



An unsupervised learning based neural network approach for a robotic manipulator

Akanshu Mahajan¹ · H. P. Singh² · N. Sukavanam³

Published online: 22 February 2017

© Bharati Vidyapeeth's Institute of Computer Applications and Management 2017

Abstract This paper presents a neural network approach for solving the inverse kinematics of a robotic manipulator. Inverse kinematics equations are more challenging than the forward kinematics equations and therefore are more computationally complex to solve. Here, we are using a neural network approach due to its ability to give more accurate results in complex situations as compared to the other approaches. Moreover, we are using this model for trajectory tracking of a two DOF robotic arm to test its validity in real life situations.

Keywords Robotic manipulator · Artificial neural network · Trajectory tracking · Kinematics · Robotic ball catching

1 Introduction

Forward kinematics problem calculates the end effector's location in the Cartesian Space using joint variables as input. Conversely, inverse kinematics problem deals with

obtaining the required manipulator joint values for a given value of end effector's position and orientation in Cartesian Space. In case of rotational joints, the joint variables are the angles between the links and in the case of prismatic joints, these are the angles between the link extensions [1, 2].

It is a quite established fact that solving the inverse kinematics problem is more challenging and complex as compared to the forward kinematics problem [1–4]. The complexity of the problem is described by the robot's geometry and the nonlinear trigonometric equations that describe the relationship between Cartesian frame and co-ordinate frame. The fundamental methods to solve inverse kinematic problem are: geometric, algebraic and iterative methods. A closed form solution is desirable for many applications but it is difficult to find it using the traditional methods as the number of degrees of robotic manipulator increases [3, 4]. Application of neural networks in the field of image recognition, speech recognition, data fitting etc. show us their capability to work with complex functions [5, 6]. A neural network architecture with 6 sub-neural networks to solve the inverse kinematics problem for robotics manipulators with 2 or higher degrees of freedom is proposed in [7]. In Ref. [8], Jacobian Matrix is used to solve the problem of inverse kinematics with neural network. In Ref. [10], an evolutionary approach based on a real-coded genetic algorithm is used to obtain the solution of the multimodal inverse kinematics problem of industrial robots. A neural network to produce the solution to the inverse kinematics problem for a three-link robotic manipulator is investigated in [14]. The neural network is trained using the data provided by the forward kinematics to learn the inverse forward mapping of the configuration space.

This paper investigates the use of neural networks to solve the inverse kinematics problem for a two degree of freedom (DOF) robotic manipulator using unsupervised

✉ H. P. Singh
harendramaths@gmail.com

Akanshu Mahajan
akanshumahajan@outlook.com

N. Sukavanam
nsukavanam@gmail.com

¹ Department of Electronics, Sri Venkateswara College, Delhi University, Delhi 110021, India

² Cluster Innovation Centre, Delhi University, Delhi 110007, India

³ Department of Mathematics, Indian Institute of Technology, Roorkee 247667, India

learning. We only give the end-effectors' Cartesian coordinates as input to the neural network and expect the neural network to give us the joint angles as output. The remainder of this paper is organized as follows. Problem formulation is presented in Sect. 2. Numerical simulation results are included in Sect. 3. Final conclusion is given in Sect. 4.

2 Problem formulation and neural Network Implementation

Consider a hypothetical robotic manipulator for testing the proposed algorithm. It is a two-link planar manipulator with rotational joints and the link lengths are given by:

$$\begin{aligned} L_1 &= 4, L_2 = 5 \quad (\text{For Problem 1}) \\ L_1 &= 0.5, L_2 = 0.5 \quad (\text{For Problem 2}) \end{aligned} \quad (1)$$

The joint angles of the robotic manipulator have a range of $[0:2\pi]$. The workspace of the robotic arm can be given by Eq. (2)

$$|L_1 - L_2| \leq (x^2 + y^2)^{1/2} \leq L_1 + L_2 \quad (2)$$

The calculation of inverse kinematic solution for any values of x-coordinate and y-coordinate satisfying the above condition is possible using the proposed algorithm. For the values of x-coordinate and y-coordinate not satisfying the above equation, the result will be ambiguous since the manipulator cannot reach that point. Figure 1 shows the representation of a two degrees of freedom robotic manipulator with $OA = L_1$ and $AB = L_2$. The axis of rotational joints of these links is perpendicular to their plane.

