Neural Network-Based Inverse Kinematics for an Industrial Robot and Its Learning Method

Fusaomi Nagata^{a,*}, Seiya Kishimoto^a, Shingo Kurita^a, Akimasa Otsuka^a, Keigo Watanabe^b

^aTokyo University of Science, Yamaguchi, 1-1-1 Daigaku-Dori, Sanyo-Onoda 756-0884, Japan

^bOkayama University, 3-1-1 Tsushima-naka, Kita-ku, Okayama 700-8530, Japan

*Corresponding Author: nagata@rs.tus.ac.jp

Abstract

The time required for the learning process of neural networks depends on the number of total weights and that of the input-output pairs in the training set. In the proposed learning process, after the learning is progressed, e.g., 1000 iterations, input-output pairs having had worse errors are extracted from the original training set and form a new temporary set. From the next iteration, the temporary set is applied instead of the original set. This means that only pairs with worse errors are used for updating the weights until the mean value of errors decreases to a level. After the learning using the temporary set, the original set is applied again instead of the temporary set. By alternately applying the two types of sets for learning that the calculation load for convergence can be efficiently reduced. The effectiveness of the proposed method is proved by applying to an inverse kinematics problem of an industrial robot.

Keywords: Neural network, Efficient weights tuning, Temporary training set, Inverse kinematics problem, Industrial robot

1. Introduction

When designing a serial link structure for a multi-legged robot, first of all, its inverse kinematics problem must be solved. Hoang et al. proposed a differential kinematics algorithm to generate omni-directional walking trajectory of a leg based on back-stepping control using Lyapunov stability⁽¹⁾. Simulation results for walking motion of one leg of a six-legged robot were shown to prove the effectiveness and applicability of the proposed controller. Tejomurtula and Kak proposed a solution of inverse kinematics concerning a simple two link manipulator⁽²⁾. Duka designed a feedforward neural network to solve the inverse kinematics problem of a

three-link planar manipulator⁽³⁾. Also, Maeda et al. designed a position control scheme for actual robot systems using high dimensional neural networks, in which complex-valued neural network and quaternion neural network learned the inverse kinematics of the robot systems. Two dimensional SCARA robot and three dimensional robot were well controlled using the inverse kinematics⁽⁴⁾. Further, Koker et al. proposed a neural network based inverse kinematics solution for a three joints manipulator⁽⁵⁾ and then neural networks and genetic algorithms were fused together to solve the inverse kinematics problem of a Stanford robotic manipulator with six-DOFs while minimize the error at the end effector⁽⁶⁾. Hasan et al. also reported the results about neural networks' based inverse kinematics solution for a three joints serial robot manipulator passing through singularities⁽⁷⁾.

Generally, in making a neural network learn a nonlinear relation properly, a desired training set prepared in advance is used. The training set consists of multiple pairs of an input vector and an output one. Each input vector is given to the input layer for forward calculation, and the corresponding output vector is compared with the vector yielded from the output layer. Also, backward calculation means updating the weights using a back propagation algorithm. One cycle consists of one forward calculation and backward one. The time required for the learning process of the neural network depends on the number of total weights in the neural network and that of the input-output pairs in the training set. Especially, it is by no means easy for some of input-output pairs to converge to a desired error level. As a research for reducing calculation load, Kanan and Khanian proposed a method to reduce training time using an adaptive fuzzy approach⁽⁸⁾. The fuzzy approach was employed to control the learning parameters such as learning rate and momentum.

This paper describes a neural network with an efficient

weights tuning ability in order to effectively learn the inverse kinematics of an industrial robot. In the proposed learning process, after the learning is progressed, e.g., 1000 iterations, input-output pairs having had worse errors are extracted from the original training set and form a new temporary set. From the next iteration, the temporary set is applied instead of the original set. In this case, only pairs with worse errors are used for updating weights until the mean value of errors reduces to a level. After the learning conducted using the temporary set, the original set is applied again instead of the temporary set. It is expected by alternately giving the two kinds of sets that the convergence time can be efficiently reduced. The effectiveness is proved through simulation experiments using a kinematic model of an industrial robot with six-DOFs.

