UNIVERSITÀ CA' FOSCARI VENEZIA
DIPARTIMENTO DI MATEMATICA APPLICATA

RENDICONTI PER GLI STUDI ECONOMICI QUANTITATIVI

2004 Volume unico

Titolo della rivista e ISSN

RENDICONTI PER GLI STUDI ECONOMICI QUANTITATIVI 1591-9773

Multi-layer perceptron learning *via* Monte Carlo approach: a proposal

Marco Corazza

Dipartimento di Matematica Applicata
Universitá Ca' Foscari di Venezia
Dorsoduro 3025/E – 30123 Venezia, Italy
corazza@unive.it
http://www.dma.unive.it/~corazza/

Abstract. From 1974 to 1986 various scholars independently proposed a learning algorithm able to determine the values of the weights of the multi-layer perceptron: the one known as *error back-propagation*. A correct and profitable utilization of this learning algorithm - and of its several variants - needs the assumption of formal hypotheses and the skilful usage of some expedients. For example, from a theoretical point of view, any activation function of a multi-layer perceptron and the related cost function have to be differentiable with respect to the weights of the multi-layer perceptron and, from an operative standpoint, the user, on the subjective basis of her experience, has to specify the training-phase dynamics of some crucial parameters. Now, considering the importance of a suitable learning approach for the usability of the multi-layer perceptron, and given the limitations peculiar to the error back-propagation algorithm, in this paper we propose a Monte Carlo-based learning algorithm which is able to run without assuming specific analytical hypotheses and without the need of user's interventions. In particular, among the features characterizing our learning approach, the proposed Monte Carlo-based algorithm allows to associate to the estimator of each weight of the multi-layer perceptron a probability distribution converging (in distribution) to the standardized normal one; moreover, our learning algorithm performs a global search in the space of the weights, and because of that the learning process is not likely to stick in correspondence of local minimum points of the cost function.
Keywords. Multi-layer perceptron, learning algorithms, Monte Carlo methods.

J.E.L. CLASSIFICATION: C15, C45, D83.
M.S.C. CLASSIFICATION: 62M45, 65C05, 68T05.

1 Introduction

A possible reconstruction of the initial phase of the scientific history of the multi-layer perceptron (in the following: MLP) can be summarized as follows:

- from 1957 to 1961 Frank Rosenblatt realized the one-layer perceptron, the ancestor of the MLP, an artificial neural network able to separate in a finite number of training trials any given linearly separable classes;
- in 1969 Marvin Minsky and Seymour Papert published a book ([6]) in which they proved the incapability of the perceptron to implement some elementary logical and predicate functions like, for instance, the XOR; this theoretical result discouraged the research on neurocomputation for several years;
- although since the late sixties it was known that the MLP's architecture could have overcame the limitations peculiar to the one-layer perceptron, only from 1974 ([8]) to 1986 ([7]) various scholars independently proposed a learning algorithm able to determine in a suitable way the values of the weights of the MLP: the algorithm known as *error back-propagation*.

It is undeniable that the introduction of this learning algorithm has strongly contributed to the definitive take-off of the MLP.

It is also undeniable that a correct and profitable utilization of the error back-propagation algorithm - and of its several variants - needs the assumption of formal hypotheses and the skilful usage of some expedients. For example, from a theoretical point of view, any activation (or gain, or transfer) function of a MLP and the related cost function have to be differentiable with respect to the weights of the MLP itself (see, for more details, [4]), and, from an operative standpoint, the user, on the subjective basis of her experience, has to specify the training-phase dynamics of some crucial parameters like, for instance, the *learning rate* and the *momentum parameter* (see, for instance, [1]).

Now, considering the importance of a suitable learning approach for the usability of the MLP, and given the peculiar limitations of the error back-propagation algorithm, in this paper we propose a Monte Carlo-based learning algorithm which is able to run without assuming specific analytical hypotheses and without the need of user's interventions. In particular, among the various features characterizing our learning approach, in this section we anticipate the following ones:

- our learning algorithm allows to associate to the estimator of each weight of the MLP a probability distribution converging (in distribution) to the standardized normal;
- as will become clear beginning from the next sections, our Monte Carlo-based algorithm performs a global search in the space of the weights; because of that, the learning process is not likely to stick in correspondence of local minimum points of the cost function.

