Uncertainties estimation and compensation on a robotic manipulator. Application on Barrett arm

Duong Dang

December 12, 2008

Abstract

In this project, two different regression methods have been used to estimate the joint friction and gravity, inertia estimation error on a 7 degree of freedom Barrett robotic arm. Using these estimations, appropriate compensation terms have been added to the servo loop to achieve an improvement of a factor of 10 in tracking error.

Acknowledgement

I would like to thank Quoc Le for his advice and a number of discussions on this problem and on machine learning in general during the quarter. I also would like to thank the AI Lab for providing the Barrett arm for this project.

1 Problem statement

The objective of this project is using learning to estimate the friction and uncertainties in dynamic parameter of the robot. This estimation will be used as a compensation term to be added to the PD controller.

The equation of motion of a manipulators consisting of n links is given by [1]

$$\tau = A(q)\ddot{q} + G(q) + b(q,\dot{q}) \tag{1}$$

Where q, $\dot{q}$ and $\ddot{q}$ are a n-dimension vector representing the joint angles,velocities and accelerations. A, G, b represent the inertia matrix, gravity term and centrifugal and Coriolis terms. The torque τ is computed based on a control signal u and the estimations of A , G and b

$$\tau = \hat{A}(q)u + \hat{G}(q) + \hat{b}(q,\dot{q}) \tag{2}$$

The control term is given from the traditional PD controller

$$u = \ddot{q}_d + K_v(\dot{q}_d - \dot{q}) + K_p(q_d - q) \tag{3}$$

where q_d is the desired trajectory. If there is no friction and all the terms are estimated precisely (i.e: $\hat{A} = A$, $\hat{b} = b$, $\hat{G} = G$), from (**??**), (**??**) and (**??**) we will have

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \tag{4}$$

Therefore the error $e = q_d - q$ will be asymptotically zero. In the real world however, it is difficult to estimate all the dynamic parameters precisely, a frictional term F must be added to the right hand side of (**??**). We then have

$$\ddot{e} + K_v\dot{e} + K_p e = f(q,\dot{q},\ddot{q}) = \hat{A}^{-1}(\Delta A\ddot{q} + \Delta b + \Delta G + F_f) \tag{5}$$

where ΔA, Δb and ΔG are estimation errors of dynamic parameters A, b, G.

To learn the function $f(q,\dot{q},\ddot{q})$, we can give the robot a desired trajectory. At each time stamp $t^{(i)}$, we record $\{q^{(i)},\dot{q}^{(i)},\ddot{q}^{(i)}\}$ and compute $f^{(i)}$ by formula (**??**). We then run a non parametric regression on the training data. The output will be the prediction f^* at any state $\{q^*,\dot{q}^*,\ddot{q}^*\}$

2 Uncertainty characteristics

Friction is a non linear, discontinuous function of $\dot{q}$. Several frictional models exist in the literature [**?**]. The simplest way is a constant force (Coulomb friction) always against the movement. A more complicated model of the friction [**?**] include Coulomb friction, viscous and Stribeck effect (Figure 2.). One thing to be noted is that all the frictional model shares a same propety. That is friction force (and therefore torque) is a asymetric function of $\dot{q}$

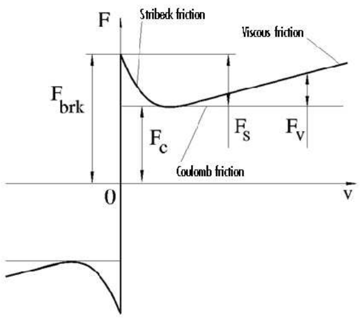

Figure 1: A complex model friction

Gravity and inertia estimation error come from incorrect estimate of joint masses and positions of center of mass. ΔG and ΔA are only function of $\mathbf{q}$

Coriolis and centrifugal forces estimation error depend upon both $\mathbf{q}$ and $\dot{\mathbf{q}}$.

3 Experiment setup

3.1 Barrett arm

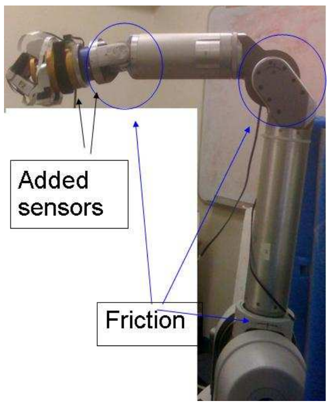

Figure 2: Sources of uncertainties on Barrett arm

Two main sources of uncertainties on the Barrett arm in this experiment are several sensors with unknown mass mounted at the hand and joint frictions.

