CMAC: Sparsely Connected Neural Network Architecture and Beyond

Tingyang Wei (G2202458H)
School of Computer Science and Engineering
Nanyang Technological University

Abstract

Cerebellar Model Articulation Controller (CMAC) is a brain-inspired algorithm modelling input-output pairs to tackle control problems. Known for its fast training behavior and hardware-targeted implementation, CMAC has been applied in many control-related applications. In this report, we briefly introduce the mechanism of CMAC and categorize the distinct strategies of improving its performance, with additional discussion covering the possible applications and its connection with other brain-inspired algorithms as well as the well-known neural network models.

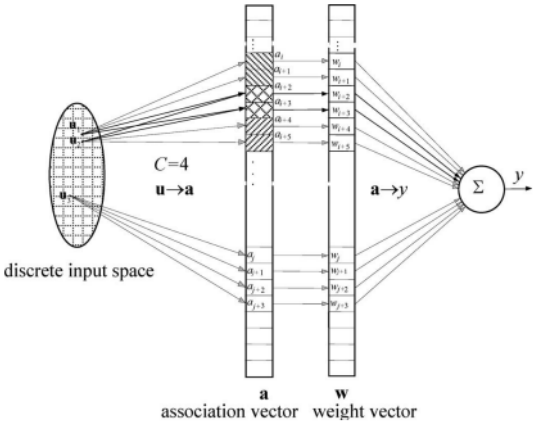

Figure 1. The CMAC architecture overview.

1. Introduction

1.1. Overview of CMAC

Cerebellar Model Articulation Controller (CMAC) is a brain-inspired algorithm that can model input-output pairs so that many control and finance-related time series prediction problems can be resolved. CMAC is a representative neuro-computing model inspired by the working mechanism of the human cerebellum. In the very first work [2], Albus mentioned the process of human beings using the cerebellum to achieve some simple movements like "pick-up glass". Albus referred to this process as two sequential operations. The first is a conscious-level operation activating a small portion of the cerebellum area and the second is a subconscious-level operation computing the degree of muscle control. Inspired by this human being control behavior, the CMAC was proposed as a control architecture with two-level, as depicted in Figure 1, where the first level is to map the input data into a series of quantization tiles in analog of conscious-level operation and the second level is to map the activated memory cells into the expected output vectors with the ability of fine-tuning in analog of the fine-grained controlled subconscious-level operation computing control precision of human beings.

The structure of the CMAC architecture can be specified in Figure 1. Mathematically, the inference process can be expressed as equation 1,

$$y(u) = a(u)^T w \tag{1}$$

where a stands for the first layer that maps input data into memory indices, association vector, u stands for the input data, and w stands for the parameter that controls the subconscious-level behavior in a fine-grained approach. From one perspective, as stated by the original work [1], the CMAC architecture can be referred to as looking up a multi-dimensional table, where the table element is the weight and the indices are the association vector, as shown in Figure 1. In Figure 1, the variable C is the dimension size of the look-up table, where a larger C value can offer a better representative ability of the CMAC model. Generally, the size of the "table" in CMAC can be really large, and thus a hash-coding solution is always adopted by projecting the large memory a into the small memory w [2]. According to the working mechanism of the CMAC model, it is generally featured as a fast-learning model [26] for small parameter size and hardware-oriented learning [9] for its consideration of memory size by hash-coding.

Although with better convergence performance compared to some classical adaptive control methods and the

Table 1. A Conclusion Remark on Representative CMAC Works

Method	Theory	White Box	Quantization	Training	Generalization	Scalability	Problem
CMAC [2]	Yes	No	Offline	LMS	Piece-wise	-	-
- [21]	No	No	Offline	-	-	-	Acrobot Learning
Analysis [16]	Yes	No	Offline	LMS	Piece-wise	-	-
GCMAC [6]	No	No	Online	LMS Growth	Fine-grained	-	Function Fitting
Kernel-CMAC [7]	Yes	No	Kernel Function	LMS Batch	Continuous	Kernel Function	-
FCMAC-Yager [20]	No	Yes	Online	Structral Hebbian	Fine-grained	-	Bank Failures
MCMAC [3]	No	No	Offline	Neighbor Momentum	Fine-grained	-	CVT Control
HCAQ-CMAC [22]	Yes	No	Nonuniform	LMS	Fine-grained	-	Vehicle Control
PSECMAC [23]	Yes	No	Nonuniform	LMS	Fine-grained	-	Financial Risks
FCMAC-AARS [27]	No	Yes	Nonuniform	LMS	Fine-grained	-	Stock Prediction

simple perceptron model [11, 12], it is still believed that the CMAC model have many deficiencies [7]. The deficiencies or limitations can be categorized into several-folds below:

- **Complexity** Since the CMAC model can be considered as a model containing a look-up table, then in real-world applications, the table size ought to be really large even with the original hash-coding trick[1]. Therefore, there should be corresponding tricks designed to resolve the large memory size of the model and the collision problem if a hash-coding trick is utilized.
- **Capability** Since the quantization space of the CMAC model is discrete, the resultant input-output landscape of the model should be discrete or piece-wise rather than continuous, which contradicts with the problem nature of most real-world applications. Therefore, the quantization space of the CMAC model should be refined for better granularity and better representation ability for more complicated problems.
- **Convergence** Since basically, the CMAC model is conducting a statistic learning task, it is important to investigate the convergence performance of the target task with training iteration proceeding. To better provide a theoretical framework or interpretation framework for the learning process for CMAC, better learning tricks can be derived by considering the error decomposition terms like Bias-Variance Decomposition [5] or VC-dim [24].

Bearing these drawbacks or limitations in mind, researchers devote much to the CMAC community by proposing both theoretical works and experimental works. In this report, we concluded several lines of inquiry for these problems to showcase better solutions to improve CMAC prediction performance in Section 2. Also, according to the similarity of CMAC with other models including canonical neural networks, radial-basis function neural networks, and even other brain-inspired neural nets like spiking neural nets and liquid state machines, we herein offer a brief discussion on the relationship between CMAC with them in Section 3, to enable better understanding the statue and value of this type of model. Section 4 concludes this report.

1.2. Overview of the Conclusion Table

According to this assignment, we also offer a brief conclusion remark in Table 1 to showcase the contributions of each representative work.

- **Theory** This column showcases whether each work covers a theoretical proof or analysis on CMAC convergence or memory size.
- **White Box** This column showcases whether each work offers an interpretable module so that the CMAC model can be interpreted by users.
- **Quantization** This column showcases how each work design the mapping mode from input to assortative vector. Herein, "Offline" refers to the quantization determined by definition in advance, "Online" refers to the quantization mode updated on the fly, and "Nonuniform" refers to a special technique that reorganizes memory cells according to the input distribution.
- **Training** This column showcases how the parameters in CMAC get tuned. Generally the weights are updated according to a pre-defined learning rate and prediction

[1]The hash-coding trick can cause collision problem and in the big data settings the collision probability is large. [7]

error, denoted as LMS, while some of the works includes refinement for better training performance.

- **Generalization** This column showcases whether the work provides a solution to improve the representative ability of CMAC. "Piece-wise" refers to the canonical CMAC since the CMAC offers only a piece-wise landscape due to the nature of quantization mode, "Fine-grained" refers to still the piece-wise landscape but with different resolutions in different positions, and "Continuous" refers to the continuous landscape.

- **Scalability** CMAC model should be able to be scalable so that it can be trained in a larger scale to be applied in more problems. However, we found that generally these works do not consider the memory complexity explicitly except for Kernel-CMAC [7].

- **Problem** This column showcases the application problem resolved by CMAC or CMAC-related [21] models.

2. Limitation and Improvements

Albeit the compact model and hardware-oriented design, the CMAC model is still restricted to several scenarios and cannot naturally be extended to more complicated problem domains. According to the listed drawbacks in Section 1, in this section, we discuss about each limitation and review the modifications of the existing works to resolve the limitation respectively.

2.1. Complexity

Complexity generally can be considered as the parameter size, and CMAC needs $M = \prod_{i=1}^{N} r_i + C - 1$ parameters where C is the quantization layer size, N is the dimension size, and r_i is the quantization size.

The original CMAC work [2], hash-coding is applied to map a large number of parameters into a small memory size, which cannot resolve the problems fundamentally but bring another collision problem.