The remainder of this paper is organized as follows: in the next section we shall present our learning algorithm and some theoretical results related to it; in section **3** we shall report the positive features characterizing our learning approach and some theoretical results related to them; in section **4** we shall describe the main shortcomings characterizing our learning algorithm; and in section **5** we shall give some concluding remarks.

2 The Monte Carlo-based learning algorithm

Before we present our Monte Carlo-based learning algorithm we need a little bit of notation which will be useful in this section and in the next ones.

Henceforth we take into account a MLP characterized by M weights. In detail:

- $w(i)$, with $i = 1, \dots, M$, is a random variable denoting the i-th weight;
- $\underline{w} = (w(1), \dots, w(i), \dots, w(M))$ is the vector of the weights.

Furthermore, in each of the N iterations the algorithm has to run:

- $w_j(i)$, with $j = 1, \dots, N$ and $i = 1, \dots, M$, is the value that the learning algorithm assigns to the i-th weight at the j-th iteration; in particular, each $w_j(i)$ can take values in a suitably pre-established real interval $[a(i), b(i)]$, with $a(i) < b(i)$ (notice that $w_j(i)$ is the realization of the random variable with $w(i)$ at the j-th iteration);
- $\underline{w}_j = (w_j(1), \dots, w(i)_j, \dots, w_j(M))$, with $j = 1, \dots, N$ and $i = 1, \dots, M$, is the vector of the realized values the learning algorithm assigns to the M weights at the j-th iteration; in particular, each $\underline{w}_j$ can take values in the weight space $\mathbf{W} = [a(1), b(1)] \times \dots \times [a(i), b(i)] \times \dots \times [a(M), b(M)]$.[1]

Now, we can present our Monte Carlo-based learning algorithm in the following itemized form:

step 0: let $D := \left\{ \left(\underline{x}_l, \underline{y}_l \right), l = 1, \dots, L \right\}$ be the input-output data set, where $\underline{x}_l$ denotes the l-th vector of the inputs, $\underline{y}_l$ denotes the l-th vector of the outputs, and L denotes the number of the input-output patterns; moreover, let $C\left(\underline{y}_1, \dots, \underline{y}_L, \hat{\underline{y}}_{1,\bar{j}}, \dots, \hat{\underline{y}}_{L,\bar{j}}\right)$ be a cost function, where $\hat{\underline{y}}_{l,\bar{j}}$, with $l = 1, \dots, L$, denotes the l-th vector of outputs determined by the MLP in correspondence to the $\bar{j}$-th vector of the realized values of the weights $\underline{w}_{\bar{j}}$ and to the l-th vector of inputs $\underline{x}_l$;

step 1: initialize the iteration counter j to 1;

step 2: determine the values $w_j(i)$, with $i = 1, \dots, M$, by generating a realization of M mutually independent random variables, each of them uniformly distributed in $[a(i), b(i)]$ respectively;

step 3: determine all MLP's vectors of outputs $\underline{y}_{l,j}$, with $l = 1, \dots, L$, in correspondence of the vector of the realized values of the weights $\underline{w}_j = (w_j(1), \dots, w(i)_j, \dots, w_j(M))$ and of the vector of inputs $\underline{x}_l$, with $l = 1, \dots, L$;

step 4: determine the value $C_j := C\left(\underline{y}_1, \dots, \underline{y}_L, \hat{\underline{y}}_{1,j}, \dots, \hat{\underline{y}}_{L,j}\right)$ of the cost function;

step 5: increase the iteration counter j by 1;

step 6: if $j \leq N$ then go to **step 2**, else go to the next step;

[1] Of course, we assume that the vector of the (unknown) true values of the weights belongs to $\mathbf{W}$.

step 7: order the vectors $\underline{w}_j = (w_j(1), \ldots, w(i)_j, \ldots, w_j(M))$, with $j = 1, \ldots, N$, according to the increasing values C_j that the cost function takes in correspondence of every vector $\underline{w}_j$ itself, and denote the so ordered vectors by $\underline{\widetilde{w}}_j = (\widetilde{w}_j(1), \ldots, \widetilde{w}(i)_j, \ldots, \widetilde{w}_j(M))$;

