# The Layered Feed-Forward Neural Networks and Its Rule Extraction

Ray Tsaih and Chih-Chung Lin

Department of Management Information Systems, National Chengchi University, Taipei, Taiwan
{tsaih,m92014}@mis.nccu.edu.tw

**Abstract.** A mathematical study of the layered feed-forward neural networks is proposed here for identifying the rules suggested in the network. The mathematical study, not a data analysis, is proposed for identifying the premise association with each rule. It, hopefully, can be used further to deal with the predicament of ANN being a black box.

## 1 The Predicament of Being a Black Box

Layered feed-forward neural networks have been widely used in many fields. When the layered feed-forward neural network is used as a modeling tool, it is interesting to check if it can display some useful information. It is necessary to have a deeper analysis of the network structure, an analysis that is rather complicated mathematically. However, as Yoon, Guimaraes, and Swales in [1] argued that, after building the layered feed-forward neural networks, reading or understanding the knowledge in layered feed-forward neural networks was difficult because the knowledge was distributed over the entire network.

There are some recent studies related with extracting rules from the trained Artificial Neural Networks (ANN). For instance, in [2] and [3] try to extract rules from a trained ANN for regression problems. To identify the premise of a single rule, however, in [2] and [3] implement a data analysis on the training data set or the generated data set. No matter the data set for extracting rules is the trained one or the generated one, the amount of data instances is still finite, and the premise of a resulted rule covers merely discrete points, not an area.

Here we present a mathematical study of the layered feed-forward neural networks for identifying the rules suggested in the network. The mathematical analysis, not a data analysis, is proposed for identifying the premise associated with each rule. This paper is organized as follows. Section 2 gives the detail of the mathematical study and Section 3 offers some conclusions and future work.

## 2 The Study for Identifying the Rules Suggested in the Network

Assume the layered feed-forward neural network $f$ is arranged as the one defined from equations (1) and (2) below, where $tanh(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Namely, given the $c^{th}$ observation ${}_c\mathbf{x}$, the corresponding value of the $i^{th}$ hidden node ${}_ca_i$ equals $tanh({}_2w_{i0} + \sum_{j=1}^{m} {}_2w_{ij}\, {}_cx_j)$ and the corresponding value of $f({}_c\mathbf{x})$ equals ${}_3w_0 + \sum_{i=1}^{p} {}_3w_i\, {}_ca_i$. In equations (1) and (2), $m$ is the number of explanatory variables $x_j$'s, $p$ is the number of adopted hidden nodes, ${}_2w_{i0}$ is the bias value of the $i^{\text{th}}$ hidden node $a_i$, ${}_2w_{ij}$ is the weight between the $j^{\text{th}}$ explanatory variable $x_j$ and the $i^{\text{th}}$ hidden node $a_i$, ${}_3w_0$ is the bias value, and ${}_3w_i$ is the weight between the $i^{\text{th}}$ hidden node $a_i$ and the output node.

$$\mathrm{a_i}(\mathbf{x}) \equiv \tanh({}_2\mathrm{w}_{i0} + \sum_{j=1}^{m} {}_2\mathrm{w}_{ij}\, \mathrm{x}_j)\ , \tag{1}$$

$$f(\mathbf{x}) \equiv {}_3w_0 + \sum_{i=1}^{p} {}_3w_i\, a_i(\mathbf{x}) = {}_3w_0 + \sum_{i=1}^{p} {}_3w_i\, tanh({}_2w_{i0} + \sum_{j=1}^{m} {}_2w_{ij}\, x_j)\ . \tag{2}$$

