International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:12, 2009

# Artificial Neural Network with Steepest Descent Backpropagation Training Algorithm for Modeling Inverse Kinematics of Manipulator

Thiang, Handry Khoswanto, Rendy Pangaldus

***Abstract*—**Inverse kinematics analysis plays an important role in developing a robot manipulator. But it is not too easy to derive the inverse kinematic equation of a robot manipulator especially robot manipulator which has numerous degree of freedom. This paper describes an application of Artificial Neural Network for modeling the inverse kinematics equation of a robot manipulator. In this case, the robot has three degree of freedoms and the robot was implemented for drilling a printed circuit board. The artificial neural network architecture used for modeling is a multilayer perceptron networks with steepest descent backpropagation training algorithm. The designed artificial neural network has 2 inputs, 2 outputs and varies in number of hidden layer. Experiments were done in variation of number of hidden layer and learning rate. Experimental results show that the best architecture of artificial neural network used for modeling inverse kinematics of is multilayer perceptron with 1 hidden layer and 38 neurons per hidden layer. This network resulted a RMSE value of 0.01474.

***Keywords*—**Artificial neural network, backpropagation, inverse kinematics, manipulator, robot.

## I. Introduction

ARTIFICIAL neural networks (ANNs) are simplified models of the central nervous system. It is believed by many researches in the field that neural network models offer the most promising unified approach to build truly intelligent computer systems. ANNs have been shown to be effective as computational processors for various tasks including pattern recognition, associative recall, classification, data compression, modeling and forecasting, adaptive control and noise filtering[1].

Reference [2] describes one of applications of ANN in control system. ANN is applied for controlling one arm robot. In this application, ANN is used to model the second order controller so that ANN can act as the controller. Architecture of ANN used in this system is fully connected multilayer perceptron with steepest descent backpropagation training algorithm. The result shows that ANN can control one arm robot successfully. Reference [3] describe about another application of ANN in control system. ANN is applied to control the speed of DC motor. In this application, ANN also acts as the controller and the result shows that ANN can control the speed of DC motor well. Felix Pasila and friends have successfully applied ANN combined with Fuzzy Logic for electrical load forecasting application [4][5][6].

This paper describes another application of ANN. Because it is not easy to derive the inverse kinematics equation of a robot manipulator, the ANN is applied for modeling the inverse kinematics equation of a robot manipulator. The ANN can learn by itself the training data so that it can create inverse kinematics model of the robot. The training data are created from the experiment. In this research, the robot has three degree of freedoms and the robot was implemented for drilling a printed circuit board. By performing the ANN to learn the training data, we do not need to derive the inverse kinematics equation. ANN can create the model of inverse kinematics automatically and of course, the model is not in form of mathematic equation, but, the model is in form of ANN structure include the weight and bias connection of the network**.**

## II. Inverse Kinematics of Robot Manipulator

In this research, the robot manipulator has three degree of freedoms, which consists of two revolute joints and one prismatic joint. Figure 1 shows picture of the robot manipulator. The prismatic joint moves the third link up and down when the drilling process is being done. A drill tool is attached at the third link

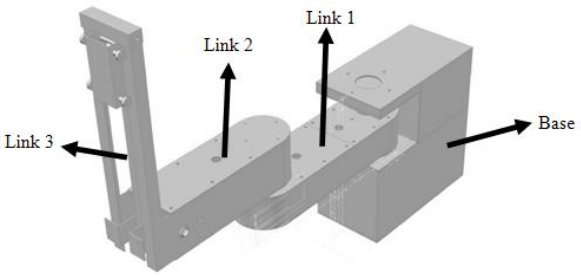


Fig. 1 Three degree of freedom robot manipulator with two revolute joints and one prismatic joint

Thiang is with the Electrical Engineering Department, Petra Christian University, Surabaya, 60236 Indonesia (phone: 62-31-2983115; fax: 62-31-8417658; e-mail: thiang@petra.ac.id).

Handry Khoswanto, is with the Electrical Engineering Department, Petra Christian University, Surabaya, 60236 Indonesia (e-mail: handry@petra.ac.id)

Rendy Pangaldus was with the Electrical Engineering Department, Petra Christian University, Surabaya, 60236 Indonesia.

