Machine Learning Handouts

Bert Kappen,
Biophysics University of Nijmegen

December 14, 2015

Contents

1 Networks of binary neurons

1.1 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. **??**Left 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. **??**Left 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. 1, we show that the curves of fig. **??**Left 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

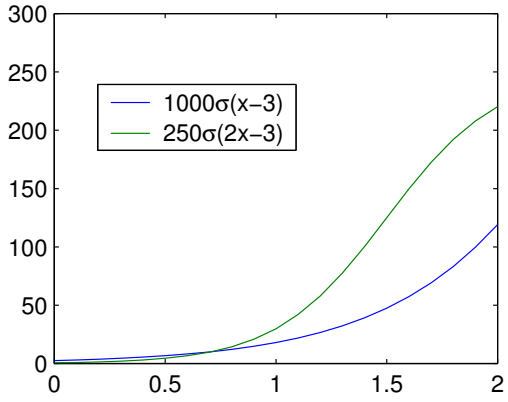


Figure 1: In the binary neuron model, the firing frequency of the neuron is a non-linear function of the input current.

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

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

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. 1 becomes

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

In Eq. 2, 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. 2 does not explicity depend on time, but only implicitly through the dependence of $\mathbf{s}$ on time.

1.1.1 Parallel dynamics: Little model

Eq. 2 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

$$T(\mathbf{s}'|\mathbf{s}) = \prod_i p(s_i', t+1|\mathbf{s}, t) \tag{3}$$

with $p(s_i', t+\tau|\mathbf{s}, t)$ given by Eq. 2. 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

$$p_{t+1}(\mathbf{s}') = \sum_{\mathbf{s}} T(\mathbf{s}'|\mathbf{s}) p_t(\mathbf{s}). \tag{4}$$

Eq 4 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. 2. This Markov process was first considered by Little [**?**].

1.1.2 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 [**?**]. 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 [**?**].

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. 2. 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

$$T(\mathbf{s}'|\mathbf{s}) \quad = \quad \frac{1}{n} p(s'_i, t+\tau|\mathbf{s}, t), \quad \text{if } \mathbf{s}' = F_i\mathbf{s} \tag{5}$$

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. 2. 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

$$T(\mathbf{s}|\mathbf{s}) = 1 - \frac{1}{n} \sum_i p(s'_i, t+\tau|\mathbf{s}, t). \tag{6}$$

Eqs. 5 and 6 together with Eq. 4 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.

1.2 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 [**?**].

1.2.1 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. 4 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:

$$\sum_{\mathbf{s}'} T(\mathbf{s}'|\mathbf{s}) = 1. \tag{7}$$

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:

$$l_\alpha^\dagger r_\beta = \delta_{\alpha\beta}. \tag{8}$$

δ_{ab} denotes the Kronecker delta: $\delta_{ab} = 1$ if $a = b$ and 0 otherwise. In Eq. 8, 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. 4:

$$p_t = T^t p_0 = \sum_\alpha \lambda_\alpha^t r_\alpha (l_\alpha^\dagger \cdot p_0) \tag{9}$$

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

$$T p_\infty = p_\infty. \tag{10}$$

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

1.2.2 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. 7. 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

$$p_\infty = \sum_{\alpha=1}^{k} \rho_\alpha r_\alpha \tag{11}$$

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. 8 and Eq. 11, 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

$$p_\infty = \sum_{\alpha=1}^{k} (l_\alpha^\dagger p_0) r_\alpha. \tag{12}$$

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. 9 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. 5 and 6 and has 4 eigenvalues. Their values as a function of w are plotted in Fig. 2a. We