In this article, character in bold represents a column vector, a matrix or a set, and the superscript T indicates the transposition: ${}_2\mathbf{w}_i^{\mathrm{T}} \equiv ({}_2w_{i1}, \ldots, {}_2w_{im})$, ${}_3\mathbf{w}^{\mathrm{T}} \equiv ({}_3w_1, \ldots, {}_3w_p)$, ${}_2\mathbf{w}^{\mathrm{T}} \equiv ({}_2\mathbf{w}_1^{\mathrm{T}}, {}_2\mathbf{w}_2^{\mathrm{T}}, \ldots, {}_2\mathbf{w}_p^{\mathrm{T}})$, and $\mathbf{w}^{\mathrm{T}} \equiv ({}_2\mathbf{w}^{\mathrm{T}}, {}_3\mathbf{w}^{\mathrm{T}})$. Here we assume $m > 1$. We also assume that ${}_3\mathbf{w}$ and ${}_2\mathbf{w}$ are non-zero vectors. Thus ${}_2\mathbf{W} \equiv ({}_2\mathbf{w}_1, {}_2\mathbf{w}_2, \ldots, {}_2\mathbf{w}_p)^{\mathrm{T}}$ is a non-zero matrix.

The range of f is at most $({}_3\mathrm{w}_0 - \sum_{i=1}^{p} |{}_3\mathrm{w}_i|, {}_3\mathrm{w}_0 + \sum_{i=1}^{p} |{}_3\mathrm{w}_i|)$ instead of R. It is because that $-1 < \mathrm{a_i} < 1$, thus $-\sum_{i=1}^{p} |{}_3\mathrm{w}_i| < \sum_{i=1}^{p} {}_3\mathrm{w}_i\, \mathrm{a}_i < \sum_{i=1}^{p} |{}_3\mathrm{w}_i|$ and ${}_3\mathrm{w}_0 - \sum_{i=1}^{p} |{}_3\mathrm{w}_i| < \mathrm{y} < {}_3\mathrm{w}_0 + \sum_{i=1}^{p} |{}_3\mathrm{w}_i|$. y is vague if it cannot be carried out with the network system; otherwise, y is non-vague. If $\mathrm{y} \in (-\infty, {}_3\mathrm{w}_0 - \sum_{i=1}^{p} |{}_3\mathrm{w}_i|) \cup ({}_3\mathrm{w}_0 + \sum_{i=1}^{p} |{}_3\mathrm{w}_i|, \infty)$, y is surely vague.

As Rumelhart and his collagues in [4] proposed the interesting Back-Propagation learning algorithm for training layered feed-forward neural networks, the information $\mathbf{x}$ coming to the input nodes is re-coded into $\mathbf{a} \equiv (a_1, a_2, \ldots, a_p)^{\mathrm{T}} \in (-1, 1)^p$ and the output $y$ is generated by $\mathbf{a}$ rather than the original pattern $\mathbf{x}$. In other words, $f$ can be viewed as the composite $g \circ \mathbf{h}$ where $\mathbf{h}$ is defined by $(\mathbf{h}(\mathbf{x}))_i$, the $i^{\text{th}}$ component of $\mathbf{h}(\mathbf{x})$, $\equiv tanh({}_2w_{i0} + \sum_{j=1}^{m} {}_2w_{ij}\, x_j)$ and $a_i = (\mathbf{h}(\mathbf{x}))_i$ for every $i \in \mathbf{P} \equiv \{1, 2, \ldots, p\}$, and $g$ is defined by $g(\mathbf{a}) \equiv {}_3w_0 + \sum_{i=1}^{p} {}_3w_i\, a_i$. The hidden-layer set is $(-1, 1)^p$ because the $tanh$ activation function is used here.

For any non-vague $y$, let $\mathbf{X}(y) \equiv f^{-1}(y)$, the set of all elements of the input space whose images under $f$ are $y$. Any input stimulus $\mathbf{x}$ in $\mathbf{X}(y)$ will result in an output value $y$. Thus, the following rule is suggested from the network:

**Rule:** If the input $\mathbf{x}$ is in the region of $\mathbf{X}(y)$, then the output value of the network is $y$.

$\mathbf{X}(y)$ can be viewed as the composite $\mathbf{h}^{-1} \circ \mathbf{g}^{-1}(y)$, where $\mathbf{g}^{-1}(y)$ is the set of all elements of the hidden-layer set whose images under $g$ are $y$, and $\mathbf{h}^{-1}(\mathbf{a})$ is the set of all elements of the input space whose images under $\mathbf{h}$ are $\mathbf{a}$.

