Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2011, Article ID 145608, 16 pages
doi:10.1155/2011/145608

Research Article

Inverse Problem with Respect to Domain and Artificial Neural Network Algorithm for the Solution

Kambiz Majidzadeh[1,2]

[1] *Institute of Applied Mathematics, Baku State University, Baku 1148, Azerbaijan*
[2] *Institute of Information Technology, ANAS, Baku 1141, Azerbaijan*

Correspondence should be addressed to Kambiz Majidzadeh, kambiz823@yahoo.com

Received 25 April 2011; Accepted 21 June 2011

Academic Editor: Gradimir V. Milovanović

We consider the inverse problem with respect to domain. We suggested a new approach for reducing the inverse problem for a domain to an equivalent problem in a variational setting and gave an effective solution algorithm for solving such problems. In order to solve boundary problem, the artificial neural network is used in each step of the iteration.

1. Introduction

A wide class of practical problems are reduced to the inverse problem with respect to domain. As an example, we can show problems of elasticity theory, diffusion problems, the problems arising in hydrodynamics [1–6], and so forth. The papers concerning inverse problems usually deal with inverse problems for an unknown function (coefficients and functions occurring in the boundary and initial conditions). But in our case, a domain is sought and the investigation of the considered problems is related with some strong difficulties. In order to avoid these difficulties and to investigate such problems, firstly, the considered inverse problem is reduced to the variational statement. As the obtained variational problem is a domain-dependent variational problem, the investigation of such problems is also related to some difficulties. Here, we give an effective algorithm for solving such problems.

2. Statement of the Problem and Main Result

Let D be an r-dimensional domain, that is, $D \subset R^r$ and $x = (x_1, x_2, \ldots, x_r) \in D$.

Denote by S_D the boundary of the domain D, $S_D = \partial D$. Assume that the boundary S_D is in the space C^2. Let us consider the following inverse problem:

$$-\Delta u + a(x)u = f(x), \quad x \in D, \tag{2.1}$$

$$u(x) = 0, \quad x \in S_D, \tag{2.2}$$

$$\frac{\partial u(x)}{\partial n} = 0, \quad x \in S_D, \tag{2.3}$$

where the functions a are f and continuously differentiable functions in R^r and $a(x) > 0$. Denote by K the set of convex domains set with a boundary from C^2.

Our goal is to find a pair $(D, u) \in K \times C^2(D)$ such that the function $u = u(x)$ satisfies (2.1) and boundary conditions (2.2), (2.3) in the domain D. As it is seen, condition (2.2) is a Dirichlet condition, and condition (2.3) is a Neumann condition. For solving inverse problem (2.1)–(2.3), at first, we consider the following unknown domain variational problem:

$$J(D, u) = \int_D F(x, u(x), u_x(x))dx \longrightarrow \min, \quad D \in K,\ u \in C^2(D), \tag{2.4}$$

$$u(x) = 0, \quad x \in S_D. \tag{2.5}$$

We will assume that the function $F(x, u, p)$ is a continuously differentiable function of its own variables in $D \times R \times R^r$.

If boundary condition is satisfied for $D \in K,\ u \in C^2(D)$, the pair (D, u) is said to be a possible pair. Denote by M all possible pairs set. The pair $(D^*, u^*) \in M$ is called an optimal pair if it gives a minimum to functional (2.4) in the set M.

Give the following theorem obtained in [7].

Theorem 2.1. *Let the pair $(D^*, u^*) \in M$ be an optimal pair for variational problem* (2.4), (2.5). *Then the function $u^* = u^*(x)$ is a solution of the following Euler equation in the domain D^*:*

$$F_u(x, u(x), u_x(x)) - \sum_{i=1}^{n} \frac{d}{dx_i} F_{u_{xi}}(x, u(x), u_x(x)) = 0, \quad x \in D^*, \tag{2.6}$$

and moreover, in the boundary S_{D^*}, the condition

$$F(x, u^*(x), u_x^*(x)) - \sum_{i=1}^{n} u_{x_i}^*(x) F_{u_{x_i}}(x, u^*(x), u_x^*(x)) = 0, \quad x \in S_{D^*}, \tag{2.7}$$

is satisfied.

As it is seen, this theorem is proved for convex domains. But one can obtain the similar result for doubly connected domain D with internal and external boundaries S_1 and S_2. For that, we must use the expansion

$$\int_D F\,dx = \int_{D_2} F\,dx - \int_{D_1} F\,dx, \tag{2.8}$$

where D_1 and D_2 are convex domains bounded by the boundaries S_1 and S_2.

