Self-Organized Neural Learning of Statistical Inference from High-Dimensional Data

Johannes Bauer, Stefan Wermter
Department of Informatics
University of Hamburg
Germany
Email: {bauer,wermter}@informatik.uni-hamburg.de

Abstract

With information about the world implicitly embedded in complex, high-dimensional neural population responses, the brain must perform some sort of statistical inference on a large scale to form hypotheses about the state of the environment. This ability is, in part, acquired after birth and often with very little feedback to guide learning. This is a very difficult learning problem considering the little information about the meaning of neural responses available at birth. In this paper, we address the question of how the brain might solve this problem: We present an unsupervised artificial neural network algorithm which takes from the self-organizing map (SOM) algorithm the ability to learn a latent variable model from its input. We extend the SOM algorithm so it learns about the distribution of noise in the input and computes probability density functions over the latent variables. The algorithm represents these probability density functions using population codes. This is done with very few assumptions about the distribution of noise. Our simulations indicate that our algorithm can learn to perform similar to a maximum likelihood estimator with the added benefit of requiring no a-priori knowledge about the input and computing not only best hypotheses, but also probabilities for alternatives.

1 Introduction

All information the brain receives about the outside world is coded in the same kind of neural responses to sensory stimuli. In order to make sense of this information the brain needs knowledge about the statistics of these responses. It needs information about the relationship between specific responses and the state of the world. Some of this knowledge is present at birth, but a vast amount is acquired during a lifetime of learning. How this knowledge is learned, represented, and used for information processing is still an open research question.

Specific decoding schemes for population responses implicitly answer the question of the representation of information at least for some cases: Often, a single quantity is encoded in the activity of an ensemble of neurons in a population code. Decoding schemes for population codes have been proposed and evaluated using neurophysiological data and computer simulations [Georgopoulos *et al.*, 1986; Seung and Sompolinsky, 1993; Deneve *et al.*, 2001]. A special case of population codes encode probability density functions (PDFs) [Pouget *et al.*, 2003]. Recent approaches have addressed the question of how the brain might process information in this special case [Cuijpers and Erlhagen, 2008; Beck *et al.*, 2008]. These studies were inspired by biological findings showing that humans and other animals perform Bayes-optimally in sensory processing and multi-sensory integration tasks [Ernst and Banks, 2002; Landy *et al.*, 2011; Gori *et al.*, 2012]. The focus of these studies is on integrating information from multiple population codes, paying special attention to sensory and neural noise. They assume that the input already codes for a PDF.

Consider, however, the problem of locating the center of a blob of light. The population response of the retina will code for the distribution of light, which is not generally equivalent to a PDF for the position of the stimulus (see Fig. 1). There has been less work on how a PDF can be computed from neural input initially, and how computing such a PDF can be learned under plausible constraints. The architecture due to Deneve *et al.* does compute an estimate from primary input, but it does not compute a PDF, and its network dynamics are hard-coded [2001]. Barber *et al.* describe a framework for creating networks which compute PDFs [2003]. Similarly, Jazayeri and Movshon propose an artificial neural network (ANN) model which recovers a PDF from noisy input [2006]. However, in both cases, there is no learning involved and knowledge of the response and noise properties of the input is required. In this paper, we propose an algorithm which lets a neural population learn to compute a PDF for a stimulus from sensory input. The network then encodes the PDF in a population code. The algorithm is self-organized and unsupervised. It does not require implicit knowledge of the nature of the stimulus, of how the stimulus is represented by the input, or of the distribution of noise. These features make ours a very versatile machine learning algorithm and a new model of natural learning and information processing.

The structure of this paper is as follows: In Section 2 we first analyze the problem of interpreting neural input from a statistical perspective. We briefly explain population coding

Figure 1: Difference between Light Distribution and PDF. **left:** stimulus, **right:** PDF for position of center.

and self-organization. In Section 3, we describe an algorithm based on the self-organizing map (SOM) [Kohonen, 1982] which combines statistical learning, population coding, and self-organization to learn how to compute and represent PDFs from neural input. In Section 4 we report on simulations carried out to validate our approach. Finally, we discuss the properties of our algorithm as a machine learning approach and as a model of neural learning. We also point out other ANN models which may help implement some of the mechanisms of our algorithm in a biologically plausible manner.

2 Theoretical Considerations

One approach to understanding biological systems is analyzing their presumed function, and what a good solution could be. Comparing the result with what one finds in nature can then give insight into how the system works, or what the task really is, or what the constraints are [Jacobs and Kruschke, 2011; Landy *et al.*, 2011]. In this section, we will present the problem faced by a single neuron interpreting sensory information and an abstract solution to that problem. We will extend the problem and the solution to the case where a population of neurons collaborate, and discuss how a population can self-organize without external feedback.

Generally, the goal of processing sensory information is learning about properties of the world. Such properties can be the position of a stimulus, or its size, or its identity. A neuron taking part in such a computation uses information from thousands of synapses connecting it to that many neurons in different parts of the brain. A fundamental problem in this is that initially a neuron has no way of knowing what the relationship is between its input activity and anything outside the skull. Some connection properties of course are hard-wired and present at birth. However, perception sharpens drastically with experience after birth [King, 2004; Gori *et al.*, 2008; Stein, 2012]. Thus, neurons performing perceptual processing have to learn this relationship from experience and over time.

