Bayesian and Information-theoretic Tools for Neuroscience

PhD Thesis

Dominik M. Endres

dme2@st-andrews.ac.uk

School of Psychology, University of St Andrews,

St Andrews KY16 9JP, U.K.

October 14, 2006

Abstract

The overarching purpose of the studies presented in this report is the exploration of the uses of information theory and Bayesian inference applied to neural codes. Two approaches were taken: Starting from first principles, a coding mechanism is proposed, the results are compared to a biological neural code. Secondly, tools from information theory are used to measure the information contained in a biological neural code.

Chapter 3: The REC model proposed by Harpur and Prager [33] codes inputs into a sparse, factorial representation, maintaining reconstruction accuracy. Here I propose a modification of the REC model to determine the optimal network dimensionality. The resulting code for unfiltered natural images is accurate, highly sparse and a large fraction of the code elements show localized features. Furthermore, I propose an activation algorithm for the network that is faster and more accurate than a gradient descent based activation method. Moreover, it is demonstrated that asymmetric noise promotes sparseness.

Chapter 4: A fast, exact alternative to Bayesian classification is introduced. Computational time is quadratic in both the number of observed data points and the number of degrees of freedom of the underlying model. As an example application, responses of single neurons from high-level visual cortex (area STSa) to rapid sequences of complex visual stimuli are analyzed.

Chapter 5: I present an exact Bayesian treatment of a simple, yet sufficiently general probability distribution model. The model complexity, exact values of the expectations of entropies and their variances can be computed with polynomial effort given the data. The expectation of the mutual information becomes thus available, too, and a strict upper bound on its variance. The resulting algorithm is first tested on artificial data. To that end, an information theoretic similarity measure is derived. Second, the algorithm is demonstrated to be useful in neuroscience by studying the information content of the neural responses analyzed in the previous chapter. It is shown that the information throughput of STS neurons is maximized for stimulus durations ≈ 60ms.

Declarations

I, Dominik Endres, hereby certify that this thesis, which is approximately 30000 words in length, has been written by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree.

October 14, 2006 signature:

I was admitted as a part-time research student in Feb. 1998 and as a candidate for the degree of PhD in Feb. 1998; the higher study for which this is a record was carried out in the University of St Andrews between 1998 and 2004.

October 14, 2006 signature:

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

date: signature of supervisor:

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker.

October 14, 2006 signature:

Contents

Chapter 1

Introduction

We live in the information age: one can hardly look around and not see some device designed for the storage, transmission and processing of information. Computers in every office, TVs and radios at home, PDAs and pagers in many people's pockets. In case one happens to forget this basic truth about modern life, the obnoxious ringtone of some mobile phone is guaranteed to remind him or her rather sooner than later.

Yet the most complex known information processing system, the one which helped to design and build the aforementioned ones, is still poorly understood: the human brain. The advancement of its understanding is the objective of neuroscience, which has gained tremendous momentum in the past few decades. Neuroscience is not limited to the study of the human brain: the nervous systems of other species are also under investigation.

1.1 Tools for neuroscience: information theory and Bayesian inference

There are many facets of neuroscience. Unlike more 'classical' disciplines like physics or chemistry, which have had relatively clearly defined domains of study, neuroscience has attracted researches with various and seemingly disparate backgrounds. The reasons for that are manyfold. One probable cause is that it is a relatively young field. Perhaps more important, though, is the fact that the brain can be approached on many different levels, ranging from the sub-atomic to the macroscopic, or from the purely descriptive to the study of its functional organization.

One of the aims of this thesis is to contribute to the understanding of the way in which the brain processes visual information. To that end, methods and principles from two other disciplines will be employed: **information theory** may loosely be defined as the mathematical study of information transmission and storage. Its fundamental concepts were developed by C. Shannon [83]. Probably his greatest contribution was the realization that information, properly defined, can be measured. Consequently, he developed what he called the 'mathematical theory of communication' [82]. Information theory will be used in two ways in this thesis:

1. In chapter 3, a (visual) coding scheme will be motivated by information-theoretic principles. The resulting codes turn out to be similar to those employed by the early processing stages of the mammalian visual system. Furthermore, it will be demonstrated that the presence of noise in the information transmission process promotes sparse codes.

2. In chapter 5, a method will be developed for the information theoretic analysis of small and noisy datasets. As an example application, it will be applied to responses of neurons in area STSa (a high level visual cortex area) to complex visual stimuli.

Bayesian inference is the provably best way of reasoning in the presence of uncertain and incomplete information [54]. It is named after its founding father, Rev. T. Bayes [5], who pondered the question of how to determine whether a coin used for tossing is 'fair', i.e. shows heads or tails with equal probability. After a long period of hibernation, it has enjoyed a revival in the last century [14, 43]. In this thesis, it will be used to develop an optimal method for information extraction from neural spike trains (more generally speaking, the algorithm presented in chapter 4 is an exact Bayesian classification scheme). Moreover, it is also an essential ingredient of the method for information theoretic response analysis (chapter 5).

Chapter 2 contains a brief overview of the mathematical methods needed for the understanding of the following chapters, along with a short introduction to Bayesian inference and information theory.

1.2 An apology to the reader

I hope that the work presented in this thesis is found interesting by neuroscientists, and that the developed methods may be considered useful for future research. Since I am very hesitant to use a method which I do not fully understand, and assuming that the reader feels similarly, I placed heavy emphasis on explaining the methods in detail. The resulting presentational style may be deemed lengthy and ponderous by a pure mathematician. However, to the mathematically inclined neuroscientist, mathematics is a research tool, not an end in itself. Thus, many things which appear 'trivial' and 'obvious' to the former, may require explanation if to be understood by the latter without undue effort. In finding the right balance between brevity and clarity, I am heavily indebted to Peter Földiák and Johannes Schindelin for their feedback. Most lengthier calculations and proofs have been relegated to various appendices.

1.3 Publications

The main findings of chapter 3 have been printed [20] and presented at the *ICANN'99* conference. Large parts of chapter 5 have appeared in the *IEEE Transactions on Information Theory* [22, 21]. The original work contained in these publications has for the most part been carried out by me, the co-authors contributed by posing the problems and, in the case of [22], by helping me with the correct mathematical formulation. We plan to publish another article focussing on the neuroscientific results obtained with the Bayesian Bin Distribution Inference method.

Moreover, a publication containing the theoretical development of chapter 4 and the resulting neuroscientific findings has been submitted.

1.4 Acknowledgments

While a thesis is written by one person, it is contributed to by many. First and foremost, I'd like to thank Peter Földiák. At the beginning of my work at St. Andrews, he posed a number of challenging questions to me, some of which subsequently turned into the research projects that are documented here. He let me develop my own ideas, always providing constructive criticism and pointing me in the right direction whenever I got stuck. I very much enjoyed his style of supervision, and sincerely hope that I did not overly strain his patience, when my theoretical investigations took me a little off course at times.

The neurophysiological recordings used in this thesis are a result of an earlier project by Christian Keysers, David Perret, Peter Földiák and Dengke Xiao. I am grateful to D. Xiao and C. Keysers for supplying me with the data and explaining the formatting details to me. Furthermore, I'd like to thank D. Perret and Mike Oram for their feedback on my ideas and results. I would like to thank M. Oram for his constructive comments on my first-year report, too.

I also owe a debt of gratitude to Johannes Schindelin, with whom I enjoyed many stimulating discussions, who proofread drafts of this thesis and who helped me to clarify several notational issues.

Moreover, I'm very thankful to Jennifer, my wife, for her patience and support all along. Finally, I'd like to thank my parents for all their support throughout my education.

Chapter 2

Methods

The research presented in this report was carried out using a variety of computational methods. In this chapter, the key concepts necessary to comprehend their working are outlined. For a more in-depth understanding, the reader is referred to the references given for each method. However, basic knowledge of linear algebra, one-dimensional calculus and probability theory are required.

2.1 Notation

Unless stated otherwise, the following notation will be used throughout this thesis:

- x, y, z denote real variables.
- $\vec{x}$ is a vector, whose components are real numbers.
- $\mathbf{A}$ is a $M \times N$ matrix.
- A,B,C are propositions, e.g. A='A neuron in area STSa begins transmitting stimulus-related information 100 ms after the stimulus onset'.
- $P(A|B)$ the conditional probability that proposition A is true given that proposition B is true.
- Logical operators: $AB = A$ and B, $A + B = A$ or B, $\bar{A} =$ not A, $A \Rightarrow B = A$ implies B.
- $p(x)$ is the probability density of x. Likewise $p(\vec{x})$. Probability densities will be denoted by a lower-case p, probabilities by P.

2.2 Gradient descent

A number of learning algorithms for artificial neural networks can be formulated as a minimization problem of a function of many variables (i.e. weights, biases etc.). In many cases, this function depends on its variables in a differentiable way. A necessary condition for a minimum of a differentiable function $f(x)$ of one variable defined on $I\!R$ is that it's first derivative be zero at x_{min}, the point where the minimum is located:

$$f'(x_{min}) = \left.\frac{df(x)}{dx}\right|_{x=x_{min}} = 0 \tag{2.1}$$

If the function depends on more than one variable, the concept of the derivative needs to be suitably generalized for this condition to be applicable. This generalization is the *gradient*. The following derivation aims at making the idea plausible, a strict mathematical treatment can be found in [9].

Since the derivative $f'(x_0)$ is the incline of a tangent on $f(x)$ at x_0, the function can be approximated by

$$f(x_0 + \Delta x) \approx f(x_0) + \Delta f(x_0) \tag{2.2}$$

where

$$\Delta f(x_0) = f'(x_0)\Delta x \tag{2.3}$$

This approximation (also known as a *Taylor expansion* to first order, see [9]) gets the better, the smaller Δx.

Consider now a function $g(x_1, x_2, \ldots, x_N)$ of N variables. Here, eqn. 2.2 becomes

$$g(x_1+\Delta x_1, x_2+\Delta x_2, \ldots, x_N+\Delta x_N) \approx g(x_1, x_2, \ldots, x_N)+\Delta g(x_1, x_2, \ldots, x_N) \tag{2.4}$$

where

$$\Delta g(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{N} \frac{\partial g(x_1, x_2, \ldots, x_N)}{\partial x_i}\Delta x_i \tag{2.5}$$

$\frac{\partial g}{\partial x_i}$ is the *partial derivative* of $g(x_1, x_2, \ldots, x_N)$ with respect to x_i. It is computed by the same rules as the derivative of a function of one variable, all variables $x_j, j \in \{1, 2, \ldots, N\}$ where $j \neq i$ are treated as constants. Example: Let

$$g(x_1, x_2) = x_1^2 + x_1x_2 + 2x_2^2 \tag{2.6}$$

Then

$$\frac{\partial g(x_1, x_2)}{\partial x_1} = 2x_1 + x_2 \tag{2.7}$$

and

$$\frac{\partial g(x_1, x_2)}{\partial x_2} = x_1 + 4x_2 \tag{2.8}$$

Lumping all the x_i, Δx_i and $\frac{\partial g}{\partial x_i}$ together in three vectors, $\vec{x} = (x_1, x_2, \ldots, x_N)$, $\vec{\Delta x} = (\Delta x_1, \Delta x_2, \ldots, \Delta x_N)$ and $\vec{h} = (\frac{\partial g(x_1,x_2,\ldots,x_N)}{\partial x_1}, \frac{\partial g(x_1,x_2,\ldots,x_N)}{\partial x_2}, \ldots, \frac{\partial g(x_1,x_2,\ldots,x_N)}{\partial x_N})$, eqn. 2.5 becomes

$$\Delta g(\vec{x}) = \vec{h}\vec{\Delta x}^T \tag{2.9}$$

i.e. $\Delta g(\vec{x})$ is the inner product of $\vec{h}$ and $\vec{\Delta x}$. Now one could ask: Which vector $\vec{\Delta x}$ gives rise to the greatest $\Delta g(\vec{x})$? Clearly, the longer $\vec{\Delta x}$, the greater the absolute value of $\Delta g(\vec{x})$. So instead, let's try to find the $\vec{\Delta x}$ which produces the greatest $\Delta g(\vec{x})$ amongst all possible $\vec{\Delta x}$ of equal length. Since the inner product of two vectors can also be expressed by their lengths and the angle between them, another form of eqn. 2.9 is

$$\Delta g(\vec{x}) = |\vec{h}||\vec{\Delta x}|\cos(\alpha) \tag{2.10}$$

where α is the angle between $|\vec{h}|$ and $|\vec{\Delta x}|$. As the length of $\vec{\Delta x}$ is fixed, $\Delta g(\vec{x})$ will assume its maximum when the cosine of α is maximal, i.e. at $\alpha = 0$. In other words, $\Delta g(\vec{x})$ is maximal, if $\vec{\Delta x}$ is parallel to $\vec{h}$ and oriented in the same direction. Moreover, $\Delta g(\vec{x})$ is directly proportional to $|\vec{h}|$. Since eqn. 2.4 states that $\Delta g(\vec{x})$ is the change of $g(\vec{x})$ brought about by a small displacement given by $\vec{\Delta x}$, the direction of $\vec{h}$ tells us in which direction to go for maximal increase of $g(\vec{x})$, its length is a measure for the magnitude of this increase.

The vector $\vec{h}$ is called the *gradient* of $g(\vec{x})$, it is usually denoted by

$$\nabla g(\vec{x}) = \frac{\partial g(\vec{x})}{\partial \vec{x}} = (\frac{\partial g(\vec{x})}{\partial x_1}, \frac{\partial g(\vec{x})}{\partial x_2}, \ldots, \frac{\partial g(\vec{x})}{\partial x_N}) \tag{2.11}$$

It shares (and generalizes) the following qualities with (of) the derivative $f'(x)$ of a function $f(x)$ of one variable:

1. $\nabla g(\vec{x})$ points in the direction of the steepest incline of $g(\vec{x})$. In the one-dimensional case, there are only two directions: To the left and to the right. If $f'(x) > 0$, then $f(x)$ increases when going to the right, if $f'(x) < 0$, it decreases.

2. $|\nabla g(\vec{x})|$ is a measure of the magnitude of the increase. $|f'(x)|$ is a measure of the steepness of the slope of $f(x)$.

3. $f'(x) = 0$ is a necessary condition for an extremum of $f(x)$. Similarly, when $\nabla g(\vec{x}) = \vec{0}$, there is no direction (to first order) in which $g(\vec{x})$ increases at this point - which is a necessary condition for an extremum in a more-than-one dimensional space as well.

In order to find a minimum of $g(\vec{x})$, one could, starting from some point $\vec{x}_0$, follow the direction of the negative gradient (i.e. the direction of the steepest decline) until $|\nabla g(\vec{x})| = 0$. This is the basic idea behind all gradient descent algorithms. In practice, however, it will be difficult to hit exactly the point where the gradient vanishes, because a step cannot be infinitesimally small, since then the algorithm would take infinitely long to reach the minimum. Hence, a downward step is usually taken to be proportional to the gradient (so that steps get smaller when the minimum is approached) and some step size parameter η (the 'learning rate' in the neural networks community). The termination criterion is commonly of the form $|\nabla g(\vec{x})| < \epsilon$, where ϵ is some small positive constant. A typical implementation of a simple gradient descent algorithm would hence look somewhat like this:

1. Choose $\vec{x}_0$ (starting point), η (step size), and ϵ (termination threshold).
2. Set $\vec{x} = \vec{x}_0$
3. Set $\vec{\Delta x} = -\eta \nabla g(\vec{x})$
4. Set $\vec{x} = \vec{x} + \vec{\Delta x}$
5. if $|\nabla g(\vec{x})| \geq \epsilon$, goto step 3
6. Return $\vec{x}$

Additionally, it is common to define a maximum number of steps after which the algorithm terminates, even if ϵ has not been reached.

2.3 Probability theory

Logic, which was first systematically developed by Aristotle in the 4th century B.C., provides the scientist with the conceptual tools of deductive reasoning. Given certain propositions, it enables one to draw conclusions. However, in natural science, as opposed to mathematics, one does seldom (if ever) have the luxury of certainty, which is necessary for its application. When conducting measurements, factors beyond the experimenter's control will inevitably influence the results. What's worse, these influences will usually not only be uncontrollable, they will also be unpredictable. Thus, another mode of reasoning is required, one that enables us to draw the best possible conclusions given the uncertitudes we have to contend with. This 'doctrine of chances', as it was referred to by Rev. Thomas Bayes [5] almost 300 years ago, finds its modern counterpart in probability theory. Set down in axiomatic form by Kolmogorov [9], it has subsequently led to the accelerated development of statistics, which is considered indispensable for the analysis of experimental data.

While the axioms of probability (and thus, the theory deduced from them) stand largely undisputed, there has been quite a debate as to its *meaning*. Broadly speaking, there are two positions:

- **Frequentist**. Also known as 'sampling theory', this approach views probability as the limit of relative frequency. Imagine the prototypical example: a coin is being tossed N times, and the number of times n_h it shows 'heads' upon landing is recorded. The relative frequency of 'heads' is then $f_h = \frac{n_h}{N}$. By virtue of the weak law of large numbers [9], f_h converges in probability to P_h, the probability of 'heads', if $N \to \infty$:

$$\lim_{N\to\infty} P(|f_h - P_h| < \epsilon) = 1 \tag{2.12}$$

  where ϵ is some arbitrarily small positive number. Tossing the coin constitutes a 'random experiment', which avails one of a sample drawn from the probability distribution (P_h, P_t) (i.e. the probabilities of observing heads or tails, respectively). Hence the name 'sampling theory'. The outcome of a toss is an instance of a random variable. In the frequentist perspective, probabilities can only be assigned to random variables.

- **Bayesian**. Here, probability is "...regarded as a real-number-valued measure of the plausibility of a proposition when incomplete knowledge does not allow us to establish its truth or falsehood with certainty." [52]. The important keyword is 'proposition'. In the Bayesian view, probabilities are *not* restricted to random variables. They can be assigned to hypotheses as well, such as 'The coin is biased in favor of heads' or 'A neuron in area STSa begins to transmit stimulus-related information 100 ms after the stimulus onset'. This does not imply that the truth or falsehood of the proposition in question is subject to random fluctuations, it only aims at quantifying the (subjective) uncertainty associated with the veracity of a statement due to a lack of information. The rules of probability theory can then be used to compute the plausibilities of logical compositions of such statements, and, most importantly, how the probabilities change when new data become available. This process is called *Bayesian inference.*

2.3.1 Cox's axioms and Jayne's Desiderata

The proof that degrees of belief (or plausibility) can be described by probabilities was given in [14]. If one is willing to accept the Cox axioms (see e.g. [54]), then one is also forced to accept probability theory as the 'grammar' of plausible reasoning. Instead of listing these axioms here, it is perhaps more instructive to consider the basic desiderata of plausible reasoning as presented in [43]:

1. Degrees of plausibility are represented by real numbers.
2. Qualitative correspondence with common sense.
3. (a) If a conclusion can be reasoned out in more than one way, then every possible way must lead to the same result.
   (b) All of the available evidence relevant to a question is taken into account.
   (c) Equivalent states of knowledge are represented by equivalent plausibility assignments.

Desideratum 1 could in principle be replaced by the weaker requirement of a total order, i.e. that the plausibility assignments (=probabilities) to any two propositions can be compared [43], but the resulting theory would then be less manageable in practice. Desiderata 2, 3a and 3c give rise to the sum an product rules for probability. Suppose we had 3 propositions A, B and C. Then

$$P(AB|C) = P(A|BC)P(B|C) = P(B|AC)P(A|C) \qquad (2.13)$$

$$1 = P(A|C) + P(\bar{A}|C) \tag{2.14}$$

$$0 \leq P(A|C) \leq 1 \tag{2.15}$$

Those rules are all that is needed to conduct inference. It is easy to show that $P(A|C) = 1$ if A is true with certainty, and likewise that $P(A|C) = 0$ if A is false with certainty. Furthermore, if A does not depend on B, i.e. $P(A|BC) = P(A|C)$, then $P(AB|C) = P(A|C)P(B|C)$. In this case, A and B are said to be independent. Proposition C stands for the evidence (desideratum 3b) relevant for the determination of $P(A|C)$ etc.. It is important to note that *all* probability assignments are based on such prior information, at least in the Bayesian view. This is the reason why, for the better part of the last century, many mathematicians and scientists have rejected the Bayesian perspective as 'subjective', and, as a remedy, the frequentist interpretation of probability was claimed to be the only valid one, because it was deemed to be more objective. However, as argued in [52], this undertaking has largely failed. What scientists are usually interested in is assessing the probability of various hypotheses, given a set of observed data. And it is precisely this kind of question which the frequentist approach cannot answer, because, by its very definition, probabilities can not be assigned to hypotheses.

In the author's opinion, the basic desiderata are sufficiently compelling to justify the use of probability theory for inference in science. Moreover, since it was proven in [14] that any other set of rules would lead to a violation of the Cox axioms, and thus be in contradiction with these desiderata, it follows that *probability theory is the only suitable tool for conducting inference.*

2.3.2 Bayesian Inference

The original objective of Bayesian Inference, in the words of Rev. T. Bayes [5], is

> "*Given* the number of times ion which an unknown event has happende and failed:
>
> *Required* the chance that the probability of its happening in a single trial lies somewhere between any two degrees of probability that can be named."

In other words, given the observed frequencies, what are the corresponding probabilities? This is an instance of a more general class of problems, namely: given a set of data, how plausible is some hypothesis? Those questions can be answered using probability theory. From the product rule (eqn. 2.13) follows

$$P(A|BC) = \frac{P(B|AC)P(A|C)}{P(B|C)} \tag{2.16}$$

which is know as Bayes' rule or Bayes' theorem. Sometimes it is also referred to as the law of inverse probability, because it states how $P(A|BC)$ is to be converted into $P(B|AC)$. Now suppose that B is a statement about a measurement of some kind, and A is a proposition which has a logical connection to B, e.g. a possible cause of B, or something that frequently coincides with B for reasons unknown. Information of this kind, represented in C, is called *prior information*. Moreover, C must allow for the assessment the probability of A before observing B. Thus, $P(A|C)$ is called the *prior probability* of A, or just the *prior*. $P(B|AC)$ is also needed, since it quantifies how likely B is deemed given A and C, it is referred to as the *likelihood*. $P(B|C)$ could either be a part of the prior information as well, or, more customary, $P(B|\bar{A}C)$ is known. It is then possible to evaluate $P(B|C)$ via eqn. (2.14):

$$\begin{aligned} P(B|C) &= P((A+\bar{A})B|C) = P(AB|C) + P(\bar{A}B|C) \\ &= P(B|AC)P(A|C) + P(B|\bar{A}C)P(\bar{A}|C) \\ &= P(B|AC)P(A|C) + P(B|\bar{A}C)(1 - P(A|C)) \end{aligned} \tag{2.17}$$

$P(B|C)$ is called the *evidence* of the data. The reasons for this name will become clear later. $P(A|BC)$ is the *posterior probability* of A, so named because it tells how probable A becomes once the veracity of B has been established.

Bayesian Inference – for the most part – boils down to nothing more than the inversion of a conditioning via eqn. (2.16).

2.3.3 The limit of certainty: deductive and plausible reasoning

In the limit of certainty, probability theory reduces to deductive reasoning. Consider e.g. the strong syllogism

$$\begin{array}{|c|}\hline \mathrm{A} \Rightarrow \mathrm{B} \\ \mathrm{A{=}true} \\ \hline \mathrm{B{=}true} \\ \hline \end{array}$$

Translated into probabilities, the implication becomes

$$\begin{aligned}
1 &= P(\overline{A\bar{B}}|C) \\
&= 1 - P(A\bar{B}|C) \\
&= 1 - P(\bar{B}|AC)P(A|C) \\
&= 1 - (1 - P(B|AC))P(A|C) \\
&\Rightarrow P(B|AC) = 1
\end{aligned}$$

i.e. B is true with certainty as soon as A is observed to be true. As an example of plausible reasoning, the weak syllogism, which is a mode of reasoning often employed in science for the purpose of hypothesis generation

$$\begin{array}{|c|}\hline \mathrm{A} \Rightarrow \mathrm{B} \\ \mathrm{B{=}true} \\ \hline \text{A becomes more plausible} \\ \hline \end{array}$$

can be expressed as

$$\begin{aligned}
1 &= P(\overline{A\bar{B}}|C) \\
\Rightarrow 0 &= P(A\bar{B}|C) \\
&= P(\bar{B}|AC)P(A|C) \\
&= (1 - P(B|AC))P(A|C) \\
&= \left(1 - \frac{P(BA|C)}{P(A|C)}\right) P(A|C) \\
&= \left(1 - \frac{P(A|BC)P(B|C)}{P(A|C)}\right) P(A|C) \\
&= P(A|C) - P(A|BC)P(B|C) \\
\Rightarrow P(A|BC) &= \frac{P(A|C)}{\underbrace{P(B|C)}_{\leq 1}} \geq P(A|C) \qquad (2.18)
\end{aligned}$$

Thus, as soon as B is observed to be true, A becomes more plausible. The probability of A does not change, however, if C already contained the information that B is true: in that case $P(B|C) = 1$, and nothing is gained by observing B. This process of updating probabilities after observing data is referred to as *Bayesian learning.* Note also that eqn. (2.18) implies that C must be such that $P(A|C) \leq P(B|C)$. In other words, if one believes firmly in the validity of an implication $A \Rightarrow B$, then one must also be sure that A is less likely than, or at most as likely as, B *a priori.* This corresponds to the common-sensical notion that A is one of the reasons which can lead to B, but not the only conceivable one. More on the correspondences between probability theory and classical logic can be found in [43].

2.3.4 Probability distributions and densities

The concept of random variables, so central to the frequentist approach, can be incorporated quite naturally into the Bayesian perspective. For the purposes of this thesis, the added generality of a measure-theroetic approach to probability offers little benefit. Thus, the following definitions will suffice:

- **Discrete random variable**. X is a discrete random variable, if it can take on countably many values x_i. Each x_i can be assumed with probability $P(X = x_i|C)$. $P(X = x_i|C)$ is the probability distribution of X, which has the properties
  - $\forall x_i : P(X = x_i|C) \geq 0$
  - $\sum_{\{x_i\}} P(X = x_i|C) = 1$
- **Continuous random variable**. X is a continuous random variable, if every instance of X $\in I\!\!R$. X is distributed according to $p(x|C)$, the probability density of X, which has the properties
  - $P(X \in [x, x + dx]|C) = p(x|C)dx$
  - $\forall x \in I\!\!R : p(x|C) \geq 0$
  - $\int_{-\infty}^{\infty} p(x|C)dx = 1$

As before, C is the prior information on which the probability assignments are based. $X = x_i$ is just another statement, namely 'X takes on the value x_i'. Therefore, random variables can be treated within the Bayesian framework. The normalization condition $\sum_{\{x_i\}} P(X = x_i|C) = 1$ can be derived from the sum and product rules on the condition that the statements are mutually exclusive, which is obviously a sensible requirement – X cannot take on more than one value at once. The usual shorthand for $P(X = x_i|C)$ is $P(x_i)$, where prior information C on which the probability assessments are based is implicitly assumed. It will be used whenever unequivocally possible.

The generalization to a real-valued random variable is also straightforward and requires little comment. The density must be integrable (let's say in the Riemannian sense [9], even though that is not the most general possible condition). What should perhaps be noted is that a further generalization to random variables which are vectors of real numbers is easily accomplished by replacing every occurrence of x with $\vec{x}$. Consequently, the normalization integral would then have to extend over $\mathbb{R}^m$, where m is the dimensionality of $\vec{x}$.

2.4 Information theory

Information theory is concerned with the theoretical aspects of information transmission, storage and, to some (small) extent, processing. This discipline was founded by C. Shannon [83], who also addressed and solved its most fundamental problems. Since the brain is an information processing system, it is natural to try to apply information theoretic principles and results in order to understand its functions. The following short overview will be phrased in terms of random variables, but the reader should keep in mind that each of those could be replaced by a set of mutually exclusive propositions.

The most central information theoretic quantity is *entropy*. In [82], it is argued that a general measure H of uncertainty of a random variable, whose probability distribution $P(x_i) = P_i$, $i = 1, \ldots, N$ is assumed to be known , should fulfill the following conditions:

1. H should be a continuous function of the P_i, i.e. if the probabilities change only by a small amount, then the uncertainty measure should qualitatively reflect the magnitude of this change.

2. If all the P_i are equal, then H should be a monotonically increasing function

of N. When uncertainty is understood as the number of questions one has to ask about X before its exact value is determined, then this number should grow with the number of possibilities.

3. H should not depend on the specific set of questions asked. Imagine sequences of questions were constructed that successively narrowed down the possible values of a given instance of X. The final determination of X must be the same for all such sequences. As an example, suppose $N = 3$. Then a possible first question would be 'is $X = x_1$?'. Having received the answer 'yes' (which would happen with probability P_1), no more information would be needed. If, however, the answer was 'no' (with probability $1 - P_1$), then a second question would be in order, e.g. 'is $X = x_2$?'. Thus, it is required of H that

$$H(P_1, P_2, P_3) = \underbrace{H(P_1, 1 - P_1)}_{\text{1st question}} + (1 - P_1) \underbrace{H(\frac{P_2}{1 - P_1}, \frac{P_3}{1 - P_2})}_{\text{2nd question}} \tag{2.19}$$

   This property is also called 'grouping' [13].

It can be proven [82] that the only form of H compatible with these conditions is

$$H(\{P_i\}) = -K \sum_{i=1}^{N} P_i \log(P_i) \tag{2.20}$$

where K is some positive constant that determines the units of H. Alternatively, K can be included in the logarithm via a change of basis

$$K \log(x) = \log_{\exp(\frac{1}{K})}(x) \tag{2.21}$$

Common choices for the basis are:

- 2, in which case H is measured in *bit.* A bit is one 'yes/no' information.
- e, the basis of the natural logarithm. The unit is then *nats.* 1 bit ≈ 0.6931 nats.

Since $0 \leq P_i \leq 1$, and $\lim_{x->0^+} x \log(x) = 0$, it is apparent that $H \geq 0$ with equality if and only if one $P_i = 1$ and all the others are 0. Thus, the entropy is zero only if there is no uncertainty about X. It is customary to denote the entropy by the random variable, not the probability distribution, i.e.

$$H(\{P_i\}) = H(X) \tag{2.22}$$

An important property of H is its relationship with codelength. It can be shown [13] that the minimum expected number of 'yes/no' (i.e. binary) questions which need to be answered to determine X is between $H(X)$ and $H(X)+1$, when the latter is measured in bit. Since entropy can be written as an expectation as well

$$H(X) = -E\left[\log(P(x))\right] \tag{2.23}$$

it follows that,

$$-\log_2(P(x_i)) \tag{2.24}$$

is the number of bits needed to encode the message $X = x_i$. In the terminology of information theory, each x_i is a symbol or code element. The set of all symbols comprises the alphabet used for message encoding.

2.4.1 Joint, conditional and relative entropy

If a probability distribution is defined for two random variables X and Y, the *joint entropy* is given by

$$H(X,Y) = -\sum_{x_i}\sum_{y_j} P(x_i,y_j)\log(P(x_i,y_j)) \tag{2.25}$$

If X and Y are independent, then

$$\begin{aligned} H(X,Y) &= -\sum_{x_i}\sum_{y_j} P(x_i)P(y_j)\left(\log(P(x_i)) + \log(P(y_j))\right) \\ &= H(X) + H(Y) \end{aligned} \tag{2.26}$$

Likewise, the *conditional entropy* is

$$H(X|Y) = -\sum_{x_i}\sum_{y_j} P(x_i,y_j)\log(P(x_i|y_j)) \tag{2.27}$$

Another important quantity is *relative entropy* or Kullback-Leibler divergence. Suppose we had two probability distributions $P(x)$ and $Q(x)$ for the same random variable, then

$$D(P(x)||Q(x)) = \sum_{x_i} P(x_i)\log\left(\frac{P(x_i)}{Q(x_i)}\right) \tag{2.28}$$

It can be shown that (e.g. via Jensen's inequality [13])

$$D(P(x)||Q(x)) \geq 0 \tag{2.29}$$

with equality if and only if $P(x) = Q(x)$ for all possible x. This is known as Gibbs' inequality [54], and it is one of the most central inequalities in information theory. $D(P(x)||Q(x))$ can thus be seen as measure for how much $P(x)$ deviates from $Q(x)$. Kullback-Leibler divergence has a number of interpretations, perhaps the most common one being that of coding inefficiency: given a random variable is distributed according to $P(x)$, it is possible to construct a code with average codelength $H(P(x))$. If instead a code optimized for $Q(x)$ was used, then the expected codelength would be $-\sum_{x_i} P_i \log(Q_i)$. Thus, the inefficiency – the superfluous codelength due to using a suboptimal code – is

$$-\sum_{x_i} P_i \log(Q_i) - \left(-\sum_{x_i} P_i \log(P_i)\right) = D(P(x)||Q(x)) \tag{2.30}$$

Note that $D(P(x)||Q(x))$ will diverge whenever $Q_i = 0$ and $P_i \neq 0$. This indicates that the optimal code for X under $Q(x)$ is not only suboptimal, but actually unsuitable for representing X under $P(x)$: impossible values of X under $Q(x)$ ($Q_i = 0$) would not need to be coded. If those values are possible under $P(x)$, then a new coding scheme is definitely necessary.

2.4.2 Mutual information

Mutual information is defined as

$$I(X;Y) = \sum_{x_i} \sum_{y_j} P(x_i, y_j) \log \left(\frac{P(x_i, y_j)}{P(x_i)P(y_i)} \right) \tag{2.31}$$

where $P(x_i) = \sum_{y_j} P(x_i, y_i)$ and likewise for $P(y_j)$. Since it can be written as $I(X;Y) = D(P(x_i, y_j)||P(x_i)P(y_i))$, it follows that

$$I(X;Y) \geq 0 \tag{2.32}$$

with equality if and only if $P(x_i, y_j) = P(x_i)P(y_i)$ for all x_i and y_i [13]. Another way of expressing it is

$$I(X;Y) \;=\; \sum_{x_i} \sum_{y_i} P(x_i, y_i) \log(\frac{1}{P(x_i)}) + \sum_{x_i} \sum_{y_j} P(x_i, y_j) \log \left(\frac{P(x_i, y_j)}{P(y_i)} \right)$$

$$
\begin{aligned}
&= -\sum_{x_i} P(x_i)\log(P(x_i)) + \sum_{x_i}\sum_{y_j} P(x_i,y_j)\log(P(x_i|y_i)) \\
&= H(X) - H(X|Y) \qquad (2.33)
\end{aligned}
$$

i.e. the difference between the uncertainties about X before and after observing Y. Thus, it quantifies the reduction in uncertainty due to the knowledge of Y, or, in other words, it measures how much can be learnt about one random variable given another. For symmetry reasons, eqn. (2.33) could also have been written as $I(X;Y) = H(Y) - H(Y|X)$, whence the adjective 'mutual'.

For given $H(X)$, $I(X;Y)$ assumes a maximum when $H(X|Y) = 0$, i.e. when X is a function of Y. In this case (which also includes the identity function $X = Y$), $I(X;Y) = H(X)$. This opens up another perspective on entropy: instead of interpreting it as uncertainty, it can also be seen as the maximum amount of information that can be gained about a random variable (i.e. everything that is not already encoded in its probability distribution). Thus, entropy is sometimes called *self-information* of X.

Mutual information is an important quantity in neuroscience. In this thesis, it will be employed to assess the information which a neural response carries about a visually presented stimulus.

2.4.3 Differential entropy

The concept of entropy can be generalized to continuous random variables, if some care is taken [13]. Since entropy measures the average amount of information required to specify an instance of a random variable exactly, it will in general be ∞ if the random variable can take on infinitely many values. To see how this problem arises, assume a continuous random variable X was distributed according to the density $p(x)$. Also, suppose that one could measure this variable with an accuracy Δx across the whole (finite) domain of X. The probability P_i of observing X in an interval of width Δx centered around x_i is then $P_i = \tilde{p}(x_i)\Delta x$, where $\tilde{p}(x_i)$ is the mean density in this interval. The entropy of the observations x_i thus obtainable would then be given by (note that $\sum_i P_i = 1$, i.e. the intervals are assumed to be non-overlapping):

$$H(X) = -\sum_i P_i \log(P_i) \qquad (2.34)$$

$$= -\sum_i \tilde{p}(x_i)\Delta x \log(\tilde{p}(x_i)\Delta x) \quad (2.35)$$

$$= -\sum_i \tilde{p}(x_i)\Delta x \log(\tilde{p}(x_i)) - \log(\Delta x) \quad (2.36)$$

Let us now consider the limiting case of exact observations, i.e. the limit $\Delta x \to 0$:

$$\lim_{\Delta x \to 0} H(X) = -\int dx\, p(x)\log(p(x)) - \underbrace{\log(\Delta x)}_{-\infty} = \infty \quad (2.37)$$

where the integration extends over the domain of X. Thus, the entropy will diverge, which indicates that knowing a continuous quantity *exactly* requires generally an infinite amount of information. This is highlighted by the fact that the diverging term describes the measurement accuracy Δx. Therefore, it is not possible to define the entropy of a continuous random variable. However, it is possible to measure the difference between entropies of continuous variables under certain conditions: assume a second random variable Y was distributed according to $p(y)$, and measurable with the same accuracy as X, Δx. Then

$$\begin{aligned} H(X) - H(Y) &= -\sum_i \tilde{p}(x_i)\Delta x \log(\tilde{p}(x_i)) - \log(\Delta x) \\ &\quad + \sum_i \tilde{p}(y_i)\Delta x \log(\tilde{p}(y_i)) + \log(\Delta x) \\ &= \sum_i \tilde{p}(x_i)\Delta x \log(\tilde{p}(x_i)) + \sum_i \tilde{p}(y_i)\Delta x \log(\tilde{p}(y_i)) \quad (2.38) \end{aligned}$$

Taking the limit $\Delta x \to 0$ might now very well be possible, because the diverging term in eqn. (2.37) has cancelled out. Thus, the *differential entropy* of $X \in I\!R$ with density $p(x)$ is defined as

$$h(X) = -\int_{-\infty}^{\infty} p(x)\log(p(x)). \quad (2.39)$$

This expression cannot be interpreted anymore as an information content or an uncertainty measure. Unlike entropy, which is a quantity on a proportional scale (i.e. statements like 'under $P(X)$, X's information content is twice as high as under $Q(X)$' can be meaningfully made), it lives on an interval scale. That implies that only the difference between two values has meaning, hence the name 'differential entropy'. For a mathematically more precise treatment of the limiting process, see [13].

Mutual information, being a difference between differential entropies in the continuous case, can thus be interpreted as before. The same is true for relative entropy.

2.5 Bayesian methods

In the following section, a brief outline of the most common Bayesian methods for data modeling will be given. A popular way of understanding those methods is to imagine the models as generators of the data. This generation process is governed by a set of parameters (in the case of parametric models). Bayesian inference is then applied to infer those parameters from a set of observed data.

2.5.1 Inferring model parameters

Assume we had reason to believe that an observable X was distributed according to $p(x|\vec{\theta}, M)$. $\vec{\theta}$ is a vector of parameters, whose values determine the exact shape of the density function. In the case of the well-known Gaussian density, $\vec{\theta}$ would have two components, one being the mean and the other one the variance. M denotes the part of the prior information which is not encoded in $\vec{\theta}$, e.g. how the components of $\vec{\theta}$ are to be interpreted. One might find it unnecessary to explicitly state this dependency on M, and indeed, it is customary in the literature not to do so. However, as we will shortly see, this can mislead one into believing that Bayesian Inference was able to provide unconditional answers, which it cannot. $p(x|\vec{\theta}, M)$ is called the *likelihood function* or the *noise model* [7].

Furthermore, assume that we also knew how to choose the prior $p(\vec{\theta}|M)$. How this is accomplished is the subject of the next subsection. If we now observe a (multi)set D of instances of X, $D = \{x_0, \ldots, x_N\}$, then we can, if the x_i are independent, write the density of D as

$$p(D|\vec{\theta}, M) = \prod_i p(x_i|\vec{\theta}, M) \tag{2.40}$$

Using Bayes' rule, the posterior density of $\vec{\theta}$ then is

$$p(\vec{\theta}|D, M) = \frac{p(D|\vec{\theta}, M)p(\vec{\theta}|M)}{p(D|M)} \tag{2.41}$$

where

$$p(D|M) = \int d\vec{\theta} p(D|\vec{\theta}, M)p(\vec{\theta}|M) \tag{2.42}$$

is called the *evidence* [54] for the model class specified by M. Had M been omitted, this would appear to be the probability (density) of the data, independent of any model class assumption. This 'absolute' probability would be a nonsensical quantity in the context of Bayesian inference.

The process of integrating out unwanted variables ($\vec{\theta}$ in eqn. (2.42)) is called *marginalization*, the name 'evidence' is motivated by the following observation: suppose we also considered another model class, denoted by Q. In order to decide which model class offers the better explanation for the data, we would have to evaluate the posterior probabilities

$$\begin{aligned} P(M|D,R) &= \frac{p(D|M)P(M|R)}{p(D|M)P(M|R)+p(D|Q)P(Q|R)} && (2.43) \\ P(Q|D,R) &= \frac{p(D|Q)P(Q|R)}{p(D|M)P(M|R)+p(D|Q)P(Q|R)} && (2.44) \end{aligned}$$

where R stands for the information that only the (mutually exclusive) model classes M and Q are take into account, and $p(D|M,R) = p(D|M)$ was assumed, i.e. the probability density of the data is fully specified once the model class is known. In the absence of further information, the priors would be chosen as

$$P(M|R) = P(Q|R) = \frac{1}{2} \tag{2.45}$$

which is an example of a non-informative prior. If $P(M|D,R)$ is greater (smaller) than $P(Q|D,R)$, M (Q) would be the model class of choice. With the non-informative prior, this is equivalent to comparing $P(D|M)$ and $P(Q|M)$. Hence the name 'evidence'.

Once the posterior (eqn. (2.41)) is known, several other quantities of interest can be computed, e.g. the expectation of $\vec{\theta}$:

$$E\left[\vec{\theta}\right] = \int d\vec{\theta}\,\vec{\theta}p(\vec{\theta}|D,M) \tag{2.46}$$

or the predictive density

$$p(x|D,M) = E\left[p(x|\vec{\theta},M)\right] = \int d\vec{\theta}p(x|\vec{\theta},M)p(\vec{\theta}|D,M) \tag{2.47}$$

where, in accordance with the model assumption, the equality $p(x|\vec{\theta},M) = p(x|\vec{\theta},M,D)$ was used.

2.5.2 Choosing the prior

The uniform model prior eqn. (2.45) was chosen according to the principle of indifference, which dates back to Bernoulli. According to this principle, when the only information available about a set of alternatives is that one of them must be true, then each of them should be assigned the same probability. Those are then called non-informative probabilities or least informative probabilities [52]. The principle of indifference is a special case of the principle of maximum entropy, which states that, given information C about the distribution of X, the latter should always be chosen such that the entropy

$$H = -\sum_i P(X = x_i|C) \log(P(X = x_i|C)) \tag{2.48}$$

assumes a maximum. It is not hard to see that, if C only specifies the number of possible values X can take on, maximum entropy reduces to the principle of indifference. There are several ways to justify the maximization of H, ranging from Shannon's original argument [82] (entropy is the most general measure of uncertainty) to demonstrating that this criterion is obtained by requiring the content of C be invariant under very general transformations [84]. Those and more have been reviewed in [43]. It should be noted, however, that while the arguments for a maximum entropy approach are compelling if X is a discrete random variable, the situation is less clear for continuous X. Here, it is usually not a priori clear which scale to choose, and thus other methods of determining the prior are often used [6].

In practice, another approach is often used when choosing priors, namely that of conjugacy. A prior is said to be conjugate to the posterior if both have the same functional form [7]. This is especially useful when this form allows for the evaluation of interesting integrals (such as eqn. (2.42)) in a closed form. It is often possible to make this choice so that the prior fulfills the maximum entropy condition when its parameters approach some limit [54]. Nevertheless, conjugacy is mostly employed for mathematical convenience.

2.5.3 Kullback-Leibler divergence in Bayesian inference

As explained above, $D(P(x)||Q(x))$ measures the inefficiency of encoding a random variable distributed according to $P(x)$ with a code optimized for $Q(x)$. Assume $Q(x)$ was the prior of X, and $P(x)$ its posterior. Then, $D(P(x)||Q(x))$ would tell us how much more efficient we can represent X given the information acquired by Bayesian inference. Conversely, it measures the additional information needed to represent the posterior knowledge under the prior, i.e. the information gain. Therefore, Kullback-Leibler divergence is frequently used to quantify 'learning'. As an example, consider the weak syllogism from section 2.3.3: The prior was $P(A|C)$, the posterior $P(A|BC) = \frac{P(A|C)}{P(B|C)}$ (i.e. the probabilities after the message 'B is true' has arrived). Thus,

$$\begin{aligned} D_A &= P(A|BC)\log\left(\frac{P(A|BC)}{P(A|C)}\right) + P(\bar{A}|BC)\log\left(\frac{P(\bar{A}|BC)}{P(\bar{A}|C)}\right) \\ &= -\frac{P(A|C)}{P(B|C)}\log(P(B|C)) + \left(1 - \frac{P(A|C)}{P(B|C)}\right)\log\left(\frac{1-\frac{P(A|C)}{P(B|C)}}{1-P(A|C)}\right) \end{aligned} \tag{2.49}$$

Compare this expression with the Kullback-Leibler divergence between the prior and posterior of B, $P(B|C)$ and $P(B|BC)$:

$$\begin{aligned} D_B &= P(B|BC)\log\left(\frac{P(B|BC)}{P(B|C)}\right) + P(\bar{B}|BC)\log\left(\frac{P(\bar{B}|BC)}{P(\bar{B}|C)}\right) \\ &= -\log(P(B|C)) \end{aligned} \tag{2.50}$$

This is just the length of the message 'B is true' coded under the prior, which is exactly the information that gave rise to the posterior. Note that

$$\begin{aligned} 1 &\geq \frac{P(A|C)}{P(B|C)} \geq P(A|C) \\ &\Rightarrow 0 \leq 1 - \frac{P(A|C)}{P(B|C)} \leq 1 - P(A|C) \\ &\Rightarrow 0 \leq \frac{1-\frac{P(A|C)}{P(B|C)}}{1-P(A|C)} \leq 1 \\ &\Rightarrow \log\left(\frac{1-\frac{P(A|C)}{P(B|C)}}{1-P(A|C)}\right) \leq 0 \end{aligned}$$

Whence

$$\begin{aligned} D_A &\leq -\frac{P(A|C)}{P(B|C)}\log(P(B|C)) \\ &\leq -\log(P(B|C)) = D_B \end{aligned} \tag{2.51}$$

Thus, the infomation gain w.r.t. A is always smaller than, or at most as large as, that w.r.t. B, which is the information-theoretic expression of the intuitive notion that this syllogism is 'weak'. The maximum gain is $D_A = -\log(P(B|C))$ if $P(A|C) = P(B|C)$, in this case $P(A|BC) = 1$ (see eqn. (2.18)) and the syllogism becomes a strong one. The gain approaches a minimum as $P(A|C) \to 0$, $P(B|C) \neq 0$, here, nothing is learned by observing B.

As noted above, $D(P(x)||Q(x))$ can diverge if $P(x) > 0$ and $Q(x) = 0$ for some x. In the context of Bayesian inference, that means that under the posterior, $X = x$ is a possible value, whereas it was deemed impossible under the prior. Divergence of $D(P(x)||Q(x))$ thus indicates an inconsistency between the prior and the posterior. If inference is conducted properly, this will of course not happen. It is, however, possible for the Kullback-Leibler divergence to become very large, if a highly informative dataset is available.

Noisy information transmission as a classification problem

Assume a discrete alphabet comprised of the x_i was used to transmit messages of the form '$X = x_i$', and that the (prior) probability distribution $P(X = x_i) = P(x_i)$ of a typical message was known to the receiver. Due to noise in the transmission process, the receiver cannot fully trust his observations, rather, he knows how likely it is that a given symbol is replaced by another one in transport, quantified by $P(Y = x_j|X = x_i) = P(x_j|x_i)$, where Y is the symbol he receives. He then is faced with a classification task, namely that of deciding which message was really sent. To that end, he would employ Bayes' theorem to compute

$$P(x_i|x_j) = \frac{P(x_j|x_i)P(x_i)}{\sum_{x_i'} P(x_j|x_i')P(x_i')} \tag{2.52}$$

and pick the x_i which maximizes this probability.

The information gained on receiving the message $Y = x_j$ is given by the Kullback divergence

$$D(P(x_i|x_j)||P(x_i)) \tag{2.53}$$

because eqn. (2.52) and $P(X = x_i)$ represent the receiver's state of knowledge after and before the observation, respectively. The average information gain can be computed by averaging (2.53) over all possible observations, weighted by their probabilities $P(x_j) = \sum_{x_i} P(x_j|x_i)P(x_i)$:

$$\begin{aligned}
& E\left[D(P(x_i|x_j)||P(x_i))\right] \\
= & \sum_{x_j} P(x_j)D(P(x_i|x_j)||P(x_i)) \\
= & \sum_{x_j} P(x_j)\sum_{x_i} P(x_i|x_j)\left(\log\left(P(x_i|x_j)\right) - \log\left(P(x_i)\right)\right)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{x_j}\sum_{x_i} P(x_i, x_j)\log\left(P(x_i|x_j)\right) - \sum_{x_j}\sum_{x_i} P(x_i, x_j)\log\left(P(x_i)\right) \\
&= -H(X|Y) + H(X) \\
&= I(X;Y) && (2.54)
\end{aligned}$$

In other words, the average amount of information transmitted through a noisy channel is given by the mutual information between input and output. If the prior distribution is chosen such that $I(X;Y)$ is maximized for a given noise, then this mutual information is called the *channel capacity* [82]. Moreover, it follows that mutual information is also a measure for expected classification performance.

2.6 Approximation techniques

Thus far, Bayesian inference is a straightforward process: choose model(s), apply sum & product rules, (eqn. (2.14) and eqn. (2.13)), get result(s). But the devil is in the detail: evaluating integrals like (2.42), (2.47) or (2.46) is often very difficult, or even infeasible (e.g. they involve numerical integrations over several hundred variables). This gave rise to the development of various approximation techniques, the most important of which will now briefly be reviewed. For notational ease, the examples will consider only densities of one-dimensional variables θ, but generalizations to $\vec{\theta} \in I\!R^m$ are possible.

2.6.1 Maximum likelihood (ML) and maximum a-posteriori (MAP)

ML and MAP are the simplest approximations to Bayesian learning. In integrals like (2.46), one might hope that the largest contributions to the expectation would come from regions where the posterior density is high. Therefore, the location of its maximum is used as an estimate of $E[\theta]$. Since this location does not change when the posterior is subjected to a monotonic transformation, it is customary to look for the minimum of its negative log instead:

$$\bar{\theta} = \min_\theta \; -\log\left(p(\theta|D,M)\right) \tag{2.55}$$

This is known as MAP, or maximum a-posteriori. Taking the logarithm has the added avantage of increased numerical stability, because probability densities can become very small, or be very sharply peaked. This may cause potential problems with optimization techniques having a finite step size and accuracy, such as computer implementations of gradient-based algorithms.

If the prior in eqn. (2.41) is uniform, finding the posterior maximum becomes equivalent to finding the maximum of the likelihood $p(D|\theta, M)$, in which case the technique is called ML (maximum likelihood). Both approaches suffer from several deficiencies, the worst one probably being that they simply might not be good approximations, if the posterior is not sharply peaked. Another drawback is their noninvariance under transformations of variables [54]. Nevertheless, they are still being used with some success in practical inference tasks, such as neural network training.

2.6.2 Laplace approximation

Assume the posterior density (2.41) had only one maximum. Laplace approximation [31] replaces the true density with a second-order logarithmic approximation centered at this maximum (located at θ_m):

$$\log\left(\tilde{p}(\theta|D, M)\right) \approx \log\left(p(\theta_m|D, M)\right) + \frac{1}{2} \left.\frac{\partial^2 \log\left(p(\theta|D, M)\right)}{\partial\theta^2}\right|_{\theta=\theta_m} (\theta - \theta_m)^2 \quad (2.56)$$

The first-order term of the expansion is zero, because θ_m is the location of an extremum. Since the approximative density $\tilde{p}(\theta|D, M)$ has to be normalized, the zeroth-order term doesn't matter either. Thus, setting $-\left.\frac{\partial^2 \log(p(\theta|D,M))}{\partial\theta^2}\right|_{\theta=\theta_m} = \frac{1}{\sigma^2}$ and inverting the logarithm on both sides, one finds that

$$\tilde{p}(\theta|D, M) = \frac{1}{N(\sigma)} \exp\left(-\frac{(\theta - \theta_m)^2}{2\sigma^2}\right) \quad (2.57)$$

where $N(\sigma)$ is the normalization constant. This is the well-known Gaussian distribution [9] with $N(\sigma) = \sqrt{2\pi\sigma^2}$.