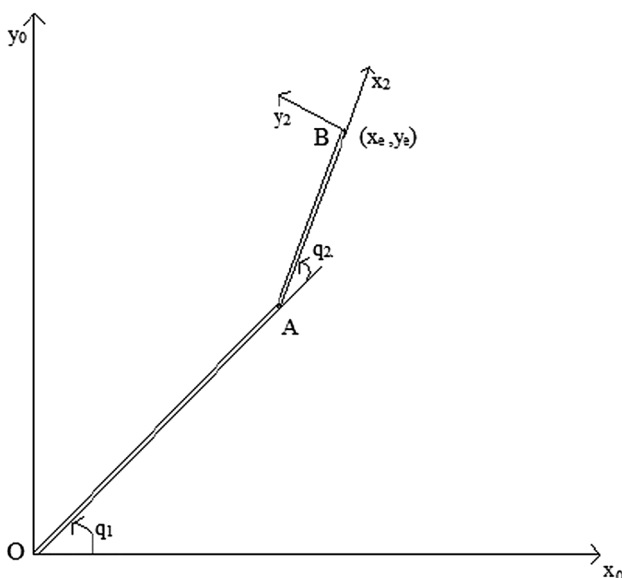


Fig. 1 A two link planar manipulator

Forward kinematics is the use of known parameters of a robotic manipulator along with q_1 and q_2 to find the x-coordinate and y-coordinate in the Cartesian coordinate system. The forward kinematic equations for the two link planar manipulator given in Fig. 1 are given as

$$\begin{aligned} x_e &= L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) \\ y_e &= L_1 \sin(q_1) + L_2 \sin(q_1 + q_2) \end{aligned} \quad (3)$$

where L_1 and L_2 represent the link lengths and q_1 and q_2 represent the known joint angles.

To test the validity of the proposed algorithm, we have the following two problems.

2.1 Problem 1

The robotic arm is required to draw a circular trajectory in its workspace. The points on this trajectory is given by the parametric equation of the circle

$$\begin{aligned} x_p &= x_c + r \cos \Theta \\ y_p &= y_c + r \sin \Theta \\ \Theta &= [0 : 2\pi] \end{aligned} \quad (4)$$

where r is the radius of the circle, (x_p, y_p) are the coordinates of the centre of circle and Θ is the parametric variable.

The neural network is executed for every increase in Θ by one degree in Eq. (4) till it reaches 2π , therefore forming a circular trajectory. The neural network gives the inverse kinematic solution every time Θ increases and thus we end up having the angles for a two degrees of freedom robotic manipulator which would result in a circular trajectory.

2.2 Problem 2

The robotic arm is required to catch a ball approaching in its workspace. The trajectory of the ball is supposed to be given by the following equation

$$\begin{aligned} x_b &= -2 + 0.15t^2 - 0.01t^3 \\ y_b &= -4 + 0.6t^2 - 0.04t^3 - 0.0225t^4 + 0.003t^5 - 0.0001t^6 \end{aligned} \quad (5)$$

The time (t) is incremented by 0.2 till it reaches its maximum value i.e. 10. For every sample value, the slope of the line joining the ball and end-effectors' position is calculated. The inverse of the slope is calculated which gives us the angle. The derived angle is used as a variable in Eq. (6) which acts as an input to the neural network.

$$\begin{aligned} x_p &= o_1 + v \cos \beta \\ y_p &= o_2 + v \sin \beta \\ v &= k * d \\ \beta &= \tan^{-1}(y_b - o_2)/(x_b - o_1) \end{aligned} \quad (6)$$

Here $(y_b - o_2)$ and $(x_b - o_1)$ are always taken to be positive, v represents the velocity of the robotic arm which keeps on decreasing as the ball approaches the end-effector and k is taken to be the constant initial velocity. Once the ball is caught by the manipulator, it follows the trajectory of the ball for some time before coming to rest. For the manipulator to show good results, it is important that the ball should enter the workspace of the manipulator as early as possible.

