Chapter 5
Learning as a Control Process

5.1 Introduction

The commonly used algorithm for the multilayer neural networks learning, the backpropagation algorithm described in the Chap. 4, is a gradient descent method for searching minimum of a performance index of learning. The performance index, being the measure of neural network learning quality, is a multimodal function. Application of this kind of algorithms causes frequent stopping at a local minimum. Various modifications of this algorithm still cannot avoid local minimal points. Until now, in practice, the only way of trying to find the near global optimum solution is to perform computation several times with different initial weight values and then to choose the best solution.

The backpropagation algorithm does not use the special layered structure of the multilayer networks. In this chapter we propose a new global algorithm for neural networks learning. The algorithm is based on the dynamic programming principle introduced by Bellman in the early 1950s (Bellman 1972, Bertsekas 1995), and allows, at least theoretically, for finding of the global minimum of the learning error. The learning of a multilayer neural network is considered as a special case of the multistage optimal control problem, first proposed by Krawczak and Mizukami (1994), and developed by Krawczak (e.g. 1995a, 1995b, 1999b, 2000a, 2001a, 2001b, 2004d, 2004g, 2005b, 2006a). The gist of the new algorithm for learning of multilayer neural networks consists of aggregating neurons within separate layers and then considering such a system as a particular multistage optimal control problem. Thus, layers become stages, while weights - controls. The problem of optimal weight adjustment is converted into a problem of optimal control.

The multistage optimal control problem can be solved by application of the dynamic programming (Bryson and Ho 1969, Cruz (1977, Roitenberg 1978, Luenberger 1984). For the new algorithm the return functions for each layer are defined, and minimization of these functions is performed layer by layer, starting from the last layer. This approach gives a real possibility of performing global optimisation. There are obstacles to the application of dynamic programming; one
is the *curse of dimensionality* – the computational burden, and the second is the memory requirement, growing exponentially with the state and control dimensionality. Fortunately, there is a way to avoid this kind of difficulties by introducing some approximation of the return functions, see Jacobson and Mayne (1979) or Yakowitz Rutherford (1984). They proposed a method to approximate the return function by considering the second-order terms in the Taylor expansion of the functions. It seems that there is possibility using the first-order method only but with application of the conjugate gradient algorithm, which converges to the inverse of the proper Hessian matrix.

In some sense, it is an application of the idea of the *neuro-dynamic programming* for the neural network learning, process introduced by Bertsekas and Tsitsiklis (1996). The term “neuro” is equivalent in this context to any kind of function approximation.

### 5.2 Multistage Neural Systems

Let us assume each neuron is given by the following expressions

\[
x_{pj(l)} = f\left(\text{net}_{pj(l)}\right)
\]

(5.1)

\[
\text{net}_{pj(l)} = \sum_{i(l-1)=1}^{N(l-1)} w_{i(l-1)j(l)} x_{pi(l-1)}
\]

(5.2)

where \(x_{pj(l)}\) is the scalar output of the \(j\)-th neuron, \(j(l) = 1, 2, ..., N(l)\), situated within the layer \(l\), \(l = 1, 2, ..., L\), the index \(p = 1, 2, ..., P\) indicates the number of a pattern, and \(f\left(\text{net}_{pj(l)}\right)\) is the differentiable activation function of the neuron \(j(l)\), while \(\text{net}_{pj(l)}\) is the input to the neuron \(j(l)\) coming from the layer \((l-1)\). The notation used is a little bit different than that used in the previous chapter in order to emphasize the stage-wise nature of the network.

Fig. 5.1 shows a multilayer neural network with distinct layers. Now, let us aggregate neurons situated within each layer \(l\), \(l = 0, 1, 2, ..., L\), in a way described by the following expressions

\[
X(l) = [x_{1l(l)}, x_{2l(l)}, ..., x_{N(l)l}]^T \quad \text{for} \quad l = 0, 1, 2, ..., L
\]

(5.3)

\[
W(l-1) = [w_{1l(l-1)}, w_{2l(l-1)}, ..., w_{N(l-1)l}]^T
\]

(5.4)

where \(w_{jl(l-1)} = [w_{j1(l-1)}, w_{j2(l-1)}, ..., w_{jN(l-1)l}]^T\), for \(l = 1, 2, ..., L\), \(j(l-1) = 1, 2, ..., N(l)\).