Maximum a Posteriori Models for Cortical Modeling:
Feature Detectors, Topography and Modularity

vorgelegt von
Diplom Physiker
Cornelius Weber
aus Bielefeld

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr.-Ing. Günter Hommel
1. Berichter: Prof. Dr. rer. nat. Klaus Obermayer
2. Berichter: Prof. Dr. sc. nat. Fritz Wysotzki

Tag der wissenschaftlichen Aussprache: 1. Mai 2001

Berlin 2001
D 83

Department of Computer Science
Technical University of Berlin
Neural Information Processing

Maximum a Posteriori Models for Cortical Modeling: Feature Detectors, Topography and Modularity

PhD Thesis by Cornelius Weber, Berlin, July 31, 2000
Defense on Mai 1, 2001

Contents

1 Introduction

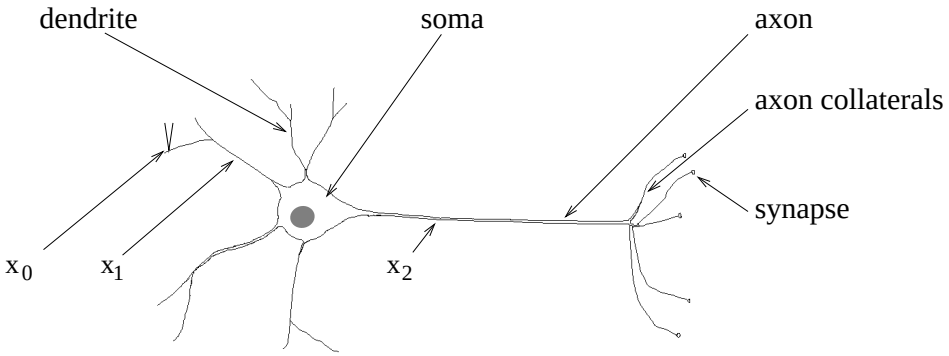


Figure 1: Schematic illustration of an example neuron. Information flow is from the dendrites, left, to the axon collaterals, right.

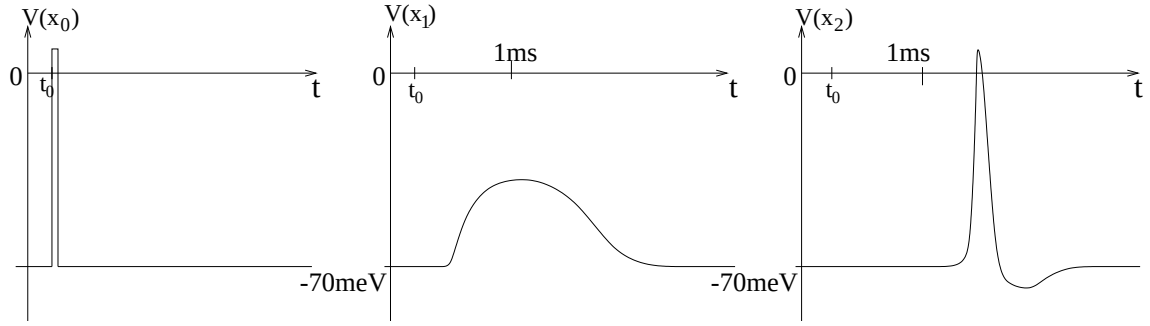


Figure 2: Membrane potential vs. time at three different positions marked in Fig. 1. If a dendrite receives a depolarizing current $V(x_0)$ at x_0, t_0, then this disturbation travels along the dendrite (x_1) and broadens to finally reach the soma. If the sum of these depolarizing potentials reaches a threshold at the soma, then the axon is triggered to carry spikes (x_2). These stereotyped impulses propagate rapidly without attenuation over long distances towards the synapses on the axon collaterals. The behavior of a cell is quantified by its size, shape and membrane properties and the types of neuro-transmitters used at synapses determine the influence to and from other cells.

Figs. 1 and 3 show cartoons of the smallest and the largest parts of the brain. The whole brain contains about 10^{10} to 10^{12} cells. Each of these use only the type of signals described in Fig. 2, and thus a single neuron carries as little information as whether a spike occurs within one milli-second or not. The complex behavior of this system arises only from the connectivity between cells. This is specified by an

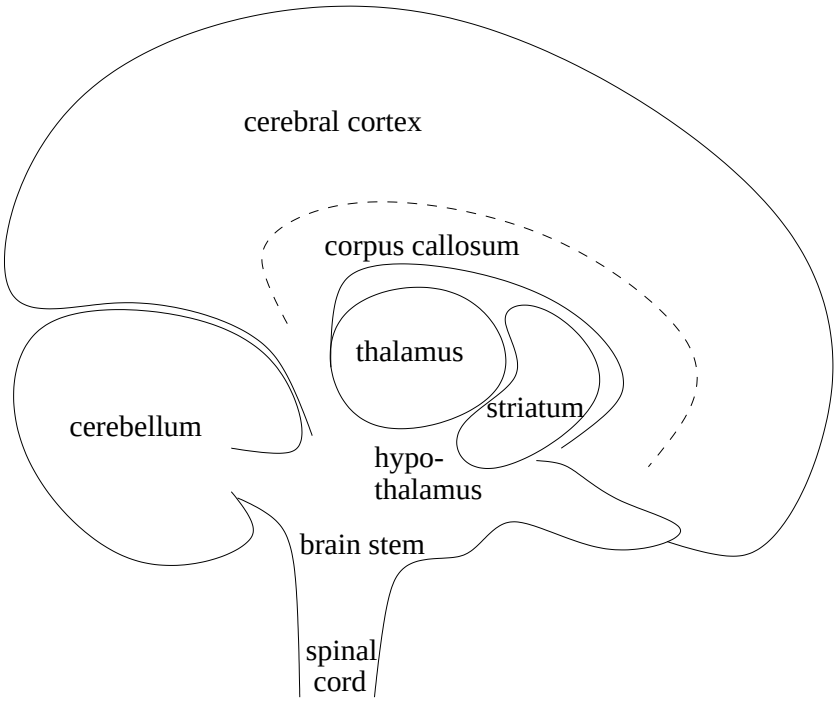


Figure 3: Schematic illustration of the human brain, drawn from a cross section [33] and a cartoon [43]. The **spinal cord** controls movement and receives sensory information from parts of the body other than the head.
The **brain stem** consists of three parts, the medulla, pons and midbrain. It controls movement and receives sensory information from the head. It also contains several collections of cell bodies, called cranial nerve nuclei, where each is specialized to convey information of distinct regions or senses. The reticular formation regulates levels of arousal and awareness. The medulla oblongata, which lies directly above the spinal cord, controls vital autonomic functions as digestion, breathing and heart rate. The pons conveys information to the cerebellum. The **cerebellum** modulates via inhibitory output only the force and range of movement and improves motor skills. The midbrain contains the tectum (superior colliculus) and controls sensory and motor functions such as eye movement and coordination of visual and auditory reflexes. The **thalamus** contains the LGN and pulvinar. It relays most information which reaches the cerebral cortex. The **hypothalamus** regulates autonomic, endocrine and visceral function. The (corpus) **striatum** contains the caudate nucleus and the putamen. Together with the globus pallidus, they make up the basal ganglia and are important for controlling movement. The **corpus callosum** interconnects the **cerebral cortices** of the two hemispheres. Description taken from [35].

even larger number of parameters. If we assume an address of a cell to be 4 bytes long (which can address $\mathbf{2^{32} = 4.3 \cdot 10^9}$ cells) then the connectivity information of

the whole brain sums up to approximately

$$10^{10}\ \text{[cells]}\ \cdot 100\ \frac{\text{[connections]}}{\text{[cell]}}\ \cdot 4\ \text{[bytes]}\ = 4 \cdot 10^{12}\ \text{[bytes]}$$

or 4 tera-bytes. Therefore, connectivity cannot be a result of a direct description by (a smaller number of) genes, but must emerge by self-organization. It is a goal of this thesis to charactierize such self-organization processes in terms of how they work and how they make use of data through learning. They are tested against which of the known structures of the brain they can explain.

At an intermediate scale, in general, we do **not** find very distinctive structures, e.g. that a cluster of 10 to 100 cells would perform its own computations independent of other cells. Connectivity is instead diffuse and often links local and distant targets in combination. In addition, recurrency causes loops, which can be observed at all scales, at the smallest of which are connections from one cell to itself via “autapses”. But again, loops are not clearly confined because of the diffuse connectivity.

However, there are certain principles observed throughout brain structures.

- Grey matter (densely packed cell bodies) can be distinguished from white matter (long-range axons). The grey matter makes up the organs of the brain or parts of them. The white matter can be organized into nerve bundles thereby maintaining functional specificity between the interconnected regions.
- In many organs, cells arrange in layers and connectivity between cells is dependent on layers and cell types.
- Every cell as well as the whole brain has a clearly defined input and output. Information is processed in more or less well defined “pathways”.
- Topographic mappings display a high degree of parallel processing of equivalent features.
- Serial arrangements between topographic map can be considered to form a hierarchy.

These properties provide a scaffold on which models of the brain can be tested. Developmental models may predict these properties based on simple developmental mechanisms. Functional models may predict physiological properties (as obtained e.g. by optical imaging or FMRI) based on the observed anatomy. In this thesis, we will explore, which architectures can emerge in developmental models but also, which functional properties these models have.

Outline of the thesis

Section 2 summarizes biological findings about the cortex. We will consider its areas, connectivity between these and resolve these down to layers. As a consequence,

we will confine to which degree intrinsic mechanisms and activity dependent mechanisms govern development.

Section 3 provides the theoretical foundation for the models used. We assume that the Bayesian maximum likelihood underlies cortical function. Accordingly, the function of the cortex is to represent the real world so that the learned stimuli can be generated autonomously.

Section 4 presents the models which implement the theoretical principles. We show that certain Bayesian priors lead to corrections to the learning rules which reflect intrinsic mechanisms.

Section 5 presents simulation results. We will present feature maps, topographic maps and the organization of modules as the results of developmental mechanisms. Additionally we will demonstrate how the activations develop during the relaxation procedures.

2 The Cortex

The cortex is the largest organ of the human brain. However, a mammal can survive without a cortex and lower animals do not even have a cortex. Essential functions like controlling inner organs and instincts reside in other parts of the brain (see Fig. 3 caption). So what does the cortex do? An other way to pose this question is: what can the lower animals **not** do? If lower animals cannot learn a complex behavior we may infer that they cannot understand a complex environment. In other words: the cortex may provide a representation of a complex environment to mammals.

This suggests that the cortex is a highly organized structure. Indeed, in the absence of input it can generate dreams and imagery from intrinsic spontaneous activity. Recurrent connectivity may be the key to these capabilities. Neurons in the cortex receive signals from other neurons rather than directly from sensory inputs. The responses to inputs are thereby context dependent and thus cannot be fully described by the direct connection to the sensory input via the receptive field only.

On a macroscopic scale, the cortex can be structurized into some dozens of areas.

2.1 Modularity

Cortical areas are hard to be distinguished by one criterion alone. Instead, the following criteria may be combined:

- Architecture: different structure revealed by staining. This method is reliable only for a few areas [20].
- Connectivity "fingerprint", the connectivity pattern to other cortical areas. Comment: if two areas had same connectivity patterns, then they would be the same.
- Topographic organization w.r.t. the visual field in visual cortical areas: half of all visual areas show a measurable degree of topography [20].
- Recent findings suggest that a “neurochemical fingerprint” [23] determines the earliest compartmentalization in corticogenesis [18].

Data

In the following, we will first collect some data from cortical areas and their mutual connectivity. The question arises, what is the distance between these areas as defined by their connections. Visualization techniques like non-metric multi dimensional scaling (NMDS) as well as simple models will help to answer this question.

Areas and connections The number of cortical areas is 65 in the cat [52] and 73 in the macaque [72]. The number of connections reported between these is 1139

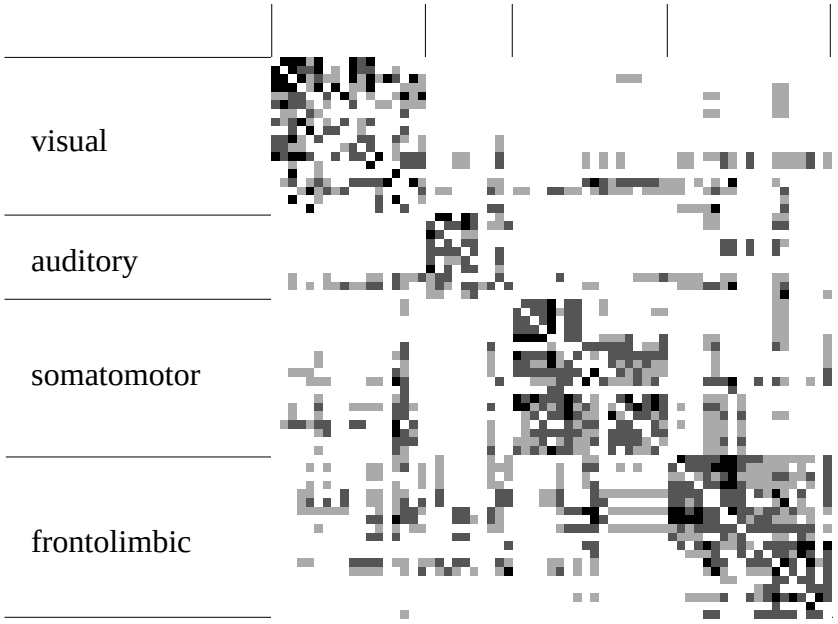


Figure 4: The cortico-cortical connection matrix of the cat. On both axes, the 65 cortical areas are arranged into the four complexes, visual system, auditory system, somato-sensory system and fronto-limbic system. Connections along the horizontal axis are afferent, and those along the vertical axis are efferent. Strong connections are depicted dark, weaker connections are depicted in a lighter grey, missing connections white. Self connections (diagonal) are omitted. Data were taken from [52].

in the cat (Fig. 4) which represents 27.4% of all possible connections (only cortico-cortical, but no contralateral connections are considered). The number of reported connections in the macaque is 758 which represents 15% of all possible connections.

Most of the connections are bidirectional: they consist of axons going into both directions. The number of one-way connections is 136, i.e. 18% of all connections (macaque). Together there is a mean of approximately 10 input- and 10 output-connections per area (macaque).

Complexes of areas The cortex can be divided into 4 complexes (macaque) [72] with different functionality and sizes: visual cortex 55%, somato-sensory 11%, motor 8%, auditory 3% $\rightarrow$ rest 23% [20].

The number of areas in the visual system of the macaque is 25 plus 7 visual-association areas. The number of connections is 305, i.e. 31% of all possible connections. There are 3 non-reciprocal projections: no V4t $\rightarrow$ V1, no V2 $\rightarrow$ FST, V4t $\rightarrow$ MSTI [20]. An “intermediate” example: the projection from V1 to V4 is robust, but the back-projection from V4 to V1 is weak and occasional, possibly dependent on eccentricity. The somato-sensory/motor system has 13 areas with 62 connections, i.e. 40% of all possible connections. Thus, connectivity between areas within one complex is higher than average.

Quantitative aspects The connection strengths between areas (i.e. density of fibers) comprise two orders of magnitude [20]. Only 30-50% of connections may be "robust". Sizes of areas: the visual cortex of the macaque has the size of a medium-sized cookie, 8cm diameter, a whole hemisphere of the cortex is the size of a large cookie, 12cm. Areas V1 and V2 take 12% of the surface area, smallest areas are 50-times smaller. Note: there is a 2-fold variability of single areas from one brain to the next [20].

Finally, it should be noted that every visual area is connected to (a) noncortical area(s). The number of these connections may outrange the number of cortico-cortical connections.

Simple models for the connections data

Let a model [52] assume all neighboring areas to be connected ("nearest neighbor model", Tab. 1). It predicts too few connections, but most of them are indeed observed in biology. Let another model assume areas to be connected to their neighbors and to the second nearest neighbors. Then there are many wrong guesses, especially between different complexes, i.e. in areas away from the diagonal in the connection matrix. Tab. 1 shows the scores of these models.

	nearest neighbor model		nearest + 2nd nearest neighbor	
% of	real connections predicted by model	right guesses by model	real connections predicted by model	right guesses by model
visual	26.1%	77.2%	56.6%	61.1%
auditory	27.9%	70.6%	61.6%	64.6%
global	19.5%	68.9%	47.8%	51.4%

Table 1: Explanation capabilities of simple geometrical models. Data are taken from [52].

The errors of these models are systematic in that they ignore the influence of activity and function: the models predict connections between functionally dissimilar but neighboring areas. The author notes that through activity based refinement these missing neighbor connections should be correctly predicted. The developmental mechanisms which underly the principles stated by these models are suspected to fulfill the following goals: minimizing genetic information, minimizing metabolic costs and minimizing the total wiring volume.

Visualization methods

A number of different methods have been applied in order to make the connections data less abstract.

NMDS The connection matrix does not supply a metric on the areas. A metric d is define by the following axioms: *(i)* $d(x,y) = 0 \Leftrightarrow x = y$, *(ii)* $d(y,x) = d(x,y)$ and *(iii)* $d(x,z) \leq d(x,y) + d(y,z)$. The last condition is not met, e.g. if areas x and y are linked and areas y and z are linked but if area x is not linked to z.

Non-metric multi dimensional scaling (NMDS) describes a class of algorithms which projects high dimensional data onto a low dimensional surface. Applied to the connection matrix (Fig. 4) the result should be a graph on 2 dimensions (where there is a metric) where the original proximities between the areas in high-dimensional space are conserved.

The input to the algorithm is an $n \times n$ matrix which contains the distances δ_{ij} between points (note: distance = inverse proximity). The goal is to fit the distances δ_{ij} with distances d_{ij} in a low-dimensional K-space, e.g. $K = 2$. Then there is a metric. The ansatz is to minimize a cost function:

$$S = \frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j} d_{ij}^2}$$

where d_{ij} is expressed by points in K-space: $d_{ij} = \sqrt{\sum_k^K (x_{ik} - x_{jk})^2} = \sqrt{(\vec{x}_i - \vec{x}_j)^2}$. The cost function S is now minimized w.r.t. the parameters $\vec{x}_i$ by an iterative process: $\vec{x}_i^{new} = \vec{x}_i^{old} - \epsilon * (dS/d\vec{x}_i)$. The solution for the primate visual system is shown in Fig. 5.

A desired property of the results is the following: if distances δ_{ij} and δ_{lm} in the high-dimensional space are similar, then distances d_{ij} and d_{lm} in the low-dimensional space should either be similar. In order to yield this property, the energy function can be complemented by a correction term, if they are not similar. This is termed the "tied approach". Its strongest effect is on the many zero-entries: they “push” each other away. In the solution, thus, areas are arranged on a ring, following from the large proportion of non-connections.

The observation that a few connections pass across the central region, whereas most pass around the rim has led to the interpretation of a dorsal and a ventral stream of processing. Accordingly, information flow re-converges rostrally, in the frontal lobe. Also, from the tied approach it can be nicely seen that early and late visual areas are segregated. In comparison, the untied solution gives a “lower bound”: the least possible degree of segregation between areas.

When tracers were injected into a cortical area in order to observe a certain density of labeling at other cortical areas then it was observed that 70% were not connected. Proximity between most areas is thus low if only a nearest neighbor basis is considered. Connecting only few nearest neighbors of a large distribution on a disk, then one easily arrives at a distribution of connections as on a ring.

Seriation analysis An algorithm is implemented which orders the rows of the connection matrix such that the cumulative mismatch between all neighboring rows is minimized. As a result, all areas align in a line which can be closed to a ring. The algorithm finds multiple orderings because of local minima [73].

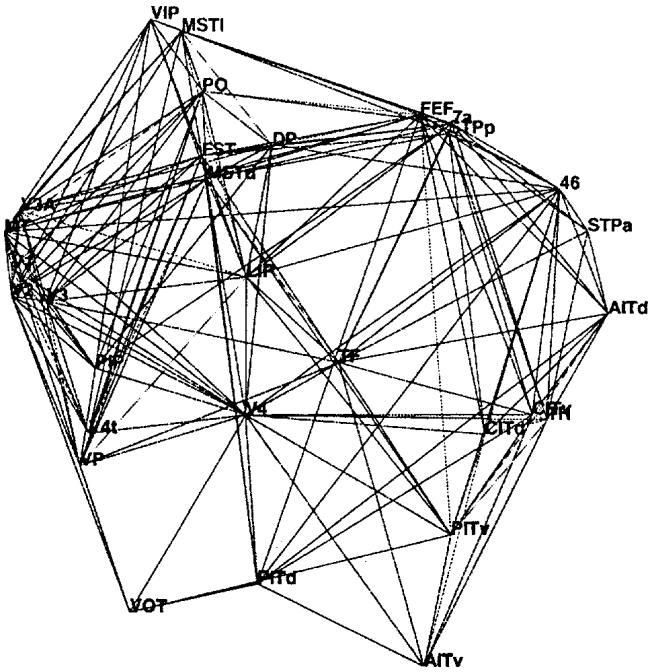


Figure 5: Visualization of areal relationships in the primate visual system through NMDS (untied approach). Lower visual areas are on the left, areas which belong to the dorsal stream are on the top, the ventral stream areas are placed toward the bottom. Figure taken from [73].

Dissimilarity of connection pattern A new distance matrix is defined according to the following criterium: "areas are similar when they are connected to similar areas". I.e. the number of different entries between rows i and j of the connection matrix is summed up. This defines a new distance between areas i and j, which is non-metric. Then, "untied" NMDS can be applied to yield results which will resemble the previous NMDS results [73].

Flattening of cortex Visualization of a curved sphere on a flat paper is not possible without changes of the surface area or angles if no cuts should be introduced. The following procedure provides a good approximation [13]:

1. Sampling of sections of the macaque brain as line drawings and digitizing of contours.

2. Reconstruction: an algorithm connects neighboring nodes. Furthermore, noise of reconstruction and sampling is reduced: nodes are smoothed by a relaxation procedure. Now the section can be displayed as a 3-dimensional lattice.

3. Unfolding: *(i)* reference lengths and angles are saved. *(i)* A recursive flattening procedure is applied where external forces make a node coplanar with its

neighbors while internal, longitudinal and torsional forces maintain similarity to the reference grid. *(ii)* Finally, the nearly flat grid is projected onto a plane "parallel to it".

4. Visualization. The flattened image is of good quality if the distortion ratio for each tile is close to 1:

$$\text{distortion ratio} = \frac{\text{area of a tile in the final map}}{\text{reference area (prior to unfolding)}}$$

2.2 Hierarchy

So far, connections between cortical areas were termed symmetric because if area “I” projects to area “II”, then, in general, area “II” projects to area “I”, too. Upon further scrutiny, this symmetry is broken: the cortical sheet (grey matter) consists of six distinguishable cell layers (Fig. 8). Connections which link areas “I” and “II” may have different layers of origin and different layers of termination (Fig. 6 **a)**). This asymmetry suggests a directional order of information processing. Based on a well established but rough notion of hierarchical information processing, those areas which lie close to the input are arranged at the bottom of the hierarchy projecting “forward” to areas in anterior cortex which are taken as "higher". While in a hierarchical model the vertical dimension is meaningful, the horizontal dimension is still arbitrary.

Acquisition of data The following steps are made to obtain a hierarchy for the macaque connection data:

1. observe individual laminar patterns by labeling techniques. Anterograde labeling: inject at cell bodies, observe tracer chemicals at synaptic targets. Retrograde labeling: axonal processes terminate at injection site, observe tracer chemicals at cell bodies. Typical tracer patterns are shown in Fig. 6 and are described in Tab. 2. The observed connectivity patterns can be irregular across subregions of an area, e.g. patches of columnar pattern can alternate with patches of multilaminar pattern (as in the projection from MT to V4).

2. matching origins and terminations. Here, origin-termination mismatches can be observed, e.g. an **S** pattern (Tab. 2) which terminates in an **M** pattern (as from area 46 to STPp and vice versa).

3. assigning reciprocal relationships. Here, reciprocity mismatches are observed, e.g. a connection which is ascending based on the anterograde pattern but which is descending based on the retrograde pattern. However, 9 of 10 of these irregular cases are questionable due to uncertainties in earlier steps like areal and laminar uncertainties.

Figure 6: Tracer patterns of connections between area "I" and area "II". Each box denotes a slice of the grey matter which includes the six cortical layers, superficial layers at the top. Pyramids denote cell-bodies, the cite of origin of the connections. Stippling denotes fiber terminations, their targets. **a)** Forward (ascending) connections from "I" to "II" and backward (descending) connections from "II" to "I". **b)** Symmetric (lateral) organization. Figure taken from [74].

	anterograde	retrograde
ascending	**F** "layer four, forward" terminations are densest in layer 4 but invade also layer 3	**S** "supra-granular, superficial": at least 70% of labeled cells in supra layers
descending	**M** "multilaminar" avoid layer 4, occasionally preference for superficial layers	**I** "infra-granular": at least 70% of labeled cells in infra layers
lateral/ambiguous	**C** "columnar" fashion, uniform density also within layer 4	**B** "bilaminar": 30-70% of labeled cells in supra and infra layers. This labeling is compatible with all 3 hierarchical relations.

Table 2: Typical tracer patterns as defined in [20].

4. setting up a constraint chart. This table contains information about which area is lower / lateral / higher than others. It is inferred solely on their direct linkages. Erroneous assignments can occur, e.g. TF must be lower than FST, but repositioning would lead to even more inconsistencies.

The resulting hierarchy of the visual system shows 33 linkages with irregularities. In the retrograde labeling studies, patterns of bilaminar origin are less specific and traverse only few levels whereas projections of more specific unilaminar origin traverse more levels. In the anterograde labeling studies, mixed termination patterns (C/M, C/F) traverse only 0.8 levels in average. Thus, structure may be only quasi-hierarchical, especially between “nearby” areas or the methods of anatomical analysis may be noisy. Fig. 7 shows differing results of hierarchical arrangements for the cat visual system from two authors.

Note that single neurons can project to more than one target area, and the number of target areas is greater for descending pathways (cat). While V1 is linked to 9 areas "only", V4 is linked to 21 areas, which are 2/3 of all visual areas.

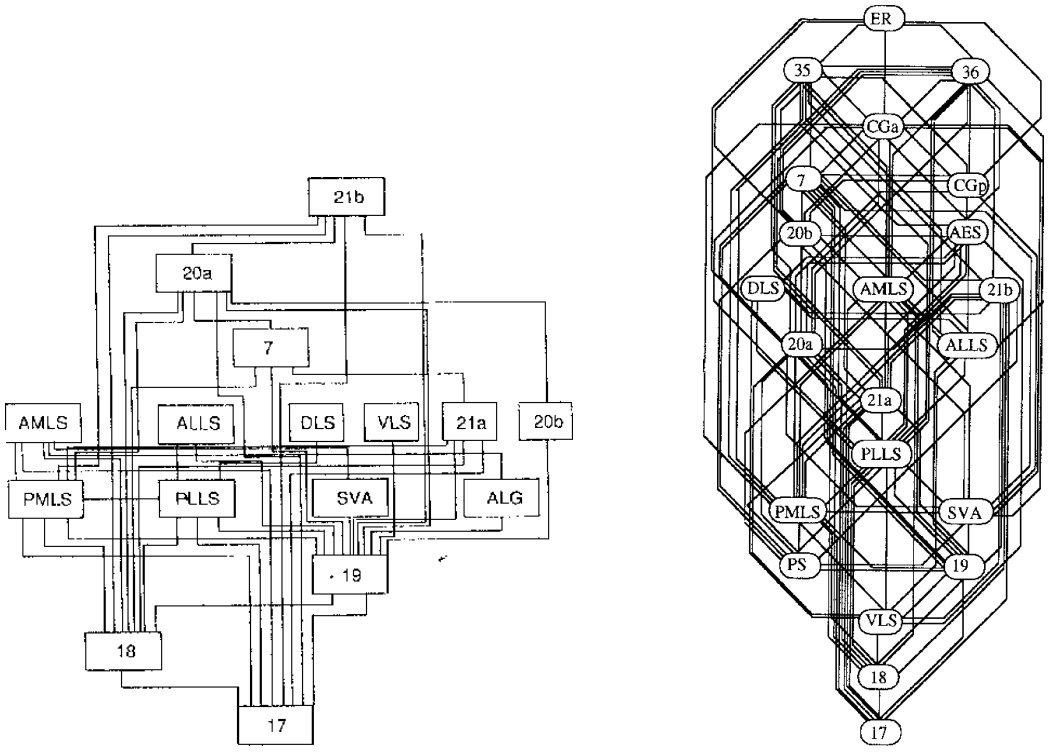


Figure 7: Hierarchy of cat visual areas, taken from **left:** Van Essen (1991) [20] and **right:** Young (1995) [52]. Van Essen depicts 16 areas on 8 hierarchical levels and 62 connections. Young also takes into account the limbic system and depicts 22 areas on 14 hierarchical levels and 224 connections.

Areas which are topographical neighbors are separated by not more than one level.

Cortical circuitry Fig. 8 shows a more detailed model of cortical circuitry (see [39],[71],[10],[50] for reviews). Between hierarchically organized areas we distinguish forward and backward connections. The forward projection exits the lower area in layers 2/3 and enters the higher area in layer 4. The backward projection exits the higher area in layers 5 and 6 and enters the lower area mainly in layer 6 and layers 2/3 (see also Fig. 6).

Within an area the direction of a projection can be derived on the basis of its origin and termination w.r.t. the entries and exits of inter area connections. Input-receiving cells in layer 4C are exclusively (spiny?) stellate cells [J.Lund]. Thus, spiny and smooth stellate cells which project from layer 4 to layers 2/3 are the inner area constituents of the forward projection. An additional processing step which is not depicted in Fig. 8 is situated within layers 2/3: layer 3 is the main recipient zone and projects to layer 2 via all cell types, while mainly layer 2 sends axons to other

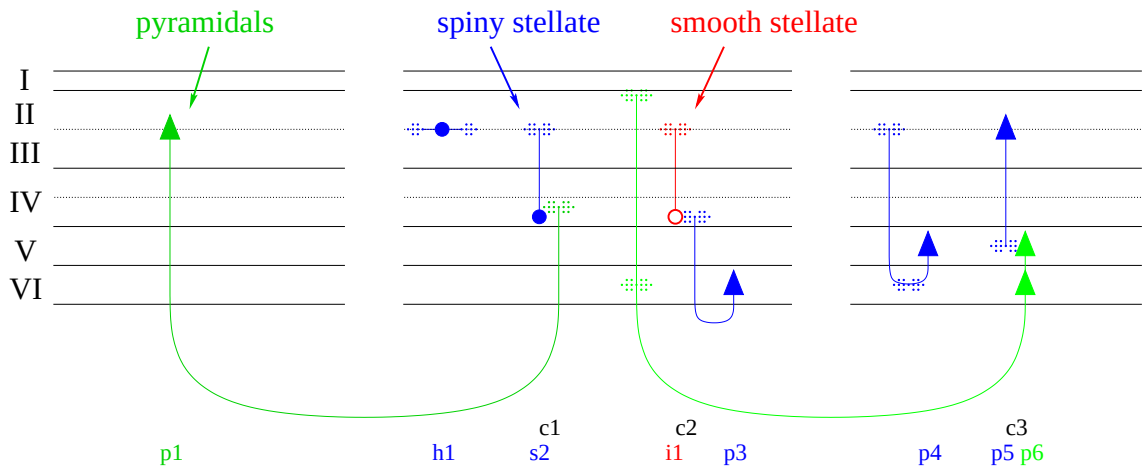


Figure 8: The six cortical layers, cells and connections. Cell types are pyramidal cells and spiny stellate cells, both excitatory, and smooth stellate cells, inhibitory. Three cortical areas are depicted where the hierarchical levels rise from left to right. The hierarchical relationships between the areas are determined by the zones of origin and termination of the long-range connections which consist of pyramidal axons exclusively [20][74]. These are distinguished from spiny stellates by an apical dendrite which points to the surface, whereas the axon shows to the white matter [16]. Horizontal connections (h1) are made up of pyramidal cells as well as spiny stellate cells in layers 2 and 3 [11]. All other features are taken from [35].

areas [J.Lund][1]. The backward projection does not need to be linked within an area, because it both enters and exits an area in layer 6.

In summary, the forward projection covers layers 2/3 and 4 while the backward projection occupies layer 6. Both pathways need to be joined: input from the forward projection into the back projection is mediated mainly through one pathway, pyramidal cells from layers 2/3 to layer 5. The influence of the back projection to the forward projection is mediated via several pathways: *(i)* terminals of the inter area back projection arrive not only in layer 6 but also in layer 2. Within an area, *(ii)* layer 6 pyramidals project to smooth stellate cells in layer 4 and *(iii)* layer 5 pyramidals project to layers 2/3 (with side branches in layer 6).

Areas are laterally connected and thus on the same level of a hierarchy, if the zones of origin and of termination of their mutual connection do not differ (Fig. 6, right). The most prominent lateral connections within one area are within layers 2/3.

As already noted above [20], the hierarchical relation between two areas is rather a continuous property which scales from zero (lateral relation) to a clear hierarchical relation. The expression of hierarchical characteristics is more pronounced with the number of levels traversed: V1 receives few “untypical” back projections from layers 2/3 of V5, but more 2/3 back projections from areas V3 and even more from V2.

[1] personal communication

The proportion of “typical” back projections from layers 5/6 decreases from V5 to V2 [J.Lund]. This renders V2 an area which is more laterally connected to V1 than V5.

Inhibitory cells Inhibitory neurons, as a group, are scattered fairly evenly throughout V1. They comprise smooth and sparsely spined stellate cells. Basket and chandelier cells belong to them named after the shape of their axonal terminations. Inhibitory contacts tend to cluster on soma and proximal dendrites, the localized nature of their axonal distribution often leads to terminations within the area of the cells own dendrites. Chandellier cells each go to axon initial segments of pyramidal cells [J.Lund]. Basket cells inhibit everybody else: other basket cells, pyramidals, other inhibitory inter-neurons [J.Lund]. Bipolar smooth stellate cells can also make excitatory contact with a recipient cell [29]. Inhibitory inter-neurons are faster than pyramidal cells because they have simpler dendrites [J.Lund].

A smooth stellate cell in layer 4 is depicted (i1) which projects to layer 2/3 [35]. Henry [29] depicts a sparsely spinous stellate cell in layers 2/3 which projects to layer 5, Lund [Lund et al.; J.Comp.Neurol.276:1-29,1988] depicts cells in layer 6 which project to layer 4C as well as [Lund, Yoshioka; J.Comp.Neurol.311:234-258,1991] cells in layer 3B/4A which project to layer 4C and 6 and furthermore [Lund; J.Comp.Neurol.257:60-92,1987] cells throughout layer 4B and C which project to layer 6. Furthermore, there are inhibitory cells between the M and the P pathway, namely from layer 4Cβ to 4Cα and vice versa [J.Lund].