2.1 The Statistical Approach

Given enough data about past activity of input neurons in different situations, the problem described above becomes a matter of simple statistics. Consider an output neuron $\mathbf{o}$ receiving as input activity $\mathbf{a}$ from some input neuron $\mathbf{i}$. Let us say that it is $\mathbf{o}$'s task to calculate whether or not some proposition $\mathfrak{A}$ is true. The statistical solution to this is to compare how often $\mathbf{i}$'s activity was $\mathbf{a}$ when $\mathfrak{A}$ was true in the past to when $\mathfrak{A}$ was false. To make this more precise, assume for a moment we have a data set $\mathbb{D}$ of the previous activity of $\mathbf{i}$ in cases when $\mathfrak{A}$ was true and in cases when $\mathfrak{A}$ was false. If $\mathbb{D}_{\mathbf{a}}$ and $\mathbb{D}_{\mathfrak{A}}$ are the subsets of only those data points in $\mathbb{D}$ where the activity was $\mathbf{a}$ and where $\mathfrak{A}$ was true, respectively, and $\mathbb{D}_{\mathbf{a},\mathfrak{A}} = \mathbb{D}_{\mathbf{a}} \cap \mathbb{D}_{\mathfrak{A}}$, then we can estimate the following probabilities:

$$P(\mathbf{a} \mid \mathfrak{A}) \approx \frac{|\mathbb{D}_{\mathbf{a},\mathfrak{A}}|}{|\mathbb{D}_{\mathfrak{A}}|}, \quad P(\mathbf{a}) \approx \frac{|\mathbb{D}_{\mathbf{a}}|}{|\mathbb{D}|}, \quad P(\mathfrak{A}) \approx \frac{|\mathbb{D}_{\mathfrak{A}}|}{|\mathbb{D}|}. \quad (1)$$

The probability of $\mathfrak{A}$ given $\mathbf{i}$'s activity $\mathbf{a}$ is (by Bayes' theorem):

$$P(\mathfrak{A} \mid \mathbf{a}) = \frac{P(\mathbf{a} \mid \mathfrak{A})}{P(\mathbf{a})} P(\mathfrak{A}). \quad (2)$$

Inserting the probability estimates from Eq. 1, we get

$$P(\mathfrak{A} \mid \mathbf{a}) \approx \frac{|\mathbb{D}_{\mathbf{a},\mathfrak{A}}|}{|\mathbb{D}_{\mathfrak{A}}|} \frac{|\mathbb{D}|}{|\mathbb{D}_{\mathbf{a}}|} \frac{|\mathbb{D}_{\mathfrak{A}}|}{|\mathbb{D}|} = \frac{|\mathbb{D}_{\mathbf{a},\mathfrak{A}}|}{|\mathbb{D}_{\mathbf{a}}|}. \quad (3)$$

Now consider the case in which our neuron receives input from a population of input neurons $\mathbf{i}_1, \mathbf{i}_2, \dots, \mathbf{i}_l$ whose joint activity is $\vec{\mathbf{a}} = \mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_l$. Assuming noise in these input neurons is uncorrelated, it is easy to extend our considerations above for this case. For one, Eq. 2 becomes

$$P(\mathfrak{A} \mid \vec{\mathbf{a}}) = \frac{P(\mathfrak{A})}{P(\vec{\mathbf{a}})} \prod_{k=1}^{l} P(\mathbf{a}_k \mid \mathfrak{A}). \quad (4)$$

We can estimate the probability $P(\mathfrak{A} \mid \vec{\mathbf{a}})$ as we did in Eq. 3:

$$P(\mathfrak{A} \mid \vec{\mathbf{a}}) \approx \frac{|\mathbb{D}_{\mathfrak{A}}|}{|\mathbb{D}|} \frac{|\mathbb{D}|}{|\mathbb{D}_{\vec{\mathbf{a}}}|} \prod_{k=1}^{l} \frac{|\mathbb{D}_{\mathbf{a}_k,\mathfrak{A}}|}{|\mathbb{D}_{\mathfrak{A}}|}, \quad (5)$$

where $\mathbb{D}_{\mathbf{a}_k}$ and $\mathbb{D}_{\mathbf{a}_k,\mathfrak{A}}$ are defined analogous to $\mathbb{D}_{\mathbf{a}}$ and $\mathbb{D}_{\mathbf{a},\mathfrak{A}}$ above and $\mathbb{D}_{\vec{\mathbf{a}}}$ is the intersection of all $\mathbb{D}_{\mathbf{a}_k}, 1 \leq k \leq l$. Note that, for large l, it can take a very big data set to make $\mathbb{D}_{\vec{\mathbf{a}}_k}$ representative. Fortunately, this will not be a problem in our case as we will explain below.