3.2 Choice of trajectory

In this project, the goal is to learn the function f in equation (**??**) and extract individual uncertainty terms to find the joint friction profile and mass offsets. In general, f is fairly complicated. Especially, the first uncertainty term depends on both q and $\ddot{q}$. That would make it difficult to extract separate uncertainty terms in the end.

The trick here is to choose a sinusoidal function $q_d = C_1 + C_2 \sin(\omega t)$ as desired trajectory. With this setup we have $\ddot{q}_d = -C_2\omega^2 \sin(\omega t)$, we then can approximate $\ddot{q} = -\omega^2(q - C_1)$. The uncertainty term $\Delta\hat{A}\ddot{q}$ now only depends on q.

For the sake of simplicity in the problem we assume that Δb is negligible compare to other Uncertainties. This is not a bad assumption since b itself is usually small compared to other terms.

With the choice of trajectory and assumption above, equation (**??**) can be written as

$$\hat{A}(\ddot{e} + K_v\dot{e} + K_p e) = f(\dot{q}, q) = F_f(\dot{q}) + F_m(q) \quad (6)$$

Where both F_f and F_m are non linear function, non parametric of $\dot{q}$ and F_m. F_f is a asymmetric function of $\dot{q}$. This fact help us to separate the terms in the right hand side of (**??**) after doing regression.

4 Results

4.1 Gaussian process regression

Given noisy observations

$$\mathcal{D} = \{X, \mathbf{y}\}, \quad (7)$$

Gaussian process regression [**?**] predicts the mean μ and covariance Σ at patterns X^* as follow

$$\mu = K(X, X^*)\left[K(X, X) + \sigma^2 I\right]\mathbf{y} \text{ and} \quad (8)$$
$$\Sigma = K(X^*, X^*) - K(X, X^*))\left[K(X, X) + \sigma^2 I\right]^{-1} K(X, X^*) \quad (9)$$

In our problem, X and X^* are $2n \times m$ and $2n \times k$ matrices represent the training set and the query points.

$$X = [x^{(1)} \dots x^{(m)}], \qquad x = (q_1 \dots q_n \dot{q}_1 \dots \dot{q}_n)^T$$
$$X^* = [x^{*(1)} \dots x^{*(m)}], \qquad x^* = (q_1^* \dots q_n^* \dot{q^*}_1 \dots \dot{q^*}_n)^T$$

the label vector **y** is the column vector of the value found in (**??**).

$$\mathbf{y} = (y^{(1)} \dots y^{(m)})^T, \quad \mathbf{y} = \mathbf{f} + \varepsilon \text{ where } \varepsilon = \mathcal{N}(0, \sigma^2 I)$$

$K(X,X)$, $K(X,X^*$, and $K(X^*,X^*)$ are respectively $m \times m$, $k \times m$ and $k \times k$ matrices corresponding to a Gaussian kernel.

$$K(X,X)_{ij} = \exp((x^{(i)} - x^{(j)})^T \Theta (x^{(i)} - x^{(j)}))$$
$$K(X,X^*)_{ij} = \exp((x^{*((i)} - x^{(j)})^T \Theta (x^{*(i)} - x^{(j)}))$$
$$K(X^*,X^*)_{ij} = \exp((x^{*(i)} - x^{*(j)})^T \Theta (x^{*(i)} - x^{*(j)}))$$

Kernel covariance matrix Θ is a diagonal matrix that we choose.

4.1.1 Robust regression, outliers detection

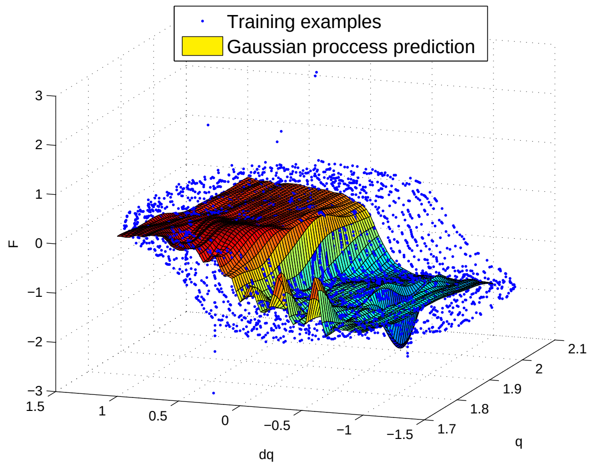

Figure 3: Naive Gaussian process regression on initial data set (5000 training examples)

Apply the naive Gaussian process regression does not provide a good prediction of uncertainty f^* (Figure **??**). The reason is outliers affect greatly Gaussian process. As a result, we need to clean up the data before doing regression. To detect outliers, we set $X^* = X$ and do regression with large bandwidth. After that we compute the error $|\mu(i) - F(i)|$ at all points. All the outliers should have a large error. We can then discard the training examples with highest error, say 1/10 of the total data set. Figure **??** show the improvement after removing outliers.

