Leipzig University of Applied Sciences
Department of Computer Science, Mathematics
and Natural Sciences

A Recurrent Neural Network Model for Pattern Recognition

Holger Arnold

A thesis submitted in partial fulfillment of the requirements
for the degree Master of Science in Computer Science

Submitted on December 15, 2003

Advisors:

Prof. Dr. Jürgen Jost
Max-Planck-Institute for Mathematics in the Sciences

and

Prof. Dr. Siegfried Schönherr
Leipzig University of Applied Sciences

Acknowledgments

I would like to thank the following people who have made writing this thesis easier: Prof. Jost for supporting my interest in neural networks and cognitive systems and for being always open for questions and discussions, Prof. Schönherr for helping to clarify many passages in the text and for his efforts to make this university a fine place for studying computer science, Susanne Schindler for reading an early manuscript and for providing helpful comments, Bent Großmann for many good discussions and talks, and Liane Fischer for supporting me in everything I do for six years now.

Contents

Chapter 1

Introduction

When we are developing technical systems or scientific methods, we often try to get inspiration from the inventions of nature. But there are many examples of natural systems having abilities that still no technical system created by man has. Aside from purely physical features, as for example materials that are remarkably strong and light, the abilities of animals to perceive their environment, to control their own behavior, and to behave socially are fascinating us above all. For most of these abilities, we cannot explain the mechanisms behind them. Although the calculation speed of our computers has grown exponentially over the last decades, even simple animals have capabilities that we are not able to reproduce. For example, how can a small insect, with only a few thousand nerve cells, process the data from thousands of eye facets, put them together to a view of its environment, and control a stable flight?

The most impressive property of the neural data processing in simple animals is its efficiency. Obviously, a handful of neurons can be enough to realize complicated tasks of perception and control. In higher animals, and especially in mammals, it is the incredible complexity of the nervous system that impresses us most. How can structures composed of billions of communicating neurons function together effectively and efficiently? The computational power of the brains of higher animals is especially obvious when we consider the processing of visual information. What animals accomplish in this area is far beyond what we can do by today's technical means.

A great part of the computing power of the natural neural systems of higher animals is based on the ability to detect patterns in unstructured, high-dimensional data. It would mean a considerable improvement for many applications if we could build systems similarly powerful in the processing of large data sets as the brains of higher animals. The analysis of data plays an important role for today's human life. The analysis of seismic data in the search for natural resource deposits and the recognition of incidents in process control are only two examples for the application of data analysis methods, and it is not hard to find many others. As the evolution has produced such powerful natural systems over millions of years, it is without doubt a wise approach to study them and to find out the principles of their function. Although we may not be able to completely understand these systems, and we may not have the technical capabilities to realize them, we can hope to discover basic

principles that we can use for our own developments.

Artificial neural networks, abstract models of the natural neural structures, have been investigated as computational models since the early days of computer science. Since then, they have been successfully applied to many different problems, especially in the area of pattern recognition. In this work, we are developing a *recurrent* neural network model to solve pattern recognition tasks. The application of a recurrent network to pattern recognition means that information is not only processed bottom-up, from the concrete data to the abstract concepts, but also in the opposite direction. It is especially important for us to develop our model from abstract concepts to a concrete neural network implementation that can be tested in computer simulations.

In Chapter 2, we explain why it is wise to investigate recurrent networks at all, and not only pure feed-forward networks. We use neuroanatomical facts and the results of neuropsychological experiments for our argumentation. At the end of the chapter we describe two recurrent network models that both had a significant influence on the development of this area of research. In Chapter 3, we develop the theoretical foundation for our network model. We apply concepts from information theory to specify how the bottom-up path and the top-down path in the network should behave. Chapter 4 is concerned with the implementation of the abstract model as a neural network. We give several implementation alternatives and test the effectiveness of the model using computer simulations of the network. Chapter 5 summarizes the results of this work and gives an outlook on possible further developments.

Chapter 2

Recurrent neural networks in nature

Why should we consider recurrent neural networks as tools for pattern recognition? The recurrent connections between the neurons make the analysis of the network's behavior much more complicated than in pure feed-forward networks. Is it worth the effort? In this chapter, we give two main arguments in favor of recurrent networks. The first argument is the neuroanatomical fact that there are actually recurrent connections in natural brains. We especially consider the recurrent connections in the cortex, which is the brain structure that makes up the greatest difference between the nervous systems of mammalian and non-mammalian animals. If there are recurrent connections in the brains of the animals with the greatest associative capabilities, there must be a reason for this. The second argument comes from results of neuropsychological experiments done with monkeys and humans. Some of these results can only be explained by the assumption that recurrent connections play an important role in the process of pattern recognition.

2.1 Basic neuroanatomy of the cortex and the thalamus

A strong indication that recurrent neural networks play an important role for some brain functions comes from the neuroanatomy of the cortex and the thalamus. It has been shown that there are extensive recurrent connections between different cortical areas, as well as between cortical areas and the thalamus. This section gives an introduction to the basic neuroanatomy of these two structures, the cortex and the thalamus. Later in Chapter 4, where we develop a neural network implementation of our abstract pattern recognition model, we shall come back to this subject using a more detailed view. Most of the information given in this section have been taken from [26] and [27].

Throughout this work, we usually refer to the brain structure of mammals. What makes them interesting for us are their higher-level cognitive functions, which are much more developed than in non-mammalian animals. Only these high cognitive capabilities enabled the various mammalian species to develop

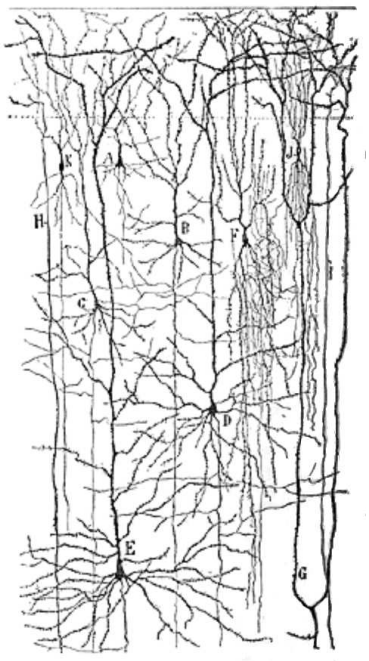


Figure 2.1: Drawing of the superficial layers in the human frontal cortex

their complex social behaviors, allows them to adapt their behavior to changing environments, lets primates use tools in a goal-directed way, and gives humans the unique ability to abstract and to communicate by language. There are a number of structural differences between the brains of mammals and those of other vertebrates, and there are even larger differences between the brains of vertebrates and the neural structures of non-vertebrate animals. In contrast to this, the brains of most mammalian species share the same fundamental blueprint.

2.1.1 The cortex

When we speak of the cortex, we actually mean the neocortex; the other, more primitive parts of the cortex, namely the paleocortex and the archicortex, play no role for us. The neocortex in adult humans forms a heavily folded, 2–3 mm thin layer in the upper part of the brain, known as the gray matter. It has a surface area of about 200000 mm^2 and an average neuron density of about 100000 neuron cells per mm^2, resulting in approximately 20 billion neocortical

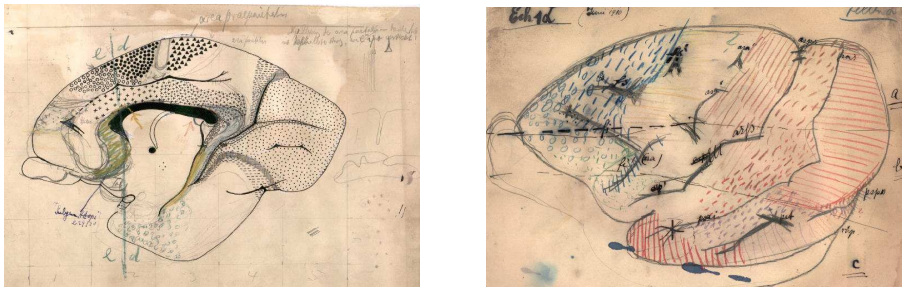


Figure 2.2: Two original brain drawings by Brodmann

neuron cells.[1]

What may be surprising is that the structure of the neocortex is nearly uniform over its whole surface. It consists of six layers that are characterized by their cell populations and their connectivity. Although more and more substructures have been identified in these layers in the recent years, the general structure has been confirmed in a number of experiments. A possible explanation of this uniformity is that it has developed by replicating the basic structure over and over in the evolution of the species. Figure 2.1 shows an old drawing of the superficial layers in the human frontal cortex. It is one of many high-quality drawings produced by Santiago Ramón y Cajal using a staining technique developed by Camillo Golgi. In recognition of their work, Ramón y Cajal and Golgi received the Nobel Prize in Physiology or Medicine in 1906.

The cortex of a mammalian brain can be divided into a number of areas. The first areas have been identified by Korbinian Brodmann and others at the beginning of the 20th century based on small differences in the cell types. Figure 2.2 shows two original brain drawings by Brodmann. His studies led him to a cytoarchitectonic map of the cortical areas, which is shown in figure 2.3. Later, this subdivision into areas has been refined and corrected using lesion studies, done for example with stroke patients, and new imaging methods. Today, it is assumed that each hemisphere of the human cortex contains about 100 areas, with approximately 100 million neurons in each area.

It has been confirmed in many experiments that each cortical area has a specific function, and many of these functions have been identified at least partially. Based on their connectivity and on results from brain imaging studies, the cortical areas can be partially ordered with respect to their relative functional level. There are lower areas that are concerned with more concrete, sensory or motor related data, and higher areas that are concerned with more abstract concepts.

Although the functional division into areas may suggest this, the way the

[1] Note that such numbers are only rough estimates and that the numbers given in different publications vary considerably.

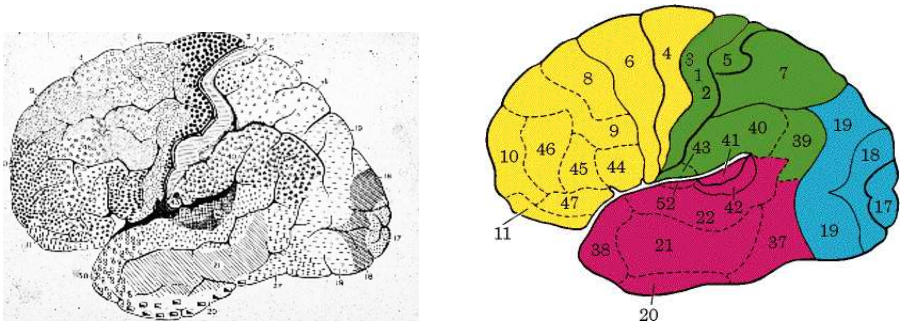


Figure 2.3: Map of the cortical areas

cortex is composed of areas is not the same as the way a computer program is composed of subprograms or the way a human-made machine is built up from parts. Whereas the parts of a computer program or a machine are usually designed as independent modules, which hide their internal state and communicate as little as possible, the cortical areas are highly interconnected. Actually, more than 60% of the neurons in the cortex project their output not only locally, but also to another area of the brain. This means that no area works isolated from the others and suggests a computational and compositional model that is quite different from the one we know from human engineering.

Despite the extensive interconnections, each area is connected only to a few other areas. It has been found in experiments that of the 10000 directed paths that are possible between the estimated 100 areas in each hemisphere only about 2000 exist. Most connections between areas are symmetric in their connectivity and in the number of connections: if area A projects to area B, then area B also projects to area A, and the number of connections has the same order of magnitude in both directions. In Chapter 4, we shall look at the different cell types in the cortex and how they form connections. The essential point for us now is that if two cortical areas are connected at all, then they are usually recurrently connected.

2.1.2 The thalamus

The thalamus is a small subcortical structure that is located in the center of the brain, at the top of the brain stem. It consists of two symmetric ellipsoidal parts, one in each cerebral hemisphere. The thalamus is composed of about 50–80 nuclei, and each nucleus contains approximately 2 million neurons.

What makes the thalamus interesting for us are its connections to the cortex. All sensory input to the cortex, with the exception of the olfactory sense, passes the thalamus before it is relayed up to the cortical areas. Most cortical areas receive their input from a specific nucleus of the thalamus. But

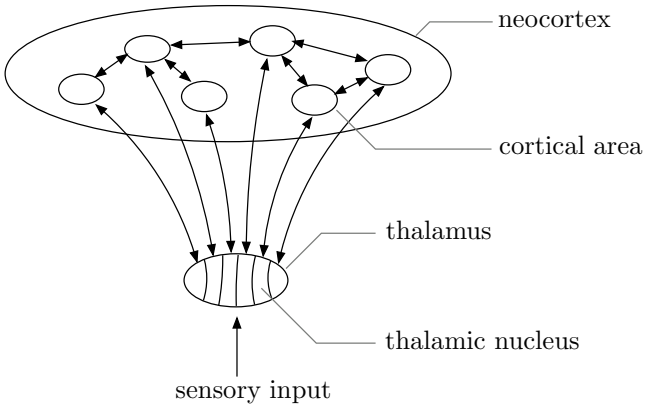


Figure 2.4: The connections between cortical areas and the thalamus

information flows not only from the thalamus to the cortex: most thalamic nuclei receive at least as many connections from cortical areas as they send to them. There are even pairs of nuclei and areas where the number of backward connections is much greater than the number of forward connections. The recurrent connections between a cortical area and its thalamic nucleus are similar in their structure to the connections between two cortical areas. Their layout suggests that to a cortical area, its thalamic nucleus appears similar to an area of a lower level.

The thalamus gets its input from multiple sources. Some nuclei of the thalamus directly receive sensory signals and transmit them to the primary sensor areas of the cortex. An example is the lateral geniculate nucleus (LGN), which receives visual sensory data coming from the retina through the optic nerve. This nucleus is connected to the primary visual area (V1) in the cortex. Other nuclei receive inputs coming from further subcortical structures. An example for this is the posterior ventral lateral nucleus (VLp), which receives motor-related signals from the cerebellum and transmits them to the primary motor area of the cortex. If we compare the number of connections that the thalamus receives from the various sources, however, we find that much more connections are coming from the cortical areas than from the other sources. This is a remarkable finding, as it means that the top-down information flow must be very important for the processing of the sensory information.

To understand the difference in number of connections from the multiple sources, let us look at some numbers which have been found for the visual system of the cat [35]. As in all mammals, the retinal information is transmitted along two pathways to the cortex, the X pathway (called P in primates), which processes shape and color data, and the Y pathway (called M in pri-

mates), which processes motion data. Taken these two pathways together, 10^5 axons transmit information from the retina to the thalamus where they form synapses with $3 \cdot 10^5$ LGN neurons, which relay the visual information up to the cortex. The pathway that runs from the cortex back to the thalamus, however, consists of $4 \cdot 10^6$ axons that form synapses with LGN cells. This means that the LGN receives 40 times as many connections from the cortex as it receives from the retina, and 13 times as many connections as it sends to the cortex. Even by taking into consideration that the retinal axons may have more synapses than other axons, it is estimated that only 10–20% of the synapses on LGN neurons are coming from retinal axons, and 80–90% from cortical axons. Although the LGN may be an extreme case among the thalamic nuclei, these figures indicate that recurrent connections must play an important role in the processing of the sensory data.

2.2 Results from neuropsychological experiments

Besides the neuroanatomical fact that there are recurrent connections in the cortex and between cortex and thalamus, there are many results from neuropsychological experiments which suggest that the recurrent connections are important to the function of these structures. We now look at some of these results and try to interpret their effects.

2.2.1 Illusory contours

There are situations, which are known as optical illusions, where our perception can not be explained solely by the visual information that reaches our retina. One of these optical illusions is the illusory contour effect. This is the effect that we perceive the border of a shape at a position in our visual image where there is no border. The Kanisza square [17] shown in figure 2.5 is an example of a visual stimulus where this effect occurs: the four corner disks form an imaginary square whose borders we perceive, although there is no visual evidence for them.

In [21], Lee and Nguyen analyze the responses of neurons in the cortical visual areas V1 and V2 of macaque monkeys, while they are exposed to a Kanisza square and several other stimuli. Figure 2.6 shows the experimental setting: neurons from V1 and from V2 were compared with respect to their firing rate over time, which is illustrated in the diagrams on the right. It was already known from earlier experiments that some neurons in area V2 respond to illusory contours. A common interpretation for this is that area V2 can perform basic shape composition because the receptive fields (the subsets of the input from which the neurons directly receive input) of V2 neurons are large enough to capture the presented shape.

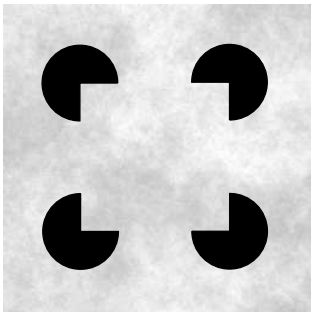


Figure 2.5: The Kanisza square

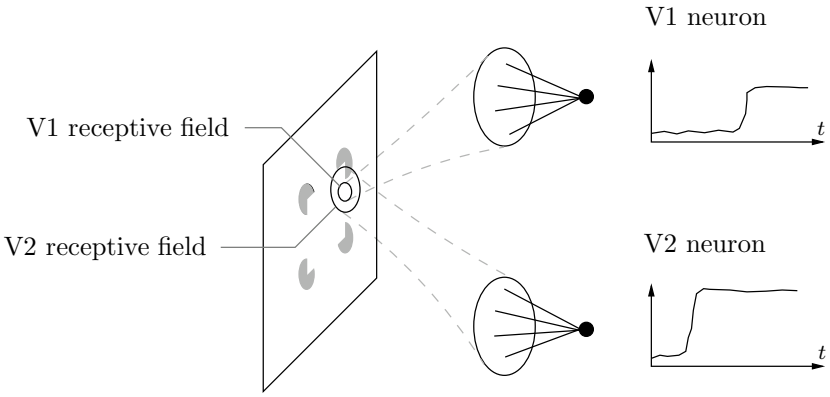


Figure 2.6: An experiment to measure the illusory contour effect

In this experiment, however, not only V2 neurons responded to the illusory contours, but also neurons in the lower-level visual area V1. The Kanisza square was scaled so that the receptive fields of the V1 neurons were smaller than the distance between the corner disks. This means that the responding neurons in V1 can not have detected the illusory border in the visual information coming from the retina. Because the square shape in the Kanisza figure is an abstract concept, it is hard to believe that the border signal has spread out laterally from the neurons that detected the corners. A convincing explanation for the effect is that the border information comes from neurons in higher-level areas, for example from the neurons in V2. This theory is supported by the observation that the responses of the neurons in V1 were weaker than the responses of the V2 neurons, and that the V1 neurons responded later than the V2 neurons (100–190 ms after the presentation of the stimulus for V1, compared to 70–95 ms for V2).