Laplace approximation works well if the density in question is unimodal (i.e. has only one maximum), reasonably symmetric, and fulfills the differentiability requirements necessary for the second-order expansion. Note that it is sufficient to know a quantity proportional to the true distribution, because a factor only changes the zeroth-order term. Thus, this method can still be applied when the normalization constant is unavailable.

2.6.3 Monte-Carlo methods

Monte-Carlo (MC) methods are basically ways of inspired sampling. For example, if the expectation of θ under the posterior is the quantity of interest, then one might try to devise a procedure for drawing a (probably large) number N of samples θ_i, and then compute

$$\bar{\theta} = \frac{1}{N}\sum_{i=1}^{N}\theta_i \tag{2.58}$$

as an approximation to the true expectation $E[\theta]$. The variance of $\bar{\theta}$ then depends only on the variance of θ and N, and *not* on the dimensionality of the problem. This is a property which has contributed somewhat to the appeal of MC methods. The main problem, however, is the sampling. In the one-dimensional case, there are various techniques: transformation method, rejection sampling [76] and importance sampling [54] to name a few. What they all have in common is that they won't work well in higher-dimensional problems, either because they cannot be generalized, or due to exponential scaling properties, i.e. the N (and thus, the computational time) required until a representative sample of the density has been collected grows exponentially with the dimensionality. This can be remedied by the Metropolis algorithms [55] and refinements thereof, such as Gibbs sampling [57] and slice sampling [59]. These approaches construct successive sample points via Markov chains, hence they are often referred to as MCMC methods.

The current popularity of MCMC is probably due to their ability to generate representative samples from virtually any density, which, in concert with the speedily increasing performance of modern computers, avails one of the possibility to evaluate very complex models, even if only approximately. In this thesis, however, they will not be used (except for comparisons) and thus not be discussed in great detail. For an excellent review, see [57].

2.6.4 Variational methods

The idea underlying variational methods is simple: if the true density $p(\theta|D, M)$ is too complicated to handle, choose a more manageable density $\tilde{p}(\theta|\vec{\gamma})$ and determine $\vec{\gamma}$ so that some suitable defined distance measure between the two is minimized [45]. A common choice is the Kullback divergence $D(\tilde{p}(\theta|\vec{\gamma})||p(\theta|D, M))$. Since

$p(\theta|D, M)$ is often so complex that it cannot even be normalized, it is replaced by $p(D|\theta, M)p(\theta|M)$, which differs from the posterior only by a factor $p(D|M)$. The objective function for the minimization therefore is

$$\begin{aligned} F &= \int d\theta \tilde{p}(\theta|\vec{\gamma}) \log\left(\frac{\tilde{p}(\theta|\vec{\gamma})}{p(D|\theta, M)p(\theta|M)}\right) \\ &= D(\tilde{p}(\theta|\vec{\gamma})||p(\theta|D, M)) - \log\left(p(D|M)\right) \end{aligned} \tag{2.59}$$

which is also known as the variational free energy, due to a formal correspondence with statistical thermodynamics, where similar techniques had been developed for the description of complex systems [54]. Since the Kullback divergence is always positive, F is an upper bound on $-\log\left(p(D|M)\right)$. This bound will be reached when $\tilde{p}(\theta|\vec{\gamma})$ is equal to the true posterior. Therefore, $\exp(-F)$ is an approximation to the evidence for M.

If eqn. (2.59) is rewritten in a slightly different form

$$F = D(\tilde{p}(\theta|\vec{\gamma})||p(\theta|M)) - \int d\theta \tilde{p}(\theta|\vec{\gamma}) \log\left(p(D|\theta, M)\right) \tag{2.60}$$

then an interpretation of Bayesian inference as a (meaningful) optimization problem becomes apparent: the first term on the r.h.s. quantifies how much must be learned to advance from the prior to the prospective posterior $\tilde{p}(\theta|\vec{\gamma})$. The second term is the expectation of the log likelihood under the prospective posterior. Thus, F will be small if:

1. the posterior deviates little from the prior, i.e. most prior believes are maintained and

2. the log likelihood is large, i.e. the data are explained well.

Therefore we might say that *Bayesian inference is the best compromise between preserving what we know already and explaining new data as well as possible.*

The main difficulty in applying a variational method to a problem lies in finding a good form for $\tilde{p}(\theta|\vec{\gamma})$. Once this has been accomplished, however, solutions can usually be computed faster than via MCMC methods, even though the latter can in principle give better results if one is prepared to be very patient.

It should be noted that there is another approach to variational inference [42] which aims at computing upper and lower bounds on probabilities, which are then minimized and maximized to find close approximations of the true distribution.

Chapter 3

Occam's razor for factorial codes

3.1 Introduction

What is sparse coding and why is it interesting? As yet, there seems to be no unique definition of sparseness, but a code is usually called sparse if most of its symbols are unused in the encoding of a single given input. Translated into the terminology of neural networks applied to the coding of images (or any other type of input), that means that in the firing pattern which represents one image, only a few out of a possibly large number of units are active.

The second requirement is that the code be evenly distributed, i.e. on average every unit gets activated approximately equally often, such that each unit is inactive for most inputs. For continuous-valued units, that means that the probability density function which describes the distribution of each unit's output value is unimodal with a sharp peak at zero. If this was not a necessary condition, then any code could be 'sparsified' by adding plenty of symbols (or units) to it which are never used.

These two requirements are also referred to [65] as *population sparseness* and *single-unit sparseness* (or lifetime sparseness). It is important to note that they do not necessarily imply each other: given that a code is sparse on a single-unit level, it need not be sparse on a population level. Consider e.g. the extreme case where all units behave like exact copies of one single unit: this code might exhibit single-unit sparseness (even though its usefulness is questionable), but it would not fulfill the requirement for population sparseness. Conversely, as stated above, adding plenty of unused units to any neural code will induce population sparseness, but it would fail to bring about single-unit sparseness. When examining a given neural code, it is therefore important to test for the two types of sparseness separately.

Sparse codes are interesting for a number of reasons: Firstly, when data are coded sparsely, subsequent classification tasks can be greatly simplified [11]. Since each code symbol corresponds to some feature of the data and only a few of these features are present in a given input [28], classification based on the presence or absence of these features is likely to be easier than classification based on the original data. For example, if one wants to determine whether a given picture shows a face or not, the absence or presence of a nose will in most cases be sufficient information. Secondly, when considering biological neural networks, the idea of 'load distribution' across the available hardware is quite an appealing one [34]. This load distribution could also be achieved by a dense code (i.e. one where most units are participating in the coding of an input), which would have the advantage of utilizing the full storage and transmission capacity of a neural network. However, as argued in [30, 29], learning a dense code is time-consuming and pattern recognition is slow as well. Furthermore, if multiple inputs are presented simultaneously, then there is a good chance that a sparse representation of these inputs will not overlap, and thus the inputs will still be distinguishable. This would be much harder to achieve with a dense representation. Thirdly, if the code is sufficiently sparse, it could also be used for data compression.

Building on a framework by Daugmann [16] and Pece [75], Harpur and Prager [33, 34, 35] constructed a neural network, which they termed the REC model (see fig. 3.1), capable of discovering sparsely distributed representations by using the principle of redundancy reduction. This principle states that the mutual information between code symbols should be as small as possible. Given a deterministic encoder (i.e. no noise is produced in the coding process), the entropy of the data before and after coding has to be equal, if no information is to be lost. If there is no mutual information between code symbols, then the sum of their individual entropies will be equal to the total entropy of the encoded data, otherwise their sum will be larger. By applying downward pressure on the entropy of the units of the REC model while requiring a faithful representation of the data at the same time, mutual information between units will be 'squeezed out' and the resulting code will hence contain little or no redundancy.

One way of applying this downward pressure is to penalize high absolute values of the units' outputs, which will give rise to an output probability density that peaks at zero. Hence the resulting code fulfills the first condition for sparseness. Whether the second one is met as well depends on the structure of the data, but for natural images this seems to be the case.

Olshausen and Field [63, 64] employed a similar technique for efficient coding of natural images and showed how the resulting response functions of the units relate to the properties of simple cells in the mammalian primary visual cortex.

In order to model the function of later stages of visual processing in mammals, the activation patterns of this network could be used as an input to another one of similar architecture. It would therefore be advantageous if these patterns could be calculated fast. Moreover, not only speed but also accuracy would be an important issue if the network was to be used in practical applications, such as image compression. In the following, the derivation of an algorithm will be presented which achieves both goals with less computational effort than gradient descent based minimizers.

In addition, an extension to the REC model will be discussed, specifically addressing the problem of how to find the correct number of independent causes of the data (here, 'causes' stands for code symbols necessary to represent the data). The resulting learning algorithm prunes all units which are not necessary for the coding, i.e. which do not contribute significantly to the quality of the code. Sparseness is promoted but not enforced, i.e. the code will only be sparse if the data allow for it. An extension of this scheme so as to grow units, should the coding not be satisfactory, is possible.

3.2 The network

The network (fig. 3.1) which was used here has a similar architecture to the REC model proposed by Harpur and Prager [35]. It consists of M units, each of which has N inputs, the receptive fields (RFs) for different units are fully overlapping. The weight vector (of dimensionality N) for unit i is denoted by $\vec{W}_i, |\vec{W}_i| = 1$. Unlike the original network interpretation of the REC model, this network has lateral connections $L_{ij} = -\vec{W}_i\vec{W}_j$, which are a consequence of the error function that is being minimized when the network is activated (see (3.5) and (3.6)). Thus, when the $\vec{W}_i$ change, the L_{ij} do so accordingly. For a given input vector $\vec{X}$, the output of

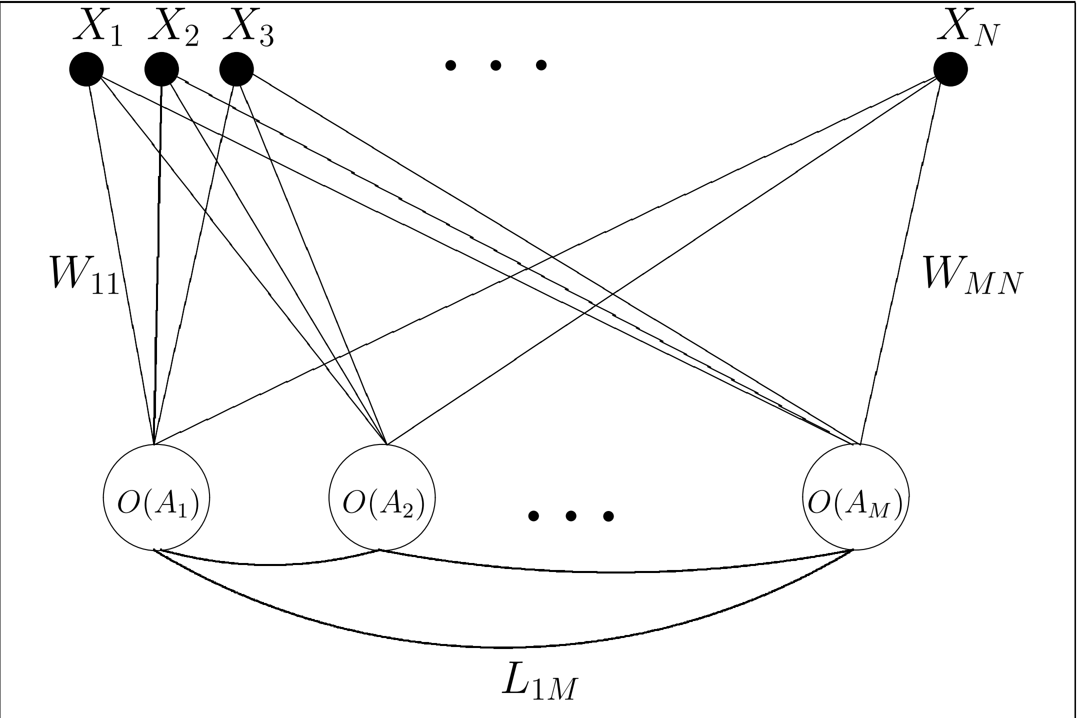


Figure 3.1: A network interpretation of the REC model. $X_1, \ldots, X_N$ are the input terminals. Each unit computes the output from its activation A_i via the transfer function $O(A)$. The symmetric and bidirectional lateral connections are given by $L_{ij} = -\vec{W}_i \vec{W}_j$.

a unit is given by

$$O_i = O(A_i) \tag{3.1}$$

$$A_i = \vec{X}\vec{W}_i + \sum_{j=1, j\neq i}^{M} L_{ij} O_j \tag{3.2}$$

where A_i is the activation of unit i and $O(A)$ (see fig. 3.2) is a piecewise linear activation function with a dead-zone of width λ_i around 0, i.e.

$$\begin{aligned} -\lambda_i \leq A_i \leq \lambda_i &: \ O_i = 0 \\ |A_i| > \lambda_i &: \ O_i = A_i - \lambda_i \mathrm{sign}\,(A_i) \end{aligned} \tag{3.3}$$

Here and in the following the sign () function is defined as

$$\begin{aligned} x > 0 &: \ \mathrm{sign}\,(x) = 1 \\ x = 0 &: \ \mathrm{sign}\,(x) = 0 \\ x < 0 &: \ \mathrm{sign}\,(x) = -1 \end{aligned} \tag{3.4}$$

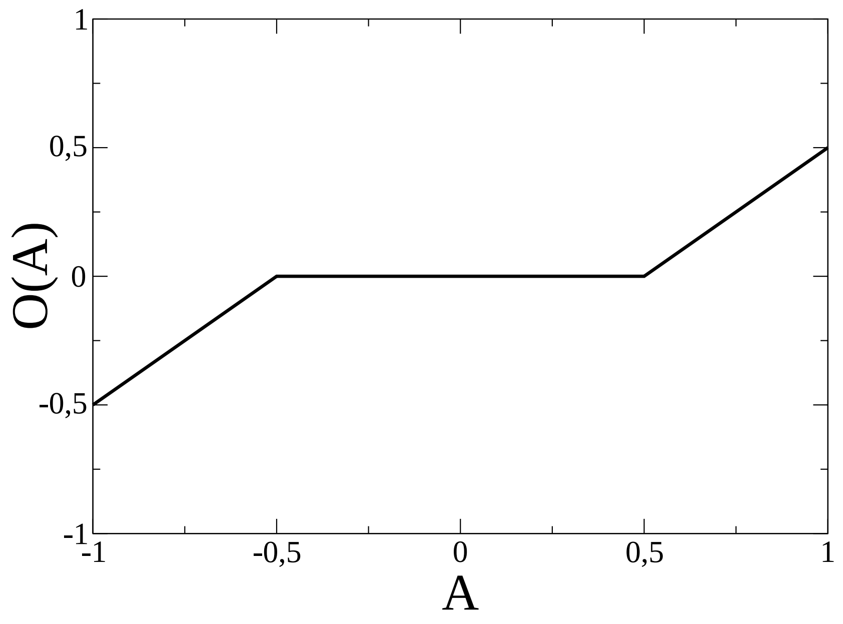


Figure 3.2: The units' activation function for $\lambda_i = 0.5$

When an input vector is presented to the network, its outputs need to be determined such that the equations (3.1) and (3.2) are fulfilled simultaneously for all units.

Harpur and Prager define an error function to measure reconstruction accuracy and sparseness. The error function that governs the dynamics of this network is similar, but a different λ_i was used for each unit which is determined by learning:

$$E = \underbrace{\frac{1}{2}\left(\vec{X} - \sum_{j=1}^{M} \vec{W}_j O_j\right)^2}_{E_{recon}} + \underbrace{\sum_{j=1}^{M} \lambda_j |O_j|}_{E_{sparse}} \tag{3.5}$$

E_{recon} measures the reconstruction accuracy, and E_{sparse} is the sparseness penalty. If (3.1) and (3.2) are fulfilled for all units, then this error function assumes a minimum. This can readily be seen by evaluating the derivate of E with respect to O_i (for a detailed proof, see appendix A):

$$\frac{\partial E}{\partial O_i} = -\vec{X}\vec{W}_i + \sum_{j=1}^{M} \vec{W}_i \vec{W}_j O_j + \lambda_i \text{sign}\left(O_i\right) \tag{3.6}$$

If $O_i \neq 0$ at the minimum, then, by substituting (3.3) and (3.2) into (3.6) (noting that in this case $\text{sign}\left(O_i\right) = \text{sign}\left(A_i\right)$), one finds that (3.6) vanishes, which is a necessary condition for a minimum of a function with a continuous first derivative.

Here, it is a sufficient condition as well as 3.5 is always positive. Should the output (at the minimum) be zero, however, then the minimum is located at a point where (3.6) is not continuous. In this case it is sufficient to require that

$$\begin{aligned} \lim_{O_i \to 0^+} \frac{\partial E}{\partial O_i} &> 0 \quad \text{and} \\ \lim_{O_i \to 0^-} \frac{\partial E}{\partial O_i} &< 0 \end{aligned} \tag{3.7}$$

which is fulfilled because $\left|\frac{\partial E}{\partial O_i}\right|_{O_i=0} = |A_i| < \lambda_i$. It is this discontinuity which gives rise to the dead-zone of the activation function: Whenever the activation of a unit is in the interval $[-\lambda_i, \lambda_i]$, the minimum of the error function is located at $O_i = 0$.

The error function (3.5) is similar to that which governs the dynamics of a Hopfield network [36, 37] and it can indeed be proven that when (3.1) and (3.2) are used for asynchronous sequential (one unit at a time) updating of the network's outputs, that the minimum of E forms a stable attractor. The proof is similar to that for an Hopfield network (see appendix B).

As the λ_i control the interval of activation in which the units are inactive, a unit will not respond to any input should its λ_i become very large. It is then possible to prune this unit without altering the code, since the code symbol it represents is then never used.

3.2.1 Activation algorithms

Three different activation algorithms were compared with respect to their speed, accuracy and sparseness of the resulting code. The employed sparseness measure is given by the number of inactive units (i.e. units with zero output). This is a suitable definition if the network is to be used for image coding and compression, as the outputs of inactive units need not to be stored.

3.2.2 Gradient descent with clipping

The first algorithm that was tried was an extension of simple gradient descent. Consider the case where $O_i = 0$ for some i at the minimum. As the sparsifier (i.e. the sparseness-promoting penalty term $\lambda_i |O_i|$) is not differentiable at this point, simple gradient descent would lead to oscillations in O_i around 0 with an amplitude that depended on the chosen step size. In order to avoid this, it simply sets $O_i = 0$ whenever the absolute value of the corresponding activation $|A_i| < \lambda_i$.

A good starting point for the search is $O_i = 0 \; \forall i = 1 \ldots M$ because a large fraction of the units can expected to be inactive at the minimum if the code is sparse.

3.2.3 Sequential updating

The aforementioned similarity between the REC model and a Hopfield network suggests to try an activation mechanism that works for the latter. Updating one unit at a time according to

$$A_i^{new} := \vec{X}\vec{W}_i - \sum_{j=1, j\neq i}^{M} \vec{W}_i \vec{W}_j O_j^{old} \tag{3.8}$$

$$O_i^{new} := O\left(A_i^{new}\right) \tag{3.9}$$

$$O_i^{old} := O_i^{new} \tag{3.10}$$

will always cause E to decrease or remain unchanged. Initially, it sets $A_i = \vec{X}\vec{W}_i$ and $O_i = 0$. Between two successive updates of one unit every other unit is reactivated. This process is repeated until either the network settles down in a stable configuration or a certain number of activation cycles is completed. Note that this algorithm does *not* require a choice of step size, which is a potential advantage. For a proof of convergence, see appendix B.

3.2.4 Quadratic programming

The error function 3.5 is quadratic in the O_i. Hence, *quadratic programming* techniques [25] can be employed for the minimization. These algorithms require a continuous first derivative of the function under consideration and converge to a point where the gradient vanishes. As $S(O_i) = |O_i|$ is not differentiable at $O_i = 0$, the minimum will not be found if it is located at a point where some of the $O_i = 0$. As mentioned above, in order to identify a minimum the gradient doesn't need to vanish, it is sufficient to require that conditions 3.7 hold. Lets assume it was already known that $\forall i \in I_0 = \{i_1, \ldots, i_K\} : O_i = 0$ and $\forall i \in I_1 = \{i_{K+1}, \ldots, i_M\} : O_i \neq 0$, $I_0 \cap I_1 = \emptyset$, $I_0 \cup I_1 = \{1, \ldots, M\}$. The sets I_0 and I_1 contain the indices of all inactive and all active units, respectively. The minimum with respect to the $M-K$ non-zero activations $\hat{\vec{O}} = \left(O_{i_{K+1}}, \ldots, O_{i_M}\right)^T$ would then be determined by setting the gradient of E with respect to these activations to zero, yielding

$$\mathbf{A}\hat{\vec{O}} = \vec{b} - \lambda\,\vec{s} \tag{3.11}$$

where $\mathbf{A} = \left(\vec{W}_{i_{K+1}}, \ldots, \vec{W}_{i_M}\right)^T \cdot \left(\vec{W}_{i_{K+1}}, \ldots, \vec{W}_{i_M}\right)$, $\vec{b} = \left(\vec{W}_{i_{K+1}}, \ldots, \vec{W}_{i_M}\right)^T \cdot \vec{X}$ and $\vec{s} = \left(s_{i_{K+1}} = \operatorname{sign}\left(O_{i_{K+1}}\right), \ldots, s_{i_M} = \operatorname{sign}\left(O_{i_M}\right)\right)^T$. The sign (x) function is here defined as

$$\operatorname{sign}(x) = \begin{cases} 1 & : \quad x \geq 0 \\ -1 & : \quad x < 0 \end{cases} \tag{3.12}$$

Of course, it is not *a priori* clear, which O_i will be zero and which will not. Thus, an iterative procedure has to be employed that produces configurations of I_0 and I_1 and terminates, when a configuration is consistent with conditions 3.7 and 3.11. A good initial guess is $I_1 = \emptyset$, because, as mentioned above, quite a large fraction of the O_i will be zero at the minimum. Putting all these considerations together leads to the following algorithm (in step 2, a simplification to the general quadratic programming algorithm was used as the O_i are decoupled with respect to the constraints) :

1. Set $K = M$, $I_0 = \{1, \ldots, M\}$, $I_1 = \emptyset$

2. Determine the minimum $\hat{\vec{O}} = \left(O_{i_{K+1}}, \ldots, O_{i_M}\right)^T$ of E (eqn. 3.5) subject to the constraints $\forall i \in I_0 : O_i = 0$ and $\forall i \in I_1 : (\text{sign}\,(O_i) = s_i \vee O_i = 0)$ using a simplified version of the *active set quadratic programming* technique [25].

3. Update $I_0 \to I_0 \cup \{i | O_i = 0 \wedge i \in I_1\}$ and $I_1 \to I_1 \setminus \{i | O_i = 0 \wedge i \in I_1\}$. Update $\hat{\vec{O}}$ and $\vec{s}$ by removing all the components that have become 0.

4. Calculate

$$\sigma = \max_{i \in I_0} \left(\text{sign}\left(z_i^+\right) \cdot \text{sign}\left(z_i^-\right) \min(|z_i^+|, |z_i^-|)\right) \tag{3.13}$$

   where $z_i^+ = \lim_{O_i \to 0^+} \frac{\partial E}{\partial O_i}$ and $z_i^- = \lim_{O_i \to 0^-} \frac{\partial E}{\partial O_i}$. If $\sigma > 0$, then condition (3.7) is violated and the corresponding $O_i \neq 0$ at the minimum. Thus, update $I_0 \to I_0 \setminus \{O_i\}$, $I_1 \to I_1 \cup \{O_i\}$, decrement K by one, add a component to $\hat{\vec{O}}$ and $\vec{s} \to (\vec{s}| - \text{sign}\left(z_i^+\right))$ and goto step 2.

5. Return $\hat{\vec{O}}$, I_0, and I_1.

6. End.

3.3 Learning

The learning process is best understood if the network is viewed as generator of the data. For a given configuration of network parameters $\vec{O} = (O_1, \ldots, O_N)$, $\vec{\lambda} = (\lambda_1, \ldots, \lambda_N)$ and $\mathbf{W} = (\vec{W}_1, \ldots, \vec{W}_N)$ the probability that it produces the output $\vec{X}$ is assumed to have Gaussian density, since this is the maximum entropy density for random noise with fixed variance:

$$p(\vec{X} | \vec{O}, \vec{\lambda}, \mathbf{W}) = \frac{1}{\sqrt{2\pi}^N} \exp\left(-\frac{1}{2}\left(\vec{X} - \sum_{j=1}^{M} \vec{W}_j O_j\right)^2\right) \tag{3.14}$$

The prior probability density for the outputs is a product of exponential distributions (this is a consequence of the linear sparsifier):

$$p(\vec{O} | \vec{\lambda}, \mathbf{W}) = \Pi_{j=1}^{M} \frac{\lambda_j}{2} \exp(-\lambda_j |O_j|) \tag{3.15}$$

i.e. the outputs are assumed to be mutually independent. Multiplying (3.14) with (3.15) yields the probability for $\vec{X}$ and $\vec{O}$ given $\vec{\lambda}$ and $\mathbf{W}$, its negative logarithm

$$-\ln(p(\vec{X},\vec{O}|\vec{\lambda},\mathbf{W})) = \underbrace{\frac{1}{2}\left(\vec{X}-\sum_{j=1}^{M}\vec{W}_jO_j\right)^2}_{E_{recon}} + \underbrace{\sum_{j=1}^{M}\lambda_j|O_j| - \sum_{j=1}^{M}\ln(\lambda_j)}_{E_{sparse}} + \frac{N}{2}\ln(2\pi) \tag{3.16}$$

is the quantity that is being minimized when the network is activated, as it is (except for contributions that do not depend on the outputs) equal to (3.5). The outputs that minimize (3.16) for input vector $\vec{X}^r$ will be denoted as $\vec{O}^r$ in the following.

For a given training set of examples $\mathbf{X} = (\vec{X}^1,\ldots,\vec{X}^R)$, $\vec{\lambda}$ and $\mathbf{W}$ are then determined by minimizing

$$E_{tot} = E\left[E^r\right]_{\mathbf{X}} = -E\left[\ln\left(p(\vec{X}^r,\vec{O}^r|\vec{\lambda},\mathbf{W})\right)\right]_{\mathbf{X}} \tag{3.17}$$

subject to the constraint

$$|\vec{W}_i| = 1 \tag{3.18}$$

The average is performed over all examples in the training set. Stochastic gradient descent (a variation of simple gradient descent where the examples are presented in random order) was used to carry out the minimization, the necessary derivatives are:

$$\frac{\partial E^r}{\partial \lambda_i} = |O_i^r| - \frac{1}{\lambda_i} \tag{3.19}$$

$$\nabla_{\vec{W}_i}E^r = -\left(\vec{X}^r - \sum_{j=1}^{M}\vec{W}_jO_j^r\right)O_i^r \tag{3.20}$$

As can be seen in (3.19), the derivative of E^r with respect to λ_i consists of two terms. The first term pulls (when gradient descent is performed) λ_i towards smaller values, the second term pulls towards higher ones, the stronger, the smaller λ_i is. This is where the pruning arises: If a unit hardly ever has an $O_i \neq 0$, the second part of (3.19) will dominate and λ_i will grow boundlessly. Hence, the unit will not get activated at all and can be pruned. For numerical stability and model comparison purposes (see discussion below), an upper bound on λ_i was imposed. The precise value of this bound is not critical, as long as it's high enough (say e.g. 10^2) to guarantee that the unit does not get activated once its λ_i has grown to this value.

A lower bound on λ^j has to be imposed as well to make sure that it does not become negative, in that case $p(\vec{X}, \vec{O}|\vec{\lambda}, \mathbf{W})$ would no longer be a probability density.

The assumption of mutually independent outputs in (3.15) might seem invalid, especially at the early stages of learning. However, Harpur and Prager argued in [35] that a factorial form of the prior probability in conjunction with a suitable constraint on the overall output entropy (here given by the quadratic part of E which measures the quality of reconstruction) will eventually give rise to a factorial posterior probability of the outputs.

3.4 Results

3.4.1 Comparison of activation algorithms

Natural image data were used to compare the performances of the three activation algorithms, so as to determine which one would be preferable in practical applications of the network. The results in this section were partly presented on the *ICANN99* conference (see [20]). Here, the λ_i were not adjusted according to 3.19 in the course of learning, but held fixed at a value of 0.5. This due to the fact that at this stage the pruning framework had not been developed.

The quadratic programming algorithm was used as a benchmark for the other two methods since the computational time it needs to terminate cannot be adjusted. On the computer which was used for the simulations (a Silicon Graphics workstation) it took between 0.44 s (at the beginning of training) and 0.034 s (towards the end of training) to converge, 0.074 s on average. This considerable time difference is due to the fact that it will generally converge faster if the code is sparser. Both the gradient descent as well as the sequential updating algorithms require always the same amount of time per step. The average convergence time allowed for approximately 38 gradient steps or 33 complete cycles of sequential updating, after which these two activation methods were terminated, regardless of whether they had converged to their final fixed points or not.

Table 3.1 shows the averaged squared reconstruction error per pixel and the average fraction of active units per example after training with 40000 examples.

activation method	reconstruction error	fraction of active units
gradient descent	0.031	0.18
sequential updating	0.030	0.11
quadratic programming	0.030	0.11

Table 3.1: Comparison of activation algorithms after training with 40000 examples w.r.t. the reconstruction error per pixel and the fraction of inactive units per example. Averages computed over 100 examples. For details, see text.

Quadratic programming and sequential updating produce codes which are almost equally sparse, whereas gradient descent does not do quite as good: For a given patch, it leaves ≈ 1.6 times as many units activated as the other two. Yet the average reconstruction errors per pixel are comparable.

As quadratic programming and sequential updating seem to do equally well, one might wonder why one would want to implement the former, which is much more complicated. The answer is that is about twice as fast (0.034 s compared to 0.074 s in this case) once the basis vectors have converged. Thus, if one wants to use this network in practice, quadratic programming seems to be the preferable choice of algorithm.

The runtime of the quadratic programming method appears to scale with the number of active units, whereas sequential updating scales with the number of units in the network. Thus, one would expect the gain of the former over the latter to increase with the network size, providing that the number of units used to encode a given input remains the same.

3.4.2 Pruning

The network was first tested on an artifical data set created by a fixed network of similar architecture. Although this 'teacher/student' scenario might seem rather contrived, it is quite useful for determining whether the learning algorithm works as desired. The teacher network had $M = 3$ units with $N = 9$ outputs, the units were randomly activated with a probability distribution given by (3.15). Its weights were normalized random vectors. The student had initially $N = 9$ units with $M = 9$ inputs, it was activated using the quadratic programming technique. After training on 100000 examples with a learning rate of 0.002 for the weights and 0.0002 for the $\vec{\lambda}$, the algorithm had pruned six units. This difference in the learning rate is due to the fact that the weights have to be stationary in comparison to the λ adaptation process for good results. At the present time, upper bounds for the learning rate have not been determined. The remaining three related to the units of the teacher as shown in the table below. Here, $\vec{W}_i^t\vec{W}_i^s$ is the overlap of the i-th weight vector of the teacher with the corresponding vector of the student. The student's units were sorted for maximum overlap with the teacher.

unit	1	2	3
teacher λ_i	0.10	0.15	0.20
student λ_i	0.10	0.16	0.21
$\vec{W}_i^t\vec{W}_i^s$	0.99	0.98	0.98

Both the weights and the $\vec{\lambda}$ model the teacher's parameters fairly well.

Secondly, the algorithm was run on image data. Training was performed on 43 natural images (people, natural scenes etc.), each image patch was individually normalized to zero mean and unit variance. Each unit had a RF of 13×13 pixels, the fields were fully overlapping. Initially, the network had 169 units, all λ_j were set to 0.5. After training on 200000 examples with a learning rate of 0.005 for the weights and 0.0005 for the $\vec{\lambda}$ (here, a λ_i was only updated when the corresponding $O_i \neq 0$, which turned out to give better reconstruction), the algorithm had pruned 31 units (a unit was considered inactive if it's mean activation was smaller than 0.01).

In fig. 3.3 a picture, which was not part of the training set, and its reconstruction are shown. Although the code is highly sparse on a population level (only $\approx 11\%$ of the units are active for a given patch), the reconstruction resembles the original still fairly well. Numerically speaking, the code accounts for about 95% of the data variance.

Figure 3.4 shows the evolution of the average total error, the average reconstruction error and the sparseness penalty (see eqn. (3.16)) in the course of training. Note that E_{recon} is not monotonically decreasing: there is a minimum at ≈ 5000

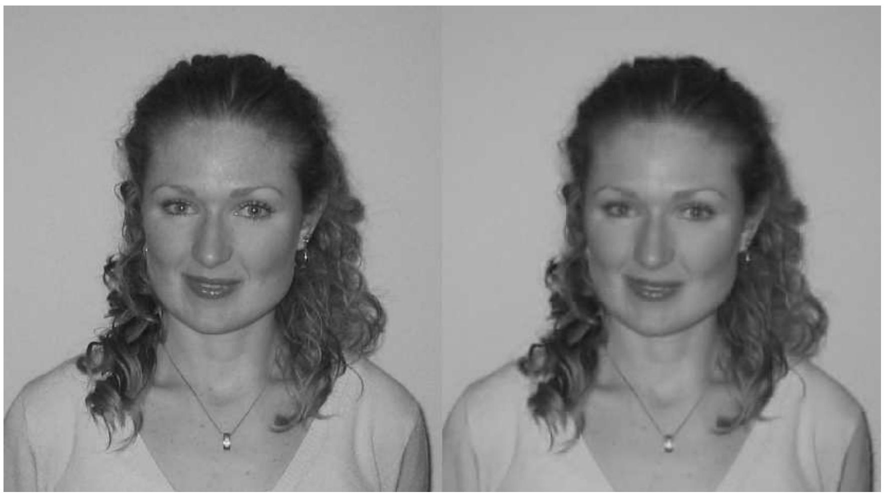


Figure 3.3: A picture (left) and its reconstruction (right).

examples, then it increases until ≈ 95000 examples. However, E_{sparse} is decreasing in that range, such that the total error E is also decreasing. This highlights the fact that the network is not just trying to minimize the reconstruction error, but seeks to achieve a compromise between good reconstruction and sparseness, as defined by eqn. (3.5). Consequently, stopping after ≈ 5000 examples, when a minimum of E_{recon} is reached, would not be conducive to the overall goal of learning a representation of the data which is both faithful and sparse.

3.4.3 Overcompleteness

It is well established that V1 contains significantly more cells than it receives inputs. While ratios differ between 25:1 [62] and 100:1 [24], there certainly is a consensus that the factor is (a lot) greater than one. In [65], the authors conjectured that this increase of code elements is used to re-represent the visual input in a sparser, *overcomplete* form. A code is said to be overcomplete whenever the number of code elements exceeds the number of possible values of the quantity to be encoded. In the case of the network used here, that means that the number of basis vectors M is greater than the number N of pixels in each unit's RF.

To determine the network's behavior in the overcomplete case, it was trained on examples drawn from natural images. Each example was individually normalized to zero mean and unit length. Furthermore, the λ_i were held constant at 0.5 and pruning was not applied to force the network into developing an overcomplete code.

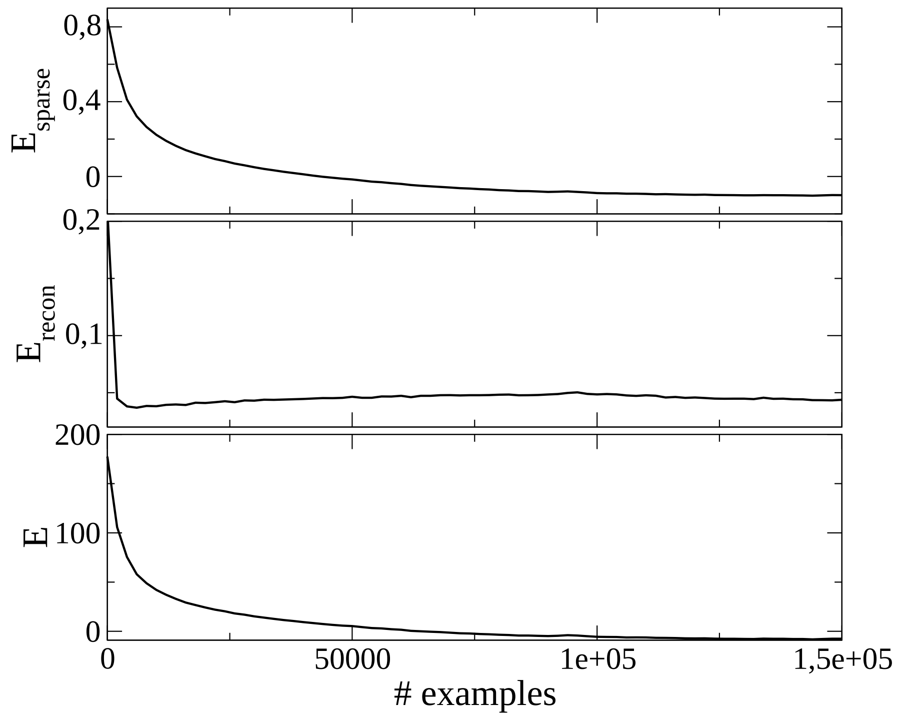


Figure 3.4: Evolution the sparseness penalty (see eqn. (3.16)) per basis vector $\frac{E_{sparse}}{M}$ (top graph), the reconstruction error per pixel $\frac{E_{recon}}{N}$ (middle graph) and the total error E (bottom graph) in the course of training. All values averaged over 2000 examples.

The learning rate was initially set to 0.005, and gradually reduced to 0.0005 in the second half of training. This was found to be necessary to get the basis vectors to converge in the overcomplete case.

Table 3.2 shows the degree of overcompleteness (i.e. the factor k such that $M = k \times N$), the number of training examples, the average squared reconstruction error per pixel and the average fraction of active units per example. All networks had RFs consisting of 13x13 pixels. The fraction of active units per example can be regarded as a measure for population sparseness, which appears to be roughly inversely proportional to the degree of overcompleteness. The average squared reconstruction error per pixel decreases with increasing overcompleteness as well, which is an indication that the added units are actually used by the network, i.e. the code is not only sparse on a population level, but also on a single-unit level.

Stronger evidence of single-unit sparseness development can be seen in fig. 3.5, which shows the average output $E\left[|O_i|\right]$ per example for each unit in the 4x over-

overcompleteness	training examples	rec. error	fraction of active units
complete	200000	0.037	0.16
2x	400000	0.026	0.077
4x	1600000	0.021	0.041

Table 3.2: Comparison of network performances for different degrees of overcompleteness. All networks had RFs with $N = 169$, i.e. 13x13 pixels. Overcompleteness is the factor between N and the number of units M, 'complete' means $M = N$. Shown are the number of training examples, the average squared reconstruction error per pixel, and the average fraction of active units per example, which is a measure for population sparseness. Averages computed over 2000 examples.

complete network. In the early stages of training, population sparseness is increased by decreasing each unit's average output (red line vs. dashed green line). Lateron, single-unit sparseness is developed by distributing the average outputs more evenly across the units (dashed green line vs. black line).

The resulting RFs for a 4x overcomplete network are shown in fig. 3.6. They are qualitatively similar to the pruned, complete and 2x overcomplete RFs: many are localized, and virtually all are oriented.

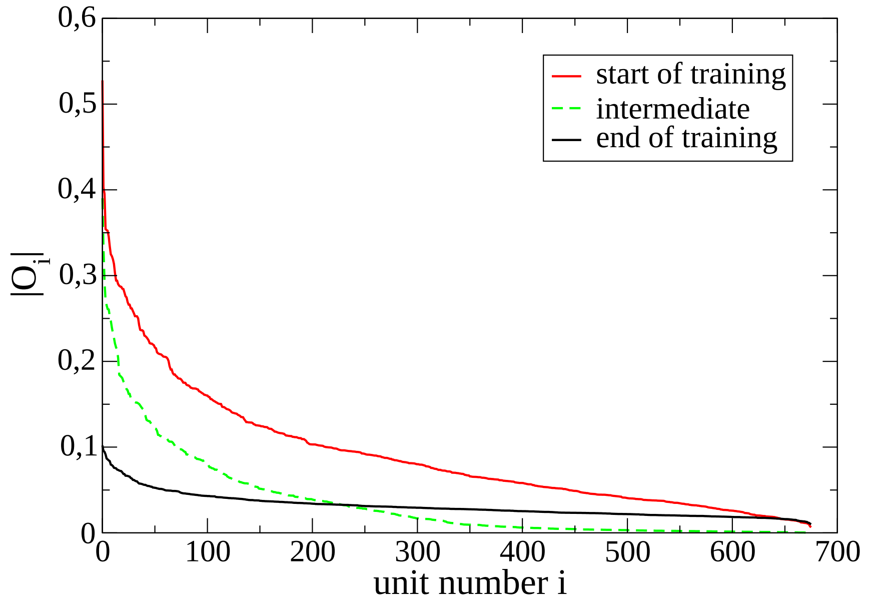


Figure 3.5: Average output $E\left[|O_i|\right]$ per example for all units of the 4x over-complete network at the beginning of training (red line), after 20000 examples (dashed green line) and at the end of training (black line, 1600000 examples). For each curve, the units were ordered descendingly by $E\left[|O_i|\right]$, thus the unit numbers i differ between curves. In the early stages of training, population sparseness is increased by decreasing each unit's average output. Lateron, single-unit sparseness is developed by distributing the average outputs more evenly across the units.

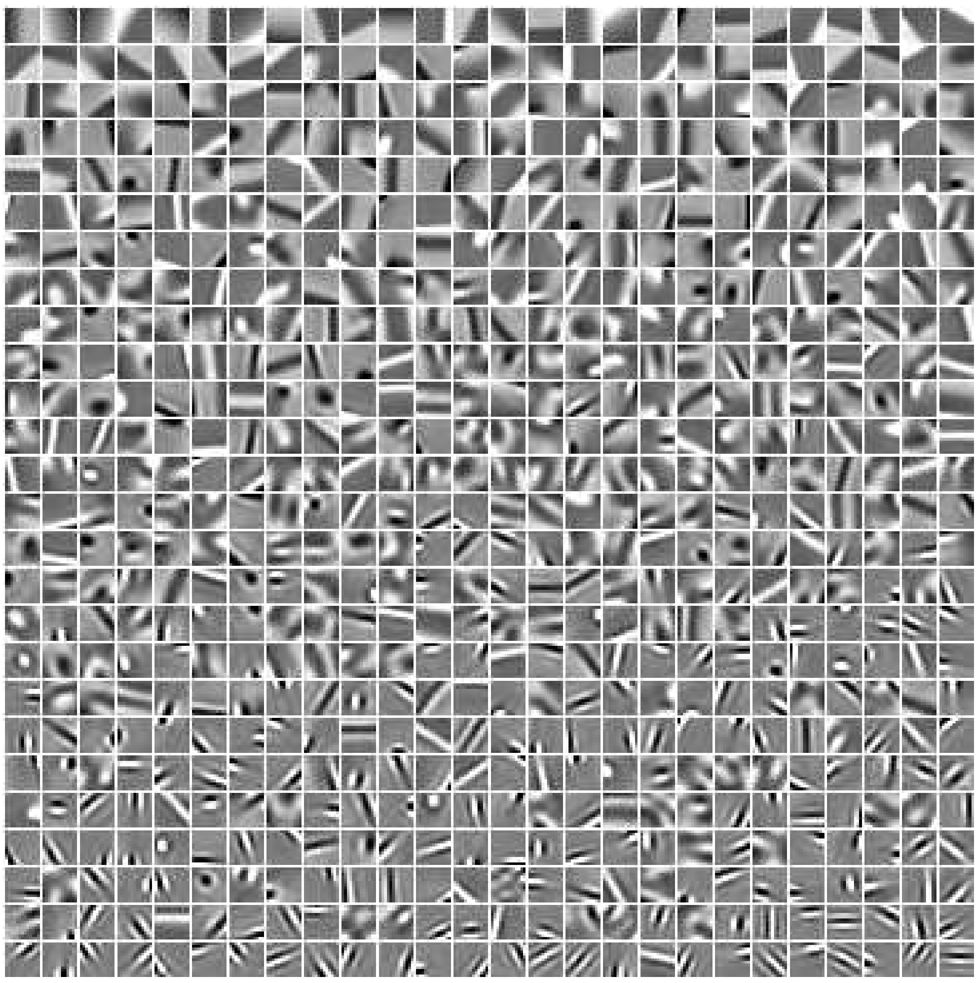


Figure 3.6: The basis vectors of a 4 times overcomplete network after 1600000 examples. The vectors are ordered from top left (greatest $E\left[|O_i|\right]$) to bottom right (smallest $E\left[|O_i|\right]$) and the grayscales are individually normalized.

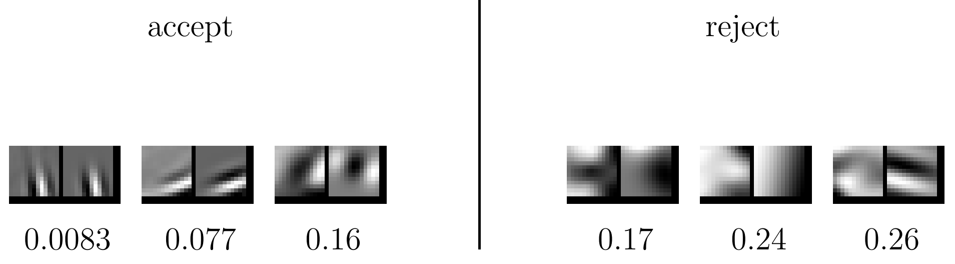


Figure 3.7: Each image represents a RF (left half) from the 2x overcomplete network and the fitted Gabor function (right half). Below the images are the corresponding fit errors. Fits with an error smaller than 0.17 were accepted, the rest was rejected.

3.4.4 Quantitative comparison to V1 data and the generative model by Olshausen & Field

For a more quantitative comparsion of the learned RFs, Gabor functions

$$
\begin{aligned}
g(x_r, y_r, \sigma_x, \sigma_y, \lambda, \phi) &= \exp\left(-\frac{x_r^2}{2\sigma_x^2} - \frac{y_r^2}{2\sigma_y^2}\right)\cos\left(2\pi\frac{x_r}{\lambda} + \phi\right) && (3.21)\\
x_r &= (x - x_0)\cos(\theta) + (y - y_0)\sin(\theta) && (3.22)\\
y_r &= -(x - x_0)\sin(\theta) + (y - y_0)\cos(\theta) && (3.23)
\end{aligned}
$$

were least-square fitted to the RFs. x, y are the coordinates within a RF, x_0, y_0 are the origin coordinates of the gabor function, θ is the rotation angle between the coordinate systems of the RF and the gabor function, σ_x, σ_y are the standard deviations of the Gaussian envelope in x_r and y_r direction, λ is the wavelength and ϕ is the phase angle. Gabor functions have long been regarded as models for RFs of simple cells from V1 [44].

The majority of the RFs could be fitted well by Gabor functions, there were however a number of cases in which no good match could be found. The fit error limit E_{lim} for acceptance/rejection was determined by visual inspection of the RFs (normalized to unit length) and fitted Gabors of the 2x overcomplete network. As illustrated in fig. 3.7, $E_{lim} = 0.17$ was found to be a reasonable value.

Furthermore, the generative model described in [63, 64, 66] and, in a similar form, in [50], which will be called the OF model in the following, was trained on the same images for comparison. The OF model is similar to the one described in this thesis, except for the sparsifier (E_{sparse} of eqn. (3.5)), which is now

$$E_{sparse} = \lambda_i \log\left(1 + O_i^2\right) \tag{3.24}$$

In [66], the authors trained their network on whitened natural images. Due to the statistical structure of natural images, whitening essentially boosts spatial frequency components in the mid to upper range. Since it was found in [77, 62] that the RFs of [66] overrepresent higher spatial frequencies (as compared to V1 RFs from macaque monkeys), I decided to work with non-whitened images. The λ_i were all set to 0.5.

network	# accepted	# rejected
pruned	135	3
complete	168	0
2x overcomp.	318	20
4x overcomp.	542	143
OF complete	156	13
OF 2x overcomp.	223	115

Table 3.3: Number of accepted and rejected Gabor fits ($E_{lim} = 0.17$) for each of the tried networks.

Table 3.3 shows the number of accepted and rejected Gabor fits for each of the tried networks. The OF network was not tested in a 4x overcomplete scenario, because I could not find a parameter setting which would lead to convergence of the basis vectors. The fraction of rejected fits increases with overcompleteness, at the current time it is not clear why this should be so and whether it could be remedied.

Spatial frequency distribution and aspect ratios

In neurophysiological experiments, spatial frequency is usually measured in cycles/degree. Since it is not sensibly possible to translate pixels into degrees, I chose, like [77], to compute the scale-free quantity

$$n_x = \frac{\sigma_x}{\lambda} \tag{3.25}$$

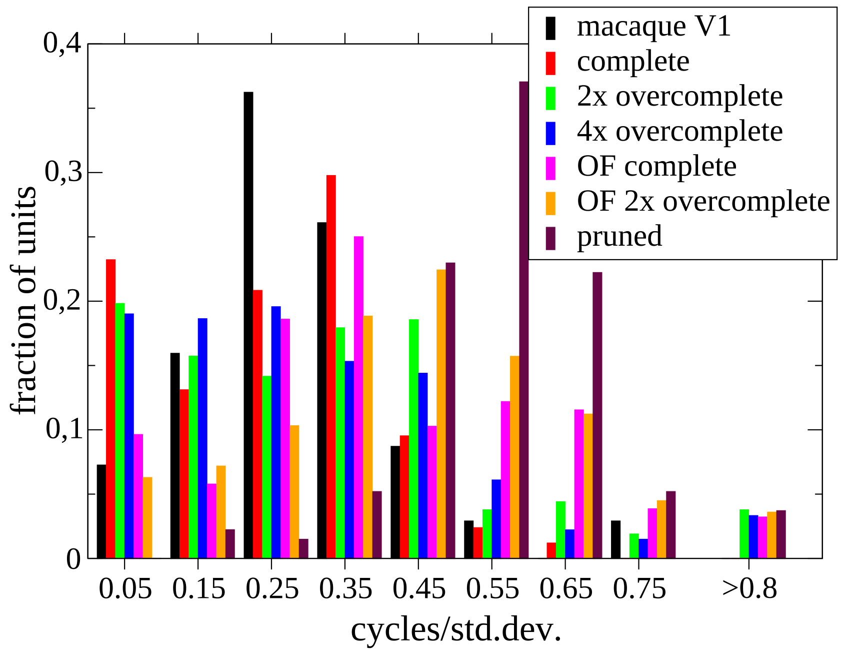


Figure 3.8: Spatial frequency distribution of the RFs of the tried networks compared to RFs from macaque V1 from [77]. The $n_x = \frac{\sigma_x}{\lambda}$ (see eqn. (3.21)) were binned into 9 bins, the first eight of which had a width of 0.1 and their centers located at the points indicated at the abscissa. The last bin contained all observations > 0.8.

where σ_x is the standard deviation of the Gaussian envelope of the Gabor function in x_r direction and λ is its wavelength (see eqn. (3.21)). This quantity measures the number of cycles (of the cosine part of the Gabor function) that fit into one standard deviation σ_x.

The distributions of n_x for the different networks, along with data from macaque V1 from [77], are plotted in fig. 3.8. All networks' n_xs are found largely in the same range as the macaque V1 data, but the peak at 0.25 is reproduced by neither of them.

Table 3.4, second and third column, lists the average n_x and their standard errors. Cat V1 data were obtained from [44]. My network's average values (complete, 2x and 4x overcomplete) are closer to macaque V1 than cat V1, the 4x overcomplete network matches the macaque V1 value within one standard error. The OF network veers towards the cat V1 value, the 2x overcomplete one is within a standard error

network	avg. n_x	std.err. n_x	avg. AR	std.dev. AR
pruned	0.547	0.013	1.36	0.40
complete	0.253	0.011	1.67	0.19
2x overcomp.	0.316	0.012	1.25	0.35
4x overcomp.	0.2985	0.0090	1.42	0.13
OF complete	0.396	0.017	1.69	0.48
OF 2x overcomp.	0.436	0.014	2.04	0.65
macaque V1	0.284	0.016	1.35	0.18
cat V1	0.479	0.037	1.62	0.73

Table 3.4: Second and third column: average cycles per standard deviation n_x of the fitted Gabor functions and their standard errors. Fourth and fifth column: average aspect ratio (AR) and standard deviation, computed by fitting a line through the origin of a σ_x, σ_y (see eqn. (3.21)) scatterplot. Standard deviation of the aspect ratio was calculated from the unexplained variance after the fit. Macaque V1 data from [77], cat V1 data from [44].

of cat V1 data. The pruned network's n_xs are significantly higher.