Features specific to V1 Fig. 9 shows cortical circuitry which is specific to V1. As it is at the bottom of the cortical hierarchy, it receives bottom-up input and sends top-down projections from/to the thalamus instead of other cortical areas (shown left). Even more striking is that V1, especially layer 4, is thicker than other cortical areas. The M-pathway is a miniature model of the forward projection 4C $\rightarrow$ 2/3 $\rightarrow$ higher areas, but fully situated within layer 4Cα, 4B (Fig. 9, middle and right). Unlike other areas, 4B receives a back projection from higher areas and, reminiscent of layers 2/3, layer 4B hosts extensive lateral connections. Concerning the part of the back projection which arrives at layers 5/6, the M-stream slightly prefers lower layer 6 while the P-stream prefers upper layer 6, but this separation is not clearly distinct [11].

Visual streams Even within single areas, different streams of parallel information processing can be distinguished. The two major visual streams, the fast M stream and the fine P stream, are displayed in Tab. 3. Within the P stream, the color sensitive P-B stream is a further subdivision. However, there is cross talk between the two streams within a single layer, in the ascending or descending connections. E.g. V5 projects back mostly to the thick stripes of V2 (from where it receives input) but also to thin- and inter-stripes.

While the two streams are arranged in parallel (horizontally), each of them form hierarchically advanced stages of processing. The stages are:

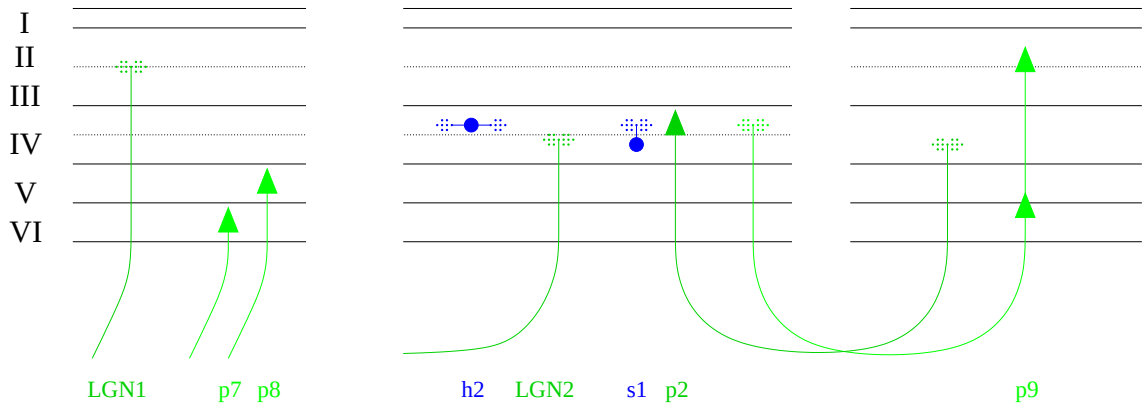


Figure 9: Specific features of V1. On the **left**, connections from and to the thalamus are shown: axons from the interlaminar zones of the LGN project directly to the blobs in layers 2/3 of V1 (LGN1). A pyramidal cell in layer 6 (p7) projects to the LGN or to the claustrum and a pyramidal cell in layer 5 (p8) projects to the superior colliculus, the pons or the pulvinar.
Middle and **right**, the M-pathway: LGN input arrives in layer 4Cα (LGN2). From there, a spiny stellate cell (s1) projects to 4B and a pyramidal cell (p2) in layer 4B projects to higher visual cortical areas. Horizontal connections (h2) in layer 4B spread even further than those in layers 2/3 (h1) [J.Lund]. Feedback connections from layer 2/3 and 5/6 in areas V2, V3 and V5 arrive in layer 4B of V1 (p9). These, too, are the largest back-projections (larger than those to layers 5/6) [J.Lund].

1. Retina: retinal ganglion cells have concentric receptive fields, with an antagonistic center-surround organization that allows them to measure the light intensity in the receptive field center relative to the surround [35].

2. LGN: cells have the same functionality as in the retina but M and P pathways as well as ON-center and OFF-center cells segregate into different layers [35].

3. V1[2]: within this area there is a gradual "sharpening of selectivity for" direction, orientation, length, binocular disparity and spatial frequency. Starting from a weak expression of these properties in the main input layer 4C, they emerge more pronounced in the other layers:

   - There is a small population of direction selective cells in upper 4Cα while the principal population of *direction selective* cells lies in layer 4B [6].
   - Many cells in layer 4C lack orientation selectivity (the proportion of orientation selective cells increases from 4Cβ to 4Cα) while *orientation selective* cells are prominent in all other layers above and below layer 4C (except in

[2]visual area 1 (= Brodmann's area 17) is also called "striate cortex", because it contains a prominent stripe of white matter in layer 4 consisting of myelinated axons from the LGN and other cortical areas [35].

	magnocellular stream	parvocellular stream
retina	M (α) ganglion cells (10%)	P (β) ganglion cells (80%)
LGN	M-layers	P-layers
V1	layer 4Cα $\rightarrow$ 4B	layer 4Cβ $\rightarrow$ 4A, 2/3; segregation into parvo-blob (P-B) / parvo-interblob (P-I)
V2	thick stripes	thin stripes (P-B) / interstripes (P-I)
higher to	V3, MT=V5, MST and V4t, V3A parietal lobe (upper posterior head-wall)	V4; P-B / P-I infero-temporal cortex
properties	high temporal resolution, high conduction velocity, high contrast sensitivity, low threshold, transient (phasic) response	small receptive fields, high spatial resolution, sustained (tonic) response, spectrally opponent receptive fields (color selective)
functions	direction of motion, computing trajectory, perception of *(i)* depth by way of motion parallax *(ii)* shape by way of structure from motion *(iii)* texture by way of dynamic reflectance changes (as in a rippling surface)	P-B: wavelength selective cells; P-I: orientation selective cells $\rightarrow$ pattern and form recognition [74]
damage to impairments	parietal cortex visuo-spatial	temporal cortex recognition
	dorsal stream	ventral stream

Table 3: The two main visual streams. Retina: The remaining 10% of retinal ganglion cells project to the superior colliculus and have different physiological properties [6]. LGN: There are 1.3 million LGN cells, which are roughly as many as afferent retinal ganglion cells. Only 10-20% of the presynaptic connections onto LGN cells are from retinal ganglion cells. The majority are from other subcortical regions and from primary visual cortex [35].

the blobs of layers 2/3) [6]. *Orientation preference* denotes a cells preference to a certain orientation of a visual stimulus. Additionally, quantizing these properties on V1 one finds characteristical patterns such as orientation preference bands and pinwheels and ocular dominance stripes.

- Many orientation selective cells in layer 4B are *simple cells* which have receptive fields clearly defined by ON and OFF zones and which respond to a bar with the edge at a certain position. Layers 2 and 3 contain populations of *complex cells* which distinguish less the position of a bar and may be

considered to integrate the outputs of several simple co-aligned cells [6].

- Cells in layer 4C are almost exclusively monocular (driven by one eye only) while cells in other layers are predominantly *binocular* with some preference to one eye [6].
- Cells in the blobs of layer 2 and 3 are markedly *color specific*, however, these get direct input from the interlaminar LGN cells [6].

4. V2: cells prefer angles and arcs, some significantly, but bars are not significantly preferred [28]; cells are responsive for subjective "virtual" contours (contour bridging gaps).

5. MT: cells are selective for motion of a complex pattern rather than individual components.

6. Infero-temporal cortex: cells are selective for faces or other complex patterns.

During these hierarchical processing stages, receptive field sizes w.r.t. the visual field increase gradually. The degree of topography (retinotopy) hereby decreases gradually.

Somato-sensory hierarchy The somato-sensory system has 13 areas, which is less than one half of the number as in visual cortex and 62 connections. There are, however, 9 levels, which is nearly as many as in the visual cortex. Ascending anterograde patterns within the motor cortex terminate in layer 3 rather than layer 4. It lacks the granular layer 4 characteristics as it does not receive sensory input.

2.3 Development

Cells – genesis The period in which cells are born can be divided into 3 phases: *(i)* symmetrical cell divisions which produce only progenitor cells, *(ii)* progenitors produce post-mitotic neurons and *(iii)* time of (post-mitotic) neuron origin.

While the number of founder population cells is roughly the same in monkey and mouse [14] there is a 360 times larger amplification factor in the monkey to yield the greater sized cortex. This can be explained by more time which the monkey spends for development [48]. The surface of the cortex ranges from 3-5 cm^2 in small insectivores to 1100 cm^2 in humans.

Cells – movement and differentiation – layers The new born cells move along glia cells at approximately 0.1 mm per day. Like an amoeba, the cell attaches to a process somewhere, the nucleus flows towards this point, and so on [16]. The cells always move until they reach the marginal zone. The result is a chronotopic ordering of layers: older layers reside inside, younger ones outside (superficial). The layers together form the cortical plate. Cell differentiation depends on the time at which they

receive signals to become pyramidal, basket or stellate cells. Spiny stellate cells start with an apical dendrite during development (as to become a pyramidal cell), but shed it later [J.Lund]. As a rule of thumb: large cells and those with long axons emerge earlier than the small local ones.

Connections The emergence of a connection between the thalamus and the cortex can be divided into the following phases:

1. Axons sprout, growth cones search target areas. Mechanisms involved are that growth cones stick to adhesive surface [55], and that growth cones follow concentration gradients of chemical markers which is evidenced in the retino-tectal projection [24].

2. The following axons (from other cells) form bundles. Hereby, they maintain their relative positional order. Thalamo-cortical and cortico-thalamic fibers meet on their way. During this "handshake", the oldest axons meet first resulting in a chronotopic ordering [41].

3. Extension of collaterals (vertical and horizontal). Before collaterals enter the cortical plate, there is a waiting period in the subcortical plate where they can interact also with cortico-cortical fibers [48].
   The growth of developing axonal arbors is specific for layers 2/3, 4, 5 and 6 from the outset and apparently dependent on cues intrinsic to the cortex, but sub-laminar specificity arises by later organization only [11].

4. Refinement. Initially exuberant development leads to a later reduction of synapse numbers [34]. 3-5 weeks young macaque spiny stellate cells have many more spines than numbers in the adult which are reached at an age of 9 months [J.Lund]. This could mean an initially higher activation level, however, synaptic efficiacies may be weaker than in the adult or activation may spread out less specifically and thus less effectively.
   Refinement occurs earlier in layer 4 than in layer 2/3 and later in higher, more frontal areas which may reflect a role of the input. Furthermore, refinement occurs earlier in layer 4Cα than in 4Cβ, as M-axons develop earlier than P-axons [J.Lund].

The following features can be regarded as a result of connections: *(i)* topography, *(ii)* areas as defined by the connection matrix and even *(iii)* folding of the cortex and thus its overall form [56].

Areas As a result of connectivity, areas will emerge. The question remains, to which extent this is due to intrinsic or to activity dependent mechanisms.

If by a successful mutation a new sensory organ evolves, but there is no corresponding part on the cortex, then the mutation would not be successful. The mutation would have to happen very often until by chance the cortex mutates accordingly.

Therefore, it seems reasonable that the cortex should always make use of an increase of input dimension. It should act as uniform and universal information processing tissue which adjusts to its input.

Following enucleation, V1 is present but smaller. This implies that V1 is genetically described and additionally, its size is regulated by in-growing fibers. Looking at barrels (which may not be regarded as proper areas), these do not emerge if their input is taken away by cutting the corresponding whisker.

Summary Tables 4 and 5 give a summary on intrinsic and activity dependent mechanisms. We will associate intrinsic mechanisms with early development and activity dependent mechanisms with late development. Key words on a time scale are: cell genesis, migration, differentiation; layers – cell number, synapse number – volume, surface area – connections, waiting period – folding – Hebbian learning.

	intrinsic	activity dependent
what is meant	genetic description	learning
mechanisms	chemical markers	Hebbian learning
target	cell movement and differentiation, connections	connections
when does it appear	early development	late development
reliability	maintainance across species and by manipulations	changed by in-growing axons
result	cortical layers, areas(?)	LGN layers, barrels, OD-patches, areas(?)

Table 4: Intrinsic and activity dependent growth mechanisms. Areas are hardly attributed to one class of mechanisms alone.

3 Theory

3.1 Stochastic neurons

A real neuron responds to its input with a change in its spiking output. Measurements cannot avoid noise. In a rate coding model neuron the spike rate represents the output activity. This leads to a deterministic neuron model: its output $\boldsymbol{u}$ of is a stereotyped response to the neurons net input $\boldsymbol{h}$ such as $\boldsymbol{u} = \boldsymbol{\varphi}(\boldsymbol{h})$. The non-deterministic spiking behavior of a real neuron is lost.

Deterministic neurons are also insufficient in neural network models, if probability distributions have to be set up, i.e. if each network activation state should appear with a certain probability. The flaw of deterministic networks can be envisioned if an

markers seem to be weak	markers seem to be strong
thalamic fibers show no selectivity for certain cortical areas	axons cross the midline
if the input is manipulated then the size of an area changes	but e.g. V1 does not vanish totally
the connection matrix obeys simple principles	areas arise at the same positions (e.g. ventral and dorsal pathway are never inverted)
exuberant growth; connection and cell death of improper connected cells	axons tend to grow in bundles
	direct evidence [2]: gradients of increasing semaphorin concentrations elicit stereotyped responses from cortical growth cones (repulsive, Semaphorin3A and attractive, Semaphorin3C)

Table 5: Evidence for chemical markers to guide the developing connections on the cortex.

energy landscape exists for the activation states: gradient descent then leads to only one state which corresponds to a minimum (which may not be the global minimum).

The use of a stochastic search procedure on this energy landscape has two advantages: *(i)* any state can be reached and a probability for this state may be assigned and *(ii)* activations can escape from a local minimum.

In particular, neurons are able to become active spontaneously, i.e. without any input. This allows for testing (or even training) of the network in the absence of data or other external input. Besides, parameters which control the noise enrich the functional properties of neurons.

3.1.1 Boltzmann distribution I

A recipe to assign different occurrence probabilities to different activation states in a unique way is the following: we assign each state $\vec{s}$ a real number, the energy value $E^{boltz}(\vec{s})$, which will be given in the next paragraph. We determine the probability for each state according to this number, e.g. the larger $E^{boltz}(\vec{s})$ the smaller the probability for the corresponding state.

A unique assignment from energy values to the occurrence probability follows from the following three conditions:

1. The entropy of the distribution $P_{\vec{s}}$ of activation states is maximal:

$$H(\{P_{\vec{s}}\}) \;:=\; -\sum_{\{\vec{s}\}} P_{\vec{s}}\, \ln P_{\vec{s}} \stackrel{!}{=} max \tag{1}$$

   This condition tends to distribute the states broadly.

2. The distribution is normalized:

$$g_1(\{P_{\vec{s}}\}) \;:=\; \sum_{\{\vec{s}\}} P_{\vec{s}} - 1 \;=\; 0 \tag{2}$$

   Demand of these two points already yields a unique solution (homogenous distribution) if the number of activation states is finite.

3. We can mould the shape of the resulting distribution if we impose a further constraint making use of the energy values $E^{boltz}(\vec{s})$. Each state $\vec{s}$ is assigned an energy $E_{\vec{s}}$. The mean energy is given:

$$g_2(\{P_{\vec{s}}\}) \;:=\; \sum_{\{\vec{s}\}} P_{\{\vec{s}\}}\, E^{boltz}(\vec{s}) \;-\; E^{mean} \;=\; 0 \tag{3}$$

   The mean energy E^{mean} certainly can only be chosen to lie within the boundaries of the maximum and the minimum possible energy. We find the homogenous distribution as a solution if $E^{mean} = (E^{max} - E^{min})/2$ and if there is a homogenous prior on states $\vec{s}$ w.r.t. the energy.

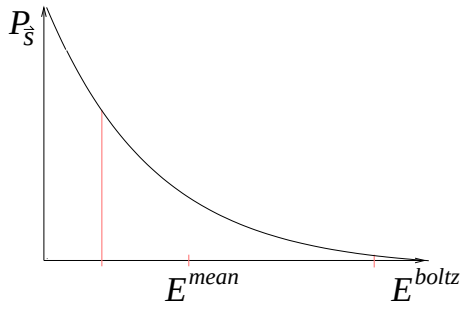


Figure 10: The Boltzmann distribution is an exponential function which is normalized between minimum and maximum energy values (dotted vertical lines).

We use the method of Lagrange multipliers to find the extreme values of Eqn. 1 under the constraints of Eqs. 2,3. It assumes that the gradient of the function which is to be maximized and the gradient of the constraints is parallel:

$$0 \;=\; \vec{\nabla} H \;+\; \lambda_1 \vec{\nabla} g_1 \;+\; \lambda_2 \vec{\nabla} g_2$$

For each coordinate k this yields

$$0 \;=\; -\ln P_k \;-1\; +\; \lambda_1 \;+\; \lambda_2 E_k$$

or

$$P_k = e^{\lambda_1 - 1 + \lambda_2 E_k} =: c e^{\lambda_2 E_k}$$

After normalization (Eqn. 2) we have the maximum entropy distribution:

$$P_k = \frac{e^{\lambda_2 E_k}}{\sum_{k'} e^{\lambda_2 E_{k'}}}$$

or if states $\vec{s}$ are continuous we have:

$$\boxed{p(\vec{s}) = \frac{e^{\lambda_2 E(\vec{s})}}{\int_{\vec{s}'} e^{\lambda_2 E(\vec{s}')}\, d\vec{s}'}} \tag{5}$$

Thus the solution is the Boltzmann distribution (see standard literature, e.g. [25]), as depicted in Fig. 10. The inverse temperature λ_2 we will denote β in the following.

Note that the probability is defined on the states $\vec{s}$ but not on the energy values $E^{boltz}(\vec{s})$. In the case of multiple states with degenerate energy we will observe the corresponding energy value more frequently. In the limit of an infinite number of states (but finite dimensionality) the sum over the states can be replaced by an integral.

Energy function We define the energy function which we use to set up the Boltzmann distribution as follows:

$$E^{boltz}(\vec{s}, W) \;=\; -\frac{1}{2} \sum_i^N \underbrace{\sum_j^N w_{ij}\, s_j}_{h_i}\; s_i \tag{6}$$

The energy is low if large pre-synaptic cell activities s_j and large receipient cell activities s_i are paired with large weights w_{ij}. Corresponding activation states are assigned high probabilities according to the Boltzmann distribution. The energy can be regarded as a sum of single neuron energies: local energies $-h_i s_i$ with $h_i = \sum_j^N w_{ij}\, s_j$ on each neuron.

3.1.2 Boltzmann distribution II

If we want to use the Boltzmann distribution practically, we need a neuron update dynamics which results into states that underly this distribution. Here we show that the Boltzmann distribution as an equilibrium distribution will result from the following two conditions:

- The neuron state transition from state s_i to s_i' occurs according to the difference in the energy ΔE between these two states:

$$P(s_i \to s_i') \;=\; \frac{1}{1+e^{-\beta \Delta E}} \tag{7}$$

  This condition is met by some stochastic transfer functions, while ΔE can be related to a neurons input.

- In equilibrium, the detailed balance principle holds:

$$P(s_i) P(s_i \to s_i') \;=\; P(s_i') P(s_i' \to s_i) \tag{8}$$

  This means that a transition from state s_i to state s_i' can be observed as often as the inverse transition.

Eqn. 8 implies:

$$\begin{aligned} \frac{P(s_i)}{P(s_i')} \;&=\; \frac{P(s_i' \to s_i)}{P(s_i \to s_i')} \\ &\overset{7}{=}\; \frac{1+e^{\beta \Delta E}}{1+e^{-\beta \Delta E}} \;=\; \frac{e^{\beta \Delta E}\left(e^{-\beta \Delta E}+1\right)}{1+e^{-\beta \Delta E}} \;=\; e^{\beta \Delta E} \end{aligned} \tag{9}$$

This is true if

$$P(\vec{s}) \sim e^{-\beta E} = \frac{1}{Z} e^{-\beta E} \tag{10}$$

(which can be seen if Eqn. 10 is inserted into the left hand side of Eqn. 9). The partition sum Z normalizes the probability distribution to the integral of 1:

$$Z = \sum_{\vec{s}} e^{-\beta E}. \tag{11}$$

Summary The distribution of activity states obeys the Boltzmann distribution, if there is an appropriate energy function and if the maximum entropy principle holds. Using a neuronal state transition function as in Eqn. 7, we can set up the Boltzmann distribution with stochastic neurons.

3.2 Generative models

We have the idea of a model which generates its input data. Fig. 11 shows the architecture of a recurrent model. A data point $\vec{x}$ is generated from hidden neuron activities $\vec{u}$ via feedback weights V:

$$\vec{x} \;=\; \vec{f}_x(V\,\vec{u}) \tag{12}$$

where the transfer function f_x is usually assumed to be linear. The generation of the data point $\vec{x}$ can then be interpreted as a linear superposition of "basis functions"

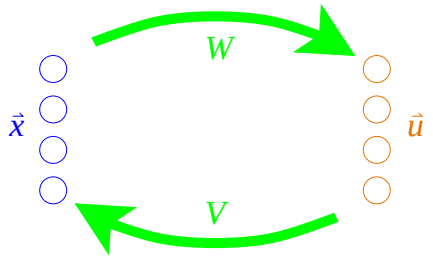


Figure 11: Architecture of a recurrent model. N input units with activations $\vec{x}$ are fully connected to H hidden units which have activations $\vec{u}$. Recurrent weights are depicted as separate feed-forward weights W which set up a recognition model and feedback weights V which set up a generative model. There is no lateral connectivity.

(columns $\vec{w}_i$ of W^T) using coordinates u_i. Weight vectors $\vec{w}_i$ (projective field of hidden neuron i) are interpreted to be possible causes of all data points $\{\vec{x}\}$, while activations u_i determine which causes generate a particular data point $\vec{x}^\mu$.

The feed-forward weights are used to recognize the data and infer the hidden code $\vec{u}$:

$$\vec{u} \;=\; \vec{f}_u(W\,\vec{x}) \tag{13}$$

with various possible forms of f_u.

This simple model does not contain lateral connections among the input units or among the hidden units.

Likelihood The probability for the model to generate this data measures the quality of its representation of the environment. Let $P^-(\vec{x}^\mu)$ denote the probability for the model to generate the data point $\vec{x}^\mu$. (Data may originally be generated by a distribution P^+.) Our model should learn to generate its input data, i.e.:

$$P^-(\{\vec{x}^\mu\}\,|\,V) \;\stackrel{!}{=}\; \text{high.} \tag{14}$$

For a set of statistically independent data points it makes sense to assign a probability to a single data point

$$P^-(\vec{x}^\mu\,|\,V) \tag{15}$$

and the likelihood of generating all data becomes

$$P^-(\{\vec{x}^\mu\}\,|\,V) \;=\; \prod_\mu P^-(\vec{x}^\mu\,|\,V) \tag{16}$$

3.2.1 Bayes theorem

Given: data distribution $\{\vec{x}^\mu\}$ drawn from a distribution $P^+_{\vec{x}}$. The model generates the data using its parameters V. Bayes theorem is:

$$P(V|\{\vec{x}^\mu\}) \;=\; \frac{P(\{\vec{x}^\mu\}\,|\,V)\;P(V)}{P(\{\vec{x}^\mu\})} \tag{17}$$

where $P(V)$ is the prior over the weights, $P(\{\vec{x}^\mu\} \mid V)$ is the likelihood (Eqn. 14), $P(\{\vec{x}^\mu\})$ is the evidence and $P(V \mid \{\vec{x}^\mu\})$ is the posterior probability distribution over model parameters V.

Log-likelihood It is usual to regard the evidence $P(\{\vec{x}^\mu\})$ of Eqn. 17 as a normalization constant and to neglect it. Furthermore, for mathematical convenience, the logarithm is taken to yield:

$$\ln P(V|\,\{\vec{x}^\mu\}) \;=\; \underbrace{\ln P(\{\vec{x}^\mu\} \mid V)}\; + \ln P(V) \tag{18}$$

The underbraced term is the log-likelihood, $\mathcal{L}$. Assuming independence of the generated data points (Eqn. 16), we write:

$$\mathcal{L}(\{\vec{x}\}) \;=\; \sum_\mu \ln P^-(\vec{x}^\mu) \tag{19}$$

For simplicity we will denote this term $P_{\vec{x}}^-$. The minus sign as upper index means that activations $(\vec{u}, \vec{x})$ emerge by some random process intrinsic to the network without influence by the environment. In the following sections we will call this the *free running phase* (Boltzmann machine) or the *sleep phase* (Helmholtz machine). $P(\vec{u}|V)$ can be regarded as a prior probability of a hidden state vector $\vec{u}$ to occur.

Maximum a posteriori With Bayes theorem, one obtains the whole posterior distribution over the model parameters. In practice, however, one is often interested in the "best" network only, i.e.:

$$V^{opt} \;=\; \operatorname*{argmax}_V \Big\{ P(V \mid \{\vec{x}^\mu\}) \Big\} \;=\; \operatorname*{argmax}_V \Big\{ \ln P(V \mid \{\vec{x}^\mu\}) \Big\} \tag{20}$$

3.2.2 Maximum likelihood

The task of a data generating model is to make $P_{\vec{x}}^-$ equal to the true data distribution which we denote $P_{\vec{x}}^+$. In other words, the Kullback Leibler divergence between the two distributions $P_{\vec{x}}^-$ and $P_{\vec{x}}^+$ should be minimized by adjusting the model parameters:

$$R_{P^+\|P^-} \;=\; \sum_\mu \ln \frac{P^+(\vec{x}^\mu)}{P^-(\vec{x}^\mu)} \;=\; \underbrace{\sum_\mu \ln P^+(\vec{x}^\mu)}\; - \sum_\mu \ln P^-(\vec{x}^\mu) \tag{21}$$

The underbraced term is not dependent of model parameters. Thus, for a derivation of the learning rules it will be omitted and in the remaining term we recognize the negative likelihood that will be minimized by learning the parameters.

If activation states are discrete then the sum over all data points $\vec{x}^\mu$ of Eqn. 19 can be replaced by a sum over all possible states $\vec{x}^i$:

$$\mathcal{L}(\{\vec{x}^\mu\}) \;=\; \sum_\mu \ln P^-(\vec{x}^\mu) \;=\; \sum_{\vec{x}^i} P^+(\vec{x}^i)\; \ln P^-(\vec{x}^i) \tag{22}$$

The term $P^+(\vec{x}^i)$ is introduced to weigh the occurrence of data point $\vec{x}^i$ among the dataset $\{\vec{x}^\mu\}$.

In the limit of an infinite number of states we can assume continuous states (also for discrete states and with use of the Dirac distribution):

$$\mathcal{L}(p^+_{\vec{x}}) \;=\; \int_{\vec{x}} p^+(x)\;\ln p^-(\vec{x})\;d\vec{x}$$

This will constitute our energy function for the weights (neglecting the prior), as the weights will be adjusted to shape P^-_x to approach P^+_x. Maximizing this log-likelihood is equivalent to minimizing the Kullback-Leibler divergence between P^-_x and P^+_x.

3.2.3 Helmholtz free energy

In the following we will see that the probability of generating a data point can be expressed as the negative Helmholtz free energy. This will show that the likelihood to generate the true data is under-estimated but not over-estimated if a wrong generative model is used for data generation. In practice, this will provide a substitute for the wake phase of the Boltzmann machine. The Helmholtz machine [17] is defined [32] as a (stochastic) generative model which learns to minimize the free energy.

We will first have a closer look at the Boltzmann distribution for the clamped phase, Eqn. 50, which is the conditional probability of a hidden state $\vec{u}$ given data $\vec{x}$:

$$P(\vec{u}|\vec{x},V) \;=\; \frac{e^{-\beta E(\vec{u},\vec{x},V)}}{\sum_{\vec{u}'} e^{-\beta E(\vec{u}',\vec{x},V)}} \tag{23}$$

Comparing this equation with Bayes rule

$$P(\vec{u}|\vec{x},V) \;=\; \frac{P(\vec{x}|\vec{u},V)\,P(\vec{u}|V)}{\sum_{\vec{u}'} P(\vec{u}'|V)\,P(\vec{x}|\vec{u}',V)} \;=\; \frac{P(\vec{x},\vec{u}|V)}{\sum_{\vec{u}'} P(\vec{x},\vec{u}'|V)}$$

we can define a Helmholtz energy for an activation state $\vec{u}$ as:

$$\beta E^H(\vec{u},\vec{x},V) \;:=\; -\ln\big(P(\vec{x}|\vec{u},V)\,P(\vec{u}|V)\big) \;=\; -\ln P(\vec{x},\vec{u}|V) \tag{24}$$

With

$$e^{-\beta E^H(\vec{u},\vec{x},V)} \overset{24}{=} P(\vec{x},\vec{u}|V) \overset{48,49}{=} \frac{e^{-\beta E^B(\vec{u},\vec{x},V)}}{Z} \;=\; e^{-\beta E^B(\vec{u},\vec{x},V)-\ln Z}$$

it is

$$-\ln Z(V) \;=\; -\beta E^H(\vec{x},\vec{u},V) \;+\; \beta E^B(\vec{x},\vec{u},V).$$

or with temperature T, using $\beta = \frac{1}{T}$:

$$E^H(\vec{x},\vec{u},V) \;=\; E^B(\vec{x},\vec{u},V) \;+\; T\;\ln Z(V)\;. \tag{25}$$

The Helmholtz energy differs from the Boltzmann energy by a constant, $\ln Z$, which does not depend on the state $(\vec{u},\vec{x})$ but which still depends on the weights. E^B corresponds to an unnormalized probability, E^H to a normalized probability.

Analogy to thermodynamics Eqn. 25 is reminiscent of the relation between thermodynamic potentials: the free energy can be obtained from the inner energy by a Legendre transformation w.r.t. the entropy (thermodynamics: entropy $= \boldsymbol{S}$):

$$\begin{aligned} \text{free energy} &= \text{inner energy} - \boldsymbol{T} \cdot \text{entropy} \\ \boldsymbol{F} &= \boldsymbol{U} - \boldsymbol{T} \cdot \boldsymbol{S} \end{aligned} \tag{26}$$

$\boldsymbol{U}(\boldsymbol{s})$, $\boldsymbol{F}(\boldsymbol{T})$ and $\boldsymbol{S}(\boldsymbol{U})$ are thermodynamic potentials, i.e. they are energy functions if they are given as functions of their "natural variables". Taking the partial derivatives of the energy functions w.r.t. the "natural variables", while fixing the values of the other variables, we can obtain the values of all other variables:

$$\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{S}} = \boldsymbol{T} \;, \qquad \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{T}} = -\boldsymbol{S} \;, \qquad \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} = \frac{\mathbf{1}}{\boldsymbol{T}} \;.$$

The other "natural variables" of these potentials are the volume $\boldsymbol{\mathcal{V}}$ and the number of particles in the system. Other state variables are obtained, e.g. the pressure $\boldsymbol{p}$:

$$\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{\mathcal{V}}} = -\boldsymbol{p} \;, \qquad \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\mathcal{V}}} = -\boldsymbol{p} \;, \qquad \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{\mathcal{V}}} = \frac{\boldsymbol{p}}{\boldsymbol{T}} \;.$$

Our neural nets do not have a volume or a pressure but weights $\{\boldsymbol{w_{ij}}\}$ and activations $\{\boldsymbol{s_i}\}$ for which the following relations can be obtained (cf. Eqs. 52,25,57 and also [51]):

$$\frac{\partial \boldsymbol{E^B}}{\partial \boldsymbol{w_{ij}}} = -\boldsymbol{s_i}\,\boldsymbol{s_j} \;, \qquad \frac{\partial \boldsymbol{E^H}}{\partial \boldsymbol{w_{ij}}} = -\boldsymbol{s_i}\,\boldsymbol{s_j} \;, \qquad \frac{\partial \ln \boldsymbol{Z}}{\partial \boldsymbol{w_{ij}}} = \frac{\langle \boldsymbol{s_i}\,\boldsymbol{s_j} \rangle}{\boldsymbol{T}} \;.$$

Note that while that partial derivatives of one of the potentials $\boldsymbol{E^B}$, $\boldsymbol{E^H}$ or $\ln \boldsymbol{Z}$ is taken, the other potentials have to be fixed.

Relation to the log-likelihood Making use of the Helmholtz energy, Eqn. 24, we can reformulate the log-likelihood (Eqn. 19). Omitting dependencies on $\boldsymbol{V}$ for clarity we write:

$$\begin{aligned} \ln \boldsymbol{P}(\vec{\boldsymbol{x}}) &= \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln \boldsymbol{P}(\vec{\boldsymbol{x}}) \\ &= \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln \boldsymbol{P}(\vec{\boldsymbol{u}}) + \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln \boldsymbol{P}(\vec{\boldsymbol{x}}|\vec{\boldsymbol{u}}) \\ &\quad - \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln \boldsymbol{P}(\vec{\boldsymbol{u}}) - \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln \boldsymbol{P}(\vec{\boldsymbol{x}}|\vec{\boldsymbol{u}}) + \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}})\, \boldsymbol{P}(\vec{\boldsymbol{x}}) \\ &= \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln(\boldsymbol{P}(\vec{\boldsymbol{u}})\, \boldsymbol{P}(\vec{\boldsymbol{x}}|\vec{\boldsymbol{u}})) - \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln \frac{\boldsymbol{P}(\vec{\boldsymbol{x}}|\vec{\boldsymbol{u}})\, \boldsymbol{P}(\vec{\boldsymbol{u}})}{\boldsymbol{P}(\vec{\boldsymbol{x}})} \\ &\overset{\text{Eqn. 24}}{=} -\sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}})\, \beta \boldsymbol{E^H}(\vec{\boldsymbol{u}}, \vec{\boldsymbol{x}}) - \sum_{\vec{\boldsymbol{u}}} \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \;\ln \boldsymbol{P}(\vec{\boldsymbol{u}}|\vec{\boldsymbol{x}}) \end{aligned}$$

Using $P_{\vec{u}} = P(\vec{u}|\vec{x}, V)$ for simple notation, we can write:

$$\ln P(\vec{x}|V) \;=\; -\sum_{\vec{u}} P_{\vec{u}}\, \beta E^H_{\vec{u}} \;-\; \sum_{\vec{u}} P_{\vec{u}}\, \ln P_{\vec{u}} \tag{27}$$

or with $\beta = \frac{1}{T}$ and a flipped sign (and the notation: entropy $= H$):

$$-\ln P(\vec{x}|V) \;=\; \langle E^H \rangle_{\vec{u}} \;-\; T \cdot H_{\vec{u}}\ .$$

In analogy to Eqs. 25,26, the left term corresponds to the Helmholtz free energy from thermodynamics. It is the difference between the Helmholtz energy averaged over the hidden code (with input $\vec{x}$) and T times the entropy of the hidden code.