Now, take the function $F(x,u,p)$ in the following form

$$F(x,u,u_x) = |u_x|^2 + a(x)u^2 - 2f(x)u, \tag{2.9}$$

where

$$|u_x|^2 = |u_{x_1}|^2 + |u_{x_2}|^2 + \cdots + |u_{x_r}|^2. \tag{2.10}$$

As the functions a and f are continuously differentiable functions in R^r, the function $F(x,u,p)$ is a continuously differentiable function of its own variablies in $D \times R \times R^r$. Apply the theorem mentioned above to this function. It is clear that

$$F_u = 2a(x)u - 2f(x)u, \qquad F_{u_{x_i}} = 2u_{x_i}. \tag{2.11}$$

Hence,

$$\frac{d}{dx_i} F_{u_{x_i}} = 2\Delta u. \tag{2.12}$$

So, from condition (2.6), we get that the function $u^* = u^*(x)$ satisfies (2.1) in the domain D^*. From condition (2.7), we get the following boundary condition

$$|u_x^*|^2 + au^{*2} - 2f(x)u^* - 2|u_x^*|^2 = 0, \quad x \in S_D. \tag{2.13}$$

If we take into account the condition $u^*(x) = 0,\ x \in S_{D^*}$, we get

$$|u_x^*| = 0, \quad x \in S_{D^*}, \tag{2.14}$$

or

$$u_x^*(x) = 0, \quad x \in S_{D^*}. \tag{2.15}$$

From the equation of the derivative with respect to the normal $\partial u(x)/\partial n$, we get that function (2.3) satisfies the boundary condition (2.3) as well. Thus, we proved the following theorem.

Theorem 2.2. *Let the pair* $(D^*, u^*) \in M$ *be an optimal pair for problem* (2.4), (2.5). *Then it is a solution of problem* (2.1)–(2.3) *as well. This theorem shows that if instead of the inverse problem* (2.1)–(2.3) *we take the function* $F(x, u, p)$ *in the form* (2.9), *we can investigate the variational problem* (2.4), (2.5). *Notice that the inverse of this fact is not true in general. Though the functional* $J(D, u)$ *is convex with respect to the functional* u, *this functional is not convex with respect to* D *in general. Using the obtained results to solving problem* (2.4), (2.5), *we give the following method.*

Let the system of the functions $\{\varphi_k(x)\}$, $k = 1, 2, \ldots$ form a basis in the space $C^2(D)$. Then function $u = u(x)$ may be expanded by this basis

$$u = \sum_{i=1}^{\infty} \alpha_k \varphi_k(x). \tag{2.16}$$

We take this into account in problem (2.4), (2.5) and get

$$I(D, \alpha) = \int_D \Phi(\mathrm{x}, \alpha)\mathrm{dx} \longrightarrow \min, \quad D \in K, \ \alpha = (\alpha_1, \alpha_2, \alpha_3, \ldots), \tag{2.17}$$

$$\sum_{i=1}^{\infty} \alpha_k \varphi_k(x) = 0, \quad x \in S_D, \tag{2.18}$$

where

$$\Phi(x, \alpha) = F\left(x, \sum_{i=1}^{\infty} \alpha_k \varphi_k(x), \sum_{i=1}^{\infty} \alpha_k \frac{\partial \varphi_k(x)}{\partial x}\right). \tag{2.19}$$

In our case, as $F(x, u, p)$ is in the form (2.9),

$$\Phi(x, \alpha) = \left|\sum_{k=1}^{\infty} \alpha_k \frac{\partial \varphi_k}{\partial x}\right|^2 + a(x)\left|\sum_{i=1}^{\infty} \alpha_k \varphi_k(x)\right|^2 - 2f(x)\sum_{i=1}^{\infty} \alpha_k \varphi_k(x). \tag{2.20}$$

For solving problem (2.17), (2.18), calculate the first variation of functional (2.17). It is clear that

$$\frac{\partial I}{\partial \alpha} = \int_D \frac{\partial \Phi(x, \alpha)}{\partial \alpha} \delta\alpha \, dx. \tag{2.21}$$

Calculate the first variation of the functional $I(D, \alpha)$ with respect to the domain D. In [5, 7], the functional of the form

$$G(D) = \int_D g(x)dx \tag{2.22}$$

is considered, and, for its first variation, the formula

$$\delta G(D) = \int_{S_D} g(x)\delta P_D(n(x))ds \tag{2.23}$$

is obtained. Here, the function $g(x)$ is a continuously differentiable function in R^r, $n(x)$ is an external normal to the surface S_D at the point x and $P_D(x)$ is a support function of the domain D and is determined as follows:

$$P_D(x) = \sup_{l \in D}(l, x), \quad x \in R^r. \tag{2.24}$$

We take into account this formula and get

$$\delta I(D, \alpha) = \int_{S_D} \Phi(x, \alpha) \delta P_D(n(x)) ds + \int_D \frac{\partial \Phi(x, \alpha)}{\partial \alpha} \delta \alpha \, dx. \tag{2.25}$$

It is seen from this formula that the numbers $\alpha_1, \alpha_2, \alpha_3, \ldots$ are found from the system of equations

$$\int_D \frac{\partial \Phi(x, \alpha)}{\partial \alpha} dx = 0. \tag{2.26}$$

In our case, as $\Phi(x, \alpha)$ is in the form (2.20), the system of (2.26) will be a system of linear equations.