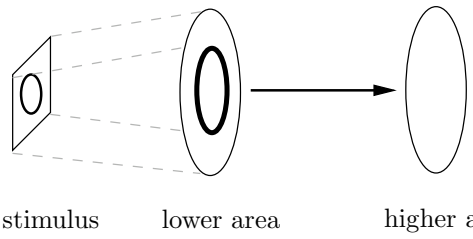
2.2.2 Backward visual masking

In a masking experiment, two stimuli, the target and the mask, are successively presented to a subject. If the target is presented before the mask, we speak of backward masking, otherwise we speak of forward masking. Depending on the delay between target and mask, called the stimulus onset asynchrony (SOA), the mask can disturb or completely prevent the perception of the target. At least the effect of backward masking is attributed to be a consequence of the recurrent computation in the cortex.

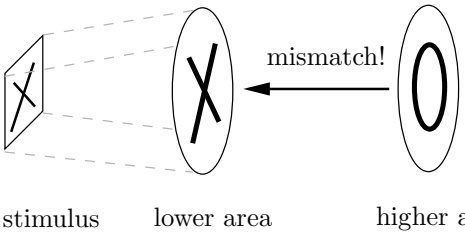
In [32], Rolls et al. investigate the effects of a backward visual masking experiment on the neural level. They recorded the responses of neurons in the inferior temporal cortex (IT) and the superior temporal sulcus (STS) of macaque monkeys. IT and STS belong to the ventral stream of visual information processing. They are part of the temporal cortex, which is commonly supposed to be the region where the recognition of complex objects like faces takes place.

The authors of the cited article used different faces as target stimuli and compared the responses of face-specific neurons under different masking conditions. They found that with decreasing SOA, the difference of the firing rates in response to the more effective and to the less effective stimuli decreased. This means that the ability to separate the different stimuli by the firing rate of the recorded neurons decreased. The authors also computed the mutual information of the stimuli and the resulting spike train. The result was that with a decreasing SOA, the resulting spike train contained less and less information about the presented target stimulus.

The reduction of information about the stimulus can be interpreted as the neurophysiological reason for the psychological masking effect. An explanation of this effect, given for example in [19], is that the mask, as it comes closer in time to the stimulus, interrupts the recurrent processing of the stimulus. When the information about the target stimulus are fed back from higher



1. The information about the target stimulus is transmitted from the lower to the higher area.



2. Some milliseconds later, the information about the target stimulus is fed back from the higher to the lower area and clashes with the information about the mask stimulus.

Figure 2.7: The backward visual masking effect

to lower areas, there is a mismatch with the data about the mask stimulus, which is now coming from even lower or sensory areas. Figure 2.7 illustrates this situation.

2.2.3 Figure-ground segmentation

The separation of figural objects from the background is an important cognitive task and a prerequisite for object recognition. In a realistic environment, where objects are partially occluded by other objects and virtual contours are generated by shadows or surface textures, figure-ground segmentation cannot be achieved using only local filter operations. Therefore, higher visual areas must be involved in this task. If we look, for example, at the famous image of the Dalmatian dog in figure 2.8, we need only a small effort to see the contours of the animal. Without the higher-level concept "dog", using only local information, we would not be able to recognize any contours, and consequently, we would see only a meaningless pattern of white and black dots.

In [20], Lee et al. attempt to verify the hypothesis that the information whether a point in the visual field belongs to an object or to the background

Figure 2.8: The famous Dalmatian dog

is already available in the lower visual areas. They recorded the responses of a number of neurons in the primary visual area V1 of macaque monkeys over a longer time interval. The stimuli they used contained textured borders, textured stripes, and textured and uniformly colored rectangles and disks over a background. Because their receptive fields were too small, the recorded neurons could not detect from the retinal information alone whether they scanned a part of a figure or the background.

In the experiments, it was found that 40–60 ms after the presentation of the stimulus the recorded neurons worked as local filters, and after 80–200 ms their responses changed depending on contextual information. Over this time, the firing rate of the neurons that represented object borders, object centers, and the interior of objects increased, while the firing rate of neurons that represented background decreased. The possibility that these effects were caused by local interactions among V1 neurons was ruled out by the design of the experiments. Therefore, a convincing explanation of the results is that they are the consequence of feedback from higher visual areas.

2.3 Models of the cortical function

The neuroanatomical and neuropsychological evidence for recurrent information processing in the cortex have led to the development of a large number of different models of the cortical function. In this section, we exemplarily look at two of these models that both had a considerable influence on this field of research.

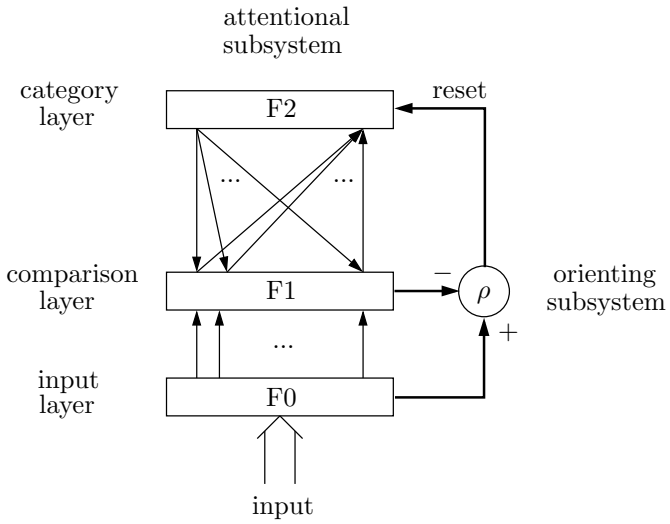


Figure 2.9: The architecture of an ART network

2.3.1 Adaptive resonance theory

Every learning system is confronted with the problem that it must be able to continuously adapt to new data without overwriting what it has learned before. This problem is called the stability-plasticity dilemma [10, 11]. To solve it, Carpenter and Grossberg developed their adaptive resonance theory (ART) [5, 6, 7], which became one of the most influential theories of the cortical function.

The adaptive resonance theory specifies a model for an unsupervised classification network. In a learning process, this network develops a set of categories from a given training data set. When the network has developed a representative category set, it can be used to classify further input vectors into the categories built in the learning process. In an ART network, learning and recognition are typically interleaved until the network has reached a stabilized state. Categories are represented by reference vectors, which are adapted to matching inputs. The stability-plasticity dilemma is solved by adapting only reference vectors that are sufficiently similar to the input. For input vectors that cannot be classified, a new category is created.

Figure 2.9 (redrawn from [2]) shows the simplified architecture of an ART network. It is composed of three layers, the input layer F0, the comparison layer F1, and the category layer F2. The input layer stores the input vector and transmits it to the comparison layer. Comparison layer and category layer are fully interconnected. The weights of these connections, which are symmetric in both directions, represent the reference vectors of the different categories.

The input in the comparison layer activates the categories in the layer F2 with different strengths. The attentional subsystem selects the category with the strongest activation and suppresses all other categories. To test whether the reference vector of the strongest category is sufficiently similar to the input, it is sent back to the comparison layer where it is compared to the input coming from F0. The threshold value for the similarity is determined by a constant ρ. If the vectors are sufficiently similar, the input is assigned to the selected category, and its reference vector is adapted in the direction of the input vector. If the vectors are not sufficiently similar, the orienting subsystem generates a reset signal, which deactivates the selected category for the whole presentation of the current input. The attentional subsystem then again selects the category with the strongest activation. This hypothesis-testing cycle continues until a matching category is found. If there is no such category, a new one with the current input as reference vector is created.

It can be shown that if a fixed set of inputs is used, the network stabilizes after a certain number of iterations. In the stabilized state, every input vector from the set can be classified without search, so that the matching category is always activated in the first cycle. Thus, in the ART model, the recurrent connections are only required for the learning process, and have no real function for the actual recognition process.

Many variants of the ART model have been developed since its introduction. A descendant of ART by Raizada and Grossberg is the LAMINART model [30], a theory of the functional role of the layered architecture of the cortex. With this model, the authors attempt to solve what they call the preattentive-attentive interface problem: Most areas in the cortex process information from different sources. There are bottom-up connections coming from sense organs or lower areas, top-down connections coming from higher areas, and lateral connections coming from neighboring areas of the same level. In the opinion of the authors, a cortical area must keep apart the different sources of information in order to know which part of the incoming data describes objects of the physical environment, and which part is only the result of feedback from higher areas.

Using a part of the visual system as an example, the authors of the model argue that data coming through the lateral connections must be able to generate activity in an area, while data coming through the top-down connections must only be able to support activity already generated by other sources. Using neuroanatomical data, the authors suggest a concrete neural circuit consisting of two feedback loops, one for lateral interaction, one for top-down interaction. These two loops interact through a common interface.

In all models that are derived from the adaptive resonance theory, bottom-up activity is enhanced when the input and the activated category match, and attenuated when they do not match. This is the opposite of what happens in the model described in the next subsection.

2.3.2 Mumford's theory

In two consecutive articles [26, 27], David Mumford presented his theory on the computational architecture of the cortex. The first article is concerned with the role of the recurrent connections between thalamus and cortex. The classical theory for the role of the thalamus was that it relays the signals coming from the sense organs up to the cortex, but this theory cannot explain the purpose of the large number of connections going from the cortex back to the thalamus.

In his article, Mumford proposes that each nucleus in the thalamus contains the view of the world for the cortical areas that this nucleus is connected to. For lower sensory areas, this view directly correlates to the signals coming from sense organs, and for higher areas, this view reflects more abstract concepts. Mumford compares the thalamus to a blackboard, a data structure that is commonly used to coordinate the work of a number of parallel and independent problem solving processes. In his model, the forward connections transmit the current view of the world from the thalamus to the cortex where it is analyzed and updated. The backward connections then transmit the modifications suggested by the cortical areas back to the thalamus where the suggestions of the different areas are integrated. Because this integration is an active process, Mumford calls the thalamus an active blackboard.

The second of the cited articles is concerned with the role of the connections between cortical areas that work on different functional levels, one on a higher level than the other (see section 2.1.1). For the top-down connections, Mumford proposes that they are used to transmit higher-level knowledge to the lower area, where it is used to eliminate noise and ambiguities from the lower-level data. He proposes that the top-down connections carry a reconstruction of lower-level data that corresponds to the state of the higher area. This reconstruction is called a template in his theory. In the case that the lower area is directly connected to sensory input from the thalamus, as for example the visual area V1, such a template would be very similar to a sensory signal.

To compensate for the variations in real-world data, Mumford assumes that the templates are parametric, and that the top-down connections transmit multiple instances of a template at a time. The lower area compares the template it receives from the higher area to its own data, possibly coming directly from sense organs, and computes the difference. This difference, which is not a simple numeric value, but in structure similar to the data itself, is called a residual in Mumford's theory. When it has been computed, it is transmitted through the bottom-up connections to the higher area where it is used to adapt the state and to generate new templates. This process continues until the template from the higher area completely matches the data of the lower area, which means that the state of the higher area fully explains the data it receives.

Although Mumford's theory has many interesting aspects, he remains rather

abstract in his suggestions, at least in the cited articles. In particular, he does not propose a detailed neuronal model or an algorithm to implement his theory.

Chapter 3

Theoretical foundation

Our goal in this work is to develop a neural network model that can be applied to solve pattern recognition tasks. In this chapter, we build up the theoretical foundation for our network model. Our starting point is the question how a pattern recognition system *should behave*. Proceeding from this question, we suggest an abstract recurrent model for the function of pattern recognition processes in the cortex, which is based on information-theoretical considerations.

3.1 Interaction of features and categories

The purpose of a pattern recognition system is to classify inputs presented to the system into a number of categories. To classify a given input vector, the system[1] evaluates certain properties of this vector; these properties are called features. Depending on the type of the input, features can have different meanings. For example, in images, the pixel values could be used as features, in visual shapes, the features could be lines and other basic elements, and in sounds, the features could be frequency values.

Based on its observations, the system assigns probabilities to the different categories, that is, the system forms a hypothesis. We assume that the categories have been developed in a supervised or unsupervised learning process. In such a process, sample inputs for the different categories, which are representative of the data that are to be analyzed, are presented to the system. During the recognition of an input pattern, the categories are fixed. Because it is not always possible to uniquely assign an input pattern to a single category, the output of the system is a probability distribution over the set of categories.

How should such a pattern recognition system behave? If we consider realistic sensory data, we find that it is usually very high-dimensional and serves multiple pattern recognition processes. For a single pattern recognition task, however, most of the information in a sensory data stream is irrelevant and makes the task more difficult. Because realistic sensory input is noisy, these additional data can even misguide the system completely. It should therefore be a good strategy not to evaluate the whole input, but only the parts that are

[1] When we speak of "the system" in this work, we refer to the pattern recognition system that we are developing.

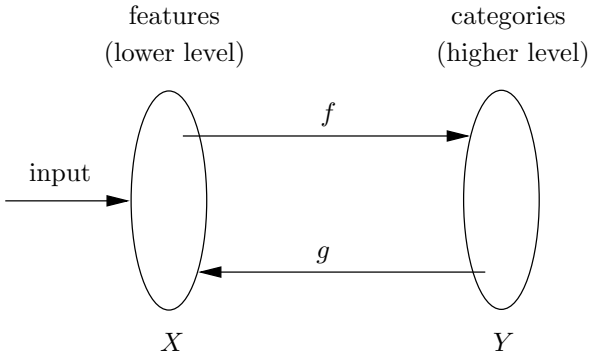


Figure 3.1: The interaction of features and categories

relevant for a specific task. Which features in the input are relevant depends on the specific categories that are considered, or more general, on the internal view of the world that the system has at a certain point in time. During the recognition process, this internal view changes due to the observation of features in the input and also due to information coming from other systems, for example from higher cognitive structures.

These thoughts lead us to a model which is based on an interaction between features and categories. We assume that there are M distinct features $X = (x_1, \ldots, x_M)$ where x_k is the value of the feature k. At a given point in time, the values of the features that have been evaluated up to that point are known to the system. The values of the other features may be unknown, but the system can use knowledge about dependencies between the features to derive information about features that have not yet been evaluated. We assume that there is a fixed set of N categories with probabilities $Y(t) = (y_1(t), \ldots, y_N(t))$ where $y_i(t)$ is the probability that the input corresponds to category i at time t. Features and categories then interact in the following manner:

1. A subset of the features in the input is evaluated. The information acquired by these observations change the probabilities of the categories.

2. The probabilities of the categories in turn determine which of the features are evaluated next.

This interaction continues until the system comes to a conclusion about the presented input.

As illustrated in figure 3.1, our pattern recognition system is composed of two subsystems: one on a lower level that deals with more local and concrete data, and one on a higher level that deals with more global and abstract data. These subsystems communicate via two pathways, the bottom-up pathway f,

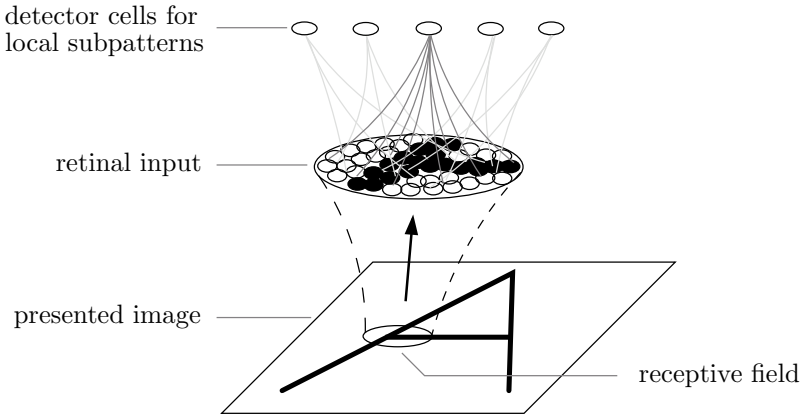


Figure 3.2: Extracting local subpatterns from visual input

which transmits information about the observed feature values, and the top-down pathway g, which transmits information about the probabilities of the categories.

3.2 Example: visual pattern recognition from local subpatterns

The recognition of visual patterns in mammals is a highly complex process that involves many cortical and non-cortical regions of the brain. In this work, we use a small part of this process as a reference example. As natural brains are much more complex than the systems we are able to simulate at this time (the visual pattern recognition processes in humans probably involve billions of neurons), we should not expect that a computer model could be similarly powerful. Our aim can only be to gain some insight into the process of visual pattern recognition by modeling a small but fundamental part of this process.

It is known that there is a stage in the processing of the visual information where local subpatterns such as lines, edges, or corners of specific orientations are extracted. Detector cells for these subpatterns, called simple cells, have been identified in the visual area V1 and other cortical areas of cats and monkeys first by David Hubel and Torsten Wiesel [15], who received the Nobel Prize in Physiology or Medicine in recognition of their work in 1981, together with Roger Sperry.

Figure 3.2 illustrates how the subpatterns are extracted from an image. Every detector cell has an associated receptive field, which is the subset of the retinal input from which it directly (through forward connections) receives information. Usually, the receptive fields of neighboring detector cells overlap.

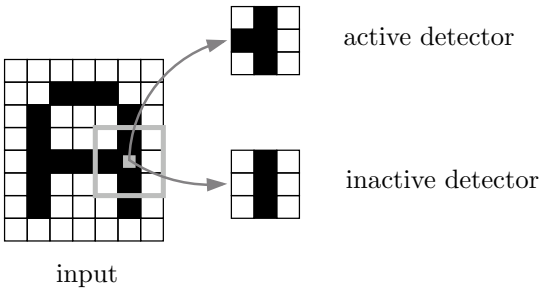


Figure 3.3: Extracting simplified features from visual input

The subpatterns that are detected in the visual input are integrated to more complex patterns in higher brain regions. In this example, we use the local subpatterns as features and the complex patterns as categories.