2.2 Population Coding and PDF

Often, hypotheses about properties of the world are not encoded in the activities of single neurons. Instead, population codes seem to be involved in many areas of the brain and have become an important modeling paradigm [Seung and Sompolinsky, 1993; Pouget *et al.*, 2003; Beck *et al.*, 2008]. In a population code, each neuron's response $\mathbf{a}_{\mathfrak{S}}$ to a stimulus $\mathfrak{S}$ is a stochastic function of $\mathfrak{S}$:

$$\mathbf{a}_{\mathfrak{S}} = f(\mathfrak{S}) + \nu(\mathfrak{S}),$$

where $\nu(\mathfrak{S})$ models random noise whose distribution may or may not be dependent on $\mathfrak{S}$. Each neuron $\mathbf{n}_j$ in the population has a different tuning function f_j, and the noise in each neuron's response is generally assumed to be independent. Usually, the neurons in a population $\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_l$ are thought to be spatially ordered. The tuning function $f_k, 1 \leq k \leq l$ of neuron $\mathbf{n}_k$ then often differs from that of other neurons only in some parameter θ_k which depends on the position of the neuron. In this paper, we assume that the input to our network is a population code for some perceived property of the world.

In a second use of the concept, we want to population-code a PDF for the property represented by the input: In a population code encoding a PDF [Barber *et al.*, 2003] for a value

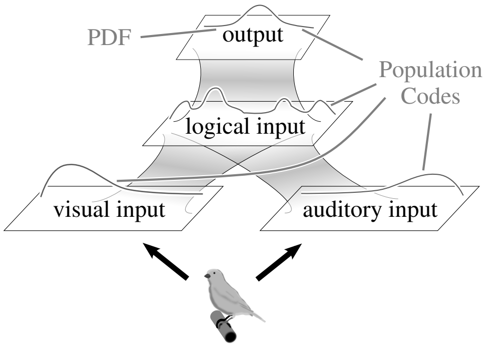


Figure 2: Population-coded PDF Computed from Multiple Physical Input Populations

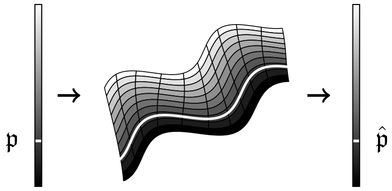


Figure 3: Value to Curve to Value:
The value $\mathfrak{p}$ of a world property $\mathfrak{P}$ is mapped to a curve (out of a set) from which $\mathfrak{p}$ can be estimated.

$\mathfrak{p}$ of a property $\mathfrak{P}$ of the outside world, each neuron $\mathbf{n}_k$ has a preferred value $\mathfrak{p}_k$ and codes with its activity $\mathbf{a}_\mathfrak{S}$ for the probability of $\mathfrak{p}_k$ being the true value of $\mathfrak{P}$ given stimulus $\mathfrak{S}$:

$$\mathbf{a}_\mathfrak{S} = g(P(\mathfrak{P} = \mathfrak{p}_k \mid \mathfrak{S})) + \nu(\mathfrak{S}),$$

where g is some function which encodes probabilities in single neural responses. To stick with the terminology of the previous section, we can also say that each neuron $\mathbf{n}_k$ codes for the probability of the proposition $\mathfrak{A}_k$ that $\mathfrak{P} = \mathfrak{p}_k$:

$$\mathbf{a}_\mathfrak{S} = g(P(\mathfrak{A}_k \mid \mathfrak{S})) + \nu(\mathfrak{S}).$$

Assuming that neurons can somehow implement the statistical calculations of Section 2.1, a population of output neurons can compute and represent a PDF from their input: Each neuron has its own preferred value $\mathfrak{p}$, determines the probability $\mathfrak{P}$ given the input, and encodes that probability in its output. This idea becomes especially interesting if we take the population of input neurons as a logical population instead of a physical one: In a biological context, some of the neurons of the input population might code for the position of an object in visual space, and others for its position in auditory space (see Fig. 2).

2.3 Self-Organization

Let us point out the following two observations about what we have discussed so far. First, Section 2.1 assumed a data set of previous pre-synaptic activity values of a neuron and truth values of the proposition for whose probability that neuron is to code. Obviously, information about the input activity is immediately available to a neuron. However, in many cases of neural learning, it is hard to argue for a teaching signal which would give a neuron information about the true value of a world property.

Second, Section 2.2 assigned each neuron in a population code a preferred value of the property whose PDF was being encoded. This poses the question of what mechanism could organize a population of neurons in such a way that they each respond to different stimuli. The development of neural connectivity is guided prenatally by chemical gradients. Such gradients could configure neurons in a population to respond to different inputs. However, it is not immediately clear that such a configuration could be fine enough. Also, it has been found in nature that the preferred values will shift with postnatal experience [Wallace and Stein, 2007]. We therefore prefer a model which produces an ordered distribution of preferred values over output neurons without requiring either pre-configuration or a teaching signal.

