Computational neuroscience

Bert Kappen,
Neurophysics, Radboud University Nijmegen

September 7, 2016

Contents

1 Introduction

How does the brain compute? Particularly in the last hundred years have we gathered an enormous amount of experimental findings that shed some light on this question. The picture that has emerged is that the neuron is the central computing element of the brain which performs a non-linear input to output mapping between its synaptic inputs and its spiky output. The neurons are connected by synaptic junctions, thus forming a neural network.

A central question is how such a neural network implements brain functions such as vision, audition and motor control. These questions are to a certain extend premature, because our knowledge of the functioning of the neuron and the synaptic process itself is only partial and much remains to be discovered. Nevertheless, it is interesting to see what emergent behavior arises in a network of very simple neurons.

In section 2 we begin with the study of the spiking behavior of biological neurons. As we will see, the neural firing is not a deterministic function of the input that it receives, but is in fact best described as a stochastic process. We discuss the Poisson process, where a spike train is viewed as a point process in time; we study the spiking behavior of a random walk model and how its interspike interval distribution can be used to model a large variety of real neuron data; and we study the integrate and fire neuron.

It will become clear that these model neurons, although they give at best an approximate description of the behavior of real neurons, are still too complex to use as basic cells in a neural network. In section 3 we set the complexity of real neurons aside and introduce the simplest neuron model possible, which is the stochastic binary neuron. The pioneering work in this direction was done by McCulloch and Pitts [1] in the '40s. Taking the thresholding property of neurons to the extreme, they proposed that neurons perform logical operations on their inputs, such as AND and OR. One can show that a network of such neurons, when properly wired, can perform any logical function and is equivalent to a Turing machine.

When considering neural networks, an important distinction is between feedforward networks and recurrent networks. In *feed-forward networks*, the neurons can be labeled such that each neuron only receives input from neurons with lower label. Thus, one can identify *input neurons*, which receive no input from other neurons and whose activity depends only on the sensory stimulus, and *output neurons* whose output does not affect other neurons. When in addition the neurons themselves are assumed to have no internal dynamics, the dynamics of

feed-forward networks is trivial in the sense that the output is a time-independent function of the input: $y(t) = F(x(t))$, where F is a concatenation of the individual neuron transfer functions and x and y are input and output activity, respectively. Examples of such networks are the perceptron [2] and the multi-layered perceptron [3, 4] and are the subject of the chapters by John Hertz, taken from his book [5].

In *recurrent networks*i, even when individual neurons have no internal dynamics, the network as a whole does, and the input-output mapping depends explicitly on time: $y(t) = F(x(t), t)$, Examples of such networks are attractor neural networks [6], topological maps [7], sequence generators [8] and Boltzmann Machines [9]. We will study important properties of such networks such as transients, equilibrium, ergodicity and periodicity in section 3.2.

An exact description of transient and stationary behavior for stochastic neural networks is not possible in general. In some special cases, however, one can compute the generic behavior of stochastic networks using mean field theory. One averages over many random instances of the network (quenched average) and describes the properties of the network with a small number of order parameters. The classical example is the attractor neural network, as proposed by Hopfield [6]. The mean field analysis was presented in a series of papers by Amit, Gutfreund and Sompolinsky [10, 11]. In section 4, we study the simplest form of mean field theory, without averaging over the quenched disorder and show how this allows us to approximately compute statistics such as means and correlations in the network. In section 4.5, we show how we can use these approximations to learn the connections in a stochastic neural network form data.

In section 5.1, we introduce the concept of Hebbian learning, as first postulated in the psychological literature to explain the phenomenon of classical conditioning. We also show some physiological evidence of Hebbian learning in the form of long-term potentiation (LTP) and long-term depression (LTD). In section 5.2 take up the question of the functional role of recurrent computation in the brain. We show how both short-term memory and long-term memory are likely to require recurrent loops and that Hebbian learning may be important to establish the correct connections.

In section 5.3 we study the implications of Hebbian learning in a recurrent network of binary neurons. These networks are also called Hopfield networks. Patterns, which are global activity states of the network, can be stored in these networks as attractors of the stochastic dynamics. In these networks, memories are content addressable, in the sense that when the network is initialized in a state that resembles one of the stored patterns, the network will converge to that

memory. Due to the symmetric connectivity of the Hebb rule, the asymptotic behavior of the network can be computed in closed form. As a result, the capacity of the network to store memories can be analyzed as a function of the number of patterns that are stored and the noise in the neuron dynamics. When the noise is too high, all attractors become unstable and the firing of the neurons becomes more or less uncorrelated (paramagnetic phase). When the number of patterns is too large, the network behaves as a *spin glass* whose minima are uncorrelated with the stored patterns.

2 Neural information processing is noisy

When a neuron is presented repeatedly with the same stimulus, the response of the neuron is not identical, as can be seen in fig. 1. An important source of the noise is the unreliable response of synapses, as can be seen in fig. 2. This unreliability can in turn be related to the stochastic nature of the channels in the membrane, as is illustrated in fig. 3. Another cause of the unreliable neural firing is that the input to the neuron is not only from the stimulus, but also from active neurons that are connected to it. Since the response of the neuron is noisy, it differs from trial to trial. By averaging the response over some time window one computes the instantaneous firing rate (fig. 1 lower two panels) of the neuron which is reproducible from trial to trial.

Instead of measuring the instantaneous firing rate of the neuron, we can also measure the time between subsequent spikes, called the inter-spike intervals (ISIs). In fig. 4, we see some examples. In some cases, subsequent ISIs are correlated, in some cases they are not.

2.1 Poisson Processes

The Poisson process is the simplest stochastic point process that you can think of. The probability of firing in any short time interval $[t, t + \delta t]$ is constant and equal to $\lambda\delta t$.

Define $p_x(t)$ =Prob{cell fires exactly x times in $[0, t]$}. Then,

$$p_0(t + \delta t) = p_0(t)(1 - \lambda\delta t)$$

Since p_0 is a smooth function of t, we can use the Taylor series expansion and write:

$$p_0(t + \delta t) = p_0(t) + \delta t \frac{\partial p_0(t)}{\partial t}$$

Combining these last two equations, we obtain $\frac{\partial p_0(t)}{\partial t} = -\lambda p_0(t)$. This can be easily solved to yield

$$p_0(t) = \exp(-\lambda t)$$

since $p_0(0) = 1$.

We can now repeat this reasoning to compute $p_x(t)$.

$$\begin{aligned} p_x(t + \delta t) &= p_x(t)(1 - \lambda\delta t) + p_{x-1}(t)\lambda\delta t \\ \frac{\partial p_x(t)}{\partial t} &= -\lambda p_x(t) + \lambda p_{x-1}(t) \end{aligned}$$

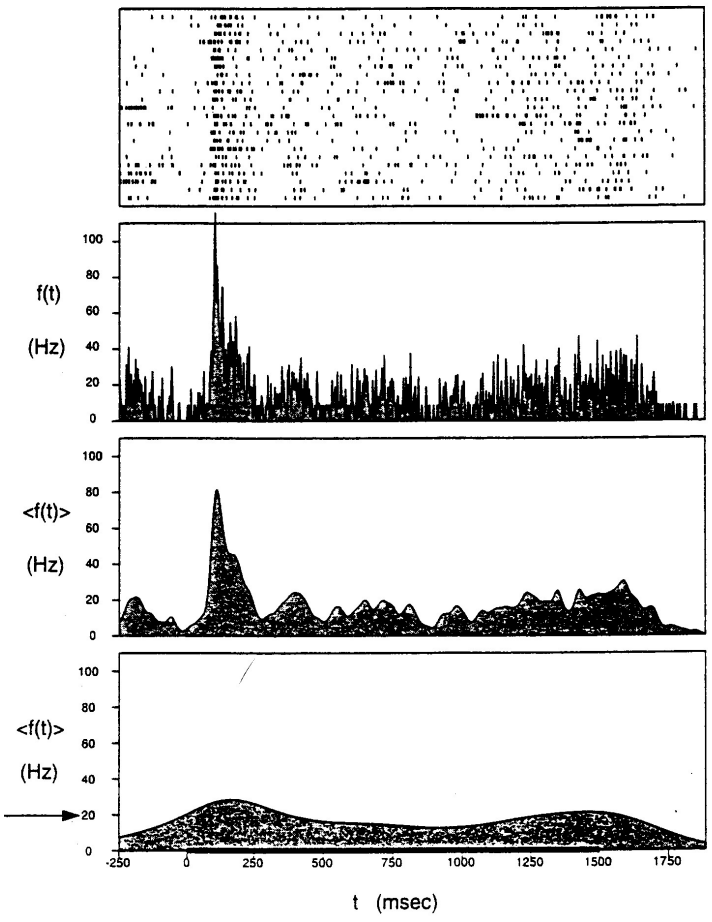


Figure 1: Monkey cortical area V4. response to 1500 ms flashed grating. Dots show spikes of the neuron. Different lines are different trials, which are each slightly different. Summing the aligned trials produces the post-stimulus time histogram (PSTH). The two lower plots illustrate an average firing rate obtained from the raster diagrams using Gaussian smoothing with σ set to 20 and 200 msec, respectively.

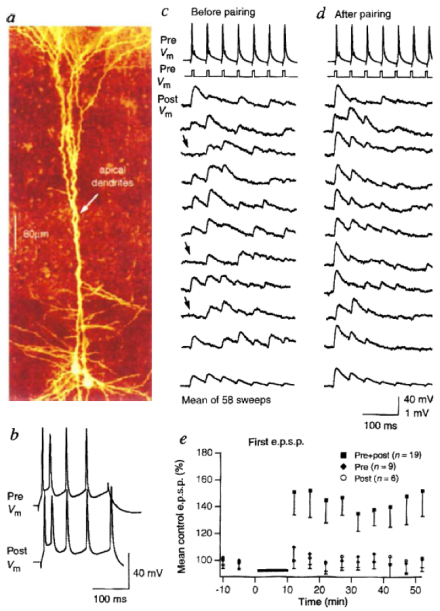


Figure 2: Layer 5 pyramidal neuron from somatosensory cortex of rat. c) shows in repeated trials the response of a single synapse to a regular train of presynaptic spikes. As can be seen, synaptic response is very unreliable. d) shows the same experiment after a period of vigorous paired stimulation of the pre- and post-synaptic cell (b). One sees that as a result of the paired stimulation the reliability of the *first* synaptic response is greatly enhanced. e) This effect lasts for hours.

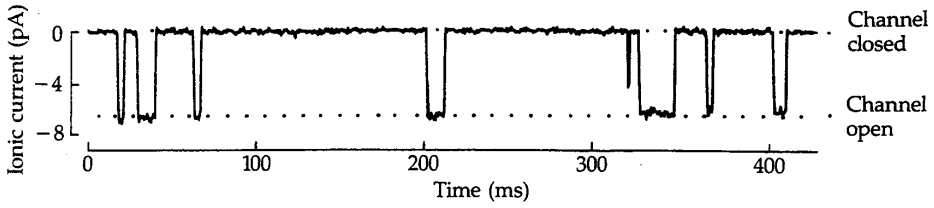


Figure 3: Ionic current across patch of excitable membrane (rat).

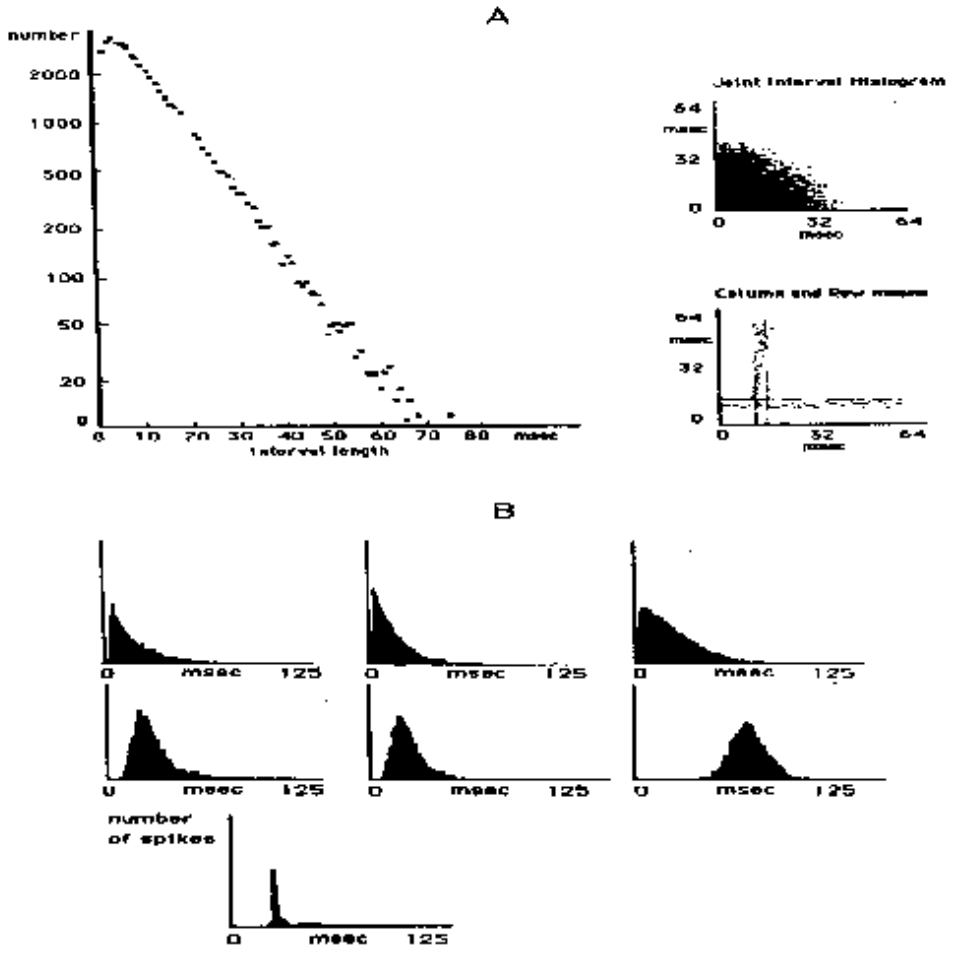


Figure 4: A) Spontaneous activity in the auditory nerve is quite variable between fibers, but all interval histograms can be described by an exponential distribution (left), and show independence of subsequent intervals (right). B) In the cochlear nucleus, a wide range of interval distributions are found ranging from exponential to nearly Gaussian.

In words, the probability to obtain x spikes in $[0, t+\delta t]$, is either to obtain x spikes in $[0, t]$ and no spikes in $[t, t+\delta t]$ or $x-1$ spikes in $[0, t]$ and one spike in $[t, t+\delta t]$. By direct substitution, one can easily verify that the solution is given by

$$p_x(t) = \frac{(\lambda t)^x}{x!} \exp(-\lambda t)$$

The expected number of spikes at time t is

$$\langle x \rangle_t = \sum_{x=0}^{\infty} x p_x(t) = \lambda t$$

The fluctuation in the number of spikes is given by

$$\sigma_t^2 = \left\langle x^2 \right\rangle_t - \langle x \rangle_t^2 = \lambda t$$

2.1.1 Interval distribution

The length of the inter-spike intervals (ISIs) can be computed for the Poisson process. The probability to observe an ISI of length t is denoted by $I(t)$ and is related to $p_0(t)$ in the following way. $I(t)\delta t$ =Prob{no firing in $[0, t]$}$\times$ Prob{firing in $[t, t+\delta t]$}= $\lambda p_0(t)\delta t$. Thus,

$$I(t) = \lambda \exp(-\lambda t)$$

The mean ISI is given by

$$\langle t \rangle = \int_0^{\infty} t I(t) dt = \lambda^{-1}$$

The variance in I is given by

$$\sigma_t^2 = \left\langle t^2 \right\rangle - \langle t \rangle^2 = \lambda^{-2}$$

The 'coefficient of variation' CV is defined as

$$CV = \frac{\sigma_t}{\langle t \rangle}.$$

$CV = 1$ for Poisson processes.

The CV is illustrated in fig. 5. Cell spiking activity was segmented according to mean firing rate (top right A and B) into 10 values. For each value, the ISI histogram is computed (top right C,D and E) and its corresponding coefficient of variation (bottom). We see that the measured CV is typically smaller than 1, and can therefore not be explained by the Poisson process. Simple extensions of the basic Poisson process can however explain these data, as we will show now.

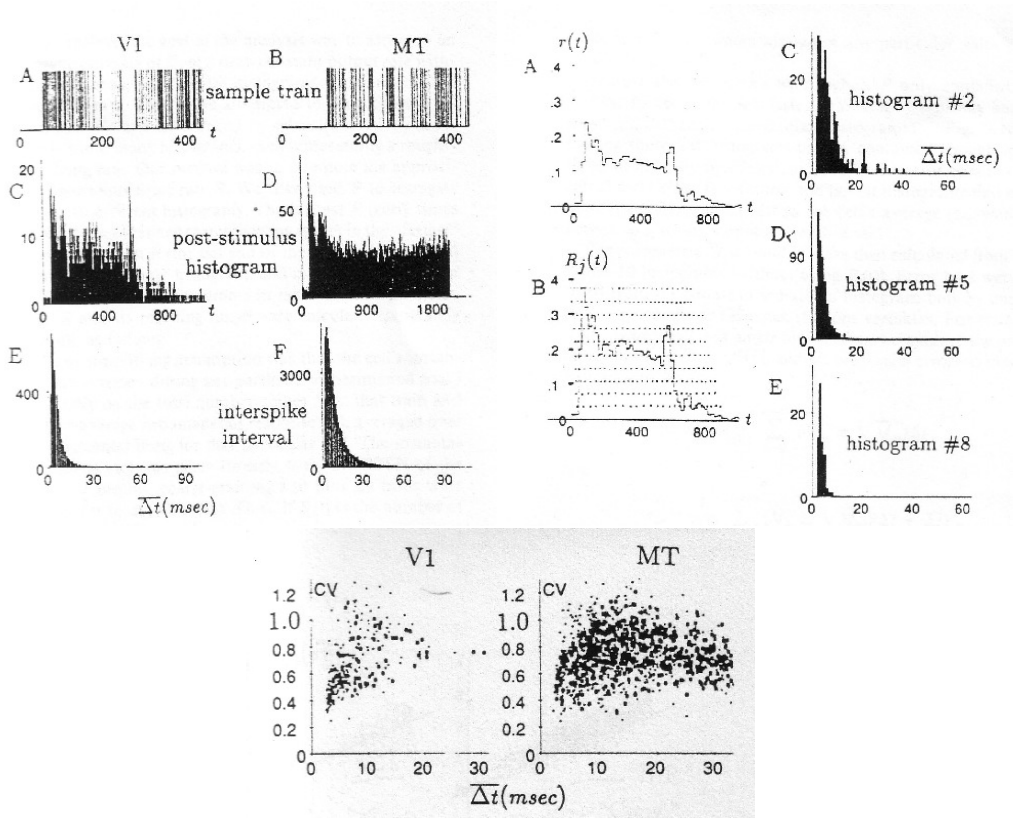


Figure 5: (Top left) Firing statistics of neurons in areas V1 (monkey; fixation task; cells were stimulated by flashed bars, Knierim and Van Essen 1992) and MT (monkey; discrimination task of movements in random dot patterns). A,B) Sample spike train C,D) PSTH; E,F) ISIs. (Top right) A,B) Instantaneous firing rates of V1 and MT cell. PSTHs were segmented according to instantaneous firing rate in 10 sets (0-9). C-E) ISI histograms are computed for different rates (segments 2, 5 and 8 shown). (Bottom) CV as a function of firing rate. Values between 20-30 msec are underestimated and are probability consistent with CV=1. For low rates, the data are in disagreement with the Poisson ansatz.

2.1.2 Integration reduces CV

Consider that the neuron receives input, which are independent Poisson spike trains with identical rates λ, and that it requires N of such input spikes before the neuron fires. The ISI distribution is now given by:

$I_N(t)\delta t$ =Prob{$N-1$ firings in $[0,t]$}$\times$ Prob{firing in $[t, t+\delta t]$}= $\lambda p_{N-1}(t)\delta t$. Thus,

$$I_N(t) = \frac{\lambda^N t^{N-1}}{(N-1)!}\exp(-\lambda t)$$

which is known as the Gamma distribution.

The mean and variance are given by

$$\langle t\rangle_N = \frac{N}{\lambda} \quad \sigma_t^2 = \frac{N}{\lambda^2} \tag{1}$$

The 'coefficient of variation' CV is therefore

$$CV = \frac{1}{\sqrt{N}}$$

Intuitively, this result can be easily understood. The time t it takes before the neuron fires is the *sum* of N independent contributions. From the law of large numbers, we know that the variance of such a sum scales as $1/\sqrt{N}$, which is what we see in CV.

2.1.3 Refractoriness

Cells must recover after firing as a result of Na inactivation which slowly restores the membrane potential to its resting value. If the refractory time is t_0, then eq. 1 is changed to

$$\langle t\rangle_N \rightarrow \langle t\rangle_N + t_0$$

and σ_t^2 is unaltered.

The effect of integration ($N > 1$) and refractoriness are illustrated in fig. 6. We see that neural integration (large N) is in disagreement with the observed large CV in fig. 5. If the neural code uses the (temporal average) firing rate as the carrier of information, then the neuron must integrate incoming activity over time to extract this rate from the incoming spike train. We see that integration implies small CV, whereas the data show rather large CV. Therefore we may conclude that the idea that mean firing rate is the carrier of information is contradicted by the above data.

