PERGAMON

AVAILABLE AT
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT

Neural Networks 16 (2003) 1059–1074

Neural Networks

www.elsevier.com/locate/neunet

Converting general nonlinear programming problems into separable programming problems with feedforward neural networks

Bao-Liang Lu[a,*], Koji Ito[b,1]

[a]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, 1954 Hua Shan Road, 200030 Shanghai, People's Republic of China*

[b]*Department of Computational Intelligence and System Science, Tokyo Institute of Technology, 4259, Nagatsuta, Midori-ku, Yokohama 226-8502, Japan*

Received 1 March 2001; revised 11 November 2002; accepted 11 November 2002

Abstract

In this paper we present a method for converting general nonlinear programming (NLP) problems into separable programming (SP) problems by using feedforward neural networks (FNNs). The basic idea behind the method is to use two useful features of FNNs: their ability to approximate arbitrary continuous nonlinear functions with a desired degree of accuracy and their ability to express nonlinear functions in terms of parameterized compositions of functions of single variables. According to these two features, any nonseparable objective functions and/or constraints in NLP problems can be approximately expressed as separable functions with FNNs. Therefore, any NLP problems can be converted into SP problems. The proposed method has three prominent features. (a) It is more general than existing transformation techniques; (b) it can be used to formulate optimization problems as SP problems even when their precise analytic objective function and/or constraints are unknown; (c) the SP problems obtained by the proposed method may highly facilitate the selection of grid points for piecewise linear approximation of nonlinear functions. We analyze the computational complexity of the proposed method and compare it with an existing transformation approach. We also present several examples to demonstrate the method and the performance of the simplex method with the restricted basis entry rule for solving SP problems.

Keywords: Nonlinear programming; Separable programming; Feedforward neural network; Piecewise linear approximation; Separable function; Simplex method

1. Introduction

Consider the following *nonlinear programming* (NLP) problem (Luenberger, 1989)

$$\text{Minimize} \quad t(\boldsymbol{X}) \quad \text{for } \boldsymbol{X} \in \mathbf{R}^n \tag{1}$$

$$\text{Subject to} \begin{cases} g_i(\mathbf{x}) \geq 0 & \text{for } i = 1, 2, \ldots, m, \\ h_i(\mathbf{x}) = 0 & \text{for } j = 1, 2, \ldots, s. \end{cases}$$

where $t(\mathbf{x})$ is called the objective function, $g_i(\mathbf{x})$ is called the inequality constraint and $h_j(\mathbf{x})$ is called the equality constraint.

NLP problems are widespread in the mathematical modeling of engineering design problems such as VLSI chip design, optimal plastic design of structures, and optimum design of chemical processing equipment and plants (Rao, 1996). For general NLP problems, however, computer programs are not available for the problems of very large size. For a class of NLP problems known as separable (Bazaraa, Sherali, & Shetty, 1993; Miller, 1963), some variation of the simplex method, a well-developed and efficient method for solving large-scale *linear programming* (LP) problems, can be used as a solution procedure. A *separable programming* (SP) problem refers to an NLP problem where both the objective function and the constraints can be expressed as the sum of single-variable functions, i.e. the sum of separable functions. An SP

* Corresponding author. Tel.: +86-21-6293-2287; fax: +86-21-6293-2406.

E-mail addresses: blu@cs.sjtu.edu.cn (B.-L. Lu), ito@dis.titech.ac.jp (K. Ito).

[1] Tel.: +81-45-924-5655; fax: +81-45-924-5655.

0893-6080/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0893-6080(02)00234-4

problem can be stated as

$$\text{Minimize} \quad \sum_{k=1}^{n} t_k(x_k) \tag{2}$$

$$\text{subject to} \begin{cases} \sum_{k=1}^{n} g_{ik}(x_k) \geq 0 & \text{for } i = 1, 2, \ldots, m, \\ \sum_{k=1}^{n} h_{ik}(x_k) = 0 & \text{for } j = 1, 2, \ldots, s. \end{cases}$$

where both the objective function and the constraints are separable functions.

The assumption of separability of the objective function and the constraints in SP problems is a serious restriction in practical optimization problems. In order to generalize the simplex method to solve general NLP problems, we need to convert nonseparable objective function and/or the constraints into separable forms. The existing approach to transforming nonseparable functions into separable functions is to use appropriate substitutions (Rao, 1996). Unfortunately, this approach cannot be used to convert nonseparable functions whose concise expression are unknown and some given nonseparable functions.

In this paper we propose an alternative method for converting general NLP problems into SP problems by using feedforward neural networks (FNNs) (Lu & Ito, 1997). The basic idea behind the method is to use two useful features of FNNs: their ability to approximate arbitrary continuous nonlinear functions with a desired degree of accuracy and their ability to express nonlinear function in terms of parameterized compositions of functions of single variables. According to these two features, the nonseparable objective functions and/or the nonseparable constraints in NLP problems can be approximately expressed as separable functions with FNNs, and therefore, any NLP problems can be transformed into SP problems. The proposed method has three prominent features. (a) It is more general than the existing transformation technique; (b) it can be used to formulate optimization problems as SP problems even when their precise analytic objective function and/or constraints are unknown; (c) the SP problems obtained by the proposed method may highly facilitate the selection of grid points for piecewise linear approximation of nonlinear functions.

The remainder of this paper is organized as follows: Section 2 introduces the idea of approximately expressing nonseparable functions as separable ones by using a class of FNNs, Section 3 presents a method for transforming nonseparable NLP problems into SP problems, Section 4 discusses the piecewise approximation of separable functions and analyze the complexity of approximation LP problems, Section 5 compares the proposed method with the existing transformation approach, Section 6 presents four examples to demonstrate the method and the performance of the simplex method with the restricted entry rule for solving large-scale SP problems, and finally Section 7 gives the conclusions.

2. Transformation of nonseparable functions

In this section we show that a class of FNNs including multilayer perceptrons (MLPs) (Haykin, 1999), radial basis function (RBF) neural networks (Bishop, 1995), and multilayer quadratic perceptrons (MLQPs) (Lu, Bai, Kita, & Nishikawa, 1993), can be used to transform nonseparable functions into separable forms.

2.1. Normalization

Let T be the training set for approximating a nonseparable function in an NLP problem

$$T = \{(x_i, y_i)\}_{i=1}^{M} \tag{3}$$

where $x_i \in \mathbf{R}^n$ is the ith training input vector, $y_i \in \mathbf{R}$ is the ith desired output, and M is the number of training data.

Without loss of generality and for simplicity of description, we assume that the same ranges, ($X_{\min}, X_{\max}$,) and ($y_{\min}$, $y_{\max}$), are respectively used to normalize the training inputs and desired outputs for all of the nonseparable functions in an NLP problem. Before learning, both training inputs and desired outputs are usually scaled to some ranges because the original training inputs and desired outputs may contain both small and large values. The original training inputs for a nonseparable function can be normalized according to the following formula (Smith, 1993)

$$\bar{x}_{ik} = X_{\min} + \frac{x_{ik} - x_{k,\min}}{x_{k,\max} - x_{k,\min}}(X_{\max} - X_{\min}) = \alpha_k + \beta_k x_{ik}, \tag{4}$$

where $x_{ik} \in \mathbf{R}$ is the kth element of the ith training input, $x_{k,\min} = \min\{x_{ik} | i = 1, 2, \ldots, M\}$, $x_{k,\max} = \max\{x_{ik} | i = 1, 2, \ldots, M\}$, $\bar{x}_{ik}$ is the kth element of the ith normalized training input, $\alpha_k = X_{\min}^k - x_{k,\min}(X_{\max} - X_{\min})/(x_{k,\max} - x_{k,\min})$, and $\beta_k = (X_{\max} - X_{\min})/(x_{k,\max} - x_{k,\min})$.

Following the same way mentioned above, the original desired outputs can be normalized by

$$\bar{y}_i = Y_{\min} + \frac{y_i - y_{\min}}{y_{\max} - y_{\min}}(Y_{\max} - Y_{\min}) = \phi + \psi y_i, \tag{5}$$

where $y_i \in \mathbf{R}$ is the ith desired output, $y_{\min} = \min\{y_i | i = 1, 2, \ldots, M\}$, $y_{\max} = \max\{y_i | i = 1, 2, \ldots, M\}$, $\bar{y}_i$ is the ith normalized desired output, $\phi = Y_{\min} - y_{min}(Y_{\max} - Y_{\min})/(y_{\max} - y_{\min})$, and $\psi = (Y_{\max} - Y_{\min})/(y_{\max} - y_{\min})$.

2.2. Approximations of nonseparable functions

It is known that multilayer FNNs such as three-layer MLPs can be trained to approximate arbitrary continuous functions with a desired degree of accuracy if a sufficient number of hidden units is provided (Cybenko, 1989; Funahashi, 1989; Hornik, Stinchcombe, & White, 1989).

Two kinds of function approximation problems are considered in this paper: (a) approximation of an unknown function from training data, and (b) approximation of a given function which has a precise analytic expression.

Let $s(\mathbf{x})$ be a unknown function or a given nonseperable function. Suppose a set of training data as defined by Eq. (3) are obtained from the environment of the unknown function $s(\mathbf{x})$ or sampled over the given function $s(\mathbf{x})$. Also suppose that before learning the training inputs and desired outputs are normalized by Eqs. (4) and (5), respectively. By Using the function approximation capability FNNs, we can train a multilayer feedforward network to approximate $s(\mathbf{x})$. After successfully learning, $\tilde{s}(\mathbf{x})$, the approximation of $s(\mathbf{x})$, can be expressed as follows

$$s(\mathbf{x}) \approx \tilde{s}(\mathbf{x}) = \frac{\mathscr{F}(\boldsymbol{\alpha} + \boldsymbol{\beta}\mathbf{x}) - \phi}{\psi} \tag{6}$$

where $\mathscr{F}(\cdot)$ denotes the forward mapping realized by the trained multilayer FNN, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_n]$, ϕ, and ψ, are constants which are determined by Eqs. (4) and (5). In the following, we will discuss how to convert nonseparable functions into separable functions by using MLPs, RBF networks, and MLQPs.

