Applications of the General Nonlinear Neural Networks in Solving the Inverse Optimal Value Problem with Linear Constraints

1Huaiqin Wu, 1Kewang Wang, 1Ning Li, 1Chongyang Wu, 1Qiangqiang Guo and 1Guohua Xu
1Department of Applied Mathematics, Yanshan University, Qinhuangdao 066004, China
2College of Economics and Finance, Huqiao University, Quanzhou 362021, China

Abstract: In this research, the aim of the study is to present a method to solve a class of inverse optimal value problem with linear constraints by using a nonlinear gradient neural network. Firstly, based on optimal theory, solving the inverse optimal value problem is changed as solving a nonlinear bilevel program problem equivalently. Then a nonlinear gradient neural network model for solving the nonlinear bilevel programming is presented. By employing Lyapunov function approach, the proposed neural network is analyzed to be globally Lyapunov stable and capable of generating approximate optimal solution to the nonlinear bilevel programming problem. Finally, numerical examples are provided to verify the feasibility and the efficiency of the proposed method in this study.

Key words: Neural networks, bilevel programming, inverse optimal value problem, global asymptotical stability, fish-burmeister function

INTRODUCTION

In the recent years, the inverse optimal value problems have received extensive attention from a lot of scholars (Tarantola, 1987; Burton and Toint, 1994; Zhang et al., 1995, 1996, Sokkalingam et al., 1999; Zhang and Liu, 1996, 1999; Yang et al., 1997; Zhang and Cai, 1998; Aluja and Orlin, 2000, 2001, 2002; Heuberger, 2004; Ahmed and Guan, 2005; Lv et al., 2008a, 2010a). Let \( \min p(x, c), x \in X \) be an optimization problem where \( X \) is the feasible region and \( c \) is an parameter vector representing costs, capacities, weights, returns, etc. The general optimization problem (also called forward optimization) is to find an \( x^* \in X \) such that the objective \( p(x, c) \) is optimal at \( x^* \). The inverse optimization problem can be described as follows. Assume that it has been known that an \( \hat{x} \) and a vector \( \hat{c} \) which is an estimate to the parameter values of the problem. We are interested in finding out whether there exist acceptable values of the parameter vector \( c \), denoted by \( \hat{c} \) which make \( \hat{x} \) optimal to problem \( \min p(x, \hat{c}) \). If the answer is positive, find the vector \( \hat{c} \) that differs from the vector \( c \) as little as possible.

Burton and Toint (1994) first investigated an inverse shortest paths problem. Since, then, many different inverse optimization problems (discrete or continuous) have been considered. Zhang et al. (1996) addressed inverse shortest paths problems with \( l \) norm for \( p \) paths using a column generation method. Zhang et al. (1996) and Sokkalingam et al. (1999) worked on inverse minimum spanning tree problems.


It should be noted that in the existing literature, almost all results is to focus mainly on the study of the classical numerical algorithm for the inverse optimal value problem. However, in lots of engineering applications many optimization problems need to be solved in real time. It is obvious that the classical methods cannot render real-time solutions to these optimization problems, especially large-scale problems. Compared with classical optimization approaches, the appearance of neural computing approach satisfies the demand of real-time optimal solutions. The main advantage of neural network approach to optimization is that the nature of the dynamic solution procedure is inherently parallel and distributed. Therefore, the neural network approach can solve optimization problems in running time at the orders
of magnitude much faster than the most popular optimization algorithms executed on general purpose digital computers. Recently, there have been some works to focus on solving varied optimization problems in engineering by using neural networks, the reader may consult (Lv et al., 2010b; Gao, 2004; Xia and Wang, 1998; Chen et al., 2002; Wu et al., 2010; Al-Bastaki, 2006; Dempe, 2002; Ljilja et al., 2011; Yedjour et al., 2011; Salazar et al., 2010; Arafà et al., 2011; Niknafs and Parsa, 2011; Lotfi and Benyettou, 2011; Liu et al., 2012) and the references therein. To the best of the authors' knowledge, solving the inverse optimal value problems by neural networks has not been investigated to date which inspires us to carry out the present study.

Motivated by the above discussion, in the present paper, a new nonlinear neural networks for solving the inverse optimal value problem with linear constraints is presented. Firstly, the inverse value problem with linear constraints is transformed into the corresponding nonlinear bilevel programming problem; Secondly, a nonlinear neural network model for solving the bilevel programming is developed. The equilibrium point of the proposed neural network is equivalent to the solution of the inverse optimal value problem. Thirdly, a new sufficient condition to ensure the global stability for the proposed neural network is given, by the use of the Lyapunov function method. Finally, numerical examples are provided to show the effectiveness of the proposed approach.

