Inverse Kinematics Learning by Modular Architecture Neural Networks

Eimei OYAMA

Robotics Department
Mechanical Engineering Laboratory
Namiki 1-2, Tsukuba Science City
Ibaraki 305-8564 Japan
eimei@mel.go.jp

Susumu TACHI

Faculty of Engineering
The University of Tokyo
Hongo 7-3-1, Bunkyo-ku
Tokyo 113-8656 Japan

Abstract

Inverse kinematics computation using an artificial neural network that learns the inverse kinematics of a robot arm has been employed by many researchers. However, conventional learning methodologies do not pay enough attention to the discontinuity of the inverse kinematics system of typical robot arms with joint limits. The inverse kinematics system of the robot arms, including a human arm with a wrist joint, is a multivalued and discontinuous function. Since it is difficult for a well-known multi-layer neural network to approximate such a function, a correct inverse kinematics model for the end-effector's overall position and orientation cannot be obtained by the conventional methods. In order to overcome the drawbacks of the inverse kinematics solver consisting of a single neural network, we propose a novel modular neural network architecture for the inverse kinematics model learning.

1 Introduction

The task of calculating all of the joint angles that would result in a specific position/orientation of an end-effector of a robot arm is called the inverse kinematics problem. An inverse kinematics solver using an artificial neural network that learns the inverse kinematics system of a robot arm has been used in many researches; however, many researchers do not pay enough attention to the discontinuity of the inverse kinematics function of typical robot arms with joint limits. The inverse kinematics function of the robot arms, including a human arm with a wrist joint, is a multivalued and discontinuous function. It is difficult for a well-known multi-layer neural network to approximate such a function. A correct inverse kinematics solution for the end-effector's overall position and orientation cannot be obtained by the inverse kinematics model consisting of a single neural network. Therefore we propose a novel modular neural network architecture for the inverse kinematics model learning and the online incremental learning method for the architecture.

Jacobs et al. proposed a modular network architecture that consists of a number of expert networks and a gating network[1] [2]. The gating network synthesizes the outputs of the expert networks appropriately. and calculates one output of the modular networks. Gomi and Kawato applied the modular architecture neural networks to the object recognition for manipulating a variety of objects and to inverse dynamics learning[3]. Kawato et al. developed the modular architecture neural networks and recently proposed Multiple Pairs of Forward and Inverse Models as a computational model of the cerebellum[4]. However, the input-output relation of their networks is continuous and the learning method of them is not sufficient for the nonlinearity of the kinematics system of the robot arm. Their architecture is not suitable for the inverse kinematics model learning. The novel architecture is necessary.

In order to evaluate the proposed architecture, numerical experiments of the inverse kinematics model learning were performed.

2 Background

Let $\boldsymbol{\theta}$ be the $m \times 1$ joint angle vector and $\boldsymbol{x}$ be the $n \times 1$ position/orientation vector of a robot arm. The relationship between $\boldsymbol{\theta}$ and $\boldsymbol{x}$ is described by $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta})$. $\boldsymbol{f}$ is a C^1 class function. Let $\boldsymbol{J}(\boldsymbol{\theta})$ be the Jacobian of the robot arm, defined as $\boldsymbol{J}(\boldsymbol{\theta}) = \partial \boldsymbol{f}(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$.When a desired hand position/orientation vector $\boldsymbol{x}_d$ is given, an inverse kinematics problem that calculates the joint angle vector $\boldsymbol{\theta}_d$ satisfying the equation $\boldsymbol{x}_d = \boldsymbol{f}(\boldsymbol{\theta}_d)$ is considered. In this paper, a function $\boldsymbol{g}(\boldsymbol{x})$ that satisfies $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{g}(\boldsymbol{x}))$ is called an inverse kinematics system of $\boldsymbol{f}(\boldsymbol{\theta})$. The acquired model of the inverse kinematics system $\boldsymbol{g}(\boldsymbol{x})$ is called an inverse kinematics model. Let $\boldsymbol{\Phi}_{im}(\boldsymbol{x})$ be the output of the inverse kinematics model.

