organization of cortex was first identified by Mountcastle (see Mountcastle [1957], and for a later review, Mountcastle [1997]) and has arguably become one of the most influential as much as controversial concepts in modern neuroscience. While the anatomical and functional columnarity of cortex stands without doubt, the debate revolves around an attractive yet potentially flawed hypothesis which is typically associated with cortical columns. The hypothesis essentially states that the intrinsic structure of neocortex is highly uniform, and that neocortex should be understood as a stereotypically repetitive system based on just a few basic templates. Even though the hypothesis has been refuted in numerous experiments, it seems to persist in popular opinion (see Rakic [2008]).

From a modeler's perspective, the hypothesis that neocortex is made of a single microstructure, replicated identically millions of times, would be very convenient. However, current research suggests that even though neocortical circuits show stereotypical features at many levels of description, the exact microcircuitry seems to depend not only on the species and age of subject, but also on brain region and exact area. Even neighboring areas within the same region seem to differ substantially in several important aspects (see Silberberg et al. [2002]). These results pose significant problems to the functional modeling of neocortical circuits, since structural variations are expected to go hand in hand with functional differences. In the worst case scenario, if neocortical microcircuits really perform highly individualized functions, then a single model will have difficulties to explain much more than a tiny brain area.

There is nevertheless hope that some of the structural variance can be explained by theories and models which incorporate learning and structural adaptation to input stimuli. The basic rationale behind this hope is that, since each brain area receives inputs with distinct statistics, at least some fraction of the observed differences in circuitry could be explained by the ubiquitousness of plasticity and adaptability in neocortex (see for example, Douglas and Martin [2004]).

2.4.2 Lateral Inhibition, Winner-Take-All (WTA) and Competitive Learning

Lateral inhibition refers to the ability of a neuron to suppress its neighbors when activated. The concept was first proposed when mutual inhibitory relationships between detectors with similar response properties were found in visual and auditory psychophysical experiments (see for example, Houtgast [1972]). Physiological evidence for lateral inhibitory processes corresponding to psychophysical findings soon consolidated the notion (see Blakemore and Tobin [1972]). A first corresponding theory was formulated in Grossberg [1973], which describes the continuous time dynamics of networks with recurrent inhibition and is the first theoretical treatment of Winner-Take-All networks.

Winner-Take-All (WTA) networks are artificial models utilizing lateral inhibition for a particular purpose, namely the selection and reinforcement of the strongest neuron among a group and the suppression of all others. Several variants of WTA exist:

K-WTA: Instead of a single winner neuron, the K strongest neurons are selected.

Soft WTA: Weaker neurons are not completely suppressed but significantly damped.

Probabilistic WTA: Each neuron is assigned a probability of winning depending on its strength, the selection is randomized. Stronger neurons will win more often due to higher winning probabilities.

A recent theoretical analysis has revealed unique computational benefits of WTA compared to other non-linearities (Maass [2000]). Given these findings and the fact that lateral inhibition is such an omnipresent phenomenon in the brain, it is not surprising that WTA-like functionality has also been postulated to play a major role in neocortical function (see Coultrip et al. [1992] and Douglas and Martin [2004]).

WTA circuits combine well with other elements of neural networks. For example, in a two-layer artificial neural network with one input layer and an output layer with WTA recurrent connectivity, the WTA functionality can be exploited for template matching. Provided that weights are normalized appropriately, finding the strongest neuron is equivalent to finding the best match between stored weights and current input (see Rumelhart and Zipser [1985]). Furthermore, endowing such a network with Hebbian learning gives rise to competitive learning, whereby each neuron tends to specialize on a small subset of input patterns. One of the first competitive learning schemes was discussed in von der Malsburg [1973]. Perhaps the most popular representative of competitive learning is the Self-Organizing-Map (Kohonen [1990]).

2.5 Simulation in Software and Hardware

In principle, research into Spiking Neural Networks could be done by integrating experimental data and devising theories alone. However, evaluating theories by simulation is helpful for a number of reasons:

- It can be used to better visualize complex processes, and to explain them to an audience.
- It helps to reveal weaknesses or undesired behaviors of a model which were not apparent in the theoretical framework.
- It assists in the justification of approximations by showing that the desired functionality is still achieved.
- It serves as an additional source of inspiration, both for abstract ideas and concrete improvements.

A quick overview of some current simulation tools and devices, is given below.

2.5.1 Software Simulation

There are plenty of simulations tools for Spiking Neural Networks available. In fact, the number of options has grown so large that there is increasing demand for a common interface language for neural network simulators. One recent project addressing this issue is PyNN (Davison et al. [2008]), which is based on the Python scripting language (Sanner [1999]), and is quickly becoming a standard for neural network interfacing, both with Software and Hardware simulators. Currently supported software simulators include (Davison et al. [2008], Brette et al. [2007]):

NEURON: The NEURON simulation environment (Hines and Carnevale [1997]) has a long tradition. Since its beginnings in the early 1990s, it has been used in countless neuroscientific experiments and publications, in particular to study observed biological phenomena like dendritic excitability or network states and oscillations in a controlled and isolated manner.

NEST: The NEST initiative is a long-term project dedicated to promote research into large-scale neural network simulation technologies. Its efforts have lead to the development of the NEST simulation tool, the reference project of the initiative (Diesmann and Gewaltig [2002]). Its main domain of usage are large-scale neural networks with biologically realistic connectivity profiles.

Brian: A simulator written purely in Python (Goodman and Brette [2008]). The scripting language approach provides unique benefits for users concerned with rapid development and testing of new models.

PCSIM: A C++/Python based simulator developed at Graz, Technical University (Pecevski et al. [2009]) which aims at combining the speed of C++ with Python's ease of configuration, interfacing and extensibility. Its predecessor CSIM was originally developed for simulating networks which implement the Liquid State Machine paradigm (Maass et al. [2002]), a highly influential theory which has established an intimate connection between diversity in neural networks and computational power. Due to its computational roots, it supports a large variety of machine learning algorithms which can be seamlessly combined with spiking neural networks in real-time simulations. It is particularly well-suited for large-scale simulations distributed on multiple machines.

For a more information on current simulation tools, the interested reader is referred to Brette et al. [2007].

2.5.2 Simulation on Special-Purpose Hardware

Software simulation tools are the natural choice for many applications due to extremely favorable properties in terms of extensibility, availability and costs.

However, there are some limitations inherent to current multi-purpose architectures which might be overcome by custom neural network implementations in Very-Large-Scale-Integration (VLSI) hardware. Massive parallelism is perhaps the most striking feature of neural computation. While there are ongoing efforts to increase parallelism also in multi-purpose architectures, it is clear that custom VLSI hardware can be more directly geared towards the massively parallel computing paradigm that neural computing suggests. Beyond the apparent advantage of exploiting parallelism explicitly in hardware, it was argued that special purpose low-precision circuits might suffice for neural networks, given that neural computation in the brain appears noisy and unreliable on the neuron level (Hammerstrom [1998], London et al. [2010]).

Early VLSI implementations of neural networks date back to the late 1980s, see for example Murray and Smith [1988] or Satyanarayana et al. [1992]. A significant step towards spike-based neural hardware implementations was the development of the Address-Event Representation (AER), described in Mahowald [1992]. The basic idea behind AER is the definition of a unified messaging protocol between circuits which includes a sender identification (Address) and a timestamp (Event) in each message. Additional payload can be appended to this minimal message format, depending on the application. AER allows for efficient temporal multiplexing by sharing a common message carrier among multiple senders. Many neurally inspired circuits have been developed on the basis of AER, including artificial cochlear and retina implementations (see for example van Schaik and Liu [2005], Delbruck and Lichsteiner [2006] and Conradt et al. [2009]), as well as large-scale neural network systems for integrated sensory processing, learning and actuating (see Serrano-Gotarredona et al. [2009], Furber et al. [2008]). A slightly different approach for temporal multiplexing was taken in the wafer-scale VLSI implementation of a spiking neural network in Ehrlich et al. [2007].

Simulating on hardware may substantially speed up development cycles and decrease associated costs. There are however a few important limitations to consider when using special purpose hardware for simulations:

- Hardware implementations are limited in the class of neuron models they can implement. In particular, many features of detailed models like dendritic processing or complex synaptic dynamics and non-standard learning rules might be difficult if not impossible to simulate on current hardware.

- Hardware elements are prone to small deviations from the desired characteristics due to process variations (see for example Pelgrom et al. [1998]). In analog circuits, these variations can severely affect the accuracy of a simulation.

- Typically spike-based hardware runs many orders of magnitude faster than a corresponding biological circuit would (for example, Furber et al. [2008]). Providing high-bandwidth input to such a network and analyzing its output in real-time may pose significant problems to the surrounding framework.

Chapter 3

Probabilistic Modeling, Inference and Learning

Symbolic logic is the language of exact reasoning. The famous “Modus Ponens“ dictates how to perform exact logical inference: If we know that some fact P holds, and we know that Q follows from P, then also Q must hold (Smullyan [1961]). In contrast to the exact nature of symbolic logic, probability theory is the language of uncertainty. P is probably true, and Q usually follows from P. Does Q hold? Probability theory provides a framework for describing and solving problems of this kind in a mathematically exact way. In many ways, probability theory may be viewed as an extended theory of logic (Jaynes and Bretthorst [2003]). Just like logic it provides us with a tool for reasoning, yet allowing for more flexible representations of beliefs and relations than its exact counterpart.

Artificial intelligence long pursued the “logic approach” to reasoning, most notably in the form of expert systems (Jordan and Russell [1999]). Only in the late 1980s, probability theory made its appearance in artificial intelligence: the highly influential book “Probabilistic reasoning in intelligent systems: networks of plausible inference” by Pearl [1988] introduced Bayesian Networks and represented a major stepping stone to modern machine learning and artificial intelligence. Today, probabilistic and statistical techniques have long become indispensable in most technical areas. Strikingly, recent neurophysiological and psychological studies (see Yang and Shadlen [2007] and Griffiths and Tenenbaum [2006]) also suggest a decisive role of probabilistic inference in human reasoning and decision making. In the light of this development it is hardly surprising that the mechanisms by which probabilistic inference and learning could be carried out in spiking neural networks have more than ever become a primary research target of computational neuroscience (see for example, Doya [2007]).

In this chapter, the basic ideas and formalities of probability theory and statistics, required for understanding the remainder of this thesis, will be briefly presented.

3.1 Basic Terminology

Before proceeding to more advanced topics, some fundamental concepts around probabilistic models, inference and learning, will be introduced.

3.1.1 Probabilistic Models

A probabilistic model captures relations among a set of random variables (Bishop [2006]; MacKay [2003]). If x and y are two random variables, a typical probabilistic model provides the *joint probability distribution* $p(x, y)$ which assigns a relative frequency of occurrence to each joint configuration of x and y. This is equivalent to a full specification of deterministic and random effects which govern these variables. From the joint distribution, one can derive other useful distributions, such as:

- *Marginal distribution*: $p(x) = \sum_y p(x, y)$: specifies how x is distributed "in disregard" of y. Analogously, $p(y) = \sum_x p(x, y)$.
- *Conditional distribution*: $p(x|y) = p(x, y)/p(y)$: specifies how x is distributed if a particular assignment of y is known. Similarly, $p(y|x) = p(x, y)/p(x)$.

In many cases, it is more convenient to formulate a model in terms of its constituent marginal and conditional distributions, rather than its complete form. For example, the specification of two distributions $p(y)$ and $p(x|y)$ instead of an explicit joint distribution $p(x, y)$ might have some advantages for a given task. Note that the split formulation is formally equivalent since $p(x, y) = p(y)p(x|y)$.

Sometimes, a full joint probability distribution is not even necessary for the task at hand. In such a scenario, a model might provide only a subset of marginal and conditional distributions. In the two-variable example from above, one might be interested exclusively in the conditional dependencies of y on x. Then a sufficient model would be $p(y|x)$.

3.1.2 Inference

An *observation* is a particular assignment to a subset of random variables in a model (MacKay [2003]). Knowledge of a subset of variables can be used to reason about other variables. This procedure is called *inference.* For instance, if the model provides $p(y|x)$ and x can be observed, then the probability distribution over y can be inferred. In a typical machine learning application, there is a fixed subset of variables which can be observed. These variables are then called *observed* variables, as opposed to *latent* (or hidden) variables which have be inferred using a model. When performing inference, the distribution over latent variables before making an observation is called *prior distribution.* In the above example, $p(y)$ would be a prior distribution. After taking all observations into account, the distribution over latent variables is called *posterior distribution* ($p(y|x)$ in the example).

3.1.3 Kullback-Leibler Divergence

Consider two probability distributions over the same set of variables, for example $p(x)$ and $q(x)$. The *Kullback-Leibler (KL) divergence* between p and q, written $D_{KL}(p||q)$, is a measure of how strongly q diverges from p. For a discrete random variable x, the definition is given by (MacKay [2003]),

$$D_{KL}(p||q) = \left\langle \log \frac{p(x)}{q(x)} \right\rangle_{p(x)} \tag{3.1}$$

$$= \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{3.2}$$

The Kullback-Leibler divergence is strictly non-negative and zero if and only if the two distributions match exactly over the entire variable range. It is strongly asymmetric since in general $D_{KL}(p||q) \neq D_{KL}(q||p)$, and is therefore not a metric. An important application arises when trying to approximate a given distribution $p(x)$ by another distribution $q(x)$. In such a case, minimization of $D_{KL}(p||q)$ is useful to the optimize the approximation.

3.1.4 Parametric Models

So far, probability distributions have been treated as rather abstract objects which can be accessed and queried regardless of their internal representation. Also the issue of setting up probability distributions and filling them with knowledge on random variables and their relations has not been raised yet. *Parametric models* constitute an elegant and unified approach to both, representation and knowledge transfer (Bishop [2006]). A parametric model is a distribution which is controlled via a number of finite parameters, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)^T$. A parametrized distribution may be written in several ways to express the dependence on a parameter set $\boldsymbol{\theta}$:

- Suggesting conditioning on parameters: $p(x, y|\,\boldsymbol{\theta})$.
- Markedly Separated: $p(x, y; \boldsymbol{\theta})$.
- As a subscript: $p_{\boldsymbol{\theta}}(x, y)$.

A simple example of a parametrized distribution is the following Poisson distribution, parametrized by the Poisson rate λ:

$$p(x|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}. \tag{3.3}$$

Note that each choice of λ yields a different distribution over x. Therefore, technically Eq. 3.3 describes a family of distributions.

3.1.5 Generative Models

If a model can be used to compute the marginal distribution over observed variables, it is called a *generative model* (Bishop [2006]). Supposing that x is observed and y is a latent variable, then $p(x, y)$ would be a generative model for x since the marginal distribution $p(x)$ can be computed from the joint distribution. Note that, due to the equivalence, also a model providing $p(y)$ and $p(x|y)$ is a generative model for x. The term "generative" reflects the fact that the model may be used to generate samples from the modeled input distribution.

3.1.6 Learning from Samples and Maximum Likelihood (ML)

Continuing the above example, suppose that a random variable x, corresponding to some measurable quantity in real life, should be modeled. A number of observations $x(1), \ldots, x(T)$ have been recorded and prior knowledge suggests that x is Poisson distributed. The only parameter which is unsettled is the rate λ. This is where *Learning* from data comes into play. Given a parametrized model like Eq. 3.3, and a number of complete observations involving all variables of interest, one would like to approximate the observed distribution as closely as possible. As has been anticipated in Section 3.1.3, a systematic approach to approximation of a probability distribution is the minimization of the KL-divergence between original and approximating distribution (MacKay [2003]). The first derivative of the negative KL-divergence with respect to a set of parameters $\boldsymbol{\theta}$ is given by,

$$\frac{\partial(-D_{KL}(p||q_{\boldsymbol{\theta}}))}{\partial\boldsymbol{\theta}} = \left\langle \frac{\partial(-\log\frac{p(x)}{q_{\boldsymbol{\theta}}(x)})}{\partial\boldsymbol{\theta}} \right\rangle_{p(x)} \tag{3.4}$$

$$= \left\langle \frac{\partial(-\log p(x) + \log q_{\boldsymbol{\theta}}(x))}{\partial\boldsymbol{\theta}} \right\rangle_{p(x)} \tag{3.5}$$

$$= \left\langle \frac{\partial \log q_{\boldsymbol{\theta}}(x)}{\partial\boldsymbol{\theta}} \right\rangle_{p(x)} \tag{3.6}$$

This can be interpreted in the following way. Optimizing the approximating distribution means decreasing the KL-divergence. In order to achieve that, a learning scheme must seek to increase the average *log-likelihood* $\log q_{\boldsymbol{\theta}}(x)$ of the data. The learning principle which is based on this rationale is therefore termed *Maximum Likelihood.* Note that in a typical learning scenario, direct access to the original distribution $p(x)$ is not available. Instead one must resort to using samples as a proxy for the real distribution which can be achieved by replacing the ideal expectation by a sample-based expectation:

$$\left\langle \frac{\partial \log q_{\boldsymbol{\theta}}(x)}{\partial\boldsymbol{\theta}} \right\rangle_{p(x)} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \log q_{\boldsymbol{\theta}}(x_i)}{\partial\boldsymbol{\theta}} \tag{3.7}$$

The procedure extends naturally to more complex models with thousands of variables and intricate relations among them. It should be noted that direct maximization by letting $\frac{\partial D_{KL}(p||q_{\boldsymbol{\theta}})}{\partial\boldsymbol{\theta}} \overset{!}{=} \mathbf{0}$ is often intractable for more involved models. In such a situation, any standard hill-climbing algorithm like gradient ascent may be used in an iterative optimization procedure.

Another note is in order concerning the observations required to perform learning according to this scheme. As was mentioned before, complete observations including all variables of the model are necessary to compute the required gradients for optimization. Clearly this presents a significant obstacle to models which include latent variables, which by definition can not be observed. The learning scheme for such a model turns out slightly more difficult. One algorithm which deals with this issue and is of particular relevance for this thesis, will be covered in Section 3.2.

3.1.7 Conditional Independence

Two random variables are statistically (unconditionally) *independent* if the observation of either variable does not alter the belief of how the other variable is distributed (Bickel and Doksum [2001]; Bishop [2006]). The following statements are equivalent:

- x and y are statistically independent.
- $p(x, y) = p(x) \cdot p(y)$.
- $p(y|x) = p(y)$.
- $p(x|y) = p(x)$.

Conditional independence is a similar concept of fundamental importance which is best explained with a simple example: Alex is an undergraduate student with a rather bad immune system. When he is sick he naturally prefers not to attend university classes. Also, he usually drinks tea to speed up his recovery. Sometimes he misses university classes also for other reasons, and occasionally he also drinks tea when healthy.

Based on this knowledge, the observation that he has not come to university increases the belief that he is drinking tea that day. Conversely, knowing whether he has drunk tea allows one to quite accurately predict whether Alex went to university. The two events, namely “tea drinking“ and “missing at university“ are strongly correlated and clearly anything but independent. However, the situation dramatically changes the moment Alex tells you he his sick (or healthy). The knowledge of his health status removes the dependence between tea and university. Knowing already that he is sick, the observation that he has missed in university has become irrelevant for predicting his tea consumption. If $Sick$ is a binary random variable indicating whether Alex is sick on a specific day or not, Tea is a binary variable indicating whether he has drunk tea that day, and Uni a variable representing his presence in university, then the following statements are equivalent:

- Tea and Uni are conditionally independent given $Sick$.
- $p(Tea, Uni|Sick) = p(Tea|Sick) \cdot p(Uni|Sick)$
- $p(Tea|Uni, Sick) = p(Tea|Sick)$
- $p(Uni|Tea, Sick) = p(Uni|Sick)$

When setting up a model, knowledge of conditional independencies among variables can be used to significantly reduce the model's complexity. This can be done by separation of the joint probability distribution into conditional distributions: To exploit conditional independencies in the above example, the full model $p(Tea, Uni, Sick)$ should be decomposed into

$$\begin{aligned} p(Tea, Uni, Sick) &= p(Sick)p(Tea, Uni|Sick) && (3.8) \\ &= p(Sick)p(Tea|Sick)p(Uni|Sick) && (3.9) \end{aligned}$$

What has been gained here is that the model from Eq. 3.9 requires at most two-dimensional objects, while the original formulation in Eq. 3.8 involved a three-dimensional object. It would be rather wasteful to directly model a full joint distribution, when there is domain knowledge suggesting conditional independencies. A frequently used independence constraint which greatly reduces model complexity, is discussed below in Section 3.1.8.

3.1.8 Naïve Bayes

In Section 3.1.7 it has been noted that implementing conditional independence assumptions may significantly reduce a model's complexity. The Naïve Bayes assumption may be regarded the strongest and most widely used type of independence assumption and its benefits in terms of computational complexity and memory consumption are remarkable (Bishop [2006], Hand and Yu [2001]). Consider a latent variable model with a number of observed variables $x_1, \ldots, x_N$ and a single latent variable z. The joint probability distribution can be written as,

$$p(x_1, \ldots, x_N, z) = p(z) \cdot p(x_1, \ldots, x_N|z). \tag{3.10}$$

The Naïve Bayes assumption states, that all observed variables are conditionally independent of each other, given the latent variable. Then the joint probability distribution simplifies to,

$$p(x_1, \ldots, x_N, z) = p(z) \cdot p(x_1|z) \cdot p(x_2|z) \cdots p(x_N|z). \tag{3.11}$$

The most immediate benefit from such a formulation is memory consumption: instead of having to handle an $(N + 1)$-dimensional object, the distribution can decomposed into at-most two-dimensional objects. Another obvious implication is the ease of generating input samples from the model: instead of sampling from $p(x_1, \ldots, x_N|z)$, which may be arbitrarily complex and require sophisticated and time-consuming sampling techniques, each “feature” can be generated independently according to $p(x_i|z)$, once the latent variable is sampled.

