Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

KCMAC-BYY: Kernel CMAC using Bayesian Ying–Yang learning

K. Tian [a], B. Guo [b,*], G. Liu [c], I. Mitchell [d], D. Cheng [b], W. Zhao [b]

[a] School of Automation, Harbin Engineering University, Harbin, China
[b] School of Electrical Engineering and Automation, Harbin Institute of Technology, Harbin China
[c] School of Computer Engineering, Nanyang Technological University, Singapore
[d] School of Engineering and Information Sciences, Middlesex University, London NW4 4BT, UK

A R T I C L E I N F O

Article history:
Received 23 November 2011
Received in revised form
7 May 2012
Accepted 20 June 2012
Communicated by L. Xu
Available online 6 September 2012

Keywords:
Bayesian Ying–Yang learning
CMAC
Kernel machine

A B S T R A C T

The Cerebellar Model Articulation Controller (CMAC) possesses attractive properties of fast learning and simple computation. In application, the size of its association vector is always reduced to economize the memory requirement, greatly constraining its modeling capability. The kernel CMAC (KMAC), which provides an interpretation for the traditional CMAC from the kernel viewpoint, not only strengthens the modeling capability without increasing its complexity, but reinforces its generalization with the help of a regularization term. However, the KCMAC suffers from the problem of selecting its hyperparameter. In this paper, the Bayesian Ying–Yang (BYY) learning theory is incorporated into KCMAC, referred to as KCMAC-BYY, to optimize the hyperparameter. The proposed KCMAC-BYY achieves the systematic tuning of the hyperparameter, further improving the performance in modeling and generalization. The experimental results on some benchmark datasets show the prior performance of the proposed KCMAC-BYY to the existing representative techniques.

1. Introduction

The Cerebellar Model Articulation Controller (CMAC) [1,2] is a type of neural network based on a model of the mammalian cerebellum. Originally, the CMAC was proposed as a function modeler for robotic controllers by James Albus in 1975, but it has been extensively used in reinforcement learning and also as a classifier. As an associative memory neural network model, the CMAC has some attractive features of fast learning, simple computation, local generalization, and the fact that it can be realized by specialized high-speed hardware [3].

The traditional CMAC applies a table-lookup technique and has local generalization ability that depends on the overlap of the association vectors [1,2]. In this technique, the input space of a CMAC is divided into levels. Each level corresponds to a defined quantization function, which is used to quantize the input space into discrete states. Levels from different dimensions combine to structure a set of overlays, the number of which is a key factor of modeling capability of the CMAC. A specific association vector is obtained from every input point using an "AND" operation in all overlays, which contains binary elements, with 1 indicating excitation and 0 no excitation. This action is also interpreted as the first primary mapping of a CMAC, which projects from the input space into association space. The second primary mapping calculates the output of a CMAC as a scalar product of the association vector and corresponding weight vector. The learning mechanism adopted in a CMAC network is based on error correction. For each training data point, the error between output of a CMAC and the desired response is computed and the weight vector is adjusted accordingly.

Practically, a crucial problem always constrains the application of a CMAC: the memory requirement grows exponentially with respect to the input dimension. In order to reduce the complexity of a CMAC, Albus introduced hash coding into his model [1]. This approach effectively reduces the size of memory, but it can result in collisions of the mapped weights and bring certain adverse impacts to the convergence of learning. Another method of decomposing a multivariate case into a group of lower dimensional ones is also widely used to reduce complexity in CMAC research [4–7]. The authors in [6] introduced a hierarchical architecture CMAC (HCMAC), which is a multilayer network with $\log_2{}^N$ layers. In each layer, a two-dimensional elementary CMAC is constructed using finite support Gaussian basis functions. Similarly, in [7] another multilayer architecture, macro structure CMAC (MS-CMAC), the one-dimensional elementary CMAC is applied. Although all the above mentioned architectures are less complex compared to the traditional CMAC, they are time consuming in the training process. For example, experimental results on the Iris data including 75 patterns demonstrated that the training time required for traditional CMAC is 0.0186 s, while that for HCMAC and S-OHCMAC are 2.604 and 0.115 s, respectively [6].