Inverse kinematics is needed for calculating the position of each joint for a set of given tool coordinate. Because the third link only moves full up and full down when the drilling process is being done, the inverse kinematics is only considered for two revolute joints, which determine the drilling position coordinate. For a given tool coordinate (*x, y*), the position of the two revolute joints ($\theta i$) can be defined as follows:

$$\theta_i = f(x, y) \tag{1}$$

Assume that there is a coordinate system for the robot manipulator as shown at figure 2.

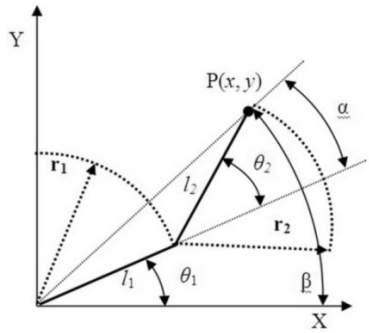


Fig. 2 Coordinate system of the robot manipulator

The position of point *P(x,y)* can be determined by using following equation:

$$P(x, y) = f(\theta_1, \theta_2) \tag{2}$$

Assume that vector *P* is determined by adding vector *r1* and vector *r2* where *r1* and *r2* can be defined as follows:

$$r_1 = [l_1 \cos\theta_1, l_1 \sin\theta_1] \tag{3}$$

$$r_2 = [l_2 \cos(\theta_1 + \theta_2), l_2 \sin(\theta_1 + \theta_2)] \tag{4}$$

then

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \tag{5}$$

$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \tag{6}$$

Equation (5) and (6) are forward kinematics equation of the robot manipulator with two revolute joints.

The inverse kinematics equation can be derived from (5) and (6). By using trigonometry equation, (5) and (6) can be written as follows:

$$x = l_1 \cos\theta_1 + l_2 \cos\theta_1 \cos\theta_2 - l_2 \sin\theta_1 \sin\theta_2 \tag{7}$$

$$y = l_1 \sin\theta_1 + l_2 \sin\theta_1 \cos\theta_2 + l_2 \cos\theta_1 \sin\theta_2 \tag{8}$$

Solving (7) and (8) to find value of $\theta_1$ and $\theta_2$ results the following equation:

$$\theta_1 = \tan^{-1}\left[\frac{y(l_1 + l_2\cos\theta_2) - x.l_2\sin\theta_2}{x(l_1 + l_2\cos\theta_2) + y.l_2\sin\theta_2}\right] \tag{9}$$

$$\theta_2 = \cos^{-1}\left[\frac{x^2 + y^2 - {l_1}^2 - {l_2}^2}{2l_1 l_2}\right] \tag{10}$$

Equation (9) and (10) are inverse kinematics equation of the robot manipulator, which has two revolute joints. For a set of given coordinate *P*(x,y), the position of each link ($\theta_1$, $\theta_2$) can be calculated by using (9) and (10)

Solving (5) and (6) in order to derive inverse kinematics equation is not so easy. The more is number of degree of freedom of the manipulator, the more is difficulty to derive the inverse kinematics equation. That is why in this research, artificial neural network is used to model the inverse kinematics equation of the manipulator.

## III. Modeling Inverse Kinematics Using Artificial Neural Network

As explained in the previous section, in this research, artificial neural network was applied for modeling the inverse kinematics of the robot manipulator and the model is only considered for the kinematic of two revolute joints. The architecture of artificial neural network used in this research is a multi layer perceptron with steepest descent backpropagation training algorithm. Figure 3 shows the architecture of artificial neural network used for modeling inverse kinematic of the robot manipulator.