To simplify the problem, we assume that the visual input consists of a rectangular matrix of pixels, which are either set (black) or unset (white). A subpattern, and thus a feature, is a 3×3 matrix of pixels. Because we use only those subpatterns whose center pixel is set, there are $2^8 = 256$ different features. We assume that for every pixel in the input, there is one detector for each feature, whose center is located at the position of this pixel. When a feature is evaluated, its detector becomes active if the pixel pattern in the input around the center of the detector matches the pixel pattern of the feature; otherwise the detector stays inactive. Active detectors have the output 1, inactive ones have the output 0. It is also possible to allow values from the interval $[0, 1]$ as output of the feature detectors, depending on the similarity between the pixel pattern of the detector and the pixel pattern in the input.

Figure 3.3 shows two feature detectors at the same pixel location in the input. The upper detector is active because its pixel pattern matches the pixel pattern in the input. The lower detector is inactive because its pixel pattern differs from the pixel pattern in the input. Obviously, at most one of the feature detectors at each pixel location can be active. Because the receptive fields of neighboring detector cells overlap, the value observed at a pixel location restricts the values that the detectors at neighboring pixel locations can observe.

3.3 A step toward translation invariance

The visual system of mammalian animals has the ability to reliably recognize objects whose images have been subjected to complex visual transformations, resulting for example from changes in the lighting conditions, viewing direction, position, or distance. There are several hypotheses about the mechanisms

in the visual system that compensate for all these changes. Two important principles that can be the basis for transformation invariant object recognition are replication and normalization [37, 40].

Replication means that a subsystem that detects a specific object is replicated over the space of transformations. For example, to compensate for changes in object position, the subsystem that detects a specific object can be replicated at different positions in the retinal data stream. To compensate for changes in object distance, multiple subsystems that all detect the same object but at different sizes can be used. Similar strategies can be applied to compensate for other transformations. It is obvious that an approach that is solely based on replication is impractical because it suffers from combinatorial problems. The number of replicated subsystems that would be required to compensate for all combinations of transformations and objects is simply too large. Nevertheless, in [39], Wang applies replication to model a neural network which recognizes patterns invariant to translation, rotation, and scaling.

Normalization, on the other hand, means that the input is transformed to a normalized representation that is independent of the input transformations. The object detecting subsystems then work on this normalized representation. In this approach, the representation of an object consists of its invariant features, which are the features that remain unchanged by the transformations that are to be compensated. Although the normalization approach may seem more elegant than simple replication, it has its own problems. Extracting features from the input that are invariant to a number of transformations is a very difficult task, while the neural circuits in the visual system have a relatively simple structure. Besides that, if the visual system could transform every input to a normalized representation, then an object that has been learned once should be immediately (without further learning) recognizable under various transformations. This, however, is not consistent with the results of some neuropsychological experiments [28].

In [37], Ullman and Soloviev describe a model for translation invariant pattern recognition that can easily be applied to our example from section 3.2. The model is simple and biologically very plausible. It uses both, replication and normalization. In this model, the detectors for local subpatterns are replicated over the whole visual field. It is important that the receptive fields of the detectors at neighboring positions overlap. The output of all detectors for a specific subpattern at different positions is then joined to a single detector. This means that for each local subpattern, it is only detected how frequently it occurs in the input, but not at which positions it occurs. In the cited article, the authors work with subpatterns similar to the ones in our example. They show that the information how frequently each subpattern occurs in the input is sufficient to uniquely identify the greatest part of the possible patterns. In simple cases, it is even sufficient to detect only whether a subpattern occurs in the input or not. As no position information is involved, the recognition process is translation invariant. The reason why it is still possible to identify most patterns is that the detectors implicitly provide relative position infor-

mation because of their overlapping receptive fields. Actually, most possible input patterns can only be composed of a single set of subpatterns. Of course the number of patterns that cannot be uniquely identified increases with the size difference between patterns and subpatterns (other parameters play also a role).

3.4 Information entropy

Before we can go on in the development of our model, we need to introduce some concepts from information theory [1, 16]. The first one is the concept of information entropy, developed by Claude Shannon and Warren Weaver in their efforts to quantitatively understand information [34]. In their theory, the information content $I(x)$ of an event x with probability $P(x) \neq 0$ is given by

$$I(x) = -\log P(x) \tag{3.1}$$

Events with a probability of 0 or 1 contain no information. The base of the logarithm is just a matter of the unit in which the information is measured. Shannon originally used base-2 logarithms and created the term "bit" for this unit of information. In this work, we use natural logarithms and add a definite value at 0:

$$\log x := \begin{cases} \ln x & \text{for } x > 0 \\ 0 & \text{for } x = 0 \end{cases} \tag{3.2}$$

Let X be a discrete random variable that can take values from the set $\{x_1, \ldots, x_n\}$ with a probability distribution $p = (p_1, \ldots, p_n)$ where $p_i := P(X = x_i)$ is the probability that X takes the value x_i. The information entropy $H(X)$ of X is then defined as the expectation of the information content of X with the probability distribution p:

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \tag{3.3}$$

For a continuous random variable X with a probability density function p, the summation is replaced by an integration over the possible values of X (we assume that X takes real values):

$$H(X) = -\int_{-\infty}^{+\infty} p(x) \log p(x)\, \mathrm{d}x \tag{3.4}$$

In cases where only the probability distribution itself is of interest and not the values of the random variable, we also speak of the entropy $H(p)$ of the probability distribution p.

The entropy of a random variable is the amount of information that we can

expect to gain when we measure its value. In this sense, entropy measures the degree of uncertainty that we have about the value of a random variable. If one of the values of X has probability 1, and consequently all other values have probability 0, then the value of X is completely determined. Because we can gain no information from measuring such a variable, its entropy is 0. If, on the other hand, all values of X have equal probability, then we are maximally uncertain about its value. In the discrete case, the entropy of such a uniformly distributed random variable is $-\log 1/n$.

We now show that the information entropy $H(X)$ is a suitable measure for the amount of uncertainty about the value of X. For this, we show that, given a discrete probability distribution p, any modification of the probabilities of p toward the uniform distribution results in a distribution with greater entropy than p.

Lemma 3.1 Let $f : \mathcal{I} \to \mathbb{R}$ be a twice differentiable real function on the interval $\mathcal{I}$ with $f''(x) > 0$ for all $x \in \mathcal{I}$. Let $a, b \in \mathcal{I}$, $\varepsilon \in \mathbb{R}$ with $a > b$ and $0 < \varepsilon < a - b$. Then

$$f(a) - f(a-\varepsilon) > f(b+\varepsilon) - f(b) \tag{3.5}$$

Proof First, we note that $f(a)-f(a-\varepsilon) = \int_{a-\varepsilon}^{a} f'(x)\,\mathrm{d}x$ and $f(b+\varepsilon)-f(b) = \int_{b}^{b+\varepsilon} f'(x)\,\mathrm{d}x$. Because $f''(x) > 0$ for all $x \in \mathcal{I}$, f' is strictly monotonic increasing in $\mathcal{I}$. This means that $f'(x) > f'(y)$ for all $x \in [a-\varepsilon, a]$ and for all $y \in [b, b+\varepsilon]$, and consequently $\int_{a-\varepsilon}^{a} f'(x)\,\mathrm{d}x > \int_{b}^{b+\varepsilon} f'(x)\,\mathrm{d}x$. □

Proposition 3.2 Let $p = (p_1, \ldots, p_n)$ and $p' = (p_1, \ldots, p_i - \varepsilon, \ldots, p_j + \varepsilon, \ldots, p_n)$ be two discrete probability distributions, let $p_i > p_j$ and $0 < \varepsilon < p_i - p_j$. Then p' has a greater entropy than p: $H(p') > H(p)$.

Proof

$$\begin{aligned} H(p') - H(p) = {} & p_i \log p_i - (p_i - \varepsilon)\log(p_i - \varepsilon) \\ & + p_j \log p_j - (p_j + \varepsilon)\log(p_j + \varepsilon) \end{aligned} \tag{3.6}$$

We set $f(x) := x \log x$, $\mathcal{I} := [0, 1]$, $a := p_i$, $b := p_j$, and get:

$$H(p') - H(p) = (f(a) - f(a-\varepsilon)) - (f(b+\varepsilon) - f(b)) \tag{3.7}$$

with $f'(x) = 1 + \log x$ and $f''(x) = 1/x$ for $x > 0$, $\lim_{x\to+0} f'(x) = -\infty$, and $\lim_{x\to+0} f''(x) = +\infty$. By applying Lemma 3.1, we see that $H(p') - H(p) > 0$ and therefore $H(p') > H(p)$. □

The information entropy of a random variable X is invariant to permutations of the values of that variable. We can therefore conclude that any modification of a probability distribution p toward the uniform distribution

results in a distribution p' with higher entropy than p. It immediately follows that the uniform distribution is the distribution with maximum entropy. This shows that we can use the entropy of a random variable X as a measure for the amount of uncertainty about X, thus as a measure for the amount of information that is contained in X. It is even possible to show that information entropy is the only consistent measure for this quantity; proofs of this proposition can be found in [1, 16].

3.5 The maximum entropy principle

The process of pattern recognition as described in section 3.1 is an example of a situation where we must draw conclusions based on data that is often incomplete and noisy. The results of these conclusions are the probability values that we assign to the different categories. To draw valid conclusions, we must take into account all the knowledge we have, while we must avoid to use information that is not contained in the data. When we assign probabilities to the categories based on the data, we must therefore keep as much uncertainty as possible in the resulting probability distribution.

We have seen in the preceding section that without further constraints the probability distribution with the greatest amount of uncertainty is the uniform distribution. If we have no information about the input, we must assign equal probabilities to the categories. But what if we have some information? In this case, we must choose a distribution with maximum uncertainty from all probability distributions that are consistent with our information. Because entropy is a measure for the uncertainty contained in a distribution, we have to maximize the entropy of the category distribution under the constraints determined by our data. This principle is called the maximum entropy principle.

Let $\{f_k : X \to \mathbb{R}\}_{k=1,\dots,m}$ be a family of m real functions on the discrete random variable $X = (x_1, \dots, x_n)$, and let us assume that the information we have about the input are the expectations F_k of the functions f_k. Then, to get the most uniform distribution that is consistent with our information, we have to find a probability distribution $p = (p_1, \dots, p_n)$ that maximizes the entropy $H(p)$ with respect to the constraints

$$\text{(i)} \quad \sum_{i=1}^{n} p_i = 1 \tag{3.8}$$

$$\text{(ii)} \quad \sum_{i=1}^{n} p_i f_k(x_i) = F_k, \quad k = 1, \dots, m \tag{3.9}$$

We can find a solution to this problem by using the method of Lagrange

multipliers [13]. We define the function L:

$$L(p_1, \ldots, p_n; \lambda_0, \lambda_1, \ldots, \lambda_m) = \\ -\sum_{i=1}^{n} p_i \log p_i - \lambda_0 \left(\sum_{i=1}^{n} p_i - 1 \right) - \sum_{k=1}^{m} \lambda_k \left(\sum_{i=1}^{n} p_i f_k(x_i) - F_k \right) \quad (3.10)$$

where $\lambda_0, \lambda_1, \ldots, \lambda_m$ are the Lagrange parameters. Solving $\partial L / \partial p_i = 0$ gives

$$\frac{\partial L}{\partial p_i} = -\log p_i - 1 - \lambda_0 - \sum_{k=1}^{m} \lambda_k f_k(x_i) = 0 \quad (3.11)$$

$$p_i = \exp\left(-1 - \lambda_0 - \sum_{k=1}^{m} \lambda_k f_k(x_i) \right) \quad (3.12)$$

Now we use constraint (i) to eliminate λ_0 and get the distribution

$$p_i = \frac{\exp\left(-\sum_{k=1}^{m} \lambda_k f_k(x_i)\right)}{Z(\lambda_1, \ldots, \lambda_m)} \quad (3.13)$$

with the partition function

$$Z(\lambda_1, \ldots, \lambda_m) = \sum_{j=1}^{n} \exp\left(-\sum_{k=1}^{m} \lambda_k f_k(x_j) \right) \quad (3.14)$$

In simple cases, the constraints (ii) can be used to eliminate $\lambda_1, \ldots, \lambda_m$, but for most practical applications we have to leave p in parametric form. Probability distributions of the form of equation 3.13 are known as Gibbs distributions, named after J. Willard Gibbs who introduced the maximum entropy principle into the field of statistical mechanics at the end of the 19th century.

We now show that $H(p)$ actually takes its maximum at the distribution given by equation 3.13. The idea of the following argumentation has been taken from [16].

Lemma 3.3 Let $p = (p_1, \ldots, p_n)$ and $q = (q_1, \ldots, q_n)$ be two discrete probability distributions. Then

$$H(p) \leq -\sum_{i=1}^{n} p_i \log q_i \quad (3.15)$$

with equality if and only if $p = q$. The function on the right hand side of equation 3.15 is called the cross entropy of p and q, denoted $H(p \,\|\, q)$.

Proof $\log x \leq x - 1$ for $0 < x < \infty$ with equality if and only if $x = 1$. Therefore, with $x = q_i/p_i$,

$$\sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} \leq \sum_{i=1}^{n} p_i \left(\frac{q_i}{p_i} - 1 \right) = 0 \tag{3.16}$$

$$H(p) \leq -\sum_{i=1}^{n} p_i \log q_i \tag{3.17}$$

with equality if and only if $p = q$. □

Proposition 3.4 Let $X = (x_1, \ldots, x_n)$ be a discrete random variable, and let D be the set of all probability distributions of X that satisfy the constraints

$$\sum_{i=1}^{n} p_i f_k(x_i) = F_k, \quad k = 1, \ldots, m \tag{3.18}$$

Then any distribution $p^* \in D$ for which $H(p) \leq H(p^*)$ holds for all distributions $p \in D$ must have the form

$$p_i^* = \frac{\exp\left(-\sum_{k=1}^{m} \lambda_k f_k(x_i)\right)}{Z(\lambda_1, \ldots, \lambda_m)}, \quad i = 1, \ldots, n \tag{3.19}$$

with

$$Z(\lambda_1, \ldots, \lambda_m) = \sum_{j=1}^{n} \exp\left(-\sum_{k=1}^{m} \lambda_k f_k(x_j) \right) \tag{3.20}$$

Proof From Lemma 3.3 we know that for all probability distributions $p = (p_1, \ldots, p_n)$ and $q = (q_1, \ldots, q_n)$, it holds that $H(p) \leq -\sum_{i=1}^{n} p_i \log q_i$, with equality if and only if $p = q$. Setting $q := p^*$ gives

$$H(p) \leq \sum_{i=1}^{n} p_i \left(\log Z(\lambda_1, \ldots, \lambda_m) + \sum_{k=1}^{m} \lambda_k f_k(x_i) \right) \tag{3.21}$$

which can also be written as

$$H(p) \leq \log Z(\lambda_1, \ldots, \lambda_m) + \sum_{k=1}^{m} \lambda_k F_k \tag{3.22}$$

with equality if and only if $p = p^*$. Consequently, any maximum entropy distribution from D must have the form of p^* in equation 3.19. □

3.6 The binary pattern recognition model

Now we have the tools available to develop our pattern recognition model. We begin with a simplified model where the feature values and the knowledge about the categories which is stored in the system are restricted to binary values. As we have described in section 3.1, our model is based on an interaction of features and categories — some features are evaluated in the input, the observed values modify the probabilities of the categories, which in turn determine the features that are evaluated next. This process continues until the system comes to a conclusion about the input.

Before the system can be used to classify input patterns, it must be fed with information about the classes. This can be done in a supervised or unsupervised learning process where sample input patterns, which are representative of the different categories, are presented to the system. The data that must be extracted from the sample inputs in the learning process are the associations between features and categories, that is, information about the features that typically occur in input patterns of the different categories. It depends, however, on the concrete application what quantities should be stored in the system and how an input pattern is classified into a category. There is no method which is optimal for all situations.

In the binary model, we define the set $\mathcal{P}$ of all possible input patterns of length M as $\mathcal{P} := \{(x_1 \dots x_M) \,|\, x_i \in \{0, 1\}\}$. The elements of an input pattern are the features that are evaluated in the input. The distance $d(p, q)$ between two input patterns $p = (p_1 \dots p_M)$ and $q = (q_1 \dots q_M)$ is the number of features that are different between p and q: $d(p, q) := |\{i \,|\, 1 \leq i \leq M, p_i \neq q_i\}|$. This function is known as the Hamming distance.

The N categories are stored in the system as reference patterns $g_1, \dots, g_N$. How these patterns are created depends on the concrete learning process used to develop the system. We consider an input pattern p correctly classified into a category c, $1 \leq c \leq N$, if the distance between p and g_c is minimal, that is, if $d(p, g_c) \leq d(p, g_i)$ for $i = 1, \dots, N$. Thus, we can say that the k-th element $g_{i,k}$ of the reference pattern g_i "predicts" the value of the feature k in the input, given that the input corresponds to category i.

Let us assume in the following sections of this chapter that exactly one feature is evaluated in each iteration. This means that in iteration n, the pattern recognition system knows the feature values $x_{\alpha_1}, \dots, x_{\alpha_n}$, where α_k is the index of the feature that has been evaluated in iteration k.

3.6.1 The bottom-up path

The function of the bottom-up path in the system is to compute the probability distribution of the categories, based on the observed feature values $x_{\alpha_1}, \dots, x_{\alpha_n}$. Following constructivist ideas, we want to adapt our system's state so that it reconstructs what the system has observed from its environment. This reconstruction is done using the stored reference patterns

$g_1, \ldots, g_N$. The state of the system at a point in time is represented by the probabilities of the categories at that point in time. We consider the observed feature values properly reconstructed when for each evaluated feature α_k, the sum of the predictions $g_{1,\alpha_k}, \ldots, g_{N,\alpha_k}$, weighted by the current probabilities, is equal to the observed feature value x_{α_k}. That means in iteration n, the probability distribution $Y(n) = (y_1, \ldots, y_N)$ must satisfy the constraints[2]

$$\sum_{i=1}^{N} y_i g_{i,\alpha_k} = x_{\alpha_k}, \quad k = 1, \ldots n \tag{3.23}$$

As the predictions g_{i,α_k} can only take the values 0 and 1, for a given k we can define the set of probabilities $Y_k := \{y_i \,|\, 1 \leq i \leq N \text{ and } g_{i,\alpha_k} = 1\}$ and rewrite equation 3.23 as

$$\sum_{y \in Y_k} y = x_{\alpha_k}, \quad k = 1, \ldots n \tag{3.24}$$

If $x_{\alpha_k} = 1$, then the probabilities in Y_k must sum up to 1 and those not in Y_k must all be 0. If $x_{\alpha_k} = 0$, then the probabilities in Y_k must all be 0 and the others must sum up to 1. This means that in the binary model, a category i must be assigned the probability 0 as soon as a feature value x_{α_k} with $x_{\alpha_k} \neq g_{i,\alpha_k}$ is observed. A consequence of this property is that the binary model can only recognize an input pattern if the observed feature values exactly match the feature values of one of the stored categories.