Let us recall the task we are trying to solve: Implicitly, we are assuming that a low-dimensional latent variable $\mathfrak{P}$ is the cause of our high-dimensional observation, the input population's response. And it is our aim to learn to recover probabilities for different values of that latent variable. That means, we are trying to learn a latent-variable model. Assume, for example, that input neurons $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_l$ respond to the value $\mathfrak{p}$ of a world property $\mathfrak{P}$ and $\mathfrak{p} \in \mathbb{R}$. If these neurons encode the value of $\mathfrak{P}$ as a noisy sine wave, say, then it is our task to learn this relationship and to transfer a sine wave back to an estimate of $\mathfrak{p}$ (see Fig. 3). The unsupervised SOM algorithm has been shown to be able to learn low-dimensional latent-variable models for observations in high-dimensional spaces [Yin, 2007; Klanke, 2007]: a SOM learns a topology-preserving mapping from points in a data space of dimension m to its units (or 'neurons'), which are ordered in an n-dimensional grid, typically with $m > n$.

The SOM algorithm uses three mechanisms to accomplish learning of latent-variable models: through a form of Hebbian learning, it gradually moves the mapping of its units from their starting position closer and closer and finally into the subspace of data points. Through competitive learning, the algorithm distributes the units within that subspace. And through neighborhood interaction, ie. by always adapting not only one neuron, but also its neighbors in the grid, it preserves the topological order of that subspace in the mapping.

In the following, we will explain how the statistical considerations from Section 2.1 can be combined with self-organized learning of latent-variable models to produce population codes for PDFs.

3 Implementation

This section will deal with the implementation of the ideas laid out so far. We will describe verbally the process of learning how to population-code PDFs and how the mechanism relates to the ideas above. For a summary, consult Algorithm 1.

Data Structures. In Section 2.1 we explained our ideas in terms of data sets. However, since only the sizes of these sets are necessary for probability computations, we implement our algorithm using counters. Let $\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_m$ and

Algorithm 1 The Full Algorithm

```
function LEARN($\mathbb{D}$ : the data set)
    for $n = 0 \to N$ do
        select $d \leftarrow (d_1, d_2, \ldots, d_l) \in \mathbb{D}$
        $b \leftarrow argmax_k(\text{RESPONSE}(k, d))$
        for $j = 1 \to m$ do
            UPDATENEURON($j, b, d, \sigma_n, s_n$)
        end for
    end for
end function

function UPDATENEURON($k, b, d, \sigma, s$)
    $\eta \leftarrow h(\delta_{k,b}, \sigma)$
    for $i = 1 \to l$ do
        $on_{k,i}[d_i] \leftarrow on_{k,i}[d_i] + s * \eta$
    end for
    $c_{off\,k} \leftarrow c_{off\,k} + s$
    $c_{on\,k} \leftarrow c_{on\,k} + s * \eta$
end function

function RESPONSE($k, d$)
    $r = (c_{off\,k}/c_{on\,k})^p$
    for $i = 1 \to l$ do
        $r \leftarrow r \cdot \frac{on_{k,i}[d_i]}{sum(on_{k,i})}$
    end for
    return $r$
end function
```

Definitions:

N	Number of training steps.
l, m	Size of input, output population.
σ_n	Neighborhood interaction width (decreasing with n, eg. linear with softmin).
s_n	Update strength (*in*creasing with n, eg. $\sqrt{n}$).
$\delta_{k,j}$	Grid distance between output neurons $\mathbf{o}_k, \mathbf{o}_j$.
$h(\delta, \sigma)$	Gaussian $\exp(-\delta^2/\sqrt{\frac{1}{2}\pi\sigma^2})$.
$p \in \mathcal{O}(l)$	Spread exponent.
$c_{off\,j}, c_{on\,j}$	Update counters for output neuron $\mathbf{o}_j$.
$on_{j,k}$	list of activity counters for output $\mathbf{o}_j$, input neuron $\mathbf{i}_k$.

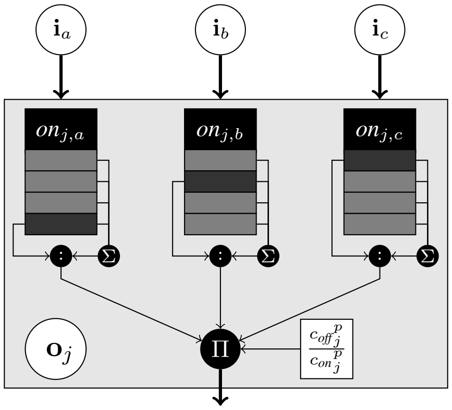


Figure 4: Output Neuron $\mathbf{o}_j$ Computes Its Activity from that of Input Neurons $\mathbf{i}_a, \mathbf{i}_b, \mathbf{i}_c$.

$\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_m$ be our population of input and output neurons, respectively. Then, each output neuron $\mathbf{o}_j$ maintains two counters, $c_{on\,j}$ and $c_{off\,j}$, and a list $on_{j,k}$ of counters per input neuron $\mathbf{i}_k$.

