A Nonlinear Neural Network for Solving Linear Programming Problems

Khanh V. Nguyen

Abstract: **This paper presents a new recurrent neural network for solving linear programming problems. The new model is simpler and more intuitive than existing models and converges very fast to the exact primal and dual solutions. The model is based on a nonlinear dynamical system and has an interesting economic interpretation.**

1. Introduction

Linear programming (LP) is an important class of optimization problems and is used extensively in economics, operations research, engineering, and many other fields. In 1985, Hopfield and Tank published a paper [1] proposing a new approach to solve Linear Programming problems using recurrent neural networks. Unlike the Simplex and other traditional methods, Hopfield and Tank model can be implemented using analog electrical components that operate in parallel. The new model therefore can potentially provide very fast solutions. Hopfield and Tank's pioneer works have generated a lot of interests in recurrent neural networks. In 1987, Kennedy and Chua [2] proposed an improved model that always guaranteed convergence. However, their new model converges to only an approximation of the optimal solution. In addition, a good approximation requires a careful selection of the system parameters. Maa and Shanblatt [3] later proposed a two-phase model that can converge to the exact solution. Their model, however, is relatively complex and still requires some parameter tuning. Recently, Xia [4] introduced a new

model that eliminates these drawbacks. Xia model solves both the primal and dual problems and requires no parameter tuning. In this paper, I will present a new LP neural network that not only retains the advantages of Xia model but also has a simpler and more intuitive architecture as well as a much faster convergence. Unlike Xia's and other previous models, which are piecewise linear, the new model is based on a nonlinear dynamical system.

2. The New Neural Network for Solving LP Problems

Consider a LP problem in the following standard form:

$$\begin{aligned} &\text{Find } x \text{ that maximizes:} && b^T x \\ &\text{Subject to the constraints:} && A\,x \le c, \qquad x \ge 0 \end{aligned} \tag{1}$$

Where x and $b \in R^n$, $A \in R^{m \times n}$, and $c \in R^m$. The dual problem of (1) is:

$$\begin{aligned} &\text{Find } y \text{ that minimizes:} && c^T y \\ &\text{Subject to the constraints:} && A^T y \ge b \qquad y \ge 0 \end{aligned} \tag{2}$$

The topology of the new neural network for solving (1) and (2) is described in figure 1. There are two layers of neurons, one for the primal variables, and one for the dual variables. The outputs from one layer are the inputs to the other layer. Unlike in Tank and Hopfield network, the primal and dual neurons in the new network are symmetrical. The bias (constant) inputs to the primal and dual neurons are vectors b and c respectively.

