AUGMENTED LAGRANGE MULTIPLIER METHOD TO SOLVE NONLINEAR PROGRAMMING PROBLEMS: 
A NEURAL NETWORK APPROACH

W. Abdul Hameed, Kaspar S*, D. Anuradha
Department of Mathematics, School of Advanced Sciences, 
Vellore Institute of Technology, Vellore, India – 632014.
e-mails: hameedvellore@yahoo.co.in, kasparsebastian@gmail.com, danuradha@vit.ac.in

Abstract: One promising approach to handle the optimization problems with high dimensions and dense 
structure is to employ artificial neural network based on circuit implementation. The neural network approach 
can solve optimization problems in running times at the orders of magnitude much faster than conventional 
optimization algorithms executed on general-purpose digital computers. The main goal of this Paper is to 
provide several illustrations of how neural networks can be used as a computationally efficient and relatively 
simple tool for implementing the well-known nonlinear programming technique, known as, augmented 
Lagrange multiplier method. In this Paper, we consider a neural network approach of solving two kinds of the 
NP problems defined relative to the forms of the nonlinear constraints namely, neural network for the NP 
problems with equality constraints and neural network for the NP problems with inequality constraints. The 
validity and transient behavior of the proposed neural networks are demonstrated by some simulation results 
using MATLAB software.

Key Words: Nonlinear Programming Problems, Augmented Lagrange Multiplier Method, Steepest Descent 
Method, Neural Network.

1. Introduction

Problems in which both objective function and constraints may be nonlinear are referred to as nonlinear 
programming problems (NP). These problems arise in a wide variety of scientific and engineering fields 
including regression analysis, function approximation, signal processing, image restoration, parameter 
estimation, filter design, robot control, etc., and hence a real time solution is often desired. However, traditional 
umerical methods might not be efficient for digital computers since the computing time required for a solution 
is greatly dependent on the dimension and the structure of the problem and the complexity of the algorithm 
used. In this Paper, we present a neural network for solving the nonlinear programming problems in real time by 
means of Augmented Lagrange Multiplier and Steepest Descent Methods.

The structure of this Paper is as follows: In Section 1, the Steepest Descent and Augmented Lagrange Multiplier 
Methods are explained. In Section 2, the basic concepts, approach, algorithms and architecture of the 
networks are presented. In Section 3, the simulation results using MATLAB software for NP problems 
are derived. Finally, some concluding remarks are drawn in Section 4.

2. Steepest Descent and Augmented Lagrange Multiplier Methods

Steepest Descent Method
The method of Steepest Descent (often referred to as the gradient method) is one of the oldest and most widely used numerical optimization techniques for minimizing a function of several variables. If we assume $E(x)$ (an energy function) as a function of several variables $x = \begin{bmatrix} x_1, x_2, \ldots, x_n \end{bmatrix}^T$, where $x \in \mathbb{R}^{n \times 1}$, and has continuous partial derivatives on $\mathbb{R}^n$, then the gradient of $E$ with respect to the vector $x$ is given as $\nabla_x E(x) \in \mathbb{R}^{n \times 1}$. The method of Steepest Descent can be defined for the discrete-time case as

$$x_{k+1} = x_k - \mu_k \nabla_x E(x)$$

where $k$ is the discrete-time index, $x_k = x(k)$, and $\mu_k$ is a non-negative scalar that minimizes $E(x_{k+1})$. Steepest Descent searches from the point $x_k$ along the direction of the ‘negative gradient’ to a minimum point, where this minimum point is taken as $x_{k+1}$.

Augmented Lagrange Multiplier Method

The Augmented Lagrange Multiplier Method [3] is one of the most effective general approaches to solving nonlinear programming problems. It was derived independently by Hestens[4] and Powell[5]. Mathematically, the constrained optimization problems can be formulated as follows:

Minimize $f(x) = f(x_1, x_2, \ldots, x_n)$

subject to $h_i(x) = 0$

where $x \in \mathbb{R}^{n \times 1}$. The Ordinary Lagrange Multiplier Method [3] transforms the above problem to an unconstrained optimization problem. It is formulated by appending the constraints to the objective function with Lagrange Multipliers used as searching factors. The new objective function is called the Lagrangian and it is of the form

