Neural Network-Based Inverse Kinematics for Motoman HS20 and Its Efficient Learning Method

FUSAOMI NAGATA†† Member, KEIGO WATANABE‡ Non-member

(Received May 5, 2016, revised July 29, 2016)

Abstract: Generally, desired training set is used to make a neural network learn nonlinear relations properly. The training set consists of multiple pairs of an input vector and 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. Weights between neurons are updated based on the errors using a back propagation algorithm in backward calculation. The time required for the learning process of a neural network 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 [1]. There were reported the results about neural networks’ 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].

Other than the inverse kinematics problems, neural networks were also applied to trajectory tracking tasks of industrial robots. Jiang and Ishida proposed a dynamic trajectory tracking control method for an industrial robot using a linear feedback controller and a neural network controller. A manifold was designed to evaluate the trajectory tracking performance. The linear feedback controller was designed with respect to the manifold. The three layered feedforward neural network controller was parallelly added to the system to make up the proposed control system [8]. Raafat et al. proposed a neural-network based robust adaptive controller for a three-axis SCARA type industrial robot. Effective robustness with respect to uncertain dynamics and nonlinearities was evaluated through trajectory tracking experiments [9]. Then, Dinary et al. developed a neural network-based identification and control scheme for trajectory tracking of a robot arm. The scheme was composed of two neural networks and a linear controller without using the model dynamics. The first neural network worked as an inverse dynamic identification of the robot and the second
one was applied to a controller in parallel with the linear PD controller [10]. Up to here, several researches using neural networks for robotic control have been surveyed, in which back propagation methods were used for training neural networks. It seems that some efficient method to accelerate the learning speed of neural network has been required.

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 between neurons 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 [11]. 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. Note that one iteration of learning uses all pairs in the training 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. Inverse Kinematics Problem

In this section, the difficulty of inverse kinematics problem is explained using two examples. One is a serial link with four-DOFs for multi-legged mobile robots [12] [13]. The other is a typical industrial robot with six-DOFs. Figure 1 shows an example of a leg with four-DOFs, which is used for graduation study of undergraduate students [14]. Figure 2 shows the kinematic model of the leg. Five coordinate systems $\Sigma_0$, $\Sigma_1$, $\Sigma_2$, $\Sigma_3$, $\Sigma_4$ are assigned at each joint and the tip of the leg. The position of the leg tip in base frame $\Sigma_0$ is defined with $x = [x \, y \, z]^T$. Table 1 tabulates the Denavit-Hartenberg (DH) notation extracted from the leg with four joints. $a$ is the link length which is the distance between two adjacent $z$-axes measured along $x$-axis, $\alpha$ is the link twist angle between two adjacent $z$-axes measured around $x$-axis, $d$ is the link offset which is the distance between two adjacent $x$-axes measured along $z$-axis and $\theta$ is the joint angle between two adjacent $x$-axes measured around $z$-axis. Actually, Fig. 2 shows the initial pose of the leg with the angles of $\theta = [\theta_1 \, \theta_2 \, \theta_3 \, \theta_4]^T = [0 \, 0 \, 0 \, 0]^T$. The homogeneous transform matrix $k^{-1}T_k$ using the four parameters is written by

$$
k^{-1}T_k = \begin{pmatrix} c\theta & -s\theta & a\theta & 0 \\
s\theta & c\theta & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
$$

where, e.g., the homogeneous transform $R(z, \theta)$ conducts the rotation of $\theta$ [rad] to right screw direction around $z$-axis; $R(x, 1, 0, d)$ translates to $x$-direction with $d$ [mm]; $S\theta$ and $C\theta$ denote $\sin \theta$ and $\cos \theta$, respectively. Hence, $0T_4$ is obtained by

$$
0T_4 = 0T_1 \cdot T_2 \cdot T_3 \cdot T_4
$$

where

$$
R_{11} = C_a(C_{\theta_1}C_{\theta_2}C_{\theta_3} - C_{\theta_1}S_{\theta_2}S_{\theta_3}) \\
+ S_a(C_{\theta_1}S_{\theta_2}C_{\theta_3} - C_{\theta_1}S_{\theta_2}S_{\theta_3}) \\
R_{12} = S_a(C_{\theta_1}S_{\theta_2}C_{\theta_3} - C_{\theta_1}C_{\theta_2}C_{\theta_3}) \\
- C_a(C_{\theta_1}S_{\theta_2}C_{\theta_3} + C_{\theta_1}C_{\theta_2}S_{\theta_3})
$$
Table 1: Denavit-Hartenberg notation designed for the leg with four joints.