The fourth and fifth colum of table 3.4 show the average aspect ratios (AR) and their standard deviations of the fitted Gabor functions, computed by fitting a line through the origin of a σ_x, σ_y (see eqn. (3.21)) scatterplot[1]. The standard deviation of the aspect ratio was calculated from the unexplained variance after the fit. The 4x overcomplete network, as well as the pruned one, are the closest matches for the macaque V1 data, whereas the complete OF network shows the smallest difference to cat V1. However, given the large unexplained variances, it is not possible to make a definite statement at this time. What can be said with some certainty is that all networks produce average aspect ratios in the same range as those found in V1.

[1]In a number of instances, σ_y of the best fit was found to be much larger than the radius of the RF, e.g. when the RF contained a single edge that ran across it. In those cases, the variation of the fit error w.r.t. σ_y was also very slight whenever σ_y was greater than the radius of the RF, making it impossible to determine a single best σ_y. These Gabor fits were excluded from the aspect ratio analysis.

Position tiling and rotation angle distribution

Figure 3.9, top, shows the spatial tiling. Each symbol represents the center coordinates x_0, y_0 (see eqn. (3.21)) of one fitted Gabor function. The black box outlines the boundaries of the RF. Most Gabor centers lie within the RF. The distribution inside the RF appears largely uniform, with the 4x overcomplete network exhibiting a weak central tendency. The distribution becomes denser with increasing overcompleteness, i.e. position space is more evenly covered as units are added to the network. The spatial tiling of the OF network exhibited the same characteristics.

In fig. 3.9, bottom, the distribution of the rotation angle θ is plotted. All network show a tendency towards horizontal and vertical orientation. This might be the result of a bias from the dataset used for training.

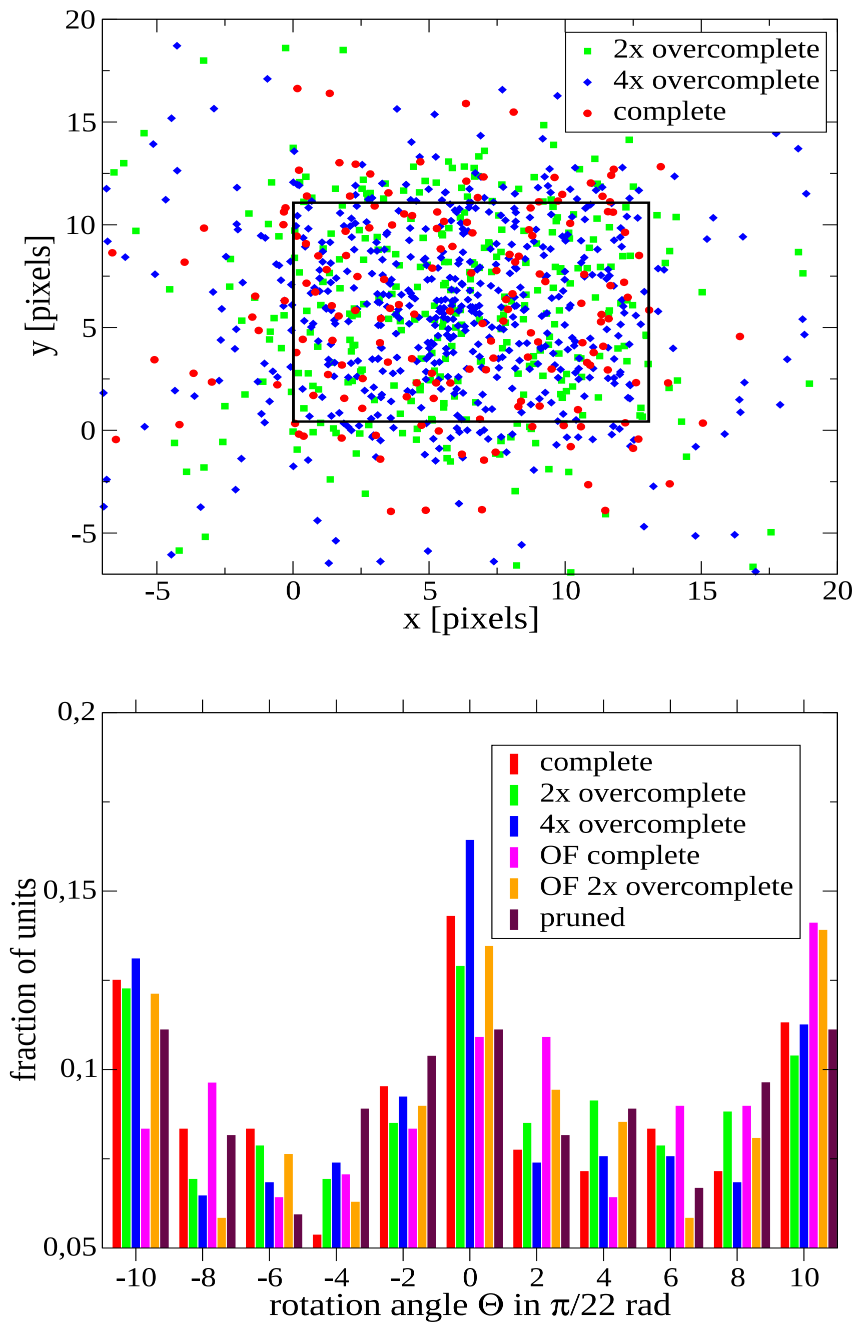


Figure 3.9: Top: position tiling. Each symbol represents the center location x_0, y_0 of a fitted Gabor function (see eqn. (3.21)). The black box outlines the boundaries of the RF. Bottom: Distribution of rotation angles $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. Each bin had a width of $\frac{\pi}{11}$, the bin center values are plotted on the abscissa in units of $\frac{\pi}{22}$.

Spatial phase distribution

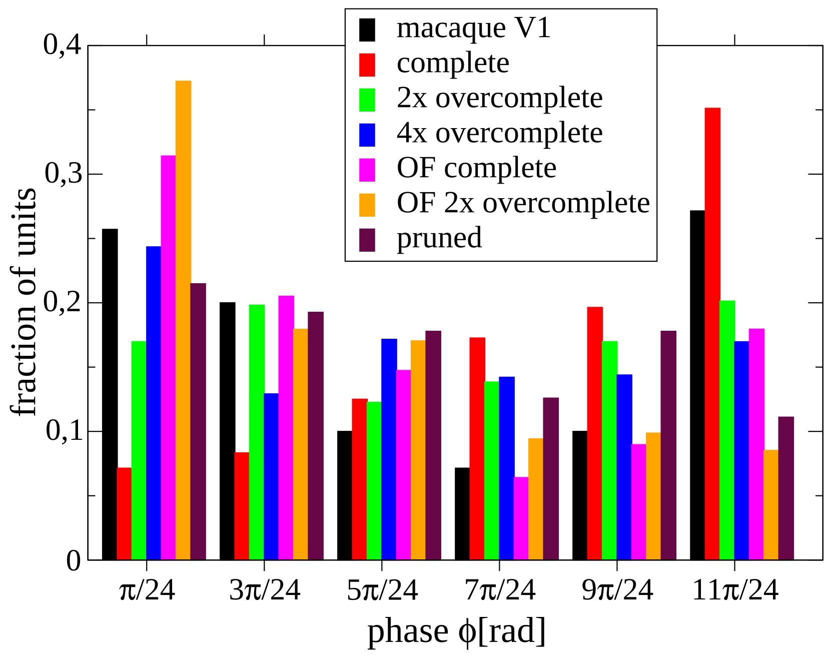


Figure 3.10: Spatial phase distribution of the networks compared to macaque V1 data from [77]. The phase angles ϕ were first mapped onto the interval $[0, \frac{\pi}{2}]$, then binned into 6 bins, each of which had a width of $\frac{\pi}{12}$. The values on the abscissa denote the bin centers. For details, see text.

The spatial phase distributions of the networks are compared to V1 data in fig. 3.10. Prior to binning, the phase angles ϕ were transformed into the interval $[0, \frac{\pi}{2}]$ via the following symmetries of the cosine part of eqn. (3.21) [77]:

$$g(x_r, y_r, \sigma_x, \sigma_y, \lambda, \phi) = g(x_r, y_r, \sigma_x, \sigma_y, \lambda, \phi + 2\pi) \tag{3.26}$$

$$g(x_r, y_r, \sigma_x, \sigma_y, \lambda, \phi) = -g(x_r, y_r, \sigma_x, \sigma_y, \lambda, \phi + \pi) \tag{3.27}$$

$$g(x_r, y_r, \sigma_x, \sigma_y, \lambda, \phi) = -g(-x_r, y_r, \sigma_x, \sigma_y, \lambda, \pi - \phi) \tag{3.28}$$

Eqn. (3.26) follows from the periodicity of the cosine function, and allows for a mapping of ϕ onto the interval $[0, 2\pi]$ without changing the RF. Eqn. (3.27) states that ϕ can always be chosen within $[0, \pi]$ if sign changes are ignored. Since sign changes do not alter the basic shape of the RF, this transformation can be applied. Finally, a further restriction of ϕ on the interval $[0, \frac{\pi}{2}]$ is possible via eqn. (3.28)[2],

[2]Eqn. (3.28) can be derived using $\cos(\frac{\pi}{2} + \alpha) = -\sin(\alpha) = \sin(-\alpha) = -(-\sin(-\alpha)) =$

if RF differences attributable to mirror symmetries around the x_r-axis are also discarded. The remaining information in ϕ is then the phase relative to the x_r origin, i.e. RFs with ϕ close to 0 have even symmetry and may be thought of as 'line detectors' ($\sigma_y \gg \sigma_x$) or 'blob detectors' ($\sigma_y \approx \sigma_x$), whereas RFs with $\phi \approx \frac{\pi}{2}$ have odd symmetry, thus these units are responsive to edges.

As can be seen in fig. 3.10, simple cells in macaque V1 tend towards the extremes of the phase angle. In contrast, the complete network learns mostly edge detectors. This trend is less pronounced in the 2x overcomplete network, and is reversed in the 4x overcomplete one. The phase angle distribution of the complete OF network reproduces the peaks at $\phi = 0$ and $\phi = \frac{\pi}{2}$, but the 2x overcomplete OF network has mostly RFs with even symmetry.

network	edge/line ratio
pruned	0.71
complete	3.54
2x overcomp.	1.01
4x overcomp.	0.84
OF complete	0.52
OF 2x overcomp.	0.33
macaque V1	0.81

Table 3.5: Ratios of edge vs. line detectors for the different networks and macaque V1 [77]. Values computed by dividing the sum of the last two bins in fig. 3.10 by the sum of the first two bins.

For further comparsion, table 3.5 lists the ratios of edge vs. line detectors of the different networks. These values are the quotients of the sums of the last two bins of fig 3.10 and of the first two, i.e. the number of RFs with $\phi \in [\frac{8\pi}{24}, \frac{\pi}{2}]$ divided by the number of RFs with $\phi \in [0, \frac{4\pi}{24}]$. The pruned network's ratio is comparable to the macaque V1 value. The complete, 2x and 4x overcomplete networks tend towards the macaque V1 ratio with increasing overcompleteness. In contrast, the OF network seems to move away from it.

$-\cos(\frac{\pi}{2} - \alpha)$.

3.5 Discussion

The presented algorithm has two main features: Firstly, it learns a sparse code when the data allow for it (e.g. natural pictures) and a non-sparse code otherwise, as in the case of the teacher/student scenario. Secondly, it is able to determine the number of units necessary for good coding. In general, a code will be called 'good' if the mutual information between the data and the code is high. A simple way of promoting this, as proposed here, is to include the reconstruction error in the error function. Furthermore, for reasons of limited bandwidth of transmission lines and/or limited storage space, codes are often required to contain as little redundancy as possible, or, in terms of information theory, the entropy of the data after coding should be minimized. One way of achieving this is ensuring that the code alphabet contains only as many letters (here: units) as needed, and that each letter's entropy is low. The algorithm reaches the second goal by assuming a prior probability for the units which is peaked around zero and the first goal through pruning unnecessary units. The pruning process is an approximate form of Bayesian model comparison (see section 2.5.1 and[53]). One is looking for the model with the highest posterior probability:

$$p(M, \vec{\lambda}, \mathbf{W}|\{\vec{X}\}) = \int d\vec{O} p(\vec{O}, M, \vec{\lambda}, \mathbf{W}|\{\vec{X})\} \tag{3.29}$$

Where $\{\vec{X}\}$ denotes the set of all input vectors and M is the number of units. The integrand on the right hand side can be rewritten as

$$p(\vec{O}, M, \vec{\lambda}, \mathbf{W}|\{\vec{X}\}) = \frac{p(\{\vec{X}\}|\vec{O}, M, \vec{\lambda}, \mathbf{W})p(\vec{O}|M, \vec{\lambda}, \vec{W})p(M, \vec{\lambda}, \mathbf{W})}{P(\{\vec{X}\})} \tag{3.30}$$

The term $p(\{\vec{X}\}|\vec{O}, M, \vec{\lambda}, \mathbf{W})$ is the likelihood of the data set, which in my case is the product of the individual likelihoods (3.14) of each example, $p(\vec{O}|M, \vec{\lambda}, \vec{W})$ is the prior of the activations (3.15) and $p(M, \vec{\lambda}, \mathbf{W})$ is the model prior. As one prefers models with a small number of units, each unit can be assigned a cost, or, in Bayesian terms, a prior probability. There is neither an upper nor a lower bound for the number units a priori, so a good assumption would be that the total prior probability for M units is proportional to the prior probability P_u for a single unit to the M-th power, i.e. the more units, the more unlikely the model. There is, however, no a priori preference for any values of the λ_i nor for the $\vec{W}_i$ (except that they be normalized), hence the model prior is equal to P_u^M except for a normalization constant. The part of the density (3.30) which is relevant to learning ($P(\{\vec{X}\})$ does not depend on the model parameters) can then be rewritten as:

$$p(\vec{X}|\vec{O}, M, \vec{\lambda}, \mathbf{W})p(\vec{O}|M, \vec{\lambda}, \vec{W})P(M) \tag{3.31}$$

When, in the course of learning, a unit turns out to be never (or at least hardly ever) activated, its contribution to the likelihood of the data will vanish, whereas it still reduces the posterior probability of the model by its presence alone (because $P_u^M < P_u^{M-1}$) . Thus, pruning this unit increases the probability of the model.

However, the above presented comparisons indicate that pruning, at least in the form presented here, does require future refinement. In several instances (spatial frequency, aspect ratio and edge/line ratio) it was the 4x overcomplete network which provided RFs whose properties were closest to those of macaque V1 RFs. It would seem necessary to invest more research effort into an exact Bayesian learning scheme for the λ_i, or at least into a better approximation than the one presented here. Something to that end, albeit for learning the basis vectors, has been undertaken in [50]. Moreover, the prior over the λ_i, which is in my approximation assumed to be uniform, should also be reconsidered.

The OF network's RFs are generally more dissimilar to macaque V1 RFs than those of my network, but in two cases (spatial frequency and aspect ratio) they seem to be reasonably close to cat V1, which was also reported in [63, 64].

The observed convergence acceleration of the inner loop (i.e. the pattern recognition process) towards the end of learning also emphasizes an important property of sparse codes. Neurophysiological evidence suggests that the visual system of primates does not spend much time on having feedback loops converge. Indeed, it has been argued that there might be hardly any recurrent connections participating in the generation of a representation of an image in area STS [69]. If the neural code was dense, convergence could potentially take very long, because many units have to 'agree' on a joint stable state, with each neuron contributing some output. In contrast, in the sparse case, only a few units take part in this competition, and therefore, convergence is faster.

3.5.1 The relationships between sparseness, overcompleteness, independence and redundancy reduction

As already mentioned in section 3.1, single-unit sparseness does not necessarily imply population sparseness. However, the former can be a sufficient condition for the latter if combined with the requirement of statistical independence amongst the code elements: given that each code element has a probability $p \ll 1$ of being used in the encoding of an input (i.e. the code is sparse on a single-unit level), and assuming that the code elements are independent (i.e. no redundancy between them) of each other, then the distribution of active units is binomial [9]. Hence, the fraction of active units per input would be

$$f_a = \frac{N(active)}{N} = \frac{Np \pm \sqrt{Np(1-p)}}{N} = p \pm \sqrt{\frac{p(1-p)}{N}} \tag{3.32}$$

i.e. population sparseness results, because $p \ll 1$ by assumption and the standard deviation vanishes as $N \to \infty$.

The same reasoning can be used to demonstrate that independence alone implies neither form of sparseness. Assume $p = 0.5$ and independence. Thus, the code is not sparse on a single-unit level. Furthermore, $f_a = 0.5 \pm \sqrt{\frac{0.5}{N}}$, i.e. no population sparseness with certainty as $N \to \infty$.

Population and single-unit sparseness together don't imply independence: assume a variable X could take on the values $\{1; 2; \ldots; M\}$ with equal probability p. A given instance of X is now encoded into a binary vector $\vec{Y} = (y_1; \ldots; y_M)$ in such a way that $y_m = 1$ for $X = m$ and all other components 0. Then, if M is large enough, the code will be sparse on a single-unit level, because $p = \frac{1}{M}$. It will also be population sparse, because exactly one unit is active for any given input. Because of this property, however, the code elements (i.e. the components of $\vec{Y}$) are not independent: the probability that two of them are active at the same time is 0, whereas $p^2 > 0$. It therefore also follows that population sparseness or single-unit sparseness alone are not sufficient to imply independence.

The aforementioned $X \to \vec{Y}$ coding scheme is overcomplete: $\vec{Y}$ could potentially represent 2^M different X, but only M are possible. Hence, overcompleteness doesn't necessarily promote independence. While one may be fairly confident that it will increase population sparseness, it might not help towards single-unit sparseness: imagine $\vec{Y}$ was extended so as to have $K > M$ components, with $\forall K \geq m > M : y_m = 0$. Then, population sparseness would have increased, but not single-unit sparseness, because the activation probability for the y_m would no longer be as evenly distributed as in the case $K = M$. This observation emphasizes the need for a good pruning scheme: from a sparseness perspective, there seems to be an optimal degree of overcompleteness, its (sensible) maximum is reached when additional units cannot increase the mutual information between input and output. Overcompleteness may be beneficial to independence under certain conditions: consider the teacher-student scenario from section 3.4.2, and assume the teacher network had only $N = 2$ outputs, but $M = 9$ units. If the student network was to learn an independent code for these outputs, it would likewise have to have $M = 9$ units, i.e. it would be overcomplete. I would therefore conclude that the question whether overcompleteness and independence together are sensible coding goals cannot be answered a priori, rather, the answer depends on the data.

3.6 Asymmetric noise: a sparseness-promoting factor

The above presented arguments (section 3.1) in favor of sparse coding focus on the noiseless case. In a real brain, however, this is an unlikely situation. Information transmission through and processing in biological neurons is fraught with uncertainties. The employed coding schemes should reflect this fact, if optimal use is to be made of the available hardware.

As detailed in section 2.5.3, mutual information is a measure for the quality of the code, i.e. $I(\vec{X}; O_1, \ldots, O_M)$ quantifies how much information the network's outputs contain about the input vector on average. If the code is factorial, and the units' noises are mutually independent given the input, then, by virtue of eqn. (2.26)

$$\begin{aligned} I(\vec{X}; O_1, \ldots, O_M) &= H(O_1, \ldots, O_M) - H(O_1, \ldots, O_M|\vec{X}) \\ &= \sum_{i=1}^{M} H(O_i) - \sum_{i=1}^{M} H(O_i|\vec{X}) \\ &= \sum_{i=1}^{M} I(O_i; \vec{X}) \end{aligned} \tag{3.33}$$

Hence, if one is looking for the optimal code under these conditions, then one can study one unit at a time. To keep things simple, only binary units are considered, i.e. they can transmit either 0 (inactivity) or 1 (firing). The encoding process can then be decomposed into two steps: first, a deterministic mapping from the input to the 'true' $\tilde{O}_i$ is performed. This mapping is assumed to be information-preserving, i.e. $\vec{X}$ can be reconstructed from the $\tilde{O}_i$. Second, the $\tilde{O}_i$ are distorted by noise to yield O_i (see fig. 3.11), resulting in some 0s being turned into 1s (with probability q) and vice versa (with probability p). The scenario of the second step is widely know as 'the binary channel with noise' in information theory [13].

The sparseness S will be measured by the average fraction of inactive units after the deterministic encoding step. Because the code is factorial

$$S = \frac{1}{M} \sum_{i=1}^{M} P(\tilde{O}_i = 0) = \frac{1}{M} \sum_{i=1}^{M} S_i \tag{3.34}$$

Now the optimal S_i for a given unit will be computed. By virtue of eqn. (3.33), the mutual information will be maximimzed by the same S_i for each unit: $\forall S_i : S = S_i$.

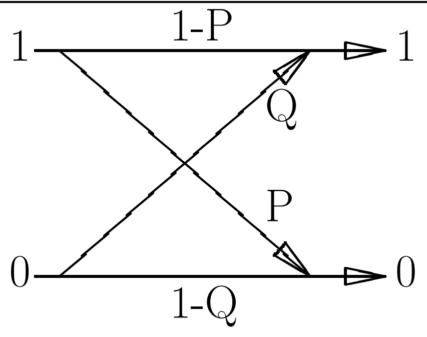


Figure 3.11: A binary channel with noise. Q is the probability of observing '1' when the input was '0', and vice versa for P. The noise is asymmetric if $Q \neq P$. In this case, sparse codes are optimal. For details, see text.

Noting that

$$P(O=0|\tilde{O}=0) = 1-Q \quad , \quad P(O=1|\tilde{O}=0) = Q \tag{3.35}$$

$$P(O=0|\tilde{O}=1) = P \quad , \quad P(O=1|\tilde{O}=1) = 1-P \tag{3.36}$$

$$P(\tilde{O}=0) = S \quad , \quad P(\tilde{O}=1) = 1-S \tag{3.37}$$

$$P(O{=}0) = S(1{-}Q) + (1{-}S)P \quad , \quad P(O{=}1) = SQ + (1{-}S)(1{-}P) \tag{3.38}$$

the mutual information between O and $\tilde{O}$ is given by

$$\begin{aligned} I(O;\tilde{O}) &= H(O) - H(O|\tilde{O}) \\ &= -\left(S(1-Q)+(1-S)P\right)\log\left(S(1-Q)+(1-S)P\right) \\ &\quad -\left(SQ+(1-S)(1-P)\right)\log\left(SQ+(1-S)(1-P)\right) \\ &\quad +S\left((1-Q)\log(1-Q)+Q\log(Q)\right) \\ &\quad +(1-S)\left(P\log(P)+(1-P)\log(1-P)\right) \end{aligned} \tag{3.39}$$

A necessary condition for an extremum w.r.t S is that the first derivative of $I(O;\tilde{O})$ vanishes:

$$\frac{\partial I(O;\tilde{O})}{\partial S} = H(P) - H(Q) - (1-(Q+P))\log\left(\frac{S(1-Q)+(1-S)P}{SQ+(1-S)(1-P)}\right) \overset{!}{=} 0 \tag{3.40}$$

where $H(P) = -P\log(P) - (1-P)\log(1-P)$ and likewise for $H(Q)$. Setting

$$C = \exp\left(\frac{H(P)-H(Q)}{1-(P+Q)}\right) \tag{3.41}$$

one finds that

$$S_m = \frac{C(1-P)-P}{(C+1)(1-(Q+P))} \tag{3.42}$$

is the location of the extremum. Since $I(O;\tilde{O}) \geq 0$ with equality if $S = 0$ or $S = 1$, the extremum must also be a maximum.

Fig. 3.12, top, shows the optimal sparseness as a function of P for four different values of Q. A code is sparse if $S_m > 0.5$, which is the case when $P > Q$. S_m reaches its maximum for $Q = 0$ and $P \rightarrow 1$, i.e. when firing is observed ($O = 1$), the unit has fired with certainty ($\tilde{O} = 1$), whereas inactivity is harder to detect. In that case, eqn. (3.41) becomes

$$C = \frac{\exp\left(\frac{-P}{1-P}\log(P)\right)}{1-P} \tag{3.43}$$

The value of the exponent in the limit $P \rightarrow 1$ can be determined by applying l'Hôpital's rule [9]:

$$\lim_{P\rightarrow 1}\frac{-P}{1-P}\log(P) = \lim_{P\rightarrow 1}\frac{-\log(P)-1}{-1} = 1 \tag{3.44}$$

Thus, C diverges and the limit of eqn. (3.42) is

$$\begin{aligned}\lim_{P\rightarrow 1} S_m &= \lim_{P\rightarrow 1}\left(\frac{C(1-P)}{(C+1)(1-P)} - \frac{P}{(C+1)(1-P)}\right) \\ &= 1 - \lim_{P\rightarrow 1}\frac{P}{\exp\left(\frac{-P}{1-P}\log(P)\right)+1-P} \\ &= 1 - \frac{1}{e} \approx 0.63212 \end{aligned} \tag{3.45}$$

Therefore, the maximum sparseness of a factorial code that can be motivated by noise alone is about 63%. While that is not nearly as much as the 84%-94% observed in the coding of natural images (see table 3.2), it still serves as an indication that noisy transmission in real neurons might be a sparsness promoting factor.

The mutual information $I(O,\tilde{O})$ is depicted in fig. 3.12, bottom. As one would expect, it decreases with increasing noise. For $q = 0$, the information gains (i.e. the Kullback divergences between $P(\tilde{O})$ and $P(\tilde{O}|O)$) on receiving '1' (dotted line) and '0' (dashed line) are plotted, too. As the code becomes sparser, progressively more information is conveyed through firing, and less by inactivity. This is a typical feature of sparse codes, which makes them appealing in neurophysiological terms: generating a neural spike requires energy. This energy is best spent when as much information as possible is transmitted in the process. Indeed, it has been demonstrated [3] that V1 and IT neurons exhibit firing rate distributions that maximize the information throughput for a fixed rate of energy expenditure.

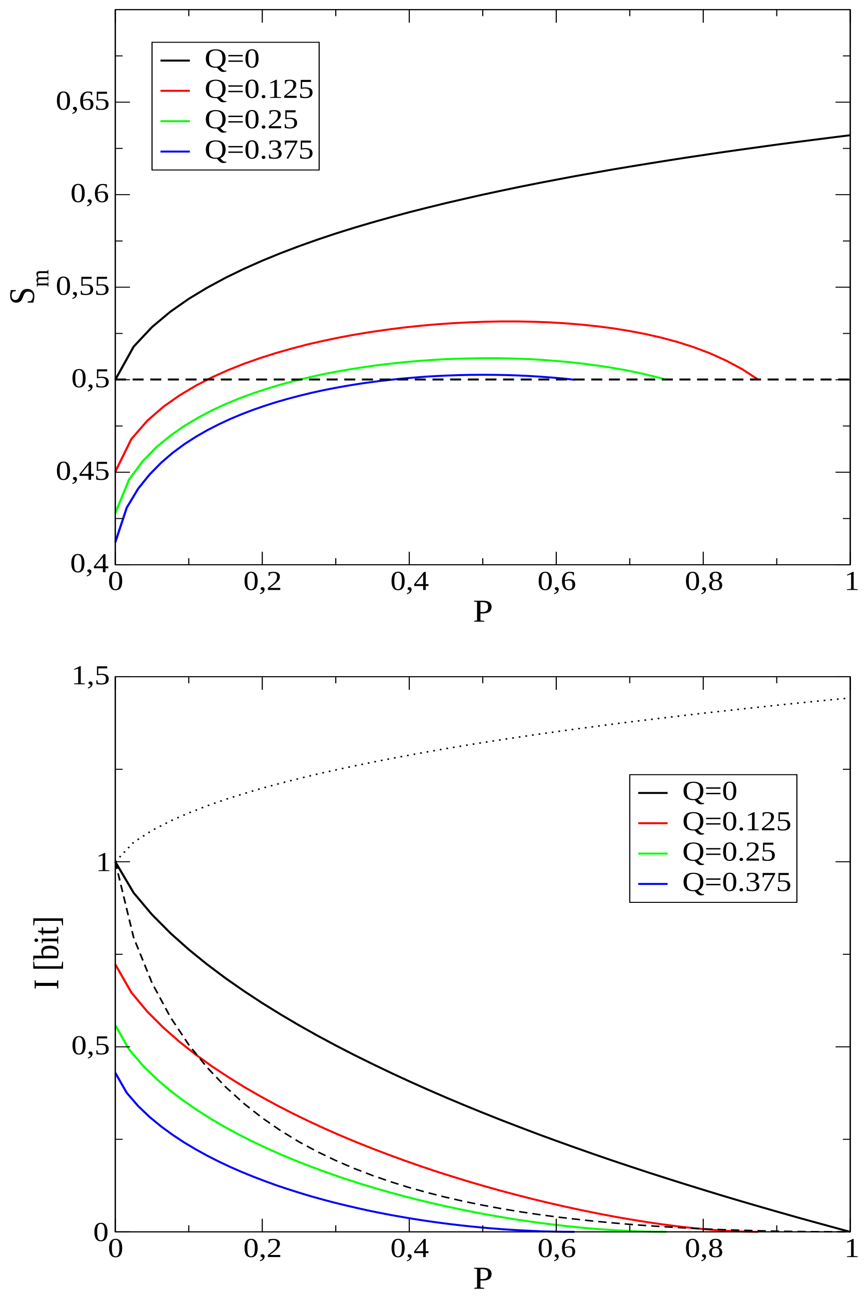


Figure 3.12: Top: optimal sparseness S_m as a function of P for four different values of Q (for the definition of P and Q, see fig. 3.11). The maximum $S = 1 - \frac{1}{e}$ is reached for $Q = 0$ and $P \rightarrow 1$ (i.e. when firing is observed, the unit has fired with certainty, whereas inactivity is harder to detect). Bottom: Transmitted information decreases with increasing noise level. For $Q = 0$, the information gains on observing '1' (dotted line) and '0' (dashed line) are plotted, too. When the code is sparse, the bulk of the information is transmitted through firing.

3.7 Conclusion

The above described network performs feature extraction on a scale defined by the dimensionality of the basis vectors (e.g. the size of the image patches). Since the learned RFs appear to model V1 RFs more closely with increasing overcompleteness, it would be interesting to investigate whether higher M:N ratios than the 4:1 studied here lead to even better agreement with experimental data – especially given the 25:1 ratios inferred from cat brain data [62]. However, this would only be feasible with significantly faster computational resources than those which I have currently at my disposal (on a 3GHz Pentium 4 system, the 4x overcomplete model took ≈ 4 days to learn).

In contrast to the network studied here, the visual cortex consists of more than one layer of neurons. It has been observed in [66] that stacking two OF networks with the same number of basis vectors on top of each other results in the top one learning an identity mapping between inputs and outputs. Something similar could be expected for my network. To integrate features on a larger scale, the top network would at least have to combine the outputs from several image patches of the lower network. More importantly, the question how the outputs are combined needs to be addressed. In [41], it was demonstrated that a square-and-add nonlinearity, inspired by a model of V1 complex cells proposed in [39], leads to top layer units that show some phase and position invariance in their RFs. Alternatively, one could also constrain the outputs of the bottom layer to positive values only, which is neurophysiologically plausible, since real neurons can not produce negative signals. In that case, one might expect the bottom layer to develop antagonistic pairs of feature detectors. The top layer could then simply marginalize over suitably chosen groups of bottom-layer outputs to generate responses that are invariant under certain transformations. Another interesting way of incorporating architectural constraints in a multilayer network model of the early visual system was explored in [90]: the optic nerve of many mammals has significantly less fibers than V1 has cells. Combining this observation with synaptic and firing rate energy constraints was shown to give rise to response characteristics of the simulated retina and V1 units which resemble those found in neurophysiological data. However, getting hierarchical networks to learn is not easy, if not generally infeasible. Thus, one will have to resort to approximations, e.g. the Helmholtz machine [18].

Given the fact that asymmetric noise is a sparseness-promoting factor, it might be interesting to investigate whether this asymmetry is present in biological neurons and how much sparseness one would expect from it. There is reason to believe that that might be so: assuming that a neuron's firing behavior can to some degree be described as a Poisson process (i.e. the probability of spike generation is constant per unit time, with the probability being a monotonically increasing function of the neuron's activation), and that another neuron receiving these spikes merely counts how many of them arrive in a given time window, then, no matter how high the firing probability p_{fire}, there is always a chance that the receiver gets no input. On the other hand, if $p_{fire} = 0$, then no spike will be generated with certainty. In other words, the situation in real brains might resemble that of the limit studied in section 3.6.

Chapter 4

Information extraction from neural spike trains I: Bayesian Bin Classification

4.1 Introduction

In the following two chapters, two Bayesian methods will be developed for information extraction from small and/or noisy datasets. As an example application, they will be employed to extract features from neural spike trains and to quantify the information contained therein. The experimental setup is schematically depicted in fig. 4.1, for a detailed description see [47, 48]: a monkey is presented with a visual stimulus, labelled y, which evokes a neural response that is recorded in the form of a spike train, i.e. a temporally ordered sequence of time indexes. Each time index marks the occurrence of a spike. This spiketrain is subjected to a function $f()$ which condenses it into a quantity x that contains as much information as possible about y. x will be used in three ways:

1. The Bayesian Bin Classification algorithm (BBCa), which is the subject of this chapter, can be used to compute $P(y|x)$, i.e. the most probable y given x can be determined. This kind of classification task must also be performed in some way by the brain, when visual object recognition is carried out.

2. If y is known, the function $f()$ can be inferred. More precisely, assume that $f() = f(x, \vec{\theta})$, where $\vec{\theta}$ is a vector of parameters whose values determine the

shape of $f()$. For example, if $f()$ counted the number of spikes in a temporal window, $\vec{\theta}$ would contain the start and end positions of this window. Given experimental data, the BBCa can then be used to compute the posterior distribution of $\vec{\theta}$. It would then be possible to pick the best $\vec{\theta}$ in a maximum-a-posteriori sense. However, one can also evaluate various expectations under the posterior, such as means and variances of the compontents of $\vec{\theta}$. This will be done so as to provide not only the expected $f()$, but also a measure of its reliability. Knowing the posterior of $\vec{\theta}$ (and thus, of $f()$) enables one to draw conclusions about how the brain encodes stimulus-related information. The necessary 'feedback signal' (see fig. 4.1) is provided by the BBCa as well.

3. Given y and x from the inferred $f()$, the mutual information $I(x;y)$ can be inferred, i.e. the amount of information a neuron transmits about a stimulus. This will be done via the Bayesian Bin Distribution Inference (BBDIa) algorithm, subject of the next chapter.

One might wonder why two separate algorithms are necessary. Wouldn't it be sufficient to infer the mutual information for a given $f()$ and then search for the function that maximizes $I(x;y)$? The answer is no. As explained in section 2.3.1, only the formalism of probability theory is suitable for conducting (Bayesian) inference (or any formalism isomorphic to it). Since mutual information is not a probability, it is thus ruled out for the purpose of inferring the posterior distribution of $f()$. To do that, the possible choices for $f()$ need to be weighted by a probability. Moreover, this probability needs to be a measure of classification performance if we are to learn which $f()$s are suited for carrying out the classification task and which ones are not. The most natural choice is thus $P(y|x)$, which will be high if, for given x and y, correct classification can be done with some certainty.

It might be argued that, since $P(y)$ is determined by the experimental setup, one could also try to infer $p(x|y)$ instead and convert it via Bayes' rule into $P(y|x)$. However, this approach is likely to meet with computational difficulties: $p(x|y)$ will usually be parameterized in some fashion. After the posterior distributions of these parameters have been inferred from the data, the marginalizations necessary for the determination of the posterior distribution of $f()$ are, in most cases, going to be intractable. Thus, two methods will be presented: BBCa, which allows for an exact

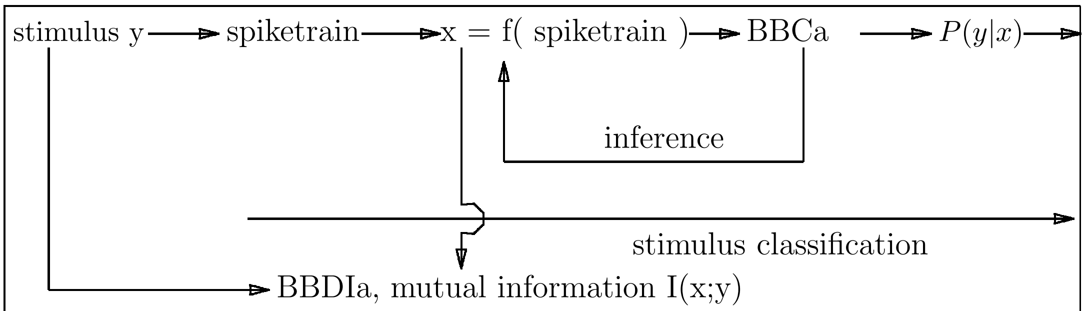


Figure 4.1: Schematic representation of the RSVP (rapid serial visual presentation) experiment and its evaluation. A monkey is presented with a visual stimulus y which evokes a neural response that is recorded in the form of a spike train. This spiketrain is subjected to a function $f()$ which condenses it into a quantity x that contains as much infomation as possible about y. The BBCa then allows for the computation of $P(y|x)$ and thus, for the determination of the most probable y. Conversely, if y is known, $f()$ can be inferred and subsequently the mutual information $I(x;y)$, too.

evaluation of the posterior distribution of $f()$, and BBDIa, which avails one of an exact Bayesian estimate of the mutual information.

Nevertheless, mutual information is an infomation-theoretic measure of average classification performance (see section 2.5.3). Thus, one would expect that probabilistic classification measures should be closely related to it. That this is indeed so will be demonstrated in the next chapter.

4.2 Bayesian classification

Bayesian classification is a widely used method in many fields of scientific inference and engineering. Suppose one wanted to determine whether an object O_k belonged to any one of C classes. To do so, a vector of features $\vec{w}_k$ is observed (which is in the following assumed to be a vector of real numbers), which is hoped to contain the information necessary to assign a class label y_k to the object in question. Usually, due to noise in the measuring process or incompleteness of the available information, this cannot be done with certainty. Hence, one tries to estimate the probability

$P(y_k|\vec{w}_k)$ that the object belongs to class $y_k \in \{1, \ldots, C\}$. A common method – Bayesian classification – estimates the class-conditional densities $p(\vec{w}|y, \vec{\theta}_y)$ ($\vec{\theta}_y$ are the parameters of the density model for class y) and then uses Bayes' theorem to predict

$$P(y|\vec{w}, \vec{\theta}_1, \ldots, \vec{\theta}_C) = \frac{p(\vec{w}|y, \vec{\theta}_y)P(y)}{\sum_{y'} p(\vec{w}|y', \vec{\theta}_{y'})P(y')} \tag{4.1}$$

where $P(y)$ is the probability of O_k belonging to class y prior to observing $\vec{w}$.

The correct way – from a Bayesian perspective – to do away with the dependency of the l.h.s. on the $\vec{\theta}_y$, is to integrate them out of the prediction:

$$P(y|\vec{w}, D) = \int_{\vec{\theta}_1} d\vec{\theta}_1 \ldots \int_{\vec{\theta}_C} d\vec{\theta}_C P(y|\vec{w}, \vec{\theta}_1, \ldots, \vec{\theta}_C)p(\vec{\theta}_1, \ldots, \vec{\theta}_C|D) \tag{4.2}$$

$p(\vec{\theta}_1, \ldots, \vec{\theta}_C|D)$ is the probability density of the $\vec{\theta}_y$ given previously observed data D (comprised of pairs $(\vec{w}_k, y_k)$) and any other available prior information regarding them (the implicit dependency on the model class is omitted here and in the following).

Performing integrals of this type is generally very difficult. Therefore, a variety of approximation methods for their evaluation have been derived in the past (see section 2.6 for a brief overview). In the following, it will be demonstrated how the problem can be circumvented (at least in part) by directly inferring $P(y|\vec{w})$ from the data.

4.3 A simple model for $P(y|\vec{w})$

Assume one had a multiset of K labeled feature vectors $D = \{(\vec{w}_k, y_k)\}$. The classification task can then be decomposed into two steps:

1. Find a suitable mapping $f(\vec{w}) \mapsto x$, $x \in [0, 1]$, such that x contains all the information from $\vec{w}$ that pertains to the classification,

2. Infer the probabilities $P(y|x, D_f)$, where $D_f = \{(x_k, y_k)\}$ are the data after the $\vec{w}_k$ have been mapped onto the x_k through $f(.)$.

While step 1 is by no means trivial, the following arguments will focus mostly on step 2. It should be noted, however, that step 1 is always possible (to any given degree of accuracy)[1], because the number of classes C is assumed to be finite. Ways of finding $f(.)$ are discussed in section 4.11.1.

4.3.1 Why do the mapping $f(\vec{w}) \mapsto x$ first?

The mappings $f(\vec{w}) \mapsto x$ and subsequently from x to y might always be possible, but is it also desirable to take this route? To answer this question, let's take a look at decision making in the presence of uncertainty. At the current stage of development of exact Bayesian Inference techniques it is (to the author's knowledge) rather unlikely to find a model for $P(y|\vec{w}, \vec{\theta}_1, \ldots, \vec{\theta}_C)$ that is both of sufficient generality to be applicable to a variety of practical inference problems and integrable in a closed form. Thus, one would employ one of the above mentioned approximation techniques. They all require – to some degree or another – to pick some (or one, in the cases of MAP and ML) models included in the model class that is considered and discard the rest. By doing so, one 'contrives' information that one doesn't really have (namely that certain models can be excluded), its amount is given by the Kullback divergence between the posterior density $p(\{\vec{\theta}_y\}|D) = p(\vec{\theta}_1, \ldots, \vec{\theta}_C|D)$ and the modified posterior density after making the decision $p(\{\vec{\theta}_y\}|C)$. This difference diverges in the cases of MAP and ML, which might offer an explanation as to why these methods often perform so poorly, especially when only a very small dataset is available.

Now assume $\vec{\theta}_y$ was split in two parts: $\vec{\theta}_y = (\vec{\theta}_y^E, \vec{\theta}_y^A)$. The first part contains all those degrees of freedom for which the integration can be carried out directly. To tackle the second part, an approximation technique is being used. Correspondingly,

[1]The most general information which $\vec{w}$ can contain about y is given by the probability distribution $P(y|\vec{w})$. For $C = 2$, choose $f(\vec{w}) \mapsto x$ such that $P(y = 1|x) = x$. For $C = 3$, the possible probability distributions can be represented by points inside a 2 dimensional simplex (i.e. a triangle). If the desired accuracy is ϵ, then cover the triangle by smaller triangles of width and height $< \epsilon$, enumerate those small triangles in some fashion (e.g. such that the probabilities change as smoothly as possible between successive triangles) and map this enumeration in an order-preserving way onto x. Each possible value of x then represents a probability distribution. The generalization of this procedure to any finite C is straightforward. While it may not be the best way of performing $f(\vec{w}) \mapsto x$, it serves as an indication that this mapping is always possible for a given ϵ.

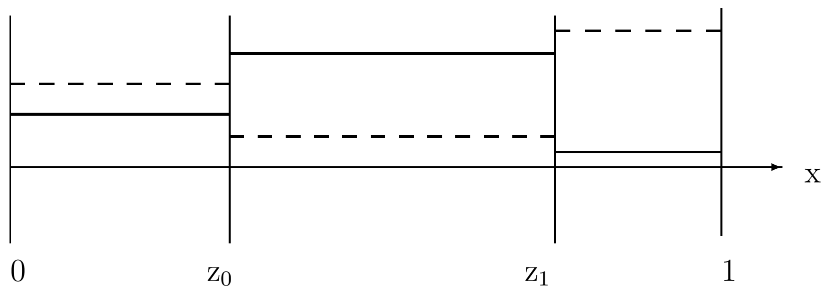


Figure 4.2: An example model. The interval between 0 and 1 is divided into three bins at points z_0 and z_1. Within each bin $j \in \{0, 1, 2\}$, x is mapped onto $P(y|x) = c_y^j$, which are constant within the bin. Two classes are possible, their probabilities are indicated by the solid and dashed lines.

decompose the posterior densities as

$$p(\{\vec{\theta}_y\}|D) = p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, D)p(\{\vec{\theta}_y^A\}|D) \tag{4.3}$$

$$p(\{\vec{\theta}_y\}|C) = p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, C)p(\{\vec{\theta}_y^A\}|C) \tag{4.4}$$

Then the Kullback divergence can be written as

$$\begin{aligned} D_{DC} &= \int d\vec{\theta}_y^A p(\{\vec{\theta}_y^A\}|C) \int d\vec{\theta}_y^E p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, C) \log\left(\frac{p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, C)p(\{\vec{\theta}_y^A\}|C)}{p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, D)p(\{\vec{\theta}_y^A\}|D)}\right) \\ &= E\left[D(p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, C)||p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, D)\right] + D(p(\{\vec{\theta}_y^A\}|C)||p(\{\vec{\theta}_y^A\}|D)) \\ &= D_E + D_A \end{aligned} \tag{4.5}$$

where the expectation for D_E in the second line is carried out w.r.t. $p(\{\vec{\theta}_y^A\}|C)$. If the first part of the decomposed posterior is treated exactly, then $D_E = 0$, because $p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, D) = p(\{\vec{\theta}_y^E\}|\{\vec{\theta}_y^A\}, C)$. Otherwise, $D_E > 0$, because it is the expectation of a Kullback divergence. It follows that one should perform exact integrations if possible, in order to keep the errors which one introduces through making approximations at a minimum. Furthermore, in the decomposition suggested here, $f()$ can likely be modeled in a simpler manner, because it no longer needs to deal with the mapping onto the class labels. As noted above, this mapping is always possible.

4.3.2 A bin model for $P(y|x)$

To carry out step 2, the following model will be employed:

Divide the interval $[0,1]$ by M points z_j, so as to get $M+1$ bins (see fig. 4.2). Within each bin, the probability $P(y|x)$ is assumed to be constant w.r.t. x. There are C such probabilities per bin, denoted by $c_y^j = P(y|x \in (z_{j-1}, z_j])$, where y is the class and j is the bin (bin j is the interval between z_j and z_{j-1}, bin 0 is the interval between 0 and z_0, hence bin M is the interval between z_{M-1} and 1). This is not as restrictive as it may seem at first, since it is possible to approximate any continuous function with arbitrary accuracy by a piecewise constant function, given that the number of bins is not limited: Define $f_n(x) \mapsto f(\frac{\xi}{n})$, where ξ is the greatest integer smaller than $x \cdot n$. Since f is continuous, $\lim_{n\to\infty} f_n(x) = f(x)$.

The c_y^j are normalized with respect to the classes within their bin, i.e.

$$\forall j \in \{0, \ldots, M\} : \sum_y c_y^j = 1$$

$\{c_y^j\}$ denotes the set of these x-conditional probabilities in bin j.

For a given configuration of M,$\{z_j\}$ and $\{\{c_y^j\}\}$, the probability of the dataset then becomes (assuming that the data points have been drawn independently of each other):

$$P(D|\{\{c_y^j\}\}, \{z_j\}, M) = \prod_k c_{y_k}^{j_{x_k}}; \tag{4.6}$$

where j_{x_k} is the bin that contains $x_k = f(\vec{w}_k)$. Once observed, one can easily reorder the data points so that $x_k < x_{k+1}$, i.e. the data is ordered according to x_k, which will be assumed from here on.

Classifying a new feature vector $\vec{w}'$ involves evaluating

$$\begin{aligned} P(y'|x', D_f) &= \frac{P(D'_f)}{P(D_f)} \\ &= \frac{\sum_M P(D'_f|M)P(M)}{\sum_M P(D_f|M)P(M)} \end{aligned} \tag{4.7}$$

where D'_f is the data(multi)set with $(x' = f(\vec{w}'), y')$ added to it, i.e. $D'_f = D_f \cup \{(x', y')\}$, and $P(M)$ is the prior probability for a model with M bin boundaries in $[0,1]$. The sums run over all values of M which one chooses to include into the calculation. This choice could be made (in the case of uniform $P(M)$) by selecting those M that have a high enough evidence $P(D_f|M)$ to contribute significantly to (4.7). As y' is the quantity to be determined, one has to evaluate (4.7) for all possible values of y' and then pick the one with the highest probability, to minimize the chance of misclassification.

Thus, one needs to compute $P(D_f|M)$, i.e. the evidence for a model with M bin boundaries:

$$\begin{aligned} P(D_f|M) &= \int d\{z_j\} \int d\{\{c_y^j\}\}\; p(D_f, \{\{c_y^j\}\}, \{z_j\}|M) \\ &= \int d\{z_j\} \int d\{\{c_y^j\}\}\; P(D_f|\{\{c_y^j\}\}, \{z_j\}, M) p(\{\{c_y^j\}\}|\{z_j\}, M) p(\{z_j\}|M) \end{aligned} \tag{4.8}$$

where

$$\int d\{z_j\} = \int_0^1 dz_{M-1} \int_0^{z_{M-1}} dz_{M-2} \ldots \int_0^{z_1} dz_0 \tag{4.9}$$

$$\int d\{c_y^j\} = \int_0^1 dc_0^j \int_0^1 dc_1^j \ldots \int_0^1 dc_{C-1}^j \delta(1 - \sum_{y=0}^{C-1} c_y^j) \tag{4.10}$$

$$\int d\{\{c_y^j\}\} = \int d\{c_y^0\} \int d\{c_y^1\} \ldots \int d\{c_y^M\} \tag{4.11}$$

The way the integration boundaries are chosen in (4.9) ensures that the ordering of the z_j is maintained, and the Dirac-delta function in (4.10) enforces normalization of the probabilities in bin j.

It is also possible, in a similar fashion, to determine confidence intervals on the predicted probabilities, via

$$\mathrm{Var}\left[P(y'|x', D_f, M)\right] = \frac{P(D_f''|M)}{P(D_f|M)} - \left(\frac{P(D_f'|M)}{P(D_f|M)}\right)^2 \tag{4.12}$$

where D_f'' is the data(multi)set with the point (x', y') added twice[2] .

Since $p(\{\{c_y^j\}\}|\{z_j\}, M)$ and $p(\{z_j\}|M)$ do not depend on the data D_f, they are prior densities which will be assigned in the following way:

- $p(\{z_j\}|M)$ depends on M only insofar as $0 \leq j < M$. Furthermore, it will be assumed that $z_j \leq z_{j+1}$, i.e. the z_j are ordered, but otherwise no preferences for their locations in the interval $[0, 1]$ will be expressed in the prior. It will also be assumed that, apart from the ordering, they are independent of each other.

[2]$\mathrm{Var}\left[(P(y'|x', D_f, M))\right] = E\left[P(y'|x')^2\right] - E\left[P(y'|x')\right]^2$, where the expectations are w.r.t the posterior of $P(y'|x')$. This posterior is given by the integrand of (4.8) divided by (4.8). To compute the expectation of $P(y'|x')$, multiply the posterior by $c_{y'}^{j_{x'}}$, where $j_{x'}$ is the bin containing x' and integrate. By virtue of (4.6), this is computationally equivalent to adding the point (x', y') to the data(multi)set. Likewise, to compute the expectation of the square of $P(y'|x')$, add this point twice, thus yielding the expectation of $\left(c_{y'}^{j_{x'}}\right)^2$.

The prior thus becomes $p(\{z_j\}|M) = \prod_j p(z_j)$, where $p(z_j) = p_z = const > 0$ if $z_j \leq z_{j+1}$ and 0 otherwise.

- $p(\{\{c_y^j\}\}|\{z_j\}, M) = \prod_j \prod_n p(c_y^j)$, i.e. the prior densities of the x-conditional probabilities in bin j are independent of the locations of the bin boundaries z_j and independent of each other, except for the constraint that the c_y^j be normalized w.r.t. the classes. Hence, it will be assumed that the $p(c_y^j) = p_c$ are constant and equal (subject to the normalization constraint).

