Mathematical Theory of Neural Networks

Eva Zerz[⋆], Uwe Helmke[†], Dieter Prätzel-Wolters[⋆]

[⋆] Fachbereich Mathematik
Universität Kaiserslautern
67663 Kaiserslautern

[†] Mathematisches Institut
Universität Würzburg
97074 Würzburg

July 2001

Contents

Chapter 1

The Perceptron

In this chapter, the perceptron is introduced as a simple model of a nerve cell. It is shown that perceptrons are closely linked with the problem of linear separation. The perceptron learning algorithm will be described, and the issue of optimal separation will be studied. It will be shown that an optimal linear separation of finite disjoint sets can be achieved by solving a linear or quadratic optimization problem.

1.1 Formal Neurons

The structure and dynamics of biological neurons is very complex. To achieve simple and easily manageable models of artificial neurons, it is necessary to comprehend the essential functional characteristics of a neuron in a drastically simplified form. Such models are referred to as formal neurons.

Definition 1.1 A **formal neuron** is a data quadruple (X, Y, σ, s), where $X \subseteq \mathbb{R}^n$ (for some positive integer n), $Y \subseteq \mathbb{R}$, and s and σ are mappings

$$s : X \to \mathbb{R} \quad \text{and} \quad \sigma : \mathbb{R} \to Y,$$

respectively. We call X and Y the **input and output value sets**, respectively, and n denotes the number of inputs. The mapping s is called **activation function** and σ is called the **output map**. Sometimes, a formal neuron will be identified with its **transfer function**, which is the composed input-to-output map

$$f = \sigma \circ s : X \to Y.$$

In 1943, Warren McCulloch and Walter Pitts developed a simple but fundamental model which has the capacity to realize the elementary logical functions NOT, AND, OR. In particular, all finite Boolean functions can be realized by using appropriate networks of such neurons.

From a biological point of view, the input signals can be viewed to stem from receptors of connected neurons. The signals are modified at the synapses and are condensed into a single signal at the soma. If the quantity surpasses a certain threshold, the neuron "fires." The simplest way to model such a neuron therefore uses $Y = \{0, 1\}$ as its output value set (the output is 1 if the neuron fires, and 0 otherwise) and the affine linear map

$$s(x) = s(x_1, \dots, x_n) = \sum_{j=1}^{n} w_j x_j - \theta \tag{1.1}$$

as the activation function. The **weights** $w_i \in \mathbb{R}$ model the influence of the synapses on the signal x_i and $\theta \in \mathbb{R}$ represents the **threshold**. If the Heaviside function $\mathrm{sat} : \mathbb{R} \to \{0, 1\}$, which is defined by

$$\mathrm{sat}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

is used as the output map, such a formal neuron transmits the signal 1 if and only if $\sum_{i=1}^{n} w_i x_i \geq \theta$. Such formal neurons are called **McCulloch-Pitts neurons** or **perceptrons**. Often, the output map σ is required to be a **sigmoid** function, i.e., one considers the continuous value set $Y = [0, 1]$, and a monotone function

$$\sigma : \mathbb{R} \to [0, 1] \quad \text{with} \quad \lim_{z \to -\infty} \sigma(z) = 0 \quad \text{and} \quad \lim_{z \to \infty} \sigma(z) = 1.$$

Common examples for sigmoid functions are:

1. the Heaviside function sat;

2. "ramp" functions, where for $\alpha > 0$

$$\sigma(z) = \begin{cases} 0 & \text{if } z \leq -\alpha \\ \frac{1}{2}(\frac{z}{\alpha} + 1) & \text{if } -\alpha < z < \alpha \\ 1 & \text{if } \alpha \leq z; \end{cases}$$

3. the Fermi function $\sigma(z) = \frac{1}{1+e^{-z}}$ and its scaled versions $\sigma(z) = \frac{1}{1+e^{-\alpha z}}$, where $\alpha > 0$;

4. the modified hyperbolic tangent $\sigma(z) = \frac{1}{2}(\frac{e^z - e^{-z}}{e^z + e^{-z}} + 1)$.

In summary, we arrive at the following definition.

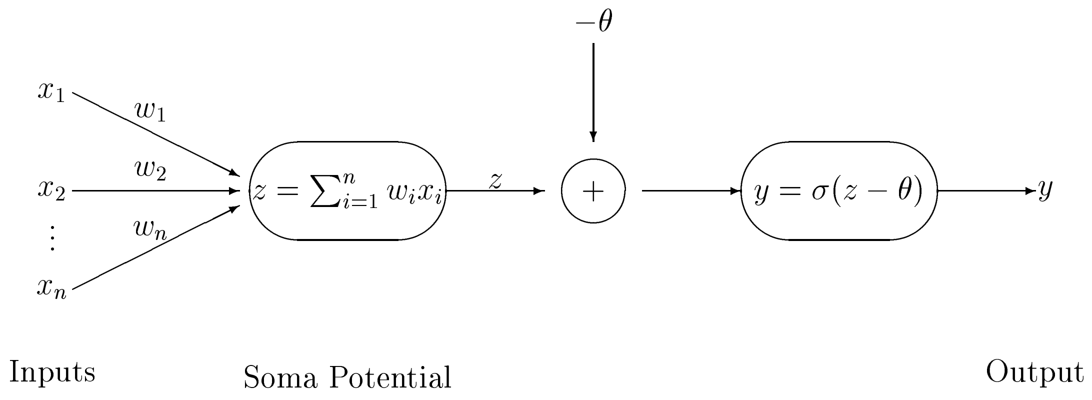


Figure 1.1: *Representation of a formal neuron with an affine activation function. For the perceptron, the Heaviside function is used as the output function σ.*

Definition 1.2 Let $s(x_1, \ldots, x_n) = \sum_{i=1}^n w_i x_i - \theta$, $w_i, \theta \in \mathbb{R}$, $i = 1, \ldots, n$. A formal neuron (X, Y, σ, s) is called a **$\boldsymbol{\sigma}$-perceptron**, and if $\sigma \equiv \mathrm{sat}$ simply a **perceptron** or **McCulloch-Pitts neuron**. Then we put $Y = \{0, 1\}$.

We first focus on McCulloch-Pitts neurons with $X = \{0,1\}^n$. If only binary inputs and outputs are used, a perceptron's transfer function is a Boolean or switching function

$$f : \{0,1\}^n \to \{0,1\}$$

of the form

$$f(x) = f(x_1, \ldots, x_n) = \mathrm{sat}\left(\sum_{i=1}^n w_i x_i - \theta\right) = \mathrm{sat}(\langle w, x \rangle - \theta),$$

with fixed weights $w = (w_1, \ldots, w_n)^T, w_i \in \mathbb{R}$ and a fixed threshold $\theta \in \mathbb{R}$. Various switching functions $f : \{0,1\}^n \to \{0,1\}$ can be realized by a perceptron, i.e., we can find a perceptron whose transfer function equals f.

Examples:

1. $f = \mathrm{NOT}$;

x	$f(x)$
0	1
1	0

f can be realized by a perceptron, e.g. $f(x) = \mathrm{sat}(-x + 0.5)$.

2. f = AND;

x_1	x_2	$f(x)$
0	0	0
0	1	0
1	0	0
1	1	1

Then, $f(x) = \text{sat}(x_1 + x_2 - 1.5)$ is an appropriate perceptron.

3. f = OR;

x_1	x_2	$f(x)$
0	0	0
0	1	1
1	0	1
1	1	1

The perceptron $f(x) = \text{sat}(x_1 + x_2 - 0.5)$ represents the given function.

Since any Boolean function can be written in disjunctive normal form, using only AND, OR, and NOT operations, this implies that all Boolean functions can be represented by a suitable network consisting of McCulloch-Pitts neurons. This does not infer, however, that all switching functions can actually be realized by a single perceptron. A counter-example is the "exclusive OR," which coincides with the addition $\oplus$ on $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$, defined by

$f =$ XOR;

x_1	x_2	$f(x) = x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Lemma 1.1 *There is no perceptron that represents the XOR-function.*

Proof: Assume that for all $(x_1, x_2) \in \{0,1\}^2$,

$$x_1 \oplus x_2 = \text{sat}(w_1 x_1 + w_2 x_2 - \theta).$$

Then we have

(i) $0 \oplus 0 = 0 \quad \Rightarrow \quad \text{sat}(-\theta) = 0 \quad \Rightarrow \quad -\theta < 0$

(ii) $0 \oplus 1 = 1 \quad \Rightarrow \quad \text{sat}(w_2 - \theta) = 1 \quad \Rightarrow \quad w_2 - \theta \geq 0$

(iii) $1 \oplus 0 = 1 \quad \Rightarrow \quad \text{sat}(w_1 - \theta) = 1 \quad \Rightarrow \quad w_1 - \theta \geq 0$

(iv) $1 \oplus 1 = 0 \quad \Rightarrow \quad \text{sat}(w_1 + w_2 - \theta) = 0 \quad \Rightarrow \quad w_1 + w_2 - \theta < 0.$

The addition of (ii) and (iii) yields

$$w_1 + w_2 - 2\theta \geq 0.$$

On the other hand the addition of (i) and (iv) gives

$$w_1 + w_2 - 2\theta < 0,$$

which is a contradiction. $\square$

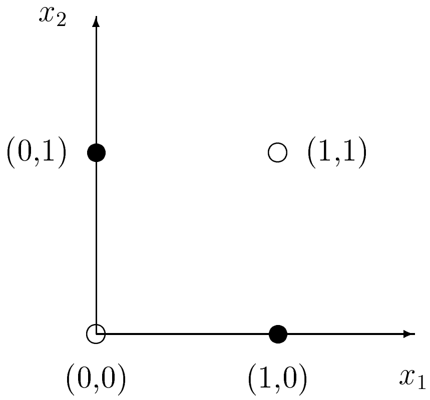


Figure 1.2: *The geometrical interpretation of the XOR-problem. There exists no affine hyperplane* $\langle w, x\rangle - \theta = 0$ *that separates* $\{(1,0),(0,1)\}$ *from* $\{(0,0),(1,1)\}$.

The realization of Boolean functions by a perceptron is closely linked to the affine separation of sets, a simple geometric concept that will be discussed now. This yields a criterion for deciding whether a given Boolean function can be realized by a McCulloch-Pitts neuron. In fact, it will turn out that the number of realizable Boolean functions is actually quite small.

1.2 Affine Separation

Definition 1.3 A set $\mathcal{A} \subset \mathbb{R}^n$ is called **affinely separable from** $\mathcal{B} \subset \mathbb{R}^n$ if there exists $(w, \theta) \in \mathbb{R}^{n+1}$ with

$$\langle w, x\rangle - \theta \begin{cases} \geq 0 & \text{for } x \in \mathcal{A} \\ < 0 & \text{for } x \in \mathcal{B}. \end{cases} \tag{1.2}$$

Then $H = \{x \in \mathbb{R}^n : \langle w, x\rangle = \theta\}$ is called a **separating hyperplane**. If the inequalities in (1.2) are both strict, we say that $\mathcal{A}$ and $\mathcal{B}$ are **strictly** affinely separable (from each other). If $\theta = 0$, we say that $\mathcal{A}$ is **linearly separable from** $\mathcal{B}$, or $\mathcal{A}$ and $\mathcal{B}$ are **strictly linearly separable**, respectively.

It is easy to see that if a set is affinely separable from a compact set, then the two sets are strictly affinely separable. Later on, we restrict to that case, and then the distinction between the two concepts becomes unnecessary.

Now we formulate a necessary and sufficient condition for $\{0,1\}$-valued functions to be realizable by McCulloch-Pitts neurons. Note that in the following, X is an arbitrary subset of $\mathbb{R}^n$.

Theorem 1.2 *Let $X \subseteq \mathbb{R}^n$. A function $f : X \to \{0,1\}$ can be represented by a perceptron if and only if X_+ is affinely separable from X_-, where*

$$X_+ := f^{-1}(1) \subseteq X \quad \textit{and} \quad X_- := f^{-1}(0) \subseteq X.$$

Proof: We have

$$f(x) = \mathrm{sat}(\langle w, x\rangle - \theta)$$

for some $(w, \theta) \in \mathbb{R}^{n+1}$ if and only if

$$\langle w, x\rangle - \theta \begin{cases} \geq 0 & \text{for } x \in X_+ \\ < 0 & \text{for } x \in X_-. \end{cases}$$

This yields the desired result. □

The degree of difficulty in separating arbitrary finite disjoint sets is measured by means of the so-called **capacity**. In order to decide whether two given sets are affinely separable, the concept of convexity plays a decisive role.

Definition 1.4 A set $\mathcal{A} \subset \mathbb{R}^n$ is called **convex** if for any $x, y \in \mathcal{A}$, the complete line segment between x and y is also contained in $\mathcal{A}$, that is,

$$x, y \in \mathcal{A} \text{ and } 0 \leq \lambda \leq 1 \quad \Rightarrow \quad \lambda x + (1-\lambda) y \in \mathcal{A}.$$

The following theorem gives sufficient conditions for the separability of convex sets, proofs can be found in [12, 16].

Theorem 1.3 *Let $\mathcal{A}$ and $\mathcal{B}$ be two non-empty, disjoint convex subsets of $\mathbb{R}^n$.*

1. *If $\mathcal{B}$ is open, then $\mathcal{A}$ is affinely separable from $\mathcal{B}$.*
2. *If one of the sets is closed and the other is compact, then they are strictly affinely separable.*

For arbitrary sets $\mathcal{A}$, it is useful to consider their convex supersets, the smallest of which is called the convex hull of $\mathcal{A}$.

Definition 1.5 Let $\mathcal{A} \subset \mathbb{R}^n$ be an arbitrary set. The **convex hull** $\operatorname{conv}(\mathcal{A})$ of $\mathcal{A}$ is defined as the intersection of all convex subsets of $\mathbb{R}^n$ that contain $\mathcal{A}$.

Thus $\operatorname{conv}(\mathcal{A})$ is the smallest convex subset of $\mathbb{R}^n$ containing $\mathcal{A}$. Moreover,

$$\operatorname{conv}(\mathcal{A}) = \left\{ \sum_{i=1}^{k} \lambda_i x_i \;:\; k \in \mathbb{N}, x_i \in \mathcal{A}, \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}. \tag{1.3}$$

Note that the convex hull of an open (compact) set in $\mathbb{R}^n$ is again open (compact) [12], whereas the convex hull of a closed set is not necessarily closed. For a counter-example, consider the closed set

$$\mathcal{A} = \{(x,0) \;:\; x \in \mathbb{R}\} \cup \{(0,1)\} \subset \mathbb{R}^2,$$

whose convex hull is

$$\operatorname{conv}(\mathcal{A}) = \{(x,y) \;:\; 0 \leq y < 1\} \cup \{(0,1)\}.$$

A necessary and sufficient criterion for the separability of non-convex sets is illustrated by the following corollary:

Corollary 1.4

1. *Let $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^n$ be non-empty, and let $\mathcal{B}$ be open. Then, $\mathcal{A}$ is affinely separable from $\mathcal{B}$ if and only if*

$$\operatorname{conv}(\mathcal{A}) \cap \operatorname{conv}(\mathcal{B}) = \emptyset.$$

2. *Let $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^n$ be non-empty, with $\mathcal{A}$ closed and $\mathcal{B}$ compact. Let $\overline{\operatorname{conv}(\mathcal{A})}$ denote the closure of* $\operatorname{conv}(\mathcal{A})$. *Then, $\mathcal{A}$ and $\mathcal{B}$ are strictly affinely separable if and only if*

$$\overline{\operatorname{conv}(\mathcal{A})} \cap \operatorname{conv}(\mathcal{B}) = \emptyset.$$

Proof: (i) Let $\mathcal{A}$ be affinely separable from $\mathcal{B}$. Then there are half-spaces $H^+ = \{x \;:\; \langle x, w\rangle \geq \theta\}$ and $H^- = \{x \;:\; \langle x, w\rangle < \theta\}$, with $\mathcal{A} \subset H^+$ and $\mathcal{B} \subset H^-$. Since H^+ and H^- are convex, $\operatorname{conv}(\mathcal{A}) \subset H^+$ and $\operatorname{conv}(\mathcal{B}) \subset H^-$ and hence $\operatorname{conv}(\mathcal{A}) \cap \operatorname{conv}(\mathcal{B}) = \emptyset$. Conversely, if $\operatorname{conv}(\mathcal{A}) \cap \operatorname{conv}(\mathcal{B}) = \emptyset$, then $\operatorname{conv}(\mathcal{A})$ is affinely separable from $\operatorname{conv}(\mathcal{B})$ according to Theorem 1.3, using that the convex hull of an open set is open again. Due to $\mathcal{A} \subseteq \operatorname{conv}(\mathcal{A})$ and $\mathcal{B} \subseteq \operatorname{conv}(\mathcal{B})$, the conclusion follows.

(ii) Let $\mathcal{A}$ and $\mathcal{B}$ be strictly affinely separable. As in part (i), we have a closed and an open half-space H^+ and H^-, respectively, such that $\operatorname{conv}(\mathcal{A}) \subset H^+$ and $\operatorname{conv}(\mathcal{B}) \subset H^-$. Now

$$\overline{\operatorname{conv}(\mathcal{A})} \subset \overline{H^+} = H^+ \quad \text{and hence} \quad \overline{\operatorname{conv}(\mathcal{A})} \cap \operatorname{conv}(\mathcal{B}) = \emptyset.$$

The converse direction follows from Theorem 1.3, since the convex hull of a compact set is compact, and the closure of a convex set is convex [12]. $\square$

Separation of Finite Sets

Finite sets and their convex hulls are compact. Hence, in view of Corollary 1.4, the condition $\mathrm{conv}(\mathcal{A}) \cap \mathrm{conv}(\mathcal{B}) = \emptyset$ is equivalent to the strict affine separability of finite non-empty sets $\mathcal{A}$ and $\mathcal{B}$. When dealing with finite sets, we may therefore omit the term "strictly," and simply speak of sets that are affinely separable (from each other).

It is convenient here to change the notation: Instead of functions $f : X \to \{0, 1\}$, we consider mappings

$$f : X \to \{\pm 1\},$$

where $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^n$. Let

$$X_\pm = f^{-1}(\pm 1)$$

and suppose that X_+ and X_- are both non-empty. Let $y_i = f(x_i)$ for $1 \leq i \leq N$. The sets X_+ and X_- are (strictly) affinely separable if and only if

$$y_i(\langle w, x_i \rangle - \theta) > 0 \quad \text{for } 1 \leq i \leq N$$

for some $w \in \mathbb{R}^n$, $\theta \in \mathbb{R}$. Equivalently,

$$f(x) = \mathrm{sign}(\langle w, x \rangle - \theta) \quad \text{for all } x \in X,$$

that is, the function f is realizable by a perceptron with the modified Heaviside or sign function,

$$\mathrm{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0. \end{cases}$$

Another characterization of separability is given in the following theorem. For any two non-empty sets $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^n$, we define $\mathcal{A} - \mathcal{B} = \{a - b \ : \ a \in \mathcal{A}, b \in \mathcal{B}\}$.

Theorem 1.5 *X_+ and X_- are affinely separable if and only if*

$$0 \notin C := \mathrm{conv}(X_+ - X_-).$$

Proof: We have $\mathrm{conv}(X_+ - X_-) = \mathrm{conv}(X_+) - \mathrm{conv}(X_-)$ and therefore, $0 \notin C$ is equivalent to $\mathrm{conv}(X_+)$ and $\mathrm{conv}(X_-)$ being disjoint. □

Still equivalently, X_+ and X_- are affinely separable if and only if the associated **weight polyhedron**

$$\Gamma := \{(w, \theta) \in \mathbb{R}^{n+1} \ : \ y_i(\langle w, x_i \rangle - \theta) > 0 \quad \text{for } 1 \leq i \leq N\}$$

is non-empty. Clearly, Γ consists of all (w, θ) that define hyperplanes which separate X_+ from X_-. In the following two sections, we treat the problem of finding an arbitrary element of Γ (in a finite number of computation steps), and we address the problem of selecting an *optimal* separating hyperplane.

1.3 Perceptron Learning Algorithm

If the finite sets X_+ and X_- are to be separated, we are interested in finding a concrete separating hyperplane. A method of constructing such a hyperplane is given by the Perceptron Learning (PL) Algorithm. The strict affine separability of the sets X_+ and X_- is assumed throughout this section. We start by reformulating the task.

PL Problem: Two finite sets $X_+, \ X_- \subset \mathbb{R}^n$ with $\operatorname{conv}(X_+) \cap \operatorname{conv}(X_-) = \emptyset$ shall be affinely separated by a perceptron. In other words: Find $(w, \theta) \in \mathbb{R}^{n+1}$ such that

$$\operatorname{sign}(\langle w, x\rangle - \theta) = \begin{cases} 1 & \text{if } x \in X_+ \\ -1 & \text{if } x \in X_-. \end{cases}$$

It is convenient to eliminate the threshold θ: Replace (x, w) by $(\hat{x}, \hat{w})$, where

$$\hat{x} = \begin{bmatrix} x \\ 1 \end{bmatrix} \quad \text{and} \quad \hat{w} = \begin{bmatrix} w \\ -\theta \end{bmatrix}.$$

Hence, the new input domain is $\mathbb{R}^n \times \{1\}$ and

$$\operatorname{sign}(\langle \hat{w}, \hat{x}\rangle) = \operatorname{sign}(\langle w, x\rangle - \theta).$$

Instead of the affine separation of X_+ and X_- in $\mathbb{R}^n$, we now face the problem of linear separation of $\hat{X}^+$ and $\hat{X}^-$ in $\mathbb{R}^{n+1}$. Let further $m := n + 1$ and

$$\hat{X} := \hat{X}_+ \cup \{-x \ : \ x \in \hat{X}_-\} \subset \mathbb{R}^m.$$

Hence, the PL problem is equivalent to the following:

Reformulated PL Problem: Find a vector $\hat{w} \in \mathbb{R}^m$ with

$$\langle \hat{w}, \xi\rangle > 0 \quad \text{for all } \xi \in \hat{X}. \tag{1.4}$$

If $\hat{X} = \{\xi_1, \dots, \xi_N\}$, define $A := [\xi_1, \dots, \xi_N]^T \in \mathbb{R}^{N\times m}$. Then (1.4) reads

$$A\hat{w} > 0,$$

where the inequality is to be understood component-wise. As will be illustrated, the Perceptron Learning Algorithm yields such a separating vector $\hat{w}$ after finitely many iteration steps.

Definition 1.6 Let $\hat{X} \subset \mathbb{R}^m$ be a finite set. A sequence

$$u : \mathbb{N} \to \hat{X}$$

is called a **training sequence** for $\hat{X}$ if for all $\xi \in \hat{X}$ and all $k_0 \in \mathbb{N}$, there is a $k \geq k_0$ with $u(k) = \xi$.

That is, each pattern $\xi \in \hat{X}$ appears infinitely often in a training sequence u for $\hat{X}$.

Perceptron Learning Algorithm: Let $A := (\xi_1, \ldots, \xi_N)^T \in \mathbb{R}^{N\times m}$ be given. Let u be a training sequence for $\{\xi_1, \ldots, \xi_N\}$, and let $\varphi : \mathbb{N} \to \mathbb{R}$ a sequence with

$$0 < \inf_{k\in\mathbb{N}} \varphi(k) \leq \sup_{k\in\mathbb{N}} \varphi(k) < \infty.$$

Step 0: Choose $w(0) \in \mathbb{R}^m$ and put $k := 0$.

Step 1: Set

$$w(k+1) = \begin{cases} w(k) & \text{if } u(k)^T w(k) > 0 \\ w(k) + \varphi(k)u(k) & \text{if } u(k)^T w(k) \leq 0. \end{cases} \tag{1.5}$$

Step 2: If $Aw(k+1) > 0$ applies component-wise, an admissible separation has been found and the algorithm stops. Otherwise, augment k by one and go to Step 1.

The following theorem shows that the Perceptron Learning Algorithm stops after finitely many steps.

Theorem 1.6 (Perceptron Convergence Theorem) *Let a matrix*

$$A = (\xi_1, \ldots, \xi_N)^T \in \mathbb{R}^{N\times m}$$

be given and suppose that there exists $w^* \in \mathbb{R}^m$ such that

$$Aw^* > 0$$

is fulfilled component-wise. Let u and φ be as described above. Let $w : \mathbb{N} \to \mathbb{R}^m$ be the (infinite) sequence of weight vectors that is determined by the recursion (1.5) from an arbitrary initial value $w(0) \in \mathbb{R}^m$. Then w becomes stationary, i.e., there is a $k_L \in \mathbb{N}$ such that for all $k \geq k_L$:

$$w(k) = w(k_L) =: \hat{w},$$

and $A\hat{w} > 0$ component-wise.

This signifies that the sequence of weights produced according to (1.5) converges to a solution of the PL problem, in finitely many steps. In view of Step 2 above, this implies that the PL Algorithm stops after finitely many iterations.

Proof: By assumption, $Aw^* > 0$, hence

$$\alpha := \inf_{k\in\mathbb{N}} \varphi(k) \cdot \min_{j\in\{1,\ldots,N\}} ({\xi_j}^T w^*) > 0.$$

Define

$$\beta := \sup_{k\in\mathbb{N}} \varphi(k) \cdot \max_{j\in\{1,\ldots,N\}} \|\xi_j\|_2 \quad \text{and} \quad \overline{w} = \frac{\beta^2}{\alpha} w^*.$$

Consider an iteration step in which the weight vector is properly updated, i.e. suppose $w(k+1) \neq w(k)$ and thus $u(k)^T w(k) \leq 0$. Then we have

$$\begin{aligned} \|w(k+1) - \overline{w}\|_2^2 &= \|w(k) + \varphi(k)u(k) - \overline{w}\|_2^2 \\ &= \|w(k) - \overline{w}\|_2^2 + 2\varphi(k)u(k)^T(w(k) - \overline{w}) + \varphi(k)^2\|u(k)\|_2^2 \\ &\leq \|w(k) - \overline{w}\|_2^2 - 2\varphi(k)u(k)^T\overline{w} + \beta^2. \end{aligned}$$

Furthermore $0 < \alpha \leq \varphi(k)u(k)^T w^*$, and therefore

$$\varphi(k)u(k)^T\overline{w} = \beta^2 \frac{\varphi(k)u(k)^T w^*}{\alpha} \geq \beta^2.$$

With the above estimate, we get

$$\|w(k+1) - \overline{w}\|_2^2 \leq \|w(k) - \overline{w}\|_2^2 - \beta^2.$$

Set $k_1 := 0$ and let $0 < k_2 < k_3 \ldots$ be the sequence of integers with

$$w(k_i - 1) \neq w(k_i).$$

Let $(w(k_1), w(k_2), \ldots)$ be the corresponding subsequence of w. Then

$$0 \leq \|w(k_{j+1}) - \overline{w}\|_2^2 \leq \|w(0) - \overline{w}\|_2^2 - j\beta^2$$

and hence

$$j \leq \frac{\|w(0) - \overline{w}\|_2^2}{\beta^2}.$$

We conclude that the length of the subsequence, and hence the number of proper updating steps is bounded. Let L denote the length of the subsequence. Then $w(k) = w(k_L)$ for all $k \geq k_L$. As u is a training sequence for the rows of A, this implies $Aw(k_L) > 0$. $\square$

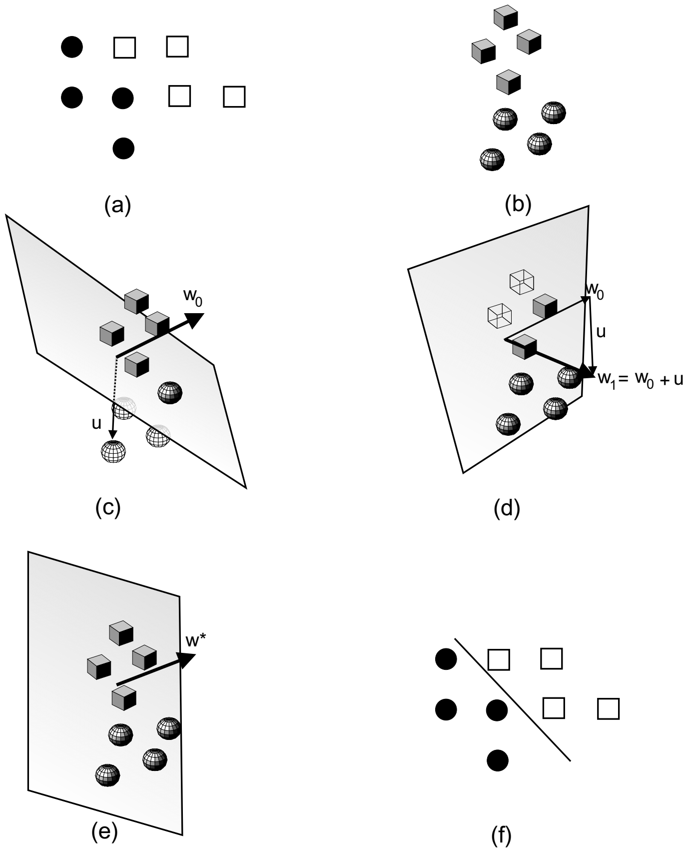


Figure 1.3: *Schematic course of the perceptron algorithm (cf. Example 1.3). The sets in (a), consisting of circles (X_-) and squares (X_+) shall be separated affinely. Equivalently, a linear half space is sought that contains all points (cubes and spheres) in (b). For this, we begin with an arbitrary half space $H_0^+ = \{x : \langle x, w(0)\rangle > 0\}$, and a point u that does not lie in H^+ is chosen (cf. (c)). A new half space $H_1^+ = \{x : \langle x, w(0)+u\rangle > 0\}$ emerges (cf. (d)). The Perceptron Convergence Theorem guarantees that a half space H_*^+ which contains all the points (cf. (e)), is reached by continuing this procedure in the case of the separability of the sets. After an inverse transformation (f), an affine separation of the original sets (a) is obtained.*

Example 1.7 An affine separation of the sets $X_+ = \{(0,0),(0,1),(1,0),(-1,1)\}$ and $X_- = \{(-2,1),(-2,0),(-1,0),(-1,-1)\}$ is sought (cf. squares and circles in Figure 1.3 (a)).

After eliminating the thresholds and reflecting the set X_- at the origin, we obtain the set

$$\begin{aligned}\hat{X} = \hat{X}^+ \cup \{-x : x \in \hat{X}_-\} &= \{(0,0,1),(0,1,1),(1,0,1),(-1,1,1),(2,-1,-1),\\ &\quad (2,0,-1),(1,0,-1),(1,1,-1)\}.\end{aligned}$$

Now a weight vector $w \in \mathbb{R}^3$ is sought with

$$\langle w, x \rangle > 0 \text{ for all } x \in \hat{X}.$$

For this, the Perceptron Algorithm is started with the randomly chosen weight vector $w(0) = (0, 2, 1)$ and for simplicity, $\varphi \equiv 1$. The hyperplane

$$H = \{x \in \mathbb{R}^3 \ : \ \langle w(0), x \rangle = 0\}$$

which is orthogonal to $w(0)$ separates the correctly classified points from the falsely classified points $(2, -1, -1), (2, 0, -1)$ and $(1, 0, -1)$ (Figure 1.3 (c)). If a falsely classified point emerges in the training sequence (e.g. $u = (2, 0, -1)$), the weight vector is corrected accordingly:

$$w(1) = w(0) + u = (0, 2, 1) + (2, 0, -1) = (2, 2, 0).$$

The algorithm only stops if a w is found with

$$\langle w, x \rangle > 0$$

for all $x \in \hat{X}$. In this example, this is obtained after 5 iteration steps with $w = (3, 3, 2)$ (Figure 1.3 (e)). Then, the corresponding affine separation of X_+ and X_- is

$$3x_1 + 3x_2 + 2 = 0$$

and the corresponding perceptron has the weights $w = (3, 3)^T$ and the threshold $\theta = -2$ (Figure 1.3 (f)).