How these counters and lists are maintained will be explained below. As a guiding intuition, assume some neuron $\mathbf{o}_j$ has the preferred value $\mathfrak{p}_j$. This means that the neuron's job is to code for the probability of proposition $\mathfrak{A}_j = (\mathfrak{P} = \mathfrak{p}_j)$ being true. The reader may think of entry $on_{j,k}[\mathbf{a}]$ as a count of the number of occasions where the activity of input neuron $\mathbf{i}_k$ was $\mathbf{a}$ and $\mathfrak{A}_j$ was true. It will therefore serve instead of $\left|\mathbb{D}_{\mathbf{a}_j, \mathfrak{A}_j}\right|$ as used in Section 2.1.

The individual counters $c_{on\,j}$ and $c_{off\,j}$ are similar; it is their role to approximate $|\mathbb{D}_{\mathfrak{A}}|$ and $|\mathbb{D}|$, respectively. Thus we can implement Eq. 5 and given activities $\vec{\mathbf{a}} = \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_l$ of the input neurons $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_l$, the probability of $\mathfrak{P}$ being $\mathfrak{p}_j$ can be calculated approximately as

$$P(\mathfrak{P} = \mathfrak{p}_j | \vec{\mathbf{a}}) \approx \frac{1}{P(\vec{\mathbf{a}})} \frac{c_{on\,j}}{c_{off\,j}} \prod_{k=1}^{l} \frac{on_{j,k}[\mathbf{a_k}]}{\sum(on_{j,k})}. \tag{6}$$

where $\sum(on_{j,k})$ denotes the sum of all entries of list $on_{j,k}$. The evidence $P(\vec{\mathbf{a}})$ is independent of $\mathfrak{p}_j$ and thus the same for all neurons $\mathbf{o}_j$. It can therefore be replaced by a constant, becoming part of the PDF encoding.

The Algorithm. We will now explain how the counters and lists of counters of our output neurons introduced above can be updated such that after learning each output neuron $\mathbf{o}_n$ has its own preferred value $\mathfrak{p}_k$ of $\mathfrak{P}$ and that the population response will represent a PDF over the possible values of $\mathfrak{P}$.

In the SOM algorithm, competitive learning is implemented by updating SOM units depending on their grid distance from the so-called best-matching unit (BMU). The BMU is that unit which reacts most strongly to the data point. The further away from the BMU, the less strongly a unit is updated. On average, units are updated in such a way that they become more responsive to the data points with which they are updated and less responsive to data points which are very different from them. This leads to specialization of units and their distribution in data space. We, too, use this strategy for our learning algorithm.

Assume neuron $\mathbf{o}_B$ was most responsive to the data point represented by input activity $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_l$ of the input units $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_l$ at some time t. Then each neuron $\mathbf{o}_j, 1 \leq j \leq m$ is updated with a strength s_t and with a neighborhood interaction $\eta_t = h_t(B, j)$ where $h_t(B, j)$ is a zero-centered Gaussian function of the distance between $\mathbf{o}_B$ and $\mathbf{o}_j$. The width of the Gaussian neighborhood interaction function h_t decreases over time, whereas s_t increases sublinearly to allow for continued learning. Updating a neuron $\mathbf{o}_j$ with an input activity $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_l$, strength s_t and neighborhood interaction η means updating $c_{on\,j}$ and $c_{off\,j}$ as well as $on_{j,k}$, for every input neuron $\mathbf{i}_k$. $c_{\mathit{off}\,j}$ and $c_{\mathit{on}\,j}$ are updated by adding s_t and $s_t\eta$, respectively. $\mathit{on}_{j,k}$ is updated by updating the entry corresponding to the input neuron's activity $\mathbf{a}_k$: We add $s_t\eta$ to $\mathit{on}_{j,k}[\mathbf{a}_k]$. We refer to Algorithm 1 for details.

What is left is to explain how an output neuron computes its activity from the input and thus how the BMU is selected. Our aim is to population-code a PDF, therefore it would be natural to set a neuron $\mathbf{p}_j$'s response proportional to the computed probability density at its preferred value $\mathfrak{p}_j$. However, we have found that noise and the complex mechanisms for calculation of activation and neuron updates can lead to an anomalous situation: Suppose during the learning process one neuron is adapted so strongly that it is much closer to the data space than all others. Then, in the next learning steps, it may be so much closer to all data points that no other unit is ever again selected as BMU and updated. Usually, this can be prevented by using a very small learning step length.

There is no such thing as a learning step length in our algorithm. Therefore, we employ another mechanism which promotes competition between neurons even if only a small number of neurons are updated: By multiplying the activity of each neuron $\mathbf{o}_j$ by the spread factor $b_j = \left(\frac{c_{\mathit{off}\,j}}{c_{\mathit{on}\,j}}\right)^p$ for some large p, we artificially increase the neurons' responses if they have not been BMU (or close to a BMU) for a long time. Thus, whenever some neuron $\mathbf{o}_j$ gets too much better than the others at representing the input, it becomes BMU very often and b_j becomes smaller. Other neurons' responses are magnified and they are updated more often such that their chances at being BMU for some inputs are increased. An equilibrium is reached when b_j is equal for all neurons $\mathbf{o}_j$. This is the case if, over many data points, every neuron has been BMU approximately the same number of times. Assuming that the data set is representative, this happens if the possible values $\mathfrak{p}$ of $\mathfrak{P}$ are distributed over the neurons such that their accumulated probability is approximately equal across neurons.