2. Industrial Robot Motoman with Six-DOFs

Figure 1 shows an industrial robot Motoman with six-DOFs. Six coordinate systems $\Sigma_k - x_k y_k z_k$ ($0 \leq k \leq 5$) are assigned at each joint. The position of the arm tip viewed in base frame $\Sigma_0 - x_0 y_0 z_0$ is defined with $\mathbf{x} = [x \ y \ z]^T$. Table 1 tabulates the Denavit-Hartenberg notation (DH) extracted from the robot⁽¹¹⁾. a is the link length which is the distance between two adjacent z -axes measured along x -axis, α is the link twist angle between two adjacent z -axes measured around x -axis, d is the link offset which is the distance

Fig. 1. Industrial robot Yaskawa Motoman.

Table 1. Denavit-Hartenberg notation for Motoman HP20⁽¹¹⁾.

k	a	α	d	θ
1	150	-90°	0	θ_1
2	760	0	0	θ_2
3	140	-90°	0	θ_3
4	0	90°	795	θ_4
5	0	90°	0	θ_5
6	0	0	-105	θ_6

between two adjacent x -axes measured along z -axis and θ is the joint angle between two adjacent x -axes measured around z -axis.

The homogeneous transform matrix ${}^{k-1}T_k$ using the four parameters is written by

$${}^{k-1}T_k = \text{Rot}(z, \theta) \text{Trans}(0, 0, d) \text{Trans}(a, 0, 0) \text{Rot}(x, \alpha)$$

$$= \begin{bmatrix} C_\theta & -S_\theta C_\alpha & S_\theta S_\alpha & a C_\theta \\ S_\theta & C_\theta C_\alpha & -C_\theta S_\alpha & a S_\theta \\ 0 & S_\alpha & C_\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where, e.g., the homogeneous transform $\text{Rot}(z, \theta)$ conducts the rotation of θ [rad] to right screw direction around z -axis; $\text{Trans}(0, 0, d)$ translates to z -direction with d [mm]; S_θ and C_θ denote $\sin\theta$ and $\cos\theta$, respectively. Hence, 0T_6 is obtained by

$${}^0T_6 = {}^0T_1 {}^1T_2 {}^2T_3 {}^3T_4 {}^4T_5 {}^5T_6 = \begin{bmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

3×3 rotational matrix $\mathbf{R}$ in 0T_6 has to be transformed into roll, pitch and yaw angles in kinematic simulation⁽¹²⁾. As is known, the inverse kinematics problem of a serial link structure becomes more complex with the increase of the DOFs.

3. Neural Network-Based Inverse Kinematics

3.1 Design of neural network

In this paper, the joint angle vector $\boldsymbol{\theta} = [\theta_1 \ \dots \ \theta_6] \Rightarrow$ the homogeneous transform matrix 0T_6 is called the forward kinematics and can be analytically calculated through ${}^0T_6 = f_{\text{kine}}(\boldsymbol{\theta})$ ⁽¹²⁾. Reversely, ${}^0T_6 \Rightarrow \boldsymbol{\theta}$ is to be the inverse kinematics. It is not easy but complex to obtain the analytical solutions of the inverse kinematics. In this section, it is tried that the mapping of inverse kinematics ${}^0T_6 \Rightarrow \boldsymbol{\theta}$ is acquired in the neural network shown in Fig. 2, in which $\mathbf{x}_d = [x_d \ y_d \ z_d \ \alpha_d \ \beta_d]^T$ is transformed from 0T_6 ; α_d and β_d are the roll and pitch angles, respectively. The yaw angle can be always fixed to 0 because of the axis symmetry of the sixth axis. The number of the hidden layers and that of neurons are not important for researching the proposed weights tuning method, so that they were tentatively set to 2 and 20, respectively. The neurons in two adjacent layers are

Fig. 2. Neural network with five inputs of $\mathbf{x}_d \in \mathbb{R}^5$ and six outputs of $\boldsymbol{\theta}_{nn} \in \mathbb{R}^6$. In simulations, the number of the hidden layers and that of neurons are tentatively set to 2 and 20, respectively.

Fig. 3. Desired points along a spiral path prepared for training set.

connected with weights randomly initialized within the range from -1 to 1 . In the learning process shown in this section, the back propagation algorithm is applied for training of the weights. After here, the neural network is called the NN.