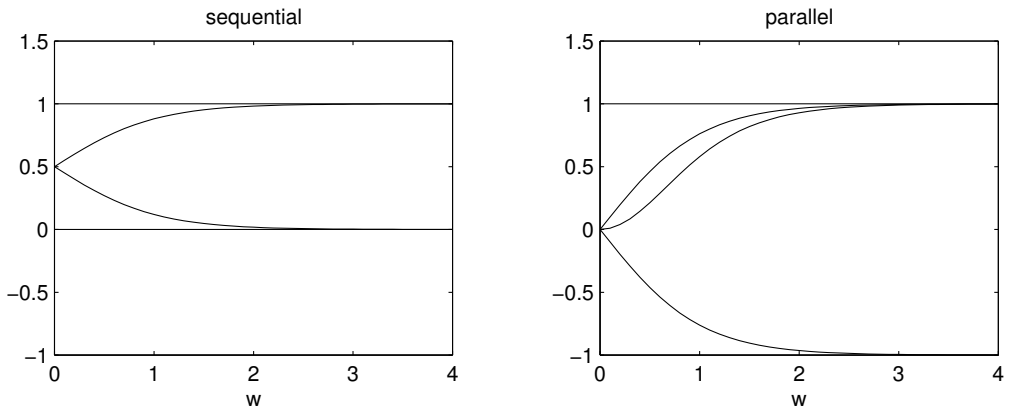


Figure 2: Eigenvalues of T as a function of w under sequential and parallel dynamics. For large w, multiple eigenvalues 1 signal ergodicity breaking.

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. 8. 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. 12:

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

$L_1 = 1$ 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. 2b. For small weights the network is again ergodic. The stationary distribution is given by Eq. 19 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

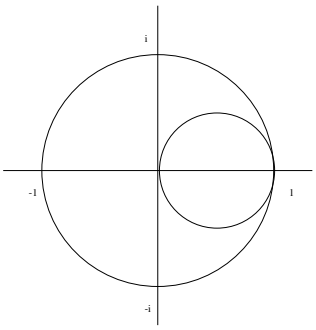


Figure 3: Eigenvalues of the transition matrix T are numbers in the complex plane with $|\lambda| \leq 1$. There is always at least one eigenvalue $\lambda = 1$. When the system is ergodic, there is only one eigenvalue 1 and the stationary distribution is unique and is reached for any initial state. 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 $\frac{1}{2}$ [**?**]. 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. 3.

1.3 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 **??** is a concrete example. The topic of Markov processes, irreducibility and ergodicity is taken from [**?**, **?**].

1.4 Exercises

1. (a) Compute the interspike interval distribution for the binary neuron as defined in Eq. 2.
   (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. 5 and 6.
   (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. 3.

(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?

2 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.

2.1 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

$$T(s|s')p(s') = T(s'|s)p(s) \text{ for all } s, s'. \tag{13}$$

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. 13 over all states s' and using Eq. 7. However, the reverse is not true: many stochastic dynamics do not satisfy detailed balance and a solution to Eq. 10 is then typically not available in analytical form, although its existence is dictated by the Perron-Frobenius theorem [**?**].

For random sequential dynamics, T is given by Eqs. 5 and 2 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

$$\frac{T(s|F_i s)}{T(F_i s|s)} = \exp(2(\sum_j w_{ij} s_j + \theta_i)s_i). \tag{14}$$

Consider the distribution

$$p(s) = \frac{1}{Z} \exp(\frac{1}{2} \sum_{ij} w_{ij} s_i s_j + \sum_i \theta_i s_i). \tag{15}$$

$p(s)$ 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:

$$-E(s) = \frac{1}{2}\sum_{ij} w_{ij}s_i s_j + \sum_i \theta_i s_i. \tag{16}$$

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

$$Z = \sum_s \exp(-E(s)) \tag{17}$$

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

$$\frac{p(s)}{p(F_i s)} = \exp(2(\sum_j w^s_{ij}s_j + \theta_i)s_i). \tag{18}$$

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]

2.2 Computing statistics

$p(s)$ in Eq. 15 and 19 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. 3. 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

$$p(s) = \frac{1}{Z}\exp(\sum_i \log\cosh(\sum_j w_{ij}s_j + \theta_i) + \sum_i \theta_i s_i). \tag{19}$$

This is the equilibrium distribution for parallel dynamics [**?**].

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

$$m_i = \langle s_i \rangle = \sum_s s_i p(s), \tag{20}$$

and correlations between neurons:

$$\chi_{ij} = \left\langle s_i s_j \right\rangle - \langle s_i \rangle \left\langle s_j \right\rangle = \sum_s s_i s_j p(s) - m_i m_j. \tag{21}$$

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 [**?**]. 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. 15 and 19 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. 20 and 21.

[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.