step 8: consider the first $\widetilde{N} \leq N$ of the vectors $\underline{\widetilde{w}}_j$, with $j = 1, \ldots, N$, *i.e.* $\underline{\widetilde{w}}_1, \ldots, \underline{\widetilde{w}}_j, \ldots, \underline{\widetilde{w}}_{\widetilde{N}}$, where $\widetilde{N}$ is the number of vectors belonging to a suitable neighbourhood of the vector of the true values of the weights $\underline{w}^* = (w^*(1), \ldots, w^*(i), \ldots, w^*(M))$ which is related to the unique[2] global minimum points of the cost function;[3]

step 9: compute the values of the weights of the MLP as follows: $\widehat{w}(i) = \frac{1}{\widetilde{N}} \sum_{j=1}^{\widetilde{N}} \widetilde{w}_j(i)$, with $i = 1, \ldots, M$;[4]

step 10: release the vector $\underline{\widehat{w}} = (\widehat{w}(1), \ldots, \widehat{w}(i), \ldots, \widehat{w}(M))$ of the values of the MLP's weights and end the learning algorithm.

Before to argue about the meaningfulness of the Monte Carlo-based learning algorithm we propose (see the next section), we spend some words about the determination of $\widetilde{N}$.

Recalling the definition of $\widetilde{N}$ given in **step 8**, we can determine it as follows:

$$\widetilde{N} = \lfloor Np \rfloor \tag{1}$$

where
p is the probability for a vector of realized values of the weights to belong to a suitable neighbourhood of $\underline{w}^*$.[5]

Notice that in such a way we define, though indirectly, the neighbourhood of $\underline{w}^*$.

In order to determine p, we give and prove the following proposition.

Proposition 1. *Let E_i denote the event "$w_j(i) \in (w^*(i) - \varepsilon^-(i), w^*(i) + \varepsilon^+(i))$", where $\varepsilon^-(i)$, $\varepsilon^+(i) > 0$, for a $j \in \{1, \ldots, N\}$ and with $i = 1, \ldots, M$. If $\underline{w}^* \in \mathbf{W}$, if the values $w_j(i)$ are generated by a realization of M mutually independent random variables, each of them uniformly distributed in $[a(i), b(i)]$ respectively, and if $\varepsilon^-(i) < w^*(i) - a(i)$ and $\varepsilon^+(i) < b(i) - w^*(i)$, then*

$$p = \Pr(E_1 \wedge \ldots \wedge E_i \wedge \ldots \wedge E_M) = \prod_{i=1}^{M} \frac{\varepsilon^+(i) + \varepsilon^-(i)}{b(i) - a(i)} \tag{2}$$

and

$$p \in (0, 1). \tag{3}$$

[2] The assumption on the uniqueness of the global minimum points of the cost function will be relaxed in section 4.

[3] In the continuation of this section we shall explain how to determine $\widetilde{N}$;

[4] By the notation "$\widehat{\cdot}$" we denote the estimator of the corresponding weight.

[5] $\lfloor \cdot \rfloor$ is the maximum integer which does not exceed the value taken by the expression inside the notation itself.

Proof. The values $w_j(i)$, for a $j \in \{1, \dots, N\}$ and with $i = 1, \dots, M$, are generated by a realization of M mutually independent random variables. Due to their mutual independence,

$$p = \Pr(E_1 \wedge \dots \wedge E_i \wedge \dots \wedge E_M) = \prod_{i=1}^{M} \Pr(E_i). \tag{4}$$

Furthermore, each of these random variables is uniformly distributed in $[a(i), b(i)]$ respectively, therefore

$$\begin{aligned} \Pr(E_i) &= \Pr\left(w_j(i) \in (w^*(i) - \varepsilon^-(i), w^*(i) + \varepsilon^+(i))\right) = \\ &= \{[w^*(i) + \varepsilon^+(i)] - [w^*(i) - \varepsilon^-(i)]\} \frac{1}{b(i) - a(i)} = \\ &= \frac{\varepsilon^+(i) + \varepsilon^-(i)}{b(i) - a(i)}, \text{ with } i = 1, \dots, M. \end{aligned} \tag{5}$$

Now, substituting (5) in (4) one obtains the thesis (2).