<table>
<thead>
<tr>
<th>k</th>
<th>a</th>
<th>α</th>
<th>d</th>
<th>θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(l_1)</td>
<td>(-90^\circ)</td>
<td>0</td>
<td>(\theta_1)</td>
</tr>
<tr>
<td>2</td>
<td>(l_2)</td>
<td>0</td>
<td>0</td>
<td>(\theta_2)</td>
</tr>
<tr>
<td>3</td>
<td>(l_3)</td>
<td>0</td>
<td>0</td>
<td>(\theta_3)</td>
</tr>
<tr>
<td>4</td>
<td>(l_4)</td>
<td>0</td>
<td>0</td>
<td>(\theta_4)</td>
</tr>
</tbody>
</table>

\[
R_{21} = C_{a1}(S_{\alpha1}C_{\theta1} - S_{\alpha1}S_{\theta1}) - S_{a1}(S_{\alpha1}S_{\theta1} + C_{\alpha1}C_{\theta1})
\]

\[
R_{22} = S_{a1}(S_{\alpha1}S_{\theta1} + C_{\alpha1}S_{\theta1}) - C_{a1}C_{\theta1}
\]

\[
R_{31} = S_{a1}(S_{\alpha1}C_{\theta3} - C_{\alpha1}S_{\theta3}) - C_{a1}(S_{\alpha1}S_{\theta3} + C_{\alpha1}C_{\theta3})
\]

\[
R_{32} = S_{a1}(S_{\alpha1}S_{\theta3} + C_{\alpha1}C_{\theta3}) + C_{a1}S_{\theta3} - C_{\alpha1}S_{\theta3}
\]

\[
x = l_1C_{\theta1} + l_2C_{\theta1}C_{\theta2} + l_3C_{\theta1}C_{\theta2}C_{\theta3} - l_3C_{\theta1}S_{\theta2}S_{\theta3} + l_4C_{\theta4}(C_{\alpha1}C_{\theta2}C_{\theta3} - C_{\alpha1}S_{\theta2}S_{\theta3}) - l_4S_{\alpha4}(S_{\alpha1}S_{\theta2}C_{\theta3} + C_{\alpha1}C_{\theta2}S_{\theta3})
\]

\[
y = l_1S_{\theta1} + l_2S_{\theta1}C_{\theta2} + l_3S_{\theta1}C_{\theta2}C_{\theta3} - l_3S_{\theta1}S_{\theta2}S_{\theta3} + l_4C_{\theta4}(S_{\alpha1}C_{\theta2}C_{\theta3} - S_{\alpha1}S_{\theta2}S_{\theta3}) - l_4S_{\alpha4}(S_{\alpha1}S_{\theta2}C_{\theta3} + S_{\alpha1}C_{\theta2}S_{\theta3})
\]

\[
z = -l_2S_{\alpha2} - l_3S_{\alpha2}C_{\theta3} - l_3C_{\theta1}S_{\alpha2} + l_4C_{\theta4}(S_{\alpha1}C_{\theta2}S_{\theta3} + C_{\alpha1}S_{\theta2}S_{\theta3}) + l_4S_{\alpha4}(S_{\alpha1}S_{\theta2}S_{\theta3} - C_{\alpha1}C_{\theta2}S_{\theta3})
\]

As can be imagined from above equations, the inverse kinematics problem to analytically obtain \(\theta = [\theta_1 \theta_2 \theta_3]^T\) from \(x = [x y z]^T\) is not easy but complex even though the number of joints is four.

### 2.2 Industrial robot Motoman with six-DOFs

Figure 3 shows an industrial robot Motoman with six-DOFs. Seven coordinate systems \(\Sigma_k - x_{k}y_{k}z_{k} (0 \leq k \leq 6)\) are assigned at the robot base origin and six joints. The position of the arm tip viewed in base frame \(\Sigma_0 - x_{0}y_{0}z_{0}\) is defined with \(x = [x y z]^T\). Table 2 tabulates the Denavit-Hartenberg notation extracted from the industrial robot Motoman HS20 [15]. Similarly in the previous subsection, \(\theta_{\theta_6}\) is obtained by

\[
0T_6 = \begin{bmatrix} 1 & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 \end{bmatrix}
\begin{bmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]

where \(3 \times 3\) rotational matrix \(R\) in \(0T_6\) has to be transformed into roll, pitch and yaw angles in kinematic simulation [16]. As is known, the inverse kinematics problem of a serial link structure becomes more complex with the increase of the DOFs.

In the next section, neural network-based inverse kinematics with an efficient learning capability is discussed using the kinematic model of Motoman HS20.

Table 2: Denavit-Hartenberg notation of Motoman HP20 with six joints [15].

