Learning the inverse dynamics of anthropomorphic robot arms via Gaussian Processes

Ella Smith

Purdue University Fort Wayne

Fort Wayne, IN 46805 USA

Introduction

Inverse dynamics is a method for determining torque given position, velocity, and acceleration. It is important in the study of the locomotion behaviour of animals. In fact, inverse dynamics is useful in studying how to replicate human joint movement through an anthropomorphic robot arm. An inverse dynamics model can be used in the following manner: a planning module decides on a trajectory that takes the robot from its start to goal states, and this specifies the desired positions, velocities and accelerations at each time. The inverse dynamics model is used to compute the torques needed to achieve this trajectory and errors are corrected using a feedback controller [4]. Generally, biomechanists use inverse dynamics to infer joint torques from kinematic variables. In particular, there exist physics-based rigid-body-dynamics models which give us a method to obtain the torques from the position, velocity and acceleration variables [4]. However, the real robot arm is actuated hydraulically and is rather lightweight and compliant, so the assumptions of the rigid-body-dynamics model are violated [4]. It is worth noting that the rigid-body-dynamics model is nonlinear, involving trigonometric functions and squares of the input variables. Therefore, standard models such a linear regression are expected to perform poorly in an inverse dynamics task. In this paper, we will use Gaussian Processes (GP) to learn the inverse dynamics of an anthropomorphic robot arm. A GP can be viewed as defining a distribution over a possibly infinite dimensional space of functions with inference taking place directly in the space of functions [4].

Methods

The SARCOS dataset

The data relates to an inverse dynamics problem for a seven degrees-of-freedom SARCOS anthropomorphic robot arm. The task is to map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques.The data is divided in training and test sets: There are 44,484 training examples and 4,449 test examples. The first 21 columns are the input variables, and the $22\mathrm{nd}$ column is used as the target variable. The inputs were linearly rescaled to have zero mean and unit variance on the training set. The outputs were centered so as to have zero mean on the training set. This dataset was previously analyzed in [5, 6, 7]. Due to Gaussian Processes being computationally expensive, five percent of the training and testing data was used. The five percent of the training and testing data was randomly selected and then used for the fitting and predictions made by the Gaussian Process Regression model.

Gaussian Processes

A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution. Furthermore, as it happens for the multivariate Gaussian distribution, a Gaussian Process is completely specified by its first and second moment. We define the mean function

$$\mathrm{m}(\mathrm{x}) := \mathbb{E}[\mathrm{f}(\mathbf{x})]$$

and a covariance function

$$\mathrm{k}(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(\mathrm{f}(\mathbf{x}) - \mathrm{m}(\mathbf{x}))\,(\mathrm{f}(\mathbf{x}) - \mathrm{m}(\mathbf{x}))^{\mathsf{T}}].$$

Given this definitions, we have that $\mathrm{f}(\mathbf{x})$ is distributed as a Gaussian Process:

$$\mathrm{f}(\mathbf{x}) \sim \mathcal{GP}(\mathrm{m}(\mathbf{x}), \mathrm{k}(\mathbf{x}, \mathbf{x}')).$$

Consider now the regression problem of the form

$$\mathrm{y} = \mathrm{f}(\mathbf{x}) + \epsilon$$

with ϵ independent and identically distributed Gaussian noise with variance σ_{n}^2. We get

$$\mathrm{cov}(\mathbf{y}) = \mathrm{K}(\mathrm{X}, \mathrm{X}) + \sigma_{\mathrm{n}}^2 \mathrm{I}.$$

With this, we obtain the predictive distribution as

$$\mathbf{f}_* \mid \mathrm{X}, \mathrm{y}, \mathrm{X}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \mathrm{cov}(\mathbf{f}_*))$$

with

$$\bar{\mathbf{f}}_* := \mathbb{E}[\mathbf{f}_* \mid \mathrm{X}, \mathrm{y}, \mathrm{X}_*] = \mathrm{K}(\mathrm{X}_*, \mathrm{X})[\mathrm{K}(\mathrm{X}, \mathrm{X}) + \sigma_{\mathrm{n}}^2 \mathrm{I}]^{-1}\mathbf{y}$$

and

$$\mathrm{cov}(\mathbf{f}_*) = \mathrm{K}(\mathrm{X}_*, \mathrm{X}_*) - \mathrm{K}(\mathrm{X}_*, \mathrm{X})[\mathrm{K}(\mathrm{X}, \mathrm{X}) + \sigma_{\mathrm{n}}^2 \mathrm{I}]^{-1}\mathrm{K}(\mathrm{X}, \mathrm{X}_*)$$

Software

The software MATLAB was used to convert the .mat files to downloadable .csv files. Then the Gaussian Regression Model analysis was achieved using the software Python and the following packages: sklearn.guassian-process, pandas, random, and matplotlib.pyplot. Code is available upon request.