A common solution for this problem is decomposing several dimensions into single-dimension problems with a modular architecture [15], a hierachical structure [14], or a multi-layer structure [8] where the basic element is to resolve one-dimension problem. These methods aim to reduce the size M in a divide-and-conquer fashion but may result in an increase in implementation time, since more complicated archtectures require more complicated training process. Although this section does not cover time complexity, simply sacrificing time complexity to alleviate memory complexity is not a perfect solution to make CMAC scalable.

Another line of inquiry is to offer a fine-grained control by enabling many external parameters to make the traditional CMAC architecture general. The most representative one is GCMAC [6], in which the quantization mode of the CMAC is parameterized and the parameters can be adapted on the fly. Other works in this topic generalize the quantization process with clustering methods [3, 20, 23], where the authors focus on re-distributing memory cells so that the memory density can be corresponding to the input data density. This direction claims that re-distributing memory cells can effective lower the computational burden but does not include explicit memory complexity analysis.

Kernel-CMAC is a satisfying framework in this regard. Different from other works only compromising the time complexity or only re-distributing memory cells, Kernel-CMAC can utilize kernel-trick that makes the problem complexity irrespective of data dimension but makes it upbounded by the number of training samples [7]. With the assistance of SVM training paradigm, the kernel-trick can reduce the memory complexity significantly (i.e. from 2^{25} to less than 2^{18}).

2.2. Capability

Due to the nature of quantization mode in the original CMAC (similar inputs activate nearly the same memory cells), the output landscape of CMAC can be coarse or piece-wise. This feature limits the representative power of the CMAC model and entails improvement.

Similar with the complexity approach, one common solution of this problem is still implementing quantization process in a fine-grained approach. GCMAC [6] can adaptively adjust the quantization mode corresponding to the input data domains, so that the quantization structure can better represent the input data distributions.

Likewise, a group of clustering-based method [2, 20, 23] can pre-define the dense area of the input domain so that more memory cells can be allocated into the dense distribution, and thereby allocating the resources to improve local precision of the input-output association.

Albeit the so-called multi-resolution approach can control the quantization in a better way, the resultant landscape is still discrete and piece-wise, which is not suitable for many problems requiring continuous attributes. Interpretating the CMAC as a fuzzy model [20, 27] can resolve this problem, since replacing quantization cell with member functions (i.e. high-order basis functions) can present the input domains in a continuous way. Also, kernel-trick can be interpreted as a continuous mapping for its ability of projecting input domains into an any-dimension space, which is also favored option when it comes to CMAC capability.

2.3. Convergence

Treating CMAC methods as a type of machine learning method, the convergence performance is really concerned to guarantee a fair learning time and a reasonable testing

precision.

Traditionally, researchers only compare the convergence results between CMAC and other adaptive control methods [11] or neural network models [12]. Afterwards, more specific analysis is provided to offer a theoretical ground for the CMAC family. The convergence analysis is conducted with the scenario not considering hash-coding component [25] and the scenario with learning rate as 1 always [19]. Also, Lin *et al* [16] assume no two tiles in CMAC are mapped into the same physical memory, coming into a conclusion that given a positive small iterative weighting the learning characteristics tends to be in a cycle form, and given an infinite learning process CMAC will converge to least square error with decreasing learning rate. As shown in Table 1, some studies also offer the convergence analysis to prove the corresponding contribution towards the overall training process [7, 22, 23]. Specially, Kernel-CMAC [7] proposes its framework based on a analyze upon the generalization error, and showcases its ability to bound the generalization error, which is important.

Since the convergence refers to good precision on unseen data samples for training data, one should not only consider the training performance (i.e. the bias), but should also consider the possible variation from training samples to unseen testing samples (i.e. the variance). Considering this aspect, the CMAC community may need more theoretical analysis and guidance on the bias-variance trade-off or generalization analyisis framework like error decomposition [5] or VC-dimension [24] to measure the model complexity and improve the model convergence in details.

Existing techniques of improving convergence performance still revolve around weight updating considering neighborhood or considering gradient-base tricks. The classical weight updating rule is equation 2,

$$\Delta w = \mu a e \tag{2}$$

where μ, a, e stands for learning rate, association vector, and prediction error. In a different way, [20] applies the Hebbian learning rule and [3] adds into the momentum trick into the weight updating process. Kernel-CMAC is distinct in this regard not only for the analysis on generalization error bound, but also for its completely distinct optimization problem formed by the kernel-trick in an SVM fashion.