The set K may satisfy some additional restrictions as well. For example, the volume K may be the mentioned domains set, and domains set with the given area of surface. In another case, the domains set K may be given as $D_0 \subset D \subset D_1$ as well, where D_0, D_1, and R^r are the given domains. In practical problems, the set K may be given in the form of the integral restrictions

$$\int_D g(x) dx = c \tag{2.27}$$

or

$$\int_D g(x) dx \leq c. \tag{2.28}$$

In general, assume that there is a domain $G \subset R^r$ such that for arbitrary $D \in K$ contained in the set K, $D \subset G$. We can state this condition in a simpler form as follows.

We know that the optimal domain is contained in a certain domain G. The obtained relations (2.25), (2.26) enable to solve problem (2.17), (2.18) approximately. For that we give the following algorithm.

3. Algorithm for Numerical Solution

Based on the provided procedure, for numerical solution of the problem (2.4)-(2.5), the following methods are proposed.

Step 1. Take arbitrary domain $D^{(0)} \in K$ and the basis functions

$$\left\{\varphi_k^{(0)}(x)\right\}, \quad k = 1, 2, \ldots. \tag{3.1}$$

Step 2. Solving the system of equations

$$\int_{D^{(0)}} \frac{\partial \Phi(x,\alpha)}{\partial \alpha} dx = 0, \tag{3.2}$$

we find the convergence $\alpha^{(0)} = (\alpha_1^0, \alpha_2^0, \alpha_3^0, \ldots)$.

Step 3. Minimizing the linear functional

$$\int_{S_{D^{(0)}}} \Phi\left(x, \alpha^{(0)}\right) P_D(x) ds \longrightarrow \min, \quad D \in K, \tag{3.3}$$

we find the convex function $P(x)$.

Step 4. The intermediate domain $\overline{D}^{(0)}$ is found as a subdifferential of the function $P(x)$ at the point $x = 0$ [8]. In other words,

$$\overline{D}^{(0)} = \partial P(0) = \{l \in R^n;\ P(x) \geq (l, x),\ \forall x \in R^n\}. \tag{3.4}$$

Step 5. A new domain $D^{(1)}$ is found as follows:

$$D^{(1)} = (1 - \mu)\overline{D}^{(0)} + \mu D^{(0)}, \quad 0 < \mu < 1. \tag{3.5}$$

Here, the domain μ may be chosen in different ways [7, 9, 10].

If a new found domain $D^{(1)}$ satisfies definite exactness conditions, the iteration process is completed. On the contrary, for a new domain $D^{(1)}$, the iteration begins from the first step. The exactness condition may be given in different ways for example,

$$\left| I\left(D^{(k+1)}, \alpha^{(k+1)}\right) - I\left(D^{(k)}, \alpha^{(k)}\right) \right| < \varepsilon. \tag{3.6}$$

Here, $\varepsilon > 0$ is said to be accuracy order of the method. Now, give some rules for choosing the quantity μ_k.

(1) In general, the numbers μ_k may be chosen from the following condition:

$$f_k(\mu) = I\left((1 - \mu_k)D^{(k)} + \mu_k \overline{D}^{(k)}, \alpha_k\right) \longrightarrow \min, \quad \mu \in [0, 1]. \tag{3.7}$$

(2) The quantity μ_k may be given as a sequence satisfying the following conditions:

$$0 \le \mu_k \le 1, \quad \lim_{k\to\infty} \mu_k = 0, \quad \sum_{k=0}^{\infty} \mu_k = \infty, \tag{3.8}$$

for example,

$$\mu_k = \frac{1}{k+1}, \quad k = 1, 2, 3, \ldots. \tag{3.9}$$

(3) The another method is to take $\mu_k = 1$ and verity that the value of the functional decreases. If the value of the functional does not decrease, then the value of μ_k decreases twice. In order to solve boundary problem (2.1), (2.2), we will use the artificial neural network in each step of the iteration.

4. Application of Neural Networks to Solving the Boundary Problem

Before passing to solution of differential equations by means of neural networks, notice the cause of this necessity. It is known that there are many methods and approaches for solving differential equations and related boundary value problems. But there exist such boundary value problems that application of the known methods to them does not enable to find the solution with high accuracy because of many reasons. This may be connected with complexity of geometric structure of the domain where these differential equations are given and with strong nonlinearity of the problem. One of the reasons is that there are a great plenty of partition points, and this may strongly influence on error. At the same time, the solution of differential equation may require numerous iterations.

In addition, application of a neural network to the solution of differential equations is stipulated also by two reasons. The first reason is that logical sequence scheme corresponds to the logical scheme in the solution of differential equations [11–14]. In other words, the solution scheme of differential equations enables to create the appropriate structure of neural networks. The second reason is that neural networks can accurately approximate the function [15, 16].