Interpreting a CMAC as a type of kernel machine, the Kernel Cerebellar Model Articulation Controller (KCMAC) can reduce the complexity of a CMAC and strengthen its modeling capability remarkably [8]. Specifically, the association space in the KCMAC is treated as the feature space in the kernel machine. Additionally, the binary basis functions can be regarded as first-order B-spline functions [9] of fixed positions, so higher-order B-spline kernel functions can be designed to

[*] Corresponding author.
E-mail address: guoben@hit.edu.cn (B. Guo).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.06.028

replace the binary basis functions of the CMAC. Due to the kernel trick, the dimension of the kernel space will not increase as the dimension of input space increases. This means we can build a kernel version of a multivariate CMAC without reducing the length of the associate vector. Thus, given any training data set KCMAC can learn without error that is independent of the dimension of the input data and, in addition, there will be no significant difference between one-dimensional and multi-dimensional cases.

In order to improve the modeling capability of the KCMAC, one hyperparameter is introduced to penalize the miss-regression/classification. Such a hyperparameter must be optimized to adapt a specific problem; however, it is always pre-defined based on empirical knowledge in the original KCMAC. To address this problem, we attempt to optimize the hyperparameter using Bayesian Ying–Yang learning theory.

First proposed in [10] and developed for over a decade, Bayesian Ying–Yang (BYY) learning provides a general framework that accommodates typical learning algorithms from a unified perspective and improved model selection criteria. BYY learning consists of two subcategories. One is featured with Ying–Yang best matching for developing typical learning algorithms, which is considered in this paper and also one major focus of Ref. [10,12],while the other is Ying–Yang best harmony featured with its favorable nature for model selection [11], for which readers are referred to Ref. [22,23]for recent systematic reviews, and especially Fig. A2 of Ref. [22]and Section 4.2 of Ref. [23], for the relations between these two BYY learning subcategories.

In [13,14], BYY has been successfully applied to the fuzzification phase of the CMAC, which is further incorporated with an online expectation–maximization (EM) algorithm to process time series data [24]. However, the weight training phase is separated from the fuzzification phase in all these models. In this paper, the KCMAC is optimized by BYY and hereafter referred to as KCMAC-BYY, in which the training data are regarded as the external observation, while the connection weight vector is the inner representation. Based on these, our proposed KCMAC-BYY achieves the systematic tuning of the hyperparameter, and further improves the performance in modeling capability and stability. The remainder of this paper is organized as follows: Section 2 briefly describes KCMAC; Section 3 introduces a probabilistic interpretation of KCMAC; Section 4 investigates parameter optimization using BYY; Section 5 includes the experiments, results and analysis; and, finally, Section 6 is the conclusion.

2. Overview of the kernel CMAC

In terms of modeling capability, the lower dimensional CMACs always provide much better performance than higher dimensional ones. A detailed analysis of the reason for this phenomenon is given here. In high-dimensional cases, considering that the memory requirement of a CMAC grows exponentially with respect to the input dimension, in implementation we usually have to reduce the size of the association vector by decreasing the number of overlays used. For example, in a 5-D CMAC with three levels in each of the input dimensions, 3^5 potential overlays will be used in learning. In order to save memory, CMAC only uses five overlays, in which the overlay-representing points are in the main diagonal of the input space. However, in lower dimensional cases a full overlay structure is always used and hence the reason why they have better modeling capabilities. To resolve the problem, a kernel version of a CMAC is introduced, eliminating the difference between the lower and higher dimensional CMACs [3,8] and strengthening the modeling capability of the latter remarkably without increasing the model complexity.