For computational reasons the correct distribution $P(\vec{x}|V)$ which is generated by our generative model may be untractable because all states of the system have to be considered. Instead it may be easier to replace the correct distribution $P_{\vec{u}}$ by an (incorrect) approximation $Q_{\vec{u}}$. In our case, $Q_{\vec{u}}$ will be drawn from a separate recognition model. The difference to Eqn. 27 can be seen from the following:

$$\ln P(\vec{x}|V) \;=\; -\sum_{\vec{u}} Q_{\vec{u}} E^H_{\vec{u}} \;-\; \sum_{\vec{u}} Q_{\vec{u}} \ln Q_{\vec{u}} \;+\; \underbrace{\sum_{\vec{u}} Q_{\vec{u}} \ln \left(\frac{Q_{\vec{u}}}{P_{\vec{u}}}\right)} \tag{28}$$

The underbraced term is the Kullback-Leibler divergence between the correct and the approximated probability distribution. The Kullback divergence is always positive or zero. Thus, if we compute the free energy, Eqn. 27, with the incorrect distribution Q, then we know that the true free energy must be smaller or equal, i.e. we have obtained a lower bound for the true likelihood.

3.2.4 Marginalization

A stochastic net generates data autonomously: an activity distribution on all states can be defined, e.g. with use of the Boltzmann distribution. The marginal distribution which considers only the activations $\vec{x}$ of the input neurons defines the probability of generating any data point by the model.

Another marginal distribution considers only the activations $\vec{u}$ of the hidden neurons. In order to compute the likelihood, Eqn. 14, it is necessary to marginalize over the hidden neuron states, because we are only interested in the occurrence of input activations $\vec{x}$ but not in the "working parameters" $\{\vec{u}^\mu\}$. For example, different states $\vec{u}$ which lead to the same (probabilities for) input unit activities $\vec{x}$ should not appear as separate terms of the likelihood.

Eqn. 15 remains independent of the hidden unit activations $\{\vec{u}\}$ if we marginalize over them:

$$P^-(\vec{x}^\mu \mid V) \;=\; \sum_{\{\vec{u}\}} P(\vec{x}^\mu, \vec{u}|V) \;=\; \sum_{\{\vec{u}\}} P(\vec{x}^\mu|\vec{u}, V)\; P(\vec{u}^\mu|V) \tag{29}$$

or in an integral version:

$$P^-(\vec{x}^\mu \mid V) \;=\; \int_{\vec{u}} P(\vec{x}^\mu \mid \vec{u}, V)\; P(\vec{u}\mid V)\; d\vec{u}$$

where $P(\{\vec{u}\})$ is the prior over the model states.

The learning procedures will have to take into account the different natures of the model parameters. The model states $\{\vec{u}\}$ depend on the weights V, W and their distribution p_x^- will change instantaneously if the weights are changed. Therefore, each time that weights are changed in a slow dynamics, the necessary terms such as p_x^- have to be computed by a fast dynamics.

Marginalization maximization In practical situations we often want to find the hidden variable distribution which maximizes the probability to generate a certain data point. For simplicity, the hidden state distribution, Eqn. 29, can be approximated by the contribution of the most effective value only:

$$\max P^-(\vec{x}^\mu \mid V) \;\approx\; \max_{\vec{u}} P(\vec{x}^\mu \mid \vec{u}, V)\; P(\vec{u})$$

The hidden state where the likelihood is maximal is:

$$\begin{aligned} \vec{u}^{opt,\mu} \;&:=\; \arg\max_{\vec{u}} P(\vec{x}^\mu \mid \vec{u}, V)\; P(\vec{u}) \\ &=\; \arg\max_{\vec{u}} (\ln P(\vec{x}^\mu \mid \vec{u}, V) \;+\; \ln P(\vec{u})) \end{aligned} \tag{30}$$

Note that here, even in a purely generative model, a hidden state $\vec{u}^{opt,\mu}$ is dependent on the data index μ. A neural net architecture which considers this dependency thus needs (in addition to generative weights) recognition weights which mediate the influence of the data towards the hidden units.

Drawbacks of "Marginalization maximization" To approximate the integral over the hidden code, Eqn. 30, means that an incorrect hidden variable distribution is used to generate the data. This insight motivates the use of the "Helmholtz approximation", Eqn. 28, to estimate the resulting loss in the likelihood [44]. In case of a factorial distribution,

$$Q(\vec{u}) \;=\; \prod_i Q_i(u_i) \tag{31}$$

the entropy becomes

$$H_Q \;=\; \sum_i H_{Q_i} \;=\; -\sum_i \int Q_i(u_i)\; \ln Q_i(u_i)\, du_i$$

Use of Eqn. 30 means that $Q_i(u_i) = \delta(u_i - u_i^{opt})$ and thus

$$H_Q \;\approx\; -\sum_i \ln Q_i(u_i^{opt})$$

the entropy becomes minus infinity which reflects the fact that the volume information of the distribution is lost.

3.3 Maximum entropy

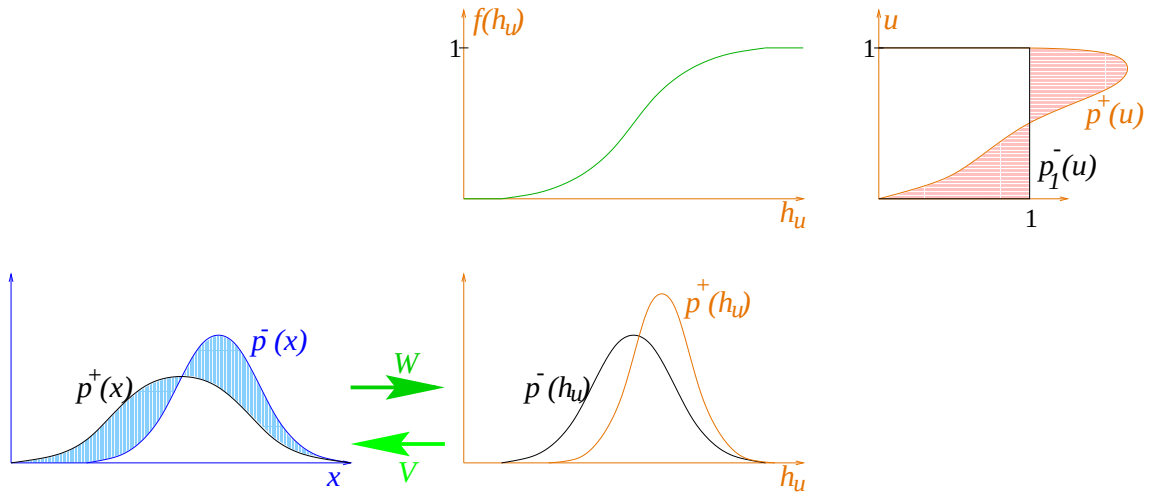


Figure 12: Probability distributions of the data, $\boldsymbol{p(x)}$, the hidden variables, $\boldsymbol{p(h_u)}$, and the second hidden variables, $\boldsymbol{p(u)}$. The transformation from data space to hidden variable space is done via $\boldsymbol{W}$, and backwards via $\boldsymbol{V}$, whereas the transformation from hidden variable space to the second hidden variable space is done via the transfer function $\boldsymbol{f(h_u)}$. If the parameters are optimal and if the prior $\boldsymbol{p^-(h_u)}$ can be mapped to the data $\boldsymbol{p^+(x)}$ by a linear transform, then the black, thick curves are transformed to each other and the colored, thin curves do not exist.
In the maximum-entropy case, if the parameters $\boldsymbol{W}$ are **not** optimal, then $\boldsymbol{p^+(x)}$ will be mapped to $\boldsymbol{p^+(h_u)}$ and $\boldsymbol{p^+(u)}$ which does not have the maximum entropy. The difference to the optimal distribution $\boldsymbol{p^-(u)}$ is indicated by the horizontally striped area and contributes to the Kullback divergence in Eqn. 32.
In the maximum-likelihood case, if the parameters $\boldsymbol{V}$ are **not** optimal, then $\boldsymbol{p^-(h_u)}$ will be mapped to $\boldsymbol{p^-(x)}$ which does not have maximum likelihood. Wrong predictions of $\boldsymbol{p^-(x)}$ are indicated by the vertically striped area and contribute to the Kullback divergence in Eqn. 34.

Maximum likelihood is only one possible basis for unsupervised learning. Another idea is to maximize the entropy of the hidden variable activity distribution. This seems reasonable in order to avoid redundancy and use resources as efficient as possible. In the following we will demonstrate the close relation between these two paradigms.

The infinitely broad homogenous distribution has maximum entropy. Thus we should not maximize the entropy of a variable that may grow infinitely large. For this reason, the inner activation $\boldsymbol{h_u}$ of a hidden neuron is fed through a bounded transfer function to obtain its outer activations $\boldsymbol{u}$. Therefore the transfer function may in this context be termed a “squashing function”. This mapping is based on the prior $\boldsymbol{p^-(h_u)}$

and is defined by

$$u \;=\; f(h_u) \;:=\; \int_{-\infty}^{h_u} p^-(h'_u)\, dh'_u$$

which ensures $0 \leq u \leq 1$. Samples taken from the exact prior distribution $p^-(h_u)$ will be mapped onto a homogenous distribution $p_1^-(u)$:

$$p^-(h_u) \;\overset{u=f(h_u)}{\mapsto}\; p_1^-(u) \;:=\; \begin{cases} 1 & 0 \leq u \leq 1 \\ 0 & \text{else} \end{cases}$$

The distribution $p_1^-(u)$ is bounded on the interval $[0,1]$ (hypercube) and has the maximum possible entropy given this bound (which is zero).

To minimize the negative entropy H_u is equivalent as to minimize the Kullback divergence between $p^+(u)$ and $p_1^-(u)$

$$-H_u \;=\; \underbrace{\int_0^1 p^+(u)\, \ln p^+(u)\, du}\; - \int_0^1 p^+(u)\, \ln p_1^-(u)\, du \;=\; K(p^+(u)||p^-(u)) \tag{32}$$

because the right term (not underbraced) is zero (because $p_1^-(u) = 1$ in the interval $[0,1]$). The hidden and second hidden variable distributions can be transformed into each other:

$$p(h_u)\, dh_u \;=\; p(u)\, du$$

which implies the transformations

$$p(h_u) \;=\; p(u)\, \frac{du}{dh_u} \;=\; p(u)\, \frac{d\, f(h_u)}{dh_u}\,, \qquad p(u) \;=\; p(h_u)\, \frac{dh_u}{du} \;=\; p(h_u)\, \frac{d\, f^{-1}(u)}{du}$$

The transformations in both directions, $f^{-1}(u)$ and $f(h_u)$, are strictly monotonously increasing functions, thus minimizing Eqn. 32 means to minimize the Kullback divergence in hidden variable space

$$K(p^+(h_u)||p^-(h_u)) \;=\; \int p^+(h_u)\, \ln p^+(h_u)\, dh_u \;-\; \int p^+(h_u)\, \ln p^-(h_u)\, dh_u \tag{33}$$

Input- and hidden variable distributions are transformed into each other:

$$p(x)\, dx \;=\; p(h_u)\, dh_u$$

by linear transformations

$$p(x) \;=\; p(h_u) \left|\frac{dh_u}{dx}\right| \;=\; p(h_u)\, |W|\,, \qquad p(h_u) \;=\; p(x) \left|\frac{dx}{dh_u}\right| \;=\; p(x)\, |V|$$

We can thus express the Kullback divergence in data space (Eqn. 21):

$$K(p^+(x)||p^-(x)) \;=\; \int p^+(x)\, \ln p^+(x)\, dx \;-\; \underbrace{\int p^+(x)\, \ln p^-(x)\, dx} \tag{34}$$

This was shown to be equivalent to maximizing the likelihood because the left term (not underbraced) is not affected by the model parameters.

In summary, the underbraced terms of Eqs. 32 and 34 denote what is maximized in each paradigm. Maximum likelihood means to shape $p^-(x)$ as close as possible to the true data distribution (Eqn. 21) using the prior $p^-(h_u)$ (this implies that $\{u\}$ obeys the hypercube distribution $p_1^-(u)$):

$$p^-(h_u) \xrightarrow{V} p^-(x) \overset{!}{\approx} p^+(x)$$

The parameters to optimize are those of a generative model, V.

In contrast, maximum entropy means to shape $p^+(h_u)$ as close as possible to the chosen prior using the data:

$$p^+(x) \xrightarrow{W} p^+(h_u) \overset{!}{\approx} p^-(h_u)$$

This is equivalent to shape $p^+(u)$ as close as possible to the hypercube distribution $p_1^-(u)$ which means at the same time that the entropy of $p^+(u)$ will be maximized. The parameters are those of a recognition model, W.

4 Models

This section describes how the theory is applied in the form of computational models. A variety of models can be derived from one theory like the maximum likelihood theory. The way down from an abstract theory to a detailed model implementation allows many possibilities:

1. Approximations within the theory. Instead of computing the whole posterior distribution over model parameters it is usual to select the parameters with maximum posterior probability only. This is usually done for the weights (as in all our models) as well as the hidden code (as in the Kalman filter model).

2. Different internal models. This is expressed in different architectures, e.g. some models assume a hierarchical organization, some do not. In particular, weight symmetry is introduced in order to make the recognition model and the generative model dependent (Kalman filter model, Boltzmann machine).

3. Approximations within the computation. The optimal hidden code within the recurrent models described here can only be obtained by iterative computations. These have to be prematurely terminated. Even more, the mean-field approximation involves the use of deterministic neurons instead of stochastic neurons.

4. Different parameterization. Finally, there is a choice of parameters. Some affect the priors which are used for the model parameters, some affect the learning procedure.

Thus, from the maximum likelihood theory one can arrive at a variety of algorithms which can be used for various purposes. It is desirable to make approximations (items 1 and 3) as weak as possible so that different approximations do not lead to different results. Then one can study the effect of different internal models (item 2) as well as different parameters (item 4) on the internal representation of data which may also provide further insight into the structure of the data.

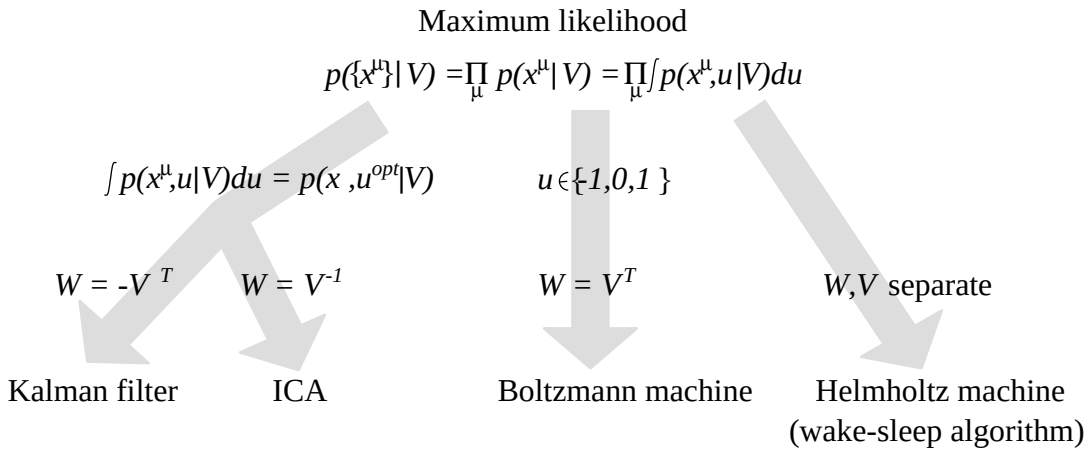


Figure 13: Approximations used to arrive at different models.

Fig. 13 displays how to arrive from the maximum-likelihood equations 16,29 to the models used here.

4.1 Non-linear Kalman filter model

Instead of correctly marginalizing over the hidden states according to Eqn. 29, this model[3] takes only the optimal hidden state, Eqn. 30, which maximizes the posterior probability of an observed data point. Also, only the most probable weights for the data set are interesting, Eqn. 20.

Using these approximations, Eqs. 18,19 yield:

$$\boldsymbol{V}^{opt} = \underset{\boldsymbol{V}}{\text{argmax}} \left(\sum_{\mu} \ln \underbrace{\int_{\vec{u}} \boldsymbol{P}(\vec{x}^{\mu} \mid \vec{u}, \boldsymbol{V})\, \boldsymbol{P}(\vec{u})\, d\vec{u}}_{\boldsymbol{P}^{-}(\vec{x}^{\mu} \mid \boldsymbol{V})} + \ln \boldsymbol{P}(\boldsymbol{V}) \right)$$

If the underbraced term, $\boldsymbol{P}^{-}(\vec{x}^{\mu} \mid \boldsymbol{V})$, is approximated using the optimal hidden vector

[3]With a Gaussian prior over the hidden states the model was related to a Kalman filter [49]. The dynamics of a Kalman filter optimally estimates a current system state $\vec{u}$ which relate to an observed process through $\vec{x} = \boldsymbol{A}\vec{u} + \boldsymbol{e}$, where $\boldsymbol{e}$ is Gaussian noise. Because of the non-Gaussian prior on the recognition states $\vec{u}$, we term our model a *nonlinear* Kalman filter. (Inofficially, it is known in the community as the "Olshausen-model".)

only, Eqn. 30, then we have

$$\boldsymbol{V}^{opt} = \underset{\boldsymbol{V}}{\text{argmax}} \left(\sum_{\mu} \ln \boldsymbol{P}(\vec{\boldsymbol{x}}^{\mu} \mid \vec{\boldsymbol{u}}^{opt,\mu}, \boldsymbol{V}) + \ln \boldsymbol{P}(\boldsymbol{V}) \right) \tag{35}$$

Eqs. 30 and 35 lead to a 2-step learning procedure:

- fast dynamics (activity relaxation): for each pattern $\boldsymbol{\mu}$ find $\vec{\boldsymbol{u}}^{opt,\mu}(\vec{\boldsymbol{x}}^{\mu})$ which maximizes the *likelihood* to generate $\vec{\boldsymbol{x}}^{\mu}$ at fixed $\boldsymbol{V}$.
- slow dynamics (weight learning): with every $\vec{\boldsymbol{u}}^{opt,\mu}(\vec{\boldsymbol{x}}^{\mu})$ found by fast dynamics modify $\boldsymbol{V}$ slowly to find $\boldsymbol{V}^{opt}$.

Even though the ansatz was purely generative, the hidden code $\vec{\boldsymbol{u}}^{opt,\mu}(\vec{\boldsymbol{x}}^{\mu})$ depends on the input. For this reason we need a recognition model with parameters $\boldsymbol{W}$. With the simple model architecture of Fig. 11, we set $\boldsymbol{W} = \boldsymbol{V}^{\boldsymbol{T}}$.

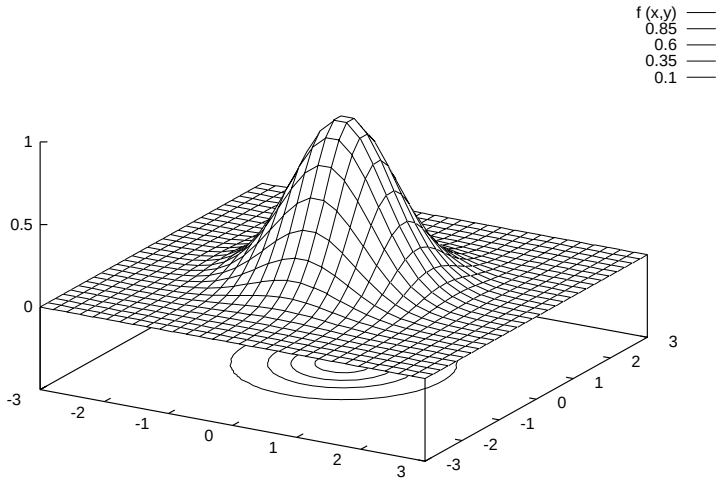


Figure 14: The *Likelihood* function 36 (z-axis) assumes Gaussian noise on the inputs. x- and y-axis denote the error $\vec{\boldsymbol{x}}^{\mu} - \vec{\boldsymbol{x}}(\vec{\boldsymbol{u}}^{\mu}, \boldsymbol{V})$ in the reconstruction of the data point along two arbitrarily selected input dimensions. The *Likelihood* is maximal where the error is small.

The Learning Rules Neglecting the priors over the model parameters, we will first consider the terms only which satisfy the *likelihood*. The learning rules for activities and weights depend on the chosen probability functions. The *Likelihood* function assumes Gaussian noise on inputs (Fig. 14). This leads to a punishment according to the squared distance.

$$\boldsymbol{P}(\vec{\boldsymbol{x}}^{\mu} \mid \boldsymbol{V}, \vec{\boldsymbol{u}}^{\mu}) \approx \frac{\boldsymbol{1}}{\boldsymbol{Z}} \boldsymbol{e}^{-\beta \boldsymbol{E}^{\mu}(\vec{\boldsymbol{x}}^{\mu}, \vec{\boldsymbol{u}}^{\mu}, \boldsymbol{V})} \tag{36}$$

where

$$E^\mu(\vec{x}^\mu, \vec{u}^\mu, V) \;:=\; \frac{1}{2}(\vec{x}^\mu - \vec{x}(\vec{u}^\mu, V))^2$$

and $Z = (\frac{2\pi}{\beta})^{N/2}$ is a normalizing constant.

On-line learning by gradient ascent yields connectionist neuron like activation and learning:

- fast dynamics: find $\vec{u}^{opt}(\vec{x}^\mu)$. The activity u_i of hidden neuron i (e.g. a cell at V1 location i) is modified iteratively according to:

$$\Delta u_i \;=\; \epsilon_u \frac{\partial}{\partial u_i} \ln\, P(\vec{x}^\mu \mid V,\, \vec{u}^\mu) \;=\; \epsilon_u \underbrace{\beta \sum_{j'} (x^\mu_{j'} - x_{j'}(V, \vec{u}^\mu)) \cdot \mathsf{v}_{j'i}}_{\text{feed-forward activation}} \quad (37)$$

  or in vector notation (with $W = V^T$):

$$\Delta \vec{u} \;=\; \epsilon_u\, \beta (\vec{x}^\mu - \vec{x}(V, \vec{u}^\mu)) W$$

  $\vec{u}$ is modified until the reconstruction error $(\vec{x}^\mu - \vec{x}(V, \vec{u}^\mu))$ is minimized by $\vec{u}^{opt}(\vec{x}^\mu)$. The activation update corresponds to the feed-forward activation of a connectionist neuron, with small deviations: *(i)* the input is not the data but the prediction error on the input, *(ii)* the activation update is not performed in one pass but instead, incremental.

- slow dynamics: find V^{opt} using $\vec{u}^{opt}(\vec{x}^\mu)$ which is obtained after completion of the fast dynamics:

$$\Delta \mathsf{v}_{ji} \;=\; \epsilon_{\mathsf{V}} \frac{\partial}{\partial \mathsf{v}_{ji}} \ln\, P(\vec{x}^\mu \mid V,\, \vec{u}^{opt,\mu}) \;=\; \epsilon_{\mathsf{V}} \underbrace{\beta\, (x^\mu_j - x_j(\vec{u}^{opt,\mu}, V))}_{\text{post}} \cdot \underbrace{u_i^{opt,\mu}}_{\text{pre}} \quad (38)$$

  This rule is a Hebbian learning rule which may ideally be repeated until the reconstruction error $x^\mu_{\vec{\alpha}} - x_{\vec{\alpha}}(\vec{u}^{opt,\mu}, V) = 0$.

The slow dynamics optimizes the weights w.r.t. the solution $\vec{u}^{opt}(\vec{x}^\mu)$ of the fast dynamics no matter whether this is really the optimal solution. This is a desirable property if the fast dynamics can find only an approximate solution, e.g. through an interrupted relaxation procedure. The major drawback of this model is shown in the following.

Drawbacks of "Marginalization maximization" In order to see what gets lost when the integral over the hidden code is approximated by the optimal value only, let us rearrange the quadratic terms of the mean expected energy, Eqn. 36, which is

to be minimized [44]. For simplicity, we let $q(u) = Q_i(u_i)$ (cf. Eqn. 31) and regard the hidden code and the data as scalars:

$$\begin{aligned}
\langle E \rangle &= \langle (x - Vu)^2 \rangle = \int q(u)(x - Vu)^2 du \\
&= x^2 - 2\,x\,V \int q(u)\,u\,du + V^2 \left(\int q(u)\,u\,du\right)^2 + V^2 \int q(u)\,u^2\,du \\
&\quad - 2\,V^2 \left(\int q(u)\,u\,du\right)^2 + \underbrace{\int q(u)\,du}\; V^2 \left(\int q(u)\,u\,du\right)^2 \\
&= (x - V \int q(u)\,u\,du)^2 + V^2 \int q(u)\,(u - \int q(u')\,u'\,du')^2\,du \\
&= (x - V\,\langle u \rangle)^2 + V^2\,\langle (u - \langle u \rangle)^2 \rangle \qquad (39)
\end{aligned}$$

With $Q_i(u_i) = \delta(u_i - u_i^{opt})$ we have $u = \langle u \rangle$ and the right term of the right hand side of Eqn. 39 is zero. Without this approximation, the variance on the hidden activations u given a data point would not vanish and the latter term would punish large weights V. There would be a growth limiting term for the weights. Because this term is missing in the approximation, it will be necessary to introduce a separate weight decay term.

4.2 A hierarchical Kalman filter model

In a generative model with two hierarchical levels, the activations $\vec{u}_1$ of the first hidden layer are generated by the activations $\vec{u}_2$ on the second hidden layer. Reminiscent of Eqn. 29, we have

$$P(\vec{u}_1|V_{12}) = \int P(\vec{u}_1|\vec{u}_2)\,P(\vec{u}_2)\,d\vec{u}_2$$

where $P(\vec{u}_2)$ is the prior over the activations on the second level. Together with a prior over activations on the first level, $P(\vec{u}_1)$, which is independent of the other terms, we obtain

$$P(\vec{u}_1|V_{12}) = P(\vec{u}_1) \int P(\vec{u}_1|\vec{u}_2)\,P(\vec{u}_2)\,d\vec{u}_2$$

The likelihood to generate a data point $\vec{x}^\mu$ then is:

$$P(\vec{x}^\mu|V) = \int P(\vec{x}^\mu|\vec{u}_1)\,P(\vec{u}_1) \int P(\vec{u}_1|\vec{u}_2)\,P(\vec{u}_2)\,d\vec{u}_2\;d\vec{u}_1 \qquad (40)$$

For computational reasons, the integrals may be evaluated at the corresponding maximal values, as in Eqn. 30. Then, the inner integral (over $\vec{u}_2$) can only be maximized if $\vec{u}_1$ is known. In order to maximize the outer integral (over $\vec{u}_1$), however, the inner integral must be known.

An approximation to resolve this dilemma, is to collect the single terms without the integrals and maximize them together using to an iterative procedure to find the

optimal hidden code $(\vec{u}_1^{\,opt,\mu}, \vec{u}_2^{\,opt,\mu})$. We adapt the joint vector $(\vec{u}_1, \vec{u}_2)$ to perform gradient descent on an energy landscape obtained by taking the negative logarithm of the single terms:

$$\Delta(\vec{u}_1, \vec{u}_2) \;=\; \frac{\partial}{\partial(\vec{u}_1, \vec{u}_2)} \left(\ln P(\vec{x}^{\,\mu}|\vec{u}_1) \;+\; \ln P(\vec{u}_1) \;+\; \ln P(\vec{u}_1|\vec{u}_2) \;+\; \ln P(\vec{u}_2) \right)$$

This leads to the following system of equations which has to be solved and which is stationary if the maximum values have been found:

$$\begin{aligned} \dot{\vec{u}}_1 &= \beta\, \boldsymbol{W}_{10} \tilde{\boldsymbol{x}}_0 \;+\; \frac{\partial}{\partial \vec{u}_1} \ln P(\vec{u}_1) \;+\; (1-\beta)\, \underbrace{(\boldsymbol{V}_{12}\vec{u}_2 - \vec{u}_1)} \\ \dot{\vec{u}}_2 &= \underbrace{(\vec{u}_1 - \boldsymbol{V}_{12}\vec{u}_2)} \;+\; \frac{\partial}{\partial \vec{u}_2} \ln P(\vec{u}_2) \end{aligned} \tag{41}$$

where $\tilde{\boldsymbol{x}}_0 = \vec{x}^{\,\mu} - \boldsymbol{V}_{01}\vec{u}_1$ and $\boldsymbol{W}_{10} = \boldsymbol{V}_{01}^{T}$. The logical information flow in these equations starts at the input with the data $\vec{x}^{\,\mu}$. This corresponds to determining $\vec{u}_1^{\,opt,\mu}$ before $\vec{u}_2^{\,opt,\mu}$ in Eqn. 40 because the representation on the first hierarchical level depends more directly on the data. In this spirit, in the system of equations 41, $\vec{u}_1$ will be replaced by $\boldsymbol{W}_{10}\tilde{\boldsymbol{x}}_0$ where it communicates with the higher level, i.e. in the underbraced terms.

4.2.1 Relaxation dynamics in a flexible hierarchy

In the case of a hierarchical model, the hidden representation spans more than one layer. Several layers are arranged hierarchically [49][36]. In a typical maximum likelihood setting, the activation of a given neuron then not only adjusts to minimize the reconstruction error on the lower level but also to match the feedback ("prediction") from a higher level representation.

Concatenation of levels In a model for the evolution of a two-level hierarchy [65] all hidden neurons have to choose whether to take the computational role of the first hidden layer, the second hidden layer, or both. Here we concatenate the first and the second hidden layer which renders weights between them lateral weights (Fig. 15). Different activations on these units, however, can belong logically either to the first hidden layer, $\vec{u}_1$, or to the second hidden layer, $\vec{u}_2$. An activity-dependent weight constraint introduces competition between all incoming weights of a hidden neuron (besides preserving sparse coding). This encourages a hidden neuron to receive input from the input neurons via $\boldsymbol{W}_{10}$ or from other lateral neurons via $\boldsymbol{W}_{11}$ but not both.

Distinguishing the modules Our idea is that neurons possess different intrinsic functional properties in different regions of the cortex. Upon learning, the neurons should code for those elements of the data which its function is best adapted to.

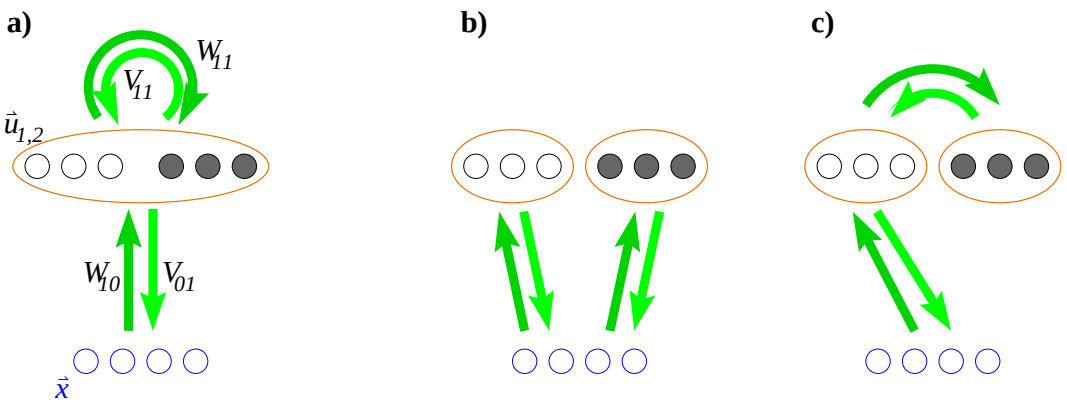


Figure 15: Three different model architectures. In each of them the activations $\vec{x}$ on the input units are represented by hidden unit activations $\vec{u}$. W are recognition weights, V are generative weights, indexed with the number of the layer of termination and origin. **Left:** architecture of our model. The lateral weights W_{11} and V_{11} (top) allow each hidden neuron to take part in a representation $\vec{u}_1$ on a lower and $\vec{u}_2$ on a higher hierarchical level. Dependent on the structure of the data training will result in one of the two other architectures shown: **middle:** parallel organization and **right:** hierarchical organization of hidden units which are segregated into two areas.

In our net the hidden neurons are distinguished into two classes, one with highly active neurons, the other with sparsely active neurons. This property is scaled by a real valued parameter of the neuronal transfer function (see Fig. 39, below). Hereby neurons are expected to specialize on data elements which are less or more sparsely distributed.

Relaxation of activations In each training step, a data point $\vec{x}$ is presented on the input layer and hidden unit activations are initialized with zero. Then activations relax iteratively until a sufficiently good hidden representation on all layers has been found.

In order to implement Eqs. 41, the following computations are done in the input, the logically first hidden layer and the logically second hidden layer. First, the negative reconstruction error $\tilde{\boldsymbol{x}}_0$ in the input neurons is computed as the difference between the data $\vec{x}$ and the reconstruction from the hidden code $\vec{u}_1$ via feedback weights V_{01}:

$$\tilde{\boldsymbol{x}}_0(t) \;=\; \vec{x} \;-\; V_{01}\vec{u}_1(t)\,. \tag{42}$$

The negative reconstruction error measured in the logically first layer is the difference between the bottom-up input and the top-down reconstruction:

$$\tilde{\boldsymbol{x}}_1(t) \;=\; W_{10}\tilde{\boldsymbol{x}}_0(t) \;-\; V_{11}\vec{u}_2(t)\,. \tag{43}$$

We term the top-down weights V_{11} instead of V_{12} because they are connections within one layer of neurons. Analogeously, we use $W_{11} = V_{11}^T$ instead of W_{21}. Activities $\vec{u}_1$ and $\vec{u}_2$, however, have to be separated even though they are also on the same layer.

The hidden code vector $\vec{u}_1$ is the hidden units' representation of the data on the first hierarchical level. It is adjusted *(i)* to account for the negative error $\tilde{x}_0$ via recognition weights W_{10} and *(ii)* to account for the prediction from the next higher level activations $\vec{u}_2$ via generative weights V_{11}:

$$\begin{aligned} \vec{h}_1(t) &= \vec{u}_1(t) + \varepsilon_u(\beta\, W_{10}\tilde{x}_0(t) - (1-\beta)\,\tilde{x}_1(t)) \\ &= \vec{u}_1(t) + \varepsilon_u((2\beta-1)\, W_{10}\tilde{x}_0(t) + (1-\beta)\, V_{11}\vec{u}_2(t)) \\ \vec{u}_1(t+1) &= \vec{f}(\vec{h}_1(t)) \end{aligned} \tag{44}$$

where ε_u is the update step size, β handles the tradeoff between bottom-up and top-down information and f is the the transfer function which transfers the inner activations $\vec{h}_1$ to the hidden code at the next time step $t+1$.

