search method has been developed based on boosting to append classifier kernels one by one in an orthogonal forward regression procedure. Experimental results presented have demonstrated the effectiveness of the proposed technique.

REFERENCES

[1] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[2] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.

[3] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Machine Learn. Res.*, vol. 1, pp. 211–244, 2001.

[4] K. Z. Mao, "RBF neural network center selection based on Fisher ratio class separability measure," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1211–1217, 2002.

[5] S. Chen, L. Hanzo, and A. Wolfgang, "Kernel-based nonlinear beamforming construction using orthogonal forward selection with Fisher ratio class separability measure," *IEEE Signal Process. Lett.*, vol. 11, no. 5, pp. 478–481, 2004.

[6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[7] R. E. Schapire, "The strength of weak learnability," *Machine Learn.*, vol. 5, no. 2, pp. 197–227, 1990.

[8] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," in *Advanced Lectures in Machine Learning*, S. Mendelson and A. Smola, Eds. Berlin, Germany: Springer Verlag, 2003, pp. 119–184.

[9] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison Wesley, 1989.

[11] K. F. Man, K. S. Tang, and S. Kwong, *Genetic Algorithms: Concepts and Design*. London, U.K.: Springer-Verlag, 1998.

[12] L. Ingber, "Simulated annealing: Practice versus theory," *Math. Comput. Model.*, vol. 18, no. 11, pp. 29–57, 1993.

[13] S. Chen and B. L. Luk, "Adaptive simulated annealing for optimization in signal processing applications," *Signal Process.*, vol. 79, no. 1, pp. 117–128, 1999.

[14] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

$O(\log_2 M)$ Self-Organizing Map Algorithm Without Learning of Neighborhood Vectors

Hiroki Kusumoto and Yoshiyasu Takefuji

***Abstract*—In this letter, a new self-organizing map (SOM) algorithm with computational cost $O(\log_2 M)$ is proposed where M^2 is the size of a feature map. The first SOM algorithm with $O(M^2)$ was originally proposed by Kohonen. The proposed algorithm is composed of the subdividing method and the binary search method. The proposed algorithm does not need the neighborhood functions so that it eliminates the computational cost in learning of neighborhood vectors and the labor of adjusting the parameters of neighborhood functions. The effectiveness of the proposed algorithm was examined by an analysis of codon frequencies of *Escherichia coli (E. coli)* K12 genes. These drastic computational reduction and accessible application that requires no adjusting of the neighborhood function will be able to contribute to many scientific areas.**

***Index Terms*—Binary search, computational reduction, codon frequency, *Escherichia coli (E. coli)*, neighborhood function, self-organizing map (SOM), subdividing method.**

I. INTRODUCTION

A self-organizing map (SOM) algorithm is one of unsupervised learning methods in the artificial neural network in order to map a multidimensional input data set into two-dimensional (2-D) space according to the neighborhood function. The first SOM algorithm was originally developed by Kohonen [1] and has been used in a variety of research areas including speech or speaker recognition [2], mathematics [3], financial analysis [4], color quantization [5], identification and control of dynamical systems [6], color clustering [7], and bioinformatics [8]–[10]. Particularly in the field of bioinformatics, many researchers have adopted SOM algorithm for analysis of gene sequences as a method of clustering, visualization, or feature extraction. Wang *et al.* clustered genes according to codon usage by SOM algorithm in order to identify highly expressed and horizontally transferred genes [8]. Sultan *et al.* and Gill *et al.* applied SOM algorithm to analyze microarray data [9], [10].

When M^2 is the size of a feature map, the number of compared weight vectors for one input vector to search a winner vector by exhaustive search is equivalent to M^2. Tree-structured SOM proposed by Koikkalainen and Oja [11] and Truong [12] to improve the winner search reduces the number of searching operations to $O(M \log M)$. Kohonen proposed a new method with the total number of comparison operations by $O(M)$ [1]. Self-organizing topological tree with $O(\log M)$ was proposed by Xu and Chang [13].

In this letter, a new SOM algorithm with $O(\log_2 M)$ is proposed where it is composed of the subdividing method and the binary search method. The proposed algorithm not only reduces the computational costs but also eliminates the time-consuming parameter tuning in the neighborhood function in SOM applications. When we use SOM for practical analyses, one of the most time-consuming tasks for effective learning is to adjust the values of several parameters, particularly in neighborhood function. In addition to that, the neighborhood function has a critical effect on the performance of SOM. In the proposed algorithm, only winner vectors are trained. The proposed algorithm not to train neighborhood vectors is completely original.