3. CMAC versus other Frameworks

Although CMAC is a ubiquitous framework for control problems [21] and many financial-related problems [20,27], it may be replaced with other models, that is, it shares many common features with several machine learning models in both the deep learning community and the neuro-computing community. This section discusses whether CMAC can be replaced by other models, and the similar features they share.

3.1. CMAC versus RBFNN

Radial-basis function neural network (RBFNN) [18] is a classical network model for machine learning before the emergence of many advanced deep learning models. As shown in Figure 2, RBFNN shares the similar two-level architecture compared to CMAC, where the final layer is exactly the same. However, for the first layer, CMAC implements an assortative mapping to navigate the weights in the final layer, while RBFNN computes the distance to every basis function. The difference here lies in that for CMAC [2] a constraint is imposed so that the linkages between the input and the basis functions are far less than the number of the radial basis functions. By converting RBFNN into a CMAC, one can refer to equation 3 where ϕ is a matrix maintaining the distances from each dimension to each basis function. Imposing a constraint $g(\phi(u)) < \epsilon$ can reduce an RBFNN to a CMAC from an architecture view so that the assumption that only a small portion of neurons can be activated is realized. Although one may claim that the CMAC can also guarantee the generalization (similar inputs get similar outputs) performance, RBFNN can also guarantee similar performance by feeding the distance information in the first layer.

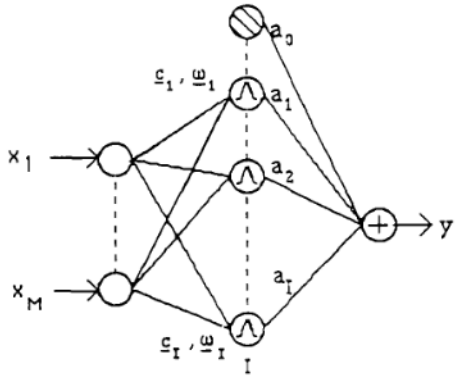

Figure 2. The RBFNN architecture overview.

$$y(u) = \phi(u)^T w \tag{3}$$

3.2. CMAC versus Neural Networks

Deep Neural Network (DNN) [13] is constructed based on perceptron in a cascade behavior. Albeit some preliminary results show perceptron can be outperformed by CMAC [12], the deep structure of DNN is more proper for big data settings with more dimensions and large sample size. Meanwhile, the two distinct features or initiatives of CMAC can also be fulfilled by DNN.

First, CMAC assumes the generalization (i.e. similar inputs generate similar outputs) structure, which can be realized by representation learning in DNN so that similar

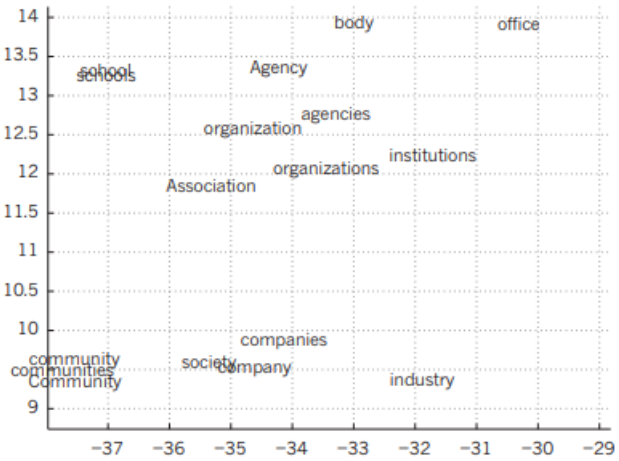

Figure 3. The word embedding overview.

inputs can generate similar intermediate outputs in embedding [13]. A representative result is word embedding [13], as shown in Figure 3, where similar words lie in similar space positions in the embedding layer.

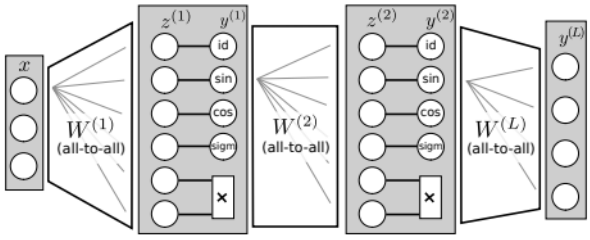

Figure 4. The equation learner network overview.

