The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

Modified Local Gaussian Process Regression for Inverse Dynamics Learning

Seung-Yoon Cho[1], Sung-Soon Yim[2], and Ju-Jang Lee[3]

[1,2,3] KAIST, Daejeon, 305-701, Korea
(Tel: 82-42-350-8032, Fax: 82-42-350-5342)
[1]jove@kaist.ac.kr
[2]matssam@kaist.ac.kr
[3]jjlee@ee.kaist.ac.kr

Abstract: Robots have been playing an important roles on our life. In various field such as entertainment, military, space and medical fields, the precise control is required according to the increase of the importance on robot. In robot manipulator position control problem, modeling robot inverse dynamics is important because it can allow accurate robot control using computed torque control and PD control with computed feedback. However, modeling rigid-body inverse dynamics is not simple and not accurate in some case, because of unmodeled nonlinearities such as hydraulic cable dynamics, complex friction or actuator dynamics. Instead of rigid-body dynamics, nonparametric regression such as Locally Weighted Projection Regression (LWPR), Gaussian Process Regression (GPR) is proposed as alternative. Locally Weighted Projection Regression is fast, but it is difficult to tune because of many user-parameters. Gaussian Process Regression has high accuracy but low computation speed. In other word, high complexity of computation is drawback of Gaussian process regression. In Gaussian Process Regression, the inverse of Gram matrix is a significant problem and it dominates the computation time. To improve the low computation speed, there are many methods such as approximation method and Local Gaussian Process Regression (LGPR). In approximation method, the approximation of inverse of Gram matrix is proposed and in local Gaussian Process Regression, the training data is divided into local training data using Gaussian kernel. It generates M local models. After partitioned the training data, the local model is trained. When test data is given, each local model predicts the local prediction. The total prediction value is weighted average of M local prediction values. The weight is a similarity measure and it can be calculated by Gaussian kernel. Local Gaussian Process Regression In this paper, Modified Local Gaussian Process Regression (MLGPR) is suggested for improving accuracy and computation time of Local Gaussian Process Regression. Modified Local Gaussian Process Regression is used adaptive method for partitioning the training data. Modified Local Gaussian Process Regression uses multiple model generation threshold w_{gen} values depending on the local target variance. Proposed method is demonstrated by 2-dimension regression example and learning inverse dynamics of SARCOS arm. The result of simulations will be compared with other method such as Gaussian Process Regression, Local Gaussian Process Regression. As a result, the accuracy of Modified Local Gaussian Process Regression is improved and computation cost is reduced. The result represent Modified Local Gaussian Process Regression has low computation cost as compared with Local Gaussian Process Regression.

Keywords: inverse dynamics, machine learning, Gaussian Process Regression, Local Gaussian Process Regression

1 INTRODUCTION

Robots have been studied and applied in various field such as entertainment, military, space and medical fields. The precise control is required according to the increase of the importance on robot. For more precise control, more precise models of inverse dynamics is needed. In other word, precise models of inverse dynamics allow the effective control [1] [2]. However, finding precise models of inverse dynamics is difficult in some cases because of unmodeled nonlinearity. This modeling error comes from hydraulic cable dynamics, complex friction or actuator dynamics [3]. To solve this problem, the alternative method which is using nonparametric regression method is proposed. The nonparametric regression method is a kind of supervised learning, and it is more flexible than parametric regression. This method predicts inverse dynamics model from pre-measured data [6]. The most common used nonparametric regression methods are Locally Weighted Projection Regression (LWPR) [4], Gaussian Process Regression (GPR) [5] and Support Vector Regression (SVR). Although LWPR is fast ,accurate and incremental learning is possible, it is hard to tune due to many user parameters. GPR has high accuracy but its computation cost is increased when the data is large. To overcome the high computation cost, Local Gaussian Process Regression (LGPR) [3][7][8][9] is proposed. LGPR is an algo-

rithm which combines the accuracy of GPR method on one region with the good capability of LWPR to split the input domain into regions [6]. The more detail regression methods are well organized and summarized in [6].

