

Weighted Networks – The Perceptron

3.1 Perceptrons and parallel processing

In the previous chapter we arrived at the conclusion that McCulloch–Pitts units can be used to build networks capable of computing any logical function and of simulating any finite automaton. From the biological point of view, however, the types of network that can be built are not very relevant. The computing units are too similar to conventional logic gates and the network must be completely specified before it can be used. There are no free parameters which could be adjusted to suit different problems. Learning can only be implemented by modifying the connection pattern of the network and the thresholds of the units, but this is necessarily more complex than just adjusting numerical parameters. For that reason, we turn our attention to weighted networks and consider their most relevant properties. In the last section of this chapter we show that simple weighted networks can provide a computational model for regular neuronal structures in the nervous system.

3.1.1 Perceptrons as weighted threshold elements

In 1958 Frank Rosenblatt, an American psychologist, proposed the *perceptron*, a more general computational model than McCulloch–Pitts units. The essential innovation was the introduction of numerical weights and a special interconnection pattern. In the original Rosenblatt model the computing units are threshold elements and the connectivity is determined stochastically. Learning takes place by adapting the weights of the network with a numerical algorithm. Rosenblatt's model was refined and perfected in the 1960s and its computational properties were carefully analyzed by Minsky and Papert [312]. In the following, Rosenblatt's model will be called the *classical perceptron* and the model analyzed by Minsky and Papert the *perceptron*.

The classical perceptron is in fact a whole network for the solution of certain pattern recognition problems. In Figure 3.1 a projection surface called the

retina transmits binary values to a layer of computing units in the projection area. The connections from the retina to the projection units are deterministic and non-adaptive. The connections to the second layer of computing elements and from the second to the third are stochastically selected in order to make the model biologically plausible. The idea is to train the system to recognize certain input patterns in the connection region, which in turn leads to the appropriate path through the connections to the reaction layer. The learning algorithm must derive suitable weights for the connections.

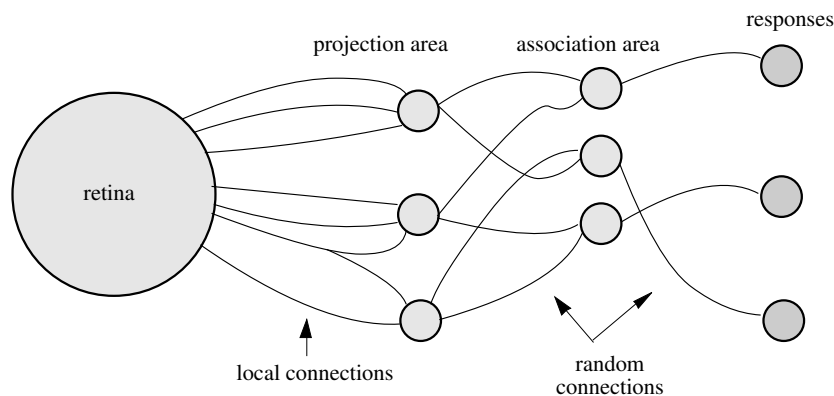


Fig. 3.1. The classical perceptron [after Rosenblatt 1958]

Rosenblatt's model can only be understood by first analyzing the elementary computing units. From a formal point of view, the only difference between McCulloch–Pitts elements and perceptrons is the presence of weights in the networks. Rosenblatt also studied models with some other differences, such as putting a limit on the maximum acceptable fan-in of the units.

Minsky and Papert distilled the essential features from Rosenblatt's model in order to study the computational capabilities of the perceptron under different assumptions. In the model used by these authors there is also a retina of pixels with binary values on which patterns are projected. Some pixels from the retina are directly connected to logic elements called predicates which can compute a single bit according to the input. Interestingly, these predicates can be as computationally complex as we like; for example, each predicate could be implemented using a supercomputer. There are some constraints however, such as the number of points in the retina that can be simultaneously examined by each predicate or the distance between those points. The predicates transmit their binary values to a weighted threshold element which is in charge of reaching the final decision in a pattern recognition problem. The question is then, what kind of patterns can be recognized in this massively parallel manner using a single threshold element at the output of the network? Are there limits to what we can compute in parallel using unlimited processing power

for each predicate, when each predicate cannot itself look at the whole retina? The answer to this problem in some ways resembles the speedup problem in parallel processing, in which we ask what percentage of a computational task can be parallelized and what percentage is inherently sequential.