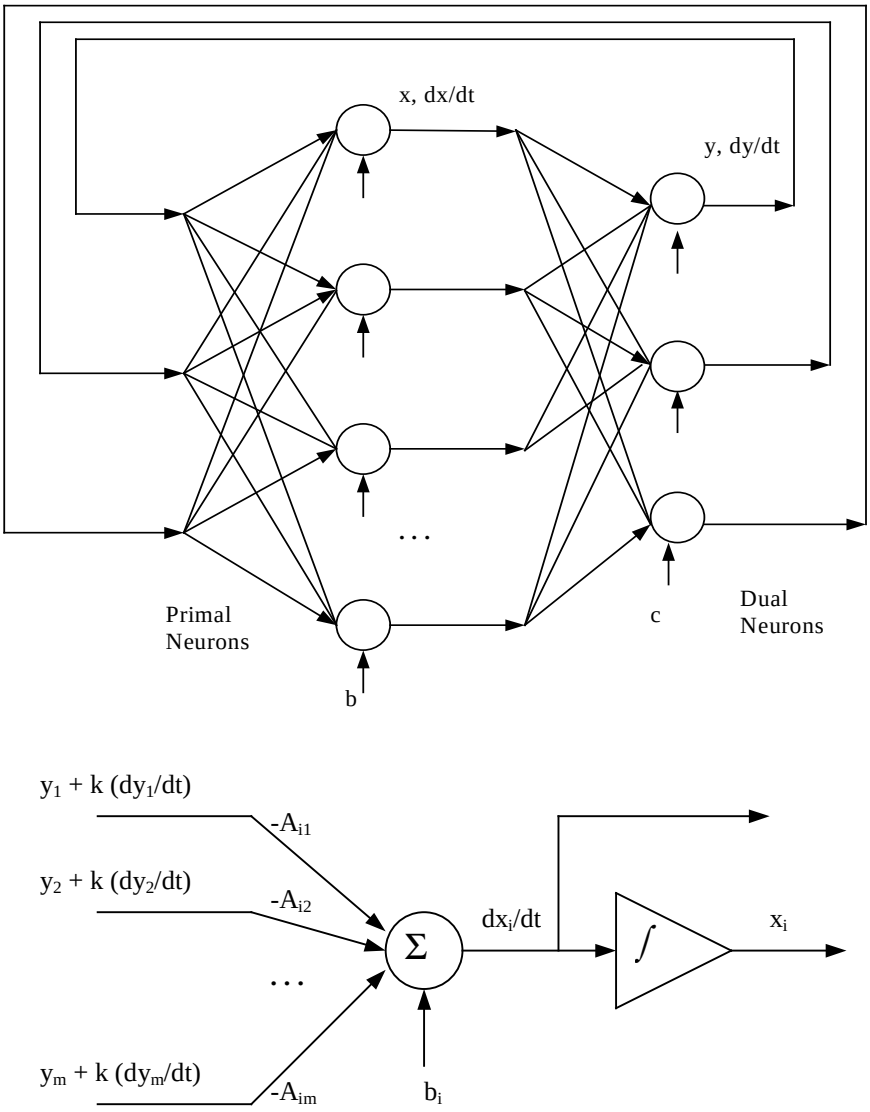

Figure 1: Topology and neuron configuration of the new network

Figure 1 also illustrates the configuration of a primal neuron. A new feature of the above neuron is that it takes as inputs not only the dual neurons' outputs y but also their derivatives dy/dt. This feature introduces nonlinearity into the system. Coefficient k is some positive constant, which will be discussed later. The dual neurons have a similar configuration. Mathematically, the outputs of the above primal and dual neurons can be described by the following nonlinear dynamical system:

$$dx/dt = b - A^T(y + k(dy/dt)) \quad , x \geq 0 \qquad (3)$$

$$dy/dt = -c + A(x + k(dx/dt)) \quad , y \geq 0 \qquad (4)$$

It can be seen that (3) and (4) are equivalent to a system of second order differential equations. The main property of the above system is stated in following theorem:

Theorem 1: If the neural network whose dynamics is described by the differential equations (3) and (4) converges to a stable state, then the convergence will be the optimal solutions for the LP problem (1) and its dual problem (2).

Proof: Let x_i be the i^{th} element of x. Equation (3) can be rewritten as:

$$dx_i/dt = [b - A^T(y + k(dy/dt))]_i \qquad \text{if } x_i > 0 \quad , \forall i \qquad (5a)$$

$$dx_i/dt = \max\{[b - A^T(y + k(dy/dt))]_i, 0\} \qquad \text{if } x_i = 0 \quad , \forall i \qquad (5b)$$

Condition (5b) is to ensure that x will be bounded from below by 0.

Let x*, y* be the limit of x and y respectively. Because of the stability of the convergence, we have dx*/dt = 0 and dy*/dt = 0. Equations (5a) and (5b) then become:

$$0 = [b - A^T y^*]_i \qquad \text{if } x_i^* > 0 \qquad (6a)$$

$$0 = \max\{[b - A^T y^*]_i, 0\} \qquad \text{if } x_i^* = 0 \qquad (6b)$$

In other words:

$$[b - A^T y^*]_i = 0 \qquad \text{if } x_i^* > 0 \qquad (7a)$$

$$[b - A^T y^*]_i \leq 0 \qquad \text{if } x_i^* = 0 \qquad (7b)$$

Or: $$b - A^T y^* \leq 0 \qquad \forall\, i \qquad (8)$$

Similarly, taking the limit of (4) we will have:

$$Ax^* - c \leq 0 \qquad (9)$$

Equations (8) and (9) shows that x* and y* are the feasible solutions for the problems (1) and (2). Furthermore, from (7a) and (7b) we have:

$$x_i^*[b - A^T y^*]_i = 0 \quad , \forall i \qquad (10)$$

Or in vector form: $b^T x^* - x^* A^T y^* = 0$ (11)

Similarly, from (4): $x^* A^T y^* - c^T y_j^* = 0 \quad , \forall j$ (12)

From (11) and (12): $b^T x^* = c^T y_j^*$ (13)

By the LP Duality theory, from (13) and the feasibility of x* and y*, we can conclude that x* and y* are the optimal solutions for the LP problems (1) and (2).