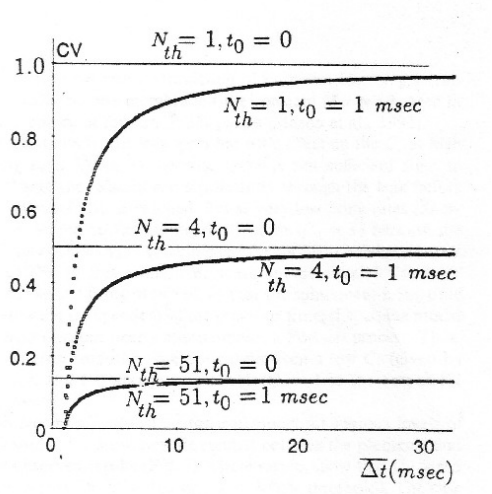


Figure 6: CV as predicted by the above model for different values of $N(= N_{\text{th}})$ and t_0

2.2 ISI distributions

We will now have a closer look at the ISI distributions as a whole, not just the means and variances.

For a Poisson process the ISI distribution is given by

$$I(t) = \lambda \exp(-\lambda t)$$

Therefore, if successive ISIs are independent, the joint probability for interval t_1 followed immediately by interval t_2 is

$$I(t_1, t_2) = \lambda^2 \exp(-\lambda(t_1 + t_2))$$

Thus, lines of constant probability are parallel to $t_1 + t_2 = 0$.

Consider now the probability density $I_k(t)$ for the interval between a spike and the kth that follows it. This is the Gamma distribution as given before:

$$I_k(t) = \frac{\lambda^k t^{k-1}}{(k-1)!} \exp(-\lambda t)$$

For large k, $I_k(t)$ tends towards a Gaussian distribution. In fig. 7, we see that cell 259-2 nicely obeys these two above properties.

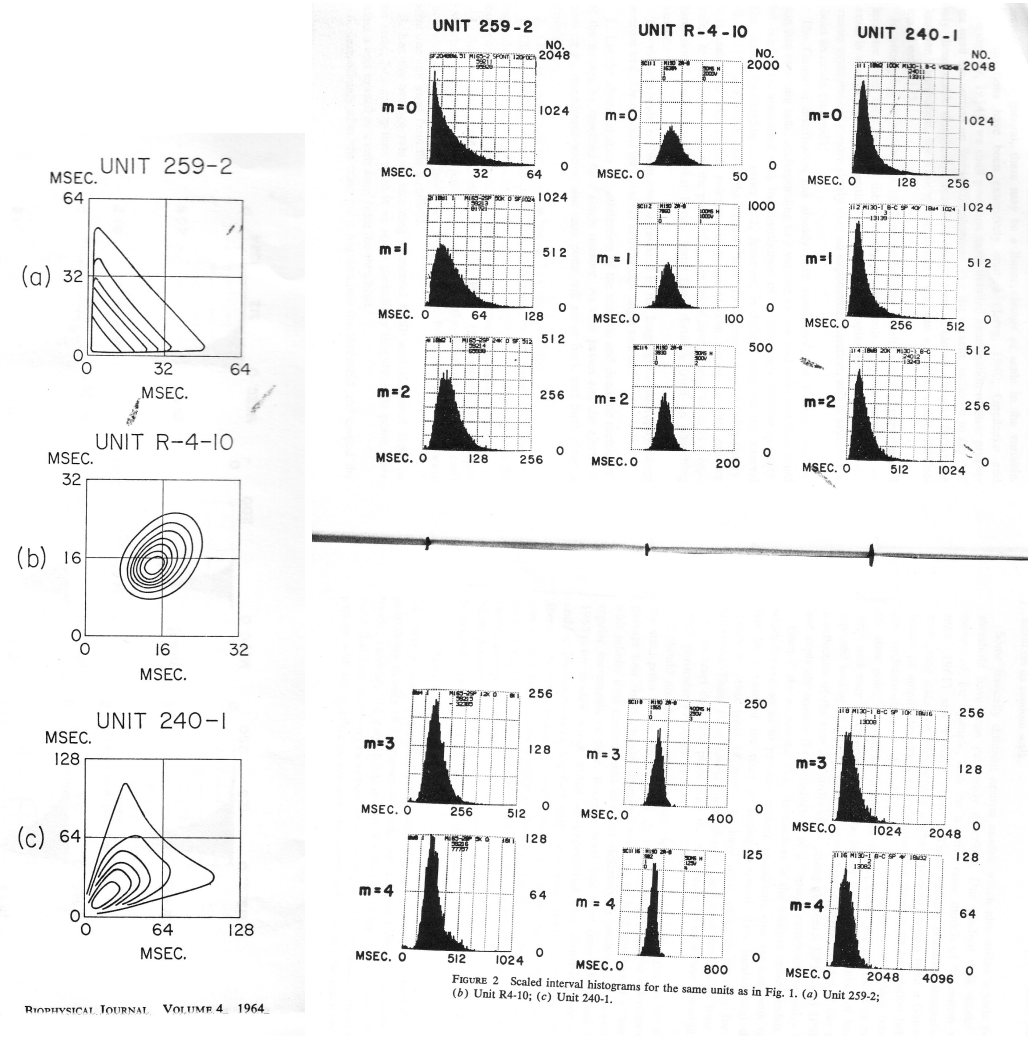


Figure 7: (Left) Observed joint ISI distribution of two subsequent intervals $I(t_1, t_2)$ for three cells from cat auditory system. (Right) $I_m(t)$ for same three cells and various m.

Cell R-4-10 has a more or less Gaussian ISI. If subsequent ISIs are independent and $I(t)$ is Gaussian, we obtain

$$I(t_1, t_2) \propto \exp(-\frac{1}{2\sigma^2}\left((t_1 - \bar{t})^2 + (t_1 - \bar{t})^2\right))$$

with $\bar{t}$ the mean ISI. This implies that contours of constant probability are circles. The convolution of a Gaussian is again a Gaussian. Therefore, $I_k(t)$ remains Gaussian. Cell R-4-10 in fig. 7 is an example (note the change of scale on the x axis).

For cell 240-1 we observe that the shape of $I_k(t)$ is more or less independent of k and only require a change of time scale by a factor 2. If we assume that succesive intervals are independent, the scaled interval histogram obeys:

$$I_2(t) = I(t) * I(t) = \int_0^t dt' I(t - t')I(t') = \frac{1}{2}I(t/2) \tag{2}$$

So, we should ask ourselves the question: Which distribution has the property that, when convoluted with itself, is of the same functional form? Only three distributions that satisfy this property are know in closed analytical form: the Gaussian distribution, the Cauchy distribution and the so-called stable distribution of order 1/2. The Gaussian distribution we already considered and does not match well the asymmetry and long tail of cell 240-1. The Cauchy distribution ($\propto (x^2 + 1)^{-1}$) has very long tails but is symmetric and therefore not easily restricted to $x \geq 0$.

The distribution of order 1/2 is defined only on the interval $x \geq 0$ and has one of the longest tails to be found in probability theory. It gives the probability density of first passage times in a one-dimensional random walk. This is the distribution that can be interpreted to result from a very simple neuron model and that we will now consider more in detail.

2.3 First passage times

The effect of a presynaptic spike on the post-synaptic neuron is a local change in the membrane potential. This change can be either positive or negative and is called the excitatory or inhibitory postsynaptic potential (PSP). The PSP is of the order of 0.05-2 mV [12] and is a stochastic event [13]: it either happens or it does not. The probability of the PSP is experimentally observed anywhere between 0.1 and 0.9 (see [14] and references there) and depends also on recent pre- and post-synaptic cell activity [15, 16].

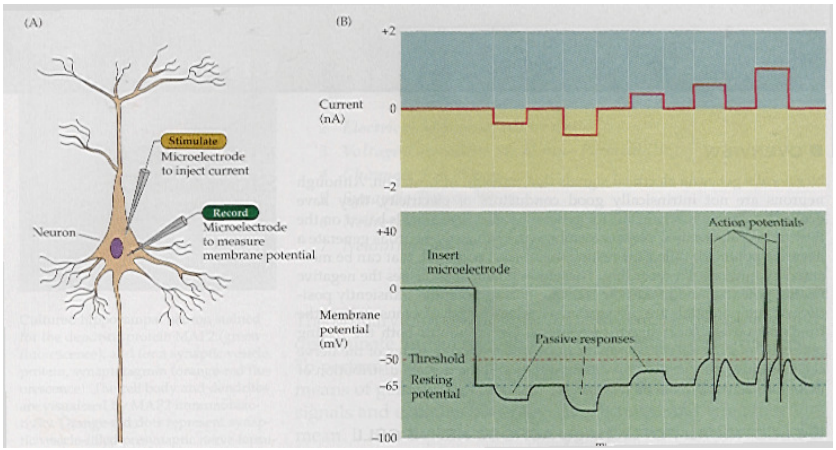


Figure 8: Basic response of a neuron (membrane potential) as a result of external current stimulation. The neuron displays two regimes. For small input currents, the response in approximately linear. For currents that exceed a threshold value, the neuron displays an autonomous response in the form of a stereotype action potential.

How these local changes in the membrane potential at synaptic junctions contribute to the spike generation process can be computed by compartmental modeling of the geometry of the cell. The dynamics is a complex spatio-temporal process involving many thousand synaptic inputs which are distributed over the dendrites and soma. A main complicating factor is that such simulations require the setting of many parameter values, many of which are not experimentally accessible. The general picture that emerges is, however, that the local PSP at the synapse propagates to the cell body with a delay of 1-2 msec and shows a temporal dispersion of about 10 msec. The dendrite acts as a low pass filter and strongly attenuates the frequency components of the PSP above 50 Hz [17].

The first passage time (FPT) distribution can be formulated as a random walk problem, which we here describe in terms of a simplified neuron model. This model mimics the linear and threshold behavior of real neurons as shown in fig. 8.

1. Let the electrical state of the neuron be specified by a single number, the membrane potential v.

2. Choose a particular point ($v = 0$) as the resting potential

3. Assume that each incoming EPSP increments v by one and each IPSP decreases v by one. These presynaptic events are random and independent:

$$v_{t+1} = v_t + \xi \tag{3}$$

where, $\xi = \pm 1$ is a random variable with mean value μ and variance σ^2.

4. Choose another point $v = v_{\text{th}}$ as the threshold. Whenever v reaches the threshold, a spike is emitted and v is reset to 0.

2.3.1 Diffusion

First consider that there is no threshold, and we wish to compute the probability $p(t, v|t_0, v_0)$, which is the probability that at time t the membrane potential is at v, *given* that at time t_0 the value was v_0.

Since v_t is a sum of $t - t_0$ independent contributions, $p(t, v|t_0, v_0)$ is Gaussian. The Gaussian is fully determined by its mean and variance. We easily compute the evolution of the mean by taking the expectation value of eq. 3 on both sides: $\langle v_{t+1} \rangle = \langle v_t \rangle + \mu$, and thus

$$\langle v \rangle_t = v_0 + \mu(t - t_0)$$

Squaring eq. 3 and taking the expectation value on both sides, we compute the time dependence of the variance: $\sigma^2_{t+1} = \left\langle v^2_{t+1} \right\rangle - \langle v_{t+1} \rangle^2 = \sigma^2_t + \sigma^2$:

$$\sigma^2_t = \sigma^2(t - t_0)$$

Thus the diffusion process is described by a Gaussian distribution with time dependent mean and variance:

$$p(t, v|t_0, v_0) = \frac{1}{\sqrt{2\pi(t - t_0)}\sigma} \exp\left(-\frac{(v - \langle v \rangle_t)^2}{2\sigma^2(t - t_0)}\right)$$

2.3.2 First passage time distribution

We can now compute $\rho(t)$: the probability distribution of the ISIs for the first passage time problem. We make use of the following identity:

$$p(t, v_{\text{th}}|0, 0) = \int_0^t dt' \, p(t, v_{\text{th}}|t', v_{\text{th}})\rho(t') \tag{4}$$

$\rho(t')$ is the probability to arrive after time t' *for the first time* to v_{th}, starting at time $t = 0$ at $v = 0$. Eq. 4 states that $\rho(t)$ is related to the diffusion distribution

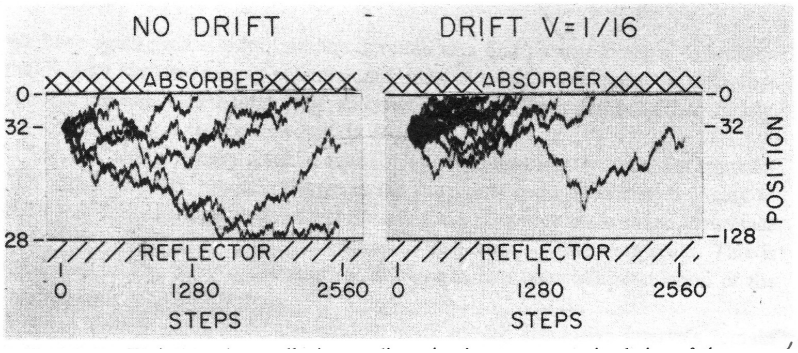


Figure 9: First passage time process. In this example, $v(t = 0) = 32$ and at each time, v is increased or decreased by one with equal probability. The first time $v(t) = 0$, a spike is emited and v is reset to 32. $\rho(t)$ gives the distribution of the first passage times.

$p(t, v_{\text{th}}|0, 0)$ by first moving in time t' from 0 to v_{th} (described by $\rho(t')$) and then at time t visiting the threshold again (described by $p(t, v_{\text{th}}|t', v_{\text{th}})$). The situation is depicted graphically in fig. 9.

The solution can be obtained using the Laplace transformation and is given by

$$\rho(t) = \frac{v_{\text{th}}}{\sqrt{2\pi}\sigma t^{3/2}} \exp\left(-\frac{(v_{\text{th}} - \mu t)^2}{2\sigma^2 t}\right) \tag{5}$$

We will now show how to obtain this solution.

2.3.3 Laplace transformation

We solve the integral equation using the Laplace transform. We briefly review its basic properties.

If $f(t)$ is a function of t, then the Laplace transform of f, denoted by $\hat{f}$ is given by

$$\hat{f}(s) = \int_0^\infty f(t) \exp(-st) dt$$

The Laplace transform is only defined for Re $s > 0$. The inverse transformation is given by

$$f(t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} ds \hat{f}(s) \exp(st)$$

with c some positive real number.

The Laplace transformation can be useful to solve integral equations of certain type. For instance, if

$$f_1(t) = \int_0^t dt' f_2(t-t') f_3(t'),$$

then

$$\hat{f}_1(s) = \hat{f}_2(s)\hat{f}_3(s)$$

Some useful Laplace relations are:

$f(t)$	$\hat{f}(s)$
$\frac{1}{\sqrt{\pi t}}$	$\frac{1}{\sqrt{s}}$
$\frac{1}{\sqrt{\pi t}} \exp(-k^2/4t)$	$\frac{1}{\sqrt{s}} \exp(-k\sqrt{s})$
$\frac{k}{2\sqrt{\pi t^3}} \exp(-k^2/4t)$	$\exp(-k\sqrt{s})$

We use the Laplace transformation to solve the integral equation eq. 4. We identify

$$\begin{aligned} f_1(t) &= p(t, v_{\mathsf{th}}|0,0) = \frac{1}{\sqrt{2\pi t}\sigma} \exp\left(-\frac{(v_{\mathsf{th}} - \mu t)^2}{2\sigma^2 t}\right) \\ f_2(t-t') &= p(t, v_{\mathsf{th}}|t', v_{\mathsf{th}}) \\ &= \frac{1}{\sqrt{2\pi(t-t')}\sigma} \exp\left(-\frac{\mu^2}{2\sigma^2}(t-t')\right) \\ f_3(t') &= \rho(t') \end{aligned}$$

We show the calculation in detail for the slightly simpler case $\mu = 0$. Then

$$\begin{aligned} \hat{f}_1(s) &= \frac{1}{\sqrt{2s}\sigma} \exp\left(-\sqrt{2s}\frac{v_{\mathsf{th}}}{\sigma}\right) \\ \hat{f}_2(s) &= \frac{1}{\sqrt{2s}\sigma} \end{aligned}$$

Thus,

$$\hat{\rho}(s) = \frac{\hat{f}_1(s)}{\hat{f}_2(s)} = \exp\left(-\sqrt{2s}\frac{v_{\mathsf{th}}}{\sigma}\right)$$

$$\rho(t) = \frac{v_{\mathsf{th}}}{\sqrt{2\pi}\sigma t^{3/2}} \exp\left(-\frac{v_{\mathsf{th}}^2}{2\sigma^2 t}\right)$$

which is in agreement with eq. 5.

2.3.4 Scale invariance

It is easy to see that without drift ($\mu = 0$), $\rho(t)$ is invariant under convolution with itself. The sum of two subsequent intervals is distribution as

$$\rho_2(t) = \int_0^t dt' \rho(t - t')\rho(t')$$

We easily solve this using the Laplace transform:

$$\hat{\rho}_2(s) = (\hat{\rho}(s))^2 = \exp\left(-2\sqrt{2s}\frac{v_{\text{th}}}{\sigma}\right)$$

from which we get

$$\rho_2(t) = \frac{2v_{\text{th}}}{\sqrt{2\pi}\sigma t^{3/2}} \exp\left(-\frac{2v_{\text{th}}^2}{\sigma^2 t}\right) = \frac{1}{4}\rho(t/4)$$

So $\rho_2(t)$ has the same shape as $\rho(t)$. However, note that the scaling factor 4 is different from the scaling factor 2 that we observe experimentally (see eq. 2). In addition, for large k we observe that the observed distribution eventually becomes Gaussian, whereas ρ_k (the distribution of the sum of k subsequent ISIs) remains scale invariant.

2.3.5 Approximate scale invariance.

When $\mu \neq 0$, the first passage time distribution is unfortunately no longer invariant under convolution with itself. However, it is easy to show that for a certain range of parameters there is approximate shape invariance with $2\times$ expansion of the time scale. This is illustrated in fig. 10.

The first passage time distribution can be fitted to the data and has two free parameters:

$$\rho(t) \propto \frac{1}{t^{3/2}} \exp\left(-\frac{a}{t} - bt\right) \tag{6}$$

The best fits are shown for three different neurons in fig. 11. Despite the great disparity of experimental conditions and anatomical location, the interval histograms of the data from both units 240-1 and 6-2 are well fitted by similar choice of parameter values. From eq. 5, we infer that $a = \frac{v_{\text{th}}^2}{2\sigma^2}$ and $b = \frac{\mu^2}{2\sigma^2}$. If we set the step-size of the random walker equal to one ($\sigma^2 = 1$), then the fit of unit 240-1 shows that the threshold is about 7 steps above the resting membrane potential

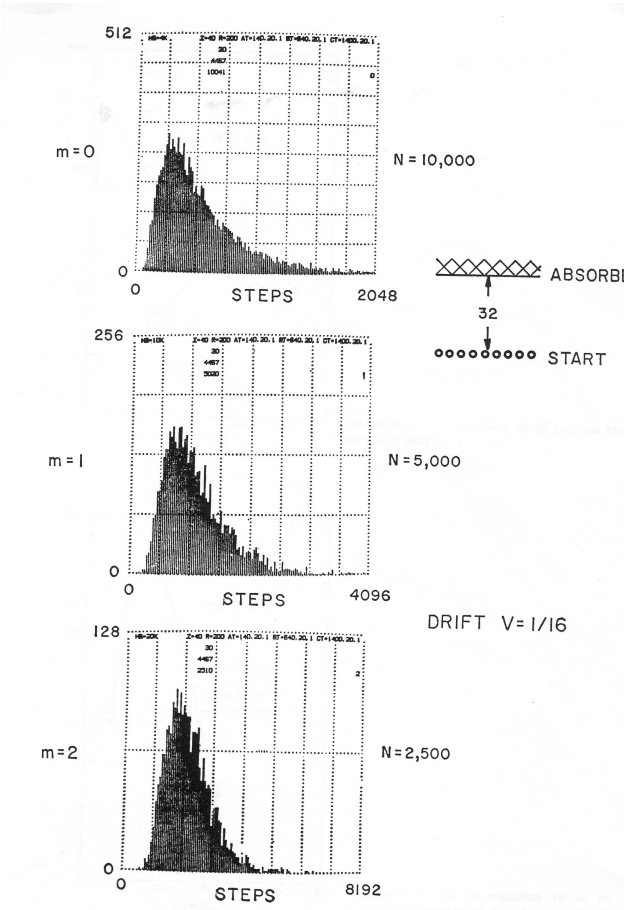


Figure 10: Simulated ISIs from first passage time distribution with drift $\mu = 1/16$ and threshold $v_{\text{th}} = 32$ shows approximate invariance under convolution with $2\times$ expansion of the time scale.

