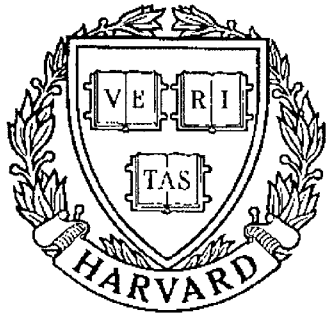
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Hierarchical Neural Networks for Monitoring Complex Dynamic Systems

by M.L. Mavrovouniotis

TR 89-95

HIERARCHICAL NEURAL NETWORKS FOR MONITORING COMPLEX DYNAMIC SYSTEMS

Michael L. Mavrovouniotis
Chemical Engineering and Systems Research Center
University of Maryland
College Park, MD 20742

ABSTRACT

With the common three-layer neural network architectures, the processing of a large number of signals requires an enormously large neural network. Such a network is very difficult to train and may not have sufficient speed for real-time applications. The computational complexity also prevents the use additional hidden layers, potentially leading to total inability of the network to capture essential complex patterns in the signals. Furthermore, the lack of internal structure in a large network makes it very difficult to discern characteristics of the knowledge acquired by the network, in order to evaluate its reliability and applicability.

We will investigate alternative neural-network structures that contain much fewer connections and are organized in a hierarchical fashion. Our hierarchical networks consist of a number of **loosely-coupled subnets**, arranged in layers; each subnet is intended to capture specific aspects of the input data. At the bottom layer, each subnet operates directly on a particular subset of the input variables. In the intermediate layers, each subnet receives its inputs from subnets of the previous layer and sends its outputs to subnets in the next higher layer. Each subnet is expected to model and summarize in its output the important characteristics of a particular set of **related input variables**.

In order to construct the subnets we start from the set of inputs and identify all its subsets which, based on our *a priori* knowledge of the structure and behavior of the system being modelled, consist of *related* inputs. We call these subsets ***input clusters***. In general, the clusters will overlap, and there will even be clusters that are fully contained in (i.e., are subsets of) other clusters. This defines the hierarchy that will be used in the construction of the network: Whenever a larger cluster is equal to the union of smaller clusters, the subnet that corresponds to the larger cluster will not receive its inputs directly from the input set, but rather from the outputs of other subnets that correspond to the smaller clusters. This should only take place if the rationale for establishing the cluster can be viewed as a combination of the rationales of the smaller cluster, i.e., if the set-subset relation of the clusters is not coincidental. The complexity of each subnet (i.e., its number of hidden and output nodes) can be adjusted based on the complexity of the relationship that led to the establishment of the cluster.

In a hierarchical neural network, we are **not hardwiring exact knowledge**, because the exact patterns and relationships among variables are still determined by the network during training. In essense, by organizing the variables into related sets and structuring the neural network as a **multiple hierarchy of subnets**, we are giving the network **hints** to look for patterns in the most promising directions, based on our prior knowledge about the structure and behavior of the system being modelled. The fact that hierarchical networks can contain **several layers** of subnets may allow these neural networks to capture patterns more complex than those captured by a three-layer network. However, due to their **small number of connections**, hierarchical neural-network structures can be trained faster and process a large number of signals in real-time. Their modular organization also makes them **easier to analyze**, because one can focus on the analysis of one subnet at a time, rather than attempt to decipher the whole network at once.

1. BACKGROUND

Most neural network applications investigated so far have involved only a small number of inputs and outputs. Use of neural networks for large problems with many inputs and complex patterns necessitates a large number of neurons in the input and hidden layers, and a number of weights that may be proportional to the *square* of the number of inputs.

As the size of the network increases, it becomes progressively harder to select a sufficiently large and diverse set of training examples, perform a sufficient number of iterations (since each iteration takes much longer), and avoid inappropriate weight patterns that hinder convergence.

The analysis and *explanation* of the predictions of the network also becomes more difficult, because a large number of weights must be dealt with all at once; the usual neural networks do not have any internal structure that can be related to the structure of the problem, i.e., one cannot *a priori* relate particular parts of the network to parts or attributes of the problem.

