# SIMULATING DATA ENVELOPMENT ANALYSIS USING NEURAL NETWORKS

## *A new paradigm of efficiency measurement*

Luiz Biondi Neto, Pedro Henrique Gouvêa Coelho
*Electronics and Telecommunications Department, State University of Rio de Janeiro, Rua São Francisco Xavier, 524, Bl. A, Sala 5036, Maracanã, 20550-013, Rio de Janeiro, RJ, Brazil*
*Email: lbiondi@uerj.br, phcoelho@uerj.br*

João Carlos C. B. Soares de Mello
*Production Engineering Department, Fluminense Federal University, Rua Passo da Pátria, 156, São Domingos, 24240-240, Niteroi, RJ, Brazil*
*Email: gmajcsm@vm.uff.br*

Eliane Gonçalves Gomes
*Satellite Monitoring Research Center, Brazilian Agricultural Research Corporation, Av. Dr. Júlio Soares de Arruda, 803, Parque São Quirino, 13088-300, Campinas, SP, Brazil*
*Email: eliane@cnpm.embrapa.br*

Keywords: Data Envelopment Analysis, Neural Networks

Abstract: This article studies the creation of efficiency measurement structures of Decision-Making Units (DMUs) by using high-speed optimisation modules, inspired in the idea of an unconventional Artificial Neural Network (ANN) and numerical methods. In addition, the Linear Programming Problem (LPP) inherent in the Data Envelopment Analysis (DEA) methodology is transformed into an optimisation problem without constraints, by using a pseudo-cost function, including a penalty term, causing high cost every time one of the constraints is violated. The LPP is converted into a differential equations system. A non-standard ANN implements a numerical solution based on the gradient method.

## 1 INTRODUCTION

Data Envelopment Analysis (DEA) is a mathematical technique used to analyse the decision-making units (DMUs) performance. It also allows the evaluation of the relative operational efficiency of the (DMUs), comparing each DMU relatively to all others comprising the investigated DMUs group (Charnes et al., 1996).

The DEA technique compares the DMU efficiencies by their abilities in transforming inputs in outputs, measuring the reached output relation in terms of the input resources.

In the end of the analysis, the DEA technique is able to tell which units are relatively efficient and which ones are relatively inefficient.

The above-mentioned technique uses Linear Programming (LP) methods to solve a group of interrelated linear programming problems LPPs, as many as the number of DMUs, with the purpose of determining the relative efficiency of each DMU. Optimisation modules called Neuro-LPs are used in the proposed neural model coined as Neuro-DEA, inspired by artificial neural network structures (Biondi, 2001).

DEA models can either be input oriented or output oriented. DEA analysis initialisation involves the choice of outputs and the model orientation. The orientation to inputs indicates one wants to reduce the inputs, keeping the outputs unaffected. On the other hand, the orientation to outputs indicates that one wants to increase the outputs without affecting

the inputs. The most important models are the following:

**CCR** – Model presented by Charnes et al. (1978) that builds a non parametrical surface, piecewise linear, over the data and determines the investigated DMUs technical efficiency over this surface. It was conceived as an input oriented model and it works with constant return of scale (CRS), which means that each variation in the inputs produces a proportional variation in the outputs. The problem consists in determining the $u_j$ and $v_i$ weight values to maximise the linear combination of the outputs divided by the linear combination of the inputs.

The process is repeated for each of the *n* DMUs, yielding the relative value of each DMU efficiency. If *u* and *v* are the optimum solution vectors, for equation (1) then $\alpha u$ and $\alpha v$ will also be optimum solution vectors, and consequently the problem will present infinite solutions.