The hidden code $\vec{u}_2$ is the hidden units' representation on the logically second hierarchical level. It adjusts to the code $\vec{u}_1$ on the first level via the lateral recognition weights W_{11} but has no feedback from a higher hierarchical level.

$$\begin{aligned} \vec{h}_2(t) &= \vec{u}_2(t) + \varepsilon_u\, W_{11}\tilde{x}_1(t) \\ \vec{u}_2(t+1) &= \vec{f}(\vec{h}_2(t))\,. \end{aligned} \tag{45}$$

The transfer function $f_i(h_i)$ for a hidden neuron i adds the corrections which are enforced by the priors $P(\vec{u}_1)$ and $P(\vec{u}_2)$ (see section "Priors", Eqn. 69). After relaxation of Eqs. (42,43,44,45) towards a stationary state we have found the optimal code $\vec{u}_{1,2}$ to reconstruct the data point, under a sparseness prior on the hidden unit activations from which the sparseness constraint can be derived [46].

We train recognition weights w_{10}^{ij} from input neuron j to hidden neuron i and lateral recognition weights w_{11}^{ik} from hidden neuron k to hidden neuron i by the following on-line update rules:

$$\begin{aligned} \Delta w_{10}^{ij} &= \varepsilon_w\big(u_1^i\,\tilde{x}_0^j - \lambda^w\,|\bar{h}^i|\,w_{10}^{ij}\,\|\vec{w}^i\|^2\big) \\ \Delta w_{11}^{ik} &= \varepsilon_w\big(u_2^i\,\tilde{x}_1^k - \lambda^w\,|\bar{h}^i|\,w_{11}^{ik}\,\|\vec{w}^i\|^2\big) \end{aligned} \tag{46}$$

where ε_w is the learn step size. The second-level weights learn to predict the contribution of the activation of the first level units which originates from the input units. They do not learn the averaged activation which take into account the back projection nor the value which is sparsified by the transfer function.

The first term on the right hand side of Eq. (46) implements Hebbian learning. The second term which is scaled by λ^w is a soft activity dependent weight constraint. $\|\vec{w}^i\|^2 = \sum_l^N (w_{10}^{il})^2 + \sum_l^H (w_{11}^{il})^2$ is the sum of the squared weights to all N input units and all H hidden units and $|\bar{h}^i| = |h_1^i| + |h_2^i|$ is the mean of absolute values of the inner activations of hidden neuron i at the final relaxation time step. The weight constraint scales the length but does not change the direction of a hidden neuron weight vector. It is local in the sense that it does not depend on any weight of any other hidden neuron. Generative weights are made symmetric to the recognition weights, i.e. $V_{01} = W_{10}^T$ and $V_{11} = W_{11}^T$.

4.3 Boltzmann machine

In the Boltzmann machine which is a stochastic net there is some probability for "spontaneous" emergence of any activation state $(\vec{u}, \vec{x})$ of hidden unit activatons $\vec{u}$ and input unit activations $\vec{x}$. After some time one can then observe a probability distribution P^-_{ux}, the Boltzmann distribution (cf. Eqn. 5), on all states:

$$P^-(\vec{u}, \vec{x}) \;=\; P^-_{\vec{s}} \;=\; \frac{e^{-\beta E^{boltz}(\vec{s})}}{\sum_{\{\vec{s'}\}} e^{-\beta E^{boltz}(\vec{s'})}} \tag{47}$$

The marginal activity distribution which shows up on the input neurons (Eqn. 29) is

$$p^-_{\vec{x}} \;=\; \int_{\vec{u}} p^-_{\vec{x}\vec{u}}\, d\vec{u} \;=\; \frac{1}{Z} \int_{\vec{u}} e^{-\frac{E_{\vec{x}\vec{u}}}{T}}\, d\vec{u} \tag{48}$$

where

$$Z := \int_{\vec{x}\vec{u}} e^{-\frac{E_{\vec{x}\vec{u}}}{T}}\, d\vec{x}. \tag{49}$$

The probability $p^-_{\vec{x}}$ of generating a data point by the model is used to calculate the likelihood (Eqn. 14) of generating the whole data set.

The minus sign as upper index in term P^-_x denotes the free running phase of the network, which means that activations $(\vec{u}, \vec{x})$ emerge by some random process intrinsic to the network without influence by the environment. According to the likelihood principle P^-_x should equal the true data distribution P^+_x. The plus sign denotes the clamped phase in which the activation of the input neurons is given by the data distribution.

When the input units are clamped we have:

$$P^+(\vec{u}|\vec{x}) \;=\; P^+_{\vec{x},\vec{u}} \;=\; \frac{e^{-\beta E^{boltz}(\vec{x},\vec{u})}}{\sum_{\{\vec{u'}\}} e^{-\beta E^{boltz}(\vec{x},\vec{u'})}} \tag{50}$$

with the partition function as denominator taking care of the normalizing condition. The distribution $P^+_{\vec{x},\vec{u}}$ of the clamped phase follows if a data point $\vec{x}$ fixes the activation values of the input neurons.

In our work we take the simple recurrent architecture of Fig. 11 to underly a Boltzmann machine. Without lateral connections input unit activations $\vec{x}$ are purely determined by hidden unit activations $\vec{u}$ and hidden unit activations $\vec{u}$ are determined by input unit activations $\vec{x}$. This is reminiscent of Eqs. 12 and 13, but the output of each neuron is computed by a stochastic transfer function from its net input. The energy function eqn. 6 becomes:

$$E^{boltz}(\vec{s}, W) \;:=\; E_{\vec{x}\vec{u}} \;=\; -\frac{1}{2}\sum_{j}^{N} \underbrace{\sum_{i}^{H} w_{ji}\, u_i}_{h_j}\; x_j \;-\; \frac{1}{2}\sum_{i}^{H} \underbrace{\sum_{j}^{N} w_{ij}\, x_j}_{h_i}\; u_i \tag{51}$$

which can again be regarded as a sum of local single neuron energies $-h_j x_j$ on input neurons and $-h_i u_i$ on hidden neurons. With $w_{ij} = w_{ji}$ we have:

$$\frac{\partial E_{\vec{x}\vec{u}}}{\partial w_{ij}} = -u_i\, x_j \tag{52}$$

which will later constitute the Hebbian learning term in the learning rule.

The Boltzmann machine in its standard definition has permanently active neurons with activations $+1$ or -1. We will introduce it here with a further activation state, 0. Furthermore, we make this state degenerate. This means that it consists out of many states which, however, cannot be distinguished. Alternatively, one can regard this as one state which can have a high prior probability to occur on the hidden units. In order to make this a general framework we can set the degeneracy (probability) of the zero-state to zero by which we obtain the classical Boltzmann machine.

High degeneracy of the zero-state on the hidden neurons will render most neurons inactive during the process of generating the data. The data are “sparsely coded”. We will motivate the necessity of this coding paradigm in the following “Priors” section. Only by sparse coding, when the visual system is modeled, edge detectors will emergence.

Model and notation The model is depicted in Fig. 11. Where convenient we will concatenate the N-dimensional activation vector $\vec{x}$ on the input neurons and the H-dimensional activation vector $\vec{u}$ on the hidden neurons to an $N+H$-dimensional state vector $\vec{s} = (\vec{x}, \vec{u})$ for all neuron activations. Recognition weights W are symmetric to the generative weights V:

$$W = V^T$$

Identities The following identities are used for the derivation of the Boltzmann learning rule:

$$\frac{\partial Z}{\partial w_{ij}} = \frac{1}{T} \int_{\vec{x}\vec{u}} e^{\frac{E_{\vec{x}\vec{u}}}{T}} \frac{\partial E_{\vec{x}\vec{u}}}{\partial w_{ij}}\, d\vec{x}\, d\vec{u} \tag{53}$$

$$p^-_{\vec{x}\vec{u}} = \frac{1}{Z} e^{\frac{E_{\vec{x}\vec{u}}}{T}} \tag{54}$$

$$\frac{p^+_{\vec{x}}}{p^-_{\vec{x}}} \overbrace{p^-_{\vec{x}\vec{u}}} = \frac{p^+_{\vec{x}}}{p^-_{\vec{x}}} \overbrace{p^-_{\vec{x}} p^-_{\vec{u}|\vec{x}}} = p^+_{\vec{x}} p^-_{\vec{u}|\vec{x}} = p^+_{\vec{x}} p^+_{\vec{u}|\vec{x}} = p^+_{\vec{x}\vec{u}} \tag{55}$$

$$\int_{\vec{x}} p^+_{\vec{x}}\, d\vec{x} = 1 \tag{56}$$

Derivation of the learning rule The learning rule for the weights of the Boltzmann machine is derived in the standard literature (e.g. [27]). We will recapitulate the derivation in a compressed form to make sure that the result is not dependent on the number of possible activation states of a neuron. Moreover, it is even valid for continuous activations. The likelihood (Eqn. 22) is maximized by gradient ascent:

$$\Delta w_{ij} \;=\; \epsilon\,\frac{\partial}{\partial w_{ij}}\int_{\mathrm{R}^N} p_x^+\,\ln p_x^-(W)\,d\vec{x} \;=\; \epsilon\int_{\vec{x}}\frac{p_{\vec{x}}^+}{p_{\vec{x}}^-}\frac{\partial p_{\vec{x}}^-}{\partial w_{ij}}\,d\vec{x}$$

At first we will look at $\frac{\partial p_{\vec{x}}^-}{\partial w_{ij}}$.

$$\frac{\partial p_{\vec{x}}^-}{\partial w_{ij}} \overset{(48),(49)}{=} \frac{1}{ZT}\int_{\vec{u}} e^{\frac{E_{\vec{x}\vec{u}}}{T}}\frac{\partial E_{\vec{x}\vec{u}}}{\partial w_{ij}}\,d\vec{u} - \frac{1}{Z^2}\frac{\partial Z}{\partial w_{ij}}\int_{\vec{u}} e^{\frac{E_{\vec{x}\vec{u}}}{T}}\,d\vec{u}$$

For the left term we have

$$\frac{1}{ZT}\int_{\vec{u}} e^{\frac{E_{\vec{x}\vec{u}}}{T}}\frac{\partial E_{\vec{x}\vec{u}}}{\partial w_{ij}}\,d\vec{u} \overset{(52)}{=} \frac{1}{T}\cdot\frac{1}{Z}\int_{\vec{u}} e^{\frac{E_{\vec{x}\vec{u}}}{T}}\,u_i\,x_j\,d\vec{u} \overset{(54)}{=} \frac{1}{T}\int_{\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{u}$$

We develop the right term using

$$\frac{1}{Z}\frac{\partial Z}{\partial w_{ij}} \overset{(53),\,(52)}{=} \frac{1}{ZT}\int_{\vec{x}\vec{u}} e^{\frac{E_{\vec{x}\vec{u}}}{T}}\,u_i\,x_j\,d\vec{x}\,d\vec{u} \overset{(54)}{=} \frac{1}{T}\int_{\vec{x}\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{x}\,d\vec{u} \quad (57)$$

So the right term can be written using Eqn. 48 as

$$\frac{1}{Z}\Big(\int_{\vec{u}} e^{\frac{E_{\vec{x}\vec{u}}}{T}}\,d\vec{u}\Big)\frac{1}{Z}\frac{\partial Z}{\partial w_{ij}} \;=\; \frac{p_{\vec{x}}^-}{T}\int_{\vec{x}\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{x}\,d\vec{u}$$

Together

$$\frac{\partial p_{\vec{x}}^-}{\partial w_{ij}} \;=\; \frac{1}{T}\int_{\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{u} - \frac{p_{\vec{x}}^-}{T}\int_{\vec{x}\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{x}\,d\vec{u}$$

Insertion into the learning rule yields

$$\begin{aligned}\Delta w_{ij} \;&=\; \frac{\epsilon}{T}\int_{\vec{x}}\frac{p_{\vec{x}}^+}{p_{\vec{x}}^-}\int_{\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{x}\,d\vec{u} - \frac{\epsilon}{T}\int_{\vec{x}} p_{\vec{x}}^+\int_{\vec{x}\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{x}\,d\vec{u}\\ &\overset{(55),(56)}{=}\; \frac{\epsilon}{T}\Big(\int_{\vec{x}\vec{u}} p_{\vec{x}\vec{u}}^+\,u_i\,x_j\,d\vec{x}\,d\vec{u} - \int_{\vec{x}\vec{u}} p_{\vec{x}\vec{u}}^-\,u_i\,x_j\,d\vec{x}\,d\vec{u}\Big)\\ &=\; \frac{\epsilon}{T}\left(\langle u_i x_j\rangle^+ - \langle u_i x_j\rangle^-\right) \;=\; \Delta w_{ji}\end{aligned}$$

For the case of no lateral connectivity within input- or hidden layer the rule for the remaining weights yields:

$$\Delta w_{ij} \;=\; \epsilon\Big(\sum_{\{\vec{x}\}}\sum_{\{\vec{u}\}} P_{\vec{x}\vec{u}}^+\,u_i\,x_j \;-\sum_{\{\vec{x}\}}\sum_{\{\vec{u}\}} P_{\vec{x}\vec{u}}^-\,u_i\,x_j\Big) \qquad (58)$$

The factor ϵ is the learning step size. Each of the two terms is the expectation value of the correlation between activations u_i and x_j of neurons connected by w_{ij}. In the first term, $P^+_{\vec{x}\vec{u}}$ means that the input data distribution determines the occurrence of activations $\{\vec{x}\}$. In the second term, $P^-_{\vec{x}\vec{u}}$ and thus all neuron activity probabilities are purely determined by the Boltzmann distribution. The learning rule constitutes Hebbian (clamped phase) or anti-Hebbian (free running phase) learning and thus leads to symmetric weights.

4.3.1 Restricted Boltzmann machine

A Boltzmann machine without lateral weights among the hidden units (Fig. 11) is called "restricted" and is a special case of a product of experts in which each expert is one hidden neuron [31]. In such a model, each expert generates a data point with a certain probability, independently of the other experts. The total probability that a data point is generated is a product of the individual probabilities of each expert to generate it as we will show in the following.

An input neuron activation x_j is generated according to the Boltzmann distribution using the energy E_j. The probability that one input unit is switched to e.g. activation "1" is:

$$\begin{aligned} P^-(x_j = 1) &= \frac{e^{-\beta E_j}}{Z_j} = \frac{1}{Z_j} e^{\beta h_j x_j} = \frac{1}{Z_j} e^{\beta \sum_i w_{ij} u_i x_j} \\ &= \frac{1}{Z_j} \prod_i e^{\beta w_{ij} u_i x_j} \end{aligned}$$

where Z_j normalizes over all activation states of input neuron j. Thus generation of the input corresponds to a product of contributions u_i from each hidden unit i. Note that $w_{ij} = w_{ji}$ was used.

The main advantage of this architecture shows up in the clamped phase of training. When data are shown, the activation of each hidden neuron depends only on the data without influence by other hidden neurons. We will show that "inference" of the hidden code given the data can be done exactly by taking the expected value of the hidden neuron activation.

4.3.2 Sparse restricted Boltzmann machine

In the following we will make the hidden neurons code the data by sparse activation [61][62]. Fig. 16 shows the simple modification of hidden neuron activations: add to the states $+1$, -1 several (n) zero-states (with degenerate energy value). It is because of their numerousness that neuron activities prefer the zero-states over active states.

Formally, there are two possibilities to deal mathematically with the situation of multiple states with identical activations. First, all states are distinguished (by some label different from activations). Then the Boltzmann distribution (Eqn. 47) remains

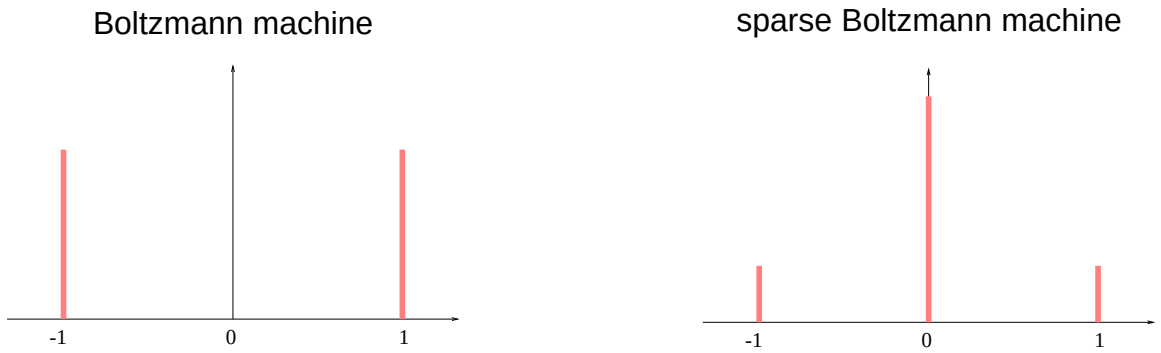


Figure 16: Prior density for the activation of one hidden neuron. The neuron can have activations -1, $+1$, and 0, the latter value is n-times (here: $n = 4$) more probable than an active state.

unchanged. The sum in the denominator (partition function) counts all states once (by the label which distinguishes them all).

Instead to distinguish states with similar activation values another possibility is to take into account multiplicity of states by a *prior* probability $P^{sp}(\vec{s})$ for a sparse activation. The Boltzmann distribution becomes

$$P_{\vec{s}} = \frac{P^{sp}(\vec{s})\, e^{-\beta E^{boltz}(\vec{s})}}{\sum_{\{\vec{s''}\}} P^{sp}(\vec{s''}) e^{-\beta E^{boltz}(\vec{s''})}} \tag{59}$$

where in the partition function the sum extends over states $\vec{s''}$ with different activation values only.

Fig. 16 can thus be interpreted as a prior function for the activation values of one neuron. The Fisher-kurtosis of the prior for a hidden neuron activation u is

$$\frac{\sum_{\{u\}} P^{sp}(u)\, u^4}{(\sum_{\{u\}} P^{sp}(u)\, u^2\,)^2} - 3 = \frac{\frac{(-1)^4}{n+2} + \frac{n \cdot 0^4}{n+2} + \frac{1^4}{n+2}}{(\frac{(-1)^2}{n+2} + \frac{n \cdot 0^2}{n+2} + \frac{1^2}{n+2})^2} - 3 = \frac{n+2}{2} - 3 = \frac{n}{2} - 2 \tag{60}$$

It has been argued that distributions with kurtosis larger than zero are suited for independent component analysis [7] and that distributions with large kurtosis produce sparse code [46]. We obtain a positive kurtosis for $n > 4$. From this point of view the standard Boltzmann machine is not suited for sparse coding because its neuron activity distributions have a negative kurtosis.

4.3.3 Gibbs sampling

The probability for a hidden neuron i to have activity u_i is

$$P(u_i) = \frac{1}{Z_i} e^{-\beta \sum_j^N w_{ij} x_j\, u_i}$$

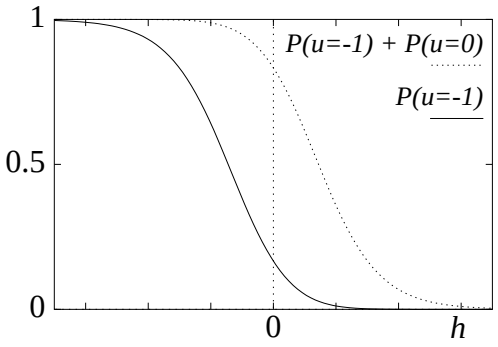


Figure 17: $P(u_i = -1)$, left curve, and $1 - P(u_i = +1)$, right curve, as a function of the net-input h. The vertical distance between both curves is the probability for 0 activation.

The probability for a hidden neuron i to have activity ± 1 or 0 is

$$\begin{aligned} P(u_i = \pm 1) &= \frac{1}{Z_i} e^{\mp \beta \sum_j^N w_{ij} x_j} \\ P(u_i = 0) &= \frac{n}{Z_i} \end{aligned}$$

(cf. Eqn. 59) where the normalizing constant considers its possible values -1, n-times zero and $+1$:

$$Z_i = e^{\beta \sum_j^N w_{ij} x_j} + n + e^{-\beta \sum_j^N w_{ij} x_j}$$

Fig. 17 shows these functions which correspond to transfer functions of stochastic neurons.

Because Eqn. 7 does not hold, we show directly that the Boltzmann distribution is realized also with the redundant zero-state:

$$\frac{P(u_i = 1)}{P(u_i = 0_k)} = \frac{\frac{1}{Z_i} e^{-\beta h_i}}{\frac{1}{Z_i}} = e^{-\beta h_i}$$

where $k \in \{1, \ldots, n\}$ is one of the n zero-states. The Boltzmann distribution is here defined on the single states and distinguishes the otherwise degenerate zero-states. If these were concatenated to one undistinguishable state, then a different distribution would result.

4.3.4 The mean-field approximation

The Boltzmann learning rule has an ill-reputation for not to be computationally tractable. If a neuron has $n + 2$ possible activation states, then a net with N neurons would have $(n + 2)^N$ possible activation states. If N is of the order of 100 or larger as in biological modeling then $P^-_{\vec{x}\vec{u}}$ cannot be computed repeatedly even for $n = 0$.

The mean-field approximation involves computing the expectation values $\langle s_k \rangle$ of the activation s_k of each neuron k. However, in an attractor net with symmetric weights and no thresholds the energy function is symmetric with respect to $s_i \to -s_i$. As a consequence, $\langle s_k \rangle^- = 0$ for a freely relaxing net. Furthermore, our input data has zero mean, thus $\langle x_j \rangle^+ = 0$.

In order to solve this problem we will find an expectation value $\langle s_i \rangle^{\mu +}$ which is "local" in one data point, i.e. the mean will be taken for each clamped data point $\vec{x}^\mu$. There are, however, no data points during the free-running phase. As a substitute we take local maxima when finding $\langle s_i \rangle^{\nu -}$ by the mean-field relaxation procedure which we describe in the next paragraph. The index ν denotes one such entity.

Formally, we derive our on-line like learning rule from the standard Boltzmann learning rule. Using $P^+_{\vec{x}\vec{u}} = P^+_{\vec{x}} P^+_{\vec{u}|\vec{x}}$ the first term of the learning rule, Eqn. 58, can be written as

$$\epsilon \sum_{\{\vec{x}\}} P^+_{\vec{x}} \sum_{\{\vec{u}\}} P^+_{\vec{u}|\vec{x}}\, u_i\, x_j \;=\; \sum_\mu \epsilon \sum_{\{\vec{u}\}} P^+_{\vec{u}|\vec{x}^\mu}\, u_i\, x_j^\mu$$

The sum over activations $\{\vec{x}\}$ which are weighted by $P^+_{\vec{x}}$ is replaced by summing actual occurrences μ of data points on the right hand side.

Equivalently, using $P^-_{\vec{x}\vec{u}} = P^-_{\vec{x}} P^-_{\vec{u}|\vec{x}}$ the second term of Eqn. 58 yields

$$\epsilon \sum_{\{\vec{x}\}} P^-_{\vec{x}} \sum_{\{\vec{u}\}} P^-_{\vec{u}|\vec{x}}\, u_i\, x_j \;=\; \sum_\nu \epsilon \sum_{\{\vec{u}\}} P^-_{\vec{u}|\vec{x}^\nu}\, u_i\, x_j^\nu$$

Analogously to above, the sum over activations $\{\vec{x}\}$ which are weighted by $P^-_{\vec{x}}$ is replaced by summing occurrences ν of data points which are generated by the free running system. (After successful learning these should equal the true data.) Defining

$$\langle u_i\, x_j \rangle^{\mu +} \;:=\; \sum_{\{\vec{u}\}} P^+_{\vec{u}|\vec{x}^\mu}\, u_i\, x_j^\mu \qquad \text{and} \qquad \langle u_i\, x_j \rangle^{\nu -} \;:=\; \sum_{\{\vec{u}\}} P^-_{\vec{u}|\vec{x}^\nu}\, u_i\, x_j^\nu$$

the learning rule writes as

$$\Delta w_{ij} \;=\; \epsilon \Big(\sum_\mu \langle u_i\, x_j \rangle^{\mu +} \;- \sum_\nu \langle u_i\, x_j \rangle^{\nu -} \Big) \tag{61}$$

This is an on-line rule: we make a learn step for each data point $\vec{x}^\mu$ and for each entity $\vec{s}^\nu$.

From Eqn. 61 we start with the mean-field approximations. First, the average of correlations between activation states is replaced by a correlation of averages:

$$\langle u_i x_j \rangle^{\mu +} \;\to\; \langle u_i \rangle^{\mu +} \langle x_j \rangle^{\mu +} \qquad \text{and} \qquad \langle u_i x_j \rangle^{\nu -} \;\to\; \langle u_i \rangle^{\nu -} \langle x_j \rangle^{\nu -}$$

The mean activation for our three-state hidden neuron is

$$\langle u_i \rangle \;=\; P(-1) \cdot (-1) \;+\; n \cdot P(0) \cdot 0 \;+\; P(+1) \cdot (+1)$$

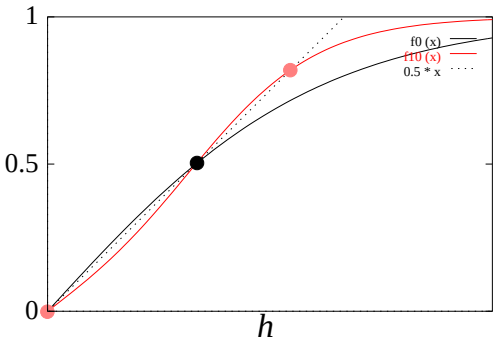


Figure 18: Transfer functions $\tanh(h)$ (solid line) and the sparse mean-field transfer function $\varphi(h)$ (dashed, curved line), here for $n = 10$. Only the positive half-axes are shown and both functions are scaled such that they intersect the identity function (dotted line) at 0.5. A single-neuron dynamics $h(t+1) = \tanh h(t)$ leads for large time t to the following stationary solutions: a stable fix-point is 0.5, an unstable fix-point is 0.0. The dynamics $h(t+1) = \varphi h(t)$ features stable fix-points at 0.0 and $\varphi(h) = 0.8$, an unstable fix-point is 0.5.

Secondly, we make the assumption that the probability (according to the Boltzmann distribution) for a neuron activation depends only on the mean of its inner activation $h_i := \sum_j^N w_{ij} x_j$ for a hidden neuron or $h_j := \sum_i^H w_{ji} u_i$ for an input neuron. The error we make here is small if the net is large. Then the neuron activation u_i will have only a minor effect on the average activities of other neurons. We obtain

$$\langle u_i \rangle \;=\; \frac{e^{\beta\langle h_i\rangle} \;-\; e^{-\beta\langle h_i\rangle}}{e^{\beta\langle h_i\rangle} \;+\; n \;+\; e^{-\beta\langle h_i\rangle}} \;=:\; \varphi_u(\langle h_i\rangle) \tag{62}$$

The effective transfer function φ_k is depicted in Fig. 18. For $n = 0$ the transfer function becomes $\tanh\langle h_i\rangle$.

For the input neurons we choose the linear/identity mean-field transfer function

$$\langle x_j \rangle \;=\; \varphi_x(\langle h_j\rangle) \;=\; \langle h_j\rangle \tag{63}$$

so that the data are not distorted. In particular in the clamped state the data are unchanged. This transfer function can be deduced if we assume a (infinite) homogeneous prior on whatever continuous or equidistantly spaced discrete input activations. Then it results from the linear superposition of $\tanh$-functions:

$$\lim_{M\to\infty} \frac{1}{M} \sum_{m=0}^{M} m \,\tanh \frac{\langle h_j\rangle}{m}$$

In the limit the majority of terms will be indexed by $m > m_0$ for arbitrary but fixed m_0. Considering the Taylor expansion

$$m \,\tanh \frac{\langle h_j\rangle}{m} \;=\; \langle h_j\rangle \;-\; \frac{m}{3}\Big(\frac{\langle h_j\rangle}{m}\Big)^3 \;+\; \ldots$$

one finds that within a fixed range of $\langle h_j \rangle$ these terms approximate the linear identity function if m_0 is chosen sufficiently large.

Resting activity It was shown by theoretical and experimental work [37][38] that endogenously active cells are required to maintain robust low, steady firing. Besides the "OFF"-states with degeneracy n, we will now assign each hidden neuron a positive resting activity, i.e. an "ON"-state with degeneracy m. The probability that a hidden neuron i is switched "ON" by its input h_i is:

$$P(u_i = 1) \;=\; \langle u_i(\vec{x}) \rangle \;=\; \frac{e^{h_i} + m}{e^{h_i} + m + n} \;=:\; \varphi_u^r(\vec{x}) \tag{64}$$

Having only states 0 and 1, the mean activation equals the probability to be 1.

Fig. 19 left, indicates that a network state which features both, highly and weakly active neurons is not robust. Highly active states alone, Fig. 19 right, are more stable, but they lead to highly active states in other neurons, too. The same is true for weakly active neurons. The parameter regime where both states co-occur robustly increases with m [37].

However, it has not been shown that Eq. 64 can lead to a Boltzmann distribution and we have run the Boltzmann machine with $m = 0$ only.

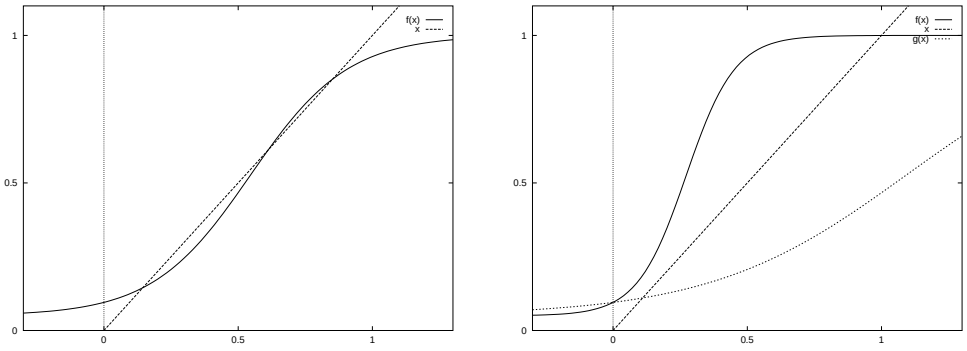


Figure 19: **Left:** Sparse positive-only mean-field transfer function with resting activity, φ_u^r, as a function of a one-dimensional input x. There are three intersections with the identity function (dotted line), the left and the right one correspond to stable fix-points. The parameter regime which features three fix-points, however, is narrow. **Right:** If neurons are active within an attractor, then they are connected to active neurons with strong weights. This leads to a steeper effective transfer function with only one stable fix-point at high activation (upper curve). Outside of an attractor, input arrives via weak weights, leading to a narrow effective transfer function with only one stable fix-point at low activation (lower curve). Without resting activity ($m = 0$), this point is closer to zero.

Attractor dynamics of continuous mean-field neurons The mean neuron activations are determined by Eqs. 62 and 63, respectively. Writing down these equations for all neurons they constitute a system of nonlinear equations. Inserting arbitrary values for all neuron activities, the equations need not be consistent. We will solve the system of equations by relaxating for each neuron k the estimate $\hat{h}_k$ of its inner activations mean $\langle h_k \rangle$. We obtain estimates $\hat{x}_j = \varphi_x(\hat{h}_j) = \hat{h}_j$ and $\hat{u}_i = \varphi_u(\hat{h}_i)$ for input and hidden neuron activations, respectively. The relaxation dynamics is defined by:

$$\frac{1}{\eta}\frac{d\hat{h}_i}{dt} = \sum_j^N w_{ij}\hat{x}_j - \hat{h}_i \qquad \text{or} \qquad \frac{1}{\eta}\frac{d\hat{h}_j}{dt} = \sum_i^H w_{ji}\hat{u}_i - \hat{h}_j \tag{65}$$

η is the update step size. $\eta = 1$ leads to fix-point iteration. If after many repetitions equilibrium is reached then the sum of the weighted input of each neuron equals its inner activation.

In the case of many neurons which are strongly interconnected by positive weights activations behave like ferromagnetic dipoles: all neurons together behave the same as if there was one neuron which is connected to itself only . Above update dynamics reveals three stable fix-points (see Fig. 18): a positive non-zero value, its negative counterpart and the fix-point zero. The latter is the result of the sparse coding ansatz. It would be an unstable fix-point if we used the $\tanh$ transfer function which we obtain for $n = 0$ as in the standard Boltzmann machine.

The relaxation procedure is simple for the clamped phase: the activations of the input neurons are set by the data. They do not respond to the feedback. Thus hidden neurons always see the clamped input data through the feed-forward weights. There is no contribution by lateral weights. Thus one iteration gives us the hidden activations as in a purely feed-forward architecture.

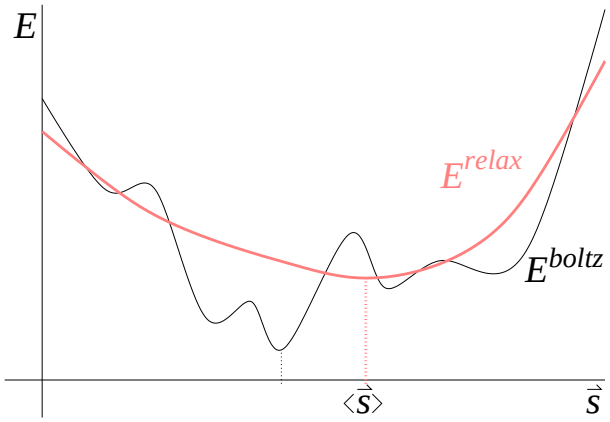


Figure 20: Energy functions for the activations which are in an abstract manner represented on the x-axis.: the Boltzmann energy E^{boltz} is used to set up the probability distribution on the (discrete) activations $\{\vec{S}\}$. The mean-field energy function E^{relax} is maximized when finding the continuous vector $\langle \vec{S} \rangle$.

Free relaxation requires a lot of iterations, because each time that activities change in one layer, activities on the other layer have to be corrected, too. It is not obvious that this procedure converges. But it can be shown that continuous relaxation (in the limit of small step size η) according to Eqs. 65 minimizes an energy function for the activity states:

$$E^{relax} = -\sum_i^H \sum_j^N w_{ij}\hat{u}_i\hat{x}_j + \sum_j^N \int_0^{\hat{x}_j} \varphi_x^{-1}(x')dx' + \sum_i^H \int_0^{\hat{u}_i} \varphi_u^{-1}(u')du'$$

The equilibrium points $\frac{d\hat{u}_i}{dt} = 0 \Leftrightarrow \frac{dE}{dt} = 0$ are the only attractors, because the energy cannot decrease eternally.