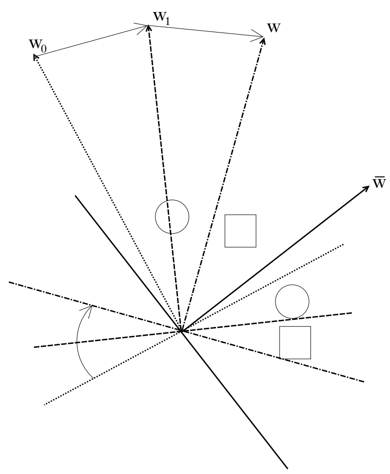


Figure 1.4: *Simplified representation of the Perceptron Algorithm with $\theta = 0$.*

Remarks

1. The scaling of w^* by $\frac{\beta^2}{\alpha}$ in the proof of the Perceptron Algorithm ensures that the actualized weight vector w_{k+1} lies closer to $\overline{w}$ than w_k. Generally however, w does not correspond to $\overline{w}$ at the end of the algorithm since the algorithm usually finds an admissible solution beforehand (cf. Figure 1.4).

2. Input vectors $u(k)$ with a large norm have a greater influence on the updating step $w(k+1) = w(k) + \varphi(k)u(k)$ than vectors of small norm. Hence, it is useful to normalize the input vectors ($\varphi(k) = \|u(k)\|_2^{-1}$) so that a faster convergence might be obtained.

3. The fractional correction rule suggests
$$\varphi(k) = \frac{\lambda\,|u(k)^T w(k)|}{\|u(k)\|_2^2}.$$
One can show that for $0 < \lambda < 2$, the resulting algorithm converges even if the data sets are not affinely separable [3].

4. The norm of the weight vector tends to increase during the algorithm. Hence, input vectors have, at first, greater influence on the location of the hyperplane. After the initial rough corrections, the algorithm becomes more sensitive with increasing $\|w(k)\|_2$.

5. Instead of choosing a random initial weight vector, we can always start with a deterministic initial value such as, for example, the centroid of $\hat{X}$.

1.4 Optimal Separation

Suppose that X_+ and X_- are finite non-empty subsets of $\mathbb{R}^n$ that are affinely separable. Let $X = X_+ \cup X_- = \{x_1, \ldots, x_N\}$ and let $y_i = \pm 1$ for $x_i \in X_\pm$. The set of separating hyperplanes is uniquely determined by the weight polyhedron

$$\Gamma = \{(w,\theta) \in \mathbb{R}^{n+1} \;:\; y_i(\langle w, x_i\rangle - \theta) > 0 \quad \text{for } 1 \le i \le N\} \neq \emptyset. \tag{1.6}$$

We define a functional that measures the quality of separation achieved by a particular separating hyperplane. Intuitively, this depends on the distance between $X = X_+ \cup X_-$ and the separating hyperplane. If this distance is large, we may expect a certain robustness of the separation with respect to noisy data.

Definition 1.7 *For $(w,\theta) \in \mathbb{R}^{n+1}$, $w \neq 0$, let*

$$\rho(w,\theta) := \min_{1\le i\le N} \rho_i(w,\theta), \quad \textit{where } \rho_i(w,\theta) = \frac{y_i(\langle w, x_i\rangle - \theta)}{\|w\|} \tag{1.7}$$

be the **separation margin** *of (w,θ).*

This notion has the following geometrical interpretation: Let $x_i \in X$ and let

$$H = \{x \,:\, \langle w, x\rangle - \theta = 0\}$$

be the hyperplane defined by $(w, \theta) \in \mathbb{R}^{n+1}$. The distance of x_i from H is

$$\operatorname{dist}(x_i, H) = \min_{x \in H} \|x_i - x\|.$$

It is easy to show that

$$\operatorname{dist}(x_i, H) = |\rho_i(w, \theta)|.$$

Recall that H separates X_+ from X_- if and only if all the $\rho_i(w, \theta)$ are positive. In that case,

$$\rho(w, \theta) = \min_{1 \le i \le N} \operatorname{dist}(x_i, H) =: \operatorname{dist}(X, H),$$

that is, $\rho(w, \theta)$ equals the distance of X from H, and thus measures the quality of separation. On the other hand, if H does not separate X_+ and X_-, then at least one of the $\rho_i(w, \theta)$ is non-positive. This yields the inequality

$$\rho(w, \theta) \le \operatorname{dist}(X, H)$$

which is true for all $(w, \theta) \in \mathbb{R}^{n+1}$; and equality holds if and only if $(w, \theta) \in \Gamma$, that is, if H is a separating hyperplane.

Therefore, the optimal separation problem is to maximize $\rho(w, \theta)$ by choice of $(w, \theta) \in \mathbb{R}^{n+1}$. Note that the consideration above shows that it is *not* necessary to restrict to $(w, \theta) \in \Gamma$. The main result of this section is stated next.

Theorem 1.8 *Suppose that the non-empty finite sets X_+, X_- are affinely separable. Then the optimal separation margin*

$$\rho := \sup\{\rho(w, \theta) \,:\, (w, \theta) \in \mathbb{R}^{n+1}, w \neq 0\} \tag{1.8}$$

is finite, and the supremum is achieved at some (w^*, θ^*). Moreover, the corresponding optimal hyperplane

$$H = \{x \in \mathbb{R}^n \,:\, \langle w^*, x\rangle - \theta^* = 0\}$$

is uniquely determined, and we have

$$\rho = \frac{1}{2} \operatorname{dist}(C_+, C_-),$$

where $C_\pm := \operatorname{conv}(X_\pm)$.

In order to prove Theorem 1.8, several preliminary steps are needed. We start by characterizing the distance between the compact convex sets C_+ and C_-.

Lemma 1.9 *Let X_+, X_-, and C_+, C_- be as in Theorem 1.8 and let $C := \operatorname{conv}(X_+ - X_-) = C_+ - C_-$. Due to the separability assumption, we have $0 \notin C$ according to Theorem 1.5. Then*

$$\operatorname{dist}(C_+, C_-) = \operatorname{dist}(C, 0) = \max_{\|v\|=1} \min_{z \in C} \langle v, z \rangle.$$

Proof: For any v with $\|v\| = 1$, we have $\langle v, z \rangle \leq \|z\|$ due to the Cauchy–Schwarz inequality and hence

$$\min_{z \in C} \langle v, z \rangle \leq \min_{z \in C} \|z\| = \operatorname{dist}(C, 0).$$

Thus

$$\max_{\|v\|=1} \min_{z \in C} \langle v, z \rangle \leq \operatorname{dist}(C, 0).$$

For the converse direction, note that C is compact, and $0 \notin C$. Hence there exists $0 \neq z^* \in C$ with

$$\|z^*\| = \min_{z \in C} \|z\| = \operatorname{dist}(C, 0).$$

For any $z \in C$ and $0 \leq \lambda \leq 1$, the convexity of C implies that $\lambda z + (1-\lambda) z^* \in C$. Hence

$$\|z^*\|^2 \leq \|\lambda z + (1-\lambda) z^*\|^2 = \|z^*\|^2 + 2\lambda \langle z^*, z - z^* \rangle + \lambda^2 \|z - z^*\|^2,$$

which implies that $0 \leq 2\langle z^*, z - z^* \rangle + \lambda \|z - z^*\|^2$ for all $z \in C$ and $0 < \lambda \leq 1$. This can only be true if $0 \leq \langle z^*, z - z^* \rangle$ for all $z \in C$. Hence

$$\|z^*\| \leq \left\langle \frac{z^*}{\|z^*\|}, z \right\rangle \tag{1.9}$$

holds for all $z \in C$, and thus there exists a vector v with $\|v\| = 1$ and

$$\operatorname{dist}(C, 0) \leq \min_{z \in C} \langle v, z \rangle.$$

However, this implies

$$\operatorname{dist}(C, 0) \leq \max_{\|v\|=1} \min_{z \in C} \langle v, z \rangle$$

as desired. □

The next lemma characterizes the weight polyhedron.

Lemma 1.10 *Let X_+, X_- be as usual, with $C_\pm = \mathrm{conv}(X_\pm)$ and $C = C_+ - C_- = \mathrm{conv}(X_+ - X_-)$. Let Γ be defined as in (1.6). Consider its projection onto the first n components*

$$W := \{w \in \mathbb{R}^n \ : \ \exists\theta \in \mathbb{R} : (w, \theta) \in \Gamma\}.$$

Then

$$W = \{w \in \mathbb{R}^n \ : \ \langle w, z\rangle > 0 \textit{ for all } z \in C\}$$

and for any $w \in W$,

$$\theta_w := \frac{1}{2}\left(\min_{x\in C_+}\langle w, x\rangle + \max_{y\in C_-}\langle w, y\rangle\right) \tag{1.10}$$

is such that $(w, \theta_w) \in \Gamma$. Moreover, for any $0 \neq w \in \mathbb{R}^n$,

$$\rho(w, \theta_w) = \frac{1}{2}\min_{z\in C}\left\langle \frac{w}{\|w\|}, z\right\rangle. \tag{1.11}$$

Proof: By definition,

$$w \in W \quad \Leftrightarrow \quad \exists\theta : \begin{cases} \langle w, x\rangle - \theta > 0 & \text{for all } x \in X_+ \\ \langle w, y\rangle - \theta < 0 & \text{for all } y \in X_-. \end{cases}$$

Thus

$$w \in W \quad \Rightarrow \quad \langle w, z\rangle > 0 \text{ for all } z = x - y \in X_+ - X_-$$

and then $\langle w, z\rangle > 0$ holds for all $z \in C$, due to convexity.

For the converse, let $w \in \mathbb{R}^n$ and let θ_w be as defined above. For $x \in X_+$, we have

$$\begin{aligned}
\langle w, x\rangle - \theta_w &= \langle w, x\rangle - \frac{1}{2}\min_{x\in C_+}\langle w, x\rangle - \frac{1}{2}\max_{y\in C_-}\langle w, y\rangle \\
&\geq \frac{1}{2}\min_{x\in C_+}\langle w, x\rangle - \frac{1}{2}\max_{y\in C_-}\langle w, y\rangle \\
&= \frac{1}{2}\min_{z\in C}\langle w, z\rangle. \qquad (1.12)
\end{aligned}$$

Similarly, for $y \in X_-$,

$$\begin{aligned}
\langle w, y\rangle - \theta_w &= \langle w, y\rangle - \frac{1}{2}\min_{x\in C_+}\langle w, x\rangle - \frac{1}{2}\max_{y\in C_-}\langle w, y\rangle \\
&\leq -\frac{1}{2}\min_{x\in C_+}\langle w, x\rangle + \frac{1}{2}\max_{y\in C_-}\langle w, y\rangle \\
&= -\frac{1}{2}\min_{z\in C}\langle w, z\rangle. \qquad (1.13)
\end{aligned}$$

Now if w is such that $\langle w, z\rangle > 0$ for all $z \in C$, we have $\min_{z\in C}\langle w, z\rangle > 0$ due to the compactness of C. Then (1.12) and (1.13) imply that $(w, \theta_w) \in \Gamma$, and in particular, $w \in W$.

Note that equality holds in (1.12), (1.13) for at least one $x \in X_+$, $y \in X_-$. This is due to the fact that $C_\pm$ are compact and convex sets. Hence $\langle w, \cdot\rangle$ achieves its minimum and maximum in one of the extreme points of $C_\pm$ [12], and these are contained in the sets $X_\pm$, respectively. Thus

$$\min_{x\in X_+}(\langle w, x\rangle - \theta_w) = \min_{y\in X_-}(-\langle w, y\rangle + \theta_w) = \frac{1}{2}\min_{z\in C}\langle w, z\rangle.$$

Let $X = X_+ \cup X_- = \{x_1, \ldots, x_N\}$ and let $y_i = \pm 1$ for $x_i \in X_\pm$. For any $0 \neq w \in \mathbb{R}^n$,

$$\begin{aligned}
\rho(w, \theta_w)\|w\| &= \min_{1\leq i\leq N} y_i(\langle w, x_i\rangle - \theta_w) \\
&= \min\left(\min_{x\in X_+}(\langle w, x\rangle - \theta_w)\,,\ \min_{y\in X_-}(-\langle w, y\rangle + \theta_w)\right) \\
&= \frac{1}{2}\min_{z\in C}\langle w, z\rangle.
\end{aligned}$$

This proves (1.11). □

Now we are able to prove part of Theorem 1.8.

Corollary 1.11 *The optimal separation margin ρ defined in (1.8) satisfies*

$$\rho = \frac{1}{2}\,\mathrm{dist}(C_+, C_-).$$

In particular, it is a finite number.

Proof: For any $0 \neq w \in \mathbb{R}^n$, we have

$$\begin{aligned}
\rho(w, \theta)\|w\| &= \min\left(\min_{x\in X_+}(\langle w, x\rangle - \theta)\,,\ \min_{y\in X_-}(-\langle w, y\rangle + \theta)\right) \\
&= \min\left(\min_{x\in C_+}(\langle w, x\rangle - \theta)\,,\ \min_{y\in C_-}(-\langle w, y\rangle + \theta)\right).
\end{aligned}$$

The second equality follows again from the fact that $\langle w, \cdot\rangle$ achieves its minimum in one of the extreme points of $C_\pm$. For any real numbers a, b, we have $\min(a, b) \leq \frac{1}{2}(a + b)$, and equality holds only if $a = b$. Using this, we obtain

$$\rho(w, \theta)\|w\| \leq \frac{1}{2}\min_{x\in C_+, y\in C_-}\langle w, x - y\rangle$$

and thus

$$\rho(w,\theta) \le \frac{1}{2}\min_{z\in C}\left\langle \frac{w}{\|w\|}, z\right\rangle .$$

Equality holds only if

$$\min_{x\in C_+}(\langle w,x\rangle - \theta) = \min_{y\in C_-}(-\langle w,y\rangle + \theta)$$

and in view of (1.10), this signifies that $\theta = \theta_w$. We conclude that for any $0 \neq w \in \mathbb{R}^n$,

$$\sup_{\theta\in\mathbb{R}}\rho(w,\theta) = \rho(w,\theta_w) = \frac{1}{2}\min_{z\in C}\left\langle \frac{w}{\|w\|}, z\right\rangle . \tag{1.14}$$

This yields the desired result, since

$$\begin{array}{rclcl}\rho & = & \sup_{w\in\mathbb{R}^n}\sup_{\theta\in\mathbb{R}}\rho(w,\theta) & = & \frac{1}{2}\sup_{w\in\mathbb{R}^n}\min_{z\in C}\left\langle \frac{w}{\|w\|}, z\right\rangle \\ & = & \frac{1}{2}\max_{\|v\|=1}\min_{z\in C}\langle v,z\rangle & = & \frac{1}{2}\operatorname{dist}(C_+,C_-)\end{array}$$

according to Lemma 1.9. □

So far, we have shown that the optimal separation margin is

$$\rho = \frac{1}{2}\operatorname{dist}(C,0) = \frac{1}{2}\min_{z\in C}\|z\|$$

where C is a compact convex set. We show below that this minimization problem possesses a unique solution. This leads to the construction of the unique optimal separating hyperplane, thus completing the proof of Theorem 1.8.

Lemma 1.12 *Let $C \subset \mathbb{R}^n$ be a non-empty closed convex set with $0 \notin C$. Then there exists a unique $z^* \in C$ with*

$$\|z^*\| = \min_{z\in C}\|z\| = \operatorname{dist}(C,0).$$

Define

$$C^{\#} = \{w \in \mathbb{R}^n \ :\ \langle w,z\rangle \ge c \textit{ for all } z \in C\}, \tag{1.15}$$

where $c > 0$. Then $C^{\#}$ is also non-empty, closed and convex, and $0 \notin C^{\#}$. Thus there exists a unique $w^* \in C^{\#}$ with

$$\|w^*\| = \min_{w\in C^{\#}}\|w\| = \operatorname{dist}(C^{\#},0)$$

and

$$w^* = \frac{cz^*}{\|z^*\|^2} \quad \textit{and} \quad z^* = \frac{cw^*}{\|w^*\|^2}.$$

Proof: Let $\zeta \in C$ be arbitrary. Define $B := \{x \in \mathbb{R}^n \ : \ \|x\| \leq \|\zeta\|\}$. Clearly

$$\inf_{z \in C} \|z\| = \inf_{z \in B \cap C} \|z\|$$

and since $B \cap C$ is compact, the infimum is achieved at some $z^* \in C$. For uniqueness, suppose that z_1^* and z_2^* are two solutions. Due to convexity, also $\frac{1}{2}(z_1^* + z_2^*) \in C$. It is easy to see that if $z_1^* \neq z_2^*$, then

$$\|\tfrac{1}{2}(z_1^* + z_2^*)\| < \|z_1^*\| = \|z_2^*\| = \min_{z \in C} \|z\|,$$

which is a contradiction. Hence we must have $z_1^* = z_2^*$. An argument from the proof of Lemma 1.9 shows that

$$\left\langle \frac{cz^*}{\|z^*\|^2}, z \right\rangle \geq c \quad \text{for all } z \in C$$

and hence, $C^{\#}$ is not empty. The remaining properties of $C^{\#}$ stated above are obvious. This guarantees the unique existence of w^*. Finally, for any $w \in C^{\#}$ and $z \in C$, we have

$$\|w\| \, \|z\| \geq \langle w, z \rangle \geq c.$$

In particular, $\|w\| \geq c\|z^*\|^{-1}$ for all $w \in C^{\#}$. Set

$$w^* := \frac{cz^*}{\|z^*\|^2},$$

then $\|w^*\| = c\|z^*\|^{-1}$ which implies that $\|w^*\|$ is the global minimum. $\square$

Corollary 1.13 *Let X_+, X_- and $C = \operatorname{conv}(X_+ - X_-)$ be as usual. Let $C^{\#}$ be defined as in (1.15) and let w^* be the unique solution of*

$$\|w^*\| = \min_{w \in C^{\#}} \|w\|.$$

Let θ_{w^*} be defined as in (1.10). Then

$$H^* = \{x \in \mathbb{R}^n \ : \ \langle w^*, x \rangle - \theta_{w^*} = 0\}$$

is the unique optimal separating hyperplane.

Proof: In view of Lemma 1.10, it is clear that $(w^*, \theta_{w^*}) \in \Gamma$, that is, H^* is a separating hyperplane. Moreover, it is optimal, since

$$\langle w^*, z \rangle \geq c \quad \text{for all } z \in C$$

and

$$\langle w^*, z^* \rangle = c, \quad \text{where } z^* = \frac{cw^*}{\|w^*\|^2}.$$

Thus

$$\rho(w^*, \theta_{w^*}) = \frac{1}{2} \min_{z \in C} \left\langle \frac{w^*}{\|w^*\|}, z \right\rangle = \frac{1}{2} \left\langle \frac{w^*}{\|w^*\|}, z^* \right\rangle = \frac{1}{2}\|z^*\| = \rho.$$

For uniqueness, let (w, θ) define another optimal separating hyperplane, that is, let $\rho(w, \theta) = \rho$. According to (1.14),

$$\rho(w, \theta) \le \rho(w, \theta_w) \quad \text{for all } \theta,$$

and equality holds only if $\theta = \theta_w$. Hence we may assume without loss of generality that $\theta = \theta_w$. Thus we obtain from (1.11)

$$\rho(w, \theta_w) = \frac{1}{2} \min_{z \in C} \left\langle \frac{w}{\|w\|}, z \right\rangle = \rho = \frac{1}{2} \min_{z \in C} \|z\| = \frac{1}{2}\|z^*\|.$$

Then

$$\|z^*\| \le \left\langle \frac{w}{\|w\|}, z^* \right\rangle \le \|z^*\|,$$

where the second inequality is due to Cauchy–Schwarz. Hence w and z^* are related via

$$w = \alpha z^* \quad \text{for some } \alpha > 0.$$

We conclude that the (w, θ) for which the optimal separation margin ρ is achieved are unique up to scalar positive multiples. □

1.5 Optimization Techniques for Optimal Separation

In the previous section, we have seen that the optimal separating hyperplane for the finite data set $X = X_+ \cup X_- = \{x_1, \dots, x_N\}$ is determined by minimizing $\|z\|$ over the compact convex set

$$C = \operatorname{conv}(X_+ - X_-) = \operatorname{conv}\{z_1, \dots, z_M\} \subset \mathbb{R}^n.$$

The vectors z_i are simply obtained by taking all differences $x - y$, where $x \in X_+$ and $y \in X_-$. According to Lemma 1.12, there exists a unique $z^* \in C$ with

$$\|z^*\| = \min_{z \in C} \|z\|.$$

The idea of the following **line search algorithm** is to consider, for a given point $z \in C$, the line segments connecting z with any of the "extreme" points $z_1, \ldots, z_M$ and to minimize $\|z\|$ along these (actually, we only have $\operatorname{ext}(C) \subseteq \{z_1, \ldots, z_M\}$, where $\operatorname{ext}(C)$ denotes the set of extreme points of C [12], but this will not be crucial in the following). Iteratively, this produces a sequence of elements of C that will eventually converge to z^*, as shown below.

Step 0: Choose $\zeta(0) := \zeta(0,0) \in C$ and set $k := 0$.

Step 1: For $j = 1, \ldots, M$: Minimize

$$\|tz_j + (1-t)\zeta(k, j-1)\|$$

over $t \in [0,1]$. Let t^* be such that the minimum is achieved, set

$$\zeta(k,j) := t^* z_j + (1-t^*)\zeta(k, j-1).$$

Step 2: Set $\zeta(k+1) := \zeta(k+1, 0) := \zeta(k, M)$. If $\|\zeta(k+1) - \zeta(k)\| < \delta$, then stop. Otherwise, augment k by one and go to Step 1.

Lemma 1.14 *Let $(\zeta(k))_{k\in\mathbb{N}}$ be the sequence generated according to the algorithm above. Then*

$$\|\zeta(k+1)\| \leq \|\zeta(k)\|$$

and if equality holds, then $\zeta(k) = z^*$.

Proof: By construction,

$$\|\zeta(k,j)\| = \min_{t\in[0,1]} \|tz_j + (1-t)\zeta(k,j-1)\| \leq \|\zeta(k,j-1)\|.$$

Inductively, we get

$$\|\zeta(k+1)\| = \|\zeta(k,M)\| \leq \ldots \leq \|\zeta(k,0)\| = \|\zeta(k)\|.$$

Assume that $\|\zeta(k+1)\| = \|\zeta(k)\|$. Then we have equality in the relations above, in particular

$$\min_{t\in[0,1]} \|tz_j + (1-t)\zeta(k,j-1)\| = \|\zeta(k,j-1)\|,$$

i.e., the minimum is achieved at $t^* = 0$. This implies that $\zeta(k,j) = \zeta(k,j-1)$ and hence

$$\zeta(k+1) = \zeta(k,M) = \ldots = \zeta(k,0) = \zeta(k).$$

Thus for all $j = 1, \ldots, M$,

$$\|\zeta(k)\| = \min_{t\in[0,1]} \|tz_j + (1-t)\zeta(k)\|$$

that is, for all $0 \le t \le 1$,

$$\|\zeta(k)\|^2 \le \|tz_j + (1-t)\zeta(k)\|^2 = t^2\|z_j - \zeta(k)\|^2 + 2t\langle z_j - \zeta(k), \zeta(k)\rangle + \|\zeta(k)\|^2.$$

This implies that

$$0 \le \langle z_j - \zeta(k), \zeta(k)\rangle$$

for $j = 1, \ldots, M$ and thus

$$\langle z, \zeta(k)\rangle \ge \langle \zeta(k), \zeta(k)\rangle$$

for all $z \in C = \operatorname{conv}\{z_1, \ldots, z_M\}$. Using the Cauchy–Schwarz inequality, we get

$$\|z\| \ge \|\zeta(k)\| \quad \text{for all } z \in C$$

showing that $\|\zeta(k)\| = \min_{z\in C} \|z\|$. Lemma 1.12 implies that $\zeta(k) = z^*$. □

Theorem 1.15 *For any initial value $\zeta(0) \in C$, the sequence $(\zeta(k))_{k\in\mathbb{N}}$ converges to z^*.*

Proof: As a sequence of vectors in the compact set C, $(\zeta(k))_k$ has a convergent subsequence $(\zeta(k_l))_l$ with

$$\lim_{l\to\infty} \zeta(k_l) = \zeta^*. \tag{1.16}$$

We wish to prove that for all $j = 1, \ldots, M$

$$\|\zeta^*\| = \min_{t\in[0,1]} \|tz_j + (1-t)\zeta^*\|. \tag{1.17}$$

As discussed in the proof of the previous lemma, this implies that $\zeta^* = z^*$. Then any convergent subsequence of $(\zeta(k))_k$ converges to z^*, implying that $(\zeta(k))_k$ itself converges to z^*, and we are finished. Assume conversely to (1.17) that there exists j and $\varepsilon > 0$ with

$$\|\zeta^*\| \ge \min_{t\in[0,1]} \|tz_j + (1-t)\zeta^*\| + 2\varepsilon.$$

Without loss of generality, we may assume that j is the least integer with that property, i.e.,

$$\|\zeta^*\| = \min_{t\in[0,1]} \|tz_i + (1-t)\zeta^*\| \quad \text{for } i = 1, \ldots, j-1. \tag{1.18}$$

Due to (1.16), there exists $l_0 \in \mathbb{N}$ such that

$$\|\zeta(k_l) - \zeta^*\| \le \varepsilon \quad \text{for all } l \ge l_0.$$

Note that Step 1 of the algorithm induces continuous mappings $f_j : C \to C$ defined by $f_j(x) = t^* z_j + (1-t^*)x$, where t^* minimizes $\|t z_j + (1-t)x\|$ on $[0,1]$. That is to say

$$\zeta(k,j) = f_j(\zeta(k,j-1)) \quad \text{and} \quad \zeta(k+1) = (f_M \circ \cdots \circ f_1)(\zeta(k)).$$

In particular

$$\|f_j(x)\| = \min_{t\in[0,1]} \|t z_j + (1-t)x\| \le \|x\|.$$

Assumption (1.18) amounts to

$$f_i(\zeta^*) = \zeta^* \quad \text{for } i = 1, \ldots, j-1.$$

Define $F_j := f_j \circ \cdots \circ f_1$ and set $F_0 := \mathrm{id}_C$. Then $F_{j-1}(\zeta^*) = \zeta^*$. Due to the continuity of F_j, there exists $l_1 \in \mathbb{N}$ such that

$$\|F_j(\zeta(k_l)) - F_j(\zeta^*)\| \le \varepsilon \quad \text{for all } l \ge l_1 \ge l_0.$$

Then

$$\begin{aligned}
\|F_j(\zeta(k_l))\| &\le \|F_j(\zeta(k_l)) - F_j(\zeta^*)\| + \|F_j(\zeta^*)\| \\
&\le \varepsilon + \min_{t\in[0,1]} \|t z_j + (1-t)F_{j-1}(\zeta^*)\| \\
&= \varepsilon + \min_{t\in[0,1]} \|t z_j + (1-t)\zeta^*\| \\
&\le \varepsilon + \|\zeta^*\| - 2\varepsilon \quad = \quad \|\zeta^*\| - \varepsilon.
\end{aligned}$$

Now

$$\|\zeta(k_{l+1})\| \le \|\zeta(k_l + 1)\| = \|F_M(\zeta(k_l))\| \le \|F_j(\zeta(k_l))\| \le \|\zeta^*\| - \varepsilon.$$

Letting l tend to infinity, we obtain $\|\zeta^*\| \le \|\zeta^*\| - \varepsilon$, which is a contradiction. □

Alternatively, the optimal separating hyperplane for $X = X_+ \cup X_- = \{x_1, \ldots, x_N\}$ can be found by minimizing $\|w\|$, or equivalently, $\frac{1}{2}\|w\|^2$, subject to

$$\langle w, z\rangle \ge c > 0 \quad \text{for all } z \in C = \mathrm{conv}(X_+ - X_-).$$

This is a consequence of Corollary 1.13. Similarly as in the proof of Lemma 1.10, one shows that

$$\langle w, z\rangle \ge 2 \text{ for all } z \in C \quad \Leftrightarrow \quad \exists \theta : \begin{cases} \langle w, x\rangle - \theta \ge 1 & \text{for all } x \in X_+ \\ \langle w, y\rangle - \theta \le -1 & \text{for all } y \in X_-. \end{cases}$$

Thus the optimal separating hyperplane can be found by solving the quadratic optimization problem

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\|w\|^2 \\ \text{subject to} & y_i(\langle w, x_i\rangle - \theta) \geq 1 \text{ for } 1 \leq i \leq N, \end{array} \tag{1.19}$$

where $y_i = \pm 1$ if $x_i \in X_\pm$.

Theorem 1.16 (Kuhn-Tucker) *Let $f : \mathbb{R}^m \to \mathbb{R}$ and $g_i : \mathbb{R}^m \to \mathbb{R}$, $1 \leq i \leq N$ be continuously differentiable convex functions. Then the minimization problem*

$$\begin{array}{ll} \textit{Minimize} & f(x) \\ \textit{subject to} & g_1(x) \leq 0, \ldots, g_N(x) \leq 0 \end{array}$$

is equivalent to the maximization problem

$$\begin{array}{ll} \textit{Maximize} & F(x,\mu) := f(x) + \sum_{i=1}^N \mu_i g_i(x) \quad \textit{with respect to } \mu \\ \textit{subject to} & G(x,\mu) := \nabla f(x) + \sum_{i=1}^N \mu_i \nabla g_i(x) = 0 \textit{ and } \mu_1, \ldots, \mu_N \geq 0. \end{array}$$

The parameters $\mu = (\mu_1, \ldots, \mu_N)$ can be interpreted as generalized Lagrange multipliers. In our particular situation, $m = n + 1$,

$$x = \begin{bmatrix} w \\ \theta \end{bmatrix}, \quad f(w,\theta) = \frac{1}{2}\|w\|^2, \quad \text{and} \quad g_i(w,\theta) = 1 - y_i(\langle w, x_i\rangle - \theta).$$

Then

$$\nabla f(w,\theta) = \begin{bmatrix} w \\ 0 \end{bmatrix} \quad \text{and} \quad \nabla g_i(w,\theta) = \begin{bmatrix} -y_i x_i \\ y_i \end{bmatrix}$$

and thus $G(w,\theta,\mu) = 0$ reads

$$w = \sum_{i=1}^N \mu_i y_i x_i \quad \text{and} \quad \sum_{i=1}^N \mu_i y_i = 0.$$

Using this, we may write

$$\begin{aligned} F(w,\theta,\mu) &= \frac{1}{2}\|w\|^2 + \sum_{i=1}^N \mu_i - \sum_{i=1}^N \mu_i y_i \langle w, x_i\rangle + \sum_{i=1}^N \mu_i y_i \theta \\ &= \frac{1}{2}\|w\|^2 + \sum_{i=1}^N \mu_i - \left\langle w, \sum_{i=1}^N \mu_i y_i x_i \right\rangle \\ &= -\frac{1}{2}\|w\|^2 + \sum_{i=1}^N \mu_i \end{aligned}$$

and thus

$$F(w,\theta,\mu) = -\frac{1}{2}\sum_{i,j=1}^N \mu_i \mu_j y_i y_j \langle x_i, x_j\rangle + \sum_{i=1}^N \mu_i =: \tilde{F}(\mu).$$

Theorem 1.17 *The optimal solution w^* of (1.19) is given by*

$$w^* = \sum_{i=1}^{N} \mu_i^* y_i x_i,$$

where $\mu^* = (\mu_1^*, \dots, \mu_N^*)$ is the solution to

$$\begin{array}{ll} \textit{Maximize} & \tilde{F}(\mu) = -\frac{1}{2}\sum_{i,j=1}^{N} \mu_i \mu_j y_i y_j \langle x_i, x_j\rangle + \sum_{i=1}^{N} \mu_i \\ \textit{subject to} & \sum_{i=1}^{N} \mu_i y_i = 0 \textit{ and } \mu_1, \dots, \mu_N \geq 0. \end{array} \tag{1.20}$$

The vectors x_i with $\mu_i^* \neq 0$ are called **support vectors**. Theorem 1.17 yields a reformulation of the original optimization problem over $\mathbb{R}^n$ as an optimization problem over $\mathbb{R}^N$. This will be advantageous for **support vector learning**. Roughly speaking, the idea is to perform non-linear separation by embedding the data in a high-dimensional Hilbert space in which they become linearly separable. Then n (the Hilbert space dimension) is typically significantly larger than N (the number of data points). Another important feature of (1.20) is the fact that the data vectors $x_1, \dots, x_N$ enter only in the form $\langle x_i, x_j\rangle$.

1.6 Support Vector Learning

Definition 1.8 A continuous symmetric map $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is called a **positive kernel** if for any finite set $\{x_1, \dots, x_N\} \subset \mathbb{R}^n$, the matrix $(k(x_i, x_j))_{i,j=1,\dots,N}$ is positive semi-definite.