$$\text{Max } h_0 = \left( \frac{\sum_{j=1}^{s} u_j Y_{j0}}{\sum_{i=1}^{r} v_i X_{i0}} \right)$$

$$\text{subject to } \frac{\sum_{j=1}^{s} u_j Y_{jk}}{\sum_{i=1}^{r} v_i X_{ik}} \le 1, \quad k = 1,...,n \qquad (1)$$

$$u_j \text{ and } v_i \ge 0 \quad \forall j,i$$

where:
$h_0$ – DMU 0 efficiency;
*r* – total amount of inputs
*s* – total amount of outputs
*n* – total amount of DMUs
$Y_{jk}$ – amount of output *j* to DMU *k*
$X_{ik}$ – amount of input *i* to DMU *k*
$u_j$ – weight to output *j*
$v_i$ – weight to input *i*

In order to solve this problem we introduce a linear transformation that allows transforming linear fractional problems into LPPs, creating the model called Multipliers, represented by equation (2).

$$\text{Max } h_0 = \sum_{j=1}^{s} u_j Y_{j0}$$

subject to

$$\sum_{i=1}^{r} v_i X_{i0} = 1 \qquad (2)$$

$$\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{r} v_i X_{ik} \le 0, \quad k = 1,...,n$$

$$u_j \text{ and } v_i \ge 0 \quad \forall j,i$$

It's possible to derive the dual model to the multiplier model also known as the primal one. So, the dual model will present a smaller number of constraints ( $s + r < n + 1$ ), because the DEA model requires that the number of DMUs be greater than the number of variables. By the exposed reasons, the dual model, called Envelope, being easily solved, is preferred compared to the Multipliers model. In the Envelope model the objective is to determine the values of $\lambda_k$, minimising $\theta$ in equation (3).

$$\text{Min } \theta$$

subject to

$$\theta X_{j0} - \sum_{k=1}^{n} X_{ik} \lambda_k \ge 0, \quad i = 1,...,r \qquad (3)$$

$$-Y_{j0} + \sum_{k=1}^{n} Y_{jk} \lambda_k \ge 0, \quad j = 1,...s$$

$$\lambda_k \ge 0, \quad \forall k$$

**BCC** – Model developed by Banker et al. (1984), allows a variable return of scale (VRS) avoiding problems in imperfect competition situations, financial constraints etc. In this case, the VRS frontier considers increasing or decreasing returns in the efficient frontier. To do this job, a convexity constraint is introduced in the CRS mode making the $\lambda$ sum equal to 1. Figure 1 compares the CRS with the VRS frontier, in a situation with 5 DMUs, one input and one output. It is easy to see that DMUs 4 and 5 are not efficient.

For BCC models adopting VRS, each DMU is compared to efficient DMUs that operate in the same scale. So, using the orientation to inputs, one verifies that the optimum projection of the DMU 4 occurs in a point that reflects the convex linear combination of DMUs 1 and 2. Using the orientation to inputs, one also sees that the optimum projection of DMU 4 happens in a point that reflects the convex linear combination of DMUs 2 and 3. Moreover, for both orientation cases, the linear combination values are given by the λs. The Envelope model, oriented to input and the primal derived model or multipliers model are given by (4) and (5), respectively.

$$\text{Min } \theta$$

subject to

$$\theta X_{j0} - \sum_{k=1}^{n} X_{ik} \lambda_k \ge 0, \quad i = 1,...,r$$

$$-Y_{j0} + \sum_{k=1}^{n} Y_{jk} \lambda_k \ge 0, \quad j = 1,...s \qquad (4)$$

$$\sum_{k=1}^{n} \lambda_k = 1$$

$$\lambda_k \ge 0, \quad \forall k$$

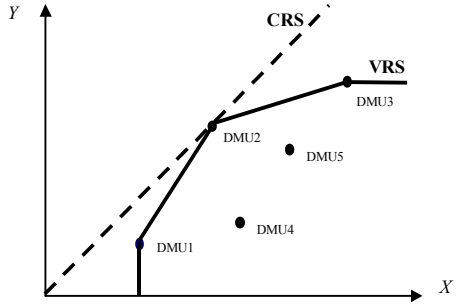


Figure 1: CRS and VRS frontiers.