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

where $\lambda = \begin{bmatrix} \lambda_1, \lambda_2, \ldots, \lambda_m \end{bmatrix}^T \in \mathbb{R}^{m \times 1}$ denotes the vector of the Lagrange Multipliers. The Augmented Lagrangian is formed by addition of extra penalty terms. The most popular form of the Augmented Lagrangian is given by Gill, Murray and Wright [2]

$$L(x, \lambda, K) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{i=1}^{m} k_i h_i^2(x)$$

(1)

Where $K = \begin{bmatrix} k_1, k_2, \ldots, k_m \end{bmatrix}^T \in \mathbb{R}^{m \times 1}$ is the vector of positive penalty parameters. After this, the optimal solution is found as its unconstrained minimum. This can be done by using any of the unconstrained optimization techniques. In the most simplistic approach, we can use Steepest Descent method [3] where the minimization of the Lagrangian is converted to a system of difference equations of the form

$$x(k+1) = x(k) - \mu_k \frac{\partial}{\partial x} L(x, \lambda, K)$$

(2)

$$\lambda(k+1) = \lambda(k) + \mu_k \left( \frac{\partial}{\partial \lambda} L(x, \lambda, K) \right)$$

(3)

where $\mu_k, \mu_j > 0$ represent the learning rule parameters. There are several important properties of the Augmented Lagrange Multiplier Method that need to be taken into consideration:

(i) Local-minimum property: The augmented Lagrange Multiplier Method guarantees convergence to the local minimum of the Augmented Lagrangian. The local minimum of the Augmented Lagrangian converges to the constrained minimum of the objective function only in the limiting case when the penalty parameter in $K$ are sufficiently large.
(ii) Choice of the penalty parameter: In general, the penalty parameter need to be chosen so that the Hessian matrix of the augmented Lagrangian defined as

$$H = \nabla_{x}^{2} L_{A}(x, \lambda, K) = \frac{\partial^{2} L}{\partial x^{2}}$$

is positive definite. If the values of the penalty parameter are too small, the algorithm may fail to converge or it may converge to a value that is a local minimum of the Augmented Lagrangian but does not minimize the objective function itself. On the other hand, if the parameters are chosen too large, the algorithm may exhibit oscillatory behavior in the vicinity of the solution.

(iii) Convergence of the Lagrange Multipliers: For the algorithm in Eqn. 2 and Eqn. 3, to find the optimal solution it is necessary that both $x$ and $\lambda$ converge to their optimal values $x^{*}$ and $\lambda^{*}$. In some cases the augmented Lagrangian can be very sensitive to the values for the multipliers, and it may take a considerable number of iterations before convergence is achieved.

3. Basic Concepts, Approach, Algorithms and Architecture of the Neural Networks

Basic Concepts and Approach

Neural networks [6] are simplified models of the biological nervous system and therefore have drawn their motivation from the kind of computing performed by a human brain. A neural network (NN), in general, is a highly interconnected network of a large number of processing elements called neurons in an architecture inspired by the brain. An NN can be massively parallel and therefore is said to exhibit parallel distributed processing. Neural networks learn by examples. They can therefore be trained with known examples of a problem to ‘acquire’ knowledge about it. Once appropriately trained, the network can be put to effective use in solving ‘unknown’ or ‘untrained’ instances of the problem.

Neural networks adopt various learning mechanisms of which supervised learning and unsupervised learning methods have turned out to be very popular. In supervised learning, a ‘teacher’ is assumed to be present during the learning process, i.e. the network aims to minimize the error between the target (desired) output presented by the ‘teacher’ and the computed output, to achieve better performance. However, in unsupervised learning, there is no teacher present to hand over the desired output and the network therefore tries to learn by itself, organizing the input instances of the problem.

Though NN architectures have been broadly classified as single layer feed forward networks, multiplayer feed forward networks, and recurrent networks, over the years several other NN architectures have evolved. Some of the well-known NN systems include back-propagation network, perceptron, ADALINE (Adaptive Linear Element), associative memory, Boltzmann machine, adaptive resonance theory, self-organizing feature map and Hopfield network.

Neural networks [9] exhibit characteristics such as mapping capabilities or pattern association, generalization, robustness, fault tolerance, and parallel and high speed information processing. Neural networks have been successfully applied to problems in the fields of pattern recognition, image processing, data compression, forecasting, and optimization.