2.2.1. Multilayer perceptrons

For simplicity of description, we consider only three-layer MLPs. The results can be extended to L-layer ($L > 3$) MLPs easily. After successfully learning, the forward mapping $\mathscr{F}(\hat{x})$ realized by a three-layer MLP can be expressed as follows

$$\mathscr{F}(\hat{x}) = f\left(\sum_{j=1}^{N_2} w_{31j} f\left(\sum_{i=1}^{N_1} w_{2ji}\hat{x}_i + \theta_{2j}\right) + \theta_{31}\right) \tag{7}$$

where $\hat{x}_i$ is the input of the ith unit in the input layer, w_{kji} the weight connecting the ith unit in the layer $(k-1)$ to the jth unit in the layer k, θ_{kj} is the bias of the jth unit in the layer k, f is the sigmoidal activation function, and N_k is the number of units in the layer k, and $k = 1,2$.

Introducing auxiliary variables b_{31}, b_{2i} for $i = 1, 2, \ldots, N_2$ into Eq. (7), we can express $\tilde{s}(\mathbf{x})$ as the following simultaneous equations

$$\begin{cases} \tilde{s}(b_{31}) = \dfrac{f(b_{31}) - \phi}{\psi} \\ b_{31} - \displaystyle\sum_{j=1}^{N_2} w_{31j} f(b_{2j}) = \theta_{31} \\ b_{21} - \displaystyle\sum_{i=1}^{N_1} w_{21i}\hat{x}_i = \theta_{21} \\ \vdots \\ b_{2N_2} - \displaystyle\sum_{i=1}^{N_1} w_{2N_2 i}\hat{x}_i = \theta_{2N_2}, \end{cases} \tag{8}$$

where b_{31}, b_{2j} for $j = 1, 2, \ldots, N_2$, and $\hat{x}_i$ for $i = 1, 2, \ldots, N_1$, are variables. We see that all of the functions in Eq. (8) are separable functions.

2.2.2. RBF networks

In general, only three-layer RBF networks with a single hidden layer are considered in neural network literature (Bishop, 1995; Haykin, 1999). The forward mapping $\mathscr{F}(\hat{\mathbf{x}})$ realized by a RBF network can be expressed as follows

$$\mathscr{F}(\hat{x}) = \sum_{j=1}^{N_2} w_j \varphi_j(\hat{\mathbf{x}}) + \theta \tag{9}$$

where w_j is the weight connecting the jth unit in the hidden layer to the output unit, θ is the bias of the output unit, and φ_j denotes the RBF.

For the case of Gaussian basis function we have

$$\varphi_j(\mathbf{x}) = \exp\left(-\frac{\|\hat{\mathbf{x}} - \mathbf{r}_j\|^2}{2\sigma_j^2}\right) \tag{10}$$

where σ_j is the width parameter of φ_j, and $\mathbf{r}_j$ is the vector determining the center of φ_j.

Introducing auxiliary variables a_i for $i = 1, 2, \ldots, N_2$ into Eqs. (9) and (10), we can express $\tilde{s}(\mathbf{x})$ as the following simultaneous equations

$$\begin{cases} \tilde{s}(b) = \dfrac{b - \phi}{\psi} \\ b - \displaystyle\sum_{j=1}^{N_2} w_j a_j = \theta \\ \ln a_1 - \displaystyle\sum_{i=1}^{N_1} \frac{(\hat{x}_i - r_{1i})^2}{2\sigma_1^2} = 0 \\ \vdots \\ \ln a_{N_2} - \displaystyle\sum_{i=1}^{N_1} \frac{(\hat{x}_i - rN_{2i})^2}{2\sigma_{N_2}^2} = 0 \end{cases} \tag{11}$$

where b, a_i for $i = 1, 2, \ldots, N_2$, and $\hat{x}_i$ for $i = 1, 2, \ldots, N_1$ are variables, and r_{ij} for $i = 1, 2, \ldots, N_2$; $j = 1, 2, \ldots, N_1$ and σ_i for $i = 1, 2, \ldots, N_2$ are constants. Clearly, each of the functions in Eq. (11) is a separable function.

2.2.3. Multilayer quadratic perceptrons

In order to improve the learning performance of MLPs, various multilayer FNN architectures having higher-order connectivity have been proposed in neural network literature (Giles & Maxwell, 1987; Lu et al., 1993). Here, we consider the following higher-order network for transforming nonseparable functions

$$\hat{x}_{kj} = f\left(\sum_{i=1}^{N_{k-1}} \sum_{h=1}^{H} w_{ijkh}\hat{x}_{k-1,i}^h + bias_{ij}\right) \tag{12}$$

for $k = 2, \ldots, L$ and $j = 1, \ldots, N_k$

where H is the order of the connectivity, w_{kjih} the hth weight connecting the ith unit in the layer $k-1$ to the jth unit in the layer k, $f(\cdot)$ is the sigmoidal activation function. When $H=2$, Eq. (12) expresses the characteristic of the unit of MLQP (Lu et al., 1993).

For simplicity of description, we consider three-layer MLQPs. Following the similar way as described in Section 2.2.1, we can express $\tilde{s}(\mathbf{x})$ as the following simultaneous equations

$$\begin{cases} \tilde{s}(b_{31}) = \dfrac{f(b_{31}) - \phi_p}{\psi_p} \\ b_{31} - \displaystyle\sum_{j=1}^{N_2} w_{31j1} f(b_{2j}) - \sum_{j=1}^{N_2} w_{31j2} f^2(b_{2j}) = \theta_{31} \\ b_{21} - \displaystyle\sum_{i=1}^{N_1} w_{21i1}\hat{x}_i - \sum_{i=1}^{N_1} w_{21i2}\hat{x}_i^2 = \theta_{21} \\ \vdots \\ b_{2N_2} - \displaystyle\sum_{i=1}^{N_1} w_2 N_{2i}\hat{x}_i - \sum_{i=1}^{N_1} w_2 N_{2i2}\hat{x}_i^2 = \theta_2 N_2, \end{cases} \tag{13}$$

where b_{31}, b_{2j} for $j=1,2,\ldots,N_2$, and $\hat{x}_i$ for $i=1,2,\ldots,N_1$, are variables. We see that all of the functions defined by Eq. (13) are separable functions.

The importance of Eqs. (8), (11) and (13) lies in the fact that they provide us with a general approach to approximately expressing nonseperable functions as seperable forms.

3. Transformation of NLP problems

According to the locations of nonseparable functions in NLP problems, NLP problems can be classified into three types:

I: only the objective function is a nonseparable function;
II: only the constraints are nonseparable functions; and
III: both the objective function and the constraints are nonseparable functions.

3.1. Nonseparable objective function

For Type I nonseparable NLP problems, we only need to transform the objective function into separable function. Replacing the objective function with its approximation such as the form defined by Eq. (8), we can transform a Type I nonseparable NLP problem into an SP problem as follows:

$$\text{Minimize} \quad \frac{f(b_{31}^{o}) - \phi^{o}}{\psi^{o}} \tag{14}$$

subject to

$$\begin{cases} b_{31}^{o} - \displaystyle\sum_{j=1}^{N_2^{o}} w_{31j}^{o} f(b_{2j}^{o}) = \theta_{31}^{o} \\ b_{21}^{o} - \displaystyle\sum_{i=1}^{N_1} w_{21i}^{o}(\alpha_i^{o} + \beta_i^{o} x_i) = \theta_{21}^{o} \\ \vdots \\ b_{2N_2^{o}}^{o} - \displaystyle\sum_{i=1}^{N_1} w_{2N_2^{o}i}^{o}(\alpha_i^{o} + \beta_i^{o} x_i) = \theta_{2N_2^{o}}^{o} \\ \displaystyle\sum_{k=1}^{n} g_{ik}(x_k) \geq 0 \qquad \text{for } i = 1,2,\ldots,m, \\ \displaystyle\sum_{k=1}^{n} h_{jk}(x_k) = 0 \qquad \text{for } j = 1,2,\ldots,s. \end{cases}$$

where b_{31}^{o}, b_{2j}^{o} for $j=1,2,\ldots,N_2^{o}$, and x_i for $i=1,2,\ldots,N_1$ are variables, and the 'o' superscript refers to quantities on the objective function.