2. MONITORING COMPLEX PROCESS SYSTEMS

Our research involves the use of neural networks for monitoring and interpretation of measurements from complex dynamic systems, such as chemical processing plants. Tasks of interest include diagnosis of the current and future qualitative trends of the system, prediction of values of unmeasured parameters, selection of control actions or parameters, and diagnosis of faults. The processing of a large number of signals for this application would ordinarily require an enormous neural network. Such a network would be very difficult to train and might be too slow for real-time applications. It would also be very difficult to analyze the weights of the network, discern characteristics of the knowledge acquired, and evaluate the reliability and range of applicability of the network.

Consider the detection of faults in a simple processing system (a distillation column, shown in Figure 1). The process fluid contains three components, A, B, and C. The variables associated with each stream i are its flowrate F_i, its pressure P_i, its temperature T_i, and the concentrations of the three components A_i, B_i, and C_i. To avoid additional variables we assume that the control system's setpoints are fixed, and its controlled and manipulated variables are within the system examined.

For detection of a particular fault, the neural network needs only one output node, which produces 1 if there is a fault, and 0 if there is no fault. The inputs to the network are all the measured stream variables, for successive points within a time-window. If t is the present time, the series of n+1 time points [t, t-Δt, t-2Δt, t-3Δt, ..., t-nΔt] covers a window with total duration nΔt. The value of a variable V_i at time t-kΔt is denoted as $V_i(k)$.

A traditional three-layer network for fault-detection for this system would be quite big. If there are 10 points in the time-window (i.e., n=9), the network has 540 input nodes, whose values are quantitative (rather than boolean). The number of hidden nodes is generally comparable to the number of input nodes, to ensure that important patterns are recognized. If we use, for example, 180 hidden nodes, the number of connections between the input layer and the hidden layer is approximately 10^5. This number increases as the square of the number of points in the time-window or the number of measured variables. For a real application involving a section of an oil refinery or chemical plant, we can have more than 200 variables. Due to the multiple time scales in the dynamics of such a complex system, we may need to work with more than 100 time-points in the time-window. This brings the number of input nodes to more than 20,000, and the number of weights to 4×10^8. Even if the number of hidden nodes is reduced, this network would be quite cumbersome to use in real-time. It would also require an enormous number of training examples and passes through the database. For instance, if 10,000 passes through a set of 2,500 examples are needed, a total of 10^{16} updates of weights must take place.

In addition to being hard to train and use, the network is also hard to analyze, partly because of its large number of weights and nodes, but mainly because of its *lack of internal structure*. It is not possible to discern any manageable subnetworks, study and analyze those separately, and then build an explanation of the behavior of the whole network. One can also doubt the very ability of the network to learn the appropriate patterns; the complexity of the system suggests that several hidden layers may be necessary, but the computational cost of additional layers is prohibitive.

How can this enormous complexity be tamed? The key lies in reducing the number of weights by organizing a network as a hierarchy of subnets. The network can extend to several hidden layers and have a large number of hidden nodes, but its organization is structured in a way that limits the number of connections (and weights). The goal of this organization is to take

advantage of prior knowledge about what *kinds of patterns* are likely in the input data – instead of searching for patterns blindly.

3. HIERARCHICAL ARCHITECTURE

Consider the initial example with 36 variables and 10 time-points. The value of variable V for stream i at time t-kΔt is denoted as $V_i(k)$ and stands for a particular input of the overall neural network. The domains of V, i, and k are: V∈ {F, P, T, A, B, C}, i∈ {1, 2, ..., 9}, k∈ {0, 2, ..., 9}.