Many researchers have employed the method that uses an acquired inverse kinematics model consisting of a single neural network in order to control a robot arm; however, the neural network architecture has a number of drawbacks. Let's consider the inverse kinematics of the 2-DOF (degrees of freedom) arm moving in a plane. The relationship between the joint angle vector $(\theta_1, \theta_2)^T$ and the end-effector position vector $(x, y)^T$ is as follows:

$$\begin{aligned} x &= x_0 + L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) \quad (1) \\ y &= y_0 + L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) \end{aligned}$$

We assume that L_1 is $0.30m$, L_2 is $0.25m$, the range of θ_1 is $[30°, 150°]$, and the range of θ_2 is $[-150°, 150°]$. Two inverse kinematics solutions of Equation (1) must be switched according to $(x, y)^T$. The inverse kinematics function of the robot arm is a multivalued and discontinuous function.

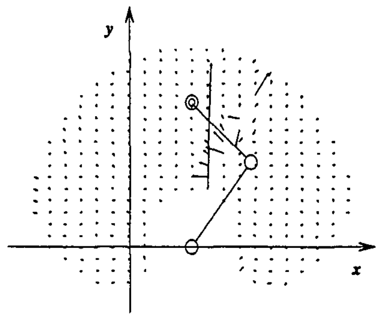


Figure 1 Position Error Vector of Inverse Kinematics Model Which Consists of a Single Neural Network

Figure 1 shows the position error vector of an inverse kinematics model learned by Forward and Inverse Modeling[5]. An arrow in the figure shows the end-effector position error by the inverse kinematics model $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{f}(\boldsymbol{\Phi}_{im}(\boldsymbol{x}))$ at each desired end-effector position $\boldsymbol{x}$. In most regions, the inverse kinematics model is precise. However, there are some regions where it is far from precise, which is caused by the discontinuity of the inverse kinematics function.

3 Modular architecture neural networks for inverse kinematics model learning

The inverse kinematics function can be constructed by the appropriate mixture of continuous functions[6]. We propose a novel modular neural network architecture that can learn a discontinuous inverse kinematics function by the appropriate switching of multiple neural networks.

3.1 Configuration of proposed inverse kinematics solver

In order to learn a discontinuous inverse kinematics function, selecting one expert has better performance than mixing all experts. Figure 2 shows the configuration of the inverse kinematics solver with the modular architecture networks for inverse kinematics model learning. Each expert network in Figure 2 approximates the continuous region of the inverse kinematics function. The expert selector selects one appropriate expert according to the desired position/orientation of the end-effector of the arm, as described in Section 3.2. The extended feedback controller calculates the inverse kinematics solution based on the output of the selected expert. When no precise solution is obtained, the controller performs a type of global search, as shown in Section 3.3. The expert generator generates a new expert network based on the inverse kinematics solution.

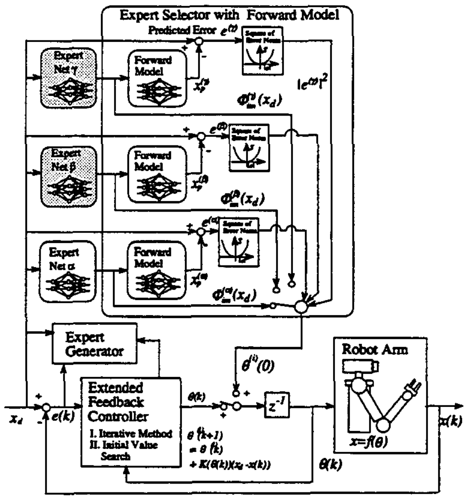


Figure 2 Inverse Kinematics Solver with Modular Architecture Networks

3.2 Configuration of expert and selection by the forward model

In order to cover the overall work space, each expert has its representative posture. The representative posture is the inverse kinematics solution obtained in the global searches by the extended feedback controller when the expert is generated. Let $\theta_r^{(i)}$ be the representative posture of the *i-th* expert and $x_r^{(i)}$ be the end-effector position/orientation that corresponds to $\theta_r^{(i)}$. Let $\Phi_{im}^{(i)}(x)$ be the output of *i-th* expert when the input of the expert is x. Each expert is trained to satisfy the following equation:

$$x_r^{(i)} = f(\Phi_{im}^{(i)}(x_r^{(i)})) \tag{2}$$

By changing the bias parameters of the output layer of the neural network, the above equation can easily be satisfied. Each expert approximates the continuous region of the inverse kinematics function in which the reaching motion can move the end-effector smoothly from its representative posture.