Inference in the model above can be performed according to,

$$p(z|x_1, \ldots, x_N) = \frac{p(x_1, \ldots, x_N, z)}{\sum_{z'} p(x_1, \ldots, x_N, z')}, \tag{3.12}$$

assuming a discrete latent variable z. Since the joint probability is decomposed with Naïve Bayes, inference can be highly parallelized and distributed, since each input i adds an independent factor $p(x_i|z)$ to the joint distribution.

Finally, also learning in a model with Naïve Bayes is greatly simplified. First, note that also parameters may be distributed among conditional distributions:

$$p_{\boldsymbol{\theta}}(x_1, \ldots, x_N, z) = p_{\boldsymbol{\theta}_0}(z) \cdot p_{\boldsymbol{\theta}_1}(x_1|z) \cdot p_{\boldsymbol{\theta}_2}(x_2|z) \cdots p_{\boldsymbol{\theta}_N}(x_N|z). \tag{3.13}$$

The log-likelihood of a data sample x is given as,

$$\log p_{\boldsymbol{\theta}}(x_1, \dots, x_N, z) = \log p_{\boldsymbol{\theta}_0}(z) + \log p_{\boldsymbol{\theta}_1}(x_1|z) + \cdots + \log p_{\boldsymbol{\theta}_N}(x_N|z). \tag{3.14}$$

As expected, the multiplicative factorization of the probability distribution results in a linear decomposition in the log-space. When differentiating with respect to $\boldsymbol{\theta}$, one obtains entirely decoupled derivatives,

$$\frac{\partial \log p_{\boldsymbol{\theta}}(x_1, \dots, x_N, z)}{\partial \boldsymbol{\theta}_0} = \frac{\partial \log p_{\boldsymbol{\theta}_0}(z)}{\partial \boldsymbol{\theta}_0}, \tag{3.15}$$

$$\frac{\partial \log p_{\boldsymbol{\theta}}(x_1, \dots, x_N, z)}{\partial \boldsymbol{\theta}_i} = \frac{\partial \log p_{\boldsymbol{\theta}_i}(x_i|z)}{\partial \boldsymbol{\theta}_i}, \quad i = 1, \dots, N. \tag{3.16}$$

When optimizing parameters according to the Maximum Likelihood principle, the resulting decoupled systems can be solved independently, thereby greatly reducing computational demands and complexity.

Having discussed the benefits of Naïve Bayes at great length, a few critical remarks are in order to complete the picture. Naïve Bayes' enjoys widespread use, even in applications where the corresponding conditional independence assumptions are inadequate. In some situations, especially when data is scarce, its use may still be justified in order to avoid overfitting (Hand and Yu [2001]). In order to illustrate this point, consider an application with N input variables, jointly Gaussian. Modeling a multivariate normal distribution involves a full covariance matrix and therefore $O(N^2)$ variables, while the corresponding Naïve Bayes model requires only $O(N)$. If too little data is available to reliably fill the covariance matrix, then Naïve Bayes becomes a viable alternative. Also in discrimination tasks, where the posterior probability distribution over the latent variable is used to decide among a set of alternatives, meticulous precision of the underlying generative model is of minor importance. For discrimination performance, it suffices that the posterior probability for the correct choice is higher than all others, regardless of exact probabilities. Hence, also in such a scenario, it might be reasonable to trade a little performance for all the advantages of a simpler model like Naïve Bayes (Hand and Yu [2001]).

3.1.9 Mixture Models

There is a glaring mismatch between the simplicity of many computationally feasible models, and the complexity of probability distributions typically encountered in real life. A common issue are multimodal probability distributions, that is distributions with multiple distinct local maxima, or peaks. Many popular parametric models are unimodal, and more complex models which support multimodality are often unattractive for practical reasons. Here, mixture models step in to provide an efficient compromise between simplicity and complexity (Bishop [2006]).

Mixture models are probabilistic models which define a joint distribution as a composition, or mixture, of a number of component distributions:

$$p(\mathbf{x}) = \sum_i a_i f_i(\mathbf{x}). \tag{3.17}$$

If the component weights a_i sum to one and all component distributions $f_i(\mathbf{x})$ are properly normalized and non-negative, then also $p(\mathbf{x})$ will be normalized and constitute a valid probability distribution. Mixture models can also be constructed and understood by introducing a latent variable which acts as a probabilistic switch between component distributions.

$$p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}\,|z) \tag{3.18}$$

In this form, the components $p(\mathbf{x}\,|z)$ are distributions conditioned on the latent variable.

Learning mixture models from data can be best understood with the formulation in Eq. 3.18 in mind. Since a latent variable is involved, direct maximization of data likelihood is impossible. Since learning in mixture models is of particular interest to this thesis, Section 3.2 will introduce the Expectation Maximization algorithm, a popular method to overcome this issue.

3.1.10 Classification

One of the most basic forms of reasoning is classification: given an observed pattern, decide among a number of categories which one fits the pattern best. The classification performance depends on two crucial factors: the quality of the model and the accuracy of inference. While inference can be performed exactly in most models which are used in practice, the quality of the model usually presents the bigger challenge. There are at least three ways to arrive at a useful classification model (Bishop [2006]; MacKay [2003]):

Learn discriminative model from data with supervision: Given a number of training input patterns $\mathbf{x}$ and corresponding labels z, learn a model for $p(z|\mathbf{x})$. When presented a new pattern $\mathbf{x}$, evaluate $p(z|\mathbf{x})$ for each z to determine the most likely label.

Learn generative model from data with supervision: Given a number of training input patterns $\mathbf{x}$ and corresponding labels z, learn models for $p(z)$ and $p(\mathbf{x}|z)$. The term "generative model" reflects the fact that the model can produce samples from the input distribution, which is conveniently given by a mixture model: $p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}|z)$. When asked for classification of a new pattern $\mathbf{x}$, compute $p(z|\mathbf{x}) \propto p(y)p(\mathbf{x}|z)$ for each category z and choose winner.

Learn generative model from data without supervision: Given a number of training input pattern $\mathbf{x}$ without labels, introduce a hidden variable z which corresponds to the unknown labels. Learn a mixture model of the form $p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}|z)$. Upon arrival of a new pattern, evaluate $p(z|\mathbf{x}) \propto p(z)p(\mathbf{x}|z)$ and choose the best label.

3.2 Expectation Maximization (EM)

In parametric latent variable models one has a number of observed variables $\mathbf{x}$ and hidden variables $\mathbf{z}$, and relations among these which are defined via a set of parameters $\boldsymbol{\theta}$. In order to obtain a generative model from a given data set (T samples which are assumed to be drawn i.i.d.), one seeks to maximize the likelihood of these data samples under the model (Bickel and Doksum [2001]),

$$p_{\boldsymbol{\theta}}(\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(T)) = \prod_t p_{\boldsymbol{\theta}}(\mathbf{x}(t)). \tag{3.19}$$

As will become clear in Section 3.2.1, likelihood optimization in latent variable models is somewhat more difficult than in fully observed models. Direct optimization turns out to be intractable, such that iterative optimization schemes have to be employed. Lower bound optimization is a general iterative optimization principle which is introduced in Section 3.2.2. A concrete algorithm for Maximum Likelihood based on this principle is Expectation Maximization (Dempster et al. [1977]). It is composed of an iteration of two steps, which will be discussed in Section 3.2.3 and 3.2.4.

3.2.1 Likelihood Gradient

Likelihood maximization is usually performed in the log-domain (Bickel and Doksum [2001]):

$$L(\boldsymbol{\theta}) = \frac{1}{T} \log p_{\boldsymbol{\theta}}(\mathbf{x}(1)\, \mathbf{x}(2), \ldots, \mathbf{x}(T)) = \frac{1}{T} \sum_t \log p_{\boldsymbol{\theta}}(\mathbf{x}(t)). \tag{3.20}$$

This is a convenient reformulation since the gradient with respect to the parameters can be decoupled across observations:

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial \sum_t \log p_{\boldsymbol{\theta}}(\mathbf{x}(t))}{\partial \boldsymbol{\theta}} \tag{3.21}$$

$$= \sum_t \frac{\partial \log p_{\boldsymbol{\theta}}(\mathbf{x}(t))}{\partial \boldsymbol{\theta}}. \tag{3.22}$$

In order to evaluate the likelihood of a single input pattern $\mathbf{x}$ in a latent variable model, one must compute,

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}). \tag{3.23}$$

The sum runs over all possible configurations of latent variables $\mathbf{z}$. Since the number of configurations may grow exponentially in the number of latent variables, Eq. 3.23 poses a major problem to Maximum Likelihood learning in many latent variable models. In mixture models, the number of possible configurations is limited by the number of components K which greatly alleviates the issue:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{k=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}, z_k). \tag{3.24}$$

However, both in the general case and in mixture models, the parameter gradient in the log-domain yields strongly coupled derivatives,

$$\begin{aligned}
\frac{\partial \log p_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \boldsymbol{\theta}} &= \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})} \frac{\partial p_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \boldsymbol{\theta}} && (3.25)\\
&= \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})} \frac{\partial \sum_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{\partial \boldsymbol{\theta}} && (3.26)\\
&= \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})} \cdot \sum_{\mathbf{z}} \frac{\partial p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{\partial \boldsymbol{\theta}} && (3.27)\\
&= \sum_{\mathbf{z}} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} \frac{\partial \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{\partial \boldsymbol{\theta}} && (3.28)\\
&= \sum_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, \mathbf{x}) \frac{\partial \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{\partial \boldsymbol{\theta}}, && (3.29)
\end{aligned}$$

since the conditional probabilities $p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, \mathbf{x})$ in Eq. 3.29 involve all parameters of the model. This usually makes direct likelihood optimization via $\frac{\partial \log p_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \boldsymbol{\theta}} \overset{!}{=} \mathbf{0}$ intractable, even for a single observation. Iterative gradient-based methods like gradient ascent are one way to get around this problem (see for example Salakhutdinov et al. [2003]). A slightly different approach will be presented below in Section 3.2.2.

3.2.2 Lower Bound Optimization

The principle of lower bound optimization is sketched below (Bishop [2006]):

1. Initialization: Consider maximization of an arbitrary function $L(\boldsymbol{\theta})$

and start with some initial guess $\boldsymbol{\theta}_{cur}$.

2. **Lower Bound:** Construct a lower bound $\hat{L}(\boldsymbol{\theta})$ which touches the original function at the current position, such that $\hat{L}(\boldsymbol{\theta}_{cur}) = L(\boldsymbol{\theta}_{cur})$.

3. **Maximization:** Find a $\boldsymbol{\theta}_{new}$ which maximizes the lower bound. Let $\boldsymbol{\theta}_{cur} \leftarrow \boldsymbol{\theta}_{new}$.

Repeat 2. and 3. until convergence.

Note that the key step in this scheme is the construction of the lower bound, which may be difficult to find for many practical problems. If, however, one can find a lower bound with the desired characteristics, then this scheme has several nice properties:

- Since the lower bound touches the original function at the current position, its gradient must match the original gradient there. Hence, the lower bound will have a local maximum at $\boldsymbol{\theta}_{cur}$ only if the current position is also a local maximum of the true optimization goal.

- Note again that $\hat{L}(\boldsymbol{\theta}_{cur}) = L(\boldsymbol{\theta}_{cur})$. Unless there is a local maximum at the current position, maximization of $\hat{L}(\boldsymbol{\theta})$ is guaranteed to increase the lower bound, such that $\hat{L}(\boldsymbol{\theta}_{new}) > \hat{L}(\boldsymbol{\theta}_{cur})$. By definition of a lower bound, one also has $L(\boldsymbol{\theta}_{new}) \geq \hat{L}(\boldsymbol{\theta}_{new})$. These facts taken together, it becomes apparent that $L(\boldsymbol{\theta}_{new}) > L(\boldsymbol{\theta}_{cur})$: *An iteration always increases the true function value, until a local maximum is found.*

- The lower bound is often concave (Ghahramani and Jordan [1997]). Convex (concave) optimization is a particularly well-studied field and many efficient algorithms have been found (Boyd and Vandenberghe [2004]). Note that this is only relevant in cases where an analytical solution of the lower bound optimization problem can not be found.

- The gradient of the true optimization goal need not computed at any point in the algorithm. This may entail computational benefits for complicated $L(\boldsymbol{\theta})$.

- The scheme is very simple. Depending on the quality of the lower bound, it may significantly outperform traditional gradient-based algorithms of comparable simplicity like gradient ascent, since it will take well-informed large steps through state space when adequate.

3.2.3 Expectation

In contrast to gradient-based methods, Expectation Maximization relies on lower bound optimization for increasing the data likelihood under a latent variable model (Dellaert [2002]). The difficulty of lower bound optimization lies in the construction of a lower bound which should be tight at the current position, and preferably concave.

The optimization objective in learning probabilistic models is the data likelihood function. For a single sample we have,

$$\begin{aligned} L(\boldsymbol{\theta}; \mathbf{x}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x}) && (3.30) \\ &= \log \sum_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}). && (3.31) \end{aligned}$$

In mixture models this would correspond to,

$$L(\boldsymbol{\theta}; \mathbf{x}) = \log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \sum_{k=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}, z_k). \qquad (3.32)$$

As has been discussed in Section 3.2.1, the logarithm of a sum makes for a difficult optimization target. In order to obtain a lower bound of $L(\boldsymbol{\theta})$, note that Jensen's inequality states,

$$\varphi\left(\sum_{x} g(x) f(x)\right) \geq \sum_{x} \varphi(g(x)) f(x), \qquad (3.33)$$

if $f(x)$ is a probability distribution over x and ϕ is a concave function over the range of interest. Since the logarithm is strictly concave, one can introduce an arbitrary artificial distribution $f(\mathbf{z})$ and apply Jensen's inequality to the likelihood function:

$$L(\boldsymbol{\theta}; \mathbf{x}) = \log \sum_{\mathbf{z}} f(\mathbf{z}) \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{f(\mathbf{z})} \geq \sum_{k=1}^{K} f(\mathbf{z}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{f(\mathbf{z})} = \hat{L}(\boldsymbol{\theta}; \mathbf{x}, \boldsymbol{\theta}_{cur}). \quad (3.34)$$

In order to make the bound tight at the current parameter estimate $\boldsymbol{\theta}_{cur}$, the distribution $f(\mathbf{z})$ must be chosen such that,

$$\log \sum_{\mathbf{z}} f(\mathbf{z}) \frac{p_{\boldsymbol{\theta}_{cur}}(\mathbf{x}, \mathbf{z})}{f(\mathbf{z})} \overset{!}{=} \sum_{\mathbf{z}} f(\mathbf{z}) \log \frac{p_{\boldsymbol{\theta}_{cur}}(\mathbf{x}, \mathbf{z})}{f(\mathbf{z})}. \quad (3.35)$$

Interestingly, this is just achieved by the posterior distribution $f(\mathbf{z}) = p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x})$ given the current input. This gives the final lower bound for the likelihood of a single sample $\mathbf{x}$:

$$\begin{aligned}
\hat{L}(\boldsymbol{\theta}; \mathbf{x}, \boldsymbol{\theta}_{cur}) &= \sum_{\mathbf{z}} p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x})} && (3.36) \\
&= \sum_{k=1}^{K} p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \sum_{\mathbf{z}} p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x}) \log p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x}) && (3.37) \\
&= \underbrace{\langle \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) \rangle_{p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x})}}_{Q(\boldsymbol{\theta}; \mathbf{x}, \boldsymbol{\theta}_{cur}) \,=\, \text{Expectation}} + \underbrace{H_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x})}_{\text{Independent of } \boldsymbol{\theta}} . && (3.38)
\end{aligned}$$

As indicated in Eq. 3.38, the second term is independent of $\boldsymbol{\theta}$ and may therefore be dropped in anticipation of the subsequent maximization step. Evaluation of the first term, which will be denoted as $Q(\boldsymbol{\theta}; \mathbf{x}, \boldsymbol{\theta}_{cur})$, is known as the *expectation step* and can be understood as follows:

- Given an input pattern $\mathbf{x}$, inference is performed according to the current parameter set $\boldsymbol{\theta}_{cur}$. This provides the posterior distribution $p_{\boldsymbol{\theta}_{cur}}(\mathbf{z} \,|\, \mathbf{x})$.
- Samples $\mathbf{z}^i$ from this posterior distribution lead to *full observation* pairs $(\mathbf{x}, \mathbf{z}^i)$.
- Then, the expectation is computed as the log-likelihood $\log p(\mathbf{x}, \mathbf{z})$, averaged over all observation pairs.

Note that the general difficulty of Maximum Likelihood in latent variable models arises from the fact that only partial observations are available. Expectation Maximization tackles this issue by filling the missing variables via inference with the current parameter set.

Note that so far only a single observation was considered. In order to account for multiple observations, the lower bound of the full likelihood can be constructed straightforwardly by linear composition,

$$\hat{L}(\boldsymbol{\theta};\boldsymbol{\theta}_{cur}) = \frac{1}{T}\sum_t \hat{L}(\boldsymbol{\theta};\mathbf{x}(t),\boldsymbol{\theta}_{cur}). \tag{3.39}$$

Analogously, the full expectation can be linearly composed of individual sample expectations:

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}_{cur}) = \frac{1}{T}\sum_t Q(\boldsymbol{\theta};\mathbf{x},\boldsymbol{\theta}_{cur}). \tag{3.40}$$

3.2.4 Maximization

During the expectation step, the average data log-likelihood in dependence of $\boldsymbol{\theta}$ is computed. The expectation runs over input samples, augmented by inferred hidden variables. Let $p^*(\mathbf{x})$ denote the true (empirical) input distribution. Then, the augmented distribution may be written as,

$$p^*_{\boldsymbol{\theta}_{cur}}(\mathbf{x},\mathbf{z}) = p^*(\mathbf{x})p_{\boldsymbol{\theta}_{cur}}(\mathbf{z}\,|\,\mathbf{x}), \tag{3.41}$$

and the expectation can be expressed as,

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}_{cur}) = \langle \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\rangle_{p^*_{\boldsymbol{\theta}_{cur}}(\mathbf{x},\mathbf{z})}\,. \tag{3.42}$$

As has been discussed in Section 3.1.6 and 3.1.8, likelihood maximization with fully observed data is straightforward for many models, especially when conditional independence assumption are implemented wisely.

3.2.5 Algorithm

The Expectation Maximization algorithm (Dempster et al. [1977]) can be summarized as follows:

1. **Initialization:** Start with an initial parameter guess $\boldsymbol{\theta}_{cur}$.

2. **Expectation:** For each data point, draw samples from the posterior distribution over the latent variable and keep track of the log-likelihood in dependence of $\boldsymbol{\theta}$. This procedure gives $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_{cur})$.

3. **Maximization:** Maximization of $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_{cur})$ can be done exactly as for fully observed data. Repeat 2. and 3. until convergence.

It should be noted that even though the lower bound is typically convex, the original likelihood landscape rarely is. For that reason, EM does not necessarily find the global optimum of a likelihood function but may get stuck in local maxima.

3.3 Exponential Families

Among all possible distributions there is a certain class of distributions which have particularly appealing properties: the exponential class of density functions (Bishop [2006]; Bickel and Doksum [2001]). Many popular distributions fall into this class, like Gaussian and Poisson distributions. This section will introduce the concept of exponential families along with useful facts and relations relevant to this thesis.

3.3.1 Definition

An exponential family is a parametrized set of distributions which can written as[1],

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = h(\mathbf{x})g(\boldsymbol{\theta}) \quad \underbrace{\exp(\boldsymbol{\theta}^T T(\mathbf{x}))}_{\text{parameter-data interaction}} \quad , \tag{3.43}$$

where $T(\mathbf{x})$ is vector-valued and has the same dimension as $\boldsymbol{\theta}$. As indicated in Eq. 3.43, the property which distinguishes an exponential family from other families is a very specific and restricted form of interaction between parameters $\boldsymbol{\theta}$ and data $\mathbf{x}$. A more common but equivalent form of an exponential family is,

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = h(\mathbf{x}) \exp(\boldsymbol{\theta}^T T(\mathbf{x}) - A(\boldsymbol{\theta})). \tag{3.44}$$

The individual elements of Eq. 3.44 are explained below:

$\mathbf{x}$	...	Modeled random variables. The domain of an exponential family (all configurations for which the probability density is defined) is often restricted. There are families with discrete and continuous domain, and many are restricted to non-negative data.
$\boldsymbol{\theta}$	...	The M parameters of the family. They are usually restricted to some parameter space, in particular some subspace of $\mathbb{R}^M$.
$T(\mathbf{x})$	...	*Sufficient statistics* (must also be M-dimensional): they gate the interaction between parameters and data. Section 3.3.2 is dedicated to the meaning and importance of sufficient statistics.
$A(\boldsymbol{\theta})$	...	*Log-partition function*: this function ensures that the distribution is properly normalized for each allowed parameter set $\boldsymbol{\theta}$. It is of utmost importance for the analysis of an exponential family.
$h(\mathbf{x})$	...	May be considered the (unnormalized) base distribution of a family, since for $\boldsymbol{\theta} = \mathbf{0}$ one has $p_{\boldsymbol{\theta}}(\mathbf{x}) \propto h(\mathbf{x})$.

[1]In this thesis, only the canonical form will be used. For a more general definition, see Bickel and Doksum [2001].