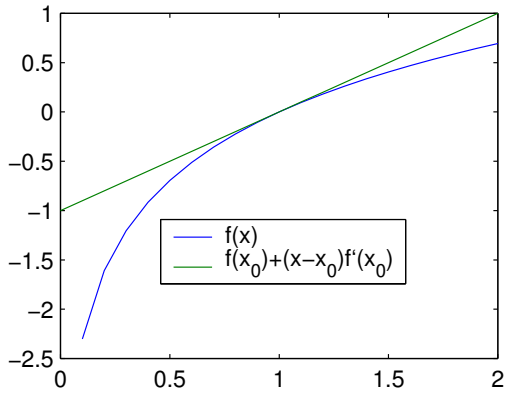


Figure 4: Illustration of the concavity property $f(x) \leq f(x_0) + (x - x_0)f'(x_0)$ for the logarithmic function and $x_0 = 1$.

2.3 Mean field theory

In this section, we will show how to approximate Z in Eq. 17 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. 17 and write

$$\begin{aligned} \log Z &= \log \sum_s \exp(-E(s)) = \log \sum_s q(s) \frac{\exp(-E(s))}{q(s)} \\ &\geq \sum_s q(s) \log \left(\frac{\exp(-E(s))}{q(s)} \right) = -\langle E \rangle_q + S_q = -F \end{aligned} \tag{22}$$

$q(s)$ 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. 4. 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

$$F = -\frac{1}{2}\sum_{ij} w_{ij} m_i m_j - \sum_i \theta_i m_i + \frac{1}{2}\sum_i \left((1 + m_i)\log(\frac{1}{2}(1 + m_i)) + (1 - m_i)\log(\frac{1}{2}(1 - m_i)) \right). \tag{23}$$

From Eq. 22, 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

$$m_i = \tanh(\sum_{j=1}^{n} w_{ij} m_j + \theta_i) \tag{24}$$

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. 20:

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

2.4 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. 21. Combining these two expressions, we can relate the correlations to the mean firing rates as

$$\chi_{ij} = \frac{\partial \langle s_i \rangle}{\partial \theta_j} \approx \frac{\partial m_i}{\partial \theta_j} \tag{25}$$

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

$$\frac{\partial \theta_i}{\partial m_j} = \frac{\delta_{ij}}{1 - m_i^2} - w_{ij}. \tag{26}$$

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

2.5 Boltzmann Machines

A well-known application of the Boltzmann-Gibbs distribution are Boltzmann Machines [**?**]. The basic idea is to treat the distribution Eq. 15 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 [**?**, **?**]:

$$\begin{aligned} \frac{\partial L}{\partial \theta_i} &= \Big(\langle s_i \rangle_c - \langle s_i \rangle \Big), \\ \frac{\partial L}{\partial w_{ij}} &= \Big(\Big\langle s_i s_j \Big\rangle_c - \Big\langle s_i s_j \Big\rangle \Big) \; i \neq j. \end{aligned} \tag{27}$$

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:

$$\langle s_i \rangle = \sum_s s_i p(s) \tag{28}$$

$$\Big\langle s_i s_j \Big\rangle = \sum_s s_i s_j p(s) \tag{29}$$

with $p(s)$ given by Eq. 15. The 'clamped' expectation values are simply the statistics computed in the training set:

$$\langle s_i \rangle_c = \frac{1}{P} \sum_\mu s_i^\mu \tag{30}$$

$$\Big\langle s_i s_j \Big\rangle_c = \frac{1}{P} \sum_\mu s_i^\mu s_j^\mu \tag{31}$$

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. 27 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. 29 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. 24 and $\left\langle s_i s_j \right\rangle$ from Eqs. 25 and 26 and insert these values into the learning rule Eq. 27. 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. 27 equal to zero and solve these equations directly. In the mean field approximation, these equations read:

$$m_i = \langle s_i \rangle_c \tag{32}$$

$$\chi_{ij} = \left\langle s_i s_j \right\rangle_c - m_i m_j, i \neq j. \tag{33}$$

m_i is a function of w_{ij} and θ_i as given by the mean field equations Eqs. 24. χ_{ij} is a function of w_{ij} and m_i as given by the linear response equations Eqs. 25 and 26. Eqs. 32 and 33 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. 33 can be computed from the data, because of Eq. 32. Thus Eq. 33 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 25 and 26. This gives an explicit solution for w_{ij} in terms of known quantities.