2.3 The artificial neural network

Since there is a significant amount of research going on to develop artificial systems that can respond to environmental changes intelligently and not in a pre-programmed manner, a lot of algorithms have been proposed by different authors for the same. Fuzzy logic and PID controller based systems have already made their way into various industries but in the past decade a considerable amount of research has been done in the area of artificial neural networks [7–10]. Artificial neural networks have proved to be more feasible and accurate in complex tasks such as image processing and have yielded great results even in an undesirable situation.

In this paper we are solving the inverse kinematics problem for a two-DOF robotic manipulator using a feed-forward neural network which has two input neurons, four hidden neurons and two output neurons as shown in Fig. 2.

In Fig. 2, the x-coordinate and y-coordinate is given as an input to the neural network and the angles q_1 and q_2 are the outputs. The transfer function for the hidden layer is taken to be a sigmoid function.

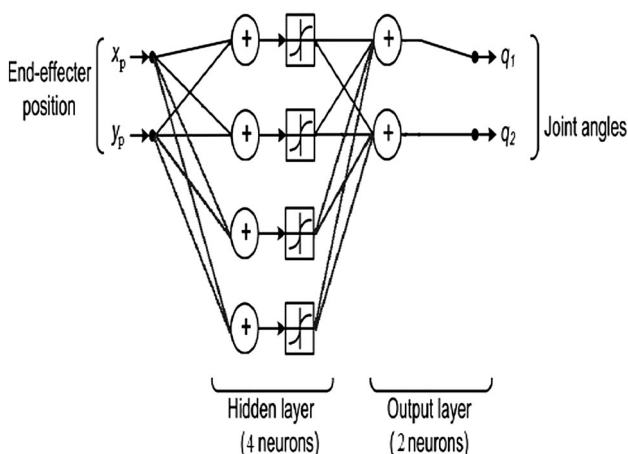


Fig. 2 The architecture of neural network

2.4 Implementation

The training algorithm used for the neural network is unsupervised learning i.e. no prior target data set is given to the neural network. This algorithm is different from other algorithms because it gives us a method to implement neural network controller on robots with less data requirement i.e. no data is required initially [8, 9, 12]. Hence, the memory consumed by desired output matrix is eliminated. Whereas, in other algorithms we require a desired output matrix for supervised learning [11–13].

The unsupervised learning is implemented by considering that the neural network has an invisible layer after the output layer having two neurons each of which contains forward kinematics equation for x-coordinate and y-coordinate respectively. The input to this invisible layer is q_1 and q_2 , respectively and the outputs are o_1 and o_2 , respectively, given by Eq. (7).

$$\begin{aligned} o_1 &= L1 * \cos(q_1) + L2 * \cos(q_1 + q_2) \\ o_2 &= L1 * \sin(q_1) + L2 * \sin(q_1 + q_2) \end{aligned} \tag{7}$$

An important fact about the invisible layer is that this layer is not directly connected to the output layer using any weight and hence updating the weight is not done in this layer. This approach can be represented by the block diagram given in Fig. 3.

The desired output values of the neural network are the inputs (x_p, y_p) only and therefore, the error of the neural network is calculated using the formula given in Eq. (8).

$$E = [(x_p - o_1)^2 + (y_p - o_2)^2] \tag{8}$$

The objective is to minimize this error using artificial neural network. This is done by updating weights using delta (δ) which is the partial differentiation of error with respect to weights for which the equations are given by

$$\delta(i, j) = \partial E / \partial W(i, j) \quad i = [1 : 4], j = [1, 2] \tag{9}$$

$$\delta(k, l) = \partial E / \partial W(k, l) \quad k = [1, 2], l = [1 : 4] \tag{10}$$