$\mathbf{h}^{-1}(\mathbf{a})$ equals $\{\mathbf{x}/ \sum_{j=1}^{m} {}_2w_{ij}\, x_j = tanh^{-1}(a_i) - {}_2w_{i0}$ for all $i \in \mathbf{P}\}$ where $tanh^{-1}(x) \equiv 0.5\, ln(\frac{1+x}{1-x})$ is the inverse function of $tanh$. When ${}_2\mathbf{w}_i$ is non-zero, there are parallel activation level hyperplanes, $\{\mathbf{x}/ \sum_{j=1}^{m} {}_2w_{ij}\, x_j = tanh^{-1}(a) - {}_2w_{i0}\}$ for all $a \in (-1, 1)$, in the input space. As stated in [5], these activation level hyperplanes make a scalar activation field in the input space. Through each point of the input space, there passes merely an activation level hyperplane that determines the associated activation value $a_i$ of that point. All points on the level hyperplane $\{\mathbf{x}/ \sum_{j=1}^{m} {}_2w_{ij}\, x_j = tanh^{-1}(a_i) - {}_2w_{i0}\}$ have the same activation value $a_i$ in the $i^{th}$ hidden node. $p$ hidden nodes set up $p$ activation fields in the input space, but these activation fields do not interfere with each other. Thus, $\mathbf{h}^{-1}(\mathbf{a})$ equals $\bigcap_{i=1}^{p} \{\mathbf{x}/ \sum_{j=1}^{m} {}_2w_{ij}\, x_j = tanh^{-1}(a_i) - {}_2w_{i0}\}$, which also equals $\{\mathbf{x}/ \sum_{j=1}^{m} {}_2w_{ij}\, x_j = tanh^{-1}(a_i) - {}_2w_{i0}$ for all $i \in \mathbf{P}\}$.

$\{\mathbf{x}/ \sum_{j=1}^{m} {}_2w_{ij}\, x_j = tanh^{-1}(a_i) - {}_2w_{i0}$ for all $i \in \mathbf{P}\}$ can also be represented as $\{\mathbf{x}/\, {}_2\mathbf{W}\, \mathbf{x} = \boldsymbol{\omega}(\mathbf{a})\}$. The function $\boldsymbol{\omega} : (-1, 1)^p \rightarrow R^p$ is defined by $\boldsymbol{\omega}(\mathbf{a}) \equiv (\omega_1(a_1), \omega_2(a_2), \ldots, \omega_p(a_p))^T$ with $\omega_i(a_i) \equiv tanh^{-1}(a_i) - {}_2w_{i0}$ for every $i$. Given the vector $\mathbf{a}$, $\boldsymbol{\omega}(\mathbf{a})$ is determined. Accordingly, given the vector $\mathbf{a}$, the system ${}_2\mathbf{W}\ \mathbf{x} = \boldsymbol{\omega}(\mathbf{a})$ is a system of $p$ linear equations in $m$ unknowns. If $rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W}) + 1$, where $rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a}))$ is the rank of the augmented matrix $({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a}))$, the system ${}_2\mathbf{W}\ \mathbf{x} = \boldsymbol{\omega}(\mathbf{a})$ has no solution. (cf. [6], p. 108) In other words, if $rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W}) + 1$, $\mathbf{h}^{-1}(\mathbf{a})$ is an empty set and the activation value $\mathbf{a}$ cannot be carried out with the neural network.

The point $\mathbf{a}$ in the hidden-layer set is vague if the activation values of $\mathbf{a}$ cannot be carried out with the neural network; otherwise, $\mathbf{a}$ is non-vague. Namely, the point $\mathbf{a}$ in the hidden-layer set is vague if $\mathbf{a}$ is mapped from nowhere in the input space. Lemma 1 gives the situation of the point $\mathbf{a}$ being vague.