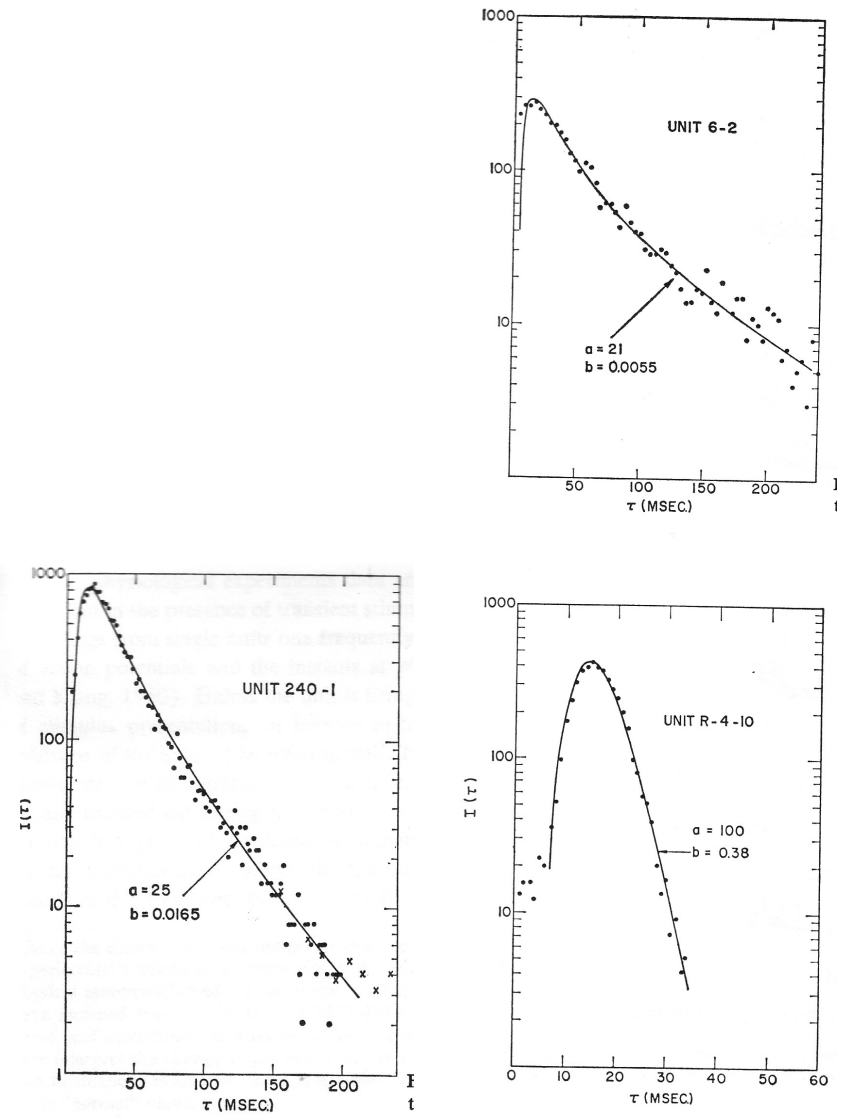


Figure 11: Unit 240-1. Cochlear nucleus anesthetized cat. Unit 6-2. Auditory cortex unanesthetized cat. The 'Gaussian' ISI distribution of Unit R-4-10 can also be rather well fitted with this model, but requires rather different values for a and b.

and the drift $\mu = 0.18$ is small. The Gaussian-like data of unit R-4-10 can alos be fitted by eq. 6, but with very different values of a and b. This neuron has a higher threshold than unit 240-1 (14 vs. 7) and a significant drift ($\mu = 0.87$). These estimates imply that the mean ISI value should be approximately $14/0.87 \approx 16$ msec, which is in agreement with the peak value of the ISI distribution in fig. 11.

2.4 Integrate and Fire Neuron

The integrate and fire model improves the random walk model, by introducing a 'leak' term in the dynamics that drives the membrane potential back to its resting value. As can be seen from fig. 8, for small currents the response is well described by a linear differential equation of the form

$$C\frac{dV}{dt} = -\frac{V}{R} + I(t)$$

When $V > V_{\text{th}}$ a spike is generated and $V \rightarrow V_{\text{rest}} = 0$.

For constant current ($I(t) = I_0, t > 0, V(0) = 0$) the differential equation in the absence of a threshold can be easily solved:

$$V(t) = I_0 R\left(1 - e^{-t/\tau}\right), \quad \tau = RC$$

Thus, for large t the membrane potential saturates at a value $V(\infty) = I_0 R$. If $V(\infty) => V_{\text{th}}$ a spike is produced at time t^* which satisfies

$$V_{\text{th}} = I_0 R\left(1 - e^{-t^*/\tau}\right),$$

or

$$t^* = \tau_{\text{ref}} - \tau \log\left(1 - \frac{V_{\text{th}}}{I_0 R}\right) \tag{7}$$

This mechanism is graphically illustrated in fig. 12.

Real neurons do not respond with a purely regular spike train of constant frequency to a constant input current, as can be in the right part of fig. 12. Instead, the firing rate is slowly decreasing with time. This can be easily incorporated in the integrate-and-fire neuron model by introducing a time-dependent conductance g_{adapt} (with reversal potential equal to the resting potential). Each spike increases this conductance by a fixed amount g_0. Between spikes, g_{adapt} decreases exponentially to zero with a time constant τ_{adapt}:

$$C\frac{dV}{dt} = -\frac{V}{R} - g_{\text{adapt}} V + I(t) \tag{8}$$

$$\tau_{\text{adapt}}\frac{dg_{\text{adapt}}}{dt} = -g_{\text{adapt}} \tag{9}$$

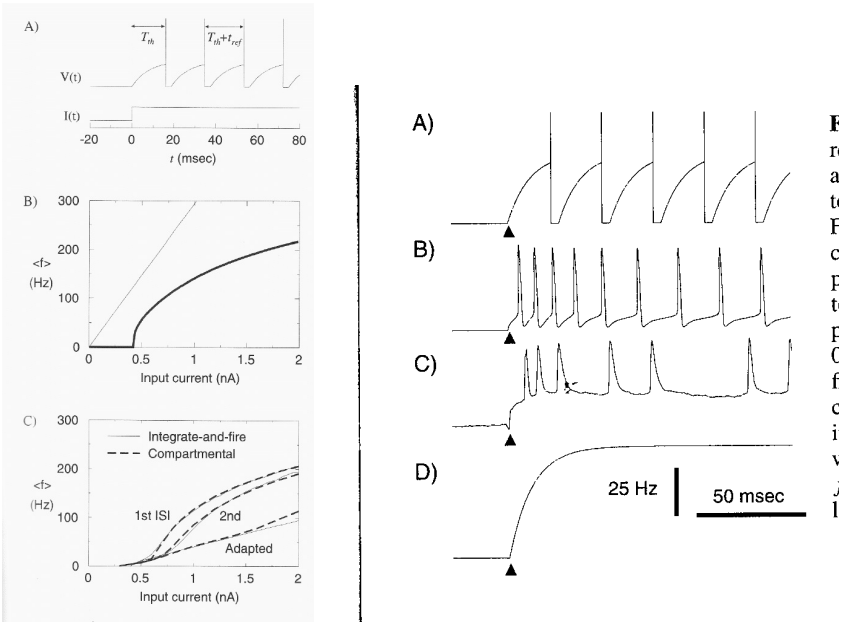


Figure 12: Behavior of the integrate-and-fire neuron when stimulated by a constant current. Left A,B) When the current exceeds a minimal value, $I_0 > V_{\text{th}}/R$, the neuron will regularly emit a spike with frequency $1/t^*$, as given by eq. 7. Right B). Somatic membrane potential in a layer 5 pyramidal cell compartmental model to a 1.5 nA current input. Right C) Response of a cell in the primary visual cortex of the anesthetized adult cat to a 0.6 nA current injection. In both cases B and C, the firing rate decreases gradually with time despite the constant input current. Left C) The adapting IF model of eq. 9 can correctly reproduce these findings.

If V reaches V_{th}, a spike is generated and g_{adapt} is incremented by g_0. Such a mechanism can be interpreted as the effect of a calcium-dependent potassium conductance. The ISIs produced by this adapting integrate-and-fire model reproduce correctly the experimental neural data, as shown in fig. 12.

In vivo, the current $I(t)$ is a sum of excitatory and inhibitory pre-synaptic events. When we assume that

$$I(t) = \xi$$

with $\xi = \pm 1$ denoting an EPSP or IPSP, respectively, the IF model becomes similar to the random walk model. However, an important difference is the leak term V/R. Whereas, for the non-leaky IF model we can solve the first passage time problem and obtain an analytical expression for the ISI distribution (see eq. 5), this is no longer true when a leak term is included.

2.5 Summary

Neuron spiking is noisy and requires a statistical description. Such a description can be done in terms of firing rates or inter-spike interval distributions. The coefficient of variation is a measure for deviation of spike train from regular firing [18]. It is zero for a regular spike train and 1 for a Poisson process. If many pre-synaptic events contribute to a spike (integration) one expects a low coefficient of variation, which is not observed experimentally.

Most observed ISI distributions can be modeled with a Poisson, Gaussian or first passage time distribution. We have assumed independence of subsequent ISIs, which is often not true in particular for short ISIs. The FPT distribution, can fit a large variety of distributions, but note that it cannot fit the Poisson distribution. When non-stationary input is presented, the distribution can be rather more complicated (multi-modal). The discussion of the FPT distribution is based on [19].

We have introduced two simplified neuron models: FPT and IF. They differ by a passive leak term, which makes the FPT model solvable for stochastic input and the IF model not. The IF model is more realistic because it takes into account that the PSP decays with time with a time constant on the order of 5-15 ms. Various approximations to the first passage time problem for the integrate-and-fire neuron are studied in [20].

Even the IF model is approximate at best. We describe the electrical state of the entire neuron by a single number. Real neurons have a complex geometry, and the spike generation process can initiate at various loci. We ignore the fact that

the size of the PSP depends on the membrane potential. This dependence is due to the fact that the membrane conductance is voltage dependent ($I = g(V)(V - V_{\mathsf{rev}})$).

2.6 Exercises

1. Show that the Gamma distribution for large N becomes the Gaussian distirbution.

2. Verify that the Laplace transformation of $f(t) = \frac{1}{\sqrt{\pi t}}$ is $\hat{f}(s) = \frac{1}{\sqrt{s}}$.

3. For a diffusion process in one dimension that starts at $t = t_0$ at location $v = v_0$, the probability to observe the membrane potential v at a later time t is given by

$$p(t, v|t_0, v_0) = \frac{1}{\sqrt{2\pi(t - t_0)}\sigma} \exp\left(-\frac{(v - \langle v\rangle_t)^2}{2\sigma^2(t - t_0)}\right)$$

   with $\langle v\rangle_t = v_0 + \mu(t - t_0)$. The first passage time distribution satisfies the integral equation

$$p(t, v_{\mathsf{th}}|0, 0) = \int_0^t dt' \, p(t, v_{\mathsf{th}}|t', v_{\mathsf{th}})\rho(t', v_{\mathsf{th}})$$

   (a) Using Laplace transforms, compute the first passage time distribution for $\mu = 0$, following the derivation in the text.

   (b) Using Laplace transforms, compute the first passage time distribution for general μ.

4. Compute the expected ISI time for the first passage time distribution.

5. Compute the output frequency of the FPT model with constant input μ and compare with the IF model.

3 Networks of binary neurons

3.1 Stochastic binary neurons and networks

We have seen that even for a very simple neuron model, such as the integrate-and-fire model, the relation between the stochastic input and the output can be too complex to be described in analytical form. Therefore, in order to study the behavior of networks of neurons we may try to find a more compact description of a neuron which ignores its internal details but retains some of its input-output behavior. Let us look again at fig. 12Left C, which shows the output frequency of a biologically realistic compartmental model of a layer 5 pyramidal cell as a function of a constant input current. A very simple model that captures this relation is to state that in each small but finite time interval Δt the neuron can either emit one spike or no spike. Thus, the output of the neuron is described by a binary stochastic variable $y = 0, 1$ which defines the number of spikes in Δt. The probability of $y = 1$ is then proportional to the firing frequency of the neuron, which as we see in fig. 12Left C is a non-linear function of the input current I:

$$p(y = 1) = \sigma(I)$$

A common choice for σ is the sigmoid function $\sigma(x) = \frac{1}{2}(1 + \tanh(x))$. As a result of the discretization of time in intervals of size Δt, the maximal firing frequency is clearly $1/\Delta t$. In fig. 13, we show that the curves of fig. 12Left C can be reproduced approximately in the binary neuron model. But it should be understood that whereas real neurons adapt their firing rate through time, the firing rate of the binary neuron is constant for any value of I and Δt.

Suppose that we have a network of binary neurons and consider neuron i. The current I_i represents the total current input to the neuron and consists of current contributions I_{ij} from all neurons that make synaptic contacts onto the neuron and that have recently fired (we will ignore temporal delays in the propagation of action potentials). Therefore, we may write

$$I_i(t) = \sum_{j \neq i} I_{ij} y_j(t) + \Theta_i$$

where Θ_i is a free parameter that must be adjusted such that $\sigma(\Theta_i)$ is equal to the firing rate of the neuron in the absence of any input. t labels the discretized time in units of Δt. The probability of firing of neuron i at the next time step is thus dependent on $y(t) = (y_1(t), \ldots, y_n(t))$ which we call the *state* of the network at

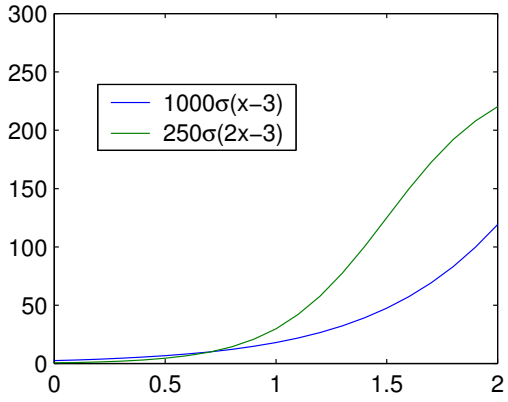


Figure 13: In the binary neuron model, the firing frequency of the neuron is a non-linear function of the input current.

time t (n is the number of neurons in the network), or

$$p(y_i = 1, t+1|y(t)) = \sigma(\sum_{j\neq i} I_{ij} y_j(t) + \Theta_i) \tag{10}$$

For our subsequent analysis, we will find it useful to replace the binary variables $y_i = 0, 1$ by the variables $s_i = \pm 1$ using the relation $y_i = \frac{1}{2}(y_i + 1)$. Then eq. 10 becomes

$$\begin{aligned} p(s_i', t+1|\mathbf{s}, t) &= \sigma(s_i' h_i(\mathbf{s}(t)) \\ h_i(\mathbf{s}) &= \sum_{j\neq i} w_{ij} s_j + \theta_i \\ w_{ij} &= \frac{1}{2} I_{ij} \\ \theta_i &= \Theta_i + \frac{1}{2}\sum_{j\neq i} I_{ij} \end{aligned} \tag{11}$$

In Eq. 11, we have made use of the property $\sigma(x) + \sigma(-x) = 1$, which allows us to write the probability for both $s_i' = \pm 1$ in one expression and $\mathbf{s} = (s_1, \ldots, s_n)$. Note, that Eq. 11 does not explicity depend on time, but only implicitly through the dependence of $\mathbf{s}$ on time.

3.1.1 Parallel dynamics: Little model

Eq. 11 describes the *conditional* probability for a single neuron to emit a spike between t and $t+1$, given an input activity $\mathbf{s}$. In a network of neurons, this equation must be updated in parallel for all neurons. Thus, the transition probability from a state $\mathbf{s}$ at time t to a state $\mathbf{s}'$ at time $t+1$ is given by

$$T(\mathbf{s}'|\mathbf{s}) = \prod_i p(s_i', t+1|\mathbf{s}, t) \tag{12}$$

with $p(s_i', t+\tau|\mathbf{s}, t)$ given by Eq. 11. T denotes the probability to observe the network in state $\mathbf{s}'$, given the fact that it was in state $\mathbf{s}$ at the previous time step. Since the dynamics is stochastic, the network will in general not be found in any one state but instead in a superposition of states. Therefore, the fundamental quantity to consider is $p_t(\mathbf{s})$, denoting the probability that the network is in state $\mathbf{s}$ at time t. The dynamics of the network is therefore defined as

$$p_{t+1}(\mathbf{s}') = \sum_{\mathbf{s}} T(\mathbf{s}'|\mathbf{s}) p_t(\mathbf{s}). \tag{13}$$

Eq 13 is known as a first order homogeneous Markov process. The first order refers to the fact that the probability of the new state only depends on the current state and not on any past history. Homogeneous means that the transition probability is not an explicit function of time, as can be verified by Eq. 11. This Markov process was first considered by Little [21].

3.1.2 Sequential dynamics

One of the drawbacks of parallel dynamics is that due to the strict discretization of time in intervals of length Δt, an external clock is implicitly assumed which dictates the updating of all the neurons. There exists another stochastic dynamics which has been used extensively in the neural network literature which is called sequential Glauber dynamics. Instead of updating all neuron in parallel, one neuron is selected at random and is updated. The neurobiological motivation that is sometimes given for this dynamics is that neurons are connected with random delays [22]. However, in my view a more important reason for the popularity of sequential dynamics is that the stationary distribution is a Boltzmann-Gibbs distribution when the connectivity in the network is symmetric. This makes the connection to statistical physics immediate and allows for all the powerful machinery of mean field theory to be applied. Also, the parameters (weights and

thresholds) in the Boltzmann-Gibbs distribution can be adapted with a learning algorithm which is known as the Boltzmann Machine [9].

The sequential dynamics is defined as follows. At every iteration t, choose a neuron i at random. Update the state of neuron i using Eq. 11. Let $\mathbf{s}$ denote the current state of the network and let F_i denote a flip operator that flips the value of the ith neuron: $\mathbf{s}' = F_i\mathbf{s} \Leftrightarrow s'_i = -s_i$ and $s'_j = s_j$ for all $j \neq i$. Thus, the network can make a transition to state $\mathbf{s}' = F_i\mathbf{s}$ with probability

$$T(\mathbf{s}'|\mathbf{s}) \quad = \quad \frac{1}{n} p(s'_i, t+\tau|\mathbf{s}, t), \quad \text{if } \mathbf{s}' = F_i\mathbf{s} \tag{14}$$

and zero if $\mathbf{s}'$ differs more thane one bit from $\mathbf{s}$. $p(s'_i, t+\tau|\mathbf{s}, t)$ is again given by Eq. 11. The factor $\frac{1}{n}$ is a consequence of the random choice of the neurons at each iteration. The probability to remain in state $\mathbf{s}$ is given by the equality $\sum_{\mathbf{s}'} T(\mathbf{s}'|\mathbf{s}) = 1$, so that

$$T(\mathbf{s}|\mathbf{s}) = 1 - \frac{1}{n} \sum_i p(s'_i, t+\tau|\mathbf{s}, t). \tag{15}$$

Eqs. 14 and 15 together with Eq. 13 define the sequential dynamics. Note, that this dynamics allows only transitions between states $\mathbf{s}$ and $\mathbf{s}'$ that differ at most at one location, whereas the Little model allows transitions between all states.

3.2 Some properties of Markov processes

In this section, we review some of the basic properties of first order Markov processes. For a more thorough treatment see [23].

3.2.1 Eigenvalue spectrum of T

Let $\mathcal{S}$ denote the set of all state vectors $\mathbf{s}$. $\mathbf{s} \in \mathcal{S}$ is a binary vector of length n and thus $\mathbf{s}$ can take on 2^n different values. Therefore, $p_t(\mathbf{s})$ in Eq. 13 is a vector of length 2^n and $T(\mathbf{s}'|\mathbf{s})$ is a $2^n \times 2^n$ matrix. Since $p_t(\mathbf{s})$ denotes a probability vector, it must satisfy $\sum_{\mathbf{s}} p_t(\mathbf{s}) = 1$. In addition, $T(\mathbf{s}'|\mathbf{s})$ is a probability vector in $\mathbf{s}'$ for each value of $\mathbf{s}$ and therefore each column must add up to one:

$$\sum_{\mathbf{s}'} T(\mathbf{s}'|\mathbf{s}) = 1. \tag{16}$$

Matrices with this property are called stochastic matrices.

Let us denote the eigenvalues and left and right eigenvectors of T by $\lambda_\alpha, l_\alpha, r_\alpha, \alpha = 1, \ldots, 2^n$, respectively [1]. In matrix notation we have

$$\begin{aligned} T r_\alpha &= \lambda_\alpha r_\alpha \\ l_\alpha^\dagger T &= \lambda_\alpha l_\alpha^\dagger \end{aligned}$$

Since T is a non-symmetric matrix, the left and right eigenvectors are different, non-orthogonal and complex valued. $\dagger$ denotes complex conjugation and transpose. The eigenvalues are also complex valued. Under rather general conditions each set of eigenvectors spans a non-orthogonal basis of C^{2^n}, the complex 2^n dimensional space. These two bases are *dual* in the sense that:

$$l_\alpha^\dagger r_\beta = \delta_{\alpha\beta}. \tag{17}$$

δ_{ab} denotes the Kronecker delta: $\delta_{ab} = 1$ if $a = b$ and 0 otherwise. In Eq. 17, a and b are simple numbers, but below we wull also see cases where they are vectors, such as the state of the network. We can therefore expand T on the basis of its eigenvectors:

$$T = \sum_{\alpha=1}^{2^n} \lambda_\alpha r_\alpha l_\alpha^\dagger$$

If at $t = 0$ the network is in a state $\mathbf{s}^0$ then we can write the probability distribution at $t = 0$ as $p_0(\mathbf{s}) = p_{t=0}(\mathbf{s}) = \delta_{\mathbf{s},\mathbf{s}^0}$. The probability vector p_t at some later time t is obtained by repeated application of Eq. 13:

$$p_t = T^t p_0 = \sum_\alpha \lambda_\alpha^t r_\alpha (l_\alpha^\dagger \cdot p_0) \tag{18}$$

where $T^t p_0$ denotes t times the multiplication of the matrix T with the vector p_0, and the $\cdot$ denotes inner product. The stationary probability distribution of the stochastic dynamics T is given by p_∞ which is invariant under the operation of T and therefore satisfies

$$T p_\infty = p_\infty. \tag{19}$$

Thus, the stationary distribution is a right eigenvector of T with eigenvalue 1.