Prediction Evaluation

The simplest method for model evaluation is possibly the *squared error loss*, which can be summarized by the *mean squared error* (MSE) by averaging over the test set. When the value of the MSE is zero, this is interpreted as the model having no errors. However, a MSE of zero is not found within practice. Instead, a lower in value MSE is an indicator of more precise predictions from a model and decreased error within the model. The MSE will be one of the methods that we will use to evaluate our predictive model. We define the MSE to be:

$$\mathrm{MSE} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i})^2}{\mathrm{n}}$$

Here, $\mathrm{y_i}$ is the i^{th} observation of the dependent variables, $\hat{\mathrm{y}}_\mathrm{i}$ is the corresponding value predicted by the model, and n is the number of observations within the data.

The other method that we will use for model evaluation is the *coefficient of determination*. The *coefficient of determination* (R^2) is the amount of explainable variance within the dependent variable that can be predicted by the independent variable. The R^2 can take any value from zero to one. The closer R^2 is to zero, the less variability the independent variable can explain in the dependent variable. And the closer R^2 is to one, the more variability the independent variable can explain in the dependent variable. We define R^2 to be:

$$\mathrm{R}^2 = \left(\frac{\mathrm{n}(\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i y_i}) - (\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i})(\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{y_i})}{\sqrt{[\mathrm{n}\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i^2} - (\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i})^2][\mathrm{n}\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{y_i^2} - (\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{y_i})^2]}}\right)^2$$

Here, n is the number of observations within the data, $\mathrm{x_i}$ is the i^{th} observation of the independent variables, and $\mathrm{y_i}$ is the i^{th} observation of the dependent variables.

Results

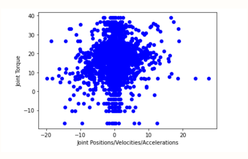

Figure 1: Plot of Test Data

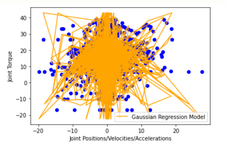

Figure 2: Plot of Gaussian Regression Model

Figure 3: Plot of Gaussian Regression Model over Test Data

Explanation

Above are three plots respectively depicting the testing data, the Gaussian Regression Model after being fit with the training data, and a plot of the testing data overlaid with the Gaussian Regression Model.

After fitting the Gaussian Regression Model and using the model to predict joint torque values given test joint movement data, the value of the MSE came out to be 53.7829 and the value of the R^2 came out to be 0.9649. These numbers seem to show that our Gaussian Regression Model is proficient. However while the MSE value could be interpreted to indicate that the model produces some but not a significant amount of error in prediction, 53.7829 provides a baseline for determining whether or not the Gaussian Regression Model will improve in reducing prediction errors in future iterations. And when considering the high R^2 with the MSE value, it appears as though overfitting is occurring within the model.

The Gaussian Regression Model could be very proficient in predicting training data, however, the model will most likely fall short with predicting testing data. Some solutions to the model's possible overfitting would be data augmentation or training the model with more data. Since utilizing Gaussian Processes is computationally expensive and a restriction of five percent of the dependent variable data was necessary in this project to avoid computation errors, training the Gaussian Regression Model with diverse data may be the best option to prevent overfitting of the model and improve the model's prediction performance in the future.

References

[1] Bengio, Yoshua and LeCun, Yann. Scaling learning algorithms towards AI. In *Large Sclae Kernel Machines.* MIT Press, 2007.

[2] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

[3] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[4] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[5] Sethu Vijayakumar and Stephan Schaal. Lwpr: An o(n) algorithm for incremental real time learning in high dimensional space. *Seventeenth International Conference on Machine Learning*, pp. 1079–1086, 2000.

[6] Sethu Vijayakumar, Aaron D'Souza, Tomohiro Shibata, Jorg Conradt, and Stephan Schaal. Statistical learning for humanoid robots. *Autonomous Robot*, 12:55–69, 22, 2002.

[7] Sethu Vijayakumar, Aaron D'Souza, and Stephan Schaal. Incremental online learning in high dimensions. *Neural Computation*, pp. 22, 24, 2005.