$$\text{Max } h_0 = \sum_{j=1}^{s} u_j Y_{j0} + \tau_0$$

subject to

$$\sum_{i=1}^{r} v_i X_{i0} = 1 \qquad (5)$$

$$\sum_{j=1}^{s} u_j Y_{jk} - \sum_{i=1}^{r} v_i X_{ik} + \tau_0 \le 0, \quad k = 1,...,n$$

$u_j$ and $v_i \ge 0 \quad \forall j,i$ and $\tau_0$ unrestricted

**Artificial Neural Networks** (ANNs), also known as connectionist systems, are structures inspired in the human brain (Haykin, 1994). ANNs are massively parallel structures, based on simple processing elements (PE), inspired in the biological neurone and densely interconnected. The main ANN characteristics are:

- Parallel search and content addressing, as with the brain that neither has memory addresses nor processes information sequentially;
- Learning capability, making the network able to learn and acquire knowledge through experience, without using any sort of explicit algorithms for the solution and execution of a job;
- Association, allowing the network to associate different patterns;
- Generalisation, enabling the network to deal with noise and distortion and correctly answer to a never seen input exploring similarity with other previously presented patterns;
- Abstraction, endowing the network with the ability to abstract the essence of an input group;
- Robustness, allowing, thanks to parallelism, that even with the loss of PEs, network operation is guaranteed. Through training, ANNs can learn by successive presentation of examples or patterns that are distributively stored in some way in the huge number of connections among the PEs, known as the neural network weights. In that way, knowledge is distributed all over the network. After training the neural network is said to be in the execution mode and that knowledge is ready to be used in many different application areas, as in the neural linear programming case (Haykin, 1994).

A structure similar to an ANN is used in Neuro-LP optimisation modules, which is part of the Neuro-DEA model, where the synaptic weights, obtained in the training mode, are basically formed by the coefficients of the problem constraint groups.

The mostly used network architecture is the feedforward one. This arrangement is composed by a group of PEs, arranged in one or more layers that interconnect themselves in sequence. The most complete configuration presents one or more intermediate or hidden layers between the input and the output layer, and it is known as a multi layer network. Hidden layers allow better results for certain problems, as well as solutions for problems impossible to be solved with single layer networks (Haykin, 1994).

Neural processing is accomplished in two main phases: Training and Execution. The training or learning phase is the updating process of the connection weights. Its goal is to acquire information and store it as a weight matrix ($W$). A Neuro LP, which is the main cell of the Neuro DEA model, skips the training phase once its weights are the LPP constraint coefficients. The execution or recall phase produces the ANNs output ($Y$) in terms of the injected stimulus in the input ($X$) and the weights. The goal of the training phase in the Neuro LP is to determine the optimum values for the LPP decision variables, which in Data Wrapping Analysis may represent the efficiency value of a DMU (Biondi, 2001).

This can be done by solving of a differential equations system, obtained by the transformation of the original LPP in an optimisation problem without constraints. The numerical method used to solve the differential equation system is the dynamic gradient method, derived from the Newton method and is very similar to the ANNs training method.

The ANN architecture used in the Neuro LP model will be presented, as well as the development of the training algorithm based on the minimisation of the sum squared errors in the network output using the gradient method and its variations. In

Neuro LP, an ANN is used in the execution phase with the weights preset to the problem constraint coefficients.

The LPP is transformed into an optimisation problem without constraints (Bazaraa et al., 1993) where function called pseudo-cost is adopted with a penalty term, causing high cost every time a constraint is violated. The new problem can be solved by the gradient method, turning it into a differential equations system, which can be numerically solved.

Convergence speed can be increased, if the proposed modules are integrated in a *VLSI* (Very Large Scale Integration) or a *CMOS* (Complementary Metal-Oxide Semiconductor) chips and connected to a free slot in a personal computer.

# 2 MATHEMATICAL BASICS

Consider an optimisation problem without constraints where we wish to find the value $x \in \Re^n$ that minimises a scalar function $E(x)$, called Pseudo-cost, Energy or Objective function.