Let $D \subset R^2$ and $S = \partial D$. Consider the following two-dimensional Poisson equation:

$$\begin{aligned} \Delta u &= f(x), \quad x \in D, \\ u(x) &= g(x), \quad x \in S. \end{aligned} \tag{4.1}$$

Find the solution of this problem as follows:

$$u(x) = \sum_{i=1}^{n} w_i \varphi_i(x), \tag{4.2}$$

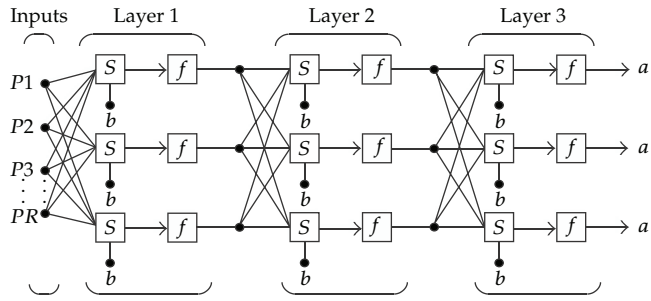


Figure 1: 1st layer is *Input layer* that contains *12 neuron* and *tansig* transfer functions. 2nd layer is *Hidden layer* that contains *18 neuron* and *tansig* transfer functions. 3rd layer is *Output layer* that contains *3 neuron* and *purelin* transfer functions.

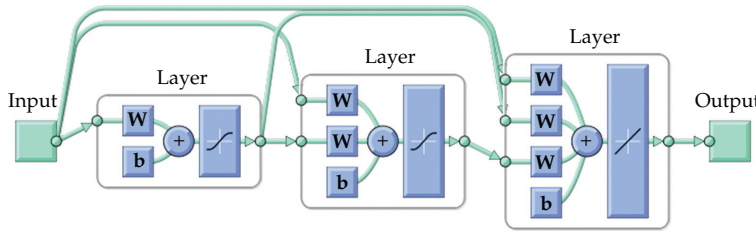


Figure 2

where the functions $\varphi_i(x)$ are usually chosen as radial-basis functions. Writing expansion (4.2) in (4.1), we get

$$\begin{aligned} \sum_{i=1}^{n} w_i \Delta\varphi_i(x) &= f(x), \quad x \in D, \\ \sum_{i=1}^{n} w_i \varphi_i(x) &= g(x), \quad x \in S. \end{aligned} \tag{4.3}$$

The weight coefficients $w_i,\ i = \overline{1,n}$ are found from the minimality condition of the following function:

$$J(w) = \int_D \left| \sum_{i=1}^{n} w_i \Delta\varphi_i(x) - f(x) \right|^2 dx + \int_S \left| \sum_{i=1}^{n} w_i \varphi_i(x) - g(x) \right|^2 ds \longrightarrow \min. \tag{4.4}$$

But, in great majority of cases, the functions

$$J(w) = \sum_{k=1}^{M_1} \left| \sum_{i=1}^{n} w_i \Delta\varphi_i(x_k) - f(x_k) \right|^2 + \sum_{k=1}^{M_2} \left| \sum_{i=1}^{n} w_i \varphi_i(s_k) - g(s_k) \right|^2 \longrightarrow \min \tag{4.5}$$

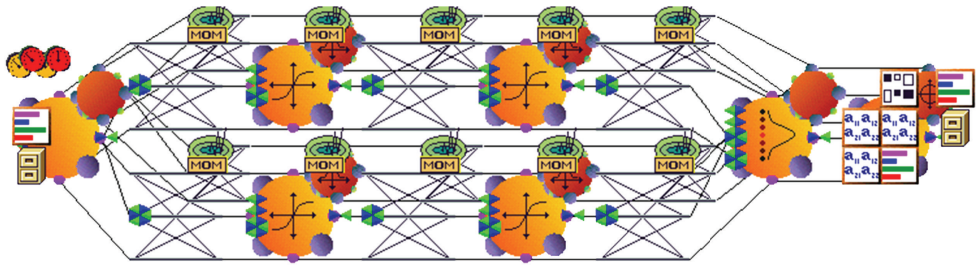


Figure 3

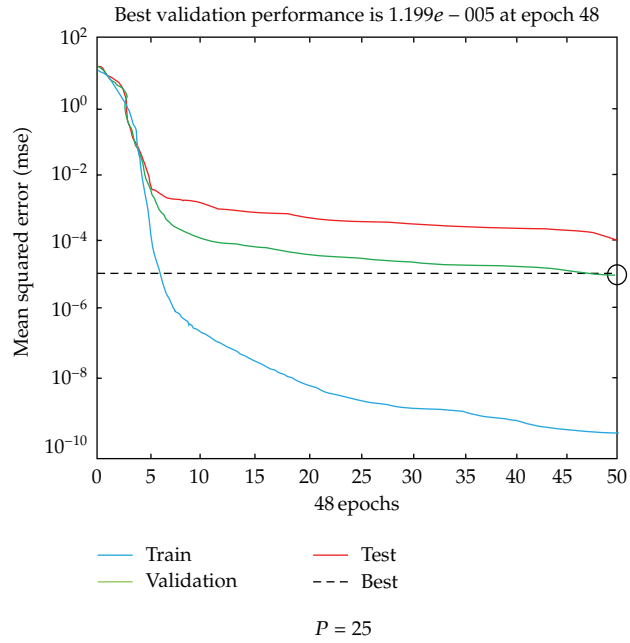


Figure 4: Validation performance.

are considered using the uniformity of the equation in the values at discrete points. In order to find the minimum of this function, we can use the gradient method. The iteration may be constructed to find the weight coefficients w_i, $i = \overline{1,n}$.