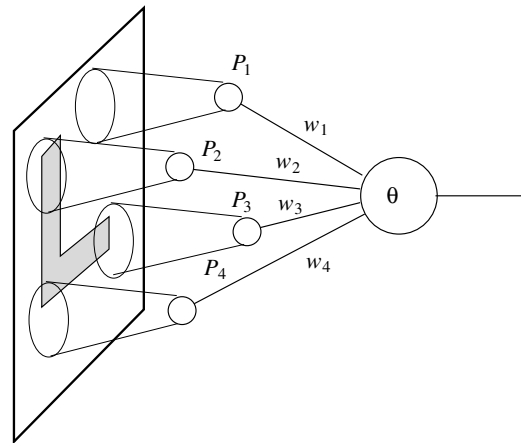


Fig. 3.2. Predicates and weights of a perceptron

Figure 3.2 illustrates the model discussed by Minsky and Papert. The predicates P_1 to P_4 deliver information about the points in the projection surface that comprise their *receptive fields*. The only restriction on the computational capabilities of the predicates is that they produce a binary value and the receptive field cannot cover the whole retina. The threshold element collects the outputs of the predicates through weighted edges and computes the final decision. The system consists in general of n predicates P_1, P_2, \dots, P_n and the corresponding weights w_1, w_2, \dots, w_n . The system fires only when $\sum_{i=1}^n w_i P_i \geq \theta$, where θ is the threshold of the computing unit at the output.

3.1.2 Computational limits of the perceptron model

Minsky and Papert used their simplified perceptron model to investigate the computational capabilities of weighted networks. Early experiments with Rosenblatt's model had aroused unrealistic expectations in some quarters, and there was no clear understanding of the class of pattern recognition problems which it could solve efficiently. To explore this matter the number of predicates in the system is fixed, and although they possess unbounded computational power, the final bottleneck is the parallel computation with a single threshold element. This forces each processor to *cooperate* by producing a partial result pertinent to the global decision. The question now is which problems can be solved in this way and which cannot.

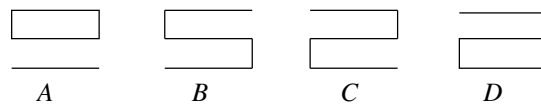
The system considered by Minsky and Papert at first appears to be a strong simplification of parallel decision processes, but it contains some of the most important elements differentiating between *sequential* and *parallel* processing. It is known that when some algorithms are parallelized, an irreducible sequential component sometimes limits the maximum achievable speedup. The mathematical relation between speedup and irreducible sequential portion of an algorithm is known as *Amdahl's law* [187]. In the model considered above the central question is, are there pattern recognition problems in which we are forced to analyze sequentially the output of the predicates associated with each receptive field or not? Minsky and Papert showed that problems of this kind do indeed exist which cannot be solved by a single perceptron acting as the last decision unit.

The limits imposed on the receptive fields of the predicates are based on realistic assumptions. The predicates are fixed in advance and the pattern recognition problem can be made arbitrarily large (by expanding the retina). According to the number of points and their connections to the predicates, Minsky and Papert differentiated between

- Diameter limited perceptrons: the receptive field of each predicate has a limited diameter.
- Perceptrons of limited order: each receptive field can only contain up to a certain maximum number of points.
- Stochastic perceptrons: each receptive field consists of a number of randomly chosen points

Some patterns are more difficult to identify than others and this structural classification of perceptrons is a first attempt at defining something like complexity classes for pattern recognition. Connectedness is an example of a property that cannot be recognized by constrained systems.

Proposition 6. *No diameter limited perceptron can decide whether a geometric figure is connected or not.*



Proof. We proceed by contradiction, assuming that a perceptron can decide whether a figure is connected or not. Consider the four patterns shown above; notice that only the middle two are connected.

Since the diameters of the receptive fields are limited, the patterns can be stretched horizontally in such a way that no single receptive field contains points from both the left and the right ends of the patterns. In this case

we have three different groups of predicates: the first group consists of those predicates whose receptive fields contain points from the left side of a pattern. Predicates of the second group are those whose receptive fields cover the right side of a pattern. All other predicates belong to the third group. In Figure 3.3 the receptive fields of the predicates are represented by circles.