3.3.2 Sufficient Statistics

A *statistic* is a measure or a number of measures computed from a set of data (Bickel and Doksum [2001]). It usually captures important characteristics of the data. Examples of statistics are the mean and variance of a set of observations. In the context of parametrized distributions, a *sufficient statistic* with respect to a certain parameter of the model, is a statistic which captures all relevant information for estimating that parameter from the data. Intuitively, a sufficient statistic may be seen as a concise summary of data, targeted at the estimation of a parameter. Formally, a statistic is sufficient with respect to a parameter θ_i if,

$$p(\mathbf{x}\,|T_i(\mathbf{x}), \theta_i) = p(\mathbf{x}\,|T_i(\mathbf{x})). \tag{3.45}$$

For a normal distribution with parameters $\boldsymbol{\theta} = (\mu, \sigma^2)^T$, the corresponding sufficient statistics are the mean and variance of a data set, since this information is all that can be extracted from the data with respect to these model parameters.

Exponential families play an outstanding role in the context of sufficient statistics. Consider a number of independent identically distributed observations from some family of distributions with unknown parameters $\boldsymbol{\theta}$. In general, the dimensionality of a sufficient statistic for these parameters may increase with the number of data samples. Among all families of distributions, only exponential families have the property that the dimension of the sufficient statistic does not increase with sample size. In practical terms this means that in exponential families, parameter estimation can be elegantly performed via a fixed number of sufficient statistics, whereas outside exponential families systematic parameter estimation is much more difficult and requires a different approach.

3.3.3 Variance Functions

In order to understand variance functions, it is useful to study a certain subclass of exponential families. A *natural exponential family* is an exponential family with the simplest possible sufficient statistic $T(\mathbf{x}) = \mathbf{x}$ (Bickel and Doksum [2001]). Hence a univariate, single-parameter natural exponential family can be written as,

$$p_\theta(x) = h(x)\exp(\theta^T x - A(\theta)). \tag{3.46}$$

Single-parameter natural exponential families are convenient to study for their simple form and many popular distribution families can be cast in this form. Just to name a few, the family of Gaussian distributions with fixed variance, the family of Poisson distributions, the family of Gamma distributions with fixed shape parameter, all can be parametrized to fit Eq. 3.46.

Recall that each parameter value θ defines a different distribution, with potentially different mean μ_θ and variance σ_θ^2. The dependence of mean and variance on θ is strictly determined by the specific model, in particular by the log-partition function $A(\theta)$ (Bickel and Doksum [2001]):

$$\mu_\theta = A'(\theta), \tag{3.47}$$

$$\sigma_\theta^2 = A''(\theta). \tag{3.48}$$

In general, the mean μ_θ can be shown to be strictly monotonic increasing with θ, such that an inverse always exists. The inverse function,

$$\theta_\mu = \Theta(\mu) = (A')^{-1}(\mu), \tag{3.49}$$

is useful for recovering the parameter associated with a given mean.

A *variance function* is a function which relates variance and mean in a family of distributions. In single-parameter natural exponential families, the variance function can be constructed from the log-partition function $A(\theta)$

via (Morris [1982]),

$$V(\mu) = A''\left(\Theta(\mu)\right). \tag{3.50}$$

Hence it is clear that each family has a unique variance function associated with it. Moreover, this association is bijective: the variance function uniquely characterizes the family (see for example David R. Clark [2004]). In fact, for a given variance function one can systematically construct the corresponding log-partition function $A(\theta)$. First, $\Theta(\mu)$ can be computed by noting that,

$$\Theta'(\mu) = \frac{\partial\theta}{\partial\mu} = \frac{1}{\frac{\partial\mu}{\partial\theta}} = \frac{1}{A''(\Theta(\mu))} = \frac{1}{V(\mu)}, \tag{3.51}$$

and therefore,

$$\Theta(\mu) = \int \frac{1}{V(\mu)} d\mu + C. \tag{3.52}$$

Since $V(\mu)$ is strictly positive, $\Theta(\mu)$ will be strictly monotone increasing and thus also possess a unique inverse, which must be $A'(\theta)$ according to Eq. 3.49, from which in turn one can derive $A(\theta)$ by integration. The integration in Eq. 3.52 adds a degree of freedom to the choice of $\Theta(\mu)$ and hence also $A(\theta)$, which can be easily interpreted as a constant parameter shift that can be chosen freely. The relation between $V(\mu)$ and $A(\theta)$ is summarized below:

$$A(\theta) \xrightarrow{\text{diff.}} A'(\theta) \xrightarrow{\text{inv.}} \Theta(\mu) \xrightarrow{\text{diff.}} \frac{1}{V(\mu)} \xrightarrow{\frac{1}{\cdot}} V(\mu) \tag{3.53}$$

$$A(\theta) \xleftarrow[\text{int.}]{} A'(\theta) \xleftarrow[\text{inv.}]{} \Theta(\mu) \xleftarrow[\text{int.}]{} \frac{1}{V(\mu)} \xleftarrow[\frac{1}{\cdot}]{} V(\mu) \tag{3.54}$$

Note that not all variance functions have a proper probability distribution associated with them. The variance function $V(\mu) = \sqrt{\mu}$, for instance, is not associated with any natural exponential family (Tweedie [1984]). Thus, a reconstructed log-partition function $A(\theta)$ for an improper variance function does not correspond to any normalized family of distributions according to Eq. 3.46.

Chapter 4

Expectation Maximization in Spiking Neural Networks

Expectation Maximization is one of the most powerful algorithms for training generative models (see Section 3.2). The challenge of transferring the algorithm to the domain of Spiking Neural Networks is to identify neural mechanisms which correspond to the two main ingredients of Expectation Maximization, namely inference and learning. In the original part of this document, a method will be presented which addresses these two issues. In this chapter, related work in this context will be introduced. Of immediate relevance to this thesis is a recent publication by Nessler et al. [2010], which consider a neural implementation of a mixture model with observed categorical variables. Importantly, the authors succeed in relating learning and inference in this model to neural integration and plasticity. A slightly different approach is taken by Deneve [2008a,b], where a Hidden Markov Model with binary variables is mapped to the spiking neural domain.

4.1 Mixture of Categoricals

In this section, the work by Nessler et al. [2010] will briefly be discussed where a mixture model with M observed categorical variables $\mathbf{x} = (x_1, \dots, x_M)^T$ is considered. A categorical random variable x can take on values from a limited discrete set of size C[1]. For simplicity, one may assume values that range from $1 \ldots C$. The distribution of such a random variable can be described by C probabilities which must sum to one:

$$\sum_{c=1}^{C} p(x = c) = \sum_{c=1}^{C} \pi_c = 1, \tag{4.1}$$

where π_c is by definition the probability of x taking on the value c. In a typical model, the π_c would constitute the parameters of the distribution.

The latent variable z in the mixture model may also be seen as a categorical variable with K different states. The underlying generative model of the approach is based on Naïve Bayes independence assumptions (see Section 3.1.8):

$$\begin{aligned} p(x_1, \ldots, x_M, z) &= p(z)p(x_1, x_2|z) &(4.2)\\ &= p(z)p(x_1|z) \cdots p(x_M|z). &(4.3) \end{aligned}$$

Below, the mapping of this generative model to the neural domain is explicated. Also, the most important limitations of the approach will be discussed as they are significant for motivating the original part of this thesis.

4.1.1 Mapping between random variables and spiking neurons

The connection between theoretical input variables and spiking neurons is established in two steps. First, a single categorical random variable is expanded into C binary variables. At any time, exactly one of these C binary variables may be active such that each categorical value is encoded by the activity of one binary variable. The authors call this transformation population coding since a single categorical variable is represented by an ensemble of C binary variables. These binary variables are denoted by $\mathbf{y} = (y_1, \ldots, y_N)^T$, where

[1]Original notation has been slightly adjusted.

with $N = C \cdot M$, taking account of M variables with C possible values each. The second step relates the binary variables to spiking neurons by introducing one neuron for each binary variable and defining that an active binary state is reflected by random firing at some fixed rate λ, while the inactive binary state is encoded by the absence of firing [2].

In an analogous fashion, the latent variable is also brought in relation to a set of K spiking neurons, each of which encodes a specific value of the latent variable. Again, only one of these variables may be active at a time. However, as opposed to the spiking input neurons, the latent variable takes on a value only at certain points in time, in particular when the network performs inference. Then, exactly one spike is emitted by the neuron which corresponds to the current value of the latent variable.

4.1.2 Inference

The parameters of the generative model are the log-prior probabilities of the latent variable z,

$$w_{k0} = \log p(z_k = 1), \tag{4.4}$$

and the conditional log-probabilities of input variables y_i given a specific latent value,

$$w_{ki} = \log p(y_i = 1 | z_k = 1). \tag{4.5}$$

The parameters are subject to several constraints, namely,

$$\sum_{k=1}^{K} \pi_k = 1, \tag{4.6}$$

[2]Firing occurs according to a homogeneous Poisson process at the given rate.

and for each group $\mathcal{G}_j$ of variables y_i which code for the same categorical variable x_j,

$$\sum_{i \in \mathcal{G}_j} \pi_{ki} = 1, \quad k = 1 \dots K;\ j = 1 \dots M \tag{4.7}$$

This choice of this parametrization, in particular of using log-probabilities instead of probabilities, is justified by noting that inference becomes very straightforward with these parameters:

$$u_k = \sum_i w_{ki} y_i + w_{k0}, \tag{4.8}$$

$$p(z_k = 1 |\, \mathbf{y}) = \frac{\exp(u_k)}{\sum_{k'=1}^{K} \exp(u_{k'})}. \tag{4.9}$$

In fact, the operation defined by Eq. 4.8 and 4.9 can be mapped to neural dynamics in a straightforward fashion:

- The weight w_{ki} corresponds to the synaptic efficacy between y_i and z_k.
- The computation of u_k corresponds to the integration of current input via the synaptic weights. u_k may be considered the (uninhibited) membrane potential of neuron k.
- The divisive normalization in Eq. 4.9 corresponds to a lateral inhibition mechanism among the output neurons z_k, implementing a Winner-Take-All (WTA) operation, as discussed in Section 2.4.2.

4.1.3 Learning

Maximum likelihood learning in the model is achieved by local synaptic plasticity rules:

$$\Delta w_{ki} = \begin{cases} \eta(\exp(-w_{ki}) - 1), & \text{if } y_i = 1 \text{ and } z_k = 1 \\ -\eta, & \text{if } y_i = 0 \text{ and } z_k = 1 \\ 0, & \text{if } z_k = 0 \end{cases} \tag{4.10}$$

where η is a learning rate. A similar rule may be used for learning the log-priors w_{k0}.

It can be shown that this learning rule converges to the optimal values in a fully observed setting. The concurrent use of inference and learning results in an online EM approach for which the authors can prove that the expected learning update after a combined inference and learning step always increases the log-likelihood of the input distribution under some mild conditions.

4.1.4 Results and Limitations

The method proposed in Nessler et al. [2010] appears to work well in practice and can cope with high-dimensional data and noisy inputs. To demonstrate the properties of the algorithm, an application on the MNIST dataset was shown. The database, which consists of small pictures of handwritten digits labeled by their digit (Lecun and Cortes [1998]), is usually used as a standard benchmark test for supervised learning. In order to test performance in the unsupervised setting, the authors employ the conditional entropy of the labels given the network output as a measure for discrimination performance. They further show that the model is able to detect changes in the input distribution and adapt its connectivity accordingly. On a reduced subset of the standard database comprising different writings of the digits 0, 3 and 4, the method achieves a classification error of 3.68% on an independent test set.

The method may be considered an early prototype of a new class of spiking neural network architectures which succeed in relating time-tested machine learning algorithms to neural activity and plasticity. As such, it naturally has a few limitations and imperfections:

Implementation of WTA: The authors do not provide a neural mechanism for computing the WTA response. The hypothesis that such a network may be implemented neurally, in particular in context of the proposed method, has yet to be confirmed.

Categorical variables: The method is limited to categorical input variables. In the context of spiking neurons, it would seem more natural to consider Poisson distributed inputs (Gerstner and Kistler [2002]).

Population coding: The requirements on the input coding seem overly strict and unrealistic in a biological context. Demanding that there are subsets of input neurons which encode underlying variables as described above in Section 4.1.1 seems unrealistic.

EPSP shapes: The proposed theory is only consistent when rectangular EPSP shapes are used. Biological neurons are known to have smoothly decaying impulse responses (see for instance the Leaky-Integrate-Fire model discussed in Section 2.2.1).

4.2 Binary Hidden Markov Model

In Deneve [2008a], the authors present a principled approach how a single neuron can perform Bayesian inference based on evidence from incoming spike trains. First, in an implicit space, a temporal generative model is constructed which relates binary input variables $\mathbf{s}_t = (s_t^1, \ldots, s_t^N)^T$ with a binary hidden state x_t. The model has temporal dynamics and may be seen as a binary instance of a Hidden Markov Model. Inference in this Hidden Markov model can be achieved by a recurrent process in time. Note that in a binary latent variable model, inference is equivalent to keeping track of the log-odds ratio $L_t = \log \frac{p(x_t=1|\,\mathbf{s}_{0\to t})}{p(x_t=0|\,\mathbf{s}_{0\to t})}$, a quantity which captures the tendency of explaining the input history with hidden state $x_t = 1$ rather than $x_t = 0$.

In the explicit neural space, the input $\mathbf{s}_t$ corresponds to a binary vector indicating which synapses were activated between time t and $t + dt$. The authors then show that the computation of the log-odds ratio L_t can be mapped to a differential equation which contains terms resembling the integration of input spikes and decay in a neuron's membrane potential:

$$\dot{L}_t = -\phi(L_t) + \sum_i w_i \delta(s_t^i - 1). \tag{4.11}$$

The neuron uses this information to emit spikes according to a predictive coding paradigm, where spikes are only generated when new information has been accumulated at the neuron which could not be predicted from the prior information. In order to be able to perform this coding scheme, they postulate that a neuron must constantly try to estimate its own state from its output spike train in order to judge how accurately another neuron could infer this state. Whenever this estimate deviates too greatly from the real state, a neuron must emit spikes to correct for the error.

The scheme allows for building hierarchies of such neurons, since the output coding is approximately equal to the input coding, a fact which is experimentally verified in Deneve [2008b]. However, belief propagation in these hierarchies is not correctly implemented by the proposed network in a strict sense, due to the lack of top-down feedback.

The parameters of the model can be learned in an online EM algorithm with Spike-Timing Dependent Plasticity. The details of the learning rule involve estimating sufficient statistics via running averages and computing functions of these sufficient statistics to update the actual parameters of the model. Note that this mechanism is somewhat unwieldy in the context of synaptic plasticity, especially when compared to the plasticity rules emerging from the previously discussed model (Section 4.1). The postulated learning scheme involves non-standard computations and therefore constitutes a major issue for implementing this method in standard neural network simulators.

Chapter 5

A Generalization to Exponential Families

In this chapter and the remainder of this work, the original part of this thesis will be presented and discussed, which will build on the concepts and formalisms discussed in the introductory part of this document. In particular, in Section 4.1, a recent publication by Nessler et al. [2010] was summarized which will provide the basis and source of inspiration for the remainder of this work. From a theoretical perspective, the most striking aspect in their approach is the establishment of a link across traditional discipline boundaries. In a simple neural architecture, they are able to relate one of the most powerful algorithms in machine learning, Expectation Maximization (see Section 3.2), to neural activity and plasticity. Its formal simplicity, which is evident in implementations of both inference and learning, makes it an attractive starting point for further investigation and improvement.

Section 5.1 is intended to provide motivation for a new theory which is then introduced in Section 5.2 and 5.2.2.

5.1 Motivation

In order to motivate the improvements which will be proposed shortly in this chapter, consider two particular limitations of Nessler et al. [2010]. First, the restriction to categorical input values, which prevents the method from being able to read, learn and reason based on continuous input firing rates. Second, the related problem of population coding. In Nessler et al. [2010], inputs have to fulfill certain requirements in order for the theory to work. It is assumed that inputs are divided in disjoint groups, and in each of these groups there may be precisely one active neuron at a time. While at first this may seem tolerable in an algorithmic setting, it is quite hard to argue in favor of such restrictions in biological networks, since precision seems particularly hard to achieve in single neurons (see for example, London et al. [2010]). Even in purely technical applications, there are practical issues with population coding. Note that usually, input data is not supplied in a form that satisfies the demanded constraints. Consequently, one has to convert data into the required format before the method can be applied. In the worst case, one has to construct a mirror neuron for each single input. The original input and the mirror neuron together could then satisfy the required group constraint. It is clear that such an overhead may severely hamper performance and is highly undesirable in real-time online processing tasks. The performance loss is particularly striking when data is already provided in a spiking protocol with sparse activity. In such a situation, the introduction of mirror neurons which always fire when the corresponding input is silent, drastically increases input bandwidth and computational demands.

Note that both issues are related to the particular choice of component distribution, namely the categorical distribution. Therefore, below a generalization of the method to exponential family components will be elaborated.

5.2 Recursive Estimation in Mixtures of Exponential Families

The new theory should preserve three critical properties of the original method. First, the parameters of the probabilistic model should directly correspond to synaptic weights. This is important in order to be able to map learning in the probabilistic model to synaptic plasticity. Second, inference should be tractable, and the interaction between input data and synaptic weights should be a weighted sum of the form $\mathbf{w}^T \mathbf{x}$. This is of relevance, since neurons may be viewed, to a first approximation, as integrators of their weighted inputs (see Section 2.2.1). Third, learning rules should be local. Only four components must enter the weight change of a synapse: pre-synaptic spikes, post-synaptic spikes, current weight, and learning rate. The model which is proposed below fulfills all requirements.

5.2.1 Mixtures of Exponential Families

Consider a generative model of the following form:

$$p(\mathbf{x}) = \sum_{k=1}^{N} p(z=k) p(\mathbf{x}\,|z=k, \boldsymbol{\theta}_k) \tag{5.1}$$

$$= \sum_{k=1}^{N} \exp(\pi_k) p(\mathbf{x}\,|z=k, \boldsymbol{\theta}_k). \tag{5.2}$$

with $\boldsymbol{\pi} = (\pi_1, \ldots \pi_N)^T$, constrained by $\sum_k \exp(\pi_k) = 1$. This defines a general parametric mixture model with "class" probabilities $p(z = k) = \exp(\pi_k)$ and component parameters $\boldsymbol{\theta}_k$. Here, the focus is on mixtures of exponential-family type distributions, such that the k-th component density can be written as

$$\begin{aligned} &p(\mathbf{x}\,|z = k, \boldsymbol{\theta}_k) = \\ &\quad h(\mathbf{x}) \exp\left(\sum_{j=1}^{M} \theta_{kj} T_j(\mathbf{x}) - A(\boldsymbol{\theta}_k)\right), \end{aligned}$$

parametrized by $\boldsymbol{\theta}_k = (\theta_{k1}, \ldots, \theta_{kM})^T$. The functions $T_j(\cdot)$ are called the sufficient statistics of $\mathbf{x}$.

For convenience, define

$$\begin{aligned} \mathbf{y} &= (T_1(\mathbf{x}), \ldots, T_M(\mathbf{x}))^T, \\ u_k(\mathbf{y}) &= \pi_k + \boldsymbol{\theta}_k^T\, \mathbf{y} - A(\boldsymbol{\theta}_k), \end{aligned}$$

and

$$u(\mathbf{y}, z) = \begin{cases} u_1(\mathbf{y}) & \text{, if } z = 1, \\ \vdots & \\ u_N(\mathbf{y}) & \text{, if } z = N. \end{cases}$$

Then the joint probability of $\mathbf{x}$ and z can be written as

$$p(\mathbf{x}, z|\, \boldsymbol{\pi}, \boldsymbol{\theta}) = h(\mathbf{x}) \exp(u(\mathbf{y}, z)) \tag{5.3}$$

with $\mathbf{y}$ as defined above.

Inference in such a model is easy:

$$\begin{aligned} p(z = k|\, \mathbf{x}, \boldsymbol{\pi}, \boldsymbol{\theta}) &= \frac{p(z = k, \mathbf{x}\,|\, \boldsymbol{\pi}, \boldsymbol{\theta}_k)}{p(\mathbf{x}\,|\, \boldsymbol{\pi}, \boldsymbol{\theta})} \\ &= \frac{p(z = k, \mathbf{x}\,|\, \boldsymbol{\pi}, \boldsymbol{\theta}_k)}{\sum_{k'} p(z = k', \mathbf{x}\,|\, \boldsymbol{\pi}, \boldsymbol{\theta}_{k'})} \\ &= \frac{h(\mathbf{x}) \exp(u_k(\mathbf{y}))}{\sum_{k'} h(\mathbf{x}) \exp(u_{k'}(\mathbf{y}))} \\ &= \frac{\exp(u_k(\mathbf{y}))}{\sum_{k'} \exp(u_{k'}(\mathbf{y}))}. \end{aligned} \tag{5.4}$$

Note how the parameter independent factor $h(\mathbf{x})$ is canceled. This will be of major importance later when a neural implementation for inference with this model is considered.

