Neurocomputing 74 (2011) 1859–1867

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Incremental online sparsification for model learning in real-time robot control

Duy Nguyen-Tuong *, Jan Peters

Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany

A R T I C L E I N F O

Available online 24 February 2011

Keywords:
Sparse data
Machine learning
Real-time online model learning
Inverse dynamics
Robot control

A B S T R A C T

For many applications such as compliant, accurate robot tracking control, dynamics models learned from data can help to achieve both compliant control performance as well as high tracking quality. Online learning of these dynamics models allows the robot controller to adapt itself to changes in the dynamics (e.g., due to time-variant nonlinearities or unforeseen loads). However, online learning in real-time applications – as required in control – cannot be realized by straightforward usage of off-the-shelf machine learning methods such as Gaussian process regression or support vector regression. In this paper, we propose a framework for online, incremental sparsification with a fixed budget designed for fast real-time model learning. The proposed approach employs a sparsification method based on an independence measure. In combination with an incremental learning approach such as incremental Gaussian process regression, we obtain a model approximation method which is applicable in real-time online learning. It exhibits competitive learning accuracy when compared with standard regression techniques. Implementation on a real Barrett WAM robot demonstrates the applicability of the approach in real-time online model learning for real world systems.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, model learning has become an important tool in a variety of robotics applications such as terrain modeling [5], sensor evaluation [11], model-based control [8,13] and many others. The reason for this rising interest is that accurate analytical models are often hard to obtain due to the increasing complexity of modern robot systems. Model learning can be a useful alternative as the model is estimated directly from measured data. Unknown nonlinearities are directly taken into account, while they are neglected either by the standard physics-based modeling techniques or approximated by hand-crafted models. Nevertheless, the excessive computational complexity of the more advanced regression techniques still hinders their widespread application in robotics. Models that have been learned offline can only approximate the model correctly in the area of the state space that is covered by the sampled data, and often do not generalize beyond that region. Thus, in order to cope with unknown state space regions online model learning is essential. Furthermore, it also allows the adaptation of the model to changes in the robot dynamics, for example, due to unforeseen loads or time-variant nonlinearities such as backlash, complex friction and stiction.

However, real-time online model learning poses three major challenges: first, the learning and prediction processes need to be sufficiently fast; second, the learning system needs to deal with large amounts of data; third, the data arrives as a continuous stream and, thus, the model has to be continuously adapted to new training examples. A few approaches for real-time model learning for robotics have been introduced in the machine learning literature, such as locally weighted projection regression [18] or local Gaussian process regression [10]. In these methods, the state space is partitioned into local regions for which local models are approximated and, thus, these methods will not make use of the global behavior of the embedded functions. As the proper allocation of relevant areas of the state space is essential, appropriate online clustering becomes a central problem for these approaches. For high dimensional data, partitioning of the state space is well-known to be a difficult issue [18,10]. To circumvent this online clustering, an alternative is to find a sparse representation of the *known* data [12,15,6]. For robot applications, however, it requires finding an incremental sparsification method applicable in real-time online learning—a major challenge tackled in this paper.

Inspired by the work in [15,3], we propose a method for incremental, online sparsification which can be integrated into several existing online regression methods, making them applicable

* Corresponding author.
E-mail address: d.nguyentuong@googlemail.com (D. Nguyen-Tuong).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.06.033

for model learning in real-time. The suggested sparsification is performed using a test of linear independence to select a sparse subset of the training data points, often called the *dictionary*. The resulting framework allows us to derive criteria for incremental insertion and deletion of dictionary data points, which are two essential operations in such an online learning scenario. For evaluation, we combine our sparsification framework with an incremental approach for Gaussian process regression (GPR) as described in [10]. The resulting algorithm is applied in online learning of the inverse dynamics model for robot control [17,9].

The rest of the paper will be organized as follows: first, we present our sparsification approach which enables real-time online model learning. In Section 3, the efficiency of the proposed approach in combination with an incremental GPR update is demonstrated by an offline comparison of learning inverse dynamics models with well-established regression methods, i.e., ν-support vector regression [16], standard Gaussian process regression [12], locally weighted projection regression [18] and local Gaussian process regression [10]. Finally, the capability of incremental GPR using online sparsification for real-time model learning will be illustrated by online approximation of inverse dynamics models for real-time tracking control on a Barrett WAM. A conclusion will be given in Section 4.

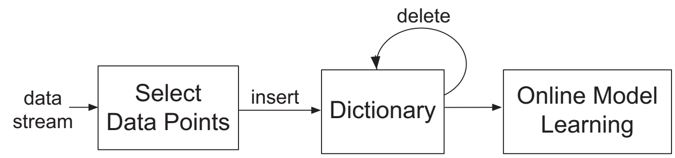

Fig. 1. Sparsification for online model learning.

2. Incremental sparsification for real-time online model learning

In this section, we introduce a sparsification method which – in combination with an incremental kernel regression – enables fast, real-time model learning. The proposed sparsification approach is formulated within the framework of kernel methods. Therefore, we first present the basic intuition behind the kernel methods and motivate the need of online sparsification. Subsequently, we describe the proposed sparsification method in details.