Although a proof for the global (Liapunov) stability of the new model has not been found yet, experiments indicated that the system always converge to a stable state if k>0. The convergence has the form of dampened oscillations (Figure 2). The convergence speed depends on the coefficient k. Intuitively, k acts like the coefficient of a kinetic friction that dampens the oscillations. If k = 0 (no friction), the system will oscillates indefinitely without converging.

A discrete simulation of the above network has been implemented. The Euler method is used to solve differential equations (3) and (4). Experiments show that the system converges very fast. For a LP problem with 5 variables and 4 constraints, the discrete model converges after about 1000 iterations. In comparison, it takes more than 200,000 iterations to solve the same problem using Xia model (Figure 2). The performance improvement of the new model is primarily due to its ability to handle larger discrete time step (dt) without becoming unstable. Finally, the

complexity of the new neural network (about n+m adders and 2n.m multipliers) is only about half of Xia network's.

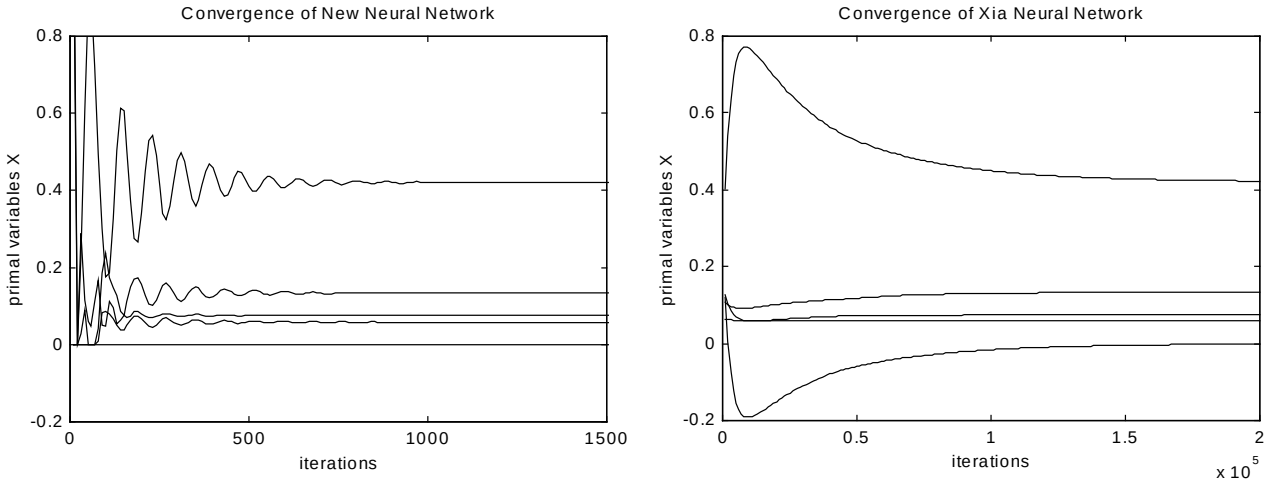

Figure 2: System Convergence

The following Matlab code describes the discrete implementation of the above neural network. Coefficient k is set to equal the time step dt to simplify the calculations.

```
for i=1:n
   dx = (b – A'*(y + dy)) * dt;
   dx = max(x + dx, 0) - x;       % to make x >=0
   x = x + dx;
   dy = (-c + A *(x + dx)) * dt;
   dy = max(y + dy, 0) - y;       % to make y >=0
   y = y + dy;
end;
```