Theorem 1.18 (Aronszajn) *If k is a positive kernel, then there exists a Hilbert space $\mathcal{H}$ and a map $\Phi : \mathbb{R}^n \to \mathcal{H}$ such that*

$$k(x, y) = \langle \Phi(x), \Phi(y)\rangle \quad \textit{for all } x, y \in \mathbb{R}^n.$$

The Hilbert space $\mathcal{H}$ is constructed in a way that it contains all linear combinations of functions

$$k_x : \mathbb{R}^n \to \mathbb{R}, \quad y \mapsto k_x(y) := k(x, y),$$

where $x \in \mathbb{R}^n$. We set

$$\langle k_x, k_y\rangle = k(x, y)$$

and $\Phi(x) = k_x$. Thus

$$\langle \Phi(x), \Phi(y)\rangle = \langle k_x, k_y\rangle = k(x, y).$$

Examples:

1. $k(x,y) = \langle x, y\rangle^d$ for $d \in \mathbb{N}$. Since
$$k(x,y) = \left(\sum_{i=1}^n x_i y_i\right)^d = \sum_{\alpha\in\mathbb{N}^n, |\alpha|=d} \binom{d}{\alpha} x^\alpha y^\alpha$$
where
$$|\alpha| = \sum_{i=1}^n \alpha_i, \qquad \binom{d}{\alpha} = \frac{d!}{\alpha_1!\cdots\alpha_n!}, \qquad x^\alpha = x_1^{\alpha_1}\cdots x_n^{\alpha_n}$$
we put
$$\Phi(x) = \left(\tbinom{d}{\alpha}^{1/2} x^\alpha\right)_{\alpha\in\mathbb{N}^n, |\alpha|=d}.$$
Thus the components of $\Phi(x)$ are (up to the scaling factor) the monomials of total degree d. Since the number of such monomials is finite, $\mathcal{H} = \mathbb{R}^M$ for some $M \in \mathbb{N}$, with the Euclidean scalar product.

2. $k(x,y) = (1 + \langle x, y\rangle)^d$ for $d \in \mathbb{N}$. Then
$$k(x,y) = \left(1 + \sum_{i=1}^n x_i y_i\right)^d = \sum_{\alpha\in\mathbb{N}^n, |\alpha|\le d} \frac{1}{(d-|\alpha|)!}\binom{d}{\alpha} x^\alpha y^\alpha$$
and we put
$$\Phi(x) = \left(\left(\tfrac{1}{(d-|\alpha|)!}\tbinom{d}{\alpha}\right)^{1/2} x^\alpha\right)_{\alpha\in\mathbb{N}^n, |\alpha|\le d}$$
and $\mathcal{H}$ is again finite-dimensional. The components of Φ correspond to the monomials of total degree at most d.

3. $k(x,y) = \tanh(\alpha\langle x, y\rangle + \beta)$ for suitable α, β.

4. $k(x,y) = \exp(-\|x-y\|^2/2\sigma^2)$.

As usual, let $X_+, X_- \subset \mathbb{R}^n$ be non-empty finite sets. No assumption is made about the affine separability of the sets. Set $X = X_+ \cup X_- = \{x_1, \ldots, x_n\}$ and let $y_i = \pm 1$ if $x_i \in X_\pm$. We say that $\Phi : \mathbb{R}^n \to \mathcal{H}$ separates X_+, X_- if $\Phi(X_+)$, $\Phi(X_-)$ are affinely separable in $\mathcal{H}$. If Φ and $\mathcal{H}$ are constructed from a positive kernel k as in Theorem 1.18, this notion depends only on k, and therefore it is justified to say that k separates X_+, X_-.

In the following, let k be a positive kernel and let Φ and $\mathcal{H}$ be as in Theorem 1.18. Suppose that k separates X_+, X_-. Then we may solve the optimal separation problem, that is,

Maximize $\tilde{F}(\mu) = -\frac{1}{2}\sum_{i,j=1}^N \mu_i\mu_j y_i y_j \langle\Phi(x_i), \Phi(x_j)\rangle + \sum_{i=1}^N \mu_i$
subject to $\sum_{i=1}^N \mu_i y_i = 0$ and $\mu_1, \dots, \mu_N \geq 0$.

Note that

$$\tilde{F}(\mu) = -\frac{1}{2}\sum_{i,j=1}^N \mu_i\mu_j y_i y_j k(x_i, x_j) + \sum_{i=1}^N \mu_i = -\frac{1}{2} z^T K z + \sum_{i=1}^N \mu_i$$

where $K \in \mathbb{R}^{N\times N}$ and $z \in \mathbb{R}^N$ are defined by

$$K_{ij} = k(x_i, x_j) \quad \text{and} \quad z_i = \mu_i y_i.$$

Let $\mu^* = (\mu_1^*, \dots, \mu_N^*)$ be a solution, then

$$w^* = \sum_{i=1}^N \mu_i^* y_i \Phi(x_i) \in \mathcal{H}$$

and

$$\theta^* := \theta_{w^*} = \frac{1}{2}\left(\min_{x\in X_+}\sum_{i=1}^N \mu_i^* y_i k(x_i, x) + \max_{y\in X_-}\sum_{i=1}^N \mu_i^* y_i k(x_i, y)\right) \in \mathbb{R}$$

define the optimal separating hyperplane

$$H^* = \{y \in \mathcal{H} \mid \langle w^*, y\rangle = \theta^*\}.$$

For $x \in X_+$, we have $\Phi(x) \in \Phi(X_+)$ and therefore

$$\langle w^*, \Phi(x)\rangle = \sum_{i=1}^N \mu_i^* y_i \langle\Phi(x_i), \Phi(x)\rangle = \sum_{i=1}^N \mu_i^* y_i k(x_i, x) > \theta^*$$

and similarly, for $y \in X_-$,

$$\sum_{i=1}^N \mu_i^* y_i k(x_i, y) < \theta^*.$$

In other words, we have realized the original function $f : X \to \{\pm 1\}$ with $f(X_\pm) = \pm 1$ by

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^N \mu_i^* y_i k(x_i, x) - \theta^*\right),$$

that is, a formal neuron with activation function $s(x) = \sum \mu_i^* y_i k(x_i, x) - \theta^*$ and output function sign. Note that the maximization problem as well as the resulting separation of X_+, X_- depend only on k. Explicit knowledge of Φ is not necessary.

Example: Let $f : \{0,1\}^2 \to \{\pm 1\}$ be the modified XOR function, that is, we have $N = 4$ and $f(x_i) = y_i$, where

$$\begin{array}{rclcrcr} x_1 & = & (0,0) & \qquad & y_1 & = & -1 \\ x_2 & = & (0,1) & & y_2 & = & 1 \\ x_3 & = & (1,0) & & y_3 & = & 1 \\ x_4 & = & (1,1) & & y_4 & = & -1 \end{array}$$

Let $k(x,y) = \langle x,y \rangle^2$ for $x, y \in \mathbb{R}^2$. Then

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 4 \end{bmatrix} \quad \text{and} \quad z = \begin{bmatrix} -\mu_1 \\ \mu_2 \\ \mu_3 \\ -\mu_4 \end{bmatrix}$$

and we have to maximize

$$\begin{aligned} \tilde{F}(\mu) &= -\tfrac{1}{2} z^T K z + \textstyle\sum_{i=1}^4 \mu_i \\ &= -\tfrac{1}{2}(\mu_2^2 + \mu_3^2) - 2\mu_4^2 + \mu_2\mu_4 + \mu_3\mu_4 + \mu_1 + \mu_2 + \mu_3 + \mu_4 \end{aligned}$$

subject to $-\mu_1 + \mu_2 + \mu_3 - \mu_4 = 0$ and $\mu_1, \dots, \mu_4 \geq 0$. We solve the constraint for μ_1 and plug in. It remains to maximize

$$\hat{F}(\mu_2, \mu_3, \mu_4) = -\frac{1}{2}(\mu_2^2 + \mu_3^2) - 2\mu_4^2 + \mu_2\mu_4 + \mu_3\mu_4 + 2\mu_2 + 2\mu_3.$$

The gradient of this function is

$$\nabla \hat{F} = \begin{bmatrix} -\mu_2 + \mu_4 + 2 \\ -\mu_3 + \mu_4 + 2 \\ -4\mu_4 + \mu_2 + \mu_3 \end{bmatrix}.$$

Solving $\nabla \hat{F} = 0$ yields $\mu_2^* = \mu_3^* = 4$, $\mu_4^* = 2$. This implies that $\mu_1^* = 6$. Recall that $\Phi(x) = \Phi(x_{(1)}, x_{(2)}) = (x_{(1)}^2, x_{(2)}^2, \sqrt{2} x_{(1)} x_{(2)})$. Thus

$$w^* = \sum_{i=1}^4 \mu_i^* y_i \Phi(x_i) = 2 \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \end{bmatrix}.$$

Similarly, one computes $\theta^* = 1$. Thus the optimal separating hyperplane in $\mathcal{H} = \mathbb{R}^3$ is

$$H^* = \{ y \in \mathbb{R}^3 \mid 2(y_{(1)} + y_{(2)} - \sqrt{2} y_{(3)}) = 1 \}.$$

With $\Phi(x) = (y_{(1)}, y_{(2)}, y_{(3)})$, this corresponds to

$$\begin{aligned} f(x) &= \operatorname{sign}(2(x_{(1)}^2 + x_{(2)}^2 - 2x_{(1)}x_{(2)}) - 1) \\ &= \operatorname{sign}((x_{(2)} - x_{(1)} - \tfrac{1}{\sqrt{2}})(x_{(2)} - x_{(1)} + \tfrac{1}{\sqrt{2}})). \end{aligned}$$

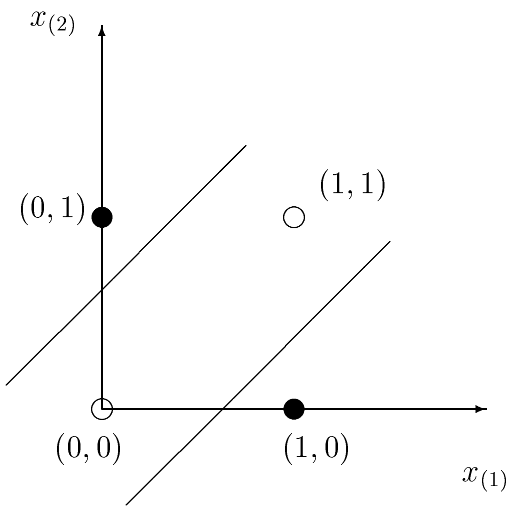


Figure 1.5: *Solution to the XOR-problem using a quadratic activation function.*

Theorem 1.19 *For any Boolean function $f : \{0,1\}^n \to \{0,1\}$, there exists a homogeneous polynomial of total degree n, that is,*

$$p(x) = \sum_{\alpha \in \mathbb{N}^n, |\alpha| = n} p_\alpha x^\alpha, \quad p_\alpha \in \mathbb{R}$$

and $\theta \in \mathbb{R}$ such that

$$f(x) = \mathrm{sat}(p(x) - \theta) \quad \textit{for all } x \in \{0,1\}^n.$$

In other words, $k(x,y) = \langle x, y \rangle^n$ separates any partition X_+, X_- of $X = \{0,1\}^n$.

The proof of this theorem makes use of the Shegalkin polynomials discussed in Appendix A, exploiting the fact that any $x \in \{0,1\}$ is idempotent, i.e., $x = x^2$.

Chapter 2

Feed-forward Networks

The efficiency of perceptrons is quite limited. The construction of multilayer networks, which have the capacity to realize all switching functions, remedies the matter. Similarly to the perceptron, linear optimization is used in forward directed networks to determine an explicit separation of arbitrary finite sets. This chapter ends with an overview of the Back-propagation Algorithm.

2.1 Structure of Feed-forward Networks

As was illustrated in the previous chapter, the representation of a Boolean function by means of a perceptron is of limited generality, as not all switching functions can be realized. It is often advantageous to link several perceptrons in order to increase the number of functions that can be represented. To this end, the following graph-theoretical definitions are necessary.

Definition 2.1 (a) A (finite, simple, directed) **graph** $G = (V, E)$ is composed of a non-empty finite set of vertices or nodes V and a set of edges $E \subseteq V \times V$.

(b) $P(i) := \{j \in V \,:\, (j,i) \in E\}$ is the **set of all direct predecessors** and $S(i) := \{j \in V \,:\, (i,j) \in E\}$ is the **set of direct successors** of the node $i \in V$. A vertex i with $P(i) = \emptyset$ is called a **source**, and a node i with $S(i) = \emptyset$ is said to be a **sink**.

(c) For $i_0, \ldots, i_l \in V$, let $e_j := (i_{j-1}, i_j) \in E$ apply for all $j = 1, \ldots, l$. The corresponding sequence of edges $(e_1, \ldots, e_l)$ is called a **path** from i_0 to i_l and the number of edges l is its **length**. A path whose first and last nodes coincide is called a **cycle**. A graph that is devoid of cycles is called **acyclic**.

Lemma 2.1 *An acyclic graph contains a source and a sink.*

Proof: Suppose that $G = (V, E)$ contains no sink. Then any vertex possesses a direct successor, which implies that there exist paths of arbitrary length. Consider a path $(e_1, \dots, e_l)$ whose length exceeds the (finite) number of edges of the graph. The edges cannot be distinct, that is, we must have $e_i = e_j$ for some $1 \le i < j \le l$. But then $(e_i, \dots, e_{j-1})$ has to be a cycle. The dual statement on sources is proven by reversing the orientation of each edge of the graph. $\square$

Let $G = (V, E)$ be an acyclic graph. There exists an integer $k \ge -1$ and a natural partition of V into non-empty sets

$$V = V_0 \,\dot\cup\, V_1 \,\dot\cup\, \dots \,\dot\cup\, V_k \,\dot\cup\, V_{k+1} \tag{2.1}$$

which is constructed as follows: V_0 contains all sources. Remove from G all vertices in V_0 and all edges that start in one of them. The resulting graph is again acyclic. Let V_1 be the set of all sources in the new graph, etc. Thus $i \in V_j$ iff the longest path from a source to i has length j. Moreover,

$$i \in V_j \quad \Rightarrow \quad P(i) \subseteq V_0 \cup \dots \cup V_{j-1} \quad \text{and} \quad S(i) \subseteq V_{j+1} \cup \dots \cup V_{k+1}.$$

The elements of V_{k+1} are sinks.

Definition 2.2 Let $G = (V, E)$ be an acyclic graph, and let $V = V_0 \cup \dots \cup V_{k+1}$ be the partition according to (2.1). A **feed-forward network** $\mathcal{F}$ is composed of the graph G and a family of formal neurons $(X_i, Y_i, \sigma_i, s_i)$, each associated to one of the non-source vertices $i \in V \setminus V_0$. We have $X_i \subseteq \mathbb{R}^{n_i}$, where $n_i := |P(i)|$, $Y_i \subseteq \mathbb{R}$,

$$s_i : X_i \to \mathbb{R} \quad \text{and} \quad \sigma_i : \mathbb{R} \to Y_i.$$

The transfer functions of these neurons are $(f_i)_{i \in V \setminus V_0}$, with $f_i = \sigma_i \circ s_i : X_i \to Y_i$. The compatibility requirement

$$\prod_{j \in P(i)} Y_j \subseteq X_i \quad \text{for all } i \in V \setminus V_0$$

will be satisfied if we set $X_i = Q^{n_i}$, $Y_i = Q$ for all i, for some set $Q \subseteq \mathbb{R}$. For simplicity, we make this assumption in the following.

Then each node $i \in V$ has a **state** $q_i \in Q$. For $i \in V_0$, the states will have to be imposed by some additional initial condition. For all $i \in V \setminus V_0$, the state is determined by the states of the direct predecessors of i, via

$$q_i = f_i\left((q_j)_{j \in P(i)}\right).$$

The nodes in V_0, $V \setminus (V_0 \cup V_{k+1})$, and V_{k+1} are called input, hidden, and output nodes, respectively. Similarly, the associated neurons are called hidden neurons or output neurons. The **transfer function** of $\mathcal{F}$ is the mapping

$$f : Q^{|V_0|} \to Q^{|V_{k+1}|}$$

which maps the vector of input states to the vector of output states.

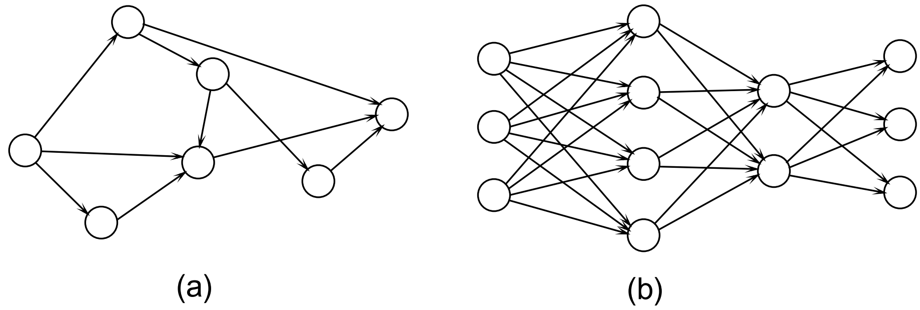


Figure 2.1: *(a) Graph of a feed-forward network without a layer structure. (b) Graph of a 2-layer feed-forward network.*

Definition 2.3 A feed-forward network $\mathcal{F}$ with

$$V = V_0 \,\dot\cup\, V_1 \,\dot\cup\, \ldots \,\dot\cup\, V_k \,\dot\cup\, V_{k+1}$$

according to (2.1) is called ***k*-layer feed-forward network** if

$$i \in V_j \quad \Rightarrow \quad P(i) \subseteq V_{j-1} \quad \text{and} \quad S(i) \subseteq V_{j+1}$$

for $j = 0, \ldots, k+1$ (putting $V_{-1} = V_{k+2} = \emptyset$). We shall restrict to the case where the layers are fully interconnected, that is,

$$i \in V_j \quad \Rightarrow \quad P(i) = V_{j-1} \quad \text{and} \quad S(i) = V_{j+1}$$

for $j = 0, \ldots, k+1$. Then V_0 is the set of all sources (by construction) and V_{k+1} is the set of all sinks. We write $V_0 = V_I$ and $V_{k+1} = V_O$ (the subscripts refer to input and output, respectively.)

Definition 2.4 A k-layer feed-forward network whose neurons are all (σ-) perceptrons, is called a ***k*-layer (σ-) perceptron**.

The transfer function of a k-layer σ-perceptron is therefore a composition

$$f = f^{(k+1)} \circ \cdots \circ f^{(1)} : Q^{|V_I|} \to Q^{|V_O|}$$

of $k+1$ mappings of the form

$$\begin{aligned} f^{(i)} : Q^{|V_{i-1}|} &\to Q^{|V_i|} \\ q &\mapsto \underline{\sigma}(W^{(i)} q - \theta^{(i)}) \end{aligned}$$

where $W^{(i)} \in \mathbb{R}^{|V_i| \times |V_{i-1}|}$ is a fixed weight matrix, $\theta^{(i)} \in \mathbb{R}^{|V_i|}$ is a threshold vector, and $\underline{\sigma}$ is a vector-valued sigmoid function defined by component-wise application of $\sigma : \mathbb{R} \to Q$, that is, for any integer $n > 0$,

$$\underline{\sigma}\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} := \begin{pmatrix} \sigma(x_1) \\ \vdots \\ \sigma(x_n) \end{pmatrix}.$$

For $i = 1, \ldots, k$, the mapping $f^{(i)}$ is called the transfer function of the i-th hidden layer, and $f^{(k+1)}$ is referred to as the transfer function of the output layer.

Unfortunately, the counting of layers for multilayer networks is not unified in the literature. Here, we will stick to counting the hidden layers only.

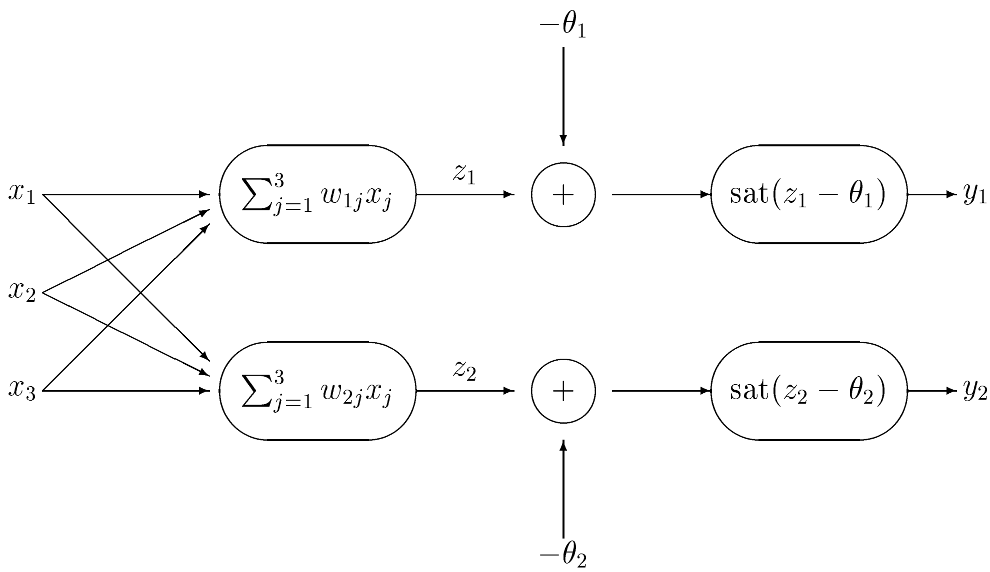


Figure 2.2: *A layer of two perceptrons in a multilayer perceptron.*

2.2 Realization by Multi-layer Perceptrons

Realization of Boolean Functions

As the XOR-problem in Lemma 1.1 shows, the representation of arbitrary switching functions by a single perceptron is impossible. The situation is different when a multilayer perceptron is used.

Here, the set of possible states of neurons will be $Q = \{0, 1\}$ and the transfer functions considered will have the form $f : \{0,1\}^n \to \{0,1\}^p$, that is, we consider feed-forward networks with n input nodes and p output neurons. A simple observation is the fact that, as long as the number of hidden neurons is unconstrained, $f = (f_1, \dots, f_p)$ can be realized by such a feed-forward network iff all its components $f_i : \{0,1\}^n \to \{0,1\}$ can be realized by a feed-forward network (with n input nodes and one output neuron).

Theorem 2.2 *Any Boolean function $f : \{0,1\}^n \to \{0,1\}^p$ can be realized by a 1-layer perceptron.*

Proof: Let $p = 1$, without loss of generality. Let $X = \{0,1\}^n$, $X_- = f^{-1}(0)$,

$$X_+ = f^{-1}(1) = \{x_1, \dots, x_k\},$$

and $X_i := \{x_i\}$ for $i = 1, \dots, k$. Then $\mathrm{conv}(X_i) = X_i$ and

$$\mathrm{conv}(X_i) \cap \mathrm{conv}(X \setminus X_i) = \emptyset.$$

This follows from the fact that x_i is an extreme point of the unit cube $\mathrm{conv}(X)$. This signifies that $\mathrm{conv}(X) \setminus X_i$ is still convex, and therefore

$$\mathrm{conv}(X \setminus X_i) \subseteq \mathrm{conv}(X) \setminus X_i.$$

Thus X_i and $X \setminus X_i$ are affinely separable. For $i = 1, \dots, k$, let

$$\langle w^{(i)}, x \rangle = \theta^{(i)}$$

be a hyperplane that separates X_i from $X \setminus X_i$, that is, for $1 \le i \le k$,

$$X_i \subset H_i := \{x \in \mathbb{R}^n \ : \ \langle w^{(i)}, x \rangle \ge \theta^{(i)}\} \quad \text{and} \quad X_i = X \cap H_i.$$

Define $g^{(1)}(x) = y = (y_1, \dots, y_k)^T$ by

$$y_i := \mathrm{sat}(\langle w^{(i)}, x \rangle - \theta^{(i)}) \quad \text{for } i = 1, \dots, k.$$

Then with

$$g^{(2)}(y) = z = \mathrm{sat}(y_1 + \cdots + y_k - 0.5)$$

we have constructed a function $g = g^{(2)} \circ g^{(1)}$ that coincides with the given function f due to

$$\begin{aligned} z = g(x) = 1 \quad &\Leftrightarrow \quad \exists i \in \{1, \dots, k\} \text{ with } y_i = 1 \\ &\Leftrightarrow \quad \exists i \in \{1, \dots, k\} \text{ with } \langle w^{(i)}, x \rangle \ge \theta^{(i)} \\ &\Leftrightarrow \quad \exists i \in \{1, \dots, k\} \text{ with } x \in X_i \\ &\Leftrightarrow \quad x \in X_+ \\ &\Leftrightarrow \quad f(x) = 1. \end{aligned}$$

□

Example: The XOR-function shall be realized as a perceptron with one hidden layer (cf. Fig. 2.3). For the XOR-problem,

$$X_+ = \{(0,1);(1,0)\}, \quad \text{and} \quad X_- = \{(0,0);(1,1)\}$$

with $X_1 = \{(0,1)\}$ and $X_2 = \{(1,0)\}$. Hence, we have the half-spaces

$$\begin{aligned} H_1 &= \{x \in \mathbb{R}^2 \ : \ x_2 - x_1 - 0.5 \geq 0\}; \\ H_2 &= \{x \in \mathbb{R}^2 \ : \ x_1 - x_2 - 0.5 \geq 0\} \end{aligned}$$

and thus

$$\begin{aligned} y_1 &= \mathrm{sat}(x_2 - x_1 - 0.5), \\ y_2 &= \mathrm{sat}(x_1 - x_2 - 0.5), \\ z &= \mathrm{sat}(y_1 + y_2 - 0.5). \end{aligned}$$

This can be summarized in the following table:

x_1	x_2	y_1	y_2	z
0	0	0	0	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

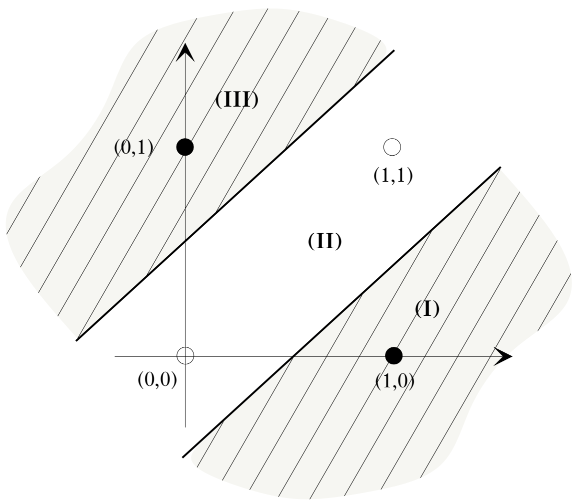


Figure 2.3: *Realization of the XOR-function by a perceptron with one hidden layer (in region (I) neuron 1 fires; in (II) no neuron fires; in (III) neuron 2 fires.)*

Remarks

1. The number of neurons that lie in the hidden layer depends on the switching function, i.e., if the number of hidden neurons is constrained, such a network cannot necessarily represent all switching functions.

2. As $\mathbb{Q}$ is dense in $\mathbb{R}$, the weights can always be chosen to be rational numbers. Since multiplication by positive numbers does not change the corresponding half-spaces, it even suffices to consider integer-valued weights.

Realization of Arbitrary Functions

Here $Q = \mathbb{R}$. Since we use the Heaviside function as output function, the states of all neurons will nevertheless be in $\{0, 1\}$. Thus we consider transfer functions $f : \mathbb{R}^n \to \{0, 1\}^p$, and the networks under consideration have n input nodes and p output neurons. Again, it is sufficient to consider the case where $p = 1$.

Theorem 2.3 *1. Let $A \subset \mathbb{R}^n$ be a polyhedron. Then there exists a 1-layer perceptron whose transfer function $f : \mathbb{R}^n \to \{0, 1\}$ satisfies $f^{-1}(1) = A$.*

2. Let $B \subset \mathbb{R}^n$ be a finite union of polyhedra. Then there exists a 2-layer perceptron whose transfer function $f : \mathbb{R}^n \to \{0, 1\}$ satisfies $f^{-1}(1) = B$.

Proof: Let $A = \{x \in \mathbb{R}^n \mid \langle x, w_1\rangle \geq \theta_1, \ldots, \langle x, w_k\rangle \geq \theta_k\}$. Let $\mathcal{F}$ be a 1-layer perceptron with k hidden neurons with transfer functions

$$g_i : \mathbb{R}^n \to \{0, 1\}, \quad x \mapsto \mathrm{sat}(\langle x, w_i\rangle - \theta_i)$$

and let the transfer function of the output neuron be

$$h : \{0, 1\}^k \to \{0, 1\}, \quad y \mapsto \mathrm{sat}(y_1 + \ldots + y_k - (k - 0.5)).$$

Let $g = (g_1, \ldots, g_k)$. Then the transfer function of $\mathcal{F}$ is $f := h \circ g$. We have

$$x \in A \Leftrightarrow g_i(x) = 1 \forall i \Leftrightarrow h(g(x)) = 1 \Leftrightarrow f(x) = 1 \Leftrightarrow x \in f^{-1}(1).$$

If $B = A_1 \cup \ldots \cup A_l$ with A_i the intersection of $k(i)$ closed half-spaces, we first build, as above, a 1-layer network with n inputs, $\sum_{i=1}^{l} k(i)$ hidden neurons and l output neurons such that the transfer function $g : \mathbb{R}^n \to \{0, 1\}^l$ of this network satisfies $g(x)_i = 1 \Leftrightarrow x \in A_i$. Then we add a new output layer with one output neuron and set

$$h : \{0, 1\}^l \to \{0, 1\}, \quad z \mapsto \mathrm{sat}(z_1 + \ldots + z_l - 0.5).$$

Let $f = h \circ g$ be the transfer function of the whole network. We have

$$f(x) = 1 \Leftrightarrow g(x) \neq 0 \Leftrightarrow \exists i : g(x)_i = 1 \Leftrightarrow \exists i : x \in A_i \Leftrightarrow x \in B.$$

□

Theorem 2.4 *Let $A \subset \mathbb{R}^n$ be closed and let $B \subset \mathbb{R}^n$ be compact, with $A \cap B = \emptyset$. Then there exists a 2-layer perceptron whose transfer function $f : \mathbb{R}^n \to \{0, 1\}$ satisfies $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$.*

Proof: Since B is compact, there exists a set $B_1 \supseteq B$ which is a finite union of polyhedra. Since A is closed, and $A \cap B = \emptyset$, this set can be chosen such that $A \cap B_1 = \emptyset$. According to Theorem 2.3, there exists a 2-layer perceptron whose transfer function f satisfies $f^{-1}(1) = B_1$. Then $f^{-1}(0) = \mathbb{R}^n \setminus B_1 \supseteq A$ and $B \subseteq B_1 = f^{-1}(1)$. □

Next we use an arbitrary output function $\sigma : \mathbb{R} \to [0, 1]$, with

$$\lim_{x \to -\infty} \sigma(x) = 0 \quad \text{and} \quad \lim_{x \to \infty} \sigma(x) = 1.$$

Theorem 2.5 *Let σ be as above, and let $0 < \varepsilon < 1$. Let $A \subset \mathbb{R}^n$ be closed, let $B \subset \mathbb{R}^n$ be compact, and let $A \cap B = \emptyset$. Then there exists a σ-perceptron with 2 hidden layers whose transfer function $f : \mathbb{R}^n \to [0, 1]$ satisfies*

$$\begin{aligned} x \in A &\Rightarrow f(x) \leq \varepsilon \\ x \in B &\Rightarrow f(x) \geq 1 - \varepsilon. \end{aligned}$$

If σ is additionally monotone, such a transfer function can be seen as a "fuzzy" membership map.

2.3 Back-propagation Algorithm

According to Theorem 2.4, disjoint finite sets can be separated by 2-layer perceptrons if the number of required neurons in the hidden layers is unconstrained. At this point, we will present another algorithm where the number of neurons is given a priori.