3.2. Nonseparable constraint function

For the sake of clarity, we define two sets:

$$U = \{i | g_i(\mathbf{x}) \text{ is a nonseparable function}\},$$

and

$$V = \{j | h_j(\mathbf{x}) \text{ is a nonseparable function}\}.$$

Then for each $i \in U$ and $j \in V$, $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ should be transformed into separable functions. By the definition, $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ for $i \in \bar{U}$ and $j \in \bar{V}$[2] are separable functions. Hence, no transformation needs to be carried out. Replacing the nonseparable constraints with their approximations such as the form defined by Eq. (8), we can transform a Type II nonseparable NLP problem into an SP problem as follows:

$$\text{Minimize} \quad \sum_{k=1}^{n} t_k(x_k) \tag{15}$$

[2] $\bar{U} = \{1,2,\cdots,m\} - U$ and $\bar{V} = \{1,2,\cdots,r\} - V$.

subject to

$$
\begin{cases}
\sum_{k=1}^{n} g_{lk}(x_k) \geq 0 & \text{for } l \in \bar{U} \\
\sum_{k=1}^{n} h_{lk}(x_k) = 0 & \text{for } l \in \bar{V} \\
\dfrac{f(b_{31}^{G,u}) - \phi^{G,u}}{\psi^{G,u}} \geq 0 & \\
b_{31}^{G,u} - \sum_{j=1}^{N_2^{G,u}} w_{31j}^{G,u} f(b_{2j}^{G,u}) = \theta_{31}^{G,u} & \\
b_{21}^{G,u} - \sum_{i=1}^{N_1} w_{21i}^{G,u}(\alpha_i^{G,u} + \beta_i^{G,u} x_i) = \theta_{21}^{G,u} & \\
\vdots & \\
b_{2N_2^{G,u}}^{G,u} - \sum_{i=1}^{N_1} w_{2N_2^{G,u}i}(\alpha_i^{G,u} + \beta_i^{G,u} x_i) = \theta_{2N_2^{G,u}}^{G,u} & \text{for } u \in U \\
\dfrac{f(b_{31}^{H,v}) - \phi^{H,v}}{\psi^{H,v}} = 0 & \\
b_{31}^{H,v} - \sum_{j=1}^{N_2^{H,v}} w_{31j}^{H,v} f(b_{2j}^{H,v}) = \theta_{31}^{H,v} & \\
b_{21}^{H,v} - \sum_{i=1}^{N_1} w_{21i}^{H,v}(\alpha_i^{H,v} + \beta_i^{H,v} x_i) = \theta_{21}^{H,v} & \\
\vdots & \\
b_{2N_2^{H,v}}^{H,v} - \sum_{i=1}^{N_1} w_{2N_2^{H,v}i}(\alpha_i^{H,v} + \beta_i^{H,v} x_i) = \theta_{2N_2^{H,v}}^{H,v} & \text{for } v \in V
\end{cases}
$$

where the 'G,u' and 'H,v' superscripts refer to quantities on the nonseparable constraints $g_u(\mathbf{x})$ for $u \in U$ and $h_v(\mathbf{x})$ for $v \in V$, respectively.

Using the similar approach mentioned above, we can transform Type III nonseparable NLP problems into SP problems and deal with unconstrained nonlinear programming problems.

3.3. Network size

For converting NLP problems into SP problems, each of the nonseparable functions in NLP problems should be approximated with an FNN. Therefore, one may ask: how many hidden units of an FNN are satisfactory for approximating a nonseparable function? In this section, we first introduce some related theoretical results, then we present an approach to cope with the curse of dimensionality in approximation of given functions.

3.3.1. Optimum number of hidden units

For any $0 < \delta < 1$, with probability greater than $1 - \delta$, Niyogi and Girosi (1996) have derived a bound on the generalization error generated by a Gaussian network as follows:

$$
\mathrm{O}\left(\frac{1}{r}\right) + \mathrm{O}\left(\left[\frac{nr\ln(rM) - \ln\delta}{M}\right]^{1/2}\right) \tag{16}
$$

where r is the number of hidden units, n is the input dimension of the function, and M the number of training data.

From the bound of Eq. (16), we may identify the following relationship between the optimum number of hidden units r^* and a given data size M:

$$
r^*(M) \propto M^{1/3} \tag{17}
$$

Note that there are various choices of r^* and M for a certain confidence parameter δ and for a fixed error bound (Niyogi & Girosi, 1996).

The importance of Eq. (17) lies in the fact that it provides us with the theoretical basis for selecting suitable network size for good generalization. It has been shown that MLPs require smaller number of parameters than RBF networks for the same degree of accuracy for function approximation (Haykin, 1999). In addition, experimental studies show that MLQPs can approximate functions with fewer number of hidden units than MLPs (Lu et al., 1993).

3.3.2. Practical considerations

As described earlier, two kinds of function approximation problems are considered in the transformation: (a) approximation of a given function which has precise analytic expression, and (b) approximation of an unknown function from training data. From the transformation's point of view, the first approximation problem is easier to be solved than the second problem. The main reason is that some given functions $h(\mathbf{x})$ can be expressed as the sum of several components as

$$
h(\mathbf{x}) = h_1(\mathbf{x}_1) + h_2(\mathbf{x}_2) + \cdots + h_K(\mathbf{x}_K) \tag{18}
$$

where $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{x}_i \in \mathbf{R}^{n_i}$, and $n_i < n$ for $i = 1, 2, \ldots, K$. For example, the objective function for minimizing the weight of speed reducer (Golinski, 1973) is stated as

$$
\begin{aligned}
f(x) = {} & 0.79x_1x_2^2(3.33x_3^2 + 14.93x_3 - 43.09) - 1.51x_1(x_6^2 + x_7^2) \\
& + 7.48(x_6^3 + x_7^3) + 0.79(x_4x_6^2 + x_5x_7^2)
\end{aligned} \tag{19}
$$

If a given nonseparable objective function or a given nonseparable constraint in NLP problems can be expressed as the form as defined by Eq. (18), then by using at most K relatively smaller FNNs, this nonseparable function can be

transformed into a separable function as follows:

$$\begin{cases} \tilde{h}(b_{31}^{h_1},\ldots,b_{31}^{h_K}) = \dfrac{f(b_{31}^{h_1}) - \phi h_1}{\psi_{h_1}} + \cdots + \dfrac{f(b_{31}^{h_K}) - \phi h_K}{\psi_{h_K}} \\ b_{31}^{h_1} - \sum_{j=1}^{N_2^{h_1}} w_{31j}^{h_1} f(b_{2j}^{h_1}) = \theta_{31}^{h_1} \\ b_{21}^{h_1} - \sum_{i=1}^{N_1^{h_1}} w_{21i}^{h_1} \hat{x}_i = \theta_{21}^{h_1} \\ \vdots \\ b_{2N_2}^{h_1} - \sum_{i=1}^{N_1^{h_1}} w_{2N_2 i}^{h_1} \hat{x}_i = \theta_{2N_2}^{h_1} \\ \vdots \\ b_{31}^{h_k} - \sum_{j=1}^{N_2^{h_K}} w_{31j}^{h_K} f(b_{2j}^{h_K}) = \theta_{31}^{h_K} \\ b_{21}^{h_K} - \sum_{i=1}^{N_1^{h_K}} w_{21i}^{h_K} \hat{x}_i = \theta_{21}^{h_K} \\ \vdots \\ b_{2N_2}^{h_K} - \sum_{i=1}^{N_1^{h_K}} w_{2N_2 i}^{h_K} \hat{x}_i = \theta_{2N_2}^{h_K} \end{cases} \tag{20}$$

where $h_k(x_k)$ is approximated by a three-layer MLP with $N_1^{h_k}$ $(=n_k)$ input units, $N_2^{h_k}$ hidden units, and one output unit, $b_{31}^{h_k}$, $b_{2j}^{h_k}$ for $j=1,2,\ldots,N_2^{h_k}$; $k=1,2,\ldots,K$, and $\hat{x}_i$ for $i=1,2,\ldots,N_1^{h_k}$, are variables.

In comparison to approximating $h(\mathbf{x})$ directly with a single FNN, the approach to approximating the components of $h(\mathbf{x})$ with relatively smaller FNNs has several merits. (a) It can circumvent the curse of dimensionality if $n_i \ll n$ for large n; (b) approximating $h_i(x_i)$ might much easier than approximating $h(\mathbf{x})$ because $h_k(x_k)$ is always simpler than $h(\mathbf{x})$, and therefore, we can use a relatively smaller FNN to approximate each component of $h(\mathbf{x})$; (c) for isomorphic components of $h(\mathbf{x})$, only one of them needs to be approximated and the approximating result can be adapted to the transformations of the others because the tasks of approximating the isomorphic components are the same from learning's point of view. Consequently, to approximate $h(\mathbf{x})$ needs at most K relatively smaller FNNs. For example, the four components of the objective function defined by Eq. (19), i.e. $-1.51x_1x_6^2$, $-1.51x_1x_7^2$, $0.79x_4x_6^2$, and $0.79x_5x_7^2$, are isomorphic to each other mathematically.

3.4. Accuracy of transformation

The accuracy of transforming NLP problems into SP problems by using the proposed method largely depends upon the performance of the trained FNNs for approximating the nonseparable functions in NLP problems. That is, their approximation errors on training data and their generalization errors on test data. Therefore, to achieve a high accuracy of transformation, both the approximation error and generalization error should be small.

It is known that the approximation and generalization errors of FNNs are affected by several factors such as the learning algorithms, the structure of FNNs, the complexity of the nonseparable functions, and the number of the training data. In recent years, many fast learning algorithms such as natural gradient learning (Amari, 1998; Amari, Park & Fukumizu, 2000) and new network architectures such as constructive neural networks (Choi & Choi, 1994) have been developed in order to avoid local minima encountered in training FNNs and improve generalization. As a result, we can combine these new techniques with the proposed method to achieve a good accuracy of transformation. Although the existing learning algorithms for FNNs are not guaranteed to produce a desired accuracy to approximate a complicated function, FNNs have become a popular tool for function approximation in high dimensions and are clearly recognized as a useful member of the toolbox of methods that one might use (Cherkassky, Gehring, & Mulier, 1996).

3.5. Problem size versus network size

Suppose that each nonseparable function is approximated by an FNN such as an MLP, an RBF network, or an MLQP with the same number of hidden units K. Also suppose that there are c_1 $(c_1 \leq m)$ nonseparable inequality constraints and c_2 $(c_2 \leq r)$ nonseparable equality constraints in Types II and III nonseparable NLP problems. From Eqs. (8), (11) and (13), we see that MLPs, RBF networks, and MLQPs lead to the same numbers of new variable and constraints for converting a nonseparable function to a separable function. The numbers of variables and constraints in the original NLP problems and the equivalent SP problems are shown in Table 1.

From Table 1 we see that as the number of hidden units grows both the number of variables and the number of constraints in the equivalent SP problems increase. Therefore, it is desirable that each nonseparable function is approximated by an FNN with as few a number of hidden units as possible.

