Neural Networks 78 (2016) 75–87

Contents lists available at ScienceDirect

Neural Networks

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

2016 Special Issue

Smart sampling and incremental function learning for very large high dimensional data

Diego G. Loyola R*, Mattia Pedergnana, Sebastián Gimeno García

German Aerospace Center (DLR), Oberpfaffenhofen, 82234 Wessling, Germany

ARTICLE INFO

Article history:
Available online 28 September 2015

Keywords:
High dimensional function approximation
Sampling discrepancy
Design of experiments
Probably approximately correct computation
Function learning
Neural networks

ABSTRACT

Very large high dimensional data are common nowadays and they impose new challenges to data-driven and data-intensive algorithms. Computational Intelligence techniques have the potential to provide powerful tools for addressing these challenges, but the current literature focuses mainly on handling scalability issues related to data volume in terms of sample size for classification tasks.

This work presents a systematic and comprehensive approach for optimally handling regression tasks with very large high dimensional data. The proposed approach is based on smart sampling techniques for minimizing the number of samples to be generated by using an iterative approach that creates new sample sets until the input and output space of the function to be approximated are optimally covered. Incremental function learning takes place in each sampling iteration, the new samples are used to fine tune the regression results of the function learning algorithm. The accuracy and confidence levels of the resulting approximation function are assessed using the probably approximately correct computation framework.

The smart sampling and incremental function learning techniques can be easily used in practical applications and scale well in the case of extremely large data. The feasibility and good results of the proposed techniques are demonstrated using benchmark functions as well as functions from real-world problems.

1. Introduction

Computer-based simulations of tremendously complex mathematical systems describing multifaceted physical, chemical, dynamical and engineering models are usually associated with very expensive costs in terms of processing time and storage. Complex mathematical models are present in a wide variety of scientific areas such as the simulation of atmospheric processes in numerical weather prediction (Han & Pan, 2011; Hsieh & Tang, 1998; Lynch, 2006; Morcrette, 1991), climate modeling (Flato et al., 2013), (Gordon et al., 2000), chemical transport (Grell et al., 2005), (Menut et al., 2013), radiative transfer (Gimeno García, Trautmann, & Venema, 2012) and large eddy simulations (Sagaut, 2006). Other scientific disciplines such as genetics, aerodynamics, or statistical mechanics also make use of highly complex models. The input space of these models can be of high dimensionality with hundreds or more components. The usage of more realistic models usually introduces new dimensions leading to an exponential increase in volume, i.e. "Big Data" (Hilbert & López, 2011; Lynch, 2008).

The speeding-up of such models is a crucial problem in practical applications like weather forecasting, remote sensing, and climate modeling among others. These complex models can be approximated using for example neural networks (Haupt, Pasini, & Marzban, 2009; Loyola, 2006; Qazi & Linshu, 2006), support vector machines (Tripathi, Srinivas, & Nanjundiah, 2006), kernel smoothing models (Cervellera & Macciò, 2013), and the resulting computational intelligence systems are being deployed in operational environments, see for example Krasnopolsky and Chevallier (2003) and Loyola (2006).

Related work found in the literature usually addresses function approximations from complex mathematical models in the general framework of statistical machine learning where it is assumed that the training samples are created independently according to an unknown probability density function (Krasnopolsky & Schiller, 2007). However, having a mathematical model to parameterize, we can freely choose the sample points that better characterize the input and output spaces of the model. The selection of samples to be used in function approximation problems is less explored in the literature, sometimes it is called design of experiments (Sacks, Welch, Mitchell, & Wynn, 1989) or active learning (Enăchescu, 2013).

In this study we develop a general function approximation framework in which the choice of the samples describing the

* Corresponding author. Tel.: +49 8153 28 1367; fax: +49 8153 28 1446.
E-mail address: Diego.Loyola@dlr.de (D.G. Loyola R).

http://dx.doi.org/10.1016/j.neunet.2015.09.001

function is an integral part of the learning problem. The goal is to find a regression model that accurately approximates a function $f : X \rightarrow Y$ with input $X \subset \mathbb{R}^n$ and output $Y \subset \mathbb{R}^m$ from a set of training samples $T = \{x_i, y_i\}$ for $i = 1$ to s, subject to the constraint of minimizing the number of calls to the target function f, i.e. minimizing the number of samples s. The dimensionality of the data needed to solve this problem is characterized by the number of samples s and the dimensionality of the mapping represented by f with n inputs and m outputs.

On one side, the complexity of a function does not necessarily increase for a high dimensional input space; moreover the limiting factor for an accurate function approximation is the intrinsic complexity of the target function and not the dimension of the input data space (Kolmogorov, 1957). Accuracy of approximation can be achieved in high dimensional cases for target functions with lower complexity (Gnecco, Kůrková, & Sanguineti, 2011). On the other side, systematically sampling the input space with k values per dimension will require in total $s = k^n$ calls to f, therefore the number of samples s for mappings with high dimensional input size grows exponentially with the input dimension n.

