

Machine learning assisted inverse kinematics of an IRB 120 robotic arm

Provided by:

Amirhossein Mohammadi

Amirhossein Zare

Amirhossein Khalgh

1. Introduction

The problem of computing the position and orientation of the end-effector of a manipulator is called direct kinematics. There are many ways to solve direct kinematics problems based on the geometry of the robotic manipulator. In these problems, the position or angle of each robot joint is known and orientation of the tool relative to the user's workstation is then determined at the given angles.

On the other hand, in lots of real world problems the end-effector's position is specified and the objective is to find the position or angle of each robot joint, which leads to inverse kinematics. Inverse kinematics are extremely important, as they allow us to determine the position of the robot joints needed to position the robot end-effector at a certain point in space.

Solving inverse kinematics problems can be extremely challenging and sometimes impossible using classic methods. These problems are somehow related to reverse engineering. In reverse engineering the initial input is unknown and only the output is available. Therefore the effort is based on determination of the initial input from the output.

The IRB 120 is one of ABB Robotics latest generation of 6-axis industrial robot, with a payload of 3 kg, designed specifically for manufacturing industries that use flexible robot-based automation. Since it's a 6-axis robot, it has 6 joints and eventually 6 degrees of freedom.



Figure 1 – IRB 120 robotic manipulator

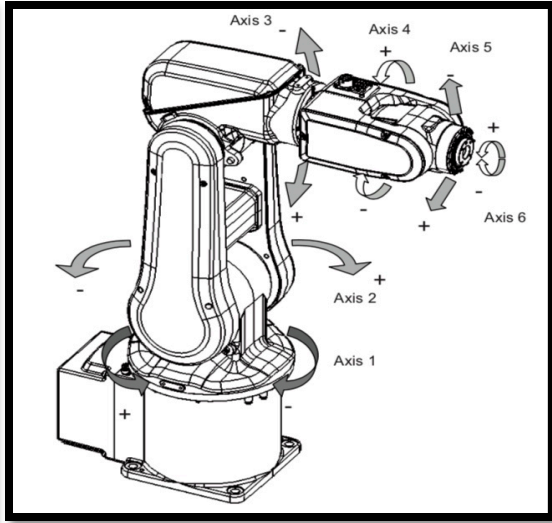


Figure 2 – Manipulator axes

Machine learning is a subset of artificial intelligence that provides algorithms to learn from data and apply that knowledge to a new data or input. These algorithms are used to predict and optimize a target based on the past information. Machine learning can be used to solve complex inverse kinematics problems.

To develop machine learning models a training phase is required. However, there are potential challenges in applying this method, such as the need for accurate and reliable data input. Therefore lots of reliable datasets are obtained from experimental tests or accurate computer simulations.

2. Methods and Data

At first algebraic solution technique applied to a manipulator with six degrees of freedom is analyzed to some extend.

$${}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= {}^0_1T(\theta_1){}_1^2T(\theta_2){}_2^3T(\theta_3){}_3^4T(\theta_4){}_4^5T(\theta_5){}_5^6T(\theta_6)$$

Therefore,

$$[{}^0_1T(\theta_1)]^{-1} {}^0_6T = {}^1_2T(\theta_2){}^2_3T(\theta_3){}^3_4T(\theta_4){}^4_5T(\theta_5){}^5_6T(\theta_6).$$

Inverting transform matrix from 1 to 0 leads to

$$\begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^1_6T$$

When looking for a solvable equation, the straightforward method of multiplying each side of a transform equation by its inverse is frequently useful for sorting out variables.

$$-s_1 p_x + c_1 p_y = d_3.$$

To solve an equation of this form, trigonometric substitutions are made:

$$p_x = \rho \cos \phi, \text{ and}$$

$$p_y = \rho \sin \phi,$$

where

$$\rho = \sqrt{p_x^2 + p_y^2},$$

$$\phi = \text{Atan2}(p_y, p_x).$$

By substituting the given equations:

$$\sin(\phi - \theta_1) = \frac{d_3}{\rho}.$$

Hence,

$$\cos(\phi - \theta_1) = \pm \sqrt{1 - \frac{d_3^2}{\rho^2}},$$

and so

$$\phi - \theta_1 = \text{Atan2} \left(\frac{d_3}{\rho}, \pm \sqrt{1 - \frac{d_3^2}{\rho^2}} \right).$$