With regard to the thesis (3), recalling that by hypothesis $\varepsilon^-(i)$, $\varepsilon^+(i) > 0$ for all $i = 1, \dots, M$, and that by construction $a(i) < b(i)$ (see section **2**), one has

$$\Pr(E_i) = \frac{\varepsilon^+(i) + \varepsilon^-(i)}{b(i) - a(i)} > 0, \text{ with } i = 1, \dots, M,$$

from which $p > 0$.[6] Moreover, by hypothesis $\varepsilon^-(i) < w^*(i) - a(i)$ and $\varepsilon^+(i) < b(i) - w^*(i)$, with $i = 1, \dots, M$, therefore

$$\varepsilon^+(i) + \varepsilon^-(i) < [b(i) - w^*(i)] + [w^*(i) - a(i)] = b(i) - a(i), \text{ with } i = 1, \dots, M,$$

from which

$$\Pr(E_i) = \frac{\varepsilon^+(i) + \varepsilon^-(i)}{b(i) - a(i)} < 1, \text{ with } i = 1, \dots, M,$$

from which $p < 1$.[7]

3 The positive features of our learning algorithm

A first positive feature of our Monte Carlo-based learning algorithm is simply the one we anticipated in section **1**: our learning approach does not require the assumption of specific analytical hypotheses concerning the activation functions of the MLP and the related cost function, and moreover it does not need any intervention from the user.

[6] Notice that if there exists $\bar{i} \in \{1, \dots, M\}$ such that $\varepsilon^-\left(\bar{i}\right) = \varepsilon^+\left(\bar{i}\right) = 0$, then $p = 0$.

[7] Notice that if $\varepsilon^-(i) = w^*(i) - a(i)$ and $\varepsilon^+(i) = b(i) - w^*(i)$ for all $i = 1, \dots, M$, then $p = 1$.

All this contributes to improve the usability of the MLP. In particular, notice that it gives to the user the possibility to choose the cost function in a way suitably depending on the specific application to implement.

A second positive feature of our learning algorithm is due to the fact that the Monte Carlo-based approach we utilize satisfies the hypotheses of the *central limit theorem*. Consequently, it is possible to associate to the estimator of each MLP's weight - in the asymptotic way which follows - a probability distribution converging (in distribution) to the standardized normal:

$$\frac{\widehat{w}(i) - w^*(i)}{\widehat{s}(i) \Big/ \sqrt{\widetilde{N}}} \xrightarrow{d} N(0,1), \text{ with } i = 1, \dots, M, \tag{6}$$

where

$\widehat{s}(i)$ is the standard deviation $\sqrt{\frac{1}{\widetilde{N}} \sum_{j=1}^{\widetilde{N}} \left(w_j(i) - \widehat{w}(i)\right)^2}$, with $i = 1, \dots, M$ (see, for instance, [2], [3], and [5]).

From a training-phase point of view, the classical result (6) has an important implication.[8] Recalling from basic inferential statistics that the standard deviation of the estimator $\widehat{w}(i)$ is $\widehat{s}(i) \Big/ \sqrt{\widetilde{N}}$, with $i = 1, \dots, M$, it is possible to associate the confidence interval which follows to each of the (unknown) true values of the weights $w^*(i)$:

$$w^*(i) \in \left(\widehat{w}(i) - z_{\alpha/2}\frac{\widehat{s}(i)}{\sqrt{\widetilde{N}}}, \widehat{w}(i) + z_{\alpha/2}\frac{\widehat{s}(i)}{\sqrt{\widetilde{N}}}\right), \text{ with } i = 1, \dots, M, \tag{7}$$

where

$z_{\alpha/2}$ is the value taken by the standardized normal distributed random variable z in correspondence of one of the pre-established customary confidence interval α (for example: if $\alpha = 0.05$, then $z_{0.025} = 1.96$).

Thanks to the determination of the confidence interval reported in (7), our learning algorithm allows to give some assessment about the meaningfulness of the estimate of the MLP's weights.

A third positive feature we emphasize with regard to our Monte Carlo-based learning approach arises from the fact that the learning approach we propose performs a global search in the space of the weights $\mathbf{W}$. In order to prove this assertion, we give the following corollary (deriving from **Proposition 1**) in which we state that, at each iteration of our learning algorithm, all the neighbourhoods contained in $\mathbf{W}$ which differ among them only for their respective centers have the same probability to be "visited" by our Monte Carlo-based approach.

Corollary 1. *Let $E_{i,1}$ denote the event "$w_j(i) \in \left(\overline{\overline{w}}_1(i) - \varepsilon^-(i), \overline{\overline{w}}_1(i) + \varepsilon^+(i)\right)$" and let $E_{i,2}$ denote the event "$w_j(i) \in \left(\overline{\overline{w}}_2(i) - \varepsilon^-(i), \overline{\overline{w}}_2(i) + \varepsilon^+(i)\right)$", where*

[8] Of course, only for values of $\widetilde{N}$ sufficiently large the left-side probability distribution can be regarded as a standardized normal one.