The predicted position/orientation error is used as the performance index of the expert. When the desired end-effector position x_d is given, the *i-th* expert calculates the output $\Phi_{im}^{(i)}(x_d)$. The expert selector selects an expert with the smallest predicted error among all experts. Let $\Phi_{fm}(\theta)$ be the output of the forward kinematics model and $\Phi_{im}^{(i)}(x_d)$ be the output of the *i-th* expert. The predicted error of the *i-th* expert p_i is calculated as $p_i = |x_d - \Phi_{fm}(\Phi_{im}^{(i)}(x_d))|$.

Let $\Phi'_{fm}(\theta)$ be the desired output for $\Phi_{fm}(\theta)$. The learning of the forward kinematics model is conducted as $\Phi'_{fm}(\theta) = x = f(\theta)$.

3.3 Extended feedback controller and expert learning

The conventional online inverse model learning methods, such as Forward and Inverse Modeling[5] and Feedback Error Learning proposed by Kawato[7], are based on the local information of the forward system near the output of the inverse model. The desired output signal provided by these methods is not always in the direction that finally reaches the correct solution of the inverse problem. An extended feedback controller avoids that drawback by employing a kind of global search technique based on the multiple starts of the iterative procedure[8][9]. When the iterative procedure from the output of the selected expert cannot reach a correct solution, the extended feedback controller repeats the iterative procedure from a number of initial values until a correct solution is obtained.

The proposed inverse kinematics solver solves an inverse kinematics problem according to the following procedural steps:

(1) When x_d is given, the expert selector selects the expert with the minimum predicted error among all the experts. The extended feedback controller moves the arm to the posture that corresponds to the output of the expert and then improves the end-effector position/orientation by using the output error feedback, as described in Section 3.4.

(2) When no precise inverse kinematics solution is obtained in step (1), the other expert is selected in increasing order of the predicted error and the iterative improvement procedure by the output error feedback is conducted.

(3) When no solution is obtained in steps (1) and (2), an expert is randomly selected and a reaching motion from the representative posture of the selected expert is conducted. The above procedure is repeated until the reaching motion is successfully conducted or all the experts are tested.

(4) If a precise solution is obtained in each iterative computation, the solution is used as the desired output signal for the expert, as shown in Section 3.4.

(5) When no solution is obtained in the above procedures, the controller starts a type of global search. The controller repeats the initial joint angle vector generation by using a uniform random number generator and the reaching motion from the generated posture, until a precise solution is obtained. When a precise solution is obtained, a new expert is generated and the solution is used as the representative posture θ_r of the expert.

3.4 Iterative procedure

An illustration of the iterative procedure follows. Let $\theta^{(0)}$ be the initial posture of the iterative procedure, which is the output of the selected expert $\Phi^{(i)}(x_d)$, the representative posture of the selected expert $\theta_r^{(i)}$, or the randomly generated posture.

Let $J^+(\theta)$ be the pseudo-inverse matrix of $J(\theta)$, which is calculated as $J^+(\theta) = J^T(\theta)(J(\theta)J^T(\theta))^{-1}$. When $|e^{(0)}| = |x_d - f(\theta^{(0)})|$ is smaller than an appropriate threshold r_{st}, the extended feedback controller conducts the iterative improvement by using the output error feedback as follows:

$$\theta^{(j+1)} = \theta^{(j)} + J^{+}(\theta^{(j)})(x_d - f(\theta^{(j)})) \quad (3)$$

where j is the iteration number. The Jacobian of the arm $J(\theta)$ can be calculated by the observation of the movement of the robot arm and the numerical differentiation technique. The output error feedback through the forward kinematics model[5] or the learning feedback controller[10][11] can also be utilized for the above iterative calculation. Let $\Phi'^{(i)}_{im}(x_d)$ be the desired output signal for the selected expert. Let r_e be the desired maximum position/orientation error norm of the arm. When a precise inverse kinematics solution $\theta^{(J)}$ which satisfies $|x_d - f(\theta^{(J)})| < r_e$ is obtained, the solution is used as the desired output for the selected expert as $\Phi'^{(i)}_{im}(x_d) = \theta^{(J)}$.