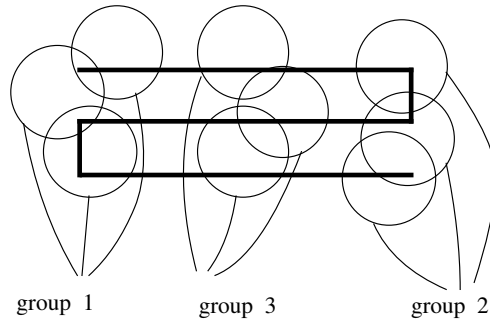


Fig. 3.3. Receptive fields of predicates

All predicates are connected to a threshold element through weighted edges which we denote by the letter w with an index. The threshold element decides whether a figure is connected or not by performing the computation

$$S = \sum_{P_i \in \text{group 1}} w_{1i} P_i + \sum_{P_i \in \text{group 2}} w_{2i} P_i + \sum_{P_i \in \text{group 3}} w_{3i} P_i - \theta \geq 0.$$

If S is positive the figure is recognized as connected, as is the case, for example, in Figure 3.3.

If the disconnected pattern A is analyzed, then we should have $S < 0$. Pattern A can be transformed into pattern B without affecting the output of the predicates of group 3, which do not recognize the difference since their receptive fields do not cover the sides of the figures. The predicates of group 2 adjust their outputs by $\Delta_2 S$ so that now

$$S + \Delta_2 S \geq 0 \Rightarrow \Delta_2 S \geq -S.$$

If pattern A is transformed into pattern C , the predicates of group 1 adjust their outputs so that the threshold element receives a net excitation, i.e.,

$$S + \Delta_1 S \geq 0 \Rightarrow \Delta_1 S \geq -S.$$

However, if pattern A is transformed into pattern D , the predicates of group 1 cannot distinguish this case from the one for figure C and the predicates of group 2 cannot distinguish this case from the one for figure B . Since the predicates of group 3 do not change their output we have

$$\Delta S = \Delta_2 S + \Delta_1 S \geq -2S,$$

and from this

$$S + \Delta S \geq -S > 0.$$

The value of the new sum can only be positive and the whole system classifies figure D as connected. Since this is a contradiction, such a system cannot exist. \square

Proposition 6 states only that the connectedness of a figure is a global property which cannot be decided locally. If no predicate has access to the whole figure, then the only alternative is to process the outputs of the predicates sequentially.

There are some other difficult problems for perceptrons. They cannot decide, for example, whether a set of points contains an even or an odd number of elements when the receptive fields cover only a limited number of points.

3.2 Implementation of logical functions

In the previous chapter we discussed the synthesis of Boolean functions using McCulloch–Pitts networks. Weighted networks can achieve the same results with fewer threshold gates, but the issue now is which functions can be implemented using a single unit.

3.2.1 Geometric interpretation

In each of the previous sections a threshold element was associated with a whole set of predicates or a network of computing elements. From now on, we will deal with perceptrons as isolated threshold elements which compute their output without delay.

Definition 1. *A simple perceptron is a computing unit with threshold θ which, when receiving the n real inputs x_1, x_2, \dots, x_n through edges with the associated weights w_1, w_2, \dots, w_n , outputs 1 if the inequality $\sum_{i=1}^n w_i x_i \geq \theta$ holds and otherwise 0.*

The origin of the inputs is not important, whether they come from other perceptrons or another class of computing units. The geometric interpretation of the processing performed by perceptrons is the same as with McCulloch–Pitts elements. A perceptron separates the input space into two half-spaces. For points belonging to one half-space the result of the computation is 0, for points belonging to the other it is 1.

Figure 3.4 shows this for the case of two variables x_1 and x_2 . A perceptron with threshold 1, at which two edges with weights 0.9 and 2.0 impinge, tests the condition

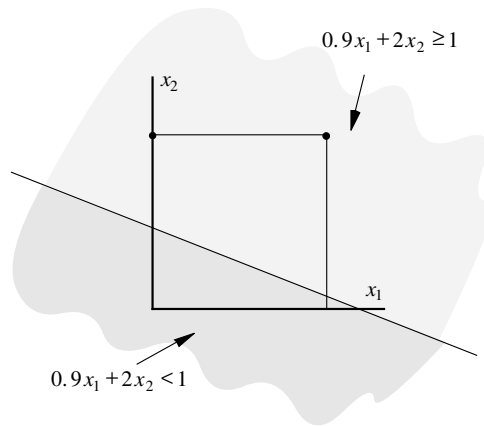


Fig. 3.4. Separation of input space with a perceptron

$$0.9x_1 + 2x_2 \geq 1.$$

It is possible to generate arbitrary separations of input space by adjusting the parameters of this example.

