4. Organizing Unorganized Machines

*Any sufficiently advanced technology is indistinguishable from magic.*


The past years have produced an impressive amount of research on machine learning. Learning generally allows a system to improve its performance by experience. Historically, the earliest forms of supervised learning (learning with a teacher) involved changing the synaptic weights so as to minimize the error of the network. Today, one of the most popular learning techniques is the back-propagation algorithm: the error of the output units is propagated back into the network to yield estimates of how much a given hidden unit contributed to the global output error. The estimates are then used to adjust the synaptic weights. There are currently a large number of learning algorithms used to change the synaptic weights of many kinds of neural networks. A lot of work is also focused on the understanding of the interaction of learning and evolution in biological systems. From an engineering point of view, the combination of learning and evolution shows that significant advantages can be gained in the adaptation of systems to an environment for a given task. We shall see in this chapter an evolutionary learning approach for Turing neural networks.

In his 1947 paper, Turing rather naively suggested that the infant human cortex should be considered as an unorganized machine. This statement might certainly be questioned since it largely oversimplifies the brain. From a biological point of view, there is a good reason to consider the embryos growing cortex as an almost unorganized network because the DNA definitely has insufficient storage capacity for the complete specification (i.e., interconnections, positions, etc.) of the entire human brain. However, the brain is neither constructed at random but the construction is guided by many self-organizing principles.

As we have seen earlier in this book, Turing’s unorganized machines can be classified into two different classes:

1. machines that allow interference by some agent (e.g., BI-type or TBI-type networks) and
2. machines without the possibility of interference (e.g., A-type machines).
Turing networks that allow interference could either be organized by an external agent (supervisor) or by self-modification. Networks that do not allow interference are, once an architecture is chosen and the nodes are initialized, no longer modifiable and will thus evolve deterministically towards an attractor. Training Turing's neural networks is a process where each interconnection is set by means of its interfering inputs into a state that allows the network to perform its desired task. Classical neural networks usually have weighted connections with weights having real values. In contrast, there is nothing smooth in switching Turing's interconnections in a network. The connection is either enabled or disabled—a savage all-or-nothing shift. Thus, useless neurons are simply disconnected from the rest of the network (known as pruning).

This chapter deals with some methods that organize unorganized machines. Turing himself already mentioned a method called "genetical" search that will be further described and applied in this chapter. Today, this method is better known under the term genetic algorithm. He was probably one of the first persons who had the idea to apply an evolutionary algorithm to neural networks. In the next section, I shall first give a very brief introduction to evolutionary algorithms. Evolutionary algorithms can be applied to neural networks to find an architecture, the connection weights, or the learning rules. Different encoding techniques shall also be introduced. Evolutionary algorithms will then be used for different pattern classification problems.

4.1 Evolutionary Algorithms

Evolutionary algorithms (EA) are a collection of methodologies inspired by the principles of the biological evolution. The basic concepts go back to the work of Charles Darwin [42]. It is interesting that Turing himself already mentioned "genetical" or "evolutionary" search in his 1948 paper [192, p. 23] in the context of problem solving.

Later, John Holland first introduced and substantiated the idea of genetic algorithms (GAs) [93]. To date, many variations and extensions of algorithms and methods inspired by the biological evolution have been proposed. One of the most recent milestones was the introduction by John Koza [106] of a method called genetic programming that deals with the automatic generation of computer code. For further reading of general interest, the reader is referred to Banzhaf [20], Michalewicz [120], Bäck [16], Mitchell and Forrest [127], Koza [107], Vose [205], and Fogel [60].

The term evolutionary algorithm usually encompasses a number of related methodologies such as genetic algorithms, evolutionary strategies, evolutionary programming, genetic programming, etc.

Mathematically speaking, evolutionary algorithms are a broad collection of optimization methods that are particularly suitable for "hard" problems