Second, CMAC assumes that the active set of connections should be sparse and far smaller than the available set of connections, which can be realized by imposing sparsity constraint on the network connections in DNN [10, 17]. As shown in Figure 4, albeit the cascade connectionism architecture is dense, it is only a conjecture space. Imposing on a sparsity constraint on the framework like only one neuron can be activated in the next layer with a maximum operator, this complicated L-layer model can simply be compressed into a equation to embody a control problem solution or a physics equation with only three or five variables [10, 17], which embraces both the sparse problem nature and the powerful representative ability of DNN.

Therefore, it can be figured out that the initiatives of CMAC that the model should be sparse and be able to smooth local structure is plausible. However, it should be noted that CMAC is not the only way to achieve this goal. To be specific, CMAC is designed for the goal in a *top-down* approach that forces model to obey certain requirements, while DNN can achieve the same goal by utilizing big data and soft constraints to obtain a satisfying model in a *bottom-up* approach.

3.3. CMAC versus Brain-inspired Methods

Herein we talk about another brain-inspired method, called Spiking Neural Nets (SNN) [4], as shown in Figure 5. SNN is also a model highly inspired by human brains, by imposing constrains on the model representations that the synaptic weights are incoming spikes with a temporal sequence. Each neuron is activated when the spike weighted sum exceeds a certain threshold determined by a complex differential equation, making it difficult to be deployed on hardware devices. Considering the nature of the spike sequence, the input domain of SNN is discrete and the connection between layers is sparse, which is similar with the assumption of CMAC. However, different from the fast-training CMAC, SNN suffers from difficult training process due to its dynamics information in each neuron, and even needs to be incorporated with DNN for an efficient training process.

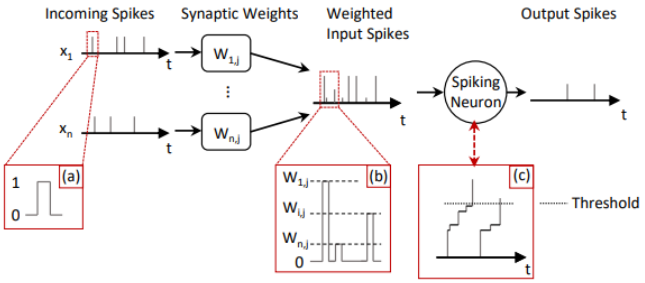

Figure 5. The spiking neural network overview.

The reason why we mention SNN here is that the limitation of SNN is caused by the strong model assumption by designers, which coincides with the problems encountered by CMAC. For CMAC, the memory size problem cannot be completely overcome because the assumption that the overall memory should depend on quantization of the whole space, but a parameterized model can resolve this problem by simply representing all the memory size implicitly using parameters. For SNN, the learning process involving the neuron dynamics can be difficult, making it infeasible in some complicated tasks so that it has to be combined with DNN for a better performance [4]. Therefore, for resolving real-world problems in the wild, in the era of big data, it may be better to simplify the initial assumptions on model to enable a more scalable architecture, instead of imposing many constraints on the initial model making it inflexible to further improve the model components.

4. Conclusion

In this paper, we taxonomies several representative works in the scope of CMAC and review the advances to resolve problems under the CMAC structure including complexity, convergence, and capability. Also, we compare CMAC with other well-known models involving DNN, SNN, and RBFNN to indicate the special features of CMAC and the limitations of CMAC.

From my personal perspective, the limitation of CMAC lies in its strong assumption of the model architecture. The memory cell settings make CMAC inflexible in terms of big data settigns with high dimension and large data sample size, and the sparsity assumption expected by Albus [2] can be offered imposing sparsity constraint on DNN model with a far less memory size using only implicit parameterized models. The so-called generalization (i.e. similar inputs generate similar outputs) assumption can be realized by many continuous model, but the strong assumption imposed by CMAC makes it hard to operate in the continuous domain, which limits further development of CMAC into a broader problem domains.