Manuscript received May 4, 2005; revised April 3, 2006.

The authors are with the Keio University, Kanagawa 252-8520, Japan (e-mail: kusu@sfc.keio.ac.jp).

Digital Object Identifier 10.1109/TNN.2006.882370

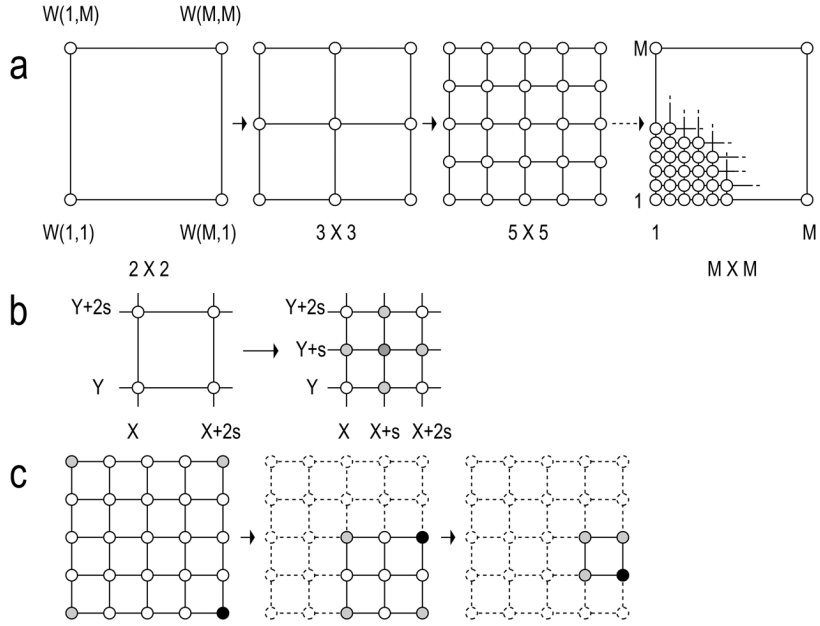

Fig. 1. Processes of the proposed algorithm. Circles denote weight vectors. (a) Process of the subdividing method. (b) New nodes by subdivision (a part of a map). (c) Process of the binary search method.

The proposed algorithm subdivides the map repeatedly, and new nodes of weight vectors emerge in every step. The idea of emerging new nodes is observed in growing SOMs such as the interpolation–extrapolation method proposed by Kohonen [1] as well. In this method, an interpolated vector is assigned a value that is calculated from two close vectors and an interpolation coefficient, when the map is linear. With a 2-D map, the value assigned to an interpolated vector is calculated from three vectors and two interpolation–extrapolation coefficients. On the other hand, in the proposed algorithm, dealing 2-D SOM, the newly emerging nodes are simply assigned the average of the closest vectors, because the map is subdivided uniformly; new nodes emerge at the center of two or four old nodes (see Section II-B).

The proposed algorithm was applied to a mapping of *Escherichia coli* genes' codon frequency data. In translation of gene into protein, codons that consist of three consecutive nucleotides occurring in messenger ribonucleic acid (mRNA) direct the incorporation of specific amino acids via transfer ribonucleic acid (tRNA) with corresponding anticodon. The combination of four types of nucleotides in the length of three makes 64 types of codons. When these 64 types of codons code for 20 types of amino acids, there are several synonymous codons corresponding to one amino acid. In bacteria, the frequency of these synonymous codons usage is not random. It has been reported that there is a positive correlation between a codon bias and the level of gene expression. Codons that correspond to abundant tRNA are preferred in highly expressed genes. The result of mapping of the codon frequency data was analyzed based on levels of gene expression identified by Sharp and Li [14].

II. Algorithm

The proposed algorithm is composed of two methods. One is a binary search for searching winner vectors and the other is a method of subdividing feature map gradually. At the initialization in the proposed algorithm, there is a 2×2 weight vector on SOM as shown in Fig. 1(a). As the process of the proposed algorithm proceeds, the feature map is subdivided by the subdividing method. At any subdivision stage in the proposed algorithm, winner vectors are searched roughly at the beginning and accurately at the end by the binary search method. The proposed algorithm does not have the neighborhood function, because one and only one winner–vector set is trained every stage. The procedure is detailed in this section.