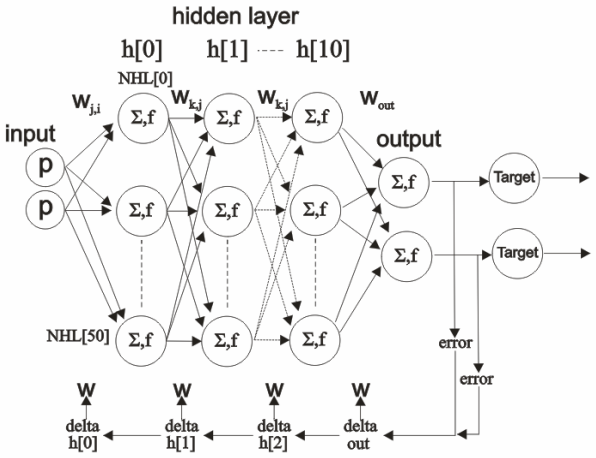


Fig. 3 Architecture of artificial neural network with backpropagation training algorithm

The network has two input neurons. Both input neurons receive the information of tool coordinate *P(x,y).* The output layer of the network has two neurons. Both neurons in the output layer give information about the position of each link ($\theta_1$, $\theta_2$). Number of hidden layer of the network varies from 1 to 10 layers and number of neuron per hidden layer varies from 1 to 50 neurons. If the artificial neural network has *m* layers and receives input of vector ***p***, then the output of the

network can be calculated by using the following equation:

$$a^m = f^m(W^m f^{m-1}(W^{m-1} f^{m-2}(...W^2 f^1(W^1 p + b^1) + b^2) + b^{m-1}) + b^m) \quad (11)$$

Where $\boldsymbol{f}^m$ is log-sigmoid transfer function of the m[th] layer of the network that can be defined as following equation:

$$f(n) = \frac{1}{1+e^{-n}} \quad (12)$$

$\boldsymbol{W}^m$ is weight of the m[th] layer of the network, and $\boldsymbol{b}^m$ is bias of the m[th] layer of the network. Equation (11) is known as the feed forward calculation.

Backpropagation algorithm is used as the training method of the designed artificial neural network. The backpropagation algorithm includes the following steps:

1. Initialize weights and biases to small random numbers.
2. Present a training data to neural network and calculate the output by propagating the input forward through the network using (11).
3. Propagate the sensitivities backward through the network:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t}-\mathbf{a}) \quad (13)$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}, \text{ for } m = M-1,...,2,1 \quad (14)$$

Where

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{s^m}^m) \end{bmatrix} \quad (15)$$

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m} \quad (16)$$

4. Calculate weight and bias updates

$$\Delta\mathbf{W}^m(k) = -\alpha\,\mathbf{s}^m(\mathbf{a}^{m-1})^T \quad (17)$$

$$\Delta\mathbf{b}^m(k) = -\alpha\,\mathbf{s}^m \quad (18)$$

Where α is learning rate.

5. Update the weights and biases

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) + \Delta\mathbf{W}^m(k) \quad (19)$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) + \Delta\mathbf{b}^m(k) \quad (20)$$

6. Repeat step 2 – 5 until error is zero or less than a limit value.

## IV. Experimental Result

Experiment was done to see performance of artificial neural network in order to model the inverse kinematic of the robot manipulator. The performance of artificial neural network is indicated by the RMSE value. Experiment was done in various training parameter value of artificial neural network, i.e., various numbers of hidden layers, various number of neuron per layer, and various value of learning rate. There were 200 training data used in this research for training the artificial neural network. The training data were created by running the robot manually and measuring the position of each link directly.

Table I shows the experimental result in various learning rate and number of neuron per hidden layer for the artificial neural network using 1 hidden layer. Table II shows the experimental result in various learning rate and number of neuron per hidden layer for the artificial neural network using 2 hidden layers, and table III shows the experimental result in various learning rate and number of neuron per hidden layer for the artificial neural network using 3 hidden layers.

TABLE I
RMSE Value of Network Using 1 Hidden Layer

| Neuron in Hidden Layer | Learning Rate | | | | |
|---|---|---|---|---|---|
| | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
| 1 | 0.05005 | 0.04715 | 0.04494 | 0.04358 | 0.04274 |
| 12 | 0.04382 | 0.0409 | 0.03936 | 0.0390 | 0.03823 |
| 25 | 0.04128 | 0.0360 | 0.03591 | 0.03535 | 0.03678 |
| 38 | 0.01474 | 0.03484 | 0.03642 | 0.03946 | 0.04029 |
| 50 | 0.0324 | 0.03658 | 0.04197 | 0.05223 | 0.05903 |

From Table I, it is shown that the best performance is resulted by artificial neural network with 1 hidden layer, 38 neurons per hidden layer, and learning rate 0.1. The best RMSE value is 0.01474. We can also see that the less number of neuron per layer, the higher RMSE value.