Finally, the solution for θ_1 may be written as

$$\theta_1 = \text{Atan2}(p_y, p_x) - \text{Atan2} \left(d_3, \pm \sqrt{p_x^2 + p_y^2 - d_3^2} \right).$$

$$\begin{aligned} c_1 p_x + s_1 p_y &= a_3 c_{23} - d_4 s_{23} + a_2 c_2, \\ -p_x &= a_3 s_{23} + d_4 c_{23} + a_2 s_2. \end{aligned}$$

If equations get squared and be added the resulting equations:

$$a_3 c_3 - d_4 s_3 = K,$$

where

$$K = \frac{p_x^2 + p_y^2 + p_x^2 - a_2^2 - a_3^2 - d_3^2 - d_4^2}{2a_2}.$$

The equation can be solved by the same kind of trigonometric substitution to yield θ_3

$$\theta_3 = \text{Atan2}(a_3, d_4) - \text{Atan2}(K, \pm \sqrt{a_3^2 + d_4^2 - K^2}).$$

$$[{}^0_3T(\theta_2)]^{-1} {}^0_6T = {}^3_4T(\theta_4) {}^4_5T(\theta_5) {}^5_6T(\theta_6),$$

or

$$\begin{bmatrix} c_1 c_{23} & s_1 c_{23} & -s_{23} & -a_2 c_3 \\ -c_1 s_{23} & -s_1 s_{23} & -c_{23} & a_2 s_3 \\ -s_1 & c_1 & 0 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^3_6T,$$

Likewise other angles are determined. The complete solution is out of context of this article.

It is obvious that algebraic method is extremely complex and doesn't always lead to an accurate solution. Therefore, machine learning algorithms can be more useful to handle complex problems.

The data is generated by uniformly randomly generating values of joints within possible ranges for each of the six joints of the robotic manipulator in radians. With these values generated, the direct kinematic equations are used to determine the x , y and z coordinate for the given joint coordinates. This pair of vectors constitutes a single data point, while the entire dataset consists of 15,000 data points generated in this manner. A single CSV file is attached and contains the values of q_1 , q_2 , q_3 , q_4 , q_5 and q_6 that are uniformly randomly generated while the values x , y , and z are calculated based on the direct kinematics model obtained through Denavit-Hartenberg operation.

1/12/24, 12:53 AM

Untitled - Jupyter Notebook

```
In [13]: import pandas as pd

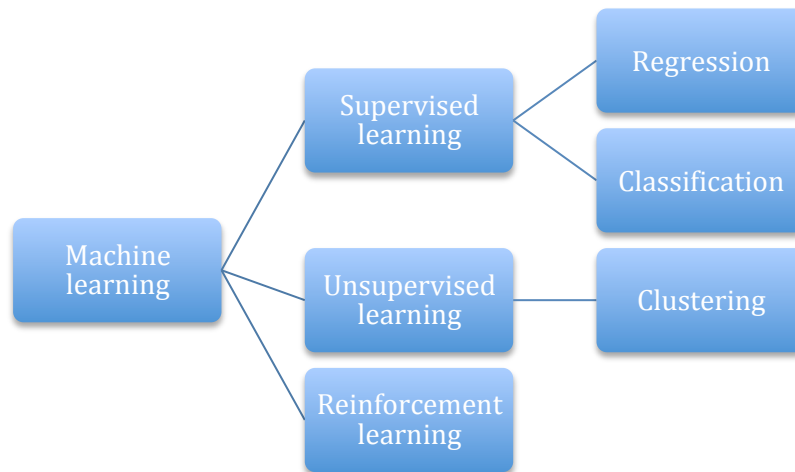
df = pd.read_csv("robot_inverse_kinematics_dataset.csv")
print("The Dataset is: ")
df
```

