정보 및 제어 심포지엄(ICS'04) 2004. 5. 22

Problem Solution of Linear Programming based Neural Network

손 준 혁*, 서 보 혁**
(Jun-Hyug Son, Bo-Hyeok Seo)

Abstract -Linear Programming(LP) is the term used for defining a wide range of optimization problems in which the objective function to be minimized or maximized is linear in the unknown variables and the constraints are a combination of linear equalities and inequalities. LP problems occur in many real-life economic situations where profits are to be maximized or costs minimized with constraint limits on resources. While the simplex method introduced in a later reference can be used for hand solution of LP problems, computer use becomes necessary even for a small number of variables. Problems involving diet decisions, transportation, production and manufacturing, product mix, engineering limit analysis in design, airline scheduling, and so on are solved using computers. This technique is called Sequential Linear Programming(SLP). This paper describes LP's problems and solves a LP's problems using the neural networks.

Key ward : neural, linear programming, problems, solution

1. Introduction

A Linear Programming(LP) problem seeks to optimize a linear objective function subject to a set of linear function constraints and non_negativity constraints. Many real-time systems, such as machine vision in robotic operations and large-scales systems, such as personnel and equipment maneuvers in military operation, require solving large-scale LP problems in real time. In such applications, existing sequential algorithms such as the classical simplex method are usually not efficient due to the limitation of sequential processing.

In recent years, Neural Network(NN)s have been proposed for solving various optimization problems[1]. A NN consists of a number of massively connected simple neurons that operate concurrently in a parallel distributed fashion. While the mainstream of the NN approach to optimization has focused on combinatorial optimization problems, a number of NN paradigms have been proposed for solving LP problems. So to speak, Tank and Hopfield demonstrate the potential of the Hopfield network for solving LP problems via an example[2]. Kennedy and Chua analyze the Tank and Hopfield LP circuit and Chua and Lin nonlinear programming circuit from a circuit-theoretic viewpoint and reconcile a modified canonical nonlinear programming circuit that can also be applied for LP[3][4]. Rodriguez et al. develop switched-capacitor NN for linear and nonlinear programming[5]. Maa and Shanblatt analysis the properties of the recurrent NNs for linear and quadratic programming[6]. The results of these investigations have demonstrated great potential and shed much light on the NN research. However, none of the paradigms can be theoretically guaranteed to generate optimal solutions to linear programs.

Recently, Wang proposed a class of recurrent NNs for optimization[7]. This particular class of recurrent NNs has been proven to be able to generate optimal solutions to linear programs. In this class of current NNs, the interconnection weights are essentially time-varying. Although a block diagram of the proposed NN is presented in which two dummy neurons are defined to circumvent the time-varying connection weight, the complexity of the NN configuration still makes circuit realization challenging[8]. Problems involving diet decisions, transportation, production and manufacturing, product mix, engineering limit analysis in design, airline scheduling, and so on are solved using computers. This technique is called Sequential Linear Programming(SLP). This paper describes LP's problems and solves a LP's problems using the neural networks.

2. Linear programming Problem

The general form of the LP problem has objective function to be minimized or maximized, and a set of constraints.

maximize : $c_1x_1 + c_2x_2 + c_3x_3 + \cdots \cdots \cdots + c_nx_n$ (1)

저자 소개
* 正 會 員 : 慶北大學校 大學院 電氣工學科 博士課程
** 正 會 員 : 慶北大學校 電子電氣工學部 敎授·工博

subject to

$c_{i1}x_1 + c_{i2}x_2 + c_{i3}x_3 + \cdots \cdots \cdots + c_{in}x_n \leq b_i$

: LE constrains$(i = 1 \ \ to \ \ l)$ (2)

$c_{j1}x_1 + c_{j2}x_2 + c_{j3}x_3 + \cdots \cdots \cdots + c_{jn}x_n \leq b_j$

: GE constrains$(j = l+1 \ \ to \ \ l+r)$ (3)

$c_{k1}x_1 + c_{k2}x_2 + c_{k3}x_3 + \cdots \cdots \cdots + c_{kn}x_n \leq b_k$

: EQ constrains$(k = j+r+l \ \ to \ \ l+r+q)$ (4)

※$x_1 \geq 0, x_2 \geq 0, \cdots \cdots \cdots, x_n \geq 0.$

We note that the number of constrains is $m = l+r+q$, c_j and a_{ij} are constant coefficients, b_i are fixed real constants which are adjusted to be non-negative, and x_j are the unknown variables to be determined. In engineering design problems, the x_j are referred to as design variables.