[1] In general, the number of eigenvalues of T can be less than 2^n. However, for our purposes we can ignore this case

3.2.2 Ergodicity and ergodicity breaking

A Markov process is called *irreducible*, or *ergodic*, on a subset of states $C \subset \mathcal{S}$ if for any state $\mathbf{s} \in C$ there is a finite probability to visit any other state $\mathbf{s}' \in C$. This means that for any two states $\mathbf{s}, \mathbf{s}' \in C$, there exists a number k and a set of intermediate states $\mathbf{s} = \mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^k = \mathbf{s}'$ such that $\prod_{i=1}^{k} T(\mathbf{s}^i|\mathbf{s}^{i-1}) > 0$. In words, between any two states in an irreducible set there exists a path of transitions with non-zero probability. A subset of states $C \subset \mathcal{S}$ is called *closed* when the Markov process can never escape from C, once entered: $T(\mathbf{s}'|\mathbf{s}) = 0$ for all $\mathbf{s} \in C, \mathbf{s}' \neg \in C$. A subset of states $\mathcal{T}$ is called *transient* when the Markov process can never enter in $\mathcal{T}$, once outside: $T(\mathbf{s}'|\mathbf{s}) = 0$ for all $\mathbf{s} \notin \mathcal{T}, \mathbf{s}' \in \mathcal{T}$. It is a property of homogeneous first order Markov processes that one can partition the state space $\mathcal{S}$ uniquely into closed irreducible subsets C_i and a transient set $\mathcal{T}$: $\mathcal{S} = \mathcal{T} \cup C_1 \cup C_2 \ldots$.

For an irreducible Markov process of *periodicity* d the Perron-Frobenius theorem states that T has d eigenvalues given by

$$\lambda_m = \exp(2\pi i m/d), m = 0, \ldots, d-1,$$

and all remaining eigenvalues of T are inside the unit circle in the complex plane: $|\lambda_\alpha| < 1$ [2] . In particular, T has exactly one eigenvalue 1. Its corresponding right eigenvector is equal to the (unique) stationary distribution. Note, that the left eigenvector with eigenvalue 1 is $\propto (1, \ldots, 1)$ as is immediately seen from Eq. 16. The right eigenvector, in contrast, is in general difficult to compute, as will be seen later.

A non-irreducible or non-ergodic Markov process has more than one eigenvalue 1 and therefore more than one left and right eigenvector with eigenvalue

[2] The fact that all eigenvalues are within the unit circle in the complex plane can be easily demonstrated in the following way. Let λ be an eigenvalue of T and l its corresponding left eigenvector. Then for all s,

$$(\lambda - T(s|s))l(s) = \sum_{s' \neq s} l(s')T(s'|s).$$

Choose s such that $|l(s)|$ is maximal. Then

$$|\lambda - T(s|s)| = \frac{1}{|l(s)|} |\sum_{s' \neq s} l(s')T(s'|s)| \leq \sum_{s' \neq s} T(s'|s) = 1 - T(s|s).$$

This statement is known as Gershgoren's Theorem. Thus, λ is within a circle of radius $1 - T(s|s)$ centered at $T(s|s)$. We do not know which s maximizes $|l(s)|$ and therefore we do not know the value of $T(s|s)$. However, since circles with smaller $T(s|s)$ contain circles with larger $T(s|s)$, λ is in the largest circle: $|\lambda| < 1$. This completes the proof.

1. Let us denote these eigenvectors by $l_1, \ldots, l_k$ and $r_1, \ldots, r_k$, respectively. Any linear combination of the right eigenvectors

$$p_\infty = \sum_{\alpha=1}^{k} \rho_\alpha r_\alpha \tag{20}$$

is therefore a stationary distribution, assuming proper normalization: $p_\infty(\mathbf{s}) \geq 0$ for all $\mathbf{s}$ and $\sum_{\mathbf{s}} p_\infty(\mathbf{s}) = 1$. Thus, there exists a manifold of dimension $k-1$ of stationary distributions.

In addition, the k left eigenvectors with eigenvalue 1 encode *invariants* of the Markov process in the following way. Let the state of the network at time t be given by p_t. Define the numbers $L_\alpha^t = l_\alpha^\dagger \cdot p_t, \alpha = 1, \ldots, k$ as the inner product of l_α with the probability distribution at time t. Then it is easy to see that the L_α^t are invariant under the Markov dynamics:

$$L_\alpha^{t+1} = l_\alpha^\dagger p_{t+1} = l_\alpha^\dagger T p_t = l_\alpha^\dagger p_t = L_\alpha^t.$$

where the forelast step follows because l_α is a left eigenvector of T with eigenvalue 1. We can thus drop the time index on L_α. One of these invariants is the left eigenvector $l_1 \propto (1, \ldots, 1)$ which ensures that the normalization of the probability vector p_t is conserved under the Markov process. The value of the remaining $k-1$ invariants are determined by the initial distribution p_0. Since their value is unchanged during the dynamics they parametrize the stationary manifold and determine uniquely the stationary distribution. We can thus compute the dependence of the stationary distribution on the initial state. Because of Eq. 17 and Eq. 20, we obtain $L_\alpha = l_\alpha^\dagger p_0 = l_\alpha^\dagger p_\infty = \rho_\alpha$. Thus, the stationary state depends on the initial state as

$$p_\infty = \sum_{\alpha=1}^{k} (l_\alpha^\dagger p_0) r_\alpha. \tag{21}$$

Note, that in the ergodic case ($k = 1$) the dependence on the initial state disappears, as it should, since $l_1^\dagger p_0 = 1$ for any (normalized) initial distribution.

The time it requires to approach stationarity is also given by the eigenvalues of T. In particular, each eigenvalue whose norm $|\lambda_\alpha| < 1$ corresponds to a transient mode in Eq. 18 with *relaxation time* $\tau_\alpha = \frac{-1}{\log \lambda_\alpha}$.

Both concepts of irreducibility and periodicity are important for neural networks and we therefore illustrate them with a number of simple examples. Consider a network of two neurons connected symmetrically by a synaptic weight

$w = w_{12} = w_{21}$ and thresholds zero. First consider sequential dynamics. The network has four states, the transition matrix T can be computed from Eqs. 14 and 15 and has 4 eigenvalues. Their values as a function of w are plotted in Fig. 14a. We

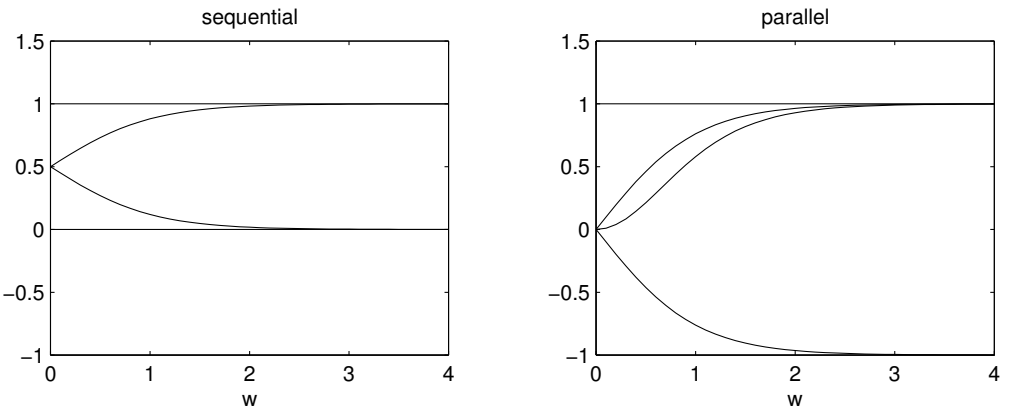


Figure 14: Eigenvalues of T as a function of w under sequential and parallel dynamics. For large w, multiple eigenvalues 1 signal ergodicity breaking.

observe, that for small w there exists only one eigenvalue 1. Its corresponding right eigenvector is the Boltzmann-Gibbs distribution $p(s_1, s_2) = \frac{\exp(ws_1s_2)}{Z}$ as will be shown below. For small weights, the dynamics is ergodic: for any initialization of the network the asymptotic stationary distribution is the Boltzmann-Gibbs distribution. The dominant relaxation time is given by the largest eigenvalue that is smaller than 1. For larger w, we observe that the relaxation time becomes infinite because a second eigenvalue approaches 1. This means that some transitions in the state space require infinite time and therefore ergodicity is broken. From the Boltzmann-Gibbs distribution, we see that the large weight prohibits the two neurons to have opposite value and therefore only the states $(1, 1)$ and $(-1, -1)$ have positive probability in this distribution. The ergodicity breaking signals the fact that transitions between $(1, 1)$ and $(-1, -1)$ become impossible.

Let us denote the 4 states $(1, 1), (1, -1), (-1, 1), (-1, -1)$ by $\mathbf{s}^\mu, \mu = 1, \ldots, 4$. The right eigenvectors with eigenvalue 1 are the Boltzmann-Gibbs distribution

$$r_1(\mathbf{s}) = \frac{1}{2}(\delta_{\mathbf{s},\mathbf{s}^1} + \delta_{\mathbf{s},\mathbf{s}^4})$$

and the vector

$$r_2(\mathbf{s}) = \frac{1}{2}(\delta_{\mathbf{s},\mathbf{s}^1} - \delta_{\mathbf{s},\mathbf{s}^4})$$

The stationary distribution is no longer unique and consists of any linear combination of r_1 and r_2 that is normalized and positive: $p_\infty = r_1 + \rho_2 r_2$, with $-1 < \rho_2 < 1$. As a result, any convex combination $\lambda\delta_{\mathbf{s},\mathbf{s}^1} + (1-\lambda)\delta_{\mathbf{s},\mathbf{s}^4}$, with $0 < \lambda < 1$ is a stationary distribution.

As we showed above, the particular stationary distribution that is attained by the network is determined by the initial condition, in particular by the invariants L_α. The left eigenvectors with eigenvalue 1 are

$$\begin{aligned} l_1(\mathbf{s}) &= 1 \\ l_2(\mathbf{s}) &= \delta_{\mathbf{s},\mathbf{s}^1} - \delta_{\mathbf{s},\mathbf{s}^4} \end{aligned}$$

It can be checked that the vectors r_α and l_α satisfy the duality relation Eq. 17. The corresponding quantities L_1 and L_2 are conserved and the dependence of the stationary distribution on the initial distribution is given by Eq. 21:

$$p_\infty = L_1 r_1 + L_2 r_2 = \frac{1}{2}(1+L_2)\delta_{\mathbf{s},\mathbf{s}^1} + \frac{1}{2}(1-L_2)\delta_{\mathbf{s},\mathbf{s}^4}$$

$L_1 = 1$ for any initial distribution that is normalized, and therefore is not of interest to determine the final distribution. In particular, the 4 pure states are mapped onto:

$$\begin{aligned} \mathbf{s}^1 : L_2 = 1 &\rightarrow p_\infty(\mathbf{s}) = \delta_{\mathbf{s},\mathbf{s}^1} \\ \mathbf{s}^{2,3} : L_2 = 0 &\rightarrow p_\infty(\mathbf{s}) = r_1(\mathbf{s}) \\ \mathbf{s}^4 : L_2 = -1 &\rightarrow p_\infty(\mathbf{s}) = \delta_{\mathbf{s},\mathbf{s}^4} \end{aligned}$$

Since there are two eigenvalues 1, there are two ergodic components, each consisting of one state ($\mathbf{s}^1$ and $\mathbf{s}^4$), and the remaining two states are transient.

For the same network with parallel dynamics, the eigenvalues are depicted in Fig. 14b. For small weights the network is again ergodic. The stationary distribution is given by Eq. 28 and is flat: independent of w and $\mathbf{s}$. For large weights ergodicity breaking occurs together with the occurence of a cycle of period 2 and two additional eigenvalues 1. Thus, there are three ergodic components. Two ergodic components are of period one and consist of one of the eigenvalues 1 (fixed points states $\mathbf{s}_1$ and $\mathbf{s}_4$: $T\mathbf{s}_{1,4} = \mathbf{s}_{1,4}$). The third ergodic component is of period 2 and consists of the eigenvalues 1 and -1 (a limit cycle of period 2 on states $\mathbf{s}_2$ and $\mathbf{s}_3$: $T^2\mathbf{s}^2 = T\mathbf{s}^3 = \mathbf{s}^2$).

In these two examples we have seen that all the eigenvalues of T are real. This is indeed in general true for both parallel and sequential dynamics when the weights are symmetric: $-1 \leq \lambda_\alpha \leq 1$. In addition, one can show for sequential dynamics (symmetric or asymmetric) that all eigenvalues are within the

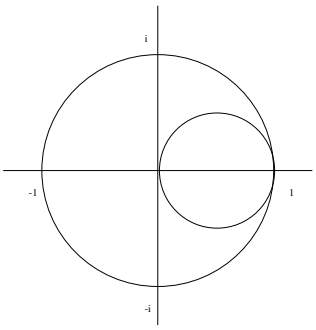


Figure 15: Eigenvalues of the transition matrix T are numbers in the complex plane with $|\lambda| \leq 1$. There is always at least one eigenvalue $\lambda = 1$. When the system is ergodic, there is only one eigenvalue 1 and the stationary distribution is unique and is reached for any initial state. When ergodicity is broken, there are more eigenvalues $\lambda = 1$ and multiple stationary distributions. The asymptotic behavior then depends on the initial state of the network. The state space is partitioned in ergodic components (plus a transition region). In addition, the ergodic components can have periodicity $d > 1$, leading to additional eigenvalues $\lambda = \exp(2\pi im/d), m = 1, \ldots, d$, all with $|\lambda| = 1$. For sequential dynamics (symmetric or asymmetric) all eigenvalues are within the circle centered at $\frac{1}{2} + 0i$ with radius $\frac{1}{2}$. Therefore, sequential dynamics has always periodicity 1. When weights are symmetric, the eigenvalues are real. Therefore, parallel dynamics with symmetric weights has at most periodicity 2. Parallel dynamics with asymmetric weights can have arbitrary periodicity.

circle centered at $\frac{1}{2} + 0i$ with radius $\frac{1}{2}$ [22]. The proof of this last statement again uses Gershgoren's Theorem and the special property of sequential dynamics that $T(F_i s|s) + T(s|F_i s) = \frac{1}{n}$. As a consequence, sequential dynamics has always periodicity 1 since other eigenvalues with $|\lambda| = 1$ are excluded. Note, that this property holds regardless of whether the network has symmetric or asymmetric connectivity. It also follows that for parallel dynamics with symmetric weights one can have at most periodicity 2 (because the eigenvalues are real). The spectrum of eigenvalues of T in the complex plane is sketched in fig. 15.

3.3 Summary

The behavior of a network of stochastic neurons can be dscribed as a first order Markov process. The Markov process is a prescription of how the probability distribution at time t over all the states of the network $s = (s_1, \ldots, s_n)$, $p_t(s)$, maps onto $p_{t+1}(s)$, and is given by the transition matrix $T(s'|s)$.

The transition matrix can be analysed in terms of its eigenvalues and eigenvectors. The right eigenvectors with eigenvalue 1 give the stationary distributions of T. When the Markov process is ergodic, there is a unique stationary distribution that is asymptotically reached from all initial states. Such a network has therefore no memory, because its stationary distribution contains no information about its past.

When the Markov process is non-ergodic, there are multiple stationary distributions. The asymptotic behavior of the network depends then on its initial state. Such a network can be used as a memory, where each memory corresponds to one of the ergodic components of the network. The attractor neural network that we discuss in section 5.3 is a concrete example. The topic of Markov processes, irreducibility and ergodicity is taken from [23, 24].

3.4 Exercises

1. (a) Compute the interspike interval distribution for the binary neuron as defined in Eq. 11.
   (b) Show that the distribution is normalized.
   (c) Discuss the similarities and differences between the binary neuron model and the Poisson process.

2. Consider a network of two neurons symmetrically connected by a synaptic weight $w = w_{12} = w_{21}$. Consider sequential Glauber dynamics as defined in Eqs. 14 and 15.
   (a) Write the transition matrix T in terms of the states s_1, s_2 and s'_1, s'_2.
   (b) Write T explicitly as a 4×4 matrix in the limit that $w \to \infty$. Show that there are three eigenvalues 1.
   (c) What are the invariants in this case?

3. Consider a network of two neurons symmetrically connected by a synaptic weight $w = w_{12} = w_{21}$. Consider parallel Glauber dynamics as defined in Eq. 12.

(a) Write the transition matrix T in terms of the states s_1, s_2 and s_1', s_2'.

(b) Write T explicitly as a 4×4 matrix in the limit that $w \to \infty$. Show that there are three eigenvalues 1.

(c) What are the invariants in this case?

4 Boltzmann-Gibbs distributions

If we consider a stochastic neural network with a specific connectivity matrix, what will the behavior of the network be? This is a rather difficult question to answer in general, but in some specific cases quite a lot is known. In particular for symmetrically connected networks with sequential dynamics, the equilibrium distribution is the Boltzmann-Gibbs distribution which plays a central role in statistical physics. In this section we derive the Boltzmann-Gibbs distribution. Then we indicate the computational problems associated with the computation of statistics of the Boltzmann-Gibbs distribution. We introduce the mean field theory as a simple approximation to compute the mean firing rates of the network and the linear response correction to approximately compute the correlations. We illustrate the use of these methods on Boltzmann Machines, which are Boltzmann-Gibbs distributions whose weights are thresholds are adapted through learning.

4.1 The stationary distribution

In the case that the synaptic connectivity is symmetric, $w_{ij} = w_{ji}$ one can compute the stationary probability distribution for the parallel and sequential dynamics explicitly. In both cases the derivation uses the argument of detailed balance, which states that for the dynamics $T(s'|s)$ there exists a function $p(s)$ such that

$$T(s|s')p(s') = T(s'|s)p(s) \quad \text{for all} \quad s, s'. \tag{22}$$

If detailed balance holds, it implies that $p(s)$ is a stationary distribution of T, which is easily verified by summing both sides of Eq. 22 over all states s' and using Eq. 16. However, the reverse is not true: many stochastic dynamics do not satisfy detailed balance and a solution to Eq. 19 is then typically not available in analytical form, although its existence is dictated by the Perron-Frobenius theorem [23].

For random sequential dynamics, T is given by Eqs. 14 and 11 and the detailed balance equation reads $T(F_i s|s)p(s) = T(s|F_i s)p(F_i s)$ for all states s and all neighbor states $F_i s$. It is easy to show that

$$\frac{T(s|F_i s)}{T(F_i s|s)} = \exp(2(\sum_j w_{ij} s_j + \theta_i)s_i). \tag{23}$$

Consider the distribution

$$p(s) = \frac{1}{Z} \exp(\frac{1}{2} \sum_{ij} w_{ij} s_i s_j + \sum_i \theta_i s_i). \tag{24}$$

$p(s)$ is called a Boltzmann-Gibbs distribution and plays a central role in statistical physics. For this reason, the expression in the exponent is often referred to as the energy:

$$-E(s) = \frac{1}{2}\sum_{ij} w_{ij}s_i s_j + \sum_i \theta_i s_i. \tag{25}$$

States of low energy have high probability. Z is a normalization constant,

$$Z = \sum_s \exp(-E(s)) \tag{26}$$

and is called the partition function. $p(s)$ only depends on the symmetric part of the weights w^s_{ij} and

$$\frac{p(s)}{p(F_i s)} = \exp(2(\sum_j w^s_{ij}s_j + \theta_i)s_i). \tag{27}$$

Thus for symmetric weights, detailed balance is satisfied between all neighboring states. Since all values of T are zero for non-neighboring states this proves that $p(s)$ is the equilibrium distribution. [3]

4.2 Computing statistics

$p(s)$ in Eq. 24 and 28 give an analytical expression of the stationary probability distribution of an arbitrary network with symmetric connectivity and sequential and parallel dynamics, respectively. From these equations we can compute any

[3] When all neurons are updated in parallel, the transition matrix is given by Eq. 12. As in the case of sequential dynamics, we can again compute the stationary distribution for symmetric weights. We use again detailed balance:

$$\frac{T(s'|s)}{T(s|s')} = \frac{\exp(\sum_{ij} w_{ij}s_j s'_i + \sum_i \theta_i s'_i)}{\exp(\sum_{ij} w_{ij}s'_j s_i + \sum_i \theta_i s_i)} \prod_i \frac{\cosh(h_i(s'))}{\cosh(h_i(s))}.$$

When the weights are symmetric, the term involving the double sum over i and and j cancels and the remainder is of the form $\frac{p(s')}{p(s)}$, with

$$p(s) = \frac{1}{Z}\exp(\sum_i \log\cosh(\sum_j w_{ij}s_j + \theta_i) + \sum_i \theta_i s_i). \tag{28}$$

This is the equilibrium distribution for parallel dynamics [21].

interesting *statistics*, such as for instance the mean firing rate of each of the neurons:

$$m_i = \langle s_i \rangle = \sum_s s_i p(s), \tag{29}$$

and correlations between neurons:

$$\chi_{ij} = \left\langle s_i s_j \right\rangle - \langle s_i \rangle \left\langle s_j \right\rangle = \sum_s s_i s_j p(s) - m_i m_j. \tag{30}$$

However, these computations are in general too time consuming due to the sum over all states, which involves 2^n terms.