The performance of the proposed efficient learning method is evaluated using the regularly prepared training set consisting of multiple pairs of $\mathbf{x}_{dj} = [x_{dj} \ y_{dj} \ z_{dj} \ \alpha_{dj} \ \beta_{dj}]^T$ and $\boldsymbol{\theta}_{dj}$ ($1 \leq j \leq 481$), i.e., the sampled number in the training set is 481. The desired points $[x_{dj} \ y_{dj} \ z_{dj}]^T$ included in $\mathbf{x}_{dj}$ generated along a spiral path can be viewed as shown in Fig. 3. Roll angle α_{dj} and pitch angle β_{dj} are obtained from normal vector at the respective points. The joint angle θ_{dj} in the training set was made by using an inverse kinematics function $\boldsymbol{\theta}_{dj} = \text{ikine}(\mathbf{x}_{dj})$ available on MATLAB⁽¹²⁾.

When the back propagation algorithm is used for adjusting weights in neural network, it is serious problem that much time is required for satisfactory convergence. Especially, undesirable stagnation of learning always depresses us. Here, an efficient training method is introduced by using the case shown in Fig. 3. In this method, when the

learning falls into the situation of stagnation, a stimulating extraction from original training set is applied and temporally creates another training set with worse evaluation.

3.2 Conventional and orthodox learning process

First of all, the conventional and orthodox learning process is described. Error E_i for criterion at the i -th learning process is defined by

$$E_i = \bar{e} = \frac{1}{481} \sum_{j=1}^{481} e_j \quad (3)$$

where e_j is the error in case that the j -th sample in training set is given to input and output layers, which is obtained by

$$e_j = \sqrt{\sum_{k=1}^6 (\theta_{djk} - \theta_{nnjk})^2} \quad (4)$$

where $\boldsymbol{\theta}_{dj} \in \mathbb{R}^6$ and $\boldsymbol{\theta}_{nnj} \in \mathbb{R}^6$ are the training vector for the output layer and the actual output from the NN, respectively. k is the number of each joint. j is the sampled number in training set. The learning, i.e., the updating of weights, is continued until the following condition is satisfied.

$$E_i < E_d \quad (5)$$

where E_d is the maximum allowable error. The stagnation of training process can be judged by

$$E_{i-1} - E_i < E_s \quad (6)$$

where E_s is the stagnation index. In the conventional learning method used in this section, all samples in the training set, i.e., 481 pairs, are sequentially given for updating 620 weights.

Figure 4 shows the learning result, in which the updating of weights by using a back propagation algorithm was

Fig. 4. Learning result using a conventional method.

conducted 481 times in one learning process on the horizontal axis. In other words, one learning process means 481 times iterative computations using 481 input and output pairs. Also note that one computation for updating the weights consists of a pair of a forward propagation to calculate the error and a back propagation to update 620 weights based on the error.

It was confirmed that totally the updating of weights was conducted $81,702 \times 481 = 39,298,662$ times. In this case, the desired maximum allowable error E_d was set to 0.03. The specification of CPU was Intel(R) Core(TM) i7 4790 3.6 GHz. In this research, the calculation time is not used for criterion because the value of the time depends on the performance of CPU built in PC.

3.3 Efficient learning process

On the other hand, in our proposed method, a new temporary training set consisting of pairs of input $\mathbf{x}_l$ and output θ_l , that have not been well trained, is extracted from the original training set. Three conditions of extraction are compared in the latter experiments, which are given by

$$e_l > \bar{e} \quad (7)$$

$$e_l > \bar{e} + e_{sd} \quad (8)$$

$$e_l > \bar{e} + 2e_{sd} \quad (9)$$

Where

$$e_{sd} = \sqrt{\frac{1}{481} \sum_{j=1}^{481} (\bar{e} - e_j)^2} \quad (10)$$

where e_{sd} is the standard deviation of e_j ($1 \leq j \leq 481$). At the present time, some standards and criterions to systematically decide the condition have not been considered.

The pairs of $\mathbf{x}_l$ and θ_l satisfying Eq. (7), (8) or (9) are extracted and form a new training set with less number of pairs than the original training set. When this new training set is applied instead of the original set, the error $\tilde{E}_i$ at the i -th learning process is calculated by

$$\tilde{E}_i = \frac{1}{m} \sum_{l=1}^m e_l \quad (11)$$

where m is the number of the pairs in the training set extracted from the original set based on Eq. (7), (8) or (9). E_i and $\tilde{E}_i$ are alternately used for the criterion in the back propagation algorithm in our proposed method. Figure 5 illustrates the conceptual comparison between the

Fig. 5. Concept of the proposed learning method, where original training set and extracted one are alternately applied.

conventional and proposed learning methods. In the proposed method, the original training set and the extracted one are alternately applied for the iterative learning 1000 and 4000 times, respectively. Note that the times of iterative learning for original and extracted sets can be arbitrarily changed according to actual problem settings. The vertical axis means the number of the pairs in the training set.