As this energy function is not minimized by gradient ascent, we give two conditions to show that the energy is minimized:

- $E^{relax} < \infty$, i.e. E^{relax} has a finite lower bound.
  This is true for each of the two terms. The first term is bounded, because the transfer function of the hidden units is bounded: $|s_k| < \infty \quad \forall k$ and thus $u_i = \varphi_i(h_i) < \infty$
  Activations of input neurons are bounded, too, due to bounded input: $x_i \leq \sum_i^H w_{ji}u_i$, if $\varphi_j(h_j) \leq const \cdot h_j$ Remember there are no lateral connections. The second term is positive, $\text{sign}(s_k)\cdot\varphi_k^{-1}(s_k) > -\infty \quad \forall k,\ s_k \in [s^{min}, s^{max}]$.

- E^{relax} de(in-)creases monotonously according to relaxation. The following proof is briefly recapitulated from the standard literature [30].

$$\frac{dE^{relax}}{dt} = \sum_i^H \sum_j^N \left(w_{ij}\frac{du_i}{dt}x_j + w_{ij}u_i\frac{dx_j}{dt} \right) \tag{66}$$

$$- \sum_k^{H+N} \frac{\partial}{\partial s_k}\left(\int_0^{s_k} \varphi_k^{-1}(s')ds' \right)\frac{\partial s_k}{\partial t}$$

$$= \sum_i^H \left(\sum_j^N w_{ij}x_j\frac{du_i}{dt} - \varphi_i^{-1}(u_i)\frac{du_i}{dt} \right) \tag{67}$$

$$+ \sum_j^N \left(\sum_i^H w_{ij}u_i\frac{dx_j}{dt} - \varphi_j^{-1}(x_j)\frac{dx_j}{dt} \right)$$

$$\overset{\varphi_k^{-1}(s_k)=h_k}{=} \sum_i^H \frac{du_i}{dt}\left(\sum_j^N w_{ij}u_j - h_i \right) + \sum_j^N \frac{dx_j}{dt}\left(\sum_i^H w_{ij}x_i - h_j \right)$$

$$\overset{\substack{\text{chain rule} \\ w_{ij}=w_{ij} \\ (65)}}{=} \sum_k^{H+N} \underbrace{\frac{ds_k}{dh_k}}_{>0} \underbrace{\frac{dh_k}{dt}\frac{1}{\epsilon}\frac{dh_k}{dt}}_{>0}$$

Note that weight symmetry, $w_{ij} = w_{ji}$, is essential for the proof.

Note that the local maxima of E^{relax} do not necessarily correspond to the local maxima of E^{boltz} (see Fig. 20). This was not the intention, instead we want to compute $\langle S \rangle$. This state could better be expressed by the energy $\langle E^{boltz} \rangle$ (but not exactly).

4.3.5 Overview of energy functions used by the Boltzmann machine

- The weights should maximize the likelihood that the observed data are generated by the model $(\{x^\mu\}, W)$ (all data).

  $$\ln P(W \mid p_x^+) = \mathcal{L}(p_x^+ \mid W) + \ln P(W)$$

  We need this to learn the weights and it leads us to the Boltzmann learning rule.

- Each activity vector is assigned a "theoretical" energy value

  $$E^{boltz}(\vec{s}, W) + E^{dif}(\vec{u})$$

  We need this to compute the distribution p_x^- according to Boltzmann distribution $e^{E^{boltz}}$.

- Each activity vector in practice (relaxation of continuous values) maximizes the energy

  $$E^{relax}(\vec{s}, W) + E^{dif}(\vec{u})$$

  We need this to approximate p_x^- by the mean-field method.

- The correction term for the weights comes to the energy function for the weights.

- The correction term for the activations comes to the energy function for the activation -> changes the Boltzmann distribution.

Note that sparseness does not enter the terms of the energy functions. Instead it is realized by introducing more available states.

4.4 Helmholtz machine and the wake-sleep algorithm

Eqn. 28 supplies a justification that an incorrect distribution may be used in order to maximize the likelihood. With this justification we shall work with two different models: *(i)* an original, generative model with weights V which defines P, and generates the true output of the network, and *(ii)* a separate recognition model with weights W which defines Q. The goal is to train this model such that Q approximates P.

Training of these two models involves two phases, one for each model: *(i)* the wake phase, to learn the generative weights V so that the data are generated correctly, and *(ii)* the sleep phase, to learn the recognition weights W so that the recognition model approximates the generative model. The algorithm is as follows.

- wake phase:

  1. initialize activations on the input layer with a data point $\vec{x}$.
  2. compute hidden unit activations $\vec{u}$ using the recognition weights W:

     $$\vec{u}^{\lambda} = \vec{f}(W^{\lambda,\lambda-1}\vec{u}^{\lambda-1})$$

     where λ is the layer index and $\vec{u}^0$ is the input vector $\vec{x}$. This procedure is done sequentially, layer for layer starting at the hierarchically bottom-most hidden layer up to the top-most layer.
  3. activations are projected back between adjacent layers using the generative weights V:

     $$\vec{s}^{\lambda} = \vec{f}(V^{\lambda,\lambda+1}\vec{u}^{\lambda+1})$$

     where $\vec{s}^{\lambda}$ is the generative model's reconstruction of the activations on layer λ using the code on layer $\lambda + 1$. Note that the code which was previously obtained by the generative weights is used as the source for each layer. The generative model thus "inverts" only one step but not more hierarchical levels.
  4. adjust the generative weights V to minimize the reconstruction error $\vec{u}^{\lambda} - \vec{s}^{\lambda}$ in each level of the hierarchy:

     $$\Delta V^{\lambda,\lambda+1} \approx (\vec{u}^{\lambda} - \vec{s}^{\lambda}) \cdot \vec{u}^{\lambda+1}$$

  Here we assume that the recognition model is correct. In this phase the generative weights V are trained to model the data distribution and to "invert" the recognition model W.

- sleep phase:

  1. initialize with a code vector $\vec{u}$ at the hierarchically top-most layer by random.
  2. compute the code vector at lower levels using the generative weights V:

     $$\vec{s}^{\lambda} = \vec{f}(V^{\lambda,\lambda+1}\vec{u}^{\lambda+1})$$

  3. project activations back towards the inner-most representation across adjacent layers using the recognition weights W:

     $$\vec{u}^{\lambda} = \vec{f}(W^{\lambda,\lambda-1}\vec{u}^{\lambda-1})$$

     where $\vec{u}^0 = \vec{x}$.
  4. adjust the recognition weights W to minimize the code error $\vec{s}^{\lambda} - \vec{u}^{\lambda}$ in each level of the hierarchy:

     $$\Delta W^{\lambda,\lambda-1} \approx (\vec{s}^{\lambda} - \vec{u}^{\lambda}) \cdot \vec{s}^{\lambda-1}$$

Here we assume that the generative model is correct, including the code. In this phase the recognition weights $\boldsymbol{W}$ are trained to model the code distribution and to “invert” the generative model $\boldsymbol{V}$.

In both phases, the model which represents a data flow into the opposite direction is “inverted” by training. The wake phase only, however, considers the data and trains the generative model accordingly. Even as long as the generative model is bad, the sleep phase tries to learn the recognition model to become the inverse of it and thus does not do optimal. However, as the generative model gets better, training of the recognition model also becomes better. Note that this algorithm is not a gradient descent on an energy function.

4.5 Priors

4.5.1 *Prior* on the activities: sparse coding

In our model architecture (Fig. 11) we recognize an auto-associator network if the goal is to reproduce (predict) the data. In this case, its output is generated by a hidden code which is computed from its input and this output is compared to the input on the input units. A trivial solution for an optimal hidden code arises if the number of hidden units equals or even exceeds the number of input units. Any data point is perfectly coded if $\boldsymbol{W}$ equals the unity matrix and the hidden code $\vec{u}$ equals the data $\vec{x}$ (superfluous hidden units may be zero) and information transfer between input and hidden units is maximal. Obviously there is a need for constraints to achieve qualitatively useful information processing.

A popular principle to underly such a constraint is the reduction of redundancy between hidden unit codes. Best-founded arguments for this, in our view, come from the biological example, the recurrent LGN - V1 projection in the visual system. Reminiscent of the idea of an "overcomplete basis set", more cells exist in V1 than in the LGN.

Neurons in the input layer (layer IV) of primary visual cortex (area V1) each receive input from a small part of the visual field only and respond to edges of a preferred orientation. Statistics about how often such a stimulus feature occurs tells about how often a cell should be active. So we can regard a neuron to respond only if there is an edge within its receptive field and if the edge has the direction which this neuron is tuned to. According to image statistics a neuron rarely encounters such a matching edge. But once there is such an edge it fits well and the neuron will be optimally stimulated, i.e. it has a strong input [4]. Figs. 21,22, left, display this behavior.

In summary, a localized edge-detector neuron is unlikely to be activated if an image is shown on the retina, but once it is activated then it is activated strong. The representation of an image within the hidden units activations is thus sparse: only a small number of neurons are active, but these are strongly active. Accordingly, we can impose a sparse coding constraint on hidden unit activations. We hope that a neuron will then try to fulfill the constraint by learning to code for localized edges.

Activations in such a sparse context are interpreted to be independent causes, a few of which constitute a data point [44]. A real world image hence consists of a relatively small number of localized edges. Independence of the causes is not restricted to second order statistics such as pairwise correlations. The interest lies on statistical independence of higher order: the "basis functions" need not be orthogonal. Hereby the number of causes may be larger than the input dimension allowing for an "overcomplete basis set" [46].

The idea so far is to model the input data as good as possible given the constraint of sparse coding on the inner representation. It underlies a growing variety of recurrent models [21][26][49][46][45][22]. Cardoso [12] and Olshausen [44] have shown the equivalence of these feedback data generating models to a class of feed-forward data analyzing models. The latter maximize the independence on the inner representation by maximizing the entropy on the hidden unit activity distribution [7][8]. Applied to the LGN – V1 projection, they develop target neurons with *localized receptive fields* and *orientation preference* which may theoretically be *ocular dominant*.

A totally different biological motivation for sparse coding is that metabolic costs of the coding neurons need to be minimized [40]. According to this interpretation, two conflicting goals need to be satisfied: to maximize information capacity and to minimize metabolic costs of the coding neurons.

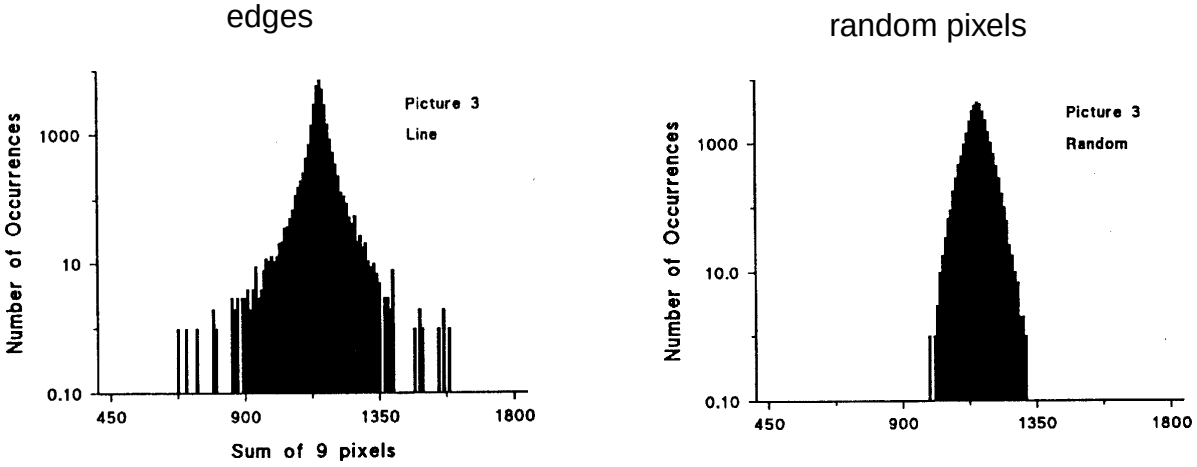


Figure 21: **Left:** activation statistics of edges in a natural image, taken from [4]. The left- and rightmost contributions to the histogram correspond to edges in the image which are co-aligned with 9 pixels in a row. **Right:** activation statistics of a random pixel pattern in the same natural image which was taken for the left diagram.

Neglecting a normalization constant, the following prior (Figs. 22,23) will enforce sparse activations:

$$P(\vec{u}) = \frac{1}{\prod_i (1+|u|_i^r)^{\frac{\lambda_u}{2}}} = e^{\frac{\lambda_u}{2}\ln(\prod_i (1+|u|_i^r))^{-1}} = e^{-\frac{\lambda_u}{2}\sum_i \ln(1+|u|_i^r)} = \prod_i e^{-\frac{\lambda_u}{2}S(u_i)} \quad (68)$$

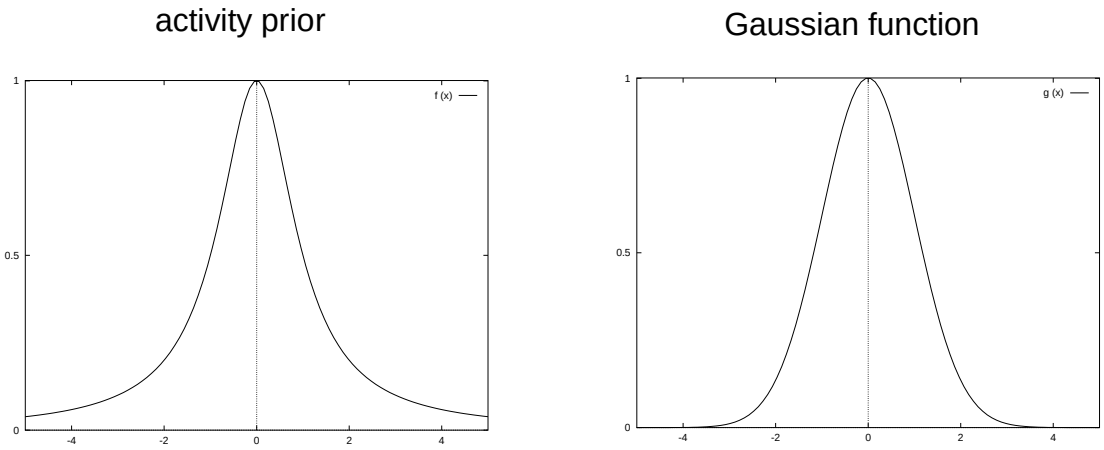


Figure 22: **Left:** a sparse prior with positive kurtosis. **Right:** a Gaussian function.

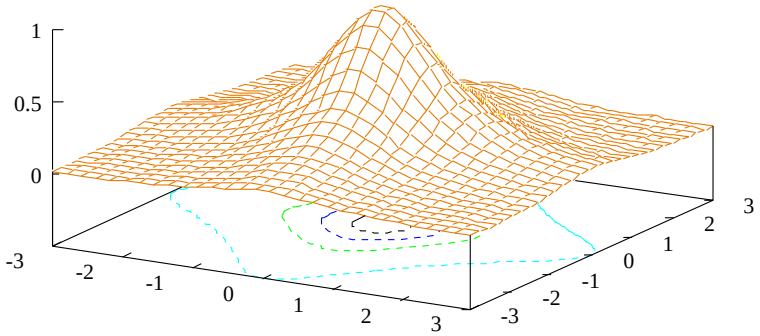


Figure 23: The *prior* on the activities (z-axis) enforces *sparse coding*. x- and y-axes denote the activities of two arbitrarily selected hidden neurons. In two dimensions it can be seen that it is not rotationally symmetric. Instead, "ridges" along the axes give rise to activation vectors with zero components.

with

$$S(u_i) \;=\; \ln(1+|u|_i^r)$$

The parameter r scales the kurtosis. It becomes smaller if r is large and larger if r is small [26]. In the limit of $r \to 0$ the function becomes a delta function. In all our simulations, $r = 2$.

An activity-decay term adds to the fast dynamics to find $\vec{u}^{opt}$:

$$\begin{aligned}\Delta u_i^{sparse} \;&=\; \frac{\partial}{\partial u_i} \ln\, P(\vec{u}) \;=\; \frac{\partial}{\partial u_i} \ln\, e^{-\frac{\lambda_u}{2}\sum_i \ln(1+|u|_i^r)} \;=\; -\frac{\lambda_u}{2}\frac{\partial}{\partial u_i}\ln(1+|u|_i^r) \\ &=\; -\frac{\lambda_u}{2}\cdot\frac{r\cdot|u_i|^{r-1}}{1+|u|_i^r} \;\overset{r=2}{=}\; -\lambda_u\frac{u_i}{1+u_i^2} \end{aligned} \tag{69}$$

The parameter λ_u can be seen here to scale the effectiveness of the prior.

For the models with discrete activations, see Eqn. 59 and Fig. 16.

$y(u)$	$\frac{\text{atan}(u)}{\pi} + \frac{1}{2}$	$\frac{e^u}{n + e^u} = \frac{1}{1 + n\, e^{-u}}$
$p^-(u)$	$\frac{1}{\pi} e^{-\ln(1+u^2)} = \frac{1}{\pi\,(1+u^2)}$	$\frac{n\, e^{-u}}{(1 + n\, e^{-u})^2}$
$S(u)$	$\ln(1+u^2)$	$u + 2\,\ln(1 + n\, e^{-u}) - \ln(n)$
$\frac{\partial}{\partial u} S(u)$	$-\frac{u}{1+u^2}$	$1 - 2\frac{n\, e^{-u}}{1 + n\, e^{-u}} = \frac{1 - n\, e^{-u}}{1 + n\, e^{-u}}$

Table 6: Priors and the relations: $\frac{\partial}{\partial u} y(u) = p^-(u) = e^{-S(u)}$.

4.5.2 *Prior* on the weights: weight constraint

The motivation to constrain weights to small values follows the lines of [46]. The sparse transfer function silences many neuron activations which incurs under-estimation of the data: they will back-project smaller input values during free-run than values given by the data. The weights try to compensate for this by growing larger regardless of the scale of the input. As a result, the net input h_i of any hidden neuron i will be larger. More neurons will take part in data generation and they will decide more decisively for the saturating states $+1$ or -1 of their transfer function. There is no sparse coding anymore.

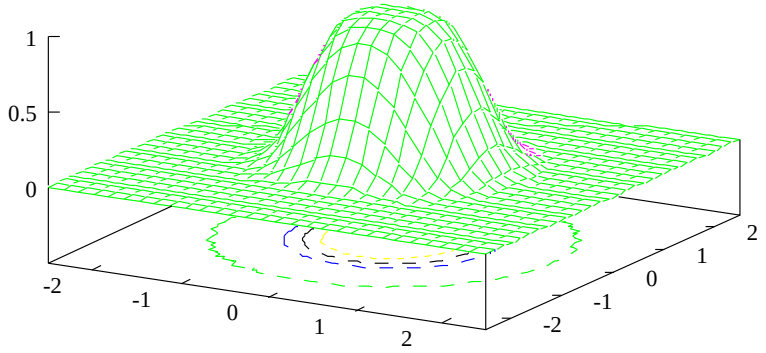


Figure 24: The *Prior* on the weights (z-axis) makes small weights likely. x- and y-axes are the strengths of two arbitrarily selected input weights of one hidden neuron. It is rotationally symmetric (see contour lines) and thus does not have an influence on the direction of the weight vector.

The following Bayesian prior (Fig. 24) constrains the (receptive/projective) field vector $\vec{w}_i$ of a hidden neuron i to small length

$$P(\vec{w}_i) \;\approx\; e^{-\frac{d_w}{4} \sum_{j'} w_{ij'}^2 \sum_{j''} w_{ij''}^2}$$

$$P(\vec{\mathsf{v}}_i) \;\approx\; e^{-\frac{d_{\mathsf{V}}}{4}\sum_{j'}\mathsf{v}_{j'i}^2\,\sum_{j''}\mathsf{v}_{j''i}^2}$$

d_w is a scaling factor. The direction of $\vec{w}_i$ (shape of the field) will not change because $P(\vec{w}_i)$ depends on its length only. Gradient ascent leads to the additive correction term to above maximum likelihood learning rules. A weight-decay term adds to the slow dynamics to find V^{opt}:

$$\Delta\mathsf{v}_{ji}^{decay} \;=\; \frac{\partial}{\partial\mathsf{v}_{ji}}\ln\,P(\vec{\mathsf{v}}_i) \;=\; -d_v\mathsf{v}_{ji}\sum_{j'}\mathsf{v}_{j'i}^2 \tag{70}$$

Relation between inverse temperature and weight length The absolute value of the temperature does not play a role: it appears in the expression $e^{-\beta h}$, always multiplied by the net input h. Weights can thus compensate e.g. for a high temperature (small β) by growing large. However, for successful annealing of the temperature, the span-width (above T_{final}) can be important.

For the case of mean-field relaxation and the linear transfer function on the input units we show that scaling the weights has the same effect on the relaxation procedure as scaling the square root of the inverse temperature β.

We express scaling of all entries for a given $H \times N$ weight matrix W by multiplication of a diagonal Matrix L. Its diagonal values l are the scaling factors. Both of the following operations scale the weights by l:

$$L_H\,W \qquad \text{or} \qquad W\,L_N$$

where L_H is a $H \times H$ matrix and L_N is a $N \times N$ matrix.

In a fix-point relaxation a new hidden activation vector $\vec{u}$ is obtained according to Eqn. 65:

$$\vec{u} \;=\; \vec{\varphi}(W\,L_N\,\vec{x})$$

Inserting $\vec{x} \;=\; L_N\,W^T\,\vec{u}$ yields

$$\vec{u} \;=\; \vec{\varphi}(W\,L_N\,L_N\,W^T\,\vec{u}).$$

With $B_N := L_N\,L_N$ we obtain (in analogy to the above notation):

$$\vec{u} \;=\; \vec{\varphi}(B_H\,W\,W^T\,\vec{u}).$$

The diagonal values of the diagonal matrix B_N or B_H are the inverse temperature. In short:

$$|\vec{w}| \to l|\vec{w}| \quad\Leftrightarrow\quad \beta \to l^2\beta$$

These considerations, however, ignore that the back-projected input vector $\vec{x}$ changes if W changes but not if β changes (assuming a linear transfer function on the input neurons). It is often the case that input values are given (by data) and hidden unit values also have desired values (by a constraint).

Examples: *(i)* If the mean activation on the input neurons is doubled, then the weight strengths have to be doubled as to make $\vec{x} = \boldsymbol{W}^T\vec{u}$. Then β must be divided by 4 such that the hidden neurons will not receive a stronger net-input $\vec{h}$. *(ii)* If the number of input neurons is doubled but mean activations remain, then weights remain but β has to be halved.

4.5.3 Priors for topographic mappings

A shortcoming of the above mathematical theory is that it ignores spatial relations among the target neurons. Thus, *retinotopy* and superimposed properties like orientation preference patterns cannot be addressed. For robust topographical mappings we need to add two biologically inspired mechanisms to the learning rule of the recurrent network. First, neighborhood interactions to ensure neighborhood relationships. Second, there must be a label on source- as well as on target cells to determine the correct polarity of the topographic map.

Neighborhood interactions can be implemented by *(i)* diffusive activity, *(ii)* diffusive weights or *(iii)* a diffusive weight update step (as in the Kohonen model). It is most elegant to consider neuronal activations to diffuse from the proper target cells towards nearby targets.

Diffusive forward activation transfer Let $s_{\vec{x}}$ denote the number of spikes heading to target cell at position $\vec{x}$, and $\mathcal{N}$ be the number of neighbors each cell has. If P^d is the probability for a spike to arrive instead at any neighbor and if all neighbors receive this spike with the same probability, then $P^d/\mathcal{N}$ is the probability for a spike to reach $\vec{x}$ instead of $\vec{y}$. Averaging over time (spikes) the activation $u_{\vec{x}}$ will be altered by

$$\Delta u^{dif}_{\vec{x}} \;=\; -P^d s_{\vec{x}} + \sum_{\vec{y}\neq\vec{x}}^{\mathcal{N}} \frac{1}{\mathcal{N}} P^d s_{\vec{y}} \tag{71}$$

We show that this correction to the update rule for the activities can be derived by a prior $P^{dif}(\vec{u})$ on the activities

$$P^{dif}(\vec{u}) \;=\; \prod_{\vec{x}} P^{dif}_{\vec{x}} \;=\; \prod_{\vec{x}} e^{-\frac{P^d}{4\mathcal{N}}\sum_{\vec{y}\neq\vec{x}}^{\mathcal{N}}(u_{\vec{y}}-u_{\vec{x}})^2}$$

Note that $P^{dif}(\vec{u})$ factorizes into terms $P^{dif}_{\vec{x}}$ one for each target neuron, but that $P^{dif}_{\vec{x}}$ contains activities of neighboring cells of $\vec{x}$. The modification of the update rule for activity $u_{\vec{x}}$

$$\Delta u^{dif}_{\vec{x}} \;=\; \frac{\partial}{\partial u_{\vec{x}}} \ln P^{dif}(\vec{u}) \;=\; \frac{\partial}{\partial u_{\vec{x}}} \ln P^{dif}_{\vec{x}} \;+\; \frac{\partial}{\partial u_{\vec{x}}} \sum_{\vec{y}\neq\vec{x}} \ln P^{dif}_{\vec{y}}$$

will take into account terms which depend on $u_{\vec{x}}$. These are the priors for the cell at location $\vec{x}$ (left term) and its neighbors (right term). This yields

$$\begin{aligned}
\Delta u_{\vec{x}}^{dif} &= \frac{\partial}{\partial u_{\vec{x}}} \underbrace{\Big(-\frac{P^d}{4\mathcal{N}} \sum_{\vec{y}\neq\vec{x}} (u_{\vec{y}} - u_{\vec{x}})^2 \Big)}_{\ln P_{\vec{x}}^{dif}} + \frac{\partial}{\partial u_{\vec{x}}} \underbrace{\Big(\sum_{\vec{y}\neq\vec{x}} -\frac{P^d}{4\mathcal{N}} (u_{\vec{x}} - u_{\vec{y}})^2 \Big)}_{\text{contribution from neighbors}} \\
&= \frac{P^d}{2\mathcal{N}} \sum_{\vec{y}\neq\vec{x}} (u_{\vec{y}} - u_{\vec{x}}) \; + \; \frac{P^d}{2\mathcal{N}} \sum_{\vec{y}\neq\vec{x}} (u_{\vec{y}} - u_{\vec{x}}) \\
&= -P^d \, u_{\vec{x}} \; + \; \frac{P^d}{\mathcal{N}} \sum_{\vec{y}\neq\vec{x}} u_{\vec{y}}
\end{aligned}$$

In the case of a 1-dimensional target layer a cell at location $\vec{x}$ has two neighbors at locations $\vec{x}+1$ and $\vec{x}-1$. The equation writes

$$\Delta u_{\vec{x}}^{dif} \; = \; P^d \, (\frac{1}{2} s_{\vec{x}-1} \; - s_{\vec{x}} \; + \frac{1}{2} s_{\vec{x}+1}) \; \hat{=} \; \frac{P^d}{2} \, \frac{\partial^2}{\partial \vec{x}^2} s_{\vec{x}}$$

which is the discrete approximation to the second derivative in space.

Chemical repulsion term Topographic maps in the brain have a predictable polarity. This means that from one animal to the next, the map is never inverted or rotated. In models, reversal of polarity corresponds to a permutation of cell indexes and is thus natural. For biological modeling thus, a term is needed which breaks permutation symmetry in the cell indexes.

A "chemical repulsion term" [59][60] may be used which is proportional to a graded concentration $c_{\vec{\alpha}}^{input}$ of hypothetical chemical markers along each axis of the input plane and $c_{\vec{x}}^{hidden}$ within the hidden units. It is reasonable to make the influence of this term on weight growth proportional to the amount (strength) of connections which are affected. This yields the following additive term to a weight learning rule:

$$\Delta w_{\vec{x}\vec{\alpha}}^{chem} \; = \; -\lambda^c \, c_{\vec{\alpha}}^{input} \, c_{\vec{x}}^{hidden} \, w_{\vec{x}\vec{\alpha}} \tag{72}$$

where λ^c is a scaling factor. Fig. 25 depicts the effective value of the resulting term for all connections.

Chemical repulsion exists in the retinotectal projection [54][3] and graded distributions of membrane attached ligands and their receptors have been found in the tectum and the retina, respectively [15][19]. However, direct evidence for chemical guidance in the thalamocortical projection is missing and other mechanisms have been suggested [41].

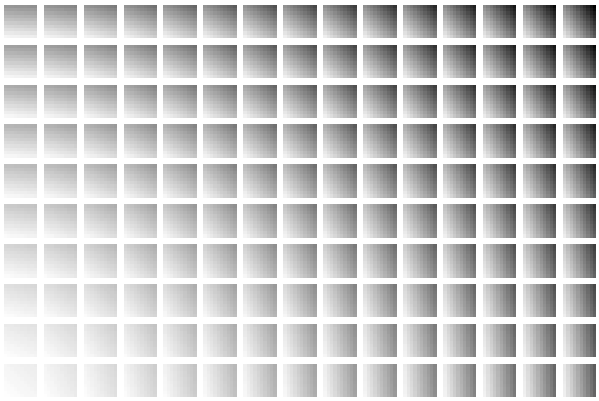


Figure 25: The chemical repulsion term biases weight growth to form a topographic map of a certain polarity. Each hidden unit is depicted by a square which shows the input layer. Concentrations of markers rise from the bottom and the left side to the upper and the right side in both, the input and the hidden units. The product of these values, $c_{\vec{\alpha}}^{input} c_{\vec{x}}^{hidden}$, is displayed dark.

4.6 Maximum entropy model

Using $p^+(x)\,dx = p^+(h)\,dh = p^+(u)\,du$ we have

$$p^+(u) \;=\; \frac{p^+(x)}{\frac{du}{dx}}$$

With Eqn. 13 and $p^-(\vec{h}) = \prod_i p^-(h_i)$ we have

$$\frac{du}{dx} \;=\; \frac{dh}{dx}\frac{du}{dh} \;=\; |W| \prod_i p^-(h_i)$$

and so

$$\begin{aligned} H_u \;&=\; -\int p^+(u)\,\ln \frac{p^+(x)}{\frac{du}{dx}}\,du \\ &=\; -\underbrace{\ln p^+(x)}\; +\; \ln|W| \;+\; \int p^+(u)\,\sum_i \ln p^-(h_i)\,dy \end{aligned}$$

where the underbraced term does not depend on the weights. With $\ln p^-(h_i) = -S(h_i)$ and

$$\frac{\partial S(h_i)}{\partial w_{ij}} \;=\; \frac{\partial S(h_i)}{\partial h_i}\frac{\partial h_i}{\partial w_{ij}}$$

and Eqn. 13 we can write

$$\frac{\partial}{\partial w_{ij}} H_u \;=\; (W^T)^{-1}_{ij} \;-\; \int p^+(u)\,\frac{\partial}{\partial h_i} S(h_i)\,x_j\,du \qquad (73)$$

If we choose, for example,

$$u(h) \;=\; \frac{1}{1+n\,e^{-h}}$$

then we have according to table 4.5.1:

$$-\frac{\partial}{\partial h}S(h) \;=\; \frac{n\,e^{-h}-1}{1+n\,e^{-h}} \;=\; \frac{1+n\,e^{-h}}{1+n\,e^{-h}} - \frac{2}{1+n\,e^{-h}} \;=\; 1-2\,u$$

If we are interested in an on-line rule then the integral over u is well approximated by the occurrences of training data. The learning rule is

$$\Delta w_{ij} \;\approx\; \frac{cof\;w_{ij}}{|W|} \;+\; x_j(1-2\,u_i)$$

This algorithm was published by Bell and Sejnowski [7] and made independent component analysis (ICA) popular. Our derivation follows closer [47]. In order to render the matrix inversion superfluous the learning rule is multiplied by W^TW [1]. This is a positive definite matrix and thus does not change the direction of the gradient. With $h^T = x^TW^T$ we obtain:

$$\Delta W \;\approx\; W \;+\; (1-2\,u)\,h^TW$$

4.6.1 Approximation by a maximum likelihood model

Olshausen [44] shows that using two simplifying assumptions, one can derive from the maximum likelihood framework the same learning algorithm which is derived from a maximum entropy principle.

The assumptions are: *(i)* The number of hidden units equals the number of input units. Furthermore, the matrix V of the data generating process is invertible. *(ii)* There is no noise on the inputs. With these assumptions, we have:

$$\begin{aligned}
P(x) \quad &= \quad \int P(x|u)\,P(u)\,du \\
&= \quad \int \delta(x-Vu)\,P(u)\,du \\
&\overset{u=V^{-1}x'}{=} \quad \int \underbrace{\delta(x-VV^{-1}x')}\;P(V^{-1}x')\;\left|\frac{d(V^{-1}x')}{dx'}\right|\,dx' \\
&\overset{x'\overset{!}{=}x}{=} \quad P(V^{-1}x)\;\left|V^{-1}\right|
\end{aligned}$$

and so

$$\begin{aligned}
\ln P(x) \quad &= \quad \ln P(V^{-1}x) \;+\; \ln\left|V^{-1}\right| \\
&\overset{Eqn.\,68}{=} \quad -\frac{\lambda_u}{2}\sum_i S((V^{-1})_i\,x) \;+\; \ln\left|V^{-1}\right|
\end{aligned}$$

Gradient ascent w.r.t. V leads to the update rule:

$$\Delta V_{ij} \propto -\frac{\lambda_u}{2}\frac{\partial}{\partial u_i}S(u_i)x_j \; + \; \frac{cofV_{ij}}{|V|}$$

which equals Eqn. 73 which was obtained by entropy maximization [4]. This rule has the following advantages over the Kalman filter model:

- No iterative procedure has to be done to find the optimal hidden code vector $\vec{u}$.
- A weight constraint is not necessary, because entropy is maximized exactly when the hidden vectors make up the prior distribution. In contrast, in the Kalman filter algorithm, best results are obtained if the hidden vectors make up the maximum of the prior distribution.

Illustration of entropy maximization Figs. 26 and 27 are the two-dimensional analogue of Fig. 12. They demonstrate that only after un-mixing which happens in a transformation from Fig. 26 to Fig. 27, an input distribution of two neurons can be transformed to a homogenous distribution which has maximum entropy.