The classical perceptron uses the Heaviside function or the sign function. Here, the functions involved have to be differentiable. Therefore, the standard sigmoid function $\sigma_c : \mathbb{R} \to (0, 1)$ with

$$\sigma_c(x) = \frac{1}{1 + e^{-cx}}$$

will be used as the activation function. The parameter c determines its steepness. An advantage of this sigmoid function is the simple calculation of its derivative:

$$\sigma_c'(x) = \frac{c\,e^{-cx}}{(1+e^{-cx})^2} = c\,\sigma_c(x)\,(1-\sigma_c(x)).$$

For simplicity, only the case $c = 1$ will be considered. We write $\sigma_1(x) = \sigma(x)$.

In the following, let a one-layer σ-perceptron be given with n input nodes, l hidden neurons and p output neurons. Suppose that the task of the network is to assign, to a given set of input vectors $x^{(i)} \in \mathbb{R}^n$, $i = 1, \ldots, N$, certain prescribed output vectors $\zeta^{(i)} \in (0,1)^p$, as precisely as possible. Typically, the network should learn to classify patterns $x \in \mathbb{R}^n$, based on the information contained in the set of given input-output-pairs $(x^{(i)}, \zeta^{(i)})$. These correspond to known correct classifications (training data). Instead of the discrete value set $\{0,1\}$, this classification admits a continuous range of output values between 0 and 1.

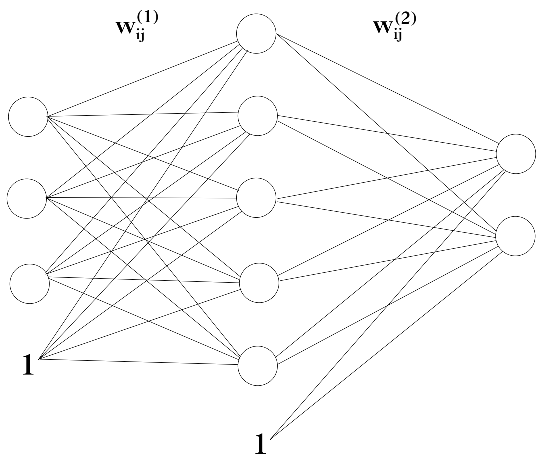


Figure 2.4: *Structure of a network for the Back-propagation Algorithm.*

Let $w_{mk}^{(1)}$ denote the weight between input node m and hidden neuron k, and $w_{kj}^{(2)}$ the weight between hidden neuron k and output neuron j. Hence, the output value at neuron j is

$$z_j^{(i)} = \sigma\left(\sum_{k=1}^{l} w_{kj}^{(2)} \sigma\left(\sum_{m=1}^{n} w_{mk}^{(1)} x_m^{(i)} - \theta_k^{(1)}\right) - \theta_j^{(2)}\right),$$

if $x^{(i)}$ is applied as an input. Just like with the perceptron, the threshold vectors can be eliminated by introducing auxiliary neurons. In order to simplify the notation, we assume this has already been done, i.e., the input layer and the hidden layer have each been extended by one neuron whose state is always 1 (cf.

Fig. 2.4). The output values of one layer are the input values of the next layer. Thus we write

$$y_j^{(i)} = \sigma\left(\sum_{m=1}^{n+1} w_{mj}^{(1)} x_m^{(i)}\right) \quad \text{and} \quad z_j^{(i)} = \sigma\left(\sum_{k=1}^{l+1} w_{kj}^{(2)} y_k^{(i)}\right).$$

In order to "train" a network with the Back-propagation Algorithm, the quality of the generated network has to be measurable. Therefore, we introduce the performance function

$$\begin{aligned} E : \mathbb{R}^{(n+1)l+p(l+1)} &\to \mathbb{R} \\ (w_{11}^{(1)}, \ldots, w_{n+1,l}^{(1)}, w_{11}^{(2)}, \ldots, w_{l+1,p}^{(2)}) &\mapsto \frac{1}{2}\sum_{i=1}^{N} \|z^{(i)} - \zeta^{(i)}\|_2^2. \end{aligned}$$

It assigns the mean squared error between the true and the desired network outputs $z^{(i)}$ and $\zeta^{(i)}$, respectively, to the vector of weights describing the network. The weights of the network are to be chosen such that E becomes minimal. The partial derivatives of E are computed as follows:

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \frac{\partial E}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} (z_j^{(i)} - \zeta_j^{(i)})\, z_j^{(i)}\, (1 - z_j^{(i)})\, y_k^{(i)}$$

and

$$\begin{aligned} \frac{\partial E}{\partial w_{kj}^{(1)}} &= \sum_{i=1}^{N} \left(\sum_{q=1}^{p} \frac{\partial E}{\partial z_q^{(i)}} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}}\right) \frac{\partial y_j^{(i)}}{\partial w_{kj}^{(1)}} \\ &= \sum_{i=1}^{N} \left(\sum_{q=1}^{p} (z_q^{(i)} - \zeta_q^{(i)}) z_q^{(i)} (1 - z_q^{(i)})\, w_{jq}^{(2)}\right) y_j^{(i)}\, (1 - y_j^{(i)})\, x_k^{(i)}, \end{aligned}$$

where we use $\sigma'(x) = \sigma(x)\,(1 - \sigma(x))$. Hence,

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \delta_j^{(i)} y_k^{(i)} \quad \text{and} \quad \frac{\partial E}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N} \Delta_j^{(i)} x_k^{(i)}$$

where

$$\delta_j^{(i)} = (z_j^{(i)} - \zeta_j^{(i)})\, z_j^{(i)}\, (1 - z_j^{(i)}) \tag{2.2}$$

$$\Delta_j^{(i)} = y_j^{(i)} (1 - y_j^{(i)}) \textstyle\sum_q \delta_q^{(i)} w_{jq}^{(2)}. \tag{2.3}$$

A well-known method of minimizing E is the gradient descent method, where

$$w_{kj}^{(i)}(t+1) = w_{kj}^{(i)}(t) - \varphi_t \frac{\partial E}{\partial w_{kj}^{(i)}}.$$

The Back-propagation Algorithm is based on such a method.

Back-propagation Algorithm for One-Layer $\boldsymbol{\sigma}$-Perceptrons: Initialize the weights $w_{kj}^{(i)}$ randomly. Set $i = 1$, $t = 0$ and choose $\varphi_t > 0$, $\varepsilon > 0$.

STEP 1: FEED-FORWARD CALCULATION. The input vector $x^{(i)}$ is fed into the network. Compute $y_k^{(i)}$ and $z_j^{(i)}$.

STEP 2: BACK-PROPAGATION TO THE HIDDEN LAYER. Calculate $\delta_j^{(i)}$ according to (2.2).

STEP 3: BACK-PROPAGATION TO THE INPUT LAYER. Calculate $\Delta_j^{(i)}$ according to (2.3). Augment i by one. If $i > N$, go to Step 4, otherwise go to Step 1.

STEP 4: EVALUATION AND UPDATE OF WEIGHTS. If $\frac{1}{2}\sum_{i=1}^{N} \|z^{(i)} - \zeta^{(i)}\|_2^2 < \varepsilon$, then stop. Otherwise, define corrected weights by

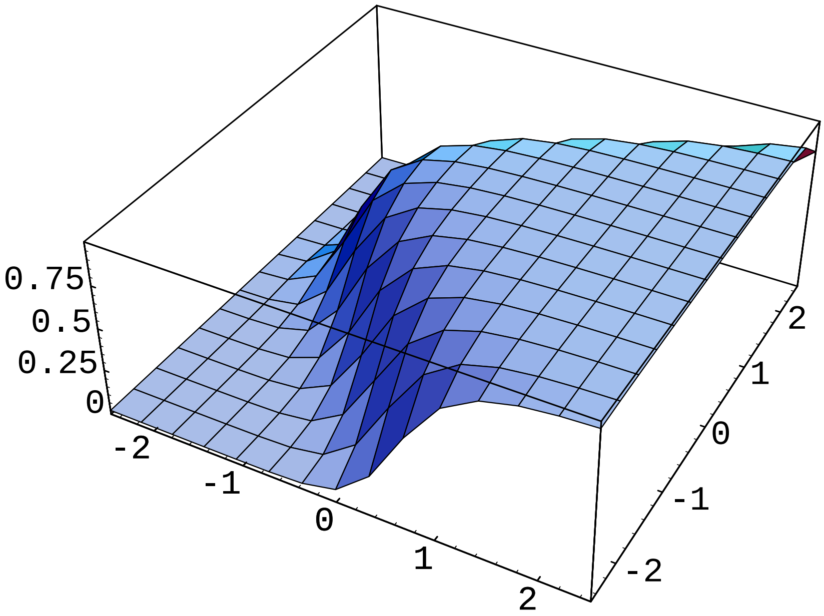
$$w_{kj}^{(2)}(t+1) = w_{kj}^{(2)}(t) - \varphi_t \sum_{i=1}^{N} \delta_j^{(i)} y_k^{(i)}$$

$$w_{kj}^{(1)}(t+1) = w_{kj}^{(1)}(t) - \varphi_t \sum_{i=1}^{N} \Delta_j^{(i)} x_k^{(i)}.$$

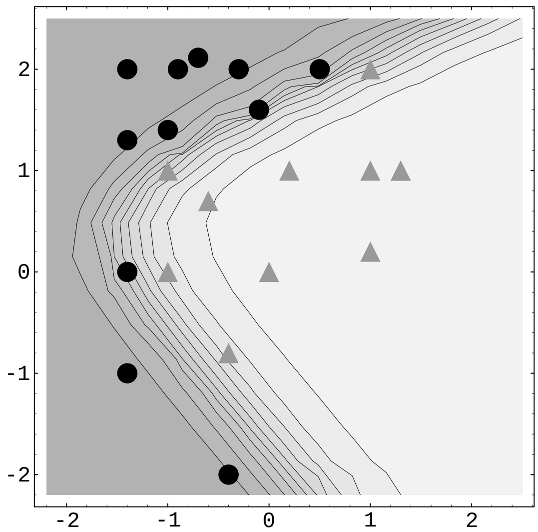
Set $i = 1$ and choose $\varphi_{t+1} > 0$. Augment t by one and go to Step 1.

It should be clear that in each iteration $t = 0, 1, \ldots,$ the current weights $w_{kj}^{(i)}(t)$ are used in Steps 1 and 3. The algorithm is constructed in such a way that each node only needs local information for the calculation of the gradient.

Remark: As is the case with all steepest descent methods, the convergence to a global minimum cannot be guaranteed by the Back-propagation Algorithm. There are various heuristic methods that improve the efficiency of the Back-propagation Algorithm by, for example, a variable step–length φ_t, or by the introduction of a momentum term. For details, see standard textbooks on optimization.



(*a*)



(*b*)

Figure 2.5: *The separation of sets by back-propagation. The network has 2 input nodes, 5 hidden neurons and one output neuron. The function (a) is determined within 4000 iteration steps by the Back-propagation Algorithm with momentum* $\alpha = 0.9$ *and variable step-length* φ*. The contour plot (b) of the function shows the separation produced by back-propagation.*

Chapter 3

Recurrent Networks

A recurrent network consists of formal neurons whose interconnection structure is given by a graph. In contrast to the previous chapter, no assumption is made about the acyclicity of this underlying network graph. Therefore, there is no well-defined direction of information propagation, and thus recurrent networks are modeled as dynamical systems. We start by reviewing some basic facts from the theory of discrete dynamical systems, and then we study the structural properties of Hopfield networks with respect to convergence to fixed points, transient length, and attractivity.

3.1 Finite Automata

Let X be a set and let $f : \mathbb{Z} \times X \to X$ be a function. Consider the difference equation

$$x(t+1) = f(t, x(t)) \tag{3.1}$$

for $t \in \mathbb{Z}$. Together with an initial condition $x(t_0) = x_0$, where $t_0 \in \mathbb{Z}$ and $x_0 \in X$, this recursively defines a unique sequence

$$x = (x(t_0), x(t_0+1), x(t_0+2), \dots)$$

of elements of X. Such sequences are called **trajectories** of (3.1), and the set of all trajectories is called a **discrete dynamical system**. We call $x(t)$ the **state** of the system at time t (when starting in state x_0 at time t_0). If X is finite, such a system is called a **finite automaton**.

A point $\bar{x} \in X$ is called a **fixed point** of (3.1) if there exists $t_0 \in \mathbb{Z}$ such that $x(t_0) = \bar{x}$ implies that

$$x(t+1) = x(t) \quad \text{for all } t_0 \leq t \in \mathbb{Z}.$$

that is, $x(t) = \bar{x}$ for all $t \geq t_0$. This is equivalent to saying that there exists $t_0 \in \mathbb{Z}$ with

$$f(t, \bar{x}) = \bar{x} \quad \text{for all } t_0 \leq t \in \mathbb{Z}.$$

For an integer $k \geq 1$, a vector $(\bar{x}_0, \dots, \bar{x}_{k-1}) \in X^k$ is called a **cycle** of (3.1) of length k if there is a $t_0 \in \mathbb{Z}$ such that $x(t_0) = \bar{x}_0$ implies $x(t_0 + l) = \bar{x}_l$ for $l = 0, \dots, k-1$ and $x(t+k) = x(t)$ for all $t_0 \leq t \in \mathbb{Z}$. A fixed point is precisely a cycle of length one.

If $f : X \to X$ does not depend on t, we write

$$x(t+1) = f(x(t)) \tag{3.2}$$

and we call such an equation **autonomous**. Then it suffices to consider the initial time $t_0 = 0$, that is, trajectories $x = (x(0), x(1), x(2), \dots)$, where $x(t) = f^t(x(0))$ for all $t \in \mathbb{N}$. Here, f^t denotes the t-fold composition of f with itself. The fixed points of (3.2) are characterized by the equation $f(\bar{x}) = \bar{x}$, i.e., they are precisely the fixed points of f, and a cycle corresponds to a vector $(\bar{x}_0, \dots, \bar{x}_{k-1})$ with

$$f(\bar{x}_0) = \bar{x}_1, \dots, f(\bar{x}_{k-2}) = \bar{x}_{k-1}, f(\bar{x}_{k-1}) = \bar{x}_0.$$

A function $E : X \to \mathbb{R}$ is called a **Lyapunov function** for f if

$$E(f(x)) \leq E(x) \quad \text{for all } x \in X.$$

Let $(x(0), x(1), x(2), \dots)$ be a trajectory of (3.2). Then a Lyapunov function E satisfies

$$E(x(t+1)) \leq E(x(t)) \quad \text{for all } t \in \mathbb{N},$$

that is, E decreases along the trajectories of (3.2). If we even have

$$x \neq f(x) \quad \Rightarrow \quad E(f(x)) < E(x),$$

then we say that E is a **strict** Lyapunov function for f. This signifies that along a trajectory x of (3.2), E decreases strictly whenever the state changes:

$$x(t) \neq x(t+1) \quad \Rightarrow \quad E(x(t+1)) < E(x(t)).$$

Theorem 3.1 *If X is finite, the existence of a strict Lyapunov function for f implies that any trajectory of (3.2) reaches a fixed point.*

Proof: Since E can only take finitely many values, there can be only a finite number of changes in the trajectory $x = (x(0), x(1), x(2), \dots)$, that is, there are only finitely many times steps t for which $x(t) \neq x(t+1)$. Let T be the largest integer for which $x(T) \neq x(T+1)$, then $x(T+1) =: \bar{x}$ must be a fixed point. □

Let x be a trajectory of (3.1). The number of time steps t with $x(t) \neq x(t+1)$ is called the **transient length** of x. Clearly, a trajectory reaches a fixed point if and only if its transient length is finite.

Now let X be equipped with a metric $d : X \times X \to \mathbb{R}$. For $\bar{x} \in X$ and $r > 0$, let

$$B_r(\bar{x}) := \{x \in X \ : \ d(x, \bar{x}) \leq r\}.$$

We say that B is a **neighborhood** of $\bar{x}$ if $\bar{x} \in B$ but $B \neq \{\bar{x}\}$. Now let $\bar{x}$ be a fixed point of (3.1). We say that $\bar{x}$ is **attractive** if there exists $t_0 \in \mathbb{Z}$ and a real number $r > 0$ such that $B_r(\bar{x})$ is a neighborhood of $\bar{x}$ and

$$x_0 \in B_r(\bar{x}) \quad \Rightarrow \quad \lim_{t\to\infty} x(t) = \bar{x}, \tag{3.3}$$

where $x(t)$ is the state at time t when starting in state x_0 at time t_0.

In the autonomous case, a fixed point $\bar{x}$ is attractive if there exists r as above with

$$x_0 \in B_r(\bar{x}) \quad \Rightarrow \quad \lim_{t\to\infty} f^t(x_0) = \bar{x}.$$

If X is additionally finite, this simplifies to

$$x_0 \in B_r(\bar{x}) \quad \Rightarrow \quad \exists t \in \mathbb{N} : f^t(x) = \bar{x}.$$

The largest $r \in \mathrm{im}(d)$ with this property is called the **radius of attraction** of $\bar{x}$. Similarly, the largest r as above with

$$x \in B_r(\bar{x}) \quad \Rightarrow \quad f(x) = \bar{x}$$

is called the **radius of direct attraction** of $\bar{x}$, and the set

$$\{x \in X \ : \ f(x) = \bar{x}\}$$

is called the **domain of direct attraction** of $\bar{x}$.

3.2 Structure and Convergence of Recurrent Networks

Definition 3.1 A recurrent network $\mathcal{R}$ consists of a (finite, directed, simple) graph $G = (V, E)$ and a family of formal neurons $(X_i, Y_i, \sigma_i, s_i)$, each associated to one of the vertices $i \in V$. We suppose that for some set $Q \subseteq \mathbb{R}$, $X_i = Q^{n_i}$ and $Y_i = Q$ for all i, where $n_i = |P(i)|$. Then each neuron has a transfer function $f_i : Q^{n_i} \to Q$. Let $n = |V|$. For $t \in \mathbb{N}$, and $i = 1, \ldots, n$, we set

$$q_i(t+1) = f_i(\,(q_j(t))_{j \in P(i)}\,)$$

and we call $q_i(t) \in Q$ the state of neuron i at time t. Accordingly, the vector $q(t) = (q_1(t), \ldots, q_n(t))$ is called the state of the network at time t.

Definition 3.2 A recurrent network is called a **Hopfield network** if the graph G is fully interconnected, that is, $E = V \times V$, and if all the neurons are perceptrons. Then $Q = \{0, 1\}$ and

$$q_i(t+1) = f_i(q_1(t), \ldots, q_n(t)),$$

which can be written in the concise form (putting $f = (f_1, \ldots, f_n)$)

$$q(t+1) = f(q(t))$$

where

$$f(x) = \underline{\mathrm{sat}}(Wx - \theta)$$

for some weight matrix $W \in \mathbb{R}^{n \times n}$ and some threshold vector $\theta \in \mathbb{R}^n$.

In the language of the previous section, the **Hopfield network equation**

$$q(t+1) = \underline{\mathrm{sat}}(Wq(t) - \theta)$$

defines an autonomous finite automaton with $X = \{0, 1\}^n$. A point $\bar{x} \in X$ is a fixed point of this equation if

$$\bar{x} = \underline{\mathrm{sat}}(W\bar{x} - \theta).$$

In the following, we will tacitly assume that W and θ are such that the components of $Wx - \theta$ are non-zero for all $x \in X = \{0, 1\}^n$. This can always be achieved by a modification of the threshold vector, without changing the values of $f(x) = \underline{\mathrm{sat}}(Wx - \theta)$ on X.

Another assumption frequently made below is that W is a symmetric matrix. This means that the strength of interconnection between neuron i and neuron j equals the strength of interconnection between j and i.

Theorem 3.2 *Let W be symmetric. Then each trajectory of a Hopfield network reaches a fixed point or a cycle of length 2.*

Proof: Define $F : X \to \mathbb{R}$ by

$$F(x) = -f(x)^T W x + (f(x)^T + x^T)\theta.$$

Then

$$F(f(x)) = -f(f(x))^T W f(x) + (f(f(x))^T + f(x)^T)\theta.$$

Using the symmetry of W, this yields

$$\begin{aligned} F(f(x)) - F(x) &= (x^T - f(f(x))^T)(Wf(x) - \theta) \\ &= \sum_{i=1}^{n} (x_i - f_i(f(x)))(Wf(x) - \theta)_i. \end{aligned}$$

For each of these summands, there are four cases, summarized in the following table:

x_i	$f_i(f(x))$	$x_i - f_i(f(x))$	$(Wf(x) - \theta)_i$
0	0	0	< 0
1	0	1	< 0
0	1	-1	> 0
1	1	0	> 0

The last column is obtained by noting that

$$f(f(x)) = \underline{\mathrm{sat}}(Wf(x) - \theta)$$

and thus

$$f_i(f(x)) = \mathrm{sat}(Wf(x) - \theta)_i.$$

We conclude that

$$x \neq f(f(x)) \quad \Rightarrow \quad F(f(x)) < F(x).$$

Since F can only take finitely many values, there exist, for every trajectory $x = (x(0), x(1), x(2), \dots)$ only finitely many time steps t with

$$x(t+2) \neq x(t).$$

Thus there exists a $t_0 \in \mathbb{N}$ such that $x(t+2) = x(t)$ for all $t \geq t_0$. □

Theorem 3.3 *Let W be symmetric and assume that $x^T W x \geq 0$ for all x in $\{-1, 0, 1\}^n$. Then*

$$E(x) = -\frac{1}{2} x^T W x + x^T \theta$$

is a strict Lyapunov function for the Hopfield network. According to Theorem 3.1, this implies that each trajectory reaches a fixed point.

Proof: Using the symmetry of W, we obtain

$$E(f(x)) - E(x) = -\frac{1}{2}(x^T - f(x)^T)W(x - f(x)) + (x^T - f(x)^T)(Wx - \theta).$$

Since $x, f(x) \in \{0,1\}^n$, we have $x - f(x) \in \{-1,0,1\}^n$ and therefore the first summand is non-positive due to our assumption on W. For the second summand, we consider again the possible cases for a particular component:

x_i	$f_i(x)$	$x_i - f_i(x)$	$(Wx-\theta)_i$
0	0	0	< 0
1	0	1	< 0
0	1	-1	> 0
1	1	0	> 0

The last column comes again from

$$f(x) = \underline{\mathrm{sat}}(Wx - \theta)$$

that is,

$$f_i(x) = \mathrm{sat}(Wx - \theta)_i.$$

Thus we obtain

$$x \neq f(x) \quad \Rightarrow \quad E(f(x)) < E(x)$$

as desired. $\square$

3.3 Asynchronous Update

The sufficient condition of Theorem 3.3 is often too restrictive. One way out is to compute the new state vector $q(t+1) = f(q(t))$ component for component, and to use the already updated components $q_1(t+1), \ldots, q_{i-1}(t+1)$ for the calculation of $q_i(t+1)$. The resulting computation procedure is:

$$\begin{array}{rcl} q_1(t+1) & = & f_1(q_1(t), q_2(t), \ldots, q_n(t)) \\ q_2(t+1) & = & f_2(q_1(t+1), q_2(t), \ldots, q_n(t)) \\ \vdots & = & \vdots \\ q_n(t+1) & = & f_n(q_1(t+1), \ldots, q_{n-1}(t+1), q_n(t)). \end{array} \tag{3.4}$$

We want to recast these equations in the form discussed in Section 3.1. For this, we set $h = 1/n$ and $x(t) = q(t)$,

$$x(t+h) = \begin{bmatrix} f_1(x(t)) \\ q_2(t) \\ \vdots \\ q_n(t) \end{bmatrix}, \ldots, x(t+(n-1)h) = \begin{bmatrix} f_1(x(t)) \\ \vdots \\ f_{n-1}(x(t+(n-2)h)) \\ q_n(t) \end{bmatrix}$$

and $x(t+nh) = x(t+1) = q(t+1)$. After re-scaling the time axis, we have

$$x_i(t+1) = \begin{cases} f_i(x(t)) & \text{if } i \equiv t+1 \bmod n \\ x_i(t) & \text{otherwise.} \end{cases} \tag{3.5}$$

This model is equivalent to (3.4), and it has the form discussed in Section 3.1. However, note that the equation is not autonomous! We call (3.5) a **sequential updating scheme**.

More generally, for a sequence $h : \mathbb{N} \to \{1, \dots, n\}$, we call

$$x_i(t+1) = \begin{cases} f_i(x(t)) & \text{if } i = h(t) \\ x_i(t) & \text{otherwise} \end{cases}$$

an **asynchronous updating scheme**. Usually, one requires that h is such that for all $i \in \{1, \dots, n\}$, and for all $k_0 \in \mathbb{N}$, there exists $k \geq k_0$ such that $h(k) = i$. This guarantees that that every component is updated infinitely many times. This notion should be compared with the concept of a training sequence, used in Chapter 1. The sequential updating scheme corresponds to the special case where $h = (1, \dots, n, 1, \dots, n, \dots)$. For practical purposes, it suffices to consider the sequential update.

Consider the Hopfield network equation with sequential update, that is,

$$x_i(t+1) = \begin{cases} \text{sat}(Wx(t) - \theta)_i & \text{if } i \equiv t+1 \bmod n \\ x_i(t) & \text{otherwise.} \end{cases} \tag{3.6}$$

First, we note that the notion of fixed points does not depend on the choice of the updating mode.

Lemma 3.4 *A point $\bar{x} \in X = \{0,1\}^n$ is a fixed point of (3.6) if and only if $\bar{x} = \underline{\text{sat}}(W\bar{x} - \theta)$.*

Theorem 3.5 *Let W be symmetric, with $W_{ii} \geq 0$ for all i. Then every trajectory of (3.6) reaches a fixed point.*

Proof: Consider again the energy function

$$E(x) = -\frac{1}{2} x^T W x + x^T \theta.$$

Let x be a trajectory of (3.6) and suppose that $x(t) \neq x(t+1)$. As usual, it suffices to show that $E(x(t+1)) < E(x(t))$. We have

$$\begin{aligned} E(x(t+1)) - E(x(t)) &= -\tfrac{1}{2}(x(t)^T - x(t+1)^T) W (x(t) - x(t+1)) \\ &\quad + (x(t)^T - x(t+1)^T)(Wx(t) - \theta). \end{aligned} \tag{3.7}$$

Due to the asynchronous update, $x(t)$ and $x(t+1)$ differ only in one component, say $x(t)-x(t+1) = \pm e_i$, where e_i is the i-th natural basis vector of $\mathbb{R}^n$. Therefore

$$E(x(t+1)) - E(x(t)) = -\frac{1}{2}W_{ii} \pm (Wx(t) - \theta)_i.$$

If $x(t)-x(t+1) = +e_i$, then $x_i(t) = 1$ and $x_i(t+1) = 0$ and thus $(Wx(t)-\theta)_i < 0$. The case $x(t) - x(t+1) = -e_i$ is analogous. □

Remarks

1. Note that

$$x^T Wx \geq 0 \text{ for all } x \in \{-1, 0, 1\}^n \quad \Rightarrow \quad W_{ii} \geq 0 \text{ for all } i.$$

   Therefore, the sufficient condition of Theorem 3.3 is strictly stronger than that of Theorem 3.5, showing that asynchronous update is really an advantage.

2. The condition $W_{ii} \geq 0$ is met by the "classical" Hopfield network, which even requires $W_{ii} = 0$. This means that no neuron in the network uses its own output as an input, which corresponds to a graph $G = (V, E)$ without self-loops, that is, $(i, i) \notin E$ for all $i \in V$.

3.4 Transient Length and Attractivity

For the following discussion, it is convenient to replace $X = \{0, 1\}^n$ by $X = \{\pm 1\}^n$ and correspondingly, we replace the Heaviside function by the sign function. All the results obtained so far remain valid.

Theorem 3.6 *Let W be symmetric, with $W_{ii} \geq 0$ for all i. According to Theorem 3.5, all trajectories of the Hopfield network with sequential update (3.6) reach a fixed point. Moreover, the transient length T satisfies*

$$T \leq \frac{\sum_{i=1}^n \sum_{j>i} |W_{ij}| + \sum_{i=1}^n |\theta_i|}{w + \varepsilon},$$

where $w := \min W_{ii} \geq 0$ and $\varepsilon > 0$ is such that

$$|(Wx - \theta)_i| \geq \varepsilon$$

for all $x \in X = \{\pm 1\}^n$ and all $i = 1, \ldots, n$.

Proof: Consider once more

$$E(x) = -\frac{1}{2}x^T W x + x^T\theta$$

and let

$$\Delta = \max_{x\in X} E(x) - \min_{x\in X} E(x).$$

If $\delta > 0$ is such that for any trajectory $x = (x(0), x(1), x(2), \dots)$

$$x(t+1) \neq x(t) \quad \Rightarrow \quad |E(x(t+1)) - E(x(t))| \geq \delta,$$

then the number of time steps t with $x(t+1) \neq x(t)$ is bounded by Δ/δ. It suffices to find an upper bound for Δ. For this, note that for $x \in X$,

$$\begin{aligned} E(x) &= -\frac{1}{2}\sum_{i=1}^n W_{ii}x_i^2 - \sum_{i=1}^n \sum_{j>i} W_{ij}x_i x_j + x^T\theta \\ &= -\frac{1}{2}\sum_{i=1}^n W_{ii} - \sum_{i=1}^n \sum_{j>i} W_{ij}x_i x_j + x^T\theta \end{aligned}$$

and thus, setting $\tilde{E}(x) = E(x) + \frac{1}{2}\sum_{i=1}^n W_{ii}$,

$$\Delta = \max_{x\in X} \tilde{E}(x) - \min_{x\in X} \tilde{E}(x).$$

One obtains

$$\Delta \leq 2\left(\sum_{i=1}^n \sum_{j>i} |W_{ij}| + \sum_{i=1}^n |\theta_i|\right).$$

On the other hand, when $x(t+1) \neq x(t)$, say, $x(t) - x(t+1) = \pm 2e_i$, we get from (3.7)

$$E(x(t+1)) - E(x(t)) = -2W_{ii} \pm 2(Wx(t) - \theta)_i$$

and thus

$$|E(x(t+1)) - E(x(t))| = 2(W_{ii} + |(Wx(t) - \theta)_i|) \geq 2(w + \varepsilon) =: \delta.$$

This yields the result. □

For discussing attractivity, we need to introduce a metric on X.

Definition 3.3 For $x, y \in X = \{\pm 1\}^n$,

$$d(x,y) = |\{i \,:\, x_i \neq y_i\}|$$

is called the **Hamming distance** between x and y.

Lemma 3.7 *We have:*

1. *d is a metric on X.*
2. *$d(x,y) \in \mathbb{N}$ and $0 \leq d(x,y) \leq n$.*
3. *$d(x,y) = \frac{1}{2}(n - \langle x, y \rangle)$.*
4. *$B_r(\bar{x})$ is a neighborhood of $\bar{x} \in X$ iff $r \geq 1$.*

Let $\bar{x}$ be a fixed point of the Hopfield network, that is, $\bar{x} = \underline{\text{sign}}(W\bar{x} - \theta)$. Let $r_1(\bar{x})$ denote the radius of direct attraction of $\bar{x}$, which is defined to be the largest integer $r \geq 1$ with

$$x_0 \in B_r(\bar{x}) \quad \Rightarrow \quad \underline{\text{sign}}(Wx_0 - \theta) = \bar{x}.$$

In that case, if we start in $x_0 \in B_r(\bar{x})$ and use a sequential update, the trajectory will reach $\bar{x}$ after at most n time steps (recall that n sequential time steps correspond to one synchronous time step).

Theorem 3.8 *Let $\bar{x}$ be a fixed point of the Hopfield network, and assume that $W \neq 0$. Let $\varepsilon > 0$ be such that*

$$|(W\bar{x} - \theta)_i| \geq \varepsilon$$

for $i = 1, \ldots, n$. Then

$$r_1(\bar{x}) \geq \left[\frac{\varepsilon}{2v}\right]$$

where $v = \max |W_{ij}|$, and $[x]$ denotes the greatest integer less than or equal to a real number x.

Proof: For any $x, \bar{x} \in X$ and any $i = 1, \ldots, n$, we have

$$\begin{aligned} |(Wx - \theta)_i - (W\bar{x} - \theta)_i| &= |(W(x - \bar{x}))_i| \\ &= |\sum_{j=1}^{n} W_{ij}(x_j - \bar{x}_j)| \\ &\leq \sum_{j=1}^{n} |W_{ij}||x_j - \bar{x}_j| \\ &\leq 2vd(x, \bar{x}). \end{aligned}$$

Therefore, if $d(x, \bar{x}) \leq \frac{\varepsilon}{2v}$, then

$$|(Wx - \theta)_i - (W\bar{x} - \theta)_i| \leq \varepsilon$$

which implies that $(Wx - \theta)_i$ and $(W\bar{x} - \theta)_i$ have the same sign, and hence

$$\underline{\text{sign}}(Wx - \theta) = \underline{\text{sign}}(W\bar{x} - \theta) = \bar{x}.$$

□

Chapter 4

Associative Memory Problem

The most important application of a Hopfield network is its use as an associative memory. We review two well-known design strategies, namely the Hebb rule and the projection rule. Then we point out some connections between the two. This leads to a learning rule for the associative memory problem.