Learning also turns out straightforward. Given fully observed data, that is a number of independent samples $\mathbf{x}^{(t)}$ and corresponding latent variables $z^{(t)}$, one can write the joint-probability of the data as

$$\begin{aligned} p(\mathbf{x}^{(1)}, z^{(1)}, \dots, \mathbf{x}^{(T)}, z^{(T)} | \, \boldsymbol{\pi}, \boldsymbol{\theta}) = \\ \prod_{t=1}^{N} p(\mathbf{x}^{(t)}, z^{(t)} | \, \boldsymbol{\pi}, \boldsymbol{\theta}) \end{aligned}$$

Maximizing this quantity w.r.t. $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$ is equivalent to maximizing

$$1/T \sum_{t=1}^{T} \log p(\mathbf{x}^{(t)}, z^{(t)} | \, \boldsymbol{\pi}, \boldsymbol{\theta})$$

or by using Eq. 5.3 also equivalent to maximizing

$$\begin{aligned} \langle u(\mathbf{y}, z) \rangle_{p(\mathbf{X}, z)} &= \\ \Big\langle \langle u(\mathbf{y}, z) \rangle_{p(\mathbf{X} \,|z)} \Big\rangle_{p(z)} &= \\ \sum_{k=1}^{N} p(z = k) \langle u_k(\mathbf{y}) \rangle_{p(\mathbf{X} \,|z=k)} &= \\ \sum_{k=1}^{N} p(z = k) u_k(\langle \mathbf{y} \rangle_{p(\mathbf{X} \,|z=k)}). \end{aligned}$$

Here, $\langle \cdot \rangle_{p(\mathbf{X}, z)}$ denotes the expectation operator w.r.t. the sample distribution, and $\langle \cdot \rangle_{p(\mathbf{X} \,|z=k)}$ the expectation conditioned on a certain value for z.

In order to account for the normalization constraint on $\boldsymbol{\pi}$ one can define the Lagrangian

$$\begin{aligned} L(\boldsymbol{\pi}, \boldsymbol{\theta}, \lambda) = & \sum_{k=1}^{N} p(z = k) u_k(\langle \mathbf{y} \rangle_{p(\mathbf{X} \,|z=k)}) \\ & + \lambda (\sum_{k} \exp(\pi_k) - 1) \end{aligned}$$

and find critical points by requiring[1]

$$\nabla L(\boldsymbol{\pi}, \boldsymbol{\theta}, \lambda) \overset{!}{=} \mathbf{0}.$$

This gives the expected result for π (compare this to the model definition in Eq. 5.1 and 5.2):

$$\pi_k^* = \log p(z = k)$$

For $\boldsymbol{\theta}$, the result depends on the particular choice of $A(\cdot)$ and $T_j(\cdot)$ through the following relation which must hold for all k and j considered:

$$\frac{\partial A(\boldsymbol{\theta}_k)}{\partial \theta_{kj}} \overset{!}{=} \langle T_j(\mathbf{x}) \rangle_{p(\mathbf{X}\,|z=k)} \tag{5.5}$$

Example: Single Poisson variable

Suppose x is distributed according to

$$p(x|z = k, \lambda_k) = \frac{\lambda_k^x e^{-\lambda_k}}{x!},$$

for non-negative integers x. One can identify the exponential form by rearranging

$$\begin{aligned} p(x|z = k, \lambda_k) &= \frac{\lambda^x e^{-\lambda_k}}{x!} \\ &= \frac{1}{x!} e^{\log(\lambda_k)x - \lambda_k} \\ &= \frac{1}{x!} e^{\theta_k x - \exp(\theta_k)} \end{aligned}$$

[1]For notational simplicity the nabla-operator will be used as the column vector of partial derivatives, contrary to common notation.

such that

$$\begin{aligned}\theta_k = \log \lambda_k, \quad h(x) = \frac{1}{x!},\\ y = T(x) = x, \quad A(\theta_k) = \exp(\theta_k).\end{aligned}$$

Then, Eq. 5.5 simplifies to

$$\begin{aligned}\exp(\theta_k) &\overset{!}{=} \langle x \rangle_{p(x|z=k)}\\ \theta_k^* &= \log(\langle x \rangle_{p(x|z=k)})\end{aligned}$$

Example: Conditionally independent Poisson variables

Consider the following distribution:

$$\begin{aligned}p(\mathbf{x}\,|z = k, \lambda_k) = \prod_j \frac{{\lambda_{kj}}^{x_j} e^{-\lambda_{kj}}}{x_j!} =\\ \left(\prod_j \frac{1}{x_j!}\right) e^{\sum_j \theta_{kj} x_j - \sum_j \exp(\theta_{kj})}\end{aligned}$$

with

$$\begin{aligned}\theta_{kj} = \log \lambda_{kj}, \quad h(x) = \prod_i \frac{1}{x_i!},\\ y_j = T_j(\mathbf{x}) = x_j, \quad A(\boldsymbol{\theta}_k) = \sum_j \exp(\theta_{kj}).\end{aligned}$$

Thanks to the factorized form, the derivatives of $A(\cdot)$ are very simple and Eq. 5.5 becomes

$$\begin{aligned}\exp(\theta_{kj}) &\overset{!}{=} \langle x_j \rangle_{p(x|z=k)},\\ \theta_{kj}^* &= \log(\langle x_j \rangle_{p(x|z=k)}).\end{aligned}$$

5.2.2 A Recursive Maximum Likelihood (ML) Approach

Again, it is assumed that access to full observations $(\mathbf{x}^{(t)}, z^{(t)})$ is provided. However, in contrast to standard batch learning according to Eq. 5.5, in neural networks, parameters should be updated incrementally while data is observed. Furthermore the update at time step t should depend only on the current data $(\mathbf{x}^{(t)}, z^{(t)})$ and the parameters from the last step $(\boldsymbol{\pi}^{(t-1)}, \boldsymbol{\theta}^{(t-1)})$.

First, let $\hat{\mathrm{p}}_k^{(t)}$ and $\hat{\mathbf{y}}_k^{(t)}$ be the empirical estimates of $p(z = k)$ and $\langle \mathbf{y} \rangle_{p(\mathbf{x}\,|z=k)}$, respectively, based on the observations up to time step t:

$$\begin{aligned} \hat{\mathrm{p}}_k^{(t)} &= \frac{1}{t} \sum_{t'=1}^{t} \delta(z^{(t')} - k), \\ \hat{\mathbf{y}}_k^{(t)} &= \frac{\sum_{t'=1}^{t} \delta(z^{(t')} - k)\, \mathbf{y}^{(t')}}{\sum_{t'=1}^{t} \delta(z^{(t')} - k)} \end{aligned}$$

with $\delta(\cdot)$ being the Dirac delta function. Now suppose that the optimal parameters up to time step t have already been computed, that is for all k,

$$\begin{aligned} \pi_k^{(t)} &= \log \hat{\mathrm{p}}_k^{(t)}, \\ \nabla A(\boldsymbol{\theta}_k^{(t)}) &= \hat{\mathbf{y}}_k^{(t)}\,. \end{aligned} \tag{5.6}$$

For the next time step $t + 1$ with $\eta = \frac{1}{t+1}$, one has,

$$\begin{aligned} \pi_k^{(t+1)} &= \log \hat{\mathrm{p}}_k^{(t+1)} \\ &= \log \Big((1 - \eta)\, \hat{\mathrm{p}}_k^{(t)} + \eta \delta(z^{(t+1)} - k) \Big) \\ &= \log \Big(\hat{\mathrm{p}}_k^{(t)} + \eta [\delta(z^{(t+1)} - k) - \hat{\mathrm{p}}_k^{(t)}] \Big) \\ &\approx \log \hat{\mathrm{p}}_k^{(t)} + \eta \frac{\delta(z^{(t+1)} - k) - \hat{\mathrm{p}}_k^{(t)}}{\hat{\mathrm{p}}_k^{(t)}} && (5.7) \\ &= \pi_k^{(t)} + \eta (e^{-\pi_k^{(t)}} \delta(z^{(t+1)} - k) - 1) \\ &= \pi_k^{(t)} + \eta \Delta_{\pi_k}^{(t+1)} && (5.8) \end{aligned}$$

with

$$\eta\Delta_{\pi_k}^{(t+1)} = \begin{cases} \eta(e^{-\pi_k^{(t)}} - 1) & \text{if } z^{(t+1)} = k, \\ -\eta & \text{otherwise.} \end{cases} \tag{5.9}$$

The approximation in Eq. 5.7 is valid for small η.

In order to obtain a recursive formula for $\boldsymbol{\theta}$, note that at time step $t+1$ the conditional expectation on the right-hand side of Eq. 5.6 changes only for the active hidden value $k = z^{(t+1)}$. Therefore one can immediately write

$$\boldsymbol{\theta}_{k'}^{(t+1)} = \boldsymbol{\theta}_{k'}^{(t)}, \quad \forall k' \in \{1 \dots N\} \backslash \{k\}. \tag{5.10}$$

For $k = z^{(t+1)}$ an update of the form $\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} + \eta\Delta_{\boldsymbol{\theta}_k}^{(t+1)}$ is desired, and can be plugged in to give:

$$\begin{aligned} \nabla A(\boldsymbol{\theta}_k^{(t)} + \eta\Delta_{\boldsymbol{\theta}_k}^{(t+1)}) &= \hat{\mathbf{y}}_k^{(t+1)} \\ &= (1-\eta)\,\hat{\mathbf{y}}_k^{(t)} + \eta\,\mathbf{y}^{(t+1)} \\ &= \hat{\mathbf{y}}_k^{(t)} + \eta[\mathbf{y}^{(t+1)} - \hat{\mathbf{y}}_k^{(t)}] \\ &= \nabla A(\boldsymbol{\theta}_k^{(t)}) + \eta[\mathbf{y}^{(t+1)} - \nabla A(\boldsymbol{\theta}_k^{(t)})] \end{aligned}$$

Now, assuming small η one may approximate the left-hand side by a first-order Taylor series

$$\nabla A(\boldsymbol{\theta}_k^{(t)} + \eta\Delta_{\boldsymbol{\theta}_k}^{(t+1)}) \approx \nabla A(\boldsymbol{\theta}_k^{(t)}) + \eta H_A(\boldsymbol{\theta}_k^{(t)})\Delta_{\boldsymbol{\theta}_k}^{(t+1)},$$

and obtain

$$\eta\Delta_{\boldsymbol{\theta}_k}^{(t+1)} = \eta H_A^{-1}(\boldsymbol{\theta}_k^{(t)})[\mathbf{y}^{(t+1)} - \nabla A(\boldsymbol{\theta}_k^{(t)})], \tag{5.11}$$

where $H_A(\cdot)$ is the Hessian of $A(\cdot)$ w.r.t. $\boldsymbol{\theta}_k$.

In summary, Eq. 5.9, 5.10 and 5.11 define an approximate recursive estimation procedure for mixtures of exponential-family type distributions. Only Eq. 5.11 depends on the particular choice of distribution.

Example: Conditionally independent Poisson variables

Using from 5.2.1

$$\begin{aligned}\theta_{kj} = \log \lambda_{kj}, \quad h(x) = \prod_i \frac{1}{x_i!},\\ y_j = T_j(\mathbf{x}) = x_j, \quad A(\boldsymbol{\theta}_k) = \sum_j \exp(\theta_{kj}).\end{aligned}$$

A particularly convenient gradient and Hessian of $A(\cdot)$ is found,

$$\begin{aligned}\nabla A(\boldsymbol{\theta}_k^{(t)}) &= (\exp(\theta_{k1}), \ldots, \exp(\theta_{kM}))^T,\\ H_A(\boldsymbol{\theta}_k^{(t)}) &= \operatorname{diag}(\exp(\theta_{k1}), \ldots, \exp(\theta_{kM})).\end{aligned}$$

Plugging this into Eq. 5.11 yields

$$\begin{aligned}\eta \Delta_{\boldsymbol{\theta}_k}^{(t+1)} &= \eta \operatorname{diag}(e^{-\theta_{k1}}, \ldots, e^{-\theta_{kM}})[\mathbf{y}^{(t+1)} - (e^{\theta_{k1}}, \ldots, e^{\theta_{kM}})^T]\\ &= \eta[(e^{-\theta_{k1}} y_1^{(t+1)}, \ldots, e^{-\theta_{kM}} y_M^{(t+1)})^T - \mathbf{1}]\end{aligned}$$

or, more explicitly,

$$\theta_{kj}^{(t+1)} = \theta_{kj}^{(t)} + \begin{cases} \eta(e^{-\theta_{kj}} x_j^{(t+1)} - 1) & \text{if } k = z^{(t+1)},\\ 0 & \text{otherwise.}\end{cases}$$

5.3 Learning Rules for Some Exponential Families

General learning and inference equations for mixtures of exponential families in a neural network setting have been derived. Here, a selection of exponential families is compared by their resulting learning and inference equations. The Poisson special case will be discussed in greater detail in section 5.5.

5.3.1 Derivation of Learning Rules for Single-Parameter Families

Here, it will be assumed that inputs x_i are conditionally independent and that the same single-parameter exponential family model can be used for all inputs (except for the multinomial distribution, see 5.3.2). Furthermore, only exponential families are covered, for which the single sufficient statistics is proportional to the mean of the distribution, that is $T(x) = x/\phi$ with some constant ϕ which will be called dispersion factor. Then, one can write

$$p(\mathbf{x}\,|z_k = 1, \boldsymbol{\theta}_k) = [\prod_j h(x_j)] \exp\left(\frac{\boldsymbol{\theta}_k^T \mathbf{x} - \sum_j A(\boldsymbol{\theta}_{kj})}{\phi}\right)$$

For conditionally independent inputs, learning rules can also be studied in a simplified single input, single parameter scenario since the derivative of the log-likelihood is decomposed when Naïve Bayes assumptions apply (see Section 3.1.8). Consider the simplified scenario,

$$p(x|\theta) = h(x) \exp\left(\frac{\theta x - A(\theta)}{\phi}\right). \tag{5.12}$$

The normalization or log-partition function $A(\theta)$ ensures that $p(x|\theta)$ is properly normalized for all possible θ. It depends on ϕ, $h(x)$ and the support $\mathcal{S}_X$ through the following relation[2],

$$A(\theta) = \phi \log \int_{x \in \mathcal{S}_X} h(x) \exp\left(\frac{\theta x}{\phi}\right) dx. \tag{5.13}$$

The log-likelihood of an ensemble of T observations $x_1, \ldots, x_T$ is given by:

$$\log p(x_1, \ldots, x_T|\theta) = \frac{1}{\phi} \sum_{t=1}^{T} \left(\log h(x_t) + \theta x_t - A(\theta)\right). \tag{5.14}$$

[2]For discrete distributions, the integral is replaced by a sum.

A maximum likelihood objective function $L_T(\theta)$ up to time step T is defined by dropping terms that are independent of θ and multiplying by ϕ/T:

$$L_T(\theta) = \frac{1}{T}\sum_{t=1}^{T}(\theta x_t - A(\theta)) \tag{5.15}$$

$$= \theta\langle x\rangle_T - A(\theta), \tag{5.16}$$

where $\langle x\rangle_T$ denotes the temporal average over samples $x_1, \ldots, x_T$. Then the maximizing θ fulfills

$$A'(\theta) = \langle x\rangle_T. \tag{5.17}$$

In order to understand the meaning of Eq. 5.17, note that $A'(\theta)$ is simply the mean of the model distribution with parameter θ:

$$A'(\theta) = \frac{\partial\left[\phi \log \int_{x\in\mathcal{S}_X} h(x)\exp\left(\frac{\theta x}{\phi}\right)dx\right]}{\partial\theta} \tag{5.18}$$

$$= \phi\frac{\partial\left[\int_{x\in\mathcal{S}_X} h(x)\exp\left(\frac{\theta x}{\phi}\right)dx\right]}{\partial\theta}\cdot\frac{1}{\int_{x\in\mathcal{S}_X} h(x)\exp\left(\frac{\theta x}{\phi}\right)dx} \tag{5.19}$$

$$= \phi\int_{x\in\mathcal{S}_X} h(x)\frac{\partial\left[\exp\left(\frac{\theta x}{\phi}\right)\right]}{\partial\theta}dx\cdot\exp\left(-\frac{A(\theta)}{\phi}\right) \tag{5.20}$$

$$= \phi\int_{x\in\mathcal{S}_X} h(x)\exp\left(\frac{\theta x}{\phi}\right)\frac{x}{\phi}dx\cdot\exp\left(-\frac{A(\theta)}{\phi}\right) \tag{5.21}$$

$$= \int_{x\in\mathcal{S}_X} h(x)\exp\left(\frac{\theta x - A(\theta)}{\phi}\right)x\ dx \tag{5.22}$$

$$= \langle x\rangle_{p(x|\theta)} := \mu(\theta) \tag{5.23}$$

Eq. 5.17 therefore simply states that θ should be chosen such that the model's mean and the observed mean match. Note that it does not directly give an explicit solution for the parameter θ, unless one can invert $A'(\theta)$ (see Section 3.3.3 for more on that). Instead, as has been shown alrady, a recursive learning rule can be derived without having to invert $A'(\theta)$.

Assume that θ is the current maximum likelihood estimate up to time

$T-1$, such that

$$A'(\theta) = \langle x \rangle_{T-1}.$$

Adding another observation generally leads to a different estimate, $\theta_{new} = \theta + \Delta_\theta$, for which holds,

$$\begin{aligned} A'(\theta + \Delta_\theta) &= \langle x \rangle_T \\ &= (1-\eta)\langle x \rangle_{T-1} + \eta x_T \\ &= (1-\eta)A'(\theta) + \eta x_T, \end{aligned}$$

with $\eta = \frac{1}{T}$. A first order Taylor approximation of the left-hand side yields

$$\begin{aligned} A'(\theta) + \Delta_\theta A''(\theta) &= (1-\eta)A'(\theta) + \eta x_T, \\ \Delta_\theta &= \eta \frac{x_T - A'(\theta)}{A''(\theta)} \end{aligned} \tag{5.24}$$

In the following, Eq. 5.24 will be used to show how various exponential families differ in their learning rules due to different normalization functions $A(\theta)$.

5.3.2 Multinomial Distribution with n Trials

The above derivation applies to most distributions that will be considered. The fact that one can model a multivariate distribution by independently modeling each input makes the method quite elegant and attractive.

The multinomial distribution, however, is a multivariate distribution that imposes additional constraints on parameters across inputs, and thus requires some extra treatment. For a multinomial distribution with n trials, defining $\theta_i = \log(n\, p_i)$ one has,

$$\begin{aligned} p(\mathbf{x} \,|\, \boldsymbol{\theta}) &= \frac{n!}{\prod_i x_i!} e^{\sum_i (\theta_i - \log n) x_i} \\ &= \left[\frac{n!}{\prod_i x_i!} n^{-\sum_i x_i} \right] e^{\sum_i \theta_i x_i} \end{aligned}$$

for $\sum_i x_i = n$, constrained by $\sum_i \exp(\theta_i) = n$. This results in a constrained maximum likelihood optimization problem, for which the Lagrangian can be defined,

$$L_1(\boldsymbol{\theta}, \lambda) = \sum_i \theta_i \langle x_i \rangle_T + \lambda(n - \sum_i \exp(\theta_i)),$$

Then one obtains

$$\begin{aligned}\frac{\partial L_1(\boldsymbol{\theta}, \lambda)}{\partial \theta_i} &= \langle x_i \rangle_T - \lambda \exp(\theta_i) \overset{!}{=} 0,\\ \sum_i \langle x_i \rangle_T - \lambda &\underbrace{\sum_i \exp(\theta_i)}_{\overset{!}{=} n} \overset{!}{=} 0\end{aligned}$$

For well-behaved input, $\sum_i x_i = n$, this yields $\lambda = 1$. Using the convenient fact that the Lagrange multiplier is a constant, a simplified objective is defined,

$$\begin{aligned}L_2(\boldsymbol{\theta}) &= \sum_i \theta_i \langle x_i \rangle_T + (1 - \sum_i \exp(\theta_i)),\\ \frac{\partial L_2(\boldsymbol{\theta})}{\partial \theta_i} &= \langle x_i \rangle_T - \exp(\theta_i) \overset{!}{=} 0,\\ \theta_i &\overset{!}{=} \log \langle x_i \rangle_T,\end{aligned}$$

giving the learning rule

$$\Delta_{\theta_i} = \eta \frac{x_{i,T} - \exp(\theta_i)}{\exp(\theta_i)} \tag{5.25}$$

$$= \eta(x_{i,T} \exp(-\theta_i) - 1). \tag{5.26}$$

Note that this generalizes the single trial result from Nessler et al. [2010].

5.3.3 Comparison of Learning Rules

Tables 5.1 and 5.2 summarize important properties and learning rules for six different distributions: Normal, Poisson, Multinomial, Gamma, Binomial and Negative binomial:

Domain: The set of admissible inputs, which can be continuous or discrete, finite or infinite.

Fixed parameters: Parameters of the distribution which are assumed fixed. Fixed parameters must come from prior knowledge of the input data, since they usually cannot be learned with a simple scheme as the one proposed.

Variable parameter (standard form): The parameter of the distribution which is assumed unknown and which can be learned from data with the proposed method.

Mean: The mean of the distribution in dependence of the variable parameter.

Variance: The variance of the distribution in dependence of the variable parameter.

Variable parameter (Exponential family form): The variable parameter expressed in the exponential family form. The exponential family form differs from the standard form of most distributions. Hence, this row accounts for the re-parametrization of the variable parameter and shows the relation between parameter spaces in original and exponential family form.

Parameter domain: The set of admissible parameters in the exponential family form. This is typically of a subset from the real numbers.

Interpretation: The meaning of the parameter in exponential family form.

Log-partition function $A(\theta)$: The corresponding log-partition function of the family.

Mean μ_θ: Shows the dependence of the mean of the distribution on the exponential family parameter.

Variance function: The mean-variance relation of the family (see Section 3.3.3). Note that the variance function is not given for the multinomial case, since a channel in a population coded input ensemble is highly dependent on other channels and therefore does not qualify as a single-parameter distributed input.

Learning rule: The learning rule which has its equilibrium point at the correct maximum likelihood estimate based on complete observations.

Figure 5.1 visualizes the learning rules for five distributions. The black line denotes the optimal parameter θ given some input x, that is the parameter that explains the input best. The arrows indicate parameter changes caused by a mismatch between current and optimal parameter.

