

Linear And Nonlinear Optimization Using Parameterized Hopfield Neural Networks

¹S.Danipoul and ²Dr.S.Pradeep Gnamam
¹Research Scholar, ²Asst.Professor

^{1,2}PG & Research Department of Computer science, Joseph Arts and Science College Thirunavallur,
Villupuram,Tamilnadu, India

Abstract-- The Hopfield computational model is almost exclusively applied to the solution of combinatorial complex linear decision problems. The ability of networks of highly interconnected simple nonlinear analog processors (Neurons) to solve complicated optimization problems. A nonlinear neural framework called the Parameterized Hopfield Network is proposed, which is able to solve in a parallel distributed manner systems of nonlinear equations. The method is applied to the general nonlinear optimization problem. We demonstrate linear PHNs Cauchy and Newton dynamics are the two most important unconstrained optimization algorithms and nonlinear PHNs implementing the three most important optimization algorithms, namely the Augmented Lagrangian, Generalized Reduced Gradient and Successive Quadratic Programming methods. The study results in a dynamic view of the optimization problem and offers a straightforward model for the parallelization of the optimization computations, thus significantly extending the practical limits of problems that can be formulated as an optimization problem and which can gain from the introduction of nonlinearities in their structure (eg. pattern recognition, supervised learning, design of content-addressable memories).

Keywords-- Hopfield neural network, constrained optimization, Parameterized solution.

I. INTRODUCTION

The ability of networks of highly interconnected simple linear and nonlinear analog processors (neurons) to solve complicated optimization problems was demonstrated in a series of papers by Hopfield and Tank (Hopfield, 1984), (Tank, 1986). The Hopfield computational model is almost exclusively applied to the solution of combinatorially complex linear decision problems (eg. Traveling Salesman Problem). Unfortunately such problems can not be solved with guaranteed quality, (Bruck, 1987), getting trapped in locally optimal solutions. Jeffrey and Rossner, (Jeffrey, 1986), extended Hopfield's technique to the nonlinear unconstrained optimization problem, using Cauchy dynamics. Linear optimization problem using Newton dynamics. Kennedy and Chua, (Kennedy, 1988), presented an analog implementation of a network solving a nonlinear optimization problem. The underlying optimization algorithm is a simple transformation method, (Reklaitis, 1983), which is known to be relatively inefficient for large nonlinear optimization problems.

II. RATIONALE

Hopfield networks consist of the previously described elements and are totally dynamical, so including the time shift and possible updating rules. The space of all possible states of the network, is called the configuration space. Updating rules hopefield network have two simplified versions. The first one Synchronous or Parallel that mean All neurons update their activity states simultaneously at discrete time steps n , where $n = 1, 2, \dots$, as if governed by a clock. The inputs of every neuron in the network are determined by the same activity

state of the network in the time interval $(n-1) < t < n$. This choice requires a central clock or pacemaker and is sensitive to timing errors. The second version is Asynchronous or Sequential (more natural for both brains and artificial networks) All neurons are updated one by one, where one can proceed in either of two ways one at each time step, select at random a unit i to be updated and apply the rule and another let each unit independently choose to update itself, with some constant probability per unit.

A. Linear Hopfield Network (LHN)

The computation in a Hopfield network is done by a collection of highly interconnected simple neurons. Each processing element, i , is characterized by the activation level, U_i , which is a function of the input received from the external environment, I_i , and the state of the other neurons. The activation level of i is transmitted to the other processors, after passing through a filter that converts U_i to a 0-1 binary value, V_i . The time behavior of the system is described by the following model:

$$C_i \left(\frac{du_i}{dt} \right) = \sum_j T_{ij} V_j - \frac{u_i}{R_i} + I_i$$

Where T_{ij} are the interconnection strengths. The network is characterized as linear, because the neuron inputs appear linearly in the neuron's constitutive equation. The steady-state of a Hopfield network corresponds to a local minimum of the corresponding quadratic Lyapunov function:

$$E = -\frac{1}{2} \sum_i \sum_j T_{ij} V_i V_j + \sum_i I_i V_i + \sum_i \left(\frac{1}{R_i} \right) \int_0^{V_i} s_i^{-1}(V) dV$$

If the matrix $[T_{ij}]$ is symmetric, the steady-state values of V_i are binary. These observations turn the Hopfield network to a very useful discrete optimization tool. Nonetheless, the linear structure poses two major limitations: The Lyapunov (objective) function can only take a quadratic form,

Whereas the feasible region can only have hypercube geometry ($-1 \leq V_i \leq 1$) Therefore, the Linear Hopfield Network is limited to solve optimization problems with quadratic objective function and linear constraints. The general nonlinear optimization problem requires arbitrarily nonlinear neural interactions.

B. The Nonlinear Optimization Problem

The general nonlinear optimization problem consists of a search for the values of the independent variables X_i , optimizing a multivariable objective function so that some conditions (equality. h_i . and inequality. g_j . constraints) are satisfied at the optimum.