When $|e^{(0)}|$ is larger than r_{st}, the extended feedback controller conducts a reaching motion from the posture $\theta^{(0)} = \Phi^{(i)}_{im}(x_d)$ to x_d. The reaching control is conducted as the tracking control to the desired trajectory of the end-effector $x_d(k)(k = 0, 1, \ldots)$. The desired trajectory $x_d(k)$ is a straight line from $x_s = f(\theta^{(0)})$ to x_d, which is generated in order to satisfy $|\Delta x_d(k)| < r_{st}$. Let $\theta^{(J)}(k)$ be a precise inverse kinematics solution at step k. $\theta^{(J)}(0) = \theta^{(0)}$ is satisfied. When a precise inverse kinematics solution $\theta^{(J)}(k-1)$ that corresponds to $x_d(k-1)$ is obtained, the initial value for step $k+1$ $\theta^{(0)}(k)$ is calculated as follows:

$$\begin{aligned} \theta^{(0)}(k) &= \theta^{(J)}(k-1) \\ &+ J^{+}(\theta^{(J)}(k-1))(x_d(k) - f(\theta^{(J)}(k-1))) \end{aligned} \quad (4)$$

When $|x_d(k) - f(\theta^{(0)}(k))|$ is not small, the iterative improvement procedure is continued as follows:

$$\begin{aligned} \theta^{(j+1)}(k) &= \theta^{j}(k) \\ &+ J^{+}(\theta^{(j)}(k))(x_d(k) - f(\theta^{(j)}(k))) \end{aligned} \quad (5)$$

The iterative improvement is repeated until the output error norm $|e^{(j)}(k)| = |x_d(k) - f(\theta^{(j)}(k))|$ is lower than the desired maximum error norm r_e. If a precise solution is obtained, the solution can be used for the desired output for the inverse kinematics model as $\Phi'_{im}(x_d(k)) = \theta^{(J)}(k)$.

When the controller cannot find a precise solution because of the singularity of Jacobian $J(\theta^{(j)}(k))$ or the joint limits, the reaching motion is regarded as a failure.

3.5 Discontinuity check

The initial status of the experts sometimes causes a situation where one expert approximates multiple regions of the inverse kinematics function. In the case that one region contacts the other region in the workspace coordinates, the result is that the expert tries to approximate a discontinuous function. This situation should be avoided. When the matrix norm of $\partial\Phi^{(i)}_{im}(x)/\partial x|_{x=x_d}$ is larger than an appropriate threshold r_{jix}, a new expert with a new representative posture θ_r corresponding to x_d is generated to approximate the region near to x_d. $\partial\Phi^{(i)}_{im}(x)/\partial x|_{x=x_d}$ is approximately calculated by using numerical differentiation and checked when the *i-th* expert is selected according to step (1) in Section 3.3.

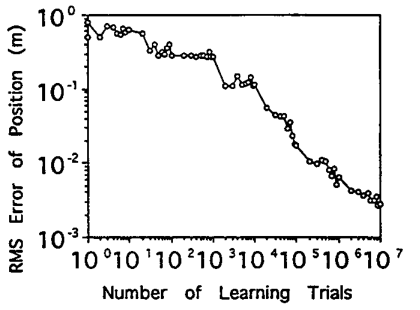
4 Simulations

4.1 Inverse kinematics learning of a 2-DOF arm

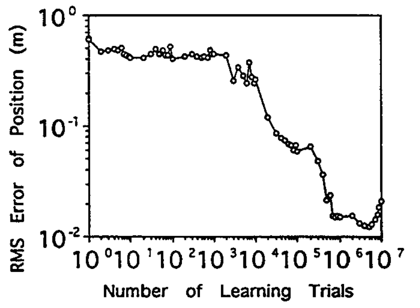
We performed simulations of the inverse kinematics model learning for a 2-DOF arm moving in the 2-DOF plane as described in Section 2.