Finally, the proposed method was applied to the same problem explained in section 3.2 to evaluate the effectiveness. As an example, Table 2 shows three results of the total times of weights updating in case that the numbers of iterative learning of original and extracted sets were set to 1,000 and 4,000, respectively. Figure 6 shows the learning result until E_i reached to 0.03 in the case of Eq. (7) in

Table 2. Three results of the total times of weights updating, in which numbers of iterative learning of original and extracted sets were 1,000 and 4,000, respectively.

Extraction	Times of weights updating
Eq. (7)	27,050,036
Eq. (8)	29,130,464
Eq. (9)	31,901,186

Fig. 6. Learning result in case of using Eq. (7) in Table 2.

Table 2, in which sequential 1000 times of E_i and 4,000 times of $\widetilde{E}_i$ are used alternately and repeatedly. It is observed that total times of weights updating were improved by about 31 %, i.e., from 39,298,662 to 27,050,036.

4. Conclusions

This paper has introduced a neural network with an efficient weights tuning ability to effectively learn the inverse kinematics of an industrial robot with six-DOFs. In the proposed learning process, after the learning is progressed, e.g., 1000 iterations, input-output pairs with worse errors are extracted from the original training set and form a new temporary set. From the next iteration, the temporary set is applied instead of the original set. After the learning conducted using the temporary set, the original set is applied again instead of the temporary set. The effectiveness was proved through simulation experiments using a kinematic model of an industrial robot MOTOMAN HP20, i.e., the convergence time could be efficiently reduced by alternately applying the two kinds of sets.

References

- (1) G. Hoang, J.H. Min, G.M. Lee, B.H. Jun, H.K. Kim and S.B. Kim : “Omni-directional walking control for a six-legged robot using differential kinematics algorithm,” Procs. of 14th International Conference on Control, Automation and Systems (ICCAS 2014), pp. 1163–1168, 2014.
- (2) S. Tejomurtula and S. Kak : “Inverse kinematics in robotics using neural networks,” Information Sciences, Vol. 116, pp. 147–164, 1999.
- (3) A.V. Duka : “Neural network based inverse kinematics solution for trajectory tracking of a robotic arm,” Procedia Technology, Vol. 12, pp. 20–27, 2014.
- (4) Y. Maeda, T. Fujiwara and H. Ito : “Robot control using high dimensional neural networks,” Procs. of SICE Annual Conference 2014, pp. 738–743, 2014.
- (5) R. Koker, C. Oz, T. Cakar and H. Ekiz : “A study of neural network based inverse kinematics solution for a three-joint robot,” Robotics and Autonomous Systems, Vol. 49, Nos. 3/4, pp. 227–234, 2004.
- (6) R. Koker : “A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization,” Information Sciences, Vol. 222, pp. 528–543, 2013.
- (7) Ali T. Hasan, Hayder M.A.A. Al-Assadi and Ahmad Azlan Mat Isa : “Neural networks’ based inverse kinematics solution for serial robot manipulators passing through singularities,” Artificial Neural Networks – Industrial and Control Engineering Applications, pp. 459–478, InTech, 2011.
- (8) H.R. Kanan and M.Y. A. Khanian : “Reduction of neural network training time using an adaptive fuzzy approach in real time applications,” International Journal of Information and Electronics Engineering, Vol. 2, No. 3, pp. 470–474, 2012.
- (9) F. Nagata, S. Inoue, S. Fujii, A. Otsuka and K. Watanabe : “Learning of inverse kinematics using a neural network with efficient weights tuning ability,” Procs. of SICE Annual Conference 2015, pp. 1271–1275, 2015.
- (10) F. Nagata, S. Inoue, S. Fujii, A. Otsuka, K. Watanabe and M.K. Habib : “Learning of inverse kinematics using neural networks and its application to kinematic control of position-based servo motor,” Procs. of The 2015 World Congress on Advances in Aeronautics, Nano, Bio, Robotics, and Energy (ANBRE15), pp. 1–13, 2015.
- (11) Y. Gan, X. Dai and D. Li : “Off-line programming techniques for multi-robot Cooperation System,” International Journal of Advanced Robotic Systems, Vol. 10, pp. 1–17, 2013.
- (12) P. Corke : “A robotics toolbox for MATLAB,” IEEE Robotics & Automation Magazine, Vol. 3, No. 1, pp. 24–32, 1996.