In many cases it is more convenient to deal with perceptrons of threshold zero only. This corresponds to linear separations which are forced to go through the origin of the input space. The two perceptrons in Figure 3.5 are equivalent. The threshold of the perceptron to the left has been converted into the weight $-\theta$ of an additional input channel connected to the constant 1. This extra weight connected to a constant is called the *bias* of the element.

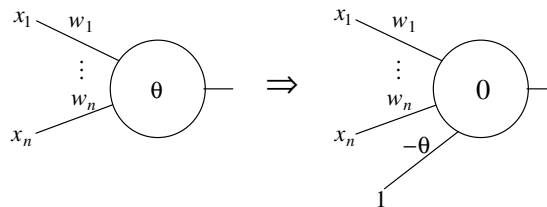


Fig. 3.5. A perceptron with a bias

Most learning algorithms can be stated more concisely by transforming thresholds into biases. The input vector (x_1, x_2, \dots, x_n) must be extended with an additional 1 and the resulting $(n + 1)$ -dimensional vector $(x_1, x_2, \dots, x_n, 1)$ is called the *extended input vector*. The extended weight vector associated with this perceptron is $(w_1, \dots, w_n, w_{n+1})$, whereby $w_{n+1} = -\theta$.

3.2.2 The XOR problem

We can now deal with the problem of determining which logical functions can be implemented with a single perceptron. A perceptron network is capable of computing any logical function, since perceptrons are even more powerful than unweighted McCulloch–Pitts elements. If we reduce the network to a single element, which functions are still computable?

Taking the functions of two variables as an example we can gain some insight into this problem. Table 3.1 shows all 16 possible Boolean functions of two variables f_0 to f_{15} . Each column f_i shows the value of the function for each combination of the two variables x_1 and x_2 . The function f_0 , for example, is the zero function whereas f_{14} is the OR-function.

Table 3.1. The 16 Boolean functions of two variables

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Perceptron-computable functions are those for which the points whose function value is 0 can be separated from the points whose function value is 1 using a line. Figure 3.6 shows two possible separations to compute the OR and the AND functions.

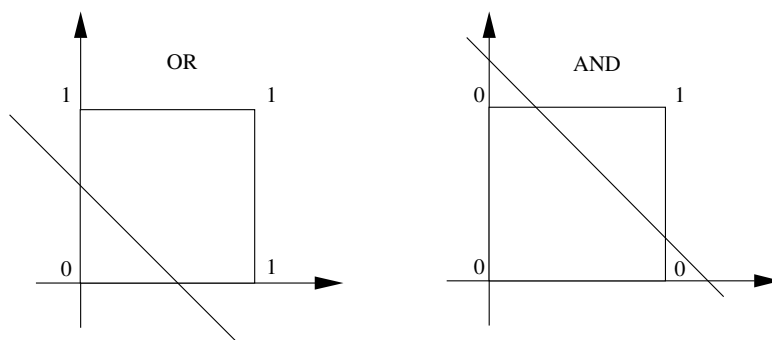


Fig. 3.6. Separations of input space corresponding to OR and AND

It is clear that two of the functions in the table cannot be computed in this way. They are the function XOR and identity (f_6 and f_9). It is intuitively evident that no line can produce the necessary separation of the input space. This can also be shown analytically.

