공학 연구논문집 제29권 2호 pp. 519~530, 울산대학교.
Journal of Engineeringl Research Vol. 29, No. 2, pp. 519~530, 1998
University of Ulsan.

Application of CMAC for Short-Term Load Forecasting

Tai-Ho Lee
School of Electrical Eng. and Automation

<Abstract>

This paper addresses the use of CMAC for load forecasting. The local generalization property, along with other merits like easy implementation and fast convergence make CMAC an attractive means for this purpose. In using CMAC, we introduced two new concepts; anisotropic generalization and validity check measure. The former is to handle the situation where the requirements for the ranges of generalizations are different between input axes. The latter is a process to discriminate a part of incorrect results. Simulation showed that the CMAC is quite capable of modeling the effects of temperature and the previous day's load on load forecasting.

Keywords: CMAC, load forecasting, artificial neural networks.

단기 부하예측을 위한 CMAC의 활용에 관하여

이 태 호
전기, 전자 및 자동화 공학부

<요 약>

이 논문에서는 전력수요를 예측하는데 있어서 CMAC의 사용을 시도하였다. 초기 인공신경망의 한가지인 CMAC은 구현이 쉽고 수렴이 빠를 뿐 아니라, 지역적 일반화 (local generalization) 의 특성을 가지고 있어 부하예측의 목적에 적절하다고 사료되었다. 본 연구에서는 CMAC 에 관하여 두 가지 새로운 개념을 도입하였다. 하나는 비등방성 일반화의 개념이며, 이것으로 CMAC 의 입력축들이 서로 다른 일반화 범위를 가질 수 있도록 되었

다. 또 하나는 유효성 평가 수단이며, 이를 이용하면 CMAC 의 교육이 불완전한 경우 이를 경고할 수 있게 된다. 씨뮬레이션 에서, 온도와 전일의 부하에 관한 정보로 CMAC 을 교육하여 부하예측의 모형을 구축하고 성능을 검증하였다.

I. INTRODUCTION

Having good prediction of the next day's load demand is of great importance for utility companies since the operation and planning of utility system are largely based on this information. There have been several approaches to this task. These include 1) the methods based on various functional forms such as regression model [1], stochastic model [2] and so on, 2) expert system method [3], and 3) artificial neural network (ANN) approach. Recent studies are largely focused on the ANN approach. Due to the variety of ANN models and flexibility of their applications there have been many different methods reported within the ANN approach [4-7].

We present still another ANN method which uses CMAC (Cerebella Model Articulation Controller), one of the earliest ANN models [8]. The CMAC consists of a memory system and a set of rules to associate this memory to inputs and an output. One of the advantages of CMAC over other 'processing element based' ANN such as multi-layer perceptron (MLP) is local generalization property [9,10]. This property makes CMAC much less sensitive to the structural parameters than MLP and similar ANNs; at the same time allowing to model highly nonlinear systems. As inputs to CMAC for our task we have chosen past load values, temperature, time, and the day of a week. Each of these inputs requires different degree of generalization, which makes it necessary to design a new rule to associate memory system to inputs. The result is an anistropic CMAC which is described in Section II.

As is true for most of other forecasting systems, our CMAC is trained with historical data in the hope of finding a similar condition in the past and expecting the same result may apply at the present. This means that the system can not cope with the situation where a new trend is incorporated or the system meets an input which is located in the outside of its range experienced. This drawback may seem to be more conspicuous in the case of CMAC, since CMAC is very conservative in extrapolation. This characteristics, however, can be counted as a merit because the mathematical models formed in the neural networks are usually not appropriate to give reliable extrapolation [6]. If an input point falls in the region where the effect of local generalization fails, CMAC can generate a corrupted data. To locate this corrupted data, we propose a verifying technique in Section III with other training details. Test result is discussed in Section IV, and conclusion is given in Section V.