Standard form of the LP is one in which all constrains are converted into the equality(EQ) type. LE-type constrains are converted into equalities by adding a non-negative variable $x_i (i = 1 \ \ to \ \ L)$, called a slack variable, on the left-hand side. GE-type constrains are converted into equalities by subtracting a non-negative variables x_j $(j = l+1 \ \ to \ \ l = r)$, called surplus variables. Variables unrestricted in sign, also called free-variable, can be brought into the standard form by expressing each, such variable as a difference of two non-negative variables and substituting into previous relations. In the standard form $b \geq 0$..

3. The dual simplex method

In some problems, basic feasible solution may mot be readily available for the primal problem while the dual feasible solution(satisfying primal optimality) is available. The dual simplex method is a natural choice for these problems. The steps are very similar to the conventional simplex method discussed earlier. Here we maintain with the primal simplex procedure, the points generated by the dual simplex procedure are always optimal during the iterations in that the cost coefficients are non-negative; however, the points are infeasible(some basic $x_i < 0$). In the dual simplex method, the iterations terminate when we achieve a feasible point. The following example and fig. 1 illustrate the dual simplex procedure. Consider the two-variable problem.

minimize $2x_1 + x_2$ (5)

subject to $2x_1 + 5x_2 \geq 20, \quad x_1 + x_2 \geq 6$ (6)

The two-variable problem is illustrated graphically in Fig. 1.

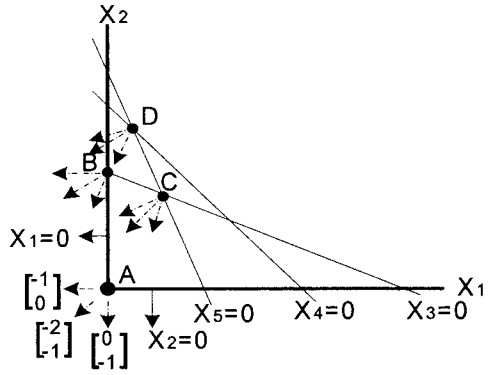

Fig. 1. Dual method

4. Linear programming of neural network

The LP NN emulates the first order dynamical systems associated with the dual simplex method: an imaginary point with no mass in a time-evolving gradient vector field $y(w,t)$, of the augmented barrier functions $F_{g(t)}$. The evolution of the point is governed by a set of differential equations involving the local gradient. Use the vector of slack variables $w(x) = b - Ax > 0$ to define the vector $w'(x,t)$. It is defined by $w'(x,t) = [g(t) \cdot w_i(x)]^{-1}$. The gradient of $f_{g(t)}(x)$ is computed as $y(x,t) = c - A^t w'(x,t)$. The essence of the dual simplex method is this: if the initial point x^0 lies in the interior of the feasible region, and if the gain term $g(t)$ increase with time, then $x(t)$ converges to an optimal solution of $\max_x c^t x$, such that $Ax \leq b$.

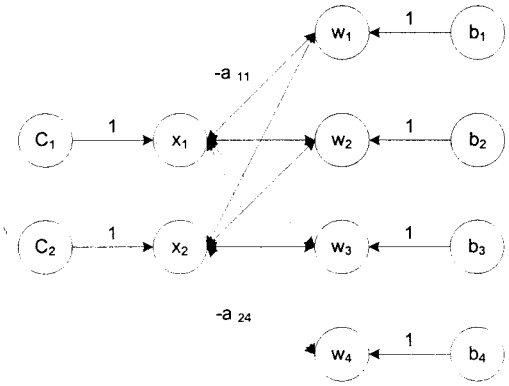

Fig. 2. Linear programming of neural network

Fig. 2 shows a small version of the LP NN, in particular, one that 4-constraints and 2-variables inequality constrained LP problem. There four types of neurons in this architecture; those that output x_j, w_i', c_j and b_i. Let us assume that x^0 denotes once again a known strictly interior feasible point$(Ax^0 < b)$. In the complete system of differential equations for the dynamical system, the x_j neurons are initialized to $x_j = x_j^0$ and governed by the

equation

$$\frac{dx_j}{dt} = c_j - \sum_{i=1}^{m} A_{ij} w_i' \quad \text{for all } j. \tag{7}$$

The w_i' neurons are initialized to satisfy $w = b - Ax^0$. They are governed by the equation

$$r\frac{dw_i}{dt} = -w_i + b_i - \sum_{j=1}^{n} A_{ij} x_j \quad \text{for all } i \tag{8}$$

and the nonlinear transformation

$$w_i' = f(w_i, t) \tag{9}$$

More explicitly,

$$f(w_i, t) = \frac{1}{g(t) w_i} \tag{10}$$

where $g(t)$ is a monotonically increasing function of time and r is vary small compared to the scale of $\frac{dg}{dt}$. The c_j and b_i neurons are kept constant at all times. Differential equation (8) is derived form the definition $w(x) = b - Ax > 0$ of the slack variables. The time derivative of $w(x)$ is equated to the deviation $d(t) = -w(x) + b - ax$. When the dynamical system stabilizes, the definition is satisfied with $d(t) = 0$. Otherwise, (8) servos $w(x)$ to reduce $d(t)$. The connection strengths are unity for all c_j and b_i neurons, while the w_i' and x_j neurons are fully interconnected with bidirectional weights $-A_{ij}$. No real-time learning or weigh adaption occurs in this system. The parameters $-A_{ij}$, c_j and b_i, where $1 \le j \le n$ and $1 \le i \le m$, are set at $t = 0$ and do not change through the convergence of the system[9].