Now, in this way, we will try to solve a boundary value problem stated for partial equations by means of a neural network. We will consider problem (4.1). If we write the solution of this problem by means of the Green function, we will see that the solution is linearly and continuously dependent on the right hand side of the boundary function. As it was noted, the neural networks allow to approximate the function with any accuracy [15, 16]. Replace the domain $D \subset R^2$ by a regular discrete network by means of a small step h. Let the value of the function $f(x)$ at the internal nodal points be f_{ij}. Let I_1 be an indices set and denote the data at the boundary point by g_{pq}, the indices set by I_2. Our goal is to construct a neural network by changing the functions $f(x), g(x)$. To get input and output data, take some $u_1(x), u_2(x), \ldots, u_M(x)$ functions. For these functions to be the solutions

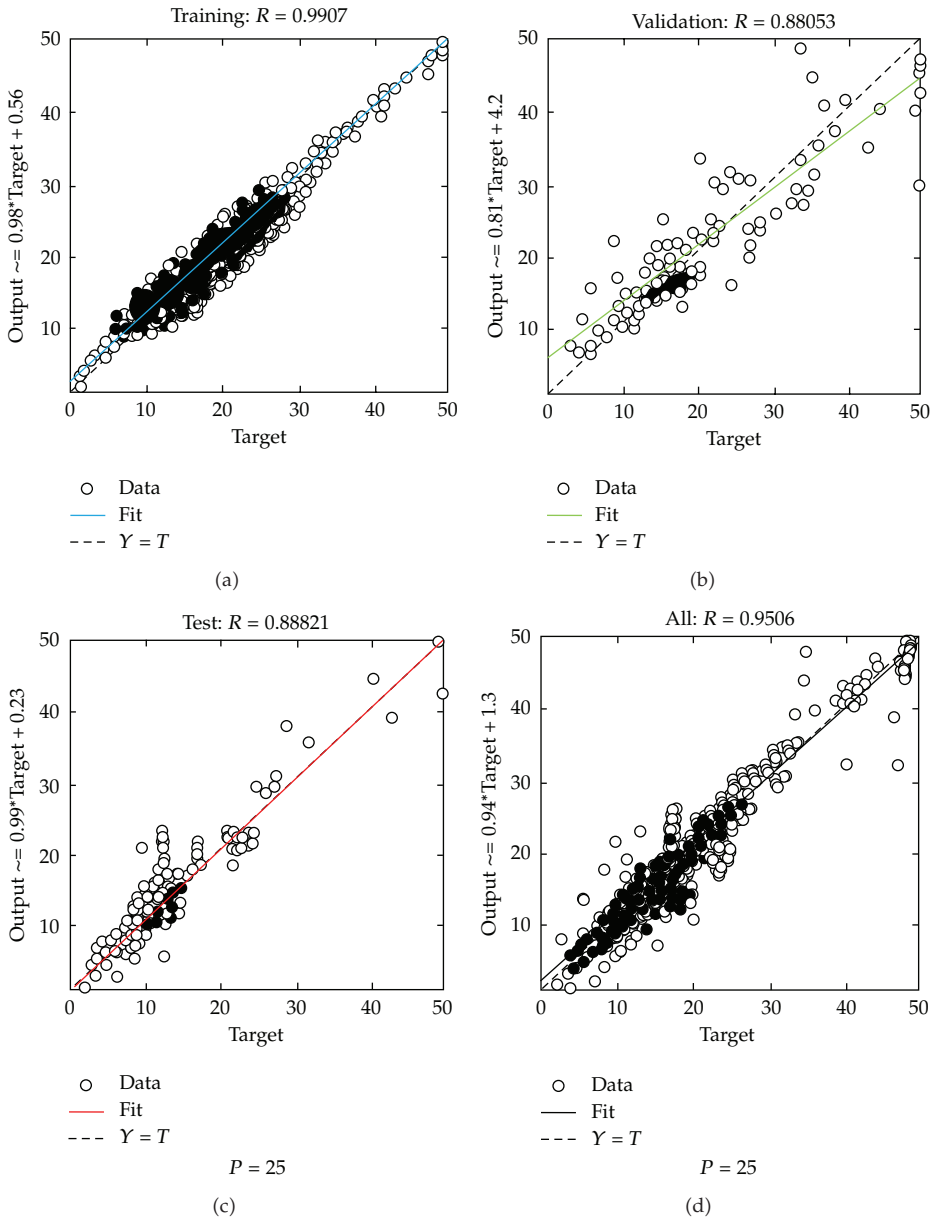
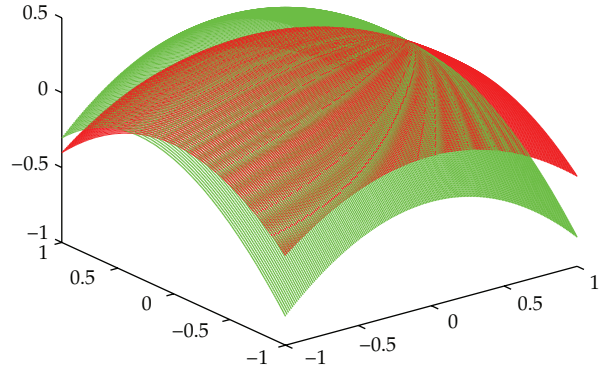


Figure 5: Regression performance.