Table 1
The number of variables and constraints in the original NLP problems and the equivalent SP problems transformed by using MLPs, RBF networks, or MLQPs

Problem	No. of variables	No. of constraints
Original	n	$m+r$
I	$n+K+1$	$m+r+K+1$
II	$n+c_1K+c_2K+(c_1+c_2)$	$(m-c_1)+(r-c_2)+c_1K+c_2K+c_1+c_2$
III	$n+K+c_1K+c_2K+(1+c_1+c_2)$	$(m-c_1)+(r-c_2)+K+c_1K+c_2K+(1+c_1+c_2)$

But, on the other hand, it may become more difficult for an FNN with fewer number of hidden units to approximate a nonseparable function with a desired degree of accuracy. Therefore, there exists a trade-off between the complexity of the equivalent SP problems and the size of FNNs.

4. Approximations of SP problems

In this section, we firstly introduce how to approximate SP problems by replacing each of separable nonlinear functions with their approximations using piecewise linear functions. Then we discuss the relationship between the accuracy of the solutions to NLP problems and the accuracy of two different kinds of approximations used for transforming NLP problems into approximating LP (ALP) problems. Finally, we analyze the complexity of the ALP problems.

4.1. Piecewise approximation

In order to use the simplex method with some variation to solve SP problems, we need to define a new problem that approximates the original SP problem. The new problem is obtained by replacing each nonlinear function with a piecewise linear approximation.

Suppose that we are interested in the values of the nonlinear function e over the interval $[r_a, r_b]$, and we wish to define a piecewise linear function $\tilde{e}$ that approximates e. The interval $[r_a, r_b]$ is partitioned into several subintervals, via the grid points $r_a = \mu_1, \mu_2, ..., \mu_p = r_b$. The nonlinear function e can be approximated over the interval $[r_a, r_b]$ via the grid points $\mu_1, \mu_2, ..., \mu_p$ by the piecewise linear function $\tilde{e}$, defined by

$$\begin{cases} \tilde{e}(\mu) = \sum_{i=1}^{p} \lambda_i e(\mu_i) \\ \sum_{i=1}^{p} \lambda_i = 1 \\ \lambda_i \geq 0 \text{ for } i = 1, 2, ..., p, \end{cases} \tag{21}$$

where, at most, two adjacent λ_is are positive.

According to the definition of $\hat{e}(\mu)$, the SP problems can be restated in equivalent more manageable forms. For example, the SP problem defined by Eq. (14) can be restated as

$$\text{Minimize} \quad \frac{\sum_{r=1}^{p} \lambda_{31,r}^{o} f(\mu_{31,r}^{o}) - \phi^{o}}{\psi^{o}} \tag{22}$$

subject to

$$\begin{cases} \sum_{r=1}^{p} \lambda_{31,r}^{o} \mu_{31,r}^{o} - \sum_{j=1}^{N_2^o} w_{31j}^{o} \sum_{r=1}^{p} \lambda_{2j,r}^{o} f(\mu_{2j,r}^{o}) = \theta_{31}^{o} \\ \sum_{r=1}^{p} \lambda_{21,r}^{o} \mu_{21,r}^{o} - \sum_{i=1}^{N_1} w_{21i}^{o} (\alpha_i^{o} + \beta_i^{o} \sum_{r=1}^{p} \lambda_{21i,r}^{x} \mu_{ir}^{x}) = \theta_{21}^{o} \\ \vdots \\ \sum_{r=1}^{p} \lambda_{2N_2^o,r}^{o} \mu_{2N_2^o,r}^{o} - \sum_{i=1}^{N_1} w_{2N_2^o i}^{o} (\alpha_i^{o} + \beta_i^{o} \sum_{r=1}^{p} \lambda_{2N_2^o i,r}^{x} \mu_{ir}^{x}) = \theta_{2N_2^o}^{o} \\ \sum_{k=1}^{n} \sum_{r=1}^{p} \lambda_{ik,r}^{g} g_{ik}(\mu_{ik,r}^{g}) \geq 0 & \text{for } i = 1, ..., m \\ \sum_{k=1}^{n} \sum_{r=1}^{p} \lambda_{jk,r}^{h} h_{jk}(\mu_{jk,r}^{h}) = 0 & \text{for } j = 1, ..., s \\ \sum_{r=1}^{p} \lambda_{31,r}^{o} = 1 \\ \sum_{r=1}^{p} \lambda_{2j,r}^{o} = 1 & \text{for } j = 1, 2, ..., N_2^o \\ \sum_{r=1}^{p} \lambda_{2ji,r}^{x} = 1 & \text{for } j = 1, 2, ..., N_2^o \text{ and } i = 1, 2, ..., N_1 \\ \sum_{r=1}^{p} \lambda_{ik,r}^{g} = 1 & \text{for } i = 1, 2, ..., m \text{ and } k = 1, 2, ..., n \\ \sum_{r=1}^{p} \lambda_{jk,r}^{h} = 1 & \text{for } j = 1, 2, ..., s \text{ and } k = 1, 2, ..., n \end{cases}$$

$\lambda \geq 0$ for all λ mentioned in the above constraints

where, at most, two adjacent λ's are positive.

With the exception of the constraint that, at most, two adjacent λ's are positive, the problem defined by Eq. (22) is a LP problem, and hence it is called approximating LP problem. It has been shown that the ALP problems can be solved by means of the simplex method with the *restricted basis entry rule* (Miller, 1963).

4.2. Accuracy of optimal solutions

For transforming nonseparable NLP problems into ALP problems, we need to carry out two different kinds of approximations: (a) each nonseparable function in NLP problem should be approximated by a FNNs, and (b) each nonlinear function in the SP problem should be approximated by piecewise linear functions. One may ask a question: What is the relationship between the optimal solutions to the original nonseparable NLP problems and the corresponding ALP problems ? In the rest of this subsection, we will answer this question.

Consider the first approximation. Suppose that the trained FNN of a nonseparable function has a small approximation error and small generalization error. The SP problems obtained by the proposed method are equivalent to their original nonseparable NLP problems because the transformation only changes the expression forms of the objective function and/or constraints.

The accuracy of the second approximation largely depends upon the number of grid points for each variable. For example, Fig. 2 shows the relationship between the number of grid points for each variable and the accuracy of the values of the objective function for the corresponding ALP problems. From this figure, we can see that the accuracy of the optimal solutions can be improved by increasing the number of grid points. However, as the number of grid points increases, the number of variables and the number of constraints in the ALP problems also increase. It has been shown that if the objective function is strictly convex and all the constraints are convex, by choosing a small grid, the solution obtained from the ALP problem can be sufficiently close to the global optimal solution of the original NLP problem (Bazaraa et al., 1993). In the non-convex case, even though optimality cannot be claimed with the restricted basis entry rule, good solutions are produced (Bazaraa et al., 1993). In addition, the value of each grid point is another factor that may largely affect the accuracy of the second approximation.

4.3. Complexity of ALP problems

For simplicity of analyzing the complexity of ALP problems and comparison of MLPs with RBF networks and MLQPs, we classify NLP problems defined by Eq. (1) into the following three types

$\mathscr{A}$. Only the objective function or one constraint is a nonseparable function, and all of the rest functions are linear functions;

$\mathscr{B}$. Among the objective function and the constraints, there are only $L(1 < L < m + s + 1)$ nonseparable functions, and there are $n_l(0 \leq n_l < n)$ input variables which are not involved in any nonlinear functions;

$\mathscr{C}$. Both the objective function and the constraints are nonseparable functions.

Suppose that each of the nonseparable functions in NLP problems is transformed into a separable function by using an MLP, a RBF network, and an MLQP with the same number of hidden units (K). Also suppose p grid points are used for each of the variables involved in the nonlinear functions. Considering Eqs. (8), (11), (13), (21), and the definition of ALP problems as stated by Eq. (22), we see that MLPs and other two networks (i.e. RBF networks and MLQPs) result in different number of new variables and new constraints in ALP problems. Tables 2 and 3 show the number of variables and the number of constraints in each of three types of ALP problems, whose original NLP problems belong to Type $\mathscr{A}$ Type $\mathscr{B}$, and Type $\mathscr{C}$ NLP problems, respectively.

From Tables 2 and 3, we see that the numbers of variables and constraints in an ALP problem not only depend upon the numbers of variables and constraints in the corresponding SP problem, but also depend upon the types of the neural networks used for converting the nonseparable functions in the original NLP problem. RBF networks and MLQPs cause the same numbers of new variables and new constraints. On the other hand, MLPs require fewer number of new variables and new constraints than both RBF networks and MLQPs in all of the ALP problems. The reason is that all of the functions involving the variables $\hat{x}_i$ for $i = 1, 2, \ldots, N_1$ in Eq. (8) are linear functions, while all of the functions involving the variables $\hat{x}_i$ for $i = 1, 2, \ldots, N_1$ in both Eqs. (11) and (13) are nonlinear functions. From computational complexity's point of view, MLPs are more suitable for converting nonseparable functions than RBF network and MLQPs.

5. Comparison with the existing approach

In this section, we compare the proposed transformation method with the existing approach and discuss the merits and demerits of each of the methods. To our best knowledge, only one transformation technique which is called the substitution approach is known in optimization literature (Rao, 1996).

Table 2
The number of variables in each of the three types of ALP problems

Type	MLP	RBF or MLQP
$\mathscr{A}$	$n + p(K + 1)$	$p(n + K + 1)$
$\mathscr{B}$	$n_l + p(K + 1)L + n - n_l)$	$p(n + (K + 1)L)$
$\mathscr{C}$	$N + p(m + r + 1)(K + 1)$	$p(n + (m + r + 1)(K + 1))$

Table 3
The number of constraints in each of the three types of ALP problems

Type	MLP		RBF or MLQP	
	No. of Eq.	No. of Ineq.	No. of Eq.	No. of Ineq.
$\mathscr{A}$	$2(K+1)$	$m+r+p(K+1)$	$n+2(K+1)$	$m+r+p(K+1+n)$
$\mathscr{B}$	$2(K+1)L+n-n_l$	$m+r+p((K+1)L+n-n_l$	$n+2(K+1)L$	$m+r+p((K+1)L+n)$
$\mathscr{C}$	$2(K+1)(m+r)$	$m+r+p(K+1)(m+r)$	$n+2(K+1)(m+r)$	$m+r+p+((K+1)(m+r)+n)$

5.1. The substitution approach

For converting a nonseparable function to a separable form, the following two kinds of basic substitutions are used in the substitution approach.