What probabilities must be assigned to the categories that have not been eliminated by the observations? To draw valid conclusions, the probability distribution $Y(n)$ must be based solely on the observed feature values and on further information the system has, but nothing must be interpreted into this data. To achieve this, we must keep as much uncertainty as possible in the category distribution and therefore choose a distribution with maximum entropy. Consequently, if the system has no information beyond the observed feature values, equal probabilities must be assigned to all categories that have not been eliminated.

On the other hand, if the system has additional information, they must of course be reflected by the category distribution. If, for example, the system has learned the long-term probabilities $\bar{y}_1, \ldots, \bar{y}_N$ of the categories (their probabilities without any knowledge about feature values), the categories should be assigned probabilities that are compatible with these long-term probabilities. This means at least that if two categories i and j have both not been eliminated by the observations and $\bar{y}_i < \bar{y}_j$, then the probabilities assigned to these categories should satisfy $y_i(n) < y_j(n)$. When we implement the abstract model in a neural network, we assume for simplicity that all categories are equally probable in the long term.

[2] We omit the index for the number of the iteration where the meaning is clear.

3.6.2 The top-down path

The function of the top-down path is to choose the feature that is evaluated in the next cycle, based on the current probabilities $y_1(n), \ldots, y_N(n)$ of the categories. Of course we want our pattern recognition system to profit maximally from the evaluation of a feature. The profit is the amount of information that the system gains from the observation. Choosing a feature that promises the maximum possible information gain also leads to a maximum expected reduction of the entropy of the category distribution.

In the binary model, only two cases can occur: either the evaluated feature is observed in the input or not. The information content of these two possible events is given by

$$I(x_{\alpha_{n+1}} = 0) = -\log P(x_{\alpha_{n+1}} = 0 \,|\, x_{\alpha_1} \ldots x_{\alpha_n}) \quad \text{and} \tag{3.25}$$

$$I(x_{\alpha_{n+1}} = 1) = -\log P(x_{\alpha_{n+1}} = 1 \,|\, x_{\alpha_1} \ldots x_{\alpha_n}) \tag{3.26}$$

where $P(x_{\alpha_{n+1}} = x^* \,|\, x_{\alpha_1} \ldots x_{\alpha_n})$ is the probability that the feature α_{n+1} evaluated in the next cycle has the value x^*, given that the feature values $x_{\alpha_1} \ldots x_{\alpha_n}$ have been observed in the previous cycles. These equations contain the implicit assumption that the observed feature values are the only information the system has about the input. We set $p^{\,x^*}_{\alpha_{n+1}} := P(x_{\alpha_{n+1}} = x^* \,|\, x_{\alpha_1} \ldots x_{\alpha_n})$. The expected information gain caused by the evaluation of feature α_{n+1} is then

$$\mathrm{E}(I(x_{\alpha_{n+1}})) = -p^{\,0}_{\alpha_{n+1}} \log p^{\,0}_{\alpha_{n+1}} - p^{\,1}_{\alpha_{n+1}} \log p^{\,1}_{\alpha_{n+1}} \tag{3.27}$$

This value is the entropy of the probability distribution $(p^{\,0}_{\alpha_{n+1}}, p^{\,1}_{\alpha_{n+1}})$, and we know that uniform distributions have maximum entropy. To maximize its expected information gain, the system must therefore choose to evaluate a feature α_{n+1} that minimizes $|\, p^{\,1}_{\alpha_{n+1}} - p^{\,0}_{\alpha_{n+1}} \,|$, which means that it must choose a feature for that occurrence and non-occurrence in the input are nearly equally probable.

Unfortunately, the pattern recognition system cannot compute the probability that a given feature occurs in the input. It can, however, make a reasonable estimation of this probability by computing the sum of the predictions for the feature, weighted by the current probabilities of the categories:

$$p^1_{\alpha_{n+1}} \approx \sum_{i=1}^{N} y_i g_{i,\alpha_{n+1}} \tag{3.28}$$

Note that $g_{i,\alpha_k} = g_{i,\alpha_l}$ for $1 \leq k, l \leq n$ for all categories i with $y_i > 0$. That means, all categories with probabilities greater than 0 make the same predictions for all features evaluated up to iteration n. Using the estimation in equation 3.28, the system must choose a feature α_{n+1} that minimizes $\left| \sum_{i=1}^{N} y_i g_{i,\alpha_{n+1}} - 0.5 \right|$ to maximize its expected information gain. This is a

feature with minimal difference between the number of active categories that predict the feature and the number of active categories that do not predict the feature. In this sense, the feature α_{n+1} is chosen to be as independent as possible from the already evaluated features $\alpha_1, \ldots, \alpha_n$.

What behavior can we expect from this model? Under the assumption that in the long term all categories are equally probable, so that $\bar{y}_1 = \cdots = \bar{y}_N$, about half of the remaining categories can be eliminated in every cycle, provided that a feature can be found for which $| p^{\,0}_{\alpha_{n+1}} - p^{\,1}_{\alpha_{n+1}} |$ is small enough. This is the case if for every subset of categories, there is a feature that occurs in about half of the categories in this subset and not in the others. In this ideal situation, the binary model behaves similar to a binary search algorithm. The number of iterations required to eliminate all but one of the N categories is then approximately $\log_2 N$.

3.7 The generalized model

In this section, we extend the domain of the feature values and the reference patterns to real values from the interval $[0, 1]$. Most of the simplifications we used in the preceding section are no longer applicable, and we must therefore try to find a more general model.

As in the binary model, we assume that inputs are classified using a suitable distance function. One possible definition of the distance $d(p, q)$ of two input patterns $p = (p_1, \ldots, p_M)$ and $q = (q_1, \ldots, q_M)$ is the Euclidean distance $d(p, q) := \sum_{i=1}^{M} (p_i - q_i)^2$, but other functions are also possible. The distance function that is used to classify input patterns implicitly defines the meaning of the predictions: they partition the input space so that a partition contains all inputs that are classified into the same category. For our purpose, it is not necessary to further interpret these values.

3.7.1 The bottom-up path

As in the binary model, we assume that to reconstruct the observed feature values, the category distribution must satisfy the constraints

$$\sum_{i=1}^{N} y_i g_{i,\alpha_k} = x_{\alpha_k}, \quad k = 1, \ldots, n \tag{3.29}$$

It is important to note that these equations are only one possibility to express the abstract concept of reconstruction by a set of constraints. They correspond to the idea that a reconstruction is a linear combination of stored concepts. The concepts are represented by the predictions $g_{i,k}$, and they are weighted by the probabilities y_i of the individual categories.

In contrast to the binary model, we can no longer simply set the probability of a category to 0 as soon as one of its predictions deviates from an

observed feature value. As we want to keep as much uncertainty as possible in the category distribution, we must maximize its entropy with respect to the constraints in equation 3.29. We have derived in section 3.5 that this leads to the Gibbs distribution

$$y_i = \frac{\exp\left(-\sum_{k=1}^{n} \lambda_k g_{i,\alpha_k}\right)}{Z(\lambda_1, \ldots, \lambda_n)} \tag{3.30}$$

with the partition function

$$Z(\lambda_1, \ldots, \lambda_n) = \sum_{j=1}^{N} \exp\left(-\sum_{k=1}^{n} \lambda_k g_{j,\alpha_k}\right) \tag{3.31}$$

As long as the parameters $\lambda_1, \ldots, \lambda_n$ are finite values, the probabilities of the categories are always greater than zero. It is not always possible to find parameters $\lambda_1, \ldots, \lambda_n$ that satisfy the constraints in the equations 3.29. The solvability of these equations depends on the stored reference patterns. To be able to reconstruct an observed feature value x_{α_k}, it must lie inside the convex closure of the predictions $g_{1,\alpha_k}, \ldots, g_{N,\alpha_k}$. This restriction expresses the natural constructive requirement that a system can only reconstruct what can be expressed in terms of its own experience.

In the following, we denote the entropy of the maximum entropy distribution by $S(x) = S(x_{\alpha_1}, \ldots, x_{\alpha_n})$ and the partition function by $Z(\lambda) = Z(\lambda_1, \ldots, \lambda_n)$. We have seen in the proof of proposition 3.4 that we can write the entropy $S(x)$ as

$$S(x) = \log Z(\lambda) + \sum_{k=1}^{n} \lambda_k x_{\alpha_k} \tag{3.32}$$

This shows that the entropy $S(x)$ and the function $\log Z(\lambda)$, which is called the free energy in some contexts, are closely related. Equation 3.32 is actually a Legendre transformation [29], which transforms between the dual coordinate systems $x_{\alpha_1}, \ldots, x_{\alpha_n}$ and $\lambda_1, \ldots, \lambda_n$. Using the constraints in equation 3.29, we can find an implicit equation for the observed feature values $x_{\alpha_1}, \ldots, x_{\alpha_n}$:

$$\begin{aligned} x_{\alpha_k} &= \sum_{i=1}^{N} \frac{\exp\left(-\sum_{k=1}^{n} \lambda_k g_{i,\alpha_k}\right)}{Z(\lambda)} g_{i,\alpha_k} \\ &= -\frac{1}{Z(\lambda)} \frac{\partial Z(\lambda)}{\partial \lambda_k} \\ &= -\frac{\partial \log Z(\lambda)}{\partial \lambda_k} \end{aligned} \tag{3.33}$$

Similarly, by differentiating $S(x)$ with respect to the feature values, we get

$$\begin{aligned}\frac{\partial S(x)}{\partial x_{\alpha_k}} &= \sum_{l=1}^{n} \frac{\partial \log Z(\lambda)}{\partial \lambda_l} \frac{\partial \lambda_l}{\partial x_{\alpha_k}} + \sum_{l=1}^{n} \frac{\partial(\lambda_l x_{\alpha_l})}{\partial x_{\alpha_k}} \\ &= -\sum_{l=1}^{n} \frac{\partial \lambda_l}{\partial x_{\alpha_k}} x_{\alpha_l} + \sum_{l=1}^{n} \frac{\partial \lambda_l}{\partial x_{\alpha_k}} x_{\alpha_l} + \sum_{l=1}^{n} \frac{\partial x_{\alpha_l}}{\partial x_{\alpha_k}} \lambda_l \\ &= \sum_{l=1}^{n} \frac{\partial x_{\alpha_l}}{\partial x_{\alpha_k}} \lambda_l\end{aligned} \tag{3.34}$$

Under the assumption that the values of the evaluated features are mutually independent (which is desirable, but will not be realizable in most applications), so that $\partial x_{\alpha_l} / \partial x_{\alpha_k} = \delta_{\alpha_l \alpha_k}$, the parameters λ_k can be expressed as

$$\lambda_k = \frac{\partial S(x)}{\partial x_{\alpha_k}} \tag{3.35}$$

Equation 3.35 gives us an interpretation for the parameters $\lambda_1, \ldots, \lambda_n$: λ_k determines how much the maximum entropy $S(x)$ changes due to a change of the feature value x_{α_k}.

3.7.2 The top-down path

In the generalized model, the expected information gain $\mathrm{E}(I(x_{\alpha_{n+1}}))$ caused by the evaluation of the feature α_{n+1} becomes

$$\mathrm{E}(I(x_{\alpha_{n+1}})) = \int_0^1 p_{\alpha_{n+1}}(x) \log p_{\alpha_{n+1}}(x) \, \mathrm{d}x \tag{3.36}$$

where $p_{\alpha_{n+1}}$ is the probability density function for the values of the feature α_{n+1}, given the feature values $x_{\alpha_1}, \ldots, x_{\alpha_n}$ that have been observed in the previous cycles. As in the binary model, $\mathrm{E}(I(x_{\alpha_{n+1}}))$ reaches its maximum at a uniform distribution $p_{\alpha_{n+1}}$. The problem is, however, that there is no way to compute the probabilities $p_{\alpha_{n+1}}(x)$ of the feature values from the stored predictions $g_{i,k}$ without making further assumptions. It might be possible to get more analytical results by making assumptions about the distribution underlying the predictions. Such an approach, however, would be rather arbitrary, as long as there is no real justification to favor one specific distribution.

A heuristic solution is to choose a feature k that maximizes the variance $\mathrm{E}((g_{.,k} - \bar{g}_{.,k})^2)$ of the predictions $g_{.,k}$ with $\bar{g}_{.,k} = \mathrm{E}(g_{.,k}) = \sum_{i=1}^{N} y_i g_{i,k}$:

$$\mathrm{E}((g_{.,k} - \bar{g}_{.,k})^2) = \sum_{i=1}^{N} y_i (g_{i,k} - \bar{g}_{.,k})^2 \to \max \tag{3.37}$$

This heuristic should at least work well if the predictions $g_{i,k}$ for the feature

values take only values from the set $[0, \varepsilon] \cup [1 - \varepsilon, 1]$ for a sufficiently small constant ε.

3.8 Some Remarks

Obviously, the generalization of predictions and features to real values leads to problems. In contrast to the binary model, we are not able to find an explicit expression for the category distribution and the features that should be evaluated. One reason is that for such a general model the parameters $g_{i,k}$ are not sufficient to fully describe a category.

Because of these problems, we concentrate our further work on the binary model. We have described in section 3.6.1 that this model only works for inputs where the evaluated features exactly match the feature values of one of the stored categories. This means that a pattern recognition system that directly implements the binary model cannot compensate for noisy inputs or errors in the recognition process — features that are required for most practical applications. When we have developed a neural network implementation of the binary model, we shall therefore try to extend it to overcome these limitations. To be precise, the binary model already has some insensitiveness to noise in the input: errors in the features that are not evaluated to recognize an input (which can be the greatest part of the features) are completely ignored. Nevertheless, we shall try to build a system that can also compensate for errors in the evaluated features.

The model we have developed in this chapter has an important property that distinguishes it from many other recurrent neural network models that have been developed for pattern recognition tasks, including the associative networks described and analyzed in Chapter 12 of [31] and the bidirectional associative memories (also known as resonance networks) [18]. Using our terminology, in these models the active categories activate (through the top-down connections) the features they predict, which are the features that typically occur in inputs corresponding to them. If we applied this strategy to our model, the feature α_{n+1} evaluated in iteration $n + 1$ would be chosen so that the weighted sum $\sum_{i=1}^{N} y_i g_{i,\alpha_{n+1}}$ is maximized. The information gain that can be expected from the evaluation of such a feature cannot be higher than the one that can be expected by applying our strategy (as our strategy *is* maximizing the expected information gain), and in most cases it will be lower. These models therefore make only suboptimal use of the information that is contained in the input.

Chapter 4

Neural network implementation

How should the neural network look like that implements the abstract model we have developed in the preceding chapter? Before we come to this question, we take another look at the anatomy of the natural neural networks that solve pattern recognition tasks in the cortex. We should not expect that we can directly derive a blueprint for our network implementation from this. But the remarkably simple structure of these powerful networks will admonish us that we should not let our own implementation become too complicated.

4.1 Further neuroanatomy

In Chapter 2, we have introduced some neuroanatomical knowledge on the level of larger brain structures and their connectivity. In this section, we scale our view by an order of magnitude and look at the specific neuron types in the cortex and how the neurons are connected. Our hope is that this can provide some inspiration for the neural network implementation of the abstract model we have derived in Chapter 3. It is important to note that the information given in this section have been simplified much and leave out many neuroanatomical and functional details that are not important for our purpose.

A neuron cell consists of the following externally visible components: the cell body, called the soma, the dendrites, and the axon. The dendrites are thin strings that form a widely branched tree-like structure. Located on the surface of the dendrites are the synapses. These are the input sites of the neurons where axon endings of other neurons can attach to. The dendrites transport the input signals from the synapses to the cell body. Recent findings, however, provide evidence that the dendrites may also be important for the computational function of the neurons [4, 24, 25, 38]. Only the widely branched structure of the dendrites makes it possible that a single neuron can receive input from thousands of other neurons.

If two neurons are connected through a synapse, the neuron that provides the input is called the presynaptic neuron, and the neuron that receives this input is called the postsynaptic neuron. There are two kinds of synapses: excitatory and inhibitory. Input from excitatory synapses increases the activation of the postsynaptic neuron, and input from inhibitory synapses decreases it.

Whether a synapse is excitatory or inhibitory is determined by the neurotransmitter (the substance used to transmit the signal over the synapse) that the presynaptic neuron produces. For most types of neurons it is assumed that they form either excitatory or inhibitory synapses with postsynaptic neurons, so that neuron types can be divided into excitatory and inhibitory types.

The axon is a thin string, usually much longer than the dendrites, with branches at its end segment. It transmits the output signals of the neuron from the cell body to the synapses that it forms with other neurons at the endings of its axon branches. There are two types of axons, myelinated and unmyelinated, which differ in the way they transport neuroelectrical signals. The myelinated axons are covered by a sheath of myelin, which is composed of layers of Glia cell membranes wrapped around the axon. In regular intervals, the myelin sheath is interrupted, resulting in gaps called the Ranvier nodes. Neuroelectrical signal travel much faster along myelinated axons than along unmyelinated axons. The higher signal speed is especially important where information must be transmitted over greater distances. For this reason, neurons that connect distant regions of the brain, brain regions to muscles, or sensory organs to brain regions, all have myelinated axons.

4.1.1 Types of neurons in the cortex

There are two main types of neurons in the cortex: pyramidal cells and interneurons, which can be further divided into subtypes. Besides the neurons, a number of other cell types can be found in the brain. These cells, which are summarized under the name Glia cells, provide the infrastructure for the neurons. For example, they take care of the nutrition of the neurons, they build the myelin sheath around the neuron's axons, and they form a skeleton that props the neural structures.