Optimize $f(x_1, x_2, \dots, x_n)$

Subject to

$$h_i(x_1, x_2, \dots, x_n) = 0 \quad i = 1, 2, \dots, k, \quad k < n$$

$$a_j \leq g_j(x_1, x_2, \dots, x_n) \leq b_j \quad j = 1, 2, \dots, m$$

The influence of the constraint geometry on the shape of the objective function is described in a unified manner by the Lagrangian Function:

$$L = f - v^T h$$

The v_j variables, also known as Lagrange multipliers, are unknown weighting parameters to be specified. In the optimum, the following conditions are satisfied

$$\nabla_x L = 0 \quad (N \text{ equations}) \quad (1)$$

$$\nabla_v L = 0 \quad (K \text{ equations}) \quad (2)$$

From (1) and (2) it is clear that the optimization problem is transformed into a nonlinear equation solving problem. In a Generalized Hopfield Network each neuron represents an independent variable. The nonlinear connectivity among them is determined by the specific problem at hand and the implemented optimization algorithm. The network is designed to relax from an initial state to a steady-state that corresponds to a locally optimal solution of the problem. Therefore, the optimization algorithms must be transformed into a dynamic model -system of differential equations - that will dictate the nonlinear neural interactions.

III. RESEARCH METHODOLOGY

Cauchy and Newton dynamics are the two most important unconstrained optimization (equation solving) methods, adopted by the majority of the existing algorithms

A. Cauchy's Method

This is the famous steepest descent algorithm, which tracks the direction of the largest change in the value of the objective function, f . The "equation of motion" for a Cauchy dynamic system is:

$$\frac{dx}{dt} = -\nabla f ; \quad x(0) = x_0$$

B. Newton's Method

If second-order information is available, a more rapid convergence is produced using Newton's approximation:

$$\frac{dx}{dt} = \pm (\nabla^2 f)^{-1} \nabla f ; \quad x(0) = x_0$$

The steepest descent dynamics are very efficient initially, producing large objective value changes, but close to the optimum they become very small, significantly increasing the convergence time. In contrast, Newton's method has a fast convergence close to the optimum, but the optimization direction is uncontrollable. The Levenberg - Marquardt heuristic, (Reklaitis, 1983), solves the problem by adopting Cauchy dynamics initially and switch to Newton dynamics near the optimum. The algorithm converges to locally optimal solutions.

IV. CONSTRAINED OPTIMIZATION

The constrained optimization algorithms attempt to conveniently manipulate the equality and inequality constraints so that the problem is finally reduced to an unconstrained optimization, which is solved using Cauchy's or Newton's methods. Three are the most important constrained optimization algorithms: The Augmented Lagrangian, the

Generalized Reduced Gradient (GRG) and the Successive Quadratic Programming (SQP). Corresponding Generalized Hopfield Networks will be developed for all of them.

A. Transformation Methods - Augmented Lagrangian

According to the transformation methods, a measure of the distance from the feasibility region is attached to the objective function and the problem is solved as an unconstrained optimization one. A transformation method was employed by Hopfield. These algorithms are proved inefficient because of numerical difficulties implicitly embedded in their structure, (Reklaitis, 1983). The Augmented Lagrangian is specifically designed to avoid these problems. The transformed unconstrained objective function becomes

$$P(x, \sigma, \tau) = f(x) + R \sum_j \{ \langle g_j(x) + \sigma_j \rangle^2 - \sigma_j^2 \} \\ + R \sum_i \{ [h_i(x) + \tau_i]^2 - \tau_i^2 \}$$

Where R is a predetermined weighting factor, and the corresponding inequality - equality Lagrange multipliers. The operator $\langle a \rangle$ returns a for $a < 0$. Otherwise it returns 0.

The design of an Augmented Lagrangian PHN requires $(N + K)$ neurons, where N is the number of variables and K is the number of constraints. The neuron connectivity of a PHN with Cauchy performance is described by the following model:

$$\frac{dx}{dt} = -\nabla_x P = -\nabla f - 2R \langle g + \sigma \rangle^T \nabla g - 2R [h + \tau]^T \nabla h$$

$$\frac{d\sigma}{dt} = +\nabla_\sigma P = 2R \langle g + \sigma \rangle - 2R \sigma$$

$$\frac{d\tau}{dt} = +\nabla_\tau P = 2R h$$

where ∇g and ∇h are matrices, eg. $\nabla h = [\nabla h_1, \dots, \nabla h_k]$.

Where ∇g and ∇h are matrices, ego $\nabla h = [\nabla h_1, \dots, \nabla h_k]$.