For some distributions, the sum can be performed efficiently. For Boltzmann-Gibbs distributions, the subset of probability distributions for which the sum over states can be performed efficiently are called decimatable distributions [25]. These include factorized distributions, trees and some other special graphs as sub sets. For factorized distributions, $p(s) = \prod_i p_i(s_i)$, the energy only depends linearly on s_i and the sum over states can be performed by factorization:

$$\sum_s \exp(\sum_i \alpha_i s_i) = \prod_i \left(\sum_{s_i} \exp(\alpha_i s_i) \right) = \prod_i 2\cosh(\alpha_i).$$

From Eqs. 24 and 28 we infer that this corresponds to the rather uninteresting case of a network without synaptic connections. [4]

In general, the sum over states can not be computed in any simple way. In this case we call the the probability distribution *intractable* and one needs to apply approximation methods to compute the partition function and statistics such as Eq. 29 and 30.

[4] The probability distribution $p(s)$ is called a *tree* when between any two neurons in the network there exists only one path, where a path is a sequence of connections. Alternatively, one can order the neurons in the graph with labels $1, \ldots, n$ such that neuron i is connected to any number of neurons with higher label but only to at most one neuron with lower label. For Boltzmann-Gibbs distributions which are trees:

$$\sum_s \exp(\sum_{(ij)} w_{ij} s_i s_j) = \sum_s \exp(\sum_i w_{ip_i} s_i s_{p_i}) = \prod_i 2\cosh(w_{ip_i}),$$

where p_i labels the parent of neuron i. For parallel dynamics, such non-trivial decimatable structures do not exist.

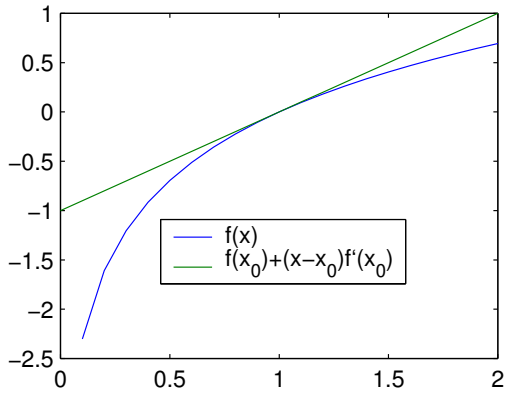


Figure 16: Illustration of the concavity property $f(x) \leq f(x_0) + (x - x_0)f'(x_0)$ for the logarithmic function and $x_0 = 1$.

4.3 Mean field theory

In this section, we will show how to approximate Z in Eq. 26 using the standard mean field theory. In fact this approximation is a lower bound on Z. As a by-product we will obtain an estimate of the mean firing rates of the neurons as well.

We can use Eq. 26 and write

$$\begin{aligned} \log Z &= \log \sum_s \exp(-E(s)) = \log \sum_s q(s) \frac{\exp(-E(s))}{q(s)} \\ &\geq \sum_s q(s) \log \left(\frac{\exp(-E(s))}{q(s)} \right) = -\langle E \rangle_q + S_q = -F \end{aligned} \tag{31}$$

$q(s)$ is an arbitrary positive probability distribution on the state space s. The inequality is called Jensen's inequality and follows from the concavity of the logarithmic function and the fact that $q(s)$ is a probability distribution: $\sum_s q(s) = 1$. For any concave function f, we have $f(x) \leq f(x_0) + (x - x_0)f'(x_0)$, as illustrated in fig. 16. Therefore, if we chose $x_0 = \langle x \rangle_q$, then $\langle f \rangle_q \leq f(\langle x \rangle_q)$. Further we have

$$\langle E \rangle_q = \sum_s q(s) E(s)$$

and

$$S_q = -\sum_s q(s) \log q(s)$$

is the entropy of the distribution $q(s)$. The bound F on $\log Z$ is called the *mean field free energy*.

Up to now, we have not specified $q(s)$, except that it must be a normalized probability distribution. We can in fact choose any probability distribution, but in order to make the mean field method tractable, we should choose a tractable distribution $q(s)$. The simples choice is to choose for $q(s)$ a factorized distribution where all neurons are independent:

$$q(s) = \prod_i q_i(s_i), \quad q_i(s_i) = \frac{1}{2}(1 + m_i s_i).$$

m_i is the expectation value of s_i under the distribution q_i: $m_i = \langle s_i \rangle_{q_i}$. The $m_i, i = 1, \ldots, n$ are undetermined parameters that specify the distribution q uniquely. F is now given by

$$F = -\frac{1}{2}\sum_{ij} w_{ij} m_i m_j - \sum_i \theta_i m_i + \frac{1}{2}\sum_i \left((1 + m_i)\log(\frac{1}{2}(1 + m_i)) + (1 - m_i)\log(\frac{1}{2}(1 - m_i)) \right). \tag{32}$$

From Eq. 31, we have $F \geq -\log Z$, for any choice of m_i. We get the tightest bound on $\log Z$ by minimizing F wrt m_i:

$$\frac{\partial F}{\partial m_i} = \sum_j w_{ij} m_j + \theta_i - \tanh^{-1}(m_i)$$

Setting $\frac{\partial F}{\partial m_i} = 0$ we obtain

$$m_i = \tanh(\sum_{j=1}^{n} w_{ij} m_j + \theta_i) \tag{33}$$

These equations are called the *mean field equations*. They consist of n non-linear equations with n unknown, which has te be solved selfconsistently. The solution m_i provides us with an approximation to the mean firing rates of the intractable Boltzmann distribution Eq. 29:

$$m_i = \langle s_i \rangle_q \approx \langle s_i \rangle_p$$

4.4 Linear response correction

We can also compute the correlations in the mean field approximation. The crucial observation is that both the mean firing rates and the correlations can be computed

as derivatives of the partition function:

$$\begin{aligned} \langle s_i \rangle &= \frac{\partial \log Z}{\partial \theta_i} \\ \chi_{ij} &= \frac{\partial^2 \log Z}{\partial \theta_i \partial \theta_j} \end{aligned}$$

with the correlations χ_{ij} defined in Eq. 30. Combining these two expressions, we can relate the correlations to the mean firing rates as

$$\chi_{ij} = \frac{\partial \langle s_i \rangle}{\partial \theta_j} \approx \frac{\partial m_i}{\partial \theta_j} \tag{34}$$

where in the last step we have used the mean field approximation for $\langle s_i \rangle$. Because the mean field equations give us an implicit relation between m_i and θ_j, we can derive

$$\frac{\partial \theta_i}{\partial m_j} = \frac{\delta_{ij}}{1 - m_i^2} - w_{ij}. \tag{35}$$

Thus the correlations can be computed by inverting this matrix. This approximation to the correlations is know as the *linear response correction*.

4.5 Boltzmann Machines

A well-known application of the Boltzmann-Gibbs distribution are Boltzmann Machines [9]. The basic idea is to treat the distribution Eq. 24 as a statistical model, and to use standard statistical tools to estimate its parameters w_{ij} and θ_i.

Let us restrict ourselves to the simplest case, that all neurons receive sensory input. The general case would be that only a subset of neurons (sensory neurons) receive input and the rest of the network (the hidden neurons) receive no direct input. The case with hidden neurons is somewhat more complex and is beyond the scope of these lectures.

Learning can be described in the following way. Consider a set of P *training patterns* $s^\mu = (s_1^\mu, \ldots, s_n^\mu)$ with $\mu = 1, \ldots, P$. We wish to find the value of the weights and thresholds, such that the Boltzmann-Gibbs distribution 'best' describes these data. The standard statistics approach to this problem is to construct the *log likelihood* of the observed data

$$L(w, \theta) = \frac{1}{P} \sum_\mu \log p(s_1^\mu, \ldots, s_n^\mu)$$

and maximize this function wrt to w and θ.

This maximization can be easily performed by computing the gradients of L wrt w_{ij} and θ_i [9, 5]:

$$\begin{aligned} \frac{\partial L}{\partial \theta_i} &= \Big(\langle s_i \rangle_c - \langle s_i \rangle \Big), \\ \frac{\partial L}{\partial w_{ij}} &= \Big(\Big\langle s_i s_j \Big\rangle_c - \Big\langle s_i s_j \Big\rangle \Big) \; i \neq j. \end{aligned} \tag{36}$$

The brackets $\langle \cdot \rangle$ and $\langle \cdot \rangle_c$ denote the 'free' and 'clamped' expectation values, respectively. The 'free' expectation values are defined as:

$$\langle s_i \rangle = \sum_s s_i p(s) \tag{37}$$

$$\Big\langle s_i s_j \Big\rangle = \sum_s s_i s_j p(s) \tag{38}$$

with $p(s)$ given by Eq. 24. The 'clamped' expectation values are simply the statistics computed in the training set:

$$\langle s_i \rangle_c = \frac{1}{P} \sum_\mu s_i^\mu \tag{39}$$

$$\Big\langle s_i s_j \Big\rangle_c = \frac{1}{P} \sum_\mu s_i^\mu s_j^\mu \tag{40}$$

The simplest learning procedure is to start at $t = 0$ with a random initial value of all weights and thresholds and to iteratively change these values in the direction of their gradients:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \eta \frac{\partial L}{\partial w_{ij}} \\ \theta_i(t+1) &= \theta_i(t) + \eta \frac{\partial L}{\partial \theta_i} \end{aligned}$$

with η a small number. This so-called gradient ascent algorithm increases the value of L at each step (for sufficiently small η) and terminates when the gradients are zero, i.e. at a local maximum. From Eq. 36 we see that at a local maximum of L, the first and second order statistics of the Boltzmann distribution p and the data are equal. It is good to be aware that there exist much more sophisticated

methods for maximizing a function, but the gradient ascent method is probability the closest to biology.

The computation of the free expectation values is intractable, because the sums in Eqs. 38 consist of 2^n terms. As a result, the exact version of the BM learning algorithm can not be applied to practical problems. We can however apply the mean field approximation as discussed in the previous section. Given the weights and thresholds at iteration t, we compute $\langle s_i \rangle$ from Eq. 33 and $\left\langle s_i s_j \right\rangle$ from Eqs. 34 and 35 and insert these values into the learning rule Eq. 36. This approach can also be applied when hidden units are present.

In the absence of hidden units we do not have to resort to an iterative learning procedure, but we can set the lhs of Eqs. 36 equal to zero and solve these equations directly. In the mean field approximation, these equations read:

$$m_i = \langle s_i \rangle_c \tag{41}$$

$$\chi_{ij} = \left\langle s_i s_j \right\rangle_c - m_i m_j, i \neq j. \tag{42}$$

m_i is a function of w_{ij} and θ_i as given by the mean field equations Eqs. 33. χ_{ij} is a function of w_{ij} and m_i as given by the linear response equations Eqs. 34 and 35. Eqs. 41 and 42 are $n + \frac{1}{2}n(n-1)$ equations with an equal number of unknowns w_{ij} and θ_i and can be solved using standard numerical routines.

The righthandside of Eq. 42 can be computed from the data, because of Eq. 41. Thus Eq. 42 is an matrix equation of the form

$$\chi = C$$

with $C_{ij} = \left\langle s_i s_j \right\rangle_c - \langle s_i \rangle_c \left\langle s_j \right\rangle_c$. If we invert this equation, we obtain

$$(C^{-1})_{ij} = (\chi^{-1})_{ij} = \frac{\delta_{ij}}{1 - m_i^2} - w_{ij}$$

where the last step is the result of Eqs 34 and 35. This gives an explicit solution for w_{ij} in terms of known quantities.

However, this procedure is incorrect, because Eq. 42 is only enforced off-diagnonally. By using the following trick we can however still use this approach. We introduce additional parameters, diagnonal weights w_{ii}, which we estimate in the learning process. Thus, in the mean field equations Eq. 33 the sum over j now also contains a term $w_{ii}m_i$. We now need n additional equations for learning, for which we propose the diagonal terms of Eq. 42: $\chi_{ii} = 1 - m_i^2$. This equation is

true by definition for the exact χ, but becomes an additional constraint on w_{ij} and θ_i when χ is the linear response approximation. Thus our basic equations become

$$m_i = \tanh(\sum_{j=1}^{n} w_{ij} m_j + \theta_i) \tag{43}$$

$$\chi_{ij}^{-1} = \frac{\partial \theta_j}{\partial m_i} = \frac{\delta_{ij}}{1 - m_i^2} - w_{ij}. \tag{44}$$

From Eq. 41-44 we can compute the solution for w_{ij} and θ_i in closed form:

$$m_i = \langle s_i \rangle_c \tag{45}$$

$$C_{ij} = \left\langle s_i s_j \right\rangle_c - \langle s_i \rangle_c \left\langle s_j \right\rangle_c \tag{46}$$

$$w_{ij} = \frac{\delta_{ij}}{1 - m_i^2} - \left(C^{-1}\right)_{ij} \tag{47}$$

$$\theta_i = \tanh^{-1}(m_i) - \sum_{j=1}^{n} w_{ij} m_j \tag{48}$$

4.5.1 Classification of digits

We demonstrate the quality of the above mean field approximation for Boltzmann Machine learning on a digit recognition problem. The data consists of 60000 training examples and 10000 test example of handwritten digits (0-9) compiled by the U.S. Postal Service Office of Advanced Technology. The examples are preprocessed to produce 28×28 binary images with noise added. See examples in fig. 17.

Our approach is to model each of the digits with a separate Boltzmann Machine. For each digit, we use approx. 6000 patterns for training using the approach outlined above. We thus obtain 10 Boltzmann distributions, each with its own parameters $W^\alpha = (w_{ij}^\alpha, \theta_i^\alpha), \alpha = 1, \ldots, 10$.

We then test the performance of these models on a classification task using 500 of the 10000 test patterns. We classify each pattern s to the model α with the highest probability:

$$\text{class}(s) = \text{argmax}_\alpha p_\alpha(s), \quad p_\alpha(s) = \frac{1}{Z(W_\alpha)} \exp(\frac{1}{2} \sum_{ij} w_{ij}^\alpha s_i s_j + \theta_i^\alpha s_i)$$

The normalization $Z(W^\alpha)$ is intractable and depends on α and therefore affects classification. We use its mean field approximation $\log Z \approx -F$, with F given by Eq. 32.

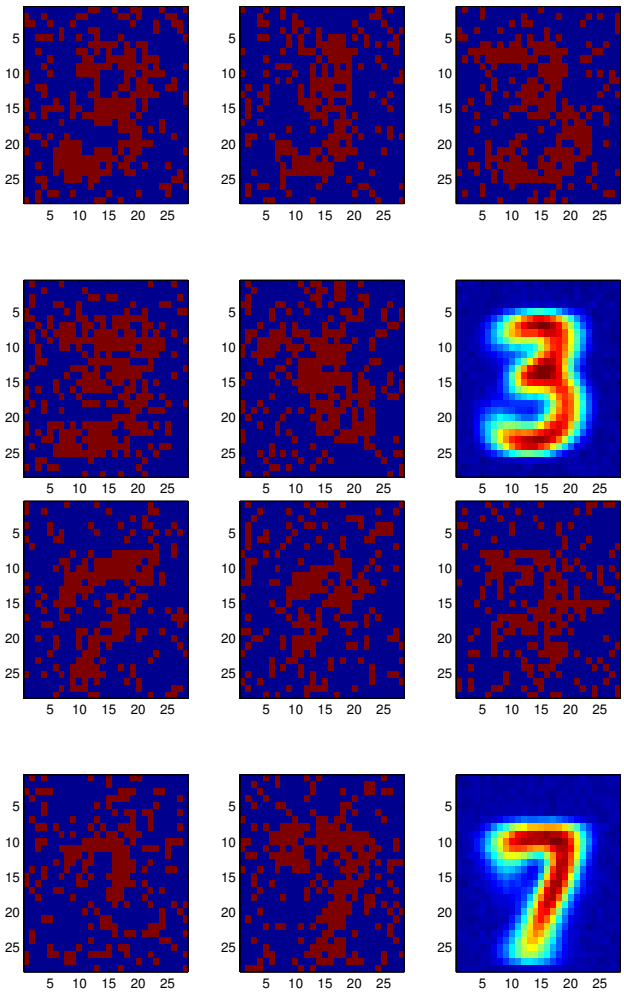


Figure 17: Sample of 60000 training patterns and 10000 test patterns of the 28 × 28 handwritten digits of the U.S. Postal Service Office of Advanced Technology. Patterns are binary and 10 % pixel noise is added.

Test the performance on 500 of the 10000 test patterns classifies 45 incorrect. Compare with simple template matching on the mean image yields 123 errors.

4.6 Summary

The Boltzmann-Gibbs distribution is the stationary distribution of the stochastic neural network, when using sequential dynamics and symmetric weights. The symmetric weights is a severe restriction, and is clearly not true for the synaptic connectivity in the brain. However, if we view the binary neurons as describing the average behavior of groups of neurons, as is customary in the connectionists approach, symmetric connectivity is not so bad. It is often observed that the time-delayed correlation between neurons shows a peak at delay zero, indication that the 'effective' connectivity is symmetric.

For non-symmetric networks the theoretical analysis is much harder and fewer results are known. Most of the results have been obtained with numerical simulations. It appears that when a sufficient amount of asymmetry is introduced, the network dynamics is dominated by periodic orbits of different length. Thus asymmetric networks are radically different from symmetric networks. The differences between symmetric and asymmetric networks are discussed in [24].

Despite the fact that the stationary distribution can be given in a closed form mathematical formula, the computation of any statistics of this distribution, such as means and correlations is intractable. The mean field method, as introduced in this chapter, allows to compute these quantities approximately. There exist more powerful approximate methods. For a discussion of the extensions of mean field theory see [26] and [27], as well as other contributions to that book.

Because the computation of means and correlations in stochastic is intractable, also any learning method is intractable. The Boltzmann Machine learning paradigm is the simplest learning method for stochastic neural networks.

4.7 Exercises

1. (a) Derive Eq. 23.

   (b) Show that the detailed balance does not hold when the weights of the neural network are not symmetric ($w_{ij} \neq w_{ji}$). In other words, show that te Boltzmann distribution is not the stationary distribution of the Glauber dynamics with asymmetric weights.

2. Study the accuracy of the mean field and linear response method for a Boltzmann distribution on 2 neurons with equal threshold $\theta_1 = \theta_2 = \theta$ and connected by a weigth w:

$$p(s_1, s_2) = \frac{1}{Z} \exp(w s_1 s_2 + \theta(s_1 + s_2))$$

   (a) Give an expression for the mean field equations to approximately compute the firing rates for this network.

   (b) Solve the mean field equations numerically for $\theta = w$ and various values of w and compare the mean field approximation with the exact result.

   (c) Compute the linear response estimate of the correlations and compare with the exact values.

3. Work out analytically the result of mean field learning with linear response correction for the case of two neurons and a data set consisting of three patterns $(1, -1), (1, 1), (-1, -1)$.

4. Take home computer exercise. The objective is to 1) make you familiar with the mean field approximation and the linear response correction and 2) to numerically compare the accuracy of a Monte Carlo sampling method with the mean field method.

   - Write a program that can compute means and correlations in an Ising model of n binary neurons using a Metropolis Hastings (MH) method. Use $s_i = \pm 1$ coding. Choose the coupling matrix 1) with random positive entries (ferromagnetic case) and 2) with random entries of either sign (frustrated case). Choose the thresholds $\theta_i = 0$.
   - Write a program that can compute means and correlations in the same Ising model using the mean field theory and linear response correction.
   - We will rely on the results of the MH method to be an good approximation of the exact result. To demonstrate the reliability of the MH method, show that the results of different MH runs with different initializations are identical within the errors of individual runs. Note, how the required length of the MH run depends on the n, on the size of the weights and on whether the weights are ferromagnetic or frustrated.

- Compare the quality of of the mean field approximation of the means and correlations as a function of n, size of the weights for the ferromagnetic and frustrated case.

Provide plots and texts for all your results and conclusions.

5 Attractor neural networks

5.1 Hebbian learning

Hebbian learning is the mechanism that neurons increase their connection strength when both pre- and post-synaptic neuron are active at the same time. The mechanism can be illustrated with the so-called *McCulloch-Pitts neuron*, which is given in fig. 18. It should be noted that the McCulloch-Pitts neuron is nothing else but

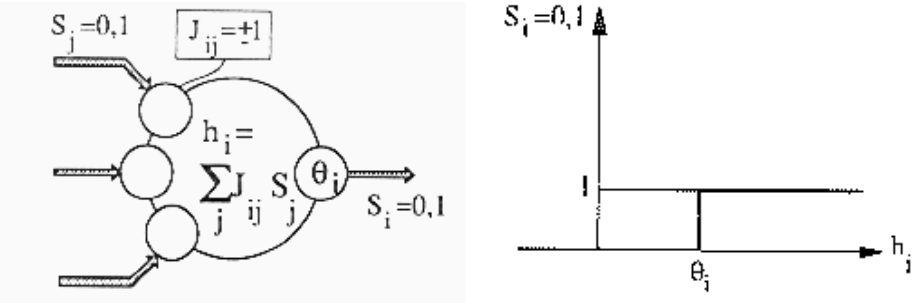


Figure 18: The McCulloch-Pitts neuron model. When the total input activity exceeds a threshold the output of the neuron is 1, otherwise it is 0

.

the noiseless limit of the binary neuron model of Eq. 11. If we scale the local field in Eq. 11 with a parameter β, and consider the limit of $\beta \to \infty$, then

$$\sigma(\beta h_i(y)) \to \Theta(h_i(y))$$

where $\Theta(x)$ is the Heavyside step function as depicted in fig. 18Right.