In this paper, Modified Local Gaussian Process Regression (MLGPR) is suggested for improving accuracy and computation time of LGPR. In section 2, Gaussian Process Regression and Local Gaussian Process Regression are presented in detail. I propose Modified Local Gaussian Process Regression (MLGPR) in section 3. MLGPR is used adaptive method for partitioning the training data. In section 4, the simulation of modified local Gaussian Process Regression is demonstrated. The result of simulation will be compared with other method such as Gaussian Process Regression, Local Gaussian Process Regression. In section 5, we suggest conclusion.

2 LOCAL GAUSSIAN PROCESS REGRESSION

2.1 Gaussian Process Regression

A Gaussian Process is completely specified its mean function and covariance function. The covariance function $k(\mathbf{x}, \mathbf{x}')$ can be any positive semi-definite functions. The square exponential(SE) covariance function is frequently used and are given by:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T \mathbf{M}(\mathbf{x_p} - \mathbf{x_q})) \qquad (1)$$

where σ_f^2 is data variance and $\mathbf{M}$ is the symmetric matrix and it represent distance measure. $\theta = [\sigma_f^2, \{\mathbf{M}\}]^{\mathbf{T}}$ is the hyperparameters of a Gaussian Process.

Consider a set of *n* training data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ and $\mathbf{y} = [y_1, y_2, ..., y_n]$. Using the training data $(\mathbf{X}, \mathbf{y})$ and a new data point $\mathbf{x}_*$, we want to know the predictive distribution of the corresponding $\mathbf{y}_*$. The joint distribution of the training outputs, $\mathbf{y}$ and the test outputs, $\mathbf{y}_*$ is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & K(\mathbf{X}, \mathbf{x}_*) \\ K(\mathbf{x}_*, \mathbf{X}) & K(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right) \qquad (2)$$

From the joint distribution, the predictive distribution is

$$\begin{aligned} &\mathbf{y}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim N(\bar{\mathbf{y}}_*, \mathrm{cov}(\bar{\mathbf{y}}_*)) \\ &\bar{\mathbf{y}}_* = K(\mathbf{x}_*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y} = {k_*}^T \boldsymbol{\alpha} \\ &\mathrm{cov}(\bar{\mathbf{y}}_*) = K(\mathbf{x}_*, \mathbf{x}_*) - K(\mathbf{x}_*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}K(\mathbf{X}, \mathbf{x}_*) \end{aligned} \qquad (3)$$

The computation cost for predictition is dominated by inverse of K^{-1}. It is about $O(n^3)$, where n is the number of training data. For large problem, computation cost increases rapidly. This is main weakness in Gaussian Process Regression. For solving this problem, Local Gaussian Process Regression [3] is suggested.

2.2 Local Gaussian Process Regression

A Local Gaussian Process Regression(LGPR) proposed for reducing computation cost of GPR is inspired by local weighted regression and Gaussian Process Regression. The procedure of LGPR is as follows: Firstly, the training data is partitioned into M local training data. here, the weight w_i which is similarity measure can be used for partitioning the training data. After local Gaussian Process model is partitioned, for each test data $\mathbf{x}_*$ the prediction of local model $\bar{y}_i(\mathbf{x}_*)$ can be calculated by using eq(3). Finally, the prediction of entire model $\hat{y}(\mathbf{x}_*)$ is given by weighted mean of all local model predictions.

$$\hat{y}(\mathbf{x}_*) = \frac{\sum_{i=1}^{M} w_i(\mathbf{x}, \mathbf{c_i})\bar{y}_i(\mathbf{x}_*)}{\sum_{i=1}^{M} w_i(\mathbf{x}, \mathbf{c_i})} \qquad (4)$$

where, i is the number of local models. The partitioning algorithm of LGPR is represented in Algorithm 1 and the prediction algorithm of LGPR is represented Algorithm 2. The method for updating Cholesky matrix $\mathbf{L}$ and prediction vector $\boldsymbol{\alpha}$ is presented in [3].