However, this procedure is incorrect, because Eq. 33 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. 24 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. 33: $\chi_{ii} = 1 - m_i^2$. 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

$$m_i = \tanh(\sum_{j=1}^{n} w_{ij} m_j + \theta_i) \tag{34}$$

$$\chi^{-1}_{ij} = \frac{\partial \theta_j}{\partial m_i} = \frac{\delta_{ij}}{1 - m_i^2} - w_{ij}. \tag{35}$$

From Eq. 32-35 we can compute the solution for w_{ij} and θ_i in closed form:

$$m_i = \langle s_i \rangle_c \tag{36}$$

$$C_{ij} = \left\langle s_i s_j \right\rangle_c - \langle s_i \rangle_c \left\langle s_j \right\rangle_c \tag{37}$$

$$w_{ij} = \frac{\delta_{ij}}{1 - m_i^2} - \left(C^{-1}\right)_{ij} \tag{38}$$

$$\theta_i = \tanh^{-1}(m_i) - \sum_{j=1}^{n} w_{ij} m_j \tag{39}$$

2.5.1 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. 5.

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 $W^\alpha = (w^\alpha_{ij}, \theta^\alpha_i), \alpha = 1, \ldots, 10$.

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:

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

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. 23.

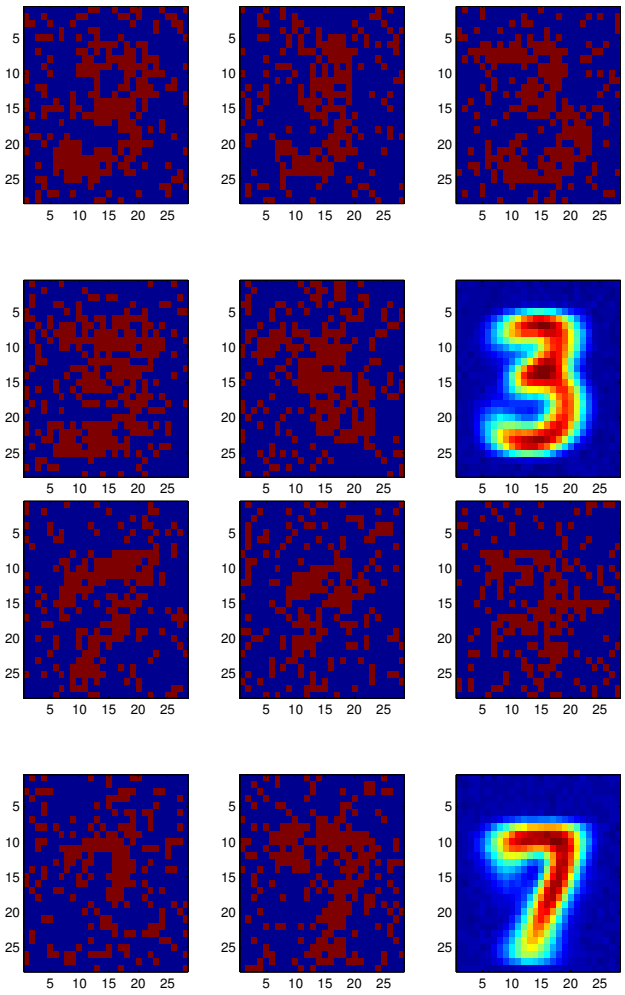


Figure 5: Sample of 60000 training patterns and 10000 test patterns of the 28 × 28 handwritten digits of the U.S. Postal Service Office of Advanced Technology. Patterns are binary and 10 % pixel noise is added.

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.

2.6 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 [**?**].

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 [**?**] and [**?**], 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.

2.7 Exercises

1. (a) Derive Eq. 14.

   (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.