With the McCulloch-Pitts neuron, it is easy to build logical circuits. For example, consider a neuron with two inputs y_1 and y_2, both with synaptic strength $w_i = 1$. The output of the neuron is

$$y = \Theta(y_1 + y_2 - \theta)$$

If we choose $\theta = 3/2$, y compute the logical AND of y_1 and y_2. If we choose $\theta = 1/2$, y compute the logical OR of y_1 and y_2. With these basic functions (plus a few others such as the inverter), one can build any logical function of some binary variables. The insight that neurons can compute any function can be used

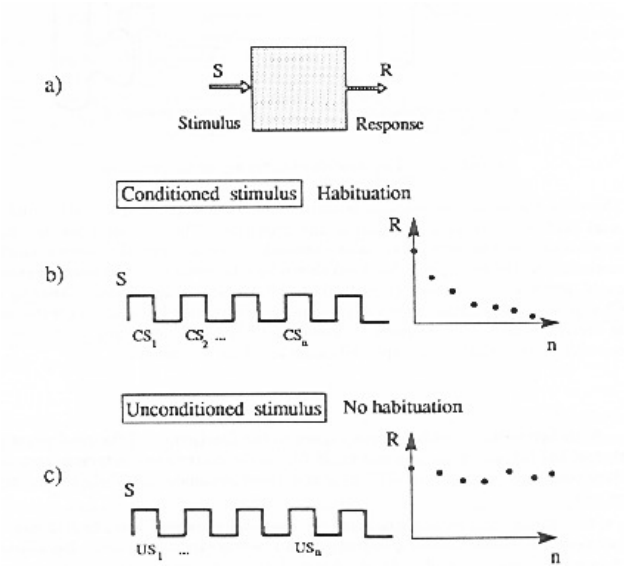


Figure 19: a) In psychology, the behavior of the subject is described in terms of stimulus and response. b) Habituation. Repeated presentation of the same stimulus gives a response that decreases with time. c) No habituation. Repeated presentation of the same stimulus gives the same response, regardless of the number of stimulations.

to show that the brain could in fact implement a Turing machine. However, we have seen that the McCulloch-Pitts neuron model is rather simple compared to the full complexity of real neurons. Therefore, the idea that the brain is a universal computer should not be taken too literaly.

The McCulloch-Pitts neuron can by nicely used to illustrate the concept of Hebbian learning which has its origin in psychology. In psychology, the behavior of 'subjects' (people or animals) is described in terms of stimulus and response (fig. 19a). The response to a stimulus may change upon repeated presentation. For instance, if you ring a bell, a dog will turn its head. But if you repeat this many times, the dog will get bored and no longer pay attention to the bell (fig. 19b). Other stimuli do not show any habituation. For instance, if you show a dog a piece of meat, the dog will salivate and will continue doing so on any future presentation of meat (fig. 19c).

Now, we turn to a little more complex experiment known as *classical conditioning*. The dog is presented with two stimuli, a nice piece of red meat and the

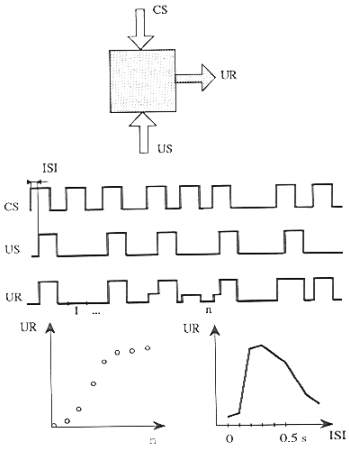


Figure 20: Classical condition. A stimulus (CS) initially does not evoke a response (UR). After repeated presentation in conjunction with another stimulus (US) that by itself does evoke a response, CS by itself will also evoke a response.

bell (fig. 20top). Upon presentation of only meat, the dog salivates. Upon presentation of only the bell, the dog does not salivate. When both stimuli are presented together, the dog will salivate because he sees the meat. After several trials, however, the dog will also start to salivate when only the bell is rung (because he is expecting the meat as well!). The meat is called the *conditioned stimulus* (CS) and the bell is called the *unconditioned stimulus* (US). The salivating response is called the *unconditioned response*. A typical trial of CS and US through time is shown in fig. 20middle and the unconditioned response to the conditioned stimulus as a function of trial number is shown in fig. 20bottom left. Fig. 20bottom right shows that the effect of classical conditioning is maximal when the CS slightly precedes the US.

Classical conditioning as well as habituation, are thought to the be result of *learning*. Learning means in broad terms that the subject adapts itself such that its response after some time is different from its initial response. Classical conditioning can be easily understood in terms of the McCulloch-Pitts neurons. Clearly, in the brain there are no individual neurons that encode 'meat', 'bell' and 'salivate'. Instead, in the psychological modeling literature it is customary to use neurons and their inputs to represent 'functional regions' in the brain (this approach to modeling is called connectionism).

The McCulloch-Pitts neuron is given by

$$y = \Theta(w_u y_u + w_c y_c - \theta)$$

where $y_{u,c} = 0, 1$ denote the activity of the unconditioned and conditioned stimuli and $y = 0, 1$ denotes the activity of the salivate neuron. $w_{u,c}$ are the effective connections between $y_{u,c}$ and y and are assumed to be subject to learning. Initially, we have $w_c < \theta$ and $w_u > \theta$ to ensure that the dog does not respond to the bell, but does respond to the meat. After classical conditioning the dog responds to either bell or meat, and thus $w_c > \theta$ and $w_u > \theta$.

The fact that w_c is increased, *only* when CS and US are both presented and not when CS is presented alone has given rise to the *Hebbian learning rule* which states that the change in the connections between neuron j and i is proportional to the product of pre- and post-synaptic neuron acitivity:

$$\Delta w_{ij} \propto y_j y_i \tag{49}$$

With $j = u, c$ and and i the salivate neuron, the stimulus CS by itself is not able to increase the connection w_c because it does not evoke a response ($y_i = 0$). When both CS and US are present, $y_c = y_i = 1$, and the connection w_c is increased. The rationale is that when a presynaptic spike contributes to the firing of the post synaptic neuron, it is likely that its contribution is of some functional importance to the animal and therefore the efficacy of the responsible synapse should be increased.

Hebbian learning is not only manifest in psychological experiments. Also at the single neuron level the Hebb rule is to a certain extend observed in several brain areas (see fig. 21) and as is also clearly seen in the experiment shown in fig. 2. However, the learning rule as given by Eq. 49 is in general considered to be too simple. In particular, synapses display an interesting short time dynamics as is evident from fig. 2 and display both strenghtening (LTP) and weakening (LTD) depending on the relative timing of the pre- and postsynaptic activity.

5.2 Short-term and long-term memory

Many mental abilities require to hold information temporarily in an accessible form. An obvious proposal to implement short-term memory on a physiological level would be to hold the corresponding neural activity over a certain duration of time. An illustrative example is shown in fig. 22. A monkey was trained to

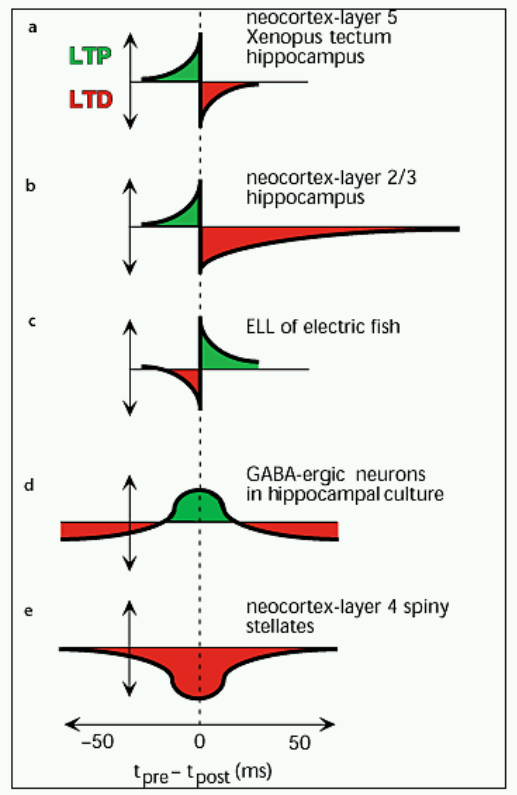


Figure 21: The amount and type of synaptic modification evoked by repeated pairing of pre-and postysnaptic action potentials in different preparations. The horizontal axis is the difference between the times of these spikes. Pairing can result in increased synaptic strength (LTP) or decreased synaptic strength (LTD).

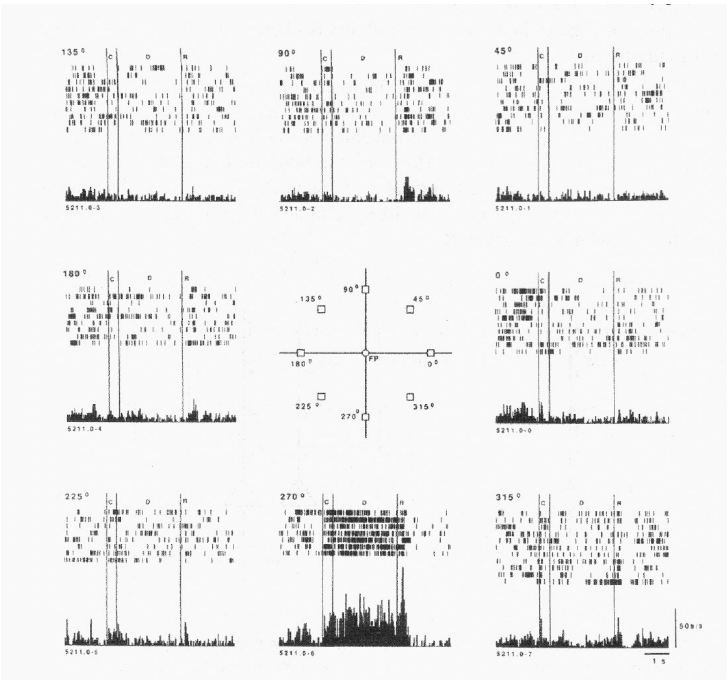


Figure 22: Short-term memory. Maintenance of neural activity in the absence of a stimulus shows evidence of recurrent dynamics in neural networks [28].

maintain its eyes on a central fixed spot until a 'go' signal, such as a tone, indicated that it should move the eyes and to focus on one of several possible targets peripheral to the fixed spot. The choice of the target to which the eye should be moved in each trial was indicated by a short flash, but the subject was not allowed to move its eyes from the time of disappearance of the target cue until the 'go' signal. Thus, the target location for each trail had to be remembered during the delay period. The experimenters recorded from neurons in the dorsolateral prefrontal cortex (area 46) and found neurons that were active during the delay period.

An area in the medial temporal lobe is called the *hippocampus* for its shape resembles a seahorse (see fig. 23). The hippocampus has been associated with the a form of long-term memory called *episodic memory*. This type of ememory refers to the storage of events, where each single event can be a vast collection of associations such as the encouter with a particular object at a certain location at a acertain time. Within the hippocampus there is an area called CA3 that has been assumed to act as a recurrent network and to perform recall and pattern completion

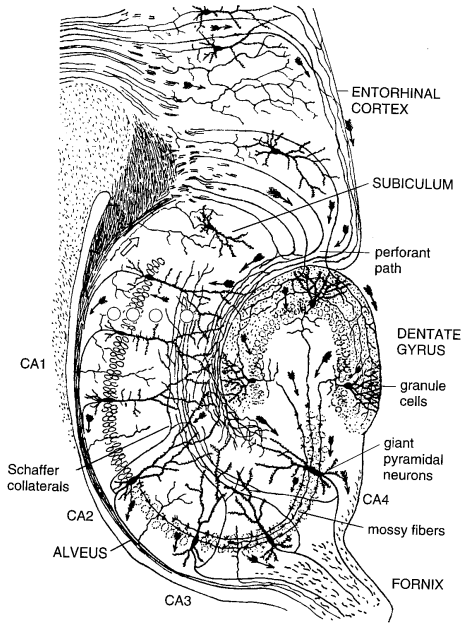


Figure 23: Neuronal elements of the hippocampal formation in rodents as drawn by Ramon y Cajal around 1900. The hippocampus has been associated with the a form of long-term memory called episodic memory.

tasks.

The hippocampus has been implicated in the acquisition of episodic long-term memory because it has been shown that a patient in whom is was necessary to remove this structure suffered subsequently from a form of amnesia marked by the inability to form new long-term memories of episodic events. In contrast, long-term memory that was acquired before the removal of this structure as well as the ability to learn new motor skills was not impaired. The precise involvement of the hippocampus in memory acquisition is still under debate, and the neural mechanisms of human episodic memory abilities are mostly unclear. So far, it seems that the hippocampus can store some forms of memory for a considerable length of time until the content is stored permanently in other cortical areas.

A possible mechanism for Hebbian learning in the CA3 area of the hippocampus is provided by *mossy fibres* from the dentate granule cells in the hippocampus onto CA3 neurons. These fibres could provide the necessary signal to indicate when learning should take place. Interestingly, mossy fibres have the largest synapses found in the mamalian brain. Thus, it is possible that signals along mossy fibres command the firing of specific CA3 neurons at the right time to enable the formation of new associations through Hebbian learning.

5.3 Attractor neural networks

The attractor neural network is an example of how recurrent connections in a neural network can make useful computation. The attractor neural network stores a number of patterns as *attractors* of the neuron dynamics by adjusting the connections between the neurons. The behavior of the attractor neural network is illustrated in fig. 24left. When the network is initialized in a certain state, the network evolves to the nearest attractor (fig. 24right).

The idea to use recurrent dynamics as a form of computation in neural networks goes back to the work of [29]. Its formulation for binary neurons was first proposed by [21] and the analysis of the behavior of the network in which the connections are the result of Hebbian learning was done by [6], and later extended in a series of papers by Amit, Gutfreund and Sompolinsky [10, 11].

Consider a network of McCulloch-Pitts neurons. The dynamics of the network is given by

$$s_i(t+1) = \text{sign}(\sum_j w_{ij} s_j(t) + \theta_i) \tag{50}$$

A state of the network $\xi = (\xi_1, \ldots, \xi_n)$ is called a *fixed point* if it is unaltered under

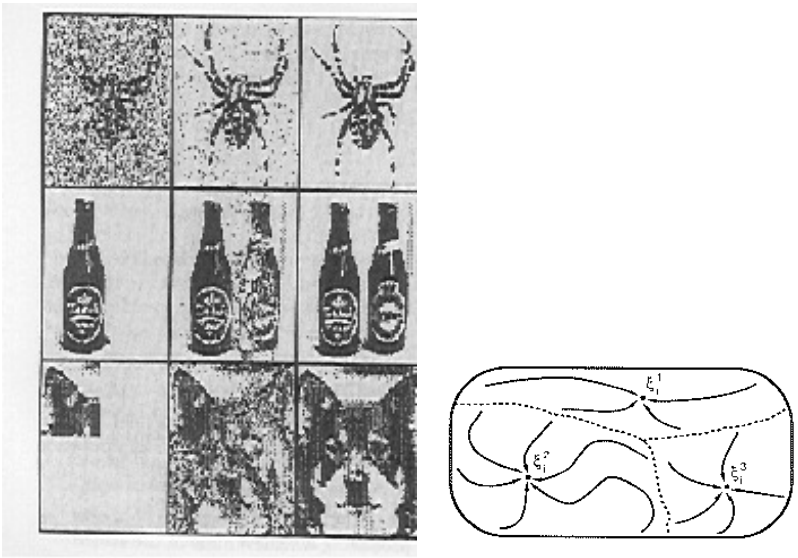


Figure 24: Left) Behavior of an attractor neural network of binary neurons. Three stored patterns are shown (right column), which can be retrieved by initializing the network in a noisy version of the patttern (top row), or part of the pattern (bottom two rows). Right) State space of the network is divided in basins of attraction for each of the stored patterns.

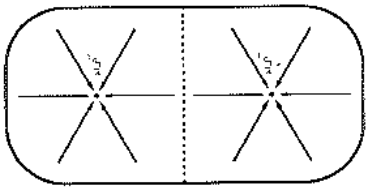


Figure 25: When a pattern ξ is stored in an attractor neural network its anti-pattern $-\xi$ is also a fixed point of the dynamics.

the dynamics eq. 50. Given a set of *patterns* $\xi_i^\mu, \mu = 1, \ldots, P$, i.e. states of the network that we wish to store, the learning task is to find the connections w_{ij} such that these patterns are stable fixed points of the dynamics.

Consider first the simple case of storing one pattern ξ. Memorisation requires stability of ξ:

$$\text{sign}(\sum_j w_{ij}\xi_j) = \xi_i$$

A suitable weight matrix is given by the outer product of the pattern with itself:

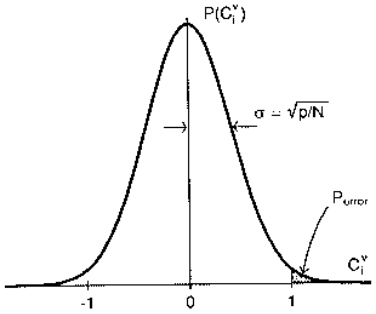
$$w_{ij} = \frac{\beta}{n}\xi_i\xi_j, \tag{51}$$

where the factor β/n is irrelevant at this moment (because of the sign operation), but will be useful shortly. The threshold $\theta_i = 0$. If we start in an arbitrary state s and update all neurons once, we have $s_i(t = 1) = \xi_i \text{sign}(\sum_j \xi_j s_j(t = 0))$. Thus, the state of the network after one time step is $\pm\xi$. Both ξ and $-\xi$ are fixed points of the dynamics. The sign depends on the overlap of s with the pattern ξ at $t = 0$ (see fig. 25).

When we want to store many patterns, we generalize Eq. 51 to

$$w_{ij} = \frac{\beta}{n}\sum_\mu \xi_i^\mu \xi_j^\mu \tag{52}$$

Suppose that we initialize the network in one of its patterns, say pattern ξ^ν and ask ourselves whether the state can be a fixed point of the dynamics. The local field is



P_{error}	p/n
0.001	0.105
0.0036	0.138
0.01	0.185
0.05	0.37
0.1	0.61

Figure 26: When P random patterns are stored, the probability that pattern ν is a fixed point decreases with P, due to destructive interference with other patterns $\mu \neq \nu$. The probability of error per neuron is given by $P_{\text{error}} = \text{Prob}\{C_i > 1\}$ and is a function of P/n.

given by

$$\begin{aligned} h_i^\nu &= \sum_j w_{ij}\xi_j^\nu = \frac{\beta}{n}\sum_{j,\mu}\xi_i^\mu\xi_j^\mu\xi_j^\nu \\ &= \beta\left(\xi_i^\nu + \frac{1}{n}\sum_{j,\mu\neq\nu}\xi_i^\mu\xi_j^\mu\xi_j^\nu\right) = \beta\xi_i^\nu(1 - C_i) \end{aligned}$$

with $C_i = -\frac{\xi_i^\nu}{n}\sum_{j,\mu\neq\nu}\xi_i^\mu\xi_j^\mu\xi_j^\nu$. Thus, the local field consists of the sum of two terms. One is the pattern ξ^ν that the network is in (called the signal). The other is a term that consists of a sum over all other patterns and all neurons (the noise). When all the bits of all the patterns are chosen randomly and independently $\xi_i^\mu = \pm 1$, C_i is the sum of a number of independent random contributions. Due to the law of large numbers, C_i becomes Gaussian distributed with mean zero and variance $\sigma = \sqrt{p/n}$. Thus we see that the signal contribution will dominate the noise contribution if the number of patterns is small compared to the number of neurons. In that case sign(h_i) will be sufficiently similar to ξ_i^ν to ensure that $\xi^n u$ is a fixed point of the dynamics.

Because C_i is Gaussian distributed, we can easily compute the probability P_{error} that sign$(h_i) \neq \xi_i^\nu$. P_{error} measures the initial stability: when we start in the fixed point a fraction of P_{error} bits will be wrong after one step. In subsequent steps, the dynamics may still return to ξ^ν or errors may accumulate with time. As we will show later, correct retrieval occurs as long as $p/n < 0.138$.

Learning rule eq. 52 is similar to the Hebbian learning rule, discussed in section 5.1 in the sense that it depends on both pre and postsynaptic activity and if we imagine that μ labels time. However, there are also some important differences. First of all, the sign of the weight can be either positive or negative whereas real synaptic connections normally cannot change from excitatory to inhibitory as a result of learning. Secondly, the connectivity is symmetric and does not obey the temporal and spatial asymmetry of fig. 21[5]. Thirdly, the rule 51 prescribes a strenghening of the connection when both pre- and postsynaptic neuron are inactive, which is not biologically plausible. Dispite these shortcomings, the model is important because it illustrates the strengths and weaknesses of recurrent computation.

5.3.1 The role of noise

As we have seen before, neuron firing is noisy. What happens to the attractor neural network when we consider noisy firing, i.e. when we replace the McCulloch-Pitts neuron by its stochastic version Eq. 11?

We have seen in section 4 that for symmetric connections the probability to observe the network in a state $p(s)$ is given by the Boltzmann-Gibbs distribution. Fortunately, the Hebb rule Eq. 52 is symmetric and we can therefore directly apply the results from that section.

Using Eqs. 24 and 52, the equilibrium distribution is given by:

$$p(s) = \frac{1}{Z}\exp(-\beta E(s)), \quad E(s) = -\frac{1}{2n}\sum_{\mu}(\xi^{\mu}\cdot s)^2 \tag{53}$$

with $\xi^{\mu}\cdot s = \sum_i \xi_i^{\mu} s_i$. Thus, the network dynamics is characterized by an energy function that has local minima at each of the p patterns ξ^{μ} (see fig. 27).