5. Solution of linear programming problem using neural network

The first step is to write the first tableau by introducing the surplus variables. We make the coefficients of the surplus variables positive by multiplying throughout by -1, and we do not require any artificial variables.

Table 1. First Tableau (corresponds to point A)

	x_1	x_2	x_3	x_4	x_5	*rhs.*
f	2	1	0	0	0	0
x_3	-2	-5	1	0	0	-20
x_4	-1	-1	0	1	0	-6
x_5	-3	-1	0	0	1	-9

Note that the first row coefficients are positive, which implies primal optimality or dual feasibility. However, the basic solution is infeasible since each of x_3, x_4, x_5 is negative.

Elementary row operations are performed for this pivot to make the number unity at the pivot and the rest of the coefficients in that column zero.(See point B in Fig. 1.)

Table 2. Second Tableau (corresponds to point B)

	x_1	x_2	x_3	x_4	x_5	*rhs.*
f	1.6	0	0.2	0	0	-4
x_2	0.4	1	-0.2	0	0	4
x_4	-0.6	0	-0.2	1	0	-2
x_5	-2.6	0	-0.2	0	1	-5

The pivot row is the third row corresponding to x_4 and the pivot column is 1. The third tableau follow naturally.

Table 3. Third Tableau (corresponds to point C)

	x_1	x_2	x_3	x_4	x_5	*rhs.*
f	0	0	0.077	0	0.615	-7.077
x_2	0	1	-0.231	0	0.154	3.231
x_4	0	0	-0.154	1	-0.231	-0.846
x_1	1	0	0.077	0	-0.385	1.923

Table 4. Fourth Tableau (corresponds to point D)

	x_1	x_2	x_3	x_4	x_5	*rhs.*
f	0	0	0	0.5	0.5	-7.5
x_2	0	1	0	-1.5	0.5	4.5
x_3	0	0	1	-6.5	1.5	5.5
x_1	1	0	0	0.5	-0.5	1.5

Primal optimality has been achieved while maintaining the dual feasibility. The solution is x_1=1.5, x_2=4.5, and the function value is 7.5(flipping the sign for the minimum problem).

6. Conclusion

In this paper, we described a NN architecture that can solve classical LP problems with a massively parallel algorithm. The algorithm is based on the logarithmic barrier function approach to LP. In other to solve the basic dynamics of these network, we simulated LP problems using the differential-equation approach. Thus far, we have simulated the effects of limited numerical precision of analogue devices and of random noise that may be described.

Reference

[1] Jun Wang, "Analysis and Design of Recurrent Neural Network for Linear Programming", IEEE Transactions on Circuits and Systems Vol. 40, pp. 613 - 618, 1993.

[2] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit", IEEE Transactions on Circuits and Systems Vol. CAS-33, no. 5, pp. 533 - 541, 1986.

[3] M. P. Kennedy and L. O. Chua, "Unifying the Tank and Hopfield linear programming circuit and the canonical nonlinear programming circuit of Chau and Lin", IEEE Transactions on Circuits and Systems Vol. CAS-34, no. 2, pp. 210 - 214, 1988..

[4] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming", IEEE Transactions on Circuits and Systems Vol. CAS-35, no. 5, pp. 554 - 562, 1988.

[5] A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, J. L. Huertas, and E. Sanchez-Sinencio, "Nonlinear switched-capacitor neural networks for optimization problems", IEEE Transactions on Circuits and Systems Vol. 37, no. 3, pp. 384 - 397, 1990..

[6] C. Y. Maa and M. A. Shanblatt, "Linear and quadratic programming neural network analysis", IEEE Transactions on Neural Networks Vol. NN-3, pp. 580 - 594, 1992.

[7] J. Wang, "On the asymptotic properties of recurrent neural network for optimization", Int. J. Pattern Recognition Artificial Intelligence, vol. 5, no. 4, pp. 581 - 601, 1991.

[8] J. Wang and V. Chankong, "Recurrent neural networks for linear programming: Analysis and principles", Compute. Operat. Res., Vol. 19, nos. 3/4, pp. 297 - 311, 1992.

[9] Caudell, T.P.; Zikan, K. "Neural network architecture for linear programming", Neural Networks, IJCNN., International Joint Conference on Vol. 3, pp. 91 - 96, 1992.