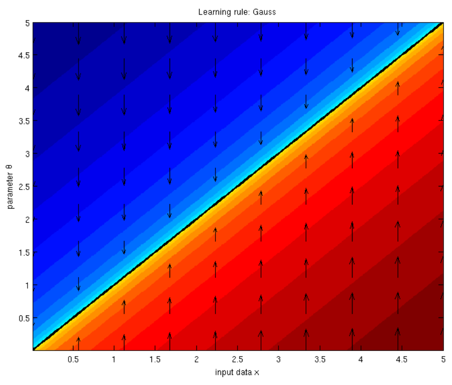
Note that, for visual clarity, the learning dynamics are shown for real-valued inputs x, even if the corresponding distribution has only discrete support.

	Normal	Poisson	Single X_i from Multinomial
Domain	$x \in \mathbb{R}$	$x \in \mathbb{N}_0$	$x \in \{0, \dots, n\}$
Fixed parameters	variance σ^2	-	No. trials n
Variable parameter (standard form)	mean μ	rate λ	prob. p
Mean	$\mathrm{E}(X) = \mu$	$\mathrm{E}(X) = \lambda$	$\mathrm{E}(X) = np$
Variance	$\mathrm{Var}(X) = \mathrm{const}$	$\mathrm{Var}(X) = \lambda$	$\mathrm{Var}(X) = np(1-p)$
Variable parameter (Exp.fam. form)	$\theta = \mu$	$\theta = \log(\lambda)$	$\theta = \log(np)$
Parameter domain	$\theta \in (-\infty, \infty)$	$\theta \in (-\infty, \infty)$	$\theta \in (-\infty, \log n]$
Interpretation	Mean	Log-rate	Log-mean
$A(\theta)$	$\theta^2/2$	$\exp(\theta)$	0
μ_θ	θ	$\exp(\theta)$	$\exp(\theta)$
Variance function $V(\mu)$	$\sigma^2(= \mathrm{const})$	μ	-
Learning rule Δ_θ/η	$x - \theta$	$\exp(-\theta)x - 1$	$\exp(-\theta)x - 1$

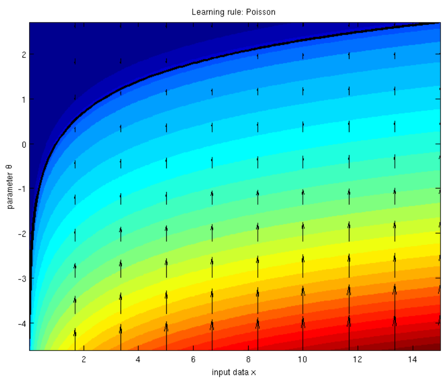
Table 5.1: Comparison of exponential families (I)

	Gamma	Binomial	Negative binomial
Domain	$x \in [0, \infty)$	$x \in \{0, \dots, n\}$	$x \in \mathbb{N}_0$
Fixed parameters	shape α	No. trials n	No. of failures n
Variable parameter (standard form)	rate β	success prob. p	success prob. p
Mean	$\mathrm{E}(X) = \alpha\beta$	$\mathrm{E}(X) = np$	$\mathrm{E}(X) = np/(1-p)$
Variance	$\mathrm{Var}(X) = \alpha\beta^2$	$\mathrm{Var}(X) = np(1-p)$	$\mathrm{Var}(X) = np/(1-p)^2$
Variable parameter (Exp.fam. form)	$\theta = -\alpha/\mu$	$\theta = \log \frac{p}{1-p}$	$\theta = \log p$
Parameter domain	$\theta \in (-\infty, 0)$	$\theta \in (-\infty, \infty)$	$\theta \in (-\infty, 0)$
Interpretation	Neg. inverse rate	Log-odd	Log-probability
$A(\theta)$	$-\alpha \log -\theta$	$n \log(1 + \exp(\theta))$	$-n \log(1 - \exp(\theta))$
μ_θ	$-\alpha/\theta$	$n/(1 + \exp(-\theta))$	$n/(\exp(-\theta) - 1)$
Variance function $V(\mu)$	$V(\mu) = \mu^2/\alpha$	$V(\mu) = \mu - \mu^2/n$	$V(\mu) = \mu + \mu^2/n$
Learning rule Δ_θ/η	$\theta(1 + \theta x/\alpha)$	$(1 + e^\theta)\left((1 + e^{-\theta})x/n - 1\right)$	$2x(\cosh\theta - 1)/n + e^\theta - 1$

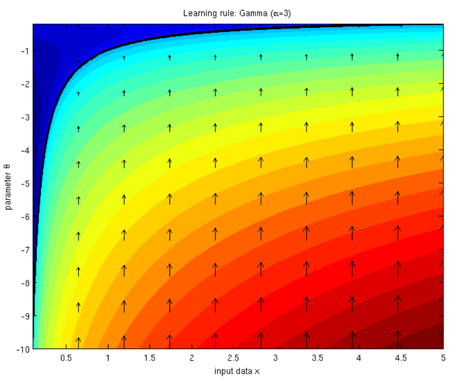
Table 5.2: Comparison of exponential families (II)



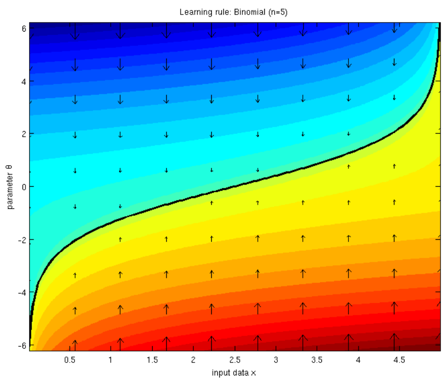
(a) Normal



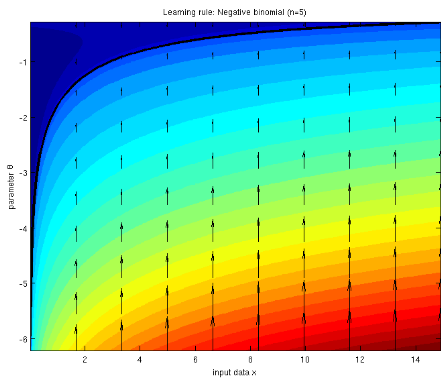
(b) Poisson



(c) Gamma (α=3)



(d) Binomial (n=5)



(e) Negative binomial (n=5)

Figure 5.1: Generalization: Comparison of learning rules for different choices of component distribution. The black line denotes the optimal parameter θ given some input x, that is the parameter that explains the input best. The arrows indicate parameter changes caused by a mismatch between current and optimal parameter. For visual clarity, learning dynamics are shown for real-valued inputs x, even if the corresponding distribution has only discrete support.

5.4 From Recursive ML to Approximate Online EM

So far, only learning with complete observations has been considered. Typically, solving the marginal ML problem $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{x} \,|\, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_i p(\mathbf{x}_i \,|\, \boldsymbol{\theta})$ for partially observed data with independent observations $\mathbf{x}_i$ and missing data z_i is a hard problem. The basic trick behind EM (see Section 3.2) that makes learning tractable is the following:

- First obtain complete data by inferring the hidden state probabilities $p(z_i)$ from the independent observed data samples $p(\mathbf{x}_i)$, using the current estimate of the model $\boldsymbol{\theta}^{(t)}$. This produces a temporary joint distribution $p^*(\mathbf{x}_i, z_i| \,\boldsymbol{\theta})$.
- Then solve the joint ML problem $\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \prod_i p^*(\mathbf{x}_i, z_i| \,\boldsymbol{\theta})$.

If these two steps are performed iteratively, one can guarantee convergence to local maxima of the likelihood function under mild assumptions (Neal and Hinton [1998]). The following EM-based learning procedure for discrete-time is proposed:

1. At time step $t+1$, consider the next input $\mathbf{x}^{(t+1)}$, for simplicity assumed to be sampled from some stationary distribution $p(\mathbf{X})$.
2. Compute a set of sufficient statistics $\mathbf{y}^{(t+1)} = T(\mathbf{x}^{(t+1)})$.
3. Sample $z^{(t+1)}$ from a multinomial distribution $p(z| \,\mathbf{y}^{(t+1)}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\theta}^{(t)})$ where

$$p(z = k| \,\mathbf{y}, \boldsymbol{\pi}, \boldsymbol{\theta}) \propto \exp(\boldsymbol{\theta}_k^T \,\mathbf{y} + \pi_k + A(\boldsymbol{\theta}_k)), \tag{5.27}$$

   with $\boldsymbol{\pi} = (\pi_k, k = 1 \ldots N)$ and $\boldsymbol{\theta} = (\boldsymbol{\theta}_k, k = 1 \ldots N)$.
4. Update neuron parameters $\boldsymbol{\pi}^{(t+1)} = \boldsymbol{\pi}^{(t)} + \eta\Delta_{\boldsymbol{\pi}_k}^{(t+1)}$.
5. Update synapse parameters $\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t-1)} + \eta\Delta_{\boldsymbol{\theta}_k}^{(t+1)}$.

Eq. 5.27 constitutes the inference step of the model and corresponds to the previously derived inference equation for MEFs (see Eq. 5.4). The learning updates with $\eta\Delta_{\boldsymbol{\pi}_k}^{(t+1)}$ and $\eta\Delta_{\boldsymbol{\theta}_k}^{(t+1)}$ are performed according to the recursive estimator equations derived earlier.

This scheme implements an online approximation to EM in the spirit of Neal and Hinton [1998] and Gilles Celeux et al. [1995]. A proof of convergence can be found in Appendix A .

5.5 Spikes, Poisson, and Continuous Time

From the above derivations, it may not be immediately clear, how the proposed method relates to spiking neural networks. This section aims at establishing the relation. First, a particular special case of the general algorithm will be elaborated that assumes Poisson distributed inputs which are independent conditioned on the hidden causes. Then, a discussion follows how and under what conditions the discrete time algorithm can be applied to Poisson processes. A spiking neural network is presented which implements the proposed model.

5.5.1 Poisson-based scheme in Discrete Time

Based on the above general learning scheme, at time step $t+1$,

1. Consider input $\mathbf{x}^{(t+1)}$.
2. Skip computation of sufficient statistics, since $y_i^{(t+1)} = x_i^{(t+1)}$.
3. Sample $z^{(t+1)}$ from $p(z=k|\,\mathbf{x},\boldsymbol{\pi},\boldsymbol{\theta}) \propto \exp(\boldsymbol{\theta}_k^T\,\mathbf{x} + \pi_k - \sum_j e^{\theta_{kj}})$.
4. Update $\pi_k^{(t+1)} = \pi_k^{(t)} + \begin{cases} \eta(e^{-\pi_k^{(t)}} - 1) & \text{if } z^{(t+1)} = k, \\ -\eta & \text{otherwise.} \end{cases}$.
5. Update $\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} + \begin{cases} \eta(e^{-\theta_{kj}} x_j^{(t+1)} - 1) & \text{if } z^{(t+1)} = k, \\ 0 & \text{otherwise.} \end{cases}$

Note that there are a few significant differences from this particular learning algorithm to the one proposed in Nessler et al. [2010]: First, the here proposed algorithm works with count data as opposed to binary inputs. This is useful in the context of spikes, as will be discussed in Section 5.5.2.

Inference in the new scheme is performed through,

$$u_k = \boldsymbol{\theta}_k^T \mathbf{x} + \pi_k - \sum_j e^{\theta_{kj}}, \tag{5.28}$$

as opposed to,

$$u_k = \boldsymbol{\theta}_k^T \mathbf{x} + \pi_k.$$

Here, it should be noted that $\sum_j e^{\theta_{kj}} = \sum_j \lambda_{kj}$, where λ_{kj} is the average rate of the input j when output k is active. Different hidden states k can specialize on patterns with different overall rates. Without the additional normalization term, those hidden neurons k would always dominate which have large θ_{kj}, even if the input has low rates. If one can guarantee that the sum of inputs is constant at all times, the term can be dropped since the sum of learned rates will converge to the same value for all k and will appear as a multiplicative constant that cancels due to the normalization in $p(z = k|\, \mathbf{y}, \boldsymbol{\pi}, \boldsymbol{\theta}) = \frac{u_k}{\sum_{k'} u_{k'}}$. If the normalization terms vary across internal neurons, they must be taken into account for correct inference.

Learning in the new model is formally very similar to the original rule, except for the fact that the new scheme allows for input values greater than 1.

5.5.2 Spiking Neural Network Implementation

The Poisson perspective is particularly appealing when a continuous-time STDP learning scheme such as in Nessler et al. [2010] is considered. If the input is supplied as an ensemble of inhomogeneous Poisson processes with instantaneous rates $\lambda_j(t)$, generating a number of events in a time window

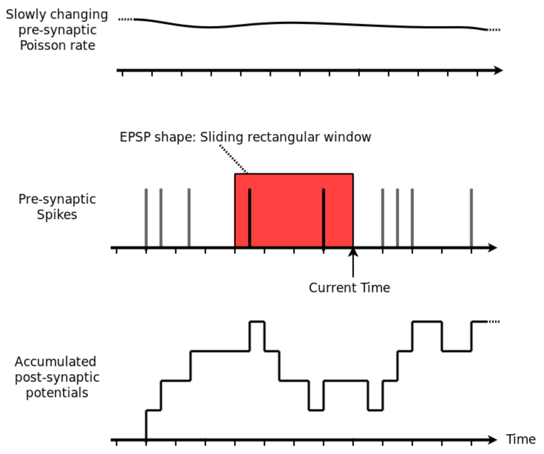
$[t - l, t]$, then on average,

$$m_j^{(t)} = \int_{t-l}^{t} \lambda_j(\tau) d\tau.$$

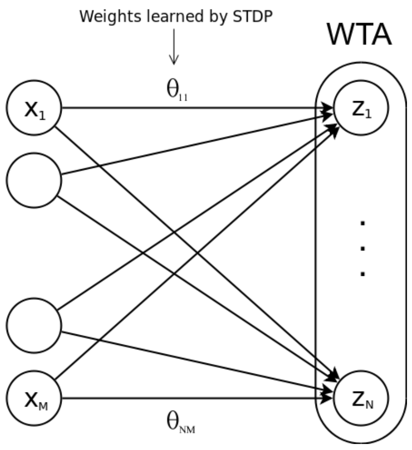
For small l and/or slowly changing Poisson rates one can approximate this by,

$$m_j^{(t)} \approx \int_{t-l}^{t} \lambda_j(t - l/2) d\tau = h\lambda_j(t - l/2).$$

This means, that if at any point t in time one has access to the number of spikes in small window of length l, then these inputs will be Poisson distributed and the model derived above can be used to perform learning and inference. What is still missing in the transition to a spiking neural network implementation is a specification of when and how often observations are taken. In accordance with the scheme in Nessler et al. [2010], for simplicity one can assume an external process which produces trigger events. At the arrival of such an event, the inference procedure is initiated, a winner is selected according to a WTA operation, and learning is performed for the synaptic weights which belong to the winning neuron.


Slowly changing pre-synaptic Poisson rate
EPSP shape: Sliding rectangular window
Pre-synaptic Spikes
Current Time
Accumulated post-synaptic potentials
Time

(a) Input: From rate to EPSP



(b) Poisson-based model

Figure 5.2: Spiking neural network implementation of Poisson-based model. The left graphic (a) shows how input spikes are assumed to be generated, and how they are accumulated by a rectangular EPSP shape. On the right side (b), the basic setup is illustrated. Inputs are connected in an bipartite all-to-all fashion to the internal layer which is controlled by a Winner-Take-All (WTA) circuit. The synaptic weights can be learned with Spike-Timing Dependent Plasticity (STDP).

The proposed spiking neural network implementation of the Poisson-based model is shown in Figure 5.2. The model corresponds to the elements of the spiking network as follows:

- The parameters θ_{kj} are the synaptic weights of the network. Since parameters θ_{kj} can be negative, one may shift them to the positive range as described in Nessler et al. [2010]. The prior parameters π_k correspond to intrinsic parameters of the internal neurons.

- Inference corresponds to neural integration, in combination with a WTA operation. The membrane potential of neuron k before WTA is given by,

$$u_k = \boldsymbol{\theta}_k^T \mathbf{x} + \pi_k - \sum_j e^{\theta_{kj}}. \tag{5.29}$$

The weighted sum $\boldsymbol{\theta}_k^T \mathbf{x}$ is equivalent to the accumulation of weighted EPSPs from all input spike trains. π_k can be understood as an intrinsic parameter of the neuron's general excitability. The remaining term $\sum_j e^{\theta_{kj}}$ is independent from the current input but depends on the synaptic weights which are attached to the neuron. As has been discussed, under appropriate conditions where the overall input rate is constant, this term can be omitted. Alternatively, one can consider heterosynaptic competition, a phenomenon which is ubiquitous in biological networks (Fiete et al. [2010]). With an appropriate mechanism, this term will be approximately equal for all neurons and is therefore canceled during the WTA operation. Yet another option to deal with this term is discussed in a further extension in Section 6.3.

- The internal output layer is activated according to some external schedule. Whenever activation of the internal layer is triggered, a winning neuron is selected by the WTA circuit. The winner emits a single spike.

- Parameter learning of θ_{kj} can be mapped to weight-dependent Spike-Timing Dependent Plasticity: each pre-synaptic spike which occurs before a post-synaptic spike within the window of the EPSP-shape increases the synaptic weight. Pre-synaptic spikes which occur after a post-synaptic spike do not have a direct effect on the weight. This behavior is reflected by the update equation,

$$\Delta_{\theta_{kj}} \propto e^{-\theta_{kj}} x_j - 1, \quad \text{if } z = k, \tag{5.30}$$

since each pre-synaptic spike from input j within the current EPSP window increments x_j. The constant decay may be seen as the negative part of standard STDP, with the slight difference that here, the negative weight contribution is triggered even if there is no pre-synaptic spike following the post-synaptic spike. The weight dependence appears as a multiplicative factor $e^{-\theta_{kj}}$ which results in the saturation of weights even for high inputs as long as they are bounded. Note that this automatically solves the problem of instability in standard Hebbian learning (see Section 2.3.1). Similar to the synaptic weights, the intrinsic parameters π_k can be learned locally at the neuron.

Chapter 6

Further Extensions

Based on the model developed in the previous chapter, three more extensions will be discussed. Section 6.1 targets at more realistic EPSP shapes. Section 6.2 is about including prior information and Maximum a Posteriori parameter estimation. Finally, Section 6.3 shows an alternative way of implementing the normalization term $\sum_j e^{\theta_{kj}}$, which appears in the inference equations in the Poisson-based model.

6.1 Incorporating Realistic EPSP shapes

The influence of an input spike on the state of a neuron is limited in time. In biological neurons, the time course of this influence is called excitatory postsynaptic potential (EPSP) and has a characteristic decaying shape (Purves [2008]). In contrast, the method by Nessler et al. [2010] proposes the equivalent of a rectangular EPSP shape. Also the Poisson-based model, presented in the previous chapter, is based on such an unrealistic assumption.

This section aims at incorporating more realistic EPSP shapes in the model. One obstacle is that the convolution of an arbitrary EPSP shape with a spike train with Poisson statistics leads to distributions which deviate from the standard Poisson case (see Section 6.1.2). These distributions may be outside the exponential family, such that exact inference may not be possible anymore. It is crucial to determine which distributions from the exponential family best approximate the real distributions encountered when using more complex EPSPs in order to keep errors small during inference. To this end, a method is proposed which systematically constructs an exponential family with a variance function identical to the desired distribution. Errors are therefore limited to higher-order moment mismatch.

6.1.1 Convolution of Poisson Spike Trains with Arbitrary EPSP shapes

As in the previous chapter, we will assume that input spike trains are generated by inhomogeneous Poisson processes with slowly varying rates. We will first consider piece-wise constant EPSP shapes $\epsilon(t)$ of finite length L which integrate to one ($\int_0^L \epsilon(t)\,dt = 1$). Suppose that there are n individual pieces with different magnitudes ϵ_i, each of length δ, such that $L = n \cdot \delta$. Formally,

$$\epsilon(t) = \sum_{i=0}^{n-1} \epsilon_i H(t - i \cdot \delta) H(-t + (i+1) \cdot \delta), \tag{6.1}$$

where $H(\cdot)$ denotes the Heaviside step function. Then, the convolution of a spike train $x(t)$ with such an EPSP can be written as,

$$\xi(t) = \int_0^L x(t-\tau)\epsilon(\tau)\,d\tau \tag{6.2}$$

$$= \sum_{i=0}^{n-1} \epsilon_i \int_0^\delta x(t - i \cdot \delta - \tau)\,d\tau. \tag{6.3}$$

Assuming a constant Poisson rate λ throughout the integration period, all sub-integrals in Eq. 6.3 will be Poisson distributed with ensemble mean and variance $\delta\lambda$. Let a single contribution to this sum be,

$$\xi_i(t) = \epsilon_i \int_0^l x(t - i \cdot l - \tau)\,d\tau. \tag{6.4}$$

Since sub-integrals are Poisson distributed, the ensemble mean and variance of $\xi_i(t)$ can be found easily as (Bickel and Doksum [2001]),

$$\mathrm{E}(\xi_i(t)) = \epsilon_i \delta \lambda \tag{6.5}$$

$$\mathrm{Var}(\xi_i(t)) = \epsilon_i^2 \delta \lambda. \tag{6.6}$$

Furthermore, since different $\xi_i(t)$ are independent, the sum of contributions is distributed with mean and variance,

$$\mathrm{E}(\xi(t)) = \sum_{i=0}^{n-1} \epsilon_i \delta \lambda = \lambda \sum_{i=0}^{n-1} \epsilon_i \delta = \lambda \int_0^L \epsilon(\tau)\, d\tau = \lambda \tag{6.7}$$

$$\mathrm{Var}(\xi(t)) = \sum_{i=0}^{n-1} \epsilon_i^2 \delta \lambda = \lambda \int_0^L \epsilon^2(\tau)\, d\tau. \tag{6.8}$$

The mean equals λ, such that $\xi(t)$ is an unbiased estimator of the underlying rate. The variance scales linearly with λ, but also depends on the shape of the EPSP. Note that this results holds irrespective of how small one makes δ, such that arbitrary EPSP shapes can be considered in the limit of $\delta \to 0$, and if required, $L \to \infty$.