II. THE ANISOTROPIC CMAC

The CMAC was first proposed by Albus as a manipulator controller based on the principles of the motor behavior of the cerebellum [8]. The fundamentals of Albus' algorithm can be described by two mappings, that is:

$$f: x \rightarrow A \tag{1}$$

$$g: A \rightarrow y \tag{2}$$

In the above equations, x is an input (or, a stimulus) vector to CMAC, y is an output (or, a response), and A is a memory association vector. The function f maps input to a set of memory addresses, and the output function g generates the response by combining the contents of memories or 'weights' associated with these addresses. A CMAC is 'trained' by adjusting its weights to produce desired responses to a given set of inputs.

The main feature of the CMAC which makes it different from a simple look-up table is the sharing of weights (or receptive fields) among neighboring input vectors. This not only reduces the amount of memory needed, but also gives the generalization properties to CMAC. The number of weights associated with each input point is decided by generalization parameter, p. The output value y of a CMAC for an input point x is

$$y = \sum_{i=0}^{p} a_i W_i(x) \tag{3}$$

where $W_i(x)$ denotes one of the weights connected to x and a_i's are usually set to unities. If the input domain is n-dimensional, each receptive field which is not located at the boundary is associated with p^n input points. On the other hand each input point is connected to p receptive fields, and thus, the memory size is reduced by a factor of approximately p^{n-1}. Most of CMAC structures so far employ the same generalization parameter for every input variable. As a result the receptive field has the shape of n-dimensional hyper-cube. In some cases, however, it may be more desirable to assign different values of generalization parameter to each dimension in the input space. To cope with these situations we introduce a modification to conventional CMAC which has anisotropic generalization in the input space.

1) Association of Inputs to Receptive Fields: The organization of receptive fields with respect to the input space is best described by the overlay concept [11]. Fig. 1(a) shows a two-input CMAC with four overlays (p = 4, only three overlays out of four

are shown in Fig. 1), where each receptive field (large squares) which is not located at boundaries in any overlay are associated with 16 input points. An input point addresses one receptive field on each overlay as shown in Fig. 1(a) as a set of small squares, and the sum of weight values of these receptive fields becomes output. Thus, designing an association structure means making a set of rules to offset the receptive fields of one overlay with respect to those of others. In the case of anisotropic CMAC, this association can be designed by modifying the shape of receptive field to reflect the unequal generalizations. Fig. 1(b) shows an example with $p_1 : p_2 = 4 : 2$, where p_i's are generalization parameters for input variables 1 and 2, respectively. The receptive fields on overlay-2 are not offset in the direction of x_2-axis which reflects the fact that the number of layers is four and the width of a receptive field corresponds only two input positions along x_2-axis.

To make overlay design systematic and easy, we devised a method in which the shape of receptive field is kept the same as the normal CMAC structure (that is, hyper-cube or square in Fig. 1(a) and the values of each variable are changed instead. An input variable x_i is transformed from its old value to new one by the following equations:

$$r_i = \frac{p_{\max}}{p_i} \tag{4}$$

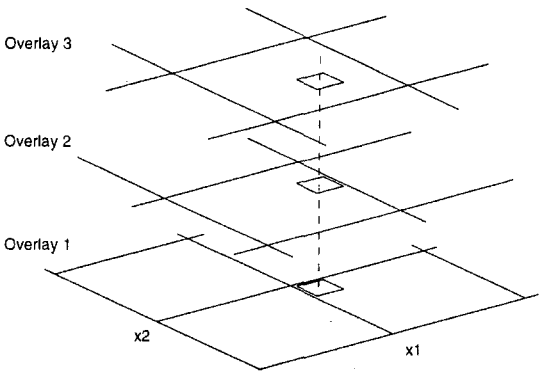
$$x_{i-new} = round[\, r_i(x_{i-old} - 1)] + 1 \tag{5}$$

In the above equations, $p_{\max}$ is the largest among generalization parameters and the rounding function, *round*[.], assigns the nearest integer to the transformed input variable when the generalization ratio r_i is not an integer. Now with the new set of input variables, generalization parameters are the same for all directions of input space, that is, $p_{\max}$, and techniques developed for the arrangement of the layers for isotropic CMAC can be used in the same manner.