Using LGP regression the computation cost can be reduced from $O(n^3)$ to $O(n_1^3 + n_2^3 + ... + n_k^3)$ where k is the number of local model, $n_1 + n_2 + ... + n_k = n$.

3 MODIFIED LOCAL GAUSSIAN PROCESS REGRESSION

3.1 Issue of Local Gaussian Process Regression

In the preceding sections the computation cost which is the problem of GPR can be reduced by using local weighted regression. But now the problem of LGPR is how to choose the number of local model, in other word, how to divide the training data. According to determination of local training data, the computation cost and accuracy of LGPR can be changed. This represents the importance for dividing of local model. In LGPR algorithm, the training data is partitioned by using similarity measure w, and model generation threshold value w_{gen}. The w_{gen} is open parameter and there's no rule to choose w_{gen}. If w_{gen} is too large, many local model will be generated and the number of training data in each local model is too small. Although it takes small computation cost, the local model with small data makes a ill distribution, i.e. accuracy of LGPR is decreased. If w_{gen} is too small, small local model with large training data is generated. Accuracy of LGPR will be increased but the computation cost will be high. LGPR loses the advantage of computation cost.

3.2 Adaptive method for partitioning of Training Data

In LGPR, every local model has only one fixed model generation threshold w_{gen}. Because the accuracy and computa-

Algorithm 1 Partitioning the training data with incremental model learning(Modified)

Input: new data point $\{\mathbf{x}_{new}, y_{new}\}$
for $i = 1$ to number of local models **do**
  Compute proximity to the ith local model:
    $w_i = k(\mathbf{x}_{new}, \mathbf{c}_i)$
end for
Take the nearest local model: $v(i) = \max_i w_i$
if $v(i) < w_{gen}(i)$ **then**
  Insert $\{\mathbf{x}_{new}, y_{new}\}$ into the nearest local model:
    $\mathbf{X}_{new} = [\mathbf{X}_i, \mathbf{x}_{new}]$, $\mathbf{y}_{new} = [\mathbf{y}_i, y_{new}]$
  Update the corresponding center: $\mathbf{c}_{new} = mean(\mathbf{X}_{new})$
  Update the Cholesky matrix, $\mathbf{L}_{new}$ and the prediction vetor, $\boldsymbol{\alpha}_{new}$ of local model
  if the maximum number of data point is reached **then**
    delete another point randomly
  end if
  Compute $\boldsymbol{\alpha}_{new}$ by back-substitution
  Update w_{gen} {This is the new part of MLGP}
  if $\mathrm{Var}(\mathbf{y}_{new}) > \mathrm{E}(\mathrm{Var}(\mathbf{y}))$ **then**
    $w_{gen}(i) = w_{gen}(i) - \delta$
  else
    $w_{gen}(i) = w_{gen}(i) + \delta$
  end if
else
  Create new model:
    $\mathbf{c}_{i+1} = \mathbf{x}_{new}$, $\mathbf{X}_{i+1} = [\mathbf{x}_{new}]$, $\mathbf{y}_{i+1} = [y_{new}]$
  Initialize of new Cholesky matrix $\mathbf{L}$ and new prediction vector $\boldsymbol{\alpha}$
end if