6.1.2 Variance Function Depends on EPSP Shape

For a rectangular EPSP of length $L = 1$, one has

$$\mathrm{Var}(\xi(t)) = \lambda \int_0^1 \frac{1}{1^2}\, d\tau = \lambda, \tag{6.9}$$

as expected. For an exponentially decaying EPSP $\epsilon(t) = \frac{1}{\tau_d} \exp(-t/\tau_d)$, one obtains,

$$\mathrm{Var}(\xi(t)) = \lambda \int_0^\infty \frac{1}{\tau_d^2} \exp(-2\tau/\tau_d)\, d\tau = \frac{1}{2\tau_d}\lambda, \tag{6.10}$$

reflecting the fact that longer integration reduces estimation variance. In general, one will obtain some variance which is proportional to λ,

$$\mathrm{Var}(\xi(t)) = \phi\lambda. \tag{6.11}$$

The factor of proportionality ϕ will be referred to as dispersion factor. Since the mean equals λ, this gives rise to the variance function (see Section 3.3.3 for an introduction to variance functions),

$$V(\mu) = \phi\mu. \tag{6.12}$$

The crucial question is whether $\xi(t)$ can be modeled by an exponential family distribution. Interestingly, one can easily construct an exponential family which has the desired variance function by multiplication of Poisson variables with the desired factor ϕ. Given a Poisson distributed variable X with rate λ and $Y = \phi X$,

$$\mathrm{E}(Y) = \phi\lambda, \tag{6.13}$$

$$\mathrm{Var}(Y) = \phi^2\lambda, \tag{6.14}$$

$$V(\mu) = \phi\mu. \tag{6.15}$$

Note that the construction of a natural exponential family from the variance function according to the above scheme is unique[1]. This means that given the parameter ϕ, there is a single natural exponential family which exactly matches the desired variance function. As a consequence, higher moments (or more precisely, cumulants) in the model typically cannot be matched to the corresponding higher moments of $\xi(t)$ with this technique. Clearly, this means that inference in the model only approximates ideal inference. Further research will be necessary to evaluate the effect of this approximation.

[1]Up to a constant parameter shift.

6.1.3 Model Adjustments for Arbitrary EPSPs

In Section 6.1.2 it was shown that for arbitrary EPSPs one can construct an exponential family which matches the variance function of the filtered input spike train. Below, this will be set more explicitly in the context of the method proposed in this thesis.

As described in the previous section, given an EPSP shape, one can compute the dispersion factor ϕ, such that the variance function of the filtered spike train can be expressed as,

$$V(\mu) = \phi\mu. \tag{6.16}$$

Now, consider the probability density function of a Poisson random variable, multiplied by a constant factor ϕ,

$$p(x|\lambda) = \frac{\lambda^{x/\phi}\exp(-\lambda)}{(x/\phi)!}, \quad x/\phi \in \mathbb{Z}. \tag{6.17}$$

This can be written in natural exponential family form with $\theta = \log\lambda$,

$$p(x|\theta) = \frac{1}{(x/\phi)!}\exp\left(\frac{\theta x - \phi e^{\theta}}{\phi}\right), \quad x/\phi \in \mathbb{Z}. \tag{6.18}$$

In this form, one can easily identify the log-partition function $A(\theta) = \phi e^{\theta}$. Using the theory developed earlier, the corresponding local maximum likelihood learning rule can be derived by application of Eq. 5.24:

$$\begin{aligned} \Delta_\theta &= \eta\frac{x - A'(\theta)}{A''(\theta)} && (6.19)\\ &= \eta\frac{x - \phi e^{\theta}}{\phi e^{\theta}} && (6.20)\\ &= \eta(\frac{x}{\phi}e^{-\theta} - 1). && (6.21) \end{aligned}$$

Note that for $\phi = 1$, the standard Poisson rule is recovered. Also during inference, one must use a slightly different normalization term in order to account for the correct log-partition function:

$$u_k = \frac{1}{\phi}\boldsymbol{\theta}_k^T \mathbf{x} - \sum_j e^{\theta_{kj}}. \tag{6.22}$$

Again, $\phi = 1$ gives the standard Poisson model.

6.2 Maximum a Posteriori Estimation

The standard maximum likelihood approach to parameter estimation is to find a parameter set θ that maximizes $p(X|\theta)$ on some dataset X. On many occasions, however, also some prior knowledge $p(\theta)$ about the parameters themselves is available. The incorporation of prior knowledge can be particularly useful when dealing with high-noise input distributions where parameters are difficult to decide on from few samples. With maximum a posteriori one can combine prior and data-dependent knowledge by maximizing $p(\theta|X)$ instead of $p(X|\theta)$ (Bickel and Doksum [2001]). Section 6.2.1 extends the theory developed in this thesis to include prior information. A simple example with a Gaussian input is illustrated in Section 6.2.2.

6.2.1 Extended Theory for Including Prior Information

As with maximum likelihood, the resulting parameter set is a point estimate as opposed to a full posterior distribution (Bayesian learning).

By application of Bayes rule one has,

$$p(\theta|X) = \frac{p(\theta)}{p(X)} p(X|\theta) \tag{6.23}$$

$$\log p(\theta|X) = \log p(\theta) - \log p(X) + \log p(X|\theta) \tag{6.24}$$

Note that the middle term $\log p(X)$ does not depend on θ. Therefore an objective function is defined,

$$L(\theta) = \log p(X|\theta) + \log p(\theta) \tag{6.25}$$

Applying this to Eq. 5.16 for exponential families yields

$$L_T(\theta) = \theta\langle x\rangle_T - A(\theta) + \frac{\phi}{T}\log p(\theta) \tag{6.26}$$

$$= \theta\langle x\rangle_T - A(\theta) - \frac{\phi}{T}B(\theta), \tag{6.27}$$

with

$$B(\theta) = -\log p(\theta). \tag{6.28}$$

Then the maximizing θ after $T-1$ input samples fulfills

$$A'(\theta) + \frac{\phi}{T-1}B'(\theta) = \langle x\rangle_{T-1} \tag{6.29}$$

An additional sample x_T leads to a new estimate $\theta + \Delta_\theta$ for which must hold

$$A'(\theta + \Delta_\theta) + \frac{\phi}{T}B'(\theta + \Delta_\theta) = \langle x\rangle_T \tag{6.30}$$

Approximating the left-hand side by a first-order Taylor series leads to the following MAP-based learning rule:

$$\Delta_\theta = \eta\frac{x_T - A'(\theta)}{A''(\theta) + \eta\phi B''(\theta)} \tag{6.31}$$

Note that, in the limit of a flat prior distribution $\log p(\theta)$, Eq. 6.31 reduces to the ML-rule (Eq. 5.24). Interestingly, learning from input data is automatically slowed down for high noise distributions ($\phi \gg 1$), thus rendering the prior distribution more influential. Also note that the parameter θ must be initialized at the (preferably unique) maximum of the prior distribution ($B'(\theta_0) \overset{!}{=} 0$), in order for the recursive argument to hold. Since MAP only tracks a point estimate, multi-modal prior distributions are discouraged.

It remains to be underlined that inference in the model is independent of the learning objective, such that the choice of MAP/ML only affects the learning step of an EM implementation.

6.2.2 Example with Gaussian Input

Consider a single Gaussian input x with unknown mean μ and known variance σ^2. The input is modeled with a Gaussian distribution, in its exponential family form:

$$p(x|\theta) = h(x) \exp\left(\frac{\theta x - \theta^2/2}{\sigma^2}\right) \tag{6.32}$$

For ML learning a learning rule derived from Eq. 5.24 was employed:

$$\Delta_\theta = \eta(x_T - \theta) \tag{6.33}$$

For MAP learning, the prior distribution is chosen as a zero-mean unit-variance Gaussian:

$$p(\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right) \tag{6.34}$$

$$B(\theta) = 0.5(\log(2\pi) + \theta^2) \tag{6.35}$$

$$B'(\theta) = \theta \tag{6.36}$$

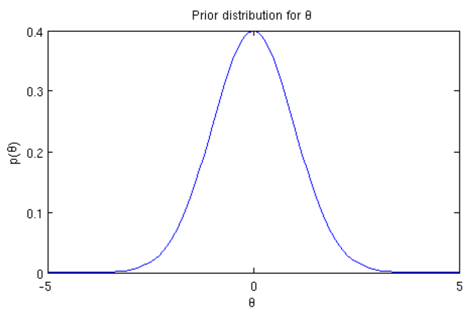
$$B''(\theta) = 1 \tag{6.37}$$

Then, Eq. 6.31 gives the MAP learning rule:

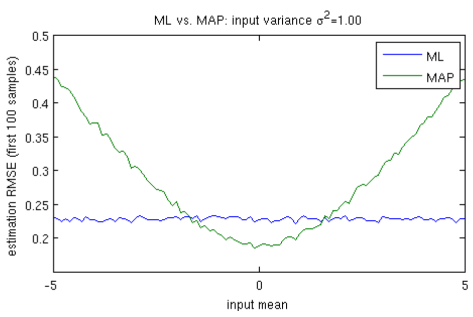
$$\Delta_\theta = \eta \frac{x_T - \theta}{1 + \eta\sigma^2} \tag{6.38}$$

In both methods, the parameter θ is initialized to zero. Each method is provided with $T = 100$ input samples, the procedure is repeated in 100 trials. For each method, the root mean squared estimation error is computed as $\sqrt{\sum_{t=1}^{T}(\theta_t - \mu)^2}$. The error is evaluated for different μ and σ^2, the results are presented in Figure 6.1.

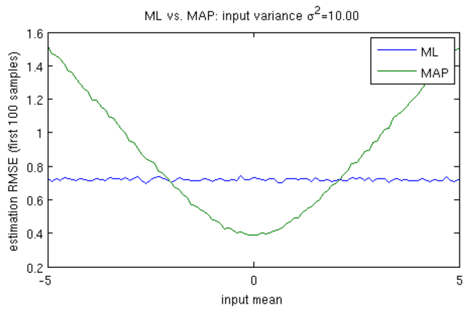
The results clearly show the potential improvement through MAP learning, especially when the input variance is high compared to the variance of the prior distribution. On the other hand, in low variance settings or if the



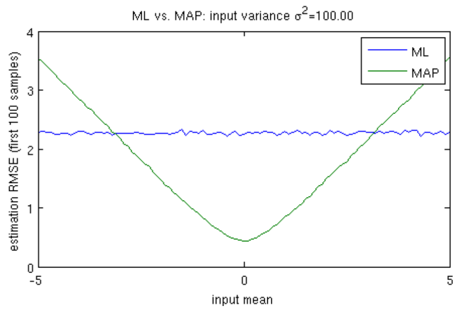
(a) Prior distribution (MAP)



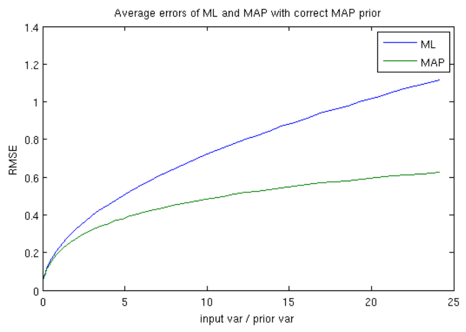
(b) RMSE: Low input variance



(c) RMSE: Medium input variance

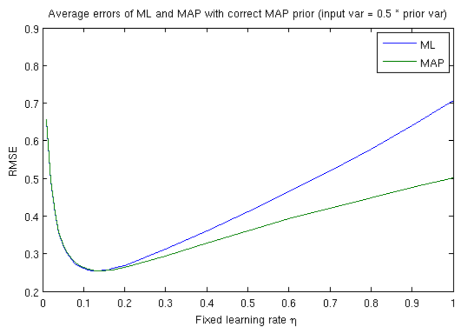


(d) RMSE: High input variance

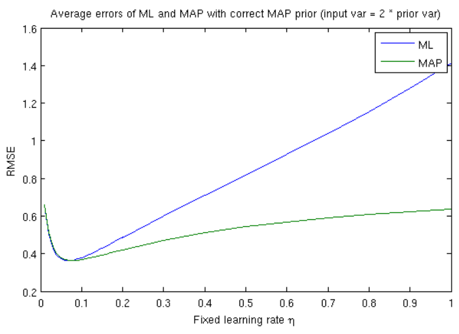


(e) Average errors with input mean drawn from MAP prior

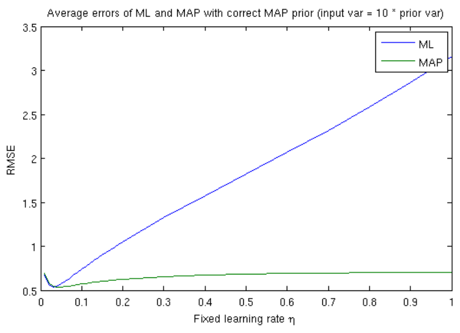
Figure 6.1: Maximum Likelihood versus Maximum a Posterior learning rules using a decaying learning rate schedule. (a) Prior distribution for parameter θ, corresponding to the distribution from which generative parameters were chosen during subsequent simulations. (b) - (d) Performance of ML and MAP learning rules in dependence of generative θs, for different input noise levels. ML does not use prior information, therefore performance does not depend on θ. MAP performs best for θ which are probable under the prior distribution. (e) Average errors for ML and MAP in dependence of input noise level. High input variance makes prior information more important.



(a) RMSE: Low input variance



(b) RMSE: Medium input variance



(c) RMSE: High input variance

Figure 6.2: Maximum Likelihood versus Maximum a Posterior with fixed learning rate. Average performance of ML and MAP learning rules under three noise conditions. MAP is more robust to the choice of learning rate and outperforms ML in all conditions.

prior distribution is unknown, ML might be the more robust choice.

Another interesting case to consider is a fixed learning rate scenario. In such a setup, η is chosen beforehand to some determined value which stays constant throughout the whole experiment, as opposed to the decaying schedule $\eta_T = 1/T$ for the above experiments. Again, the RMSE is computed from the first 100 samples. Figure 6.2 shows the comparison in performance between the ML and MAP learning rules for different input variances. Like in the decaying learning rate scenario, the superiority of the MAP learning rule is clearly visible for all settings under study and becomes even more pronounced for higher input variances.

6.3 An Approximation of the Normalization Term

Examining the activation function,

$$u_k = \exp(\boldsymbol{\theta}_k^T \mathbf{x} + \pi_k - \sum_j e^{\theta_{kj}}),$$

one may recognize that the normalization term $\sum_j e^{\theta_{kj}}$ may be quite impractical for some implementations as it requires the computation of a sum over all synapses, additional to the usual input path $\boldsymbol{\theta}_k^T \mathbf{x}$. This could be particularly inconvenient in hardware implementations where, as in biology, synapses and neurons might be spatially separated. Creating an additional communication channel will generally complicate the design and increase costs.

A few ways of dealing with this inconvenience have been mentioned already in Section 5.5.2. In particular, if the overall input rate stays nearly constant, the term can be dropped altogether. Heterosynaptic competition may serve the same purpose as constant overall input rates by equalizing the normalization term across all neurons, thus making it dispensable. A third option will be discussed here, which enables the neuron to estimate its synapse weights based exclusively on observed EPSPs.

6.3.1 A First Approximation

First, one must recognize that any systematic modification of the inference step does not interfere with the optimality of the learning rule with respect to the joint maximum likelihood. Certainly, the algorithm will generally not perform correct EM due to the defective inference and "erroneous" activations of hidden units. Still it is important to see that, due to the optimal learning step, one can assume that after sufficient time the algorithm converges, and

$$\theta_{kj} \approx \log \hat{x}_{kj}, \tag{6.39}$$

with $\hat{x}_{kj} = \langle x_j \rangle_{p(x|Z=k)}$. Then, using the observed EPSP from synapse j to neuron k, one has,

$$\langle \theta_{kj} x_j \rangle_{p(x|Z=k)} \approx f(\hat{x}_{kj}) = \hat{x}_{kj} \log \hat{x}_{kj}$$

Note that this average is taken over all input configurations that caused the neuron k to spike. Therefore this will be called the spike-induced EPSP average.

In principle, one could observe this quantity for each k and j and try to infer $\hat{x}_{kj}$ (as well as θ_{kj}) from that. Unfortunately, the function $f(\hat{x}_{kj})$ is not injective, thereby prohibiting such an approach. To get around this difficulty, shifted weights are introduced,

$$\theta'_{kj} = m + \theta_{kj},$$

limited by $\theta'_{kj} \geq 0$. The parameter m has to be chosen large enough to prevent cutting away too much of the input range, as the limitation implies that $\theta_{kj} \geq -m$. Adapting the learning and inference equations to use the shifted weights is trivial.

For the new spike-induced EPSP average, which will be denoted by ψ_{kj}, one can write,

$$\psi_{kj} = \langle \theta'_{kj} x_j \rangle_{p(x|Z=k)} \approx g(\hat{x}_{kj}) = \hat{x}_{kj}(m + \log \hat{x}_{kj}) \tag{6.40}$$

Now one can inspect the derivative of $g(\cdot)$ to analyze monotonicity:

$$g'(\hat{x}_{kj}) = m + \log \hat{x}_{kj} + 1.$$

Apparently, in order for $g(\cdot)$ to be strictly monotone increasing, one has to ensure that

$$m > -(1 + \log \hat{x}_{min}),$$

where $\hat{x}_{min}$ is the smallest ensemble averaged input at any time[2]. Then, $g(\cdot)$ is injective on the interval $[\hat{x}_{min}, \infty[$ and one can use the inverse

$$\hat{x}_{kj} = \frac{g(\hat{x}_{kj})}{W(e^m g(\hat{x}_{kj}))},$$

where $W(\cdot)$ denotes the Lambert W function. Using Eq. 6.40 gives,

$$\hat{x}_{kj} \approx \frac{\psi_{kj}}{W(e^m \psi_{kj})},$$

By Eq. 6.39 one also has,

$$e^{\theta_{kj}} \approx \frac{\psi_{kj}}{W(e^m \psi_{kj})},$$

and therefore the normalization term can be expressed as,

$$\sum_j e^{\theta_{kj}} \approx \sum_j \frac{\psi_{kj}}{W(e^m \psi_{kj})}. \tag{6.41}$$

At this point it is useful to recall what has been gained by using the approximation in Eq. 6.41. If one assumes that the synapse can communicate with the neuron just over the EPSP "channel", the above approximation enables the use of that channel for two purposes, namely

- Read out current EPSPs in order to decide on momentary spiking behavior.
- Use spike-induced EPSP averages for normalization and, consequently, correct inference.

Also, since the approximate behavior of the Eq. 6.41 is only due to $\theta_{kj} \approx \log \hat{x}_{kj}$, but otherwise exact, one may expect the approximation error to vanish when convergence is reached.

[2]If the input is noisy, this requirement is easy to fulfill, since $\hat{x}_{kj}$ will always be greater than some noise-dependent constant ϵ.

Still, the method is not quite satisfactory for two reasons. Firstly, the computation of the Lambert W function is somewhat time-consuming, as it is usually based on fixed-point iterations (see Chapeau-Blondeau and Monir [2002] for an alternative method). Secondly, one has to maintain and update an average for each single synapse j connected to neuron k. This is both computationally expensive and biologically hard to justify.

6.3.2 A More Realistic Neural Approximation

Motivated by the shortcomings of the approximation derived in the previous section, here a slightly different way of estimating $\sum_j e^{\theta_{kj}}$ through the observation of the spike-induced average of $\boldsymbol{\theta}_k^T \mathbf{x}$ is presented. The goal is to replace the approximation,

$$\sum_j e^{\theta_{kj}} \approx \sum_j g^{-1}(\psi_{kj}),$$

by something of the form,

$$\sum_j e^{\theta_{kj}} \approx h(\sum_j \psi_{kj}).$$

This step would be possible exactly, if $g(\cdot)$ was an affine map. However, as was found before, $g(\hat{x}_{kj}) = \hat{x}_{kj}(m + \log \hat{x}_{kj})$. Accordingly, the approach will be to see whether one can find a good linear approximation of $g(\cdot)$ such that,

$$g(\hat{x}_{kj}) \approx a\hat{x}_{kj} + b,$$

on a certain range of interest $[\hat{x}_{min}, \hat{x}_{max}]$.

Fortunately, $g(\cdot)$ seems to behave increasingly linear as m grows. Figure 6.3 demonstrates this effect and also shows that the $\hat{x}_{min} = e^{-m-1}$ corresponding to each m (small asterisk) becomes smaller for larger m, making the method more suitable for lower input noise levels.

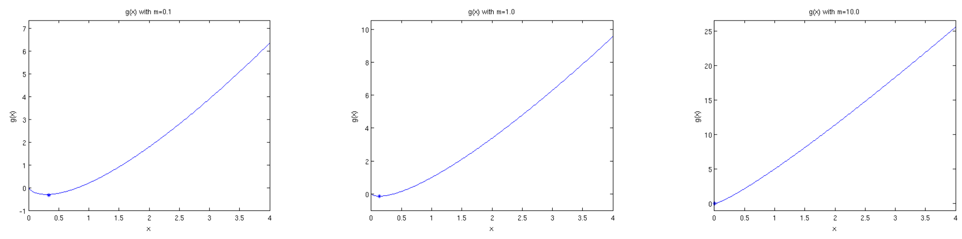


Figure 6.3: Function $g(\cdot)$ for three different parameter shifts m: 0.1, 1 and 10. The higher m, the better the approximation by a linear function.