The transformation on the data $\{\vec{x}\}$ in Fig. 26 yields a non-optimal output distribution $\{\vec{u}_x\}$. The hidden neurons in Fig. 27 receive unmixed data: the components h_1 and h_2 are made independent of each other, i.e. $P(\vec{h}) = \prod_{\vec{i}} P(h_i)$ by un-mixing them by the weights: $\vec{h} = W\vec{x}$. This is the purpose of the parameter matrix W and the target of learning.

In the maximum likelihood case (generative model) we start with the hidden activations $\{\vec{h}\}$ and transform them directly to the data $\{\vec{x}\}$ via the parameter matrix V. The transfer functions are irrelevant (except that they relate to the prior of the distribution of $\{\vec{h}\}$).

[4] $cofV_{ij}$ is the cofactor of V_{ij} defined by $(-1)^{i+j}$ times the determinant of the $(n-1) \times (n-1)$ matrix formed by eliminating row i and column j from A.

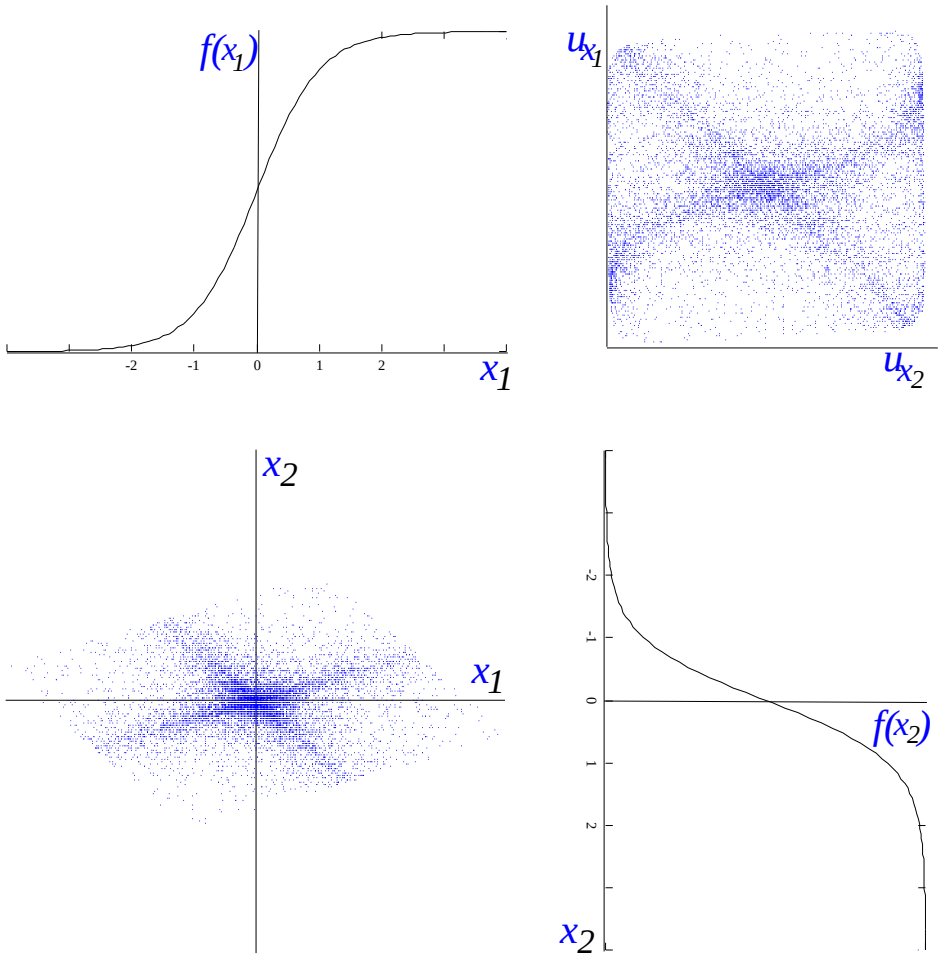


Figure 26: Data points $\{\vec{x}\}$ of two-dimensional example data, **bottom left**. For an experiment of thought (but in none of the models presented) each of the data coordinates are directly transformed via transfer functions $f(x_1)$ and $f(x_2)$ to a distribution $\{\vec{u}_{\vec{x}}\}$, **top right**. The output-distribution is unregular (low entropy), because the inputs show dependencies.

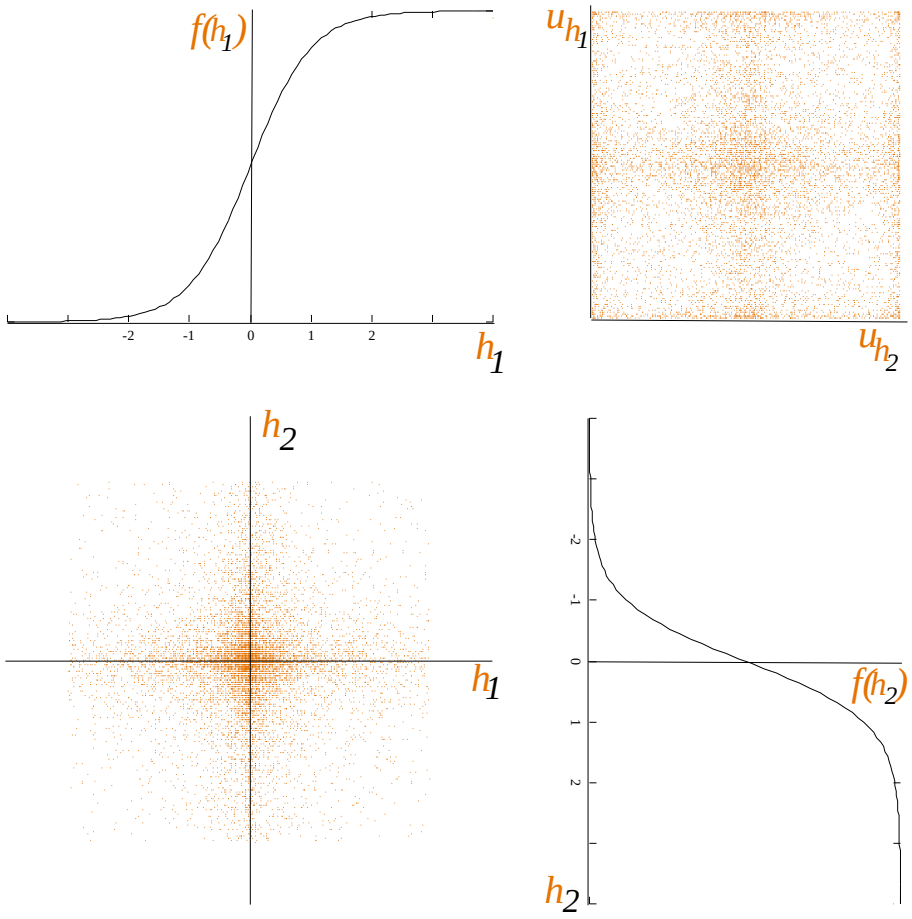


Figure 27: **Bottom left:** inner activations $\{\vec{h}\}$ of hidden neurons after making the components independent through the transformation $\vec{h} = \boldsymbol{W}\vec{\boldsymbol{x}}$. These activations are fed through the transfer functions $\boldsymbol{f}(\boldsymbol{h_1})$ and $\boldsymbol{f}(\boldsymbol{h_2})$ to obtain the output-distribution $\{\vec{u}_{\vec{h}}\}$, **top right**, which is close to a homogenous distribution.

5 Results

In this section we will first describe methods which have not yet been described. Then we show that all of the proposed models are capable of developing feature detectors as, for example, seen in the primary visual cortex and we will explore the model's behavior. Finally, in a more abstract setting, we show that a modular architecture can self-organize.

5.1 Methods

In this section we will consider some simulation details which we did not cover in the theory or model sections.

5.1.1 Preprocessing of images

For practical computation, the data have to be preprocessed and the supplementary steps which are considered here are a collection of heuristics rather than derived from principles. The following steps should be considered if images are used for training.

1. Select grey-scale images of natural scenes. Images should have a rich structure. In particular, monotonous structure (as in artificial environments) or iso-frequency patterns (as grass on a lawn) should be avoided. The more images are chosen, the less will these kind of irregularities corrupt the statistics. Images should never have been compressed (e.g. as a jpeg file) to avoid hidden artificial structure.

2. Subtract the mean luminance of each image from its pixels. This means that darkness is represented by negative activity or, as in our case, this negative activity is associated with (positive) activity of LGN-OFF cells.

3. Sphere each image, i.e. scale the pixels such they have variance 1. This is a convenient range for neuronal activations.

4. Generate each data point by selecting an image randomly and cutting out a random part which has the size of the input layer of neurons.

5. Subtract the mean luminance of each tile from its pixels. Description of the data will hereby be reduced to intensity differences. The biological justification is luminance adaptation within the retina. Trained receptive fields will look less principle component-like. This must not be done for the ICA algorithm. Note that this operation changes the direction of the input vector.

6. Sphere the input vector, i.e. scale the pixels such that the pixels of the individual tile have variance 1. This procedure was dropped for the following reason: those tiles with only little intensity differences are strongly scaled. These are often

the ones where structure just touches the tile at an edge. As a consequence, trained receptive fields show over-proportionally strong weights to the corners.

7. Reject a tile from an image if intensity differences are small, e.g. less than a tenth of the intensity differences of the image. This speeds up training as only data with an interesting structure are used (this should, however, used with care, because it corrupts the statistical structure of the data).

8. With 0.25 probability, rotate the tile by 90° or 180° or 270° in order to obtain more data from a given set of images (this should be omitted if properties which are not invariant to these rotations should be observed).

9. Flip the sign of all activations in a tile (should again be used in an exploratory phase, only). The chosen pictures seem to be rather dark with sparse but strong bright contrasts and thus asymmetric w.r.t. brightness. If hidden neurons have only positive activations, then, omitting this procedure leads to strong positive weights within a small region and weaker and broader negative weights.

The principles underlying these preprocessing procedures apply to all kinds of data: data should be rich, the whole data space should be sampled and also, one can multiply the data if symmetries are considered. Furthermore, the data are scaled in order to ease the change between different data sets. In particular, the learning step size does not need to be changed between data sets if the variance of the data in one input dimension is fixed.

5.1.2 Appropriate weight constraints

A weight constraint term in a learning rule can be derived by a prior on the weights. However, different priors can be chosen. In the following, we explore heuristically whether we should apply the weight constraint to the weight vector of a hidden neuron, i.e. constrain the receptive fields of V1 cells, or whether we should constrain the weight vector of an input neuron, i.e. constrain the receptive fields of LGN cells. Because of weight symmetry, one of these two constraints is necessary only.

Constraint on the LGN field of a V1 cell:

$$P(\vec{v}_{\vec{x}}) \approx e^{-\frac{d_v}{4}\sum_{\vec{\beta}} v_{\vec{\beta}\vec{x}}^2 \sum_{\vec{\gamma}} v_{\vec{\gamma}\vec{x}}^2}$$

Additive term for the slow dynamics:

$$\Delta v_{\vec{\alpha}\vec{x}}^{const} \approx -d_v v_{\vec{\alpha}\vec{x}} \sum_{\vec{\beta}} v_{\vec{\beta}\vec{x}}^2$$

$$w_{\vec{x}\vec{\alpha}} = v_{\vec{\alpha}\vec{x}}$$

Constraint on the V1 field of a LGN cell:

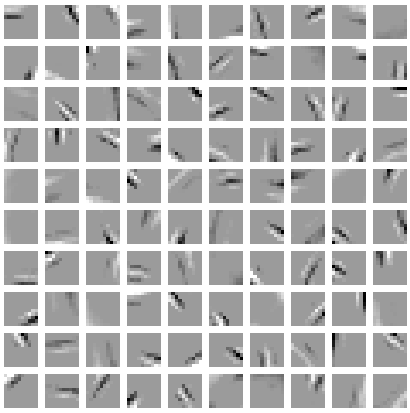
$$P(\vec{v}_{\vec{\alpha}}) \approx e^{-\frac{d_v}{4}\sum_{\vec{y}} v_{\vec{\alpha}\vec{y}}^2 \sum_{\vec{z}} v_{\vec{\alpha}\vec{z}}^2}$$

Additive term for the slow dynamics:

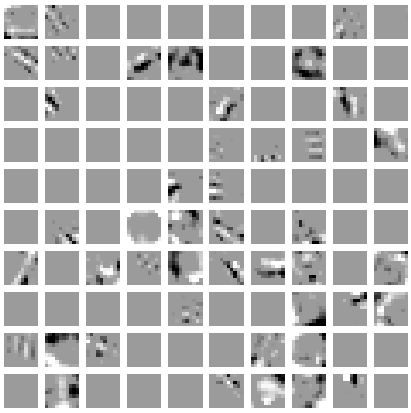
$$\Delta v_{\vec{\alpha}\vec{x}}^{const} \approx -d_v v_{\vec{\alpha}\vec{x}} \sum_{\vec{y}} v_{\vec{\alpha}\vec{y}}^2$$

$$w_{\vec{x}\vec{\alpha}} = v_{\vec{\alpha}\vec{x}}$$

Result:

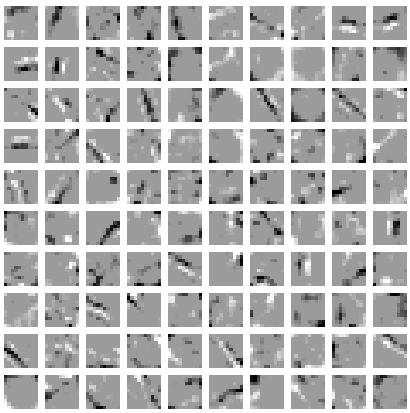


Result:

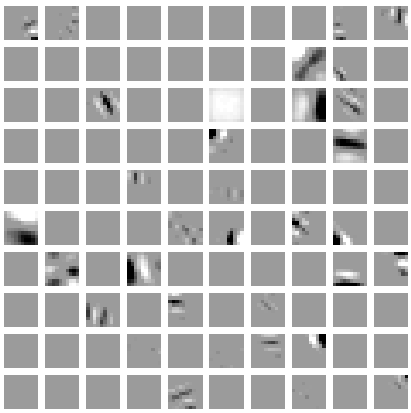


The results clearly show that the weight vector of the hidden units which spans across the input units has to be constrained. The constraint which we use does not change the direction of this vector. If the weight vector of the input units which spans across the hidden units is changed then the direction of the hidden units weight vector is changed by the constraint and the receptive fields are poor edge detectors.

A biologically more plausible strategy is to abandon weight symmetry and treat the recognition weights and the generative weights separately. Separate weight constraints are then used for both weight sets. In the following we will explore two paradigms, a “dendritic” constraint on a receptive field of a cell and an “axonic” constraint on the projective field of a cell. The following results are obtained.



The receptive field within the input of a hidden neuron is constrained via the recognition weights $\boldsymbol{W}$ and
the receptive field within the hidden units of an input cell is constrained via the generative weights $\boldsymbol{V}$.



the projective field within the hidden units of an input cell is constrained via the recognition weights $\boldsymbol{W}$ and
the projective field within the input units of a hidden cell is constrained via the generative weights $\boldsymbol{V}$.

These results show that the “dendritic” constraint, on the left, leads to better results than the “axonic” constraint. However, the receptive fields of the hidden neurons are not such clean edge detectors as when symmetric weights are used.

5.1.3 Suppressing divergence (Kalman filter model)

For biological modeling on a macroscopic scale, networks are usually chosen as large as tolerable for computation. Many commonly used algorithms, however, are tested using small nets only and have to be modified for large networks.

For a large-scale simulation of the Kalman filter model we made modifications on both steps of the learning procedure, the fast dynamics and the slow dynamics:

- The fast dynamics (activity relaxation) involves computation of the reconstruction error on the input units and an incremental update of the hidden unit activations (cf. Eqs. 37,69):

$$\begin{aligned} \tilde{x}_j &= x_j - \eta\, \vec{v}_j \cdot \vec{u} \\ \Delta u_i &= \vec{w}_i \cdot \tilde{x} - \lambda_u \frac{u_i}{1+u_i^2} \end{aligned}$$

  The first equation computes the reconstruction error. For small nets (< 150 Neurons in input and hidden layer) η equals 1. Recursive update thus corresponds to fix-point iteration. For large nets an overestimation of the prediction, i.e. $\vec{v}_j \cdot \vec{u} > x_j$, leads to oscillating activations. This can be suppressed by $\eta < 1$, typically $\eta = 0.5$.

  The second equation recalculates the internal representation $\vec{u}$. Hereby $\vec{u}$ converges to $\vec{u}^{opt}$ which maximizes the *likelihood* to generate $\vec{x}$ at fixed V.

  Convergence is not suppressed by $\eta < 1$ but the optimal hidden code will be obtained later. Furthermore, for computational reasons, the number of iterations are limited (5 to 10 iterations are usually performed) which impairs our estimation of the optimal hidden code.

- The slow dynamics is to learn the weights using our estimation of the optimal hidden code $\vec{u}^{opt} := \vec{u}$ which has been found through the fast dynamics (cf. Eqs. 38,70):

$$\Delta \vec{v}_j = \epsilon\, \tilde{x}_j\, \vec{u}^{opt} - d_v\, \vec{v}_j\, \vec{v}_j^2$$

  For optimal reconstruction, the recognition weights are usually set equal to the generative weights: $W = V^T$ (weight symmetry). Only one set of weights is then actually trained and the "inverse" weights are hereby determined. Alternatively, symmetric weights are obtained by a symmetric initialization and if both sets are trained by Hebbian learning only. The Hebbian term is symmetric. However, the postsynaptic constraint $d_v \vec{v}_j \vec{v}_j^2$ which has to be applied effects dendritic input (of LGN cells) and destroys symmetry. For biological plausibility, also the feed-forward weights W should have a postsynaptic constraint (on the input of V1 cells):

$$\Delta \vec{w}_i = \epsilon\, \vec{x}\, u_i^{opt} - d_w\, \vec{w}_i\, \vec{w}_i^2$$

  This gives two constraint parameters, d_v and d_w, which can differ if we assume that LGN and V1 cells behave differently.

Choosing $d_v > d_w$ leads to weights $\vec{v}_j$ smaller than weights $\vec{w}_i$ and thus makes the prediction $\vec{v}_j \cdot \vec{u}$ smaller. This furthermore prevents oscillating behavior because it reduces overestimation of the data.

Applying both modifications on the algorithm and using a smaller learning step ϵ than usual, we are able to learn nets of size 400 (input) $\times 900$ (hidden neurons). However, this scale is not suited for parameter explorations because computation time increases too much.

5.2 Feature detectors

For computational modeling of cortical features, we will concentrate on the LGN-V1 projection. First, because the input data are simple, consisting of only weakly preprocessed natural images. Secondly, cell properties of neurons in the input layer (layer IV) of primary visual cortex (area V1) are well described. The most striking characteristics of V1 cells on which we concentrate here are orientation preference and a topographic arrangement w.r.t. the visual field (see chapter “The cortex”).

5.2.1 Localized edge detectors

Fig. 28 on the following page shows a large scale Boltzmann machine which has developed edge detector shaped receptive fields.

As usual, weights have been initialized with small random values which were homogeneously distributed within an interval around zero. Learning (Eqn. 58 on page 45) consists of two parts: one clamped "relaxation" was done with an image, which consisted of one feed-forward sweep (Eqn. 13 on page 27 with the mean activation transfer function Fig. 17 on page 48). The free-running relaxation was done with 15 relaxation iterations and again the mean transfer function for the hidden units but the linear function for the input units. With the resulting activations an update of all weights was performed. The learn step ϵ for the weight update was set to 0.005 (cf. Eqn. 58 on page 45).

The number of relaxation steps is not crucial for the results, even though the minimum of the energy function will not be reached, because sparseness is mediated to the activity state by the derivative of the transfer function of the hidden neurons: the smallest activations have a stronger tendency to decrease than larger values. Sparseness is thus imposed on the net even if the relaxation procedure does not converge.

The number (degeneracy) of zero-activity states of a hidden neuron is $n = 10$ (cf. Eqn. 62 on page 50). This leads to a Fisher kurtosis (Eqn. 60 on page 47) of 3 for the activity prior on a neuron. Finally, the constraint on the weights is scaled by $d_w = 0.25$ (cf. Eqn. 70 on page 60).

Fig. 28 on the following page shows the network after training on natural images. The characteristics of the receptive fields changes with the length of the weight vectors. The largest vectors cover the entire input field with low frequency and a weak

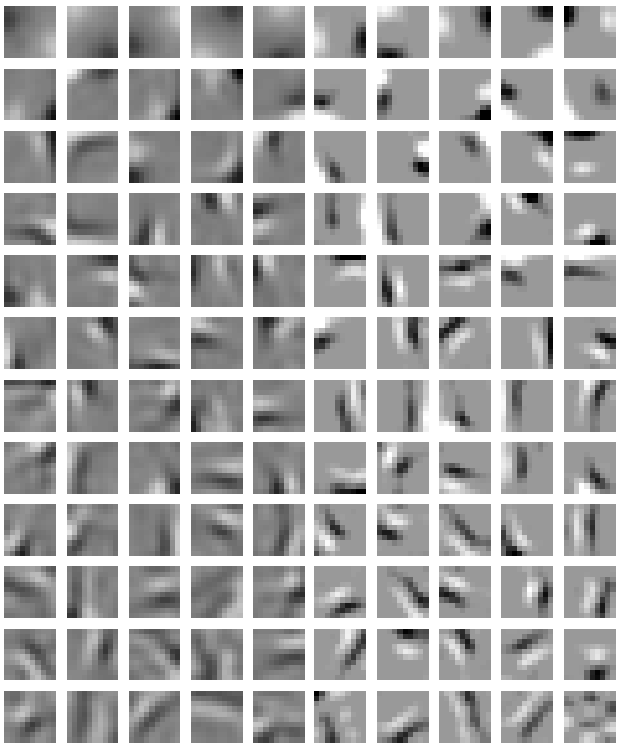


Figure 28: The weight matrix of a Boltzmann machine after training. Each square shows the receptive field of one of the 12×10 hidden neurons, black indicating negative, white positive weights to one of the 10×10 input neurons. Fields are ordered left-to-right, top-to-bottom in the order of the length of the weight vector. A larger number of hidden neurons than input neurons allow for an *overcomplete representation* of the input. **Left half:** brightness values are linearly proportional to weight values. **Right half:** brightness values are obtained through piecewise linear functions. Strong weights look black or white and weights weaker than 10 percent of the maximum weight value are not distinguished from zero. The zero value is brightened.

tendency for localization. The majority of smaller vectors form higher frequency edge detectors which are well localized within the input field. Together, the space of position, orientation and scale is roughly homogeneously filled by weight vectors.

5.2.2 Auditory feature detectors

Fig. 29 on the next page shows the network (sparse Kalman filter) after training on auditory signals. Signals were 24 sounds which were each decomposed into 30 frequency bins at 90 time steps.[5] Input to the network was then a 30×30 vector which

[5] I thank Holger Prante for giving me these data.

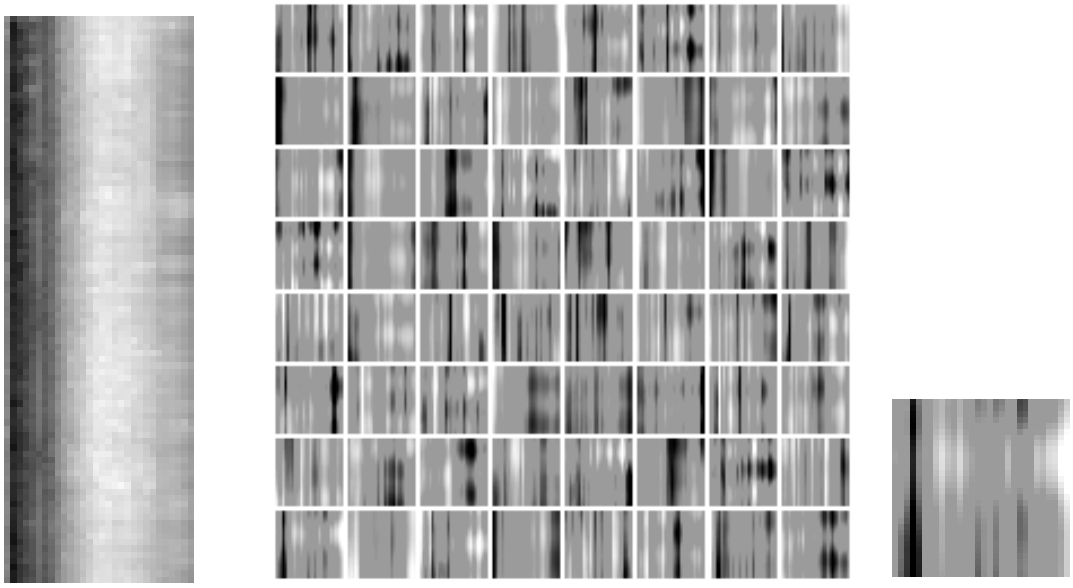


Figure 29: **Left:** an example auditory stimulus. Brightness codes the amplitude of a frequency. The frequency axis is from left (low frequencies) to right. Time advances from top to bottom. The input to the network is a randomly chosen time slice. **Middle:** the weight matrix after training on auditory signals. Strong weights are black, negative weights are white, zero weights grey. **Right:** an example receptive field (from bottom left of the map). Within each field, time advances from top to bottom, frequency from left to right.

represented the amplitudes of the 30 frequencies at 30 consecutive time steps. The mean of the input values was subtracted for each data point and values were divided by their standard deviation. The receptive fields show more pronounced contours than the original stimulus and a relatively large number of weights is zero.

5.3 Running the Boltzmann machine

5.3.1 Learning in the clamped and in the free-running phase

Fig. 30 on the following page, top row, shows some receptive fields of a training result if only the clamped phase is used for learning. Hidden neurons do not interact in this phase. As all of them see the same data their weight vectors will learn the same direction (except for the sign), i.e. identical fields develop. It is known (see standard literature, e.g. [27]) that the linear Hebbian neuron learns the first principle component of the data. Fig. 30 on the next page, top right item, shows the corresponding result which we obtained by training one linear hidden neuron using only the clamped phase. Similarity of both results demonstrates that the clamped phase corresponds to simple Hebbian learning of linear neurons.

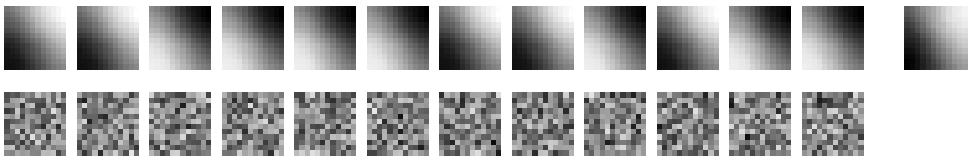


Figure 30: **Top:** a row of receptive field of a net which had been trained by the clamped phase only. **Top, right:** the receptive field of an alone-standing linear neuron for comparison. **Bottom:** a row of fields of a net which had been trained by the free-running phase only.

The last row of Fig. 30 shows a training result if only the free running phase is used for learning. Note that this does not involve any data. In order to prevent all weights from converging to zero we changed the correction to the learning rule for constraining the weights to

$$\Delta w_{ij}^{constr'} \approx +d_w \frac{w_{ij}}{\sum_{j'} w_{ij'}^2}$$

Using $d_w = 0.25$, maximum weight values of the trained net were slightly larger than for the normally trained net.

The field-vectors of hidden cells were maximally different, i.e. perpendicular. The average scalar product between normalized weight vectors[6] was well below 0.0001 if as many hidden neurons as input dimensions were chosen and after η was reduced. The obtained value was 0.033491 for overcomplete coding of above architecture. For a normally trained net we obtained 0.136125 and for the net which was trained by the clamped phase the overlap was 1.0.

5.3.2 Non-convergence of weights

After receptive fields reach their mature appearance there is still fluctuation in the weights. In order to rule out the influence of stochastic fluctuation in the data we trained the net for a test with a fixed set of 1000 data points. The total weight change upon showing the full set each time is plotted over time in Fig. 31 on the next page. It shows that the weight change does not converge to zero (solid, top line).

If the algorithm is "split" into the two phases one can see that the net which is trained by only the clamped phase does converge (dotted, lower line). Learning by the free-running phase, however, does not lead to a stable state; weights change continuously at a baseline level after initial maturation (dashed, middle line).

An interpretation of this phenomenon is the following: every attractor in the free phase will be weakened by anti-Hebbian learning. Other attractors will emerge. Every possible state thus should at some time become an attractor. Our net has 3^{N+H}

[6] We define the average mutual overlap of normalized receptive fields by: $\frac{1}{H\,(H-1)} \sum_{i'}^{H} \sum_{i''\neq i'}^{H-1} \frac{|\vec{w}_{i'}^T \vec{w}_{i''}|}{||\vec{w}_{i'}|| \, ||\vec{w}_{i''}||}$.

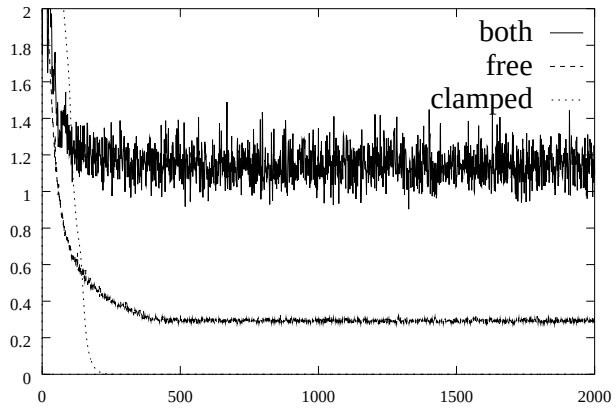


Figure 31: Total squared weight change $\sum_{i,j}(w_{ij}^{new} - w_{ij})^2$ over time. w_{ij}^{new} are the weights after an always repeating set of data is once shown for training. Solid line (top): learning by the complete algorithm; dashed line (middle): learning by the free-running phase only; dotted line (bottom): learning by the clamped phase only leads to convergence. The same learning step size ϵ has been used in all three cases.

possible different states, which are many more than relaxations can be done. In this case we will hardly observe weights to return to former values.

The effectiveness to weaken existing attractors should depend on the effectiveness to reach them in the free-running phase. For example, a mean-field relaxation procedure with simulated annealing would reliably find the optimal attractor. This would happen repeatedly until it finally vanish to be the glogal optimum by anti-Hebbian learning, independent of the learning step size ϵ.

This “try-something-new” principle of anti-Hebbian learning continues in the presence of the clamped phase with Hebbian learning. Even when the task to generate certain data constrains the weights, the optimal setting is still ambiguous: for example hidden neurons can exchange their weights (as if the indices of two neurons are exchanged). The “try-something-new” principle tells them to do so continuously.

5.3.3 Non-convergence of activities

It takes many iterations to relax activations $(\hat{x}, \hat{u})$ to reach a fix-point such that Eqs. 62 on page 50 and 63 on page 50 are consistent. For computational reasons the estimations for the mean activations are taken from an interrupted relaxation procedure. The above result was obtained with 15 iterations only for each data point or free relaxation procedure.

Relaxation interruption does not impede the effects of sparse coding: already the tendency of smallest activations to decrease fastest due to the mean-field transfer function (not taking into account very large values in the saturation regime) enforces sparse coding.

Fig. 32 on the following page plots the summed activation updates over time for

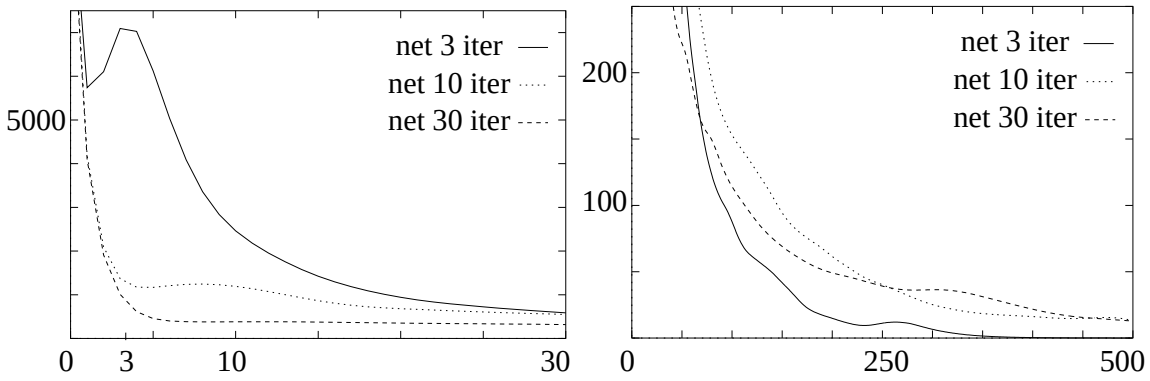


Figure 32: Summed absolute values of activation changes averaged over 70 relaxation entities over the relaxation progress. Three nets were relaxated which have been trained using different total relaxation times in the free running phases: 3 iterations (solid line), 10 iterations (dashed line) and 30 iterations (dotted line). **Right:** same as left but over longer time and with a different scaling.

three nets, each have been trained using different relaxation times: 3, 10 and 30 iterations. There is a tendency for nets which are trained with many iterations to relaxate slow at the beginning and to converge later. Therefore it seems impossible to extend the relaxation time used for learning until convergence.

The weights seemingly adapt to the time of the interruption of the relaxation procedure: the data should be best modeled after the number of iterations which are used for learning. Some hidden unit activations tend to relax towards zero, others towards "extreme" values in the saturation regimes of their transfer function. It seems, however, more favorable to model the continuous valued input data with a broad range of hidden unit activations. The weights develop such that hidden unit activities have not converged at the number of iterations which are used for learning.

5.3.4 The effect of the inverse temperature / weight length

As shown previously, weight length and inverse temperature are dependent parameters for the case of recurrent computations with the mean-field approximation. Setting these parameters properly leads to the following dilemma: *(i)* Small weights (small β) lead to underestimation of the data. In order to reconstruct the gain of the data the receptive fields will shape like the first principal components upon learning. *(ii)* Large weights (large β) leads to a data reproduction using small hidden activations. Such values are non-stationary by the relaxation dynamics.

In the case of Gibbs relaxation of the Boltzmann machine $\sqrt{\beta}$ and l are independent. It is now possible to decrease β (make the net prefer "zero"-states) without underestimating the data as with too small feedback weights.