**Lemma 1:** If $rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W}) + 1$, the point $\mathbf{a}$ is vague.

The fact that $rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W})$ implies $\boldsymbol{\omega}(\mathbf{a})$ is in the linear hull spanned by column vectors of ${}_2\mathbf{W}$ (cf. [6], p. 92). Therefore, as stated in Lemma 2 and Lemma 3, $\{\mathbf{a}/\ rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W})\}$ is either the whole hidden-layer set, or a manifold[1] or a linear hull bounded by $(-1, 1)^p$. Moreover, whether there are vague points in the hidden-layer set can be determined by comparing values of $rank({}_2\mathbf{W})$ and $p$. In other words, the range of $\mathbf{h}$ is $\{\mathbf{a}/\ rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W})\}$ instead of $(-1, 1)^p$.

**Lemma 2:** If $rank({}_2\mathbf{W}) = p$, $\{\mathbf{a}/\ rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W})\}$ equals $(-1, 1)^p$ and there are no vague points in the hidden-layer set.

**Lemma 3:** If $rank({}_2\mathbf{W}) = r$ and $r < p$, $\{\mathbf{a}/\ rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W})\}$ is a $r$-manifold or a linear hull of dimension $r$ bounded by $(-1, 1)^p$ and there are vague points in the hidden-layer set.

Lemma 4 states that $\mathbf{h}^{-1}(\mathbf{a})$ is a single point in the input space when $\mathbf{a}$ is a non-vague point and $rank({}_2\mathbf{W}) = m$; Lemma 5 states that $\mathbf{h}^{-1}(\mathbf{a})$ is an affine space of dimension $m - r$ in the input space when $\mathbf{a}$ is a non-vague point, $rank({}_2\mathbf{W}) = r$ and $r < m$. In other words, $\mathbf{h}^{-1}(\mathbf{a})$ is an affine space of dimension $m - rank({}_2\mathbf{W})$ when $\mathbf{h}^{-1}(\mathbf{a})$ is non-empty.

**Lemma 4:** Each non-vague point $\mathbf{a}$ is mapped from a single point $\mathbf{x}$ in the input space provided that $rank({}_2\mathbf{W}) = m$.

**Lemma 5:** Each non-vague point $\mathbf{a}$ is mapped from any point $\mathbf{x}$ in an affine space of dimension $m - r$ in the input space provided that $rank({}_2\mathbf{W}) = r$ and $r < m$.

Conversely, $\mathbf{g}^{-1}(y)$ equals $\{\mathbf{a}/ \sum_{i=1}^{p} {}_3w_i\, a_i = y - {}_3w_0,\ \mathbf{a} \in (-1, 1)^p\}$, and $\mathbf{g}^{-1}(y)$ is a hyperplane bounded by $(-1, 1)^p$. When ${}_3\mathbf{w}$ is non-zero, there are parallel hyperplanes, $\mathbf{g}^{-1}(y)$s for all $y \in ({}_3w_0 - \sum_{i=1}^{p} |{}_3w_i|,\ {}_3w_0 + \sum_{i=1}^{p} |{}_3w_i|)$, in $(-1, 1)^p$. These hyperplanes make a scalar field (activation field) in the hidden-layer set. Through each point of the hidden-layer set, there passes merely one hyperplane that determines the associated activation value $y$ of that point. All points on the $\mathbf{g}^{-1}(y)$ hyperplane have the same value $y$.

$\mathbf{g}^{-1}(y) \equiv \mathbf{A}_v(y) \cup \mathbf{A}_{nv}(y)$ where $\mathbf{A}_v(y)$ and $\mathbf{A}_{nv}(y)$ are the sets of vague and non-vague points in $\mathbf{g}^{-1}(y)$, respectively. From Lemma 1, $\mathbf{A}_{nv}(y) \equiv \{\mathbf{a}/\ \mathbf{a} \in \mathbf{g}^{-1}(y), rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W})\}$. $\mathbf{A}_{nv}(y)$ can be viewed as $\{\mathbf{a}/\ rank({}_2\mathbf{W} : \boldsymbol{\omega}(\mathbf{a})) = rank({}_2\mathbf{W})\} \cap \mathbf{g}^{-1}(y)$. From Lemma 6 and Lemma 7, $\mathbf{A}_{nv}(y)$ is either a hyperplane bounded by $(-1, 1)^p$ or an intersection of a hyperplane and a $r$-manifold or a linear hull of dimension $r$ in $(-1, 1)^p$.