Algorithm 2 Prediction for a query point

Input:query data point $\mathbf{x}_*$, M
Determine M local models closest to $\mathbf{x}_*$
for $i = 1$ to M **do**
  Compute proximity to the ith local model:
    $w_i = k(\mathbf{x}_*, \mathbf{c}_i)$
  Compute local prediction using kth local model:
  $\bar{y}_i(\mathbf{x}_*) = \mathbf{k}(\mathbf{X_i}, \mathbf{x}_*)^T \boldsymbol{\alpha}_i$
end for
Compute weighted prediction using M local models:
$\hat{y}(\mathbf{x}_*) = \frac{\sum_{i=1}^{M} w_i \bar{y}_i(\mathbf{x}_*)}{\sum_{i=1}^{M} w_i}$

tion cost of LGPR are changed depending on w_{gen}, using same w_{gen} on all local model is inefficient. When a local target variance $\mathrm{Var}(\mathbf{y}_{new})$ is large, the result of LGPR can't predict the original distribution. To use small w_{gen} for improving accuracy of LGPR makes the number of local training data large and makes computation cost high. On the other hand, when a local target variance $\mathrm{Var}(\mathbf{y}_{new})$ is small, the result of LGPR is good, even use of large w_{gen}. So it is required change of w_{gen} depending on local target variance $\mathrm{Var}(\mathbf{y}_{new})$ and use different w_{gen} for local models. Therefore it makes a sense to set large w_{gen} to reduce computation cost, when local target variance is small. When local target variance is large, small w_{gen} is used for improving accuracy of prediction. This is a key idea of MLGPR. Now every model has own w_{gen} values and it can be changed according to local target variance. If the w_{gen} of ith local model is bigger than mean of local target variance $\mathrm{E}(\mathrm{Var}(\mathbf{y_i}))$, w_{gen} is decreased as much as δ. It means that accuracy of LGPR is increased using more training data, when the target variance is high. If the w_{gen} of ith local model smaller than mean of local target variance $\mathrm{E}(\mathrm{Var}(\mathbf{y_i}))$, w_{gen} is increased as much as δ. It means that computation cost of LGPR is decreased by reducing the number of training data, when the target variance is low. Modified method for partitioning training data is included in Algorithm1.

3.3 Accuracy and Computation Cost of MLGP

Table.1 shows a comparison of computation cost GPR, LGPR and MLGPR. In GPR, computation cost for calculation of K^{-1} is about $O(n^3)$. In LGPR, the computation cost can be reduced from $O(n^3)$ to $O(n_1^3 + n_2^3 + ... + n_M^3)$ due to using a local prediction, where n is the number of training data, M is the number of local model and n_i the number of local training data. In MLGP, the computation cost for calculation of K^{-1} is $O(n_{m1}^3 + n_{m2}^3 + + n_{mM}^3)$ where n_{mi} the number of local training data , mM is the number of MLGPR local model. The computation cost for prediction is $O(n)$ in GPR, $O(n_i M)$ in LGPR and $O(n_{mi} M)$ in MLGPR. Here, n_{mi} is bigger than n_i, when the variance of ith model is bigger than variance of target. Otherwise n_{mi} is smaller than n_i. Thus the computation cost of MLGPR is usually lower than one of LGPR.

Table 1: The table of computation cost

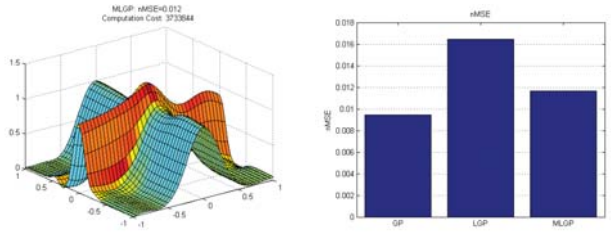
Method	For calculation of K^{-1}	For prediction
GPR	$O(n^3)$	$O(n)$
LGPR	$O(n_1^3 + n_2^3 + ... + n_M^3)$	$O(n_i M)$
MLGPR	$O(n_{m1}^3 + n_{m2}^3 + + n_{mM}^3)$	$O(n_{mi} M)$

4 SIMULATION

In this section, simulation of Modified Local Gaussian Process Regression is demonstrated by 2-