Substitution 1. Replace any nonseparable function of the form

$$p_1(x_1)\cdot p_2(x_2) \tag{23}$$

by

$$y_1^2 - y_2^2 \tag{24}$$

where two new variables y_1 and y_2 are defined as

$$y_1 = \frac{p_1(x_1) + p_2(x_2)}{2} \tag{25}$$

$$y_2 = \frac{p_1(x_1) - p_2(x_2)}{2} \tag{26}$$

Substitution 2. Replace any nonseparable function of the form

$$p_1(x_1)\cdot p_2(x_2)\cdots p_M(x_M) \tag{27}$$

by y along with the constraint

$$\ln y = \ln p_1(x_1) + \ln p_2(x_2) + \cdots + \ln p_M(x_M) \tag{28}$$

where $p_i(x_i)$ for $i = 1, 2..., M$ are separable functions and $p_i(x_i) > 0$.

5.2. Generality

Although many nonseparable functions can be transformed into separable forms by the substitution approach, this approach suffers the following two deficiencies that limit its usefulness. One is that it cannot convert some nonseparable functions. For example, it cannot convert the nonseparable function $p(x_i)^{q(x_j)}$ into a separable function when $p(x_i) \leq 0$. The other one is that it is not capable of transforming the functions whose concise mathematical descriptions are unknown.

In some practical optimization problems, the objective function and/or the constraints may be complex 'black boxes' of implicit unknown forms. In such cases, it is impossible to formulate the optimization problems as NLP problems by using the exact system models. Fortunately, from Eqs. (8), (11), and (13), we can see that the proposed transformation method may provide an efficient technique for dealing with this difficulty. Suppose that a set of training data for an unknown function is given. By training a multilayer FNN, the following two tasks can be performed at the same time: (a) the unknown function is approximated, and (b) the unknown function is transformed into a separable function. That is, training FNNs succeed in doing both the system identification and function transformation.

From optimization's point of view, the importance of the proposed method is that it may provide us a new way for formulating complex optimization problems as SP problems. As a result, the simplex method can be applied to solving large-scale and complex optimization problems, whose solutions are hard to be found by using the conventional NLP techniques.

5.3. Selection of grid points

In order to accurately approximate each nonlinear function in SP problems by using piecewise linear functions, it is important to choose an appropriate number of grid points and suitable grid points.

From Eq. (8), we can see that the separable functions transformed by MLPs have two useful features for selection of grid points. (a) All of the nonlinear components of the separable functions, i.e. $f(b_{31})$ and $f(b_{2j})$ for $j = 1, 2, ..., N_2$, are the same type of sigmoid functions; (b) all of the nonlinear components are completely independent of the training data and the parameters of MLPs. According to the first feature, we need only to select one set of grid points for all of the variables. From the second feature, one set of suitable grid points can be used for any nonlinear components in any separable functions transformed by MLPs. In a word, these features lead to an important advantage of easily selecting grid points for piecewise linear approximation. For example, the grid points ± 16, ± 8, ± 5, ± 4, ± 3, ± 2, and ± 1 are used for all of the corresponding variables to approximate the sigmoid function in four different problems as illustrated in Examples 1–4 in Section 6. In addition, these features may highly facilitate automatic transformation of SP problems into ALP problems and its software implementation.

From Eq. (13), we can see that the separable functions transformed by MLQPs have the same features as MLPs mentioned above. However, some nonlinear components of

the separable functions transformed by RBF networks, i.e. $(\hat{x}_i - r_{1i})^2/2\sigma_1^2$ for $i = 1, 2, \ldots, N_1$, do not possess the second useful feature since they largely depend on r_{1i} which is determined by training data. Consequently, different grid points should be selected for each of the variables.

Generally speaking, the separable functions converted by the substitution approach do not possess the two useful features given above. Therefore, we should carefully select the grid points for each variable. In some cases, one variable might included in different types of nonlinear functions. If their inflection points are located at different intervals, then more number of grid points are required for each variable because the grid lengths should be smaller in the neighborhood of the inflection points than other places in the interesting interval. For example, three inflection points of $\sin(x^2)$ are distributed at -1.3, 0 and 1.3, while four inflection points of $\cos(x^2)$ are distributed at -1.8, -0.6, 0.6, and 1.8. So that, a large number of grid points are required for x in order to approximate $\sin(x^2)$ and $\cos(x^2)$ accurately.

From the point of view of selecting grid points, the proposed method based on MLPs or MLQPs is superior to the substitution technique because it can avoid the problem of selecting miscellaneous sets of grid points and it may use fewer number of grid points.

5.4. Computational complexity and accuracy

The transforming mechanisms used in the substitution approach and the proposed method are completely distinct. The transformation realized by the substitution approach is to directly replace the nonseparable components of a function with separable ones. The advantages of this approach are that (a) the transformation procedure is simple and straightforward, and (b) no errors exist between the nonseparable functions and separable functions. The transformation performed by the proposed method is to approximate nonseparable functions through learning. In comparison with the substitution approach, the proposed method might require more computer memories and times. In addition, there might exist small errors between the nonseparable functions and separable functions.

In summary, the advantages of the substitution approach over the proposed method are its simplicity and its transformation accuracy. However, its disadvantages are that (a) it cannot transform the functions whose concise expressions are unknown and some given nonseparable functions, and (b) it needs to select miscellaneous sets of grid points for different types of nonlinear functions.

6. Simulation results

In this section four examples are presented. The first example is used to illustrate how to transform a nonseparable NLP problem into an SP problem by the proposed method, and how to solve the SP problem by the simplex method with the restricted basis entry rule. The second example is simulated to demonstrate how to improve the accuracy of the solutions to NLP problems by increasing the number of training data and the number of grid points. The third example is used to demonstrate how to formulate a practical optimization problem as an SP problem, in which the constraints have no precise analytic expression. The last example is used to show the performance of the simplex method with the restricted basis entry rule for solving large-scale SP problems.

In the following simulations, $X_{\min}$, $X_{\max}$, $Y_{\min}$, and $Y_{\max}$ were set to 0.001, 0.9999, 0.1, and 0.9, respectively. All of the MLPs were trained by the standard backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986).

6.1. Example 1

Consider the following NLP problem

$$\text{Minimize} \quad 2 - \sin^2 x_1 \sin^2 x_2 \tag{29}$$

$$\text{subject to} \begin{cases} 0.5 \le x_1 \le 2.5 \\ 0.5 \le x_2 \le 2.5 \end{cases}$$

Clearly, the above problem is a Type I nonseparable NLP problem.

In order to transform the problem into an SP problem, the nonseparable objective function was approximated by a three-layer MLP with two input, five hidden, and one output units. The training data set consists of 625 (25×25) input–output data which were created by sampling the original objective function over the input space $[0.5, 2.5] \times [0.5, 2.5]$ in a uniform grid. The training inputs and the desired outputs were normalized by Eqs. (4) and (5), respectively, where $x^1_{\min} = x^2_{\min} = 0.5$, $x^1_{\max} = x^2_{\max} = 2.5$, $y_{\min} = 1.000314$, and $y_{\max} = 1.94717$. In the simulation, the learning is considered complete when the sum of the squared error between the target and actual outputs becomes less than 0.025. The parameters of the trained network are as follows

$$W_2 = \begin{bmatrix} -5.101386, & -0.650895 \\ -3.985364, & -3.489580 \\ -4.685999, & 3.113827 \\ -2.336488, & 4.702585 \\ 1.291487, & 5.230563 \end{bmatrix}$$

$$W_3 = [-5.129751, 5.657458, 5.074175, -5.235262, 4.964876]$$

$$\theta_2 = [4.106874, 2.906349, -0.301974, -0.217906, -4.580654]^{\mathrm{T}}$$

$$\theta_3 = [1.881747]$$

According to Eq. (14), by replacing the nonseparable objective function with its approximation realized by

the trained network, we obtain the corresponding SP problem as follows

$$\text{Minimize} \quad \frac{f(b_{31}) - \phi}{\psi} \tag{30}$$

subject to

$$\begin{cases} b_{31}^{O} + 5.12975 f(b_{21}) - 5.657458 f(b_{22}) - 5.074175 f(b_{23}) + 5.235262 f(b_{24}) - 4.964867 f(b_{25}) = 1.881747 \\ b_{21}^{O} + 5.101386(\alpha_1 + \beta_1 x_1) + 0.650895(\alpha_2 + \beta_2 x_2) = 4.106874 \\ b_{22}^{O} + 3.985364(\alpha_1 + \beta_1 x_1) + 3.489580(\alpha_2 + \beta_2 x_2) = 2.906349 \\ b_{23}^{O} + 4.685999(\alpha_1 + \beta_1 x_1) - 3.113827(\alpha_2 + \beta_2 x_2) = -0.301974 \\ b_{24}^{O} + 2.336488(\alpha_1 + \beta_1 x_1) - 4.702585(\alpha_2 + \beta_2 x_2) = -0.217906 \\ b_{25}^{O} - 1.291487(\alpha_1 + \beta_1 x_1) - 5.230563(\alpha_2 + \beta_2 x_2) = -4.580654 \\ 0.5 \le x_1 \le 2.5 \\ 0.5 \le x_2 \le 2.5 \end{cases}$$

where $\phi = 0.881957$, $\psi = 1.18357$, $\alpha_1 = \alpha_2 = -0.24985$, $\beta_1 = \beta_2 = 0.4999$, and $f(\cdot)$ denotes the following sigmoidal activation function

$$f(z) = \frac{1}{1 + \exp(z)} \tag{31}$$

Approximating the sigmoidal activation function in the SP problem over the interval $[-16, 16]$ via 14 grid points ± 16, ± 8, ± 5, ± 4, ± 3, ± 2, and ± 1, as shown in Fig. 1, the SP problem can be stated as an ALP problem as defined by Eq. (22). Solving the ALP problem by the simplex method with the restricted basis entry rule (Miller, 1963), we obtain an optimal solution: $b_{31}^{*} = -1.791901$, $x_1^{*} = 1.583305$ and $x_2^{*} = 1.556218$. The corresponding value of the objective function for the SP problem is 1.051018, whereas the value of the objective function for the original NLP problem is 1.000369. Solving the original NLP problem directly by the Powell (1978) method, we obtain an optimal solution: $x_1^{*} = 1.570801$ and $x_2^{*} = 1.570801$. The corresponding optimal value is 1.0.