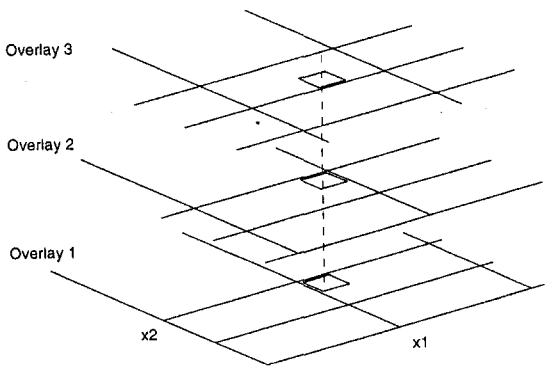
2) Memory Reduction: As mentioned earlier, memory requirements of a CMAC is reduced by a factor of approximately p^{n-1}[11]. For an anisotropic CMAC, this reduction ratio, R, is changed and can be expressed by the following equation:

$$R \approx \left(\prod_{i=1}^{n} p_i \right) / p_{\max} \tag{6}$$

The proof of (6) is straightforward. If the i-th input variable is quantized into N_i intervals, the required number of receptive fields is given by



(a) Normal CMAC



(b) Anisotropic CMAC

Fig. 1. Examples of overlay design.

$$m = \sum_{j=0}^{p_{max}} \prod_{i=1}^{n} \left(\left[\frac{N_i - d_{ij}}{P_i} \right] + 1 \right), \tag{7}$$

where m is the size of memory, d_{ij} is the i-th component of the displacement vector for the j-th overlay, and $[x]$ denotes the nearest integer not less than x. If $N_i >> p_i$, then $m \approx p_{\max} \prod_{i=1}^{n} (N_i / p_i)$ and, since the memory size for a simple look-up table is $\prod_{i=1}^{n} N_i$, we get the reduction ratio shown in (6).

III. TRAINING OF CMAC

1) Structure and Training Process: A four dimensional nonlinear form was assumed as load model, that is,

$$y(i,t) = f[y(i-1,\ t'),\ T(i,t),\ d,\ t] \tag{8}$$

where

t and t': the times in hour,

d : the day of a week,

$y(i,t)$: load value at day i, time t , and,

$T(i,t)$: temperature at day i, time t.

Two alternatives were tried for t', that is, $t' = t$ and $t' =$ a fixed time. The latter gave better results and the data presented in the next section are obtained using $t' =$ 24. After actual training we also found out that the generalization over the day of a week gave undesirable effect and the generalization parameter for this input was set to unity, which made CMAC a set of separate tables with respect to variable d. The generalization parameters for y, T and t were set to 18, 18 and 6, respectively. Preliminary training was performed over the past five years' data. Data were supplied in daily sequence. Because the amount of data was fairly large and the daily sequence did not necessarily mean the orderly sequence in the input space, the unwanted effect of sequential training could be minimized. The rms error for the training data was around 1.5% of maximum load. This error could be expected because the model described by equation (8) obviously could not explain all the behavior of the target system which is inherently random in nature. The input space of CMAC should be quantized into finite number of bins and this also can be a source of error. In our case, the previous load value to be used as one of the inputs was quantized into 120 intervals over its dynamic range, which corresponds to a quantization error of approximately 1/3% of maximum load.

2) Verification: As pointed out in [6], it is not desirable to operate a neural network in the outside of its trained range where a kind of extrapolation is performed. In the case of CMAC this corresponds to the situation when the input point falls on the place where local generalization fails.

To discriminate these corrupted data, we ran two CMAC systems with different initial state, one for 'forecasting'

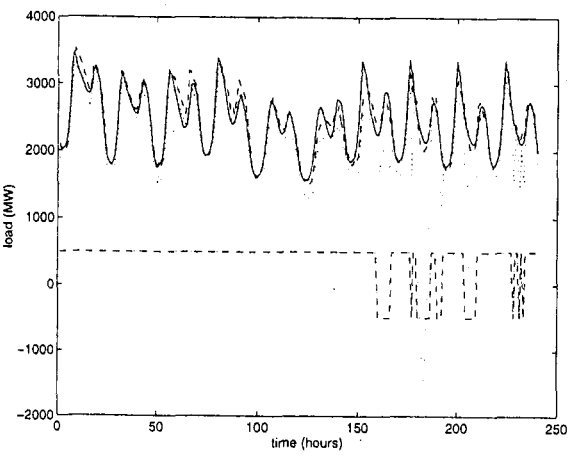


Fig. 2. An example of validity check. Solid line: actual load, dashed line: forecasted load (y), dotted line: verifying output (, shifted downward to show other curves), dash-dotted line: validity check function C.