4.4 Computing $p(\{\{c_y^j\}\}|\{z_j\}, M)$

The prior $p(\{\{c_y^j\}\}|\{z_j\}, M)$ has to be normalized w.r.t. $\{\{c_y^j\}\}$, i.e.

$$\int d\{\{c_y^j\}\} p(\{\{c_y^j\}\}|\{z_j\}, M) = 1 \tag{4.13}$$

Since the sets $\{c_y^j\}$ are assumed to be independent of each other, the integration over each set can be performed independently of the others. In the following, the superscript j (which denotes the bin) will therefore be dropped. As $0 \leq c_y \leq 1$, one needs to compute integrals of the form

$$\begin{aligned} I(l_0, \ldots, l_{C-1}) &= \int_0^1 dc_0 (c_0)^{l_0} \ldots \int_0^1 dc_{C-1} (c_{C-1})^{l_{C-1}} \times \\ &\times \; p_c^C \delta(1 - \sum_{y=0}^{C} c_y) \end{aligned} \tag{4.14}$$

where l_c means the number of data points which belong to class c, and the δ-function ensures that the set of c_y are always a probability distribution. This yields the well known result for the normalization constant of a Dirichlet density (see e.g. [40] or appendix D)

$$I(l_0, \ldots, l_{C-1}) = p_c^C \frac{\prod_y l_y!}{(\sum_y l_y + C - 1)!} \tag{4.15}$$

The integral over the prior then becomes:

$$\int d\{\{c_y^j\}\} p(\{\{c_y^j\}\}|\{z_j\}, M) = (I(0, \ldots, 0) p_c^C)^{M+1} \tag{4.16}$$

Hence, $p_c = (C-1)!^{\frac{1}{C}}$ and the prior is

$$p(\{\{c_y^j\}\}|\{z_j\}, M) = (C-1)!^{M+1} \tag{4.17}$$

4.5 Computing $p(\{z_j\}|M)$

This prior is subject to the normalization constraint

$$\int d\{z_j\}p(\{z_j\}|M) = 1 \tag{4.18}$$

Since $p(\{z_j\}|M)$ is assumed to be constant,

$$p(\{z_j\}|M) = M! \tag{4.19}$$

4.6 Computing the evidence

Now the evaluation of (4.8) can be continued. Assume that one chose a particular binning of the interval $[0, 1]$, represented by the set $\{z_j\}$. It then becomes possible to carry out the integrations over the $\{\{c_y^j\}\}$:

$$p(D_f|\{z_j\}, M) = \int d\{\{c_y^j\}\} \underbrace{P(D_f|\{\{c_y^j\}\}, \{z_j\}, M)}_{eqn.(4.6)} \underbrace{p(\{\{c_y^j\}\}|\{z_j\}, M)}_{eqn.(4.17)} \tag{4.20}$$

Due to the assumed form of the prior and of the model, this probability is a product, consisting of one factor for each bin. Each factor is of the same form as (4.15), the exponents of each c_n being the number of data points (in the following denoted by l_n^j) belonging to class n in the bin j, i.e. the posterior of the c_n is a Dirichlet density . Hence one finds

$$P(D_f|\{z_j\}, M) = \prod_{j=0}^{M} \underbrace{(C-1)!I(l_0^j, \ldots, l_{C-1}^j)}_{I_{k_1,k_2}} \tag{4.21}$$

where $x_{k_1}, ..., x_{k_2-1}$ are the data points in bin j.

What remains is the integration over all possible configurations of bins:

$$P(D_f|M) = \int d\{z_j\}P(D_f|\{z_j\}, M) \underbrace{p(\{z_j\}|M)}_{eqn.(4.19)} \tag{4.22}$$

4.6.1 Integrating over $\{z_j\}$

Note that the integrand of (4.22) is constant as long as the z_j move between data points. Only when a bin boundary crosses over a data point does its value change.

Hence, this integral can be written as a sum. Each summand is the product of the integrand and the total volume occupied by the z_j for that value of the integrand.

Let

$$\hat{x}_{k,m} := \frac{(x_k - x_{k-1})^m}{m!} \tag{4.23}$$

with $x_{-1} = 0$ and $x_K = 1$. If the z_j are distributed in such a way that at most one z_j lies between two adjacent data points, then the total volume occupied by this configuration is the product of the differences between the two data points enclosing z_j, i.e. the product of the corresponding $\hat{x}_{k,1}$. In case there are several of the z_j between the same two adjacent data points, the contribution to the volume will be $\hat{x}_{k,m}$, where m is the number of z_j found in this interval. This follows directly from the ordering constraint imposed upon the z_j.

Now assume $M = 1$. With $\mathcal{C}_{C,M} := (C-1)!^{M+1}M!$, (4.22) then becomes:

$$P(D_f|M=1) = \mathcal{C}_{C,1} \cdot \sum_{k=0}^{K} I_{0,k} \underbrace{\hat{x}_{k,1} I_{k,K}}_{\hat{E}_{k,1}} \tag{4.24}$$

For $M = 2$, it would be:

$$\begin{aligned} P(D_f|M=2) &= \mathcal{C}_{C,2} \cdot \sum_{k=0}^{K} I_{0,k} \underbrace{\hat{x}_{k,2} I_{k,K}}_{\hat{E}_{k,2}} \\ &+ \mathcal{C}_{C,2} \cdot \sum_{k=0}^{K-1} I_{0,k} \underbrace{\hat{x}_{k,1} \sum_{\kappa=k+1}^{K} I_{k,\kappa} \underbrace{I_{\kappa,K} \hat{x}_{\kappa,1}}_{\hat{E}_{\kappa,1}}}_{\hat{E}^{new}_{k,1} = \alpha \hat{x}_{k,1}} \end{aligned} \tag{4.25}$$

$\hat{E}_{k,m}$ is the sub-evidence of a sub-model with m bin boundaries (not counting those at 0 and 1) which includes only the points $x_i \geq x_{k-1}$, and α is the sum over all sub-models from the previous M which include only points to the right of x_k. Therefore, when $M = 2$, the evidence for $M = 1$ can be evaluated as well with little extra computational overhead. Furthermore, while computing $P(D_f|M = 1)$, one also evaluates the first part of $P(D_f|M = 2)$, i.e. those configurations where both z_j lie between the same two data points. Then the array $\hat{E}_{k,1}$ can be reused for the computation of the second part, where the z_j are between different pairs of data points. One can proceed in this fashion until the desired M is reached (see fig. 4.3). More generally: when evaluating the contribution of a sub-model which has m_0 bin boundaries between x_k and x_{k+1}, and m bin boundaries to the right of x_{k+1}, then

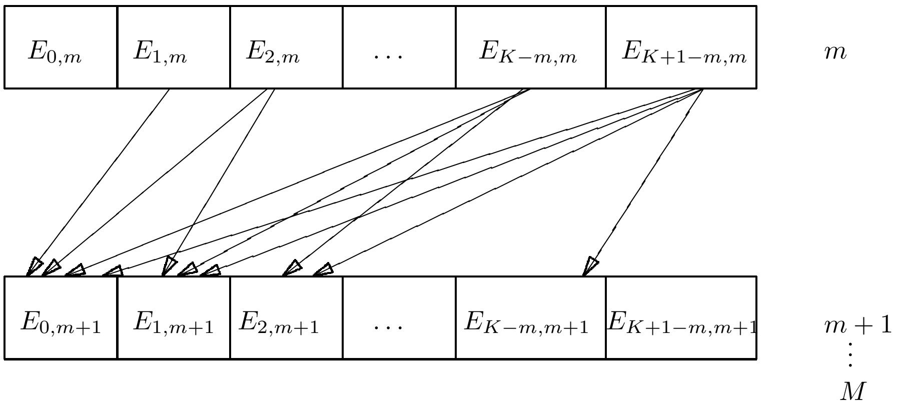


Figure 4.3: When evaluating the evidence contribution of sub-models with $m+1$ bin boundaries, one of which is found between x_k and x_{k+1}, then all contributions of models with more than one bin boundary between x_k and x_{k+1} can be evaluated reusing α (see text). The arrows indicate which sub-evidences sum up to a sub-evidence for more than m bin boundaries.

one can reuse α to compute the contributions of all sub-models which have $m_1 > m_0$ bin boundaries between x_k and x_{k+1}, because they differ only by a factor $\hat{x}_{k,m_0}$ (for m_0 boundaries) versus $\hat{x}_{k,m_1}$ (for m_1 boundaries).

Should $M > K$, then it is not possible to construct submodels which have at least one data point between adjacent z_j. Thus, the iteration over M can be stopped earlier. In pseudo-code:

```
1. For $k := 0$ to $K$, $m := 1$ to $M$: compute $\hat{E}_{k,m} := I_{k,K}\hat{x}_{k,m}$

2. For $m := 1$ to $M$: initialize $E_m := \sum_{k=0}^{K} I_{0,k}\hat{E}_{k,m}$

3. For $m := 2$ to $\min(M, K+1)$, $k := 0$ to $K+1-m$:

   (a) For $\mu := m$ to $M$: reset $\hat{E}_{k,\mu} := 0$

   (b) For $\mu := m$ to $M$

       i. Compute $\alpha := \sum_{\kappa=k+1}^{K+2-m} \hat{E}_{\kappa,\mu-1} I_{k,\kappa}$

       ii. For $\nu := \mu$ to $M$: add $\alpha\hat{x}_{k,\nu-\mu+1}$ to $\hat{E}_{k,\nu}$

   (c) For $\mu := m$ to $M$: add $I_{0,k}\hat{E}_{k,\mu}$ to $E_\mu$

4. return $E_m$
```

This yields $P(D_f|m) = E_m \cdot \mathcal{C}_{C,m}$ for all $1 \leq m \leq M$. A close look at step 3(b)i reveals that the computational complexity is $O(K^2M^2)$, because one expects $M < K$ for real world applications, or even $M \ll K$.

This way of organizing the calculation is computationally similar to the sum-product algorithm for factor graphs [49]. The 'messages' passed on from one m-level to the next are $\hat{E}_{\kappa,\mu-1}$, whereas within one level, a sum over all the 'messages' from the previous level is performed.

4.7 Comparison to other classification methods

The performances of two other classification methods, support vector machines (SVM) and Gaussian process classification (GPC), were compared to that of the BBCa. SVMs [88, 89] have enjoyed great popularity due to their successes in many applications, e.g. handwritten digit recognition [12] or 3D object recognition [8]. They operate by mapping the feature vectors $\vec{w}$ (see section 4.2) into a higher (or infinite) dimensional space and then finding a hyperplane in that space that separates all mapped $\vec{w}$s belonging to one class from the rest. There is usually more than one such hyperplane, in which case the one with the maximum margin (i.e. the distance to the nearest points on either side) is selected. Classification of previously unseen data is performed by determining on which side of the hyperplane the mapped $\vec{w}$ is found. For the comparison presented here, *libsvm 2.8*[3] was used. This package also includes a python script which performs data scaling and model selection for a C-SVM using a radial basis function kernel (for details, see [38]).

Gaussian processes were originally designed for regression problems [54]. In brief, the idea is to define a Gaussian process prior over a function space such that the joint density of any number of points drawn from the functions in this space is a multivariate Gaussian. If the noise model (i.e. the probability density of the observed data given the values produced by the Gaussian process) is also Gaussian, then the predictions for new data points can be evaluated analytically in polynomial time. This is often the case in regression tasks. In classification problems, however, the predictions have to be probabilities of class lables, which necessitates the use of a 'squashing function' that maps $[-\infty, \infty]$ onto $[0, 1]$. A popular choice for two-

[3]available at *http://www.csie.ntu.edu.tw/˜cjlin/libsvm/*

class scenarios is the logistic sigmoid $\frac{1}{1+\exp(-x)}$, where x is the value predicted by the Gaussian process. Multi-class problems can e.g. be tackled by the softmax function [92]. Unfortunately, the required integrations can now no longer be carried out analytically, and thus approximations must be employed. For the comparsion presented here, the Monte-Carlo approach implemented in the package *fbm-2004-11-10*[4][58] was used.

Training and test data were generated by simulating a neuron's response to eight stimuli. The 'strongest' stimulus evoked 60 spikes/s, the 'second strongest' 40 spikes/s, the 'weakest' 15 spikes/s and the remaining stimuli evoked 30 spikes/s. Responses were recorded over a time window of 100 ms, during which the firing rate did not change. The resulting average firing rate was used as the input to the three algorithms. The classification target was the stimulus label. All stimuli were presented equally often. This rather simple scenario was chosen so as to allow for an *a-priori* determination of the expected classification performance limits.

The BBCa was trained with a maximum number of 50 intersections. Both a uniform prior over the number of intersections M and a prior $\propto \frac{1}{M^2}$ (i.e. the prior probability for a model is inversely proportional to the computational effort required to evaluate it) were tried, they produced very similar results. For the GPC, prior (hyper)parameters for the covariance function need to be chosen. I experimented with various settings and eventually chose a covariance function comprised of a linear part and a squared-exponential part. The linear part was given by a gaussian prior with standard deviation 10 and mean 0. The scale and relevance parameters of the exponential part were Gaussian with mean 0 and variances drawn from broad inverse-gamma distributions with mean 20 and 5, respectively. For details of the possible prior choices, the reader is referred to the extensive documentation of the *fbm-2004-11-10* package. Changing these prior parameters within 2 orders of magnitude did not affect the classification performance in any substantial way. As noted above, the SVM package contained python scripts to perform parameter selection via cross-validation in an automated manner.

The average percentage of correctly classified stimuli as a function of the trials per stimulus (i.e. the number of times a response to a given stimulus appeared in

[4]available at *http://www.gaussianprocess.org*

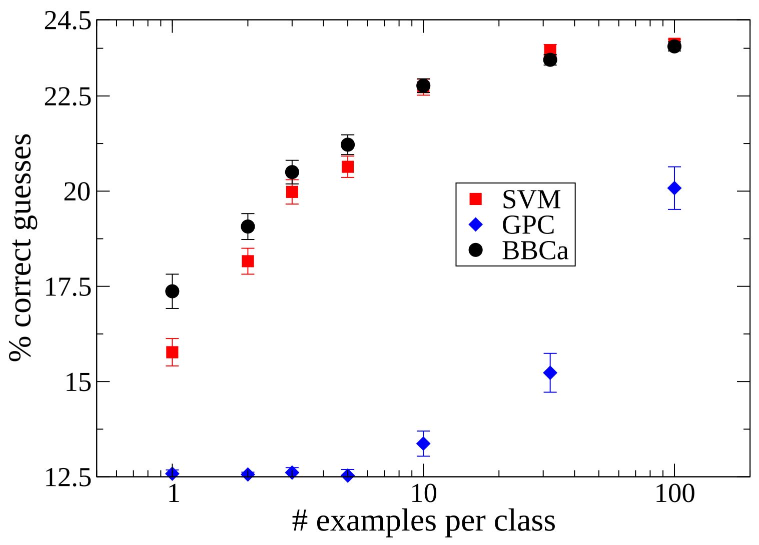


Figure 4.4: Comparison of the percentages of correctly classified stimuli as a function of the number of trials per stimulus. Black circles: BBCa, red squares: SVM, blue diamonds: GPC. Error bars are standard errors computed from 100 repetitions. The theoretical performance limits are 12.5 % (uninformed guess based only on prior knowledge of the stimulus distribution) and ≈ 24.5% (best possible expected performance if the response generating distributions were known). Especially in the neurophysiologically relevant range of only a few available trials per stimulus, the BBCa outperforms SVM and GPC. For details, see text.

the training set) is shown in fig. 4.4. For a given number of trials per stimulus, each classifier was first trained on a training data set, then its performance was evaluated on a test data set containing 100 trials per stimulus. This procedure was repeated on 100 different training/test data sets to allow for an evaluation of means and standard errors. Since all 8 stimuli were equally likely, one expects a performance of 12.5 % based on this information alone. If the response-generating distributions were known, the optimal performance as predicted by Bayes' rule would be ≈ 24.5%. All three methods seem to converge towards this value, even though GPC is doing notably worse than the other two. For 1000 trials per stimulus, BBCa and SVM have virtually reached the theoretical optimum (not shown). However, especially in the neurophysiologically relevant range of only a few available trials per stimulus, BBCa outperforms both SVM and GPC. This indicates that BBCa is a more suitable method for neural response classification than the other two. Moreover, as detailed below, it allows for an exact evaluation of the evidence eqn. (4.8), which is necessary if subsequent stages of Bayesian inference are to be conducted without introducing approximation errors. This is an additional advantage which SVM cannot offer.

4.8 Application to neural spike data

To see the algorithm in action, it was used to analyze RSVP (rapid serial visual presentation) spike train data recorded from single cells of area STSa of the visual cortex of monkeys. These data were obtained through [48]. The monkey was presented with a continuous stream of natural pictures which were drawn from a set of eight different images selected for each neuron and the resulting firing pattern was recorded. This raw signal was turned into distinct samples, each of which contained the spikes from -250 ms before to 500 ms after the stimulus onset. The temporal resolution of the recording was 1 ms. Time indexes were aligned to the stimulus onset. Here and in the following, a 'dataset' denotes the collection of responses of a cell to a given stimulus set. A 'datapoint' or 'trial' is a stimulus-response tuple. The 'target stimulus' is the stimulus identity to be determined from a given response (i.e. the class label in the classification task). The stimuli were presented multiple times in pseudorandom order. Table 4.1 shows some of the characteristics of the available data. Stimulus onset asynchrony (SOA) is the time difference between the onsets of two consecutive stimuli. In all datasets used here for analysis, the stimuli were presented without gaps. Thus, SOA was equal to the duration of the stimulus. There was one dataset available for a given cell and SOA, but not all cells were tested at all SOAs. The stimulus set for a given dataset was chosen from 68 complex stimuli prior to recording. For a more detailed description of the recording procedure, see [48, 47].

A conventional way of decoding such spike trains, i.e. determine the stimuli from the neural resposes, is to count spikes in a time window. Let this window be denoted by $f_0(l, e)$, where l and e are the first and last time indexes included. The signal extracted from a spike train $s(t_i)$, $-250 \text{ ms} \leq t_i \leq 500$ ms, $s(t_i) \in \{0; 1\}$ is then

$$x = \frac{\sum_{t_i=l}^{e} s(t_i)}{e - l + 1} \tag{4.26}$$

i.e. the average firing rate. To find the expected window by means of Bayesian analysis, it is necessary to evaluate the posterior

$$P(f_0(l, e)|D) = \frac{P(D|f_0(l, e))P(f_0(l, e))}{\sum_l \sum_{e \geq l} P(D|f_0(l, e))P(f_0(l, e))} \tag{4.27}$$

where D is the set of recorded spike trains. The summation in the denominator runs over all start/end times which one chooses to include. All possible values of $l \geq l_{min}$

SOA	no. of cells	avg. no. of trials per stimulus
222 ms	40	24
112 ms	42	48
56 ms	40	92
42 ms	28	136
28 ms	40	190
14 ms	28	353

Table 4.1: Number of cells and trials per stimulus for each of the available stimulus onset asynchronies (SOA). SOA is the time difference between the onsets of two consecutive stimuli. Average trials per stimulus are rounded to the next nearest integer. There was one dataset available per cell and SOA. Each dataset contained responses to a set of 8 stimuli.

and $e \leq e_{max}$ were assumed to be equally likely prior to observing data, with the restriction $l \leq e$.

First, $P(D|f_0(l,e))$ must be evaluated. To do so, all spiketrains in the dataset were subjected to the transformation (4.26), thus yielding D_f. This transformed dataset was then fed into the classification algorithm. Hence, the evidences $P(D|m, f_0(l,e))$ became available. Assuming $P(m, f_0(l,e)) = P(m)P(f_0(l,e))$, i.e. the prior over the number of bins is independent of the prior over the window parameters, one can then write

$$P(D|f_0(l,e)) = \sum_{m=M_0}^{M_1} P(D,m|f_0(l,e)) = \sum_{m=M_0}^{M_1} P(D_f|m)P(m) \tag{4.28}$$

where M_0 and M_1 are the minimum and maximum number of bin boundaries, respectively. In this application, it was sufficient to choose $M_0 = 0$ and $M_1 = 10$, because the maximum of $P(D|m, f_0(l,e))$ was in most cases between $m = 1$ and $m = 7$. The evidence $P(D)$ is obtained by summing the numerator over all l, e and m.

With this posterior, one can evaluate the expectations

$$E[l] = \sum_l \sum_{e \geq l} P(f_0(l,e)|D)l \tag{4.29}$$

$$\mathrm{Var}[l] = \sum_l \sum_{e \geq l} P(f_0(l,e)|D)l^2 - E[l]^2 \tag{4.30}$$

and likewise for e and $e - l + 1$. This yields the expected latency, window end and window width, along with their variances.

4.8.1 Multiple datasets, joint and marginal expectations

In neurophysiological experiments, the number of stimulus repetitions that can be achieved is often quite limited. Thus, pooling data from different runs, possibly recorded in response to different stimulus sets, is desirable. If the parameters to be estimated can be expected to have similar values across datasets, then (4.28) can be replaced by

$$P(\{D^j\}|f_0(l,e)) = \prod_j \sum_{m_j=M_0}^{M_1} P({D_f}^j|m_j)P(m_j) \tag{4.31}$$

where j runs over all datasets, assuming that the datasets were drawn independently. This allows for the computation of expectations of the joint distribution.

When pooling data from the same cell recorded at different presentation rates, it would seem sensible to assume that the latencies should be more alike than the response durations. Thus, one would compute a single latency by marginalization:

$$\begin{aligned} P(\{D^j\}, f_0(l,.)) &= \prod_j \sum_{e_j \geq l} \sum_{m_j=M_0}^{M_1} P(D^j|m_j, f_0(l,e_j))P(m_j)P(f_0(l,e_j)) & (4.32) \\ P(f_0(l,.)|\{D^j\}) &= \frac{P(\{D^j\}, f_0(l,.))}{\sum_l P(\{D^j\}, f_0(l,.))} & (4.33) \end{aligned}$$

where the denominator of the r.h.s. of (4.33) is the evidence of this hypothesis (i.e. all datasets have the same latency but different response durations). Thus

$$E[l] = \sum_l P(f_0(l,.)|\{D^j\})l \tag{4.34}$$

Likewise, an evaluation of the overall response duration can be performed by marginalizing over the latencies.

4.9 Results on artificial data

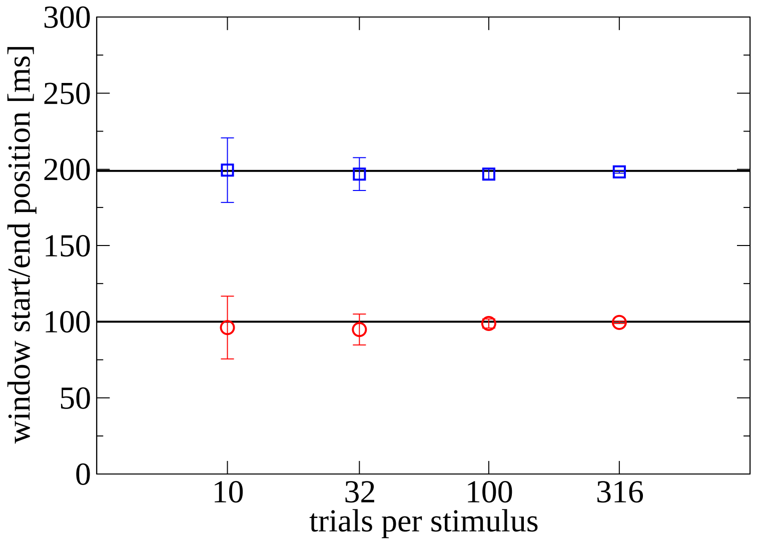
The algorithm was tested on artificial data first. Those were generated by a simulated neuron having a response latency of 100 ms and a response length of 100 ms. The firing probability, i.e. the probability of observing an event in a given time bin, was assumed to be maximal at the beginning of the response. It then decreased by

50% within 50 ms, and stayed constant at that value for the remaining 50 ms. The maximum firing rates were 60 spikes/s for the 'strongest' stimulus, 40 spikes/s for the 'second strongest', 15 spikes/s for the 'weakest' and 30 spikes/s for the rest (5 stimuli). These values resemble those found in the real data. Similar to the RSVP data analyzed below, stimuli were presented without gaps. Therefore, firing rates before the response latency and after the response offset were assumed to evoked by another, randomly drawn stimulus. Fig. 4.5, top, shows the results. For the analysis, it was assumed that the neuron starts responding somewhere between 50 ms and 150 ms after the stimulus onset, and that the response offset is found between 50 ms and 150 ms after the latency.

Even for a dataset containing as little as 10 trials per stimulus (i.e. the number of times each stimulus was presented), the expected window boundaries are close to their real values. The standard deviations are small enough ($\approx$ 20 ms) to yield useful results in neurophysiological experiments. As the number of trials per stimulus grows, the standard deviations decrease and the expected start/end positions move closer to their real values. From 100 trials per stimulus onward, the correct window parameters are determined by the algorithm almost with certainty.

The results for different numbers of datasets are depicted in fig. 4.5, bottom. Each dataset contained 10 trials per stimulus. The behavior is similar to that observed in fig. 4.5, top. Performance at 3 datasets with 10 trials per stimulus each is comparable to 1 dataset with 32 trials per stimulus. It appears that the total number of available trials (i.e. number of trials per stimulus in a dataset $\times$ number of datasets) is the determining factor for the quality of the results. At 3 and 10 datasets, the expected start positions are a few ms before the response onset. This indicates that it is better for classification performance to include a few time bins from the response to the previous stimulus than to exclude the first few ms from the response to the target stimulus, which are the most informative.

Classification performance for two datasets of different sizes is plotted in fig. 4.6. If a stimulus was to be identified by the response it evoked, one would pick the stimulus that maximizes $P(s|x)$, where s is the stimulus and x is the response. At 10 trials/stimulus, the only distinction possible is that between 'strongest' ($x > 44.5$), 'weakest' ($x < 11.5$) and 'rest'. At 316 trials per stimulus, 4 classes can be separated: 'weakest' ($x < 18.0$), '2nd strongest' ($32.0 \leq x < 36.5$) and 'strongest' ($x \geq 36.5$). The range $18.0 \leq x < 32.0$ would then be assigned to 'rest'. The optimal decision boundaries, which can be computed via Bayes' rule from the generating distributions, are: 'strongest', if $x \geq 40.0$, '2nd strongest', if $30.0 \leq x < 40.0$, 'rest', if $20.0 \leq x < 30.0$ and 'weakest' if $x < 20.0$. Note that these are in good agreement with the boundaries found by the BBCa.



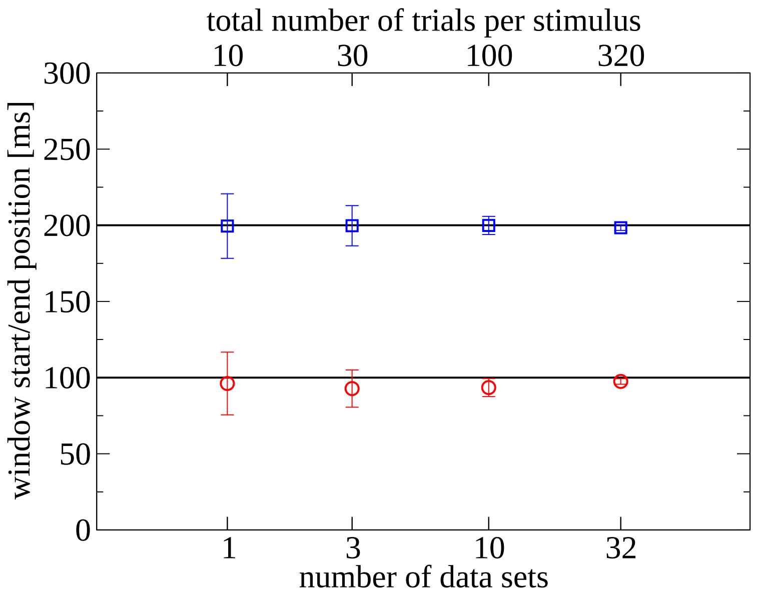
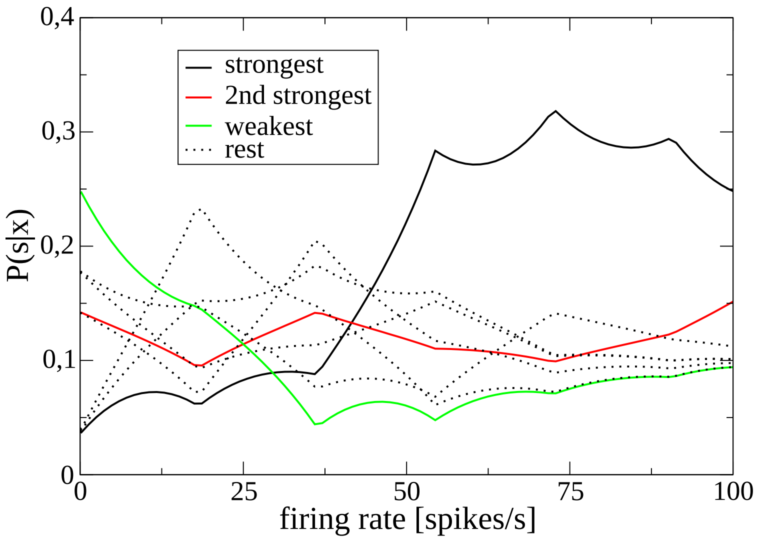


Figure 4.5: Top: expected window start (circles) and end (squares) as a function of the number of trials per stimulus. Bottom: expected window start (circles) and end (squares) as a function of the number of datasets. Each dataset contained 10 trials per stimulus. Error bars are ± 1 standard deviation. Averages computed over at least 100 runs.



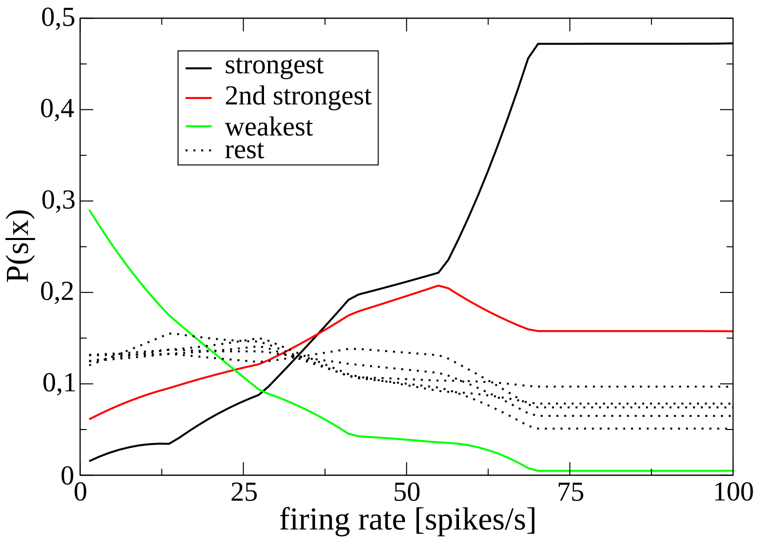


Figure 4.6: Expected classification performances for two datasets of different sizes. Top: 10 trials/stimulus. The only distinction possible is that between 'strongest' and 'weakest' and 'rest'. Bottom: 316 trials/stimulus. Here, 4 classes ('strongest', '2nd strongest', 'weakest' and 'rest') can be separated.

4.10 RSVP results

4.10.1 How similar are cells?

The available data contained measurements for 6 different stimulus onset asynchronies (SOA): 14 ms, 28 ms, 42 ms, 56 ms, 112 ms and 222 ms. As explained above, here SOA is the time interval between the onsets of two consecutive stimuli. In the following, the terms 'information latency' (IL) and 'information response duration' (IRD) refer to the time indexes of the window start and window length as determined by the classification algorithm. The term 'response latency' (RL) denotes the time index at which a response was detected by a method described in hypothesis H2 (see below). Six hypotheses were compared:

- H1 joint: for a given SOA, all cells have the same IL and IRD. While previous results (see e.g. [48]) indicate that this assumption is likely to be wrong, it served as a 'null hypothesis' against which the other hypotheses were compared.

- H2 IL=RL: this is a popular hypothesis in neurophysiology. The response latency was determined on a dataset by dataset basis by a method described in [48]: The spike train is smoothed by a Gaussian of width 10 ms to yield a spike density function. Its mean μ and standard deviation σ are computed in a time window of 250 ms directly prior to the stimulus onset. The RL is defined as the first 1 ms time bin after which the spike density function exceeds $\mu + 2.58\sigma$ for at least 25 ms. Moreover, the response duration was taken to be equal to the stimulus duration.

- H3 latency: for a given SOA, all cells have the same IL, but possibly different IRDs. This assumption allows for the computation of an overall IL as a function of the presentation rate.

- H4 window length: for a given SOA, all cells have the same IRD, but possibly different ILs. This assumption allows for the computation of an overall IRD as a function of the SOA.

- H5 single: for a given SOA, each cell has a different IL and IRD.

H6 latency per cell: each cell has an IL, which does not change with the presentation rate. The IRD may change with the presentation rate. This hypothesis is similar to the one used in [48] for the determination of the latency of each cell.

For H2, only the datasets with SOAs 42 ms - 222 ms were used, since the faster presentation rates do in most cases not yield enough signal to allow for a determination of the RL. Thus, two sets of comparisons were performed: the first included H1, H2, H3, H4 and H5 (left half of table 4.2), the second H1, H3, H4, H5 and H6 (right half of table 4.2).H1 was used as the point of reference for all others, i.e.

$$\begin{aligned} \log_{10}(\text{evidence})\text{ gain of HX} &= \log_{10}(\text{evidence of HX}) \\ &- \log_{10}(\text{evidence of H1}) \end{aligned} \tag{4.35}$$

	42 ms - 222 ms	14 ms - 222 ms
hypothesis	$\log_{10}$(evidence) gain	$\log_{10}$(evidence) gain
H1	0.00	0.00
H2	47.5	-
H3	167.6	199.3
H4	301.9	400.5
H5	367.1	494.4
H6	-	456.7

Table 4.2: Evidence comparisons for the different hypotheses described in the text. $\log_{10}$(evidence) gains are computed w.r.t the evidence of H1. For the evidence comparisons in the left half of the table, only recordings with SOAs between 42ms - 222ms were used, because the shorter presentation rates do in most cases not yield enough signal to allow for a determination of the RL. For the comparisons in the right half, recordings at all available SOAs were included.

As expected, H1 is the most unlikely one. IL=RL already results in a substantial evidence gain (being $\approx 10^{47}$ times more probable). However, compared to the remaining hypotheses, it can still safely be discarded, indicating that this way of information extraction will yield suboptimal results.

Comparing H3 and H4 shows that the IRDs seem to be more alike than the ILs, indicating that the latency depends more strongly on the cell (and perhaps on the stimulus parameters) than the response duration does. This is consistent with H6 being more probable than either of them. Nevertheless, as noted above, they serve to compute the expected ILs and IRDs across all cells.

H5, despite its additional degrees of freedom (one IL and IRD per cell and SOA), appears to provide the best explanation of the data. In other words, both IL and IRD depend strongly on both the cell and the SOA. This result should not be understood to mean that averaging over cells will generally yield meaningless results, but rather that when such averaging is performed (e.g. to compute information flows), it is advisable to compute the IL and IRD anew for each cell and SOA.

Figure 4.7 shows the expected classification performance for a single cell at SOA 56 ms. Three stimuli can be separated. Ranking was done by expected certainty of stimulus identification, i.e. by the probability $P(s|f_0(l,e))$ that stimulus s had been presented given the response computed from the spike train via $f_0(l,e)$. Thus, the 'second best' stimulus is not the one with the second strongest response, but rather the one with the weakest (this way of ranking stimuli is different from [48], where stimuli were ordered by response strength). This begets the question: is information transmitted through not firing as well as through firing, and if so, how much? If the neural code implemented by the visual system is sparse, an assumption for which experimental evidence has been collected [3], then the analysis presented in section 3.6 implies that there should hardly be any information conveyed by neural silence.

Moreover, note that even the 'best' stimulus can only be classified correctly with a probability of at most ≈ 0.3. While this allows for a decision with greater certainty than a decision based solely on prior knowledge ($p = 0.125$ for 8 stimuli), it is still fairly low. In a 'best' vs. 'rest' decision task, one would therefore always choose 'rest'. Thus, additional information is required if the stimulus is to be identified. That that is possible at SOA 56 ms has been shown in [48] via human psychophysical experiments using the same stimulus set as that employed for the single-cell recordings which yielded the data for the analysis presented here.

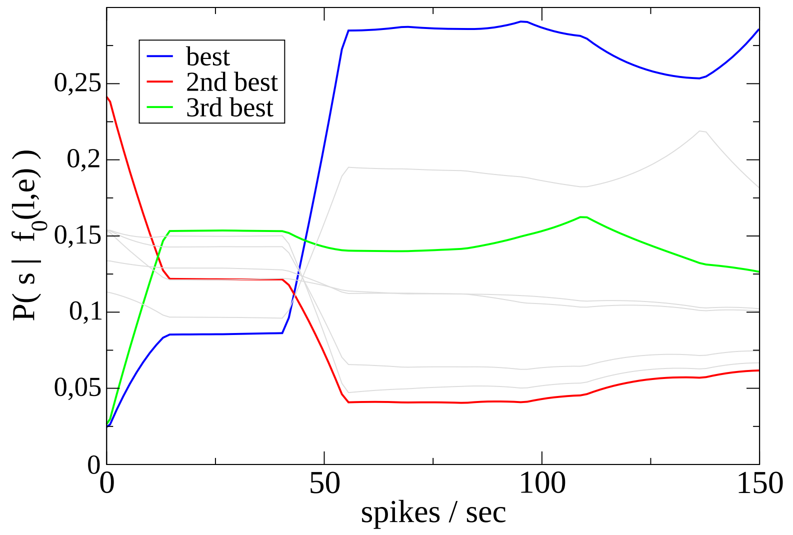


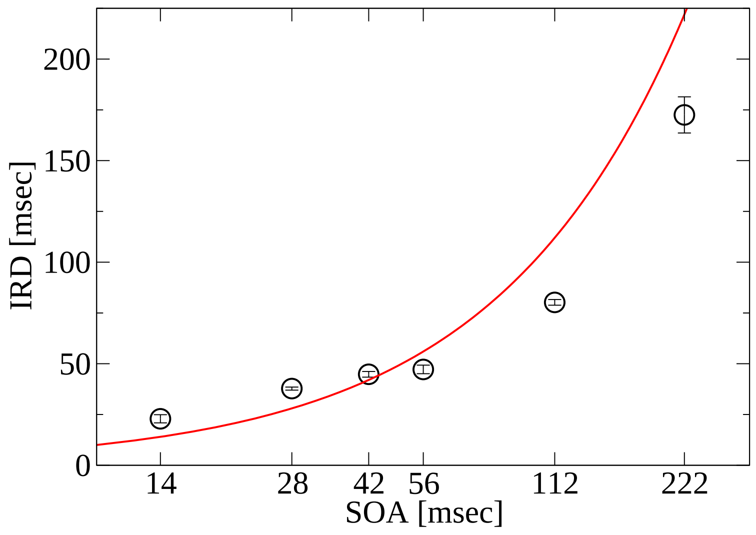
Figure 4.7: Expected classification performance for cell 144.4. Three stimuli can be separated. Stimulus ranking was done by $P(s|f_0(l,e))$, i.e. the best stimulus is the one which can be identified with the greatest certainty given the response computed from the spike train via $f_0(l,e)$. For details, see text.

4.10.2 Latencies and response durations

The overall ILs and IRDs computed via H3 and H4, respectively, are shown in fig. 4.8. In accordance with previous results on the same data [48], the IRD increases with the SOA. For SOAs greater than $\approx$ 50 ms, the IRD was found to be shorter than the SOA, this behavior is reversed for shorter SOAs. This appears at first contradictory to [48], where the response was reported to outlast the SOA by $\approx$ 60 ms. That might, however, be due to a difference in method: in the aforementioned study, response offset was defined as the start of the first 30 ms time window in which all 1 ms bins failed a t-test ($p>0.05$) performed on the spike density functions of the best stimulus and all other stimuli together. In contrast, the approach presented here calculates the expected response length by averaging over all possible windows weighted by their classification performance. Therefore, it will tend to exclude parts of the response which, while they may still contain some information about the stimulus, will decrease the signal-to-noise ratio compared to the 'better' parts available. Thus, the windows can be expected to be somewhat shorter. That the excluded part does indeed contain some information about the stimulus will be demonstrated in sections 4.11.2 and 5.14.2 where the temporal structures of the classification evidence and the mutual information are analyzed.

The IL seems to increase with the SOA, too. For SOA=41 ms and 14 ms recordings were made only on a subset of the available cells, this might offer an explanation for the slight increase of the IL at 41 ms SOA compared to 56 ms SOA. Nevertheless, the overall trend of the IL is still clearly discernible. The large standard deviation of the IL at 14 ms SOA also indicates that at this very high presentation rate, classification information in the response becomes increasingly hard to localize.

Furthermore, note that RL $\geq$ IL (within standard error) for all SOAs for which RL could be determined. Yet, as demonstrated in section 4.11.2, the information flow does seem to begin at the IL. This indicates that beginning the response analysis at the RL might truncate a rather informative part of the signal. The reason for the increase of RL with SOA might be due to the fact that both the mean and the standard deviation of the firing rate, averaged across stimuli, tend to increase with SOA. This would, in turn, lead to a higher RL.



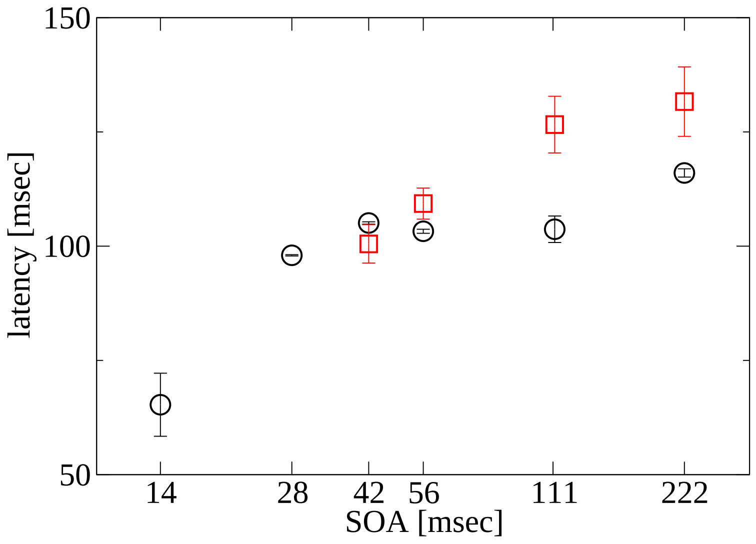


Figure 4.8: Top: Expected IRDs ± one standard deviation, averaged over all cells. The solid line marks the SOA. IRD is shorter than the SOA when the latter is ≥ 56 ms, and longer for SOAs ≤ 46 ms. Bottom: Expected ILs ± one standard deviation (circles) and average RLs (squares). Shorter stimuli appear to result in a quicker information flow onset. For 42 ms and 14 ms SOA, data were available only for a fraction of the cells that were tested at the other SOAs.

4.11 Conclusion

4.11.1 Algorithm

The presented algorithm exactly computes evidences and expected classification probabilities with a computational effort of $O(K^2M^2)$, where K is the number of data points and $M+1$ is the number of bins. This is significantly faster than a naïve approach of simply trying all possible permutations of the bin boundaries between the data points, which would take $O(M^K)$. This acceleration is accomplished by reusing intermediate results. Moreover, it yields the evidence of the data given the considered model class, which is here quantified by M, the number of bin boundaries. This evidence, which is required if further stages of inference are to be performed, is usually not available when using methods that compute the class-conditional probabilities first and convert them afterwards into the probabilities of the class labels via Bayes' theorem.

Note that the expected probabilities (4.7) are, by virtue of (4.8) and (4.23) polynomials of order M in x: for example, (4.25) is quadratic. The coefficients of these polynomials change whenever x passes over a data point, in such a way that $P(y|x)$ remains continuous. Thus, one could also look at this algorithm as a form of piecewise Bayesian polynomial interpolation.

As mentioned above, finding a suitable mapping from the feature vector onto a scalar, $f(\vec{w}) \mapsto x$, is generally not trivial. Most likely, exact inference of $f(\vec{w})$ will not be possible, and thus an approximation scheme needs to be employed. No matter which technique one decides to try, it will usually be necessary to evaluate a quantity proportional to

$$p(f(\vec{w})|D) \tag{4.36}$$

i.e. (4.27) with $f_0(l,e)$ replaced by $f(\vec{w})$. The denominator of (4.27) will probably be intractable, but the numerator is readily accessible, once a prior over the possible functions has been chosen. Then, a Monte-Carlo technique can be applied to determine e.g. the expected $f(\vec{w})$.

Another possibility, which might speed up the inference process, is to find the maximum of (4.36) w.r.t. $f(\vec{w})$ via a gradient-based technique. Lets assume $f(\vec{w}) = f(\vec{w},\vec{\theta})$, where $\vec{\theta}$ is a vector of parameters that governs the exact form of $f(\vec{w},\vec{\theta})$.

Since (4.8) is a polynomial in the x_k, its gradient w.r.t the x_k can be evaluated by an algorithm similar to that described above. Thus, using the chain rule,

$$\nabla_{\vec{\theta}}\log(p(f(\vec{w},\vec{\theta})|D)) = \sum_{k=0}^{K-1}\frac{\partial\log(P(D|f(\vec{w},\vec{\theta})))}{\partial x_k}\nabla_{\vec{\theta}}f(\vec{w}_k,\vec{\theta}) + \nabla_{\vec{\theta}}\log(p(\theta)) \quad (4.37)$$

$\frac{\partial\log(P(D|f(\vec{w},\vec{\theta})))}{\partial x_k}$ can, by virtue of (4.28), be computed via the derivative of (4.8) w.r.t x_k. $p(\vec{\theta})$ is the prior of $\vec{\theta}$. $\nabla_{\vec{\theta}}f(\vec{w}_k,\vec{\theta})$ depends on the exact functional form of $f(\vec{w},\vec{\theta})$. Assuming, for instance, that it is a feedforward neural network with a single output, then $\vec{w}_k$ would be the vector of inputs for the k-th example, x_k the resulting output and $\vec{\theta}$ stands for the weights. $\nabla_{\vec{\theta}}f(\vec{w}_k,\vec{\theta})$ could then be computed by some variant of the backpropagation algorithm [81], and thus gradient ascent on $\nabla_{\vec{\theta}}\log(p(f(\vec{w},\vec{\theta})|D))$ becomes feasible. Once the weights maximizing the posterior have been determined, refinements are of course possible, such as using a Laplace approximation for the weight posterior, or Monte-Carlo techniques. The important point is that, since a feedforward network with sigmoid transfer functions can model any continuous mapping onto a scalar to any desired degree of accuracy given that the number of hidden units is allowed to grow [15], one might hope that this approach could in principle solve any classification task for which $f(\vec{w},\vec{\theta})$ can be reasonably well approximated by a continuous function.

The applicability of the algorithm for the analysis of neural spike train data has been demonstrated. It yields useful results even when only $O(10)$ trials per stimulus are available, which can usually be accomplished in recordings of single cells. Thus, analyses can be conducted not only on a population level, but also on a cell by cell basis.

4.11.2 STSa neuron populations adapt their processing speed to the presentation rate

It is well known that response latencies differ notably between cells in STSa [48, 69]. In addition, it has been shown [70] that latency encodes information about the contrast with which a stimulus is presented. It has been demonstrated (fig. 4.8) that there appears to be a quasi-monotonic relationship between IL and SOA, i.e. the shorter the presentation time, the faster the information flow onset. The found ILs are within the boundaries established by other investigators: in [69] it is demonstrated that the latency in STSa cells should be $\geq$ 71 ms. One might wonder how the visual system accomplishes this adaptation, and if it is an adaptation at all: in [69], it has been argued that the information flow from retinal output to STSa has to be predominantly feedforward in order to achieve the observed short latencies. On the other hand, it is known from experiments with artificial neural networks [35, 20, 26] that lateral or feedback connections can give rise to sparse, almost factorial neural codes that facilitate pattern recognition and bear a qualitative resemblance to the behavior of cells in V1. If those feedback connections are employed by the visual system, then it is unlikely that the later processing stages would wait until the earlier ones have converged, i.e. there would be a signal arriving in STSa even though V1 is still busy making sense of its input. This early signal could convey some information about the stimulus, but not as much as the later part, which would become available if the stimulus was presented long enough. Presuming this happened in the visual system, then one would expect that the method used here for the determination of the best window should focus on the later part of the response if the SOA is long enough, and move the IL forward when the SOA is reduced. To shed some light on this hypothesis, see fig. 4.9: here, the logarithm of the evidence (4.28) divided by the number of data points is plotted for $l + 10$ ms $= e$, i.e. the log evidence for a 10 ms sliding window (thin dotted lines) and also its running average over 11 ms (thick lines). This quantity will be high if good classification is possible, and low otherwise. Thus its value indicates how much information the spike count carries about the stimulus. The log evidence values have been aligned so that their average in the interval -250 ms $\leq$ e-IL $\leq$ -150 ms is 0. The responses in this time interval should contain no information about the stimulus, given that the longest IL was $\approx$ 116 ms (for SOA 222 ms), and can thus provide baseline values for the curves. Note that the rising slopes which indicate the onset of the information flow are almost perfectly aligned with each other for SOAs 222 ms, 112 ms and 56 ms. Since the IL at SOA 222 ms is $\approx$ 13 ms larger than those at SOA 112 ms and 56 ms, this indicates that the cell population really begins transmitting later at the longest SOA. The maximum, too, appears to be reached later, which might be taken as another piece of evidence that some form of adaptation is taking place.

The situation is similar for the three shorter SOAs. Even though the rising slopes for 42 ms and 28 ms are not perfectly aligned, their temporal offset is certainly smaller than ≈ 7 ms, which is the difference in their ILs. This is true to an even stronger degree for the shortest SOA: IL(SOA=28 ms)-IL(SOA=14 ms) ≈ 33 ms, yet their rising slopes are within a few ms of each other.

One might still wonder if there is some signal before the IL, which is discarded by the analysis method used here, because it is too noisy. This is also unlikely, as can be seen in table 4.3, where some of the features of fig. 4.9 are listed. The information onset (ION) and offset (IOFF) relative to the IL were computed in the following manner: to estimate a noise level, the standard deviation σ_ν of each curve in the time interval -250 ms $\leq$ e-IL $\leq$ -150 ms was computed. ION and IOFF are the start and end time (rounded to the next nearest ms) of the interval in which the baseline-aligned log evidences are consistently greater than $2.58\sigma_\nu$. In other words, ION marks the beginning of stimulus-related information, whereas IOFF marks the end. Within standard errors, ION coincides well with IL, i.e. there is very little evidence for information transmission before IL. Thus, an 'early part' of the response could not be detected.

IOFF outlasts IRD (the latter is indicated by the arrows in fig. 4.9) quite significantly for the longer SOAs. This difference decreases with decreasing SOA, and disappears (within standard error) for SOA 14 ms. One might thus say that with respect to the duration of the signal which is best suited for stimulus identification (i.e. IRD), the cell population shows a transient response at the longer SOAs, and a sustained response at the shorter ones. The latter part of the response (i.e. between IRD and IOFF) might perhaps be employed by the cell population to signal that 'the stimulus is still there', but since the stimulus identity has already been established (as well as possible) by that time, this latter part needs not be as informative (and could thus possibly be produced with less energetic effort). When the total duration of the information transmission is considered, the response is sustained (fifth column of table 4.3): IOFF-SOA is always greater than 0 ms. Given that the sliding window was 10 ms long, this indicates that the response outlasts the stimulus at most by a few ms.

As noted above, IRD $<$ SOA if SOA $>\approx 50$ ms. Thus, for longer stimuli, the visual system should be able to separate the responses to successive stimuli, because

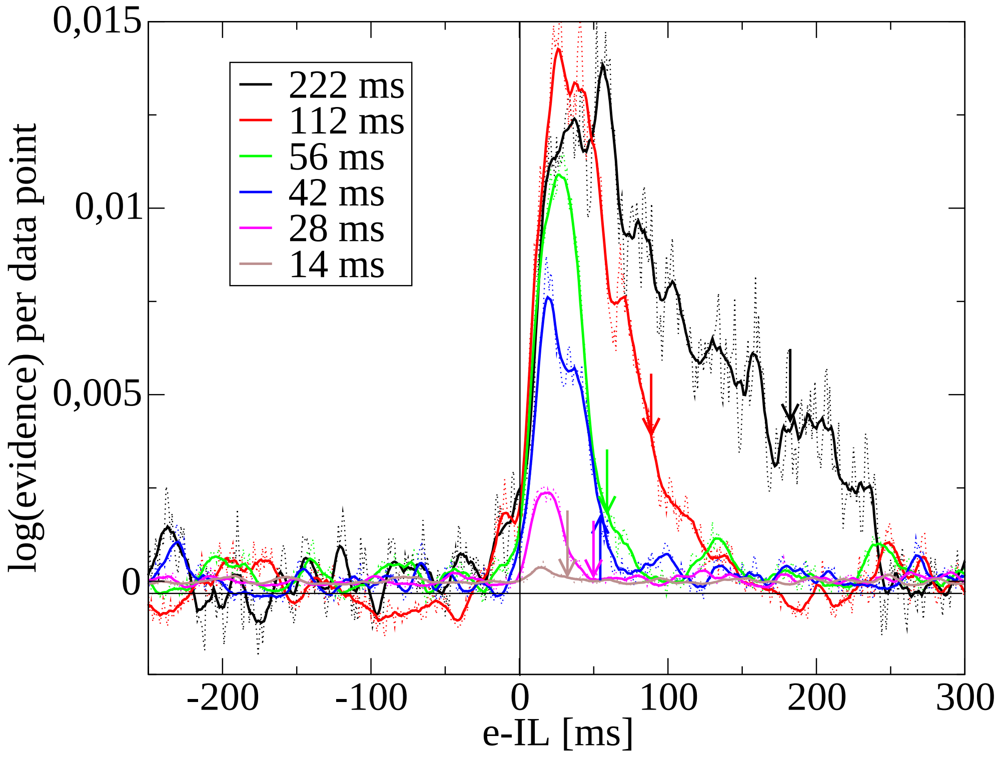


Figure 4.9: $\log(P(D_f|f_0(e-10\text{ ms},e))$ per data point, i.e. log evidence per data point in a sliding time window of length 10 ms (thin dotted lines) and their running averages in an 11 ms window (thick lines). Values are aligned so that the average for e-IL $<$ -150 ms are zero. There is no indication of stimulus-related information when e $<$ IL. The arrows indicate the end of the optimal time window for stimulus discrimination. Some information seems to be transmitted after that, but including this latter part of the response would reduce classification performance.