Equation (9) calculates delta for hidden layer and Eq. (10) calculates delta for input layer. The updated weights are given by the formula

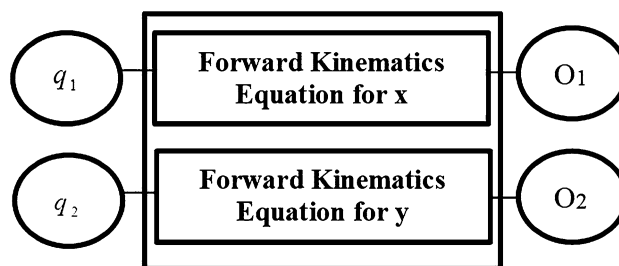


Fig. 3 Block diagram representation of invisible layer

$$W(n + 1) = W(n) - (\text{learning rate}) * (\delta) \tag{11}$$

Here n represents the iteration number and learning rate is taken according to the problem. The maximum number of iterations is 5000 and the neural network stops processing if the error reaches below a certain threshold value set by the user.

3 Results

3.1 Problem 1

The coordinates of the points on the desired trajectory, which is a circle of radius one having center at (3, 4) are given as an input to the neural network and Θ is varied from $[0:2\pi]$ in steps of one degree. We were able to obtain satisfactory results using the above proposed algorithm. Figure 4 shows the desired trajectory and the actual trajectory of the robotic manipulator.

It is evident from Fig. 4 that there is a negligible amount of difference in both the trajectories. Figure 5 shows the value of error for each input sample.

It is evident from Fig. 5 that for every input sample, the error converged to a value less than the threshold value which was taken to be 0.0002.

Artificial neural networks are considered to be computationally complex and time consuming due to high number of iterations. But in our model, the maximum number of iterations taken to reach the threshold error value is 451, thereby proving its efficiency. The number of iteration along with input sample is given in Fig. 6.

The convergence of error with respect to the number of iterations for first input value (which took the maximum no. of iterations) is shown in Fig. 7. As expected from a neural network controller, the error converges to zero.