TABLE II
RMSE Value of Network Using 2 Hidden Layers

| Neuron in Hidden Layer | Learning Rate | | | | |
|---|---|---|---|---|---|
| | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
| 1 | 0.05047 | 0.06854 | 0.05752 | 0.05151 | 0.04695 |
| 12 | 0.04419 | 0.04235 | 0.04052 | 0.04021 | 0.03932 |
| 25 | 0.04303 | 0.03806 | 0.03819 | 0.03726 | 0.03656 |
| 38 | 0.03816 | 0.03507 | 0.0351 | 0.03491 | 0.03569 |
| 50 | 0.03841 | 0.0336 | 0.03433 | 0.03514 | 0.03486 |

Table II shows that the best performance is resulted by artificial neural network with 2 hidden layers, 50 neurons per hidden layer, and learning rate 0.3. The best RMSE value is

0.0336. And Table III shows that the best performance is resulted by artificial neural network with 3 hidden layers, 50 neurons per hidden layer, and learning rate 0.3. The best RMSE value is 0.03377. Overall, the best performance is resulted by artificial neural network with 1 hidden layer and 38 neurons per hidden layer.

Another experiment was done to see performance of overall system. Artificial neural network was trained by using 200 training data. Training process results the weight and bias of the network. This indicates inverse kinematic model of robot manipulator. For testing, the network was tested by using data that are not used for training process. And experimental result shows that artificial neural network can model the inverse kinematic of robot manipulator with average error of 2.16%.

## V. Conclusion

From the experimental result, it can be concluded that the artificial neural network can be used to model the inverse kinematic of robot manipulator. By using artificial neural network, we do not need to derive mathematic model of inverse kinematic of the robot. The inverse kinematic of the robot is modeled into form of number of hidden layer, number of neurons per layer, weight and bias value of the network.

In this case, the best model is achieved by artificial neural network with 1 hidden layer, 38 neurons per layer. And this model gives RMSE value of 0.01474.

## References

[1] Dan W. Patterson. *Artificial Neural Networks, Theory and Applications*. Singapore: Prentice Hall, 1996.

[2] Thiang, Rianto Chandra, Iwan Njoto Sandjaja, "One Arm Robot Position Control Using Artificial Neural Network," in *Proceedings of National Seminar: The Application of Technology Toward Better Life*. Yogyakarta, 2005.

[3] Thiang, Indar Sugiarto, Hendrik Chandra, "DC Motor Speed Control System Using Artificial Neural Network," in *Proceedings of National Seminar of Computer Science and Information Technology,* Jakarta, 2004.

[4] Felix Pasila, "Multivariate Inputs for Electrical Load Forecasting on Hybrid Neuro-Fuzzy and Fuzzy C-Means Forecaster," in *Proceedings of International Conference on Fuzzy Systems*, Hongkong, 2008.

[5] Felix Pasila, Sautma Ronni, Thiang, Lie Hendra Wijaya, "Long-term Forecasting in Financial Stock Market using Accelerated LMA on Neuro-Fuzzy Structure and Additional Fuzzy C-Means Clustering for Optimizing the GMFs," in *Proceedings of International Joint Conference on Neural Networks*. Hongkong, 2008.

[6] Felix Pasila, "Neuro-Fuzzy Forecaster for Modeling and Forecasting Electrical Load Competition Using Multivariate Inputs on Takagi-Sugeno Networks," in *Proceedings of International Conference on Soft Computing, Intelligent System, and Information Technology*. Bali, 2007.

[7] A.M.S., Zalzala and Morris, A.S., *Neural Network For Robotic Control, Theory And Application*. London: Ellis Horwood, 1996.

[8] Craig, John J., *Introduction to Robotics: Mechanics and Control*. New Jersey: Prentice Hall, 2005.

[9] 9. Martin T. Hagan, *Neural Network Design.* Boston: PWS Publishing, 1996.

[10] Zurada, J.M., *Introduction To Artificial Neural Systems*. Boston: PWS Publishing, 1992.