PROBLEM STATEMENT AND PRELIMINARIES

Consider the following linear program problem:

\[
\begin{align*}
\min & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad x \geq 0
\end{align*}
\]  

(1)

where, \( c, x \in \mathbb{R}^n; A \in \mathbb{R}^{m \times n}; b \in \mathbb{R}^n \). Given a set \( C \subseteq \mathbb{R}^n \) and a real number \( z^* \). Arguing as in Tarantola (1987), the inverse optimal value problem corresponding to Eq. 1 is to find \( c \in C \) such that the optimal objective value \( Q(c) \) of the linear programming Eq. 1 is closed to \( z^* \). It is obvious that the inverse optimal value problem can be written as:

\[
\min_{c \in C} f(c)
\]

(2)

\[
\text{subject to } \quad c \in C
\]

where, \( f(c) = ||Q(c) - z^*|| \), \( || \cdot || \) denotes the Euclidean norm of \( \mathbb{R}^n \).

Combining Eq. 1 and 2, the inverse optimal value problem corresponding the linear program can be described as follows:

\[
\begin{align*}
\text{(UP)} \min & \quad f(c) \\
& \quad \text{subject to } \quad c \in C \\
\text{(LP)} \min & \quad c^T x \\
& \quad \text{subject to } \quad Ax \leq b \\
& \quad x \geq 0.
\end{align*}
\]

(3)

The model Eq. 3 is a class of the bilevel program problem with the optimal value of the lower level problem feeding back to the upper level. (UP) is called the upper level problem and (LP) is called the lower level problem.

Throughout, the rest of this study, the following assumptions are made:

- **H1**: The set of cost vectors \( C \) is nonempty, \( C = \{c: Dc = d\}, D \in \mathbb{R}^{m \times n}, d \in \mathbb{R}^n \)

- **H2**: The feasible region \( \{x: Ax \leq b, x \geq 0\} \) is nonempty bound

Without losing generality, replacing \( f(c) = ||Q(c) - z^*|| \) with \( f(c) = (Q(c) - z^*)^T (Q(c) - z^*) \), then the bilevel program problem Eq. 3 can be rewritten as:

\[
\begin{align*}
\text{(UP)} \min & \quad (Q(c) - z^*)^T (Q(c) - z^*) \\
& \quad \text{subject to } \quad Dc = d \\
\text{(LP)} \min & \quad c^T x \\
& \quad \text{subject to } \quad Ax \leq b \\
& \quad x \geq 0.
\end{align*}
\]

(4)

For fixed \( c \in C \), by using the Kuhn-Tucker optimality conditions to replace (LP), the problem Eq. 4 can be changed equivalently as the following program problem:

\[
\begin{align*}
\min & \quad (c^T x - z^*)^T \\
& \quad \text{subject to } \quad Dc = d \\
& \quad Ax \leq b \\
& \quad uA - v = -c^T \\
& \quad u(b - Ax) + v x = 0 \\
& \quad u, v, x \geq 0.
\end{align*}
\]

(5)

where, \( u \in \mathbb{R}^m \), \( v \in \mathbb{R}^m \) are row vectors. It is obvious that the problem (Eq. 5) is a convex program problem.

From the above discussion, it is easy to see that the inverse optimal value problem corresponding to the linear program (Eq. 1) is equivalent to a special class of the convex program (Eq. 5). Hence, solving the inverse optimal value problem corresponding to the linear program (Eq. 1) can be changed as solving the convex program (Eq. 5).

In order to obtain the main results of this study, the following definitions and lemmas are introduced.
\textbf{Definition 1}: The Fish-Burmeister function is \( \Phi: \mathbb{R}^2 \to \mathbb{R} \) defined by \( \Phi(a, b) = a + b - \sqrt{a^2 + b^2} \) and the perturbed Fish-Burmeister function is \( \Phi: \mathbb{R}^3 \to \mathbb{R} \) defined by \( \Phi(a, b, c) = a + b - \sqrt{a^2 + b^2} + c \).