$\{\overline{\overline{w}}_1(1), \ldots, \overline{\overline{w}}_1(i), \ldots, \overline{\overline{w}}_1(M)\}$, $\{\overline{\overline{w}}_2(1), \ldots, \overline{\overline{w}}_2(i), \ldots, \overline{\overline{w}}_2(M)\} \in \mathbf{W}$ and $\varepsilon^-(i)$, $\varepsilon^+(i) > 0$, for a $j \in \{1, \ldots, N\}$ and with $i = 1, \ldots, M$. If the values $w_j(i)$ are generated by a realization of M mutually independent random variables, each of them uniformly distributed in $[a(i), b(i)]$ respectively, and if $\varepsilon^-(i) < \overline{\overline{w}}_h(i) - a(i)$ and $\varepsilon^+(i) < b(i) - \overline{\overline{w}}_h(i)$, with $h = 1, 2$, then

$$\Pr(E_{1,1} \wedge \ldots \wedge E_{i,1} \wedge \ldots \wedge E_{M,1}) = \Pr(E_{1,2} \wedge \ldots \wedge E_{i,2} \wedge \ldots \wedge E_{M,2}) = \\ = \prod_{i=1}^{M} \frac{\varepsilon^+(i) + \varepsilon^-(i)}{b(i) - a(i)}. \tag{8}$$

Proof. It is sufficient to mime the proof provided for the thesis (2) substituting $\overline{\overline{w}}_h(i)$ to $w^*(i)$, with $h = 1, 2$ and with $i = 1, \ldots, M$.

Since our learning algorithm performs a global search in $\mathbf{W}$, the learning process is not likely to stick in correspondence of local minimum points of the cost function.

4 Some shortcomings of our learning algorithm

A first shortcoming of our learning approach is closely linked to the nature of the Monte Carlo-based methods. In detail, in order to reduce the standard deviation of the estimator $\widehat{w}(i)$ from $\frac{\widehat{s}(i)}{\sqrt{\widetilde{N}}}$ to $\frac{\widehat{s}(i)}{c\sqrt{\widetilde{N}}}$, where $c > 1$ and with $i = 1, \ldots, M$ (and consequently to reduce the amplitude of the confidence interval reported in (7), one has to increase $\widetilde{N}$ from $\lfloor Np \rfloor$ to $\lceil c^2 \lfloor Np \rfloor \rceil$ (see, for instance, [2] and [3]).[9],[10]

Because of it, sometimes profitable applications of our Monte Carlo-based learning algorithm could be time-consuming. In order to mitigate the undesirable effects of this feature, noticing from (1) and **Proposition 2** that for given M, $[a(i), b(i)]$ and $\widetilde{N}$, with $i = 1, \ldots, M$, the required number of iterations decreases as $\varepsilon^+(i) + \varepsilon^-(i)$ increases,[11] our learning algorithm could be utilized in accordance with the two-stage procedure we propose in the following itemized form:

- at first, perform a rough determination of the values of the weights of the MLP by using our learning approach with a not-so-particularly-small value of the width $\varepsilon^+(i) + \varepsilon^-(i)$;

[9] $\lceil \cdot \rceil$ is the minimal integer which exceeds the value taken by the expression inside the notation itself.

[10] Alternative approaches are focused on the reduction of $\widehat{s}(i)$, with $i = 1, \ldots, M$, but their application in not always straightforward (see, for more details, [2], [3], and [5]).

[11] We recall that $\varepsilon^+(i) + \varepsilon^-(i)$, with $i = 1, \ldots, M$, is the width of the real interval to whom the true value of the weight $w(i)$ belongs.

- then, refine the determination of the value of the weights of the MLP by using some suitable post-optimization numerical methods (see, for more details, [5]).