To understand the effect of noise on the network, consider the case of one pattern ($p = 1$). The energy is given by

$$E(s) = -\frac{1}{2n}\sum_{ij}\xi_i\xi_j s_i s_j = -\frac{1}{2n}\sum_{ij}\eta_i\eta_j$$

where we have introduced the new binary variable $\eta_i = \xi_i s_i = \pm 1$. Thus in terms of η, we have a fully connected network where each neuron is connected to all other neurons with connections strength β. This network is known as a ferro-magnet where the neuron states ± 1 correspond to spin up or down (see fig. 28). For

[5]Note that for the case of fig. 21a if firing of neuron i precedes firing of neuron j, w_{ji} will increase and at the same time w_{ij} will decrease. Thus w_{ij} and w_{ji} are unlikely to be identical.

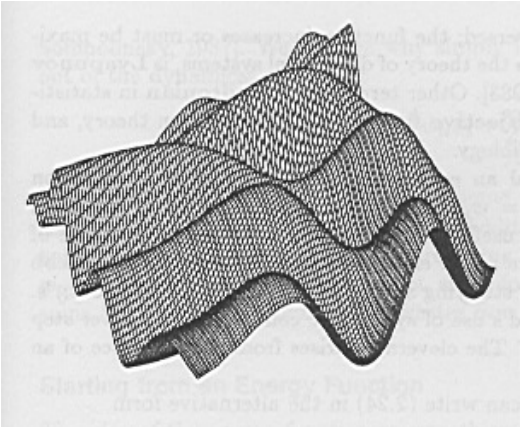


Figure 27: The network dynamics is characterized by an energy function that has local minima at each of the p patterns ξ^μ.

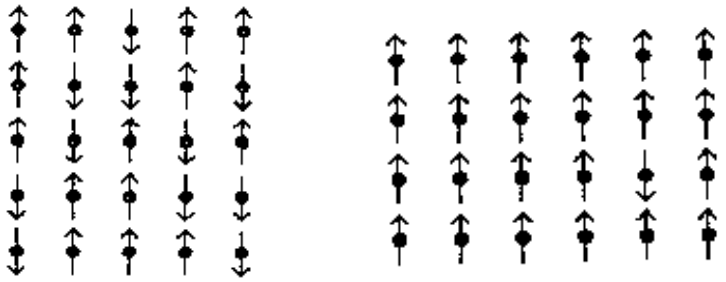


Figure 28: Attractor network with one pattern is equivalent to a (fully connected) ferro-magnet. For small coupling (small β), neurons behave independently and the mean firing rate is zero. For strong coupling, neurons 'allign' to all $+1$ or all -1.

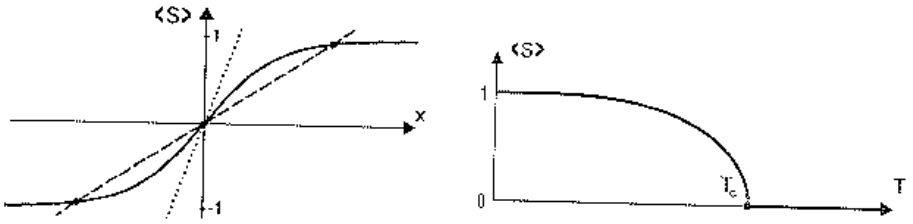


Figure 29: Left) Graphical solution of $m = \tanh(\beta m)$. For small β, only $m = 0$ is a solution. For $\beta > 1$ the equation has two stable solutions with $m \neq 0$ and the solution $m = 0$ is unstable. Right) Solution m versus 'temperature' $T = \frac{1}{\beta}$. For small T only one of the two solutions is shown.

small coupling (small β), neurons behave independently and the mean firing rate is zero. For strong coupling, neurons 'allign' to all $+1$ or all -1. This behavior can be easily understood from the mean field equations that we derived in section 4.3. We apply Eq. 33 with $w_{ij} = \frac{\beta}{n}$ and obtain

$$m = \tanh(\beta m), \quad m = \frac{1}{n} \sum_j m_j$$

For $\beta < 1$, this equation has one stable solution $m = 0$. For $\beta > 1$, this equation has two stable solution $m \neq 0$ and the solution $m = 0$ is unstable. The solutions are graphically illustrated in fig. 29. Since

$$m = \frac{1}{n} \sum_i \xi_i \langle s_i \rangle$$

it measures the inner product of the average network state with the stored pattern ξ. We see that for small noise (small T) the memory is retrieved and for large noise the memory is lost. These two phases are known as the ferromagnetic and paramagnetic phase, respectively.

For large P, Eq. 53 will not only have local minima at each of the $2P$ patterns (patterns and anti-patterns), but in addition two other types of local minima are introduced: the *(odd) mixture states* and the *spin-glass states*. The odd mixture states are linear combinations of an odd number of memories. For instance, if ξ^μ, ξ^ν and ξ^ρ are three memories, a 3-mixture is any of the eight possibilities $\xi_i^3 =$

sign($\pm\xi_i^\mu \pm \xi_i^\nu \pm \xi_i^\rho$). The odd mixtures have higher energies and are therefore meta stable at finite temperature. The odd mixtures have a finite overlap with the memories. The spin-glass states are other local minima of the energy that have a vanishing overlap with any of the memories when $n \to \infty$.

We have seen that patterns can be stored as fixed points in the attractor neural network as long as 1) the number of patterns is not too large and 2) the noise is not too large. One can analyze the stability of the fixed points for any value of β and $\alpha = p/n$. The result is given in fig. 30. In region A and B, memories are stable fixed points. In A they are global minima. In B they are only local minima: there are spin-glass states with lower energy than the memories. In C, the memories are unstable, but there are stable spin-glass states. In D the noise is so large that all memories are lost and only the state $m_i = 0$ is stable.

5.4 Summary

We have seen that Hebbian learning is widespread throughout the brain and is also observed at the functional level in the form of classical conditioning. The topic of classical conditioning and its relation to Hebbian learning is taken from [22]. It is unlikely the only form of plasticity in the brain. Other forms are fast accomodation of the eye to changing light conditions changing neuro-transmitter concentrations that may affect the global behavior of groups of neurons.

Our discussion on short-term and long-term memory is taken from [30]. We showed an example of short term memory: a delayed reward task that requires recurrent computation to maintain the activity of the specific neurons. Although this indicates that recurrent computation is present in the brain, it is unclear how adaptation of synaptic connections operate to achieve this phenomenon.

It is commonly suggested that long-term memory is the result of permanent changes in the synaptic connectivity between neurons, such as observed physiologically as LTP and LTD. Associative memory, where retrieval occurs when triggered by part of the content of the memory, can then be implemented as an attractor neural network, where each of the attractors represent one memory. We have discussed an implementation of this idea due to John Hopfield, the attractor neural network [5]. It should be stated, however, that there is at present not much direct experimental evidence of attractor neural networks in the brain.

5.5 Exercises

1. (a) Show that C_i^ν in section 5.3 has mean zero and variance $\sqrt{p/N}$.

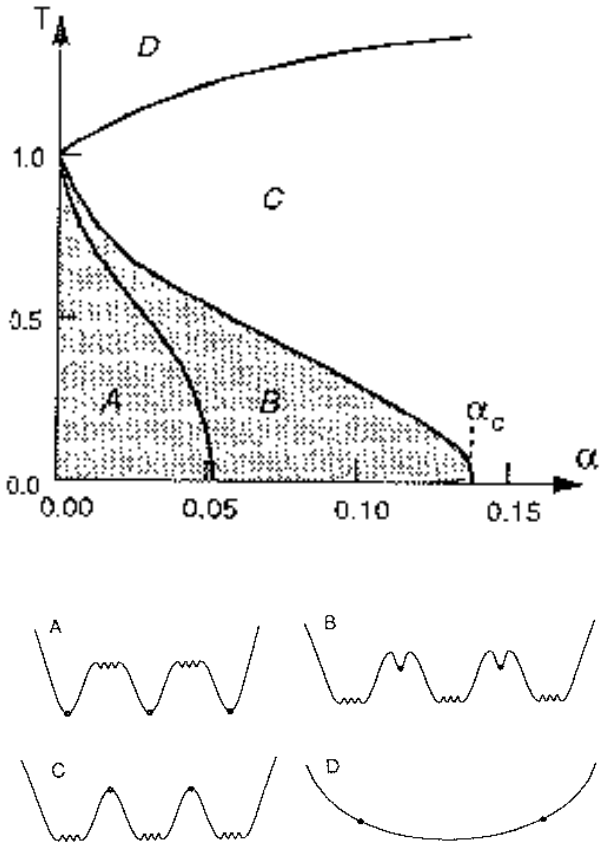


Figure 30: Storage capacity of the attractor neural network for various values of $\alpha = \frac{p}{n}$ and $\beta = \frac{1}{T}$.

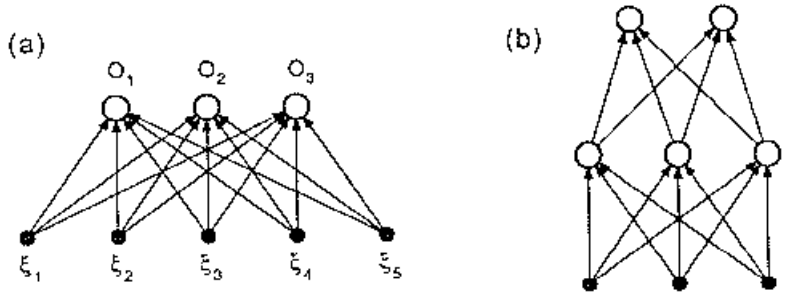


Figure 31: A) Simple Perceptron B) Multi-layered Perceptron

(b) Compute P_{error} in fig. 26.

2. (a) Show that the energy function $E(s)$ in Eq. 53 decreases (or remains constant) with each step of the sequential dynamics Eq. 50.

   (b) Use this fact to show that the dynamics Eq. 50 converges to a fixed point.

3. (a) Compute the energy of pure memories.

   (b) Compute the energy of 3-mixture states (Hint: first compute the inner product of the mixture with one of the pure states and ignore the inner product with all other patterns) and thus show that the energy of 3-mixtures is higher than of pure memories.

6 Perceptrons

Perceptrons are feed-forward neural networks. Examples are given in Fig. 31. Consider a simple perceptron with one output:

$$o = g(h) = g\left(\sum_{j=1}^{n} w_j \xi_j - \theta\right) = g\left(\sum_{j=0}^{n} w_j \xi_j\right)$$

with weights w_j and inputs ξ_j. $\xi_0 = -1$ and $\theta = w_0$. g is a non-linear function.

Learning: Given a number of input-output pairs $(\xi_j^\mu, \zeta^\mu), \mu = 1, \ldots, P$, find w_j such that the perceptron output o for each input pattern ξ^μ is equal to the desired

output ζ^μ:

$$o^\mu = g\left(\sum_{j=0}^{n} w_j \xi_j^\mu\right) = \zeta^\mu, \quad \mu = 1, \ldots, P$$

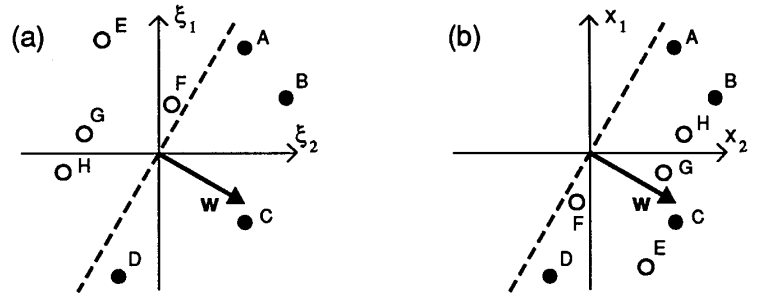
6.1 Threshold units

Consider the simplest case of binary threshold neurons:

$$g(h) = \text{sign}(h)$$

Then, the learning condition becomes

$$\text{sign}(w \cdot \xi^\mu) = \zeta^\mu, \quad \mu = 1, \ldots, P$$



Since $\zeta^\mu = \pm 1$, we have

$$\text{sign}(w \cdot \xi^\mu \zeta^\mu) = 1 \quad \text{or} \quad w \cdot x^\mu > 0$$

with $x_j^\mu = \xi_j^\mu \zeta^\mu$.

6.2 Linear separation

Classification depends on sign of $w \cdot \xi$. Thus, decision boundary is hyper plane:

$$0 = w \cdot \xi = \sum_{j=1}^{n} w_j \xi_j - \theta$$

Perceptron can solve linearly separable problems. An example of a linearly separable problem is the AND problem: The output of the perceptron is 1 if all inputs are 1, and -1 otherwise (see Fig. 32).

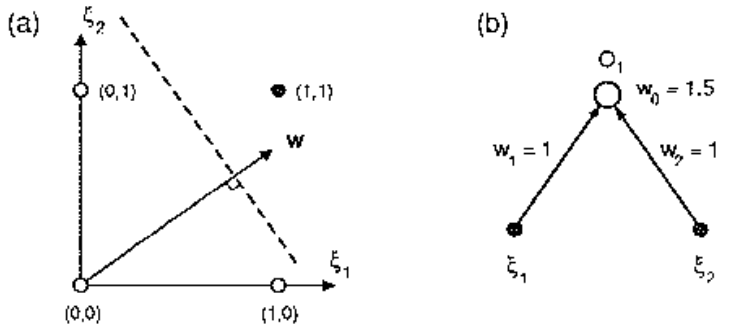
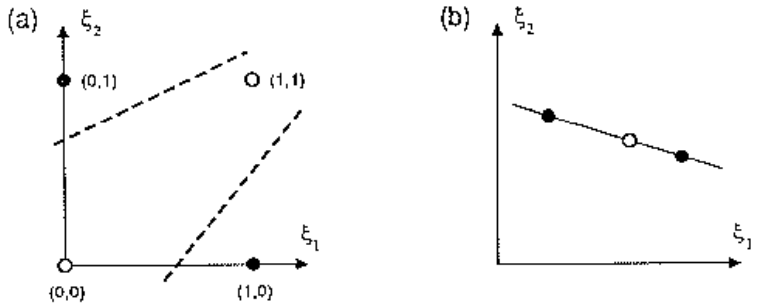


Figure 32: The AND problem for two inputs is linearly separable.

By definition, problems that are not linearly separable need more than one separating hyper plane to separate the two classes. An example of a non-linearly separable problem is the XOR problem: The output is equal to the product of the input values (see Fig. 32A). Other problems that are not linearly separable occur when three or more input patterns are linearly dependent (see Fig. 32B).



6.3 Perceptron learning rule

We have seen that the desired weight vector satisfies

$$w \cdot x^{\mu} > 0, \quad \text{all patterns} \tag{54}$$

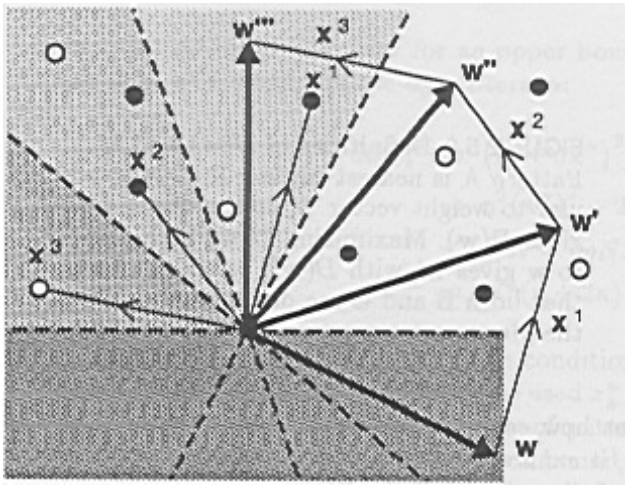


Figure 33: The perceptron learning rule in action. Learning rule Eq. 55 is applied to all patterns in some random or given order. Learning stops, when a weight configuration is found that has positive inner product with all training patterns.

We define the following perceptron learning rule:

$$\begin{aligned} w_j^{\text{new}} &= w_j^{\text{old}} + \Delta w_j \\ \Delta w_j &= \eta \Theta(-w \cdot x^\mu) \xi_j^\mu \zeta^\mu = \eta \Theta(-w \cdot x^\mu) x^\mu \end{aligned} \tag{55}$$

η is the learning rate. This learning rule is Hebbian in the sense that the change in weight is proportional to the product of input and output activity. The function Θ is 1 for positive arguments and zero otherwise: When presenting pattern μ, learning only occurs, when the condition $w \cdot x^\mu > 0$ is not satisfied for that pattern.

In Fig. 33 we show the behavior of the perceptron learning rule with $\eta = 1$. The dataset consists of three data patterns x^1, x^2 and x^3. The initial weight vector is w. Presenting pattern x^1, we note that $w \cdot x^1 < 0$ and therefore learning occurs. The resulting weight vector is $w' = w + x^1$. Presenting pattern x^2 and x^3 also result in learning steps and we end up in weight configuration w'''. This weight vector has positive inner product with all training patterns and learning terminates.

Depending on the data, there may be many or few solutions to the learning problem, or non at all! In Fig. 34 we give examples of two data sets and their solutions Eq. 54. In Fig. 34A there are more admissible weight vectors and they can have a larger inner product with all training patterns than in Fig. 34B. We define

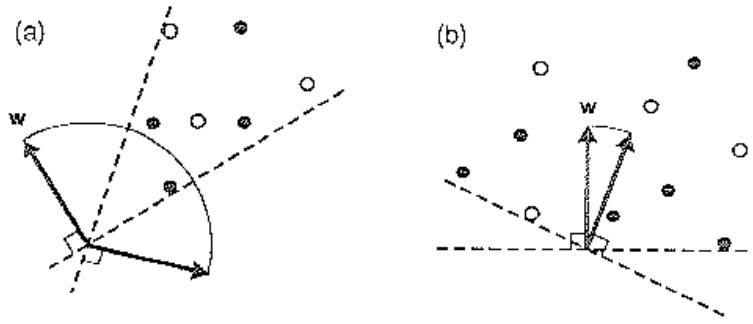


Figure 34: Two examples of data sets and the sets of w that satisfy condition Eq. 54. A) Many solutions B) Few solutions.

the quality of the solution w by the pattern that has the smallest inner product with w. Since the solution does not depend on the norm of w, we define the quality as

$$D(w) = \frac{1}{||w||} \min_{\mu} w \cdot x^{\mu}$$

The best solution is given by $D_{\max} = \max_w D(w)$.

In Fig. 35, we illustrate this for a given data set and two admissible solutions w and w' and their values of D respectively. Since $D(w') > D(w)$, w' is the preferred solution.

If we can find a w such that $D(w) > 0$ the problem is linearly separable and learnable by the perceptron learning rule. If the problem is not linearly separable not such solution exists.

6.3.1 Convergence of Perceptron rule

In this section we show that if the problem is linearly separable, the perceptron learning rule converges in a finite number of steps. We start with initial value $w = 0$. At each iteration, w is updated only if $w \cdot x^{\mu} < 0$. After some number of iterations, let M^{μ} denote the number of times pattern μ has been used to update w. Thus,

$$w = \eta \sum_{\mu} M^{\mu} x^{\mu}$$

$M = \sum_{\mu} M^{\mu}$ is the total number of iterations in which the weight vector is updated. If the learning rule converges, it means that M is finite and does not

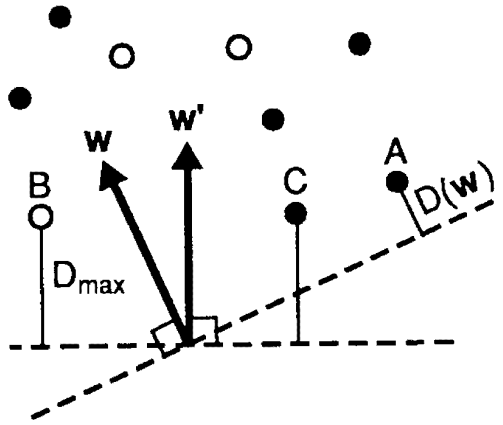


Figure 35: Two admissible solutions w and w' and their values of D respectively. Since $D(w') > D(w)$, w' is the preferred solution.

grow indefinitely.

The proof goes as follows. Assume that the problem is linearly separable, so that there is a solution w^* with $D(w^*) > 0$. We will show that

$$O(\sqrt{M}) \leq \frac{w \cdot w^*}{||w||||w^*||} \leq 1$$

where the second inequality follows simply from the definition of the inner product, and we will show the first inequality below. Thus, M can not grow indefinitely and the perceptron learning rule converges in a finite number of steps.

The proof of the first inequality is elementary:

$$\begin{aligned} w \cdot w^* &= \eta \sum_{\mu} M^{\mu} x^{\mu} \cdot w^* \geq \eta M \min_{\mu} x^{\mu} \cdot w^* = \eta M D(w^*) ||w^*|| \\ \Delta ||w||^2 &= ||w + \eta x^{\mu}||^2 - ||w||^2 = 2\eta w \cdot x^{\mu} + \eta^2 ||x^{\mu}||^2 \leq \eta^2 ||x^{\mu}||^2 = \eta^2 N \end{aligned}$$

The inequality in the second line makes use of the fact that for each training pattern where learning takes place $w \cdot x^{\mu} < 0$. The norm of w is thus bounded by

$$||w||^2 \leq \eta^2 N M$$

Combining these two inequality, we obtain Thus,

$$\frac{w \cdot w^*}{|w||w^*|} \geq \sqrt{M} \frac{D(w^*)}{\sqrt{N}} \tag{56}$$

which completes the proof. Note, that the proof makes essential use of the existence of w^* with $D(w^*) > 0$. If $D(w^*) < 0$ the bound Eq. 56 becomes a trivial statement and does not yield a bound on M.

If the problem is linearly separable, we can in conclude that the number of weight updates:

$$M \leq \frac{N}{D^2(w^*)}$$

where N is some trivial constant. We see that convergence takes longer for harder problems (for which $D(w^*)$ is closer to zero).