The Fish-Burmeister function \( \Phi(a, b) \) has the property: \( \Phi(a, b) = -a \geq 0, b \geq 0, ab = -\frac{e}{2} \) if and only if:

\[
a \geq 0, \ b \geq 0, \ ab = -\frac{e}{2}
\]

for \( e \geq 0 \). It is easy to see that \( \Phi(a, b) \) is non-differentiable at \( a = b = 0 \). However, \( \Phi(a, b, c) \) is smooth with respect to \( a, b \) for \( e > 0 \).

Let \( \Phi(a, b, c) = \sqrt{a^2 + b^2 + c - a - b} \). It is obvious that \( \Phi(a, b, c) \) has the same property with the function \( \Phi(a, b, c) \). By using \( \Phi(a, b, c) \), the problem (Eq. 5) can be approximated by:

\[
\begin{align*}
\min & \ (c^T x - z)^2 \\
\text{subject to} & \ uA - v + c^T = 0 \\
& \sqrt{h_i^2 + u_i^2 + c - h_i - u_i} = 0, i = 1, \ldots, p \\
& \sqrt{v_j^2 + x_j^2 + c - v_j - x_j} = 0, j = 1, \ldots, n \\
& Dc = d.
\end{align*}
\]

where \( h(x) = (h_1(x), \ldots, h_n(x))^T = b - Ax \).

To simplify the discussion, set:

\[
\begin{align*}
F(c, u, v, x) &= (c^T x - z)^2 \\
H(c, u, v, x) &= \begin{cases} \\
uA - v + c^T \\
\Phi(h(x), u, c), i = 1, \ldots, p \\
\Phi(v, x, c), j = 1, \ldots, n \\
Dc = d
\end{cases}
\end{align*}
\]

Then the program problem (Eq. 6) can be written as:

\[
\begin{align*}
\min & \ y^T \\
\text{subject to} & \ H(y) = 0, 1 = 1, \ldots, p + 2n + m
\end{align*}
\]

\textbf{Definition 2}: Let \( \{y^j\} \) be a feasible point of the program problem (Eq. 7). \( y \) is said to be a regular point if the gradients \( \nabla H(y) \), \( \cdots, \nabla H_{\text{trim}}(y) \) are linearly independent.

By using theorem in Dempe (2002) the following lemma can be obtained.

\textbf{Lemma 1}: Let \( \{y^j\} \) be a sequence of solution of the program problem (Eq. 7). Suppose that the sequence \( \{y^j\} \) converges to some \( y \) for \( e \to 0^+ \). If \( y \) is a regular point, then \( y \) is a Bouligand stationary solution for problem (Eq. 4).

\textbf{NEURAL NETWORK MODEL FOR THE PROGRAM PROBLEM}

\textbf{Neural network model}: Define the Lagrange function of the program problem (Eq. 7) as \( L(y, \lambda) = F(y) + \lambda H(y) \), where \( \lambda \in \mathbb{R}^{2n+m} \) is referred as the Lagrange multiplier. Based on the well-known Saddle point theorem, we have.

\textbf{Theorem 1}: If there exists \( (y^*, \lambda^*) \), such that \( \Delta_L(y^*, \lambda) = 0, \ H(y^*) \) hold. Then, \( y^* \) is an optimal solution of the program problem (Eq. 7).

By Theorem 1, the energy function of the program problem (Eq. 7) can be constructed as:

\[
E(y, \lambda) = \frac{1}{2} \left( \nabla yL(y, \lambda) \right)^T + \frac{1}{2} \left( \nabla yL(y, \lambda) \right)
\]

in the following, a neural network model is proposed which is said to be the gradient network, for solving the problem (7), whose dynamical equation is described as follows:

\[
\frac{dy}{dt} = -\nabla yE(y, \lambda), \frac{d\lambda}{dt} = -\nabla yE(y, \lambda)
\]

That is:

\[
\frac{dy}{dt} = -\nabla yL(y, \lambda), \frac{d\lambda}{dt} = -\nabla yL(y, \lambda).
\]

According to Theorem 1, it is easy to obtain.

\textbf{Theorem 2}: \( (y^*, \lambda^*) \) is an equilibrium point of the neural network (Eq. 8) if and only if \( y^* \) is an optimal solution of the program problem (Eq. 7).

\textbf{Proof}: Firstly, if \( (y^*, \lambda^*) \) is an equilibrium point of the neural network (Eq. 8), that means:

\[
E(y^*, \lambda^*) = \frac{1}{2} \left( \nabla yL(y^*, \lambda^*) \right)^T + \frac{1}{2} \left( \nabla yL(y^*, \lambda^*) \right) = 0,
\]

i.e., \( \nabla yL(y^*, \lambda^*) = 0 \) and \( \nabla yL(y^*, \lambda^*) = 0 \).