SOA	ION	IOFF	IOFF-IRD	IOFF-SOA
222	0 ± 0.91	238 ± 0.91	65.5 ± 8.9	16 ± 0.91
112	-3 ± 2.9	118 ± 2.9	37.8 ± 3.2	6 ± 2.9
56	-3 ± 0.44	73 ± 0.44	25.8 ± 2.1	17 ± 0.44
42	0 ± 0.26	58 ± 0.26	13.2 ± 1.4	16 ± 0.26
28	1 ± 0.14	41 ± 0.14	3.4 ± 0.76	13 ± 0.14
14	7 ± 6.9	25 ± 6.9	2.1 ± 7.2	11 ± 6.0

Table 4.3: Second and third column: population averages of the information onset (ION) and offset (IOFF) relative to the IL. Those are defined as the start and end of the time interval within which the log evidence per datapoint in a 10 ms sliding window (see fig. 4.9) is consistently greater than the noise level. ION coincides well with IL. The part of the response after IRD but before IOFF (fourth column) contains some information about the stimulus, but will yield suboptimal classification performance. In the fifth column, the difference between IOFF and SOA is shown. Given that the sliding window was 10 ms long, it is likely that the information-carrying response outlasts the stimulus duration by a few ms. For details, see text.

IRD is the duration of the response needed for best stimulus discrimination. This is no longer the case for the shorter SOAs: here responses to stimuli will begin to overlap, and thus optimal classification performance can no longer be attained.

One possible mechanism that might explain the observed variations in IL and IRD is threshold adaptation or, to the same end, residual activation: suppose the firing threshold of the cell increased with SOA. Then it would begin to fire sooner as the SOA decreases, and also longer w.r.t SOA. It is currently unclear what might bring about such an adaptation. But if one assume that the cell behaves to some degree like a 'leaky integrator', then in the shorter SOA conditions, when it is exposed to a 'good' stimulus, there might still be some activation left from its last presentation. This activation would have 'leaked' out of the cell had the SOA been longer.

As shown in fig. 4.7, the probability of correctly identifying the stimulus given the response computed from the spike train via $f_0(l,e)$ does not exceed 0.3 (this result is of course conditioned on the cell and the stimulus set). The cell from whose responses this classification graph was computed is a fairly average specimen of the population that was sampled. It is thus evident, that even for a stimulus set as limited as the one used in this experiment, more information is required to identify the stimulus with reasonable certainty (say, 99%). This information would become available if one observed not the activity of a single cell, but that of a population. As demonstrated in [27, 68], combining the signals of a few cells via Bayes' theorem on the assumption that their responses (given the stimulus) are conditionally independent can yield the desired increase of certainty. Since simultaneous multi-cell recordings were not available to me, I proceeded from the assumption that other neurons in the relevant population behaved similarly to the one in question, in that they are able to tell the same 'best' stimulus with the same certainty. Using Bayes' theorem, one thus obtains (for a derivation, see appendix C)

$$P(s = s_b | x_1, \ldots, x_N) \leq \frac{P_b^N}{P_b^N + \frac{(1-P_b)^N}{(C-1)^{N-1}}} \tag{4.38}$$

where $x_1, \ldots, x_N$ are the responses of the (hypothetical) population, s_b is the 'best' stimulus, P_b the probability of correctly identifying it from the output of a single cell, and C is the size of the stimulus set. A uniform prior over the stimuli within the set was assumed. (4.38) is fulfilled with equality if the cells behave like 'grandmother cells' [4], i.e. the probability for correctly identifying another stimulus $s \neq s_b$ is $\frac{1-P_b}{C-1}$. This probability approaches 0 for large C, in other words: each cell responds only to one stimulus.

Setting $P_b = 0.3$ and $C = 8$ and plotting $P(s = s_b | x_1, \ldots, x_N)$ (given the 'grandmother cell' assumption) over N yields fig. 4.10. In the best case, as little as six independent 'confirmations' of the stimulus identity are enough to be more than 99% certain. This lends further credibility to the hypothesis that the visual system employs sparse codes for the representation of stimuli [29], because only a small number of units needs to be active to code a given stimulus. It should be noted, however, that the small number of units necessary for reasonable certainty is likely due to the unrealistically small stimulus set (8 stimuli). As explained in [68], the visual system does not have the luxury of knowing the prior distribution which an experimenter chooses. Hence, the number of neurons involved in the representation of a stimulus can be expected to be somewhat larger, estimates are of $O(100)$ [69].

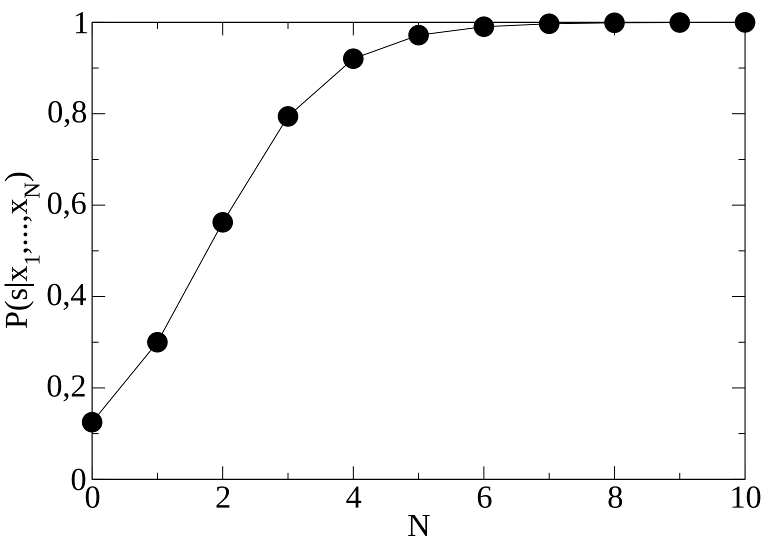


Figure 4.10: Probability of correctly identifying a stimulus given the responses of a (hypothetical) population of cells of size N. For details, see text.

Chapter 5

Information extraction from neural spike trains II: Bayesian Bin Distribution Inference and Mutual Information

5.1 Introduction

Computing entropies and mutual information from data of limited sample size, such as recordings of mammalian brain cell activity, is a difficult task. A related problem is the estimation of probability distributions and/or densities. In fact, once a good density estimate is available, one could also expect the entropy estimates to be satisfactory[1]. One of the several approaches proposed in the past is kernel based estimation, having the advantage of being able to model virtually any density, but suffering from a heavy bias [80]. Another category consists of parameterized estimation methods, which choose some class of density function and then determine a set of parameters that best fit the data by maximizing their likelihood. However, maximum likelihood approaches are prone to over-fitting. A common remedy for this problem is cross-validation [85], which, while it appears to work in many cases, can at best be regarded as an approximation.

[1]Note, however, that a good density estimate is not a prerequisite for a satisfactory entropy estimate [61].

Over-fitting occurs especially when the size of the dataset is not large enough compared to the number of degrees of freedom of the chosen model. Thus, as a compromise between the two aforementioned methods, *mixture models* have recently attracted considerable attention [87]. Here, a mixture of simple densities (Gaussians are quite common) are used to model the data. The most popular method for determining its parameters is Expectation Maximization [19], which, while having nice convergence properties, is still aiming at maximizing the likelihood. The question of how to determine the best number of model parameters therefore remains unanswered in this framework. Some progress has been made in this direction by employing Dirichlet process mixture models [60]. Nevertheless, exact solutions are hard to come by with this approach, so one is usually required to use MCMC (Markov Chain Monte Carlo) techniques (see section 2.6.3) for the estimation of quantities of interest. Two other noteworthy approaches to dealing with the overfitting problem are MML [91] (Minimum Message Length) and MDL [78] (Minimum Description Length), which are similar to Bayesian inference.

In this chapter, an exactly tractable model will be presented. In section 5.2, the model is introduced. While still simple in construction, it is sufficiently general to model any distribution (or density), which will be illustrated in section 5.10, along with a discussion of some of the model's limiting properties. Sections 5.3 and 5.4 describe the computational framework for iterating over all possible choices of the model's parameters in polynomial time, which is a necessary prerequisite for the Bayesian estimation of entropies and the mutual information. The developed BBDI algorithm can then be applied directly to perform model selection and probability distribution/density inference (see sections 5.5 to 5.7). Thus, while density estimation was not the motivation for the development of the BBDIa, it can be used to that end with little extra effort. In section 5.8, the BBDIa is extended to yield exact expectations of entropies and their variances. Thus, an exact expectation of the mutual information and a strict upper bound on its variance can be computed as well, as shown in sections 5.11 to 5.13. In section 5.13, the performance of the algorithm is also compared with two competing methods. Finally, in section 5.14, it is used for an information-theoretic evaluation of the same neural spike train data which were studied in the previous chapter. To that end, the expected temporal windows

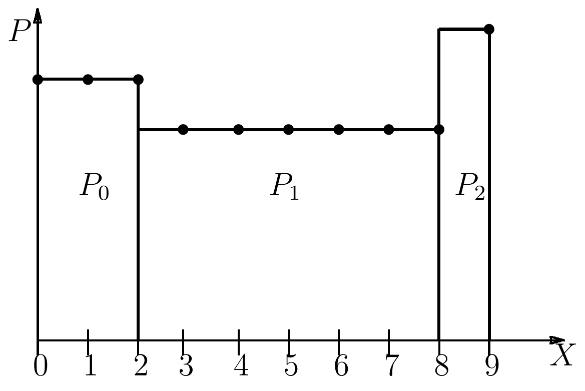


Figure 5.1: An example configuration for $K = 10$ values of X, three bins ($M = 2$ interval boundaries), containing the probabilities P_m. Here, $P(0) = P(1) = P(2) = \frac{P_0}{3}, P(3) = \ldots = P(8) = \frac{P_1}{6}, P(9) = P_2$. The points indicate which values of X belong to a bin. The algorithm iterates over all possible configurations to find the expected probability distribution and other relevant averages.

as determined by the BBCa from the last chapter are employed to compute the cells' responses, and the mutual information between response and stimulus label is evaluated.

5.2 Bayesian binning

Suppose X was a discrete random variable , which could take on the values $k = 0, \ldots, K-1$. Furthermore, assume that a notion of similarity between any two instances x_1, x_2 of X could be quantified in a meaningful way by $x_1 - x_2$. It is then natural to try to model the distribution of X by $M + 1 \leq K$ contiguous, non-overlapping bins, such that the bins cover the whole domain of X. Each bin has a probability $P_m, m = 0, \ldots, M$, subject to the normalization constraint $\sum_{m=0}^{M} P_m = 1$. This probability is evenly distributed amongst the possible values of X in the bin (see fig. 5.1). If a bin ends at k_m and the previous one at k_{m-1}, then

$$\forall k_{m-1} < k \leq k_m : P(X = k) = \frac{P_m}{\Delta k_m} \tag{5.1}$$

with $\Delta k_m = k_m - k_{m-1}$, $k_{-1} = -1$ and $k_M = K-1$, i.e. the first bin starts at $X = 0$ and the last one includes $X = K-1$. Assume now a multiset $D = \{x_1, x_2, \ldots, x_N\}$

of N points drawn independently from the distribution which is to be inferred was available. Given the model parameterized by M and $\{(P_m, k_m)\}$, the probability of the data then is given (up to a factor) by a multinomial distribution

$$P(D|M, \{(P_m, k_m)\}) = \prod_{m=0}^{M} \left(\frac{P_m}{\Delta k_m}\right)^{n_m} \tag{5.2}$$

where n_m is the number of points in bin m. One might argue that a multinomial factor should be inserted here if the data points are not ordered. This would, however, only amount to a constant factor that is cancelled out when averages (posteriors, expectations of variables etc.) are computed. It will therefore be dropped. The factors $\Delta k_m^{-n_m}$ express the intention of modeling Δk_m possible values of X by the same probability. From an information theoretic perspective, we are trying to find a simpler coding scheme for the data while preserving the information present in them: The message '$X = k$' for all $k : k_{m-1} < k \leq k_m$ would be represented by a code element of the same length $\log\left(\frac{\Delta k_m}{P_m}\right)$. In contrast, were the $\Delta k_m^{-n_m}$ absent, then the message '$X = k$' for each $k : k_{m-1} < k \leq k_m$ would be represented by the same code element, i.e. an information reduction transformation would have been applied to the data.

Next, the evidence of a model with M bins will be computed, i.e. $P(D|M)$. It is obtained by multiplying the likelihood (eqn. (5.2)) with a suitable prior $p(\{(P_m, k_m)\}|M)$ to yield $p(D, \{(P_m, k_m)\}|M)$. This density is then marginalized w.r.t. P_m and k_m, which is done by integration and summation, respectively. The summation boundaries for the k_m have to be chosen so that each bin covers at least one possible value of X. Since the bins may not overlap, $k_0 = 0 \ldots K - 1 - M$, $k_1 = k_0 + 1 \ldots K - M$ etc.. Because the P_m represent probabilities, their integrations run from 0 to 1 subject to the normalization constraint, which can be enforced via a Dirac $\delta()$ function:

$$P(D|M) = \sum \sum\nolimits_{\{k\}} \int_0^1 d\vec{P} P(D|M, \{(P_m, k_m)\}) p(\{(P_m, k_m)\}|M) \tag{5.3}$$

where

$$\int_0^1 d\vec{P} = \int_0^1 dP_0 \int_0^1 dP_1 \ldots \int_0^1 dP_M \delta(1 - \sum_{m=0}^{M} P_m) \tag{5.4}$$

and

$$\sum\sum\nolimits_{\{k\}} = \sum_{k_0=0}^{K-1-M} \sum_{k_1=k_0+1}^{K-M} \cdots \sum_{k_{M-1}=k_{M-2}+1}^{K-2} \tag{5.5}$$

Note that the prior $p(\{(P_m, k_m)\}|M)$ is a probability density, because the P_m are real numbers.

5.3 Computing the prior $p(\{(P_m, k_m)\}|M)$

The prior will be assumed to be non-informative, i.e. all possible configurations of $\{(P_m, k_m)\}$ are equally likely prior to observing the data. The prior can be written as

$$p(\{(P_m, k_m)\}|M) = p(\{P_m\}|\{k_m\}, M)P(\{k_m\}|M) \tag{5.6}$$

Note that the second factor on the r.h.s. is not a probability density, because there is only a finite number of configurations for the k_m. In the following, it shall be assumed that the probability contained in a bin is independent of the bin size, i.e. $p(\{P_m\}|\{k_m\}, M) = p(\{P_m\}|M)$. This is certainly not the only possible choice, but a common one: in the absence of further prior information, independence assumptions can be justified, since they maximize the prior's entropy. Thus, eqn. (5.6) becomes

$$p(\{(P_m, k_m)\}|M) = p(\{P_m\}|M)P(\{k_m\}|M) \tag{5.7}$$

Since all models are to be equally likely, the prior is constant (denoted by $c(M)$) w.r.t. k_m and P_m, and normalized:

$$\sum\sum\nolimits_{\{k\}} \underbrace{\int_0^1 d\vec{P}}_{\frac{1}{M!}} \times c(M) = 1 \tag{5.8}$$

Carrying out the integrals is straightforward (see (D.8) with all $n_m = 0$) and yields the normalization constant of a Dirichlet distribution. The value of the sums is given by (using the identity $\sum_{i=0}^{b} \binom{a+i}{i} = \binom{a+b+1}{b}$)

$$\begin{aligned}
\sum_{k_0=0}^{K-1-M} \cdots \sum_{k_{M-1}=k_{M-2}+1}^{K-2} 1 &= \sum_{k_0=0}^{K-1-M} \cdots \sum_{k_{M-2}=k_{M-3}+1}^{K-3} (K-2-k_{M-2}) \\
&= \sum_{k_0=0}^{K-1-M} \cdots \sum_{k_{M-2}=k_{M-3}+1}^{K-3} \frac{(K-2-k_{M-2})!}{(K-3-k_{M-2})!1!}
\end{aligned}$$

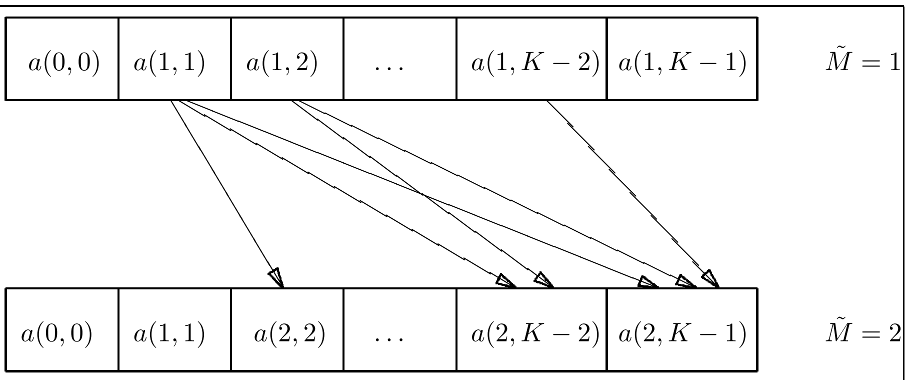


Figure 5.2: After all the evidence contributions $a(\tilde{M},\tilde{K})$ have been evaluated to compute the evidence of a model with $\tilde{M}$ intersections, they can be reused to compute the $a(\tilde{M}+1,\tilde{K})$. The arrows indicate which $a(\tilde{M},\tilde{K})$ enter into the calculation for an $a(\tilde{M}+1,\tilde{K})$.

$$\begin{aligned}
&= \sum_{k_0=0}^{K-1-M} \cdots \sum_{i=0}^{K-3-(k_{M-3}+1)} \frac{(1+i)!}{i!1!} \\
&= \sum_{k_0=0}^{K-1-M} \cdots \sum_{k_{M-3}=k_{M-4}+1}^{K-4} \binom{K-2-k_{M-3}}{K-4-k_{M-3}} \\
&= \binom{K-1}{K-M-1} = \binom{K-1}{M} \qquad (5.9)
\end{aligned}$$

This is of course just the number of possibilities of distributing M ordered bin boundaries across $K-1$ places. Due to the assumed independence between P_m and k_m, it is possible to identify

$$p(\{P_m\}|M) = M! \qquad (5.10)$$

$$P(\{k_m\}|M) = \frac{(K-M-1)!M!}{(K-1)!} \qquad (5.11)$$

and thus the prior is

$$c(M) = \frac{(K-M-1)!M!^2}{(K-1)!} \qquad (5.12)$$

5.4 Computing the evidence

To compute (5.3), rewrite it as

$$P(D|M) = \sum \sum\nolimits_{\{k\}} P(D|\{k_m\},M)P(\{k_m\}|M) \qquad (5.13)$$

where

$$P(D|\{k_m\},M) = \int_0^1 d\vec{P} \prod_{m=0}^{M} \left(\frac{P_m}{\Delta k_m}\right)^{n_m} p(\{P_m\}|M) \tag{5.14}$$

Given a configuration of the $\{k_m\}$, the $\{n_m\}$ are fixed, and thus the integrals can be carried out (see (D.8) and 5.10) to yield the normalization integral of a Dirichlet distribution:

$$\begin{aligned} P(D|\{k_m\},M) &= \prod_{m'=0}^{M} (k_{m'} - k_{m'-1})^{-n_{m'}} \times \frac{\prod_{m=0}^{M} n_m!}{(N+M)!} M! \\ &= \frac{M!}{(N+M)!} \prod_{m=0}^{M} \frac{n_m!}{\Delta k_m^{n_m}} \end{aligned} \tag{5.15}$$

For a speedy evaluation of the sums in (5.13), consider the following iterative scheme: let

$$a(0,\tilde{K}) = \frac{n_0!}{(\tilde{K}+1)^{n_0}} \tag{5.16}$$

where n_0 is the total number of data points for which $k \leq \tilde{K}$ (i.e. the number of points in the current bin 0). Furthermore, define

$$a(\tilde{M}+1,\tilde{K}) = \sum_{\tilde{k}=\tilde{M}}^{\tilde{K}-1} a(\tilde{M},\tilde{k}) \frac{n_{\tilde{M}+1}!}{(\tilde{K}-\tilde{k})^{n_{\tilde{M}+1}}} \tag{5.17}$$

where $n_{\tilde{M}+1}$ is the total number of data points for which $\tilde{k} < k \leq \tilde{K}$ (i.e. the number of points in the current bin $\tilde{M}+1$). In other words, to compute the contribution to (5.13) which has $\tilde{M}+1$ bin boundaries in the interval $0,\ldots,\tilde{K}$, let boundary $\tilde{M}+1$ move between position $\tilde{M}$ (because the previous $\tilde{M}$ boundaries must at least occupy the positions $0,\ldots,\tilde{M}-1$) and $\tilde{K}-1$ (because bin $\tilde{M}+1$ must at least have width 1). For each of these positions, multiply the factor for bin $\tilde{M}+1$ (which ranges from $\tilde{k}+1$ to $\tilde{K}$) with the contribution for $\tilde{M}$ bin boundaries in the interval $0,\ldots,\tilde{k}$ and add.

By induction, one hence obtains

$$\begin{aligned} a(\tilde{M},\tilde{K}) &= \sum_{k_{\tilde{M}-1}=\tilde{M}-1}^{\tilde{K}-1} \sum_{k_{\tilde{M}-2}=\tilde{M}-2}^{k_{\tilde{M}-1}-1} \cdots \sum_{k_0=0}^{k_1-1} \prod_{m=0}^{M} \frac{n_m!}{\Delta k_m^{n_m}} \\ &= \sum_{k_0=0}^{\tilde{K}-\tilde{M}} \cdots \sum_{k_{\tilde{M}-1}=k_{\tilde{M}-2}+1}^{\tilde{K}-1} \prod_{m=0}^{M} \frac{n_m!}{\Delta k_m^{n_m}} \end{aligned} \tag{5.18}$$

Inserting (5.15) into (5.13) and using (5.11) yields

$$P(D|M) = \frac{(K-M-1)!M!^2}{(K-1)!(N+M)!} a(M,K-1) \tag{5.19}$$

This way of organizing the calculation is computationally similar to the sum-product algorithm [49], the messages being passed from one $\tilde{M}$-level to the next are the $a(\tilde{M}, \tilde{K})$, whereas within one level, a sum over the messages from the previous level (times a factor) is performed.

To compute $a(M, K-1)$, all the $a(M-1, \tilde{K})$, $M-1 \leq \tilde{K} < K-1$ are needed (see fig. 5.2). While calculating them, $a(M-1, K-1)$ can be evaluated just as well, which does not increase the overall computation complexity. Thus the evidences for models with less than M intersections are obtained with little extra effort. Moreover, in an implementation it is sufficient to store the $a(M, \tilde{K})$ in an one-dimensional array of length K that is overwritten in reverse order as one proceeds from $M-1$ to M (because $a(M-1, K-2)$ is no longer needed once $a(M, K-1)$ is computed etc.). In pseudo-code (where the getCount(k1,k2) function returns the number of observed points for which $k1 < k \leq k2$):

```
1 Initialize a[0...K-1]:=0,evidences[0...M]:=0
2 for k:=0 to K-1 do
    (a) n:=getCount(-1,k)
    (b) a[k]:=$\frac{n!}{(k+1)^n}$
3 evidences[0]:=a[K-1]/N!
4 for m:=1 to M do
    (a) if m=M then lb:=K-1 else lb:=m
    (b) for k:=K-1 downto lb do
        i. a[k]:=0
        ii. for kk:=m-1 to k-1 do
            A. n:=getCount(kk,k)
            B. a[k]:=a[k]+a[kk]×$\frac{n!}{(k-kk)^n}$
    (c) evidences[m]:=a[K-1]×$\frac{(K-1-m)!m!^2}{(K-1)!(N+m)!}$
```

Step 4a is not essential, but it saves some computational effort: once the main loop 4 reaches M, only $a(M, K-1)$ needs to be calculated. The $a(M, k < K-1)$ would only be necessary, if the evidence for $M + 1$ bin boundaries were to be evaluated.

In a real-world implementation, it might be advisable to construct a lookup-table for $\text{getCount(k1,k2)}!/(k2 - k1)^{\text{getCount(k1,k2)}}$, since these quantities would otherwise be evaluated multiple times in step 4(b)iiB, as soon as $M > 2$.

A look at the main loop (4.) shows that the computational complexity of this algorithm is $\mathcal{O}(MK^2)$ (provided that $M \ll K$, i.e. the number of bins is much smaller than the number of possible values of X), or more generally $\mathcal{O}(K^3)$ (because $M \leq K - 1$). This is significantly faster than the naive approach of simply trying all possible configurations, which would yield $\mathcal{O}(K^M)$.

5.5 Evaluating the model posterior $P(M|D)$, the predictive distribution $P(k|M, D)$ and its variance

Once the evidence is known, one can proceed to determine the relative probabilities of different M, the *model posterior*:

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)} \tag{5.20}$$

where $P(M)$ is the model prior and $P(D) = \sum_m P(D|m)P(m)$. $P(M|D)$ is needed for model comparsion purposes. The sum over m includes all models which one chooses to include into the inference process, therefore the conditioning on a particular m is marginalized. It is thus customary to refer to $P(D)$ as 'the probability of the data'. This is somewhat misleading (as noted already in section 2.5.1), because $P(D)$ is still not only conditioned on the chosen set of m, but also on the general model class under consideration (here: probability distributions that consist of a number of bins).

The predictive distribution of X, i.e. the Bayesian estimate of the distribution which generated the data, can be calculated via the evidence as well. Note that

$$P(k_1, k_2, \ldots, k_R|M, D) = E\left[P(k_1)P(k_2)\ldots P(k_R)|M, D\right] \tag{5.21}$$

i.e. the joint predictive probability of $k_1, k_2, \ldots, k_R$ is the expectation of the product of their probabilities given M and D. Thus, if the value of the predictive distribution of X at a particular k' is to be determined, simply add this k' to the multiset D. Call this extended multiset $D' = D \cup k'$, then

$$P(k'|D,M) = \frac{P(D'|M)}{P(D|M)} \tag{5.22}$$

The choice of $P(M)$ will usually be non-informative (e.g. uniform), unless there is reason to prefer certain models over others.

Likewise, to obtain the variance, and thus a confidence interval on $P(k'|D,M)$, add k' twice: $D'' = D \cup k' \cup k'$. Then

$$\mathrm{Var}\left[P(k'|D,M)\right] = \frac{P(D''|M)}{P(D|M)} - \left(\frac{P(D'|M)}{P(D|M)}\right)^2 \tag{5.23}$$

5.6 Inferring probability densities

The algorithm can also be used to estimate probability densities. To do so, replace all occurrences of Δk_m with $(\Delta k_m)\Delta x$, where Δx is the interval between k and $k+1$. This yields a discretized approximation to the density. The discretization is just another model parameter which can be determined from the data:

$$P(\Delta x|D) = \frac{\sum_M p(D|M,\Delta x)P(M)P(\Delta x)}{\sum_{\Delta x}\sum_M p(D|M,\Delta x)P(M)P(\Delta x)} \tag{5.24}$$

where $\sum_{\Delta x}$ runs over all possible values of Δx which one chooses to include. $p(D|\Delta x)$ is computed in the same fashion as $P(D)$ in (5.20) for a given Δx, except that the data are now assumed to be continuous, hence the probability turns into a density.

The dependence of $P(D|M,\Delta x)$ on Δx is through K (see (5.3)): if one tries to find a suitable discretization of the interval $[0,b]$, then $K = \frac{b}{\Delta x}$.

5.7 Model selection versus model averaging

Once $P(M|D)$ is determined, it can be used to choose M. A fairly common procedure is to select the M for which $P(M|D)$ is maximized, usually referred to as MAP. It is closely related to the maximum likelihood approach (the only difference being that now the posterior is maximized, not the likelihood) and suffers from similar deficiencies. Strictly speaking, selecting a single model versus all others is only permitted if $P(M|D) = 1$ for this model and $P(M|D) = 0$ for the rest. It is, however, unlikely that this situation will be encountered (unless in some limiting cases, for which one needs not bother to employ probability theory at all). Rather, there will be various M with nonzero probability. From a Bayesian point of view, it is then required to include all those (denoted by the set $\{M\}$) into any prediction, weighted by their corresponding model posteriors. The predicted probability for $X = k$ then becomes

$$\begin{aligned} P(k|D, \{M\}) &= \sum_{m \in \{M\}} P(k|D, m) P(m|D) \\ &= \sum_{m \in \{M\}} \frac{P(D'|m)}{P(D|m)} \frac{P(D|m)P(m)}{\sum_{m' \in \{M\}} P(D|m')P(m')} \\ &= \frac{\sum_{m \in \{M\}} P(D'|m)P(m)}{\sum_{m \in \{M\}} P(D|m)P(m)} \end{aligned} \tag{5.25}$$

The drawback of this scheme is that, if $|\{M\}|$ is very large, it might take a long time to carry out the necessary computations. Therefore, one would like to be able to reduce the number of models that need to be taken into account.

One possible approach is based on probabilistic considerations. Suppose one splits the set $\{M\}$ in two parts, $\{M\}_i$ containing those models which are to be included, and $\{M\}_e = \{M\} \backslash \{M\}_i$. The probability of each set given the data can then be computed:

$$P(\{M\}_i|D) = \frac{\sum_{m \in \{M\}_i} P(D|m)P(m)}{\sum_{m \in \{M\}} P(D|m)P(m)} \tag{5.26}$$

$$P(\{M\}_e|D) = 1 - P(\{M\}_i|D) \tag{5.27}$$

(The implicit conditioning on the model class has again been omitted). Now, in analogy to the significance levels of orthodox statistics, choose an α, which is to represent the probability of accidentally rejecting models although they provide a good description of the data. Then, $\{M\}_i$ would be constructed so that

$$P(\{M\}_i|D) \geq 1 - \alpha \tag{5.28}$$

The straightforward way of doing this is to start with an empty set and iteratively add the models from $\{M\}_e$ that lead to a maximum increase of $P(\{M\}_i|D)$.

Another approach is to look at the entropy H of the distribution comprised of $P(\{M\}_i|D)$ and $P(\{M\}_e|D)$. Measured in bits, it has to be in the interval $[0, 1]$. Entropy is the most general measure of uncertainty of a random variable. It therefore tells how much information one has to 'contrive' when one chooses one set and discards the other. One would thus keep adding models to $\{M\}_i$ until the entropy is low enough (using the same scheme as before). What exactly 'low enough' means, is, as in the case of significance levels, a matter of convention.

These two methods of model selection are of course closely related to each other, via

$$H = \alpha \log_2(\alpha) + (1-\alpha)\log_2(1-\alpha) \tag{5.29}$$

Provided that $P(\{M\}_i|D) \geq 0.5$, it is thus possible to compute an entropy for each α, and vice versa.

5.8 Computing the entropy and its variance

To evaluate the entropy of the distribution $P(X)$, the entropy of (5.1) is computed first:

$$\begin{aligned} H(P(X|M,\{P_m,k_m\})) &= -\sum_{k=0}^{K-1} P(k|M,\{P_m,k_m\})\log(P(k|M,\{P_m,k_m\})) \\ &= -\sum_{m=0}^{M}\sum_{k=k_{m-1}+1}^{k_m}\frac{P_m}{\Delta k_m}\log\left(\frac{P_m}{\Delta k_m}\right) \\ &= -\sum_{m=0}^{M} P_m\log(P_m) + \sum_{m=0}^{M} P_m\log(\Delta k_m) \end{aligned} \tag{5.30}$$

where $\log(x)$ is the natural logarithm. This expression must now be averaged over the posterior distributions of $\{(P_m,k_m)\}$ and possibly M to obtain the expectation of $H(P(X|D))$. Instead of carrying this out for (5.30) as a whole, it is easier to do it term-by-term, i.e. to calculate the expectations of $P_m\log(P_m)$ and $P_m\log(k_m - k_{m-1})$. Generally speaking, if one wants to compute the average of any quantity that is a function of the probability in a bin m' for a given M, one can proceed in the following fashion: call this function $f(P_{m'})$. Its expectation w.r.t. the $\{P_m\}$is then

$$E\left[f(P_{m'})|\{k_m\},M,D\right] = \frac{\int_0^1 d\vec{P} f(P_{m'})P(D|\{(P_m,k_m\},M)}{I(\{k_m\})} \tag{5.31}$$

where, by virtue of (5.14) and (D.8)

$$I(\{k_m\}) = \int_0^1 d\vec{P} P(D|\{(P_m,k_m\},M)$$

$$= \frac{1}{(N+M)!}\prod_{m=0}^{M}\frac{n_m!}{\Delta k_m^{n_m}} \tag{5.32}$$

because $p(\{P_m\})$ is a constant, otherwise it would have to be included in the integrations. Note that, as far as the counts in the bins n_m are concerned, $E\left[f(P_{m'})|\{k_m\},M,D\right]$ depends only on $n_{m'}$ (and possibly N, the total number of datapoints). This can be verified by integrating the numerator of (5.31) over all $P_m, m \neq m'$. Therefore

$$\begin{aligned} E\left[f(P_{m'})|M,D\right] &= \sum\sum\nolimits_{\{k\}}\int_0^1 d\vec{P} f(P_{m'})p(\{(P_m,k_m)\}|M,D) \\ &= \frac{\sum\sum_{\{k\}}P(\{k_m\})p(\{P_m\})E\left[f(P_{m'})|\{k_m\},M,D\right]I(\{k_m\})}{P(D|M)} \end{aligned} \tag{5.33}$$

When (5.32) is inserted here, it becomes apparent that this expectation has the same form as (5.13) after inserting (5.15). Combined with the fact that $E\left[f(P_{m'})|\{k_m\},M,D\right]$ depends only on $n_{m'}$, this means that the above described iterative computation scheme (eqns. (5.16) and (5.17)) can be employed for its evaluation. All one needs to do is to substitute $n_{m'}! \rightarrow n_{m'}!E\left[f(P_{m'})|\{k_m\},M,D\right]$, i.e. (5.17) is replaced by

$$a(\tilde{M}+1=m',\tilde{K}) = \sum_{\tilde{k}=\tilde{M}}^{\tilde{K}-1} a(\tilde{M},\tilde{k})\frac{n_{m'}!}{\Delta k_{m'}^{n_{m'}}}E\left[f(P_{m'})|\{k_m\},M,D\right] \tag{5.34}$$

where $\Delta k_{m'} = \tilde{K}-\tilde{k}$. The $a(\tilde{M},\tilde{K})$ for $\tilde{M} \neq m'$ are the same as before. Note that (5.34) can also be used if $f(P_{m'})$ depends not only on $P_{m'}$, but also on the boundaries of bin m' (i.e. $k_{m'-1}$ and $k_{m'}$). Generally speaking, (5.34) can be employed whenever the expectation of a function (given M and D) is to be evaluated, if this function depends only on the parameters of one bin.

For fixed $\{k_m\}$, some of these expectations have been computed before in [40].

5.8.1 Computing $E\left[P_{m'}\log(P_{m'})|\{k_m\},M,D\right]$

Using (5.1) and defining $q(\{k_m\}) = \prod_{m=0}^{M}\Delta k_m^{-n_m}$ one obtains

$$E\left[P_{m'}\log(P_{m'})|\{k_m\},M,D\right] = \frac{q(\{k_m\})\int_0^1 d\vec{P}P_{m'}^{n_{m'}+1}\log(P_{m'})\prod_{m\neq m'}P_m^{n_m}}{I(\{k_m\})} \tag{5.35}$$

To compute the integrals, note that

$$\begin{aligned} &\int_0^1 d\vec{P}P_{m'}^{n_{m'}+1}\log(P_{m'})\prod_{m\neq m'}P_m^{n_m} \\ &= \frac{\partial}{\partial n_{m'}}\int_0^1 d\vec{P}P_{m'}^{n_{m'}+1}\prod_{m\neq m'}P_m^{n_m} \end{aligned} \tag{5.36}$$

which is, by using (D.8), (D.6) and (D.30)

$$\begin{aligned}
&= \frac{\partial}{\partial n_{m'}} \frac{\Gamma(n_{m'}+2)\prod_{m\neq m'}^{M}\Gamma(n_m+1)}{\Gamma(\sum_{m=0}^{M} n_m + M + 2)} \\
&= \frac{\partial}{\partial n_{m'}} \frac{\Gamma(n_{m'}+2)\Gamma(\sum_{m\neq m'} n_m + M)}{\Gamma(\sum_{m=0}^{M} n_m + M + 2)} \frac{\prod_{m\neq m'}^{M}\Gamma(n_m+1)}{\Gamma(\sum_{m\neq m'} n_m + M)} \\
&= \frac{\partial B(n_{m'}+2, \sum_{m\neq m'} n_m + M)}{\partial n_{m'}} \frac{\prod_{m\neq m'}^{M}\Gamma(n_m+1)}{\Gamma(\sum_{m\neq m'} n_m + M)} \\
&= \frac{\prod_{m=0}^{M} n_m!}{(N+M)!} \frac{n_{m'}+1}{N+M+1} h_{n_{m'}+2}^{N+M+1} \qquad (5.37)
\end{aligned}$$

where $B(a, b)$ is the Beta function, $\Gamma(a)$ is the Gamma function (see appendix D for details), and

$$h_a^b = \sum_{i=a}^{b} \frac{1}{i} \qquad (5.38)$$

is the difference between the partial sums of the harmonic series with upper limits $a-1$ and b. The first part of (5.37) is the normalization integral of the density of the P_m. Hence, their entropy for fixed $\{k_m\}$ is given by (5.37) divided by (D.8)

$$E[H(\{P_m\})|\{k_m\}, M, D] = \sum_{m=0}^{M} \frac{n_m+1}{N+M+1} h_{n_m+2}^{N+M+1} \qquad (5.39)$$

which is a rational number. In other words, entropy changes in rational increments as we observe new data points.

Thus,

$$E[P_{m'}\log(P_{m'})|\{k_m\}, M, D] = \frac{n_{m'}+1}{N+M+1} h_{n_{m'}+2}^{N+M+1} \qquad (5.40)$$

5.8.2 Computing $E[P_{m'}\log(\Delta k_{m'})|\{k_m\}, M, D]$

Here,

$$E[P_{m'}\log(\Delta k_{m'})|\{k_m\}, M, D] = \frac{q(\{k_m\})\log(\Delta k_{m'})\int_0^1 d\vec{P} P_{m'}^{n_{m'}+1}\prod_{m\neq m'} P_m^{n_m}}{I(\{k_m\})} \qquad (5.41)$$

and, by using the same identities as above,

$$E[P_{m'}\log(\Delta k_{m'})|\{k_m\}, M, D] = \frac{n_{m'}+1}{N+M+1}\log(\Delta k_{m'}) \qquad (5.42)$$

5.8.3 Computing the variance

To compute the variance, the square of the entropy is required:

$$
\begin{aligned}
H^2(P(X|M,\{P_m,k_m\})) &= \left(-\sum_{k=0}^{K-1} P(k|M,\{P_m,k_m\})\log(P(k|M,\{P_m,k_m\}))\right)^2 \\
&= \left(\sum_{m=0}^{M} P_m\log(P_m) - \sum_{m=0}^{M} P_m\log(\Delta k_m)\right)^2 \\
&= \sum_{m'=0}^{M} P_{m'}^2\log^2(P_{m'}) \\
&+ 2\sum_{m'=0}^{M}\sum_{m''=m'+1}^{M} P_{m'}P_{m''}\log(P_{m'})\log(P_{m''}) \\
&+ \sum_{m'=0}^{M} P_{m'}^2\log^2(\Delta k_{m'}) \\
&+ 2\sum_{m'=0}^{M}\sum_{m''=m'+1}^{M} P_{m'}P_{m''}\log(\Delta k_{m'})\log(\Delta k_{m''}) \\
&- 2\sum_{m'=0}^{M} P_{m'}^2\log(P_{m'})\log(\Delta k_{m'}) \\
&- 4\sum_{m'=0}^{M}\sum_{m''=m'+1}^{M} P_{m'}P_{m''}\log(P_{m'})\log(\Delta k_{m''}) \qquad (5.43)
\end{aligned}
$$

Hence, one needs to evaluate the expectations of

1 $P_{m'}^2\log^2(P_{m'})$. See 5.8.4.

2 $P_{m'}P_{m''}\log(P_{m'})\log(P_{m''})$. See 5.8.5.

3 $P_{m'}^2\log^2(\Delta k_{m'})$. Can be evaluated along the same lines as (5.42) and yields

$$\frac{(n_{m'}+1)(n_{m'}+2)}{(N+M+1)(N+M+2)}\log^2(\Delta k_{m'}) \qquad (5.44)$$

4 $P_{m'}P_{m''}\log(\Delta k_{m'})\log(\Delta k_{m''})$. Follows from (D.8) by replacing $n_{m'}$ with $n_{m'}+1$, $n_{m''}$ with $n_{m''}+1$ and yields

$$\frac{(n_{m'}+1)(n_{m''}+1)}{(N+M+1)(N+M+2)}\log(\Delta k_{m'})\log(\Delta k_{m''}) \qquad (5.45)$$

5 $P_{m'}^2\log(P_{m'})\log(\Delta k_{m'})$. Can be evaluated along the same lines as (5.40) and yields

$$\frac{(n_{m'}+1)(n_{m'}+2)}{(N+M+1)(N+M+2)}h_{n_{m'}+3}^{N+M+2}\log(\Delta k_{m'}) \qquad (5.46)$$

6 $P_{m'}P_{m''}\log(P_{m'})\log(\Delta k_{m''})$. Can be evaluated along the same lines as (5.40) and yields

$$\frac{(n_{m'}+1)(n_{m''}+1)}{(N+M+1)(N+M+2)} h_{n_{m'}+2}^{N+M+2} \log(\Delta k_{m''}) \tag{5.47}$$

5.8.4 Computing $E\left[P_{m'}^2 \log^2(P_{m'}) | \{k_m\}, M, D\right]$

Since

$$\frac{\partial^2 B(a+1,b+1)}{\partial a^2} = \int_0^1 dx \log^2(x) x^a (1-x)^b \tag{5.48}$$

the same method of calculation as above can be employed. Thus, noting (D.31) one obtains:

$$E\left[P_{m'}^2 \log^2(P_{m'}) | \{k_m\}, M, D\right] = \frac{(n_{m'}+1)(n_{m'}+2)}{(N+M+1)(N+M+2)} \left[\left(h_{n_{m'}+3}^{N+M+2}\right)^2 + {}_2h_{n_{m'}+3}^{N+M+2}\right] \tag{5.49}$$

where

$${}_2h_a^b = \sum_{i=a}^{b} \frac{1}{i^2} \tag{5.50}$$

5.8.5 Computing $E\left[P_{m'}P_{m''} \log(P_{m'}) \log(P_{m''}) | \{k_m\}, M, D\right]$

To compute this expectation, one needs to evaluate

$$\begin{aligned} &\int_0^1 d\vec{P} P_{m'}^{n_{m'}+1} P_{m''}^{n_{m''}+1} \log(P_{m'}) \log(P_{m''}) \prod_{m \neq m', m''} P_m^{n_m} \\ = &\frac{\partial^2}{\partial n_{m'} \partial n_{m''}} \left(\int_0^1 d\vec{P} P_{m'}^{n_{m'}+1} P_{m''}^{n_{m''}+1} \prod_{m \neq m', m''} P_m^{n_m} \right) \end{aligned} \tag{5.51}$$

and then divide it by a normalization constant (see (D.8)). The integrals in the last expression can be rewritten, using Beta functions, to yield

$$B(n_{m'}+2, n_{m''}+2) B(n_{m'}+n_{m''}+4, N_R+M-1) \times R \tag{5.52}$$

where $N_R = \sum_{m \neq m', m''} n_m$ and R is the part that does not depend upon $n_{m'}$ and $n_{m''}$ and thus is not affected by the differentiation:

$$R = \frac{\prod_{m \neq m', m''} n_m!}{(N_R+M-2)!} \tag{5.53}$$

Using (D.30), (D.31) and (D.32) yields

$$
\begin{aligned}
& \frac{\partial^2}{\partial n_{m'} \partial n_{m''}} B(n_{m'}+2, n_{m''}+2) B(n_{m'}+n_{m''}+4, N_R+M-1) \\
= \; & \frac{\partial^2 B(n_{m'}+2, n_{m''}+2)}{\partial n_{m'} \partial n_{m''}} B(n_{m'}+n_{m''}+4, N_R+M-1) \\
+ \; & \frac{\partial B(n_{m'}+2, n_{m''}+2)}{\partial n_{m'}} \frac{\partial B(n_{m'}+n_{m''}+4, N_R+M-1)}{\partial n_{m''}} \\
+ \; & \frac{\partial B(n_{m'}+2, n_{m''}+2)}{\partial n_{m''}} \frac{\partial B(n_{m'}+n_{m''}+4, N_R+M-1)}{\partial n_{m'}} \\
+ \; & \frac{\partial^2 B(n_{m'}+n_{m''}+4, N_R+M-1)}{\partial n_{m'} \partial n_{m''}} \\
= \; & \left(h_{n_{m'}+2}^{n_{m'}+n_{m''}+3}\right)\left(h_{n_{m''}+2}^{n_{m'}+n_{m''}+3}\right) + {}_2h_1^{n_{m'}+n_{m''}+3} - \frac{\pi^2}{6} \\
+ \; & \left(h_{n_{m'}+2}^{n_{m'}+n_{m''}+3}\right)\left(h_{n_{m'}+n_{m''}+4}^{N+M+2}\right) \\
+ \; & \left(h_{n_{m''}+2}^{n_{m'}+n_{m''}+3}\right)\left(h_{n_{m'}+n_{m''}+4}^{N+M+2}\right) \\
+ \; & \left(h_{n_{m'}+n_{m''}+4}^{N+M+2}\right)^2 + {}_2h_{n_{m'}+n_{m''}+4}^{N+M+2} \\
= \; & \left(h_{n_{m'}+2}^{n_{m'}+n_{m''}+3} + h_{n_{m'}+n_{m''}+4}^{N+M+2}\right)\left(h_{n_{m''}+2}^{n_{m'}+n_{m''}+3} + h_{n_{m'}+n_{m''}+4}^{N+M+2}\right) \\
+ \; & {}_2h_1^{N+M+2} - \frac{\pi^2}{6} \\
= \; & \left(h_{n_{m'}+2}^{N+M+2}\right)\left(h_{n_{m''}+2}^{N+M+2}\right) + {}_2h_1^{N+M+2} - \frac{\pi^2}{6} \qquad (5.54)
\end{aligned}
$$

The desired expectation can thus be computed in two runs of the iteration: first, let

$$E\left[f(P_{m'})|\{k_m\}, M, D\right] = (n_{m'}+1)h_{n_{m'}+2}^{N+M+2} \qquad (5.55)$$

$$E\left[f(P_{m''})|\{k_m\}, M, D\right] = (n_{m''}+1)h_{n_{m''}+2}^{N+M+2} \qquad (5.56)$$

and divide the result by $(N+M+1)(N+M+2)$.

Second, let

$$E\left[f(P_{m'})|\{k_m\}, M, D\right] = (n_{m'}+1) \qquad (5.57)$$

$$E\left[f(P_{m''})|\{k_m\}, M, D\right] = (n_{m''}+1) \qquad (5.58)$$

and multiply the result with $\frac{{}_2h_1^{N+M+2} - \frac{\pi^2}{6}}{(N+M+1)(N+M+2)}$.

5.9 An information-theoretic similarity measure

In the next section, the above described algorithm will be shown in action. To that end, data were generated from some known densities, which were then used for inference. To test the algorithm's performance, a distance measure between densities (or distributions) will be used, which is motivated in the following way: suppose an impartial observer was told that either $p(x)$ or $q(x)$ was the density which generated the data. Given no more information, the prior probabilities for each of them being the correct density would then be assigned as $P(p(x)) = P(q(x)) = \frac{1}{2}$. Upon observing a datapoint x_1, this knowledge would be updated via Bayes' rule, giving rise to the posterior

$$P(p(x)|x_1) = \frac{p(x_1)P(p(x))}{p(x_1)P(p(x)) + q(x_1)P(p(x))} = \frac{p(x_1)}{p(x_1) + q(x_1)} \tag{5.59}$$

$$P(q(x)|x_1) = \frac{q(x_1)P(q(x))}{p(x_1)P(p(x)) + q(x_1)P(p(x))} = \frac{q(x_1)}{p(x_1) + q(x_1)} \tag{5.60}$$

The amount of information thus received can be quantified by the Kullback divergence (see section 2.5.3) between posterior and prior:

$$\begin{aligned} D &= P(p(x)|x_1)\log\left(\frac{P(p(x)|x_1)}{P(p(x))}\right) + P(q(x)|x_1)\log\left(\frac{P(q(x)|x_1)}{P(q(x))}\right) \\ &= \frac{p(x_1)}{p(x_1)+q(x_1)}\log\left(\frac{2p(x_1)}{p(x_1)+q(x_1)}\right) \\ &+ \frac{q(x_1)}{p(x_1)+q(x_1)}\log\left(\frac{2q(x_1)}{p(x_1)+q(x_1)}\right) \end{aligned} \tag{5.61}$$

The observer may now ask 'How much can I expect to learn from observing x_1, given what I already know?'. This expected information gain is obtained by averaging eqn. (5.61) over the expected distribution of x_1 (given the prior knowledge, i.e. before the observation of x_1 is made), which is $p(x_1)P(p(x)) + q(x_1)P(q(x))$:

$$\begin{aligned} D^2_{pq} &= \int dx_1 \frac{1}{2}(p(x_1) + q(x_1))D \\ &= \frac{1}{2}\int dx_1 p(x_1)\log\left(\frac{2p(x_1)}{p(x_1)+q(x_1)}\right) + q(x_1)\log\left(\frac{2q(x_1)}{p(x_1)+q(x_1)}\right) \\ &= \frac{1}{2}\left(D\left(p(x)\left|\left|\frac{1}{2}(p(x)+q(x))\right.\right.\right) + D\left(q(x)\left|\left|\frac{1}{2}(p(x)+q(x))\right.\right.\right)\right) \end{aligned} \tag{5.62}$$

In other words, the expected information gain is the average of the Kullback divergences between $p(x)$ and $q(x)$ and the expected distribution of x given the prior. It is symmetric in $p(x)$ and $q(x)$ and vanishes if $p(x) = q(x)$. Moreover, it was shown in [22], that its square root fulfills the triangle inequality (hence the square in eqn. (5.62)), i.e. D_{pq} is a metric. The proof, along with a coding-theroetic motivation and some of its properties, can be found in appendix E. For the current purpose, the most important of these are:

- The maximum of D^2_{pq}, measured in bit, is 1. It is reached when $p(x)$ and $q(x)$ are distinct, i.e. such that a sample drawn from one density could not possibly have come from the other. In this case, one datapoint is enough to decide which density generated it.

- $N_{min} = \frac{1}{D^2_{pq}}$ (D^2_{pq} again measured in bit) is a lower bound on the sample size needed to ascertain which density gave rise to the data.

5.10 Examples

Fig. 5.3 shows examples of the predictive density and its variance, compared to the density from which the data points were drawn ($K = 100, M = 5$, data point abscissas were rounded to the next lower discretization point), as well as the model posteriors $P(M|D)$. The prior $P(M)$ was chosen uniform over the maximum range of M, here $M = 0, \ldots, 99$[2]. Inference was conducted with $\alpha = 0.01$ (see eqn. 5.28). Note that the curves that represent the expected density plus/minus one standard deviation are not densities anymore. Furthermore, probability densities do not need to be ≤ 1, but they have to be normalized, i.e. the integral of the density over the whole domain of the random variable which it describes must be 1. The data were produced by first drawing uniform random numbers with the generator sran2() from [76], those were then transformed by the inverse cumulative density (a method also described in [76]) to be distributed according to the desired density. For very small datasets (fig. 5.3, top left), only the largest structures of the distribution can vaguely be seen (such as the valley between 0.15 and 0.58). Furthermore, the density peaks at each data point (at 0.7, two points were observed very close to each other). One might therefore imagine the process by which the density comes about as similar to kernel-based density estimates. It does, however, differ from a kernel-based procedure insofar as that the number of degrees of freedom does not necessarily grow with the dataset, but is determined by the data's structure. The model posterior (fig. 5.3, top right) is very broad, reflecting the large remaining

[2]On a 2.6Ghz Pentium 4 system running SuSE Linux 8.2, computing the evidence took $\approx 0.26s$ (without initializations). The algorithm was implemented in C++ and compiled with the GnU compiler.

uncertainties due to the small dataset. Models with $0 \leq M \leq 97$ were included in the predictions.