According to Bazaraa et al. (1993), the point $x^*$ will be the global minimum of $E(x)$ if $E(x^*) <= E(x)$ for all $x \in \Re^n$ and a local minimum if the relationship $E(x^*) <= E(x)$ is kept for a certain interval $\varepsilon > 0$.

If the first and the second $E(x)$ derivatives exist, the point $x^*$ will be a local minimum if the gradient $\nabla E(x^*)=0$ and the Hessian matrix $\nabla^2 E(x^*) > 0$.

The necessary and sufficient conditions for the existence of a local minimum are: For $\nabla^2 E(x)$ non singular for the point x*, $E(x^*)$ will be $<= E(x)$ for all $0 < \| x - x^* \| < \varepsilon$, $\varepsilon > 0$ if the gradient $\nabla E(x^*) = 0$ and the Hessian matrix is symmetric and positive, $\nabla^2 E(x^*) > 0$.

The Dynamic Gradient Method is the most popular method inspired in the *Steepest Descent technique* and the Newton Method (Bazaraa et al., 1993). It is based in the transformation of the optimisation problem without constraints in a first order ordinary differential equations system, represented as in (6).

$$\frac{dx_j}{dt} = -\mu_{ji} \sum_{i=1}^{n} \frac{\partial E(x)}{\partial x_j},$$

$$\text{where } x_j(0) = x_j^{(0)} \text{ are a initial conditions} \qquad (6)$$

$$\text{and } \mu_{ji} \text{ is a learning matrix.}$$

So, to find the value $x^*$ that minimises $E(x)$, it is necessary to solve or simulate the solution of a differential equation system subjected to initial conditions. One can conclude that $x^*$ can be determined by the "solution path or trajectory curve" of the proposed system (7).

$$x^* = \lim_{t \to \infty} x(t) \qquad (7)$$

# 3 MODELLING

A LPP can be interpreted as an optimisation problem with constraints. In order to solve it using ANN techniques, it is necessary to build a new function called pseudo-cost or energy function $E(x)$, which global minimum is the optimum solution of the LPP.

In order to build the new function $E(x)$, a penalty term $P_i[R_i(x)]$ is incorporated to the original objective function (Chen et al., 1992; Cichocki et al., 1996; Zhu et al., 1992). This penalty term causes high cost to that new function every time a constraint is violated and zero cost if a constraint is satisfied.

So the LPP is transformed into an optimisation problem without constraint, where its desirable to find $x^* \in \Re^n$ that minimises the new function $E(x)$. The penalty term penalises the objective function for feasible solutions and inhibits it for feasible LPP solutions (Dennis and Schnabel, 1996; Rheinboldt, 1998).

The optimisation problem without constraints with penalty term can be solved similarly to the ANN training phase, applying the decreasing gradient method. The problem is written as an ordinary differential equations system and solved numerically. In that case, the solution of the path equation $x_j(k+1) = x_j(k) + \Delta(x_j)$, $j=1...n$, produces in the convergence the value of the problem decision variables. $n$ is the number of variables and is $\Delta(x_j)$ the product of the cost function gradient in relation to $x_j$ by a constant factor related to the convergence speed and the stability of the method).

To ensure accuracy of the method, the penalty parameter $p$ must be very large. However, practice shows that very large values for $p$ are not convenient from a computing point of view. Cichocki and Unbehauen (1996) show that, with careful choice of $p$ values, the minimum of the pseudo-cost function $E(x, p)$ is equivalent to the optimum solution of the original LPP. The same authors say that a good choice is to consider the pseudo-cost function (8).

$$E(x,p) = \sum_{j=1}^{n} C_j x_j - p \sum_{i=1}^{m} \min\{0, R_i(x)\}, \qquad (8)$$

where $p > 0$

Considering $P$ DMUs, $R$ inputs, $S$ outputs, the $i^{th}$ DMU representation by a column vector of inputs $X_i$ and outputs $Y_i$ , the relationship involving all inputs and outputs can be obtained for each DMU as

$\mathbf{u}Y_i/\mathbf{v}X_i$ , where **u** and **v** are output weight vectors and input weight vectors, respectively. The optimum values for these weights are obtained solving a LPP for each DMU (Kallrath, 1997).