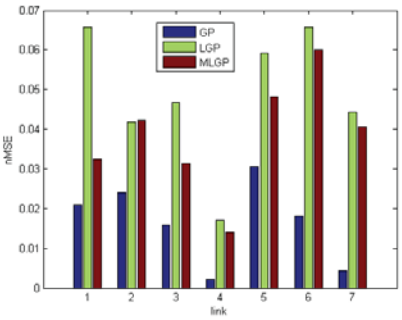
dimension regression example and learning inverse dynamics of SARCOS arm. 500 noisy training data comes from the two dimensional function[4]: $y = \max\left[\exp(-10{x_1}^2), \exp(-50{x_2}^2), 125\exp(-5({x_1}^2 + {x_2}^2))\right] + N(0, 005)$. Fig.1 shows the result of regression using MLGPR. Its nMSE is 0.012 and it is lower than the LPGR one.

Fig.2 shows the result for inverse dynamics learning of SARCOS arm. The data set consists of 45,000 training data and 5,000 test data[1]. Fig.2(a) shows the nMSE of each link of robot. Fig.2(b) is Computation cost of MLGPR and LGPR. nMSE of MLGPR is lower than one of LGPR,

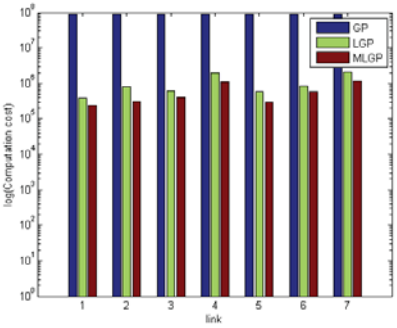


(a) Result of regression using MLGPR (b) nMSE of GPR, LGPR, MLGPR

Fig. 1: 2-dimension regression example



(a) nMSE of each link of robot



(b) Computation cost of MLGPR and LGPR

Fig. 2: Learning inverse dynamics of SARCOS arm

5 CONCLUSION

Modified Local Gaussian Process Regression(MLGPR) uses multiple model generation threshold w_{gen} values depending on the local target variance. It means each local model has own w_{gen}. If the w_{gen} of ith local model is bigger than mean of local target variance , w_{gen} is decreased for improving the accuracy. Otherwise, w_{gen} is increased for reducing computation cost. MLGPR is compared with GPR and LGPR. As a result, the accuracy of MLGP is improved and computation cost is reduced in simulation results.

REFERENCES

[1] R. M. Murray(2006), A Mathematical Introduction to ROBOTIC MANIPULATION. CRC Press, pp.191-195

[2] Rafael Kelly, Ricardo Salgado (1994), PD control with Computed Feedforward of Robot manipulators:A Design Procedure, IEEE Transactions on Robotics and Automation, voulum.10 pp.566-571.

[3] D. Nguyen-Tuong, M. Seeger, J. Peters (2009), Model Learning with Local Gaussian Process Regression. Advanced Robotics, Volume 23, Number 15, pp. 2015-2034(20)

[4] S. Vijayakumar, S. Schaal (2000), Locally Weighted Projection Regression: An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space, Proceeding of Seventeenth International Conference on Machine Learning, pp.1079-1086

[5] C. E. Rasmussen, C. K. I. Williams (2006), Gaussian Processes for Machine Learning. MIT Press

[6] O. Sigaud, C. Salaun, V. Padois (2011), On-line regression algorithms for learning mechanical models of robots: A survey, Robotics and Autonomous Systems 59, pp.1115-1129

[7] D. Nguyen-Tuong, J. Peters, M. Seeger (2008) Local Gaussian Process Regression for Real Time Online Model Learning and Control. In Advances in Neural Information Processing Systems, pp. 1193-1200

[8] D. Nguyen-Tuong, J. Peters (2008) Local Gaussian process regression for real-time model-based robot control. Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on , pp. 380 - 385

[9] D. Nguyen-Tuong, M. Seeger, J. Peters (2008), Computed Torque Control with Nonparametric Regression Models. American Control Conference, pp. 212 - 217

[1]This dataset is available at 'http://www.gaussianprocess.org/gpml/data'