In the limit one recognizes that,

$$\lim_{m\to\infty} g(\hat{x}_{kj})/m = \lim_{m\to\infty} \hat{x}_{kj}(m + \log \hat{x}_{kj})/m = \hat{x}_{kj}, \tag{6.42}$$

providing a nice explanation of the observation made above. Assuming large m one can therefore write

$$\begin{aligned} g(\hat{x}_{kj}) &\approx m\hat{x}_{kj}, \\ g^{-1}(\psi_{kj}) &\approx \psi_{kj}/m, \\ \sum_j g^{-1}(\psi_{kj}) &\approx 1/m \sum_j \psi_{kj} =: h(\sum_j \psi_{kj}) \end{aligned} \tag{6.43}$$

Following the logic developed above, one might ask whether m can be made arbitrarily large to improve the approximation. In fact, there are some apparent practical considerations concerning the choice of m. First, the EPSPs scale practically linearly with m, so in some implementations there might be a physical limit to m. Also, increasing m leads to a smaller influence of the underlying weight θ_{kj} on the EPSP, giving rise to numerical issues during inference, learning, as well as during the normalization term estimation itself. Altogether it seems reasonable to use intermediate values for m. In practice, $m \in [7, 10]$ seem to work well for most purposes.

Usually the approximation in Eq. 6.43 can be further improved, especially when m is relatively low and inputs are known to occur on a fixed range $[\hat{x}_{min}, \hat{x}_{max}]$. Then, fitting a linear model in the mean squared error sense by minimizing

$$e(a,b) = \int_{\hat{x}_{min}}^{\hat{x}_{max}} \left((ax+b) - g(x)\right)^2 dx,$$

will provide parameters a and b for an approximation,

$$\begin{aligned} g(\hat{x}_{kj}) &\approx a\hat{x}_{kj} + b, \\ g^{-1}(\psi_{kj}) &\approx \psi_{kj}/a - b/a, \\ \sum_j g^{-1}(\psi_{kj}) &\approx 1/a \sum_j \psi_{kj} - Mb/a =: h(\sum_j \psi_{kj}) \end{aligned} \tag{6.44}$$

where M is the number of synapses j for neuron k. In practice, the resulting constant term Mb/a can be omitted as it is canceled during the WTA-normalization.

In summary, the presented approximate method can be easily implemented by equipping each output neuron k with a running estimator $\psi_k \approx \langle \boldsymbol{\theta}'^T_k \mathbf{x} \rangle$, and using a slightly altered computation of the activation function:

$$u_k = \exp(\boldsymbol{\theta}'^T_k \mathbf{x} + \pi_k - \beta\psi_k), \tag{6.45}$$

The constant β will typically be $\beta \ll 1$ and is given by $\beta = 1/m$ or $\beta = 1/a$ for the limit and mean squared error approximations, respectively.

6.3.3 Comparison of Methods

For convenience, Table 6.1 revisits the four methods discussed above shortly. The methods are abbreviated by **UN** (Unnormalized), **EX** (Exact), **A1** and **A2** (Approximation 1 and 2, respectively).

ID	New parameters	New estimators	Activation u_k
UN	–	–	$\boldsymbol{\theta}_k^T \mathbf{x} + \pi_k$
EX	–	–	$\boldsymbol{\theta}_k^T \mathbf{x} + \pi_k - \sum_j e^{\theta_{kj}}$
A1	$m, \quad \theta'_{kj} := \theta_{kj} + m$	$\psi_{kj} \approx \langle \boldsymbol{\theta}'_{kj} x_j \rangle_{p(x\|Z=k)}$	$\boldsymbol{\theta'}_k^T \mathbf{x} + \pi_k - \sum_j \frac{\psi_{kj}}{W(e^m \psi_{kj})}$
A2	$m, \beta, \quad \theta'_{kj} := \theta_{kj} + m$	$\psi_k \approx \langle \boldsymbol{\theta'}_k^T \mathbf{x} \rangle_{p(x\|Z=k)}$	$\boldsymbol{\theta'}_k^T \mathbf{x} + \pi_k - \beta\psi_k$

Table 6.1: Approximation of normalization term: Four methods for inference in the Poisson-model. **UN** ignores the normalization term. **EX** corresponds to the theoretically exact method. **A1** and **A2** estimate the normalization term from the post-synaptic activity.

Below, three of the four methods, **UN**, **EX** and **A2**, will be compared experimentally on two different data sets. In all simulations a constant learning rate η and very small constant noise in the inference step is used to support exploration. The whole data set is provided in the same order, epoch after epoch. The performance measure is the average log-likelihood of the data under the model.

Random Poisson Mixture

Data was generated as a mixture of Poisson patterns,

$$p(Z = k, \mathbf{x}) = \frac{1}{N} p(\mathbf{x} \,| Z = k) = \frac{1}{N} \prod_j \text{Pois}(x_j; \lambda_{kj}), \tag{6.46}$$

15 base patterns were generated by drawing λ_{kj} randomly from $[0, 7]$. From these base patterns, a data set of 10000 patterns was randomly generated according to Eq. 6.46. 20 hidden units were used in all models. For the approximate method, a parameter shift $m = 7$ was used. The results are shown in Figure 6.4. As expected, the exact method performs best, closely followed by the approximation. The unnormalized method quickly falls behind and converges to a suboptimal solution.

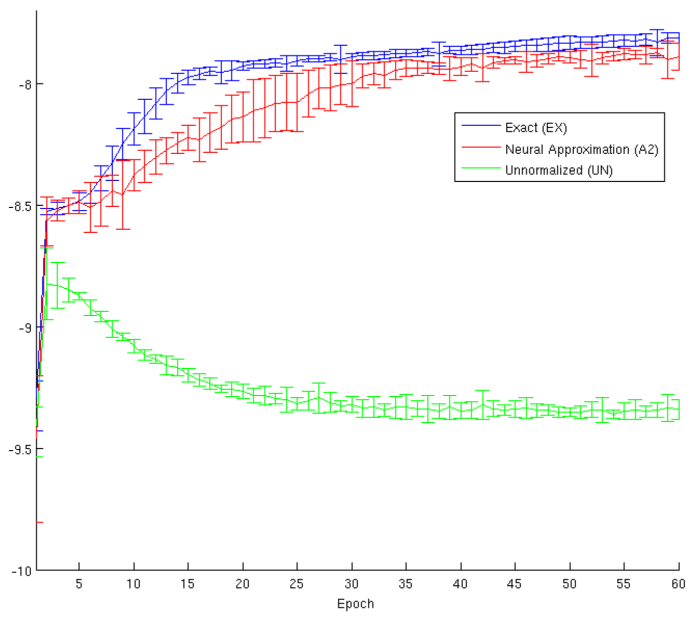


Figure 6.4: Approximation of normalization term: Performance comparison for random Poisson patterns. The data log-likelihood under the model is plotted over the course of simulation for three different methods. Error bars denote standard deviation over 20 trials. The exact method performs best, closely followed by the proposed approximation. Neglecting the approximation term as in the unnormalized method drastically impairs the quality of the learned model.

MNIST without Population Coding

Here the MNIST database digits 0,3,4 from Lecun and Cortes [1998] were used, with values discretized to 0 and 1 as input to the three algorithms. In contrast to Nessler et al. [2010] no population coding was performed prior to application of the method. Note that, since inputs are restricted to 0 and 1, the unnormalized method corresponds exactly to learning and inference as proposed in the original model by Nessler et al. [2010]. 20 hidden units were used in all models.

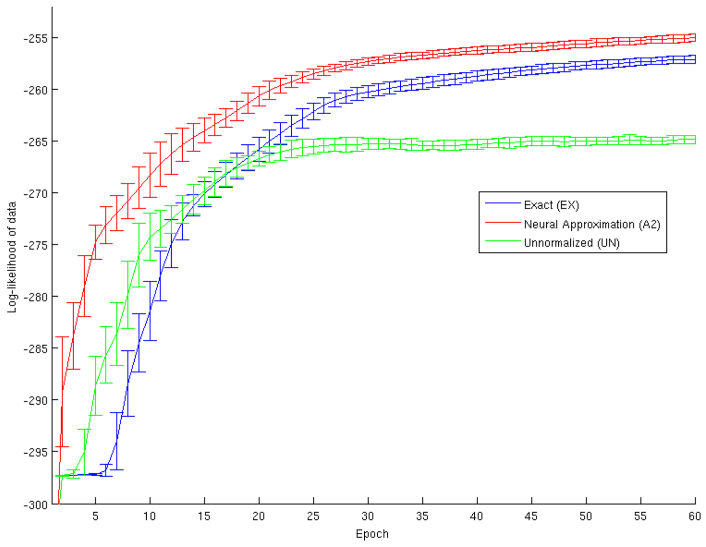


Figure 6.5: Approximation of normalization term: Performance comparison for handwritten digits. The data log-likelihood under the model is plotted over the course of simulation for three different methods. Error bars denote standard deviation over 20 trials. Both exact method and approximation perform well compared to the unnormalized method. The approximation benefits from an initial boost which is related to underestimation of the normalization term in early stages of learning, which in turn facilitates recruitment of competitors for patterns. The unnormalized method converges to an inferior solution.

The exact normalization procedure converges slower during the initial phase than the unnormalized algorithm. However, since the unnormalized version converges to a sub-optimal solution, the effect can only be seen during the first 15 epochs of training.

Note that the approximation converges faster than the exact algorithm. This can be credited to the fact that the approximation initially underestimates the normalization term. This facilitates the recruitment of more competitors for the same pattern in early stages and leads to an accelerated convergence.

Chapter 7

Conclusion and Outlook

In this thesis I have presented my research work at the crossroads of spiking neural networks and probabilistic models. After an introduction to the field, Chapter 5 presented a generalization of the model proposed by Nessler et al. [2010] to exponential family components. Not all exponential families are equally suitable in the context of neurons. The most obvious candidate for the use with spiking neurons is the Poisson distribution. A spiking neural architecture which implements learning and inference according to the Poisson-based model was presented in Section 5.5.2. The proposed generalization preserves several important aspects of the original model. Synaptic weights correspond to model parameters, inference is tractable and is mediated by a weighted sum of inputs, and learning rules are local.

The two most significant benefits of the Poisson model compared to the original model are:

- To a first approximation, spike trains can be understood from the perspective of inhomogeneous Poisson processes (Gerstner and Kistler [2002]). In accordance with this fact, the here proposed method uses Poisson distributions to model inputs, as opposed to categorical or binary variables in the original model.

- The restriction of population coding has been lifted. Arbitrary ensembles of spiking neurons can be directly connected to the network proposed in Section 5.5.2.

Apart from the Poisson special case, the generalization to exponential families entails additional flexibility in the context of spiking input. This could become relevant in the light of deviations from Poisson statistics in natural spike trains which may occur due to neural refractoriness (Gerstner and Kistler [2002]). Exponential families other than Poisson may be even more suitable for biologically realistic networks, depending on the exact nature of input coding.

In Chapter 6, three further extensions to the model were developed. First, it was shown how realistic EPSP shapes can be incorporated in the model. Second, the inclusion of prior information about synaptic weights was discussed. Third, an approximative way of performing inference in the Poisson model was discussed, which aimed at minimizing additional assumptions on how neurons communicate with synapses in spiking neural networks.

The theory developed in this thesis is largely consistent with biological data in some important aspects, such as basic neural integration (Purves [2008]), Spike-Timing Dependent Plasticity (STDP) (Gerstner and Kistler [2002]) and the existence of Winner-Take-All (WTA) circuits in cortex (Douglas and Martin [2004]). Hence, it adds evidence to the hypothesis that spiking neural networks can carry out complex probabilistic computations.

There are some attractive avenues to continue and build on the research work of this thesis. Technically, it would be interesting to consider dynamic input processes with fast rate variations and refractoriness. Also, many aspects of biological spiking neural networks are still not covered by the method. It is unclear whether and how mechanisms like short term plasticity (Zucker and Regehr [2002]) or structural developmental processes (Dehay and Kennedy [2007]) could benefit the performance of spiking neural networks for probabilistic tasks.

Deepening the connections between theory and neuroscientific data will be of utmost importance to guide further improvements of the model. Also, an implementation of the method on special purpose hardware (for example, Furber et al. [2008]) could be taken into consideration.

In the broader picture, the computational capabilities of currently researched spiking neural networks, including the here presented method, are still very limited. The vision of artificial neural networks which are capable of learning and reasoning at the level of real-world complexity, appears yet distant. In order to approach this exciting goal, more powerful probabilistic models need to be brought in relation to spiking neural networks.

Appendix A

Convergence Proof

In the most general case, vector-valued inputs $\hat{\mathbf{x}}$ are provided with sufficient statistics $\mathbf{x} = (T_1(\hat{\mathbf{x}}), \ldots, T_N(\hat{\mathbf{x}}))^T$. Assuming a mixture model with k exponential family components gives,

$$
\begin{aligned}
p(\hat{\mathbf{x}}|\,\mathbf{\Theta}, \boldsymbol{\pi}) &= \sum_k \exp(\pi_k) p(\mathbf{x}\,|z_k = 1, \boldsymbol{\theta}_k) && \text{(A.1)}\\
&= \sum_k \exp(\pi_k) h(\hat{\mathbf{x}}) \exp(\boldsymbol{\theta}_k^T\,\mathbf{x} - A(\boldsymbol{\theta}_k)) && \text{(A.2)}\\
&= h(\hat{\mathbf{x}}) \sum_k \exp(\underbrace{\pi_k + \boldsymbol{\theta}_k^T\,\mathbf{x} - A(\boldsymbol{\theta}_k)}_{u_k}) && \text{(A.3)}\\
&= h(\hat{\mathbf{x}}) \sum_k \exp(u_k) && \text{(A.4)}
\end{aligned}
$$

The normalization or log-partition function $A(\cdot)$ is a scalar function of the parameter vector $\boldsymbol{\theta}_k$, which decomposes into a sum for conditionally independent sufficient statistics $\mathbf{x}$.

Our goal is to maximize the expected log-likelihood of the data under the model, $\langle \log p(\hat{\mathbf{x}}|\,\boldsymbol{\pi}, \boldsymbol{\theta})\rangle_{\hat{\mathbf{x}}}$ under the constraint $\sum_k e^{\pi_k} = 1$. To this end one can define the Lagrangian,

$$L(\boldsymbol{\pi}, \boldsymbol{\Theta}, \lambda) = \langle \log p(\hat{\mathbf{x}}|\, \boldsymbol{\pi}, \boldsymbol{\theta}) \rangle_{p(\hat{\mathbf{x}})} - \lambda(\sum_k e^{\pi_k} - 1), \tag{A.5}$$

and obtain critical points by requiring

$$\nabla L(\boldsymbol{\pi}, \boldsymbol{\Theta}, \lambda) \overset{!}{=} \mathbf{0}. \tag{A.6}$$

As was shown previously, at a local maximum $\lambda = 1$. Using the fact that λ does not depend on parameters or inputs, and dropping terms which are independent of parameters, one can define a simplified objective function

$$l(\boldsymbol{\pi}, \boldsymbol{\Theta}) = \langle \sum_k e^{u_k} \rangle_{p(\mathbf{x})} - \sum_k e^{\pi_k}. \tag{A.7}$$

for which it is easy to show that the local maxima coincide with the original constrained optimization goal. Maximization of $l(\boldsymbol{\pi}, \boldsymbol{\Theta})$ then yields the following necessary conditions:

$$\pi_k \overset{!}{=} \log p(z_k = 1), \tag{A.8}$$

$$\nabla A(\boldsymbol{\theta}_k) \overset{!}{=} \langle \mathbf{x} \rangle_{p(\mathbf{x}\,|z_k=1)}. \tag{A.9}$$

Now consider the following learning rule,

$$\Delta_{\pi_k} = \eta(z_k e^{-\pi_k} - 1), \tag{A.10}$$

$$\Delta_{\boldsymbol{\theta}_k} = \eta\, \mathbf{H}_A^{-1}(\boldsymbol{\theta}_k)\, [\mathbf{x} - \nabla A(\boldsymbol{\theta}_k)]\,. \tag{A.11}$$

$\mathbf{H}_A$ denotes the Hessian matrix of $A(\cdot)$,

$$h_{A,ij}(\boldsymbol{\theta}_k) = \frac{\partial^2 A(\boldsymbol{\theta}_k)}{\partial \theta_{ki} \partial \theta_{kj}}, \tag{A.12}$$

which can be also interpreted as the covariance matrix of the k-th mixture component, since the cumulant-generating function of an exponential family is $g(\mathbf{t}) = A(\boldsymbol{\theta}_k + \mathbf{t}) - A(\boldsymbol{\theta}_k)$, and

$$\mathrm{Cov}(x_i, x_j) = \frac{\partial^2 g(\mathbf{t})}{\partial t_i \partial t_j}|_{\mathbf{t}=\mathbf{0}} = \frac{\partial^2 A(\boldsymbol{\theta}_k)}{\partial \theta_{ki} \partial \theta_{kj}} \tag{A.13}$$

As a consequence, if the inverse of $\mathbf{H}_A(\boldsymbol{\theta}_k)$ exists, it is guaranteed to be symmetric and positive definite, an important property that will be used below[1].

In the following, it will be shown that the expected update of the above learning rule converges to a local optimum of $l(\boldsymbol{\pi}, \boldsymbol{\Theta})$. First note that the equilibrium points of the learning rule coincide with the necessary optimality conditions developed above:

$$\langle \Delta_{\pi_k} \rangle_{p(\mathbf{x})} = 0 \Leftrightarrow \pi_k = \log p(p_k = 1), \tag{A.15}$$

$$\langle \Delta_{\boldsymbol{\theta}_k} \rangle_{p(\mathbf{x})} = \mathbf{0} \Leftrightarrow A(\boldsymbol{\theta}_k) = \langle \mathbf{x} \rangle_{p(\mathbf{x}\,|z_k=1)}. \tag{A.16}$$

From that it is concluded that the expected learning update is always non-zero for non-optimal settings, and zero at local optima. Next, it will be proven that the learning rule always drives the parameters in the right direction, by showing that the dot product $\nabla_{\pi,\boldsymbol{\Theta}} l(\pi, \boldsymbol{\Theta}) \cdot \langle \Delta_{\boldsymbol{\pi},\boldsymbol{\Theta}} \rangle_{p(\mathbf{x})} \geq 0$. The derivative

[1]For more complicated models, for example when the covariance between inputs is directly controlled by a subset of the parameters, there might be pathological cases where the Hessian matrix becomes rank-deficient, such that the inverse cannot be computed. Luckily, the rescaling of the error vector through the inverse Hessian matrix in Eq. A.11 is not essential in terms of maximizing $l(\boldsymbol{\pi}, \boldsymbol{\Theta})$, but rather enforces the learning speed imposed by the current η. In such a rare event one can perform an alternative learning update

$$\Delta_{\boldsymbol{\theta}_k} = \eta\, \mathbf{H}_k^{-1} \left[\mathbf{x} - \nabla A(\boldsymbol{\theta}_k)\right], \tag{A.14}$$

which is still guaranteed to increase data likelihood as long as $\mathbf{H}_k^{-1}$ is a small enough positive-definite matrix, for example a scaled identity matrix or the inverse Hessian computed with the last successful parameter vector.

of $l(\pi, \mathbf{\Theta})$ with respect to π_k is,

$$\frac{\partial l(\boldsymbol{\pi}, \mathbf{\Theta})}{\partial \pi_k} = \langle \frac{\partial [\sum_k' e^{u_k'}]}{\partial \pi_k} \rangle_{p(\mathbf{x})} - e^{\pi_k} \tag{A.17}$$

$$= \langle \frac{e^{u_k}}{\sum_k' e^{u_k'}} \rangle_{p(\mathbf{x})} - e^{\pi_k} \tag{A.18}$$

$$= \langle p(z_k = 1 | \mathbf{x}) \rangle_{p(\mathbf{x})} - e^{\pi_k} \tag{A.19}$$

$$= p(z_k = 1) - e^{\pi_k} \tag{A.20}$$

Similarly, differentiating with respect to $\boldsymbol{\theta}_k$ gives,

$$\frac{\partial l(\boldsymbol{\pi}, \mathbf{\Theta})}{\partial \boldsymbol{\theta}_k} = \langle \frac{\partial [\sum_k' e^{u_k'}]}{\partial \boldsymbol{\theta}_k} \rangle_{p(\mathbf{x})} \tag{A.21}$$

$$= \langle \frac{e^{u_k}}{\sum_k' e^{u_k'}} (\mathbf{x} - \nabla A(\boldsymbol{\theta}_k)) \rangle_{p(\mathbf{x})} \tag{A.22}$$

$$= \langle p(z_k = 1 | \mathbf{x}) (\mathbf{x} - \nabla A(\boldsymbol{\theta}_k))^T \rangle_{p(\mathbf{x})} \tag{A.23}$$

$$= p(z_k = 1) [\langle \mathbf{x} \rangle_{p(\mathbf{x} | z_k = 1)} - \nabla A(\boldsymbol{\theta}_k)]^T \tag{A.24}$$

Then,

$$\nabla_{\boldsymbol{\pi}, \mathbf{\Theta}} l(\boldsymbol{\pi}, \mathbf{\Theta}) \cdot \langle \Delta_{\boldsymbol{\pi}, \mathbf{\Theta}} \rangle_{p(\mathbf{x})} =$$

$$= \sum_k \frac{\partial l(\boldsymbol{\pi}, \mathbf{\Theta})}{\partial \pi_k} \langle \Delta_{\pi_k} \rangle_{p(\mathbf{x})} + \sum_k \frac{\partial l(\boldsymbol{\pi}, \mathbf{\Theta})}{\partial \boldsymbol{\theta}_k} \langle \Delta_{\boldsymbol{\theta}_k} \rangle_{p(\mathbf{x})} \tag{A.25}$$

$$\begin{aligned} &= \sum_k [p(z_k = 1) - e^{\pi_k}] \eta_{k0} [e^{-\pi_k} p(z_k = 1) - 1] + \\ &\quad \sum_k p(z_k = 1) [\langle \mathbf{x} \rangle_{p(\mathbf{x} | z_k = 1)} - \nabla A(\boldsymbol{\theta}_k)]^T \cdot \\ &\quad \cdot \underbrace{\operatorname{diag}(\eta_{k1}, \ldots, \eta_{kj}) \, \mathbf{H}_A(\boldsymbol{\theta}_k)^{-1}}_{\mathbf{B}_k} \underbrace{[\langle \mathbf{x} \rangle_{p(\mathbf{x} | z_k = 1)} - \nabla A(\boldsymbol{\theta}_k)]}_{\mathbf{v}_k} \end{aligned} \tag{A.26}$$

Since both $\mathbf{H}_A(\boldsymbol{\theta}_k)^{-1}$ and $\operatorname{diag}(\eta_{k1}, \ldots, \eta_{kj})$ are symmetric and positive definite, also $\mathbf{B}_k$ is. Consequently,

$$\begin{aligned}
&\nabla_{\boldsymbol{\pi},\boldsymbol{\Theta}} l(\boldsymbol{\pi},\boldsymbol{\Theta}) \cdot \langle \Delta_{\boldsymbol{\pi},\boldsymbol{\Theta}} \rangle_{p(\mathbf{x})} = \\
&\quad = \sum_k \underbrace{\eta_{k0} e^{-\pi_k} [p(z_k = 1) - e^{\pi_k}]^2}_{\geq 0} + \sum_k \underbrace{p(z_k = 1)\, \mathbf{v}_k^T\, \mathbf{B}_k\, \mathbf{v}_k}_{\geq 0} \qquad \text{(A.27)} \\
&\quad \geq 0, \qquad \text{(A.28)}
\end{aligned}$$

completing the proof.