The Neuro-DEA architecture model is totally based in the Neuro-LP model as shown in Figure 2 (Biondi, 2001). So, for *P* DMUs, we will have *P* Neuro-LP modules.

Each LPP of the Neuro-DEA model will represent a LPP in the Neuro-LP model and will be able to determine the relative efficiency of one DMU among *P* DMUs comprising the system. Figure 3 shows the proposed block diagram of the Neuro-DEA module.

# 4 IMPLEMENTATION AND RESULTS

The implementation was done using the input oriented CRS Envelope model. The choice of the model is due to the reduction the number of constraints since in the envelope model there is only one constraint for each input/output.

An example involving 5 DMUs with two inputs and one output is shown in Table 1. Table 2 shows the results according to Neuro-DEA, using two commercial softwares (*Lindo* and *Frontier Analyst*), in terms of the error percentage.

# 5 CONCLUSIONS

The example shown in section 4 was selected among several others indicating the consistency the method. The results obtained with our prototype were validated comparing them with those produced by softwares such as *Lindo*, to separately solve the LPPs referring to the DMUs and the *Frontier* to directly solve the DEA. In that case, the observed error was never over 0.5%. Presently, a study using Lagrange Multipliers is being developed in order to optimise the step function.

The solution method for the ordinary differential equation system used in the Neuro-LP model is similar to the technique used in ANN training phase because they use the gradient method. The evolution of the solution method for the differential equations system, represented by the solution path curve, indicates in the convergence, the value of the decision variables for the problem.

Finally, its important to highlight that convergence speed can increase, if the proposed modules are integrated in a chip and connected to a free slot in a personal computer.

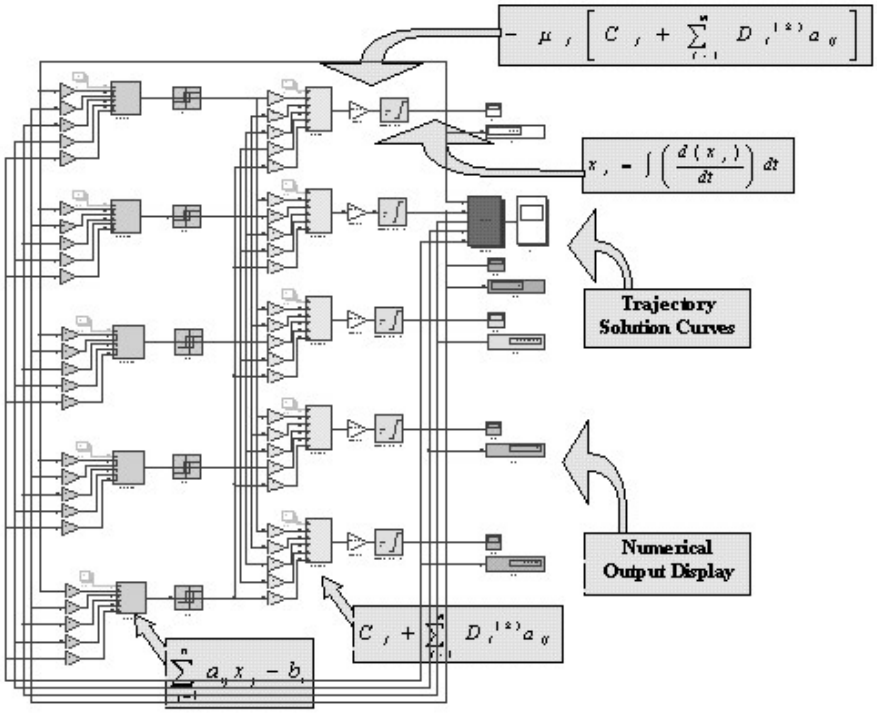


Figure 2: Model up to 5 variables.