We wish to use a recurrent network in order to store information ("memory"). Certain patterns are to be memorized and they should be implemented as fixed points of the network. The term "associative" refers to the requirement that the network should be able to reproduce the stored information even if the input is perturbed, provided that the disturbance is small enough.

The recurrent network to be designed should have the following properties:

- Each pattern to be stored should be a fixed point of the network.
- The radius of attraction of these fixed points should be as large as possible.
- Given an arbitrary initial state, the network should converge to the closest fixed point, and it should do so as fast as possible.
- The number of fixed points that are not among the patterns to be stored ("spurious fixed points") should be kept small.

4.1 Fixed Point Synthesis

Let us start with the most basic requirement. Given a family of patterns

$$\xi^{(1)}, \ldots, \xi^{(N)} \in X = \{\pm 1\}^n,$$

we wish to design a Hopfield network that has the $\xi^{(j)}$ as fixed points, that is, we need to construct a weight matrix $W \in \mathbb{R}^{n\times n}$ and a threshold vector $\theta \in \mathbb{R}^n$ such that

$$\xi^{(j)} = \underline{\text{sign}}(W\xi^{(j)} - \theta) \quad \text{for } j = 1, \ldots, N.$$

Thus we need to find a solution (W, θ) to the inequalities

$$\xi_i^{(j)} = \text{sign}(W\xi^{(j)} - \theta)_i$$

or

$$\xi_i^{(j)}(W\xi^{(j)} - \theta)_i = \xi_i^{(j)} \left(\sum_{k=1}^{n} W_{ik}\xi_k^{(j)} - \theta_i \right) > 0$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, N$. This can be reformulated as a linear separation problem, and therefore a solution (if there exists one) can be found by means of the perceptron learning algorithm. In the following, we will discuss more systematic ways to construct W and θ. This will yield a deeper insight into the behavior of the resulting networks.

4.2 Hebb Rule

Let $\xi^{(1)}, \ldots, \xi^{(N)} \in X = \{\pm 1\}^n$ be given. Define a matrix

$$U = (\xi^{(1)}, \ldots, \xi^{(N)}) \in \{\pm 1\}^{n\times N}. \tag{4.1}$$

The **Hebb rule** suggests to choose

$$W = UU^T = \sum_{j=1}^{N} \xi^{(j)}\xi^{(j)T} \quad \text{and} \quad \theta = 0. \tag{4.2}$$

In the literature, we find several variants, for instance, the **weighted** version

$$W = U\Lambda U^T = \sum_{j=1}^{N} \lambda_j \xi^{(j)}\xi^{(j)T} \quad \text{and} \quad \theta = 0,$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix. Often, this is simply used for a scaling of the entries of W, e.g., by choosing $\Lambda = \frac{1}{N}I$ one can guarantee that $|W_{ij}| \leq 1$ for all i, j.

Here, we stick to the simplest case (4.2). An immediate observation is that according to Hebb's rule, the weight matrix is always symmetric and positive semi-definite. Based on the results of the previous chapter, we have the following conclusion.

Lemma 4.1 *All trajectories of a Hopfield network designed according to the Hebb rule reach a fixed point (in the synchronous as well as in the sequential updating mode).*

However, note that the trajectories do not necessarily converge to one of the given states $\xi^{(1)}, \ldots, \xi^{(N)}$. We cannot even guarantee that every $\xi^{(j)}$ will be a fixed point of the network. At least, we can give a sufficient condition.

Theorem 4.2 *Let the patterns be such that*

$$n > \sum_{\substack{j=1 \\ j \neq k}}^{N} |\langle \xi^{(j)}, \xi^{(k)} \rangle|$$

for all $k = 1, \ldots, N$. *Then each vector* $\xi^{(k)}$ *is a fixed point of the Hopfield network resulting from Hebb's rule.*

Proof: We have

$$W\xi^{(k)} = \sum_{j=1}^{N} \xi^{(j)} \xi^{(j)T} \xi^{(k)} = \sum_{j=1}^{N} \xi^{(j)} \langle \xi^{(j)}, \xi^{(k)} \rangle = n\xi^{(k)} + \sum_{j \neq k} \xi^{(j)} \langle \xi^{(j)}, \xi^{(k)} \rangle$$

and thus

$$|(W\xi^{(k)} - n\xi^{(k)})_i| = |\sum_{j \neq k} \xi_i^{(j)} \langle \xi^{(j)}, \xi^{(k)} \rangle| \leq \sum_{j \neq k} |\langle \xi^{(j)}, \xi^{(k)} \rangle| < n.$$

Therefore, $(W\xi^{(k)})_i$ and $n\xi_i^{(k)}$ have the same sign, in other words,

$$\underline{\mathrm{sign}}(W\xi^{(k)}) = \underline{\mathrm{sign}}(n\xi^{(k)}) = \underline{\mathrm{sign}}(\xi^{(k)}) = \xi^{(k)}.$$

□

The subsequent results follow from Theorems 3.6 and 3.8.

Corollary 4.3 *If a Hopfield network is constructed according to the Hebb rule, and if a sequential update is used, then the transient length satisfies*

$$T \leq \frac{Nn(n-1)}{2(N+\varepsilon)}, \tag{4.3}$$

where ε is such that

$$|(Wx)_i| \geq \varepsilon$$

for all $x \in X = \{\pm 1\}^n$ and all $i = 1, \ldots, n$.

Proof: In view of Theorem 3.6, it suffices to note that $|W_{ij}| \leq N$ and $W_{ii} = N$ for all i, j. Therefore

$$\sum_{i=1}^{n} \sum_{j>i} |W_{ij}| \leq \frac{Nn(n-1)}{2}$$

and we may put $w = N$. □

Corollary 4.4 *In the situation of Theorem 4.2, let*

$$M_k := n - \sum_{\substack{j=1 \\ j \neq k}}^{N} |\langle \xi^{(j)}, \xi^{(k)} \rangle| > 0$$

for all $k = 1, \ldots, N$ and set $M = \min M_k$. The radius of direct attraction of the fixed point $\xi^{(k)}$ satisfies

$$r_1(\xi^{(k)}) \geq \left[\frac{M}{2N}\right].$$

Therefore, if the patterns are such that $M \geq 2N$, then $\xi^{(k)}$ is an attractive fixed point.

Proof: Noting that $|W_{ij}| \leq N$ and $W_{ii} = N$ for all i, j, we obtain

$$N = \max |W_{ij}|.$$

Since

$$\begin{aligned}
|(W\xi^{(k)})_i| &= |n\xi_i^{(k)} + \sum_{j \neq k} \xi_i^{(j)} \langle \xi^{(j)}, \xi^{(k)} \rangle| \\
&\geq n - |\sum_{j \neq k} \xi_i^{(j)} \langle \xi^{(j)}, \xi^{(k)} \rangle| \\
&\geq n - \sum_{j \neq k} |\langle \xi^{(j)}, \xi^{(k)} \rangle| \;=\; M_k \;\geq\; M,
\end{aligned}$$

the result follows from Theorem 3.8 putting $v = N$ and $\varepsilon = M$. □

Corollary 4.5 *If the patterns are orthogonal, i.e., if*

$$j \neq k \quad \Rightarrow \quad \langle \xi^{(j)}, \xi^{(k)} \rangle = 0, \tag{4.4}$$

then each $\xi^{(k)}$ is a fixed point of the Hopfield network constructed according to Hebb's law. Moreover,

$$r_1(\xi^{(k)}) \geq \left[\frac{n}{2N}\right].$$

If $n \geq 2N$, this implies that $\xi^{(k)}$ is attractive.

This corollary shows that it is desirable to have orthogonal patterns. This can be achieved by appending extra components to the patterns, i.e., by increasing n, the size of the network.

An advantage of the Hebb rule is that the weight matrix is particularly easy to compute. Moreover, it can be adapted without much effort when a new pattern $\xi^{(N+1)}$ is added; one simply puts

$$W_{\text{new}} = W + \xi^{(N+1)}\xi^{(N+1)T}.$$

Similarly, patterns may be removed. The biggest disadvantage is that in general, not every $\xi^{(j)}$ is a fixed point of the network. Therefore, an alternative approach has been proposed, called the projection or pseudo-inverse rule.

4.3 Projection Rule

Let $\xi^{(1)}, \dots, \xi^{(N)} \in \{\pm 1\}^n$ be given, and let the matrix U be defined as in (4.1). The **projection rule** says that

$$W = UU^+ \quad \text{and} \quad \theta = 0,$$

where U^+ is the pseudo-inverse of U, see Appendix B. Then W is a symmetric projection operator and therefore positive semi-definite.

Lemma 4.6 *All trajectories of a Hopfield network designed according to the projection rule reach a fixed point (in the synchronous as well as in the sequential updating mode).*

Theorem 4.7 *Each vector $\xi^{(k)}$ is a fixed point of the Hopfield network resulting from the pseudo-inverse rule.*

Proof: We have $WU = UU^+U = U$ and therefore $W\xi^{(k)} = \xi^{(k)}$ for $k = 1, \dots, N$. Thus

$$\underline{\text{sign}}(W\xi^{(k)}) = \underline{\text{sign}}(\xi^{(k)}) = \xi^{(k)}.$$

$\square$

Remark: For $i = 1, \dots, n$, we have

$$W_{ii} - W_{ii}^2 = \sum_{j \neq i} W_{ij}^2$$

and therefore $0 \leq W_{ii} \leq 1$ and $\sum_{j\neq i} W_{ij}^2 \leq \frac{1}{4}$. In fact, we even have $W_{ii} > 0$. (If W_{ii} was zero, then the whole i-th row of W would be zero. This contradicts $WU = U$, since the entries of U are in $\{\pm 1\}$ and therefore, no row of U can be zero.) Moreover, since $\|x\|_1 \leq \sqrt{m}\|x\|_2$ holds for any $x \in \mathbb{R}^m$, we obtain

$$\sum_{j\neq i} |W_{ij}| \leq \sqrt{n-1}\left(\sum_{j\neq i} W_{ij}^2\right)^{1/2} \leq \frac{\sqrt{n-1}}{2} \quad \text{for } i = 1, \ldots, n,$$

and thus

$$\sum_{i=1}^{n}\sum_{j\neq i} |W_{ij}| \leq \frac{n\sqrt{n-1}}{2}$$

or

$$\sum_{i=1}^{n}\sum_{j>i} |W_{ij}| \leq \frac{n\sqrt{n-1}}{4}.$$

Now the following result is an immediate consequence of Theorem 3.6.

Corollary 4.8 *If a Hopfield network is constructed using the projection rule, and if a sequential update is used, then the transient length T satisfies*

$$T \leq \frac{n\sqrt{n-1}}{4(w+\varepsilon)} \tag{4.5}$$

where $w = \min W_{ii} > 0$ and ε is such that

$$|(Wx)_i| \geq \varepsilon$$

for all $x \in X = \{\pm 1\}^n$ and all $i = 1, \ldots, n$.

Similarly, we have the following corollary to Theorem 3.8.

Corollary 4.9 *The radius of direct attraction of the fixed point $\xi^{(k)}$ of the Hopfield network designed according to the projection rule satisfies*

$$r_1(\xi^{(k)}) \geq \left\lceil \frac{1}{2v} \right\rceil$$

where $v = \max |W_{ij}|$. If $v \leq \frac{1}{2}$, then $\xi^{(k)}$ is attractive.

Proof: We have

$$|(W\xi^{(k)})_i| = |\xi_i^{(k)}| = 1$$

and thus we may put $\varepsilon = 1$ in Theorem 3.8. □

4.4 Connection between Hebb Rule and Projection Rule

Suppose that the data are orthogonal, as in (4.4). Then

$$U^T U = nI$$

and

$$UU^+ = U(U^T U)^{-1} U^T = \frac{1}{n} UU^T.$$

In this case, the Hebb rule and the projection rule coincide, up to a scaling factor.

In general, the projection method is better than the Hebb rule in the sense that any $\xi^{(k)}$ is a fixed point (without further conditions on the data). Also, we have found a sharper upper bound for the transient length in the case of the projection rule: To make them comparable, we use $\varepsilon \geq 0$ in (4.3) and (4.5), and we get

$$T_{\text{Hebb}} \leq \frac{n(n-1)}{2} \quad \text{and} \quad T_{\text{Proj}} \leq \frac{n\sqrt{n-1}}{4w}.$$

It is clear that the asymptotic behavior of the latter expression is better when n becomes large.

On the other hand, UU^+ is harder to compute than UU^T. Therefore it is desirable to have a simple recursive procedure which updates the weights $W(t)$ such that UU^+ is obtained after finitely many time steps, or at least, in the limit as $t \to \infty$.

Greville's Algorithm

Greville [9] proposed the following incremental technique for calculating the pseudo-inverse in finitely many steps. Let $U = (\xi^{(1)}, \ldots, \xi^{(N)}) \in \mathbb{R}^{n \times N}$ be given. For $k = 1, \ldots, N$, define

$$U_k = (\xi^{(1)}, \ldots, \xi^{(k)}) \in \mathbb{R}^{n \times k}.$$

We have

$$U_1^+ = \begin{cases} \frac{\xi^{(1)T}}{\|\xi^{(1)}\|^2} & \text{if } \xi^{(1)} \neq 0 \\ 0 & \text{if } \xi^{(1)} = 0. \end{cases}$$

If $U \in \{\pm 1\}^{n \times N}$, this simplifies to

$$U_1^+ = \frac{1}{n} \xi^{(1)T}.$$

Theorem 4.10 *We have*

$$U_k^+ = \begin{bmatrix} U_{k-1}^+(I - \xi^{(k)} p_k^T) \\ p_k^T \end{bmatrix}$$

where, with $z(k) := (I - U_{k-1}U_{k-1}^+)\xi^{(k)}$,

$$p_k = \begin{cases} \frac{z(k)}{\|z(k)\|^2} & \text{if } z(k) \neq 0 \\ \frac{U_{k-1}^{+T} U_{k-1}^+ \xi^{(k)}}{1 + \|U_{k-1}^+ \xi^{(k)}\|^2} & \text{if } z(k) = 0. \end{cases}$$

Inductively, this yields a procedure for computing U^+ in N iteration steps. For a proof, see [9]. An important consequence of Greville's theorem is that there is no need to completely re-compute the weight matrix when a new pattern $\xi^{(N+1)}$ is added, similarly as with the Hebb rule.

Neumann Series Approach

This algorithm produces an – in general infinite – sequence of matrices which converges to the pseudo-inverse of a full column rank matrix U. This corresponds to the case where the data vectors are linearly independent. Then

$$U^+ = (U^T U)^{-1} U^T.$$

Define a matrix

$$R := I - 2\varphi U^T U$$

where $0 < \varphi < 1/\|U\|_2^2$. Note that $\|U\|_2^2$ equals the largest eigenvalue of $U^T U$. Therefore all eigenvalues λ of R satisfy $|\lambda| < 1$. In other words, R is discrete-time asymptotically stable. Therefore we may write

$$(I - R)^{-1} = \sum_{i=0}^{\infty} R^i.$$

The right hand side of this equation is called **Neumann series** (the matrix version of a geometric series). By the definition of R, this implies

$$(U^T U)^{-1} = 2\varphi \sum_{i=0}^{\infty} R^i$$

and thus

$$U^+ = 2\varphi \sum_{i=0}^{\infty} R^i U^T.$$

Therefore, the sequence $U^+(t) := 2\varphi \sum_{i=0}^{t} R^i U^T$ converges to U^+. Similarly, since

$$UU^+ = 2\varphi \sum_{i=0}^{\infty} UR^i U^T$$

we define

$$W(t) = 2\varphi \sum_{i=0}^{t} UR^i U^T. \tag{4.6}$$

Then $W(0) = 2\varphi UU^T$ and $W(\infty) := \lim_{t\to\infty} W(t) = UU^+$, that is, $W(0)$ corresponds to the Hebb rule (up to a scaling factor), and $W(\infty)$ corresponds to the projection rule.

4.5 Projection Rule Learning

The previous section suggests the following learning rule in the case of linearly independent patterns:

$$W(0) = 2\varphi UU^T \quad \text{and} \quad W(t) = W(t-1) + 2\varphi UR^t U^T. \tag{4.7}$$

This recursively defines a sequence of weight matrices which converges to UU^+. We may rewrite (4.6) as follows:

$$\begin{aligned} W(t) &= 2\varphi \sum_{i=0}^{t} U(I - 2\varphi U^T U)^i U^T \\ &= 2\varphi \sum_{i=0}^{t} (I - 2\varphi UU^T)^i UU^T \\ &= \sum_{i=0}^{t} (I - W(0))^i W(0). \end{aligned}$$

Therefore (4.7) can be reformulated as

$$W(0) = 2\varphi UU^T \quad \text{and} \quad W(t) = W(t-1) + (I - W(0))^t W(0).$$

If the data are not linearly independent, we use the following alternative learning rule.

Theorem 4.11 *Let $0 \neq U \in \mathbb{R}^{n\times N}$ be given. Define*

$$W(0) = UU^T \quad \textit{and} \quad W(t+1) = W(t) + 2\varphi(I - W(t))UU^T,$$

where $0 < \varphi < 1/\|U\|_2^2$. Then $W(\infty) = \lim_{t\to\infty} W(t) = UU^+$.

Proof: Let

$$U = P\Sigma Q^T, \quad \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}$$

be a singular value decomposition of U. Then

$$UU^+ = P\Sigma\Sigma^+ P^T = P \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} P^T$$

and

$$UU^T = P\Sigma\Sigma^T P^T = P \begin{bmatrix} \Sigma_1^2 & 0 \\ 0 & 0 \end{bmatrix} P^T.$$

We have

$$W(t+1) = W(t)(I - 2\varphi UU^T) + 2\varphi UU^T$$

which implies

$$W(t) = W(0)(I - 2\varphi UU^T)^t + 2\varphi UU^T \sum_{i=0}^{t-1} (I - 2\varphi UU^T)^i.$$

We have

$$I - 2\varphi UU^T = P \begin{bmatrix} I - 2\varphi\Sigma_1^2 & 0 \\ 0 & I \end{bmatrix} P^T.$$

By the assumption on φ, the matrix $S := I - 2\varphi\Sigma_1^2$ is discrete-time asymptotically stable. This yields

$$\begin{aligned} W(t) &= W(0)P \begin{bmatrix} S^t & 0 \\ 0 & I \end{bmatrix} P^T + P \begin{bmatrix} 2\varphi\Sigma_1^2 & 0 \\ 0 & 0 \end{bmatrix} \sum_{i=0}^{t-1} \begin{bmatrix} S^i & 0 \\ 0 & I \end{bmatrix} P^T \\ &= W(0)P \begin{bmatrix} S^t & 0 \\ 0 & I \end{bmatrix} P^T + P \begin{bmatrix} I - S & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} (I-S)^{-1}(I - S^t) & 0 \\ 0 & tI \end{bmatrix} P^T \\ &= W(0)P \begin{bmatrix} S^t & 0 \\ 0 & I \end{bmatrix} P^T + P \begin{bmatrix} I - S^t & 0 \\ 0 & 0 \end{bmatrix} P^T \end{aligned}$$

and therefore

$$W(\infty) = W(0)P \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} P^T + P \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} P^T.$$

Finally, we use that $W(0) = UU^T$ to show that the first summand vanishes, and thus $W(\infty) = UU^+$. □

Remarks

1. The learning rule proposed in Theorem 4.11 is actually another gradient descent method. Consider the energy function

$$E : \mathbb{R}^{n\times n} \to \mathbb{R}, \quad W \mapsto E(W) := \|WU - U\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Clearly, this functional is minimized by any W with $WU = U$, for instance, by $W = UU^+$. We have

$$\frac{\partial E}{\partial W} = 2(W - I)UU^T.$$

Thus the gradient descent approach $W(t+1) = W(t) - \varphi\frac{\partial E}{\partial W}$ leads precisely to the discussed learning rule.

2. A variant of the learning rule of Theorem 4.11 is given by

$$W(t+1) = W(t) + 2\varphi(I - W(t))u(t)u(t)^T$$

where

$$u : \mathbb{N} \to \{\xi^{(1)}, \ldots, \xi^{(N)}\}, \quad t \mapsto u(t)$$

is a training sequence for the data vectors. Recall that this means that any data vector $\xi^{(k)}$ appears infinitely often in the sequence u. This version is more within the neural network spirit, because for learning a set of patterns, a human being would also look at one pattern at a time, repeating this procedure several times (as opposed to looking at all patterns simultaneously).

Chapter 5

Capacity

The concept of capacity plays a vital role in assessing the efficiency of a neural network. Hence, it is the subject of this chapter. First, the capacity of a perceptron will be considered. This is closely related to a famous combinatorial theorem by Schläfli. Then we present some results for multilayer perceptrons.

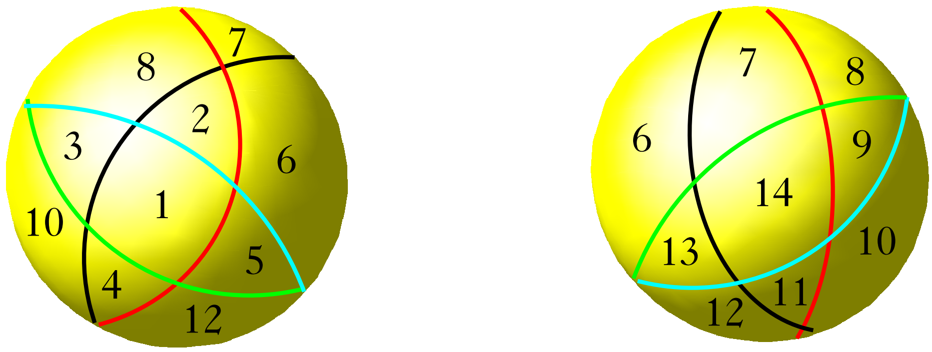
In Chapter 1, we have demonstrated that a Boolean function $f : \{0,1\}^2 \to \{0,1\}$ can be realized by a perceptron if and only if the sets $X_+ = f^{-1}(1)$ and $X_- = f^{-1}(0)$ are affinely separable. We have seen that this is the case for the AND and OR functions, but not for the XOR function. How many of the 16 switching functions $f : \{0,1\}^2 \to \{0,1\}$ can be represented by a perceptron? In order to answer this question, we consider the hyperplanes determined by $\langle w, x\rangle = \theta$ for $x \in \{0,1\}^2$, with $w_3 = -\theta$:

$$\begin{aligned} w_1 \cdot 0 + w_2 \cdot 0 + w_3 &= 0 \\ w_1 \cdot 0 + w_2 \cdot 1 + w_3 &= 0 \\ w_1 \cdot 1 + w_2 \cdot 0 + w_3 &= 0 \\ w_1 \cdot 1 + w_2 \cdot 1 + w_3 &= 0. \end{aligned} \tag{5.1}$$

The OR function is realized, for example, by weights with

$$\begin{aligned} w_1 \cdot 0 + w_2 \cdot 0 + w_3 &< 0 \\ w_1 \cdot 0 + w_2 \cdot 1 + w_3 &\geq 0 \\ w_1 \cdot 1 + w_2 \cdot 0 + w_3 &\geq 0 \\ w_1 \cdot 1 + w_2 \cdot 1 + w_3 &\geq 0. \end{aligned}$$

Thus, the number of switching functions which can be realized corresponds to the number of regions defined by the four hyperplanes (5.1). In Figure 5.1, the resulting regions are illustrated. It turns out that 14 out of 16 Boolean functions of two variables are realizable. What can be said in general?



(a) (b)

Figure 5.1: *The number of switching functions which can be realized by a perceptron where one domain represents one switching function. The XOR function and its negation cannot be calculated by a perceptron; the remaining 14 possibilities can be realized. (a) Front view and (b) rear view of the Boolean sphere.*

5.1 Capacity of Perceptrons

In the following, we shall consider perceptrons with the **sign function** sign : $\mathbb{R} \to \{-1, 0, 1\}$ as output function, where

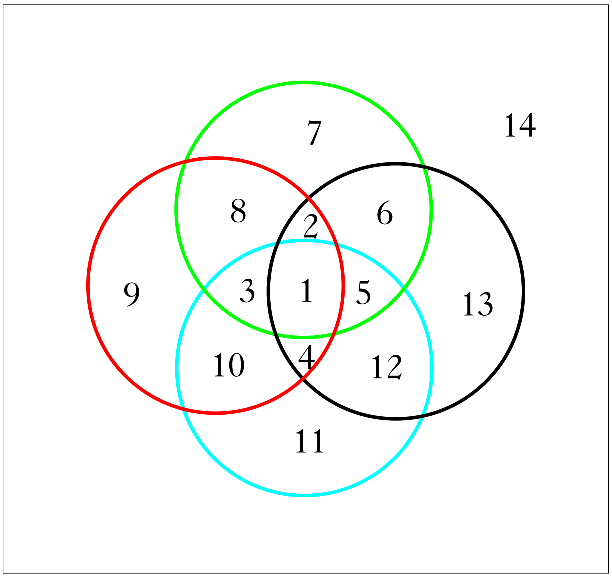
$$\operatorname{sign}(x) = \begin{cases} 1 & \text{for} \quad x \geq 0 \\ 0 & \text{for} \quad x = 0 \\ -1 & \text{for} \quad x < 0. \end{cases}$$

A transfer function of a (sign-)perceptron is then a function

$$f_{w,\theta} : \mathbb{R}^n \to \{-1, 0, 1\}, \quad x \mapsto f_{w,\theta}(x) = \operatorname{sign}(\langle w, x \rangle - \theta),$$

defined by the weight vector $w \in \mathbb{R}^n$ and threshold $\theta \in \mathbb{R}$. If $\theta = 0$, the perceptron is called **homogeneous**. In this case, we also write f_w instead of $f_{w,0}$. A **dichotomy** (X_-, X_+) of a set $X \subset \mathbb{R}^n$ is a decomposition $X = X_+ \cup X_-$ into disjoint subsets. Obviously, each perceptron $f_{w,\theta}$ induces a dichotomy on a subset $X \subset \mathbb{R}^n$ with $\{x \in X : \langle w, x \rangle = \theta\} = \emptyset$, using

$$X_+ := \{x \in X : \langle w, x \rangle > \theta\} \quad \text{and} \quad X_- := \{x \in X : \langle w, x \rangle < \theta\}.$$



(a)

1	0	5	AND	9	NAND	13	OR
2	$\bar{x}_1 \wedge x_2$	6	x_2	10	$\bar{x}_2$	14	1
3	NOR	7	$\bar{x}_1 \vee x_2$	11	$x_1 \vee \bar{x}_2$		
4	$x_1 \wedge \bar{x}_2$	8	$\bar{x}_1$	12	x_1		

(b)

Figure 5.2: *(a) Schematic representation after projection of the Boolean sphere onto the plane. The marked areas correspond to those in Fig. 5.1. (b) Switching functions which can be realized by a perceptron.*

A dichotomy (X_+, X_-) of a subset $X \subset \mathbb{R}^n$ is called **(strictly) affinely separable**, if there is a perceptron $f_{w,\theta} : \mathbb{R}^n \to \mathbb{R}$, with

$$X_+ = \{x \in X \ : \ f_{w,\theta}(x) = 1\} \quad \text{and} \quad X_- = \{x \in X \ : \ f_{w,\theta}(x) = -1\}$$

and hence,

$$X_0 = \{x \in X \ : \ f_{w,\theta}(x) = 0\} = \emptyset.$$

A dichotomy (X_+, X_-) of $X \subset \mathbb{R}^n$ is referred to as **linearly separable**, if there is an homogeneous perceptron $f_w : \mathbb{R}^n \to \mathbb{R}$, so that

$$X_+ = \{x \in X \ : \ f_w(x) = 1\} \quad \text{and} \quad X_- = \{x \in X \ : \ f_w(x) = -1\}.$$

The hyperplane

$$\{x \in \mathbb{R}^n \ : \ \langle w, x \rangle = \theta\}$$

is called a **separating hyperplane** with regard to (X_+, X_-).

Remark: For a subset $X \subset \mathbb{R}^n$, let

$$\hat{X} := \left\{ \begin{pmatrix} x \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1} \ : \ x \in X \right\}.$$

Then, a dichotomy (X_+, X_-) of a subset $X \subset \mathbb{R}^n$ is affinely separable if and only if the corresponding dichotomy $(\hat{X}_+, \hat{X}_-)$ of $\hat{X} \subset \mathbb{R}^{n+1}$ is linearly separable. This corresponds to the elimination of the threshold as discussed in Chapter 1.

Definition 5.1 Let X be a finite subset of $\mathbb{R}^n$. The **affine capacity** $c_a(X)$ of the perceptrons $f_{w,\theta} : \mathbb{R}^n \to \mathbb{R}, w \in \mathbb{R}^n, \theta \in \mathbb{R}$ with regard to X is the number of all affinely separable dichotomies (X_+, X_-) of X. Analogously, the **linear capacity** $c(X)$ of the homogeneous perceptrons $f_w : \mathbb{R}^n \to \mathbb{R}, w \in \mathbb{R}^n$, is the number of all linearly separable dichotomies of X.

Due to the above remark,

$$c_a(X) = c(\hat{X})$$

always applies. Hence, from now on, we may restrict to the calculation of the linear capacity, i.e., homogeneous perceptrons. The proof of Theorem 5.2 below requires the following lemma.

Lemma 5.1 *Let (X_+, X_-) be a dichotomy of a finite set $X \subset \mathbb{R}^n$ and let $y \in \mathbb{R}^n \setminus (X \cup \{0\})$. Then, $(X_+ \cup \{y\}, X_-)$ and $(X_+, X_- \cup \{y\})$ are linearly separable dichotomies of $X \cup \{y\}$ if and only if there is a hyperplane $H \subset \mathbb{R}^n$ containing y, which linearly separates (X_+, X_-).*

Proof: The set of all linearly separating weight vectors for (X_+, X_-) is

$$W(X_+, X_-) = \{w \in \mathbb{R}^n \ : \ \langle w, x\rangle > 0 \text{ for } x \in X_+, \langle w, x\rangle < 0 \text{ for } x \in X_-\}.$$

The dichotomy $(X_+ \cup \{y\}, X_-)$ is linearly separable if and only if there is a $w_+ \in W(X_+, X_-)$ with $\langle w_+, y\rangle > 0$. Likewise, $(X_+, X_- \cup \{y\})$ is linearly separable if there is a $w_- \in W(X_+, X_-)$ with $\langle w_-, y\rangle < 0$. Now let

$$w^* := -\langle w_-, y\rangle w_+ + \langle w_+, y\rangle w_-.$$

Then $w^* \in W(X_+, X_-)$ and $\langle w^*, y\rangle = 0$. Conversely, let w^* be an arbitrary element of $W(X_+, X_-)$ with $\langle w^*, y\rangle = 0$. Then

$$\begin{aligned} w_+ &:= w^* + \varepsilon y \in W(X_+ \cup \{y\}, X_-) \\ w_- &:= w^* - \varepsilon y \in W(X_+, X_- \cup \{y\}) \end{aligned}$$

holds for all sufficiently small $\varepsilon > 0$. $\square$

Now, an explicit combinatorial formula for the linear capacity is to be found. For this, we limit ourselves to finite subsets $X \subset \mathbb{R}^n$ whose elements are in general position.

Definition 5.2 A set X in $\mathbb{R}^n$ is said to be in **general position** if any subset $X' \subseteq X$, with no more than n elements, is linearly independent.

Theorem 5.2 (Winder 1966, Cover 1965) *There are exactly*

$$C(N,n) = 2\sum_{k=0}^{n-1} \binom{N-1}{k}$$

linearly separable dichotomies of N-element subsets $X \subset \mathbb{R}^n$ in general position.