A second shortcoming of our Monte Carlo-based learning approach arises when the MLP's cost function is characterized by more than one global minimum point $\underline{w}_1^* = (w_1^*(1), \ldots, w_1^*(i), \ldots, w_1^*(M)), \ldots, \underline{w}_k^* = (w_k^*(1), \ldots, w_k^*(i), \ldots, w_k^*(M)), \ldots$, where $k > 1$. In such a case, the first $\widetilde{N}$ vectors of the weights $\underline{\widetilde{w}}_1, \ldots, \underline{\widetilde{w}}_j, \ldots, \underline{\widetilde{w}}_{\widetilde{N}}$ (see **steps 7** and **8**) could belong to neighbourhoods corresponding to different global minimum points; consequently, the sample mean $\widehat{w}(i)$, with $i = 1, \ldots, M$, would be meaningless.

In order to avoid this shortcoming, the learning algorithm presented in section **2** can be improved by inserting the following supplementary steps between **step 8** and **step 9**:

step 8.1: given the set of $\widetilde{N}$ vectors $\widetilde{\mathbf{W}} = \left\{\underline{\widetilde{w}}_1, \ldots, \underline{\widetilde{w}}_j, \ldots, \underline{\widetilde{w}}_{\widetilde{N}}\right\}$, perform some kind of cluster analysis on it in order to specify $Q > 1$ subsets $\widetilde{\mathbf{W}}_1, \ldots, \widetilde{\mathbf{W}}_q, \ldots, \widetilde{\mathbf{W}}_Q$, such that $\bigcup_{q=1}^{Q} \widetilde{\mathbf{W}}_q = \widetilde{\mathbf{W}}$;[12]

step 8.2: on the basis of some suitable cluster indicator, detect the "best" subset among the former Q ones, $\widetilde{\mathbf{W}}_{q^*}$, with $q^* \in \{1, \ldots, Q\}$, and detect its cardinality;

step 8.3: update $\widetilde{\mathbf{W}}$ to $\widetilde{\mathbf{W}}_{q^*}$ and update the value of $\widetilde{N}$ to the cardinality of $\widetilde{\mathbf{W}}_{q^*}$;

step 8.4: if $\widetilde{N} < \lfloor Np \rfloor$ then go to **step 2**, else go to **step 9**.

This improvement of our Monte Carlo-based learning algorithm allows to determine the sample means $\widehat{w}(i)$, with $i = 1, \ldots, M$, by using only the vectors of the weights belonging to the neighbourhood corresponding to the "best" of the global minimum points of the cost function.

5 Concluding remarks

A first remark we present arises from the (simple) consideration that the Monte Carlo-based learning algorithm we propose offers evidence for possible extensions. In fact, it could be easily modified in order to be able to perform learning in a wide variety of inferential approaches.

A second remark we give concerns with the fact that, from a methodological point of view, our learning algorithm does not require any a priori - both objective and subjective - probabilistic information about the (unique) process generating the input-output data set D (like, for instance, it occurs in the Bayesian approach instead).

[12] Of course, the specific kind of cluster analysis to perform has to be chosen in a manner such that all the vector(s) $\underline{\widetilde{w}}_{\bar{j}}$, with $\bar{j} \in \{1, \ldots, \widetilde{N}\}$, belonging to the same neighbourhood of a given global minimum point of the cost function has to belong to the same subset $\widetilde{\mathbf{W}}_{\bar{q}}$, with $\bar{q} \in \{1, \ldots, Q\}$.

A third remark we report is related to the fact that our learning algorithm could also be utilized as a post-optimization numerical method.

References

1. Belcaro, P.L., Canestrelli, E., and Corazza, M.: Artificial neural network forecasting models: an application to the Italian stock market. Badania Operacyjne i Decyzje **3-4** (1996) 29-48
2. Boyle, P.: Options: a Monte Carlo approach. Journal of Financial Economics **4** (1977) 323-338
3. Fishman, G.S.: Monte Carlo. Concepts, algorithms, and applications. Springer, New York (1996)
4. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the theory of neural computation. Addison-Wesley, Reading (1991)
5. Liu, J.S.: Monte Carlo strategies in scientific computing. Springer, New York (2001)
6. Minsky, M., Papert, S.: Perceptrons. MIT Press, Cambridge (1969)
7. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L., PDP Research Group (eds.): Parallel distributed processing: explorations in the microstructure of cognition. Foundations, vol. I, MIT Press, Cambridge (1986) 318-362
8. Werbos, P.J.: Beyond regression: new tools for prediction and analysis in the behavioral sciences. Doctoral Dissertation. Department of Applied Mathematics of the Harvard University, Cambridge (1974)