That is \( \nabla yL(y^*, \lambda^*) = 0 \) and \( H(y^*) = 0 \).

By theorem 1, we have that \( y^* \) is an optimal solution of program problem (Eq. 7).
Conversely, if \( y^* \) is an optimal solution of program problem (Eq. 7), from K-T condition we have that there exist \( \lambda^* \), such that \( V_y(y^*, \lambda^*) = 0 \) and \( V_y(y^*, \lambda^*) = 0 \). i.e., \( V_y(y^*, \lambda^*) = 0 \) and \( H(y^*) = 0 \). Then, the equations -\( \nabla_y L(y^*, \lambda) \), \( \nabla_L L(y^*, \lambda) = \nabla^2 (y^*) \), \( \nabla^2 (y^*) = 0 \) and -\( \nabla_y L(y^*, \lambda) \). \( -\nabla_y L(y^*, \lambda) = 0 \) are existence. This means \( (y^*, \lambda^*) \) is an equilibrium point of the neural network (Eq. 8).

**Stability analysis:** Since, the program problem (Eq. 7) is a class of convex programs, there exists an unique optimal solution. Hence, by Theorem 2, the neural network (Eq. 8) has an equilibrium point.

**Theorem 3:** If \( (y^*, \lambda^*) \) is the equilibrium point of the neural network (Eq. 8), then \( (y^*, \lambda^*) \) is globally asymptotically stable.

**Proof:** Let \( \eta = (y, \lambda) \). Consider Lyapunov function:

\[
V(\eta) = E(\eta) - E(\eta^*) = E(\eta)
\]

It is obvious that \( V(\eta) \geq 0, \forall \eta - \eta^* \) and \( V(\eta^*) = 0 \). Calculate the derivative of \( V(\eta) \) along the solution \( \eta(t) \) of the neural network (Eq. 8), we have:

\[
\frac{d}{dt} V(\eta) = \nabla V(\eta) \cdot \frac{d\eta}{dt} = -\| \nabla E(\eta) \|^2 < 0, \eta = \eta^*
\]

Based on the Lyapunov stability theory, we can obtain that \( \eta^* \) is globally asymptotically stable.

**ILLUSTRATIVE EXAMPLES**

Here, three examples will be given to illustrate the effectiveness of the proposed approach for solving the inverse optimal value problem.

**Example 1:** Consider the following inverse optimal value problem:

\[
\begin{align*}
\text{(UP) } & \min_{\nu, x_i} (c_i x_i + c_j x_j - 14)^2 \\
\text{subject to } & \quad c_i + 2c_j = 8.
\end{align*}
\]

\[
\begin{align*}
\text{(LP) } & \max_{\nu, x_i} \sum_{i=1}^n c_i x_i \\
\text{subject to } & \quad x_i + 2x_j \leq 8 \\
& \quad 0 \leq x_i \leq 4 \\
& \quad 0 \leq x_j \leq 3.
\end{align*}
\]

Let \( z^* = 14 \). The problem (Eq. 6) corresponding to (Eq. 9) is:

**Example 2:** Consider the following inverse optimal value problem:

\[
\begin{align*}
\text{(UP) } & \min_{\nu, x_i} (c_i x_i + c_j x_j - 14)^2 \\
\text{subject to } & \quad c_i + c_j = 3. \\
\text{(LP) } & \max_{\nu, x_i} \sum_{i=1}^n c_i x_i \\
\text{subject to } & \quad 4x_i + 3x_j \leq 12 \\
& \quad 4x_i + 8x_j \leq 8 \\
& \quad x_i, x_j \geq 0.
\end{align*}
\]

By using the classical fourth-order Runge-Kutta method in MATLAB to solving the neural network (Eq. 8), we can obtain the transient behavior of the state trajectories \( (x_1, x_2, c_i, c_j) \) (Fig. 1). Take the initial values \( y = (3, 5, 1, 3, 2, 0, 0, 0, 0, 0, 0) \) and \( \lambda = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \). Let \( \epsilon \) be 0.01, 0.001 and 0.00001, respectively; we can get the different optimal solutions of the problem (10) (Table 1). It is easy to see that optimal solution \( (x^*, y^*, c^*, c_j^*) \) of the problem (10) converges to the solution (4, 2, 2, 3) of the inverse optimal value problem (9).

