GAUSSIAN PROCESSES/ROBOT ARM DYNAMICS INTRODUCTION AND METHODS

Jacob Lister
Department of Electrical and Computer Engineerning
Department of Mathematical Sciences
Purdue University Fort Wayne
Fort Wayne, IN 46805, USA
aldrjt01@pfw.edu

ABSTRACT

In this work, I analyze the inverse dynamics model of a SARCOS seven joint robotic arm. The technique that was used to perform this analysis was Guassian Processes Regression, as the functions that dictate the movement of robotic arms are complex and almost always non-linear. In addition, the Fully Independent Conditional Approximation was used due to the large size of the dataset.

1 INTRODUCTION

In this section, I provide an overview of the inverse dynamics problem, a brief description of kinematics, and a brief explanation of the analysis that was performed.

Inverse dynamics is used when working to model joint movement, often of a robotic arm or the arms and legs of an animal or person. This technique is used to determine the torque being applied to the joints in a system based on the position, velocity, and accelation of a body (or end effector in the case of a robotic arm). Using inverse dynamics, you can compute the torques needed to follow a certain trajectory, which provides the ability to simulate biological models or allows robotic arms to ”plan” the path they will take when moving from one position to another.

The specific system I am looking at is that of a SARCOS robotic arm. This arm has 7 joints, which means 7 torques to find. Additionally, this means that there are 7 positions, velocities, and accelerations as outputs. Because I was working with inverse dynamics, the positions, velocities, and accelerations will be used as inputs, and the torques will be used as outputs. This means I will be creating a mapping of 21 inputs of 7 outputs.

Inverse dynamics can be performed in a variety of ways. The most straight-forward way is to analytically compute the torque functions of each joint in the system. This method requires full knowledge of the system and often becomes extremely complicated for many-jointed systems, so this method cannot always be used. There are also methods of estimating the torques using statistical methods. Due to the fact that the composition of torque functions is of trigonometric functions mainly, these functions end up being highly non-linear. To see more information about how the torque functions are composed, see the **Appendix**. Due to the non-linearity of the torques, simple statistic methods such as linear regression are expected to perform poorly. Guassian Processes Regression is a more robust statistical method and will be the focus of this manuscript. Guassian Processes Regression was used to model the torques of the SARCOS arm given the 21 inputs.

I chose to work on this project with the interest of learning about a method through the inverse dynamics of a robotic arm can be modeled non-analytically. As I have taken an undergraduate robotics course, and am currently taking a graduate level robotics course, I have worked with forward kinematics and a small amount of inverse dynamics. The inverse dynamics I have worked with has been solely analytic, working out each torque value in a system by hand. I have been curious about learning about non-analytic methods for a while now, and this was the perfect oppurtunity to do so.

The remaining part of this manuscript is organized as follows. Section 2 is dedicated to the methods that were used, Section 3 to my results and a discussion, and in Section 4 I will draw conclusions.

2 METHODS

In this section, I describe the data source, Guassian Processes Regression (GPR), and the Fully Independent Conditional Approximation.

2.1 DATA SOURCE AND SOFTWARE

The SARCOS dataset was sourced from the website for the textbook *Gaussian Processes for Machine Learning* Rasmussen & Williams (2006b). The analysis was performed using the software MATLAB R2023A and the `Statistics and Machine Learning Toolbox`. I switched from Python to MATLAB dues to MATLAB's strong documentation and implmentation of GPR being already present, alongside the fact that I am more comfortable with MATLAB than Python. The dataset is publicly available and the code will be publicly available in a Github Repository (Lister).

2.2 GUASSIAN PROCESSES REGRESSION

To estimate the torque functions of the SARCOS arm, I used a Guassian Processes Regression (GPR). GPR is a technique used in machine learning and more broadly in statistics. It is often used a supervised learning technique when working with non-linear systems. Because Guassian Processes (GPs) can interpreted as a distribution over a function space (Rasmussen & Williams, 2006a), we can use them to create a more robust method of regression. There are some additional advantages to working with guassian processes (scikit-learn), some of which are detailed below:

- The prediction is probabilistic (Gaussian) so that one can compute empirical confidence intervals and decide based on those if one should refit (online fitting, adaptive fitting) the prediction in some region of interest.
- Versatile: different kernels can be specified. Common kernels are provided, but it is also possible to specify custom kernels.