With 100 data points (fig. 5.3, second row), more structures begin to emerge (e.g. the high plateau between 0.58 and 0.68), and the variance decreases, as one would expect. The model posterior exhibits a much narrower maximum, here the included models were those with $4 \leq M \leq 17$.

At 1000 data points (fig. 5.3, third row), all structures of the original density are modelled, and the variance is yet smaller. The algorithm is now fairly certain about the correct number of intersections, $5 \leq M \leq 8$, with a maximum at 5. Moreover, due to this restriction on the degrees of freedom, the predicted density no longer looks like a kernel-based estimate. It should be noted that in a small number of instances ($\approx 2.6\%$, estimated from results on 1000 datasets), the algorithm fails to discover the peak at 0.14, and consequently the posterior maximum is at $M = 4$. This can happen if, due to random fluctuation, a dataset does not contain enough points in the vicinity of the peak to justify a separate bin at this location.

At 10000 data points (fig. 5.3, bottom row), all structures of the distribution are faithfully modelled and the variance of the predicted density is nearly zero.

Fig. 5.4 depicts density estimates from a mixture of two Gaussians. Even though these densities can – strictly speaking – not be modelled anymore by a finite number of bins, the method still produces fairly good estimates of the discretized ($K = 100$) densities. Observe how the maximum of the posterior (fig. 5.4, right column) shifts towards higher values as the number of data points grows. The algorithm thus picks a range of M which, depending on the amount of available information, is best suited for explaining the underlying structure of the data.

For a more quantitative representation of the relationship between N and the 'closeness' of expected and true densities, fig. 5.5, black diamonds, shows the value of D^2_{pq} where $p(x)$ is the expected density and $q(x)$ is the discretized true density. This expected information gain goes to zero, following approximately a power law in N (with exponent ≈ -1 for the bin density and ≈ -0.72 for the mixture of Gaussians). Therefore, given a large enough dataset, expected and true density cannot be distinguished anymore. Moreover, in the case of the bin density, the gain is inversely proportional to the number of data points. In other words, the decision as to which density the sample was drawn from becomes twice as hard when N is doubled.

As noted above, $\frac{1}{D^2_{pq}}$ is a lower limit on the number of datapoints which are needed to determine which density is the generating one. For the bin density, this lower limit is roughly equal to N, i.e. an impartial observer would need as many datapoints to decide between $p(x)$ and $q(x)$ as were used to infer $p(x)$. In the case of the mixture of Gaussians, the minimum number of datapoints is somewhat lower, but still growing with N.

Furthermore, fig. 5.4 also shows the performances of two kernel-based density estimation methods: a mixture of (at most) 20 Gaussians (MOG) (red circles) and a Dirichlet process [23] mixture of Gaussians (DP-MOG) (green squares). The latter can be understood as the former in the limit of infinitely many mixture components, and should thus be able to model virtually any density. I used the implementation provided in the package *fbm-2004-11-10* [60]. When the true density is comprised of bins, the BBDIa outperforms the other two methods significantly. This might be expected, since the BBDIa can model the true density exactly, whereas the other two methods cannot. What is somewhat remarkable is the BBDIa performance when the true density is a mixture of two Gaussians. Now the MOG and the DP-MOG outperform the BBDIa, since they are able to model the true density exactly. However, their performance gain is never quite as high as that of the BBDIa in the bin density scenario. Moreover, for relatively small datasets (up to ≈ 100 points) which are customary in neurophysiological experiments, the BBDIa performance is comparable to that of the other two methods.

In fig. 5.6, the expected differential entropies (see section 2.4.3) are plotted as a function of the number of data points. Error bars represent ± 1 expected standard deviation. Both expectations were computed individually for each data set and then averaged over 100 runs. In every case, the true entropy is well within the error bars. For $N \geq 100$, the expected entropy is fairly close to its true value, thus eliminating the need for finite-size corrections. Note that the standard deviation of the entropy is plotted, *not* the empirical standard deviation of its expectation, i.e. the error bars serve as an indication of the remaining uncertainty in the entropy, which goes to 0 as N increases. This is due to the fact that entropy is treated here as a deterministic quantity, not as a random variable (because samples are drawn from some fixed, albeit possibly unknown, density). To illustrate this point further, look at fig. 5.7: For small datasets, the standard deviation of the expectation of the entropy is somewhat smaller than the standard deviation of the entropy. When evaluating experimental results, using the former instead of the latter would thus possibly mislead one to believe in a higher accuracy of the results than can be justified by the data, a danger that is especially preeminent when using methods such as cross-validation in place of exact calculations.

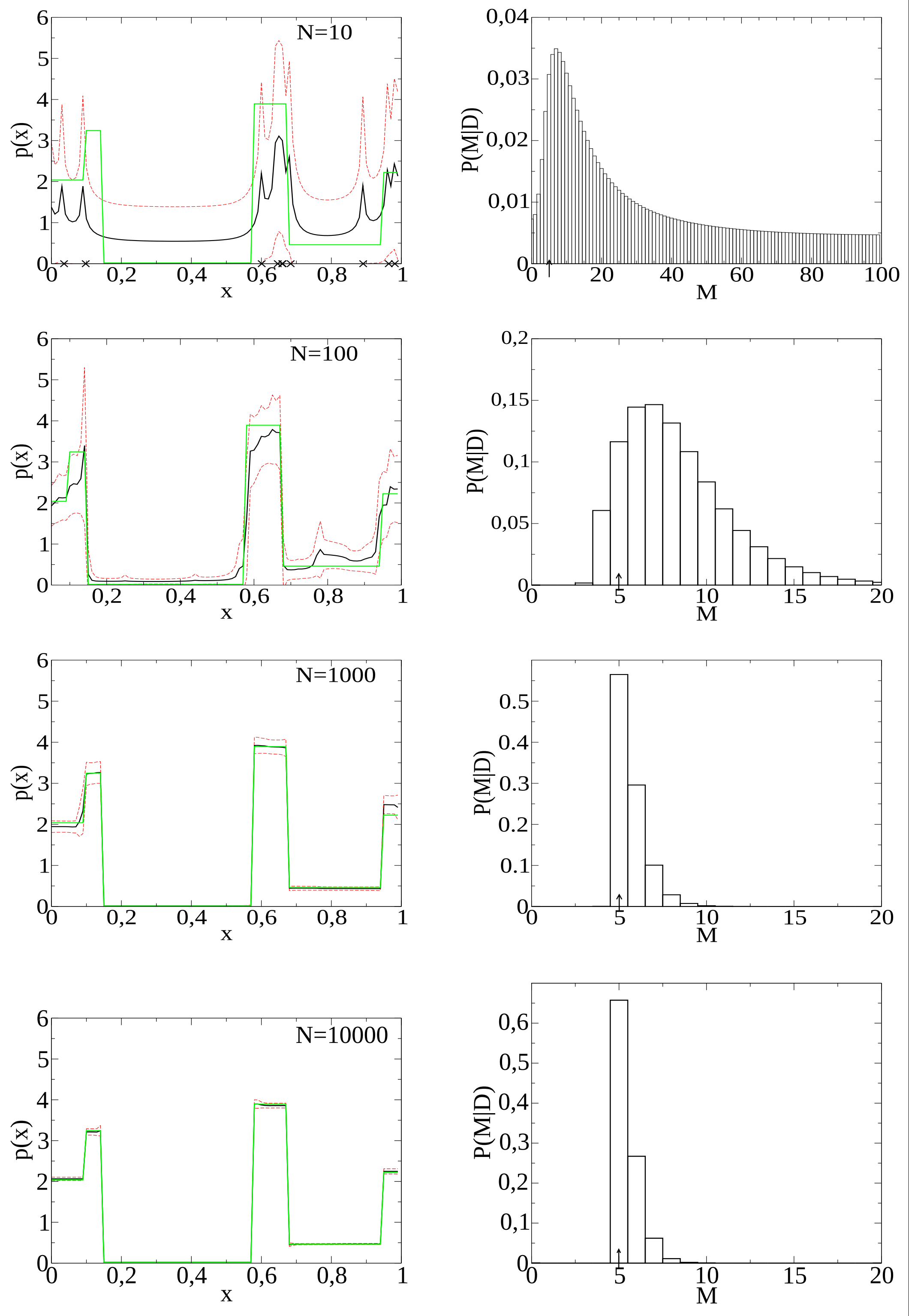


Figure 5.3: From top to bottom: densities and posteriors for 10,100,1000 and 10000 data points. Left column: expected probability density (black) plus/minus one standard deviation (red), compared to the true density (green). Left column: posterior distribution of M, the number of intersections. The arrows indicate the true value of $M = 5$. The X's on the abscissa of the graph for 10 data points mark the data points.

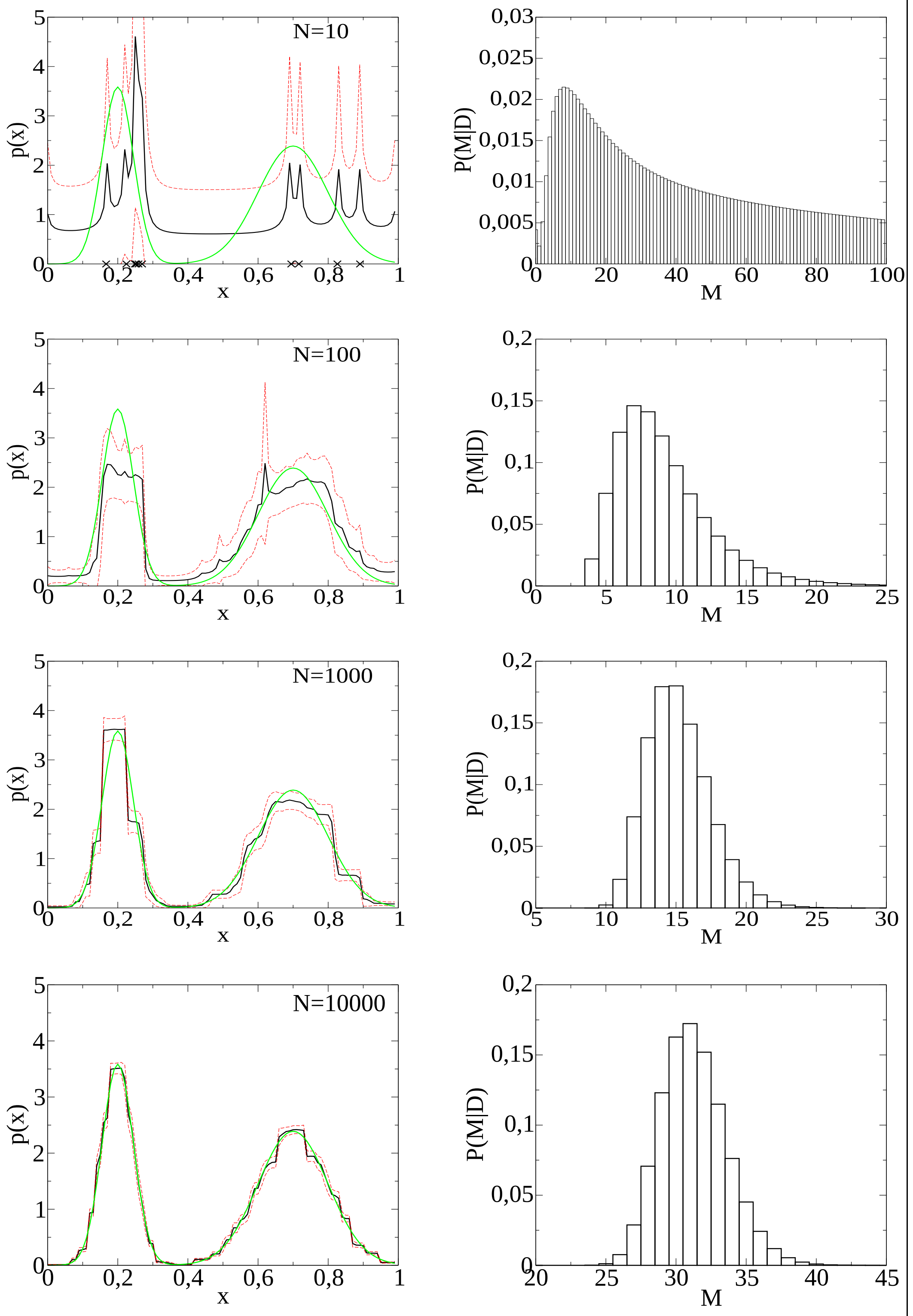
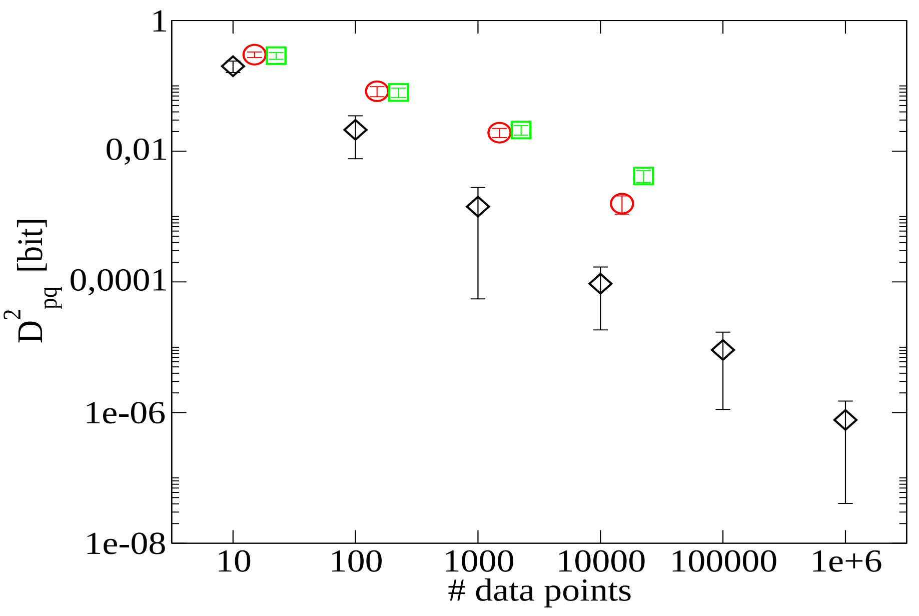


Figure 5.4: From top to bottom: densities and posteriors for 10,100,1000 and 10000 data points. Left column: expected probability density (black) plus/minus one standard deviation (red), compared to the true density (green). Left column: posterior distribution of M, the number of intersections. The X's on the abscissa of the graph for 10 data points mark the data points.



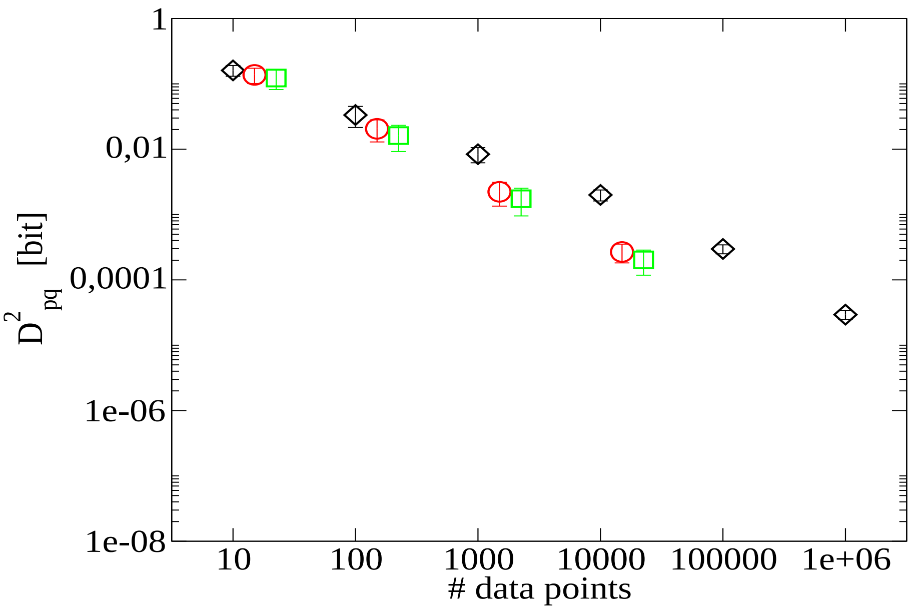
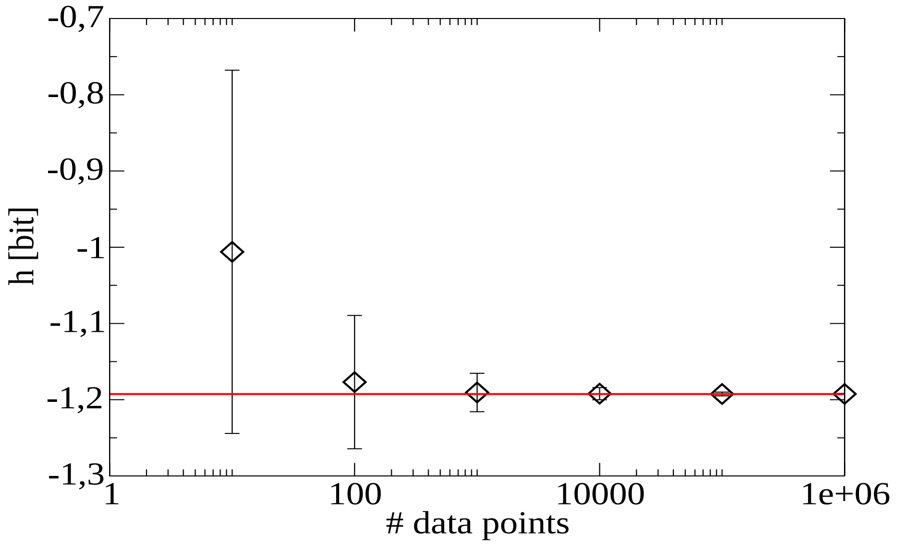


Figure 5.5: D^2_{pq} between true and expected density as a function of the number of data points. The error bars were obtained from averaging over 100 different datasets drawn from the same density. Top: 5-bin density, bottom: mixture of 2 Gaussians. Black diamonds: BBDI, red circles: mixture of (at most) 20 Gaussians, green squares: Dirichlet process mixture of Gaussians. For details, see text.



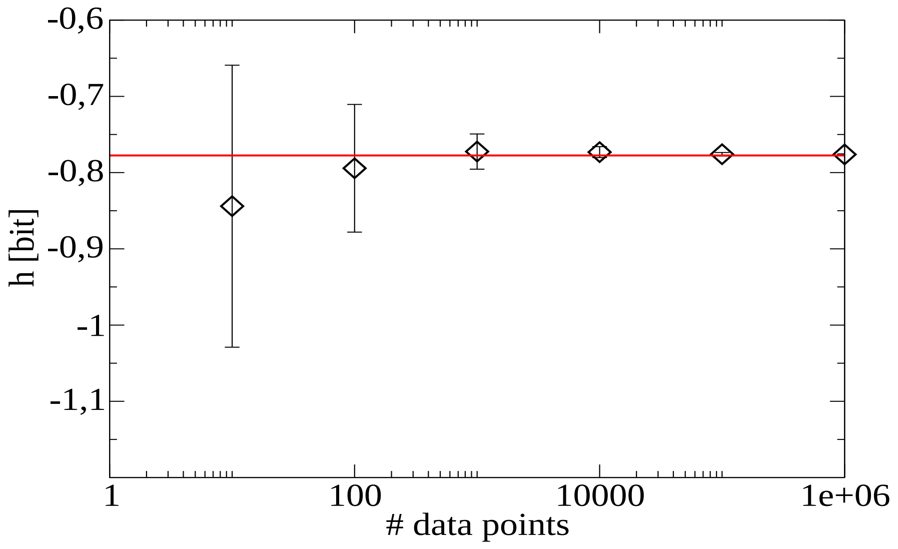
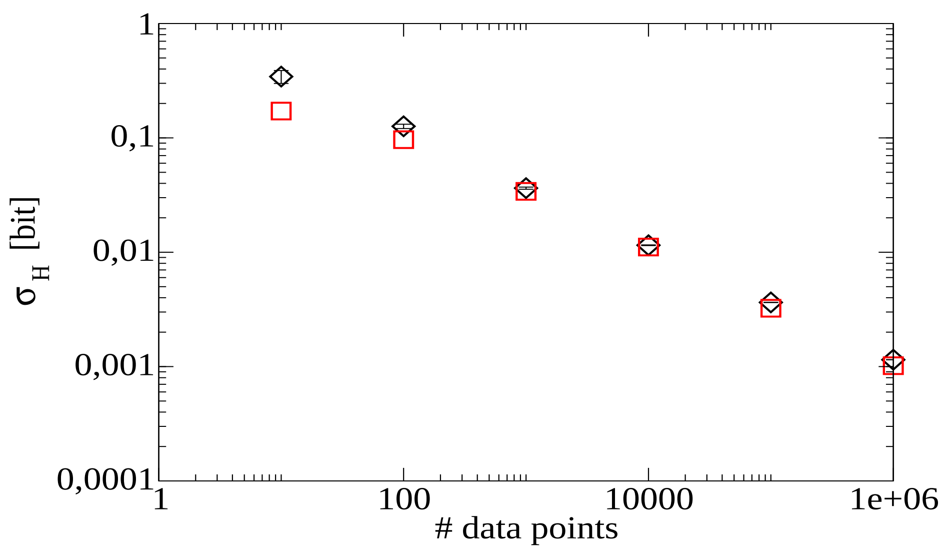


Figure 5.6: Expected differential entropy (diamonds) and expected standard deviation (error bars) as a function of the number of data points. Both expectations were computed for one dataset at a time, then averaged over 100 datasets. The red line represents the true differential entropy. Top: 5-bin density, bottom: mixture of 2 Gaussians.



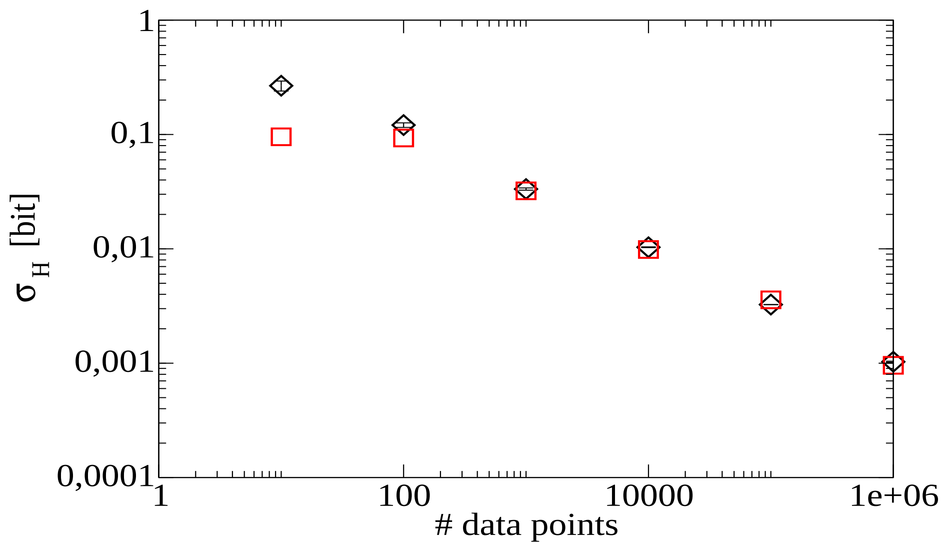
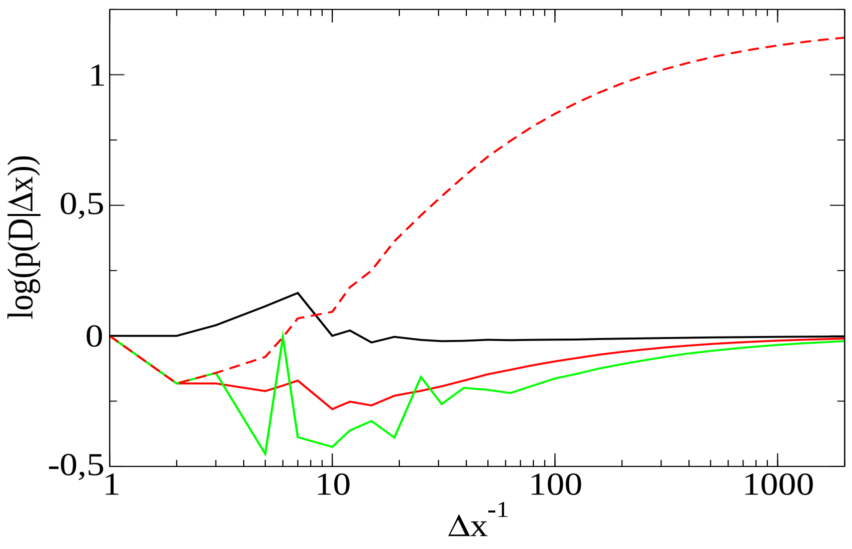


Figure 5.7: Expected standard deviation (black diamonds) ± 1 standard deviation and empirical standard deviation (red squares) of the expectation of the differential entropy as functions of the number of data points. Error bars and empirical standard deviations are averages over 100 datasets. Top: 5-bin density, bottom: mixture of 2 Gaussians.

5.10.1 The limit $K \to \infty$

Assume one tried to find the best discretization of the data via (5.24). If there is no lower limit on Δx (other than 0) given by the problem being studied (e.g. accuracy limits of the equipment used to record the data), then we might wonder how the discretization posterior behaves as $\Delta x \to 0$ or, conversely, as $K \to \infty$. In this case, given that N stays finite, the data's probability density will diverge. However, since the number of possible bin configurations also diverges, it is not clear whether $p(D|\Delta x)$ stays finite or not. I will not try to give an analytic answer to this question here. Instead, look at fig. 5.8, top: the solid lines show the log evidence $\log(p(D|\Delta x))$ as a function of Δx^{-1}. To get an impression of the limit $K \to \infty$, $N \ll K$, the datasets contained one (black), two (red) and three (green) distinct datapoints, drawn from the interval $[0, 1]$. The datasets were constructed in the following fashion: three distinct points x_1, x_2, x_3 were drawn. Then $D_1 = \{x_1\}$, $D_2 = \{x_1; x_2\}$ and $D_3 = \{x_1; x_2; x_3\}$, i.e. the dataset with one point was a subset of the datasets with two and three points, the two-point dataset was a subset of the three-point dataset. It appears that the log evidence is converging to zero from below as $K \to \infty$. Most importantly, the maximum of the log evidence is in every case located at a small value of K. One might wonder if and how the location of this maximum depends on N. The answer depends of course on the structure of the generating density. In fig. 5.8, bottom, the log evidence is plotted for $N = 1000$ and a generating density which consisted of 5 bins, with the bin boundaries b_i chosen so that $b_i \times 20$ was an integer. Consequently, the best discretization is found to be $\Delta x = \frac{1}{20}$. This value of Δx is ≈ 90 times more probable than the second best value $\Delta x = \frac{1}{40}$. Should the b_i be irrational, or should the generating density not have a bin structure, then we might expect the maximum of the log evidence to move towards smaller Δx (greater K) with increasing N, much like the M-posterior in fig. 5.4.

Another quantity of interest is the predictive density $p(x|D, \Delta x)$. By virtue of (5.22), this quantity can be computed via an evidence ratio. Thus, the log predictive density at x_3 is given by the differences between the log evidences of D_3 and D_2. For large K, this difference seems to converge to 0 from below. The situation is different at the location of the datapoints: the dashed line in fig. 5.8 shows the log evidence for $D_4 = \{x_1; x_1; x_2\}$. It appears to diverge sub-logarithmically slowly as $K \to \infty$. Since $p(x_1|D_2, \Delta x) = \frac{p(D_4|\Delta x)}{p(D_2|\Delta x)}$, this means that the predictive density has a singularity at the datapoints. However, divergence at these singularities seems slow enough to allow for an integration of the predictive density across these points, i.e. the probability contained within a non-infinitesimal interval can still be evaluated.



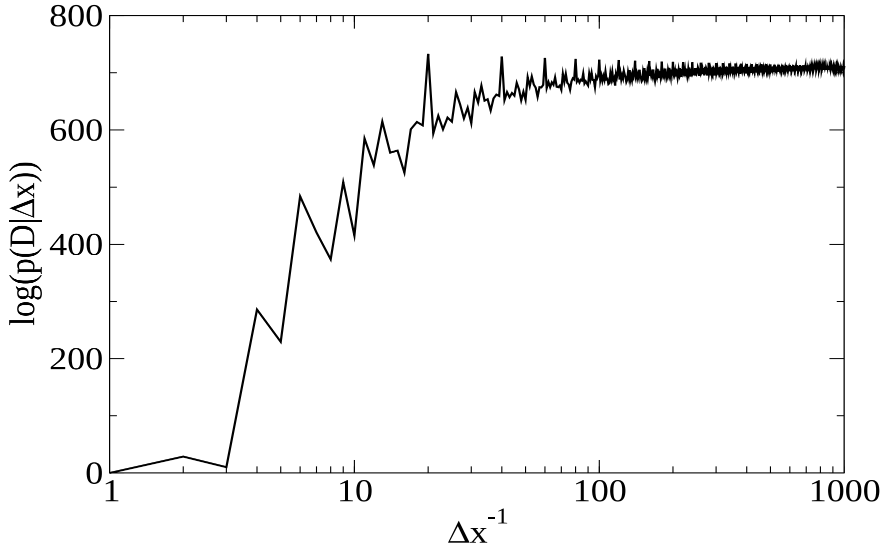


Figure 5.8: Top: Log evidence $\log(p(D|\Delta x))$ as a function of $K = \frac{1}{\Delta x}$. The range of X was $[0, 1]$. Solid lines: datasets containing one (black), two (red) and three (green) distinct datapoints. For large K, the log evidence appears to converge towards 0 from below. Dashed line: dataset containing three datapoints. This dataset was created by using the two-point dataset and doubling the first point. Here, the log evidence seems to diverge with K, although sub-logarithmically slow. Bottom: Log evidence $\log(p(D|\Delta x))$ as a function of $K = \frac{1}{\Delta x}$, $N = 1000$. The range of X was $[0, 1]$. The generating density consisted of 5 bins, with the bin boundaries b_i chosen so that $b_i \times 20$ was an integer. Consequently, the best discretization is found to be $\Delta x = \frac{1}{20}$.

5.11 Extension to multiple classes

Assume each data point x_i was labeled, the label being $y_i \in \{1, \ldots, C\}$. In other words, each x_i is drawn from one of C classes. The algorithm will now be extended so as to infer the joint distribution (or density) $P((x, y)|M, D)$, where $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

Instead of $M + 1$ probabilities, there are now $(M + 1)C$. P^y_m is the probability between k_m and k_{m-1} in class y, i.e. it is assumed that the bins are located at the same places across classes. This might seem like a rather arbitrary restriction. It shall, however, be imposed for two reasons:

1 The algorithm for iterating over the bin configurations keeps a simple form, similar to (5.16) and (5.17).

2 If different sets of k_m for the classes were allowed, computing marginal distributions and entropies became exceedingly difficult. While possible in principle, it would involve confluent hypergeometric functions and a significantly increased computational cost.

Let n^y_m be the number of data points in class y and bin m, and $\tilde{n}_m = \sum_{y=1}^{C} n^y_m$. The likelihood (5.2) now becomes

$$P(D|M, \{P^y_m, k_m\}) = \prod_{m=0}^{M} \frac{\prod_{y=1}^{C} (P^y_m)^{n^y_m}}{\Delta k_m^{\tilde{n}_m}} \tag{5.63}$$

Thus, following the same reasoning as above, the iteration rules now are:

$$a(0, \tilde{K}) = \frac{\prod_{y=1}^{C} n^y_0!}{(\tilde{K} + 1)^{\tilde{n}_0}} \tag{5.64}$$

where n^y_0 is the total number of data points in class y for which $k \leq \tilde{K}$ and $\tilde{n}_0 = \sum_{y=1}^{C} n^y_0$. Furthermore,

$$a(\tilde{M} + 1, \tilde{K}) = \sum_{\tilde{k}=\tilde{M}}^{\tilde{K}-1} a(\tilde{M}, \tilde{k}) \frac{\prod_{y=1}^{C} n^y_{\tilde{M}+1}!}{(\tilde{K} - \tilde{k})^{\tilde{n}_{\tilde{M}+1}}} \tag{5.65}$$

where $n^y_{\tilde{M}+1}$ is the total number of data points in class y for which $\tilde{k} < k \leq \tilde{K}$, and $\tilde{n}_{\tilde{M}+1} = \sum_{y=1}^{C} n^y_{\tilde{M}+1}$.

Now one can evaluate the expected joint distribution and its variance at any (k, y), using (5.22) and (5.23) as before. To compute the marginal distribution, note that

$$P(k|M, D) = \sum_{y=1}^{C} P((k, y)|M, D) \tag{5.66}$$

and likewise for its square.

5.12 Computing the mutual information and an upper bound on its variance

The mutual information between class label y and x is given by

$$I(X; Y) = H(X, Y) - H(X) - H(Y) \tag{5.67}$$

which has to be averaged over the posterior distribution of the model parameters, $p(\{P^y_m, k_m\}|M, D)$. This can be accomplished term-by-term, as described above, yielding the exact expectation of the mutual information under the posterior. The evaluation of its variance is somewhat more difficult, due to the mixed terms $E[H(X)H(Y)], E[H(X)H(X, Y)]$ and $E[H(Y)H(X, Y)]$. For the time being, an upper bound shall thus suffice. Using the identity (for a derivation, see (D.42))

$$\mathrm{Var}\left[\sum_{i=1}^{N} X_i\right] \leq N \sum_{i=1}^{N} \mathrm{Var}[X_i] \tag{5.68}$$

yields

$$\mathrm{Var}[I(X; Y)] \leq 3(\mathrm{Var}[H(X)] + \mathrm{Var}[H(Y)] + \mathrm{Var}[H(X, Y)]) \tag{5.69}$$

All terms on the r.h.s can be computed as above.

Figure 5.9 shows the results of some test runs on artificial data. Points were drawn from two classes with equal probability. Within each class, a three bin distribution was used to generate the data. The probabilities in the bins were varied to create four different values of the mutual information. Before inferring the mutual information from the data, the best discretization stepsize (via (5.24)) was determined first. The depicted values are individual averages over 100 datasets. In all cases, its true value lies well within the error bars. However, especially for small sample sizes the error bars seem rather too large – an indication that the upper bound

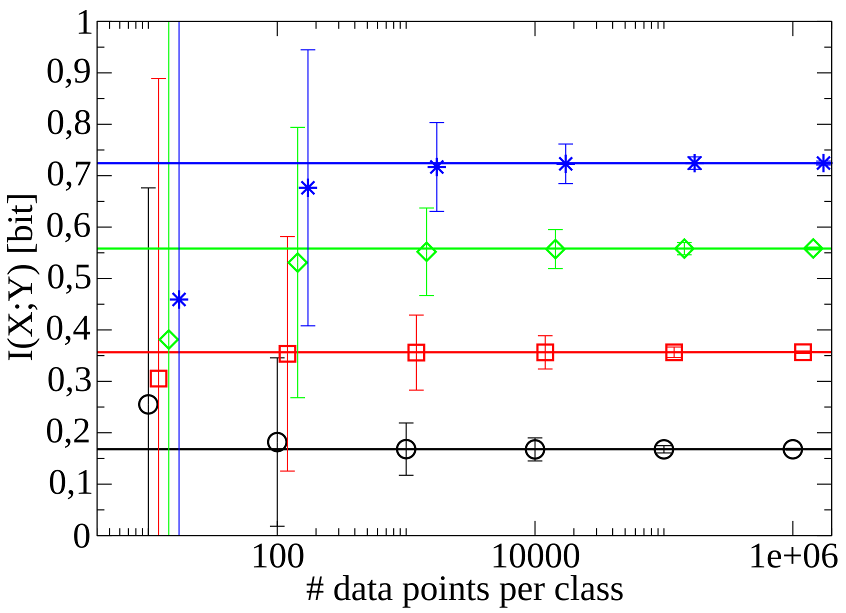


Figure 5.9: Expected mutual informations (symbols) and upper bounds on the standard deviations (error bars, computed via (5.69)) for different data set sizes. Solid line of the same color as the symbol: mutual information of the generating density. Circles: $I(X;Y) = 0.169$ bit, squares: $I(X;Y) = 0.357$ bit, diamonds: $I(X;Y) = 0.558$ bit, stars: $I(X;Y) = 0.724$ bit. Dataset sizes were 10,100,1000,10000,100000 and 1000000 datapoints, symbols are shifted to disentangle the error bars.

given by (5.69) needs future refinement. Nevertheless, observe that the expectation of $I(X;Y)$ is close to its true value from 100 data points per class onwards. One might argue that acquiring 100 data points per class would be difficult in most neurophysiological experiments, and thus, reliable mutual information estimates could not be obtained with the BBDIa. This problem can be remedied by changing the prior, as will be demonstrated in section 5.13 for sparse distributions. Another way towards faster convergence of the mutual information estimates is the incorporation of a bias towards certain values of the Fano factor (i.e. the variance of X divided by its mean), which has been shown to assume values between ≈ 1.1 and ≈ 1.8 for neurons from many cortical areas [32]. While it would in principle be possible to do that (through a properly chosen mixture of Dirichlet priors), it is rather difficult and thus this option will not be explored here.

A closer upper bound on the variance of the mutual information seems to be given by the variance of the joint entropy:

$$\mathrm{Var}\left[I(X;Y)\right] \leq \mathrm{Var}\left[H(X,Y)\right] \tag{5.70}$$

as can be seen in fig. 5.10.

Each type of symbol represents the values of the empirical standard deviation (dashed connecting lines) and the expected standard deviation of the joint entropy (solid connecting lines) for a given mutual information (see legend). The bound held in all tried cases. Moreover, for large datasets, the connecting lines seem to be parallel, which indicates that the two standard deviations may just differ by a factor. Note also that this behavior differs from that of the empirical standard deviations of the differential entropy (see fig. 5.7), which appear to approach the exact expected standard deviations in the limit of large datasets. However, there is no strict proof available for these observations at the moment.

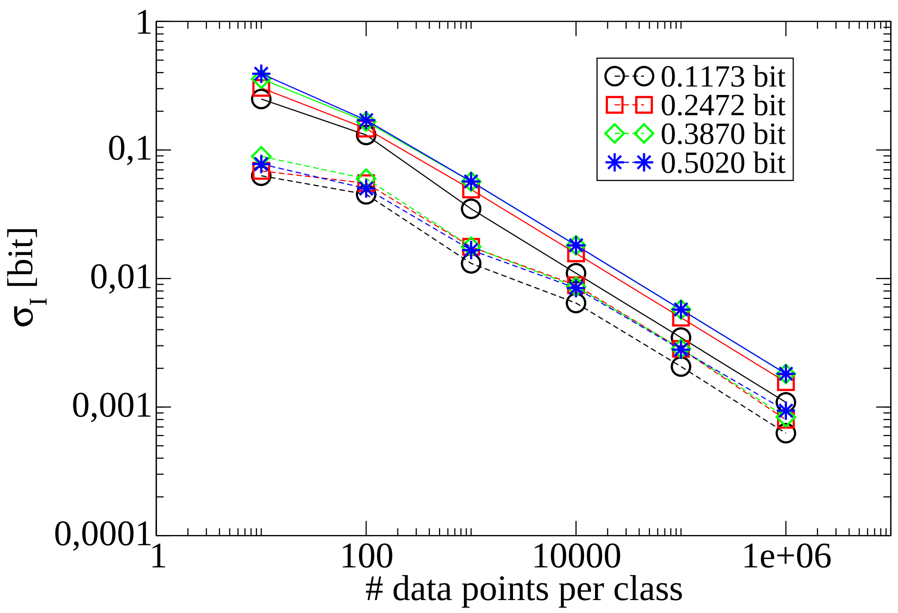


Figure 5.10: Expected standard deviations of the joint entropy (solid lines) and empirical standard deviations of the mutual information (dashed lines), computed by averaging over 100 datasets. The former seems to be an upper bound on the latter. Each type of symbol represents the values for a given mutual information (see legend). Note that the connecting lines for a given mutual information appear to be parallel for large datasets, which indicates that the empirical and expected standard deviation may just differ by a factor.

5.13 Sparse Priors

The uniform prior (5.8) is a reasonable choice in circumstances where no information is available about the data *a priori* other than their range. Should more be known, it is sensible to incorporate this knowledge into the algorithm so as to speed up the inference process (i.e. allow for better predictions from small datasets). In the following, symmetric Dirichlet priors of the form

$$p(\{P_m\}|M) \propto \prod_{m=0}^{M} P_m^{\theta-1} \tag{5.71}$$

will be examined, where $\theta > 0$ to avoid divergence of the normalization constant. Fig. 5.11 shows the resulting priors for $M = 1$ (two bins) and two different values of θ. The symmetry arises from the condition $P_0 + P_1 = 1$. The solid curve for $\theta = 0.5$ exhibits the typical behavior expected for $\theta < 1$: Extreme values are favored because $p(P_0) \to \infty$ as P_0 approaches 0 or 1. Thus, this range of θ will generally favor distributions where few bins contain most of the probability mass, i.e. *sparse* distributions. Conversely, the dashed curve for $\theta = 1.5$ indicates the general behavior expected for $\theta > 1$: the prior now promotes moderate values of P. Therefore, the probability mass will tend to be more evenly distributed (dense distributions). For $\theta = 1$, the uniform prior is recovered.

In typical single-cell neurophysiological experiments, sparse distributions are frequently encountered. Consider the following setup: a mammal is presented with a visual stimulus and the firing events produced by one of its visual cortex neurons are recorded in some suitably chosen time window. Assume that the temporal resolution of the recording equipment was 1ms and the window width 100ms. A simple model for the firing behavior is the Poisson process, i.e. a constant probability P_{fire} of observing an event at any given time within the window. The expected distribution of the number of observed events is then governed by a binominal distribution. Fig. 5.12, top, depicts three such distributions for small to medium values of P_{fire}. While up to 100 observed events in the window are possible in principle, those values are extremely unlikely. Hence, a sparseness promoting prior can be expected to speed up convergence when such (or similar) distributions are to be inferred from data.

The algorithm can be generalized to include $\theta \neq 1$. Using (D.12) instead of

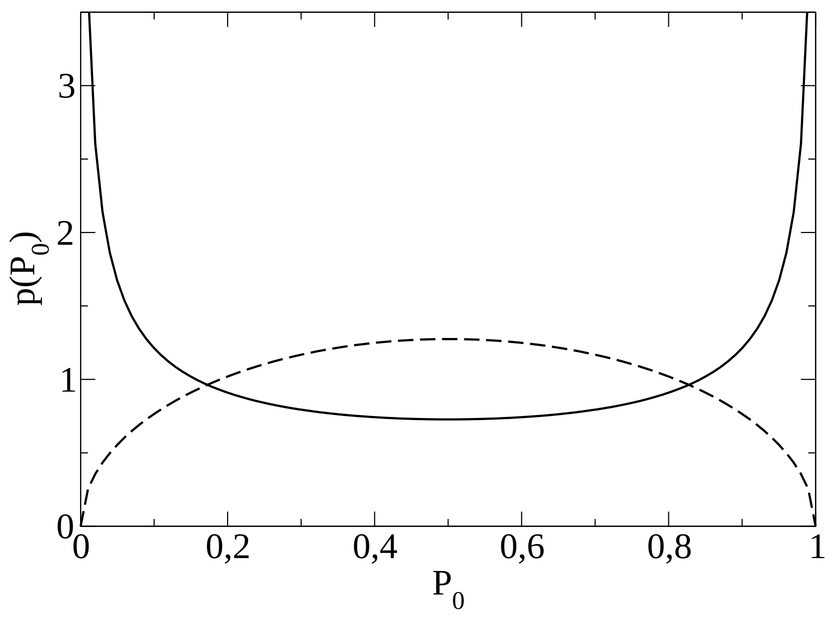


Figure 5.11: Prior for $M = 1$ and two choices of θ. Solid line: $\theta = 0.5$. The prior favors extreme values of P, i.e. sparse distributions. Dashed line: $\theta = 1.5$. Here, moderate values of P are promoted (dense distributions).

(D.8), (5.10) now becomes

$$p(\{P_m\}|M) = \Gamma(\theta_M) \tag{5.72}$$

where

$$\theta_M = (M+1)\theta \tag{5.73}$$

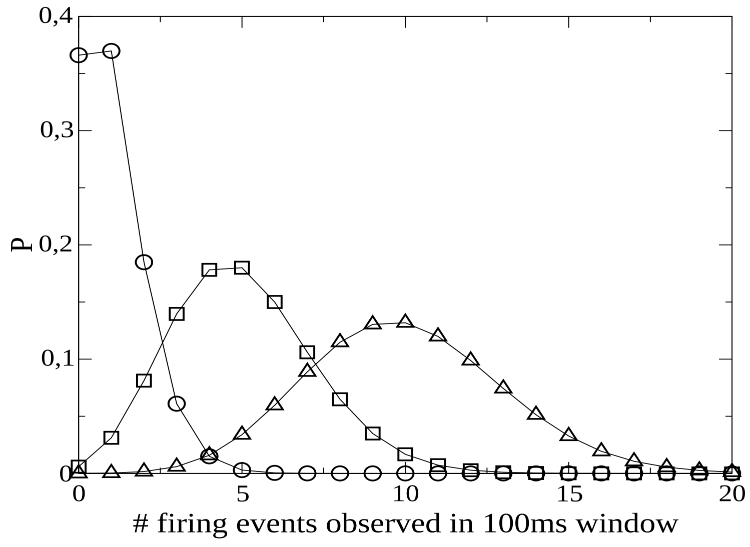
Likewise, (5.15) is to be replaced by

$$P(D|\{k_m\}, M) = \frac{\Gamma(\theta_M)}{\Gamma(N+\theta_M)} \prod_{m=0}^{M} \frac{\Gamma(n_m+\theta)}{\Delta k_m^{n_m}} \tag{5.74}$$

This expression is again a product of the counts observed in different bins and the bin width, times a factor which only depends on the total number of bins and data points. Hence, the same sum-product decomposition scheme as before can be used, with (5.16) and (5.17) now being

$$a(0,\tilde{K}) = \frac{\Gamma(n_0+\theta)}{(\tilde{K}+1)^{n_0}} \tag{5.75}$$

$$a(\tilde{M}+1,\tilde{K}) = \sum_{k=\tilde{M}}^{\tilde{K}-1} a(\tilde{M},\tilde{k}) \frac{\Gamma(n_{\tilde{M}+1}+\theta)}{(\tilde{K}-\tilde{k})^{n_{\tilde{M}+1}}} \tag{5.76}$$



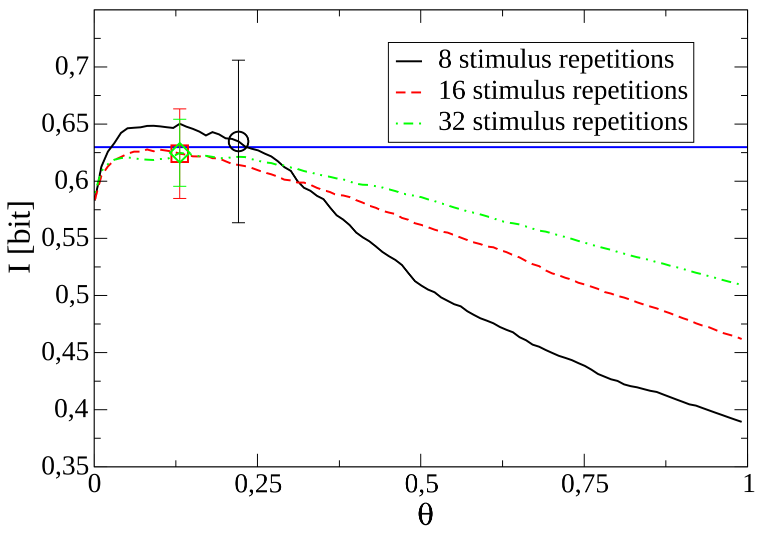


Figure 5.12: Top: Binominal distributions are simple models for the distributions of spike counts observed in a fixed time window. The curves were computed for $P_{fire} = 0.01$ (circles), $P_{fire} = 0.05$ (squares) and $P_{fire} = 0.1$ (triangles). Values are plotted only up to a spike count of 20, even though a maximum of 100 would have been possible. However, the probabilities for counts greater than 20 are virtually 0. Thus, most possible values are never assumed, i.e. the distributions are sparse. Bottom: dependence of the expected mutual information on the sparseness parameter θ, for 8, 16 and 32 stimulus repetitions. Datasets contained simulated responses to the presentation of 8 different stimuli. Symbols of the same colour as the lines mark the location of the posterior maximum of θ, error bars are empirical standard deviations of the mutual information for those values of θ. The horizontal blue line represents the true mutual information. For details, see text.

This allows for the computation of evidences and expected probabilities and their variances. To compute entropies and mutual informations and their variances, the relevant averages have to be adapted as well: after some tedious but straightforward calculus, one finds that (5.40) (using (D.37) instead of (D.30)) has to be substituted by

$$E\left[P_{m'}\log(P_{m'})|\{k_m\},M,D\right] = \frac{n_{m'}+\theta}{N+\theta_M}\left[h_0^N(\theta_M) - h_0^{n_{m'}}(\theta) + \Psi\left(\theta_M\right) - \Psi(\theta)\right] \tag{5.77}$$

where $\Psi()$ is the digamma function (see appendix D) and

$$h_a^b(\theta) = \sum_{i=a}^{b}\frac{1}{i+\theta} \tag{5.78}$$

$$_2h_a^b(\theta) = \sum_{i=a}^{b}\frac{1}{(i+\theta)^2} \tag{5.79}$$

While the term in the square brackets could be written as the difference between two digamma functions, decomposing it into a part that depends both on the prior and on the data and a part that depends only on the prior (i.e. M and θ) allows for a precomputation of the latter. (5.42) becomes

$$E\left[P_{m'}\log(\Delta k_{m'})|\{k_m\},M,D\right] = \frac{n_{m'}+\theta}{N+\theta_M}\log(\Delta k_{m'}) \tag{5.80}$$

Likewise, substituting (D.38) for (D.31) in (5.49) yields

$$\begin{aligned} E\left[P_{m'}^2\log^2(P_{m'})|\{k_m\},M,D\right] &= \frac{(n_{m'}+\theta)(n_{m'}+1+\theta)}{(N+\theta_M)(N+1+\theta_M)} \\ &\times\Bigg[\Big(h_0^{N+1}(\theta_M) - h_0^{n_{m'}+1}(\theta) + \Psi(\theta_M) - \Psi(\theta)\Big)^2 \\ &\quad + {}_2h_0^{N+1}(\theta_M) - {}_2h_0^{n_{m'}+1}(\theta) + \Psi'(\theta) - \Psi'(\theta_M)\Bigg] \end{aligned} \tag{5.81}$$

and finally, employing (D.39) in place of (D.32) in the derivation of (5.54), one obtains for (5.55)

$$E\left[f(P_{m'})|\{k_m\},M,D\right] = (n_{m'}+\theta)\left(h_0^{N+1}(\theta_M) - h_0^{n_{m'}}(\theta) + \Psi(\theta_M) - \Psi(\theta)\right) \tag{5.82}$$

$$E\left[f(P_{m''})|\{k_m\},M,D\right] = (n_{m''}+\theta)\left(h_0^{N+1}(\theta_M) - h_0^{n_{m''}}(\theta) + \Psi(\theta_M) - \Psi(\theta)\right) \tag{5.83}$$

then divide the results of the averaging by $(N+\theta_M)(N+1+\theta_M)$, and for (5.57)

$$E\left[f(P_{m'})|\{k_m\},M,D\right] = (n_{m'}+\theta) \tag{5.84}$$

$$E\left[f(P_{m''})|\{k_m\},M,D\right] = (n_{m''}+\theta) \tag{5.85}$$

and multiply the averages by $\frac{2h_0^{N+1}(\theta_M)-\Psi'(\theta_M)}{(N+\theta_M)(N+1+\theta_M)}$.

The question which remains is how to choose θ. Since the sparseness of a distribution will usually not be known a priori, a feasible way is to treat it as another hyperparameter to be inferred from the data. However, integrating over θ is computationally rather expensive. Instead, a maximum a posterior approach was tried (the posterior density of θ was unimodal in all observed cases), which seems to work quite well, as shown in fig. 5.12, bottom: for a set of 8 (purely abstract) 'stimuli', distributions of responses, i.e. 'mean firing rates' (minimum 5 spikes/sec, maximum 380 spikes/sec, average 60 spikes/sec), were generated by drawing random variables from binominal distributions with different values of P_{fire}. These firing probabilities were functions of the stimulus label Y. The expected mutual informations between Y and the responses X were subsequently computed for 8, 16 and 32 stimulus repetitions per dataset. Expectations were averaged over 100 datasets, and θ was varied between 0 and 1. The symbols of the same color as the lines mark the values of the expected mutual information at the locations of the posterior maxima of θ, which are close to the true value of the mutual information (horizontal blue line) and well within the error bars. Two other features of the graph are noteworthy: firstly, the dependency of the expected mutual information becomes less pronounced with increasing number of stimulus repetitions, i.e. the more data are available, the more will the estimates be determined by the data, and not by the prior parameter θ. Secondly, setting $\theta = 1$ will tend to underestimate the mutual information, which reflects the fact that a sparse prior ($\theta < 1$) is beneficial if the generating distributions are sparse.