Pyramidal cells are comparatively large neurons of the excitatory kind. Their dendrites are separated into a basal and an apical group. The basal dendrites spread out laterally and form connections in the same cortical layer as the neuron or in adjacent layers; the apical dendrites rise to the superficial layers and form connections there. Apical and basal dendrites together form a pyramidal shape, which gave these neurons their name. A single pyramidal cell can have a few thousand input synapses, which are located on the dendrites on small bulges called spines. The output of the pyramidal cells is transmitted locally through axon collaterals and to more distant locations through a long myelinated axon.[1]The myelin sheath around the axon makes it possible that pyramidal cells can transmit their output over a distance of a few centimeters in the cortex, while unmyelinated axons are usually only a fraction of a millimeter long. Most of the connections between cortical areas and between the cortex and subcortical structures are therefore made up of

[1]There is also a subtype called small pyramidal cells that has no long myelinated axon. The number of small pyramidal cells in the cortex, however, is much smaller than the number of "real" pyramidal cells.

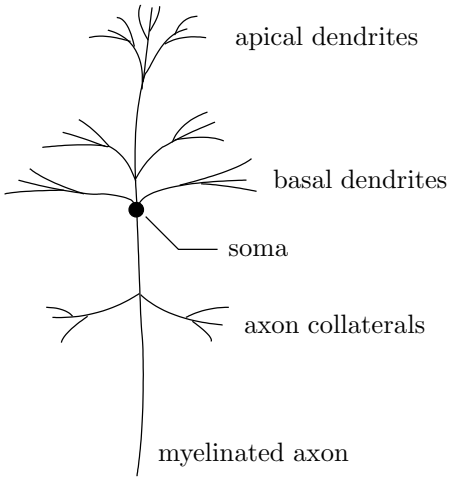


Figure 4.1: A cortical pyramidal cell

long pyramidal cell axons. Figure 4.1 shows a schematic drawing of a cortical pyramidal cell (note that the axon extends further down beyond the border of the drawing).

The interneurons are smaller cells of the inhibitory kind. They have no spines on their dendrites, so that their input synapses directly attach to the dendritic surface. Their axons are much shorter than those of the pyramidal cells and lack a myelin sheath; it is not required because they transmit their output only locally. Figure 4.2 shows a schematic drawing of a cortical interneuron.

It has been estimated that the excitatory pyramidal cells make up 60%–

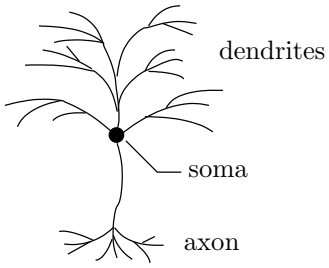


Figure 4.2: A cortical interneuron

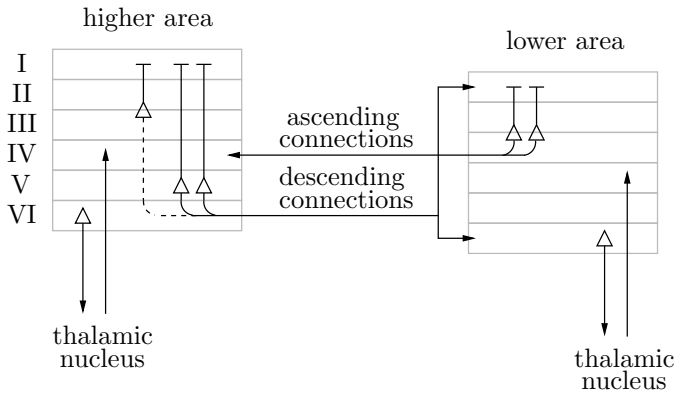


Figure 4.3: Connections between two cortical areas

80% of all neurons in the cortex of mammalian brains [27]. This is especially remarkable because in other brain structures, for example in the cerebellum, there are usually much more inhibitory interneurons than excitatory cells. That most of the neurons in the cortex are pyramidal cells means that the majority of the cortical neurons has (through the long axon) connections to neurons in other cortical areas. This supports the theory that the cortex works in a very distributed manner.

4.1.2 Connections between cortical areas

The cortical areas can be ordered into lower areas, which are concerned with more concrete sensory or motor concepts, and higher areas, which are concerned with more abstract concepts. The relative functional level of two connected cortical areas is also reflected in the structure of their connections. In Chapter 2, we have already discussed Mumford's theory of the cortical function. In one of his articles [27], he describes the following three pathways between two connected areas A and B where area A has a higher functional level than area B:

- The ascending pathway from area B to area A: Superficial pyramidal cells in the layers II and III of area B send connections up to layer IV of area A. As the incoming connections from the thalamus also terminate in layer IV, it can generally be considered the layer where the input from lower structures arrives. For this reason, layer IV is called the standard input layer. In Mumford's theory, the function of the ascending pathway is to transmit the difference, called the residual, between the data coming from the higher area A and the real input that arrives in area B from lower structures.

- The standard descending pathway from area A to area B: Deep pyramidal cells in layer V of area A send connections down to the layers I and VI of area B. The deep pyramidal cells, mainly in layer VI, are also the point of origin of the area's connections to the thalamus, so that the layers V and VI can generally be considered the layers where the output to lower structures originates. The function of the descending pathway is to transmit a reconstruction of area B's input data, which is based on the state of area A. Mumford calls these reconstructions templates; in our terminology, they would be called predictions.

- The extra descending pathway from area A to area B: If the functional level of area A is only slightly higher than that of area B, then the superficial pyramidal cells in the layers II and III of area A also send connections down to the layers I and VI of area B. The function of this pathway is to transmit a residual that area A sends to an even higher area C also to the lower area B.

Figure 4.3 illustrates the connections between two cortical areas, as they are described in the cited article. The small triangles show the pyramidal cells, the arrows show their long axons, and the T-shaped lines going from the top of the triangles upward show their apical dendrites that terminate in layer I.

In [30], Raizada and Grossberg describe a more detailed model of the neural circuits in the thalamic nucleus LGN and the visual areas V1 and V2 of primates, which is based on the results of a large number of studies on macaque monkeys. The authors propose that the processing of visual information in these regions is based on small subcircuits, each with a well-defined functional role. Two of these subcircuits, which can possibly be useful for our model, are illustrated in the figures 4.4 and 4.5. The white-filled circles show excitatory neurons, the black-filled circles show inhibitory interneurons, and the lines with the T-shaped endings show the neuron's axons. The subcircuits are:

- On-center, off-surround filter: Input from lower structures arrives in layer IV along two routes. First, there is a strong direct connection from the lower structure to the layer IV neurons. Second, there are connections to layer VI neurons that transmit the signals up to layer IV through a circuit with on-center, off-surround characteristic, which means that at every input location the central signals are enhanced at the expense of peripheral signals. The on-center, off-surround circuit has a contrast and edge enhancing effect on the input. Because the direct connection is stronger than the indirect one, however, the filter effect is only modulatory and the direct input is determinant. Figure 4.4 shows the on-center, off-surround circuit for input coming from the LGN to the visual area V1.

- Folded feedback: Feedback coming from higher structures does not directly activate the input neurons in layer IV. Instead, connections from

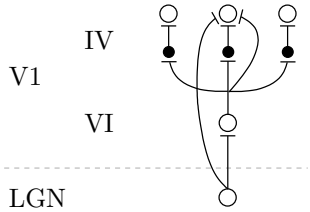


Figure 4.4: On-center, off-surround circuit

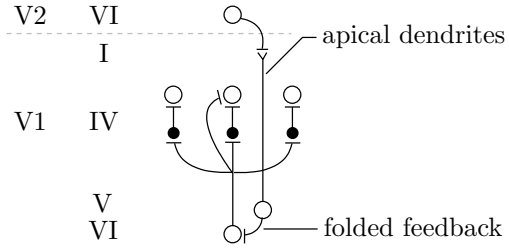


Figure 4.5: Folded-feedback circuit

the deep layers of the higher structure synapse in layer I on the apical dendrites of layer V pyramidal cells. These cells feed their input into the on-center, off-surround path from layer VI to layer IV. This indirect "folded" feedback has two important effects. First, because of the filter effect of the on-center, off-surround path, the feedback enhances the activity of neurons that support it at the expense of neighboring neurons. Second, because the on-center, off-surround path is only modulatory, feedback can only support activation caused by input, but cannot lead to activation without input. This prevents the feedback from only confirming itself, which would lead to hallucinations in the system. Figure 4.5 shows the folded feedback circuit for feedback coming from the higher area V2 to the lower area V1.

In section 4.3.1, we shall see that the folded feedback subcircuit is realized in the part of our network that selects the features to be evaluated. The on-center, off-surround subcircuit is at least compatible with the part of our network that activates the categories, which is described in subsection 4.3.2.

4.2 The neuron model

Numerous models of neurons and networks of neurons have been described in the neural network literature, ranging from networks of simple binary thresh-

old gates studied first by Warren McCulloch and Walter Pitts [23] to the bioelectrical model for the generation of action potentials in the giant nerve fibers of squids by Alan Hodgkin and Andrew Huxley [14], who received the Nobel Prize in Physiology or Medicine for this achievement in 1963, together with John Eccles. The neuron model that we use in this work is very abstract and simple, compared to the complex dynamic properties of real neurons that are captured by some more biological models (a good reference for these models is Part II of [8]).

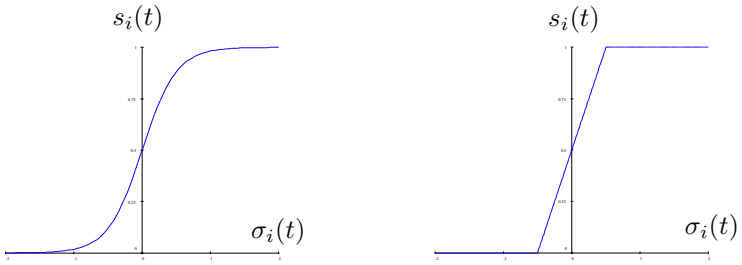
We define a neural network $\mathcal{N}$ as a set of interconnected computational units, called neurons, where the state of each unit at a given point in time depends on the state of the network at earlier points in time. Depending on the state of the network or other quantities, some of the network parameters can change over time. This can be used, for example, to adapt the strengths of the connections between the units or the sensitivity of the neurons to input signals. Some researchers explicitly avoid the term neuron for the computational units because they want to emphasize that the behavior of natural neurons is much more complex than that of the modeled units [33]. Nevertheless, we use the term neuron in this work, as it should be clear from the context what it refers to.

In our model, the state of a neuron corresponds to the neuroelectrical activity of natural neurons. In response to input signals, natural neurons generate local changes in the electrical potential of their cell membranes. These local changes, which are called spikes or action potentials, move along the neurons' axons to the synapses. When an action potential reaches a synapse, it triggers the release of a neurotransmitter that transmits the signal to the postsynaptic cell. Action potentials are the only means by which neurons can communicate over greater distances. It is known today that neurons do not encode their output solely in the frequency with which they are generating action potentials, but also in their precise timing. To keep our model simple, however, we ignore the timing of single action potentials and interpret the state of a neuron in our model as the spike-count rate of the neuron, which is the number of action potentials generated in a time interval of fixed length divided by the length of that interval.

The input to a neuron is computed from the states of the neurons in the network from which the neuron receives input. This includes the possibility that a neuron receives input from itself through self-recurrent connections. The input $\sigma_i(t)$ to neuron i at time t is defined as

$$\sigma_i(t) := \sum_{j=1}^{|\mathcal{N}|} w_{ij}(t) s_j(t - \Delta t_{ij}) \tag{4.1}$$

$|\mathcal{N}|$ is the number of neurons in the network. $s_j(t)$ is the state of the neuron j at time t. The weight $w_{ij}(t)$ determines the strength of the connection from neuron j to neuron i at time t. A positive weight means that the connection



(a) sigmoid (b) piecewise linear

Figure 4.6: Neuron state functions

is excitatory; a negative weight means that it is inhibitory. If there is no connection from neuron j to neuron i, the weight w_{ij} is set to 0. The delay Δt_{ij} is the time that the signals need to travel along the connection from neuron j to neuron i; usually, we set $\Delta t_{ij} := 1$ for all connections. In this work, we use only discrete time values. External input to the network is provided by special input neurons, which have a fixed state and do not receive input from other neurons.

The state of a neuron, as we understand it, depends on the input in the following manner: If the sum of the input signals is below a threshold value, the neuron produces no or only a very small output, which is called the resting activity. Each neuron can have a different threshold value. If the input is above this threshold, the activity of the neuron increases monotonic with the input until a maximum activity is reached. If the input is above this saturation value, the activity of the neuron remains at the maximum level. This behavior can be modeled by using the classical sigmoid state function, which is applied in many application-oriented neural network models (see for example [31], Chapter 7); it is defined as

$$s_i(t) := \frac{1}{1 + \exp(-\beta\,(\sigma_i(t) - b_i(t)))} \tag{4.2}$$

$b_i(t)$ is the threshold value (also called the bias) of neuron i at time t; it defines the input value where the neuron's state is exactly 0.5. $\sigma_i(t)$ is the input to the neuron at time t, as defined in equation 4.1. The parameter β controls the slope of s_i at $\sigma_i(t) = 0$. Figure 4.6 (a) shows a plot of the sigmoid state function for $\beta = 1$ and $b_i(t) = 0$.

Depending on the value of the parameter β, two interpretations of the sigmoid state function in equation 4.2 are possible. If β is large ($\beta \gg 1$), then s_i has a large slope at $\sigma_i(t) = 0$, and the state function can be interpreted as a continuous version of the step function. In this interpretation, a neuron is active if its input exceeds its threshold value, and inactive otherwise. If β is

small, then the state function can be interpreted as a continuous version of a piecewise linear function of the neuron input, whose value is restricted to the interval $[0, 1]$.

When doing computer simulations, the use of a sigmoid state function turns out to be problematic in some situations. For example, a neuron that receives connections from many other neurons can be activated although all neurons from which it receives input have only a very small activation. This is a consequence of the property that the value of a sigmoid state function is always greater than 0. Such side effects can sometimes hide the relevant effects that we want to capture in a simulation. To avoid this, we shall sometimes directly use step functions or piecewise linear functions to compute the state of the neurons in our computer simulations. Figure 4.6 (b) shows a plot of the piecewise linear state function with threshold value 0 and slope 1.

4.3 Implementation of the binary model

In the following sections, we build the components of a neural network that is based on the theoretical foundation we have developed in Chapter 3. It is not so crucial that the neural network implements every detail of the theoretical model. Much more important is that the neural network shows the same qualitative behavior as the theoretical model without becoming too complicated. The natural neural circuits have a remarkably simple structure (at least when we consider their power and the huge number of neurons they are composed of), and we should attach importance to keep this property in our network. We begin with the implementation of the binary model (see section 3.6) where the feature values and the predictions can take only the values 0 and 1. We assume that the parameter β in the neuron model is large, so that the state function is close to a step function. This means that at a given point in time, a neuron can be either active or inactive.

In the description of the subcircuits of the network, we give conditions that the parameters of the circuit must satisfy. Some of the conditions could be weakened by introducing additional parameters and by considering special cases. For example, in the category activation circuit (subsection 4.3.2), each category neuron could have its own threshold value. When the parameters of the network are developed in a learning process, such a differentiation should happen automatically. When the network is constructed, however, it is desirable to keep the number of parameters small.

4.3.1 Feature evaluation

The first module we develop for our neural network is the subcircuit for the evaluation of the features. This subcircuit has actually two functions: evaluating the features and storing their values. When a feature is selected for evaluation, which is signaled by the activation of a select neuron, the circuit

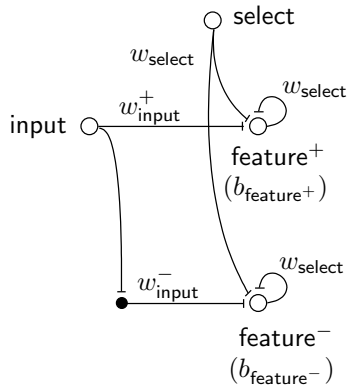


Figure 4.7: Feature evaluation circuit

must read the corresponding feature value. Each feature can have one or more assigned select neurons, which are activated by a feature selection subcircuit (see section 4.3.3). Once a feature has been evaluated, its value must be stored, so that it can be used in the following cycles without being read again. In natural networks, the feature values are coming from sensory organs or from other regions of the nervous system; in our network, we use special input neurons to provide them.

Figure 4.7 shows the feature evaluation circuit (only one select neuron is shown). The network contains this subcircuit once for every feature that can be evaluated. In the figure, the white-filled circles show excitatory neurons, the black-filled circles show inhibitory neurons, and the lines with the T-shaped endings show the neuron's axons. The variable names inside parentheses beneath the neuron names are the threshold values of the corresponding neurons. The variable names at the connections are the strengths of these connections. The default threshold value is 0; the default connection strength is 1 for excitatory connections and -1 for inhibitory connections.

In the feature evaluation circuit, we find again the indirect, "folded" feedback we have described in subsection 4.1.2: Feedback from higher structures does not directly activate the feature neurons. Instead, the feedback activates special select neurons, which then trigger the evaluation of the selected features. This prevents the categories from confirming themselves through their own feedback.

To function properly, the parameters of the feature evaluation circuit must satisfy the following conditions:

- The feature^{+} neuron is activated if the feature has been selected (at least one associated select neuron is active) and the feature has been detected

in the input (the corresponding input neuron is active):

$$b_{\text{feature}^+} < w^+_{\text{input}} + w_{\text{select}} \tag{4.3}$$

- The feature^+ neuron cannot be activated if the feature has not been selected or the feature has not been detected in the input:

$$w^+_{\text{input}} < b_{\text{feature}^+} \tag{4.4}$$

$$n_{\text{select}} w_{\text{select}} < b_{\text{feature}^+} \tag{4.5}$$

  n_{select} is the maximum number of select neurons assigned to this feature that can be active at the same time.

- The feature^- neuron is activated if the feature has been selected and the feature has not been detected in the input:

$$b_{\text{feature}^-} < w_{\text{select}} \tag{4.6}$$

- The feature^- neuron cannot be activated if the feature has been detected in the input:

$$w^-_{\text{input}} + n_{\text{select}} w_{\text{select}} < b_{\text{feature}^-} \tag{4.7}$$

It is clear that either the feature^+ neuron or the feature^- neuron in the subcircuit can be active, but not both. Once a feature^+ neuron or a feature^- neuron has been activated, the self-recurrent connections with weight w_{select} guarantee that it remains active.