This versatility helps GPR work well in non-linear systems. There are some downsides to GPR, one being the fact that it uses all samples provided to perform its prediction, so the computational complexity becomes difficult to manage with large datasets; additionally, GPR loses efficiency in higher dimensional spaces (scikit-learn), but this should not be an issue as I am working with a dataset with a relatively small amount of dimensions. Because of the fact that the dataset I am working with is large, having over 40,000 samples, an approximation method for GPR will be used, which is detailed in the next subsection.

Now for an in-depth explanation of GPR. If we interpret a GP as being a distribution over a function space, we can write it as:

$$\begin{aligned} f(\mathbf{x}) &\sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \\ \mathbb{E}[f(\mathbf{x})] &= m(\mathbf{x}) \\ \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] &= k(\mathbf{x}, \mathbf{x}') \end{aligned}$$

where $\mathbf{x}$, $\mathbf{x}'$ are 2 points chosen from the set of possible inputs. Then, if we have a number of input points X_*, we can generate a random Guassian vector $\mathbf{f}_* = f(\mathbf{x}_*) \sim \mathcal{N}(0, K(X_*, X_*))$, where K is the covariances matrix between the different inputs. This becomes the prior for the function we are trying to approximate. If we take into account noise and condition the prior on the observations (training data), we have (Rasmussen & Williams, 2006a):

$$\mathbf{f}_* \mid X_*, X, \mathbf{f} \sim \mathcal{N}(K(X_*, X)K(X, X)^{-1}\mathbf{f}, K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*))$$

This means that by evaluating the mean and covariance matrices above, the function values $\mathbf{f}_*$ corresponding to the inputs X_* can be sampled from the posterior distribution to estimate $f(\mathbf{x})$ (Rasmussen & Williams, 2006a). This provides the main basis for GPR.

2.3 Fully Independent Conditional Approximation

I was originally planning to use Projected Processes as my approximation method, but ended up changing to Fully Independent Conditional Approximation. This is due to a mix of personal and school stuff happening reducting the amount of time I had to work on this, and the fact the Fully Independent Conditional Approximation was built into MATLAB, which reduced the amount of time I had to work on this.

The approximation method for GPR I used was the Fully Independent Conditional (FIC) Approximation. This approximation method modfies the Subset of Regressors method detailed in (Rasmussen & Williams, 2006a) in a way that avoids the degeneracy problem as in the Subset of Regressors method, so it is operating on a GP still. In FIC, only a subset of the full dataset is used, and depending on the data being inputted into the kernel, it may either be approximated or calculated exactly. The kernel for the FIC approximation is shown below (Quiñonero-Candela & Rasmussen, 2005):

$$\widehat{k}_{\text{FIC}}(x_i, x_j \mid \theta, A) = (1 - \delta_{ij})\widehat{k}_{\text{SR}}(x_i, x_j \mid \theta, A) + \delta_{ij}k(x_i, x_j \mid \theta, A)$$

As seen above, the kernel is approximated when the $i \neq j$, which is equivalent to the Subset of Regressors method, and calculated exactly when $i = j$. The FIC approximation of the $K(X, X \mid \theta, A)$ matrix is given as (fic):

$$\widehat{K}_{\text{FIC}}(X, X \mid \theta, A) = \widehat{K}_{\text{SR}}(X, X \mid \theta, A) + \delta_{ij}(k(x_i, x_j \mid \theta, A))$$

This method was chosen over the other approximation methods for a number of reasons (besides what I mentioned at the beginning of this section). This method was chosen over the Subset of Regressors Method due to FIC being a direct improvement to Subset of Regressors, as it has the same computational complexity as the Subset of Regressors while fixing the issue with the degeneracy of the covariance matrix (Quiñonero-Candela & Rasmussen, 2005). The Projected Processes Approximation was not used due to reasons mentioned earlier, and the Bayesian Committee Machine Method was not chosen due to it's higher computational complexity for similar performance. The last method that was not chosen was the Subset of Dataset method, and while it does have the same computational complexity, it is based on a degenerate GP, so I chose FIC over it for the same reasons as Subset of Regressors. Overall, the FIC Approximation provided a strong middle ground of the different approximation methods, leading to it being chosen.

3 Analysis and Results

In this discussion I discuss the results of my analysis. In **Table 1**, the loss of each predicted torque function is given.