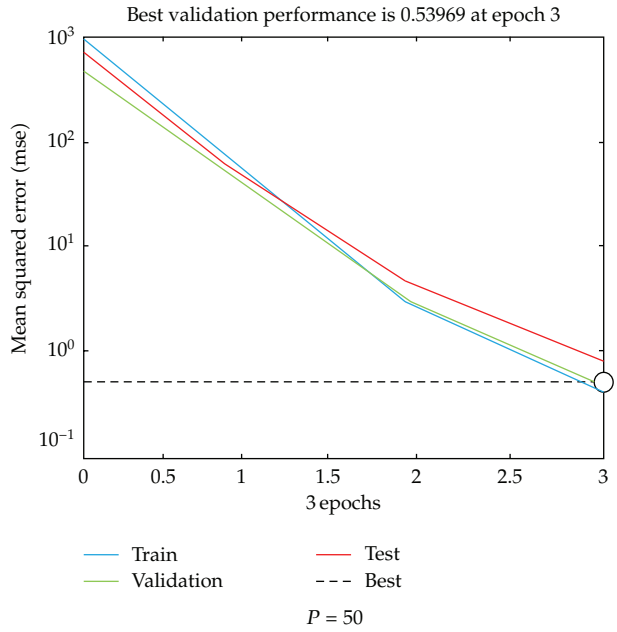
of (2.3), the function $f(x)$ should be chosen as the functions $f^{(1)}(x), f^{(2)}(x), \ldots, f^{(M)}(x)$, respectively. For satisfying the boundary condition, the function $f(x)$ must be defined as $g^{(1)}(x), g^{(2)}(x), \ldots, g^{(M)}(x)$ in the boundary. In other words, if, in problem (2.3), (2.4), we take $f(x) = f_k(x)$ and $g(x) = g_k(x)$, we get the appropriate solution $u(x) = u_k(x)$. So we must construct such a neural network that it could associate the M number of inputs

$$H_k = \left\{ f_{ij}^{(k)}, (i,j) \in I_1,\ g_{pq}^{(k)}, (p,q) \in I_2 \right\} \tag{4.6}$$



$P = 25$

Figure 6



$P = 50$

Figure 7: Validation performance.

to the M number of outputs

$$U_k = \left\{ u_{ij}^{(k)}, (i,j) \in I_1 \right\}. \tag{4.7}$$

Here, $u_{ij}^{(k)}$ is the value of the chosen solution $u_k(x)$ at the nodal points $(i,j) \in I_1$. The constructed neural network enables to find an approximate solution of problem (2.3), (2.4) for arbitrarily given $f(x), g(x)$. For attaining it, we must insert the set of