The circuit in figure 4.7 detects both, the occurrence and the non-occurrence of a feature in the input. Occurrence is signaled by an activation of the feature^+ neuron; non-occurrence is signaled by an activation of the feature^- neuron. It is unlikely that in natural circuits the non-occurrence is detected for all features because only a small part of all features can occur in the input at one point in time. Natural circuits, however, can compensate for this loss of information by the tremendously large number of different features they can detect.

After a feature has been evaluated, its value is stored in the self-recurrent connections of the feature neurons. In natural circuits, feature values may be stored using a different mechanism. Neurons that have a very slow activation decay are, for example, a conceivable alternative.

4.3.2 Category activation

The subcircuit for the activation of the categories must activate those categories whose predictions match all evaluated feature values and deactivate all other categories. In our model, each category is represented by a single category neuron. If a feature k is predicted by a category i, that is, $g_{i,k} = 1$,

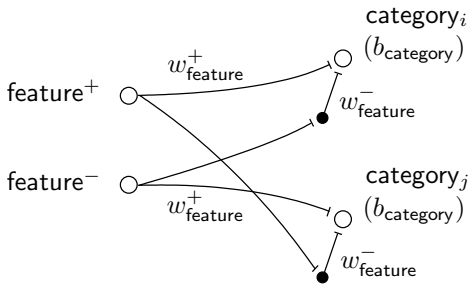


Figure 4.8: Category activation circuit

then there are an excitatory connection from the feature$^+$ neuron k to the category neuron i and an inhibitory connection from the feature$^-$ neuron k to the category neuron i. If a feature k is not predicted by a category i, that is, $g_{i,k} = 0$, then there are an inhibitory connection from the feature$^+$ neuron k to the category neuron i and an excitatory connection from the feature$^-$ neuron k to the category neuron i. Figure 4.8 shows the category activation circuit for one feature neuron and two category neurons i and j; category i predicts the feature, and category j does not predict the feature ($g_{i,k} = 1$ and $g_{j,k} = 0$).

The parameters of the category activation circuit must satisfy the following conditions:

- A category neuron is activated if and only if it receives at least one excitatory input and no inhibitory input:

$$w^-_{\text{feature}} + (n_{\text{feature}} - 1)\, w^+_{\text{feature}} < b_{\text{category}} < w^+_{\text{feature}} \tag{4.8}$$

  n_{feature} is the maximum number of different features that can occur in the input at the same time.

Thus, a single matching feature in the input can be sufficient to activate a category. On the other hand, a single non-matching feature is sufficient to prevent the activation of a category.

In subsection 4.1.2, we have described that the bottom-up connections in the cortical areas V1 and V2 work as on-center, off-surround filters, which have a contrast and edge enhancing effect on the visual input. Although this effect is specific to visual signals, similar filters are possibly applied to the data processed in other cortical regions. The category activation circuit in our network model is at least compatible with this neuroanatomical finding. If we assume that a feature neuron is not connected to all category neurons, but only to a small, locally constrained subset of neurons, we get structures in our network that are similar to the filter structures in V1 and V2. Such locality constraints are biologically very plausible and can even been applied

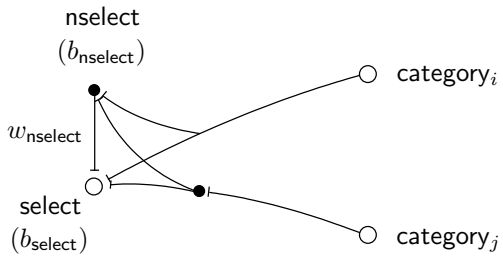


Figure 4.9: Feature selection circuit, implementation 1

to build network structures. The proposition that the connections between feature neurons and category neurons in our model correspond to the filter structures observed in the cortex is, however, very speculative.

4.3.3 Feature selection

The feature selection subcircuit selects the features that are evaluated in the next iteration. To select those features that maximally reduce the uncertainty in the category distribution, this subcircuit must estimate the information gain that can be expected from the evaluation of each feature. In contrast to the abstract model where we assumed for simplicity that exactly one feature is evaluated in every iteration, our feature selection circuit should select all features whose expected information gain is greater than a certain threshold value. The reason for this deviation from the abstract model is that selecting exactly one feature in every iteration is difficult to realize in a neural network; besides this, it is hard to believe that the features are evaluated in a strict sequential manner in natural neural networks.

We give two alternative implementations for this subcircuit. The first one is directly derived from the abstract model. It selects all features for which the difference between the number of active categories that predict the feature and the number of active categories that do not predict the feature is less than a certain constant. Figure 4.9 shows the first implementation for two category neurons i and j and a single select neuron; category i predicts the feature, and category j does not predict the feature. In this implementation, each feature neuron has exactly one assigned select neuron. If a category i predicts a feature k, that is, $g_{i,k} = 1$, then there are excitatory connections from the category neuron i to the nselect neuron k and from the category neuron i to the select neuron k. If a category i does not predict a feature k, that is, $g_{i,k} = 0$, then there are inhibitory connections from the category neuron i to the nselect neuron k and from the category neuron i to the select neuron k. All these connections have the same strength.

The parameters of the first implementation of the feature selection circuit

must satisfy the following condition:

- Let d be the difference between the number of active categories that predict the corresponding feature and the number of active categories that do not predict the corresponding feature. Then, the select neuron is activated if and only if $d_{\min} < d < d_{\max}$:

$$b_{\mathsf{select}} = d_{\mathsf{min}} \tag{4.9}$$

$$b_{\mathsf{nselect}} = d_{\mathsf{max}} \tag{4.10}$$

$$w_{\mathsf{nselect}} + b_{\mathsf{nselect}} < b_{\mathsf{select}} \tag{4.11}$$

The problem with the first implementation of the feature selection circuit is that the threshold values b_{select} and b_{nselect} are constant during the recognition process. If the interval $(b_{\mathsf{select}}, b_{\mathsf{nselect}})$ is too small, the difference between the number of active categories that predict a feature and the number of active categories that do not predict a feature can get outside this interval for all features. When this happens, no new feature is selected although there can still be more than one active category. On the other hand, if the interval is too large, too many features are evaluated when the number of active categories decreases. The situation could be improved by adapting the range that is defined by the threshold values b_{select} and b_{nselect} depending on which categories are active. This, however, would require a neural circuit which is much more complicated.

The second implementation, which is shown in figure 4.10, does not have this problem. In this implementation, each select neuron has an assigned category subset, from which it receives excitatory connections. The select neuron is activated when all categories in this subset are active. In contrast to the first implementation, there is no one-to-one correspondence between features and select neurons. Instead, each select neuron selects the feature whose evaluation contains the greatest amount of information under the assumption that the categories in the assigned category subset are active. This is a feature for which the difference between the number of categories in the subset that predict the feature and the number of categories in the subset that do not predict the feature is minimal. Thus, a feature can have zero, one, or more assigned select neurons.

The parameters of the second implementation of the feature selection circuit must satisfy the following condition:

- A select neuron is activated if and only if all category neurons from which it receives input are active:

$$n_{\mathsf{category},k} - 1 < b_{\mathsf{select},k} < n_{\mathsf{category},k} \tag{4.12}$$

  $n_{\mathsf{category},k}$ is the number of category neurons in the category subset assigned to the select neuron k.

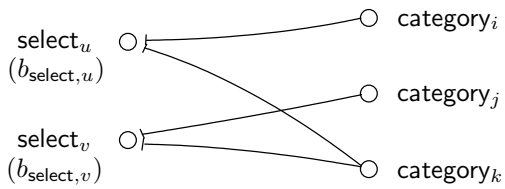


Figure 4.10: Feature selection circuit, implementation 2

A question that arises when using this implementation is how to choose the category subsets that are assigned to the select neurons. As the number of subsets grows exponentially with the number of categories, it is certainly not a good idea to have a select neuron for every possible subset. A better approach is to assign one select neuron to each pair of categories, resulting in $N(N-1)/2$ select neurons for N categories. If this approach is applied, the resulting network needs only 2 iterations to recognize any input pattern that corresponds to one of the stored categories. The reason is that for each pair of categories that are active after the first iteration, a feature which separates this pair is selected and evaluated in the second iteration. As this is done for all pairs of active categories, at most one category can be active in the network after the second iteration.

4.3.4 Initial activation and synchronization

The network we have described so far requires that some category neurons are already active, so that features can be selected for evaluation. There are different possibilities to create an initial activation of the category neurons. When the network has performed a pattern recognition task before and a new input is presented to it, the simplest solution is to use the category activation from the previous task. This requires that there are enough category neurons active to activate some select neurons. In larger, more realistic networks this will normally be the case. Another possibility is to explicitly activate some select neurons. These neurons should select the features with the highest expected information gains resulting from their evaluation, averaged over the categories. The activation of these select neurons can be triggered by an external signal or by a special category neuron that represents the "unknown" category.

Although in principle the neurons in natural neural networks can all work in parallel, cortical areas have been found where the neural activity oscillates with specific frequencies [9]. This can be an indication that there are mechanisms in the cortex that serialize and synchronize the activity of groups of neurons. In [27], Mumford suggests that the claustrum, a small subcortical structure

which is connected to the whole cortex, plays a role in the serialization of bottom-up and top-down processing between connected cortical areas. In our network model, we assign the neurons to different neuron groups. All neurons in the same group work synchronously, which means that they all compute their states at the same time step. The state computation of the different neurons groups is serialized. In a single iteration, they are activated in the following order: feature neurons, category neurons, nselect neurons (if the first implementation of the feature selection circuit is used), select neurons.

4.3.5 Network size

In the computer model, a single neuron can form excitatory as well as inhibitory connections with postsynaptic neurons. This means that some inhibitory neurons and their connections in the described model can be replaced by inhibitory connections in the computer model, which reduces the number of neurons and connections. If we ignore the mechanism that generates the initial activations, a network for M features and N categories that uses the first implementation of the feature selection circuit (one select neuron for every feature) contains $5M + N$ neurons and $7M + 4MN$ connections. A network that uses the second implementation of the feature selection circuit (one select neuron for every pair of categories) contains $3M + (N^2 + N)/2$ neurons and $4M + 2MN + 2N^2 - 2N$ connections.

4.4 Simulation of the binary model

Our binary network model is now complete, so that we can test it with a practical example. The task of the network is to recognize two-dimensional patterns of black and white pixels. As we have already described in section 3.2, the features that are extracted from the input are small subpatterns of 3×3 pixels. Because we use only patterns whose center pixel is set, there are $2^8 = 256$ different features. To make the recognition process translation invariant, we ignore any position information and detect only whether a feature occurs in the input or not.

The figures 4.11 and 4.12 show two sets of 8×8 pixel images that we use to test the network model. The images have been created in this small size to make it easier to analyze the behavior of the network. Because the binary model is unable to compensate for noise or other variations in the input, the same image data is used to build a network and to test its recognition capabilities.

The terms we have defined in the description of the abstract model in section 3.6 have the following meaning in this example: An input pattern p is a vector $p = (x_1, \ldots, x_{256})$ of 256 elements. Its ith element is 1 if the 3×3 pixel pattern that corresponds to feature i occurs in the image that defines

Figure 4.11: Digit images

Figure 4.12: Letter images

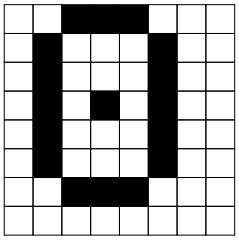
the pattern, and 0 if not. The reference patterns are the input patterns that are defined by the images in the figures 4.11 and 4.12.

From the image data, two networks have been created, one for the recognition of the digit images, one for the recognition of the letter images. Both networks use the second implementation variant of the feature selection circuit (one select neuron for each pair of categories) because the first variant does not work well in the binary model; we have explained the reasons in subsection 4.3.3. A step function is used to compute the state of the neurons. In the network, there is one select neuron for each pair of categories; each of these neurons selects a feature that occurs in one category, but not in the other. The network for the recognition of the digit images contains 823 neurons and 6324 connections; the network for the recognition of the letter images contains 1119 neurons and 15636 connections.

As predicted in the description of the feature selection circuit, both networks need only 2 iterations to recognize an input pattern. Of the 256 features, the digit network evaluates 5 features per input and 8 different features to recognize all digit images. The letter network evaluates 11–13 features per input and 17 different features to recognize all letter images. This shows that most of the possible 256 features are irrelevant for the recognition of the patterns in our sample data sets. If this property is also valid for more complex patterns, which is a reasonable assumption, this justifies our approach to evaluate the features selectively. The fact that most of the features are never evaluated could be used in a learning process to remove many unnecessary neurons and connections from the network to considerably reduce the network's complexity. In our example, the size of the digit network could be reduced to 79 neurons and 372 connections, and the size of the letter network could be reduced to 402 neurons and 2252 connections.

Figure 4.13 illustrates the recognition process for a single input in more detail. The input that is presented to the network is the image of the digit

Input



Iteration 1

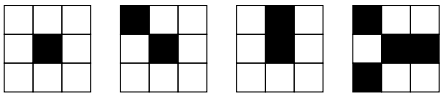
Evaluated features:

#68

Categories:

Iteration 2

Evaluated features:



#0 #1 #2 #49

Categories:

Figure 4.13: Recognition of the image of the digit “0” in the binary model

"0". In the first iteration, the feature #68 is selected and evaluated.[2] In our example network, the selection signal for this feature is generated externally, but it could also be generated by a special "unknown" category, which is activated when all other categories are inactive. The reason that the feature #68 is selected first is that it occurs in 5 of the 10 categories, so that a maximum information gain can be expected from its evaluation. Feature #68 occurs in the input; it activates the categories "0", "6", "7", "8", "9", and deactivates the other categories (inactive categories are shown in gray). In the next iteration, for each pair of active categories one feature is selected and evaluated. As some pairs of categories select the same feature, only 4 additional features are evaluated, namely #0, #1, #2, and #49. After the evaluation of these features, only category "0" is still active.

It is even possible to further reduce the number of features that need to be evaluated to recognize an input pattern. In the construction of the feature selection circuit, we required only that each pair of categories selects a feature that separates both. For each pair of categories there is usually more than one feature that separates them. Therefore, to lower the number of evaluated features, pairs of categories should select features that also separate as many other pairs as possible.

4.5 Implementation of an extended model

Although our binary network model works well, it has the limitation that it can only be used to classify inputs where the evaluated features exactly match those in one of the stored categories. For many applications, however, it is required that a pattern recognition system is able to compensate for small errors in the input. In Chapter 3, we have seen that the attempt to generalize our abstract model has lead only to very limited results. In particular, it is unclear how the extension of the abstract model could be implemented in a neural network.

In the binary model, there are three elements that can be extended from binary values to real values from the interval $[0, 1]$: the feature values in the input $x_1, \ldots, x_M$, the predictions $g_{i,k}$, and the probabilities $y_1, \ldots, y_N$ of the categories. Of course the probabilities are also real values in the binary model. But as we assumed that all categories are equally probable in the long term, a category could have only two possible values at every point in time, namely 0 or $1/n$, where n is the number of categories whose probability is greater than 0 at that point in time. For this reason we can say that the probabilities are also "binary" values in the binary model.

Extending the feature values $x_1, \ldots, x_M$ to real values is not really necessary for our purpose because in most situations, it is enough to know that a feature value is in a certain range. Thus, a real-valued feature can be replaced by

[2] The identifiers of the features are computed by interpreting their pixel patterns as binary numbers.

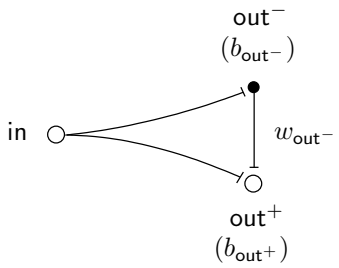


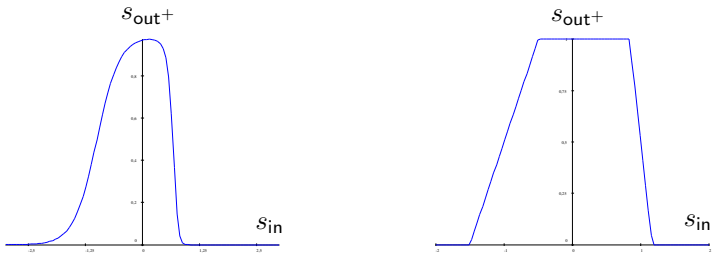
Figure 4.14: Range detection circuit

a number of binary features that represent different ranges in the original feature.

Figure 4.14 shows a neural circuit that detects a range of feature values. The state of the in neuron is the input to the circuit; the state of the out^+ neuron is used as the state of the whole circuit. We have already used such a circuit in the binary network model in the first implementation of the feature selection circuit. The range that this circuit is sensitive for is determined by the threshold values b_{out^-} and b_{out^+}. We assume that the connection from the out^- neuron to the out^+ neuron is much shorter than the other connections, so that we can neglect its delay.

The state function of such a range detection circuit depends on the state functions of the individual neurons and on the strengths of their connections. If the neurons' state functions are step functions (as in the binary model), the resulting state function of the circuit has the form of a gate function; if they are sigmoid functions, the state function of the circuit is similar to a Gauss function. Figure 4.15 (a) shows the state function of a circuit where the neurons have sigmoid state functions; figure 4.15 (b) shows the state function of a circuit where the neurons have piecewise linear state functions. Note that the graphs of the functions are not symmetric with respect to a vertical line. The reason is that the inhibitory connection from the out^- neuron to the out^+ neuron must be strong enough to bring the state of the circuit to 0 when the input exceeds the upper bound of the circuit. It is also possible to build a range detection circuit with a symmetric state function, but such a circuit needs two additional neurons, which seems to be too much for this task.