The Dataset is:

```
Out[13]:
```

	q1	q2	q3	q4	q5	q6	x	y	z
0	-1.510	-0.763	1.85	-0.817	0.9120	2.320	-0.0947	0.15000	0.301
1	-2.840	0.520	1.58	-1.270	-1.3900	0.617	0.1420	-0.10000	0.225
2	-1.230	0.695	1.22	-1.130	0.0343	6.270	-0.0833	0.22300	0.206
3	-1.990	1.060	1.74	-1.760	-1.2400	4.760	0.1350	-0.03140	0.370
4	1.050	0.836	1.34	-1.890	0.4840	4.380	-0.0560	-0.22900	0.260
...
14995	0.314	-0.534	1.76	1.970	-0.6990	3.870	-0.1130	-0.12800	0.257
14996	2.450	1.360	1.55	2.780	-0.3210	5.310	0.0633	-0.03160	0.450
14997	2.620	1.410	1.56	2.540	1.0600	5.870	0.1310	-0.16000	0.362
14998	-1.890	1.850	1.51	1.090	0.6970	4.070	0.0829	-0.01600	0.441
14999	2.680	-1.790	1.79	2.620	1.5900	2.640	-0.1570	-0.00369	0.254

Machine learning algorithms are generally divided to 3 categories.



Explanation about these subsets in details is out of context. The inverse kinematics problem is a regression problem. Therefore, supervised learning algorithms must be used with the purpose of regression. Two of the most famous and powerful machine learning algorithms that can be used for supervised problems, are **Decision tree** and **Random forest**.

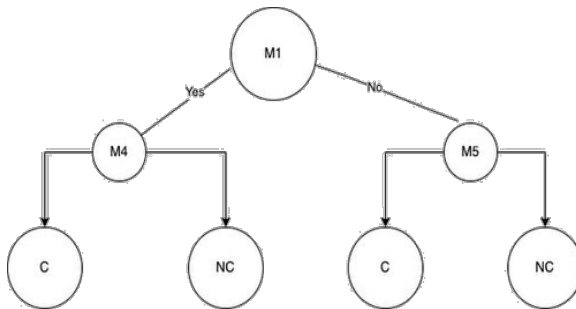


Figure 3 – Decision tree

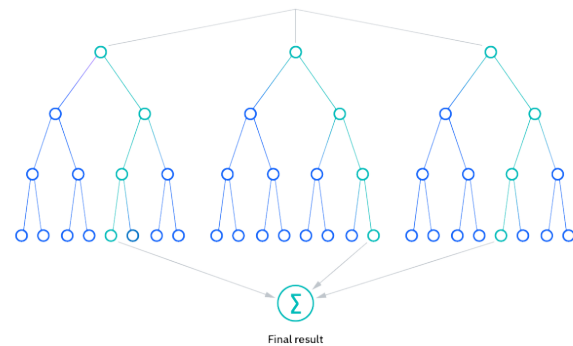


Figure 4 – Random forest

These algorithms are written in python and scikit-learn library and the results are shown in this article. Note that one of the most important concepts is how an algorithm or a method is reliable. Hence, there is a parameter in machine learning called coefficient of determination also known as R squared that determines how