$$H = \left\{ f_{ij}, (i,j) \in I_1,\ g_{pq}, (p,q) \in I_2 \right\} \tag{4.8}$$

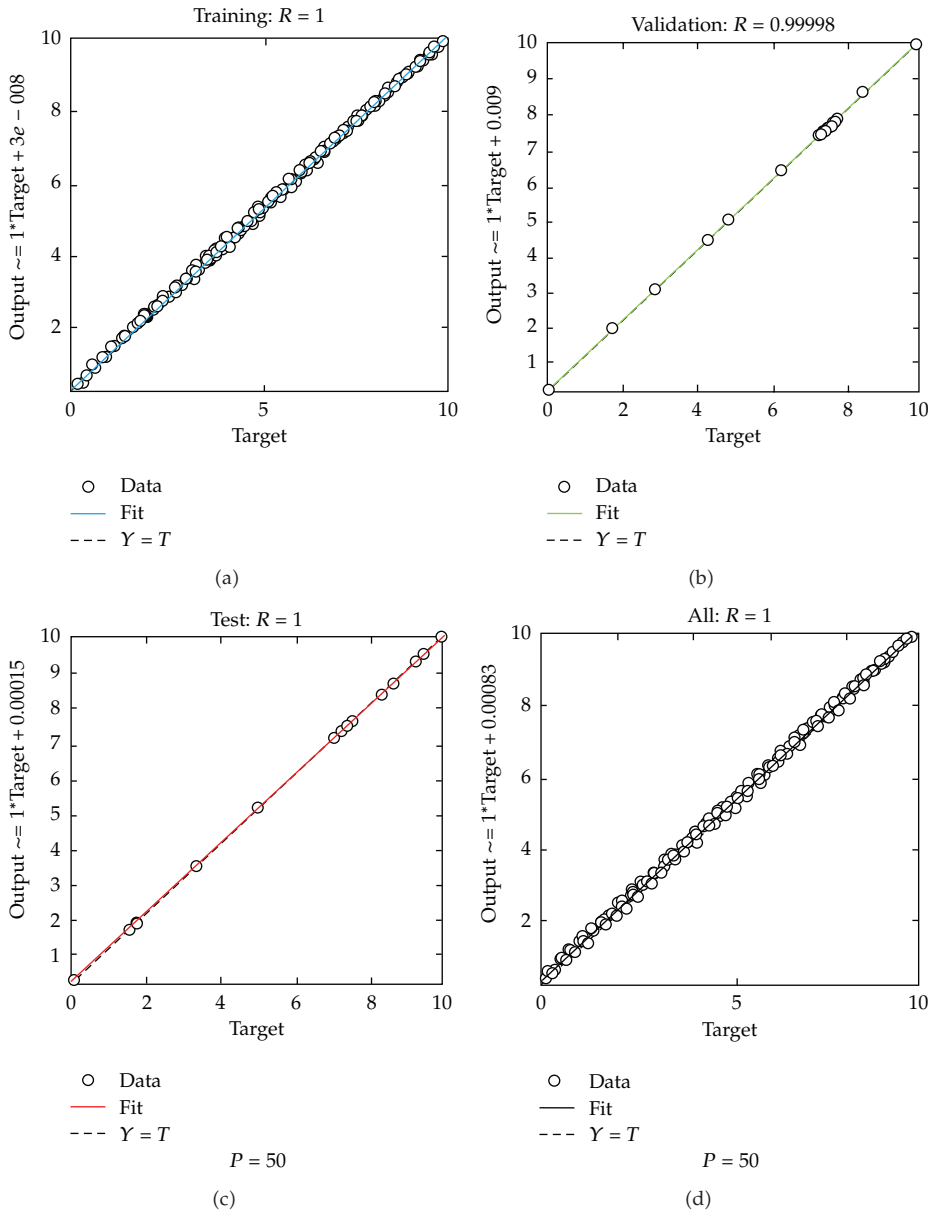


Figure 8: Regression performance.

as an input variable and, in this case, the set of

$$U = \{u_{ij}, (i,j) \in I_1\} \tag{4.9}$$

as an output variable to the neural network. Then, the neural network will be an approximate solution of problem (4.1).

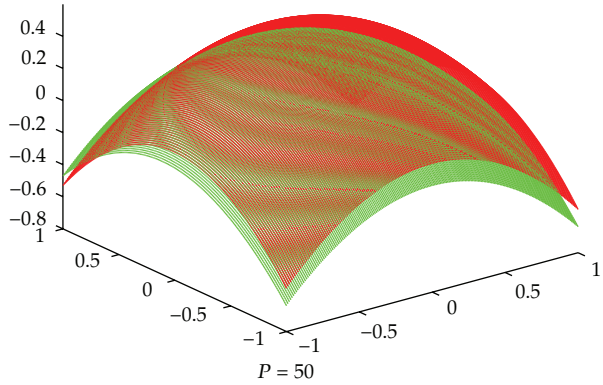


Figure 9

5. Model Example

Now let us consider the following example.

Consider the following problem:

$$\begin{aligned} \Delta u &= -2, \quad x \in D, \\ u(x) &= 0, \quad x \in S, \end{aligned} \tag{5.1}$$

where

$$D = \left\{ x = (x_1, x_2) \in R^2 : x_1^2 + x_2^2 \leq 1 \right\}. \tag{5.2}$$

The used neural network is nonliner "*cascade feed forward—distributed time delay—backpropagation*" with levenberg-marqwardt algorithm (train LM) that needs 3 layers (Figure 1).

Backpropagation is the generalization of the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined.

The relationship between I/O, weight, and biases is shown as (Figure 2), include a weight connection from the input to each layer, and from each layer to the successive layers.

The additional connections might improve the speed at which the network learns the desired relationship. Each layer has a weight matrix **W**, a bias vector **b**, and an output vector **a**. Networks with wight, biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities.

Now we can start creating network through command structure:

```
NET=newcf (minmax(P),[12, 18, 3], {"tansig", "tansig", "purelin"},"trainlm");
NET=init (NET);
NET.trainparam.show=50;
NET.trainparam.lr=0.05;
```

```
NET.trainparam.lr_inc=1.05;
NET.trainparam.mc=0.9;
NET.trainparam.epochs=300;
NET.trainparam.goal=1e − 5;
[NET,tr]=train (NET,P,T);

Network =
  Neural Network object:

  architecture:
  numInputs: 1
  numLayers: 3
  biasConnect: [1; 1; 1]
  inputConnect: [1; 1; 1]
  layerConnect: [0 0 0; 1 0 0; 1 1 0]
  outputConnect: [0 0 1]
  numOutputs: 1 (read-only)
  numInputDelays: 0 (read-only)
  numLayerDelays: 0 (read-only)

  subobject structures:
  inputs: {1 × 1 cell} of inputs
  layers: {3 × 1 cell} of layers
  outputs: {1 × 3 cell} containing 1 output
  biases: {3 × 1 cell} containing 3 biases
  inputWeights: {3 × 1 cell} containing 3 input weights
  layerWeights: {3 × 3 cell} containing 3 layer weights

  functions:
  adaptFcn: "trains"
  divideFcn: "dividerand"
  gradientFcn: "gdefaults"
  initFcn: "initlay"
  performFcn: "mse"
  plotFcns: {"plotperform","plottrainstate","plotregression"}
  trainFcn: "trainlm"
```