Bibliography

Abbott, L.F. and S.B. Nelson [2000]. *Synaptic plasticity: taming the beast. nature neuroscience*, 3, pages 1178–1183. (Cited on page 13.)

Bi, G.Q. and M.M. Poo [2001]. *Synaptic modification by correlated activity: Hebb's postulate revisited. Annual review of neuroscience*, 24, pages 139–166. (Cited on page 13.)

Bickel, PJ and K. Doksum [2001]. *Mathematical Statistics: Basic Ideas and Selected Topics. 2d. ed. vol. 1 Prentice Hall. Upper Saddle River, NJ.* (Cited on pages 29, 35, 41, 42, 43, 44, 83 and 86.)

Bienenstock, E.L., L.N. Cooper, and P.W. Munro [1988]. *Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex.* In *Neurocomputing: foundations of research*, page 455. MIT Press. (Cited on page 13.)

Bishop, C.M. [2006]. *Pattern recognition and machine learning.* Springer New York. (Cited on pages 1, 2, 8, 24, 26, 27, 29, 30, 33, 34, 36 and 41.)

Blakemore, C. and E.A. Tobin [1972]. *Lateral inhibition between orientation detectors in the cat's visual cortex. Experimental Brain Research*, 15(4), pages 439–440. (Cited on page 16.)

Boyd, S.P. and L. Vandenberghe [2004]. *Convex optimization.* Cambridge Univ Pr. (Cited on page 37.)

Brette, R., M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J.M. Bower, M. Diesmann, A. Morrison, P.H. Goodman, F.C. Harris, et al. [2007]. *Simulation of networks of spiking neurons: a review of tools and strategies. Journal of computational neuroscience*, 23(3), pages 349–398. (Cited on pages 18 and 19.)

Carlson, N.R. and J. Braun [2002]. *Foundations of physiological psychology.* Allyn and Bacon Boston, MA. (Cited on pages 1 and 5.)

Chapeau-Blondeau, F. and A. Monir [2002]. *Numerical evaluation of the Lambert W function and application to generation of generalized Gaussian noise with exponent 1/2. Signal Processing, IEEE Transactions on*, 50(9), pages 2160–2165. ISSN 1053-587X. doi:10.1109/TSP.2002.801912. (Cited on page 94.)

Churchland, P.S. and T.J. Sejnowski [1994]. *The computational brain.* The MIT press. (Cited on page 6.)

Conradt, J., R. Berner, M. Cook, and T. Delbruck [2009]. *An Embedded AER Dynamic Vision Sensor for Low-Latency Pole Balancing.* In *IEEE Workshop on Embedded Computer Vision (ECV09).* (Cited on page 20.)

Coultrip, R., R. Granger, and G. Lynch [1992]. *A cortical model of winner-take-all competition via lateral inhibition*. Neural Networks*, 5(1), pages 47–54. (Cited on page 17.)

Dan, Y. and M.M. Poo [2006]. *Spike timing-dependent plasticity: from synapse to perception. Physiological reviews*, 86(3), page 1033. (Cited on page 14.)

David R. Clark, Charles A. Thayer [2004]. *A Primer on the Exponential Family of Distributions. CAS Discussion Paper Program*, pages 117–148. (Cited on page 45.)

Davison, A.P., D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger [2008]. *PyNN: a common interface for neuronal network simulators. Frontiers in Neuroinformatics*, 2. (Cited on page 18.)

Dehay, C. and H. Kennedy [2007]. *Cell-cycle control and cortical development. Nature Reviews Neuroscience*, 8(6), pages 438–450. (Cited on page 102.)

Delbruck, T. and P. Lichsteiner [2006]. *Freeing vision from frames. Neuromorphic Eng*, 3, pages 3–4. (Cited on page 20.)

Dellaert, F. [2002]. *The expectation maximization algorithm. Georgia Institute of Technology, Technical Report Number GIT-GVU-02-20.* (Cited on page 38.)

Dempster, A.P., N.M. Laird, D.B. Rubin, et al. [1977]. *Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1), pages 1–38. (Cited on pages 35 and 41.)

Deneve, S. [2008a]. *Bayesian spiking neurons I: inference. Neural computation*, 20(1), pages 91–117. (Cited on pages 2, 47 and 52.)

Deneve, S. [2008b]. *Bayesian spiking neurons II: learning. Neural computation*, 20(1), pages 118–145. (Cited on pages 47 and 53.)

Diesmann, M. and M.O. Gewaltig [2002]. *NEST: An Environment for Neural Systems Simulations. Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen*, pages 43–70. (Cited on page 19.)

Douglas, RJ and KA Martin [2004]. *Neuronal circuits of the neocortex. Annual review of neuroscience*, 27, page 419. (Cited on pages 16, 17 and 102.)

Doya, K. [2007]. *Bayesian brain: Probabilistic approaches to neural coding.* The MIT Press. (Cited on page 24.)

Drubach, D. [1999]. *The brain explained.* Prentice Hall. (Cited on page 8.)

Ehrlich, M., C. Mayr, H. Eisenreich, S. Henker, A. Srowig, A. Grübl, J. Schemmel, and R. Schüffny [2007]. *Wafer-scale VLSI implementations of pulse coupled neural networks.* In *Proceedings of the International Conference on Sensors, Circuits and Instrumentation Systems.* (Cited on page 20.)

Fiete, I.R., W. Senn, C.Z.H. Wang, and R.H.R. Hahnloser [2010]. *Spike-time-dependent plasticity and heterosynaptic competition organize networks to produce long scale-free sequences of neural activity. Neuron*, 65(4), pages 563–576. (Cited on page 79.)

Filler, A.G. [2009]. *The History, Development and Impact of Computed Imaging in Neurological Diagnosis and Neurosurgery: CT, MRI, and DTI.* (Cited on page 7.)

Furber, S. et al. [2008]. *SpiNNaker: Mapping NNs onto a MassivelyParallel Chip Multiprocessor.* In *IEEE World Congress on Computational Intelligence.* (Cited on pages 20, 21 and 103.)

Gerstner, W. [2001]. *A framework for spiking neuron models: The spike response model. Handbook of Biological Physics*, 4, pages 469–516. (Cited on page 11.)

Gerstner, Wulfram and Werner M. Kistler [2002]. *Spiking Neuron Models.* 1 Edition. Cambridge University Press. ISBN 0521890799. `http://www.amazon.com/exec/obidos/redirect?tag=`

citeulike07-20&path=ASIN/0521890799. (Cited on pages 10, 52, 101 and 102.)

Ghahramani, Z. and M. Jordan [1997]. *Mixture models for learning from incomplete data. MIT Press*, 4, pages 67–85. (Cited on page 37.)

Gilles Celeux, Didier Chauveau, and Jean Diebolt [1995]. *On Stochastic Versions of the EM Algorithm*. Research Report RR-2514, INRIA. http://hal.inria.fr/inria-00074164/en/. (Cited on page 75.)

Goodman, D. and R. Brette [2008]. *Brian: a simulator for spiking neural networks in Python. Frontiers in Neuroinformatics*, 2. (Cited on page 19.)

Griffiths, T.L. and J.B. Tenenbaum [2006]. *Optimal predictions in everyday cognition. Psychological Science*, 17(9), page 767. (Cited on page 24.)

Grossberg, S. [1973]. *Contour enhancement, short term memory, and constancies in reverberating neural networks. Studies in applied Mathematics*, 52(3), pages 213–257. (Cited on page 16.)

Hammerstrom, D. [1998]. *Digital VLSI for neural networks*. In *The handbook of brain theory and neural networks*, page 309. MIT Press. (Cited on page 20.)

Hand, D.J. and K. Yu [2001]. *Idiot's Bayes-not so stupid after all? International Statistical Review*, 69(3), pages 385–398. (Cited on pages 30 and 32.)

Haykin, S. [1994]. *Neural networks: a comprehensive foundation*. Prentice Hall PTR Upper Saddle River, NJ, USA. (Cited on pages 1 and 8.)

Hebb, Donald O. [1949]. *The Organization of Behavior: A Neuropsychological Theory*. New edition Edition. Wiley, New York. ISBN 0805843000. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0805843000. (Cited on page 13.)

Hines, M.L. and N.T. Carnevale [1997]. *The NEURON simulation environment. Neural computation*, 9(6), pages 1179–1209. (Cited on page 18.)

Hodges, A. [2007]. *Alan Turing. Stanford Encyclopedia of Philosophy*. (Cited on page 6.)

Houtgast, T. [1972]. *Psychophysical evidence for lateral inhibition in hearing. The Journal of the Acoustical Society of America*, 51, page 1885. (Cited on page 16.)

I.B.M.B.G. Team [2008]. *Overview of the IBM Blue Gene/P project. IBM Journal of Research and Development*, 52(1/2), pages 199–220. (Cited on page 8.)

Izhikevich, E.M. [2004]. *Which model to use for cortical spiking neurons? IEEE transactions on neural networks*, 15(5), pages 1063–1070. (Cited on page 11.)

Jaynes, E.T. and G.L. Bretthorst [2003]. *Probability theory: the logic of science.* Cambridge Univ Pr. (Cited on page 23.)

Jolivet, R., R. Kobayashi, A. Rauch, R. Naud, S. Shinomoto, and W. Gerstner [2008]. *A benchmark test for a quantitative assessment of simple neuron models. Journal of neuroscience methods*, 169(2), pages 417–424. (Cited on page 12.)

Jolivet, R., A. Rauch, H.R. Lüscher, and W. Gerstner [2006]. *Predicting spike timing of neocortical pyramidal neurons by simple threshold models. Journal of computational neuroscience*, 21(1), pages 35–49. (Cited on page 12.)

Jones, E.G. and A. Peters [1990]. *Comparative structure and evolution of cerebral cortex.* 269–280 pages. (Cited on page 15.)

Jordan, M.I. and S. Russell [1999]. *Computational intelligence. The MIT encyclopedia of the cognitive sciences.* (Cited on page 23.)

Judenhofer, M.S., H.F. Wehrl, D.F. Newport, C. Catana, S.B. Siegel, M. Becker, A. Thielscher, M. Kneilling, M.P. Lichy, M. Eichner, et al. [2008]. *Simultaneous PET-MRI: a new approach for functional and morphological imaging. Nature medicine*, 14(4), pages 459–465. (Cited on page 7.)

Koch, C. and K. Hepp [2006]. *Quantum mechanics in the brain. Nature*, 440(7084), page 611. (Cited on page 7.)

Kohonen, T. [1990]. *The self-organizing map. Proceedings of the IEEE*, 78(9), pages 1464–1480. (Cited on page 17.)

Lecun, Yann and Corinna Cortes [1998]. *The MNIST database of handwritten digits.* `http://yann.lecun.com/exdb/mnist/`. (Cited on pages 51 and 98.)

London, M., A. Roth, L. Beeren, M. Häusser, and P.E. Latham [2010]. *Sensitivity to perturbations in vivo implies high noise and suggests rate coding in cortex. Nature*, 466(7302), pages 123–127. (Cited on pages 20 and 56.)

Maass, W. [1997]. *Networks of spiking neurons: the third generation of neural network models*. *Neural Networks*, 10(9), pages 1659–1671. (Cited on page 10.)

Maass, W. [2000]. *On the computational power of winner-take-all*. *Neural Computation*, 12(11), pages 2519–2535. (Cited on page 17.)

Maass, W., T. Natschläger, and H. Markram [2002]. *Real-time computing without stable states: A new framework for neural computation based on perturbations*. *Neural computation*, 14(11), pages 2531–2560. (Cited on page 19.)

Maass, Wolfgang [2004]. *On the computational power of circuits of spiking neurons*. *Journal of Computer and System Sciences*, 69(4), pages 593–616. ISSN 00220000. doi:10.1016/j.jcss.2004.04.001. http://dx.doi.org/10.1016/j.jcss.2004.04.001. (Cited on page 10.)

Maass, Wolfgang and Christopher M. Bishop [2001]. *Pulsed Neural Networks*. The MIT Press. ISBN 978-0-262-63221-8. (Cited on pages 1, 2, 10 and 15.)

MacKay, D.J.C. [2003]. *Information theory, inference, and learning algorithms*. Cambridge Univ Pr. (Cited on pages 2, 24, 25, 26, 27 and 34.)

Mahowald, M. [1992]. *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. PhD Thesis, California Institute of Technology. (Cited on page 20.)

Markram, H. [2006]. *The blue brain project*. *Nature Reviews Neuroscience*, 7(2), pages 153–160. (Cited on page 7.)

Martin, SJ, PD Grimwood, and RG Morris [2000]. *Synaptic plasticity and memory: an evaluation of the hypothesis*. *Annual review of neuroscience*, 23, page 649. (Cited on page 12.)

Meier, K. [2005]. *The FACETS Project*. http://facets.kip.uni-heidelberg.de. (Cited on page 8.)

Miesenbock, G. [2009]. *The optogenetic catechism*. *Science*, 326(5951), page 395. (Cited on page 7.)

Monchi, O., H. Benali, J. Doyon, and A.P. Strafella [2008]. *Recent Advances in Neuroimaging Methods*. *International Journal of Biomedical Imaging*, 2008. (Cited on page 7.)

Moore, G. E. [1965]. *Cramming More Components onto Integrated Circuits. Electronics*, 38(8), pages 114–117. doi:10.1109/JPROC.1998.658762. `http://dx.doi.org/10.1109/JPROC.1998.658762`. (Cited on page 7.)

Morris, C.N. [1982]. *Natural exponential families with quadratic variance functions. The Annals of Statistics*, 10(1), pages 65–80. (Cited on page 45.)

Morton, P.A. [1997]. *A historical introduction to the philosophy of mind: readings with commentary*. Broadview Pr. (Cited on pages 1 and 6.)

Mountcastle, V.B. [1957]. *Modality and topographic properties of single neurons of cat's somatic sensory cortex. J Neurophysiol*, 20(4), pages 408–434. (Cited on page 15.)

Mountcastle, V.B. [1997]. *The columnar organization of the neocortex. Brain*, 120(4), page 701. (Cited on page 15.)

Murray, A.F. and A.V.W. Smith [1988]. *Asynchronous VLSI neural networks using pulse-stream arithmetic. IEEE Journal of Solid-State Circuits*, 23(3), pages 688–697. (Cited on page 20.)

Murre, J.M.J. and D.P.F. Sturdy [1995]. *The connectivity of the brain: Multi-level quantitative analysis. Biological Cybernetics*, 73(6), pages 529–545. (Cited on page 10.)

Narayan, S. [2009]. *Supercomputers: past, present and the future. Crossroads*, 15(4), pages 7–10. (Cited on page 8.)

Neal, Radford and Geoffrey E. Hinton [1998]. *A View Of The Em Algorithm That Justifies Incremental, Sparse, And Other Variants*. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers. (Cited on pages 74 and 75.)

Nessler, B., M. Pfeiffer, and W. Maass [2010]. *STDP enables spiking neurons to detect hidden causes of their inputs*. In *Proc. of NIPS 2009: Advances in Neural Information Processing Systems*, volume 22, pages 1357–1365. MIT Press. (Cited on pages 2, 3, 47, 51, 55, 56, 68, 76, 77, 78, 81, 98 and 101.)

Neves, G., S.F. Cooke, and T.V.P. Bliss [2008]. *Synaptic plasticity, memory and the hippocampus: a neural network approach to causality. Nature Reviews Neuroscience*, 9(1), pages 65–75. (Cited on page 14.)

Oja, E. [1982]. *Simplified neuron model as a principal component analyzer. Journal of mathematical biology*, 15(3), pages 267–273. (Cited on page 13.)

Pearl, J. [1988]. *Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA*, page 552. (Cited on page 23.)

Pecevski, D., T. Natschläger, and K. Schuch [2009]. *PCSIM: a parallel simulation environment for neural circuits fully integrated with Python.* (Cited on page 19.)

Pelgrom, M.J.M., H.P. Tuinhout, M. Vertregt, et al. [1998]. *Transistor matching in analog CMOS applications. IEDM Tech. Dig*, pages 915–918. (Cited on page 21.)

Petersen, R.S., S. Panzeri, and M.E. Diamond [2001]. *Population coding of stimulus location in rat somatosensory cortex. Neuron*, 32(3), pages 503–514. (Cited on page 9.)

Purves, Dale [2008]. *Neuroscience.* 4th Edition. Sinauer. ISBN 0878936971. `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0878936971`. (Cited on pages 1, 3, 9, 11, 15, 81 and 102.)

Rakic, P. [2008]. *Confusing cortical columns. Proceedings of the National Academy of Sciences of the United States of America*, 105(34), pages 12099–12100. (Cited on page 15.)

Rosenblatt, F. [1958]. *The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review*, 65(6), pages 386–408. (Cited on page 8.)

Rumelhart, D.E. and D. Zipser [1985]. *Feature discovery by competitive learning. Cognitive Science*, 9(1), pages 75–112. (Cited on page 17.)

Salakhutdinov, R., S. Roweis, and Z. Ghahramani [2003]. *Optimization with EM and expectation-conjugate-gradient.* In *Machine Learning-Inernational Workship Then Conference.*, volume 20, page 672. (Cited on page 36.)

Sanner, M.F. [1999]. *Python: a programming language for software integration and development. J. Mol. Graphics Mod*, 17, pages 57–61. (Cited on page 18.)

Satyanarayana, S., Y.P. Tsividis, and H.P. Graf [1992]. *A reconfigurable VLSI neural network. IEEE Journal of Solid-State Circuits*, 27(1), pages 67–81. (Cited on page 20.)

Serrano-Gotarredona, R., M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas, T. Delbrück, S-C. Liu, R. Douglas, P. Häfliger, G. Jimenez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco [2009]. *CAVIAR: A 45K-neuron, 5M-synapse, 12G-connects/sec AER hardware sensory-processing-learning-actuating system for high speed visual object recognition and tracking. IEEE Transactions on Neural Networks*, 20(9), pages 1417–1438. (Cited on page 20.)

Sherrington, C. [1941]. *Man on his Nature. The Journal of Nervous and Mental Disease*, 94(6), page 762. (Cited on page 5.)

Silberberg, G., A. Gupta, and H. Markram [2002]. *Stereotypy in neocortical microcircuits. Trends in Neurosciences*, 25(5), pages 227–230. (Cited on page 16.)

Smullyan, R.M. [1961]. *Theory of formal systems.* Princeton University Press, 1–6 pages. (Cited on page 23.)

Turing, A.M. [1937]. *On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society*, 2(1), page 230. (Cited on page 6.)

Tweedie, M. C. K. [1984]. *An index which distinguishes between some important exponential families.* In Ghosh, J. K. and J. Roy (Editors), *Statistics: Applications and New Directions.* Proceedings of the Indian Statistical Institute Golden Jubilee International Conference, Calcutta: Indian Statistical Institute. (Cited on page 45.)

van Schaik, A. and S. C. Liu [2005]. *AER EAR: A matched silicon cochlea pair with address event representation interface.* In *IEEE International Symposium on Circuits and Systems*, volume V. (Cited on page 20.)

von der Malsburg, C. [1973]. *Self-organization of orientation sensitive cells in the striate cortex. Kybernetik*, 14(2), pages 85–100. (Cited on page 17.)

Yang, T. and M.N. Shadlen [2007]. *Probabilistic reasoning by neurons. Nature*, 447(7148), pages 1075–1080. (Cited on page 24.)

Zucker, RS and WG Regehr [2002]. *Short-term synaptic plasticity. Annual review of physiology*, 64, page 355. (Cited on page 102.)