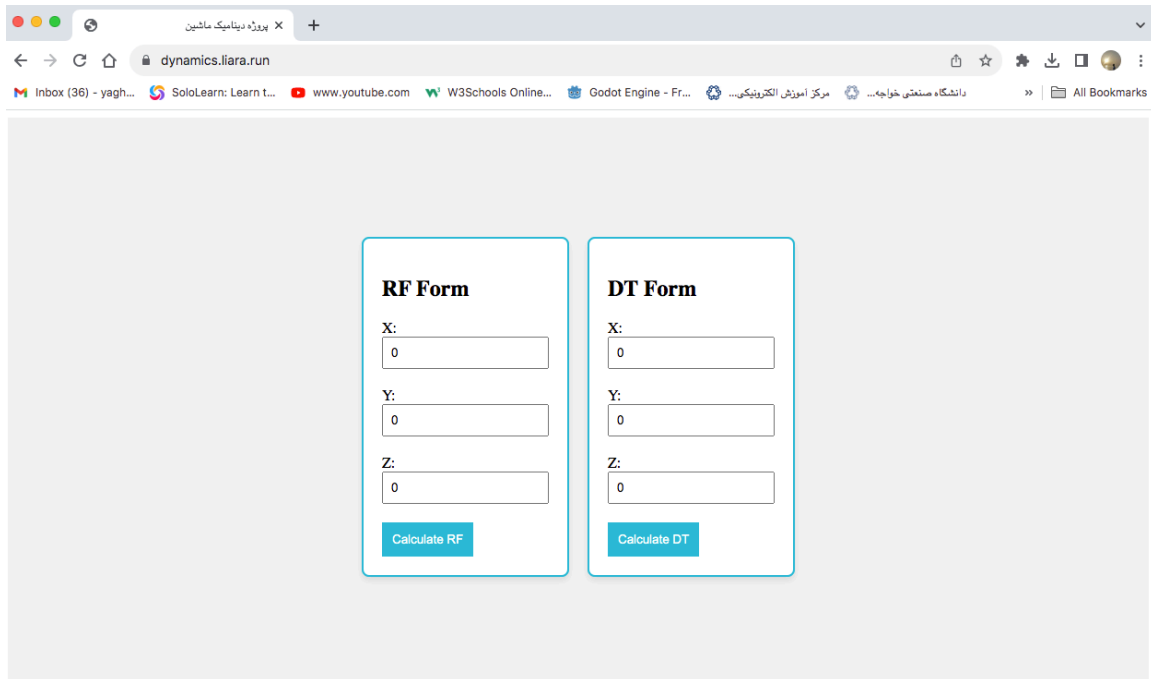
well the model fits the dataset. This parameter has a value between -1 and 0 or between 0 and 1. The closer it is to 1 or -1, it means that the algorithm in question is more accurate and reliable.

3. Results and Conclusion

A web application is provided by our team for inverse kinematics of a 6 degree of freedom manipulator. The frontend of this app is created using HTML, CSS and JavaScript and also React and Axios Frameworks. The Backend and APIs are developed using python and flask web framework. As mentioned before machine learning algorithms used for this project are decision tree and random forest written in python using scikit-learn.

All of the web app's frontend and backend code files and also machine learning notebook is attached to this report.

The web application is also deployed on the web and can be used by everyone. It is available on <https://dynamics.liara.run/>



The screenshot shows a web browser window with the URL `dynamics.liara.run`. The browser's address bar and tabs are visible at the top. The main content area displays two side-by-side forms. The left form is titled "RF Form" and contains three input fields labeled "X:", "Y:", and "Z:", each with the value "0". Below these fields is a blue button labeled "Calculate RF". The right form is titled "DT Form" and also contains three input fields labeled "X:", "Y:", and "Z:", each with the value "0". Below these fields is a blue button labeled "Calculate DT". The background of the page is a light gray.

In [14]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import r2_score
from sklearn.inspection import permutation_importance
import pandas as pd

df = pd.read_csv("robot_inverse_kinematics_dataset.csv")
features = ["x", "y", "z"]
X = df[features]
y = [df['q1'], df['q2'], df['q3'], df['q4'], df['q5'], df['q6']]

def DT(X, y, testData):
    output = []
    reg = DecisionTreeRegressor(random_state=0)
    reg.fit(X.values, y.values)

    predictedValue = reg.predict(testData)
    output.append(predictedValue[0])

    r2 = r2_score(y, reg.predict(X.values))
    output.append(r2)

    importance = reg.feature_importances_
    for i in range(3):
        output.append(importance[i])

    return output

def RF(X, y, testData):
    output = []
    reg = RandomForestRegressor(random_state=0, n_estimators=100)
    reg.fit(X.values, y.values)
    predictedValue = reg.predict(testData)
    output.append(predictedValue[0])

    r2 = r2_score(y, reg.predict(X.values))
    output.append(r2)

    importance = reg.feature_importances_
    for i in range(3):
        output.append(importance[i])

    return output
```

In [24]:

```
from IPython.display import display
test = [[-0.0947, 0.15000, 0.301]]
result = pd.DataFrame({"Predicted 01 (rad)": [DT(X, y[0], test)[0], RF(X, y[0], test)[0]],
                       "Predicted 02 (rad)": [DT(X, y[1], test)[0], RF(X, y[1], test)[0]],
                       "Predicted 03 (rad)": [DT(X, y[2], test)[0], RF(X, y[2], test)[0]],
                       "Predicted 04 (rad)": [DT(X, y[3], test)[0], RF(X, y[3], test)[0]],
                       "Predicted 05 (rad)": [DT(X, y[4], test)[0], RF(X, y[4], test)[0]],
                       "Predicted 06 (rad)": [DT(X, y[5], test)[0], RF(X, y[5], test)[0]]})

df1 = pd.DataFrame({"Real 01 (rad)": [y[0][0]], "Real 02 (rad)": [y[1][0]],
                    "Real 04 (rad)": [y[3][0]], "Real 05 (rad)": [y[4][0]]})

df2 = pd.DataFrame({"Decision tree coefficient of determination (R squared)": [DT(X, y[0], test)[1], DT(X, y[1], test)[1], DT(X, y[2], test)[1], DT(X, y[3], test)[1], DT(X, y[4], test)[1], DT(X, y[5], test)[1]],
                    "Random forest coefficient of determination (R squared)": [RF(X, y[0], test)[1], RF(X, y[1], test)[1], RF(X, y[2], test)[1], RF(X, y[3], test)[1], RF(X, y[4], test)[1], RF(X, y[5], test)[1]]})

result.index = ["Decision Tree", "Random Forest"]
df1.index = ["Value"]
df2.index = ["For 01 prediction", "For 02 prediction", "For 03 prediction", "For 05 prediction", "For 06 prediction"]

display(df1)
print("-" * 100)
display(result)
print("-" * 100)
display(df2)
```

	Real 01 (rad)	Real 02 (rad)	Real 03 (rad)	Real 04 (rad)	Real 05 (rad)	Real 06 (rad)
Value	-1.51	-0.763	1.85	-0.817	0.912	2.32

	Predicted 01 (rad)	Predicted 02 (rad)	Predicted 03 (rad)	Predicted 04 (rad)	Predicted 05 (rad)	Predicted 06 (rad)
Decision Tree	-1.51000	-0.763000	1.8500	-0.81700	0.912000	2.320000
Random Forest	-1.38802	-0.505554	1.8213	-0.71344	0.660144	2.833229

The first table above shows the real values of joint angles (θ_1 , θ_2 , θ_3 , θ_4 , θ_5 and θ_6) in radian of a test case and the second table shows predicted values of these angles by decision tree and random forest. The coefficient of determination or R squared of each algorithm in prediction of each joint angle is shown at the table below.

	Decision tree coefficient of determination (R squared)	Random forest coefficient of determination (R squared)
For θ_1 prediction	1.0	0.865935
For θ_2 prediction	1.0	0.907527
For θ_3 prediction	1.0	0.920667
For θ_4 prediction	1.0	0.845902
For θ_5 prediction	1.0	0.848114
For θ_6 prediction	1.0	0.845358

The results show that the coefficient of determination of decision tree algorithm is 1, which means it is surprisingly 100% reliable for each of the 6 angles prediction and the model completely fits the dataset.

On the other hand, random forest is also reliable enough because of high values of R squared but in this case of study decision tree is more accurate and also faster in performance.

“Genius is 1 percent inspiration and 99 percent perspiration”
-Thomas Edison-

References:

1. Introduction to robotics mechanics and control – 4th edition – John Craig
2. Machine Learning-Assisted Tensile Modulus Prediction for Flax Fiber/Shape Memory Epoxy Hygromorph Composites
3. Baressi Šegota S, Anđelić N, Mrzljak V, Lorencin I, Kuric I, Car Z. Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator. International Journal of Advanced Robotic Systems. 2021 Aug 13
4. Car Z, Baressi Šegota S, Anđelić N, Lorencin I, Musulin J, Štifanić D, Mrzljak V. Determining Inverse Kinematics of a Serial Robotic Manipulator Through the Use of Genetic Programming Algorithm. 8th International Congress of the Serbian Society of Mechanics. 2021 July 21
5. Product specification IRB 120 catalog by ABB