Neural Network for the Nonlinear Programming Problems with Equality Constraints

The standard form of the nonlinear programming problem (NP) is as follows:

Minimize $f(x) = f(x_{1}, x_{2}, \ldots, x_{n})$

subject to $h_{i}(x) = 0, \quad i = 1, 2, \ldots, m$

where $x \in \mathbb{R}^{n \times d}$ is the vector of the independent variables, $f(x) : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}$ is the objective function, and function $h_{i}(x) : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}$ represents constraints. To simplify the derivations of the algorithms, we will assume that both the objective function and the constraints as smooth differential functions of independent variables. Evaluating the gradients specified in Eqn. 2 and Eqn. 3 using Eqn. 1 yield the following update equations.
\[ x_j(k + 1) = x_j(k) - \mu_i \left( \frac{\partial f(x(k))}{\partial x_j} + \sum_{i=1}^{m} S_i(k) + k h_i(x(k)) \right) \]

\[ \lambda_j(k + 1) = \lambda_j(k) + \mu_s S_j \left( h_j(x(k)) \right) \]

where \( \mu_i, \mu_s > 0 \) represent the learning rate parameters.

**Neural Network for the Nonlinear Programming Problems with Inequality Constraints**

The nonlinear programming problem with inequality constraints is given as follows:

Minimize \( f(x) = f(x_1, x_2, \ldots, x_n) \)

subject to \( h_i(x) \leq 0, \quad i = 1, 2, \ldots, m \)

where \( x \in \mathbb{R}^n \) is the vector of the independent variables, \( f(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is the objective function, and function \( h_i(x) : \mathbb{R}^m \rightarrow \mathbb{R} \) represents constraints. The method of the Augmented Lagrange Multiplier can be extended to nonlinear programming problems with inequality constraints. To accomplish this, the augmented Lagrangian is modified according to

\[
L(x, \lambda, K) = f(x) + \sum_{i=1}^{m} \lambda_i \max \left( 0, h_i(x) \right) + \frac{k}{2} \sum_{i=1}^{m} \lambda_i h_i(x)^2 \quad (4)
\]

where \( \lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T \in \mathbb{R}^m \) denotes the vector of the Lagrange Multipliers and \( K = (k_1, k_2, \ldots, k_m)^T \in \mathbb{R}^m \) is the vector of positive penalty parameters. As can be seen from Eqn. 4, any violation of the constraints increasing the value of the Lagrange, that is, only violated constraints are considered to be active. In a more compact form, Eqn. 4 can be written as

\[
L(x, \lambda, K) = f(x) + \sum_{i=1}^{m} S_i \left[ \lambda_i h_i(x) + \frac{k_i}{2} h_i(x)^2 \right] \quad (5)
\]

where

\[
S_i = \begin{cases} 0, & \text{if } h_i(x) \leq 0 \\ 1, & \text{if } h_i(x) > 0 \end{cases}
\]

After substitution of the appropriate gradients of Eqn. 5 into Eqn. 2 and Eqn. 3, we have

\[ x_j(k + 1) = x_j(k) - \mu_i \left( \frac{\partial f(x(k))}{\partial x_j} + \sum_{i=1}^{m} S_i(k) + k h_i(x(k)) \right) \]

\[ \lambda_j(k + 1) = \lambda_j(k) + \mu_s S_j \left( h_j(x(k)) \right) \]

**4. Simulation**

In order to verify the feasibility and efficiency of the above discrete-time neural networks, four examples are carried out using MATLAB software [8] to solve the NP problems in standard form and as well as the problems with inequality constraints.

**Example - 1**
Consider the following NP problem in standard form taken from Richard Bronson and Govindasami Nadimuthu [7]

Maximize \( f(x) = x_1^2 + 2x_2^2 + x_3^2 + x_1x_2 + x_1x_3 \)

subject to
\[
\begin{align*}
x_1^2 + x_2^2 + x_3^2 &= 25 \\
8x_1 + 14x_2 + 7x_3 &= 56
\end{align*}
\]