These difficulties are usually referred as "the curse of dimensionality" problem (Vapnik, 2006) and it has severe consequences not only for the time needed to create the training dataset but also for the algorithms needed to solve function approximation problems that must deal with very large number of samples of high dimensionality, i.e. "Big Data".

The data volume component of Big Data is a hot topic in machine learning, see for example Jin and Hammer (2014) and O'Leary (2013) and the references therein. But the main focus on the current literature is the sample size and not the sample dimension problem. Only a few studies address the big dimension problem such as for example theoretical investigations of computational models efficiency in high dimensional context (Kainen, Kůrková, & Sanguineti, 2012) and for classification tasks wherein the explosion of features brings about new challenges to computational intelligence (Yiteng, Yew-Soon, & Tsang, 2014).

In this article we focus on the under-explored topic of big dimensionality for regression tasks. The paper is organized as follows: Section 2 gives an overview of data sampling methods and discrepancy measurements. Section 3 shows a comparison of sampling methods and evaluates their performance for high dimension input problems. Section 4 presents the smart sampling and incremental function learning (SSIFL) algorithm that optimally solves this kind of function approximation problems and Section 5 shows the results of applying SSIFL to a number of benchmark and real-world functions. Finally, the conclusions are given in Section 6.

2. Data sampling methods

Generally speaking a good sampling method should create training data that accurately represents the underlying function preserving the statistical characteristics of the complete dataset. This section presents first a survey of sampling methods grouped in four categories that can be applied to generating high dimensional sample data.

2.1. Stochastic methods

Pseudo-Random number generators (PR) (Marsaglia & Tsang, 2000; Matsumoto & Nishimura, 1998) are commonly used to create samples in a given range. PR methods are fast and simple to use, but they are not distributed uniformly enough especially for the cases of low number of sampling points and/or large number of dimensions. PR sampling is sometimes called "pure" or "plain" Monte Carlo (Swiler, Slepoy, & Giunta, 2006).

2.2. Deterministic methods

Uniform populations can be created using quasi-random or sub-random sequences that cover the input space quickly and evenly, the uniformity and coverage improves continually as more data points are added to the sequence. Deterministic methods can be divided into two subcategories (Kazimipour, Li, & Qin, 2013) described in the next subsections.

2.2.1. Low discrepancy methods

Low Discrepancy methods have the support of theoretical upper-bounds on discrepancy. Halton, Sobol, Niederreiter, Hammersley, and Faure are well known sequences from this category.

In this work we use Halton sequences, shortened as (HA), which are constructed according to a deterministic algorithm that uses prime numbers as bases for each dimension (Halton, 1960). Halton sequences work well in low dimensionality, but they lose uniformity in high dimensions. Workaround solutions such as using big prime numbers, setting leap values, scrambling and shuffling improve the sampling uniformity in such cases. The HA method is computationally efficient even for very high dimensional spaces.

2.2.2. Experimental design methods

Experimental Design methods are commonly used for initializing the population of evolutionary algorithms in order to accelerate convergence speed and improve stability. The most representative methods in this category are:

- Uniform Design: a space-filling method based on prime numbers that generates points uniformly scattered on the input domain (Peng, Wang, Dai, & Cao, 2012).
- Orthogonal Design: based on Latin squares for creating orthogonal arrays (Leung & Wang, 2001).

2.3. Geometrical methods

2.3.1. Uniform grid

Uniform grid (UG) is the simplest sampling method in which the samples are created using node points at fixed intervals uniformly distributed for every dimension. UG is commonly used for creating look-up tables to accelerate complex model computations (Perkins et al., 2012; Richter, Heege, Kiselev, & Schläpfer, 2014).

Using a uniform grid is convenient for storing the look-up tables in multi-dimensional arrays. As we will show in Sections 3.2 and 3.3, the coverage of the input space using UG is very poor for low number of sampling points and high dimensions.

2.3.2. Latin hypercube

Latin hypercube sampling (McKay, Beckman, & Conover, 1979) partitions the input space into bins of equal probability and distributes the samples in such a way that only one sample is located in each axis-aligned hyperplane. This method and variations like nearly-orthogonal Latin hypercube (Cioppa, 2002) and Distributed Hypercube and Improved Hypercube sampling (Beachkofski & Grandhi, 2002) are very popular in computer model problems. A Particle Swarm Optimization algorithm for solving large-scale Latin hypercube design problems was proposed recently (Aziza & Tayarani-N., 2014).

Latin hypercube sampling is useful when the underlying function has a low order distribution but this method produces clustering of sampling points at high dimensions.