output and the other for 'verifying' output. The verifying output is to be used to test if the forecasting output is reliable or not. Though the initial state does not have much effect on the training of CMAC, the output of CMAC in the neighborhood of untrained weight can be changed by 'biasing' the initial value. For forecasting output we set the bias value at the minimum load level to minimize this effect. For verifying output, however, the bias is set to a large negative value to make the output very different from the forecasting one when the system is operating in an error prone region. Thus, the outputs of these two systems are nearly same when the CMAC is operating in the properly trained regions. But, if the point moves out of these regions, the difference between them can be very large. Using this property a validity check function is defined as following:

$$C = -b \cdot sign[|y - y_v| - \theta], \tag{9}$$

where y and y_v are forecasting and verifying outputs, respectively, b is an arbitrary positive constant, and θ is a threshold value which can be chosen by experiment. When CMAC is operating in the outside of a trained region, $|y - y_v| > \theta$, and C becomes negative value. An example is shown in Fig. 2 where (θ =300 is used to calculate C. In Fig. 2, the intervals with negative values of C correspond to the insufficiently trained regions. It is noticeable that even in these regions the forecasting curve can follow the actual load fairly close.

IV. SIMULATION RESULTS

The simulation was carried out using six years' data (from 1985 to 1990) supplied by Pudget Sound and Light Company. After a preliminary training over five year's data, additional training was carried out every day using one year's data. There existed tendencies slightly to overestimate or underestimate, according to the season. We made a compensation for this using the result of regression analysis over the previous year.

1) Peak Load Forecasting: Table I and Fig. 3 show the results of forecasting for the daily peak load. Some of the data were omitted in the statistics when the verifying output showed too much corruption. As seen in the table, the mean absolute percentage error (MAPE) was 3.01%, which is comparable to the back-propagation model in [6].

2) Effect of Temperature: A comparison test is performed to see the effect of temperature information. In this experiment CMAC is trained and tested assuming constant temperature. Fig. 4 compares the normalized peak load forecasting error for one month. The effect of temperature input is obvious.

3) Hourly Load Forecasting: The monthly MAPE values for hourly forecasting are given in Table II. Result was not quite satisfactory even though part of the corrupted data were omitted in this statistics. Fig. 5 shows some of the typical daily load patterns. It is seen clearly that the pattern varies drastically from day to day which makes it very difficult to predict with the input set consisting of so few items. A better result is expected if additional information is incorporated.

4) Refinement using MLP: As an attempt to compensate the trends of error, we tried a back-propagation trained MLP with one hidden layer. The network was to accept a set of 24-hour data forecasted by CMAC and produce a new 24-hour forecast data. Training was carried out over 22 sets of previous data. The number of hidden nodes used was 24. Some of the best results are given in Fig. 5. Although no gains were made on the average, MLP produced promising results in many portions of test range. Better conditioning of training data set and a better measure to discriminate the bad results from good ones would greatly help in improving the performance of MLP.

Table I
AVERAGE FORECASTING ERRORS FOR PEAK LOAD

Month	Normalized rms error	Average error(MAPE/100)
January	0.0420	0.0378
February	0.0365	0.0287
March	0.0356	0.0287
April	0.0325	0.0303
May	0.0351	0.0298
June	0.0397	0.0341
July	0.0387	0.0313
August	0.0327	0.0269
September	0.0299	0.0246
October	0.0324	0.0300
November	0.0316	0.0285
December*	0.0313	0.0302
Average	0.0361	0.0301