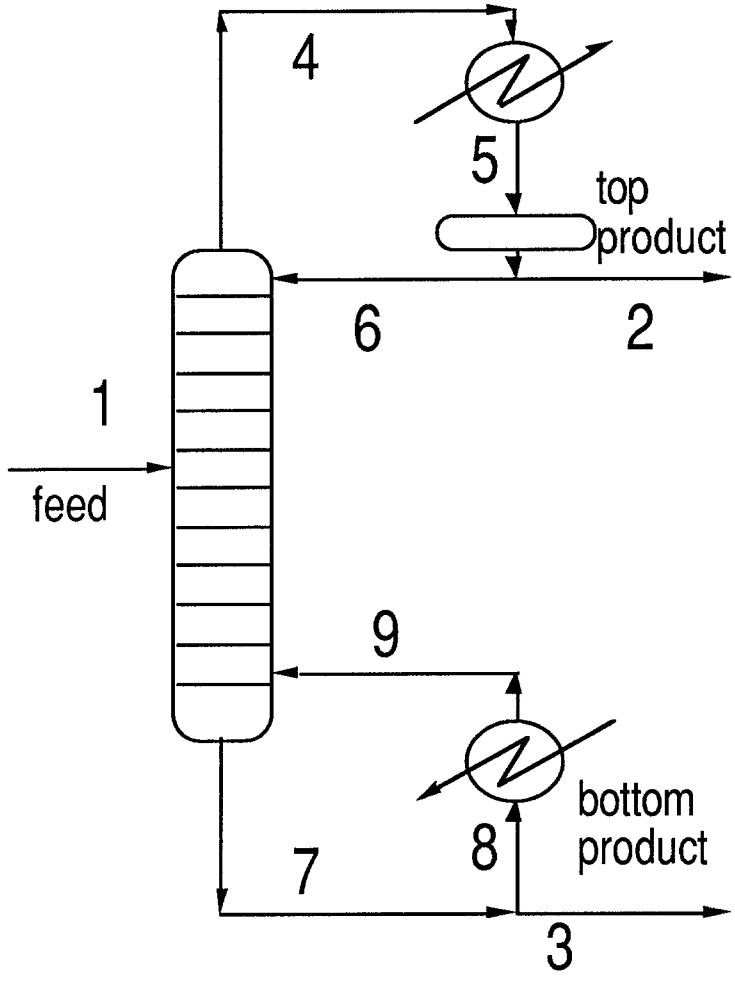


Figure 1: A distillation column, as an example of a complex dynamic system.

The proposed hierarchically structured neural network consists of a number of subnets, which are arranged in layers, much the same way that neurons are arranged in layers within a simple

neural network. At the bottom layer, the subnets receive directly the input variables, and at the very top a subnet provides the final output. In the intermediate layers, each subnet receives inputs from subnets in the previous layer and send its outputs to subnets of the next layer up. If we trace the inputs of a subnet all the way back to the original input variables, we can identify the ultimate set of input variables that could possibly affect the output of the subnet; the subnet is expected to model and summarize in its output the important characteristics of this ultimate set of input variables. No two subnets can have identical set of variables; however, each variable may participate in many subnets of each layer.

In order to construct the subnets we select all those clusters of inputs which, based on our *a priori* knowledge of the structure and behavior of the system being modelled, consist of *related* inputs. In general, the clusters will overlap, and there will even be clusters that are fully contained in (i.e., are subsets of) other clusters. This defines the hierarchy that will be used in the construction of the network: Whenever a larger cluster is equal to the union of smaller clusters, the subnet that corresponds to the larger cluster will not receive its inputs directly from the input set, but rather from the outputs of other subnets that correspond to the smaller clusters. This should only take place if the rationale for establishing the cluster can be viewed as a combination of the rationales of the smaller cluster, i.e., if the set-subset relation of the clusters is not coincidental. In general, a subnet of a particular cluster may receive some of its inputs from other subnets (in any of the previous layers) and some from the original input set. The smallest clusters receive all their inputs directly from the input set.

4. SELECTION OF CLUSTERS OF INPUTS

In this example, we will only form the most obvious clusters. We should, however, provide a brief justification for the the ways of clustering the inputs:

- **For any given process variable, its measurements at different time points** form an important cluster, because they uncover the dynamics of the variable. Taken together, the measurements can indicate the average value of the variable and its time-derivative, the presence of abrupt spikes or step-changes, the extent of fluctuation of the measurements, etc.

- **For any given process stream and any fixed time point, the measurements for all the variables** of the stream can be combined to yield derivative quantities, such as the enthalpy and heat-capacity of the stream, the molar

flowrate of each component, etc., as well as qualitative information such as the phase of the stream.

• **For any specific *kind* of variable** (such as flowrate or temperature) **and a fixed time point, the measurements of that kind of variable for all the streams** form a cluster. Measurements of the flowrates of the streams, taken together, are interrelated through a mass balance around the processing unit. Slightly subtler is the importance of relating measurements for each of the other kinds of variables. In empirical reasoning, we often use approximate or order-of-magnitude relationships among process parameters. For example, we may base our conclusions on the fact that the concentrations of A in two streams are approximately equal, or the fact that one concentration is much larger than the other. By clustering the concentrations of A in all the streams together, we allow the respective subnet to capture such approximate relationships, as well as higher level constructs that use such relationships.