Extending the predictions $g_{i,k}$ can be useful if the association strengths between features and categories vary among the different features. For example, let us assume that the categories represent hand-written character images. Usually, there are variations in the writing of each character. Because of these variations, some features *may* occur in an input corresponding to a given character, while other features *must* or *must not* occur to make the input



(a) sigmoid (b) piecewise linear

Figure 4.15: State functions of a range detection circuit

classifiable. To capture these differences, we define the weights $c_{i,k}$ as

$$c_{i,k} := \frac{2n_{i,k}}{|\mathcal{T}|} - 1 \tag{4.13}$$

where $|\mathcal{T}|$ is the number of patterns in the training set $\mathcal{T}$, and $n_{i,k}$ is the number of patterns from the training set that are examples for category i and contain feature k. The absolute value of the weight $c_{i,k}$ measures the association strength between feature k and category i. Features k with $c_{i,k} = 1$ (meaning that feature k must occur in inputs of category i) or $c_{i,k} = -1$ (meaning that feature k must not occur in inputs of category i) contribute the maximum amount of information to the decision whether the input belongs to category i or not. On the other hand, features k with $c_{i,k} = 0$ contribute no information to this decision. Based on the weights $c_{i,k}$, we get the "old" predictions by setting $g_{i,k} := 0$ if $c_{i,k} \leq 0$, and $g_{i,k} := 1$ if $c_{i,k} > 0$.

For our purpose, the most important extension of the binary model is the extension of the category probabilities $y_1, \ldots y_n$. By allowing the categories to have probabilities other than 0 and $1/n$, where n is the number of categories with non-zero probabilities, the pattern recognition system can tolerate deviations from the stored reference patterns. When the prediction $g_{i,k}$ of a category i for a feature k deviates from the value x_k that has actually been observed for this feature, the probability of the category i is not necessarily set to 0. Instead, the probability y_i is reduced by an amount that depends on the association strength between category i and feature k. The aim of this process is to assign the highest probability to the category whose predictions have the smallest weighted deviation (weighted by the $|c_{i,k}|$) from the feature values that have been observed.

In the following subsections, we describe the differences of the extended network model to the binary model. If not stated differently, we assume that the state functions of the neurons are piecewise linear functions with threshold value 0 and slope 1.

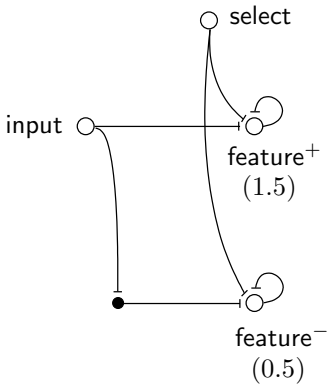


Figure 4.16: Feature evaluation circuit

4.5.1 Feature evaluation

In the binary network model, the threshold values of the feature$^+$ and feature$^-$ neurons have to be large enough to prevent these neurons from being activated solely by select neurons. To still allow the input signals to activate the feature neurons, the positive or negative input signals are amplified by the relatively large weights of the connections between input and feature neurons. In the extended model, however, this approach does not work. As our aim is to tolerate errors in the input, these errors would be amplified as well, which is certainly not what we want.[3]A solution to this problem is to use only one select neuron for each feature. If there is more than one signal that can select a feature, these signals must be collected by a single select neuron. As a consequence, we can set the weights and thresholds of this subcircuit to fixed values. Figure 4.16 shows the feature evaluation circuit of the extended model.

4.5.2 Category activation

The category activation circuit in the binary network model has to activate those category neurons whose predictions match all observed feature values, and deactivate all other category neurons. In the extended model, the circuit should guarantee the following behavior: if the weighted deviation of the predictions of a category from the observed feature values is greater for a category i than for a category j, then category i is assigned a lower probability than category j, which means that the activation of category neuron i must be lower than that of the category neuron j. The weighted deviation $d_i(n)$ of

[3]We use the term "error" here for all deviations of the input patterns from the reference patterns stored in the system, even though they need not be errors in the strict sense of the word.

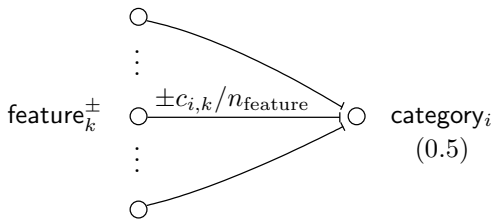


Figure 4.17: Category activation circuit

the predictions of category i from the feature values observed in the iterations $1, \ldots, n$ is defined as

$$d_i(n) := \sum_{k=1}^{n} |c_{i,\alpha_k}| \, d(x_{\alpha_k}, g_{i,\alpha_k}) \tag{4.14}$$

where $d(x_k, g_{i,k}) := 1$ if $x_k \neq g_{i,k}$ and $d(x_k, g_{i,k}) := 0$ if $x_k = g_{i,k}$ (as described above, the predictions $g_{i,k}$ and the feature values x_k remain binary values in the extended model).

Figure 4.17 sketches a circuit that realizes the desired behavior (inhibitory neurons are replaced by inhibitory connections). Every feature neuron is connected to every category neuron. The strength of the connection from the feature$^+$neuron k to the category neuron i is set to $c_{i,k}/n_{\text{feature}}$; the strength of the connection from the feature$^-$ neuron k to the category neuron i is set to $-c_{i,k}/n_{\text{feature}}$. The constant n_{feature} should be set to a value not less than the maximum number of features that must be evaluated to identify an input.

A possible extension of the circuit shown in figure 4.17, which can be useful in some situations, is the introduction of local or global competition between the categories. This can be done by inserting inhibitory connections either between all category neurons, or between local subsets of category neurons. We shall see in the simulations how this influences the behavior of the network.

4.5.3 Feature selection

As we have already seen in section 3.7.2, the top-down path is the part of the model that is most difficult to generalize. The reason is that to determine whether a feature is to be selected or not, we have to estimate the expected information gain resulting from its evaluation.[4] Without making further assumptions, however, it is very hard to find a reasonable estimation for this value. Therefore, we simply use the two implementations of the feature eval-

[4] As the neurons' states can take all values from the interval $[0, 1]$ in the extended model, we cannot really say that a feature is "selected" by this circuit. Its task can more appropriately be characterized as amplifying some features and attenuating others.

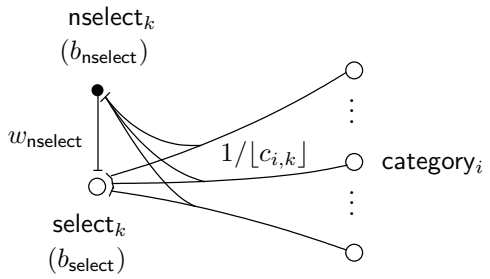


Figure 4.18: Feature selection circuit, implementation 1

uation circuit that we have developed for the binary model and see if we can do something useful with them.

The first implementation of the binary feature selection circuit estimates the probability that the feature k occurs in the input by the weighted sum $\sum_{i=1}^{N} y_i g_{i,k}$ and selects all features for which this value is in a certain fixed interval. In the extended model, the probability p_k that the feature k occurs in the input can be estimated as

$$p_k \approx \sum_{i=1}^{N} y_i \tilde{c}_{i,k}, \quad \tilde{c}_{i,k} = (c_{i,k} + 1)/2 \tag{4.15}$$

It is important to note that the dependencies between the features are completely ignored by the estimation in equation 4.15. To take the dependencies into account, however, we had to introduce additional parameters into the model, which is something we want to avoid.

The estimation 4.15 suggests that the feature selection circuit should select features k that minimize $\left|\sum_{i=1}^{N} y_i c_{i,k}\right|$. On the other hand, the circuit should favor features with larger absolute weights because they contribute more information (in the sense that we have described above at the begin of section 4.5); this can be achieved by selecting features k that maximize $\sum_{i=1}^{N} y_i |c_{i,k}|$. How can these two objectives, minimizing $\left|\sum_{i=1}^{N} y_i c_{i,k}\right|$ and maximizing $\sum_{i=1}^{N} y_i |c_{i,k}|$, be integrated in the circuit? Assuming that we want to keep the structure of the circuit from the binary model (otherwise we could be tempted to create a more complex circuit), we can only adapt the weights of the top-down connections. A heuristic approach to integrate the two objectives is to select features that minimize $\left|\sum_{i=1}^{N} y_i / \lfloor c_{i,k} \rfloor\right|$, where $\lfloor c_{i,k} \rfloor := c_{i,k}$ if $|c_{i,k}| > \varepsilon$, $\lfloor c_{i,k} \rfloor := \varepsilon$ if $0 \leq c_{i,k} \leq \varepsilon$, and $\lfloor c_{i,k} \rfloor := -\varepsilon$ if $-\varepsilon \leq c_{i,k} < 0$ for some small $\varepsilon > 0$. This can be achieved by setting the weight of the connection from the category neuron i to the select and nselect neurons k to $1/\lfloor c_{i,k} \rfloor$ and

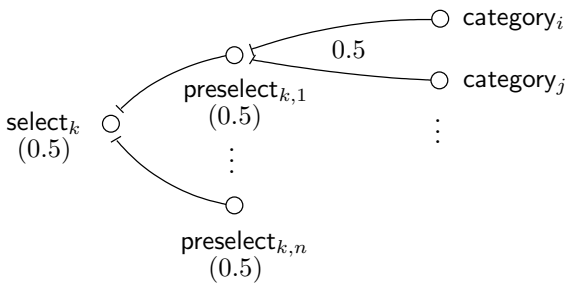


Figure 4.19: Feature selection circuit, implementation 2

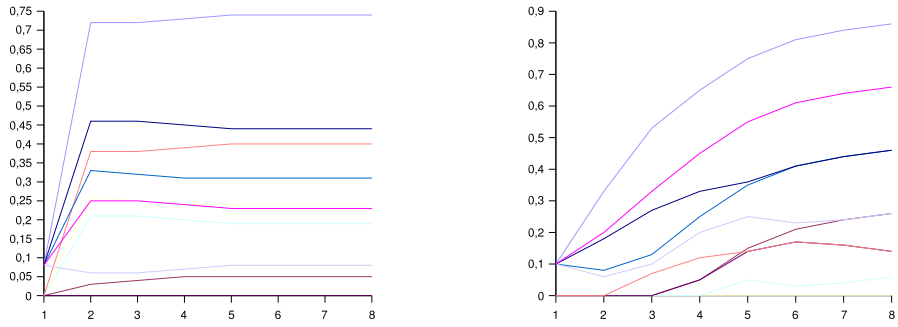
setting the neuron's threshold values so that they satisfy $b_{\mathsf{select}} < 0 < b_{\mathsf{nselect}}$. Of course the problems with the fixed interval $[b_{\mathsf{select}}, b_{\mathsf{nselect}}]$ remain in this implementation. Figure 4.18 shows the resulting circuit.

The second implementation of the feature selection circuit differs from the one in the binary model in two aspects. First, there is only a single select neuron for every feature. If more than one pair of categories selects the same feature, then for each of these pairs, an additional preselect neuron is inserted into the path from the category neurons to the select neuron. Thus, the select neuron collects the signals from all pairs of category neurons that select the corresponding feature. Figure 4.19 shows the feature selection circuit for feature k, which is selected by the category pair (i, j) and $n - 1$ other pairs.

The other aspect where the circuit for the extended model differs from the one in the binary model is how the feature is determined that is selected by a pair (i, j) of categories. In the binary model, it was sufficient to choose any feature k with $g_{i,k} \neq g_{j,k}$. In the extended model, the category pair (i, j) should select a feature for which the predictions of i and j are maximally different, that is, a feature k which maximizes $|c_{i,k} - c_{j,k}|$.

4.6 Simulation of the extended model

As in the binary model, we use computer simulations to examine whether the neural circuit we have proposed in the preceding section actually works as expected. To test the different aspects of the circuit we use the same task and the same image data as in our simulations of the binary circuit in section 4.4. It should be noted that to make the results of the different experiments easily comparable, we repeatedly show state traces that are based on the same input data. Of course, the results that are described in the text are based on experiments with many different inputs.



(a) Implementation 1

(b) Implementation 2

Figure 4.20: States of the category neurons during the recognition of the image of the digit "0" using different feature selection circuits

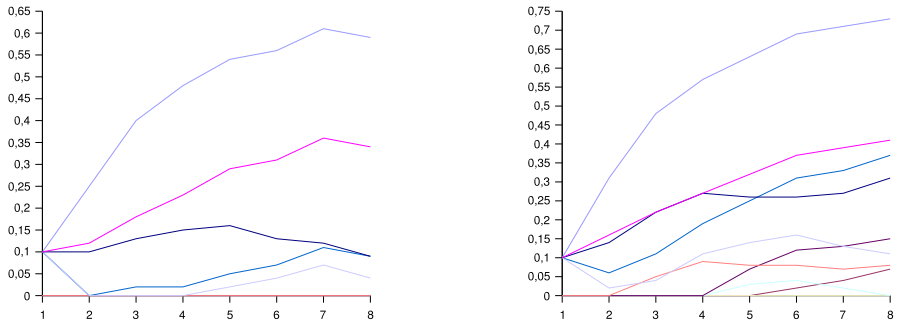
4.6.1 Implementation of the feature selection circuit

As a first experiment, we compare the two implementation alternatives for the feature selection circuit. To see which alternative works better, two networks have been created from the digit image data. Figure 4.20 shows how the states of the 10 category neurons develop over 8 iterations when the image of the digit "0" is used as input; the other inputs produce similar curves.

In this experiment, the category neurons with the strongest activations always corresponded to the categories that the input belonged to. This means that both implementations of the circuit were able to correctly identify all presented inputs. In the diagrams, we can see that the circuits need only about 2–3 iterations to determine the category of the input. The different shapes of the state curves can be explained as follows:

- Using implementation 1, the circuit collects nearly all of its information from the input in the first 2 iterations. If we trace the states of the select neurons, we see that in the first two iterations, a relatively large number of features is selected (25–31 in the example), but only very few of the select neurons have a strong activation (states greater than 0.7). After the second iteration, none of the select neurons has a state greater than 0.2, which means that no more features are selected and evaluated. The reason behind this behavior is that the interval defined by the threshold values of the circuit's select and nselect neurons cannot be adapted during the recognition process.

- Using implementation 2, the circuit evaluates less features (6–9 in the example) than using implementation 1. In every iteration, the states of



(a) Global competition

(b) Local competition

Figure 4.21: States of the category neurons during the recognition of the image of the digit "0" using different competition strategies

some select neurons are strengthened. Thus, using implementation 2, the circuit collects some information from the input in every iteration.

During the simulations, no set of parameters could be found for which the circuit using implementation 1 for feature selection did not show the problem described above. Additionally, implementation 1 turned out to be very sensitive to the exact parameter values. For example, the relation of the strengths of the feature-category connections to the interval defined by the select and nselect thresholds must be chosen carefully to result in a working circuit. Implementation 2, on the other hand, is relatively insensitive to the parameter values. The problem with this implementation, however, is that it is not so easy to decide which feature should be selected by a given pair of categories when there is more than one possibility. We shall see below in subsection 4.6.3 how this decision can affect the performance of the network.

4.6.2 Competition between the categories

In section 4.5.2, we have mentioned that it can be a useful strategy to introduce competition between the categories. We can distinguish between two forms: global and local competition. Global competition means that every category competes with every other; local competition means that only subsets of categories compete. In the network model, such a competition strategy can be realized by inserting inhibitory connections between the competing category neurons.

Figure 4.21 shows the results of the same experiment as in the preceding subsection, with the difference that additional inhibitory connections have been

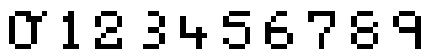


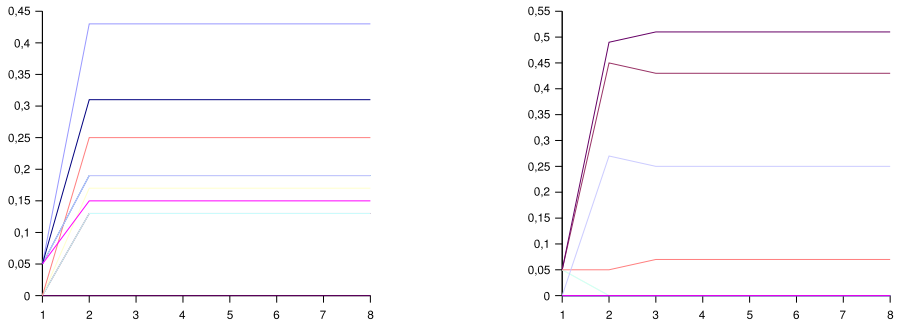
Figure 4.22: Some manually changed digit images

inserted between the category neurons. The network was built using implementation 2 of the feature selection circuit. The left diagram shows what happens when inhibitory connections are inserted between all category neurons. First, we can see that the relative distance between the states of the neurons with the stronger activations and the ones with the weaker activations is increased, compared to the situation without competition as shown in figure 4.20 (b). Second, the states of the neurons with weaker activations remain on a constant low level over the iterations, whereas without competition the states of all neurons increased. At least the first effect is quite useful because it allows a better separation of the most probable hypotheses from the less probable ones. The right diagram shows that this effect also occurs when there is only local competition between the categories. To produce this diagram, inhibitory connections have been inserted only between neighboring category neurons.

4.6.3 Inputs that contain errors

In the preparation of this work, many experiments have been carried out to analyze how the network behaves in the presence of inputs that deviate from the predictions of all stored categories. The desired behavior in such a situation would be that the input is assigned to the category whose predictions have the smallest difference to the features in the input. To test the network, specific errors have been manually added to the digit and letter images from section 4.4. Figure 4.22 shows some of the changed digit images that we used in the simulations. It should be kept in mind that because we do not evaluate individual pixels in the input but overlapping subpatterns, changing a single pixel can result in more than one changed feature in an input pattern (actually, a single pixel can directly influence up to 9 different features).

We begin with a network that uses implementation 1 of the feature selection circuit. Figure 4.23 shows the states of the category neurons in response to modified images of the digits “0” and “4”. When we compare these state traces with the one in figure 4.20 (a), we see that the network behaves very similar to the situation where the input contains no errors. The only difference is that if the input contains errors, the absolute values of the states of the category neurons are smaller. This behavior is consistent with our expectations, as an input that deviates from the predictions of all stored categories must be rated lower than an input that fully matches the predictions of a category. The most important result of this experiment is that for all analyzed examples the network behaved as expected and assigned the highest probability to the category whose predictions had the smallest difference to the input.