A. Initialization

The input is a data set of k-dimensional vectors $\boldsymbol{X}_i (i = 1, 2, 3, \ldots, n)$. A feature map is a 2-D layer of $M \times M$ nodes $(M = 2^m + 1, m = 1, 2, 3, \ldots)$. M can be determined arbitrarily as long as m is a natural number. The proposed algorithm allows to take a big enough size map, because the size of the map does not make much difference in computational costs in the proposed algorithm. At the beginning, only four nodes on the coordinates $(1, 1), (1, M), (M, 1)$, and (M, M) have k-dimensional weight vectors $\boldsymbol{W}(x, y)$ whose values are arbitrary in the extent of the distribution of input data and other nodes do not appear as shown in Fig. 1(a). These four weight vectors are trained by the same method as the basic SOM with total $O(1)$ computation.

B. Subdividing Method

The subdividing method draws center lines between all neighboring two lines on the feature maps, so that it subdivides an $M' \times M'$ feature map into a $(2M' - 1) \times (2M' - 1)$ feature map. Fig. 1(a) shows the process of the subdivisions from a 2×2 map to a $M \times M$ map. Every new node is assigned a weight vector, whose value is the average of the values of weight vectors of the closest nodes to the new node. The values of the new gray nodes in Fig. 1(b) are defined by

$$\boldsymbol{W}(X, Y + s) = \frac{\boldsymbol{W}(X, Y) + \boldsymbol{W}(X, Y + 2s)}{2} \quad (1)$$

$$\boldsymbol{W}(X + s, Y) = \frac{\boldsymbol{W}(X, Y) + \boldsymbol{W}(X + 2s, Y)}{2} \quad (2)$$

$$\boldsymbol{W}(X + s, Y + 2s) = \frac{\boldsymbol{W}(X, Y + 2s) + \boldsymbol{W}(X + 2s, Y + 2s)}{2} \quad (3)$$

$$\boldsymbol{W}(X + 2s, Y + s) = \frac{\boldsymbol{W}(X + 2s, Y) + \boldsymbol{W}(X + 2s, Y + 2s)}{2} \quad (4)$$

TABLE I
COMPUTATIONAL TIMES (S) AND ORDERS

Size of feature map	9^2	17^2	33^2	65^2	129^2	Order
The basic SOM	9.8	34.7	132.5	509.4	2149.7	M^2
The proposed algorithm	1.5	1.8	2.1	2.4	2.7	$\log_2 M$

$$\boldsymbol{W}(X+s,Y+s)=\frac{\boldsymbol{W}(X,Y)+\boldsymbol{W}(X,Y+2s)}{4}+\frac{\boldsymbol{W}(X+2s,Y)+\boldsymbol{W}(X+2s,Y+2s)}{4} \tag{5}$$

where s denotes the distance of the neighboring two nodes after the subdivision.

After each subdivision, winner vectors searched by the binary search are trained with the total $T(p)$ times. T is an overall total learning times and p is the number of subdivision stages when the feature map size is $(2^{p-1}+1)\times(2^{p-1}+1)$. This final size of the map is determined at the initialization, and the subdivision stops at the size.

C. Binary Search

When an $M''\times M''$ map (that means that $M''\times M''$ nodes on a map have weight vectors and other nodes do not appear) and an input data $\boldsymbol{X}(t)$ are given, a winner vector is searched by following procedures.

At the initialization, the search space of the map is extended by

$$0\le x\le M \quad \text{and} \quad 0\le y\le M. \tag{6}$$

All space of the map is the subject of the search. After repeating Steps A and B $\log_2(M''-1)+1$ times, respectively, the final closest vector $\boldsymbol{W}(x_c,y_c)$ is the winner vector.