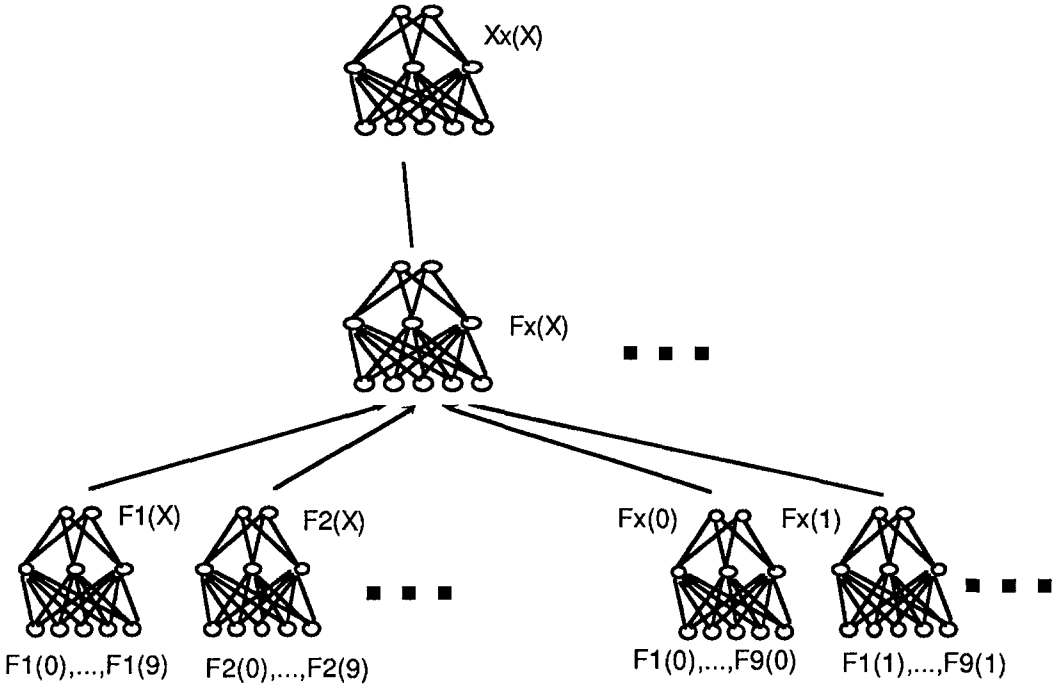


Figure 2: Part of the structure of a hierarchical neural network for monitoring the distillation column of Figure 1.

In the symbol $V_i(k)$, we can substitute X for one or more of V, i, and k to denote a subnet that ultimately receives and models the set of inputs derived by letting each X take all its allowable values. Consider two of the subnets shown in Figure 2:

- $F_x(0)$ denotes a subnet with direct inputs $F_1(0)$, $F_2(0)$, $F_3(0)$, $F_4(0)$, $F_5(0)$, $F_6(0)$, $F_7(0)$, $F_8(0)$, $F_9(0)$; similarly, $F_x(1)$ denotes a subnet with direct inputs $F_1(1)$, $F_2(1)$, $F_3(1)$, $F_4(1)$, $F_5(1)$, $F_6(1)$, $F_7(1)$, $F_8(1)$, $F_9(1)$; to generalize, for any k, $F_x(k)$ denotes a subnet with direct inputs $F_1(k)$, $F_2(k)$, $F_3(k)$, $F_4(k)$, $F_5(k)$, $F_6(k)$, $F_7(k)$, $F_8(k)$, $F_9(k)$. Each of these subnets belongs to the input layer of subnets and describes the state of the flowrates of the process, for a particular time-point.

- $F_1(X)$ denotes a subnet with direct inputs $F_1(0)$, $F_1(1)$, $F_1(2)$, $F_1(3)$, $F_1(4)$, $F_1(5)$, $F_1(6)$, $F_1(7)$, $F_1(8)$, $F_1(9)$; generalizing, $F_i(X)$ denotes a subnet with direct inputs $F_i(0)$, $F_i(i)$, $F_i(2)$, $F_i(3)$, $F_i(4)$, $F_i(5)$, $F_i(6)$, $F_i(7)$, $F_i(8)$, $F_i(9)$. Each of these subnets belongs to the input layer of subnets and describes the state and dynamics of a particular process variable (F_i) over the whole time-window.