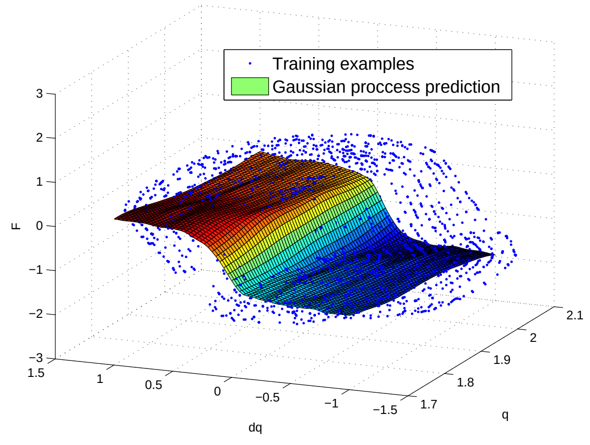

Figure 4: Regression after detecting and removing outliers (4500 training examples)

4.1.2 Hyperparameters selection

To optimize the hyperparameters, the leave-one-out-cross validation technique has been used [**?**]. In this experiment, only joint 4 is chosen to move, all other joints are fixed ($x = (q_4 \quad \dot{q}_4)^T$). We need to optimize three hyperparametters: the noise level σ, and the bandwidths θ_1 and θ_2. For each set $\{\sigma, \theta_1, \theta_2\}$, the data is randomly divided into 10 trunks then choose each of those trunks as testing points with the other nine as training data set. The total error is computed with σ ranging from 0.01 to 0.2 and θ_1,θ_2 ranging from 0.02 to 0.5. The approximate optimal parameters are $\sigma = 0.05$, $\theta_1 = 0.1$ and $\theta_2 = 0.2$

4.2 Support vector regression

The technique in section **??** does a decent job removing outliers in our particular data set. However, in general we do not know a priori the portion of outliers. Therefore, in fixing the portion of data to be removed, we might not remove all the outliers in some cases or discard too much useful data in others. Instead of Gaussian process, a support vector regression can be used to reduce the influence of outliers.

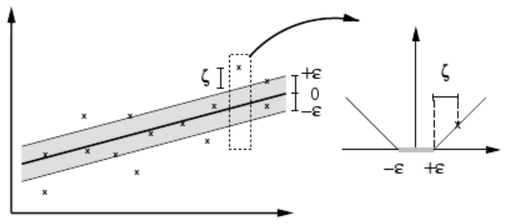

Figure 5: The soft margin loss setting for a linear SVM (from [?])

The key idea is finding the optimal line (in feature space) through the data set, with a loss function define in figure ?? Writing the optimization problem and using the Kernel trick, the dual problem can be written as

$$\text{maximize} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^* k(x_i, x_j) \\ -\epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{cases} \tag{10}$$

$$\text{subject to } \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

where x and x^* are training examples and testing points k is a Gaussian Kernel as in section ??;ϵ and C is the parametters of our choice. The predictions are

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)k(x_i, x) + b \tag{11}$$

where b is determined by the support vectors [?]. To solve problem (??) I used `CVX`, a package for specifying and solving convex programs [?, ?]. Figure ?? shows that support vector regression (red line) is much less sensitive to noise than Gaussian process regression (green line).

4.3 Friction profile and gravity, inertial estimation error

To extract the contribution of F_f and F_m to the total uncertainty in (??), we can use the fact that

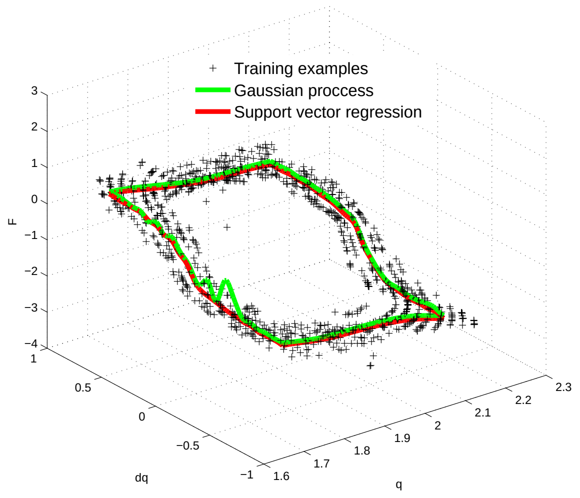

Figure 6: Comparison between Gaussian process regression and support vector regression on a noisy data set.