Proof: The proof works by induction on N and n. Let $C(N,n)$ be the number of all linearly separable dichotomies of $X = \{x_1, \dots, x_N\}$. Let $x_{N+1} \in \mathbb{R}^n$ be a new point, such that $X \cup \{x_{N+1}\}$ is in general position, in particular, $x_{N+1} \neq 0$. The linearly separable dichotomies of X then split into two classes with respect to x_{N+1}. Let (X_+, X_-) be a separable dichotomy of X and let $W(X_+, X_-)$ be as defined above. If we have either $\langle w, x_{N+1}\rangle > 0$ for all $w \in W(X_+, X_-)$ or $\langle w, x_{N+1}\rangle < 0$ for all $w \in W(X_+, X_-)$, then either $(X_+ \cup \{x_{N+1}\}, X_-)$ or $(X_+, X_- \cup \{x_{N+1}\})$ is separable, respectively. On the other hand, if there exists a $w \in W(X_+, X_-)$ with $\langle w, x_{N+1}\rangle = 0$, then, according to Lemma 5.1, both $(X_+ \cup \{x_{N+1}\}, X_-)$ and $(X_+, X_- \cup \{x_{N+1}\})$ are separable. This second case requires the following consideration.

Let $L = \{w : \langle w, x_{N+1}\rangle = 0\}$ and let $\pi : \mathbb{R}^n \to L$ be the orthogonal projection onto L with $\ker(\pi) = \mathbb{R} \cdot x_{N+1}$. Then for $w \in L$, we have $w \in W(X_+, X_-)$ if and only if $w = \pi(w) \in W(\pi(X_+), \pi(X_-))$. Therefore it suffices to count the separable dichotomies of $\pi(X) \subset L$. There is an isomorphism $\phi : L \to \mathbb{R}^{n-1}$, and $\{\phi(\pi(x_1)), \dots, \phi(\pi(x_N))\} \subset \mathbb{R}^{n-1}$ is in general position. Therefore, the number of these separable dichotomies is $C(N, n-1)$.

Hence, summarizing the two cases, we obtain

$$C(N+1,n) = C(N,n) + C(N,n-1).$$

Using

$$C(1,n) = 2 \quad \text{for all } n \geq 1$$

and

$$C(N,1) = 2 \quad \text{for all } N \geq 1$$

we have a unique solution of the recursion, namely

$$C(N,n) = 2\sum_{k=0}^{n-1} \binom{N-1}{k}$$

for all $n, N \geq 1$. This proves the statement. □

Remark: A similar statement to that in Theorem 5.2 can also be shown for formal neurons whose activation function is a polynomial of higher degree.

Asymptotic Behavior of $C(N,n)$

Evidently, for the linear capacity of a perceptron

$$C(N,n) \leq 2^N,$$

is always valid. Moreover, $C(N,n) = 2^N$ for $N \leq n$. For $N \geq n$, we obtain the following estimate.

Lemma 5.3 *For $n, N \in \mathbb{N}$ with $N \geq n \geq 1$,*

$$C(N,n) = 2\sum_{k=0}^{n-1} \binom{N-1}{k} \leq 4\frac{(N-1)^{n-1}}{(n-1)!}$$

is valid.

In the proof, we use the inequality $2^m \leq 2\frac{m^m}{m!}$ for $m \geq 1$, which can easily be proven by induction, using the well-known inequality $2 \leq (1+\frac{1}{m})^m$ for $m \geq 1$.

Proof: The claim is equivalent to

$$\sum_{k=0}^{n} \binom{N}{k} \leq 2\frac{N^n}{n!}$$

for all $N \geq n \geq 0$. We prove this statement by induction on n and N. For $n = 0$, the statement is valid, since

$$\sum_{k=0}^{0} \binom{N}{k} = \binom{N}{0} = 1 \leq 2 = 2\frac{N^0}{0!}.$$

Suppose that the statement has been proven for $n = 0, \ldots, m-1$ and consider the case $n = m$. The induction on N starts with $N = n$. In that case, we have to show that

$$\sum_{k=0}^{m} \binom{m}{k} = 2^m \leq 2\frac{m^m}{m!}.$$

This is true according to the remark preceding the proof. Assume that the statement has been proven for all $N = n, \ldots, l-1$. Then, we need to show that it also holds for $N = l$. Thus, we consider

$$\sum_{k=0}^{m} \binom{l}{k} = \sum_{k=0}^{m} \binom{l-1}{k} + \sum_{k=0}^{m} \binom{l-1}{k-1} = \sum_{k=0}^{m} \binom{l-1}{k} + \sum_{k=0}^{m-1} \binom{l-1}{k}.$$

The inductive hypothesis implies that

$$\sum_{k=0}^{m} \binom{l}{k} \leq 2\frac{(l-1)^m}{m!} + 2\frac{(l-1)^{m-1}}{(m-1)!}.$$

Therefore it is sufficient to show that

$$2\frac{(l-1)^m}{m!} + 2\frac{(l-1)^{m-1}}{(m-1)!} \leq 2\frac{l^m}{m!}.$$

Multiplying both sides of this inequality by $m!/2$, one obtains the equivalent inequality

$$(l-1)^m + m(l-1)^{m-1} \leq l^m$$

which can also be written as

$$(l-1)^{m-1}(l-1+m) \leq l^m. \tag{5.2}$$

However, this is true, because the binomial formula yields

$$1 + \frac{m}{l-1} \leq \left(1 + \frac{1}{l-1}\right)^m$$

and thus

$$\frac{l-1+m}{l-1} \leq \left(\frac{l}{l-1}\right)^m$$

which is equivalent to (5.2) after multiplication by $(l-1)^m$. □

If X is an N-element set, there are 2^N dichotomies of X. Suppose that $X \subset \mathbb{R}^n$ is in general position. Then

$$P(N, n) := \frac{C(N, n)}{2^N}$$

is the probability of a dichotomy of X to be linearly separable. The largest N with the property

$$P(N, n) = 1$$

is also referred to as the **dimension of separability**, see the next chapter for the relation of this concept with the so-called VC-dimension.

Note that for fixed n, $C(N, n)$ is a polynomial of N of degree $n-1$. Since $\lim_{N\to\infty} \frac{N^{n-1}}{2^N} = 0$, the probability $P(N, n)$ converges to 0 with increasing N. In the case of switching functions, we have $N = 2^n$, and then $\lim_{n\to\infty} P(2^n, n+1) = 0$ since $\lim_{n\to\infty} \frac{(2^n)^n}{2^{2^n}} = 0$. Therefore, a perceptron is far from being sufficient for representing arbitrary switching functions (cf. Fig. 5.4).

The formula in Theorem 5.2 is only applicable to X in general position. The object of the next subsection is to obtain an estimation of $c(X)$, with $X \subset \mathbb{R}^n$ being arbitrary.

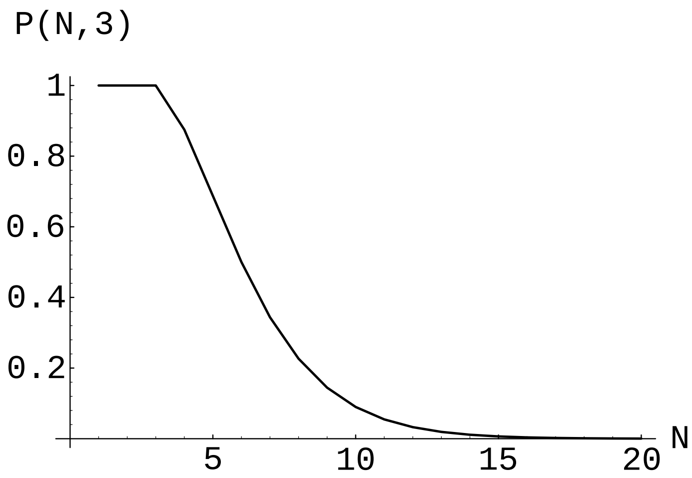


Figure 5.3: $P(N,3)$ *converges to* 0 *with increasing cardinality* N.

Number of variables n	Number of switching functions 2^N with $N = 2^n$	Affine capacity $C(N, n+1)$	$P(N, n+1)$
2	16	14	0.875
3	256	128	0.5
4	65536	3882	0.06
5	4294967296	412736	0.0001

Figure 5.4: *Realization of switching functions by perceptrons. By increasing the number of variables* n*, the percentage of switching functions which can be realized by a perceptron is rapidly reduced.*

Degenerate Position

We have determined the linear capacity $C(N,n)$ of an N-element subset X of $\mathbb{R}^n$, supposing that X is in general position. If we drop that assumption, the following observation is quite immediate.

Corollary 5.4 *For all* N*-element subsets* X *of* $\mathbb{R}^n$ *the following is valid:*

$$c(X) \leq C(N,n).$$

The aim of this subsection is to make more precise the influence of degeneracy of X on $c(X)$.

We start with the subsequent lemma, which gives a simple geometric description of the linear capacity of a finite set (not necessarily in general position).

Lemma 5.5 *Let $X = \{x_1, \dots, x_N\}$ be a finite subset of $\mathbb{R}^n \setminus \{0\}$ and let*

$$H_{x_i} := H_i := \{w \in \mathbb{R}^n \ : \ \langle w, x_i \rangle = 0\}, \quad i = 1, \dots, N,$$

be the corresponding hyperplanes. Then the linear capacity $c(X)$ of X is equal to the number of connected components of $\mathbb{R}^n \setminus \bigcup_{i=1}^N H_i$.

Proof: A dichotomy (X_+, X_-) of X is linearly separable if and only if there exists $w \in \mathbb{R}^n$ with

$$X_+ = \{x \in X \ : \ \langle w, x \rangle > 0\} \quad \text{and} \quad X_- = \{x \in X \ : \ \langle w, x \rangle < 0\}.$$

Let

$$W(X_+, X_-) := \{w \in \mathbb{R}^n \ : \ \langle w, x \rangle > 0 \text{ for } x \in X_+, \langle w, x \rangle < 0 \text{ for } x \in X_-\}.$$

This is a convex set and, in particular, a connected subset of $\mathbb{R}^n \setminus \bigcup_{i=1}^N H_i$. In addition, $W(X_+, X_-)$ is not empty if and only if (X_+, X_-) is linearly separable. Hence,

$$(X_+, X_-) \mapsto W(X_+, X_-)$$

yields a bijection of the set of all linearly separable dichotomies of X onto the set of all connected components of $\mathbb{R}^n \setminus \bigcup_{i=1}^N H_i$. □

This yields Schläfli's classical result as a direct corollary to Theorem 5.2.

Corollary 5.6 (Schläfli, 1852) *Let $H_1, \dots, H_N \subset \mathbb{R}^n$ be linear hyperplanes in $\mathbb{R}^n$ in general position. Then $\mathbb{R}^n \setminus \bigcup_{i=1}^N H_i$ splits into precisely $C(N, n)$ connected components.*

Remark: The calculation of the number of all connected components of the complement $\mathbb{R}^n \setminus \bigcup_{i=1}^N H_i$ of a system of linear hyperplanes in $\mathbb{R}^n$, is a classic subject of combinatorial geometry. Classic contributions can be found in the works of Steiner (1826), Schläfli (1852) and Roberts (1888). Background material to this can be found in Grünbaum (1967) and a good general overview of the modern development in this field can be found in Zaslavsky (1975) and Cartier (1980/81).

Winder (1966) and Zaslavsky (1975) proved the following formula for the number $c(\mathcal{H})$ of connected components of the complement $\mathbb{R}^n \setminus \bigcup_{H \in \mathcal{H}} H$ of an arbitrary system $\mathcal{H} = \{H_1, \dots, H_N\}$ of linear hyperplanes.

Definition 5.3 For any subset $\mathcal{S} \subseteq \mathcal{H}$ let $d(\mathcal{S}) := \dim \bigcap_{S \in \mathcal{S}} S$. For $\mathcal{S} = \emptyset$, we set $d(\mathcal{S}) = n$. The **characteristic polynomial of $\mathcal{H}$** is then defined by

$$P_{\mathcal{H}}(\lambda) := \sum_{\mathcal{S} \subseteq \mathcal{H}} (-1)^{|\mathcal{S}|} \lambda^{d(\mathcal{S}) - d(\mathcal{H})}.$$

Theorem 5.7 (Zaslavsky, 1975) *Let $\mathcal{H}$ be a finite set of linear hyperplanes in $\mathbb{R}^n$. Then, for the number $c(\mathcal{H})$ of all connected components of $\mathbb{R}^n \setminus \bigcup_{H\in\mathcal{H}} H$,*

$$\begin{aligned} c(\mathcal{H}) &= (-1)^{n-d(\mathcal{H})} P_{\mathcal{H}}(-1) \\ &= \sum_{\mathcal{S}\subseteq\mathcal{H}} (-1)^{|\mathcal{S}|+n+d(\mathcal{S})}. \end{aligned}$$

is valid.

Definition 5.4 A set $\mathcal{S}$ of linear hyperplanes in $\mathbb{R}^n$ is called **non-degenerate** if

$$d(\mathcal{S}) = n - |\mathcal{S}|.$$

Clearly, this implies that $|\mathcal{S}| \leq n$. In that case, $\mathcal{S}$ is non-degenerate if and only if the normal vectors of the hyperplanes $S \in \mathcal{S}$ are linearly independent. If

$$d(\mathcal{S}) \equiv n - |\mathcal{S}| \bmod 2,$$

then $\mathcal{S}$ is called **even-degenerate**. For

$$d(\mathcal{S}) \not\equiv n - |\mathcal{S}| \bmod 2,$$

the system $\mathcal{S}$ is referred to as **odd-degenerate.** In particular, non-degenerate systems of hyperplanes and the empty set are even-degenerate.

Obviously, for any system $\mathcal{S} \subseteq \mathcal{H}$ of hyperplanes, $\mathcal{S}$ is even-degenerate if and only if $n - |\mathcal{S}| + d(\mathcal{S})$ is even, or equivalently,

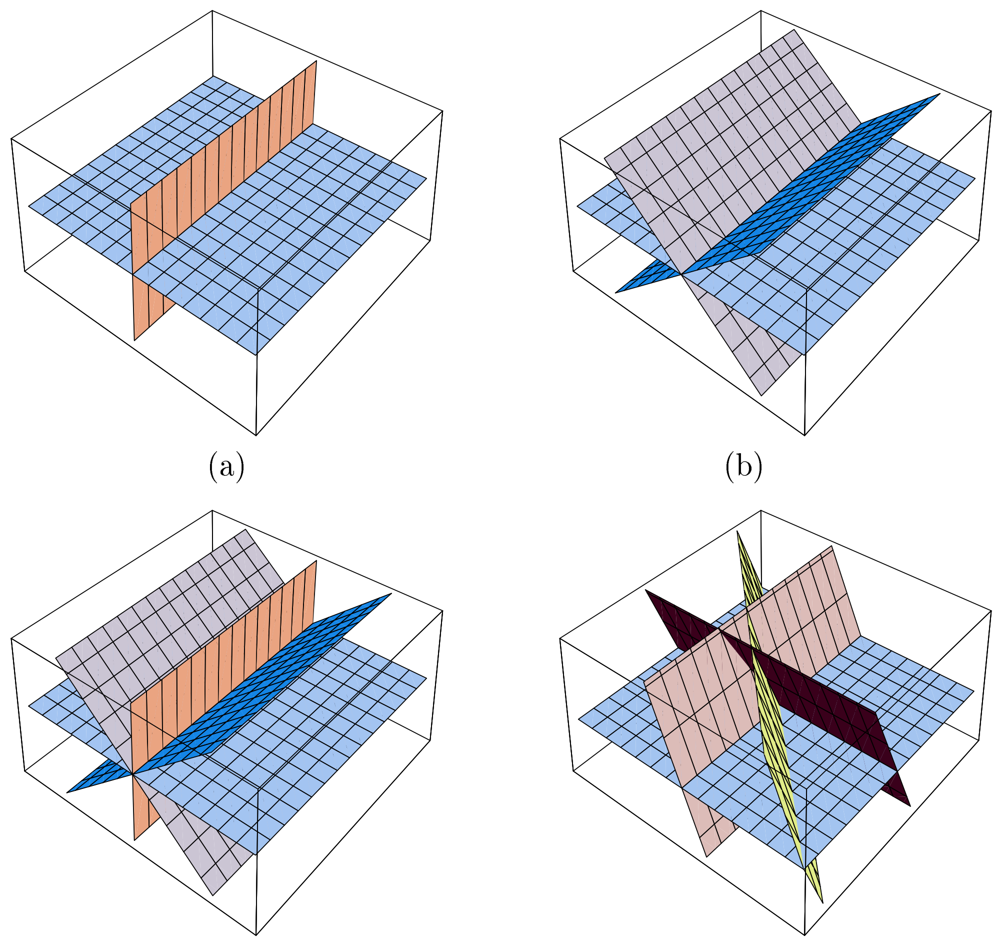
$$|\mathcal{S}| + n + d(\mathcal{S}) \equiv 0 \bmod 2.$$

Hence, Zaslavsky's theorem is equivalent to the following theorem by Winder.

Theorem 5.8 (Winder, 1966) *Using the same notation as in the above theorem, $c(\mathcal{H})$ equals the number of even-degenerate subsets $\mathcal{S} \subseteq \mathcal{H}$ minus the number of odd-degenerate subsets $\mathcal{S} \subseteq \mathcal{H}$.*

With Lemma 5.5 we obtain:

Corollary 5.9 *Let $X \subset \mathbb{R}^n \setminus \{0\}$ be a finite subset and $\mathcal{H}_X = \{H_x \subset \mathbb{R}^n | x \in X\}$. Then $c(X) = c(\mathcal{H}_X)$, which equals the number of even-degenerate subsets $\mathcal{S} \subseteq \mathcal{H}_X$ minus the number of odd-degenerate subsets $\mathcal{S} \subseteq \mathcal{H}_X$.*



(a) (b) (c) (d)

Figure 5.5: *Example of degenerate and non-degenerate systems of linear hyperplanes. The system in (a) is non-degenerate, in (b) odd-degenerate and in (c) even-degenerate. In (d),* $\mathcal{H}_X$ *is shown with* $X = \{0,1\}^2 \times \{1\}$*. With the exception of* $\mathcal{H}_X$ *itself, all systems* $\mathcal{S} \subseteq \mathcal{H}_X$ *are even-degenerate. Then, according to Winder's theorem,* $c(\mathcal{H}_X) = 15 - 1 = 14$*. This is exactly the number of regions which are defined by* $\mathcal{H}_X$*.*

Theorem 5.2 can be directly concluded from Zaslavsky's and Winder's theorems. For this, let $\mathcal{H} = \{H_1, \ldots, H_N\}$ be a set of linear hyperplanes in general position. Then for any $\mathcal{S} \subseteq \mathcal{H}$,

$$d(\mathcal{S}) = \begin{cases} n - |\mathcal{S}| & \text{if } n - |\mathcal{S}| \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently,

$$d(\mathcal{S}) = \max(0, n - |\mathcal{S}|).$$

Therefore we have

$$\begin{aligned} c(\mathcal{H}) &= (-1)^{n-d(\mathcal{H})} P_{\mathcal{H}}(-1) \\ &= \textstyle\sum_{k=0}^{N} \sum_{\substack{\mathcal{S} \subset \mathcal{H} \\ |\mathcal{S}|=k}} (-1)^{n-k+\max(0,n-k)} \\ &= \begin{cases} \sum_{k=0}^{n} \binom{N}{k} + \sum_{k=n+1}^{N} \binom{N}{k} (-1)^{n-k} & \text{for } n < N \\ \sum_{k=0}^{N} \binom{N}{k} & \text{for } n \geq N. \end{cases} \end{aligned}$$

Using the identity

$$\binom{N}{k} = \binom{N-1}{k-1} + \binom{N-1}{k},$$

$c(\mathcal{H}) = c(N, n)$ can be shown.

5.2 Capacity of 1-layer Perceptrons

The capacity of a single perceptron is quite small in comparison with the number of possible dichotomies of a set. This has already been indicated in Chapter 1 by the insolubility of the XOR problem by a perceptron. The XOR problem can however, be solved by a 1-layer perceptron. In fact, it was shown in Theorem 2.2 that arbitrary switching functions can be realized by 1-layer perceptrons. The number of hidden neurons needed for this is however, dependent on the switching function. In this section, we will examine the capacity of a 1-layer perceptron with an *a priori* given, fixed number of hidden neurons k. For this, let sign : $\mathbb{R} \to \{-1, 0, 1\}$ be the sign function again. Consider a 1-layer perceptron with n inputs, k hidden neurons and one output neuron, that is, a transfer function of the form

$$f : \mathbb{R}^n \to \{-1, 0, 1\}, \quad x \mapsto f(x) = \operatorname{sign}\left(\sum_{i=1}^{k} c_i \operatorname{sign}(\langle w_i, x \rangle - \theta_i) - \eta \right),$$

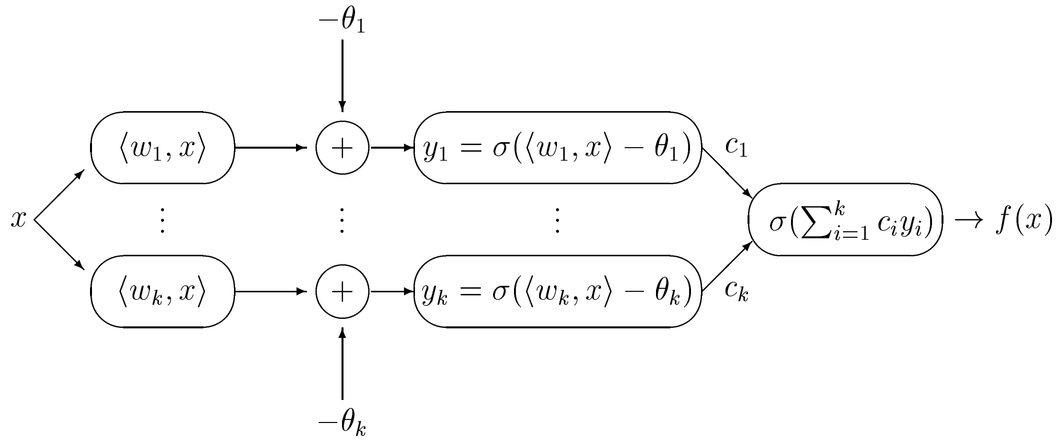


Figure 5.6: *1-layer perceptron.*

with real numbers $c_1, \dots, c_k \in \mathbb{R}$, weight vectors $w_1, \dots w_k \in \mathbb{R}^n$ and thresholds $\theta_1, \dots, \theta_k, \eta \in \mathbb{R}$. Now let $\mathcal{F}_k$ denote the set of all transfer functions of such 1-layer perceptrons. Let $X \subset \mathbb{R}^n$ be a finite subset. A dichotomy (X_+, X_-) of X is called **$\mathcal{F}_k$-separable** if there exists $f \in \mathcal{F}_k$ with

$$X_+ = \{x \in X \ : \ f(x) = 1\} \quad \text{and} \quad X_- = \{x \in X \ : \ f(x) = -1\}.$$

Definition 5.5 Let $X \subset \mathbb{R}^n$ be a finite subset. The **capacity** $c_{\mathcal{F}_k}(X)$ of X with respect to $\mathcal{F}_k$ is the number of all $\mathcal{F}_k$-separable dichotomies of X.

Obviously, $c_{\mathcal{F}_k}(X) \geq c_a(X)$ always applies. The following considerations are restricted to $n = 1$.

Lemma 5.10 *Let $f : \mathbb{R} \to \{\pm 1\}$ be a function. The $f \in \mathcal{F}_k$ if and only if f has at most k discontinuities.*

Proof: The only discontinuities of $f \in \mathcal{F}_k$ are the points $x_i = \frac{\theta_i}{w_i}$, for $w_i \neq 0$. To see this, let x_0 be a point with $w_i x_0 \neq \theta_i$ for all i. Then there exists $\varepsilon > 0$ such that $w_i x \neq \theta_i$ for all i and all x with $|x - x_0| < \varepsilon$. Therefore $\text{sign}(w_i x - \theta_i)$, $\sum c_i \text{sign}(w_i x - \theta_i)$ and thus $f(x)$ are all constant functions on $|x - x_0| < \varepsilon$, in particular, they are continuous there.

Conversely, let $a_1 < \cdots < a_k$ be arbitrary. Set $a_0 := -\infty$ and $a_{k+1} := \infty$. Let $(s_0, \dots, s_k) \in \{\pm 1\}^{k+1}$ be given. Define

$$g(x) := \frac{1}{2} \sum_{i=0}^{k} s_i(\text{sign}(x - a_i) - \text{sign}(x - a_{i+1})).$$

Then

$$g(x) = \begin{cases} s_0 & \text{if } x < a_1 \\ s_1 & \text{if } a_1 < x < a_2 \\ \vdots & \vdots \\ s_{k-1} & \text{if } a_{k-1} < x < a_k \\ s_k & \text{if } x > a_k \end{cases}$$

and hence

$$f(x) := \mathrm{sign}(g(x)) = g(x) = s_i \text{ for } a_i < x < a_{i+1}, i = 0, \ldots, k.$$

It remains to note that g has the desired form, that is, $g(x) = c_0 + \sum_{j=1}^{k} c_j \mathrm{sign}(x - a_j)$. □

In particular, this shows that a function $f : \mathbb{R} \to \{\pm 1\}$ is realizable by a 1-layer perceptron iff it has a finite number of discontinuities.

Theorem 5.11 *The capacity of an N-element set $X \subset \mathbb{R}$ with respect to $\mathcal{F}_k$ satisfies*

$$c_{\mathcal{F}_k}(X) = 2 \sum_{j=0}^{k} \binom{N-1}{j}.$$

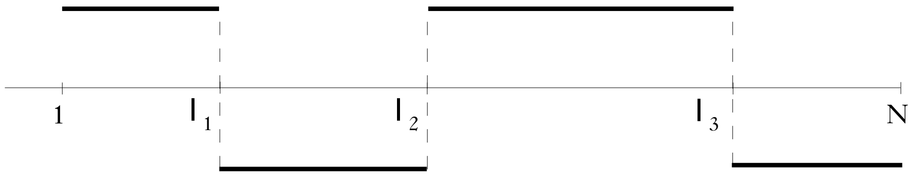
Proof: Let $X = \{1, \ldots, N\}$ without loss of generality. Then, any dichotomy (X_+, X_-) of X uniquely corresponds to a function

$$g : \{1, \ldots, N\} \to \{\pm 1\}$$

according to $x \in X_+ \Leftrightarrow g(x) = 1$. We extend this to a function on $f : \mathbb{R} \to \{\pm 1\}$ by putting

$$f(x) = \begin{cases} g(1) & \text{if } x < 3/2 \\ g(2) & \text{if } 3/2 < x < 5/2 \\ \vdots & \vdots \\ g(N) & \text{if } x > N - 1/2. \end{cases}$$

Hence, (X_+, X_-) is $\mathcal{F}_k$-separable if and only if $f \in \mathcal{F}_k$. According to Lemma 5.10, this is true iff f has at most k discontinuities. Assume that f has exactly $j \leq k$ discontinuities $1 < l_1 < l_2 < \cdots < l_j < N$, $l_i \in \{3/2, \ldots, N - 1/2\}$. Then f is uniquely determined by $(l_1, l_2, \ldots, l_j)$, up to multiplication by ± 1.



The number of all such sequences is

$$\binom{N-1}{j}.$$

Hence, the number of all $\mathcal{F}_k$-separable dichotomies is

$$2\sum_{j=0}^{k}\binom{N-1}{j}.$$

□

5.3 Capacity of Hopfield Networks

Here, we only consider the capacity of a Hopfield network as an associative memory with the Hebb rule. For this, let $\xi^{(1)},\dots,\xi^{(N)} \in \{\pm 1\}^n$ be given and define

$$U = (\xi^{(1)},\dots,\xi^{(N)}) \in \{\pm 1\}^{n\times N}.$$

Definition 5.6 We say that U is **Hebb-storable** if its columns are fixed points of the Hopfield network that results from the Hebb rule, i.e.,

$$\underline{\mathrm{sign}}(UU^T\xi^{(k)}) = \xi^{(k)}$$

for $k = 1,\dots,N$. The **capacity of the Hebb rule** $c_H(N,n)$ is defined to be the number of Hebb-storable matrices $U \in \{\pm 1\}^{n\times N}$. There are 2^{nN} such matrices, and therefore

$$P_H(N,n) = \frac{c_H(N,n)}{2^{nN}}$$

can be seen as the probability of a matrix $U \in \{\pm 1\}^{n\times N}$ to be Hebb-storable.

Consider

$$\begin{aligned}
(UU^T\xi^{(k)})_i &= \sum_{j=1}^{n}(UU^T)_{ij}\xi_j^{(k)} \\
&= \sum_{j=1}^{n}\sum_{l=1}^{N}\xi_i^{(l)}\xi_j^{(l)}\xi_j^{(k)} \\
&= n\xi_i^{(k)} + \sum_{j=1}^{n}\sum_{l\neq k}\xi_i^{(l)}\xi_j^{(l)}\xi_j^{(k)} \\
&= n\xi_i^{(k)}(1 + \frac{1}{n}\xi_i^{(k)}\sum_{j=1}^{n}\sum_{l\neq k}\xi_i^{(l)}\xi_j^{(l)}\xi_j^{(k)})
\end{aligned}$$

Therefore

$$\operatorname{sign}((UU^T\xi^{(k)})_i) = \xi_i^{(k)} \quad \Leftrightarrow \quad q_i^{(k)} := -\frac{1}{n}\xi_i^{(k)}\sum_{j=1}^{n}\sum_{l\neq k}\xi_i^{(l)}\xi_j^{(l)}\xi_j^{(k)} < 1.$$

Let P_{err} denote the error probability, that is, the probability of $q_i^{(k)} \geq 1$ when U, with $U_{ik} = \xi_i^{(k)}$, runs through all of $\{\pm 1\}^{n\times N}$. For a rough estimate of P_{err}, we note that the number $q_i^{(k)}$ equals $\frac{1}{n}$ times the sum of $n(N-1)$ numbers that can take the values ± 1, and we make the simplifying assumption that these two possibilities occur with equal probability. Then one can show that $q_i^{(k)}$ is binomially distributed with mean zero and variance $\frac{N-1}{n}$. As Nn becomes large, the binomial distribution tends to the Gaussian distribution, and

$$P_{\text{err}} \simeq \frac{1}{2}\left(1 - \operatorname{erf}\left(\sqrt{\frac{n}{2(N-1)}}\right)\right)$$

where

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt.$$

Further asymptotics lead to “rules of thumb” like

$$N \leq \frac{n}{4\ln(n)}$$

for the maximum number N of patterns to be stored, via the Hebb rule, in a Hopfield network with n neurons [11].

Chapter 6

Learnability and VC-Dimension

The Vapnik-Chervonenkis dimension (VC-dimension) is a term which stems from the theory of empirical stochastic processes. Vapnik and Chervonenkis introduced the term in order to prove generalizations of the central limit theorem. The significance of the VC-dimension for the theory of neural networks is seen in the fact that the VC-dimension is a measure of the learning capability of a network.

6.1 Learning Algorithms

The object of a learning algorithm is to determine an hypothesis h, which approximately corresponds to a given target property t, using only a finite number of examples. We start by stating this more precisely in mathematical terms.

Definition 6.1 1. Let X be an arbitrary set. A **concept** c on the set X is a function $c : X \to \{0, 1\}$.

2. Let t be a concept on X. A **training sample** s, of length m, for the concept t, is an element in $(X \times \{0,1\})^m$, with $s = ((x_1, t(x_1)), \ldots, (x_m, t(x_m)))$. Here, $x_i \in X$, $i = 1, \ldots, m$, are called **examples**, and if $t(x_i) = 1$, **positive** examples. The set of all training samples of length m for a concept t is denoted by $S(m, t)$.

3. Let two sets of concepts $\mathcal{C}$ and $\mathcal{H}$ on X be given. A **$(\mathcal{C}, \mathcal{H})$-learning algorithm** L is a mapping

$$L : \quad \mathcal{S} := \bigcup_{t \in \mathcal{C}} \bigcup_{m \geq 1} S(m, t) \to \mathcal{H},$$

which assigns a concept $h \in \mathcal{H}$ to any training sample $s \in \mathcal{S}$. Here, $\mathcal{S}$ is called **sample set**, $\mathcal{C}$ is the **concept set**, and $\mathcal{H}$ is called **hypothesis set** of L. The concept $h := L(s) \in \mathcal{H}$ is called the **hypothesis** generated by L from the sample s.