parameters:

adaptParam: .passes

divideParam: .trainRatio, .valRatio, .testRatio

gradientParam: (none)

initParam: (none)

performParam: (none)

trainParam: .show, .showWindow, .showCommandLine, .epochs,
.time, .goal, .max_fail, .mem_reduc,
.min_grad, .mu, .mu_dec, .mu_inc,

weight and bias values:

IW: {3 × 1 cell} containing 3 input weight matrices

LW: {3 × 3 cell} containing 3 layer weight matrices

b: {3 × 1 cell} containing 3 bias vectors.

After initializing and adjusting the train and weight parameter, the final created form of the artificial neural network is shown in Figure 3.

After the training network with $P = 25$ input and output data, Figure 4 shows the result. The result here isn't reasonable, because the test set error and the validation set error do not have similar characteristics, and it does appear that any significant overfitting has occurred.

Figure 5 shows the net results to perform some analysis of the network response. In this case, there are four outputs, so there are four regressions.

Solution of the considered model example is $u(x) = (1/2)(1 - x_1^2 + x_2^2)$.

We found an approximate solution of this problem for $P = 25$ by using a neural network. These results are shown in Figure 6.

For reaching the high accuracy, we should train the network with more data. We retrain the net with $P = 50$ data again. The results are shown in Figures 7 and 8.

The artificial neural network's error is reasonable, because the test set error and the validation set error have similar characteristics, and it does not appear that any significant overfitting has occurred.

We found an approximate solution for $P = 50$ again. These results are shown in Figure 9.

For improving the high accuracy of approximate solution, the network is capable to train with more data.

References

[1] J. Sea, "Numerical method in mathematical physics and optimal control," *Publishing House Nauka*, vol. 240, pp. 64–74, 1978.

[2] J. Sokolowski and J.-P. Zolesio, *Introduction to Shape Optimization. Shape Sensitivity Analysis*, Springer, Heidelberg, Germany, 1992.

[3] J. Haslinger and R. A. E. Makinen, *Introduction to Shape Optimization: Theory, Approximation and Computation*, SIAM, Philadelphia, Pa, USA, 2003.

[4] Y. S. Gasimov, A. Nachaoui, and A. A. Niftiyev, "Non-linear eigenvalue problems for p-Laplacian with variable domain," *Optimization Letters*, vol. 4, no. 1, pp. 67–84, 2009.
[5] F. A. Aliev, M. M. Mutallimov, I. M. Askerov, and I.S. Ragumov, "Asymptotic method of solution for a problem of construction of optimal gas-lift process modes," *Mathematical Problems in Engineering*, vol. 2010, Article ID 191153, 11 pages, 2010.
[6] A. A. Niftiyev and E. R. Akhmadov, "Variational statement of an inverse problem for a domain," *Journal Differential Equation*, vol. 43, no. 10, pp. 1410–1416, 2007.
[7] S. Belov and N. Fujii, "Summery and sufficient conditions of optimality in a domain optimization problem," *Control and Cybernetics*, vol. 26, no. 1, pp. 45–56, 1997.
[8] F. A. Aliev, A. A. Niftiyev, and J. I. Zeynalov, "Optimal synthesis problem for the fuzzy systems in semi-infinite interval," *Applied and Computational Mathematics*, vol. 10, no. 1, pp. 97–105, 2011.
[9] V. F. Demyanov and A. M. Rubinov, *Bases of Non-Smooth Analysis and Quasidifferential Calculus*, Nauka, Moscow, Russia, 1990.
[10] F. P. Vasilyev, *Optimization Methods*, Factorial Press, Moscow, Russia, 2002.
[11] D. A. Tarkhov, *Neural Networks: Models and Algorithms*, Radio-Technical, Moscow, Russia, 2005.
[12] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.
[13] R. M. Alguliev, R. M. Aliguliev, and R. K. Alekperov, "New approach to optimal appointment for distributed system, informatics and computer science," *Problems*, no. 5, pp. 23–31, 2004.
[14] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks: controllability and stabilization," *IEEE Transactions on Neural Networks*, vol. 4, no. 2, pp. 192–206, 1993.
[15] H. Lee and I. S. Kang, "Neural algorithm for solving differential equations," *Journal of Computational Physics*, vol. 91, no. 1, pp. 110–131, 1990.
[16] A. N. Gorban, "A generalized approximation theorem and computing possibilities of neural networks," *Siberian Journal of Calculate Mathematics*, vol. 1, no. 1, pp. 12–24, 1998.