Table 1: Table detailing the MSE for each torque variable (denoted q_i) estimated, kernel function for each variable, and basis function for each variable.

i	MSE for predicted q_i	Kernel function for q_i	Basis function for q_i
1	12.7200	Rational Quadratic	Pure Quadratic
2	5.0697	Matern52	Constant
3	1.3051	Matern52	Linear
4	0.8692	Matern32	Quadratic
5	0.0271	Exponential	None
6	0.3276	Rational Quadratic	Pure Quadratic
7	0.0962	Rational Quadratic	Pure Quadratic

Overall, as the model progressed through the joint torques the MSE substantially decreased. This could possibly be due to the fact that earlier joints in the system (assuming earlier indexed joint

variables are closer to the base as is standard) have a more dramatic effect on the location of the end effector, as small changes in q_1 or q_2 could cause large changes in the position, velocity, and acceleration of the arm, especially if the arm is large. The extremely low MSE for the later joints indicates that GPR performed well in terms of predicting their respective torque functions. It is possible that if I were to spend more time fine-tuning the optimization of the hyperparemeters or defined custom kernel functions that I could have achieved better performance on the earlier joint torques. Overall, the GPR seems to be able to well predict the torque variables for the SARCOS robotic arm.

4 CONCLUSIONS

In this project, I studied the problem of inverse dynamics, specifically on robotic arms. Using GPR, I was able to generate predictions for the functions of each torque variable of the arm using the position, velocity, and acceleration of each joint. It was found that GPR was better able to predict joints further from the base of the robot; future analysis could be performed in an attempt to provide predictions that generate a lower MSE for the earlier torques, possibly by defining a custom kernel for them or altering how the hyperparameters are optimized.

REFERENCES

Fully independent conditional approximation for gpr models. URL https://www.mathworks.com/help/stats/fully-independent-conditional-approximation-for-gpr-models.html.

Jacob Lister. URL https://github.com/jacobtlister/stat512-project.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005. URL http://jmlr.org/papers/v6/quinonero-candela05a.html.

Carl Rasmussen and Christopher Williams. *Guassian Processes for Machine Learning*. MIT Press, 2006a. URL https://gaussianprocess.org/gpml/.

Carl Rasmussen and Christopher Williams. Guassian processes for machine learning - data, 2006b. URL https://gaussianprocess.org/gpml/data/.

scikit-learn. 1.7. guassian processes. URL https://scikit-learn.org/stable/modules/gaussian_process.html.

5 APPENDIX

When transforming coordinate systems between different joints on a robotic arm, what ends up occuring is a series of rotations and translations. Transformations of this nature on a coordinate system are considered homogeneous transformations. Given points $P_0, P_1 \in \mathbb{R}^3$ where $P0$ is being transformed into P_1, one can express rotations and translations as matrix multiplication on the points, shown below:

$$P_0 = A_{\text{Rot, } x, \theta} P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad P_0 = A_{\text{Trans, } x, d} P_1 = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_0 = A_{\text{Rot, } y, \theta} P_1 = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad P_0 = A_{\text{Trans, } y, d} P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_0 = A_{\text{Rot, } z, \theta} P_1 = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad P_0 = A_{\text{Trans, } z, d} P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Additionally, there exists a procedure for defining the forward kinematics of a robotic arm called the Denavit-Hartenburg (DH) Procedure. In it, the following transformation is used when transforming between coordinate systems of the different robotic links (joint $i-1$ to i):

$$\begin{aligned} A_{i-1}^{i} &= A_{\text{Rot, } z_{i-1}, \theta_i} A_{\text{Trans, } z_{i-1}, d_i} A_{\text{Trans, } x_i, a_i} A_{\text{Rot, } x_i, \alpha_i} \\ &= \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

where θ_i, d_i are possible joint variables. Additionally, their exists a nice property relating different joint coordinate systems to each other. Say we have a n joint robotic arm and $0 \leq j < k \leq n$. Then the transformation matrix starting at link j and going to link m is given by:

$$A_j^k = A_j^{j+1} A_{j+1}^{j+2} \cdot A_{k-2}^{k-1} A_{k-1}^{k}$$

This means when transforming from our base coordinate to the end effector frame (end point on the robot where a robotic gripper / arm is often located), we have:

$$A_0^n = A_0^1 A_1^2 \cdot A_{n-2}^{n-1} A_{n-1}^{n}$$

Additionally, we can obtain information about the end effector coordinate system and location relative to the base coordinate system via the transformation matrix. If end effector coordinate basis vectors are denoted as **x**, **y**, **z**, and the end effector gripper location is **p**, then:

$$A_0^n = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From the way that the DH matrices are composed, alongside their relationship to the end effector coordinate system, it is clear to see that the system will be highly non-linear, especially as more rotational joints (joints with θ_i as the joint variable) are included in the system.