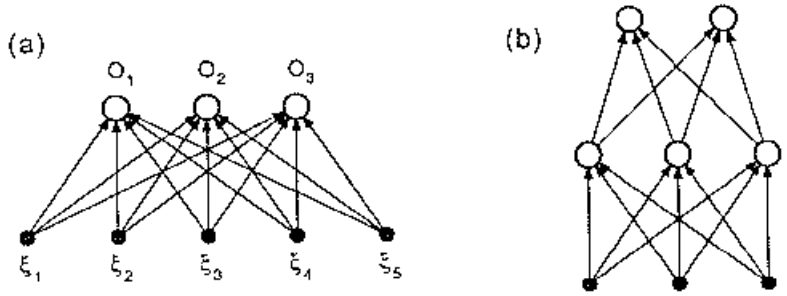


Figure 6: A) Simple Perceptron B) Multi-layered Perceptron

3 Perceptrons

Perceptrons are feed-forward neural networks. Examples are given in Fig. 6. 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:

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

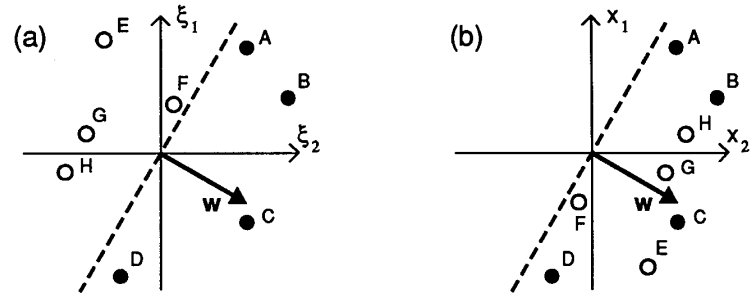
3.1 Threshold units

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

3.2 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. 7).

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. 7A). Other problems that are not linearly separable occur when three or more input patterns are linearly dependent (see Fig. 7B).

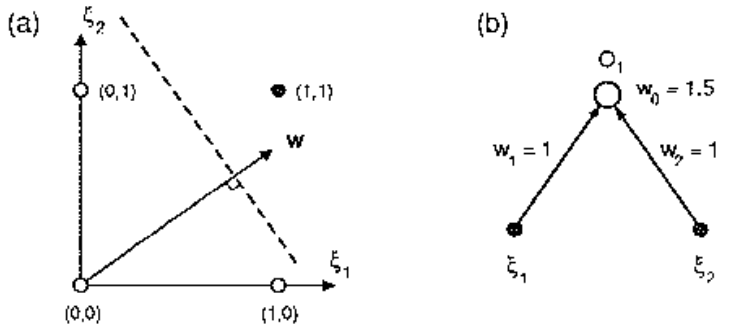
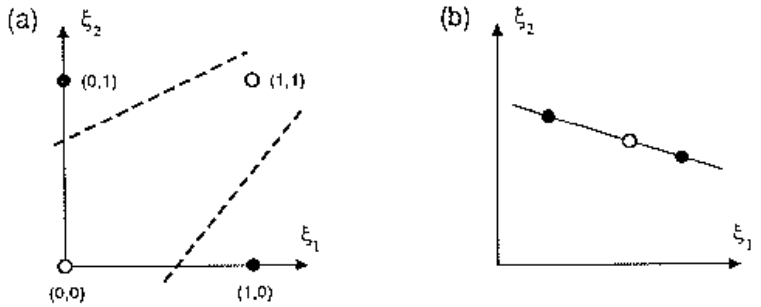


Figure 7: The AND problem for two inputs is linearly separable.



3.3 Perceptron learning rule

We have seen that the desired weight vector satisfies

$$w \cdot x^\mu > 0, \quad \text{all patterns} \tag{40}$$

We define the following perceptron learning rule:

$$\begin{aligned} w_j^{\text{new}} &= w_j^{\text{old}} + \Delta w_j \\ \Delta w_j &= \eta \Theta(-w \cdot x^\mu) \xi_j^\mu \zeta^\mu = \eta \Theta(-w \cdot x^\mu) x^\mu \end{aligned} \tag{41}$$

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.

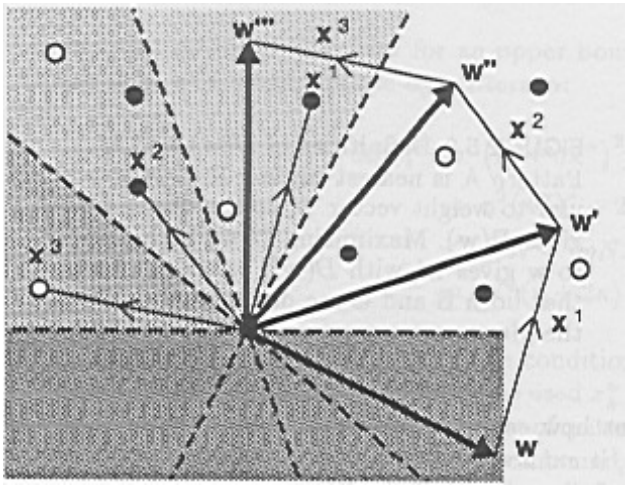


Figure 8: The perceptron learning rule in action. Learning rule Eq. 41 is applied to all patterns in some random or given order. Learning stops, when a weight configuration is found that has positive inner product with all training patterns.