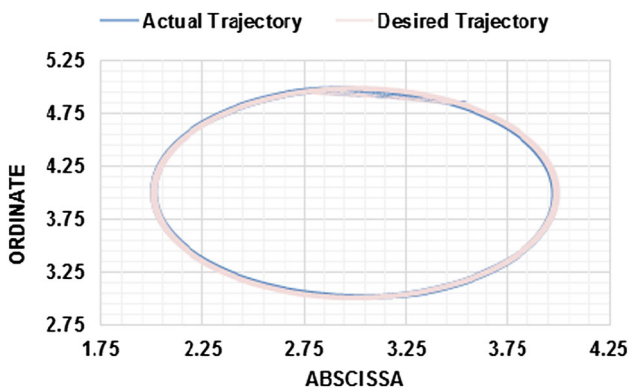


Fig. 4 Actual trajectory versus desired trajectory

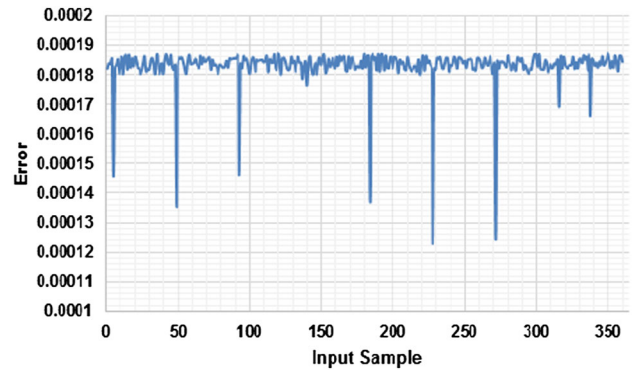


Fig. 5 Error/input sample

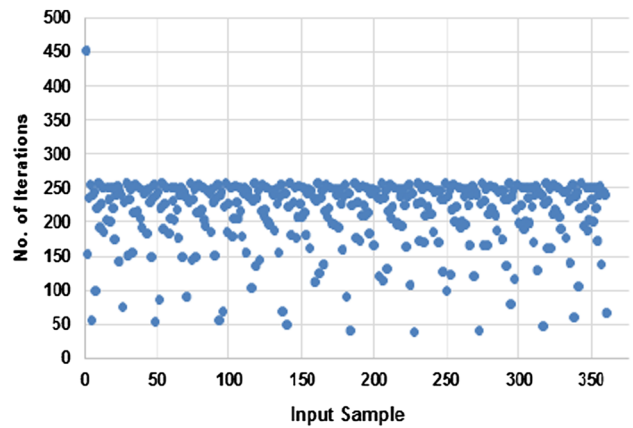


Fig. 6 Number of iterations versus input Sample

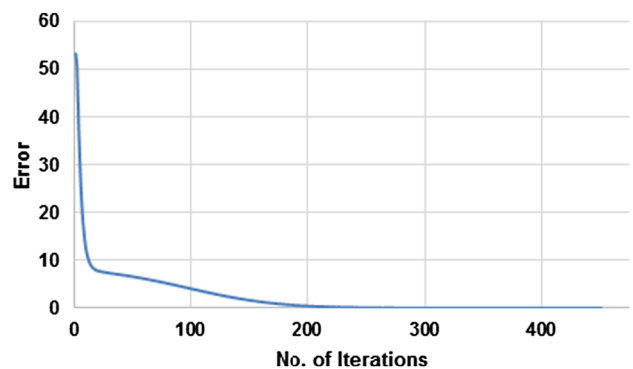


Fig. 7 Error versus number of iterations

3.2 Problem 2

In this problem, the coordinates of the trajectory of ball is given by Eq. (5) and the robotic manipulator trajectory is given by Eq. (6). The above proposed algorithm performed exceptionally well and gave satisfactory results. Figure 8 shows the ball trajectory and the actual trajectory of the robotic manipulator whereas Fig. 9 shows the number of iterations with respect to the input sample.