4 Experiments

We conducted two sets of experiments to validate our algorithm. In the first set, we simulated one large, noisy input population. These experiments were designed to be simple and easy to benchmark against an artificial, optimal maximum likelihood estimator (MLE) read-out mechanism. In the second set of experiments, we simulated two input populations. These populations were smaller and somewhat less noisy. They were meant to stand for input from different sensory modalities. We therefore refer to our different sets of experiments as 'uni-sensory' and 'multi-sensory' experiments. We report numbers and show figures for one run of the simulations for each set. These runs were representative and other runs with different parameters went qualitatively similar.

4.1 Uni-sensory Experiments

In our uni-sensory experiments, we simulated a network consisting of an input population $I = (\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_{100})$ and an output population $O = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{110})$. The output population was arranged in a one-dimensional grid. Stimuli were random decimals $\mathfrak{p} \in [0, 1]$ and the tuning function $f_k()$ of each input neuron $\mathbf{i_k}$ with preferred value $\mathfrak{p}_k = k/100$ was a Gaussian function of $\mathfrak{p}$:

$$f_k(\mathfrak{p}) = a \exp\left(\frac{-(\mathfrak{p}_k - \mathfrak{p})^2}{\frac{1}{2}\sigma^2}\right).$$

with $a = 3, \sigma = 0.3$. The actual neural response of each input neuron $\mathbf{i}_k$ was then Poisson distributed around $f_k(\mathfrak{p})$.

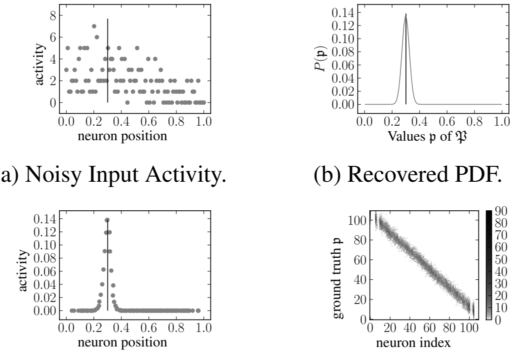
Without optimizing the parameters of the network and simulation for fast convergence, the network learned a rough mapping after a thousand training steps. This mapping sharpened and stabilized within the next few thousand steps (see Fig. 5d). After 100 000 learning steps, we first analyzed our network. The mapping from neurons to input values realized by the network cannot be predicted beforehand because of the process of self-organization. Therefore, we simulated inputs at each position $\mathfrak{p}_l = l/100\,000$ for $l \in [1\ ..\ 100\,000]$ and recorded the neurons with the strongest response. The preferred value $\mathfrak{p}_k$ of each output neuron $\mathbf{o}_k$ was then defined as the mean of all positions $\mathfrak{p}_l$ for which $\mathbf{o}_k$ was the BMU.

This mapping was then used to analyze the performance of the network. Again, we simulated inputs at regular intervals $\mathfrak{i}_{\mathfrak{l}} = l/12\,000$, this time for $l \in [1\ ..\ 12\,000]$. We then calculated the mean squared error (MSE) as the mean of the squared difference between the simulated input and the preferred value of the BMU determined in the previous step.

As a benchmark, we implemented recovering the PDF directly from the input activity (see Fig. 5b). This is possible knowing the distribution of inputs $\mathfrak{i}$, the transfer functions of the input neurons, and the noise model. The network does not have access to any of these. By recovering the PDF for test inputs and selecting the maximum, we obtained an MLE to which we could compare our network's performance. We found that the MSE of the network was $\mathit{mse}_n = 8.829 \times 10^{-4}$ and that of the MLE was $\mathit{mse}_{mle} = 7.425 \times 10^{-4}$. Since MLE is the optimal decoding scheme for this kind of input [Seung and Sompolinsky, 1993; Deneve *et al.*, 1999], this shows that our network is capable of learning how to near-optimally combine the information contained in a population response without knowledge of the specifics of the code. Figs. 5a, 5b, and 5c show an activity profile of the input population, the PDF recovered externally, and the response of the network. The activity of each neuron is plotted at the position we assigned when we analyzed the network organization. The population response was normalized such that the sum of activities was 1. Fig. 5d shows how input stimuli were mapped to neurons during testing: The figure indicates which value of $\mathfrak{P}$ was mapped to which neuron how often during testing.