Furthermore, the validity of the upper bound eqn. (5.70) was also tested under sparse conditions, for mutual informations in the range $[0.25, 1.93]$ bit. It was found that the bound held in all cases, with a factor of at least ≈ 1.8 and at most ≈ 4.0 between the empirical standard deviation of the mutual information and the expected standard deviation of the joint entropy.

To examine the performance of the BBDIa with sparse priors on a range of sparse distributions, datasets for several values of the mutual information were generated in the same way as above (which is similar to that described in [74]).

For each dataset, the optimal θ was determined by the maximum of its poste-

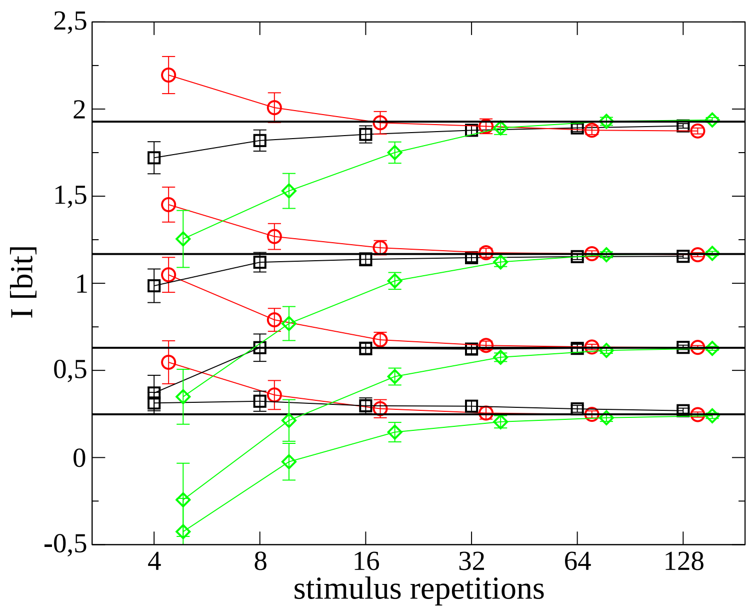


Figure 5.13: Results of the BBDIa with sparse priors on artificially created 'neural responses' to 8 'stimuli' (black squares), compared with the method described in [74](red circles). and the NSB method [61](green diamonds). The response distributions were binominals with different P_{fire}s. Error bars are ± 1 standard deviations, obtained from averaging over 100 datasets.

rior distribution in the interval $[0.0001, 1]$. Using this θ, the best M was searched for in the same fashion. Given those two parameters, the mutual information was computed. The results were averaged over 100 datasets (calculating θ and M anew for each dataset), which also allowed for the estimation of error bars. Fig. 5.13, shows the results. The error bars are the estimated standard deviations of the mutual information, not the standard errors. The black squares are the expectations computed by the BBDIa. The red circles depict the mutual information estimates computed from the observed frequencies corrected by the first-order bias term reported in [74]. Those frequencies, as described in [74], were obtained by a simple discretization of the response space. The green diamonds represent the mutual information estimates obtained with the NSB (Nemenman-Shafee-Bialek) method[3] [61], which is a Bayesian entropy estimation method using a multinomial noise model and an (almost) uniform prior over the entropy.

[3]available at *http://nsb-entropy.sourceforge.net*

Even for a rather small number of stimulus repetitions, the BBDIa performs quite well, getting better as the size of the dataset increases. In the great majority of cases it appears to be more accurate than the finite-size corrected estimates and the NSB estimates, even though the latter two yield reasonable results as well if the number of stimulus repetitions is greater than the number of stimuli. Moreover, the small error bars indicate that reliable results can be expected with the method proposed here even if only a single dataset is available. Note that the NSB method, at least in its current implementation, can produce inconsistent estimates of the mutual information: for very small datasets (8 and 16 stimulus repetitions), it yields negative results, even though the mutual information is a strictly positive quantity. I would attribute this behavior to a separate estimation of marginal $H(X)$ and conditional $H(X|Y)$ entropy (which are then used to compute the mutual information via $I(X;Y) = H(X) - H(X|Y)$) in the current implementation of the NSB method. If instead a joint prior over X and Y was used, this problem might be overcome.

5.13.1 Can (or should) finite-size effects be avoided?

In the field of non-Bayesian statistics, a great deal of effort has been expended on the search for so-called 'unbiased estimators'. An estimator of a function of a random variable is said to be unbiased, if its expectation is equal to its 'true' value, i.e. the value that could be computed if the generating density of the random variable was known [9]. From a sampling theoretic perspective, this property is usually considered desirable, because it is hoped that one might get a better estimate of the quantity of interest with shorter samples than with a biased estimator. If the bias disappears as the sample size grows to ∞, then the estimator is said to be asymptotically unbiased. The existence of a bias for a sample of limited size is then referred to as a 'finite-size effect', which is also taken to be an unwanted property of the estimation procedure.

When trying to estimate the mutual information from a sample, many estimators are found to have a heavy bias. This is mostly due to the fact that entropy is a concave function of the distribution [13] and will therefore (by virtue of Jensen's inequality) tend to be underestimated when the distribution is computed from the observed frequencies. This leads in turn to an overestimation of the mutual information, loosely speaking because due to fluctuations, one is likely to see more structure in the data than there really is.

From a Bayesian perspective, this view is not unconditionally shared. The source of bias is the prior. Thus, when the sample size is limited, the results of an inference process should be biased, because they still depend on information that was available a priori. What is important is that this bias is sensible, i.e. that the prior assumptions are justified. Once those assumptions have been made, they can be translated into a probability distribution (or density), see section 2.5.2.

When the sample size grows to ∞, the bias should disappear, because in this limit, inference results should be completely determined by the data. More formally, the data-dependent term in eqn. (2.60) can be expected to grow larger with the sample, thus the Kullback divergence between posterior and prior will contribute less to the inference process. In other words, asymptotic unbiasedness is a sensible requirement in the Bayesian view as well. Note, however, that it arises automatically as a consequence of how inference is conducted, and needs not be introduced as an added desideratum.

For example, consider fig. 5.13. The inferred values of $I(X;Y)$ (red squares) move towards the true values (lines) as the number of trial per stimulus n^y grow, i.e. the bias disappears asymptotically. For very small n^y, a bias is clearly visible: an expression of the fact that not enough information had been gathered to discard all prior beliefs. I would tend to argue that this bias is sensible – all that is known about the mutual information a priori is that it is positive and $\leq$ some upper bound. Thus, if $I(X;Y)$ of the generating distribution is very small, an upward bias can be expected for very small n^y. Conversely, large mutual informations will receive a downward bias for small n^y. Both features can be observed in fig. 5.13.

In summary, a bias should not be avoided, but rather, it should be properly interpreted. Procedures like stimulus shuffling (i.e. randomizing the stimulus/response relationship in the data, which should lead to zero mutual information) to determine the bias (see e.g. [67]), and then subtracting this bias to get a more 'correct' estimate of the mutual information are questionable at best. In [47] it was found, after applying this procedure, that the shuffled mutual information was actually higher than the unshuffled one in the SOA 14 ms condition – an indication that the mutual information values are no more correct after the bias subtraction than they were before.

5.14 RSVP results

The BBDIa with sparse priors will now be applied to the same single cell recordings that were analyzed in the last chapter.

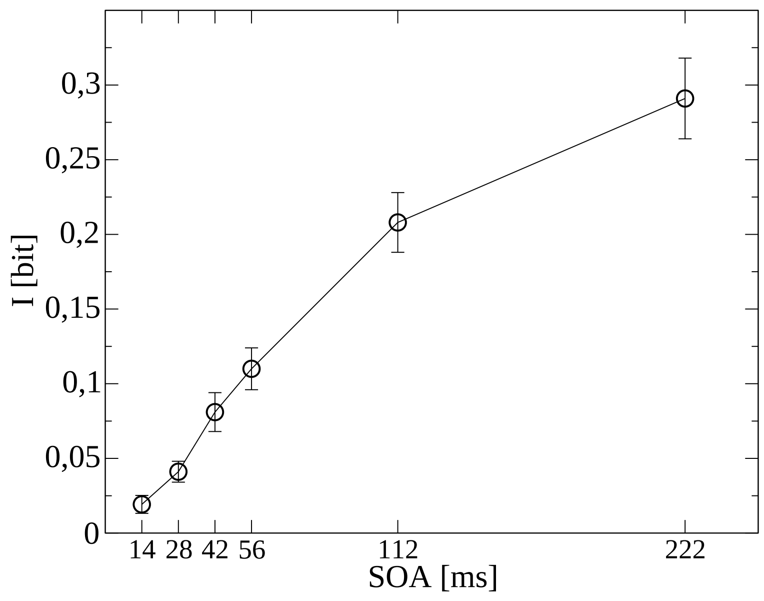
5.14.1 Mutual information and information transmission rate

Fig. 5.14, top, shows the population averages of the mutual information between stimulus label and response. In accordance with the results presented in the previous chapter, the response was computed from the spiketrain via $f_0(l, e)$, with l and e determined anew for each cell and SOA by the BBCa. Error bars were computed via eqn. (5.70) (i.e. remaining uncertainty due to limited dataset size for a given cell) and the between-cells variation of the expectation of $I(X; Y)$, treating them as independent sources of variance.

As one would expect, the mutual information increases with SOA. At SOA 56 ms, the mutual information is ≈ 0.11 bit. Assuming that the brain implements a factorial code, it would thus be necessary to observe the responses of at least $\frac{1}{0.11} \approx 9$ cells until 1 bit of information is gained. Given a noninformative prior, this bit could then be used to make a binary decision, e.g. 'best stimulus' versus 'rest'. This bound is consistent with that computed previously (at least 6 cells, eqn. (4.38)), which was derived based on an optimality assumption (see appendix C). In contrast, the mutual information measures the average information gain in a classification task, and can thus be expected to yield a lower bound on the number of cells which is somewhat higher.

Especially in the faster conditions, the mutual information turns out to be rather small. It is therefore natural to wonder whether the small deviations from zero are not some residual finite size effect. This question will be addressed in section 5.14.3.

The increase of $I(X; Y)$ with SOA is not quite linear, as can be seen in fig. 5.14, bottom. Here, the information transmission rates, i.e. mutual information per stimulus time, are plotted as a function of SOA. If the increase was linear, then the information transmission rate would have to be constant. It does, however, appear



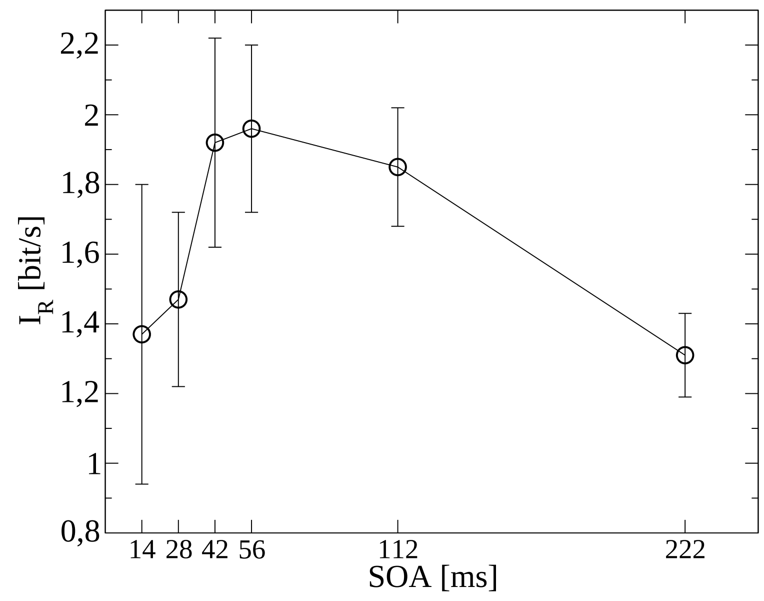


Figure 5.14: Top: population average of the mutual information between stimulus label y and response x. Responses were computed from the spiketrain via $x = f_0(l, e)$, with l and e computed anew for each cell and SOA by the BBCa described in the previous chapter. Bottom: Information transmission rate, i.e. mutual information per unit of stimulus time.

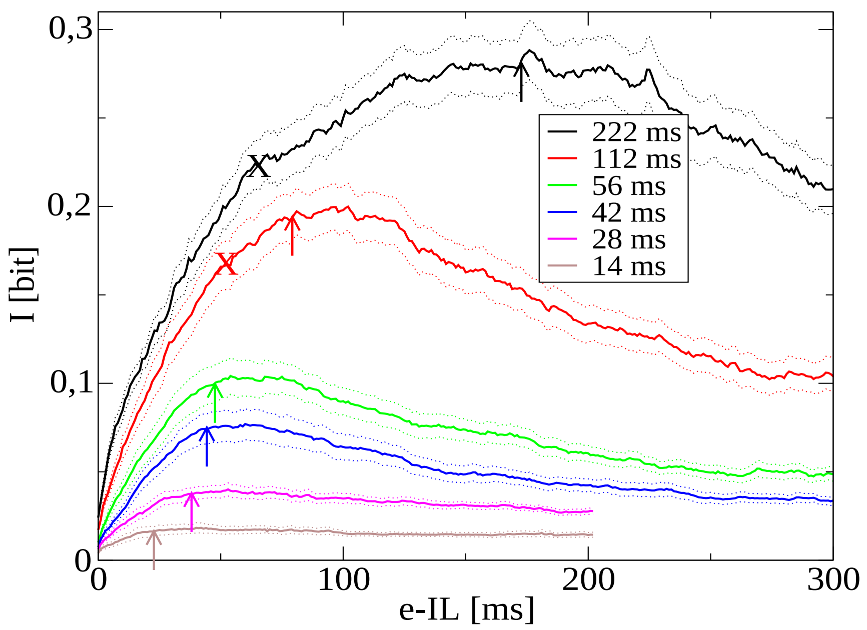


Figure 5.15: Mutual information between stimulus label and response (solid lines) ± one standard error (dotted lines) in a window of varying length, starting at the IL. Standard errors were computed from the population variance alone. The arrows indicate the end of the expected window e-IL=IRD (see also fig. 4.8). The 'X' on the curves for SOA 222 ms and 112 ms marks the time of the first steep downward slope in the log evidence per datapoint graphs (fig. 4.9), i.e. the times when most of the classification information has been transmitted. For details, see text.

to be maximal at SOA ≈ 56 ms (strictly speaking, a maximum is bracketed in the SOA interval [42 ms,112 ms]). This indicates that both the very early as well as the later parts of the response contain less information than those around 50 ms after the IL (the IRD for SOA 56 ms was 47.2 ms, see fig. 4.8).

5.14.2 Temporal structure of the mutual information

Figure 5.15 shows the population averages of the cumulative $I(X;Y)$ (solid lines) ± one standard error (dotted lines) in a window of variable length. Responses were individually IL aligned. Due to computational time constraints, standard errors reflect only the variability between cells. The arrows indicate the end of the expected window e-IL=IRD as computed in the previous chapter (see also fig. 4.8). Within standard error, $I(X;Y)$ does not increase significantly after the arrows. This shows that the part of the response which is found to be best suited for classification is also the part which contains (almost) all of $I(X;Y)$. The 'X' on the curves for SOA 222 ms and 112 ms marks the time of the first steep downward slope in the log evidence per datapoint graphs (fig. 4.9), i.e. the times when most of the classification information has been transmitted. At those time indexes, the cumulative $I(X;Y)$ for SOA 222 ms also notably changes slope (to a lesser degree, this can be observed in the curve for SOA 112 ms, too).

Note that the curve for SOA 14 ms is almost flat, and close to zero. With a sufficiently strong prior, one might be able to perceive a maximum somewhere between the arrow and ≈ 50 ms, but it is hardly discernible. It is thus natural to wonder:

5.14.3 Is the mutual information zero?

Given the fact that the above presented mutual informations – especially in the faster presentation conditions – are rather small, one might wonder if they are not really zero altogether. The expectation of the mutual information is ill suited to answer this question, because $I(X;Y)$ is a continuous variable in a bounded interval (here: $[0,3]$ bits). Therefore, unless its posterior density diverges at $I(X;Y) = 0$, its expectation will always be greater than 0. Conversely, if the density is finite at that point, then the probability of $I(X;Y) = 0$ is zero. If, on the other hand, this posterior was known, then questions of the form 'what is the probability that $I(X;Y) < 0.01$ bits?' could be addressed. Unfortunately, computing this posterior is very difficult, if not infeasible.

There is, however, another way to circumvent this problem. As stated in section 2.4.2, the mutual information is zero if and only if the joint density is equal to the product of the marginals. This observation allows for the construction of a Bayesian hypothesis test:

H0 The joint density is the product of the marginal densities, i.e. $I(X;Y) = 0$.

H1 The joint density is not the product of the marginal densities, i.e. $I(X;Y) > 0$.

To compute the posterior probabilities of these hypotheses, one must first assign a prior to both of them. In the following, the maximum entropy choice $P(\mathrm{H0}) = P(\mathrm{H1}) = 0.5$ will be used. This is sensible from the perspective of the hypotheses, because there are only two alternatives. However, when the possible densities included in each hypothesis are considered, then one might argue that this choice is heavily biased towards H0: as noted above, there are many more densities for which $I(X;Y) > 0$, i.e. a prior $P(\mathrm{H0}) \ll P(\mathrm{H1})$ could also be motivated. Thus, if the posterior probabilities favor H1 given the uniform prior, it might be taken as strong evidence that the mutual information is not zero.

Next, given a dataset, the probabilities (or densities) $P(D|\mathrm{H0})$ and $P(D|\mathrm{H1})$ need to be evaluated. For H1, this is given by

$$P(D|\mathrm{H1}) = \sum_m P(D|m)P(m) \tag{5.86}$$

where $P(D|m)$ is eqn. (5.3) (extended to multiple classes, as explained in section 5.11), $P(m)$ is the (uniform) prior over the number of bin boundaries, and the sum over m was chosen to run from 0 to $K-1$, i.e. all possible density models are considered.

$P(\mathrm{H0})$ can be calculated by treating the stimulus label independent of the response, i.e.

$$P(D|\mathrm{H0}) = P_s \sum_m P(\tilde{D}|m)P(m) \tag{5.87}$$

where $\tilde{D}$ is the dataset which contains only the responses, thus $P(\tilde{D}|m)$ is eqn. (5.3) in its original form, and

$$P_s = \frac{\prod_{y=1}^{C} n^y!}{(N+C-1)!}(C-1)! \tag{5.88}$$

is the evidence of the distribution of the stimulus labels (for a derivation, see section 4.6). n^y is the number of times stimulus y was presented, C is the size of the stimulus set (here: 8) and $N = \sum_{y=1}^{C} n^y$. Now the posterior

$$P(\mathrm{H0}|D) = \frac{P(D|\mathrm{H0})P(\mathrm{H0})}{P(D|\mathrm{H0})P(\mathrm{H0}) + P(D|\mathrm{H1})P(\mathrm{H1})} \tag{5.89}$$

$$P(\mathrm{H1}|D) = \frac{P(D|\mathrm{H1})P(\mathrm{H1})}{P(D|\mathrm{H0})P(\mathrm{H0}) + P(D|\mathrm{H1})P(\mathrm{H1})} \tag{5.90}$$

can be computed.

If L datasets are available, the probability that all of them contain no mutual information between stimulus and response is

$$P(\mathrm{H0}|D_1,\ldots,D_L) = \prod_{l=1}^{L} P(\mathrm{H0}|D_l) \tag{5.91}$$

given that $P(D_1,\ldots,D_L|\mathrm{H0}) = \prod_{l=1}^{L} P(D_l|\mathrm{H0})$ (and likewise for $P(D_1,\ldots,D_L|\mathrm{H1})$), i.e. the probability of a particular dataset given one of the hypotheses does not depend on any of the other datasets. This is a justifiable assumption if the datasets contain recordings from different cells.

SOA [ms]	$P(\mathrm{H0}|D)$
222	≈ 0
110	≈ 0
56	≈ 0
42	≈ 0
28	4.0904×10^{-37}
14	0.99997

Table 5.1: Probabilities that the mutual information between stimulus label and neural response are zero in all available cell recordings. ≈ 0 means that the probability was close to the smallest number representable by a variable of type 'double' (ca. 10^{-300}).

Table 5.1 shows the results for all available cell recordings. For SOAs 42 ms - 222 ms, the probabilities of H0 were found to be close to the smallest number representable by a variable of type 'double' (ca. 10^{-300}), i.e. one can be very certain that there is nonzero mutual information in (at least one of) the datasets. This is true as well to a slightly lesser degree for SOA 28 ms, but the probability is still close enough to 0 to justify the statement that the responses contain stimulus-related information at this SOA.

The situation is somewhat different at SOA 14 ms. Here, the posterior probability is strongly in favor of H0, which indicates that stimulus-related information is hard to detect in the responses. As noted above, this does not necessarily mean that the mutual information is really zero (due to the bias towards H0), but it certainly has to be very small. This is supported by the findings in [48], where a small signal was found at this SOA. Furthermore, the log evidence graph (fig. 4.9) also suggests that a very small amount of classification information is transmitted at SOA 14 ms.

5.15 Conclusion

5.15.1 Algorithm

The presented algorithm computes exact evidences and relevant expectations of probability distributions/densities with polynomial computational effort. This is a significant improvement over the naïve approach, which requires an exponential growth of the number of operations with the degrees of freedom. It is also able to find the optimal number of degrees of freedom necessary to explain the data, without the danger of overfitting. Furthermore, the expectations of entropies and mutual information have been shown to be close to their true values for relatively small sample sizes. In the past, a variety of methods for dealing with systematic errors due to small sample sizes have been proposed, such as stimulus shuffling in [67] or regularization of neural responses by convolving them with Gaussian kernels. What most of these approaches have in common is that the marginal $P(X)$ and the conditional $P(X|Y)$ distributions of the responses are estimated first, and the mutual information is then computed from these estimates via

$$I(X;Y) = H(X) - H(X|Y) \tag{5.92}$$

where $H(X)(H(X|Y))$ is computed from $P(X)(P(X|Y)$ and $P(Y)$, which is usually set by the experimenter) via eqn. (2.20)(eqn. (2.27)). The problem with such a procedure is that the entropy is a nonlinear function of the probabilities, hence the expectation of the entropy is not equal to the entropy of the expected probabilities. More precisely, from a Bayesian perspective, this exchange of the order of computing the expectations would only be justified if the posterior density of the distributions was concentrated at a single point. This, however, is not likely to happen with the small datasets usually available from neurophysiological experiments. Thus, while these approaches work to some degree, they lack the sound theoretical foundation of exact Bayesian treatments.

In [74], finite size corrections are given based on the number of effective bins, i.e. the number of bins necessary to explain the data. Therein, it is also demonstrated that this leads to information estimates which converge much more rapidly to the true value than the other techniques mentioned (shuffling and convolving). However, [74] as themselves admit, their method of choosing the number of effective bins is only 'Bayesian-like'. Furthermore, the initial regularization applied to the data – choosing a number of bins that is equal to the number of stimuli and then setting the bin boundaries so that all bins contain the same number of data points, a procedure also used by [79] – is debatable (it should, however, be pointed out that this equi-probable binning procedure is not an essential ingredient for the successful application of the finite-size corrections of [74]. It has recently been demonstrated [2] that, given a decent amount of data is available, the methods of [74] can be used to yield reasonably unbiased estimates for $M = K - 1$.). On the one hand, one might argue that the posterior of M is still broad when the data set is small (see fig. 5.3, top row), so choosing the wrong number of bins will do little damage. On the other hand, the bin boundaries must certainly not be chosen in such a way that all bins contain the same number of points. Doing so will destroy the structure present in the data. Consider e.g. fig. 5.3, third row: there are many more data points in the interval $[0.58, 0.68]$ than there are between $[0.15, 0.58]$, which reflects a feature of the distribution from which the data were drawn and should thus be modeled by any good density estimation technique. This will, however, not be the case if the boundaries are chosen as proposed: there would be a boundary somewhere at ≈ 0.63 instead of 0.58, and the step at this point would be replaced by a considerably smaller one at ≈ 0.63, thus misrepresenting the underlying distribution. In other words, this procedure would not even converge to the correct distribution as the data set size grows larger. Consequently, mutual information estimates calculated from those estimated distributions must be interpreted with great care.

The author believes to have shown that those drawbacks can be overcome by a Bayesian treatment, which also shows improved performance over finite-size corrections. Thus, the algorithm should be useful in several areas of research where large datasets are hard to come by, such as neuroscience.

Another interesting Bayesian approach to removing finite size effects in entropy estimation is the Nemenman-Shafee-Bialek (NSB) method [61]. It exploits the fact that the typical distributions under the symmetric Dirichlet prior (5.71) have very similar entropies, with a variance that vanishes as K grows large. This observation is then employed to construct a prior which is (almost) uniform in the entropy. The resulting entropy estimator is demonstrated to work very well even for relatively small datasets. However, as demonstrated above, it yields (at least in the current implementation) inconsistent estimates for the mutual information.

In contrast to the NSB method, my approach deals with finite size effects by determining the model complexity (i.e. the posterior of M). It might be interesting to combine the two: since the NSB prior depends only on θ and K, the required numerical integration (eqn. (9) in [61]) could be carried out, with eqn. (10) in [61] replaced by $P(D|\theta)$ (i.e the denominator of (5.20) for a given θ).

It was proven in [73] that uniformly (over all possible distributions) consistent entropy estimators can be constructed for distributions comprised of any number of bins M, even if $M \gg N$. The above presented results (fig. 5.6) suggest that the expected entropies computed with our algorithm are asymptotically unbiased and consistent. Furthermore, the true entropy was usually found within the expected standard deviation. It remains to be determined how the algorithm performs if $M \gg N$.

Since the upper bound (5.69) on the variance of the mutual information is rather large for small sample sizes, it might be interesting to invest some more work into computing the exact variance of the mutual information. This, however, turns out to be difficult.

5.15.2 The information throughput of STSa neurons is maximized at SOA $\approx$ 60 ms

The examined neuron population transmits the most information per stimulus time for SOA in the interval [42 ms,112 ms] in RSVP experiments. Furthermore, the presented results suggest that most of $I(X;Y)$ has been transmitted at a presentation time of $\approx$ 60 ms: the cumulative $I(X;Y)$ curves for SOA 222 ms and 112 ms show a change in slope around that time. More quantitatively, the amounts of transmitted information up to the 'X' in fig. 5.15 are 77% for SOA 222 ms, and 84% for SOA

112 ms. This observation is confirmed by the log evidence graphs (fig. 4.9), which also show a sharp drop about 50 ms - 60 ms after the IL. Therefore, the maximum information transmission rate can be expected to reach its peak at $\approx$ IL+60 ms. These results are consistent with those found by other researchers: in [32], it was determined that IT neurons transmit most stimulus-related information within 50 ms of the response onset. According to [47], 60% of the information are contained within the first 50 ms.

In section 4.10.2, it was demonstrated that IRD $\geq$ SOA if SOA $\leq$ 42 ms, and IRD $\leq$ SOA otherwise. This is another indication that information throughput can be maximized for SOA $\approx$ 60 ms: if the IRD is longer than SOA, responses to consecutive stimuli will overlap, and information is likely to be lost. Conversely, if IRD is shorter than SOA, then time is wasted 'waiting' for new input, thus the neurons are not used to maximum capacity. Nevertheless, longer SOAs will still reduce the interference between responses to subsequent stimuli, as demonstrated in fig 5.14: within the studied SOA range, the transmitted information increases with SOA.

Given the evidence, it is questionable whether $I(X;Y) > 0$ at SOA 14 ms. However, for the above mentioned reasons (strong bias of the test in favor of $I(X;Y) = 0$), it should not be ruled out completely. Moreover, the log evidence graphs (fig. 4.9) suggest a small signal. This is consistent with the findings presented in [48], where human psychophysical experiments showed an above chance performance at this SOA.

Chapter 6

Overall conclusion

The main objective of this thesis has been to demonstrate the utility of Bayesian and information theoretic methods for neuroscience. Two approaches were taken: firstly, starting from first principles, a coding scheme was constructed that exhibited qualitative similarities to neural codes found in the mammalian visual cortex. Furthermore, it was shown that the omnipresent noise in real biological neurons can be used to motivate sparse codes. While this argument is so far purely theoretic, it is conceivable to develop an artificial neural network that could explicitly use this assumption for learning sparse codes from natural images, possibly in concert with the already established sparseness promotion through redundancy reduction.

Secondly, two Bayesian methods were proposed for analyzing a real neural code. I hope to have demonstated that the results which can be obtained via exact Bayesian methods justify the considerable effort necessary for their development. Needless to say, this work is far from finished. One possible interesting generalization of the BBCa (chapter 4) is that towards multi-feature classification. Assuming that real brains do approximate factorial codes, one could use a generalized BBCa for naïve Bayesian classification (i.e. assuming that the code features are independent given the stimulus). This might offer new and exciting insights into the coding strategies employed by the brain.

Moreover, combining the theoretical considerations of chapter 3 and the results from chapters 4 and 5, it might also become possible to have a suitably designed artificial neural network (motivated by information theoretic principles, similar to the one studied in this thesis) learn the code implemented by the mammalian visual cortex. Since there are several processing stages between retinal input and STSa, this network would have to be fairly complex. However, due to the recent advances in computer technology, this undertaking is now much more feasible that it had been a few years ago, especially when fast coding schemes, such as the quadratic programming algorithm from chapter 3, are used.

Appendix A

Proof of minimum property

As mentioned in section 3.2, the activation dynamics of the extended REC model are governed by the error function

$$E = \frac{1}{2}\left(\vec{X} - \sum_{j=1}^{M} \vec{W}_j O_j\right)^2 + \sum_{j=1}^{M} \lambda_j |O_j| \tag{A.1}$$

where M is the number of units, $\vec{W}_j$ is the weight vector which connects the j-th unit to the input terminals, λ_j are the activation thresholds and O_j are the outputs. $\vec{X}$ is the current input vector.
The outputs were given by

$$O_i = O(A_i) \tag{A.2}$$

$$A_i = \vec{X}\vec{W}_i + \sum_{j=1, j\neq i}^{M} L_{ij} O_j \tag{A.3}$$

A_i are the unit's activations, the activation function $O(A)$ is

$$\begin{aligned} -\lambda_i < A_i < \lambda_i &: \quad O_i = 0 \\ |A_i| \geq \lambda_i &: \quad O_i = A_i - \lambda_i \mathrm{sign}\,(A_i) \end{aligned} \tag{A.4}$$

Note that $L_{ij} = -\vec{W}_i\vec{W}_j$ and $\vec{W}_i^2 = 1$, as defined in section 3.2.

Theorem: For a given input vector $\vec{X}$, the error function E defined by eqn. A.1 assumes a minimum, if the outputs O_i are given by eqns. A.2 and A.3.

Proof: E is an almost everywhere twice continuously differentiable function with respect to the O_i, all higher derivatives are zero. At $O_i = 0$, E is not differentiable.

Thus, for the purpose of this proof, the unit indices will be divided in two disjoint sets:

$$I_0 = \{i \in \{1,\dots,M\} | O_i = 0\} \tag{A.5}$$

$$I_1 = \{i \in \{1,\dots,M\} | O_i \neq 0\} \tag{A.6}$$

- $i \in I_1$

  Here, the output value lies in the differentiable region of E. A necessary condition for a minimum of a differentiable function is the vanishing of the first derivative, which is given by

$$\frac{\partial E}{\partial O_i} = -\vec{X}\vec{W}_i + \sum_{j=1}^{M} \vec{W}_i\vec{W}_j O_j + \lambda_i \text{sign}\,(O_i) \tag{A.7}$$

  Only the units with nonzero output contribute to the sum in this term, hence it is sufficient to run the sum over $j \in I_1$:

$$\frac{\partial E}{\partial O_i} = -\vec{X}\vec{W}_i + \sum_{j\in I_1} \vec{W}_i\vec{W}_j O_j + \lambda_i \text{sign}\,(O_i) \tag{A.8}$$

  By splitting the sum into the parts where $j = i$ and $j \neq i$, one obtains:

$$\frac{\partial E}{\partial O_i} = -\vec{X}\vec{W}_i + \underbrace{\vec{W}_i^2}_{=1} O_i + \lambda_i \text{sign}\,(O_i) + \sum_{j\in I_1, j\neq i} \vec{W}_i\vec{W}_j O_j \tag{A.9}$$

  With the eqns. A.2,A.3 and A.4, the output O_i can be written in the following form (noting that sign (A_i) = sign (O_i) because $O_i \neq 0$):

$$O_i = \vec{X}\vec{W}_i - \sum_{j\in I_1, j\neq i} \vec{W}_i\vec{W}_j O_j - \lambda_i \text{sign}\,(O_i) \tag{A.10}$$

  Again, the sum over all possible i was replaced with the sum over $i \in I_1$, since the other units give no contribution.

  Finally, substituting A.10 into A.9, one obtains

$$\begin{aligned} \frac{\partial E}{\partial O_i} &= -\vec{X}\vec{W}_i + \left(\vec{X}\vec{W}_i - \sum_{j\in I_1, j\neq i} \vec{W}_i\vec{W}_j O_j - \lambda_i \text{sign}\,(O_i)\right) + \lambda_i \text{sign}\,(O_i) \\ &+ \sum_{j\in I_1, j\neq i} \vec{W}_i\vec{W}_j O_j \\ &= 0 \end{aligned} \tag{A.11}$$

  Hence, if the output of a unit is nonzero and given by the eqns. A.2,A.3 and A.4 , it will assume a value such that the derivative of E with respect to this output vanishes. In this case, it is also a sufficient condition for a minimum, as E is a quadratic form which is > 0 for a sufficiently large $|O_i|$.

- $i \in I_0$

  Here, the output is zero and E is not differentiable at that point. Hence, there is no derivative that could possibly vanish. But instead of requiring this, it is sufficient that the derivative be negative in an infinitesimal vicinity to the left of the point, and positive to the right. If $O_i = 0$ then eqn. A.4 requires that

$$|A_i| = |\vec{X}\vec{W}_i - \sum_{j \in I_1} \vec{W}_i \vec{W}_j O_j| < \lambda_i \tag{A.12}$$

  The condition $j \neq i$ could be dropped from the sum, because $i \in I_0$. Hence,

$$\begin{aligned} \lim_{O_i \to 0^-} \frac{\partial E}{\partial O_i} &= -\vec{X}\vec{W}_i + \sum_{j \in I_1} \vec{W}_i \vec{W}_j O_j + \lambda_i \mathrm{sign}\,(O_i) \qquad \text{(A.13)} \\ &= -A_i - \lambda_i \\ &< 0 \end{aligned}$$

  And, for the right side limit

$$\begin{aligned} \lim_{O_i \to 0^+} \frac{\partial E}{\partial O_i} &= -\vec{X}\vec{W}_i + \sum_{j \in I_1} \vec{W}_i \vec{W}_j O_j + \lambda_i \mathrm{sign}\,(O_i) \qquad \text{(A.14)} \\ &= -A_i + \lambda_i \\ &> 0 \end{aligned}$$

  Thus, E assumes a minimum in the direction of this O_i at $O_i = 0$.

Since every O_i belongs to either I_0 or I_1, a minimum of E is located at the O_i's given by eqns. A.2, A.3 and A.4.

$\square$

Appendix B

Convergence of sequential updating

In section 3.2.1, a sequential updating algorithm for activating the network discussed in this chapter is introduced. It remains to show that this algorithm is indeed converging towards the minimum of the network's error function:

$$E = \frac{1}{2}\left(\vec{X} - \sum_{j=1}^{M} \vec{W}_j O_j\right)^2 + \sum_{j=1}^{M} \lambda_j |O_j| \tag{B.1}$$

The update rules are:

$$A_i^{new} = \vec{X}\vec{W}_i - \sum_{j=1, j\neq i}^{m} \vec{W}_i \vec{W}_j O_j^{old} \tag{B.2}$$

$$O_i^{new} = O\left(A_i^{new}\right) \tag{B.3}$$

$$O_i^{old} = O_i^{new} \tag{B.4}$$

where $O(A)$ is defined as

$$\begin{aligned} -\lambda_i < A_i < \lambda_i &: \quad O_i = 0 \\ |A_i| \geq \lambda_i &: \quad O_i = A_i - \lambda_i \mathrm{sign}\left(A_i\right) \end{aligned} \tag{B.5}$$

(All $\lambda_i > 0$, all $|\vec{W}_i| = 1$).

Starting from an arbitrary configuration of outputs, one unit is updated at a time. In the following, it will be demonstrated that application of these rules either decreases E or keeps it constant.

A unit's output O_i can be positive, negative or zero before the update. Afterwards, it will either have retained or reversed its sign, or gone to zero. Each of these cases will be considered separately.

1. O_i positive before the update, positive afterwards.

   Firstly, E is rewritten as a sum of the parts that contain O_i, which will be called E_i and the rest, E_r:

$$\begin{aligned}
E &= \frac{1}{2}\left(\vec{X}^2 - 2\vec{X}\sum_{j=1}^{M}\vec{W}_jO_j + \left(\sum_{j=1}^{M}\vec{W}_jO_j\right)^2\right) + \sum_{j=1}^{M}\lambda_j|O_j| \\
&= \underbrace{\frac{1}{2}\left(-2\vec{X}\vec{W}_iO_i + \vec{W}_i^2O_i^2 + 2\sum_{j\neq i}^{M}\vec{W}_i\vec{W}_jO_iO_j\right) + \lambda_i|O_i|}_{E_i} \\
&\quad \underbrace{+\frac{1}{2}\left(\vec{X}^2 - 2\vec{X}\sum_{j\neq i}^{M}\vec{W}_jO_j + \sum_{j\neq i}^{M}\sum_{k\neq i}^{M}\vec{W}_j\vec{W}_kO_jO_k\right) + \sum_{j\neq i}\lambda_j|O_j|}_{E_r}
\end{aligned}$$

   Upon updating O_i, only E_i will change, since E_r does not depend on O_i. Let E_i^{old} and E_i^{new} be E_i's values before and after the update, respectively. Then ΔE, the change of E brought about by the update, is:

$$\begin{aligned}
\Delta E &= E_i^{new} - E_i^{old} \\
&= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) + \underbrace{(-\vec{X}\vec{W}_i + \sum_{j\neq i}^{M}\vec{W}_i\vec{W}_jO_j)}_{-A_i^{new}}(O_i^{new} - O_i^{old}) \\
&\quad +\lambda_i(|O_i^{new}| - |O_i^{old}|) \qquad \text{(B.6)}
\end{aligned}$$

   All O_j where $j \neq i$ do not change with the update, hence $O_j^{new} = O_j^{old}$. Thus, using eqn. B.2 and noting that both O_i^{new} and O_j^{old} are positive, ΔE can be written as:

$$\Delta E = \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - (A_i^{new} - \lambda_i)(O_i^{new} - O_i^{old}) \qquad \text{(B.7)}$$

   Now $A_i^{new} - \lambda_i = O_i^{new}$ (see eqn. B.5), and hence

$$\begin{aligned}
\Delta E &= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - O_i^{new}(O_i^{new} - O_i^{old}) \\
&= -\frac{1}{2}(O_i^{old^2} - 2O_i^{old}O_i^{new} + O_i^{new^2}) \\
&= -\frac{1}{2}(O_i^{old} - O_i^{new})^2 \qquad \text{(B.8)} \\
&\leq 0
\end{aligned}$$

   Therefore, E either decreases or remains constant.

2. O_i positive before the update, zero afterwards.

   In this case, ΔE (eqn. B.6) becomes

$$\Delta E = -\frac{1}{2}O_i^{old^2} + A_i^{new}O_i^{old} - \lambda_i O_i^{old} \tag{B.9}$$

   Since $O_i^{new} = 0$, $A_i - \lambda_i \leq 0$ (eqn. B.5). Thus

$$\begin{aligned}\Delta E &= \underbrace{-\frac{1}{2}O_i^{old^2}}_{<0} + \underbrace{(A_i^{new} - \lambda_i)}_{\leq 0}\underbrace{O_i^{old}}_{>0} \\ &< 0\end{aligned}$$

   Therefore, E decreases.

3. O_i positive before the update, negative afterwards.

   Here, ΔE (eqn. B.6) is

$$\begin{aligned}\Delta E &= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - A_i^{new}(O_i^{new} - O_i^{old}) + \lambda_i(-O_i^{new} - O_i^{old}) \\ &= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - (A_i^{new} + \lambda_i)O_i^{new} + O_i^{old}(A_i^{new} - \lambda_i)\end{aligned}$$

   Since $O_i^{new} < 0$, $O_i^{new} = A_i^{new} + \lambda_i$ (eqn. B.5). Furthermore, $0 > O_i^{new} = A_i^{new} + \lambda_i > A_i^{new} - \lambda_i$. Thus

$$\begin{aligned}\Delta E &= \underbrace{-\frac{1}{2}(O_i^{new^2} + O_i^{old^2})}_{<0} + \underbrace{O_i^{old}}_{>0}\underbrace{(A_i^{new} - \lambda_i)}_{<0} \\ &< 0 \end{aligned} \tag{B.10}$$

   Therefore, E decreases.

4. O_i negative before the update, negative afterwards

   This case is similar to (1). ΔE is (noting that $O_i^{new} = A_i^{new} + \lambda_i$)

$$\begin{aligned}\Delta E &= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - A_i^{new}(O_i^{new} - O_i^{old}) - \lambda_i(O_i^{new} - O_i^{old}) \\ &= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - O_i^{new}(O_i^{new} - O_i^{old}) \\ &= -\frac{1}{2}(O_i^{new^2} - 2O_i^{new}O_i^{old} + O_i^{old^2}) \\ &= -\frac{1}{2}(O_i^{new} - O_i^{old})^2 \\ &\leq 0\end{aligned}$$

   Therefore, E either decreases or remains constant.

5. O_i negative before the update, zero afterwards

   This case is similar to (2). ΔE becomes

$$\Delta E = -\frac{1}{2}O_i^{old^2} + A_i^{new}O_i^{old} + \lambda_i O_i^{old} \tag{B.11}$$

   Since $O_i^{new} = 0$, $A_i + \lambda_i \geq 0$ (eqn. B.5). Thus

$$\begin{aligned} \Delta E &= \underbrace{-\frac{1}{2}O_i^{old^2}}_{<0} + \underbrace{(A_i^{new} + \lambda_i)}_{\geq 0}\underbrace{O_i^{old}}_{<0} \\ &< 0 \end{aligned}$$

   Therefore, E decreases.

6. O_i negative before the update, positive afterwards

   This case is similar to (3). ΔE becomes

$$\begin{aligned} \Delta E &= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - A_i^{new}(O_i^{new} - O_i^{old}) + \lambda_i(O_i^{new} + O_i^{old}) \\ &= \frac{1}{2}(O_i^{new^2} - O_i^{old^2}) - (A_i^{new} - \lambda_i)O_i^{new} + O_i^{old}(A_i^{new} + \lambda_i) \end{aligned}$$

   Since $O_i^{new} > 0$, $O_i^{new} = A_i^{new} - \lambda_i$ (eqn. B.5). Furthermore, $0 < O_i^{new} = A_i^{new} - \lambda_i < A_i^{new} + \lambda_i$. Thus

$$\begin{aligned} \Delta E &= \underbrace{-\frac{1}{2}(O_i^{new^2} + O_i^{old^2})}_{<0} + \underbrace{O_i^{old}}_{<0}\underbrace{(A_i^{new} + \lambda_i)}_{>0} \\ &< 0 \end{aligned} \tag{B.12}$$

   Therefore, E decreases.

7. O_i zero before the update, zero afterwards.

   $\Delta E = 0$ in this case. Hence, E remains constant.

8. O_i zero before the update, positive afterwards.

   Here, ΔE becomes

$$\begin{aligned} \Delta E &= \frac{1}{2}O_i^{new^2} - A_i^{new}O_i^{new} + \lambda_i O_i^{new} \\ &= \frac{1}{2}O_i^{new^2} - (A_i^{new} - \lambda_i)O_i^{new} \end{aligned}$$

Since $O_i^{new} > 0$, $O_i^{new} = A_i^{new} - \lambda_i$, and hence

$$\begin{aligned}\Delta E &= \frac{1}{2}O_i^{new^2} - O_i^{new^2} \\ &= -\frac{1}{2}O_i^{new^2} \\ &< 0\end{aligned}$$

Therefore, E decreases.

9. O_i zero before the update, negative afterwards.

   This case is similar to (8). ΔE is

$$\begin{aligned}\Delta E &= \frac{1}{2}O_i^{new^2} - A_i^{new}O_i^{new} - \lambda_i O_i^{new} \\ &= \frac{1}{2}O_i^{new^2} - (A_i^{new} + \lambda_i)O_i^{new}\end{aligned}$$

   Since $O_i^{new} < 0$, $O_i^{new} = A_i^{new} + \lambda_i$, and hence

$$\begin{aligned}\Delta E &= \frac{1}{2}O_i^{new^2} - O_i^{new^2} \\ &= -\frac{1}{2}O_i^{new^2} \\ &< 0\end{aligned}$$

   Therefore, E decreases.

Hence an update step according to eqns. B.2 will either decrease E or leave it constant.

□

Appendix C

Population decoding

Suppose we had a population of N cells, each of which was able to identify the 'best' stimulus s_b, member of a stimulus set S of size C, with probability P_b, and that the remaining probability was evenly shared between the other stimuli, which is the maximum entropy assumption in the absence of further information:

$$\begin{aligned} P(s = s_b | x_i) &= P_b && \text{(C.1)} \\ P(s \neq s_b | x_i) &= \frac{1 - P_b}{C - 1} && \text{(C.2)} \end{aligned}$$

where x_i is the response of the i-th cell to the presentation of s_b. Assuming with [27] that the responses are independent given the stimulus

$$P(x_1, \ldots, x_N | s) = \prod_{i=1}^{N} P(x_i | s) \tag{C.3}$$

we then find

$$P(s_b | x_1, \ldots, x_N) = \frac{\prod_{i=1}^{N} P(x_i | s_b) P(s_b)}{\sum_{s \in S} \prod_{i=1}^{N} P(x_i | s) P(s)} \tag{C.4}$$

Since

$$P(x_i | s) = \frac{P(s | x_i) P(x_i)}{P(s)} \tag{C.5}$$

and, assuming an uniform prior over the stimuli in S, $P(s) = \frac{1}{C}$, eqn. (C.4) can be rewritten as

$$\begin{aligned} P(s | x_1, \ldots, x_N) &= \frac{\prod_{i=1}^{N} \frac{P(s_b|x_i)P(x_i)}{P(s_b)} P(s_b)}{\sum_{s \in S} \prod_{i=1}^{N} \frac{P(s|x_i)P(x_i)}{P(s)} P(s)} \\ &= \frac{P_b^N C^{N-1} \prod_{i=1}^{N} P(x_i)}{P_b^N C^{N-1} \prod_{i=1}^{N} P(x_i) + \sum_{s \in S, s \neq s_b} \left(\frac{1-P_b}{C-1}\right)^N C^{N-1} \prod_{i=1}^{N} P(x_i)} \\ &= \frac{p_b^N}{p_b^N + \frac{(1-P_b)^N}{(C-1)^{N-1}}} && \text{(C.6)} \end{aligned}$$

If the maximum entropy assumption (C.2) had not been made, then (C.6) would be

$$P(s|x_1,\ldots,x_N) = \frac{P_b^N}{P_b^N + \sum_{s\in S, s\neq s_b} P_s^N} \tag{C.7}$$

where $P_s = P(s|x_i)$, i.e. it is still assumed that all cells behave alike. Let

$$Z = \sum_{s\in S, s\neq s_b} P_s^N = \sum_{s\in S, s\neq s_b, s\neq r} P_s^N + (1 - P_b - \sum_{s\in S, s\neq s_b, s\neq r} P_s)^N \tag{C.8}$$

where the last equality is due to the normalization constraint $P_b + \sum_{s\in S, s\neq s_b} P_s = 1$ and r is the stimulus whose probability is therefore given by all the others. The extremum of Z is found by setting its derivatives w.r.t. the P_s to 0:

$$\frac{\partial Z}{\partial P_s} = NP_s^{N-1} - N(1 - P_b - \sum_{s\in S, s\neq s_b, s\neq r} P_s)^{N-1} \overset{!}{=} 0 \tag{C.9}$$

which is fulfilled for $P_s = \frac{1-P_b}{C-1}$, i.e. (C.2). Moreover, the second derivatives at this point are

$$\begin{aligned} \frac{\partial^2 Z}{\partial P_s \partial P_{s'}} &= \delta_{ss'} N(N-1)\left(\frac{1-P_b}{C-1}\right)^{(N-2)} + N(N-1)\left(\frac{1-P_b}{C-1}\right)^{(N-2)} \\ &= (1+\delta_{ss'})N(N-1)\left(\frac{1-P_b}{C-1}\right)^{(N-2)} \end{aligned} \tag{C.10}$$

where $\delta_{ss'}$ is the Kronecker delta. Therefore, the Hessian is positive definite, and (C.2) is the location of the minimum of Z. Hence, (C.6) is an upper bound on $P(s|x_1,\ldots,x_N)$.

Appendix D

Dirichlet densities

D.1 Normalization Integral of a Dirichlet density

Priors, evidences, entropies and expected values of Dirichlet-distributed probability distributions can be expressed by Beta and Gamma functions and their derivatives (see e.g. [17] for a comprehensive collection of their properties). Let

$$p(\{P_m\}) = \frac{1}{N(\{n_m\})} \prod_{m=0}^{M} P_m^{n_m} \delta(1 - \sum_{m=0}^{M} P_m) \tag{D.1}$$

where $n_m \in I\!N_0$ and $N(\{n_m\})$ is a normalization constant. Thus

$$N(\{n_m\}) = \int_0^1 dP_0 \dots \int_0^1 dP_M p(\{P_m\}) \tag{D.2}$$

Due to the $\delta()$, the rightmost integral can be carried out immediately if $0 \leq 1 - \sum_{m=0}^{M-1} P_m = P_M \leq 1$. Otherwise, the integral is zero. Therefore, by rewriting the integration boundaries to include only possible nonzero contributions, one obtains:

$$\begin{aligned} N(\{n_m\}) &= \int_0^1 dP_0 \dots \int_0^{1-P_0} dP_1 \dots \int_0^{1-\sum_{m=0}^{M-2} P_m} dP_{M-1} \times \\ &\times \prod_{m=0}^{M-1} P_m^{n_m} (1 - \sum_{m=0}^{M-1} P_m)^{n_M} \\ &= \int_0^1 dP_0 P_0^{n_0} \dots \int_0^{1-\sum_{m=0}^{M-3} P_m} dP_{M-2} P_{M-2}^{n_{M-2}} \times \\ &\times \int_0^{1-\sum_{m=0}^{M-2} P_m} dP_{M-1} P_{M-1}^{n_{M-1}} (1 - \sum_{m=0}^{M-2} P_m - P_{M-1})^{n_M} \end{aligned} \tag{D.3}$$

By substituting $z = 1 - \sum_{m=0}^{M-2} P_m$, the rightmost integral now becomes

$$\int_0^z P_{M-1}^{n_{M-1}} (z - P_{M-1})^{n_M} dP_{M-1} \tag{D.4}$$

which can be rewritten as (setting $\tilde{P} = \frac{P_{M-1}}{z}$)

$$z^{n_{M-1}+n_M+1} \int_0^1 \tilde{P}^{n_{M-1}}(1-\tilde{P})^{n_M} d\tilde{P} = z^{n_{M-1}+n_M+1} B(n_{M-1}+1, n_M+1) \tag{D.5}$$

where $B(a,b)$ is the *Beta function.* It is connected to the Gamma function via

$$B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{D.6}$$

If the arguments are $\in I\!N_0$, the value of the Gamma function is given by a factorial:

$$\Gamma(n+1) = n! \tag{D.7}$$

Therefore, by successively integrating over all the P_m, eqn. (D.3) becomes

$$\begin{aligned} N(\{n_m\}) &= \frac{n_{M-1}!n_M!}{(n_{M-1}+n_M+1)!} \int_0^1 dP_1 P_1^{n_1} \dots \int_0^{1-\sum_{m=1}^{M-3} P_m} dP_{M-2} P_{M-2}^{n_{M-2}} \times \\ &\quad \times (1 - \sum_{m=1}^{M-3} P_m - P_{M-2})^{(n_{M-1}+n_M+1)} \\ N(\{n_m\}) &= \frac{\prod_{m=0}^M n_m!}{(\sum_{m=0}^M n_m + M)!} = \frac{\prod_{m=0}^M \Gamma(n_m+1)}{\Gamma(\sum_{m=0}^M n_m + M + 1)} \end{aligned} \tag{D.8}$$

In (D.1), a uniform prior over the P_m was assumed. The generalization to priors of the form

$$p(\{P_m\}) \propto \prod_{m=0}^M P_m^{\theta-1} \tag{D.9}$$

where $\theta \in I\!R^+$ is straightforward: (D.1) now becomes

$$p(\{P_m\}) = \frac{1}{N(\{n_m\},\theta)} \prod_{m=0}^M P_m^{n_m+\theta-1} \delta(1 - \sum_{m=0}^M P_m) \tag{D.10}$$

and thus (D.5) is

$$\begin{aligned} & z^{n_{M-1}+n_M+1+2\theta-2} \int_0^1 \tilde{P}^{n_{M-1}+\theta-1}(1-\tilde{P})^{n_M+\theta-1} d\tilde{P} \\ = \; & z^{n_{M-1}+n_M+1+2\theta-2} B(n_{M-1}+\theta, n_M+\theta) \end{aligned} \tag{D.11}$$

Since the Beta function diverges if one of its arguments is ≤ 0, θ has to be greater than zero. Continuing the calculation in the same way as above yields

$$N(\{n_m\},\theta) = \frac{\prod_{m=0}^M \Gamma(n_m+\theta)}{\Gamma(\sum_{m=0}^M n_m + (M+1)\theta)} \tag{D.12}$$

(D.8) is recovered for $\theta = 1$.