In Fig. 8 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. 9 we give examples of two data sets and their solutions Eq. 40. In Fig. 9A there are more admissible weight vectors and they can have a larger inner product with all training patterns than in Fig. 9B. We define 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. 10, 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.

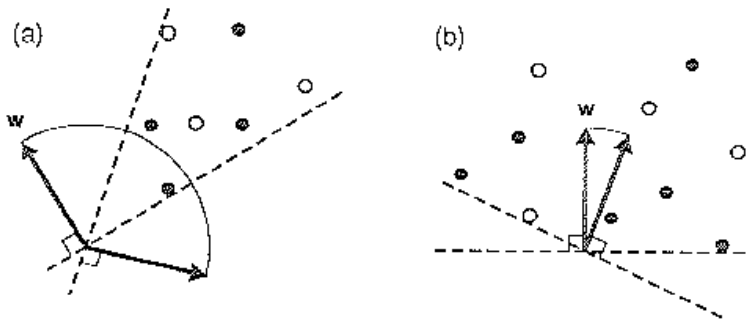


Figure 9: Two examples of data sets and the sets of w that satisfy condition Eq. 40. A) Many solutions B) Few solutions.

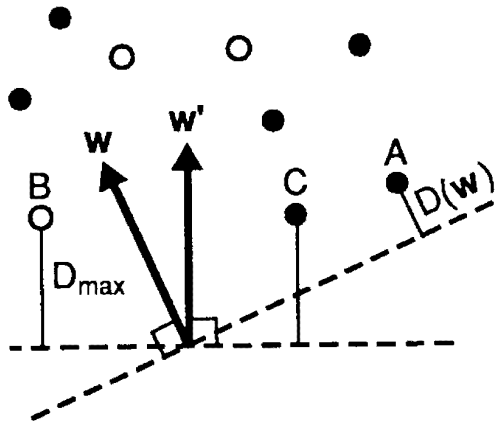


Figure 10: Two admissible solutions w and w' and their values of D respectively. Since $D(w') > D(w)$, w' is the preferred solution.

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.

3.3.1 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 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,

$$\frac{w \cdot w^*}{|w||w^*|} \geq \sqrt{M} \frac{D(w^*)}{\sqrt{N}} \tag{42}$$

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. 42 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).

3.4 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

$$w_j = \frac{1}{N} \sum_{\rho\nu} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^\nu, \quad Q_{\rho\nu} = \frac{1}{N} \sum_j \xi_j^\rho \xi_j^\nu \tag{43}$$

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

To verify that Eq. 43 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 $P \leq N$. [5]

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 $\xi^\perp$:

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

3.4.1 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.

[5] In addition, the input patterns must be linearly independent. If the input patterns are linearly dependent, solution Eq. 43 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. 43 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'

$$w_i = w_i + \Delta w_i \qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{44}$$

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).

3.4.2 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. 11, 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. 44 becomes

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

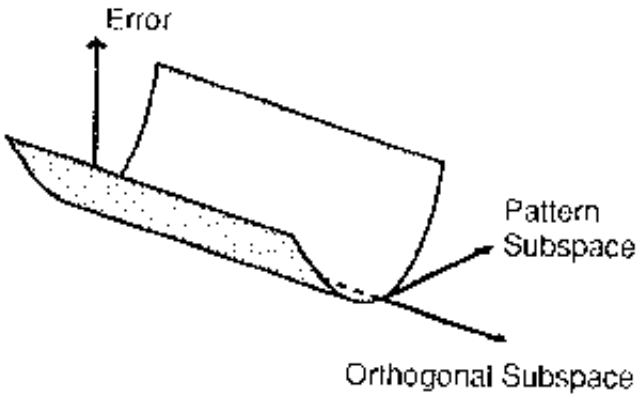


Figure 11: Cost landscape $E(w)$ with different curvatures in different directions.