The outputs of these subnets are fed to higher-level subnets, which correspond to larger clusters of variables. For example, we denote as $F_x(X)$ a subnet which corresponds to the cluster of all flowrate measurements (for all streams and all time points). This subnet belongs to the intermediate layer of subnets. Its inputs are the outputs of all subnets of the form $F_i(X)$ and $F_x(k)$, which were described above. The role of $F_x(X)$ is to receive information about the dynamics of individual flowrates – from $F_i(X)$ – and information about the state of all flowrates together at particular time points – from $F_x(k)$ – and produce an overall picture of the behavior of all flowrates over all the time points. Thus, the output of $F_x(X)$ ultimately depends on the values of all the flowrates over the time window. The hierarchy of subnets is a *multiple* hierarchy, because each subnet (and each input variable) can be used by several subnets at the next layer.

Beyond the specific examples of clusters mentioned above, other process variables are used by subnets in the same way. There is a subnets describing the state of each particular stream; for example, $X_1(0)$ takes as direct inputs the parameters of stream 1 at time t (i.e., t-0Δt). Its output is used by subnet $X_x(0)$ which describes the state of the whole process at time t; it is also used by subnet $X_1(X)$ which describes the state and dynamics of stream 1 over the whole

time-window, using all its parameters. The output of the last subnet, $X_1(X)$, ultimately depends on all the values of the parameters of stream 1.

For this particular system, we structure the subnets in three layers, according to the number of X's in their symbol. In the first layer, we have 186 subnets that have only one X in their symbol and receive their inputs directly from the input variables. In the second layer, we have 25 subnets with two X's in their symbols; each of those receives its inputs from several first-layer subnets. In the third layer, we have only one subnet, namely $X_x(x)$, which receives inputs from all second-layer subnets. The internal structure of each subnet can be as intricate or simple as desired. The subnet's complexity (i.e., its number of hidden and output nodes) can be adjusted based on the complexity of the relationship that led to the establishment of the cluster.

In the above example, we only formed sets of inputs that are comprised of either measurements of a given process variable (at different time points), measurements that refer to a given process stream, or measurements of the same kind of variable in different streams. In general, the formation of good sets of related input variables depends on the depth of prior knowledge about the system that is modelled. For example, if there is a small set of variables that participate in a constraint (such as a mass or energy balance) or relate to a particular physical phenomenon (such as diffusion of a component) then it is appropriate to use a subnet at the bottom layer to capture characteristics of the phenomenon. If there is a set of constraints which are used together, or a set of related phenomena, then a subnet at the next layer, receiving inputs from the subnets of the individual constraints is appropriate.

Subnets should also be used to model distinct **loosely-interacting subsystems.** Most real-life complex systems can be recursively decomposed into loosely-coupled subsystems. each subsystem can be modelled by a subnet that By constructing a corresponding hierarchy of subnets,

5. ADVANTAGES OF HIERARCHICAL NETWORKS

If we assume that each subnet has three layers of neurons, with two nodes in the output layer, and the number hidden nodes equal to one half of the number of the input nodes, the total number of connections in the whole hierararchical network is less than 7,000 (or at least 14 times fewer than the connections of an unstructured three-layer network). As a simple generalization, consider a system with n streams, n process variables per stream, and n time-

points to examine, there are n^3 inputs to be processed. The common three-layer network architecture would require $O(n^6)$ connections, but the proposed architecture requires only $O(n^4)$ connections. This architecture actually has a total of 7 layers of neurons, with O(n) neurons per layer, but it does not require many connections, because it is organized into subnets in a hierarchical fashion. The network should be easier to train and use in real-time because of the small number of connections and the logical structure that is built into the network's architecture. The required number of examples should also be significantly smaller.