In the simulations, joint angle vectors were generated by using a uniform random number generator, and the end-effector positions that correspond to the generated vectors were used as the desired end-effector positions. In order to evaluate the performance of the solver, 1,000 desired end-effector positions were generated for the estimation of the root mean square (RMS) error of the end-effector position $e = x_d - f(\Phi_{im}(x_d))$. A 4-layered neural network was used for the simulations. The 1st layer, i.e., the input layer, and the 4th layer, i.e., the output layer, consisted of linear neurons. The 2nd and the 3rd layers had 15 neurons each. The back-propagation method was utilized for the learning. Before the learning, the inverse kinematics solver had one expert the representative posture of which was $(0.0, 0.0)^T$. r_e was $0.001m$, r_{st} was $0.05m$, and r_{jix} was 10^2.

Figure 3 shows the progress of the inverse kinematics model learning. Figure 3(a) shows the RMS error of the end-effector position $\sqrt{E[e^Te]}$ by using the inverse kinematics model. It can be seen that the RMS error decreases and the precision of the inverse model becomes higher as the number of trials increases. Figure 3(b) shows the RMS error of the forward kinematics model $\sqrt{E[e^T_{fm}e_{fm}]}$. e_{fm} is the error vector of the forward kinematics model defined as $e_{fm} = \boldsymbol{f}(\boldsymbol{\theta}) - \boldsymbol{\Phi}_{fm}(\boldsymbol{\theta})$. The number of the experts which constructs the inverse kinematics model became 3 after 6 times learning trials.



(a)RMS position error



(b) RMS position error of forward model
Figure 3 Performance Change of Inverse Model

Figure 4 shows the position error vector of one example of an inverse kinematics model acquired by the proposed method. We were able to obtain a precise inverse kinematics model of the overall points that the robot arm can reach. Figure 5 illustrates how the expert is selected. Because the representative posture of the first expert was $(0.0, 0.0)^T$ and the Jacobian of the posture is singular, the expert is rarely used. The second expert and the third expert covers almost all region. Figure 5(b) shows the region where the predicted output error of the second expert is lower than $0.01m$. The graphics of the robot arm in Figure 5(b) shows the representative posture of the second expert. Figure 5(c) shows the region where the predicted output error of the third expert is lower than $0.01m$.

The RMS error became $1.50 \times 10^{-3}m$ after 5×10^7 learning trials. The simulations of the inverse kinematics model learning, consisting of a single neural networks, by using the Forward and Inverse Modeling, were also performed. The learning was performed from 10 different initial states of the neural network. The minimum RMS error after 10^9 learning trials was $1.20 \times 10^{-2}m$. Forward and Inverse Modeling with a single neural network cannot obtain a precise inverse kinematics model of the arm (as shown in Figure 1), whereas the proposed method can obtain a precise model.

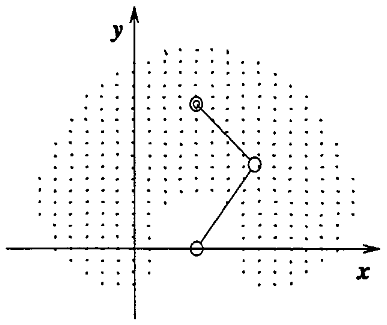
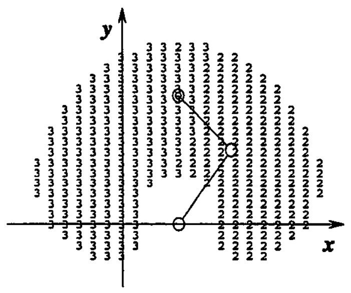
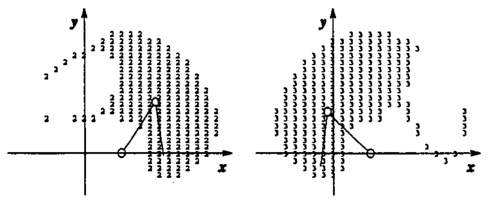


Figure 4 Error Vectors of Learned Inverse Kinematics Model Consisting Modular Architecture Networks



(a)Selected experts



(b) Second expert (c) Third expert
Figure 5 Configuration of Learned Inverse Model

4.2 Inverse kinematics learning of a 7-DOF arm

We considered the inverse kinematics model learning of a 7-DOF arm (Mitsubishi Heavy Industries, Ltd.'s "PA-10"). The configuration of the arm is illustrated in Figure 6. The 2nd and the 3rd layers of the forward kinematics model and the experts had 40 neurons each. 16,384 desired end-effector positions were generated by using a uniform random number generator for the estimation of the RMS of $e(k)$.