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

$$|1 - 2\eta a_i| < 1. \tag{45}$$

We must find an η that satisfies Eq. 45 for all i. When $1 - 2\eta a_i < 0$, δw_i changes sign in each iteration. The behavior is illustrated in Fig. 12 with $E(w_1, w_2) = w_1^2 + 20w_2^2$ for different values of η.

3.5 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:

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

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

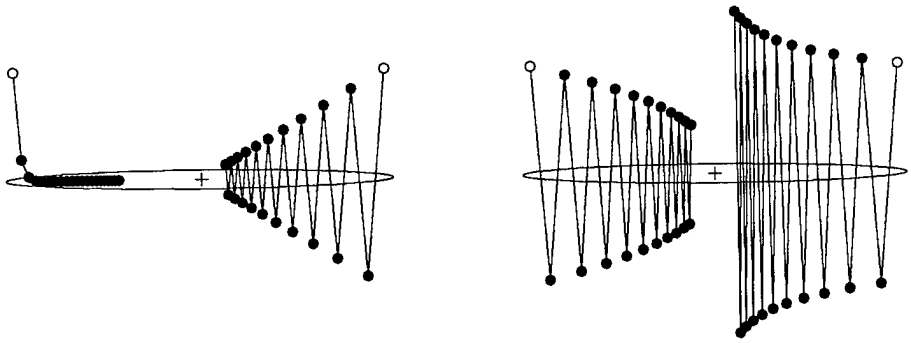


Figure 12: Behavior of the gradient descent learning rule Eq. 44 for the quadratic cost function $E(w_1, w_2) = w_1^2 + 20w_2^2$ for $\eta = 0.02, 0.0476, 0.049, 0.0505$.

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.

3.6 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. 6B. 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:

$$o_i = g\left(\sum_j w_{ij} v_j\right) = g\left(\sum_j w_{ij} g\left(\sum_k w_{jk}\xi_k\right)\right) \tag{46}$$

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:

$$E(w) = \frac{1}{2} \sum_i \sum_\mu \left(o_i^\mu - \zeta_i^\mu \right)^2 \tag{47}$$

with o_i^μ the output on node i for input pattern ξ^μ as given by Eq. 46.

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 [**?**] 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. 47 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. 13 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. 13A. The solution does not pass through all the data points. For a larger number of hidden units, the solution may look something like Fig. 13B. 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.

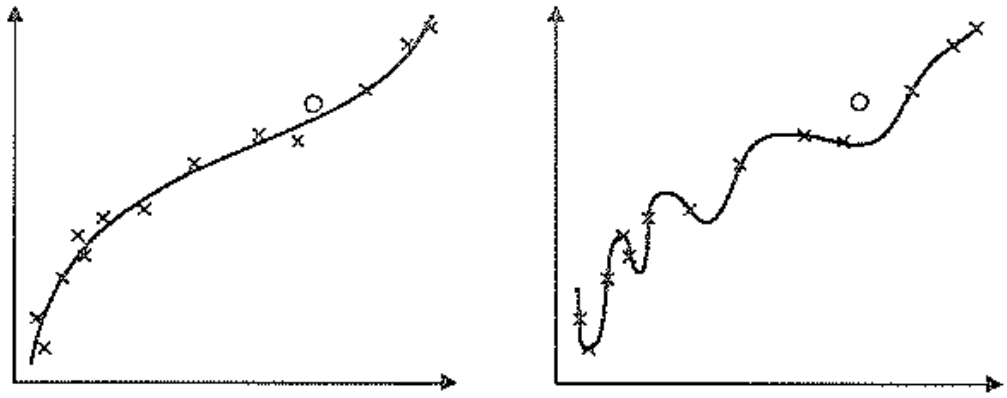


Figure 13: Network output versus network input. A) Network with a small number of hidden units. B) Network with a large number of hidden units. Networks with more hidden units can implement more complex functions and can better fit a given training set. However, more complex networks do not necessarily generalize better on novel data.

3.7 Summary

This chapter is based on [**?**]. 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.

3.8 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 3.4) 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. 44 asymptotisch convergeert.