References

[1] James S Albus. A theory of cerebellar function. *Mathematical biosciences*, 10(1-2):25–61, 1971. 1

[2] James S Albus. A new approach to manipulator control: The cerebellar model articulation controller (cmac). 1975. 1, 2, 3, 4, 6

[3] Kai Keng Ang and Chai Quek. Improved mcmac with momentum, neighborhood, and averaged trapezoidal output. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 30(3):491–500, 2000. 2, 3, 4

[4] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2):1–35, 2019. 5

[5] Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th international conference on machine learning*, pages 231–238. Morgan Kaufmann Stanford, 2000. 2, 4

[6] Francisco J Gonzalez-Serrano, Anibal R Figueiras-Vidal, and Antonio Artés-Rodríguez. Generalizing cmac architecture and training. *IEEE Transactions on Neural networks*, 9(6):1509–1514, 1998. 2, 3

[7] Gbor Horvath and Tams Szabo. Kernel cmac with improved capability. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):124–138, 2007. 2, 3, 4

[8] Shih-Lin Hung, JC Jan, et al. Ms_cmac neural network learning model in structural engineering. *Journal of computing in civil engineering*, 13(1):1–11, 1999. 3

[9] Jar-Shone Ker, Yau-Hwang Kuo, Rong-Chang Wen, and Bin-Da Liu. Hardware implementation of cmac neural network with reduced storage requirement. *IEEE transactions on neural networks*, 8(6):1545–1556, 1997. 1

[10] Samuel Kim, Peter Y Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE transactions on neural networks and learning systems*, 32(9):4166–4177, 2020. 5

[11] LG Kraft and David P Campagna. Comparison of convergence properties of cmac neural networks and traditional adaptive controllers. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 1744–1745. IEEE, 1989. 2, 4

[12] LG Kraft and David P Campagna. Comparison of cmac architectures for neural network based control. In *29th IEEE Conference on Decision and Control*, pages 3267–3269. IEEE, 1990. 2, 4

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 4, 5

[14] Hahn-Ming Lee, Chih-Ming Chen, and Yung-Feng Lu. A self-organizing hcmac neural-network classifier. *IEEE Transactions on neural networks*, 14(1):15–27, 2003. 3

[15] Chien-Kuo Li and Ching-Tsan Chiang. Neural networks composed of single-variable cmacs. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 4, pages 3482–3487. IEEE, 2004. 3

[16] Chun-Shin Lin and Ching-Tsan Chiang. Learning convergence of cmac technique. *IEEE Transactions on neural networks*, 8(6):1281–1292, 1997. 2, 4

[17] Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016. 5

[18] Mohamad T Musavi, Wahid Ahmed, Khue Hiang Chan, Kathleen B Faris, and Donald M Hummels. On the training of radial basis function classifiers. *Neural networks*, 5(4):595–603, 1992. 4

[19] PC Parks and J Militzer. Convergence properties of associative memory storage for learning control systems. In *Adaptive Systems in Control and Signal Processing 1989*, pages 377–384. Elsevier, 1990. 4

[20] J Sim, WL Tung, and Chai Quek. Fcmac-yager: A novel yager-inference-scheme-based fuzzy cmac. *IEEE Transactions on Neural Networks*, 17(6):1394–1410, 2006. 2, 3, 4

[21] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8, 1995. 2, 3, 4

[22] Sintiani Dewi Teddy, EM-K Lai, and Chai Quek. Hierarchically clustered adaptive quantization cmac and its learning convergence. *IEEE transactions on neural networks*, 18(6):1658–1682, 2007. 2, 4

[23] Sintiani Dewi Teddy, Chai Quek, and EM-K Lai. Psecmac: A novel self-organizing multiresolution associative memory architecture. *IEEE Transactions on Neural Networks*, 19(4):689–712, 2008. 2, 3, 4

[24] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994. 2, 4

[25] Yiu-fai Wong and Athanasios Sideris. Learning convergence in the cerebellar model articulation controller. *IEEE Transactions on Neural Networks*, 3(1):115–121, 1992. 4

[26] Bo Yang and Huatao Han. A cmac-pd compound torque controller with fast learning capacity and improved output smoothness for electric load simulator. *International Journal of Control, Automation and Systems*, 12(4):805–812, 2014. 1

[27] Guo Zaiyi, Chai Quek, and Douglas L Maskell. Fcmac-aars: A novel fnn architecture for stock market prediction and trading. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2375–2381. IEEE, 2006. 2, 3, 4