* Half month average

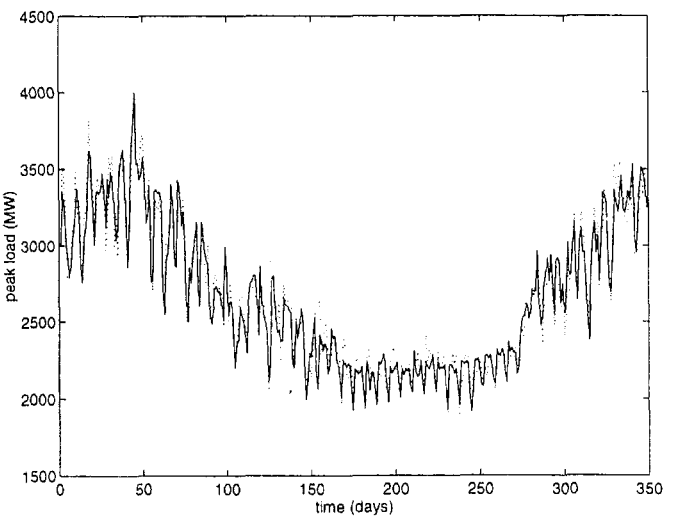


Fig. 3. Peak load forecasting. Solid line represents actual load and the dotted line represents forecasted load.

Table II
AVERAGE FORECASTING ERRORS FOR HOURLY LOAD

Month	Normalized rms error	Average error (MAPE/100)
January	0.0465	0.0334
February	0.0585	0.0433
March	0.0534	0.0433
April	0.0472	0.0392
May	0.0442	0.0361
June	0.0643	0.0492
July	0.0673	0.0512
August	0.0504	0.0374
September	0.0445	0.0384
October	0.0510	0.0404
November	0.0459	0.0399
December*	0.0663	0.0498
Average	0.0553	0.0417

* Half month average

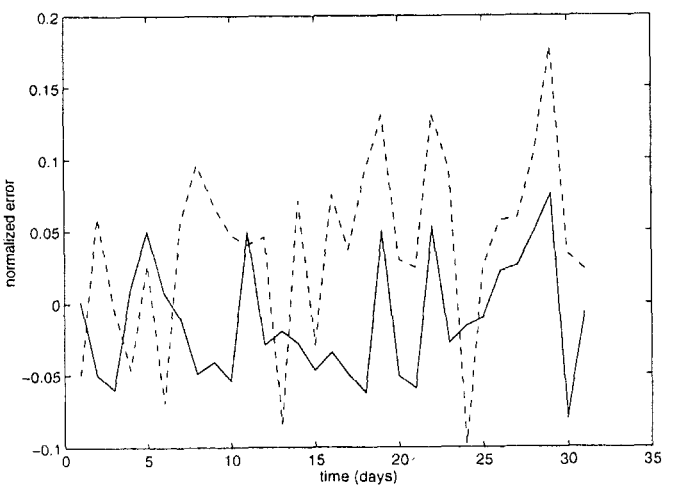


Fig. 4. Effect of temperature in model. The model with temperature input(solid line) shows less error than the model without one (dashed line)

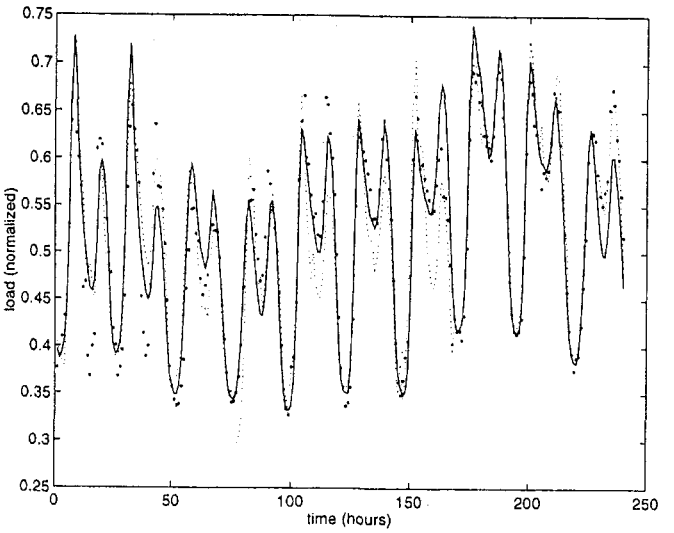


Fig. 5. A result of refinement process by MLP. Solid line: actual load, large dots: output of CMAC, dotted line: output of refining MLP.