**Lemma 6:** $\mathbf{A}_{nv}(y)$ equals $\mathbf{g}^{-1}(y)$ provided that $rank({}_2\mathbf{W}) = p$.

**Lemma 7:** $\mathbf{A}_{nv}(y)$ is an intersection of a hyperplane and a $r$-manifold or a linear hull of dimension $r$ in $(-1, 1)^p$ provided that $rank({}_2\mathbf{W}) = r$ and $r < p$.

Let $\mathbf{h}^{-1}(\mathbf{g}^{-1}(y)) \equiv \{\mathbf{h}^{-1}(\mathbf{a})/\ \mathbf{a} \in \mathbf{g}^{-1}(y)\}$ and $\mathbf{h}^{-1}(\mathbf{A}_{nv}(y)) \equiv \{\mathbf{h}^{-1}(\mathbf{a})/\ \mathbf{a} \in \mathbf{A}_{nv}(y)\} = \{\mathbf{x}/\ {}_2\mathbf{W}\,\mathbf{x} = \boldsymbol{\omega}(\mathbf{a})$ with all $\mathbf{a} \in \mathbf{A}_{nv}(y)\}$. $\mathbf{h}^{-1}(\mathbf{g}^{-1}(y))$ equals $\mathbf{h}^{-1}(\mathbf{A}_{nv}(y))$ because that $\mathbf{g}^{-1}(y) \equiv \mathbf{A}_v(y) \cup \mathbf{A}_{nv}(y)$ and $\mathbf{A}_v(y)$ is mapped from nowhere in the input space. Thus $\mathbf{X}(y)$ equals $\{\mathbf{x}/\ {}_2\mathbf{W}\,\mathbf{x} = \boldsymbol{\omega}(\mathbf{a})$ with all $\mathbf{a} \in \mathbf{A}_{nv}(y)\}$. Theorems 1 and 2 show that, when $rank({}_2\mathbf{W}) = p$, $\mathbf{X}(y)$ is either a hyperplane or a manifold in the input space and the dimension of $\mathbf{X}(y)$ is $m$-1.

**Theorem 1:** $\mathbf{X}(y)$ is a hyperplane in the input space provided that $m > p$, $p = 1$ and $rank({}_2\mathbf{W}) = 1$.

[1] The definition of manifold please refer to [7].

**Theorem 2:** $\mathbf{X}(y)$ is a $(m\text{-}1)$-manifold in the input space provided that $m \geq p$, $p > 1$ and $rank({}_2\mathbf{W}) = p$.

For any non-vague $y_1$ and $y_2$ with ${}_3w_0 - \sum_{i=1}^{p} |{}_3w_i| < y_2 \leq y_1 < {}_3w_0 + \sum_{i=1}^{p} |{}_3w_i|$, the prediction $y \leq y_2$ is activated by any $\mathbf{a}$ in $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{A}_{nv}(y)$ or any input $\mathbf{x}$ in $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{X}(y)$. In other words, there is a rule:

**Rule:** If the input $\mathbf{x}$ is in the region of $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{X}(y)$, then the output value of the network is less than or equals $y_2$. Similarly, the prediction $y_2 \leq y \leq y_1$ is activated by any $\mathbf{a}$ in $\bigcup_{y=y_2}^{y_1} \mathbf{A}_{nv}(y)$ or any input $\mathbf{x}$ in $\bigcup_{y=y_2}^{y_1} \mathbf{X}(y)$; the prediction $y \geq y_1$ is activated by any $\mathbf{a}$ in $\bigcup_{y=y_1}^{{}_3w_0+\sum_{i=1}^{p}|{}_3w_i|} \mathbf{A}_{nv}(y)$ or any input $\mathbf{x}$ in $\bigcup_{y=y_1}^{{}_3w_0+\sum_{i=1}^{p}|{}_3w_i|} \mathbf{X}(y)$.