4.2 Multi-sensory Experiments

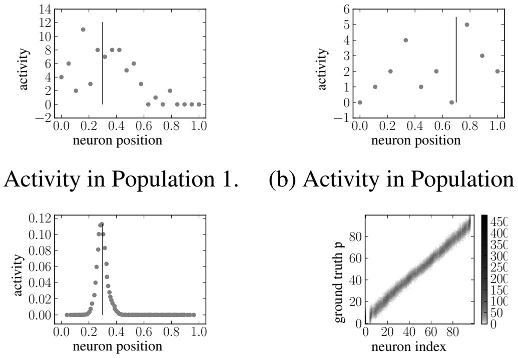
In a second set of experiments, we simulated two input populations representing different sensory modalities. Accordingly, neurons in these input populations had different tuning functions: Neurons in both populations had Gaussian tuning functions, but the width and amplitude were different. The amplitudes in population 1 and population 2 were $a_1 = 8$ and $a_2 = 4$, resp. The widths were $\sigma_1 = 0.2$ and $\sigma_2 = 0.3$. Also, the size of the input population differed. Population 1 contained 20, whereas population 2 contained only 10. Finally,



(a) Noisy Input Activity. (b) Recovered PDF.

(c) Network Output. (d) Mapping.

Figure 5: Uni-sensory Trials.



(a) Activity in Population 1. (b) Activity in Population 2.

(c) Network Output. (d) Mapping.

Figure 6: Multi-Sensory Trials.

to demonstrate that the order of neurons did not matter, we switched the orientation of population 2: An input neuron $\mathbf{i}_k$ in population 2 had the preferred value $\mathfrak{p}_k = 1 - k/10$ as opposed to $k/20$ in population 1.

Both populations were combined into one logical population as shown in Fig. 2. The output neurons and learning algorithm had no access to either the population an input neuron belonged to or the input neurons' position in the population. As expected, the network learned to combine the information from the different input populations. Figs. 6a and 6b show the noisy activities in populations 1 and 2. Fig. 6c shows the PDF recovered by the network and Fig. 6d shows the mapping.

5 Discussion

We have presented a well-motivated ANN model which uses self-organization to learn a latent variable model for a high-dimensional population response. The network computes a PDF and represents it as a population code. The model needs no embedded knowledge about the specific code of the input population or the noise properties. There are two requirements for this algorithm to succeed: one is that the noise is uncorrelated between input neurons (or correlation at least be low). In theory, this can be a serious limitation. However, Naïve Bayes algorithms like ours have proven surprisingly effective in practice. In fact, Zhang has shown that Naïve Bayes is optimal under certain realistic conditions, even if noise is correlated [2005]. The second requirement is that the noise must be low enough so the topological structure of the data space can still be recognized. Both requirements are inherited from the SOM algorithm to which ours is closely related.

As a general machine learning algorithm, our algorithm is applicable wherever a small number of latent variables of the domain manifest themselves in many noisy observables. It is unsupervised and therefore especially useful where the kind of noise and the relationship between latent variables and observables is either unknown or difficult to model precisely. The algorithm can be used both for learning of statistical inference and for extracting and exploring the statistics of large stochastic datasets. It is hard to predict in advance the final spatial organization of the network. This can be seen as a limitation in some applications. However, as we have shown, it is easy to extract the organization of the network after learning, which should be sufficient in most cases.

As a model of neural information processing, the approach presented here has the potential to explain several cognitive phenomena and capabilities found in developing and adult animals. One example at the behavioral level is multi-sensory integration [Ernst and Banks, 2002]: As we have shown, our model can learn how to combine information from different sensory modalities. On the neurophysiological level, there is a wealth of data about neural responses eg. in multi-sensory integration [Stein and Meredith, 1993] in the superior colliculus (SC). Some of the properties of these responses have been explained theoretically within a probabilistic framework (eg. [Anastasio *et al.*, 2000]). A detailed discussion of how behavioral or neurophysiological phenomena can be accommodated in our model will be the topic of future work.

Our neural model makes use of counters and lists of counters. Of course, biological neurons do not have such data structures—at least not explicitly. Soltani and Wang showed how synapses could learn probabilistic inference from single binary cues through reward-dependent learning [2010]. Extensions of this or similar work for multiple, continuous cues and unsupervised learning could be used to implement the mechanisms used here in a biologically plausible way.

In the future, we will further develop our model towards various instances of natural sensory processing. Initial results indicate that our model can well be used to build upon recent approaches to bio-mimetic sound source localization [Dávila-Chacón *et al.*, 2012]. Also, we will explore applications in multi-sensory integration. Since our network learns to process input without a notion of where that input comes from it seems a promising candidate for modeling eg. multi-sensory integration in the integrative deep layers of the SC.

Acknowledgements

This work is funded by the DFG German Research Foundation (grant #1247) – International Research Training Group CINACS (Cross-modal Interactions in Natural and Artificial Cognitive Systems).

References

[Anastasio *et al.*, 2000] Thomas J. Anastasio, Paul E. Patton, and Kamel Belkacem-Boussaid. Using Bayes' rule to model multisensory enhancement in the superior colliculus. *Neural Computation*, 12(5):1165–1187, May 2000.

[Barber *et al.*, 2003] Michael J. Barber, John W. Clark, and Charles H. Anderson. Neural representation of probabilistic information. *Neural Computation*, 15(8):1843–1864, August 2003.

[Beck *et al.*, 2008] Jeffrey M. Beck, Wei J. Ma, Roozbeh Kiani, Tim Hanks, Anne K. Churchland, Jamie Roitman, Michael N. Shadlen, Peter E. Latham, and Alexandre Pouget. Probabilistic population codes for bayesian decision making. *Neuron*, 60(6):1142–1152, December 2008.