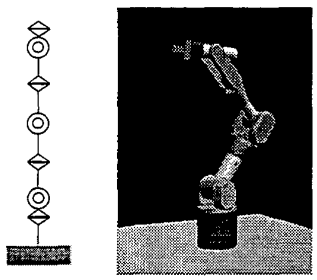


Figure 6 Configuration of 7-DOF Robot Arm

Figure 7 shows the change of the RMS error of the end-effector position by the learning. After 10^7 learning trials, 6 experts were generated. After 10^7 learning trials, the RMS end-effector position error became $9.80 \times 10^{-3} m$. We concluded that the proposed method succeeded in the inverse kinematics model learning of a 7-DOF manipulator.

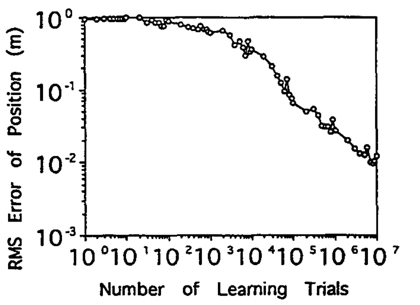


Figure 7 Change of RMS Error

5 Conclusions

We proposed a novel modular neural network architecture for the inverse kinematics model learning and tested it by numerical experiments. The proposed neural networks can approximate the inverse kinematics function that conventional methods cannot learn precisely. The proposed architecture can easily be applied to the tracking control. The slight modification of the learning algorithm for the tracking control will be reported in near future.

References

[1] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, Vol.3, pp.79-87, 1991.

[2] R. A. Jacobs and M. I. Jordan, " Learning Piecewise Control Strategies in a Modular Neural Network Architecture," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.23, pp.337-345, 1993.

[3] H. Gomi and M. Kawato, "Recognition of Manipulated Objects by Motor Learning with Modular Architecture Networks," *Neural Networks*, Vol. 6, pp.485-497, 1993.

[4] M. Kawato, "Multiple Paired Forward-Inverse Models in the Cerebellum," *Proc. of The Fifth International Conference on Neural Information Processing (ICONIP'98)*, Vol.3, pp.1177-1180, 1998

[5] M. I. Jordan, "Supervised Learning and Systems with Excess Degrees of Freedom," *COINS Technical Report, 88-27*, pp.1-41, 1988.

[6] D. DeMers and K. Kreutz-Delgado, "Solving Inverse Kinematics for Redundant Manipulators," in *Neural Systems for Robotics*, O. Omidvar and P. v. d. Smagt ed., Academic Press, 1997.

[7] M. Kawato, K. Furukawa and R. Suzuki, "A Hierarchical Neural-network Model for Control and Learning of Voluntary Movement," *Biological Cybernetics*, Vol.57, pp.169-185, 1987

[8] A. W. Moore, "Fast, Robust Adaptive Control by Learning only Forward Models," in J. E. Moody and S. J. Hanson and R. P. Lippmann ed., *Advances in Neural Information Processing Systems 4*, pp.571-578, 1992.

[9] E. Oyama and S. Tachi, "Inverse Model Learning by Using Extended Feedback System," *Proc. of the fifth International Symposium on Measurement and Control in Robotics(ISMCR'95)*, pp.291-296, 1995.

[10] M. Kawato, "A Optimization and Learning in Neural Networks for Formation and Control of Coordinated Movement," ATR Technical Report, TA-A-0086, 1990.

[11] E. Oyama and S. Tachi, "Coordinate Transformation Learning of Hand Position Feedback Controller by Using Change of Position Error Norm," *Advances in Neural Information Processing Systems 11*, 1999.