It should be noted that the solutions of the ALP problems will be slightly different from the solutions of the original NLP problems since the nonseparable functions and the nonlinear functions in the original problems are replaced with their approximations. In the following example, we will show that this difference can be improved by increasing the number of training data and the number of grid points.

6.2. Example 2

Consider the following NLP problem (McCormick, 1983)

$$\text{Minimize} \quad x_1^2 - x_2^2 - 2x_1x_2 + 8x_1 - 3x_2 + \exp(-x_1) + 16 \tag{32}$$

$$\text{subject to} \begin{cases} -6x_1^2 + 9x_1x_2 - 4x_2^2 + 5x_2 + 6 \ge 0 \\ 2x_1 - 2x_2 + 3 \ge 0 \\ -2 \le x_1 \le 2 \\ -2 \le x_2 \le 2 \end{cases}$$

Obviously, this problem is a Type III nonseparable NLP problem since the objective function and one constraint are nonseparable functions.

In order to convert the nonseparable objective function and the nonseparable constraint into separable forms, two three-layer MLPs were selected each of which has two input, five hidden and one output units. In the following two comparative simulations, different numbers of training data were used. The learning is considered complete when the sum of the squared error was smaller than 0.01.

In the first simulation, two training data sets were created by sampling the nonseparable objective function and the nonseparable constraints over the input space $[-2, 2] \times [-2, 2]$ in a uniform grid, respectively. Each of the two training sets consists of 100 (10×10) training data. After successfully training of the networks, the original NLP problem was converted into an SP problem by replacing the nonseparable objective function and the nonseparable constraint with their approximations realized by the trained networks. Approximating the nonlinear functions in the SP problem with 14, 28, 40, 62, 72, and 100 grid points, respectively, we obtain six related ALP problems. Solving the ALP problems by the simplex method with the restricted basis entry rule, we obtain six optimal solutions, which are shown in Table 4 (left part) and plotted in Fig. 2. From Fig. 2, we can easily see that the accuracy of the optimal solutions is improved as the number of grid points increases.

In the second simulation, the number of training data in each of the two training sets was largely increased in order to improve the accuracy of approximating the nonseparable objective function and the nonseparable constraint. Here, each of the two training sets consists of 1600 (40×40) training data. Following the same way as described in the first simulation, we obtain six optimal solutions, which are

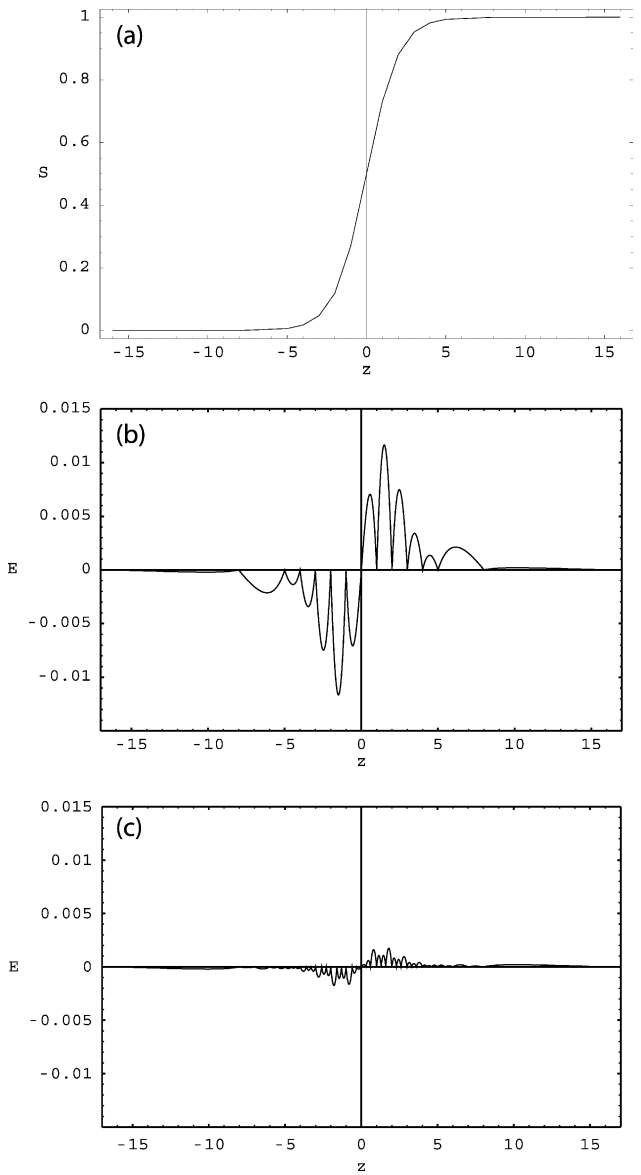

Fig. 1. (a) Shows the piecewise linear approximation of the sigmoidal activation function over the interval $[-16, 16]$ via 14 grid points ±16, ±8, ±5, ±4, ±3, ±2, ±1, (b) and (c) show the errors between the original sigmoidal activation function and its piecewise linear approximations with 14 and 40 grid points, respectively.

also shown in Table 4 (right part) and plotted in Fig. 2. From Fig. 2, we see that better solutions than those of the first simulation were obtained by increasing the number of training data for approximating the nonseparable functions.

McCormick (1983) solved the original NLP problem by using the variable-reduction method, and reported the following optimal solution: $x_1^* = -1.014423$, $x_2^* = -0.044096$. The corresponding optimal value is $t(x^*) = 11.712723$. Fig. 3 shows the NLP problem and its four different solutions.

The simulation results mentioned above indicate that, in order to get good solutions, not only a sufficient number of training data are required for approximating the nonseparable functions in NLP problems, but also a sufficient number of grid points are necessary for approximating the nonlinear functions in the corresponding SP problems.

6.3. Example 3

Consider the inverse kinematics problem of a redundant manipulator (Schilling, 1994). If its precise analytic forward kinematic function is given, the inverse kinematics problem can be formulated as the following NLP problem (Lu &Ito, 1996)

$$\text{Minimize} \quad t(\mathbf{q}) \tag{33}$$

$$\text{Subject to} \begin{cases} \bar{\mathbf{p}} = \mathbf{z}(\mathbf{q}) \\ \Gamma \le \mathbf{q} \le \Omega, \end{cases}$$

where $\mathbf{q}$ is the joint-angle variables, $\bar{\mathbf{p}}$ is the desired end-effector position, $\mathbf{z}(\mathbf{q})$ is the forward kinematic function, Γ and Ω are joint-angle limits, respectively, and $t(\mathbf{q})$ is the objective function.

It has been shown that the NLP problem stated by Eq. (33) can be transformed into an SP problem by introducing auxiliary variables (Lu & Ito, 1996). For some complex redundant manipulators, however, their precise analytic forward kinematic functions are usually unknown. In such cases, it is difficult to formulate the inverse kinematics problem as an NLP problem defined by Eq. (33). In the following simulation, we demonstrate that the proposed method provides a useful approach to dealing with this difficulty.

Table 4
Comparison of the solution of the NLP problem, whose nonseparable functions were approximated with different numbers of training data, and the nonlinear functions in the corresponding SP problems were approximated under different numbers of grid points

Grid points	No. of training data = 100			No. of training data = 1600		
	Optimal solution		Optimal value $t(x^*)$	Optimal solution		Optimal value $t(x^*)$
	x_1^*	x_2^*		x_1^*	x_2^*	
14	−0.648782	0.010571	13.125986	−0.708971	−0.025542	12.903833
28	−0.926872	−0.156283	12.174273	−0.981604	−0.189123	12.011294
40	−1.011951	−0.207331	11.924760	−1.055762	−0.233617	11.804844
62	−1.022869	−0.213881	11.894318	−1.057518	−0.234671	11.800164
72	−1.024989	−0.215154	11.888444	−1.059833	−0.23609	11.794005
100	−1.036463	−0.222038	11.856924	−1.070795	−0.242637	11.765095

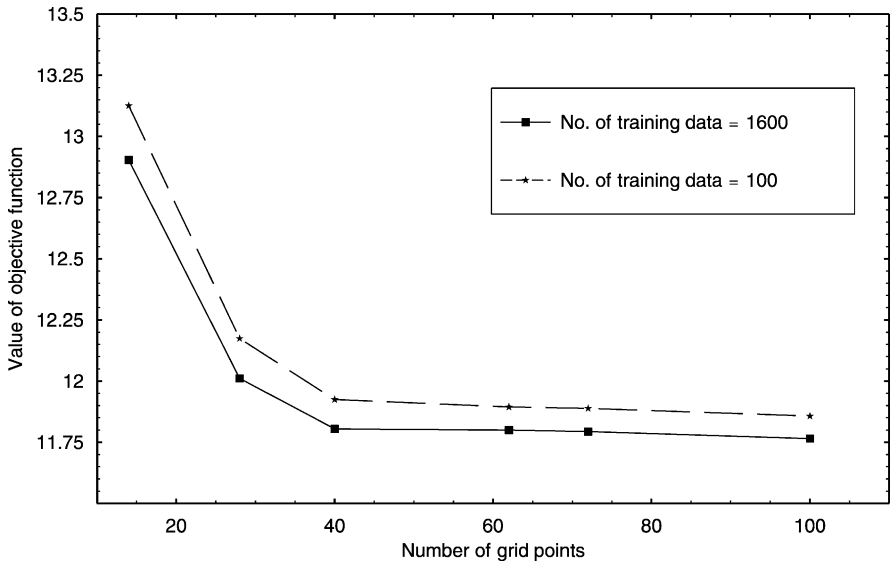

Fig. 2. Values of the objective function vs. number of grid points for two data sets with different numbers of training data used for approximating the nonseparable objective function and the nonseparable constraint. The dashed line shows the case where 100 training data were used. The solid line shows the case where 1600 training data were used.