[Cuijpers and Erlhagen, 2008] Raymond H. Cuijpers and Wolfram Erlhagen. Implementing Bayes' rule with neural fields. In *Proceedings of the 18th international conference on Artificial Neural Networks, Part II*, ICANN '08, pages 228–237, Berlin, Heidelberg, 2008. Springer-Verlag.

[Dávila-Chacón *et al.*, 2012] Jorge Dávila-Chacón, Stefan Heinrich, Jindong Liu, and Stefan Wermter. Biomimetic binaural sound source localisation with ego-noise cancellation. In Alessandro E. Villa, Włodzisław Duch, Péter Érdi, Francesco Masulli, and Günther Palm, editors, *Artificial Neural Networks and Machine Learning – ICANN 2012*, volume 7552 of *Lecture Notes in Computer Science*, pages 239–246. Springer Berlin Heidelberg, 2012.

[Deneve *et al.*, 1999] Sophie Deneve, Peter E. Latham, and Alexandre Pouget. Reading population codes: a neural implementation of ideal observers. *Nature Neuroscience*, 2(8):740–745, August 1999.

[Deneve *et al.*, 2001] Sophie Deneve, Peter E. Latham, and Alexandre Pouget. Efficient computation and cue integration with noisy population codes. *Nature Neuroscience*, 4(8):826–831, August 2001.

[Ernst and Banks, 2002] Marc O. Ernst and Martin S. Banks. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433, January 2002.

[Georgopoulos *et al.*, 1986] Apostolos P. Georgopoulos, Andrew B. Schwartz, and Ronald E. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, September 1986.

[Gori *et al.*, 2008] Monica Gori, Michela Del Viva, Giulio Sandini, and David C. Burr. Young children do not integrate visual and haptic form information. *Current Biology*, 18(9):694–698, May 2008.

[Gori *et al.*, 2012] Monica Gori, Giulio Sandini, and David Burr. Development of visuo-auditory integration in space and time. *Frontiers in Integrative Neuroscience*, 6, 2012.

[Jacobs and Kruschke, 2011] Robert A. Jacobs and John K. Kruschke. Bayesian learning theory applied to human cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(1):8–21, 2011.

[Jazayeri and Movshon, 2006] Mehrdad Jazayeri and Anthony A. Movshon. Optimal representation of sensory information by neural populations. *Nature Neuroscience*, 9(5):690–696, May 2006.

[King, 2004] Andrew J. King. Development of multisensory spatial integration. In *Crossmodal Space and Crossmodal Attention*, chapter 1, pages 1–24. Oxford University Press, USA, May 2004.

[Klanke, 2007] Stefan Klanke. *Learning Manifolds with the Parametrized Self-Organizing Map and Unsupervised Kernel Regression*. PhD thesis, University of Bielefeld, March 2007.

[Kohonen, 1982] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, January 1982.

[Landy *et al.*, 2011] Michael S. Landy, Martin S. Banks, and David C. Knill. Ideal-observer models of cue integration. In Julia Trommershäuser, Konrad Körding, and Michael S. Landy, editors, *Sensory Cue Integration*, chapter 1, pages 5–29. Oxford University Press, Oxford, August 2011.

[Pouget *et al.*, 2003] Alexandre Pouget, Peter Dayan, and Richard S. Zemel. Inference and computation with population codes. *Annual review of Neuroscience*, 26(1):381–410, 2003.

[Seung and Sompolinsky, 1993] H. Sebastian Seung and Haim Sompolinsky. Simple models for reading neuronal population codes. In *Proceedings of the National Academy of Sciences*, volume 90, pages 10749–10753, AT&T Bell Laboratories, Murray Hill, NJ 07974., November 1993.

[Soltani and Wang, 2010] Alireza Soltani and Xiao-Jing Wang. Synaptic computation underlying probabilistic inference. *Nature Neuroscience*, 13(1):112–119, January 2010.

[Stein and Meredith, 1993] Barry E. Stein and M. Alex Meredith. *The Merging Of The Senses*. Cognitive Neuroscience Series. MIT Press, 1 edition, January 1993.

[Stein, 2012] Barry E. Stein. Early experience affects the development of multisensory integration in single neurons of the superior colliculus. In Barry E. Stein, editor, *The New Handbook of Multisensory Processing*, chapter 33, pages 589–606. MIT Press, Cambridge, MA, USA, June 2012.

[Wallace and Stein, 2007] Mark T. Wallace and Barry E. Stein. Early experience determines how the senses will interact. *Journal of neurophysiology*, 97(1):921–926, January 2007.

[Yin, 2007] Hujun Yin. Learning nonlinear principal manifolds by self-organising maps. In *Principal Manifolds for Data Visualization and Dimension Reduction*, Lecture Notes in Computational Science and Engineering, chapter 3, pages 68–95. Springer, Dordrecht, 2007.

[Zhang, 2005] Harry Zhang. Exploring conditions for the optimality of Naïve Bayes. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(2):183–198, 2005.