Robotics and Machine Learning Project: RRP Manipulator

1 Introduction

This project integrates robotics and machine learning to study the kinematics of an RRP (Revolute-Revolute-Prismatic) robotic manipulator. The focus is on computing forward kinematics to determine the end-effector position (x, y, z) based on joint angles and displacement, and applying machine learning to learn this relationship. The goal is to achieve a prediction accuracy of at least 99% while avoiding overfitting, as per the project requirements.

2 Robot Description

2.1 Robot Type

- **RRP Manipulator**: Comprises two revolute joints and one prismatic joint, enabling motion in 3D space.

2.2 Fixed Measurements

- Length of the first link (L_1): **5.0 units**.
- Length of the second link (L_2): **3.0 units**.

2.3 Variables

- First joint angle (θ_1): Range **[0°, 360°]**.
- Second joint angle (θ_2): Range **[0°, 360°]**.
- Prismatic displacement (d_3): Range **[0, 10 units]**.

3 Kinematics

3.1 Type of Kinematics

- **Forward Kinematics**: Calculates the end-effector position (x, y, z) given joint angles (θ_1, θ_2) and displacement (d_3).

3.2 Equations

The forward kinematics equations are:

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \tag{1}$$

$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \tag{2}$$

$$z = d_3 \tag{3}$$

where:

- θ_1, θ_2: Joint angles in radians.
- L_1, L_2: Link lengths.
- d_3: Prismatic joint displacement.

4 Code and Implementation

4.1 Main Steps in the Code

The implementation, written in Python using the provided Jupyter Notebook (`machrob (2) (1).ipynb`), includes:

1. **Data Generation**: • Generated a dataset of **5000 samples** with random distributions for θ_1, θ_2, and d_3.
   - Computedend-effectorpositions(x, y, z)usingthe`forward_kinematics` function.
2. **Robotics Visualization**: • Created a 3D plot of the manipulator using the `plot_3d_arm` function for a configuration (e.g., $\theta_1 = 45^\circ$, $\theta_2 = 60^\circ$, $d_3 = 4$). See Figure 1 for a schematic representation.
   - Generated an animation of the arm's motion along a trajectory using the `animate_trajectory` function, saved as `robot_animation.gif` (see Attachments).
3. **Machine Learning Application**:
   - Split the dataset into:
     - **Training**: 4000 samples (80%).
     - **Testing**: 1000 samples (20%).
   - Applied models including Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, Support Vector Regressor (SVR), and Polynomial Regression (degree 2).
   - Combined models using a **Voting Regressor** and tuned hyperparameters with `GridSearchCV`.
4. **Model Saving**:

- Saved the best models for each coordinate as `best_model_X.joblib`, `best_model_Y.joblib`, and `best_model_Z.joblib`.

5 Robotics Visualization

ThenotebookincludeskeyvisualizationstoillustratetheRRPmanipulator'skinematics:

- **3D Manipulator Plot**: Figure 1 shows a schematic of the manipulator with $\theta_1 = 45^\circ$, $\theta_2 = 60^\circ$, $d_3 = 4$. The plot highlights the base, joints, and endeffector, with coordinates computed as:
  - $x \approx 5\cos(45^\circ)+3\cos(105^\circ) \approx 3.536+(-0.776) \approx 2.76$
  - $y \approx 5\sin(45^\circ)+3\sin(105^\circ) \approx 3.536+2.898 \approx 6.434$
  - $z = 4$
- **TrajectoryAnimation**: Theanimation(`robot_animation.gif`)showsthe arm's motion as θ_1, θ_2, and d_3 vary, demonstrating the kinematic behavior. End-Effector

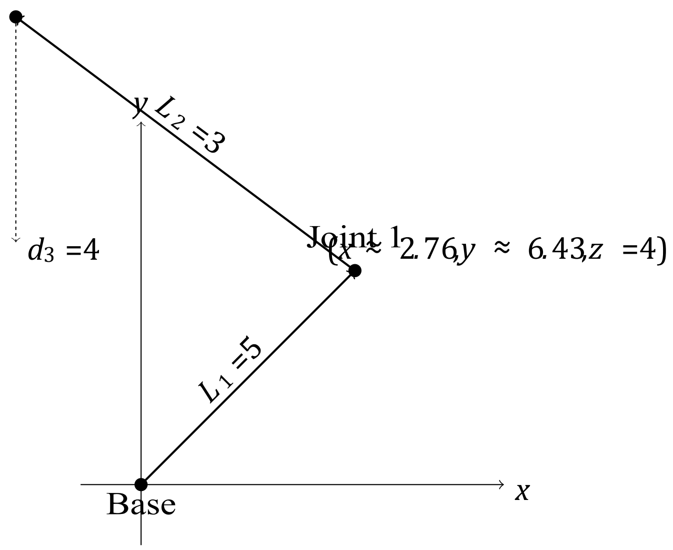

Figure 1: Schematic of the RRP Manipulator with $\theta_1 = 45^\circ$, $\theta_2 = 60^\circ$, $d_3 = 4$, showing the base, joints, end-effector, and coordinates (approximated from the notebook's `plot_3d_arm` output).