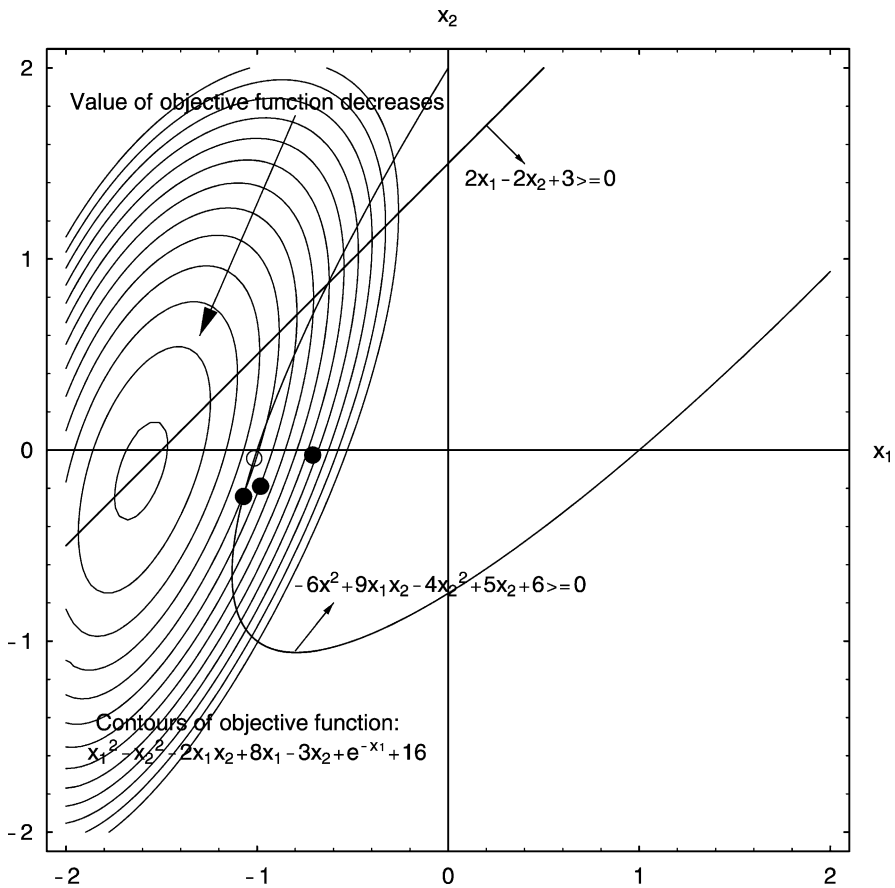

Fig. 3. Graphical representations of the NLP problem stated by Eq. (32) and its four different solutions. The small open circle denotes the optimal solution obtained by solving the NLP problem with the variable-reduction method. Three filled disks denote the optimal solutions obtained by solving the corresponding ALPs problem by means of the simplex method with the restricted basis entry rule, where 1600 training data and three different numbers of grid points were used. The disks from right to left correspond to the solutions of the ALP problems with 14, 28 and 100 grid points for each variable, respectively.

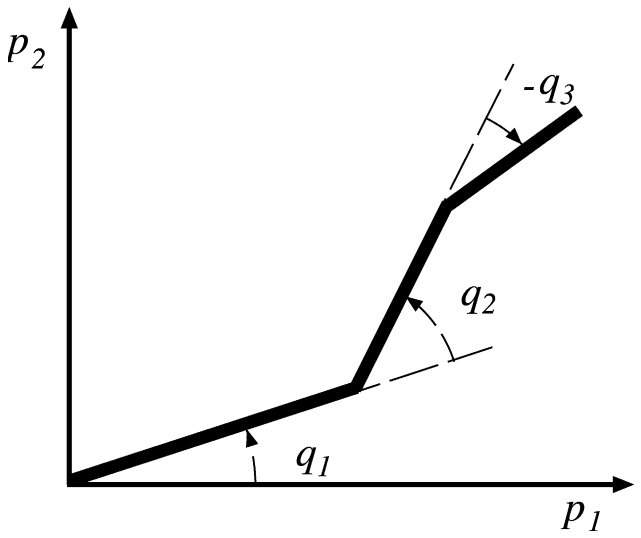

Fig. 4. A three-joint planar manipulator.

Without loss of generality and for simplicity of illustration, let us consider a three-angle planar manipulator shown in Fig. 4. Suppose that the precise analytic forward kinematic function of the manipulator is unknown, and the motion of the joints q_1, q_2, and q_3 are restricted to the intervals $[-\pi/6, 2\pi/3]$, $[0, 5\pi/6]$, and $[-\pi/6, \pi/6]$, respectively. For approximating the forward kinematic function by using FNNs, 1331 training data were collected. Each of the training data consists of a pair of joint-angle and end-effector position.

In order to speed up the training process for approximating the forward kinematic function and to reduce the computational cost for finding inverse kinematics solutions, a modular network architecture was used (Lu & Ito, 1995).

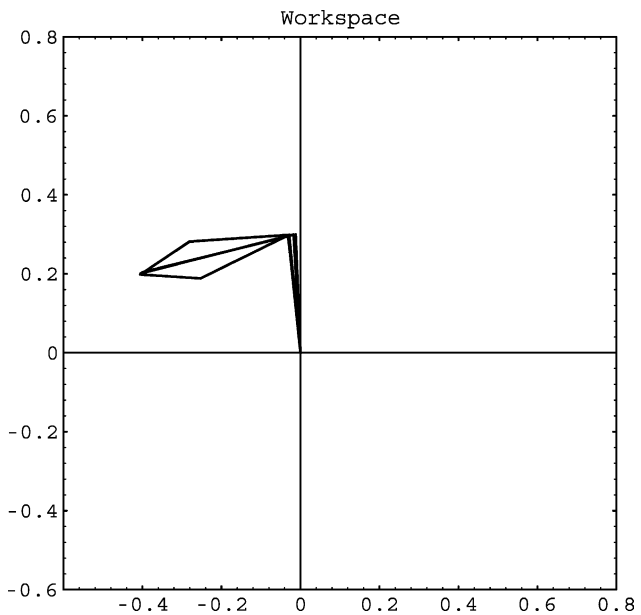

Fig. 5. Four different inverse kinematic solutions obtained by solving the SP problem defined by Eq. (34). Note that two of solutions are quite near and overlapped in the figure.

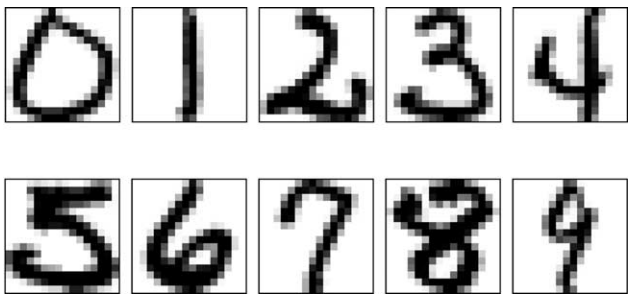

Fig. 6. Ten handwritten digits whose actual outputs were selected as the given outputs.

According to this architecture, the learning task was divided into eight subtasks by partitioning the configuration space $\mathbf{Q}$ into eight overlapping subregions via the grid points: $(-\pi/6, 3\pi/12, 2\pi/3)$, $(0, 5\pi/12, 5\pi/6)$, and $(-\pi/6, 0, \pi/6)$. Over each of the subregions, about 200 training data were used. Eight network modules were selected for approximating the forward kinematic function. Each of them is a three-layer MLP with 3 input, 10 hidden, and 2 output units.

From network inversion's point of view, if the feedforward kinematic function is precisely approximated by all the network modules, solving the inverse kinematic problem is equivalent to inverting the corresponding trained network modules. The inverse problem for MLPs can be formulated as the following NLP problem (Lu, Kita, & Nishikawa, 1999)

$$\text{Minimize} \quad t(\mathbf{q}) \tag{34}$$

$$\text{Subject to} \begin{cases} \bar{\mathbf{b}}_3 - W_3\mathbf{f}_2(\mathbf{b}_2) = \boldsymbol{\theta}_3 \\ \mathbf{b}_2 - W_2(\boldsymbol{\alpha} + \boldsymbol{\beta}\mathbf{q} = \boldsymbol{\theta}_2 \\ \qquad \Gamma \le \mathbf{q} \le \Omega \end{cases}$$

where W_3, W_2, $\boldsymbol{\theta}_3$, $\boldsymbol{\theta}_2$, $\bar{\mathbf{b}}_3$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, Γ, and Ω are constants, $\mathbf{q}$ and $\mathbf{b}_2$ are variables. It has been shown that the NLP problem defined by Eq. (34) is an SP problem (Lu et al., 1999), provided that $t(\mathbf{q})$ is a separable function.