In the KCMAC, the association space is treated as the feature space of a kernel machine [9]. Considering that the binary basis functions can be regarded as first-order B-spline functions of fixed positions, the higher-order B-spline kernel functions can be designed artificially to replace the binary basis functions of a CMAC. In the KCMAC a structured mapping function (kernel function) is adopted, replacing the “AND” operation of the traditional CMAC, to project the input data into the feature space (association space). In addition to the traditional binary CMAC, this kernel interpretation can also be used in higher order CMACs [16–18] with higher order basis functions. A CMAC with the *k*th-order B-spline basis function corresponds to a kernel machine with the *2k*th-order B-spline kernels.

The weight vector **w** in the KCMAC is determined by the following constrained optimization using a quadratic loss function [19]:

$$\underset{\mathbf{w},\mathbf{e}}{Min} J(\mathbf{w},\mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^{n} e_i^2 \tag{1}$$

Such that

$$z_i = \mathbf{w}^T \phi(\mathbf{u}_i) + e_i \frac{\gamma}{2} \tag{2}$$

where γ is the miss-regression penalty parameter of the model, e_i is the error of the *i*th input point, and $\Phi(\mathbf{u}_i)=a\,(\mathbf{u}_i)$ corresponds to the mapping function in the kernel machine. Then the Lagrangian is introduced

$$L(\mathbf{w},\mathbf{e},\alpha) = J(\mathbf{w},\mathbf{e}) - \sum_{i=1}^{n} \alpha_i(\mathbf{w}^T\phi(\mathbf{u}_i) + e_i - z_i) \tag{3}$$

where a_i are the lagrange multipliers. These a_i s are determined with the following linear system:

$$\boldsymbol{\alpha} = \left[\mathbf{K} + \frac{\mathbf{1}}{\gamma\mathbf{I}}\right]^{-1}\mathbf{z} \tag{4}$$

where $\mathbf{K}=\mathbf{A}\mathbf{A}^T$ is the kernel matrix, and $\mathbf{I}$ is an $n \times n$ identity matrix. The response of the network is

$$y(\mathbf{u}) = \phi(\mathbf{u})\sum_{i=1}^{n} \alpha_i \phi(\mathbf{u}_i) = \sum_{i=1}^{n} \alpha_i K(\mathbf{u},\mathbf{u}_i) \tag{5}$$

In addition, by adding a regularization term into Eqs. (1) and (2), the KCMAC can be easily extended to a regularized version which has a better generalization capability.

In the KCMAC, the modeling capability can be reinforced greatly without increasing its complexity, but the hyperparameter γ, which is the miss-regression penalty parameter of the model, is introduced into the model. In the original KCMAC, this hyperparameter is usually determined empirically, while different values of γ will result in quite different performances. To guarantee the modeling capability and stability, the BYY learning is embedded into the KCMAC in this paper, as it is able to systematically optimize γ, and a novel KCMAC-BYY is proposed.

3. The kernel CMAC with the BYY learning

In the BYY supervised learning [12–14], there are three primary elements: the inner representation **w**, the external observations **u**, and the output action *z*. The **u** and *z* are known (visible), but the **w** is unknown (invisible). All these elements are treated as random variants, and the joint distribution $p(\mathbf{u}, z, \mathbf{w})$ can be calculated in two ways

$$\begin{cases} p_{ying}(\mathbf{u},z,\mathbf{w}) = p(\mathbf{w})p(\mathbf{u}|\mathbf{w})p(z|\mathbf{w},\mathbf{u}) \\ p_{yang}(\mathbf{u},z,\mathbf{w}) = p(\mathbf{u})p(z|\mathbf{u})p(\mathbf{w}|\mathbf{u},z) \end{cases} \tag{6}$$

Practically, the results of these two equations are always not equal unless **w** is the optimal solution. Notice that **u** and **w** are dialectical: in training **u** and *z* are known but **w** is unknown, and **w** is obtained in terms of **u** and, while in testing or running, **w** is known but **u** and are unknown, and **w** decides what **u** and are. This interesting phenomenon fits well with the famous Chinese ancient Ying–Yang philosophy.