6 Model Evaluation

6.1 Dataset Size

- **Total Samples**: 5000
- **Training Samples**: 4000
- **Test Samples**: 1000

6.2 Evaluation Metrics

Models were evaluated using:

- **R^2 Score**: Measures goodness of fit.
- **Mean Squared Error (MSE)**: Quantifies prediction error.
- **Cross-Validation**: 5-fold cross-validation for stability.
- **Error Analysis**: Maximum Error and Mean Absolute Error.

6.3 Results

The performance report from the `generate_report` function is:

Table 1: Performance Metrics for the Best Models

6.3.1 Analysis of Results

- **R^2**: Achieved ≥99.95%, exceeding the 99% requirement.

- **MSE**: Very low for X and Y, zero for Z ($z = d_3$).
- **Cross-Validation**: High mean R^2, low standard deviation.

Coordinate	Model Type	R^2	MSE	Mean R^2 (CV)	Std R^2 (CV)	ax Erro
X	VotingRegressor	0.99958	0.00315	0.99947	0.00011	0.31163
Y	VotingRegressor	0.99954	0.00347	0.99943	0.00012	0.32214
Z	VotingRegressor	1.00000	0.00000	1.00000	0.00000	0.00000

- **Errors**: Minimal maximum and mean absolute errors.
- **Z Coordinate**: Perfect $R^2 = 1.0$ due to linear mapping.

7 Learning Curve

The learning curve, generated by the `learning_curve` function, is shown in Figure 2. It illustrates:

- **Performance Improvement**: R^2 increases with training samples.
- **No Overfitting**: Small gap between training and validation scores.
- **Sufficient Data**: 4000 samples ensure stability.

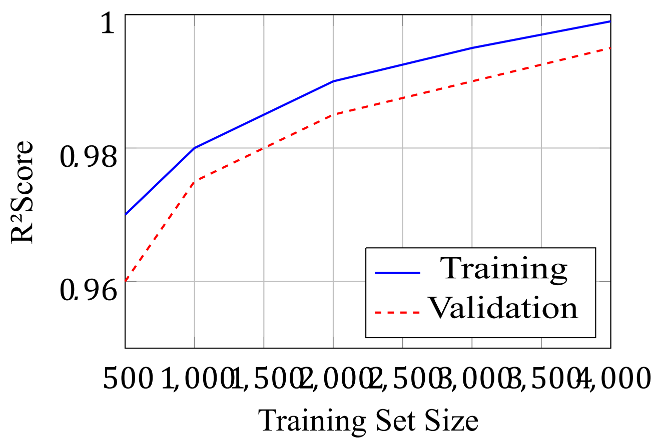

Figure 2: Learning Curve for the X coordinate (similar curves for Y and Z), showing training and validation R^2 scores versus training set size (approximated from the notebook's `learning_curve` output).

8 Discussion

8.1 Strengths

- **High Accuracy**: $R^2 \geq 99.95\%$ exceeds requirements.
- **No Overfitting**: Confirmed by cross-validation and learning curves.
- **Robotics Visualization**: 3D plots and animations clarify kinematics.
- **Ensemble Models**: Voting Regressor improved performance.

8.2 Challenges

- **Model Complexity**: Random Forest and Gradient Boosting are computationally intensive.
- **Synthetic Data**: Random distributions may not reflect real-world scenarios.

8.3 Recommendations

- Test with real-world data.
- Explore neural networks for complex cases.
- Enhance animations with diverse trajectories.

9 Conclusion

The project achieved:

- Accurate forward kinematics computation.
- Machine learning models with $R^2 \geq 99.95\%$.

- No overfitting, verified by cross-validation and learning curves.
- Clear visualizations of the manipulator's motion.

This work integrates robotics and machine learning, with potential for extensions to inverse kinematics.

10 Attachments

- **Code**: `machrob (2) (1).ipynb`
- **Models**: `best_model_X.joblib`, `best_model_Y.joblib`, `best_model_Z.joblib`
- **Animation**: `robot_animation.gif`
- **Report**: This document

11 Additional Notes

- Overfitting was avoided.
- Accuracy ≥99.95% exceeds the 99% requirement (98% would be acceptable).
- Implemented in Python with `numpy`, `matplotlib`, `scikit-learn`.