Particularly significant are learning algorithms whose hypothesis h coincides with the given concept t on the set of used examples.

Definition 6.2 A hypothesis $h \in \mathcal{H}$ is called **consistent** with a training sample $s = ((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m))$ if for all $i = 1, \ldots, m$,

$$h(x_i) = b_i.$$

A $(\mathcal{C}, \mathcal{H})$-learning algorithm is called **consistent** if $L(s)$ is consistent with s, for all $s \in \mathcal{S}$.

Example: Let $X = \mathbb{R}^n$ and let $\mathcal{C}$ be the set of all concepts $c : \mathbb{R}^n \to \{0, 1\}$, for which there are $w \in \mathbb{R}^n$, $\theta \in \mathbb{R}$, such that

$$c(x) = 1 \quad \Leftrightarrow \quad \langle w, x \rangle \geq \theta.$$

Further let $\mathcal{H}$ be the set of all perceptrons. Let $t \in \mathcal{C}$, $m \in \mathbb{N}$ be arbitrary, and let $s = ((x_1, t(x_1)), \ldots, (x_m, t(x_m))) \in S(m, t)$ an arbitrary training sample with $x_i \in \mathbb{R}^n$ for $i = 1, \ldots, m$. The perceptron learning algorithm finds an affine separation of $X_+ = \{x_i \ : \ t(x_i) = 1\}$ and $X_- = \{x_i \ : \ t(x_i) = 0\}$ and hence a perceptron $h \in \mathcal{H}$ with $h(x) = 1$ for $x \in X_+$ and $h(x) = 0$ for $x \in X_-$, i.e., the perceptron learning algorithm is consistent.

If the set X is finite, a consistent learning algorithm produces an hypothesis h which coincides exactly with the target concept t, provided that an appropriate training sample is chosen. For this, let $X = \{x_1, \ldots, x_n\}$ and let L be a consistent $(\mathcal{C}, \mathcal{H})$-learning algorithm. Let $t \in \mathcal{C}$ be arbitrary. Then, for $s := ((x_1, t(x_1)), \ldots, (x_n, t(x_n)))$, we have

$$L(s) = t.$$

In general however, it is impossible to include all elements of X in a training sample. Hence, we generally accept hypotheses h which approximately represent the target concept, i.e., the error of hypothesis h with respect to concept t

$$\mathrm{er}_\mu(h, t) := \mu \{x \in X \ : \ h(x) \neq t(x)\}$$

should not exceed a given value ε. Here, μ is a probability measure on X with σ-algebra Σ. If the probability measure μ on X is given, μ^m denotes the product measure on X^m induced by μ. Thereby, the components of a tuple $(x_1, \ldots, x_m)$ can be interpreted as independent μ-distributed variables. Then $\mu^m(Y)$, where $Y \subseteq X^m$, is the probability that an m-tuple $(x_1, \ldots, x_m)$ lies in the set Y, where each x_i was chosen from X by means of the distribution μ.

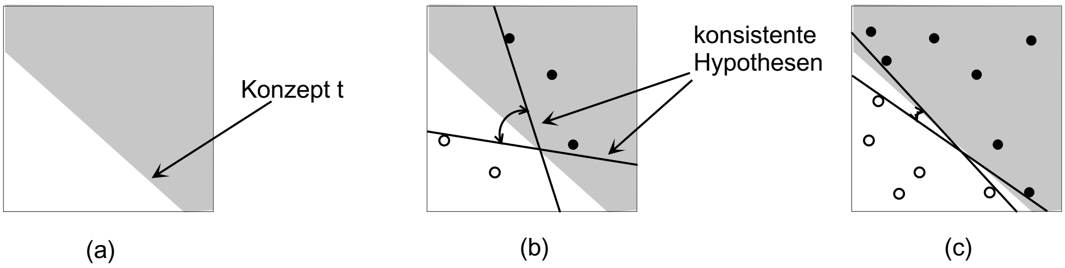


Figure 6.1: *The determination of an affine separation. (a) Let t be a given separation to be learned. (b) With only a few examples, a hypothesis (= affine separation), determined by the perceptron learning algorithm, can still differ greatly from t. (c) The hypotheses h produced come more into line with each other, when the number of training examples is increased. This is due to the PAC-property of the perceptron algorithm, which will be illustrated in the following.*

Notation: A concept c is uniquely determined by the set $c^{-1}(1) \subseteq X$. Therefore, the hypothesis set $\mathcal{H}$ can also be seen as a collection of subsets of X. With this interpretation, it makes sense to write, for $A \subseteq X$,

$$A \cap \mathcal{H} := \{A \cap H \ : \ H \in \mathcal{H}\}.$$

Then, we define

$$c_n(\mathcal{H}) := \max\{|A \cap \mathcal{H}| \ : \ A \subseteq X, |A| = n\}.$$

A basic statement about the quality of an algorithm is given by the following lemma.

Lemma 6.1 *[1, 19] Let $t \in \mathcal{C}$, μ and ε be arbitrary but fixed. Let s be a training sample for t and let $H[s] \subseteq \mathcal{H}$ be the set of all hypotheses consistent with s. Define*

$$B_\varepsilon[t] := \{h \in \mathcal{H} \ : \ \mathrm{er}_\mu(t, h) \geq \varepsilon\}.$$

Suppose that $m \geq \frac{8}{\varepsilon}$. Then we have

$$\mu^m\{x \in X^m \ : \ H[s[x,t]] \cap B_\varepsilon[t] \neq \emptyset\} \leq 2c_{2m}(\mathcal{H})2^{-\varepsilon m/2},$$

where $s[x,t] := ((x_1, t(x_1)), \ldots, (x_m, t(x_m)))$.

Definition 6.3 A $(\mathcal{C}, \mathcal{H})$-learning algorithm is called **probably approximately correct (PAC)**, if for any given $\varepsilon, \delta > 0$ there exist an $m_0(\varepsilon, \delta) \in \mathbb{N}$, such that for all $m \geq m_0(\varepsilon, \delta)$, all $t \in \mathcal{C}$, and all probability measures μ on X, the following applies:

$$\mu^m \{x \in X^m \ : \ \mathrm{er}_\mu(t, L(s[x,t])) \geq \varepsilon\} \leq \delta.$$

The smallest $m_0(\varepsilon, \delta)$ with the above property is called the **sample complexity** of L. We say that $(\mathcal{C}, \mathcal{H})$ is **PAC-learnable** if there exists $(\mathcal{C}, \mathcal{H})$-learning algorithm with the PAC-property, and **consistently PAC-learnable** if any consistent $(\mathcal{C}, \mathcal{H})$-learning algorithm is PAC.

Corollary 6.2 *Let L be a consistent $(\mathcal{C}, \mathcal{H})$-learning algorithm. Moreover, suppose that for any $\varepsilon, \delta > 0$, there is an $m \in \mathbb{N}$, with*

$$2c_{2m}(\mathcal{H})2^{-\varepsilon m/2} \leq \delta.$$

Then, L has the PAC-property.

Proof: Since L is consistent, the following applies to any training sample $s[x, t]$:

$$L(s[x, t]) \in H[s[x, t]]$$

and according to Lemma 6.1,

$$\begin{aligned} \mu^m\{x \in X^m \,:\, \mathrm{er}_\mu(t, L(s[x, t])) \geq \varepsilon\} &\leq \mu^m\{x \in X^m \,:\, H[s[x, t]] \cap B_\varepsilon[t] \neq \emptyset\} \\ &\leq 2c_{2m}(\mathcal{H})2^{-\varepsilon m/2} \quad \leq \quad \delta. \end{aligned}$$

$\square$

If a learning algorithm is PAC, we can guarantee that the error of a hypothesis h with respect to a target concept t is smaller than the value ε with probability greater than $1 - \delta$, as long as the length of the training sample used is large enough. Both the accuracy parameter ε and the confidence parameter δ can be chosen arbitrarily small. Thereby, both the error and the examples are calculated and chosen, by means of an arbitrary but fixed probability measure μ. In the following, necessary and sufficient conditions for the PAC-property of a learning algorithm will be investigated. The VC-dimension is of great significance here.

6.2 VC-Dimension

Let $\mathcal{C}$ be a set of functions from X to $\{0, 1\}$. Since any $c \in \mathcal{C}$ is uniquely determined by the set $c^{-1}(1) \subseteq X$, the function (or concept) set $\mathcal{C}$ can be identified with a collection of subsets of X.

Now let $\mathcal{C} \subseteq 2^X$ be an arbitrary system of subsets of X, where 2^X is the power set of X. Define, for $A \subseteq X$,

$$A \cap \mathcal{C} := \{A \cap C \,:\, C \in \mathcal{C}\} \subseteq 2^A.$$

Definition 6.4 We say that A is **shattered** by $\mathcal{C}$ if

$$A \cap \mathcal{C} = 2^A,$$

or equivalently,

$$|A \cap \mathcal{C}| = 2^{|A|}.$$

Notation: Let $A \subseteq X$, and let $\mathcal{C}$ be a set of subsets of X. Then we set

$$c_A(\mathcal{C}) := |A \cap \mathcal{C}|.$$

For $n \in \mathbb{N}$, we define

$$c_n(\mathcal{C}) := \max\{c_A(\mathcal{C}) \ : \ A \subseteq X, |A| = n\}.$$

Definition 6.5 The **VC-dimension VC-dim($\mathcal{C}$)** of $\mathcal{C} \neq \emptyset$ is defined as the largest number $n \in \mathbb{N}$, for which there is still a subset $A \subseteq X$ with $|A| = n$, such that A is shattered by $\mathcal{C}$. If there is no such integer, we set VC-dim$(\mathcal{C}) = \infty$. Thus

$$\text{VC-dim}(\mathcal{C}) = \sup\{n \ : \ c_n(\mathcal{C}) = 2^n\}.$$

We set VC-dim$(\emptyset) = -1$.

The following examples show that the VC-dimension of a finite set is always finite, whereas the VC-dimension of an infinite set can be both finite or infinite.

Examples:

1. Let $\mathcal{C}$ be finite. Then,

   $$\text{VC-dim}(\mathcal{C}) \leq \log_2 |\mathcal{C}|.$$

   To see this, let A be an arbitrary subset of X, which is shattered by $\mathcal{C}$. Then

   $$|\mathcal{C}| \geq |A \cap \mathcal{C}| = 2^{|A|}$$

   which implies that

   $$\log_2 |\mathcal{C}| \geq |A|.$$

2. Let X be an infinite set and let
$$\mathcal{C} := \{C \subset X \,:\, |C| < \infty\}$$
be the set of all finite subsets of X. Then VC-dim$(\mathcal{C}) = \infty$.

3. Let X be an infinite set and
$$\mathcal{C}_m := \{C \subset X \,:\, |C| \leq m\}.$$
Then, VC-dim$(\mathcal{C}_m) = m$. To see this, let $A \subseteq X$ given, with $|A| = m$. Then, any subset of A is in $\mathcal{C}_m$ and hence A is shattered by $\mathcal{C}_m$. For $A \subseteq X$ with $|A| > m$ there is, on the other hand, no $C \in \mathcal{C}_m$, with
$$A \cap C = A,$$
and hence VC-dim$(\mathcal{C}_m) = m$.

Lemma 6.3 *If $c_n(\mathcal{C}) = 2^n$, then $c_m(\mathcal{C}) = 2^m$ for all $m \leq n$.*

Proof: Let A be a set with n elements which is shattered by $\mathcal{C}$. Then any subset A' of A is also shattered by $\mathcal{C}$. (Otherwise, a set $A'' \subseteq A'$ would exist with $A'' \notin A' \cap \mathcal{C}$. Then, $A'' \notin A \cap \mathcal{C}$ would follow and this would be a contradiction.) Hence, there exist sets with $m = 0, \ldots, n$ elements which are shattered by $\mathcal{C}$, that is, $c_m(\mathcal{C}) = 2^m$. □

Remark: From the above lemma, it follows that
$$\begin{aligned} c_n(\mathcal{C}) = 2^n \quad &\Leftrightarrow \quad n \leq \text{VC-dim}(\mathcal{C}) \\ c_n(\mathcal{C}) < 2^n \quad &\Leftrightarrow \quad n > \text{VC-dim}(\mathcal{C}). \end{aligned}$$

Example: Let $X = \mathbb{R}^k$ and let
$$\mathcal{C} := \{[a_1, b_1] \times \cdots \times [a_k, b_k] \,:\, a_i, b_i \in \mathbb{R}, a_i < b_i, i = 1, \ldots, k\}$$
be the system of rectangles in $\mathbb{R}^k$. The VC-dimension of $\mathcal{C}$ is to be determined. Firstly, consider the k-dimensional unit-cube in $\mathcal{C}$ and choose $2k$ points $A = \{P_1, \ldots, P_{2k}\}$ such that they each lie in the center of a face of the cube. Then A is shattered by $\mathcal{C}$. Now let an arbitrary set of $2k+1$ vectors $A = \{P_1, \ldots, P_{2k+1}\} \subset \mathbb{R}^k$ be given and let $C' \in \mathcal{C}$ be the smallest rectangle which contains all the points of A. Then there is a point $P \in A$ which either lies in the interior of C' or on the same face as another point of A. This signifies however, that any rectangle $C \in \mathcal{C}$ which contains the points $A \setminus \{P\}$ also contains P. Hence
$$A \setminus \{P\} \notin A \cap \mathcal{C},$$
that is, A is not shattered by $\mathcal{C}$. Altogether, we find that VC-dim$(\mathcal{C}) = 2k$.

Lemma 6.4 *Let $\emptyset \neq A \subseteq X$, and let $x \in A$ be an arbitrary, but fixed element. Set $B = A \setminus \{x\}$. Then*

$$|A \cap \mathcal{C}| - |B \cap \mathcal{C}| = |\mathcal{D}|$$

where $\mathcal{D}$ is defined by

$$\mathcal{D} = \{D \subseteq B \ : \ D \in A \cap \mathcal{C} \text{ and } D \cup \{x\} \in A \cap \mathcal{C}\}.$$

Moreover, if $E \subseteq B$ is shattered by $\mathcal{D}$, then $E \cup \{x\}$ is shattered by $\mathcal{C}$. Therefore

$$\text{VC-dim}(\mathcal{D}) \leq \text{VC-dim}(\mathcal{C})$$

(with strict inequality if VC-dim$(\mathcal{C}) < \infty$).

Proof: Consider the map

$$\pi : A \cap \mathcal{C} \to B \cap \mathcal{C}, \quad Y \mapsto Y \setminus \{x\},$$

which is well-defined and surjective. Two sets $Y_1 \neq Y_2$ satisfy $\pi(Y_1) = \pi(Y_2)$ iff $Y_1 = Y_2 \cup \{x\}$ (or the other way round). The number of such $\{Y_1, Y_2\}$ equals the number of pairs $(D, D \cup \{x\})$, where $D \subseteq B$ is such that both D and $D \cup \{x\}$ are in $A \cap \mathcal{C}$. This proves the first statement of the lemma.

Now assume that $E \subseteq B$ is shattered by $\mathcal{D}$. Set $F := E \cup \{x\}$. We need to show that F is shattered by $\mathcal{C}$. For this, let E' be any subset of E. We show that both E' and $E' \cup \{x\}$ are in $F \cap \mathcal{C}$. By assumption, $E' = E \cap D$ for some $D \in \mathcal{D}$. Any $D \in \mathcal{D}$ has the form $D = A \cap C$ for some $C \in \mathcal{C}$. Thus

$$E' = E \cap D = E \cap A \cap C = E \cap C = F \cap C \in F \cap \mathcal{C},$$

where the last equality follows from $D \subseteq B$, which yields $x \notin D$ and $x \notin C$. Similarly,

$$E' \cup \{x\} = (E \cap D) \cup \{x\} = F \cap (D \cup \{x\}).$$

By the definition of $\mathcal{D}$, there is a $C \in \mathcal{C}$ with $D \cup \{x\} = A \cap C$ and therefore

$$E' \cup \{x\} = F \cap A \cap C = F \cap C \in F \cap \mathcal{C}.$$

Thus $2^F = F \cap \mathcal{C}$ as desired. □

Lemma 6.5 *Let $k, n \in \mathbb{N}$ be such that $c_n(\mathcal{C}) > \sum_{j=0}^{k-1} \binom{n}{j}$, then*

$$c_k(\mathcal{C}) = 2^k.$$

Proof: The assumption is proven by induction on n and k. Firstly, let $k = 1$. From

$$\sum_{j=0}^{0} \binom{n}{j} = 1 < c_n(\mathcal{C}),$$

it follows that $\mathcal{C}$ contains two sets C_1 and C_2, at least. Without loss of generality, $C_1 \neq \emptyset$. Let $x \in C_1$ be such that $x \notin C_2$ and set $A = \{x\}$. Then

$$A \cap \mathcal{C} = \{\{x\}, \emptyset\}$$

and thus $c_A(\mathcal{C}) = c_1(\mathcal{C}) = 2$, as desired.

Now, assume that the statement is valid for all $k \leq K$. Set $k := K + 1$. The induction on n follows. For $n < k$ the assumption applies due to

$$2^n = \sum_{j=0}^{n} \binom{n}{j} = \sum_{j=0}^{k-1} \binom{n}{j} \geq c_n(\mathcal{C}).$$

Hence, let the assumption be fulfilled for all $n \leq N$ and choose $n := N + 1$. Further let $H_n \subseteq X$ be such that $|H_n| = n = N + 1$, and

$$|H_n \cap \mathcal{C}| = c_n(\mathcal{C}) > \sum_{j=0}^{K} \binom{n}{j}. \tag{6.1}$$

For an arbitrary, but fixed element $x \in H_n$, set $H_N = H_n \setminus \{x\}$, then $|H_N| = N$.

Case 1:

$$|H_N \cap \mathcal{C}| > \sum_{j=0}^{K} \binom{N}{j}.$$

Since $|H_N \cap \mathcal{C}| \leq c_N(\mathcal{C})$, the induction hypothesis implies that $c_k(\mathcal{S}) = 2^k$ with $k = K + 1$ and the process is finished.

Case 2:

$$|H_N \cap \mathcal{C}| \leq \sum_{j=0}^{K} \binom{N}{j}.$$

Now choose $\mathcal{C}_n := H_n \cap \mathcal{C}$ and let $\mathcal{D}$ be the set of all subsets D of H_N, for which

$$D \in \mathcal{C}_n \quad \text{and} \quad D \cup \{x\} \in \mathcal{C}_n.$$

According to Lemma 6.4,

$$|H_n \cap \mathcal{C}| = |H_N \cap \mathcal{C}| + |\mathcal{D}|. \tag{6.2}$$

Case 2.1:

$$|\mathcal{D}| = |H_N \cap \mathcal{D}| > \sum_{j=0}^{K-1} \binom{N}{j}.$$

This means that

$$c_N(\mathcal{D}) > \sum_{j=0}^{K-1} \binom{N}{j}.$$

According to the induction hypothesis, this implies $c_K(\mathcal{D}) = 2^K$ and thus there exists $G \subseteq H_N$ exists with $|G| = K$ and $|G \cap \mathcal{D}| = 2^K$. Lemma 6.4 shows that for $J := G \cup \{x\}$, we have $|J| = K+1$ and $|J \cap \mathcal{C}| = 2^{K+1}$, which yields $c_{K+1}(\mathcal{C}) = 2^{K+1}$ as desired.

Case 2.2:

$$|\mathcal{D}| = |H_N \cap \mathcal{D}| \leq \sum_{j=0}^{K-1} \binom{N}{j}.$$

Using (6.2) and the binomial identity

$$\binom{N}{j} = \binom{N-1}{j} + \binom{N-1}{j-1},$$

we obtain

$$\begin{aligned} |H_n \cap \mathcal{C}| &\leq \sum_{j=0}^{K} \binom{N}{j} + \sum_{j=1}^{K} \binom{N}{j-1} \\ &= \sum_{j=0}^{K} \binom{N+1}{j} = \sum_{j=0}^{K} \binom{n}{j}. \end{aligned}$$

However, this is a contradiction to (6.1), and altogether the assumption follows.

□

Lemma 6.6 (Sauer) *If $d :=$ VC-dim$(\mathcal{C}) < \infty$, then for all $n \in \mathbb{N}$ we have*

$$c_n(\mathcal{C}) \leq \sum_{j=0}^{d} \binom{n}{j}.$$

Proof: Let $k := \text{VC-dim}(\mathcal{C}) + 1$. Then $c_k(\mathcal{C}) < 2^k$ and hence

$$c_n(\mathcal{C}) \leq \sum_{j=0}^{k-1} \binom{n}{j}$$

as stated in Lemma 6.5. □

Lemma 6.7 *Let $n \geq d = VC\text{-}dim(\mathcal{C}) \geq 1$. Then:*

$$c_n(\mathcal{C}) < \left(\frac{en}{d}\right)^d,$$

where e is the base of the natural logarithm.

Proof: We have

$$c_n(\mathcal{C}) \leq \sum_{j=0}^{d} \binom{n}{j}$$

and, from Lemma 5.3,

$$\sum_{j=0}^{d} \binom{n}{j} \leq \frac{2n^d}{d!}.$$

It is sufficient to show

$$\frac{2n^d}{d!} < \left(\frac{en}{d}\right)^d.$$

This results by induction on d.

For $d = 1$, the inequality is correct. Now let the statement be fulfilled for all $d \leq d'$. Then, according to the induction hypothesis,

$$(d'+1)! = (d'+1)d'! > (d'+1)2\left(\frac{d'}{e}\right)^{d'}.$$

Hence, it is sufficient to show

$$(d'+1) \cdot 2\left(\frac{d'}{e}\right)^{d'} > 2\left(\frac{d'+1}{e}\right)^{d'+1}.$$

This inequality is however, equivalent to the validity of

$$\left(1 + \frac{1}{d'}\right)^{d'} < e,$$

which, as is well-known, applies to all $d' \in \mathbb{N}$. □

Finally, we make the link between PAC-learning and the VC-dimension.

Theorem 6.8 *Let L be a consistent $(\mathcal{C}, \mathcal{H})$-learning algorithm and let $1 \leq d := \mathcal{V}(\mathcal{H}) < \infty$. Then, L has the PAC-property.*

Proof: According to Corollary 6.2, it is sufficient to show for arbitrary $\varepsilon > 0$:

$$\lim_{m\to\infty} 2c_{2m}(\mathcal{H})2^{-\varepsilon m/2} = 0.$$

According to Lemma 6.7, for sufficiently large m,

$$c_{2m}(\mathcal{H}) < \left(\frac{2em}{d}\right)^d .$$

Now let $a := (2e/d)^d$ and $b(\varepsilon) := 2^{-\varepsilon/2}$. Then

$$2c_{2m}(\mathcal{H})2^{-\varepsilon m/2} < 2a \cdot m^d \cdot b(\varepsilon)^m =: a_m(\varepsilon)$$

and due to $1 \leq d < \infty$ and $|b(\varepsilon)| < 1$,

$$\lim_{m\to\infty} a_m(\varepsilon) = 0$$

applies for all $\varepsilon > 0$. This gives the assumption. □

Theorem 6.8 shows that $(\mathcal{C}, \mathcal{H})$ is consistently PAC-learnable if VC-dim$(\mathcal{H})$ is finite. On the other hand, the following has been shown [1, 19]: If a – not necessarily consistent – $(\mathcal{C}, \mathcal{H})$-learning algorithm has the PAC-property, then the VC-dimension of both $\mathcal{H}$ and $\mathcal{C}$ is finite.

Chapter 7

Unsupervised Learning

Most of the learning algorithms discussed so far belong to the category of supervised learning. This chapter gives a short overview over an alternative approach to the concept of learning, known as learning without a teacher.

There are two general learning paradigms:

- Supervised Learning. Here the learning is done on the basis of direct comparison of the network's output with known correct answers. The quality of the output can therefore be directly assessed, and the network receives feedback about its error, such that "good" outputs are reinforced and "bad" outputs are punished. On the basis of this information, the network parameters (weights, thresholds) are updated. This is also known as *learning with a teacher.* Most of the learning algorithms discussed so far belong into this class: the perceptron learning algorithm, the back-propagation algorithm, and the projection rule learning algorithm.

- Unsupervised Learning. The goal of unsupervised learning is less obvious. Suppose that there is no feedback from the environment *(no teacher)* which tells the network what the correct outputs should be, or how good the actual network outputs are. This reflects the problem that, in many practical applications, there is an abundance of raw input data, but, e.g., no a priori known correct classification. The idea is that the network itself should discover patterns, features, regularities, categories and so on in the input data, and that it should code this extracted information in the output. In the following, we will give a short overview over different approaches within this class.

7.1 Hebbian Learning

Consider a σ-perceptron. Its transfer function has the form

$$y(t) = \sigma(x(t)^T w(t))$$

where $x(t), w(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}$ denote the input, weight, and output at time t, respectively. For simplicity, we assume that the threshold has already been eliminated. The **Hebb rule** suggests to change the weights according to

$$w(t+1) - w(t) = \varphi x(t) y(t) = \varphi x(t) \sigma(x(t)^T w(t)).$$

Here, $\varphi > 0$ is a proportionality constant, called the **learning rate**. A special case is obtained when σ is the identity map on $\mathbb{R}$. Then

$$y(t) = x(t)^T w(t)$$

and

$$w(t+1) - w(t) = \varphi x(t) y(t) = \varphi x(t) x(t)^T w(t). \tag{7.1}$$

The idea behind this rule is: Suppose that we have a sequence of input vectors $x = (x(0), x(1), x(2), \dots)$. The network should determine the similarity between the $x(i)$. After a training phase of, say, T time steps, the network should respond to a new input $x(T)$ by producing a number which measures the familiarity of $x(T)$ with $x(0), \dots, x(T-1)$. The more similar $x(T)$ is to the prototypes $x(0), \dots, x(T-1)$, the larger the output $y(T) = x(T)^T w(T)$ should become. Now the Hebb rule guarantees that, for $x(t) = \xi$,

$$w(t+1)^T \xi = w(t)^T \xi + \varphi w(t)^T \xi \xi^T \xi = (1 + \varphi \|\xi\|^2) w(t)^T \xi.$$

This means that if the pattern ξ was presented at time t, and if it was evaluated with the similarity measure $y(t) = w(t)^T \xi$, then, if ξ is presented again at time $t+1$, it receives

$$y(t+1) = w(t+1)^T \xi = (1 + \varphi \|\xi\|^2) y(t) \geq y(t)$$

as its output, showing that the similarity measure has been increased. Thereby, patterns which appear often in the sequence of prototypes, will come to produce a large output.

In a stochastic setup, we may also consider x as a random variable with a certain probability density, and then we set

$$C := E(xx^T)$$

where E denotes the expectation, and C is called **correlation matrix** of x. In particular, if x can only take finitely many values $\xi^{(1)}, \ldots, \xi^{(N)}$, with equal probability, then

$$C = \frac{1}{N}\sum_{j=1}^{N} \xi^{(j)}\xi^{(j)T}$$

which displays the connection with the Hebb rule as discussed in Chapter 4. The equation for $\hat{w} := E(w)$ becomes

$$\hat{w}(t+1) = \hat{w}(t) + \varphi C\hat{w}(t) = (I + \varphi C)\hat{w}(t) \tag{7.2}$$

which implies

$$\hat{w}(t) = (I + \varphi C)^t \hat{w}(0).$$

Note that C is symmetric and positive semi-definite, hence its eigenvalues are real and non-negative. The eigenvalues of $I + \varphi C$ are precisely the numbers $1 + \varphi\lambda$, where λ is an eigenvalue of C. Therefore $I + \varphi C$ is not discrete-time stable (unless $C = 0$, but this case is not interesting). Similarly, (7.1) has the form

$$w(t+1) = A(t)w(t) \quad \text{with } A(t) = I + \varphi x(t)x(t)^T.$$

The eigenvalues $\lambda(t)$ of $A(t)$ are real and satisfy $\lambda(t) \geq 1$. Therefore, we may expect some problems with the stability of (7.1). This is also indicated by (putting $x(t) = \xi$ again)

$$\begin{aligned} \|w(t+1)\|^2 &= \|w(t) + \varphi\xi\xi^T w(t)\|^2 \\ &= \|w(t)\|^2 + 2\varphi(\xi^T w(t))^2 + \varphi^2\|\xi\|^2(\xi^T w(t))^2 \end{aligned}$$

that is,

$$\|w(t+1)\|^2 - \|w(t)\|^2 = \varphi(\xi^T w(t))^2(2 + \varphi\|\xi\|^2) \geq 0.$$

The problem with Hebb's rule is that the weights tend to grow unboundedly as $t \to \infty$. Therefore, the following modification has been proposed.

Oja's Rule

The simplest way to keep the weight vectors from growing unboundedly is to normalize the weight vectors after each iteration step. Thus we set, starting from (7.1) again,

$$v(t) := \frac{w(t)}{\|w(t)\|}.$$

Clearly, this guarantees that $\|v(t)\| = 1$ for all t. Let us apply this to the averaged equation (7.2). Then with

$$\hat{v}(t) := \frac{\hat{w}(t)}{\|\hat{w}(t)\|}$$

one obtains

$$\hat{v}(t+1) = \frac{(I + \varphi C)\hat{v}(t)}{\|(I + \varphi C)\hat{v}(t)\|}.$$

In numerical linear algebra, this procedure is known as **vector iteration**. Recall that the eigenvalues of $I + \varphi C$ are precisely the numbers $1 + \varphi\lambda$, where λ is an eigenvalue of $C \in \mathbb{R}^{n \times n}$. Since C is symmetric and positive semi-definite, such λ is real and non-negative. Let $\lambda_n \leq \ldots \leq \lambda_2 \leq \lambda_1$ be the eigenvalues of C. We say that C has a **dominant eigenvalue**, or that λ_1 is a dominant eigenvalue of C if λ_2 is strictly less than λ_1. This means that the largest eigenvalue of C is simple, i.e., it has multiplicity one. If C has a dominant eigenvalue, then so does $I + \varphi C$, and then one can show that the vector iteration converges, for generic initial vectors $\hat{v}(0)$, to an eigenvector of $I + \varphi C$ (or of C, equivalently) that belongs to the dominant eigenvalue. The term “generic” refers to the fact that there is a hyperplane H in $\mathbb{R}^n$ such that convergence cannot be guaranteed for $\hat{v}(0) \in H$. However, since $\dim(H) = n - 1$, this means that the algorithm converges for almost all initial conditions.

Oja’s rule is also based on this normalization. However, the idea comes from considering the continuous version of (7.1), that is,

$$\dot{w}(t) = \varphi x(t)x(t)^T w(t).$$

We set

$$v(t) = \frac{w(t)}{\|w(t)\|}$$

again. Then

$$\dot{v}(t) = \varphi(I - v(t)v(t)^T)x(t)x(t)^T v(t).$$

We translate that back to the discrete situation and obtain **Oja’s rule**

$$v(t+1) - v(t) = \varphi(I - v(t)v(t)^T)x(t)x(t)^T v(t).$$

Using the output $y(t) = x(t)^T v(t)$, this can also be written as

$$v(t+1) - v(t) = \varphi(x(t)y(t) - v(t)y(t)^2).$$

The averaged version is again

$$\hat{v}(t+1) - \hat{v}(t) = \varphi(I - \hat{v}(t)\hat{v}(t)^T)C\hat{v}(t). \tag{7.3}$$

Note that any normalized eigenvector of C is an equilibrium of this difference equation: Set $f(v) = (I - vv^T)Cv$. We need to show: If $C\bar{v} = \lambda\bar{v}$ and $\bar{v}^T\bar{v} = 1$, then $f(\bar{v}) = 0$. However, it is easy to check that indeed

$$(I - \bar{v}\bar{v}^T)C\bar{v} = 0.$$

Suppose that C has a dominant eigenvalue, say λ_1. We show that a normalized eigenvector $\bar{v}$ is an *asymptotically stable* equilibrium iff $\bar{v}$ belongs to λ_1. To see this, we linearize (7.3) near $v = \bar{v}$ using

$$\frac{\partial f}{\partial v} = C - (v^T C v)I - 2vv^T C.$$

For $u(t) := \hat{v}(t) - \bar{v}$, this yields

$$u(t+1) - u(t) = \varphi(C - \lambda I - 2\lambda\bar{v}\bar{v}^T)u(t).$$

Thus, we need to investigate the matrix

$$A := I + \varphi(C - \lambda I - 2\lambda\bar{v}\bar{v}^T)$$

with respect to discrete-time stability. Since C is symmetric, there exists an orthonormal basis of $\mathbb{R}^n$ which consists of eigenvectors of C, say

$$CU = UD$$

where $U = (\bar{v}_1, \ldots, \bar{v}_n) \in \mathbb{R}^{n\times n}$ is orthogonal and $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n$. Suppose that $\bar{v} = \bar{v}_i$ and $\lambda = \lambda_i$ for some fixed, but arbitrary i. Since

$$(C - \lambda I - 2\lambda\bar{v}\bar{v}^T)\bar{v}_j = \begin{cases} -2\lambda\bar{v}_j & \text{if } i = j \\ (\lambda_j - \lambda)\bar{v}_j & \text{otherwise} \end{cases}$$

we conclude that the spectrum of A is given by

$$\{1 - 2\varphi\lambda\} \cup \{1 + \varphi(\lambda_j - \lambda) \ : \ j = 1, \ldots, n, j \neq i\}.$$

Therefore, for asymptotic stability, we need $\lambda_j < \lambda$ for all $j \neq i$, that is, $\lambda = \lambda_1$. On the other hand, if $\lambda = \lambda_1$, we can guarantee asymptotic stability if we make φ small enough.