Let w_1 and w_2 be the weights of a perceptron with two inputs, and θ its threshold. If the perceptron computes the XOR function the following four inequalities must be fulfilled:

$$\begin{array}{llll} x_1 = 0 \ x_2 = 0 \ w_1x_1 + w_2x_2 = 0 & \Rightarrow & 0 < \theta \\ x_1 = 1 \ x_2 = 0 \ w_1x_1 + w_2x_2 = w_1 & \Rightarrow & w_1 \geq \theta \\ x_1 = 0 \ x_2 = 1 \ w_1x_1 + w_2x_2 = w_2 & \Rightarrow & w_2 \geq \theta \\ x_1 = 1 \ x_2 = 1 \ w_1x_1 + w_2x_2 = w_1 + w_2 & \Rightarrow & w_1 + w_2 < \theta \end{array}$$

Since θ is positive, according to the first inequality, w_1 and w_2 are positive too, according to the second and third inequalities. Therefore the inequality $w_1 + w_2 < \theta$ cannot be true. This contradiction implies that no perceptron capable of computing the XOR function exists. An analogous proof holds for the function f_9 .

3.3 Linearly separable functions

The example of the logical functions of two variables shows that the problem of perceptron computability must be discussed in more detail. In this section we provide the necessary tools to deal more effectively with functions of n arguments.

3.3.1 Linear separability

We can deduce from our experience with the XOR function that many other logical functions of several arguments must exist which cannot be computed with a threshold element. This fact has to do with the geometry of the n -dimensional hypercube whose vertices represent the combination of logic values of the arguments. Each logical function separates the vertices into two classes. If the points whose function value is 1 cannot be separated with a linear cut from the points whose function value is 0, the function is not perceptron-computable. The following two definitions give this problem a more general setting.

Definition 2. *Two sets of points A and B in an n -dimensional space are called linearly separable if $n + 1$ real numbers w_1, \dots, w_{n+1} exist, such that every point $(x_1, x_2, \dots, x_n) \in A$ satisfies $\sum_{i=1}^n w_i x_i \geq w_{n+1}$ and every point $(x_1, x_2, \dots, x_n) \in B$ satisfies $\sum_{i=1}^n w_i x_i < w_{n+1}$*

Since a perceptron can only compute linearly separable functions, an interesting question is how many linearly separable functions of n binary arguments there are. When $n = 2$, 14 out of the 16 possible Boolean functions are linearly separable. When $n = 3$, 104 out of 256 and when $n = 4$, 1882 out of 65536 possible functions are linearly separable. Although there has been extensive research on linearly separable functions in recent years, no formula for

expressing the number of linearly separable functions as a function of n has yet been found. However we will provide some upper bounds for this number in the following chapters.

3.3.2 Duality of input space and weight space

The computation performed by a perceptron can be visualized as a linear separation of input space. However, when trying to find the appropriate weights for a perceptron, the search process can be better visualized in weight space. When m real weights must be determined, the search space is the whole of \mathbb{R}^m .

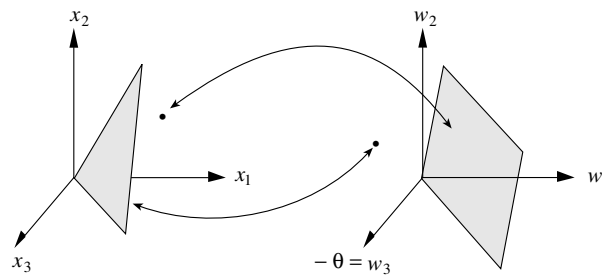


Fig. 3.7. Illustration of the duality of input and weight space

For a perceptron with n input lines, finding the appropriate linear separation amounts to finding $n + 1$ free parameters (n weights and the bias). These $n + 1$ parameters represent a point in $(n + 1)$ -dimensional weight space. Each time we pick one point in weight space we are choosing one combination of weights and a specific linear separation of input space. This means that every point in $(n + 1)$ -dimensional weight space can be associated with a hyperplane in $(n + 1)$ -dimensional extended input space. Figure 3.7 shows an example. Each combination of three weights, w_1, w_2, w_3 , which represent a point in weight space, defines a separation of input space with the plane $w_1x_1 + w_2x_2 + w_3x_3 = 0$.

There is the same kind of relation in the inverse direction, from input to weight space. If we want the point x_1, x_2, x_3 to be located in the positive half-space defined by a plane, we need to determine the appropriate weights w_1, w_2 and w_3 . The inequality

$$w_1x_1 + w_2x_2 + w_3x_3 \geq 0$$

must hold. However this inequality defines a linear separation of weight space, that is, the point (x_1, x_2, x_3) defines a cutting plane in weight space. Points in one space are mapped to planes in the other and vice versa. This complementary relation is called *duality*. Input and weight space are dual spaces and we