F_f is an asymmetric function of $\dot{q}$, therefore

$$F_m(q) = \frac{1}{2}\left[F(q, \dot{q}) + F(q, -\dot{q})\right] \tag{12}$$

$$F_f(\dot{q}) = F(q, \dot{q}) - F_m(q) \tag{13}$$

Use equations (??) and (??) to the prediction in section ?? and ??, we obtain Figure ?? and ??. Each line in Figure ??, correspond to a fix value of $\dot{q}$ in equation (??).

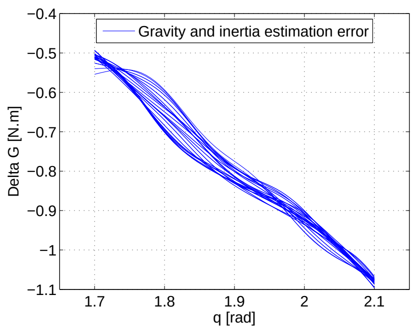

Figure 7: Gravity and inertia estimation error

One should note that if the friction profile found in Figure ?? is inherent to this particular joint and will not depend on trajectory, the gravity - inertia estimation error found here is.It is based

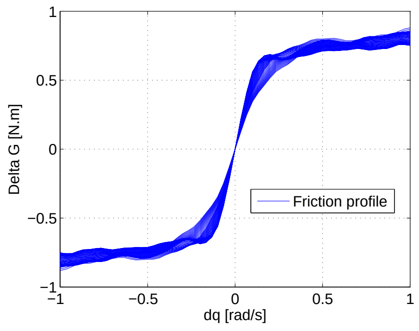

Figure 8: Friction profile

on assumption of sinusoidal trajectory and is valid only for this class of trajectory.

In our particular case, it is possible to make these estimations trajectory independent. We know that the ΔG and ΔA come mainly from the mass of various sensors mounted on the robot's hand. We can then estimated this mass by the prediction of F_m at $\ddot{q} = 0$: $F_m(q, \dot{q})\big|_{\ddot{q}=0} = \Delta G(q) = \Delta mgl$, where $\Delta m = -\sum m_{sensor}$ and l is the moment arm from the gravity force to joint 4 at the given point. The resulting offset is $\Delta m = 0.254 \pm 0.009$ kg.

4.4 Validation

Having found the friction profile and the mass offset of the arm, we can run a trajectory with the appropriate compensation terms. Figure **??** show the improvement made by the compensation term. A sinusoidal trajectory at different a frequency ($\omega = 2rad/s$) has been used for validation. The maximum tracking error has been reduced from 0.06 rad to 0.005 rad, or a factor of 10.

5 Future works

Both regression methods work well with the problem. Support vector regression is not sensitive to noise so does not requires preprocessing data. The compensation terms reduces significant tracking error. The next steps will be estimating frictions and mass error for all joints and use the compensation terms in the next version of the Barrett Arm controller that I am helping develop.

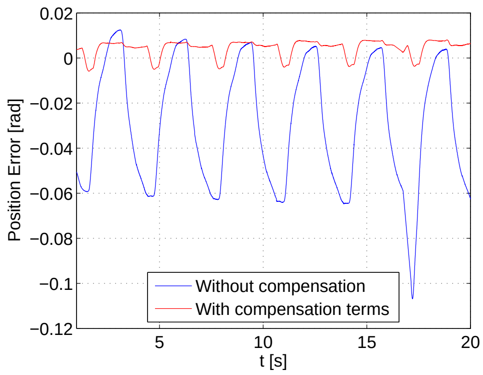

Figure 9: Tracking error with compensation terms

References

[1] Andrew Ng, *Machine Learning* CS229 Lecture Notes.

[2] Alex J. Smola, Bernhard Scholkopf *A Tutorial on Support Vector Regression* statistics and Computing, 2001.

[3] Oussama Khatib, *Introduction to Robotics.* CS223A Course Reader.

[4] C. E. Rasmussen & C. K. I. Williams, *Gaussian Processes for Machine Learning* the MIT Press, 2006

[5] H. Olsson K.J. Astrom C. Canudas de Wit M. Gafvert P. Lischinsk *Friction Models and Friction Compensation* Eur. J. Control, vol. 4, no. 3, pp. 176-195, 1998.

[6] M. Grant and S. Boyd. *CVX: Matlab software for disciplined convex programming (web page and software).* http://stanford.edu/ boyd/cvx, December 2008.

[7] M. Grant and S. Boyd. *Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar)*

[8] V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, *Lecture Notes in Control and Information Sciences*, Springer, 2008. http://stanford.edu/ boyd/graph_dcp.html.