In this section, we have restricted to the case of a dominant eigenvalue. Generalizations to the case of a degenerate largest eigenvalue have been studied, and more generally, the problem of finding an orthonormal basis of the eigenspace belonging to the largest k eigenvalues $\lambda_1 \geq \ldots \geq \lambda_k > \lambda_{k+1} \geq \ldots \geq \lambda_n$. This is known as **principal component analysis** in statistics.

7.2 Competitive Learning

Consider a feed-forward network with n inputs and p output neurons (and no hidden neurons). Its transfer function has the form

$$y(t) = \sigma(W(t)x(t))$$

where $x(t) \in \mathbb{R}^n$ is the input at time t, $y(t) \in \{0,1\}^p$ is the output at time t, and $W(t) \in \mathbb{R}^{p\times n}$ is the weight matrix at time t. Again, we suppose that there is no threshold. Note that here, $\sigma : \mathbb{R}^p \to \{0,1\}^p$ is not restricted to the case where $\sigma = \underline{\tau}$ for some $\tau : \mathbb{R} \to \{0,1\}$.

We wish to solve a classification problem with p classes $\{1, \ldots, p\}$. The network's response to an input $x(t)$ should be the number of the class to which $x(t)$ belongs. For this, we have to guarantee that at each time t, one and only one of the output neurons fires, that is, for some $i^* \in \{1, \ldots, p\}$, we have

$$y_{i^*}(t) = 1 \quad \text{and} \quad y_i(t) = 0 \text{ for } i \neq i^*.$$

Then we say that neuron i^* is the **winner neuron** at time t. Equivalently, the input $x(t)$ has been classified into class i^*.

Note that, in contrast to the perceptron learning problem, the classes are not predefined! The network itself should find clusters in the input data, and it should generate a classification directly from the data (therefore this is truly an example of unsupervised learning).

Suppose that the choice of the winner at time t depends only on the current activation vector $z(t) = W(t)x(t)$, according to some function

$$i^* : \mathbb{R}^p \to \{1, \ldots, p\}, \quad z \mapsto i^*(z).$$

Since the output of the winner neuron is one, and zero for all other neurons, we may write

$$y(t) = \sigma(z(t)) = e_{i^*(z(t))}$$

where e_i, $i = 1, \ldots, p$ denotes the i-th natural basis vector of $\{0,1\}^p$. Thus we define σ accordingly, that is, $\sigma(z) = e_{i^*(z)}$.

The winner neuron is usually chosen to be the one with the largest activation, that is, i^* is chosen such that

$$z_{i^*}(t) \geq z_i(t) \quad \text{for all } i = 1, \ldots, p. \tag{7.4}$$

Of course, this may not suffice do determine i^* uniquely, and then some random choice has to be made. If we assume additionally that the rows of $W = (w_1, \ldots, w_p)^T$, $w_i \in \mathbb{R}^n$ are normalized, then (7.4) is equivalent to

$$\|w_{i^*}(t) - x(t)\| \leq \|w_i(t) - x(t)\|.$$

In other words, the winner is the neuron whose weight vector is closest to the input. This yields an idea for a learning rule to solve this problem: The weight vector should be pushed into the direction of the input. We set

$$\begin{aligned} w_{i^*}(t+1) - w_{i^*}(t) &= \varphi(x(t) - w_{i^*}(t)) \\ w_i(t+1) - w_i(t) &= 0 \quad \text{for } i \neq i^*. \end{aligned} \tag{7.5}$$

Then, if $x(t) = \xi$, we have

$$\begin{aligned} \|w_{i^*}(t+1) - \xi\| &= \|w_{i^*}(t) + \varphi(\xi - w_{i^*}(t)) - \xi\| \\ &= \|(1-\varphi)(w_{i^*}(t) - \xi)\| \\ &= |1-\varphi| \cdot \|w_{i^*}(t) - \xi\|. \end{aligned}$$

Since we may assume without loss of generality that $0 < \varphi < 2$, we have

$$\|w_{i^*}(t+1) - \xi\| \leq \|w_{i^*}(t) - \xi\|$$

showing that the new weight vector is indeed closer to the input ξ than the previous one.

One can show that (7.5) is another gradient descent method, coming from minimizing the energy function

$$E(W) = \frac{1}{2} \sum_{j=1}^{N} \|\xi^{(j)} - w_{i^*(W\xi^{(j)})}\|^2$$

where we assume once more that the input can only take finitely many values $\xi^{(1)}, \ldots, \xi^{(N)}$. To see this, introduce a matrix M defined by

$$M_{jl} = \begin{cases} 1 & \text{if } l = i^* = i^*(W\xi^{(j)}) \\ 0 & \text{otherwise} \end{cases}$$

and write

$$\begin{aligned} E(W) &= \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{n} (\xi_k^{(j)} - W_{i^*k})^2 \\ &= \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{n} \sum_{l=1}^{p} M_{jl} (\xi_k^{(j)} - W_{lk})^2. \end{aligned}$$

Since M is a piecewise constant function of W, this is in fact only piecewise differentiable. However, on any such piece,

$$\frac{\partial E}{\partial W_{\mu\nu}} = \frac{\partial}{\partial W_{\mu\nu}} \left(\frac{1}{2} \sum_{j=1}^{N} M_{j\mu} (\xi_\nu^{(j)} - W_{\mu\nu})^2\right) = -\sum_{j=1}^{N} M_{j\mu} (\xi_\nu^{(j)} - W_{\mu\nu}).$$

Therefore,

$$\frac{\partial E}{\partial W_{\mu\nu}} = -\sum_j (\xi_\nu^{(j)} - W_{\mu\nu}),$$

where the summation runs over all over $j = 1, \ldots, N$ with $\mu = i^*(W\xi^{(j)})$, that is, over all patterns for which μ is the winner neuron. If there is no such j, the summation is empty and it evaluates to zero. Putting

$$W_{\mu\nu}(t+1) - W_{\mu\nu}(t) = -\varphi \frac{\partial E}{\partial W_{\mu\nu}}$$

as usual, we obtain

$$w_\mu(t+1) - w_\mu(t) = \varphi \sum_j (\xi^{(j)} - w_\mu).$$

This can be seen as an averaged version of (7.5).

Note that in (7.5), only the weight vector of the winner is updated, whereas the other weight vectors remain unchanged. This is known as a *winner-take-all* rule. Kohonen proposed a variant in which not only the winner weight vector is modified, but also some neighborhood of the winner.

Self-organization

Instead of the winner-take-all rule (7.5) with $w_i(t+1) = w_i(t)$ for $i \neq i^*$, Kohonen [14] proposed the learning rule

$$w_i(t+1) - w_i(t) = \varphi \Lambda(i, i^*)(x(t) - w_i(t))$$

for all $i = 1, \ldots, p$. We assume that $n = 1$ in the following. The **neighborhood function** Λ satisfies

- $0 \leq \Lambda(i,j) \leq 1$ for all $i, j = 1, \ldots, p$.
- $\Lambda(i,j)$ depends only on $|i-j|$, that is, it can be written as

  $$\Lambda(i,j) = \lambda(|i-j|).$$

- λ is a decreasing function, that is,

  $$r \leq r' \quad \Rightarrow \quad \lambda(r) \geq \lambda(r').$$

  Moreover, $\lambda(0) = 1$.

A typical choice for Λ is $\Lambda(i,j) = e^{-|i-j|^2/2\sigma^2}$. Here, we only consider

$$\Lambda(i,j) = \begin{cases} 1 & \text{if } |i-j| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Then the **Kohonen rule** says

$$w_i(t+1) - w_i(t) = \begin{cases} \varphi(x(t) - w_i(t)) & \text{if } |i - i^*| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

The winner i^* is again determined by

$$|w_{i^*} - x| \leq |w_i - x| \quad \text{for } i = 1, \dots, p.$$

Since $n = 1$, the input x and the weights w_i are scalar quantities.

The **index of disorder** of the sequence $w = (w_1, \dots, w_p)$ is defined by

$$D(w) = \sum_{i=2}^{p} |w_i - w_{i-1}| - |w_p - w_1|.$$

Then $D(w) \geq 0$ for all $w \in \mathbb{R}^p$, since

$$|w_p - w_1| = |\sum_{i=2}^{p}(w_i - w_{i-1})| \leq \sum_{i=2}^{p} |w_i - w_{i-1}|$$

and equality holds, i.e., $D(w) = 0$, iff all summands $w_i - w_{i-1}$ have the same sign, that is, iff w is monotonic, i.e., $w_1 \leq \ldots \leq w_p$ or $w_p \leq \ldots \leq w_1$.

Under certain mild assumptions on the distribution of the input x, one can show that Kohonen's rule leads almost surely (i.e., with probability one) to an ordering of the weights, i.e., the index of disorder tends to zero. This explains the term *self-organization*.

Here, we have only considered the one-dimensional case ($n = 1$). For higher dimensions, Kohonen's approach yields promising results, but no formal proof of convergence has yet been given.

Appendix A

Boolean Logic

A fundamental notion in Boolean logic is that of a binary variable. This is concerned with variables that can only take two different values, such as true/false or 0/1. The natural operations on such binary variables are conjunction (AND), disjunction (OR), and negation (NOT). This leads to the subject of Boolean lattices, a fundamental concept in mathematical logic.

Definition A.1 A **Boolean lattice** B is a set that contains at least two elements 0 and 1, and is provided with three operations

$$\wedge : B \times B \to B, \qquad \vee : B \times B \to B, \qquad {}^{-} : B \to B$$

such that for all $x, y, z \in B$:

1.	$x \wedge x = x$, $x \vee x = x$	(Idempotency)
2.	$x \wedge y = y \wedge x$, $x \vee y = y \vee x$	(Commutativity)
3.	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$ $x \vee (y \vee z) = (x \vee y) \vee z$	(Associativity)
4.	$x \wedge (x \vee y) = x$, $x \vee (x \wedge y) = x$	(Absorption)
5.	$x \wedge (y \vee (x \wedge z)) = (x \wedge y) \vee (x \wedge z)$ $x \vee (y \wedge (x \vee z)) = (x \vee y) \wedge (x \vee z)$	(Modularity)
6.	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	(Distributivity)
7.	$x \wedge 0 = 0$, $x \vee 0 = x$ $x \wedge 1 = x$, $x \vee 1 = 1$	
8.	$x \wedge \overline{x} = 0$, $x \vee \overline{x} = 1$	(Complementarity)
9.	$\overline{\overline{x}} = x$	(Involution)
10.	$\overline{x \wedge y} = \overline{x} \vee \overline{y}$, $\overline{x \vee y} = \overline{x} \wedge \overline{y}$	(De Morgan rules)

These properties are redundant and dependencies exist. For example, the idempotency (1) follows from the absorption laws (4). For this, let $y = x \wedge x$. Then,

$$x \wedge (x \vee (x \wedge x)) = x.$$

On the other hand, $x \vee (x \wedge x) = x$, and thus $x \wedge x = x$.

As can be seen from the above list of rules, statements about binary variables are highly symmetric with respect to replacing $\wedge$ by $\vee$, 0 by 1, and vice versa. More precisely, the following duality principle is valid.

Theorem A.1 (Duality Principle) *Each statement about Boolean lattices in which only* $\wedge, \vee, \overline{}, 0, 1$ *appear, remains true if* $(\wedge, \vee)$ *is replaced by* $(\vee, \wedge)$ *and* $(0,1)$ *by* $(1,0)$.

Each Boolean lattice has a canonical ordering **(lattice ordering)** defined by

$$a \leq b \quad :\Leftrightarrow \quad a \wedge b = a$$

for $a, b \in B$. This is a partial ordering on B.

Remark A.2 *We have [5]*

$$a \wedge b = \inf\{a, b\} \quad \textit{and} \quad a \vee b = \sup\{a, b\}.$$

Due to Remark A.2, the lattice operations $\wedge$ and $\vee$ are uniquely determined by the order. Therefore, a Boolean lattice can be alternatively defined as follows: A **Boolean lattice** is a partially ordered set $(B, \leq)$ that contains a least element 0, and a greatest element 1, and has the properties

1. For all $x, y \in B$, there exists $\inf\{x, y\}$ and $\sup\{x, y\}$.
2. For all $x, y, z \in B$, we have
$$\begin{aligned} \inf\{x, \sup\{y, z\}\} &= \sup\{\inf\{x, y\}, \inf\{x, z\}\} \\ \sup\{x, \inf\{y, z\}\} &= \inf\{\sup\{x, y\}, \sup\{x, z\}\} \qquad \text{(Distributivity)}. \end{aligned}$$
3. For each $x \in B$, there exists a unique $\overline{x} \in B$ with $\inf\{x, \overline{x}\} = 0$ and $\sup\{x, \overline{x}\} = 1$ (Complementarity).

Also these requirements are partially redundant [4]. Still equivalently, a Boolean lattice can be regarded as a **Boolean algebra**. A Boolean algebra is a $\mathbb{Z}/2\mathbb{Z}$–algebra B whose elements are idempotent, that is, $x^2 = x$ for all $x \in B$. Now it can easily be shown:

Theorem A.3 *Each Boolean lattice is a Boolean algebra by means of*

$$\begin{aligned} x+y &:= (x \wedge \overline{y}) \vee (\overline{x} \wedge y) \quad \textit{(exclusive or)} \\ x \cdot y &:= x \wedge y. \end{aligned}$$

Conversely, any Boolean algebra defines a Boolean lattice by

$$\begin{aligned} x \wedge y &:= x \cdot y \\ x \vee y &:= x + y + x \cdot y \\ \overline{x} &:= 1 + x. \end{aligned}$$

Since Boolean algebras are $\mathbb{Z}/2\mathbb{Z}$–algebras, we have

$$2x = x + x = 0 \quad \text{for all } x \in B.$$

Basic examples of Boolean lattices are:

1. The *switching algebra* $B = \{0, 1\}$ is a Boolean lattice with the operations defined by the following tables:

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$^{-}$	0	1
	1	0

   As a Boolean algebra, B is provided with the operations

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

   that is, B coincides with the field $\mathbb{Z}_2 := \mathbb{Z}/2\mathbb{Z}$.

2. For any $n \in \mathbb{N}$, the power set $\mathcal{P}(n)$ is the set of all subsets of $[n] := \{1, \ldots n\}$. It is a Boolean lattice, where for $A, B \subseteq [n]$,

$$\begin{aligned} A \wedge B &= A \cap B \\ A \vee B &= A \cup B \\ \overline{A} &= [n] \setminus A. \end{aligned}$$

Example 2 is already the general example of finite Boolean lattices. This follows from the subsequent theorem, whose proof can be found in [8].

Theorem A.4 ***(Representation theorem of finite Boolean lattices)*** *Any finite Boolean lattice B is isomorphic to a finite lattice of sets, i.e., there exists $n \in \mathbb{N}$ with $B \cong \mathcal{P}(n)$.*

According to Theorem A.4, for any Boolean lattice, we can find $n \in \mathbb{N}$ and a bijection

$$\Psi : B \to \mathcal{P}(n)$$

with

$$\begin{aligned} \Psi(x \wedge y) &= \Psi(x) \cap \Psi(y) \\ \Psi(x \vee y) &= \Psi(x) \cup \Psi(y) \\ \Psi(\overline{x}) &= [n] \setminus \Psi(x) \\ \Psi(0) &= \emptyset \\ \Psi(1) &= [n]. \end{aligned}$$

In particular, any finite Boolean lattice has the cardinality 2^n for some $n \in \mathbb{N}$.

Definition A.2 Let B be a Boolean lattice and let $n \in \mathbb{N}$. An ***n*-ary Boolean function** is a mapping $f : B^n \to B$. When $B = \{0, 1\}$, Boolean functions are also called **switching functions**.

Note: There are 2^{2^n} n-ary switching functions.

Disjunctive Normal Form

In this section, only the elementary switching algebra $\{0, 1\}$ will be considered. Each Boolean function

$$f : \{0, 1\}^n \to \{0, 1\}$$

is uniquely determined by the fiber $f^{-1}(1)$ (or $f^{-1}(0)$). For the coding of the 2^n elements of $\{0, 1\}^n$, the binary representation (dyadic expansion) may be used.

Definition A.3 For $i \in I := \{0, 1, \ldots, 2^n - 1\}$, let $i = i_1 2^{n-1} + i_2 2^{n-2} + \cdots + i_n 2^0$ be the unique **binary expansion** of i. This defines a bijective mapping

$$\mathrm{Bin} : I \to \{0, 1\}^n, \quad i \mapsto \mathrm{Bin}(i) := (i_1, \ldots, i_n).$$

With this definition, we obtain a bijection of the set of all switching functions onto the set of all functions from I to $\{0, 1\}$ by

$$\begin{aligned} \mathrm{Map}(\{0, 1\}^n, \{0, 1\}) &\cong \mathrm{Map}(I, \{0, 1\}) \\ f &\longmapsto f \circ \mathrm{Bin}. \end{aligned}$$

Due to this bijective correspondence, any switching function f can also be identified with a step function on the interval $[0, 2^n)$, namely by putting

$$f(x) = \begin{cases} f(\mathrm{Bin}(0)) & \text{for } 0 \le x < 1 \\ f(\mathrm{Bin}(1)) & \text{for } 1 \le x < 2 \\ \quad\vdots & \quad\quad\vdots \\ f(\mathrm{Bin}(2^n - 1)) & \text{for } 2^n - 1 \le x < 2^n. \end{cases}$$

A switching function f is represented uniquely by its table of values:

x	x_1	x_2	$\cdots$	x_{n-1}	x_n	$y = f(x_1, \ldots, x_n)$
0	0	0	$\cdots$	0	0	$f(0, \ldots, 0, 0)$
1	0	0	$\cdots$	0	1	$f(0, \ldots, 0, 1)$
2	0	0	$\cdots$	1	0	$f(0, \ldots, 1, 0)$
3	0	0	$\cdots$	1	1	$f(0, \ldots, 1, 1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	i_1	i_2	$\cdots$	i_{n-1}	i_n	$f(i_1, \ldots, i_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^n - 1$	1	1	$\cdots$	1	1	$f(1, \ldots, 1)$

It will now be shown that any switching function

$$f : \{0,1\}^n \to \{0,1\}$$

can be expressed in terms of the operations $\wedge$, $\vee$, $\overline{}$.

Theorem A.5 (Disjunctive normal form) *Each switching function*

$$f : \{0,1\}^n \to \{0,1\}$$

has, up to the order of the terms, a unique representation

$$f(x) = \bigvee_{i \in f^{-1}(1)} x^i,$$

where $x = (x_1, \ldots, x_n)$, $i = (i_1, \ldots, i_n)$, $x^i = x_1^{i_1} \wedge \cdots \wedge x_n^{i_n}$, *and for all* j,

$$x_j^1 = x_j \quad \textit{and} \quad x_j^0 = \overline{x_j}.$$

Proof: It is easy to see that for $\xi, i \in \{0,1\}$

$$\xi^i = \begin{cases} 1 & \text{if } \xi = i \\ 0 & \text{otherwise.} \end{cases}$$

By conjunction, the same holds for $\xi, i \in \{0,1\}^n$, that is,

$$\xi^i = \xi_1^{i_1} \wedge \cdots \wedge \xi_n^{i_n} = \begin{cases} 1 & \text{if } \xi = i \\ 0 & \text{otherwise.} \end{cases}$$

Hence for any $J \subseteq \{0,1\}^n$,

$$f(\xi) = \bigvee_{i \in J} \xi^i = \begin{cases} 1 & \text{if } \xi \in J \\ 0 & \text{otherwise.} \end{cases}$$

Note that any function of the form $\bigvee_{i \in J} x^i$ is uniquely determined by the set J. The desired result follows with $J = f^{-1}(1)$. □

The proof for Theorem A.5 also yields a method for calculating the disjunctive normal form. Let, for example, f be given by

x	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Then $f^{-1}(1) = \{(0,0,1), (1,0,1), (1,1,1)\}$ and hence,

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1^0 x_2^0 x_3^1 \vee x_1^1 x_2^0 x_3^1 \vee x_1^1 x_2^1 x_3^1 \\ &= \overline{x}_1 \overline{x}_2 x_3 \vee x_1 \overline{x}_2 x_3 \vee x_1 x_2 x_3 \end{aligned}$$

is the disjunctive normal form of f.

The disjunctive normal forms of some elementary Boolean functions are given in the following table:

	Boolean function	Disjunctive normal form
AND	$x_1 \wedge x_2$	$x_1 \wedge x_2$
OR	$x_1 \vee x_2$	$(x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2) \vee (x_1 \wedge x_2)$
XOR	$(x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$	$(x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$
NAND	$\overline{x_1 \wedge x_2}$	$(\overline{x}_1 \wedge \overline{x}_2) \vee (x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$
NOR	$\overline{x_1 \vee x_2}$	$\overline{x}_1 \wedge \overline{x}_2$

Similarly, the **conjunctive normal form** of a switching function f is obtained by computing the disjunctive normal form of $\overline{f}$, defined by $\overline{f}(x) := \overline{f(x)}$, and by negating the resulting expression using De Morgan's laws. One obtains the following table:

	Boolean function	Conjunctive normal form
AND	$x_1 \wedge x_2$	$(x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_2) \wedge (x_1 \vee \overline{x}_2)$
OR	$x_1 \vee x_2$	$x_1 \vee x_2$
XOR	$(x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$	$(x_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2)$
NAND	$\overline{x_1 \wedge x_2}$	$\overline{x}_1 \vee \overline{x}_2$
NOR	$\overline{x_1 \vee x_2}$	$(\overline{x}_1 \vee x_2) \wedge (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_2)$

Shegalkin Polynomials

This section gives a representation of Boolean functions as polynomials, called Shegalkin polynomials. Observe that any polynomial $P \in \mathbb{Z}[x_1, \ldots, x_n]$ having the form

$$P(x) = \sum_{i \in \{0,1\}^n} a_i x^i, \quad x = (x_1, \ldots, x_n), \quad x^i = x_1^{i_1} \cdots x_n^{i_n}$$

(now with $x_i^0 = 1$, $x_i^1 = x_i$) is dependent on exactly 2^n coefficients a_i. Such a polynomial is said to be **multi-affine**.

Theorem A.6 *Each Boolean function $f : \{0,1\}^n \to \{0,1\}$ has a unique extension $f : [0,1]^n \longrightarrow [0,1]$ as a multi-affine polynomial*

$$f(x) = \sum_{i \in \{0,1\}^n} a_i x^i$$

with coefficients $a_i \in \mathbb{Z}$.

Proof: We define recursively

$$D_1 := \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \in \mathbb{Z}^{2\times 2} \quad \text{and} \quad D_k := \begin{bmatrix} D_{k-1} & 0 \\ D_{k-1} & D_{k-1} \end{bmatrix} \in \mathbb{Z}^{2^k \times 2^k} \text{ for } k > 1.$$

It is easy to check that computing the 2^n coefficients of the Shegalkin polynomial from the 2^n function values of $f : \{0,1\}^n \to \{0,1\}$, amounts to inverting the matrix D_n. In fact, we have to solve $a = D_n b$, where a is the vector of unknown coefficients and b is the vector of given function values (both arranged according to the order induced by the function Bin). However, we have $\det(D_n) = 1$ for all n, and therefore, the matrices D_n are invertible as matrices over $\mathbb{Z}$. Indeed, we have

$$D_1^{-1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad D_k^{-1} = \begin{bmatrix} D_{k-1}^{-1} & 0 \\ -D_{k-1}^{-1} & D_{k-1}^{-1} \end{bmatrix} \text{ for } k > 1.$$

Therefore, $b = D_n^{-1}a$ is well-defined and uniquely determined, and it has only integer entries. Since each function $x_i \mapsto f(x_1, \dots, x_n)$ (with fixed values of the remaining variables) is affine, its maxima and minima are assumed on the boundary of $[0,1]$. Thus the representation extends to a continuous mapping from $[0,1]^n$ to $[0,1]$. □

Remark: There is a simple procedure for computing the Shegalkin polynomial from a Boolean expression for $f : \{0,1\}^n \to \{0,1\}$, for example, from its disjunctive normal form. If we put, similarly as in Theorem A.3,

$$\begin{aligned} x \wedge y &= x \cdot y \\ x \vee y &= x + y - x \cdot y \\ \overline{x} &= 1 - x, \end{aligned}$$

we obtain a polynomial whose values on $\{0,1\}^n$ coincide with the values of the given function. Since $x^2 = x$ for $x = 0, 1$, squares and higher powers can be eliminated, i.e., the polynomial can be reduced to multi-affine form without changing its values on $\{0,1\}^n$.

We obtain polynomial representations of the following switching functions:

	Boolean function	Shegalkin polynomial
AND	$x_1 \wedge x_2$	$x_1 x_2$
OR	$x_1 \vee x_2$	$x_1 + x_2 - x_1 x_2$
XOR	$(x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$	$x_1 + x_2 - 2x_1 x_2$
NAND	$\overline{x_1 \wedge x_2}$	$1 - x_1 x_2$
NOR	$\overline{x_1 \vee x_2}$	$1 - x_1 - x_2 + x_1 x_2$

For instance, the XOR function is first expressed as

$$x_1(1 - x_2) + (1 - x_1)x_2 - x_1(1 - x_2)(1 - x_1)x_2.$$

Using $x_1^2 = x_1$, $x_2^2 = x_2$, one obtains the multi-affine form given above.

Appendix B

Generalized Inverses

For every matrix $A \in \mathbb{R}^{m\times n}$, there exists a unique matrix $A^+ \in \mathbb{R}^{n\times m}$ with

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. AA^+ and A^+A are symmetric.

This matrix is called the **generalized inverse** or **pseudo-inverse** of A.

There are a few special cases in which an explicit formula for the pseudo-inverse can be given: If A is invertible, then of course $A^+ = A^{-1}$. If A has full column rank, then A^TA is non-singular, and the generalized inverse can be computed by $A^+ = (A^TA)^{-1}A^T$. Then we have $A^+A = I$, that is, A^+ is even a **left inverse** of A. Similarly, if A has full row rank, we put $A^+ = A^T(AA^T)^{-1}$ and then $AA^+ = I$ and A^+ is a **right inverse** of A. For computing the generalized inverse of a matrix without full rank, one uses the following.

For every matrix $A \in \mathbb{R}^{m\times n}$, there exists a factorization

$$A = U\Sigma V^T, \tag{B.1}$$

where $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}$$

with $\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ for some $\sigma_1 \geq \ldots \geq \sigma_r > 0$, where r denotes the rank of A. The representation (B.1) is called **singular value decomposition** of A, and the real numbers $\sigma_1, \ldots, \sigma_r$ are called **singular values** of A. They are uniquely determined by A (whereas U and V are not). We have $\sigma_1^2 = \|A\|_2^2 = \lambda_{\max}(AA^T) = \lambda_{\max}(A^TA)$.

Define $\Sigma_1^+ = \Sigma_1^{-1} = \mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1})$ and

$$\Sigma^+ = \begin{bmatrix} \Sigma_1^+ & 0 \\ 0 & 0 \end{bmatrix}.$$

Finally, for A, we set

$$A^+ = V\Sigma^+ U^T.$$

It is easy to check that this matrix is indeed the pseudo-inverse of A.

The **Frobenius norm** of a matrix $A \in \mathbb{R}^{m\times n}$ is defined by

$$\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n |A_{ij}|^2 = \mathrm{trace}(A^T A) = \mathrm{trace}(AA^T).$$

The pseudo-inverse A^+ is the best approximation of an inverse of A in the sense that the functions

$$\begin{aligned} f &: \mathbb{R}^{n\times m} \to \mathbb{R}, \quad X \mapsto \|AX - I_m\|_F \\ g &: \mathbb{R}^{n\times m} \to \mathbb{R}, \quad X \mapsto \|XA - I_n\|_F \end{aligned}$$

both achieve their minimum at A^+.

The matrix $P := AA^+ \in \mathbb{R}^{m\times m}$ is an **orthogonal projector** of $\mathbb{R}^m$ onto the vector space $V = \mathrm{im}(A) \subset \mathbb{R}^m$, which is spanned by the columns of A. The term "projector" refers to the fact that $P^2 = P$ and "orthogonal" means that

$$\langle Px, (I-P)y\rangle = 0 \quad \text{for all } x, y \in \mathbb{R}^m,$$

that is, $P^T(I-P) = 0$ or $P^T = P^T P = P$. Clearly, a projector is orthogonal if and only if it is symmetric. There is an orthogonal direct sum decomposition

$$\mathbb{R}^m = V \oplus V^\perp, \quad x = Px + (I-P)x.$$

In particular, $V = \mathrm{im}(A) = \mathrm{im}(P)$ and $V^\perp = \ker(A^T) = \mathrm{im}(I-P)$. Moreover, the relation $P = P^T P$ has the following consequences:

1. P is positive semi-definite, since for any $x \in \mathbb{R}^m$,

$$x^T P x = x^T P^T P x = \|Px\|^2 \geq 0.$$

2. We have $P_{ii} = \sum_{k=1}^m P_{ki}^2$ for all i. Equivalently,

$$P_{ii} - P_{ii}^2 = \sum_{k\neq i} P_{ki}^2.$$

This implies that $0 \leq P_{ii} \leq 1$, and $\sum_{k\neq i} P_{ki}^2 \leq \frac{1}{4}$. In particular, $|P_{ki}| \leq \frac{1}{2}$ for all $k \neq i$.

Bibliography

[1] M. Anthony and N. Biggs. *Computational Learning Theory.* Cambridge Tracts in Theoretical Computer Science 30 (1992).

[2] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press (1999).

[3] M. Basu and Q. Liang. The Fractional Correction Rule: A New Perspective. *Neural Networks* 11, 1027–1039 (1998).

[4] G. Birkhoff. *Lattice Theory.* American Mathematical Society Colloquium Publications (1940).

[5] G. Birkhoff and T. C. Bartee. *Modern Applied Algebra.* McGraw-Hill (1970).

[6] N. K. Bose and P. Liang. *Neural Network Fundamentals.* McGraw-Hill (1996).

[7] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery* 2, 121–167 (1998).

[8] G. Grätzer. *Lattice Theory.* Freeman (1971).

[9] T. N. E. Greville. Some Applications of the Pseudoinverse of a Matrix. *SIAM Review* 2, 15–22 (1960).

[10] J. Gruber. *Analyse und Synthese von Hopfield-Netzen.* M. Sc. Thesis, University of Regensburg, Germany (1995).

[11] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley (1991).

[12] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I.* Springer (1993).

[13] Y. Kamp and M. Hasler. *Recursive Neural Networks for Associative Memory.* Wiley (1990).

[14] T. Kohonen. *Self-organizing Maps.* Springer (1995).

[15] M. L. Minsky and S. A. Papert. *Perceptrons.* MIT Press (1988).

[16] W. Rudin. *Functional Analysis.* McGraw-Hill (1973).

[17] B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods: Support Vector Learning.* MIT Press (1999).

[18] K.-Y. Siu, V. Roychowdhury and T. Kailath. *Discrete Neural Computation: A Theoretical Foundation.* Prentice Hall (1995).

[19] M. Vidyasagar. *A Theory of Learning and Generalization.* Springer (1997).

[20] K. Yosida. *Functional Analysis.* Springer (1980).