2.1. Model learning with kernel methods

By learning a model, we want to approximate a mapping from the input set $\mathbf{X}$ to the target set $\mathbf{Y}$. Given n training data points $\{\mathbf{x}_i, y_i\}_{i=1}^n$, we intend to discover the latent function $f(\mathbf{x}_i)$ which transforms the input vector $\mathbf{x}_i$ into a target value y_i given by the model $y_i = f(\mathbf{x}_i) + \varepsilon_i$, where ε_i represents a noise term. In general, it is assumed that $f(\mathbf{x})$ can be parametrized as $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^T\mathbf{w}$, where $\boldsymbol{\phi}$ is a feature vector mapping the input $\mathbf{x}$ into some high dimensional space and $\mathbf{w}$ is the corresponding weight vector [15,12]. The weight $\mathbf{w}$ can be represented as a linear combination of the input vectors in the feature space, i.e., $\mathbf{w} = \sum_{i=1}^n \alpha_i \boldsymbol{\phi}(\mathbf{x}_i)$ with α_i denoting the linear coefficients. Using these results, the prediction $\hat{y}$ of a query point $\mathbf{x}$ can be given as

$$\hat{y} = \hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \rangle, = \sum_{i=1}^{n} \alpha_i \mathrm{k}(\mathbf{x}_i, \mathbf{x}). \quad (1)$$

As indicated by Eq. (1), the inner product of feature vectors $\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \rangle$ can be represented as a kernel value $\mathrm{k}(\mathbf{x}_i, \mathbf{x})$ [15]. Thus, instead of finding a feature vector, only appropriate kernels need to be determined. An often used kernel is, for example, the Gaussian kernel

$$\mathrm{k}(\mathbf{x}_p, \mathbf{x}_q) = \exp(-\tfrac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T \mathbf{W} (\mathbf{x}_p - \mathbf{x}_q)), \quad (2)$$

where $\mathbf{W}$ denotes the kernel widths [15,12]. For employing kernel methods in model learning, however, one needs to estimate the linear coefficients α_i using training examples. State-of-the-art kernel methods, such as kernel ridge regression, support vector regression (SVR) or Gaussian process regression (GPR), differ in the approaches for estimating α_i [15,12,4]. While support vector regression estimates the linear coefficients by optimization using training data [15], kernel ridge regression and Gaussian process regression basically solve the problem by matrix inversion [4,12] (see the Appendix for a short review of GPR). The complexity of model learning with kernel methods, i.e., the estimation of α_i, depends largely on the number of training examples. In GPR, for example, the computational complexity is $\mathcal{O}(n^3)$, if the model is obtained in batch learning.

However, online model learning requires incremental updates, e.g., incremental estimation of α_i, as the data arrives sequentially. There have been many attempts to develop incremental, online algorithms for kernel methods, such as incremental SVM [2], sequential SVR [19], recursive kernel learning with NARX form [6] or the kernel recursive least-squares algorithm [3], for an overview see [15]. However, most incremental kernel methods do not scale to online learning in real-time, e.g., for online learning with model updates at 50 Hz or faster. The main reason is that they are neither sparse [2,19], as they use the complete data set for model training, nor do they restrict the size of the sparse set [3]. To overcome these shortcomings, we propose the setup illustrated in Fig. 1.

To ensure real-time constraints, we train the model using a dictionary with a fixed budget. The budget of the dictionary, i.e., the sparse set, needs to be determined from the intended learning speed and available computational power. To efficiently make use of the stream of continuously arriving data, we select only the most informative data points for the dictionary. If the budget limit is reached, dictionary points will need to be deleted. Finally, for the model training using dictionary data, most incremental kernel regression methods can be employed, e.g., incremental GPR as described in [10], sequential SVR [19] or incremental SVM [2].

Inspired by the work in [3,14], we use a linear independence measure to select the most informative points given the current dictionary. Based on this measure, we derive criteria to remove data points from the dictionary, if a given limit is reached. The following sections describe the proposed approach in detail.

2.2. Sparsification using linear independence test

The main idea in our sparsification approach is that we intend to cover the relevant state space at the best, given a limited number of dictionary points. At any point in time, our algorithm maintains a dictionary $\mathcal{D} = \{\boldsymbol{d}_i\}_{i=1}^m$ where m denotes the current number of dictionary points $\boldsymbol{d}_i$. The choice of the dictionary element $\boldsymbol{d}_i$ might be crucial for particular application and will be discussed in Section 2.5. To test whether a new point $\boldsymbol{d}_{m+1}$ should be inserted into the dictionary, we need to ensure that it cannot be approximated in the feature space spanned by the current dictionary set. This test can be performed using a measure δ defined as

$$\delta = \left\| \sum_{i=1}^{m} \mathrm{a}_i \boldsymbol{\phi}(\boldsymbol{d}_i) - \boldsymbol{\phi}(\boldsymbol{d}_{m+1}) \right\|^2, \quad (3)$$

(see, e.g., [14,15] for background information), where a_i denote the coefficients of linear dependence. Eq. (3) can be understood as

Download English Version:

https://daneshyari.com/en/article/410282

Download Persian Version:

https://daneshyari.com/article/410282

Daneshyari.com