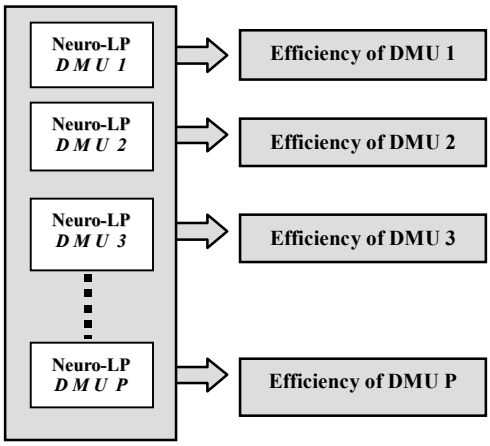


Figure 3: Model block diagram.

Table 1 - Database.

| | OUTPUT *Y* | INPUT *X*1 | INPUT *X*2 |
|---|---|---|---|
| DMU-1 | 2 | 6 | 8 |
| DMU-2 | 4 | 4 | 8 |
| DMU-3 | 3 | 6 | 6 |
| DMU-4 | 1 | 4 | 3 |
| DMU-5 | 4 | 12 | 4 |

Table 2 - DMU's efficiency.

| | Lindo | Frontier | Neuro DEA | % Error |
|---|---|---|---|---|
| DMU 1 | 0.453 | 0.454 | 0.452 | 0.44 |
| DMU 2 | 1.00 | 1.00 | 1.00 | 0.00 |
| DMU 3 | 0.832 | 0.833 | 0.831 | 0.24 |
| DMU 4 | 0.500 | 0.500 | 0.500 | 0.00 |
| DMU 5 | 1.00 | 1.00 | 1.00 | 0.00 |

# REFERENCES


Banker, R., Charnes, A., Cooper, W. W., 1984. Some Models for Estimating Technical an Scale Inefficiencies in Data Envelopment Analysis, *Management Science*, v.30, 1078-1092.

Bazaraa, M.S., Sherali, H.D., Shetty, C.M., 1993. *Nonlinear Programming Theory and Algorithms*, New York, USA, John Wiley & Sons, Inc.

Biondi L.N., 2001. *Neuro-DEA: Nova Metodologia para Determinação da Eficiência Relativa de Unidades Tomadoras de Decisão*. D.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil.

Charnes, A., Cooper, W. W., Rhodes, E., 1978. Measuring the Efficiency of Decision-Making Units, *European Journal of Operational Research*, v.2, 429-444.

Charnes, A., Cooper, W.W., Lewin, A.Y., Seiford,L.M., 1996. *Data Envelopment Analysis: Theory, Methodology, and Application*, Boston, Kluwer Academic Publishers.

Chen, J., Shanblatt, M., Maa, C., 1992. Improved Neural Network for Linear and Nonlinear Programming, *International Journal of Neural Systems*, n. 2, 331-339.

Cichocki, A., Unbehauen, R., 1996. *Neural Networks for Optimisation and Signal Processing*, New York, USA, John Wiley & Sons, Inc.

Cichocki, A., Unbehauen, R., Weinzierl, K., Hölzel, R., 1996. A New Neural Network for Solving Linear Programming Problems, *European Journal of Operational Research*, n. 93, 244-256.

Dennis, J.E., Schnabel, R.B., 1996. *Numerical Methods for Unconstrained Optimisation and Nonlinear Equations*, Englewood Cliffs, N.J., USA, Prentice Hall, Inc.

Haykin, S. 1994. *Neural Networks a Comprehensive Foundation*, London, Macmillan College Publishing Co.

Kallrath, J., 1997. *Business Optimisation using Mathematical Programming*, London, Macmillian Press Ltd.

Rheinboldt, W.C., 1998. *Methods for Solving Systems of Nonlinear Equations*, Philadelphia, SIAM Editors.

Rosenblatt, F., 1962. *Principles of Neurodynamics*, New York, Spartan Editions.

Zhu, X., Zhang, S., Constantinides, A.G., 1992. Lagrange neural Networks to linear programming, *Journal of Parallel Distributed Computing*, n. 14, 354-360.