<table>
<thead>
<tr>
<th>k</th>
<th>a</th>
<th>α</th>
<th>d</th>
<th>θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
<td>(-90^\circ)</td>
<td>0</td>
<td>(\theta_1)</td>
</tr>
<tr>
<td>2</td>
<td>760</td>
<td>0</td>
<td>0</td>
<td>(\theta_2)</td>
</tr>
<tr>
<td>3</td>
<td>140</td>
<td>(-90^\circ)</td>
<td>0</td>
<td>(\theta_3)</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>90°</td>
<td>795</td>
<td>(\theta_4)</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>90°</td>
<td>0</td>
<td>(\theta_5)</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>(-105)</td>
<td>(\theta_6)</td>
</tr>
</tbody>
</table>

### 3. Neural Networked-Based Inverse Kinematics

#### 3.1 Design of neural network

In this paper, the joint angle vector \(\theta = [\theta_1 \cdots \theta_6]\) \(\Rightarrow\) the homogeneous transform matrix \(0T_6\) is called the forward kinematics and can be analytically calculated through \(0T_6 = f_{\text{kinet}}(\theta)\) introduced in robotics tool box [16]. Reversely, \(0T_6 \Rightarrow \theta\) is to be the inverse kinematics. As is discussed in the previous section, 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 \(0T_6 \Rightarrow \theta\) is acquired in the neural network shown in Fig. 4, in which the \(x_d = [x_d y_d z_d \alpha_d \beta_d]^T\) is transformed from \(0T_6\); \(\alpha_d\) and \(\beta_d\) 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 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 $x_d = [x_d, y_d, z_d, \alpha_d, \beta_d]^T$ and $\theta_d$ ($1 \leq j \leq 481$), i.e., the sampled number in the training set is 481. The desired points $[x_d, y_d, z_d]^T$ included in $x_d$, generated along a spiral path can be viewed as shown in Fig. 5. Roll angle $\alpha_d$ and pitch angle $\beta_d$ are obtained from normal vector at the respective points. The joint angle vector $\theta_d$ in the training set was made by using an inverse kinematics function $\theta_d = \text{kine}(x_d)$ available on MATLAB [16].

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. 5. 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.

![Figure 4: Neural network with five inputs of $x_d \in \mathbb{R}^3$ and six outputs of $\theta_{on} \in \mathbb{R}^6$. In simulations, the number of the hidden layers and that of neurons are tentatively set to 2 and 20, respectively.](image)

![Figure 5: Desired points along a spiral path prepared for training set.](image)

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

$$E_i = \bar{e} = \frac{1}{481} \sum_{j=1}^{481} e_j$$

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

$$e_j = \sqrt{\sum_{k=1}^{6} (\theta_{dk} - \theta_{mn})^2}$$

where $\theta_{dj} \in \mathbb{R}^6$ and $\theta_{mn} \in \mathbb{R}^6$ 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.

$$E_i < E_d$$

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

$$E_{i-1} - E_i < E_s$$

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 6 shows the learning result, in which the updating of weights by using a back propagation algorithm was 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 81,702×481=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.

![Figure 6: Learning result using a conventional method.](image)
3.3 Efficient learning process

On the other hand, in our proposed method, a new temporary training set consisting of pairs of input \( x_l \) and output \( \theta_l \), 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

\[
\begin{align*}
    e_l &> \bar{e} \\
    e_l &> \bar{e} + \epsilon_{\text{sd}} \\
    e_l &> \bar{e} + 2\epsilon_{\text{sd}}
\end{align*}
\]

(17)\hspace{1cm}(18)\hspace{1cm}(19)

where

\[
\epsilon_{\text{sd}} = \frac{1}{481} \sum_{j=1}^{481} (\bar{e} - e_j)^2
\]

(20)

where \( \epsilon_{\text{sd}} \) 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 \( x_l \) and \( \theta_l \) satisfying Eq. (17), (18) or (19) 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, i.e., iteration, is calculated by

\[
\tilde{E}_i = \frac{1}{m} \sum_{l=1}^{m} e_l
\]

(21)

where \( m \) is the number of the pairs in the training set extracted from the original set based on Eq. (17), (18) or (19). \( E_i \) and \( \tilde{E}_i \) are alternately used for the criterion in the back propagation algorithm in our proposed method. Figure 7 illustrates the conceptual comparison between the 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 3 shows three results of the total times of weights updating in case that the numbers of iterative learning of original and extracted sets were 1,000 and 4,000, respectively. Figure 8 shows the learning result until \( E_i \) reached to 0.03 in the case of Eq. (17) in Table 3, in which sequential 1000 times of \( E_i \) and 4,000 times of \( \tilde{E}_i \) are used alternately and repeatedly. It is observed that the 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. In future work, better conditions for extraction from original training set will be investigated. Also, how to systematically determine the times of iterative learning for original training set and extracted one will be considered in order to further reduce the load of weights updating.

As can be seen, this paper has focused on how to accelerate the learning speed. As for the generalization abil-
ity of the trained neural network, it was already reported that some passable generalization was performed for inverse kinematics of a serial link with four DOFs [12][13]. The deeper evaluation on contribution of the proposed learning method to the generalization ability will be planned.

References