6.4 Linear units

We now turn to a possibly simpler case of linear units:

$$o^\mu = \sum_j w_j \xi_j^\mu$$

Desired behavior is that the perceptron output equals the desired output for all patterns: $o^\mu = \zeta^\mu, \mu = 1, \ldots, P$. In this case, we can compute an explicit solution for the weights. It is given by

$$w_j = \frac{1}{N} \sum_{\rho\nu} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^\nu, \quad Q_{\rho\nu} = \frac{1}{N} \sum_j \xi_j^\rho \xi_j^\nu \tag{57}$$

Q is a matrix with dimension $P \times P$ and contains the inner products between the input patterns.

To verify that Eq. 57 solves the linear perceptron problem, we simply check for one of the input patterns (ξ^μ) whether it gives the desired output:

$$\begin{aligned} \sum_j w_j \xi_j^\mu &= \frac{1}{N} \sum_{\rho,u,j} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^u \xi_j^\mu \\ &= \sum_{\rho,u} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} Q_{\nu\mu} \\ &= \sum_\rho \zeta^\rho \delta_{\rho\mu} = \zeta^\mu \end{aligned}$$

For this solution to exist, Q must be invertible which means that Q must be of maximal rank (rank P). Therefore $P \leq N$. [6]

When $P < N$ the solution $w_j = \frac{1}{N} \sum_{\rho\nu} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^u$ is not unique. In fact, there exists a linear space of dimension $N - P$ of solutions w. Namely, let

$$\begin{aligned} w_j^0 &= \frac{1}{N} \sum_{\rho\nu} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^u \\ w_j &= w_j^0 + \xi^\perp \end{aligned}$$

with $\xi^\perp$ an n-dimensional vector that is perpendicular to all training patterns: $\xi^\perp \perp \{\xi^\mu\}$. Then the output of the perceptron is unaffected by $\xi^\perp$:

$$\zeta^\mu = \sum_j w_j \xi_j^\mu = \sum_j (w_j^0 + \xi_j^\perp)\xi_j^\mu = \sum_j w_j^0 \xi_j^\mu$$

6.4.1 Gradient descent learning

Often $P > N$, and thus patterns are linearly dependent. In general, one can define a learning rules through a cost function, that assigns a cost or quality to each possible weight vector. A common cost function is the quadratic cost:

$$E(w) = \frac{1}{2} \sum_\mu \left(\zeta^\mu - \sum_j w_j \xi_j^\mu \right)^2$$

which is minimized when the actual perceptron output $\sum_j w_j \xi_j^\mu$ is as close as possible to the desired output ζ^μ for all patterns μ.

[6] In addition, the input patterns must be linearly independent. If the input patterns are linearly dependent, solution Eq. 57 does not exist unless the corresponding outputs are also linearly dependent. Linear dependence of the inputs implies that there exists α^μ such that

$$\sum_\mu \alpha^\mu \xi_j^\mu = 0$$

This implies that

$$\sum_\mu \alpha^\mu \zeta^\mu = \sum_{\mu j} w_j \alpha^\mu \xi_j^\mu = 0$$

in other words, that the outputs cannot be chosen at freely. For problems with linearly dependent inputs and matched linearly dependent output Eq. 57 can be used by restricting the training set to a linearly independent subset that spans the training set, and computing Q for this subset.

The cost function can be minimized by the so-called gradient descent procedure. We start with an initial random value of the weight vector w and we compute the gradient in this point:

$$\frac{\partial E}{\partial w_i} = -\sum_\mu \left(\zeta^\mu - \sum_j w_j \xi_j^\mu \right) \xi_i^\mu$$

We change w according to the 'learning rule'

$$w_i = w_i + \Delta w_i \qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{58}$$

and repeat this until the weights do not change any more.

When η is sufficiently small, it is easy to verify that this gradient descent procedure converges. The proof consists of two observations. One is that for small η, $E(w)$ decreases in each step, and the other is that $E(w)$ is bounded from below, so that it has a smallest value. Therefore E cannot continue decreasing indefinitely and must converge to some stationary value (see Exercises).

6.4.2 The value of η

What is a good value form η? Clearly, when η is very small, convergence is guaranteed, but in practice it may take a very long time. If η is too large, however, convergence is no longer guaranteed. The problem is further complicated by the fact that the optimal choice of η is different for different components of the weight vector w. This is illustrated in Fig. 36, where E as a function of w is drawn. This valley has a unique minimal value for E, but the curvature in two directions is very different. In the long (flat) direction, large steps can be made, but in the orthogonal direction only small steps are allowed. We can analyze the problem, by assuming that the energy has the form

$$E(w) \quad = \quad \frac{1}{2} \sum_i a_i \left(w_i - w_i^* \right)^2 + E_0$$

with w^* the location of the minimum, and a_i the curvatures in the two directions $i = 1, 2$. Eq. 58 becomes

$$\Delta w_i \quad = \quad -\eta \frac{\partial E}{\partial w_i} = -2\eta a_i \left(w_i - w_i^* \right) = -2\eta a_i \delta w_i$$

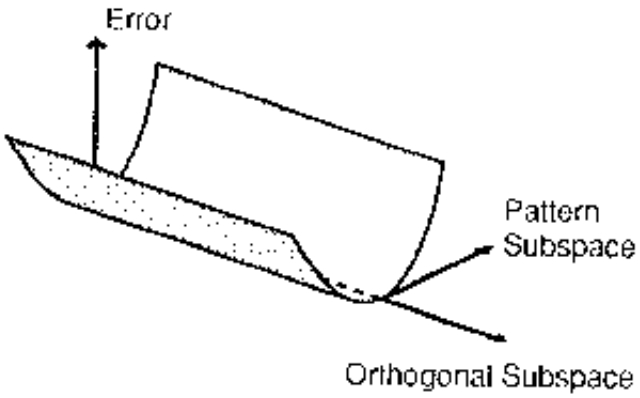


Figure 36: Cost landscape $E(w)$ with different curvatures in different directions.

with $\delta w_i = w_i - w_u^*$. The effect of learning step on δw_i is

$$\delta w_i^{\text{new}} = w_i^{\text{new}} - w_i^* = w_i^{\text{old}} - 2\eta a_i \delta w_i^{\text{old}} - w_i^* = (1 - 2\eta a_i)\delta w_i^{\text{old}}$$

thus, δw_i converges asymptotically to zero iff

$$|1 - 2\eta a_i| < 1. \tag{59}$$

We must find an η that satisfies Eq. 59 for all i. When $1 - 2\eta a_i < 0$, δw_i changes sign in each iteration. The behavior is illustrated in Fig. 37 with $E(w_1, w_2) = w_1^2 + 20w_2^2$ for different values of η.

6.5 Non-linear units

We can extend the gradient descent learning rule to the case that the neuron has a non-linear output:

$$o^\mu = g(h^\mu), \quad h^\mu = \sum_j w_j \xi_j^\mu$$

We use again the quadratic cost criterion:

$$E_1(w) = \frac{1}{2} \sum_\mu (\zeta^\mu - o^\mu)^2$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \sum_\mu (\zeta^\mu - o^\mu)\, g'(h^\mu) \xi_i^\mu$$

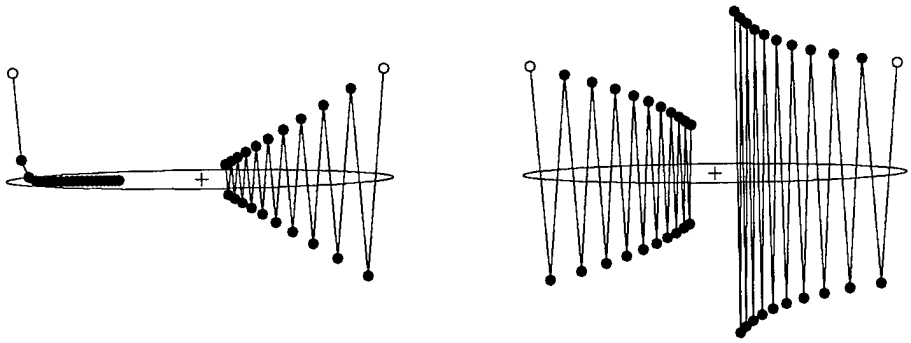


Figure 37: Behavior of the gradient descent learning rule Eq. 58 for the quadratic cost function $E(w_1, w_2) = w_1^2 + 20w_2^2$ for $\eta = 0.02, 0.0476, 0.049, 0.0505$.

When the function g is a monotonous function, it is invertible and one could also formulate a different cost criterion by observing the identity

$$\begin{aligned} \zeta^\mu &= g(h^\mu) \Leftrightarrow g^{-1}(\zeta^\mu) = h^\mu \\ E_2(w) &= \frac{1}{2} \sum_\mu \left(g^{-1}(\zeta^\mu) - h^\mu \right)^2 \end{aligned}$$

Note, that E_2 has a quadratic dependence on w, as in the linear case (but with transformed targets $g^{-1}(\zeta^\mu)$ instead of ζ^μ). In general, optimizing either E_1 or E_2 yield different optimal solutions.

6.6 Stochastic neurons

For $o = \pm 1$:

$$p(o|\xi) = \frac{1}{2} \left(1 + \tanh(ho)\right), \quad h = \sum_j w_j \xi_j$$

When the target distribution is given by $q(o|\xi)$, we use the (conditional) KL divergence as cost function for learning:

$$E = \sum_\mu \sum_{o=\pm 1} q(o|\xi^\mu) \log \left(\frac{q(o|\xi^\mu)}{p(o|\xi^\mu)} \right)$$

For a specific training set, we have $q(\zeta^\mu|\xi^\mu) = 1$ and $q(-\zeta^\mu|\xi^\mu) = 0$. Thus,

$$E = -\sum_\mu \log p(\zeta^\mu|\xi^\mu)$$

Gradient descent on this learning rule yields

$$\begin{aligned}
\frac{\partial p(\zeta^\mu|\xi^\mu)}{\partial w_j} &= 2p(\zeta^\mu|\xi^\mu)p(-\zeta^\mu|\xi^\mu)\zeta^\mu\xi^\mu_j \\
\frac{\partial E}{\partial w_j} &= -2\sum_\mu p(-\zeta^\mu|\xi^\mu)\zeta^\mu\xi^\mu_j \\
&= -\sum_\mu (\zeta^\mu - \langle o\rangle_\mu)\xi^\mu_j
\end{aligned}$$

This is equivalent with (5.57-58).

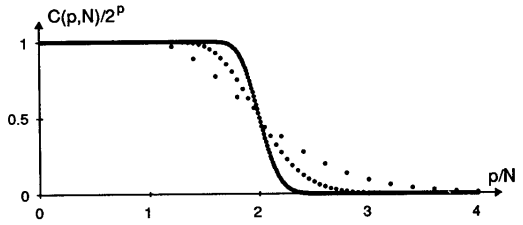
6.7 Capacity of the Perceptron

How many patterns can be perfectly mapped by a perceptron:

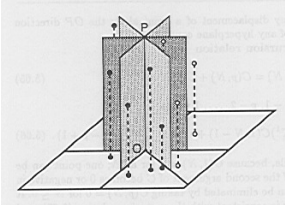
- Linear perceptron $P_{\max} = N$
- Binary perceptron $P_{\max} = 2N$

Consider P patterns in N dimensions. Each pattern can be either class (B/W). How many of the 2^P colorings are linearly separable?

- P small, then $C = 2^P$
- P large, then $C << 2^P$



Proof by induction. Define $C(P, N)$ the number of linearly separable colorings on P points in N dimensions.



Add one point X. The set $C(P, N)$ consists of

- colorings with separator through X (A)
- rest (B)

Thus,

$$\begin{aligned} C(P+1,N) &= 2A + B = C(P,N) + A \\ &= C(P,N) + C(P,N-1) \end{aligned}$$

Yields

$$C(P,N) = 2\sum_{i=0}^{N-1} \left(\begin{array}{c} P-1 \\ i \end{array} \right)$$

6.8 Multi-layered perceptrons

The gradient descent learning procedure can be trivially extended to the perceptron with multiple layers and multiple outputs as shown in Fig. 31B. In addition to the input variables ξ_k and the output variable o_i, we have a layer of hidden variables v_j for which no training data are observed. The value of the hidden variables is computed in terms of the input variables, and the outputs are computed in terms of the hidden variables:

$$o_i = g\left(\sum_j w_{ij} v_j\right) = g\left(\sum_j w_{ij} g\left(\sum_k w_{jk}\xi_k\right)\right) \tag{60}$$

The output is now a complex function of the input pattern ξ_k and the weights w_{jk} in the first layer of the network and the weights w_{ij} in the second layer of the network.

Given a set of P training patterns $(\xi_k^\mu, \zeta_i^\mu), \mu = 1, \ldots, P$, we again use the gradient descent procedure to find the weights that minimize the total quadratic error:

$$E(w) = \frac{1}{2} \sum_i \sum_\mu \left(o_i^\mu - \zeta_i^\mu \right)^2 \tag{61}$$

with o_i^μ the output on node i for input pattern ξ^μ as given by Eq. 60.

For large neural networks with many hidden units, the simple gradient descent procedure can be quite slow. However, there exist well-known algorithms that significantly accelerate the convergence of the gradient descent procedure. One such method is the conjugate gradient method. Treatment of this method is beyond the scope of this course (see however [5] or Matlab for further details).

Note, that the optimal solution that is found depends on the number of hidden units in the network. The more hidden units, the more complex functions between input and output can be learned. So, for a given data set, we can make the error Eq. 61 as small as we like by increasing the number of hidden units. In fact, one can show that the multi-layered perceptron can learn any smooth function, given a sufficiently large number of hidden units.

However, the objective of a learning algorithm is to use the neural network to predict the output on novel data, that were not previously seen. Increasing the number of hidden units does not necessarily improve the prediction on novel data. The situation is illustrated in Fig. 38 for the case of one input variable and one output variable. The crosses denote the data points that were used for training and the smooth curve is the neural network solution. For a small number of hidden units, the solution may look something like Fig. 38A. The solution does not pass through all the data points. For a larger number of hidden units, the solution may look something like Fig. 38B. The solution does pass through all the data points and is more complex. However, the prediction of the more complex network is less accurate than the simple network for the data point indicated by the circle, which was not part of the training set. The extend to which the trained neural network is capable of predicting on novel data is called the generalization performance. The network with the optimal generalization performance must balance two opposing criteria: minimization of the error on the training data requires a large number of hidden units, but the solution should also be sufficiently smooth to give good prediction.

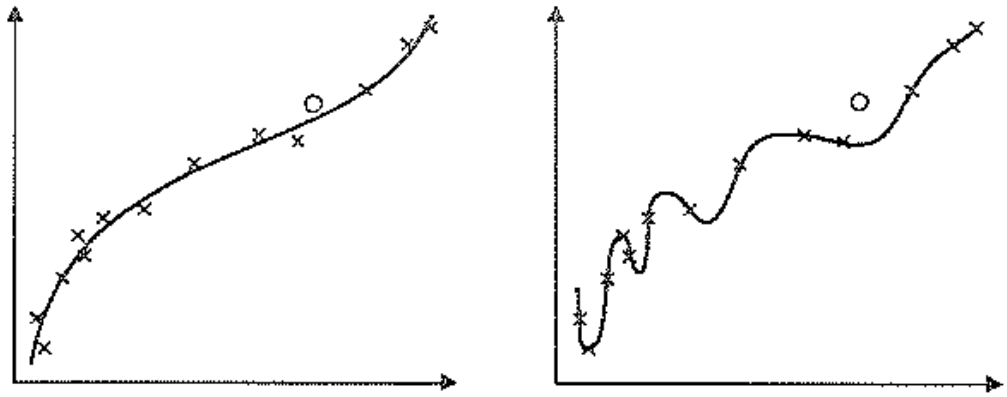


Figure 38: Network output versus network input. A) Network with a small number of hidden units. B) Network with a large number of hidden units. Networks with more hidden units can implement more complex functions and can better fit a given training set. However, more complex networks do not necessarily generalize better on novel data.

6.9 Summary

This chapter is based on [5]. Perceptrons are simple models of feed-forward computation in a network of neurons. Binary perceptrons can be used for classification problems. Learning is done using the perceptron learning rule. The learning rule converges in a finite number of iterations if and only if the problem is linearly separable.

Perceptrons can also be constructed with continuous output, either using a linear or non-linear transfer function. These perceptrons can be learned using the gradient descent method. Gradient descent converges asymptotically for any data set.

The quality of the perceptron can be significantly improved by using multiple layers of hidden units. The multi-layered perceptron can learn any function by using a sufficiently large number of hidden units. However, prediction quality on novel data does not generally increase with the number of hidden units. Optimal generalization is obtained for a finite number of hidden units.

6.10 Exercises

1. Check dat $D_{max} = \frac{1}{\sqrt{3}}$ voor het AND probleem en $D_{max} = -\frac{1}{\sqrt{3}}$ voor het XOR probleem. Het AND probleem in de $\xi_i = \pm 1$ codering is gedefinieerd als $\zeta = 1$ als $\xi_1 = \xi_2 = 1$ and $\zeta = -1$ in alle overige gevallen. Het XOR probleem is gedefinieerd als $\zeta = \xi_1 * \xi_2$. Gebruik voor de gewichten vector $w = (w_0, w_1, w_2)$. (Hint: gebruik $w_1 = w_2$ vanwege symmetrie).

2. Beschouw gradient descent in een kostenlandschap gegeven door $E = a_1 x^2 + a_2 y^2$. Bereken de leerparameter η zodanig dat de convergentie in zowel x als y richting even snel is.

3. Beschouw een lineair perceptron (sectie 6.4) om de AND functie te leren.

   - wat zijn de optimale gewichten en drempels? wat is de optimale kosten E?
   - laat zien dat $E > 0$ impliceert dat de inputpatronen lineair afhankelijk zijn.

4. Toon aan dat het gradient descent algoritme Eq. 58 asymptotisch convergeert.

References

[1] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical biophysics*, 5:114–133, 1943.

[2] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[3] S. Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16:299–307, 1967.

[4] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[5] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computation*, volume 1 of *Santa Fe Institute*. Addison-Wesley, Redwood City, 1991.

[6] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings Nat. Acad. Sci. USA*, 79:2554–2558, 1982.

[7] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[8] H. Sompolinsky and I. Kanter. Temporal association in asymmetric neural networks. *Physical Review Letters*, 57:2861–2864, 1986.

[9] D. Ackley, G. Hinton, and T. Sejnowski. A learning algorithm for Boltzmann Machines. *Cognitive Science*, 9:147–169, 1985.

[10] D.J. Amit, H. Gutfreund, and H. Sompolinsky. Spin-glass models of neural networks. *Physical Review A*, 32:1007, 1985.

[11] D.J. Amit, H. Gutfreund, and H. Sompolinsky. Storing infinite numbers of patterns in a spin glass model of neural networks. *Physical Review Letters*, 55:1530–1533, 1985.

[12] A. Mason, A. Nicoll, and K. Stratford. Synaptic transmission between individual pyramidal neurons of the rat visual cortex in vitro. *Journal of Neuroscience*, 11:72–84, 1990.

[13] B. Katz. *Nerve, muscle and synapse*. McGraw-Hill, 1966.

[14] Chr. Koch. *Biophysics of computation*. Oxford University Press, 1999.

[15] L.F. Abbott, J.A. Varela, K. Sen, and S.B. Nelson. Synaptic depression and cortical gain control. *Science*, pages 220–224, 1997.

[16] H. Markram and M. Tsodyks. Redistribution of synaptic efficacy between enocortical pyramidal neurons. *Nature*, pages 807–810, 1996.

[17] W. Rall and J. Rinzel. Branch input resistance and steady attenuation for input to one branch of a dendritic neuron model. *Biophysics Journal*, pages 648–688, 1973.

[18] W.R. Softky and Chr. Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsps. *Journal of Neuroscience*, 13:334–350, 1993.

[19] G.L. Gerstein and B. Mandelbrot. Random walk models for the spike activity of a single neuron. *The Biophysical Journal*, 4:41–68, 1964.

[20] H.E. Plesser and W. Gerstner. Noise in itegrate-and-fire neurons, from stochastic input to escape rates. *Neural Computation*, 12:367–384, 2000.

[21] W.A. Little. The existence of persistent states in the brain. *Math. Biosci.*, 19:101–120, 1974.

[22] P. Peretto. *An introduction to the modeling of neural networks*. Cambridge University Press, 1992.

[23] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes*. Clarendon Press, Oxford, 1992.

[24] H.J. Kappen. An introduction to stochastic neural networks. In F. Moss and S. Gielen, editors, *Neuro-Informatics and Neural Modelling*, pages 517–552. North-Holland, 2001.

[25] L. Saul and M.I. Jordan. Learning in Boltzmann trees. *Neural Computation*, 6:1174–1184, 1994.

[26] H.J. Kappen. An introduction to stochastic neural networks. In F. Moss and S. Gielen, editors, *Neuro-informatics and neural modelling*, volume 4 of *Handbook of biological physics*, pages 517–552. Elsevier, 2001.

[27] H.J. Kappen and W. Wiegerinck. Mean field theory for graphical models. In D. Saad and M. Opper, editors, *Advanced mean field theory*, pages 37–49. MIT Press, 2001.

[28] S. Funahashi, C.J. Bruce, and P.S. Goldman-Rakic. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *Journal of physiology*, 61, 1989.

[29] D.J. Wilshaw, O.P. Buneman, and H.C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–962, 1969.

[30] T.P. Trappenberg. *Fundamentals of computational neruoscience*. Oxford University Press, 2002.