5.3.5 Generation of images

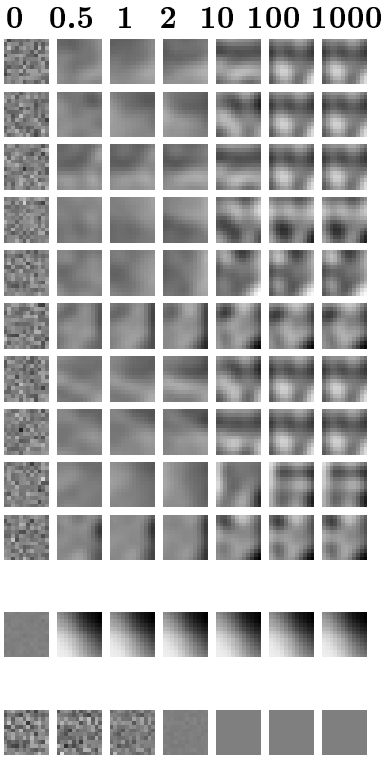


Figure 33: Each row represents the development of activations on the input sheet. The numbers which mark the columns give the iteration number in multiples of the relaxation time which was used for learning, starting with initial random activation at iteration 0. Images are generated by: top ten rows: a normally trained net; second last row: a net trained by the clamped phase only; last row: a net trained by the free running phase only.

Activations $\hat{x}$ of input neurons of a trained net during ten test relaxations are depicted in Fig. 33, top. One can clearly see that activations continue to change after the number of iterations used for training, some considerably. During early times activations differ, still influenced by the random initialization. Finally they converge to a small number of attractors (we can count three different states in the last column). They are to be compared with image patches from the training data.

The net which has been trained in only the clamped phase generates images, the development one of which is shown in the second last row of Fig. 33. Certainly, as they are composed of weight vectors (projective fields) which all resemble the first principle component of the data, they have this appearance.

Fig. 33, bottom row, shows images which are generated by the net trained in the free-running phase. The observations are *(i)* activation strengths converge towards zero and *(ii)* qualitatively, the appearance of the activation pattern (direction of the

vector) changes only little. The latter observation emphasizes that learning by free-run clears must have cleared away strong attractors and thus supports there to be a greater variety of attractors, i.e. nearly every state is an attractor.

5.3.6 Sparse coding in the Boltzmann machine and Kalman filter model

In order to run the Kalman filter model using the discrete activations $\{-1, 0, 1\}$ for the hidden units as in the Boltzmann machine, we chose the prior to consist of three Gaussian functions, Fig. 34 on the following page, top right. The figure also depicts the corresponding transfer function and the observed distribution of hidden unit activations. The observed distributions of the sparse Boltzmann machine are shown in Fig. 35 on page 81.

We see that the regular Kalman filter model distribution corresponds to the continuous distribution of the mean-field Boltzmann machine. However, the Boltzmann machine does not allow activity values larger than 1. The distribution of the Kalman filter model with three Gaussian functions as a prior corresponds closely to the distribution of the regular (stochastic) Boltzmann machine. However, for two reasons, other values than the three maximum values of the Gaussians still do occur. First, the relaxation procedure is prematurely terminated, so that the optimal values are not found. Second, it is not the prior distribution of the hidden states which is maximized but the posterior given a data point. Thus, the prior and the likelihood together are to be maximized.

5.4 Topographic mappings

It has been shown by the author and others [69],[70],[57],[58],[68],[9] that the mechanisms proposed in section “Priors for topographic mappings” can account for robust topographic mappings even under experimentally altered conditions. In these models, the most important factor during refinement of the topographic maps was observed to be neural activity.

The models were bottom-up approaches to understand the system and the equations were thus heuristic. Nevertheless, the terms in the growth equations were carefully selected to yield a description which is as small and as simple as possible. As a consequence, simple growth terms were additively combined to yield the full growth equation. They were: *(i)* an intrinsic fiber-target interaction which formulates chemospecific adhesion between afferent fibers and target cells, *(ii)* an activity-dependent fiber-fiber interaction which implements correlation-based Hebbian learning, *(iii)* constraints on the synaptic weights. These consist of a constant growth term and of pre-synaptic as well as postsynaptic decay. Finally, *(iv)* in order to simulate one experiment which has been done on the retino-tectal projection only, there has been introduced an intrinsic fiber-fiber interaction which describes mutual selective adhesion between the afferent fibers.

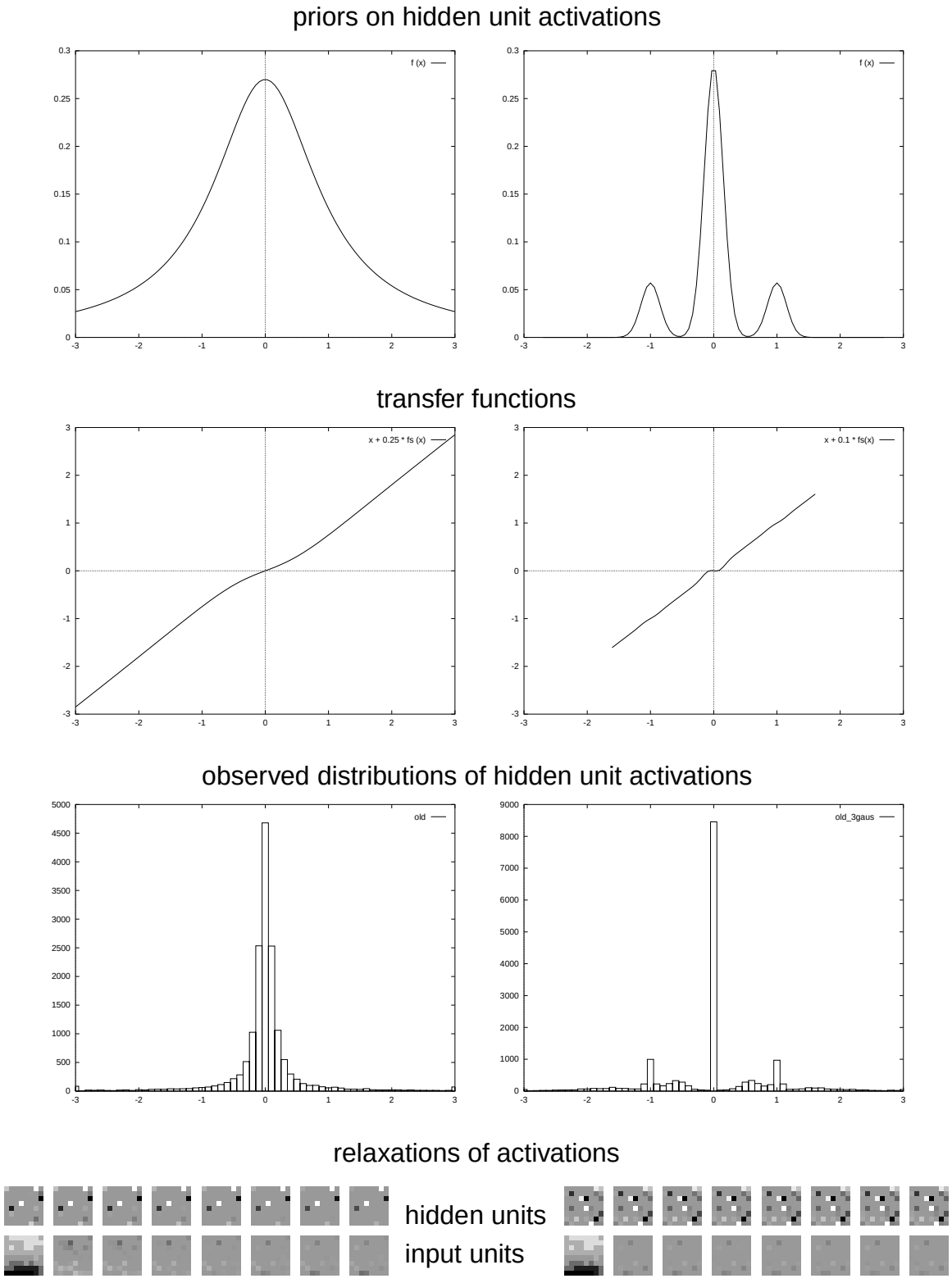


Figure 34: Kalman filter model. **Left:** sparse prior. **Right:** 3 Gaussians as prior.

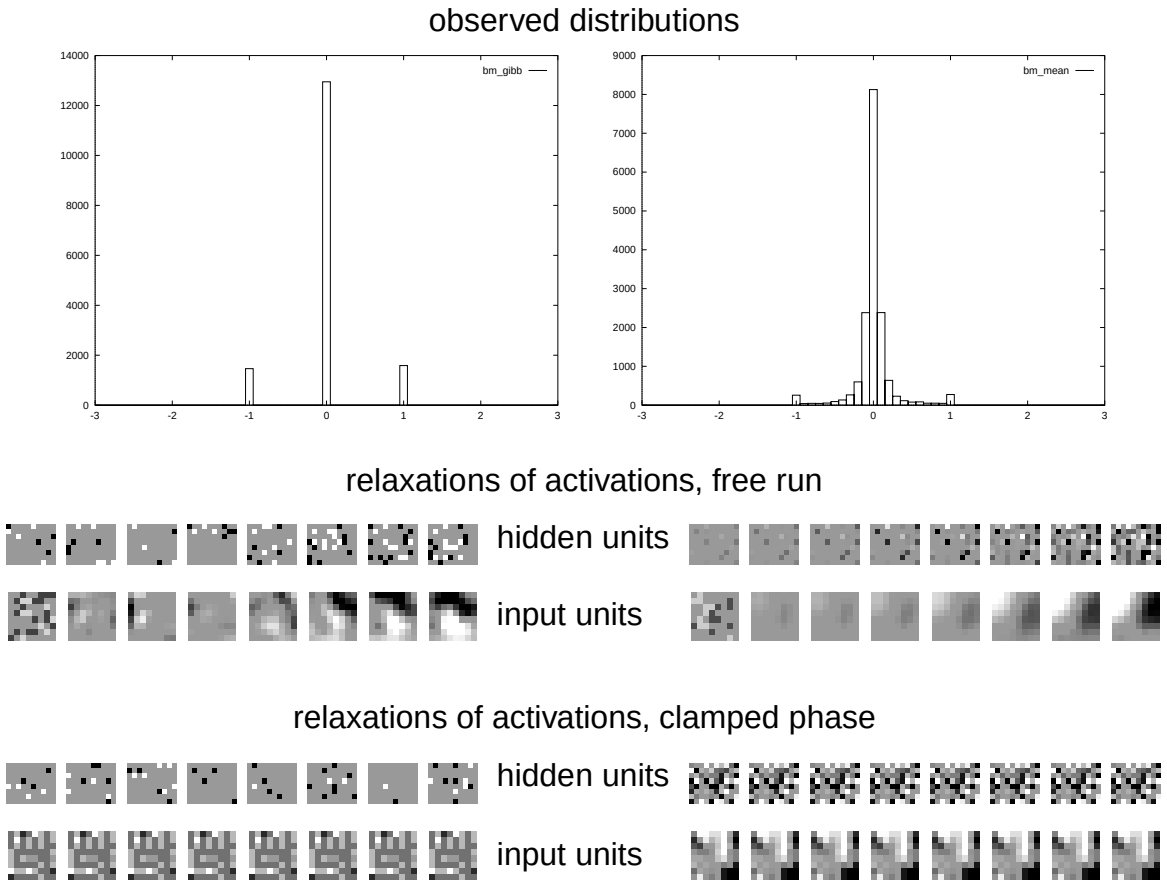


Figure 35: Boltzmann machine. **Left:** Gibbs sampling, **right:** mean-field approximation.

In this work, we use maximum-likelihood derived algorithms in the sense of a top-down approach. We have seen that activity dependent mechanisms, i.e. Hebbian learning, coincide with information theoretic goals. In addition, recurrent and feed-forward models with learning rules derived by a maximum-likelihood or a maximum-entropy principle develop target neurons with orientation preference. In contrast, neurons in the abovementioned models develop circularly symmetric receptive fields. Thus it will be interesting in the case of non-symmetric receptive fields how these fields will arrange if they are driven to form a topographic map.

We used the recurrent Kalman filter model as a basis for topographical modeling (Fig. 11 on page 27). For each image vector which is presented on the input layer, first the algorithm iteratively relaxates activities (Eqn. 37 on page 38) to obtain the optimal hidden code to predict the image, given a prior on the activities for *sparse coding* which is scaled by $\boldsymbol{\lambda}^{s}$.

We add two biologically inspired mechanisms to the learning rule for the weights: *(i)* A diffusion term [42] of each weight to its neighboring cells in V1. The noise in the

precision of wiring due to this effect is expressed by the term $\lambda^d \cdot \frac{\partial^2 w_{\vec{x}\vec{\alpha}}}{\partial \vec{x}^2}$. The effect of this term is comparable to the effect of activity diffusion (Eqn. 71 on page 61) but it is faster to compute if activations have to relax in several steps. *(ii)* A chemical repulsion term (Eqn. 72 on page 62, Fig. 25 on page 63) proportional to graded concentrations of chemical markers along each axis of the LGN (c^{LGN}) and V1 (c^{V1}) accounts for the polarity of the topographic projection. The soft constraint on the weights which is scaled by $\lambda^w = 10.0$ is added to Eqn. 38 on page 38: $\Delta w_{ij} \propto -\lambda^w w_{ij} \sum_{j'} w_{ij'}^2$.

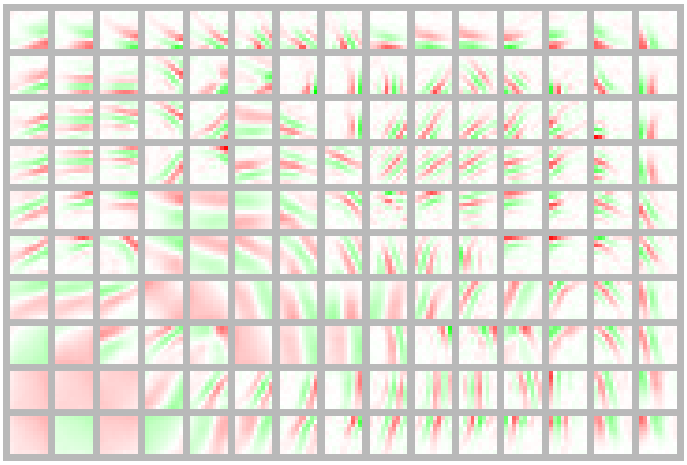


Figure 36: The resulting map where the chemical term was active, $\lambda^c > 0$, but without diffusion term, $\lambda^d = 0$. Green indicates positive, red negative connection strengths to the input neurons. Receptive fields are mostly topographically arranged edge detectors but there are discontinuities from one V1 cell to its neighbors.

Figs. 36 and 37 on the next page show the map after training on natural images where each square shows the receptive field of a hidden neuron. A larger number of hidden neurons, 10×15, then input neurons, 10×10, form an *overcomplete representation* of the input.

The properties of the receptive fields of the target neurons, retinotopy and orientation preference, change more or less continuously along space impeded mainly by the small net size. The map in Fig. 36, however, was obtained without a diffusion term, i.e. $\lambda^d = 0$. Only the chemical repulsion term was present, $\lambda^c = 2.0$ initially, but it decreased linearly towards zero during the simulation. Surprisingly, even without direct neighbor interaction, the properties of the receptive fields change continuously across space in large areas. There are discontinuities, as well. A possible explanation for this result is the following: while $\lambda^c > 0$ was decreasing, more and more connections could survive. The continuity of this process follows the continuity of the chemical repulsion function.

In the case with diffusion, the receptive fields seem to be superimposed by neighboring fields which adds noise to the elsewise Gabor-like receptive fields.

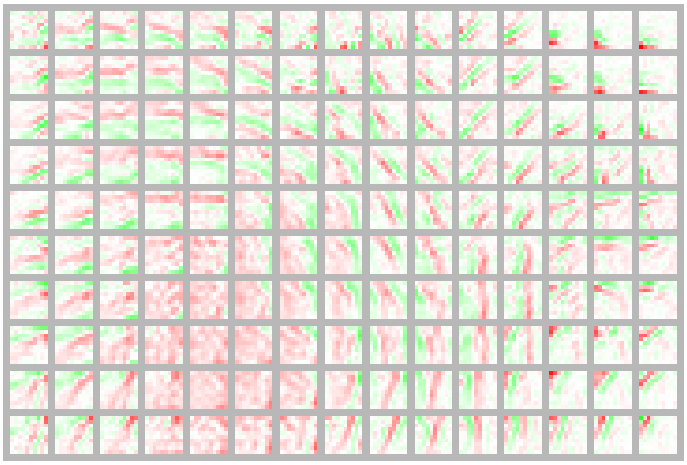


Figure 37: The resulting map with λ^c as above and with diffusion, $\lambda^d = 5 \cdot 10^{-6}$. The properties of the receptive fields change continuously in a pattern which seems to be larger than that observed above. There are no marked discontinuities, but "superposition" of receptive fields impairs maturation of "clean" edge detectors.

5.5 Modularization

5.5.1 Generation of parallelly and of hierarchically organized data

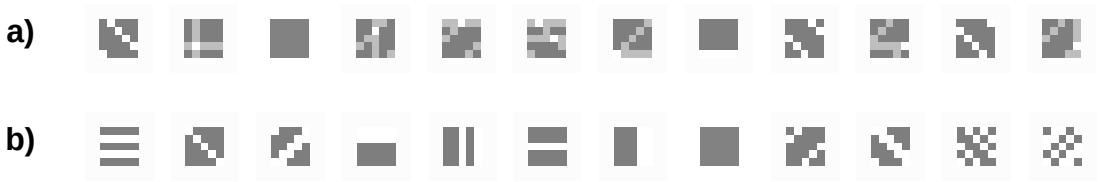


Figure 38: Examples of stimuli $\vec{x}$ used for training. **a)** Stimuli generated by two models in parallel. **b)** Stimuli generated by a hierarchical model.

Artificial data are generated by two different paradigms. First, in a non-hierarchical manner, data consist of discrete, sparsely generated elements. These elements are lines of 4 different orientations on a 5×5 grid of input units resulting in a total number of 20 different elements. For the first experiment each line is chosen with a fixed probability independently of any other. Thus there is no structure among the code elements. For the purpose of structuring our network into two distinct groups of hidden neurons we form two groups of the elements. One group, horizontal and $45°$ lines are generated with probability 0.1 whereas the other group, vertical and $135°$ lines are generated half as often, with probability 0.05 each. Fig. 38 a) shows example data.

For the second experiment data is generated in a hierarchical manner [32]. First, one of 4 orientations are chosen, which represents a decision process within a higher

hierarchical level. Then, on the lower level, lines from the formerly chosen orientation only are generated with probability 0.3 each, i.e. for each of the 5 positions on the input array a line is switched ON with this probability. Fig. 38 on the preceding page b) shows example data.

5.5.2 Results from the Kalman filter model

Weights were initialized with small random values with mean zero. Then the following on-line learning procedure was repeated $5 \cdot 10^6$ times. A data point was shown to the input neurons and Eqs. (44 on page 42, 45 on page 42) were iterated 10 times after initialization of hidden unit activations with zero to obtain an approximate estimate for the best hidden code. Using these values the weights were trained according to Eqs. (46 on page 42).

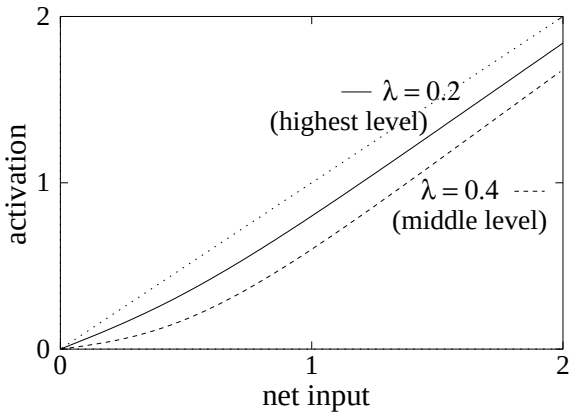


Figure 39: The transfer functions of the hidden units with different sparsity parameters λ_u.

Distinguishing the modules Fig. 39 displays the transfer function (Eqn. 69 on page 58) for two different sparseness values λ_u. The solid line shows the transfer function of neurons which are designed for the hierarchically higher level. The function displayed as a dashed line is intended for middle layer neurons. Higher level units fire more often because one of the four orientations occurs more often in the data set as one of the four$\times$five lines. The sparsity parameters in the simulations are chosen as: $\lambda_u = 0.2$ for one half of the hidden neurons, $\lambda_u = 0.4$ for the others.

Other parameters are: step-size for the update of activations $\varepsilon_u = 0.1$, tradeoff for bottom-up/top-down input $\beta = 0.9$, learning step-sizes $\varepsilon_{w01} = 0.03$, $\varepsilon_{w11} = 0.003$, constraint on the weights $d_w = 0.03$.

Areas organize in parallel The net which had been trained on the parallel data extracted all lines from the data. Fig. 40 on the following page a) shows some example

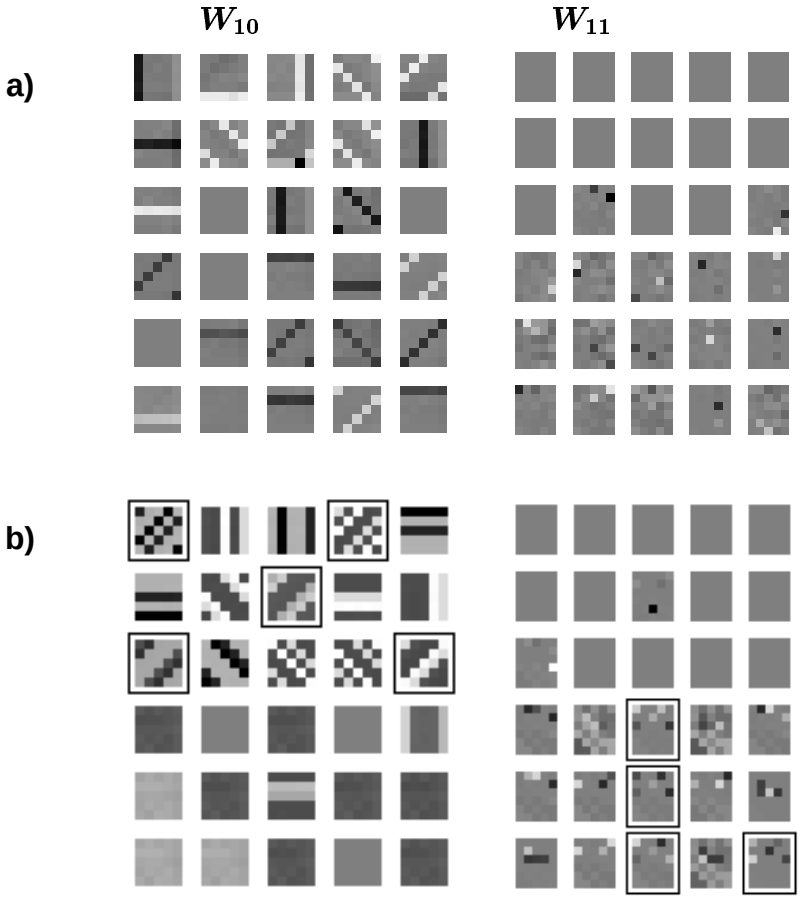


Figure 40: **Left:** the recognition weight matrices W_{10} and **right:** the lateral recognition weight matrices W_{11} after training. Each square of the weight matrices shows the receptive field of one of the 5×6 hidden neurons, black indicating negative, white positive weights. **Middle**, weights to one of the 5×5 input neurons and **right**, lateral weights. A larger number of hidden neurons than input neurons allow for an overcomplete representation.
a) Parallel organization of areas: weights W_{10} to the inputs concentrate on $0°$ and $45°$ lines in the lower half and on $90°$ and $135°$ lines in the upper half. **b)** Hierarchical organization of areas: neurons in the upper half code for the input via W_{10} while neurons in the lower half organize the code from the upper units via W_{11}. Neurons which code for lines of $45°$ orientation are marked by a frame.

data and the weights after training. Each code element (one of the 5×4 possible lines) is represented by a weight vector of the matrix W_{10}. Due to overcomplete coding a small number of neurons is not connected to the input and some lines are represented by two hidden neurons. The bottom half of the hidden neurons which have stronger activations ($\lambda_u = 0.2$) specialize on the input features which occur more often (lines of $0°$ and $45°$). The upper neurons which have weaker activations

($\lambda_u = 0.4$) specialize on the rare features (lines of $90°$ and $135°$). On both layers, there are a small number of exceptions from the rule.

If prior knowledge that the data has no hierarchical structure was assumed then $\vec{u}_2$ would not have to be computed by Eq. (45 on page 42). For consistency with the next experiment, however, we included the second hierarchical level and $\vec{u}_1$ takes into account $\vec{u}_2$. Hereby lateral weights W_{11} emerge through which in general the activation $\vec{u}_2$ of one neuron supports the activation $\vec{u}_1$ of an arbitrary other neuron. The core result of this experiment as described above is unchanged if a second hierarchical level is omitted (results not shown).

Areas organize hierarchically The net shown in Fig. 40 on the preceding page b) was trained on the hierarchical data set. It has structure among the weights W_{10} to the input in the area with less active neurons, and has structure among the lateral recognition weights W_{11} in the area with stronger activation. The former neurons have not discovered single lines as input elements because a larger number of them were presented which have the same direction compared to the first experiment (the average number of lines in each stimulus is 1.5 in both settings). Thus, elementary features could as well be the dark lines in between. Another argument for a different representation is that the less active neurons form an undercomplete representation of the input.

Neurons in the more active area join together those neurons in the less active region which code for the same orientation by the lateral weights W_{11}. As the more active neurons they code for the orientation of a stimulus because one of four orientations is statistically chosen more often than a single line.

We did not adjust the sizes of the areas to the expected outcome which we could have done by setting the parameters of exactly four neurons to have strong activations. As a consequence of a too large number of highly active neurons more neurons redundantly represent the second hierarchical level.

5.5.3 Results from the Helmholtz machine

Concatenation of hierarchical levels Our model is a Helmholtz machine with two hidden layers. However, from the architecture or the algorithm only, the two hidden layers cannot be distinguished. The two layers are concatenated to one hidden layer (Fig. 11, top). Lateral weights, both recognition W_{11} and generative V_{11}, as well as possibly missing (zero-value) weights to the input allow some neurons to represent the code of other hidden neurons and thus to belong logically to a second hierarchical level.

Recognition of data involves two consecutive steps which distinguish the two hidden layers: activations $\vec{u}_1$ evoked from data $\vec{x}$ via input weights W_{10} are assigned to the first (lower) layer. Activations $\vec{u}_2$ evoked from $\vec{u}_1$ via lateral weights W_{11} are assigned to the second (higher) layer.

Data generation also distinguishes two hidden layers: a hidden code $\vec{s}_2$ on the logically higher hidden layer evokes via lateral, generative weights V_{11} a hidden code $\vec{s}_1$ on the logically lower layer. Reconstructed data $\vec{s}_0$ is then generated via V_{01} using $\vec{s}_1$.

Note that a neuron can be active at both times consecutively. In order to discourage this during recognition, an activity-dependent weight constraint (Eq. 82 on page 89) introduces competition between all incoming weights of a hidden neuron. This encourages a hidden neuron to receive input from the input neurons via W_{10} or from other lateral neurons via W_{11} but not both.

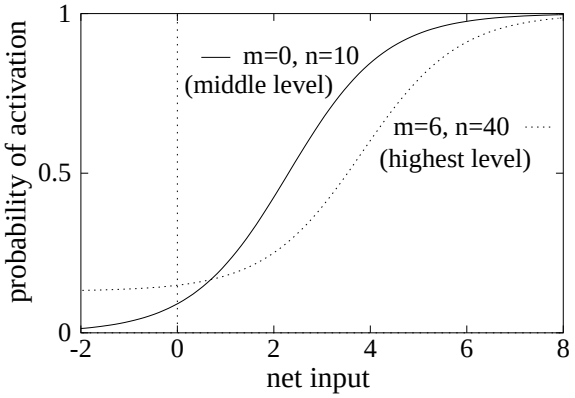


Figure 41: The binary stochastic transfer functions $\vec{f}_b^{w,s}$ with two different sets of parameters n and m which are used in the two subgroups of the hidden neurons. Given an input (x-axis) they represent the probability (y-axis) that a neuron is "ON". This is also the mean output value used in $\vec{f}_m^{w,s}$. The dotted line shows the transfer function of neurons which are designed for the hierarchically higher level. There is a prominent resting activity ($m = 6$) but a weak gain ($n = 40$, i.e. high sparsity of firing). The function displayed as a solid line is intended for middle layer neurons. There is no resting activity ($m = 0$) but a stronger gain ($n = 10$). In the wake phase, these latter values were used for the function $\vec{f}_m^w$ for all neurons.

Distinguishing the modules The idea to distinguish the hidden neurons to form two modules takes advantage of data generation in the sleep phase: higher-level neurons do not receive any input, but lower-level neurons receive input from the spontaneous activity of the higher-level neurons. Thus, a neuron which tends to be spontaneously active is well suited for the hierarchically highest level. In contrast, a neuron which becomes strongly active from input only is well suited to represent the lower level. Two stochastic transfer functions which display these differential behaviors are depicted in Fig. 41 (Eq. 74). Underlying are three states a hidden neuron can choose from stochastically: an "ON"-state evoked by its input h_i, an "ON"-state evoked spontaneously with degeneracy m and an "OFF"-state with degeneracy n.

It was shown that degenerate “OFF”-states introduce sparse coding which is used to let feature detectors emerge in a stochastic net [63]. On the other hand, spontaneous (resting) activity was shown to be necessary in attractor nets [37] as well as *in vitro* [38] to maintain robust low frequency firing. Our model functions accommodate both features, “OFF”-states with degeneracy n and spontaneous “ON”-states with degeneracy m. The probability that a hidden neuron i is switched “ON” by its input h_i is (cf. Eq. 64 on page 51):

$$P(u_i = 1) \;=\; \frac{e^{h_i} + m}{e^{h_i} + m + n} \;=:\; f_m(h_i) \tag{74}$$

The corresponding stochastic transfer function which assigns the two binary values is denoted by f_b. The corresponding mean-field transfer function is f_m.

Wake-sleep algorithm The detailed processing steps of the wake-sleep algorithm on the flexible architecture are as follows. Wake phase, inferring the hidden code from a data point $\vec{x}$:

$$\begin{aligned} \vec{u}_1^{\,w} &= \vec{f}_m^{\,w}(W_{10}\vec{x}) \\ \vec{u}_2^{\,w} &= \vec{f}_m^{\,w}(W_{11}\vec{u}_1^{\,w}) \end{aligned} \tag{75}$$

Reconstruction of the input:

$$\begin{aligned} \vec{s}_1^{\,w} &= V_{11}\vec{u}_2^{\,w} \\ \vec{s}_0^{\,w} &= V_{01}\vec{u}_1^{\,w} \end{aligned} \tag{76}$$

Update of the generative weights:

$$\begin{aligned} \Delta V_{11} &= \varepsilon_{11}\,(\vec{u}_1^{\,w} - \vec{s}_1^{\,w}) \cdot (\vec{u}_2^{\,w})^T \\ \Delta V_{01} &= \varepsilon_{01}\,(\vec{x} - \vec{s}_0^{\,w}) \cdot (\vec{u}_1^{\,w})^T \end{aligned} \tag{77}$$

Sleep phase, initiation of the hidden code at the hierarchically highest level:

$$\vec{s}_2^{\,s} \;=\; \vec{f}_b^{\,s}(0) \tag{78}$$

Generation of an input code:

$$\begin{aligned} \vec{s}_1^{\,s} &= \vec{f}_b^{\,s}(V_{11}\vec{s}_2^{\,s}) \\ \vec{s}_0^{\,s} &= V_{01}\vec{s}_1^{\,s} \end{aligned} \tag{79}$$

Reconstruction of the hidden code:

$$\begin{aligned} \vec{u}_1^{\,s} &= \vec{f}_m^{\,s}(W_{10}\vec{s}_0^{\,s}) \\ \vec{u}_2^{\,s} &= \vec{f}_m^{\,s}(W_{11}\vec{s}_1^{\,s}) \end{aligned} \tag{80}$$

Update of the recognition weights:

$$\begin{aligned} \Delta W_{10} &= \varepsilon_{10} \left(\vec{s}_1^s - \vec{u}_1^s\right) \cdot \left(\vec{s}_0^s\right)^T \\ \Delta W_{11} &= \varepsilon_{11} \left(\vec{s}_2^s - \vec{u}_2^s\right) \cdot \left(\vec{s}_1^s\right)^T \end{aligned} \tag{81}$$

Note the choice of mean activation functions $\vec{f}_m$ and stochastic binary functions $\vec{f}_b$. Where possible, $\vec{f}_m$ was chosen, for efficacy reasons. In the sleep phase, initiation of the hidden code must be done stochastically, using $\vec{f}_b$.

The wake-sleep algorithm is not a gradient descent in an energy space and easily gets stuck in local minima. The following weight constraints were applied to improve the solutions found: *(i)* generative weights V_{01} and V_{11} as well as lateral recognition weights W_{11} were rectified, i.e. negative weights were set to zero. *(ii)* a soft weight constraint was applied to the recognition weights in order to preserve sparse coding and to enforce competition between coding on the two hierarchical levels. It adds to Eq. 81 and treats positive and negative weights separately. With the use of the Heaviside function $\Theta(x) = 1$, if $x > 0$, otherwise 0, it can be written as:

$$\Delta w_{ij}^{decay} = \lambda^w \, h_i \, \Theta(w_{ij}) \, w_{ij} \sum_{j'} \Theta(w_{ij'}) w_{ij'}^2, \tag{82}$$

where $\vec{h} = \vec{u}_1^w + \vec{u}_2^w + \vec{u}_1^s + \vec{u}_2^s$ is the sum of all activations which have been induced by the recognition weights. The indices j, j' extend over all input and hidden units.

The algorithm often gets stuck in suboptimal solutions. During learning, single weights temporarily get very strong. We could obtain good results faster if maximal weight values were clipped at the values which were later maximal.

Note that the second-level weights learn to predict the contribution of the activation of the first level units which originates from the input units. They do not learn the averaged activation which take into account the back projection nor the value which is sparsified by the transfer function.

Training Weights were initialized with small positive random values. On-line learning was performed with $5 \cdot 10^6$ randomly generated data points. For each data point, Eqs. (75) to (82) were computed, where the sleep phase was repeated 8 times in order to obtain an average of the stochastic functions, Eqs. (78,79).

One third of the hidden neurons (upper third in Figs. 42,43,44) were designed to code on the highest hierarchical level (see the caption of Fig. 41 for description), the other two thirds for the lower level. Other parameters were: learning step sizes $\varepsilon_{01} = \varepsilon_{10} = 0.01$, $\varepsilon_{11} = 0.001$, both were decreased linearly to zero in the second half of training. Weight decay parameter $\lambda^w = 0.01$.