3. Economic Interpretation

The intuition of the above neural network can be demonstrated by the following economic interpretation of the system. Consider a resource allocation problem in which there are n different types of products made from m resources of limited supply. The LP problem is to maximize the value of products made $b^T x$, given the limitations of the resources $Ax \leq c$, where A defines the amounts of resources needed to make one unit of each product, b is the products' unit values, and c is the supply limitations of the resources. In this interpretation, each primal neuron

plays the role of a producer of a given type of products, and each dual neuron plays the role of a resource owner selling a given resource. Primal variables x are the amounts of the products made, and dual variables y are the prices of the resources. Equation (3) specifies the strategy of a producer: the amount of products made is proportional to the profitability of that product (value minus cost = $b - A^T y$). The more profitable, the higher increase in production (dx/dt). The term k(dy/dt) signifies the fact that the producers not only use the current resource prices in their cost calculation but also take into account the trend of these prices. Given the same resource prices, increasing prices will result in less productivity than declining prices.

Similarly, the resource owner's pricing strategy is described by (4). The price of a resource is based on the supply and demand law (demand minus supply = $Ax - c$). The higher demand, the higher the price. Like the producers, the resource owners base their decisions not only on the current demands Ax but also on the trend of these demands Adx/dt. As a result of these market interactions among the producers and resource owners, the system will eventually reach a state where the optimal amounts of products are made, which are the optimal solution to the underlying LP problem. It's interesting to note that if these market players (producers and resource owners) do not take into account the future trends in their calculation (i.e. k=0), the system will oscillate indefinitely without converging. This shows how business forecasting is important to the market economy.

4. Conclusions

The new neural network proposed in this paper has many advantages compared to existing neural networks for solving linear programming problems. It converges to the exact solutions of the LP

problem and the dual problem without requiring any parameter tuning. The new model is also very simple. Yet despite its simplicity, the system converges very fast. Experiment with a limited number of cases showed that the new model outperformed Xia model by more than two orders of magnitude. Another advantage of the new model is that it is very intuitive and can be explained in common sense without formal mathematics. As one potential application, the new model can be used to understand better the dynamics of the free market economy and to explain what Adam Smith [5] called "the invisible hand" that made the market economy achieve the best efficiency. Finally, more studies need to be done to see whether the model can be extended to solve other optimization problems, including convex optimization and other nonlinear programming problems

REFERENCES

[1] Tank, D. and J. Hopfield. "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit." IEEE Trans. on Circuits and Systems, May 1986.

[2] Kennedy, M. and L. Chua. "Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin." IEEE Trans. on Circuits and Systems, Feb 1987.

[3] Maa, C. and M. Shanbaltt. "A Two-phase Optimization Neural Network." IEEE Trans. on Neural Networks. Nov. 1992.

[4] Xia, Y. "A New Neural Network for Solving Linear Programming Problems and Its Application." IEEE Trans. on Neural Networks, Mar 1996.

[5] Smith, A. "The Wealth of Nations." Modern Library, Feb 1994.

Note: The following appendixes are provided to facilitate the reviewers' duplication of the research results and may not need to be included in the final publication.

Appendix 1: LP problem used in performance evaluation (Figure 2).

Maximizes: $b^T x$

Subject to: $A\,x \le c, \quad x \ge 0$

Where:

```
A = [ 14,  4,   1,   3,  2
      7.5, 15,  3.3, 3,  1.5
      2.5, 3,   17,  2,  2.3
      3.2, 1.6, 2.5, 5,  7  ];
b = [  32, 43, 36, 28, 29 ]';
c = [  3.5, 3.6, 2.4, 2.8 ]';
```

Primal optimal solution (apprx): `[0.1340, 0.0760, 0.0586, 0.4207, 0.0000]'`

Dual optimal solution (apprx): `[0.0215, 2.2308, 1.1247, 3.7987]'`

Appendix 2: Discrete implementation of the new neural network in Matlab code.

```
x = [ 0,0,0,0,0 ]';
y = [ 0,0,0,0 ]';
dy= [ 0,0,0,0 ]';
dt = 0.02;
n = 2000;

for i=1:n
    dx = dt*( b – A'*(y + dy));
    dx = max(x+dx, 0) – x;    % to make x >= 0
    x = x + dx;
    dy = dt*(-c + A *(x + dx));
    dy = max(y+dy, 0) – y;    % to make y >= 0
    y = y + dy;
end;
```