(a) Digit "0"

(b) Digit "4"

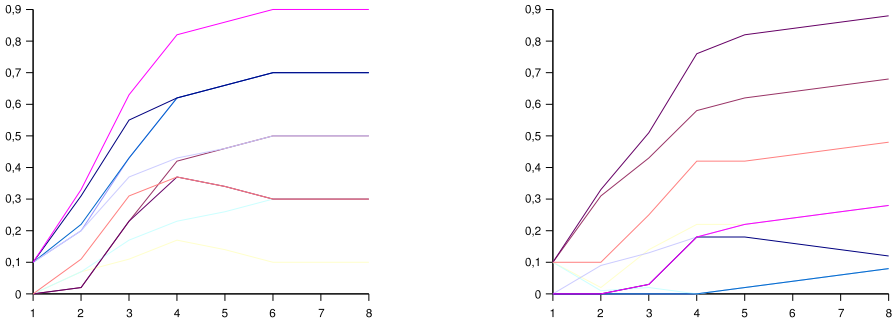
Figure 4.23: States of the category neurons during the recognition of modified images of the digits "0" and "4" using implementation 1 of the feature selection circuit

For networks that use implementation 2 of the feature selection circuit, the situation is not so clear. As we have mentioned before, the performance of such a network depends on the concrete features that are selected by the category pairs. The problem arises when for a pair (i, j) of categories there is more than one feature k that maximizes $|c_{i,k} - c_{j,k}|$, a situation that can occur rather frequently. As the connections are not allowed to change during the recognition of an input, it must be decided during the construction or development of the network which features are then selected.

Figure 4.24 shows what can happen when a naive strategy is applied to determine the feature that is selected by a pair of categories. The network was built so that whenever there was more than one feature for which the predictions of two categories were maximally different, the one with the smallest index was chosen. This works well for inputs without errors, but it fails for some inputs that deviate from the predictions. An example for which the recognition fails is shown in figure 4.24 (a): although category "0" is the one whose predictions have the smallest difference to the input, the network assigns the input to category "9".[5] The reason for this behavior is that too few of the features that separate the categories with higher probabilities are evaluated. In this specific example, the network evaluates only one feature that separates the categories "0" and "9", but 8 features for which these two categories make the same predictions.

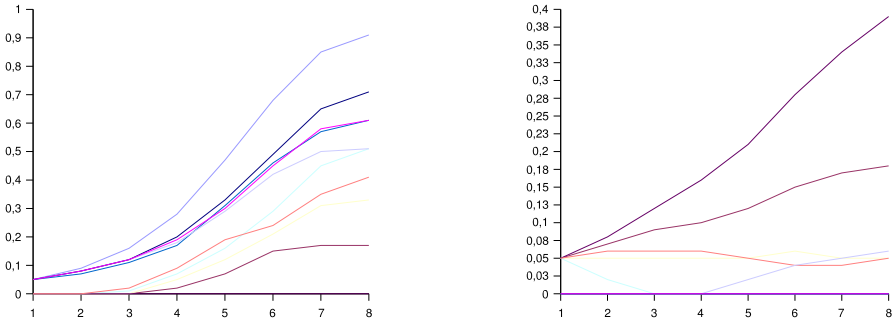
It is unlikely that there is a solution to this problem that works in all

[5] It depends on the printing whether this can actually be recognized from the diagram (the color of the strongest curve differs between the diagram 4.24 (a) and the corresponding diagram 4.20 (b)).



(a) Digit “0” (b) Digit “4”

Figure 4.24: States of the category neurons during the recognition of modified images of the digits “0” and “4” using implementation 2 of the feature selection circuit and a naive strategy to build the network



(a) Digit “0” (b) Digit “4”

Figure 4.25: States of the category neurons during the recognition of modified images of the digits “0” and “4” using implementation 2 of the feature selection circuit and a random strategy to build the network

situations. The best approach is probably to use a learning process to develop a feature selection circuit that performs well for a given set of examples. If the network must be explicitly constructed, however, a simple strategy is the following: if there is more than one feature for which the predictions of two categories are maximally different, select one of those features randomly. For all examples that were analyzed in the preparation of this work, this strategy actually produced better results than the naive strategy. Figure 4.25 shows two state traces produced by a network that was built using the random strategy.

4.7 Learning

Up to now, we have assumed that the network is constructed once and does not change during the recognition process. The strengths that were chosen for the connections were based on a fixed set of training patterns. For many applications, however, this is not a realistic scenario. For example, the full set of training patterns is often not known in advance. For such applications, it is important that the categories can be adapted by a learning process.

Generally, we can distinguish between supervised and unsupervised learning methods. In a supervised learning process, the system to be adapted receives an external training signal that contains information about the objective of the adaption process. In our case, the external signal would tell the network which category a given input belongs to. The network can then use these information to adapt its neurons and connections so that it becomes able to correctly classify inputs alone, without relying on the training signal. In an unsupervised learning process, the system to be adapted receives no training signal. Instead, the objective of the system is to develop a suitable set of categories from the inputs alone. To do this, the system has to analyze the structure of its input space.

In natural neural networks, there are probably many different adaption processes of both types (supervised and unsupervised) running in parallel. In this work, we restrict ourselves to supervised learning. Usually, the individual adaption steps are very small and change the parameters of the network only slightly. A significant change of the behavior of the network can therefore happen only over a large number of iterations.

4.7.1 Hebbian learning

It is widely accepted that in natural neural networks, changes of the synaptic strengths which depend on the activities of the presynaptic and postsynaptic neurons are a major source of adaption. In his 1949 book [12], Donald Hebb suggested a principle for this adaption process, which is now applied in many learning methods:

> "When an axon of cell A is near enough to excite cell B or repeatedly or consistently takes part in firing it, some growth or

> metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

According to this principle, an excitatory connection from a neuron A to a neuron B is strengthened when the activity of neuron A contributes to the activity of neuron B. Hebb considered only the strengthening of connections. Weakening can be introduced by assuming some form of competition between the connections that provide input to the same neuron. With competition, an excitatory connection from a neuron A to a neuron B is weakened when neuron A does not contribute to the activity of neuron B.

Since Hebb formulated his principle, it has been confirmed in many experiments. For example, a more recent observation is that the change of the strength of a synapse actually depends on the precise timing of presynaptic and postsynaptic activity [3, 36]. Measurements in cultivated neuron cells have shown that the strength of a synapse is increased if a presynaptic spike occurs in a small time interval before a postsynaptic spike, and decreased if a presynaptic spike occurs in a small time interval after a postsynaptic spike. This effect is called spike-timing-dependent plasticity (STDP). As in our neuron model, the state of a neuron corresponds to an average over the number of spikes in a time interval, however, we do cannot model this effect directly.

What about inhibitory connections? A simple approach is to treat them inversely to the excitatory connections. When doing this, however, it should be kept in mind that natural neural networks do not show such a symmetry between excitatory and inhibitory connections. As inhibitory synapses are often located at more central places at the postsynaptic neuron, they can have a greater influence on the activity of this neuron than excitatory synapses, which are usually located more peripherally at the dendrites. For our purpose, these details are not important; we treat excitatory and inhibitory connections symmetrically. Additionally, we allow the connections to change the type of their synapse during learning. As we have already described in section 4.1, this is also not possible in natural neural networks.

4.7.2 The learning rule

The learning rule that we use to adapt our network is very simple. After computing its state in iteration t, every adapting neuron i sets the weights w_{ij} of all its input connections to

$$w_{ij}(t+1) := w_{ij}(t) + \Delta w_{ij}(t) \tag{4.16}$$

where $\Delta w_{ij}(t)$ is defined as

$$\tau \Delta w_{ij}(t) := s_i(t)(2s_j(t - \Delta t_{ij}) - 1) - s_i(t)\alpha w_{ij}(t) \tag{4.17}$$

The constant $\tau > 0$ controls the rate of the weight adaption. The first term of Δw_{ij} implements the principle of Hebbian learning: If presynaptic activity is

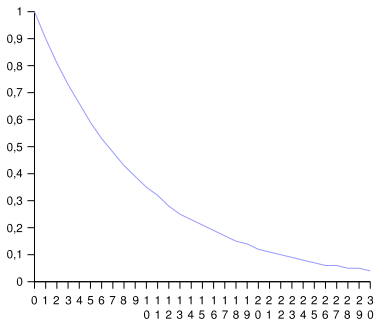


Figure 4.26: Average difference between synthetic and developed weights when learning the digit images over 30 training cycles

followed by postsynaptic activity, then $s_i(2s_j-1)$ is positive, which means that the connection should be strengthened. If, on the other hand, postsynaptic activity is not preceded by presynaptic activity, then $s_i(2s_j-1)$ is negative, which means that the connection should be weakened. If the postsynaptic neuron is not active at all, then $s_i(2s_j-1)=0$ and the connection strength is not changed. The decay term $s_i\alpha w_{ij}$ with $\alpha > 0$ is necessary to stabilize the connection strengths. Without it, the weights could grow to arbitrarily large positive or negative values as long as $s_i(2s_j-1) > 0$.

4.8 Simulation of the learning process

When the network must learn new or changed categories, the connections that must be adapted are the bottom-up connections from the feature neurons to the category neurons and the top-down connections from the category neurons to the select, nselect, or preselect neurons (depending on the implementation of the feature selection circuit). For the bottom-up path, it is reasonable to assume that there is an external training signal which activates the category that an input belongs to. In the human cortex, for example, such a training signal could originate in another cortical area. In contrast to this, we cannot assume that the network receives a training signal for the top-down path which tells the network what feature should be selected. Instead, the development of "good" top-down connections is probably a more subtle process. For this reason, we concentrate on the development of the bottom-up connections, which fits better in the scheme of supervised learning.

The objective of the learning process is to develop bottom-up weights that are similar to the weights that we used in our category activation circuit. In section 4.5.2, we have described that the strength of the connection from the feature$^+$ neuron k to the category neuron i should be set to $c_{i,k}/n_{\text{feature}}$, and

the strength of the connection from the feature$^-$ neuron k to the category neuron i to $-c_{i,k}/n_{\text{feature}}$. To measure the progress of the learning process, we use the average absolute difference between these synthetic weights and the weights that have been developed in the network.

Figure 4.26 shows the result of a learning simulation. Over 30 training cycles, the digit images from figure 4.11 have been presented to the network, and the corresponding category neurons have been activated by an external signal. The learning parameters were set to $\tau = 10$ and $\alpha = 1.0$. The weights from the feature neurons to the category neurons were initialized to small positive or negative random values. As can be seen in the figure, the difference between the synthetic weights and the weights in the network becomes very small. Thus, the learning process works as expected. What can also be recognized from the figure is the influence of the decay term: with the number of training cycles, the absolute values of the weights tend to increase, and the adaption of the weights slows down more and more.

Chapter 5

Summary

Let us make a short summary of the preceding chapters, of what we have achieved in this work and what not. Our aim was to develop a neural network model that can be applied to solve practical pattern recognition tasks. In the development of our model, we wanted to make use of principles that can be found in natural neural networks. Our hope was that this way we could get an insight into the inventions of nature.

Using knowledge from neuroanatomy and results from neuropsychological experiments, we concluded that the recurrent connections in natural neural networks must play an important role in pattern recognition processes. Consequently, we aimed at developing a recurrent network as well. By using information-theoretical considerations, we then formulated what features should be evaluated so that the network profits maximally from the information that are contained in the input, and how the observation of a feature value should influence the probabilities assigned to the categories so that only information that is actually contained in the input is taken into account. The model we developed is constructive in the sense that its objective is to find a combination of stored concepts which matches the actual observations; this combination corresponds to a probability distribution over the categories.

Next, we constructed a concrete neural network which implements the abstract model. The network consists of modules that solve the subtasks feature evaluation, category activation, and feature selection. We extended the network to support non-binary feature values, non-binary associations between features and categories, and to compensate for errors in the input to a certain extend. Using computer simulations, we have shown that both, the binary and the extended network, can be used to solve pattern recognition tasks.

Realistic sensory data is very high-dimensional and contains errors. What should have become clear is that every pattern recognition system that processes such data must concentrate on specific aspects of its input. Without a mechanism to ignore the parts of the data that do not contribute to the recognition of the input, such a system would hardly be able to successfully recognize anything at all. Naturally, which parts of the input are important and which can be ignored depends on the state of the pattern recognition system itself. In this work, we have demonstrated how a mechanism that follows these principles can be implemented in a neural network.

Suggestions for possible future work

The problems of our model that have arisen show the way for possible future work. Our analysis of the generalized model in section 3.7 did not lead to anything that could directly be implemented in a neural network. Possibly, more analytical results can be obtained by formulating additional constraints, for example, by making reasonable assumptions about the probability distribution of the feature values. Besides this, it could be interesting to work out how our model fits into the framework of Bayesian analysis [16, 22].

On the practical side, the performance of the network could be improved by developing a better feature selection circuit. As we have seen in the simulations in section 4.6.1, the information gain of the network falls down after a few iterations if we use the implementation that we have derived from the theoretical considerations. Making an improvement at this point probably requires an extension of the neuron model to support a maximum or minimum computation in the network. This should be completed by showing how the top-down connections in the network can be developed in a learning process.

In this work, we have examined a network with essentially two layers where one layer represented the observed features and the other layer represented the probabilities of the categories. It should be an interesting approach to use several of such pattern recognition networks as modules to build up a more complex system. If these pattern recognition modules could also be successfully applied to solve control tasks, then such a system would even be able to interact with its environment.

Bibliography

[1] David Applebaum. *Probability and Information.* Cambridge University Press, 1996.

[2] Guszti Bartfai. An ART-based modular architecture for learning hierarchical clusterings. Technical Report CS-TR-95/3, Department of Computer Science, Victoria University of Wellington, New Zealand, 1995.

[3] Guo-quiang Bi and Mu-min Poo. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Reviews in Neuroscience*, 24:139–66, 2001.

[4] Kim T. Blackwell, Thomas P. Vogel, and Daniel L. Alkon. Pattern matching in a model of dendritic spines. *Network: Computation in Neural Systems*, 9:107–121, 1998.

[5] Gail Carpenter and Stephen Grossberg. ART2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930, 1987.

[6] Gail Carpenter and Stephen Grossberg. A massively parallel architecture for a self organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

[7] Gail Carpenter and Stephen Grossberg. ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3:129–152, 1990.

[8] Peter Dayan and L. F. Abbot. *Theoretical Neuroscience.* MIT Press, 2001.

[9] Charles M. Gray and Wolf Singer. Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex. *Proceedings of the National Acadamy of Sciences*, 86:1698–1702, 1989.

[10] Stephen Grossberg. Adaptive pattern classification and universal pattern recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[11] Stephen Grossberg. Adaptive pattern classification and universal pattern recoding: II. Feedback, expectation, olfactation, and illusions. *Biological Cybernetics*, 23:187–202, 1976.

[12] Donald Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949.

[13] Harro Heuser. *Lehrbuch der Analysis, Teil 2*. B. G. Teubner, 11th edition, 2000.

[14] Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[15] David H. Hubel and Torsten N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[16] Edwin T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, Fragmentary Edition of March 1996.

[17] Gaetano Kanisza. *Organization in Vision*. Praeger, 1979.

[18] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18:49–60, 1988.

[19] Victor A. F. Lamme and Pieter R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neuroscience*, 23(1):571–579, 2000.

[20] Tai Sing Lee, David Mumford, Richard Romero, and Victor A. F. Lamme. The role of the primary visual cortex in higher level vision. *Vision Research*, 38:2429–2454, 1998.

[21] Tai Sing Lee and My Nguyen. Dynamics of subjective contour formation in the early visual vortex. *Proceedings of the National Acadamy of Sciences of the USA*, 98(4):1907–1991, 2001.

[22] David J. C. MacKay. Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6:469–505, 1995.

[23] Warrent McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[24] Bartlett W. Mel. Why have dendrites? A computational perspective. In G. Stuart, N. Spruston, and M. Hausser, editors, *Dendrites*. Oxford University Press, 1999.

[25] Bartlett W. Mel, Daniel L. Ruderman, and Kevin A. Archie. Translation-invariant orientation tuning in visual 'complex' cells could derive from intradendritic computations. *Journal of Neuroscience*, 18:4325–4334, 1998.

[26] David Mumford. On the computational architecture of the neocortex, I. The role of the thalamo-cortical loop. *Biological Cybernetics*, 65:135–145, 1991.

[27] David Mumford. On the computational architecture of the neocortex, II. The role of cortico-cortical loops. *Biological Cybernetics*, 66:241–251, 1992.

[28] Tatjana A. Nazir and J. Kevin O'Regan. Some results on translation invariance in the human visual system. *Spatial Vision*, 5:81–100, 1990.

[29] Christoph Paus. Notes for advanced mechanics. Lecture notes for course 8.21, MIT, available at http://web.mit.edu/8.21/www/intro.html, 2002.

[30] Rajeev D. S. Raizada and Stephen Grossberg. Towards a theory of the laminar architecture of cerebral cortex: Computational clues from the visual system. *Cerebral Cortex*, 13:100–113, 2003.

[31] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1996.

[32] Edmund T. Rolls, Martin J. Tovée, and Stefano Panzeri. The neurophysiology of backward visual masking: Information analysis. *Journal of Cognitive Neuroscience*, 11(3):300–311, 1999.

[33] D. Rumelhart and J. McClelland. *Parallel Distributed Processing*. MIT Press, 1986.

[34] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communications*. University of Illinois Press, 1949.

[35] Gordon Shepherd. *The synaptic organization of the brain, 3rd edition*. Oxford University Press, 1990.

[36] Sen Song, Kenneth D. Miller, and L. F. Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926, 2000.

[37] Shimon Ullman and Sergei Soloviev. Computation of pattern invariance in brain-like structures. *Neural Networks*, 12:1021–1036, 1999.

[38] Arjen van Ooyen, Jacob Duijnhouwer, Michiel W. H. Remme, and Jaap van Pelt. The effect of dendritic topology on firing patterns in model neurons. *Network: Computation in Neural Systems*, 13:311–325, 2002.

[39] Ruye Wang. A hybrid learning network for shift, orientation, and scaling invariant pattern recognition. *Network: Computation in Neural Systems*, 12:493–512, 2001.

[40] Laurenz Wiskott. How does our visual system achieve shift and size invariance? In J. L. van Hemmen and T. J. Sejnowski, editors, *Problems in Systems Neuroscience*. Oxford University Press, 2002.