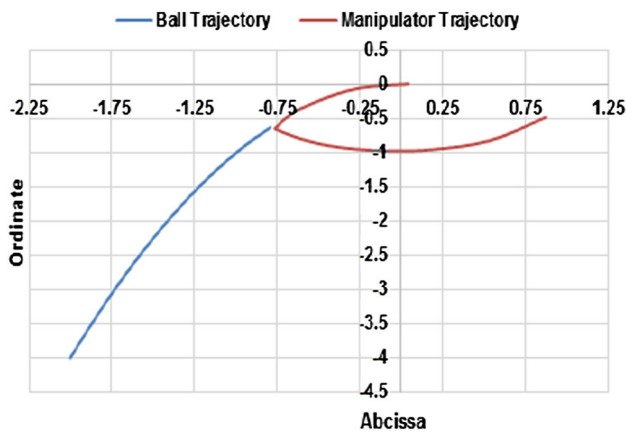


Fig. 8 Ball trajectory versus manipulator trajectory

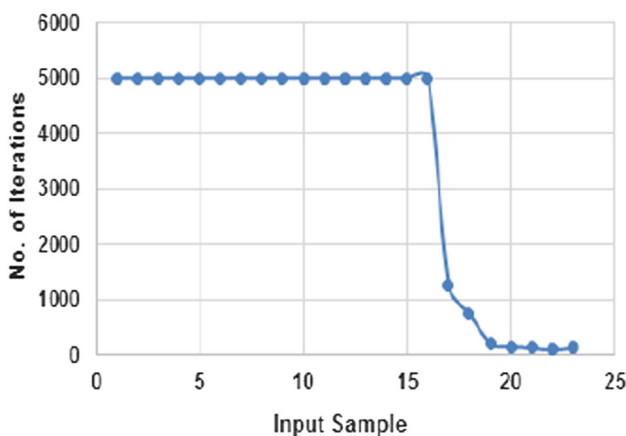


Fig. 9 No. of iterations versus input sample

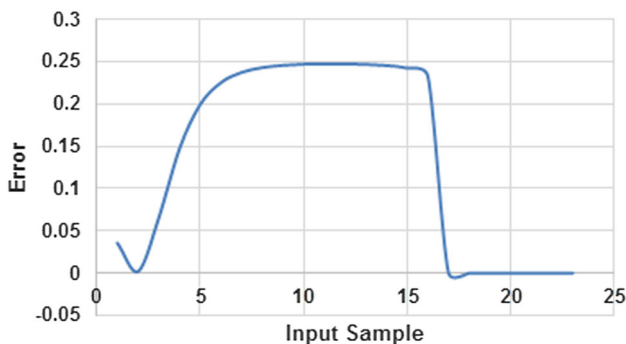


Fig. 10 Error versus input sample

As can be seen in the above figure, some the samples stopped only when the number of iteration exceeded the threshold value which was set to be 5000. This is because of the limitation of manipulator's workspace. The desired location of the end-effector's was out of reach due to constrained motion and hence the neural network could not converge. But as soon as the ball enters the workspace of the robotic arm, the algorithm starts to converge within very little iteration.

Figure 10 shows that the error remained above the threshold value when the ball was outside the workspace of the robotic manipulator. But as soon as it entered the workspace of the robotic manipulator, the error fell below the threshold value. The threshold value of error in this problem is 0.0002.

4 Conclusion

The results shown in the previous section are obtained using Microsoft Excel. The inverse kinematics problem of a robotic manipulator is solved by an artificial neural network using unsupervised learning. The solution is useful for practical purposes since the error is negligibly small. It would also be better than already used methods because of its ability to adapt itself with the change in environmental conditions. The results show promising trajectory tracking capabilities and hence the authors will try to apply this work in real world problems.

Acknowledgements This work is financially supported by university of Delhi, New Delhi, India.

References

1. Lee GCS (1982) Robot arm kinematics, dynamics and control. *Computer* 15(12):62–79
2. Fu KS, Gonzalez RC, Lee CSG (1987) *Robotics-control, sensing, vision and intelligence*. McGraw-Hill, Singapore
3. Aristidou A, Lasenby J (2009) *Inverse kinematics: a review of existing techniques and introduction of a new iterative fast solver*. Cambridge University Engineering Department, Technical Report
4. Kucuk S, Bingul Z (2006) Robot kinematics: forward and inverse kinematics. In: Cubero S (ed) *Industrial robotics: theory, modelling and control*. InTech, pp 117–148
5. Daugman JG (1988) Complete discrete 2D Gabortransforms by neural networks for image analysis and compression. *IEEE Trans ASSP* 36:1169–1179
6. Lippmann RP (1989) Review of neural networks for speech recognition. *Neural Comput* 1:1–39
7. Daya B, Khawandi S, Akoum M (2010) Applying neural network architecture for inverse kinematics problem in robotics. *J Softw Eng Appl* 3:230–239
8. Hasan AT, Al-Assadi HM, Isa AAM (2011) Neural networks based inverse kinematics solution for serial robot manipulators passing through singularities. In: Suzuki K (ed) *Artificial neural networks-industrial and control engineering applications*. InTech, pp 460–477
9. Jack H, Lee DMA, Buchal R, Elmaraghy WH (1993) Neural networks and the inverse kinematics problem. *J Intell Manuf* 4:43–66
10. Kalra P, Mahapatra BP, Aggarwal DK (2006) An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots. *Mech Mach Theory* 41:1213–1229
11. Köker R (2011) A neuro-genetic approach to the inverse kinematics solution of robotic manipulators. *Sci Res Essays* 6:2784–2794

12. Pajaziti A, Cana H (2014) Robotic arm control with neural networks using genetic algorithm optimization approach. *Int J Mech Aerosp Ind Mechatron Manuf Eng* 8:1431–1435
13. SreenivasTejomurtula SubhashKak (1999) Inverse kinematics in robotics using neural Networks. *Inf Sci* 116:147–164
14. Duka AV (2014) Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. *Proc Technol* 12:20–27