Comparing Eq. (33) with Eq. (34), we see that the MLPs play two important roles in formulating the inverse kinematics problem. One is to approximate the forward kinematic function from data, and the other is to transform the forward kinematic function into a separable function. To compute multiple inverse kinematic solutions, we select the objective function $t(\mathbf{q})$ of Eq. (34) as: $t(\mathbf{q}) = \pm q_i$ for $i = 1, 2, 3$. This objective function allows us to minimize and maximize the movement of the ith link. For example, let us compute the inverse kinematic solutions for the desired end-effector position $\bar{\mathbf{p}} = (-0.4, 0.2)$. According to the partition

Table 5
Comparison of the sizes of the ALP problems and CPU times

No of grid points	No of variables	No of constraints	CPU time (s)
14	676	576	841
28	1096	576	1521
40	1456	576	2323

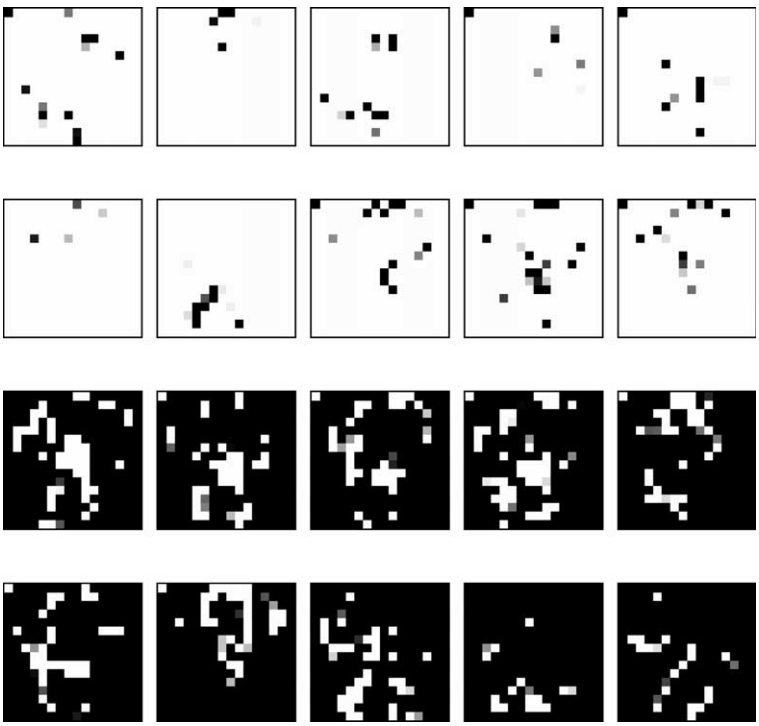

Fig. 7. Network inversions corresponding to the actual outputs of the handwritten digits '0' through '9' shown in Fig. 6. The 10 panels in the top two rows and the 10 panels in the bottom two rows show the inversions obtained by minimizing $\sum_{i=1}^{256}$ and maximizing $\sum_{i=1}^{256} x_i$, respectively.

of the configuration space, four network modules were selected for inverting. By approximating the sigmoidal activation functions including in the SP problems over the interval $[-16, 16]$ via 14 grid points ± 16, ± 8, ± 5, ± 4, ± 3, ± 2, ± 1, we solve the corresponding ALP problems by the simplex method with the restricted basis entry rule, and obtain four inverse kinematic solutions shown in Fig. 5. The results demonstrate that even if we have no precise analytic forward kinematic function, we can formulate the inverse kinematic problem as an SP problem by the proposed method and obtain the inverse kinematic solutions successfully.

6.4. Example 4

One of the most important aims of transforming NLP problems into SP problems is to generalize the simplex method to solve large-scale NLP problems, because of the lack of reliable and efficient computer-implemented algorithms for solving large-scale NLP problems. In order to show the performance of the simplex method with the restricted basis entry rule for solving large-scale SP problems, we consider a relatively larger SP problem in the following simulations.

Consider the problem of inverting the trained three-layer MLP for handwritten ZIP code recognition (Le Cun et al., 1989). The network has 256 input, 30 hidden and 4 output units. It has been shown that this inverse problem can be formulated as the following SP problem (Lu et al., 1999):

$$\underset{x_i, b_{2i}}{\text{Minimize}} \ \pm \sum_{i=1}^{256} x_i \tag{35}$$

$$\text{Subject to} \begin{cases} \bar{b}_{3i} - \sum_{j=1}^{30} w_{3ij} f(b_{2j}) = \theta_{3i}, \ i = 1, \ldots, 4 \\ b_{2i} - \sum_{j=1}^{256} w_{2ij} x_j = \theta_{2i}, \ i = 1, \ldots, 30 \\ 0.01 \le x_i \le 0.99, \ i = 1, \ldots, 256, \end{cases}$$

where $\bar{b}_{3i} = f^{-1}(\bar{y}_i)$, w_{3ij}, θ_{3i}, w_{2ij}, and θ_{2i} are constants, b_{2i} and x_i are variables, and $f(\cdot)$ denotes the sigmoidal activation function stated by Eq. (31).

In the simulation, the actual outputs of the network corresponding to the ten handwritten digits shown in Fig. 6 were selected as the given outputs $\bar{\mathbf{y}}$. Thus, solving the inverse problem means to find some typical inputs which will give rise to the given outputs. Approximating the sigmoidal functions with 14, 28, and 40 grid points, we obtain the corresponding three ALP problems. The number of variables and the number of constraints[3] in each of the ALP problem are shown in Table 5. Solving the ALP problems by the simplex method with the restricted entry rule, we get 60 network inversions corresponding to the ten given outputs.

[3] In this example, all of the inequivalent constraints $\lambda \ge 0$ were omitted in the simulations.

The average CPU time on a SUN Ultra2 workstation for solving each of the ALP problems is also shown in Table 5. Fig. 7 illustrates 20 network inversions corresponding to the actual outputs of the 10 handwritten digits shown in Fig. 6, which were obtained by solving the SP problem with 28 grid points. Presenting the 60 network inversions to the trained network as novel inputs, we see that the outputs produced by the trained network are almost the same as the given outputs. This shows that the 60 network inversions are reasonable optimal solutions of Eq. (35).

7. Conclusions

We have shown that how FNNs such as MLPs, RBF networks, and MLQPs can be used to convert nonseparable functions into separable functions. Applying this useful feature and their universality of function approximation to nonlinear programming problems, we have presented a new method for transforming nonseparable nonlinear programming problems into separable programming problems. We have compared the proposed method with existing transformation techniques. The most important advantage of the proposed method is that it can be used to convert the functions whose concise expressions are unknown by training a network on a set of data. As a result, complex optimization problems can be formulated as separable programming problems even when their exact models are unknown. We have demonstrated the method and its applications through four examples. The simulation results show that our method might largely broaden the applicability of the simplex method to solving nonlinear programming problems and open up a new area of applying FNNs to mathematical programming and system optimization.

Acknowledgements

This work was partially supported by RIKEN Frontier Program. The first author would like to thank Dr M. Ichikawa and Prof. G. Matsumoto for their encouragement and support.

References

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*, 251–276.

Amari, S., Park, H., & Fukumizu, K. (2000). Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, *12*, 1399–1409.

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (1993). *Nonlinear programming: Theory and algorithms* (2nd ed). New York: Wiley.

Bishop, C. W. (1995). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Cherkassky, V., Gehring, D., & Mulier, F. (1996). Comparison of adaptive methods for function estimation from samples. *IEEE Transactions of Neural Networks*, *7*(4), 969–984.

Choi, C. H., & Choi, J. Y. (1994). Constructive neural networks with piecewise interpolation capabilities for function approximation. *IEEE Transactions of Neural Networks*, *5*(6), 936–944.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*, 303–314.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2*, 183–192.

Giles, C. L., & Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, *26*, 4972–4978.

Golinski, J. (1973). An adaptive optimization system applied to machine synthesis. *Mechanism and Machine Synthesis*, *8*, 419–436.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed). New York: Macmillan.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366.

Le Cun, Y., Boser, Y. B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten ZIP code recognition. *Neural Computation*, *1*(4), 541–551.

Lu, B. L., Bai, Y., Kita, H., & Nishikawa, Y. (1993). An efficient multiplayer quadratic perceptron for pattern classification and function approximation. *Proceedings of International Joint Conference on Neural Networks, Nagoya, Japan*, 1385–1388.

Lu, B. L., & Ito, K. (1995). Regularization of inverse kinematics for redundant manipulators using neural network inversions. *Proceedings of IEEE International Conference on Neural Networks, Perth*, 2726–2731.

Lu, B. L., & Ito, K. (1996). Solving inverse kinematics problem of redundant manipulators in an environment with obstacles using separable nonlinear programming. *Proceedings of Japan–USA Symposium on Flexible Automation, Boston*, 79–82.

Lu, B. L., & Ito, K. (1997). Transformation of nonlinear programming problems into separable ones using multilayer neural networks. In S. W. Ellacott, J. C. Mason, & I. J. Anderson (Eds.), *Mathematics of neural networks: Models, algorithms and applications* (pp. 235–239). Dordrecht: Kluwer Academic Publishers.

Lu, B. L., Kita, H., & Nishikawa, Y. (1999). Inverting feedforward neural networks using linear and nonlinear programming. *IEEE Transactions on Neural networks*, *10*(6), 1271–1290.

Luenberger, D. G. (1989). *Linear and nonlinear programming* (2nd ed). Reading, MA: Addison-Wesley.

McCormick, G. P. (1983). *Nonlinear programming*. New York: Wiley.

Miller, C. E. (1963). The simplex method for local separable programming. In R. L. Graves, & P. Wolfe (Eds.), *Recent advances in mathematical programming* (pp. 89–100). New York: McGraw-Hill.

Niyogi, P., & Girosi, F. (1996). On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. *Neural Computation*, *8*, 819–842.

Powell, M. J. D. (1978). A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Waston (Ed.), *Proceedings of the 1977 Dundee Conference on Numerical Analysis. Lecture Notes in Mathematics 630*, Berlin: Springer.

Rao, S. S. (1996). *Engineering optimization: Theory and practice* (3rd ed). New York: Wiley.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by backpropagating errors. *Nature*, *323*, 533–536.

Schilling, R. J. (1994). *Fundamentals of robotics: Analysis and control*. New Jersey: Prentice Hall.

Smith, M. (1993). *Neural networks for statistical modeling*. New York: Van Nostrand Reinhold.