V. CONCLUSION

In this study a CMAC neural network for load forecasting was investigated. The result showed that the CMAC is quite capable of modeling the effects of temperature and the previous day's load on the load forecasting. At the same time it was also seen that the model by itself was not sufficient to give reliable data. That is, a composite system is needed, in which various information including weather conditions other than temperature, social occasions like holidays, festivals and sports events and so on, is needed to be incorporated. A successful composite system is given in [12] based on functional models, and the researches on the composite system using ANN models are also appearing [13]. In a composite system, various types of models can be used for the implementation of its partial systems and for the fusion of information from the partial systems. The CMAC presented in this study is considered to be appropriate for a partial system. Most ANNs with MLP-like structures can be very sensitive to the structural parameters like number of nodes and the initial conditions; which makes it a painstaking work to find out an optimal structure and, usually needs numerous trials to reach an acceptable minimum. Since CMAC suffers less of these difficulties and also has other merits like easy implementation and fast convergence, we expect that CMAC can be conveniently used in a practical composite load forecasting system.

REFERENCES

[1] N. F. Hubele and C. S. Cheng, "Identification of Seasonal Short-Term Forecasting Models Using Statistical Decision Functions", *IEEE Trans. on Power Systems*, vol. 5, no. 1, pp. 40-45, 1990.

[2] F. Galina, E. Handschin and A. Fiechter "Identification of Stochastic Electric Load Models for Physical Data", *IEEE Trans. on Automatic Control*, vol. AC-19, no. 6, pp. 887-893, 1974.

[3] S. Rahman and R. Bhatnagar, "An Expert System Based Algorithm for Short-Term Load Forecast", *IEEE Trans. on Power Systems*, vol. 3, no. 2, pp. 392-399, 1988.

[4] K. Y. Lee, Y. T. Cha and J. H. Park, "Short-Term Load Forecasting Using an Artificial Neural Network", *IEEE Trans. on Power Systems*, vol. 7, no. 1, pp.124-130, 1992.

[5] T. M. Peng, N. F. Hubele and G. G. Karady, "Advancement in the Application of Neural Networks for Short-Term Load Forecasting", *IEEE Trans. on Power Systems*, vol. 7, no. 1, pp. 250-257, 1992.

[6] D. K. Ranaweera, N. F. Hubele and A. D. Papalexopoulos, "Application of Radial Base Function Neural Network Model for Short-Term Load Forecasting", *IEE Proceedings-C*, vol. 142, no. 1, pp. 45-50, 1995.

[7] K. Y. Lee, T. I. Choi, C. C. Ku and J. H. Park, "Short-Term Forecasting Using Diagonal Recurrent Neural Network", *Proceedings of the Second International Forum on Applications of Neural Networks to Power Systems*, 1993, pp. 227-232.

[8] J. S. Albus, "A New Approach to Manipulator Control: the Cerebella Model Articulation Controller (CMAC)", *Transactions of the ASME, Series G.* vol. 97, pp. 220-227, 1975.

[9] M. Brown, C. J. Harris and P. C. Parks, "The Interpolation Capabilities of the Binary CMAC", *Neural Networks*, vol. 6, pp. 429-440, 1993.

[10] W. T. Miller, III, F. H. Glanz and L. G. Kraft, III, " CMAC: An Associative Neural Network Alternative to Backpropagation", *Proceedings of IEEE*, vol. 78, no. 10, pp. 1561-1567, 1990.

[11] M. Brown and C. J. Harris, *Neurofuzzy Adaptive Modelling and Control*, Hertfordshire, UK: Prentice Hall, 1994, pp. 219-229.

[12] J. H. Park, Y. M. Park and K. Y. Lee, "Composite Modeling for Adaptive Short-Term Load Forecasting", *IEEE Trans. on Power Systems*, vol. 6, no. 2, pp. 450-457, 1991.

[13] A. Piras, A. Germond, B. Buchenel, K. Imhof and Y. Jaccard, "Heterogeneous Artificial Neural Network for Short Term Electrical Load Forecasting", *IEEE Trans. on Power Systems*, vol. 11,.no. 1, pp.397-402, 1996.