Initial conditions are assumed as \( x = (1, 1, 1) \) and the Lagrange multiplier, \( \lambda = (1, 1) \). The learning rate parameter is chosen as \( \mu = 0.001 \) and the penalty parameter, \( K = 2 \). The Figure-1 shows the trajectories for each of the independent variables. The neural network converges after 2000 iterations. The optimal solution to the NP problem is given as \( x = [3.5085, 0.2047, 3.5656] \) and \( f(x) = 38.3349 \) but correspondingly in the classical Lagrangian Method optimal solutions \( x^* = [3.512, 0.217, 3.55] \) and \( f(x^*) = 38.28 \)

Example - 2

Consider the following NP problem in standard form taken from Richard Bronson and Govindasami Nadimuthu [7]

Minimize \( f(x) = x_1^2 + x_2^2 + x_3^2 \)

subject to
\[
\begin{align*}
x_1x_2x_3 &= 3 \\
x_1 + x_2x_3 &= 3
\end{align*}
\]

Initial conditions are assumed as \( x = (1, 1, 1) \) and the Lagrange multiplier, \( \lambda = (1, 1) \). The learning rate parameter is chosen as \( \mu = 0.001 \) and the penalty parameter, \( K = 55 \). The Figure-2 shows the trajectories for each of the independent variables. The neural network converges after 20 iterations. The optimal solution to the NP problem is given as \( x = [1.8993, 1.8993, 0.8271] \) and \( f(x) = 7.898 \) but correspondingly in the classical Lagrangian Method optimal solutions \( x^* = [1.911, 1.911, 0.822] \) and \( f(x^*) = 7.980 \)

Example - 3

Consider the following NP problem with inequality constraints taken from Gass[1]

Minimize \( f(x) = 2x_1 + 3x_2 - (x_1^2 + x_2^2 + x_3^2) \)

subject to
\[
\begin{align*}
x_1 + x_2 &\leq 1 \\
2x_1 + 3x_3 &\leq 6
\end{align*}
\]

Initial conditions are assumed as \( x = (0, 0, 0.5) \) and the Lagrange multiplier, \( \lambda = (1, 1) \). Learning rate parameter is chosen as \( \mu = 0.001 \) and the penalty parameter, \( K = 650 \). The Figure-3 shows the trajectories for each of the independent variables. The neural network converges towards optimal solution after 2500 iterations. The optimal solution is given as \( x = [0.2504, 0.7504, 0] \) and \( f(x) = 2.1261 \), but correspondingly in the classical Lagrangian Method optimal solutions \( x^* = [0.25, 0.75, 0] \) and \( f(x^*) = 2.125 \)

Example - 4

Consider the following NP problem with inequality constraints taken from Sharma[10]

Maximize \( f(x) = 2x_1 + 3x_2 \)

subject to
\[
\begin{align*}
x_1^2 + x_2^2 &\leq 20 \\
x_1x_3 &\leq 8
\end{align*}
\]

Zero initial conditions are assumed for \( x \) and the Lagrange multiplier, \( \lambda \). Learning rate parameter is chosen as \( \mu = 0.001 \) and the penalty parameter, \( K = 194 \). The Figure-4 shows the trajectories for each of the independent variables. As can be seen the neural network converges towards optimal solution after 1600 iterations. The
optimal solution is given as $x = [2.0007, 4.0006]$ and $f(x) = 16.0034$, but correspondingly in the classical Lagrangian Method optimal solutions $x^* = (2, 4)$ and $f(x^*) = 16$.

5. CONCLUSION

In this paper, we have shown how the neural network technique is applied to solve nonlinear programming problems in standard form and as well as with inequality constraints. We have presented the graph to understand the updating process of the neural network architecture and trajectories of the independent variables.

The neural algorithms proposed in this paper have several advantages over the conventional parallel algorithms. First, the neural algorithms are much simpler than others. The only algebraic operations required in the neural algorithms are addition and multiplication. Inverse and other complicated logic operations are not needed. Second, the neural algorithms are the most parallel, compared with conventional parallel algorithms. Simplicity in implementation of the neural algorithms is the third advantage.

![Figure 1](image1.png)

Figure 1: Trajectories of the solution for the NP problem with equality constraints in Example 1

![Figure 2](image2.png)

Figure 2: Trajectories of the solution for the NP problem with inequality constraints in Example 1
Figure 2: Trajectories of the solution for the NP problem with equality constraints in Example-2

Figure 3: Trajectories of the solution for the NP problem with inequality constraints in Example-3

Figure 4: Trajectories of the solution for the NP problem with inequality constraints in Example-4

References