Another important advantage of the hierarchical organization is the way it can be analyzed, to decipher the knowledge captured by the network. Each subnet as its specific role to summarize a specified set of inputs into a small number of outputs. By observing the output of the subnet for special test-inputs, we can draw conclusions about what attributes and patterns in the input are recognized by the subnet. Consider, for example, the subnet $F_1(X)$ which has the direct inputs $F_1(0)$, $F_1(1)$, $F_1(2)$, $F_1(3)$, $F_1(4)$, $F_1(5)$, $F_1(6)$, $F_1(7)$, $F_1(8)$, $F_1(9)$ and describes the behavior of F_1 over the whole time-window. Once the network converges, we can try several test-inputs for the subnet, such as: a flat profile, i.e., $F_1(k)=a$ for all k; a step change, i.e., $F_1(k)=a$ for $k\leq m$, $F_1(k)=b$ for $k>m$; a pulse, i.e., $F_1(k)=a$ for $k=m$, $F_1(k)=b$ for $k\neq m$;and a linear profile, i.e., $F_1(k)=a+bk$. For each kind of profile, different values of the parameters (a, b, m) can be tested. The dependence of the output values on the nature and parameters of the profile allows one to determine how the subnet is coding its inputs, i.e., what important attributes of the input the subnet has learned. For example, it may be found that a subnet's two outputs provide roughly: the average value and standard deviation of the inputs; the average value and the average slope; or the presence of abrupt changes in the inputs.

Having analyzed the behavior of the subnets in the bottom layer (which receive their inputs directly from the measured parameters), one can proceed to the next layer. Here, each subnet receives its inputs from subnets of the previous layer, which have already been deciphered. Thus, test inputs can be generated and the behavior of each subnet can be analyzed. Layer by layer, one can accomplish an analysis of the behavior and knowledge of the whole network. There is no harm (other than the computational cost) in introducing subnets of uncertain value. It is in fact expected that proper analysis of the subnets, as described above, will show that for any given example only some subnets play an important role.

6. RESEARCH PLAN

Our next step will be the computer implementation of the outlined techniques (using the computing facilities of SPL), followed by proof-of-concept computational study, to validate the capabilities of the proposed hierarchical multi-level nets and the constraint propagation approach. The study will employ simulation results for a relatively simple system (such as a single processing unit) to detect faults or predict trends. Our long term plan includes further formalization of these methods in a mathematical framework and theoretical investigation of their properties (convergence, computational complexity, etc.). The theoretical investigation will facilitate the application of the techniques to a variety of systems. As we advance in determining the theoretical robustness of our approach, we also intend to use experimental data from a (sufficiently complex) industrial system to validate the ability of the techniques to handle the non-idealities and uncertainties of real systems.

7. REFERENCES

Cohen, P. R. and E. A. Feigenbaum. The Handbook of Artificial Intelligence, Kaufman, Los Altos (1982).

Feldman, J. A. and D. H. Ballard. Connectionist Models and Their Properties, Cognitive Science 6, 205-254 (1982).

Gorman R., and Sejnowski T., Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets, Neural Networks, Vol. 1, 75-89 (1988)

Hinton G., and Anderson J. (Eds.), Parallel Models of Associative Memory, Erlbaum (1981)

Hopfield, J. J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings of the National Academy of Science 79, 2554-2558 (1982).

Hoskins, J. C. and D. M. Himmelblau. Artificial Neural Network Models of Knowledge Representation in Chemical Engineering. Computers & Chemical Engineering 12:9/10, 881-890 (1988a).

Hoskins, J. C. and D. M. Himmelblau. Automatic Chemical Process Control Using Reinforcement Learning in Artificial Neural Networks. Presented at The First Annual International Neural Network Society Meeting, Boston, MA (1988c).

Kohonen T. Self-Organization and Associative Memory, Second Edition, Springer- Verlag, Berlin (1987)

Lippmann R., An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, 4-22 (April 1987)

Minsky, M. and S. Papaert. Perceptrons: An introduction to computational geometry. MIT Press, Cambridge (1969).

Nilsson, N. J. Learning Machines. McGraw-Hill, New York (1965).

Rumelhart, D. E. and J. L. McClelland. (Eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. The MIT Press, Cambridge (1986).

Sejnowski T., and Rosenberg C., NETtalk: A Parallel Network That learns to Read Aloud, Johns Hopkins Univ. Technical Report HU/EECS-86/01 (1986)

Tou, J. T. and R. C. Gonzalez. Pattern Recognition Principles. Addison-Wesley, Reading (1977).

Widrow B., and Hoff M., Adaptive Switching Circuits, IRE WESCON Conv. Record, Part 4, 96-104 (1960)

Widrow, B. and S. D. Sterns. Adaptive Signal Processing, Prentice-Hall, Englewood Cliffs (1985).