D.2 Marginal densities

The marginal densities of (D.1) are themselves Dirichlet densities. Assume we wanted to compute the marginal density of a sum over a subset of the P_m, called $\tilde{P}$, and the rest . First, permute the indices such that $\tilde{P} = \sum_{m=k}^{M} P_m$. Due to the normalization constraint, $\tilde{P} = 1 - \sum_{m=0}^{k-1} P_m$, and therefore

$$\begin{aligned} p(\{P_0, \ldots, P_{k-1}, \tilde{P}\}) &= p(\{P_0, \ldots, P_{k-1}, 1 - \sum_{m=0}^{k-1} P_m\}) \\ &= \int_0^{1-\sum_{m=0}^{k} P_m} dP_k \ldots \int_0^{1-\sum_{m=0}^{M-2} P_m} dP_{M-1} p(\{P_m\}) \end{aligned} \tag{D.13}$$

Carrying out the integrations in the same fashion as above and setting $\tilde{n} = \sum_{m=k}^{M} n_m$, this yields

$$\begin{aligned} p(\{P_0, \ldots, P_{k-1}, \tilde{P}\}) &= \frac{1}{N(\{n_m\})} \frac{\prod_{m=k}^{M} n_m}{(\tilde{n} + M - k)} P_0^{n_0} \ldots P_{k-1}^{n_{k-1}} \tilde{P}^{\tilde{n}} \\ &= \frac{(\sum_{m=0}^{M} n_m + M)!}{\prod_{m=0}^{k-1} n_m! (\tilde{n} + M - k)!} P_0^{n_0} \ldots P_{k-1}^{n_{k-1}} \tilde{P}^{\tilde{n}} \\ &= \frac{1}{N(\{n_0, \ldots, n_{k-1}, \tilde{n}\})} P_0^{n_0} \ldots P_{k-1}^{n_{k-1}} \tilde{P}^{\tilde{n}} \end{aligned} \tag{D.14}$$

The generalization for $\theta \neq 1$ is thus:

$$p(\{P_0, \ldots, P_{k-1}, \tilde{P}\}) = \frac{P_0^{n_0+\theta-1} \ldots P_{k-1}^{n_{k-1}+\theta-1} \tilde{P}^{\tilde{n}+(M-k+1)(\theta-1)}}{N(\{n_0, \ldots, n_{k-1}, \tilde{n}\}, \theta)} \tag{D.15}$$

where

$$N(\{n_m\}, \theta) = \frac{\prod_{m=0}^{k-1} \Gamma(n_m + \theta) \Gamma(\tilde{n} + (M - k + 1)\theta)}{\Gamma(\sum_{m=0}^{M} n_m + (M + 1)\theta)} \tag{D.16}$$

D.3 Derivatives of Gamma and Beta functions

The Gamma function can be defined via the recurrence relation (see e.g. [9])

$$\begin{aligned} \Gamma(0) &= 1 \\ \Gamma(x + 1) &= x\Gamma(x) \end{aligned} \tag{D.17}$$

which – for $x \in I\!N_0$ – evaluates to $\Gamma(x+1) = x!$. Thus, its derivative w.r.t. x is given by

$$\Gamma'(x+1) = \frac{d\Gamma(x+1)}{dx} = \Gamma(x) + x\Gamma'(x) \tag{D.18}$$

This equation is fulfilled by

$$\forall x \in I\!N_0 : \Gamma'(x+1) = x!(\sum_{i=1}^{x} \frac{1}{i} + \Gamma'(1)) \tag{D.19}$$

which can be verified easily by substituting (D.19) into (D.18):

$$\begin{aligned}\Gamma'(x+1) &= (x-1)! + x(x-1)!(\sum_{i=1}^{x-1} \frac{1}{i} + \Gamma'(1)) \\ &= x!(\frac{1}{x} + \sum_{i=1}^{x-1} \frac{1}{i} + \Gamma'(1)) \\ &= x!(\sum_{i=1}^{x} \frac{1}{i} + \Gamma'(1))\end{aligned} \tag{D.20}$$

To compute $\Gamma'(1)$, one can employ an identity due to Weierstraß:

$$\frac{\Gamma'(x)}{\Gamma(x)} = -\gamma - \frac{1}{x} + \sum_{i=1}^{\infty} \frac{x}{i(i+x)} \tag{D.21}$$

Since $\Gamma(1) = 1$ and $\sum_{i=1}^{\infty} \frac{1}{i(i+1)} = 1$,

$$\Gamma'(1) = -\gamma \tag{D.22}$$

where $\gamma = 0.57721566\ldots$ is the *Euler-Mascheroni* constant. Differentiating (D.18) again yields

$$\Gamma''(x+1) = 2\Gamma'(x) + x\Gamma''(x) \tag{D.23}$$

which is fulfilled by

$$\forall x \in I\!N_0 : \Gamma''(x+1) = 2x! \sum_{i=1}^{x} \frac{1}{i}(\sum_{j=1}^{i-1} \frac{1}{j} - \gamma) + x!\Gamma''(1) \tag{D.24}$$

This can be seen by substituting (D.24) and (D.19) into (D.23):

$$\begin{aligned}\Gamma''(x+1) &= 2(x-1)!(\sum_{i=1}^{x-1} \frac{1}{i} - \gamma) + x2(x-1)! \sum_{i=1}^{x-1} \frac{1}{i}(\sum_{j=1}^{i-1} \frac{1}{j} - \gamma) \\ &\quad +x(x-1)!\Gamma''(1) \\ &= 2\frac{x!}{x}(\sum_{i=1}^{x-1} \frac{1}{i} - \gamma) + 2x! \sum_{i=1}^{x-1} \frac{1}{i}(\sum_{j=1}^{i-1} \frac{1}{j} - \gamma) + x!\Gamma''(1) \\ &= 2x! \sum_{i=1}^{x} \frac{1}{i}(\sum_{j=1}^{i-1} \frac{1}{j} - \gamma) + x!\Gamma''(1)\end{aligned} \tag{D.25}$$

To obtain $\Gamma''(1)$, multiply (D.21) with $\Gamma(x)$ and differentiate on both sides:

$$\begin{aligned}\Gamma''(x) &= -\gamma\Gamma'(x) - \frac{\Gamma'(x)}{x} + \frac{\Gamma(x)}{x^2} + \Gamma(x)\sum_{i=1}^{\infty}\left(\frac{1}{i(x+i)} - \frac{x}{i(x+i)^2}\right) \\ &\quad +\Gamma'(x)\sum_{i=1}^{\infty}\frac{x}{i(x+i)} \qquad \text{(D.26)}\end{aligned}$$

Thus, using (D.22) and noting that $\sum_{i=1}^{\infty}\frac{1}{i^2} = \frac{\pi^2}{6}$ (see e.g. [9])

$$\begin{aligned}\Gamma''(1) &= \gamma^2 + \gamma + 1 + \sum_{i=1}^{\infty}\frac{1}{(i+1)^2} - \gamma \\ &= \gamma^2 + 1 + \sum_{i=1}^{\infty}\frac{1}{i^2} - 1 \\ &= \gamma^2 + \frac{\pi^2}{6} \qquad \text{(D.27)}\end{aligned}$$

Equipped with these results, it is now possible to compute the derivatives of $B(a+1, b+1)$ for $a, b \in \mathbb{N}_0$: let $c = \frac{a!b!}{(a+b+1)!} = B(a+1, b+1)$ and define

$$h_a^b = \sum_{i=a}^{b}\frac{1}{i} \qquad \text{(D.28)}$$

$${}_2h_a^b = \sum_{i=a}^{b}\frac{1}{i^2} \qquad \text{(D.29)}$$

Then

$$\begin{aligned}\frac{\partial B(a+1,b+1)}{\partial a} &= \Gamma(b+1)\left(\frac{\Gamma'(a+1)}{\Gamma(a+b+2)} - \frac{\Gamma(a+1)\Gamma'(a+b+2)}{\Gamma^2(a+b+2)}\right) \\ &= c\left(h_1^a - \gamma - h_1^{a+b+1} + \gamma\right) \\ &= -c\, h_{a+1}^{a+b+1} \qquad \text{(D.30)}\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 B(a+1,b+1)}{\partial a^2} &= \Gamma(b+1)\left(\frac{\Gamma''(a+1)}{\Gamma(a+b+2)} - 2\frac{\Gamma'(a+1)\Gamma'(a+b+2)}{\Gamma^2(a+b+2)}\right. \\ &\quad \left.-\frac{\Gamma(a+1)\Gamma''(a+b+2)}{\Gamma(a+b+2)} + 2\frac{\Gamma(a+1)\Gamma'^2(a+b+2)}{\Gamma^3(a+b+2)}\right) \\ &= c\left[2\sum_{i=1}^{a}\frac{1}{i}\left(h_1^{i-1} - \gamma\right) + \gamma^2 + \frac{\pi^2}{6}\right. \\ &\quad -2\left(h_1^a - \gamma\right)\left(h_1^{a+b+1} - \gamma\right) \\ &\quad -2\sum_{i=1}^{a+b+1}\frac{1}{i}\left(h_1^{i-1} - \gamma\right) - \left(\gamma^2 + \frac{\pi^2}{6}\right)\end{aligned}$$

$$
\begin{aligned}
& +2\left(h_1^{a+b+1}-\gamma\right)^2\Big] \\
= &\ 2c\sum_{i=a+1}^{a+b+1}\frac{1}{i}h_i^{a+b+1} \\
= &\ c\left(h_{a+1}^{a+b+1}\right)^2+c\,{}_2h_{a+1}^{a+b+1} \qquad \text{(D.31)}
\end{aligned}
$$

(D.45) was used for the last step.

$$
\begin{aligned}
\frac{\partial^2 B(a+1,b+1)}{\partial a\partial b} = &\ \frac{\Gamma'(b+1)\Gamma'(a+1)}{\Gamma(a+b+2)}-\frac{\Gamma(b+1)\Gamma'(a+1)\Gamma'(a+b+2)}{\Gamma^2(a+b+2)} \\
&-\frac{\Gamma'(b+1)\Gamma(a+1)\Gamma'(a+b+2)}{\Gamma^2(a+b+2)} \\
&-\frac{\Gamma(b+1)\Gamma(a+1)\Gamma''(a+b+2)}{\Gamma^2(a+b+2)} \\
&+2\frac{\Gamma(b+1)\Gamma(a+1)\Gamma'^2(a+b+1)}{\Gamma^3(a+b+2)} \\
= &\ c\left[\left(h_1^a-\gamma\right)\left(h_1^b-\gamma\right)-\left(h_1^a-\gamma\right)\left(h_1^{a+b+1}-\gamma\right)\right. \\
&-\left(h_1^b-\gamma\right)\left(h_1^{a+b+1}-\gamma\right) \\
&-2\sum_{i=1}^{a+b+1}\frac{1}{i}\left(h_1^{i-1}-\gamma\right)-\left(\gamma^2+\frac{\pi^2}{6}\right) \\
&\left.+2\left(h_1^{a+b+1}-\gamma\right)^2\right] \\
= &\ c\left[h_1^a h_1^b-h_1^{a+b+1}\left(h_1^a+h_1^b\right)-\frac{\pi^2}{6}\right. \\
&\left.-2\sum_{i=1}^{a+b+1}\frac{1}{i}\left(h_1^{i-1}\right)+2\left(h_1^{a+b+1}\right)^2\right] \\
= &\ c\left[\left(h_1^{a+b+1}\right)^2+\left(h_{b+1}^{a+b+1}\right)\left(h_{a+1}^{a+b+1}\right)\right. \\
&\left.-2\sum_{i=1}^{a+b+1}\frac{1}{i}h_1^{i-1}-\frac{\pi^2}{6}\right] \\
= &\ c\left[\left(h_{b+1}^{a+b+1}\right)\left(h_{a+1}^{a+b+1}\right)+{}_2h_1^{a+b+1}-\frac{\pi^2}{6}\right] \qquad \text{(D.32)}
\end{aligned}
$$

Generalizing these results to allow for non-uniform priors of type (D.9) is feasible. Define

$$
h_a^b(\theta) = \sum_{i=a}^{b}\frac{1}{i+\theta} \qquad \text{(D.33)}
$$

$$
{}_2h_a^b(\theta) = \sum_{i=a}^{b}\frac{1}{(i+\theta)^2} \qquad \text{(D.34)}
$$

After some lenghty but straightforward algebra – along the same lines as the calculation for $\theta = 1$ – one obtains ($x \in I\!N_0$):

$$\frac{\Gamma'(x+\theta)}{\Gamma(x+\theta)} = h_0^{x-1}(\theta) + \Psi(\theta) \tag{D.35}$$

$$\frac{\Gamma''(x+\theta)}{\Gamma(x+\theta)} = \left(h_0^{x-1}(\theta)\right)^2 - {}_2h_0^{x-1}(\theta) + 2h_0^{x-1}(\theta)\Psi(\theta) + \frac{\Gamma''(\theta)}{\Gamma(\theta)} \tag{D.36}$$

where $\Psi(\theta) = \frac{d}{dx}\log(\Gamma(\theta)) = \frac{\Gamma'(\theta)}{\Gamma(\theta)}$ is the digamma function. Hence, (D.30), (D.31) and (D.32) become (now setting $c = B(a+\theta, b+\theta)$ and noting that $\Psi'(\theta) = \frac{\Gamma''(\theta)}{\Gamma(\theta)} - \Psi^2(\theta)$)

$$\begin{aligned} & \frac{\partial B(a+\theta, b+\theta)}{\partial a} \\ = \; & -c\left(h_0^{a+b-1}(2\theta) - h_0^{a-1}(\theta) + \Psi(2\theta) - \Psi(\theta)\right) \end{aligned} \tag{D.37}$$

$$\begin{aligned} & \frac{\partial^2 B(a+\theta, b+\theta)}{\partial a^2} \\ = \; & c\Big[\left(h_0^{a+b-1}(2\theta) - h_0^{a-1}(\theta) + \Psi(2\theta) - \Psi(\theta)\right)^2 \\ & + {}_2h_0^{a+b-1}(2\theta) - {}_2h_0^{a-1}(\theta) + \Psi'(\theta) - \Psi'(2\theta)\Big] \end{aligned} \tag{D.38}$$

$$\begin{aligned} & \frac{\partial^2 B(a+\theta, b+\theta)}{\partial a \partial b} \\ = \; & c\left[\left(h_0^{a+b-1}(2\theta) - h_0^{a-1}(\theta) + \Psi(2\theta) - \Psi(\theta)\right)\right. \\ & \times \left(h_0^{a+b-1}(2\theta) - h_0^{b-1}(\theta) + \Psi(2\theta) - \Psi(\theta)\right) \\ & \left. + {}_2h_0^{a+b-1}(2\theta) - \Psi'(2\theta)\right] \end{aligned} \tag{D.39}$$

D.4 An upper bound on the variance of a sum of random variables

Since

$$\begin{aligned} \mathrm{Var}[x-y] &= E\left[x^2\right] - 2E[xy] + E[y]^2 - E[x]^2 + 2E[x]E[y] - E[y]^2 \geq 0 \\ &\to \mathrm{Var}[x] + \mathrm{Var}[y] \geq 2(E[xy] - E[x]E[y]) \end{aligned} \tag{D.40}$$

we have

$$\begin{aligned}\mathrm{Var}\left[x+y\right] &= E\left[x^2\right]+2E\left[xy\right]+E\left[y\right]^2-E\left[x\right]^2-2E\left[x\right]E\left[y\right]-E\left[y\right]^2\\ &\leq 2(\mathrm{Var}\left[x\right]+\mathrm{Var}\left[y\right]) \qquad \text{(D.41)}\end{aligned}$$

This can be generalized to

$$\begin{aligned}\mathrm{Var}\left[\sum_{i=1}^{N}x_i\right] &= \sum_{i=1}^{N}(E\left[x_i\right]^2-E\left[x_i\right]^2)+2\sum_{i=1}^{N}\sum_{j=i+1}^{N}(E\left[x_ix_j\right]-E\left[x_i\right]E\left[x_j\right])\\ &\leq \sum_{i=1}^{N}\mathrm{Var}\left[x_i\right]+\sum_{i=1}^{N}\sum_{j=i+1}^{N}(\mathrm{Var}\left[x_i\right]+\mathrm{Var}\left[x_j\right])\\ &= \sum_{i=1}^{N}\mathrm{Var}\left[x_i\right]+\sum_{i=1}^{N}\mathrm{Var}\left[x_i\right](N-i)+\sum_{j=2}^{N}\sum_{i=1}^{j-1}\mathrm{Var}\left[x_j\right]\\ &= \sum_{i=1}^{N}\mathrm{Var}\left[x_i\right](1+N-i)+\sum_{j=2}^{N}(j-1)\mathrm{Var}\left[x_j\right]\\ &= \sum_{i=2}^{N}\mathrm{Var}\left[x_i\right](1+N-i+i-1)+N\mathrm{Var}\left[x_1\right]\\ \to \mathrm{Var}\left[\sum_{i=1}^{N}x_i\right] &\leq N\sum_{j=1}^{N}\mathrm{Var}\left[x_i\right] \qquad \text{(D.42)}\end{aligned}$$

D.5 Some identities for squares of sums

Since

$$\begin{aligned}\left(h_a^b\right)^2 &= \sum_{i=a}^{b}\sum_{j=a}^{b}\frac{1}{ij}\\ &= \sum_{i=a}^{b}\frac{1}{i^2}+2\sum_{i=a}^{b}\sum_{j=a}^{i-1}\frac{1}{ij}\\ &= \sum_{i=a}^{b}\frac{1}{i^2}+2\sum_{i=a}^{b}\sum_{j=a}^{i}(\frac{1}{ij}-\frac{1}{i^2})\\ &= 2\sum_{i=a}^{b}\sum_{j=a}^{i}\frac{1}{ij}-\sum_{i=a}^{b}\frac{1}{i^2} \qquad \text{(D.43)}\end{aligned}$$

we have

$$\sum_{i=a}^{b}\sum_{j=a}^{i}\frac{1}{ij}=\frac{1}{2}\left(\left(h_a^b\right)^2+{}_2h_a^b\right) \qquad \text{(D.44)}$$

Likewise, exploiting $\sum_{i=a}^{b}\sum_{j=a}^{i}\frac{1}{ij}=\sum_{j=a}^{b}\sum_{i=j}^{b}\frac{1}{ij}$, we can also write

$$\sum_{i=a}^{b}\sum_{j=i}^{b}\frac{1}{ij}=\frac{1}{2}\left(\left(h_a^b\right)^2+{}_2h_a^b\right) \qquad \text{(D.45)}$$

Appendix E

Proof of the metric properties of D_{pq}

This appendix contains the proof that D_{pq} is a metric, its coding-theoretic motivation and some of its limiting properties, e.g. its boundedness. Metrics are the prerequisites for several important convergence theorems for iterative algorithms, e.g. Banach's fixed point theorem [10], which is the basis of various pattern-matching algorithms. Boundedness is a valuable property, too, when numerical applications are considered. The presentation of this material closely resembles the published form [22].

E.1 Motivation

The motivation presented in this section is aimed at providing the reader with an idea of the meaning of the metric in coding-theoretic terms. As such it is not to be understood as a derivation in a strict mathematical sense. However, mathematical rigor will be observed in the following section, which contains the actual proof of the metric properties.

Let X be a discrete random variable which can take on N different values $\in \Omega_N = \{\omega_1, \ldots, \omega_N\}$. Now, draw an *i.i.d.* sample $\tilde{X}$, where each observation is drawn from one of two known distributions, P and Q. Each of those is used with equal probability. However, it is unknown which one is used when. Now one wishes to find the coding strategy that gives the shortest average codelength for the representation of the data. In other words, what is the most *efficient* distribution R?

Let this code be called κ. The codelengths are $\kappa_i = -\log r_i$, where $i \in \{1, \ldots, N\}$ and r_i is the probability of $X = \omega_i$ under R. Denoting the expectation of κ w.r.t. P by $\mathcal{E}(\kappa, P)$, the average codelength $<\kappa>$ is then $\frac{1}{2}\mathcal{E}(\kappa, P) + \frac{1}{2}\mathcal{E}(\kappa, Q)$. By the very definition of the entropy, the *minimum* $<\kappa>$ is obtained by setting $R = \frac{1}{2}(P + Q)$, i.e. $<\kappa>= H(R)$.

An ideal observer, i.e. one who knows which distribution is used to generate the individual data, could reach an even shorter average codelength $\frac{1}{2}H(P) + \frac{1}{2}H(Q)$. Hence the redundancy of κ is $H(R) - \frac{1}{2}H(P) - \frac{1}{2}H(Q)$. The distance which will be studied is that redundancy

$$\begin{aligned} D^2_{PQ} &= H(R) - \frac{1}{2}H(P) - \frac{1}{2}H(Q) \\ &= \frac{1}{2}(D\left(P\| R\right) + D\left(Q\| R\right)) \\ &= \frac{1}{2}\sum_{i=1}^{N}\left(p_i \log\frac{2p_i}{p_i+q_i} + q_i \log\frac{2q_i}{p_i+q_i}\right) \qquad \text{(E.1)} \end{aligned}$$

Since the Kullback divergence $D\left(P\| R\right)$ can be interpreted as the inefficiency of assuming that the true distribution is R when it really is P, D^2_{PQ} could be seen as a minimum inefficiency distance.

This distance measure has been introduced before. Topsøe, in [86], called $2D^2_{PQ}$ *capacitory discrimination* and introduced it from an information transmission point of view. In that paper, its properties are studied in depth. His results will be related to those presented here in the discussion. Now D^2_{PQ} is obviously symmetric and vanishes for $P = Q$, but it does not fulfill the triangle inequality. However, its square root, D_{PQ}, does. The proof of the metric properties of D_{PQ} is the subject of the next section.

E.2 Proof of metric properties of $\mathbf{D_{PQ}}$

In the following, $\mathbb{R}^+$ includes 0.

Definition 1 *Let the function $L(p, q) : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ be defined by*

$$L(p, q) := p \log\frac{2p}{p+q} + q \log\frac{2q}{p+q}. \qquad \text{(E.2)}$$

This function can be taken to be any one of the summands of D^2_{PQ} (see eqn. (E.1)). By virtue of the standard inequality $\log(x) \geq 1 - \frac{1}{x}$ one realizes that $L(p,q) \geq 0$ with equality only for $p = q$.

Theorem 1 uses some properties of the partial derivative of $L(p,q)$ and to show these let the function $g : \mathbb{R}^+ \backslash \{1\} \to \mathbb{R}$ be defined by

$$g(x) := \frac{\log \frac{2}{x+1}}{\sqrt{L(x,1)}}.$$

Lemma 1 *Let g be defined as above. Then*

1. $\lim_{x\to 1^\mp} g(x) = \pm 1$, *i.e. g jumps from $+1$ to -1 at $x = 1$.*

2. *The derivative $\frac{\mathrm{d}}{\mathrm{d}x} g$ is positive for $x \in \mathbb{R}^+ \backslash \{1\}$.*

A consequence of this lemma is that $|g(x)| \leq 1$ with equality only at $x = 1$. Also, it is easy to see that $|g|$ is continuous, but not g.

Proof: First note that g changes sign at $x = 1$.

A straightforward application of l'Hôspital's rule (differentiate twice) yields $\lim_{x\to 1} g^2(x) = 1$.

By differentiation one finds that $\frac{\mathrm{d}}{\mathrm{d}x} g$ is positive if and only if $f < 0$ where f is given by

$$f(x) = 2\left(x \log \frac{2x}{1+x} + \log \frac{2}{1+x}\right) + (1+x) \log \frac{2}{1+x} \log \frac{2x}{1+x} \tag{E.3}$$

Thus,

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \log \frac{2}{1+x} \log \frac{2x}{1+x} + \log \frac{2x}{1+x} + \frac{1}{x} \log \frac{2}{1+x} \tag{E.4}$$

and

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x) = \frac{-1}{x^2(1+x)} \left(\log \frac{2}{1+x} + x^2 \log \frac{2x}{1+x}\right) \tag{E.5}$$

Hence, $f(1) = f'(1) = 0$. Using the standard inequality $\log a \geq 1 - \frac{1}{a}$, one finds that $f'' < 0$, hence f is concave. Combined with the first found facts, $f < 0$ for $x \neq 1 \diamond$.

It will now be proven that

Theorem 1 *Let $\mathcal{F}_N$ be the set of all discrete probability distributions over Ω_N, $N \in I\!N$. The function $D_{PQ} : \mathcal{F}_N \times \mathcal{F}_N \longrightarrow I\!R^+$ is a metric.*

Proof: To show this, recall that $D\,(P\|\,Q)$ is 0 for $P = Q$ and strictly positive otherwise (see e.g. [13]). In addition, D^2_{PQ} is symmetric in P, Q and so is D_{PQ}. Therefore, it only remains to be shown that the triangle inequality holds.

Lemma 2 *Let $p, q, r \in I\!R^+$. Then*

$$\sqrt{L(p,q)} \leq \sqrt{L(p,r)} + \sqrt{L(r,q)}.$$

Proof: It is easy to see that this holds if $p = q$ or $r = 0$. Now, assume $p \leq q$ and denote by **rhs** the right hand side as a function of r. It will be shown that

1. **rhs** has 2 minima, namely one at $r = p$ and one at $r = q$, and
2. only 1 maximum somewhere between p and q.

This can be accomplished by way of the derivative

$$\frac{\partial \mathbf{rhs}}{\partial r} = \frac{\log \frac{2r}{p+r}}{2 \cdot \sqrt{L(p,r)}} + \frac{\log \frac{2r}{q+r}}{2 \cdot \sqrt{L(q,r)}}. \tag{E.6}$$

With g as in Lemma 1 and $x := \frac{p}{r}$ and $\beta \cdot x := \frac{q}{r}$ ($\beta > 1$), one finds that

$$2 \cdot \sqrt{r} \cdot \frac{\partial \mathbf{rhs}}{\partial r} = g(x) + g(\beta x).$$

With $|g(x)| \leq 1$ with equality only at $x = 1$, and the fact that g jumps from $+1$ to -1 at $x = 1$ (see lemma 1), the derivative $\frac{\partial \mathbf{rhs}}{\partial r}$ indeed changes sign at $r = p$, because then $x = 1$ and $|g(x)| > |g(\beta x)|$, and likewise at $r = q$. Those extrema are minima because r is reciprocal to x.

Also, $\frac{\mathrm{d}}{\mathrm{d}x} g(x) \geq 0$, therefore between $x = \frac{1}{\beta}$ and $x = 1$, $g(x) + g(\beta x)$ is monotonic increasing and as a consequence has at most one sign change. This reasoning also holds if $p = 0$ or $q = 0$, because $g(0) = \sqrt{\log(2)} < 1$. $\diamond$

Applying Minkowski's inequality to the square root of the sum which defines D_{PQ}, one sees that the triangle inequality is fulfilled.

Whence D_{PQ} is a metric.$\diamond$

The generalization of this result to continuous random variables is straightforward. Let P and Q be probability measures defined on a measurable space (Ω, A)

and let $p = \frac{dP}{d\mu}$, $q = \frac{dQ}{d\mu}$ be their Radon-Nikodym derivatives w.r.t. a dominating σ-finite measure μ. Then

$$D_{PQ} = \sqrt{\frac{1}{2}\int_{\Omega}\left(p\log\frac{2p}{p+q} + q\log\frac{2q}{p+q}\right)d\mu} \tag{E.7}$$

is a metric, too.

An alternative proof could be constructed using results presented in [46]. Since D_{PQ}^2 is an instance of a class of distances known as f-divergences (cf. [1]) (let $f(t) = t\log\frac{2t}{1+t} + \log\frac{2}{1+t}$, then $D_{PQ}^2 = \frac{1}{2}\sum_{i=1}^{N} q_i f(\frac{p_i}{q_i})$), the theorems proven in [46] apply.

Now the maxima and minima of D_{PQ} will be studied. Its minimum is, of course, located at $P = Q$, where $D_{PQ} = 0$. To find its maximum, rewrite (E.2) in the form

$$L(p,q) = \underbrace{(p+q)\log 2}_{\geq 0} + \underbrace{p\log\left(\frac{p}{p+q}\right)}_{\leq 0} + \underbrace{q\log\left(\frac{q}{p+q}\right)}_{\leq 0} \tag{E.8}$$

It follows that when P and Q are two distinct deterministic distributions, D_{PQ} assumes its maximum value $\sqrt{\log 2}$. If the logarithm to base 2 is used, then the maximum value of D_{PQ}^2 is 1 bit.

E.3 Asymptotic approximation

Next, the limit

$$\lim_{P\to Q} D_{PQ}^2 \tag{E.9}$$

shall be investigated. A term-by-term expansion of D_{PQ} to second order in p_j yields:

$$D_{PQ}^2 \approx \sum_{j=1}^{N}\frac{1}{8q_j}(p_j - q_j)^2 = \frac{1}{8}\chi^2(P,Q) \tag{E.10}$$

where $\chi^2(P,Q)$ is the well-known χ^2-distance (see e.g [51]).

E.4 Discussion

The D_{PQ} metric can also be interpreted as the square root of an entropy approximation to the logarithm of an evidence ratio when testing if two (equally long) samples have been drawn from the same underlying distribution [56]. In that paper, it is also argued that D^2_{PQ} should be named Jensen-Shannon divergence, or rather, a special instance of that divergence, which is defined as

$$\begin{aligned} D_\lambda(P,Q) &= \lambda D\,(P\|\,R) + (1-\lambda)D\,(Q\|\,R) \\ R &= \lambda P + (1-\lambda)Q \end{aligned}$$

and therefore $D^2_{PQ} = D_{\frac{1}{2}}(P,Q)$.

Topsøe [86] has interpreted capacitory discrimination as twice an information transmission rate and related it to a variety of other distance measures, such as the Kullback divergence, triangular discrimination, variational distance and Hellinger distance. Many of the inequalities found by him can now be rewritten to become relationships between metrics.

Österreicher, in [71], proved the triangle inequality for square roots of f_β divergences defined by the functions

$$f_\beta(t) = \frac{(1+t^\beta)^{\frac{1}{\beta}} - 2^{\frac{1-\beta}{\beta}}(1+t)}{1-\frac{1}{\beta}} \tag{E.11}$$

for $\beta > 1$. Since the f_β divergence one obtains by taking the limit $\beta \to 1$ is $2D^2_{PQ}$, the results presented here extend the theorem proven in [71] to include the case $\beta = 1$.

Another way of looking at D^2_{PQ} is from the viewpoint of Bayesian inference. Consider the following scenario: Draw a sample $\tilde{X}_1 = \{x_1\}$ of length 1 from an unknown distribution R. What is assumed to be known about the distribution is that it is either P or Q, hence assigning each distribution the prior probability $\frac{1}{2}$. Using Bayesian inference, the posterior probabilities $P(P|\tilde{X}_1)$,$P(Q|\tilde{X}_1)$ of each distribution given the observation $\tilde{X}_1$ can then be calculated:

$$\begin{aligned} P(P|\tilde{X}_1) &= \frac{\frac{1}{2}P(x_1)}{\frac{1}{2}P(x_1)+\frac{1}{2}Q(x_1)} \\ P(Q|\tilde{X}_1) &= \frac{\frac{1}{2}Q(x_1)}{\frac{1}{2}P(x_1)+\frac{1}{2}Q(x_1)} \end{aligned} \tag{E.12}$$

The information gain $\Delta I(\tilde{X}_1)$ resulting from the observation of $\tilde{X}_1$ is given by the Kullback divergence between the posterior and the prior

$$\Delta I(\tilde{X}_1) = \frac{P(x_1)\log\frac{2P(x_1)}{P(x_1)+Q(x_1)} + Q(x_1)\log\frac{2Q(x_1)}{P(x_1)+Q(x_1)}}{P(x_1)+Q(x_1)} \tag{E.13}$$

To find the expected value of this gain, average $\Delta I(x_1)$ over the prior distribution of x_1, which is given by $\frac{1}{2}P + \frac{1}{2}Q$. This yields, noting that $P(x_1 = \omega_i) = p_i$ and likewise for Q:

$$\begin{aligned}
\mathcal{E}(\Delta I(\tilde{X}_1)) &= \frac{1}{2}\sum_{i=1}^{N} p_i \log \frac{2p_i}{p_i + q_i} \\
&\quad + \frac{1}{2}\sum_{i=1}^{N} q_i \log \frac{2q_i}{p_i + q_i} \\
&= D^2_{PQ} \qquad \text{(E.14)}
\end{aligned}$$

Therefore, another interpretation of D^2_{PQ} is that it is the expected information gain when deciding (by means of a sample of length 1) between two distributions given a uniform prior over the distributions. Consider now the case that P and Q are such that D_{PQ} is maximized. Then, as stated above, $D^2_{PQ} = 1$ (when using $\log_2$), i.e. the information gain is one bit. Thus, a sample of length 1 is sufficient to make the (binary) decision as to which distribution is the correct one. More general formulas than (E.14) can be found in [72], where relations between arbitrary f-divergences and information gains in decision problems are studied.

One interesting generalization of E.14 is that to more than 1 datapoint. Let $\tilde{X}_K$ be an i.i.d. sample of N points. Then

$$\begin{aligned}
P(\tilde{X}_K|P) &= \prod_{j=1}^{K} P(x_j) && \text{(E.15)} \\
P(\tilde{X}_K|Q) &= \prod_{j=1}^{K} Q(x_j) && \text{(E.16)} \\
P(\tilde{X}_K) &= \frac{1}{2}\prod_{i=1}^{K} P(x_i) + \frac{1}{2}\prod_{i=1}^{K} Q(x_i) && \text{(E.17)} \\
P(P|\tilde{X}_K) &= \frac{\prod_{j=1}^{K} P(x_j)}{\prod_{i=1}^{N} P(x_j)) + \prod_{j=1}^{K} Q(x_j)} && \text{(E.18)} \\
P(Q|\tilde{X}_K) &= \frac{\prod_{j=1}^{K} Q(x_j)}{\prod_{i=1}^{N} P(x_j)) + \prod_{i=1}^{K} Q(x_j)} && \text{(E.19)} \\
P(P) &= P(Q) = \frac{1}{2} && \text{(E.20)}
\end{aligned}$$

As before, all those probabilities are conditioned on the prior knowledge. Denoting the K-fold sum over all x_js by

$$\sum_K = \underbrace{\sum_{i=1}^{N} \ldots \sum_{i=1}^{N}}_{\text{K sums}} \tag{E.21}$$

the difference $D_{PQ}(K)$ between the expected information gains from samples of length $K+1$ and K is

$$\begin{aligned}
D^2_{PQ}(K) &:= \mathcal{E}(\Delta I(\tilde{X}_{K+1})) - \mathcal{E}(\Delta I(\tilde{X}_K)) \\
&= \frac{1}{2}\sum_{K+1} P(\tilde{X}_{K+1}|P)\log(2P(P|\tilde{X}_{K+1})) + \frac{1}{2}\sum_{K+1} P(\tilde{X}_{K+1}|Q)\log(2P(Q|\tilde{X}_{K+1})) \\
&- \frac{1}{2}\sum_{K} P(\tilde{X}_{K}|P)\log(2P(P|\tilde{X}_{K})) - \frac{1}{2}\sum_{K} P(\tilde{X}_{K}|Q)\log(2P(Q|\tilde{X}_{K})) \\
&= \frac{1}{2}\sum_{K+1} P(\tilde{X}_{K+1}|P)\log\left(\frac{P(P|\tilde{X}_{K+1})}{P(P|\tilde{X}_K)}\right) \\
&+ \frac{1}{2}\sum_{K+1} P(\tilde{X}_{K+1}|Q)\log\left(\frac{P(Q|\tilde{X}_{K+1})}{P(Q|\tilde{X}_K)}\right)
\end{aligned} \tag{E.22}$$

In deriving the last line, the equality

$$\begin{aligned}
\sum_K P(\tilde{X}_K|P)\log(P(P|\tilde{X}_K)) &= \sum_K \prod_{i=1}^{K} P(x_i)\log(P(P|\tilde{X}_K)) \\
&= \sum_{K+1} \prod_{i=1}^{K+1} P(x_i)\log(P(P|\tilde{X}_K)) \\
&= \sum_{K+1} P(\tilde{X}_{K+1}|P)\log(P(P|\tilde{X}_K))
\end{aligned} \tag{E.23}$$

was used, which holds because $P(x_i)$ is normalized and the argument of the logarithm does not depend on x_{K+1}. It will now be shown that this difference is monotonically decreasing with K:

$$\begin{aligned}
D^2_{PQ}(K+1) - D^2_{PQ}(K) &= \frac{1}{2}\sum_{K+2} P(\tilde{X}_{K+2}|P)\log\left(\frac{P(P|\tilde{X}_{K+2})P(P|\tilde{X}_K)}{P(P|\tilde{X}_{K+1})^2}\right) \\
&+ \frac{1}{2}\sum_{K+2} P(\tilde{X}_{K+2}|Q)\log\left(\frac{P(Q|\tilde{X}_{K+2})P(Q|\tilde{X}_K)}{P(Q|\tilde{X}_{K+1})^2}\right)
\end{aligned} \tag{E.24}$$

where E.23 was employed again to simplify the expression. Noting (E.15)-(E.20), one finds

$$\frac{P(P|\tilde{X}_{K+2})P(P|\tilde{X}_K)}{P(P|\tilde{X}_{K+1})^2} = \frac{P(x_{K+2})}{P(x_{K+1})}\frac{P(\tilde{X}_{K+1})^2}{P(\tilde{X}_{K+2})P(\tilde{X}_K)} \tag{E.25}$$

and likewise for the second term of (E.24). Since

$$\sum_{K+2} P(\tilde{X}_{K+2}|P)\log(P(x_{K+2})) - \sum_{K+2} P(\tilde{X}_{K+2}|P)\log(P(x_{K+1})) = 0 \tag{E.26}$$

(E.24) becomes, using (E.17)

$$D^2_{PQ}(K+1) - D^2_{PQ}(K) = \sum_{K+2} P(\tilde{X}_{K+2})\log\left(\frac{P(\tilde{X}_{K+1})^2}{P(\tilde{X}_{K+2})P(\tilde{X}_K)}\right) \tag{E.27}$$

For the final step, the equality

$$\begin{aligned}\sum_{K+2} P(\tilde{X}_{K+2})\log\left(\frac{P(\tilde{X}_{K+1})}{P(\tilde{X}_K)}\right) &= \sum_{K+2} P(\tilde{X}_{K+2})\log\Big(P(x_{K+1}|\tilde{X}_K)\Big) \\ &= \sum_{K+2} P(\tilde{X}_{K+2})\log\Big(P(x_{K+2}|\tilde{X}_{2,K+1})\Big)\end{aligned} \tag{E.28}$$

will be used, where $\tilde{X}_{2,K+1}$ is the sample comprised of the datapoints $2,\ldots,K+2$. This equality holds basically due to the interchangeability of the datapoints in the sample. Note that

$$\tilde{P}(\tilde{X}_{K+2}) = P(x_{K+2}|\tilde{X}_{2,K+1})P(\tilde{X}_{K+1}) \tag{E.29}$$

is a probability distribution over $\tilde{X}_{K+2}$: it is ≥ 0 everywhere and normalized. Thus

$$\begin{aligned}D^2_{PQ}(K+1) - D^2_{PQ}(K) &= \sum_{K+2} P(\tilde{X}_{K+2})\log\left(\frac{\tilde{P}(\tilde{X}_{K+2})}{P(\tilde{X}_{K+2})}\right) \\ &= -D\Big(P(\tilde{X}_{K+2})|\tilde{P}(\tilde{X}_{K+2})\Big) \leq 0\end{aligned} \tag{E.30}$$

Thus, the expected increase of the information gain decreases as the sample grows. In other words, each newly observed point is expected to add less to the already gained knowledge than the point before it. Thus $D^2_{PQ}(0) = D^2_{PQ}$ is an upper bound on the expected information gain per sample point. Due to the uniform prior over P and Q, exactly 1 bit of information has to be extracted from the sample before it is known which distribution is the one that generated the data. When D^2_{PQ} is measured in bit as well, then

$$N_{min} = \frac{1}{D^2_{PQ}} \tag{E.31}$$

is a lower bound on the sample size that is required, before the decision between P and Q can be made with certainty. Strictly speaking, the bound can only be reached when $D^2_{PQ} = 1$ bit, i.e. the distributions are distinct. Otherwise, a sample of inifinite size is necessary before certainty is attained. It could, however, be expected, that a sample size of $\mathcal{O}(N_{min})$ should be enough to reach a reasonable degree of certainty.

Bibliography

[1] S.M. Ali and S.D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society, Series B*, 28:131–142, 1966.

[2] E. Arabzadeh, S. Panzeri, and M.E. Diamond. Whisker vibration information carried by rat barrel cortex neurons. *Journal of Neuroscience*, 24(26):6011–6020, 2004.

[3] R. Baddeley, L.F. Abbott, M.C.A. Booth, F. Sengpiel, T. Freeman, E.A. Wakeman, and E.T. Rolls. Responses of neurons in primary and inferior temporal visual cortices to natural scenes. *Proceedings of the Royal Society of London, series B - biological sciences*, 264(1389):1775–1783, 1997.

[4] H. D. Barlow. Single units and perception. *Perception*, 1:371–394, 1972.

[5] T. Bayes. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 1763.

[6] J.O. Berger. *Statistical Descision theory and Bayesian analysis.* Springer, New York, 1985.

[7] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[8] V. Blanz, B. Schölkopf, H. Bülthoff, Burges C., Vapnik V., and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3d models. In C. von der Malsburg, W. von der Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial neural networks – ICANN96*, pages 251–256. Springer-Verlag, Berlin, 1996.

[9] I. N. Bronstein and K. A. Semendyayev. *Handbook of mathematics.* Harri Deutsch, Frankfurt/Main, 24th edition, 1989.

[10] R.F. Brown. *A topological introduction to nonlinear analysis.* Birkhäuser, Kassel, 1993.

[11] B. G. Burton. Problems and solutions in early visual processing. In R. Baddeley, P. Hancock, and P. Földiák, editors, *Information theory and the brain*, chapter 2. Cambridge University Press, New York, 2000.

[12] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[13] T. M. Cover and T. A. Joy. *Elements of Information Theory.* John Wiley & Sons, New York, 1991.

[14] R.T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

[15] G. Cybenko. Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:304–314, 1989.

[16] J. G. Daugmann. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988.

[17] P.J. Davis. Gamma function and related functions. In M. Abramowitz and I.A. Stegun, editors, *Handbook of mathematical functions.* Dover, New York, 1972.

[18] P. Dayan, Hinton G., Neal R., and Zemel R. The helmholtz machine. *Neural Computation*, 7:889–904, 1995.

[19] A.P. Dempster, N.M. Laird, and D.B Rubin. Maximum likelihood for incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[20] D. Endres and P. Földiák. Quadratic programming for learning sparse codes. In *Proceedings of the ninth international conference on artificial neural net-*

works (ICANN99), IEE Conference Publication No. 470, pages 593–596, London, 1999. Institution of Electrical Engineers.

[21] D. Endres and P. Földiák. Baysian bin distribution inference and mutual information. *IEEE Transactions on Information Theory*, 51(11), 2005.

[22] D. Endres and J.E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, 2002.

[23] T. S. Ferguson. Prior distributions on spaces of probability measures. *Annals of Statistics*, 2(4):615–629, 1974.

[24] B. L. Finlay and S. L. Pallas. Control of cell number in the developing mammalian visual system. *Progress in Neurobiology*, 32:207–234, 1989.

[25] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, New York, 2nd edition, 1987.

[26] P. Földiák. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

[27] P. Földiák. The 'ideal homunculus': statistical inference from neural population responses. In F. Eeckman and J. Bower, editors, *Computation and Neural Systems*, pages 55–60. Kluwer Academic Publishers, Norwell, MA, 1993.

[28] P. Földiák. Representation in neurons. In L. Nadel, editor, *Encyclopedia of Cognitive Science*. Macmillan, Nature Publishing Group, London, 2002.

[29] P. Földiák. Sparse coding in the primate cortex. In M. A. Arbib, editor, *The Handbook of brain theory and neural networks*. MIT Press, Cambridge, MA, 2nd edition, 2002.

[30] P. Földiák and M. Young. Sparse coding in the primate cortex. In M. A. Arbib, editor, *The Handbook of brain theory and neural networks*. MIT Press, Cambridge, MA, 1995.

[31] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, 1995.

[32] E.D. Gershon, M.C. Wiener, P.E. Latham, and Richmond B.J. Coding strategies in monkey v1 and inferior temporal cortices. *Journal of Neurophysiology*, 79:1135–1144, 1998.

[33] G. F. Harpur and R. W. Prager. Techniques in low entropy coding with neural networks. Technical Report CUED/F-INFENG/TR 197, Cambridge University Engineering Department, 1995.

[34] G. F. Harpur and R. W. Prager. Development of low entropy coding in a recurrent network. *Network: Computation in Neural Systems*, 7(2):277–284, 1996.

[35] G. F. Harpur and R. W. Prager. Experiments with low-entropy neural networks. In R. Baddeley, P. Hancock, and P. Földiák, editors, *Information theory and the brain*, chapter 5, pages 84–100. Cambridge University Press, New York, 2000.

[36] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[37] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81:3088–3092, 1984.

[38] C. Hsu, C. Chang, and C. Lin. *A practical guide to support vector classification.* http://www.csie.ntu.edu.tw/~cjlin/libsvm/, 2005.

[39] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 73:218–226, 1962.

[40] M. Hutter. Distribution of mutual information. In *Advances in Neural Information Processing Systems 14*, pages 339–406, Cambridge, MA, 2002. MIT Press.

[41] A. Hyvarinen and P. O. Hoyer. Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720, 2000.

[42] T.S. Jaakkola and M.I. Jordan. Computing upper and lower bounds on likelihoods in intractable networks. Technical Report AIM-1571, 1996.

[43] E. T. Jaynes. *Probability theory: the logic of science.* Cambridge University Press, New York, 2003.

[44] J.P. Jones and Palmer L. A. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1233–1258, 1987.

[45] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[46] P. Kafka, F. Österreicher, and I. Vince. On powers of f-divergences defining a distance. *Studia Scientia Mathematica Hungariae*, 26:415–422, 1991.

[47] C. Keysers. *The Speed of Sight.* PhD thesis, School of Psychology, University of St. Andrews, U.K., 2000.

[48] C. Keysers, D. Xiao, P. Földiák, and D. I. Perrett. The speed of sight. *Journal of Cognitive Neuroscience*, 13(1):90–101, 2001.

[49] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[50] M. S. Lewicki and B. A. Olshausen. Probabilistic framework for the adaptation and comparison of image codes. *Journal of the Optical Society of America*, 16(7):1587–1601, 1999.

[51] F. Liese and I. Vajda. *Convex Statistical Distances.* B.G. Teubner Verlagsgesellschaft, Leipzig, 1987.

[52] T. J. Loredo. From laplace to supernova sn1987a: Bayesian inference in astrophysics. In P. F. Fougére, editor, *Maximum Entropy and Bayesian Methods*, pages 81–142. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.

[53] D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, Pasadena, CA, 1992.

[54] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, New York, 2003.

[55] M. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[56] T. P. Minka. *Bayesian inference, entropy, and the multinomial distribution*. http:// www.stat.cmu.edu/˜minka/papers/multinomial.html, 2001.

[57] R. M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

[58] R. M. Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification. Technical Report 9702, Dept. of Computer Science, University of Toronto, 1997.

[59] R. M. Neal. Slice sampling. Technical Report 2005, Dept. of Computer Science, University of Toronto, 2000.

[60] R.M. Neal. Markov chain sampling methods for dirichlet process mixture models. Technical Report 9815, Dept. of Statistics, University of Toronto, 1998.

[61] I. Nemenman, W. Bialek, and R.R. van Steveninck. Entropy and information in neural spike trains: Progress on the sampling problem. *Physical Review E*, 69(5), 2004.

[62] B. A. Olshausen. Principles of image representation in visual cortex. In L. M. Chalupa and J. S. Werner, editors, *The Visual Neurosciences*, pages 1603–1615. MIT Press, Boston MA, 2003.

[63] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[64] B. A. Olshausen and D. J. Field. Natural image statistics and efficient coding. *Network: Computation in Neural Systems*, 7(2):333–339, 1996.

[65] B. A. Olshausen and D. J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 7(2):333–339, 1996.

[66] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[67] L.M. Optican, T.J. Gawne, B.J. Richmond, and P.J. Joseph. Unbiased measures of transmitted information and channel capacity from multivariate neuronal data. *Biological Cybernetics*, 65:305–310, 1991.

[68] M. W. Oram, P. Földiák, D. I. Perrett, and F. Sengpiel. The 'ideal homunculus': Decoding neural population signals. *Trends in Neuroscience*, 21:259–265, 1998.

[69] M. W. Oram and D. I. Perrett. Time course of neural responses discriminating different views of the face and head. *Journal of Neurophysiology*, 68(1):70–84, 1992.

[70] M. W. Oram, D. Xiao, B. Dritschel, and K.R. Payne. The temporal resolution of neural codes: does response latency have a unique role? *Philosophical Transactions of the Royal Society, Series B*, 357:987–1001, 2002.

[71] F. Österreicher. On a class of perimeter-type distances of probability distributions. *Kybernetika*, 32:389–393, 1996.

[72] F. Österreicher and I. Vajda. On a class of perimeter-type distances of probability distributions. *IEEE Transactions on Information Theory*, 36:1036–1039, 1993.

[73] L. Paninski. Estimating entropy on m bins given fewer than m samples. *IEEE Transactions on Information Theory*, 50(9):2200–2203, 2004.

[74] S. Panzeri and A. Treves. Analytical estimates of limited sampling biases in different information measures. *Network: Computation in Neural Systems*, 7:87–107, 1996.

[75] A. E. C. Pece. Redundancy reduction of a gabor representation: A possible computational role for feedback from primary visual cortex to lateral geniculate nucleus. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks 2*, pages 865–868. Elsevier, Amsterdam, 1992.

[76] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, New York, 1986.

[77] D. L. Ringach. Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of Neurophysiology*, 88:455–463, 2002.

[78] J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[79] E.T. Rolls, H.D. Critchley, and A. Treves. The representation of olfactory information in the primate orbitofrontal cortex. *Journal of Neurophysiology*, 75:1982–1996, 1995.

[80] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–837, 1956.

[81] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart, J.L. McClelland, and the PDP research group, editors, *Parallel distributed processing: explorations in the microstructure of cognition*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[82] C. E. Shannon. The mathematical theory of communication. *The Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[83] C.E Shannon and W. Weaver. *The Mathematical Theory of Communication.* University Press, Urbana, IL, 1949.

[84] J. E. Shore and R. W. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on Information Theory*, 26(1):26–37, 1980.

[85] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B*, 36(1):111–147, 1974.

[86] F. Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory*, 46:1602–1609, 2000.

[87] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[88] V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, New York, 1995.

[89] V. Vapnik. *Statistical learning theory.* John Wiley and Sons, New York, 1998.

[90] B.T. Vincent, R.J. Baddeley, T. Troscianko, and I.D. Gilchrist. Is the early visual system optimised to be energy efficient? *Network: Computation in Neural Systems*, 16(2/3):175–190, 2005.

[91] C.S. Wallace and D.M. Boulton. An information measure for classification. *Computer Journal*, 11:185–194, 1968.

[92] C. K. I. Williams and D. Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.