Theorem 3 shows that, when $rank({}_2\mathbf{W}) = p$, the set $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{A}_{nv}(y)$ is an union of adjacent hyperplanes bounded by $(-1, 1)^p$. Combined with Theorem 1, the set $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{X}(y)$ is an union of adjacent hyperplanes in the input space when $m > p$, $p = 1$ and $rank({}_2\mathbf{W}) = 1$; combined with Theorem 2, the set $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{X}(y)$ is an union of adjacent $(m\text{-}1)$-manifolds in the input space when $m \geq p$, $p > 1$ and $rank({}_2\mathbf{W}) = p$.

**Theorem 3:** For any non-vague $y_1$ and $y_2$ with ${}_3w_0 - \sum_{i=1}^{p} |{}_3w_i| < y_1 \leq y_2 < {}_3w_0 + \sum_{i=1}^{p} |{}_3w_i|$, sets of $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{A}_{nv}(y)$, $\bigcup_{y=y_2}^{y_1} \mathbf{A}_{nv}(y)$ and $\bigcup_{y=y_1}^{{}_3w_0+\sum_{i=1}^{p}|{}_3w_i|} \mathbf{A}_{nv}(y)$ are convex polytopes provided that $rank({}_2\mathbf{W}) = p$.

## 3 Summary

In conclusion, for any non-vague $y_1$ and $y_2$ with ${}_3w_0 - \sum_{i=1}^{p} /{}_3w_i/ < y_1 \leq y_2 < {}_3w_0 + \sum_{i=1}^{p} /{}_3w_i/$, the following four rules are suggested from the trained network:

**Rule 1:** If the input **x** is in the region of $\mathbf{X}(y)$, then the output value of the network is $y$.

**Rule 2:** If the input **x** is in the region of $\bigcup_{y={}_3w_0-\sum_{i=1}^{p}|{}_3w_i|}^{y_2} \mathbf{X}(y)$, then the output value of the network is less than or equals $y_2$.

**Rule 3:** If the input **x** is in the region of $\bigcup_{y=y_2}^{y_1} \mathbf{X}(y)$, then the output value of the network is within $[y_2, y_1]$.

**Rule 4:** If the input **x** is in the region of $\bigcup_{y=y_1}^{{}_3w_0+\sum_{i=1}^{p}|{}_3w_i|} \mathbf{X}(y)$, then the output value of the network is greater than or equals $y_1$, where $\mathbf{X}(y) = \{\mathbf{x}/\ {}_2\mathbf{W}\ \mathbf{x} = \boldsymbol{\omega}(\mathbf{a})$ with all $\mathbf{a} \in \mathbf{A}_{nv}(y)\}$.

## References

1. Yoon, Y., Guimaraes, T., Swales, G.: Integration Artificial Neural Networks with Rule-Based Expert System in Decision Support Systems. Vol. 11. (1994) 497-507
2. Setiono, R., Leow, W. K., Zurada, J. M.: Extraction of Rules from Artificial Neural Networks for Nonlinear Regression. IEEE Transactions on Neural Networks, Vol. 13. (2002) 564-577
3. Saito, K., Nakano, R.: Extracting Regression Rules from Neural Networks in Neural Networks. Vol. 15. (2002) 1279-1288
4. Rumelhart, D. E., Hinton, G. E., Williams, R.: Learning Internal Representation by Error Propagation in Parallel Distributed Processing. Vol. 1, Cambridge, MA: MIT Press (1986) 318-362
5. Tsaih, R.: An Explanation of Reasoning Neural Networks in Mathematical and Computer Modelling. Vol. 28. (1998) 37-44
6. Murty, K.: Linear Programming. John Wiley & Sons, New York (1983)
7. Munkres, J.: Topology: A First Course. Prentice-Hall, Englewood Cliffs, New Jersey (1975)