**Example 2:** Consider the following inverse optimal value problem:

\[
\begin{align*}
\text{(UP) } & \min_{\nu, x_i} (c_i x_i + c_j x_j - 14)^2 \\
\text{subject to } & \quad c_i + c_j = 3. \\
\text{(LP) } & \max_{\nu, x_i} \sum_{i=1}^n c_i x_i \\
\text{subject to } & \quad 4x_i + 3x_j \leq 12 \\
& \quad 4x_i + 8x_j \leq 8 \\
& \quad x_i, x_j \geq 0.
\end{align*}
\]
Table 1: The comparison of the optimal solution of Example 1 and 2

<table>
<thead>
<tr>
<th>ε</th>
<th>x^1_0, x^2_0, e^1_0, e^2_0</th>
<th>x^1_0, x^2_0, e^1_0, e^2_0</th>
<th>x^1_0, x^2_0, e^1_0, e^2_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε1</td>
<td>(3.9902, 2.0032, 2.0634, 2.0983)</td>
<td>(3.9909, 2.0003, 2.0004, 2.9998)</td>
<td>(3.9999, 2.0000, 2.0001, 2.9999)</td>
</tr>
<tr>
<td>ε2</td>
<td>(1.4943, 2.0013, 1.9781, 1.0981)</td>
<td>(1.4995, 2.0000, 1.9978, 1.0922)</td>
<td>(1.4999, 2.0000, 1.9996, 1.0004)</td>
</tr>
</tbody>
</table>

Different optimal solution (x^1_0, x^2_0, e^1_0, e^2_0) corresponding to different ε

![Fig. 2: The transient behavior of the variables in Example 2](image)

Table 2: The comparison of the optimal solution of Example 3

<table>
<thead>
<tr>
<th>ε</th>
<th>x^1_0, x^2_0, e^1_0, e^2_0</th>
<th>x^1_0, x^2_0, e^1_0, e^2_0</th>
<th>x^1_0, x^2_0, e^1_0, e^2_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε1</td>
<td>(3.2431, 2.1355, 1.4836, 1.0285, 5.9805, 4.0966)</td>
<td>(3.2382, 2.1428, 1.4761, 1.0074, 5.9818, 4.0107)</td>
<td>(3.2376, 2.1438, 1.4750, 1.0069, 5.9819, 4.0111)</td>
</tr>
</tbody>
</table>

Different optimal solution (x^1_0, x^2_0, e^1_0, e^2_0) corresponding to different ε

![Fig. 3: The transient behavior of the variables in Example 3](image)

Using the same method of Example 1, we can obtain the transient behavior of the state trajectories (x_i, x_j, c_i, c_j) (Fig. 2). Take the initial values y = (1, 2, 2, 1, 0, 0, 0, 0) and λ = (0, 0, 0, 0, 0, 0, 0, 0). Let ε be 0.01, 0.001 and 0.0001 respectively, we can get the different optimal solutions of the problem (11) (Table 1). It is easy to see that optimal solution (x^*_1, x^*_2, e^*_1, e^*_2) of the problem (11) converges to the solution (1.5, 2, 2, 1).

**Example 3:** Consider the following inverse optimal value problem:

![LaTeX equation](image)

Take the initial values y = (2, 0, 3, 3, 3, 4, 0, 0, 0, 0, 0, 0) and λ = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0). Let ε be 0.01, 0.001 and 0.0001 respectively, we can get the different optimal solutions of the problem (12) (Table 2). The transient behavior of the state trajectories (x_i, x_j, x_k, c_i, c_j, c_k) (Fig. 3).

Table 2 the comparison of the optimal solution of example 3.

From the above examples we can find that the computed results converge to the different optimal solution with the decreasing of ε. It shows that the neural network approach is feasible to the inverse optimal value problem.

**CONCLUSION**

In this study, a neural network model has been developed for solving the inverse optimal value problem with linear constraints. Based on Lyapunov stability theory, the proposed neural network has been proved to be globally asymptotically stable and capable of generating approximal optimal solution of the inverse optimal value problem. Three examples have been given to show the effectiveness of the proposed approach. The results obtained in this paper are highly valuable in both theory and practice for solving the inverse optimal value problem.

**ACKNOWLEDGMENT**

The authors are extremely grateful to the Editor and the Reviewers for their valuable comments and suggestions which help to enrich the content and improve the presentation of this study. This study was supported.
by the Natural Science Foundation of Hebei Province of China (A201203103) and the Hebei Province Education Foundation of China (2009157).

REFERENCES