Areas organize in parallel After training on parallelly organized data we find that neurons in the upper third (Fig. 42) predominantly represent the more frequent stimuli (lines of $90°$ and $135°$) while neurons in the lower two thirds represent the less

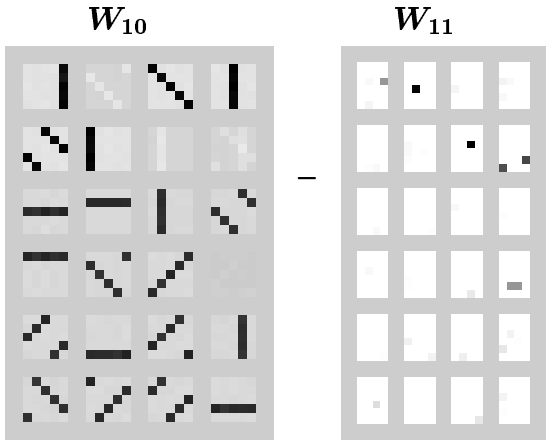


Figure 42: The recognition weight matrix W_{10} (**left**) and the lateral recognition weight matrix W_{11} (**right**) after training on parallelly organized data. Each square of the weight matrices shows the receptive field of one of the 5×6 hidden neurons; **left**, weights from the 5×5 input neurons and **right**, lateral weights. Negative weights are brighter than the background (frame), positive weights are darker (**left**). For lateral weights (**right**), zero weights are white (there are no negative weights).
Areas have organized parallelly: weights W_{10} to the inputs code for $90°$ and $135°$ lines in the upper third and predominantly for $0°$ and $45°$ lines in the lower two thirds.

frequent stimuli (lines of $0°$ and $45°$). Fig. 42 shows an exemplary result of the trained recognition weights W_{10}, and also lateral recognition weights W_{11} which have been included for consistency with the next experiment and which have not learned a meaningful structure here.

This result is surprising, because we would expect the neurons in the upper third to code for the less frequent stimuli, because they are less susceptible to input. Instead, it seems to be substantial that they are more active spontaneously and when input is low (Fig. 41, $m = 6$, $n = 40$). This is the case in the initial phase of learning when the weights have not organized. The neurons which are more active will now learn faster the most frequent stimuli.

We repeated this simulation ten times with different initial random values which changes the initial weights as well as the randomly generated stimuli. Table 7 shows the scores. Because there are more frequent lines than there are neurons in the upper third, some neurons in the lower two thirds, too, code for frequent lines. Altogether, there are four more neurons than stimulus types and so some receptive fields remain empty.

Areas organize hierarchically The net shown in Fig. 43 was trained on the hierarchical data set and has self-organized accordingly. Neurons in the upper third are more active spontaneously and four of them have vanishing connections W_{10} to the input neurons. Instead, they code on the higher level by integrating via lateral weights

	$0°, 45°$	$90°, 135°$	none
upper third	10%	70%	20%
lower two thirds	56%	28%	16%

Table 7: The neurons from the two regions and the stimulus classes which they choose to code for in the parallel setting. The percentage states how often a neuron selects a class. Numbers are averaged over ten runs and over all neurons within each region.

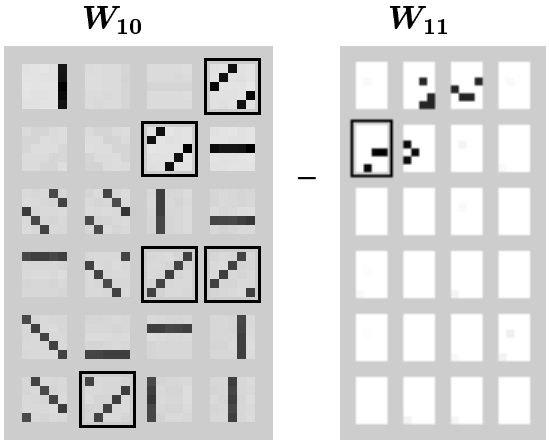


Figure 43: Same as Fig. 42, but trained on hierarchically organized data (same initialization of weights). Areas have organized hierarchically: neurons in the lower two thirds code for the input via W_{10} while four neurons in the upper third each integrate via W_{11} units from the lower two thirds which code stimuli of one direction. Neurons which code for lines of $45°$ orientation are marked by a frame.

W_{11} neurons from the lower level which code for the same orientation.

These higher-level neurons were never observed to have weights to other neurons in the upper third even if these had matching orientation. This would not be a flaw if only four neurons had been given the proper parameters for the higher level. In some experiments, however, only three neurons in the upper third managed to code on the higher level while the others were connected to the input neurons. Then one neuron within the lower two thirds was superfluous and did not code on the higher level.

Areas organize from intrinsic noise Without data ($\vec{x}$ set to zero), only intrinsic stochastic neuronal dynamics remains for training. As the generative weights V_{01} towards the input become zero, also the recognition weights W_{10} will decay in order to "invert" the generative model. Prevalently low activation values favor neurons in the upper third (Fig. 44) to develop lateral recognition weights W_{11} because these have stronger resting activity. The resulting structure, however, is not reflected in the

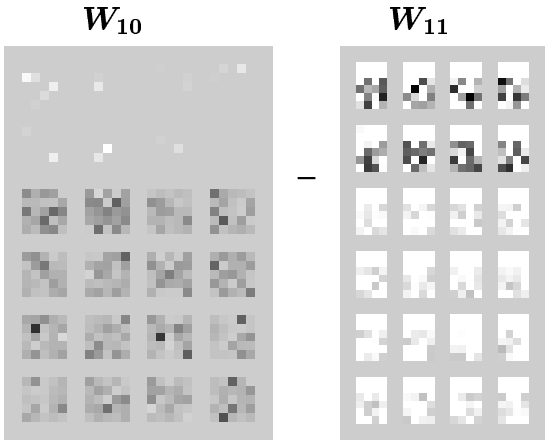


Figure 44: Same as Fig. 42, but trained with zero-valued data. Neurons in the upper third receive input via W_{11} from the units in the lower two thirds. Weights W_{10} to the input neurons are much smaller but still visible in the lower two thirds because of normalization of the brightness scale.

generative weights V_{11} (not shown), because in the wake phase, when these are trained, parameters of both regions do not differ.

5.5.4 Effect of initialization

In separate experiments with the Kalman filter model we have perceived a strong influence of developmental constraints when the sparsity parameter λ_u was unchanged across the hidden neurons (results not shown). In the parallel setting, a hidden area where neurons are initialized with larger weights and where weights grow faster will look at those input features which occur more often. Likewise, in the hierarchical setting, a desired relationship between areas can be induced by a strong initialization and faster growth of favored weights.

Similar can be observed with the Helmholtz machine, with constant parameters n and m across the hidden neurons: in the parallel setting, a hidden area where weights grow faster will look at those input features which occur more often (results not shown). This effect parallels the effect of stronger activity at low input which amplifies only the Hebbian learning term. Likewise, in the hierarchical setting, a desired relationship between areas can be induced by a strong initialization and faster growth of favored weights. This renders our model a model for the influence of topological neighborhood.

6 Discussion

Boltzmann machine

With the generation of localized edge detectors on a large array of cells we have shown that the Boltzmann machine is suited for biological modeling.

Approximations used for the Boltzmann machine

Using the architecture of a restricted Boltzmann machine, the clamped phase of the learning rule can be computed exactly by activating each neuron only once. However, we had to use several approximations along the way to calculate the correlations $\langle u_i x_j \rangle^-$ during free run.

- The discrete states which obey a Boltzmann distribution were rendered by the mean-field approximation into discrete states which follow a trajectory towards a stationary mean value which is used for learning.
- The temperature remains constant during relaxation. The global maximum for the mean-field relaxation can be guaranteed to be found, if the technique of simulated annealing is used. This involves scaling a temperature parameter slowly towards a final value during relaxation. We did not introduce this parameter which has the effect of having a constant non-zero temperature value. In effect we do not generally find the global maximum of E^{relax} but get stuck in one of its numerous local maxima.
- Fix-point iteration (the new activations are directly inserted) rather than continuous update of activities with small learning steps is performed. Step sizes are thus very large but the direction of the gradient is followed.
- Early stopping of the relaxation procedure in general does not even allow the local optimum to be reached.

For our modeling purpose, the approximations which we used were not crucial to learning. The task is not to calculate $\langle u_i x_j \rangle^-$ once perfectly. Instead, many repetitions of the relaxation procedure and the weight update are performed. Remembering that $\Delta w_{ij} \approx -\langle u_i x_j \rangle^-$, not reaching the global maximum during one free relaxation means that the corresponding point in state space remains an imprinted memory item. It will be reached more easily in a following relaxation procedure. Thus, if the optimal representation is missed and a weight update performed, then this representation is more likely to be hit for the next update.

The two phases of the Boltzmann machine

Gradient descent on the maximum likelihood energy function leads to the Boltzmann machine learning rule which consists of two terms. Each term in isolation does not have a mathematical justification.

It is tempting to identify the clamped phase of the Boltzmann learning rule to awakeness and the free-running phase to sleep and dreaming. We have investigated both phases under the conditions of our simulations: *(i)* symmetric feed-forward-/feedback weights but no lateral connectivity and *(ii)* mean-field approximation on single data points. Awake learning only makes neurons react monotonously to the input – there is no crosstalk and thus no decorrelation between them. Sleep induces variety: on the one hand the number of attractors increases, on the other hand weights change continuously.

Whether there can be two kinds of learning for given connections in the cortex will be difficult to determine. A list of some dozen of sleep associated chemical substances [35] encourages the idea. The control of the two phases may be served by the thalamus. Thalamic LGN cells exhibit two distinct responde modes [53]: a *relay* mode, which we associate to the wake phase, in which cells replicate retinal input more or less faithfully, and a *burst* mode, which we associate to the sleep phase, in which cells burst in a rhythmic pattern that bears little resemblance to the retinal input.

Comparison of the Boltzmann machine and the Kalman filter model

A most powerful form of the maximum-likelihood principle is that our model generates the whole dataset freely, that is each data point with its corresponding probability. Neurons have to become active when no input arrives, which can be determined in a computationally expensive manner from the Boltzmann distribution.

The Kalman filter model is computationally modest because the distribution over hidden states, Eqn. 29, is approximated by only one optimal state, Eqn. 30. In order to find this state, a data point must be given. This data generating model thus has the goal to reconstruct a data point while it is its actual input. Neurons are inactive without input.

For the reconstruction of a known data point $\vec{x}$, the optimal hidden neuron activations $\vec{u}$ can be found by minimizing the squared reconstruction error:

$$E \;=\; \frac{1}{2}(\vec{x} - W^T\vec{u})^T\,(\vec{x} - W^T\vec{u}) \tag{83}$$

Gradient descent with respect to $\vec{u}$ leads to an update dynamics for the activations in time

$$\dot{\vec{u}} \;=\; \eta\, W\,(\vec{x} - W^T\vec{u})$$

with learn step η. This equation characterizes the model as follows: *(i)* input neurons carry the reconstruction error $\vec{x} - W^T\vec{u}$. If the data point is perfectly recognized then input neurons become inactive. *(ii)* Feed-forward weights W transfer the remaining

activations to the hidden units and *(iii)* feed-forward and feedback weights are "symmetric" but with opposite sign. It is a goal of the model not to generate but rather to extinguish a given data point.

For simple comparison we compare the learning rules which are used by the data reconstructing model and the Boltzmann rule in the convergent state. The energy function of the first model, Eqn. 83 on the page before, has a minimum if

$$\sum_{\mu} \vec{x}^{\mu} \;=\; \sum_{\mu} W^T \vec{u}^{opt}$$

$\vec{u}^{opt}$ is the hidden activation vector which is optimal to reconstruct the data point $\vec{x}^{\mu}$. Multiplying both sides by this value yields

$$\sum_{\mu} (\vec{u}^{opt})^T \vec{x}^{\mu} \;=\; \sum_{\mu} (\vec{u}^{opt})^T W^T \vec{u}^{opt}$$

In our case, learning is complete if Eqn. 61 on page 49 yields

$$\sum_{\mu} \langle u_i \, x_j \rangle^{\mu +} \;=\; \sum_{\nu} \langle u_i \, x_j \rangle^{\nu -}$$

While in the upper equation an activation vector $\vec{u}^{opt}$ is used which is optimal for reconstruction the latter equation uses a mean activation.

Two circumstances make it a harder goal to satisfy the latter equation. First, any activation value which is used for learning has to settle by a free relaxation procedure without the use of input. As we have seen, however, the fix-points are not "sparse", instead all neurons tend to take either strong or zero activations. Secondly, the set of free relaxation entities should reflect the data set. This is achieved by the relaxation procedure to converge to different local minima. The first problem is herby seemingly solved if fix-points alterate between strong values only and zero values only. This satisfies Eqn. 61 on page 49 because strong values cancel out zero values and thus leads to convergence of mean-field learning.

Biological considerations

There is a fundamental difference in the interpretation of the network architecture between the recurrent models which we have used. The Boltzmann machine generates data through positive feedback projections, i.e. feedback weights have the same sign as feed-forward weights. The Kalman filter model extinguishes a given data point by subtraction of its prediction. Feedback weights have opposite sign compared to feed-forward weights.

Biological evidence for symmetry or anti-symmetry of weights is difficult to obtain. Long range cortico-cortical and cortico-thalamic projections are made up of pyramidal cell axons which are thought to be excitatory. In general, however, they do not contact the input neurons (layer 4 of cortex) directly but via cells in superficial layers and layer

6 as well as cells with local arborization in layer 4. Inhibitory cells are among these populations.

Physiological observations support the Boltzmann machine. First cortical and thalamic neurons are active even during sleep, that is when no input data arrive. Another observation is that an expected stimulus is more likely to be seen than an unexpected stimulus. A feedback prediction thus should rather generate than extinguish the expected pattern.

An interpretation to the Kalman filter model is that an unexpected stimulus evokes further interest because non-predicted activation remains on the input. This is an attractive theory especially for the superior colliculus which is attributed to defining the focus of attention [5]. However, it may govern the function of the cortex during awakeness.

Topographic maps

We have shown that within an information theoretic theory it is possible to make biological modeling. More precise, Bayesian priors can introduce the mechanisms which are needed to obtain functionally specific neurons and topographic mappings. The results are promising to parallel orientation patterns observed in cats and primates if net sizes are chosen larger.

Modularization

We have shown that a net trained according to a maximum likelihood framework can organize in *a)* two parallelly or *b)* two hierarchically organized areas dependent on statistics of the data. Parameters are unchanged between these experiments. The hidden area with more frequently active neurons *a)* looks at input features which occur more often or *b)* takes the function of the hierarchically higher area in our setting.

We take this as a model for the development of connection patterns and relations between cortical areas like the parallel segregation of the visual stream into a lateral and a dorsal pathway or the hierarchical relation between V1 and V2. Different functional properties, e.g. in a spike coding model the refractory period or burst duration, could be varied across areas, to account for a broader parallel organization and for more hierarchical stages.

In contrary to models which are based on neighborhood only [52], however, we model activity based and data driven growth and refinement of connections. The results on parallel and hierarchical organization are similar to those obtained by our previous work [66]. They show that simple parameter changes within a neural network can conduct the data flow to structure the network via Hebbian learning to form an adequate representation of the data.

The Helmholtz machine is advanced w.r.t. the Kalman filter model because of its biologically plausible activation dynamics. Its stochastic neurons are activated reminiscent of a synfire chain, whereas in the Kalman filter model, each hidden neuron

has to keep track of two activation values at a time. Also, internal structuring without any data can to a certain degree be done using the Helmholtz machine.

For simplicity we have neglected many additional learning mechanisms of the biological system. Most important are topographic constraints which favor neighboring cortical areas to be connected [52]. These determine however the gross connectivity pattern between cortical areas only. Within an area a topographic constraint can force a minority of neurons to code conform with the majority. Such a constraint could eliminate the outliers in our results. Time behavior like a possible flow of a stimulus induced signal from the back of the cortex towards the front could alternatively or as an additional mechanism give rise to hierarchical relationships.

By our approach which omits as much as possible internal mechanisms we could display a large potential which the data has in the organization of structure. On a high level of abstraction it demonstrates how much influence the structure of the data may have on the emerging structure of the brain.

Connectivity

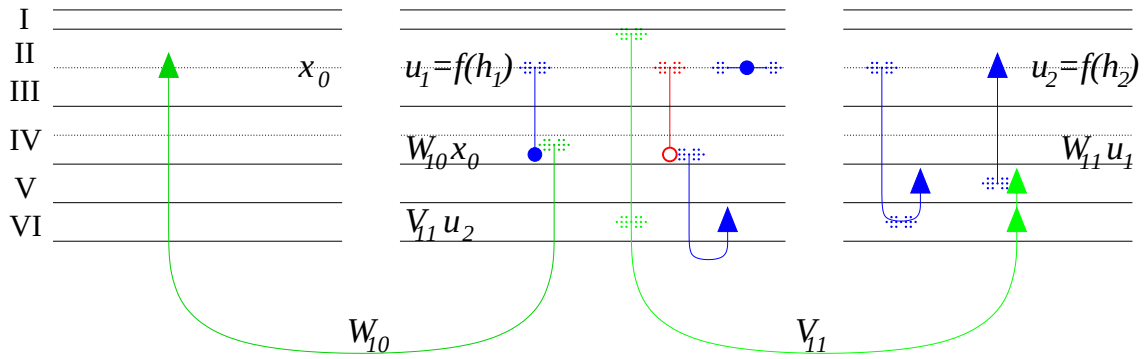


Figure 45: Proposed computations of neurons. The notation is the same as in the model description of the hierarchical Kalman filter model and the Helmholtz machine. The connections within the six cortical layers as well as hierarchical relationships are as in Fig. 8 (section *The Cortex*). The area depicted to the left corresponds to the input layer and does not have sub-layers in the models.

Fig. 45 demonstrates how the architecture and the computations of a model can be identified with the neural circuitry of the cortex (cf. [36]). Model weights are identified with inter area connections. The figure shows only connection which terminate in the middle area, bottom-up connection $\boldsymbol{W_{10}}$ and top-down connections $\boldsymbol{V_{11}}$ (which are not termed $\boldsymbol{V_{12}}$ only because in our model a hierarchy evolved instead of being fixed). Computations which are local in the model are identified to computations within a cortical “column”. This means that they are localized along the surface of the cortex but extend through the six layers.

We identify layer 4 as the locus of bottom-up information receival to capture the values $\vec{h}_1 = \boldsymbol{W}_{10}\vec{x}_0$. Layer 6 (as well as upper layer 2) is the locus of top-down information receival to receive $\boldsymbol{V}_{11}\vec{u}_2$. Layers 2/3 integrate both of these inputs (in a model-dependent way) and are the source of the (sparsified) outputs $\vec{u}_1$. These values go directly to a higher area and via layer 5 (not considered in the model) to a lower area.

This identification also displays restrictions of the model: only the main streams are considered, lateral connections are omitted. Furthermore, according to the model, no learning takes place within a cortical “column”.

Both models, the Kalman filter model and the Helmholtz machine, can be identified with this connection scheme and the notations of the model equations can be applied to Fig. 45. Both models use basic computations like the scalar product of the inter-area weights with the activations from the source area of an input. These computations follow directly from the anatomy.

The differences of the models lie in the way how bottom-up and top-down input is integrated within one area. The input to the neurons in layers 2/3, $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$, is different. In the Kalman filter model, it is a difference taken from both input streams and furthermore, the current activations, $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are considered to compute activations at the next time step. In the Helmholtz machine, there is no integration of the bottom-up and the top-down stream for the computation of the activations but only for the learning rules, the difference is taken.

Finer differences between the two models, of course, cannot show up in the connection scheme: the time when computations are made, the initialization of activities and the learning rules.

Acknowledgments

This work was done within the neural information processing group at the Technische Universität Berlin which was founded in 1995 and which developed prosperously under the direction of Prof. Klaus Obermayer who spent an overwhelming effort into this task. Traces of this effort were surely delegated to all group members and teaching and organization of tutorials was hard but rewarding work through which I acquired founded knowledge on neural nets. A high-level scientific environment was supplied which allowed international visitors to be heard on-site as well as visits to international collaborators as well as conferences. Prof. Obermayer let me freely choose my own subjects of interest even though the promising “Goldfish project” suffered from my ignorance.

Prof. Jennifer Lund gave talks in Berlin and in London and answered patiently to any questions. Her talks were at the same time introductory and in depth to cortical biology and contributed to advancing the biological section of this dissertation. Hauke Bartsch was open to frequent and extensive discussions on any aspects. In particular, his instructions to better programming eased interactive programming and testing significantly. Discussions with Thore Graepel served for a better understanding and elaboration of the theory part of this dissertation. In our very busy group it was difficult to find colleagues to read and correct my publications. Gregor Wenning gave away some of his time to correct an earlier version of this long manuscript. It should also be noted that the computer network within the computer science department was maintained very well by the service team so that it was permanently possible to run simulations on several UNIX SUN-workstations in parallel.

References

[1] S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 757–763. MIT Press, 1996.

[2] D. Bagnard, N. Thomasset, M. Lohrum, A.W. Püschel, and J. Bolz. Spatial distributions of guidance molecules regulate chemorepulsion and chemoattraction of growth cones. *J. Neurosci.*, 20(3):1030–1035, 2000.

[3] H. Baier and F. Bonhoeffer. Axon guidance by gradients of a target-derived component. *Science*, **255**:472–475, 1992.

[4] H. Barlow. *Large Scale Neuronal Theories of the Brain.*, chapter What is the computational Goal of the Neocortex?, pages 1–22. MIT Press, 1994.

[5] M.A. Basso and R.H. Wurtz. Modulation of neuronal activity in superior colliculus by changes in target probability. *J. Neurosci.*, 18(18):7519–7534, 1998.

[6] Ute Bauer. *Computational Models of Neural Circuitry in the Macaque Monkey Primary Visual Cortex*. PhD thesis, Universität Bielefeld, 1999.

[7] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neur. Comp.*, 7(6):1129–1159, 1995.

[8] A.J. Bell and T.J. Sejnowski. Edges are the 'independent components' of natural scenes. In *Advances in Neural Information Processing Systems 9*, 1996.

[9] B. Bentzien, C. Weber, and K. Obermayer. Analysis of a model for the development and regeneration of the retinotectal projection. In *Forum of European Neuroscience, Proceedings*, 1998.

[10] G.G. Blasdel and J.S. Lund. Termination of afferent axons in macaque striate cortex. *J. Neurosci.*, 3(7):1389–1413, 1983.

[11] E.M. Callaway. Local circuits in primary visual cortex of the macaque monkey. *Annu. Rev. Neurosci.*, 21:47–74, 1998.

[12] J.F. Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Letters on Signal Processing*, 4(4):112–114, 1997.

[13] G.J. Carman, H.A. Drury, and D.C. Van Essen. Computational methods for reconstructing and unfolding the cerebral cortex. *Cerebral Cortex*, 5:506–517, 1995.

[14] V.S. Caviness, T. Takahashi, and R.S. Nowakowski. Numbers, time and neocortical neuronogenesis: a general developmental and evolutionary model. *Trends in Neurosci.*, 18(9):379–383, 1995.

[15] H.J. Cheng, M. Nakamoto, A.D. Bergemann, and J.G. Flanagan. Complementary gradients in expression and binding of ELF-1 and Mek4 in development of the toppographic retinotectal projection map. *Cell*, **82**:371–381, 1995.

[16] W.M. Cowan. Die entwicklung des gehirns. *Spektrum der Wissenschaft: Gehirn und Nervensystem*, 8:100–110, 1987.

[17] P. Dayan, G. E. Hinton, R. Neal, and R. S. Zemel. The Helmholtz machine. *Neur. Comp.*, 7:1022–1037, 1995.

[18] M. J. Donoghue and P. Rakic. Molecular evidence for the early specification of presumptive functional domains in the embryonic primate cerebral cortex. *J. Neurosci.*, 19(14):5967–79, 1999.

[19] U. Drescher, C. Kremoser, C. Handwerker, J. Löschinger, M. Noda, and F. Bonhoeffer. In vitro guidance of retinal ganglion cell axons by RAGS, a 25 kDa tectal protein related to ligands for Eph receptor tyrosine kinases. *Cell*, **82**:359–370, 1995.

[20] D.J. Felleman and D.C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

[21] P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biol. Cybern.*, 64:165–170, 1990.

[22] C. Fyfe and R.J. Baddeley. Finding compact and sparse-distributed representations of visual images. *Network*, 6:333–344, 1995.

[23] S. Geyer, M. Matelli, G. Luppino, A. Schleicher, Y. Jansen, N. Palomero-Gallagher, and K. Zilles. Receptor autoradiographic mapping of the mesial motor and premotor cortex of the macaque monkey. *J. Comp. Neurol.*, 397:231–250, 1998.

[24] A. Gierer. Directional cues for growing axons forming the retinotectal projection. *Development*, **101**:479–489, 1987.

[25] H. Haken. *Synergetics. An Introduction.* Springer-Verlag, 1982.

[26] G. Harpur. Development of low entropy coding in a recurrent network. *Network – Computation in Neural Systems*, 7(2):277–284, 1995.

[27] S. Haykin. *Neural Networks. A Comprehensive Foundation.* Macmillan College Publishing Company, 1994.

[28] J. Hegdé and D.C. Van Essen. Selectivity for complex shapes in primate visual area v2. *J. Neurosci.*, 20(5):1–6, 2000.

[29] G.H. Henry. Neural processing in cat striate cortex. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(5):888–900, 1983.

[30] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation*. Addison Wesley, 1991.

[31] G. Hinton. Products of experts. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *Proceedings ICANN*, pages 1–6, 1999.

[32] G. E. Hinton, P. Dayan, B. J. Frey, and R. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.

[33] E. Hunziker and G. Mazzola. *Ansichten eines Hirns; Aktuelle Perspektiven der Hirnforschung*. Birkhäuser, 1990.

[34] G.M. Innocenti. Exuberant development of connections, and its possible permissive role in cortical evolution. *Trends in Neurosci.*, 18(9):397–402, 1995.

[35] E.R. Kandel, J.H. Schwartz, and T.M. Jessell. *Principles of neural science*. Prentice-Hall, 1991.

[36] M. Kawato, H. Hayakawa, and T. Inui. A forward-inverse optics model of reciprocal connections between visual cortical areas. *Network*, 4:415–422, 1993.

[37] P.E. Latham, B.J. Richmond, P.G. Nelson, and S. Nirenberg. Intrinsic dynamics in neuronal networks. I. Theory. *J.Neurophysiol.*, 83:808–827, 2000.

[38] P.E. Latham, B.J. Richmond, S. Nirenberg, and P.G. Nelson. Intrinsic dynamics in neuronal networks. II. Experiment. *J.Neurophysiol.*, 83:828–835, 2000.

[39] J.B. Levitt, J.S. Lund, and T. Yoshioka. Anatomical substrates for early stages in cortical processing of visual information in the macaque monkey. *Behavioral Brain Reseearch*, 76:5–19, 1996.

[40] W.B. Levy and R.A. Baxter. Energy efficient neural codes. *Neur. Comp.*, 8(3):531–543, 1996.

[41] Z. Molnar and C. Blakemore. How do thalamic axons find their way to the cortex? *Trends in Neurosci.*, 18(9):389–396, 1995.

[42] J.D. Murray. *Mathematical Biology*. Springer-Verlag, 1987.

[43] W.J.H. Nauta and M. Feirtag. Die architektur des gehirns. *Spektrum der Wissenschaft: Gehirn und Nervensystem*, 8:88–98, 1987.

[44] B.A. Olshausen. Learning linear, sparse, factorial codes. A.I. Memo 1580, Massachusetts Institute of Technology., 1996.

[45] B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[46] B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

[47] B.A. Pearlmutter and L.C. Parra. A context-sensitive generalization of ICA. In *Proc. ICONIP*, 1996.

[48] P. Rakic. A small step for the cell, a giant leap for mankind: a hypothesis of neocortical expansion during evolution. *Trends in Neurosci.*, 18(9):383–388, 1995.

[49] R.P.N. Rao and D.H. Ballard. Dynamic model of visual recognition predicts neural response properties of the visual cortex. *Neur. Comp.*, 9(4):721–763, 1997.

[50] K.S. Rockland and J.S. Lund. Intrinsic laminar lattice connections in primate visual cortex. *J. Comp. Neurol.*, 216:303–318, 1983.

[51] S.M. Rüger. *Aspekte Neuronalen Lernens*, volume M-08 of *Reihe Mathematik*, chapter Effizientes Lernen und Schließen in dezimierbaren Boltzmann-Maschinen. Fakultät für Mathematik, Naturwissenschaften und Informatik, Brandenburgische Technische Universität Cottbus, 1996.

[52] J.W. Scannell, C. Blakemore, and M.P. Young. Analysis of connectivity in the cat cerebral cortex. *J. Neurosci.*, 15(2):1463–1483, 1995.

[53] S.M. Sherman and C. Koch. The control of retinogeniculate transmission in the mammalian lateral geniculate nucleus. *Exp. Brain Res.*, 63:1–20, 1986.

[54] B. Stahl, Y. von Boxberg, B. Müller, J. Walter, U. Schwartz, and F. Bonhoeffer. Directional cues for retinal axons. *Cold Spring Harbor Symp. Quant. Biol.*, **55**:351–357, 1990.

[55] E. Tanaka and J. Sabry. Making the connection: Cytoskeletal rearrangements during growth cone guidance. *Cell*, **83**:171–176, 1995.

[56] D.C. Van Essen. A tension-based theory of morphogenesis and compact wiring in the central nervous system. *Nature*, 385(23):313–318, 1997.

[57] C. Weber, H. Ritter, K. Obermayer, and J. Cowan. A model for intrinsic and activity dependent mechanisms underlying the regeneration of the retinotectal map in goldfish. In *ICANN'95 Proceedings, Vol. 2*, pages 491–496. International Conference on Artificial Neural Networks, 1995.

[58] C. Weber, H. Ritter, K. Obermayer, and J. Cowan. Topographic and disturbed development of the retino-tectal projection of goldfish: a computational model. In *1. Kongress der Neurowissenschaftlichen Gesellschaft*, 1996.

[59] C. Weber, H. Ritter, J. Cowan, and K. Obermayer. Development and regeneration of the retinotectal map in goldfish: A computational study. *Phil. Trans. Roy. Soc. Lond. B.*, 352:1603–1623, Nov 1997.

[60] C. Weber and K. Obermayer. Retinotopic orientation maps emerge by diffuse connectivity in a framework derived by maximum-likelihood. In *New Neuroethology on the Move (Proceedings of the 26th Göttingen Neurobiology Conference)*, page 490, 1998.

[61] C. Weber and K. Obermayer. Orientation selective cells emerge in a sparsely coding Boltzmann machine. In *Proceedings of the 27th Göttingen Neurobiology Conference*, page 499, 1999.

[62] C. Weber and K. Obermayer. Orientation selective cells emerge in a sparsely coding Boltzmann machine. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *Proceedings ICANN*, pages 286–291, 1999.

[63] C. Weber and K. Obermayer. Orientation selective cells emerge in a sparsely coding Boltzmann machine. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *Proceedings ICANN*, pages 286–291, 1999.

[64] C. Weber and K. Obermayer. Emergence of modularity within one sheet of intrinsically active stochastic neurons. In *Proceedings ICONIP*, pages 732–37, 2000.

[65] C. Weber and K. Obermayer. Structured models from structured data: emergence of modular information processing within one sheet of neurons. In *Proceedings IJCNN*, 2000.

[66] C. Weber and K. Obermayer. Structured models from structured data: emergence of modular information processing within one sheet of neurons. In *Proceedings IJCNN*, 2000.

[67] C. Weber and K. Obermayer. *Emergent Neural Computational Architectures*, chapter Emergence of modularity within one sheet of neurons: a model comparison, pages 53–76. Springer-Verlag Berlin Heidelberg, 2001.

[68] C. Weber, H. Ritter, J. Cowan, and K. Obermayer. Development and regeneration of the retinotectal map in goldfish: A computational study. *Phil. Trans. Roy. Soc. Lond. B.*, 352:1603–1623, Nov 1997.

[69] C. Weber, H. Ritter, and K. Obermayer. A model for the development of the retinotectal projection in goldfish. In *Tagungsband zur 23. Göttinger Neurobiologentagung*, page 90. Göttinger Neurobiologentagung, 1995.

[70] C. Weber, H. Ritter, K. Obermayer, and J. Cowan. A model for the regeneration of the retinotectal projection in goldfish. In *Snowbird Abstracts*, 1995.

[71] T. Yoshioka, J.B. Levitt, and J.S. Lund. Independence and merger of thalamocortical channels within macaque monkey primary visual cortex: Anatomy of interlaminar projections. *Vis. Neurosci.*, 11:467–489, 1994.

[72] M.P. Young. The organization of neural systems in the primate cerebral cortex. *Proc. R. Soc. Lond. B*, 252:13–18, 1993.

[73] M.P. Young, J.W. Scannell, M.A. O'Neill, C.C. Hilgetag, G. Burns, and C. Blakemore. Non-metric multidimensional scaling in the analysis of neuroanatomical connection data and the organization of the primate cortical visual system. *Phil. Trans. R. Soc. Lond. B*, 348:281–308, 1995.

[74] S. Zeki and S. Shipp. The functional logic of cortical connections. *Nature*, 335:311–317, 1988.

Maximum a Posteriori Models for Cortical Modeling:
Feature Detectors, Topography and Modularity

PhD Thesis by Cornelius Weber, Berlin 2000

7 Summary

The thesis shows in a top-down modeling approach that unsupervised learning rules of neural networks can account for the development of cortical neural connections.

Sections 1, “Introduction”, and 2, “The Cortex”, comprise biological foundations about the cortex: its areas and their mutual connectivity, cell layers and the mechanisms which govern the development of neural connections. These data supply the goal of modeling as well as the motivation for the methods which are used.

Sections 3, “Theory”, and 4, “Models”, describe the theory and how to derive models from it. In a maximum likelihood framework, the model cortex learns to represent the real world. With neuronal activation states that follow the Boltzmann distribution, the model can then generate the incoming stimuli by itself. Using Bayesian priors on the model parameters, the learning rules are modified to account for biologically observed developmental mechanisms so that biologically observed structures develop.

Section 5, “Results”, presents simulation details and results and in section 6, “Discussion”, the models are compared and related to the biological findings [67].

In this work it is demonstrated that using a sparsely coded Boltzmann machine, neurons emerge which have localized and orientation selective receptive fields like those observed in primary visual cortex [62]. Another highlight is the demonstration of a high adaptability of model structures to the environment. Either parallelly or hierarchically organized modules will arise as an appropriate adaptation to the organization of the training data set. This is shown using two different models, a non-linear Kalman filter model [65] and a Helmholtz machine [64].