B. Generalized Reduced Gradient

According to the GRG method, K variables (basics, X) are determined by solving the K nonlinear constraint equations, as functions of the rest $(N - K)$ variables (non-basics, i). Subsequently the problem is solved as a reduced-dimension unconstrained optimization problem. Equations (1) and (2) are transformed to:

$$\nabla \bar{f} = \nabla \bar{f} - \nabla \hat{f} (\nabla \hat{h})^{-1} \nabla \bar{h} = 0$$

$$h(x) = 0$$

The constraint equations are solved using Newton's method. Note that the Lagrange multipliers are explicitly eliminated. The design of a GRG PHN requires N neurons, each one representing an independent variable. The neuron connectivity using Cauchy dynamics for the unconstrained optimization is given by:

$$\frac{d\bar{x}}{dt} = -\nabla \bar{f} = -\nabla \bar{f} + \nabla \hat{f} (\nabla \hat{h})^{-1} \nabla \bar{h} \quad (3)$$

$$h(x) = 0 \quad (\rightarrow \frac{d\hat{x}}{dt} = h (\nabla \hat{h})^{-1}) \quad (4)$$

$$x(0) = x_0$$

System (3)-(4) is a differential - algebraic system, with an inherent sequential character: for each small step towards lower objective values, produced by (3), the system of nonlinear constraints should be solved, by relaxing equations

(4) to a steady-state. The procedure is repeated until both equations (3) and (4) reach a steady state.

C. Successive Quadratic Programming

In the SQP algorithm equations (1) and (2) are simultaneously solved as a nonlinear system of equations with both the independent variables, x , and the Lagrange multipliers, v , as unknowns. The solution is determined using Newton's method.

The design of an SQP PHN requires $(N + K)$ neurons representing the independent variables and the Lagrange multipliers. The connectivity of the network is determined by the following state equations:

$$\frac{dz}{dt} = \pm [\nabla^2 L]^{-1} (\nabla L)$$

$$z(0) = z_0$$

Where z is the augmented set of independent variables: $z = [x;v]$

D. Comparison of the Networks

The Augmented Lagrangian network is very easily programmed. Newton dynamics should be used very carefully because the operator $\langle a \rangle$ is not smooth at $a = 0$. The GRG network requires K fewer neurons compared to the other networks. It requires more programming effort because of the inversion of the constraint Jacobian.

The SQP network is algorithmically the most effective, because second order information is used in the determination of both the variables and the multipliers. It is the most tedious to program because of the inversion of the Lagrange Hessian. All the PHNs are proved to be stable, (Tsirikis, 1989). The following example was solved by all three networks.

$$\text{minimize } f(\mathbf{x}) = -x_1 x_2^2 x_3^3 / 81$$

$$\text{subject to}$$

$$h_1(\mathbf{x}) = x_1^3 + x_2^2 + x_3 - 13 = 0$$

$$h_2(\mathbf{x}) = x_2^2 x_3^{-1/2} - 1 = 0$$

Convergence was achieved by all the networks starting from both feasible and infeasible initial points.

V. OPTIMIZATION & PARALLEL COMPUTATION

The presented model can be directly translated into a parallel nonlinear optimizer - nonlinear equation solver - which efficiently distributes the computational burden to a large number of digital processors (at most $N+K$). Each one of them corresponds to an optimization variable, continuously updated by numerically integrating the state equations:

$$x_j^{(t+1)} = \phi(x^{(t)}, x^{(t+1)})$$

Where ϕ depends on the optimization algorithm and the integration method. After each update the new value is communicated to the network.

CONCLUSION

The presented algorithm has some unique features: The state equations are differentials of the same function, the Lagrangian. Therefore, a simple integration method (eg. explicit) can be used for the steady-state computation. Also, the integration in each processor can be done asynchronously, independent of the state of the other processors. Thus, the

algorithm is robust to intercommunication and execution delays.

References

- [1] Bruck, J. and J. Goodman (1988). On the Power of Neural Networks for Solving Hard Problems. Neural Information Processing Systems, D2. Anderson (ed.), American Institute of Physics, New York, NY, 137-143.
- [2] Hopfield J.I. (1984), Neurons with Graded Response have Collective Computational Properties like those of Two-state Neurons, Proc. Natl. Acad. Sci. USA, vol. 81, 3088-3092.
- [3] Jeffrey, W. and R. Rosner (1986), Neural Network Processing as a Tool for Function Optimization, Neural Networks for Computing. J.S. Denker (ed.), American Institute of Physics, New York, NY, 241-246.
- [4] Kennedy, M.P. and L.O. Chua (1988), Neural Networks for Nonlinear Programming, IEEE Transactions on Circuits and Systems, vol. 35, no. 5, pp. 554-562.
- [5] Reklaitis, G.V., A. Ravindran and K.M. Ragsdell (1983), Engineering Optimization Methods and Applications. Wiley - Interscience.
- [6] Tank, D.W. and J.J. Hopfield (1986), Simple "Neural" Optimization Networks: An AID Converter. Signal Decision Circuit. and a Linear Programming Circuit. IEEE Transactions on circuits and systems, CAS-33, no. 5.
- [7] Tsirikis, A. G., Reklaitis, G.V., and Tenorio, M.F. (1989). Computational properties of Generalized Hopfield Networks applied to Nonlinear Optimization. Tech. Rep. IREE 89-69, School of Electrical Engineering, Purdue University.
- [8] Tsirikis, A. G., Reklaitis, G.V., and Tenorio, M.F. (1990). Nonlinear Optimization using Generalized Hopfield Networks. Neural Computation, vol. I, no. 4.