Step A: When search space is extended by

$$x_1\le x\le x_2 \quad \text{and} \quad y_1\le y\le y_2 \tag{7}$$

the closest vector to $\boldsymbol{X}(t)$ is searched from four weight vectors on the vertices of the search space

$$\|\boldsymbol{X}(t)-\boldsymbol{W}(x_c,y_c)\|=\min\{\|\boldsymbol{X}(t)-\boldsymbol{W}_i\| \mid \boldsymbol{W}_i=\boldsymbol{W}(x_1,y_1),\boldsymbol{W}(x_1,y_2),\boldsymbol{W}(x_2,y_1),\boldsymbol{W}(x_2,y_2)\}. \tag{8}$$

Step B: The search extent is divided into four quarters and the proposed algorithm assumes that a winner vector is on a quarter space where the closest vector $\boldsymbol{W}(x_c,y_c)$ exits. The extent of the search space is changed into

$$x_1'\le x\le x_2' \quad \text{and} \quad y_1'\le y\le y_2' \tag{9}$$

$$x_1'=\min\left(x_c,\frac{x_1+x_2}{2}\right) \quad x_2'=\max\left(x_c,\frac{x_1+x_2}{2}\right) \tag{10}$$

$$y_1'=\min\left(y_c,\frac{y_1+y_2}{2}\right) \quad y_2'=\max\left(y_c,\frac{y_1+y_2}{2}\right). \tag{11}$$

Fig. 1(c) shows the process of the binary search when $M'=5$. The solid lines mean the searched space, the weight vectors of the black and gray nodes are compared. The three weight vectors of the black nodes are the closest vectors to the input vector $\boldsymbol{X}(t)$ at each step and the final closest vector $\boldsymbol{W}_{\rm win}$ is a winner.

D. Learning

The proposed algorithm only trains winner vectors by

$$\boldsymbol{W}_{\rm win}(t+1)=\boldsymbol{W}_{\rm win}(t)+\alpha(t)(\boldsymbol{X}(t)-\boldsymbol{W}_{\rm win}(t)). \tag{12}$$

Because neighborhood vectors do not require training, learning rate α $(0<\alpha<1)$ does not have arguments of the coordinates of weight vectors. Value of α can decrease in inverse proportion to time argument t meaningfully and constant α can also work valuably.

III. SIMULATIONS AND RESULTS

A. Computational Costs

The proposed algorithm was tested by a data set consisting of 2400 three-dimensional (3-D) input vectors. The programs of the basic SOM adapting an exhaustive search and the proposed algorithm were written with C programming language and simulated on a PC of Pentium III 600 MHz CPU. Five kinds of size of feature maps from 9×9 to 129×129 were used for comparisons. At each simulation, total 50 000 input data chosen from 2400 data, randomly, was used for learning of weight vectors with constant learning rate $\alpha=0.00025$. Table I shows the computational times for each simulation. The proposed algorithm is a batch method. In this simulation, total times from the input of the data and the output of the results were measured.

The most of the computational costs of SOM are for searching of the winner vectors. The number of comparison operations for finding out the winner vectors by the proposed algorithm is $2^2\log_2 M$ and the order of the computational cost of the proposed algorithm is $\log_2 M$. This order is justified by the result of computational costs as shown in Table I.

B. Codon Frequencies of E. Coli K12 Genes

The proposed algorithm was applied to a feature extraction from *E. coli* K12 genome taken from GenBank. We obtained the 59-dimensional input vectors from the codon frequency of 4299 genes that contain at least 100 codons. The codon frequency of the vth gene for the $t(u)$th codon $R_{vt(u)}$ was calculated by

$$R_{vt(u)}=\frac{Z_{vt(u)}}{\frac{1}{n(u)}\sum_{t=1}^{n(u)}Z_{vt(u)}} \tag{13}$$

where $Z_{\rm vt(u)}$ is the $t(u)$th synonymous codon number for the uth amino acid and $n(u)$ is the number of codons for the uth aminoacid. The three-stop codons and the two codons encoding methionine (Met) and tryptophan (Trp) were excluded.

Fig. 2(a) shows the result of the mapping by the proposed algorithm, when frequency of iterative learning is 1 000 000, constant learning rate 0.001, and map size 33×33. X and Y axes denote a map and Z axis denotes the number of genes mapped on each weight vector. Fig. 2(b)–(f) shows the breakdowns by five categories based on the expression level: "very highly expressed genes," "highly expressed genes," "moderate codon usage bias," and "low codon usage biases, " classified by Sharp and Li [14], and another functional category "ribosomal protein" from Riley's gene catalogue [15].

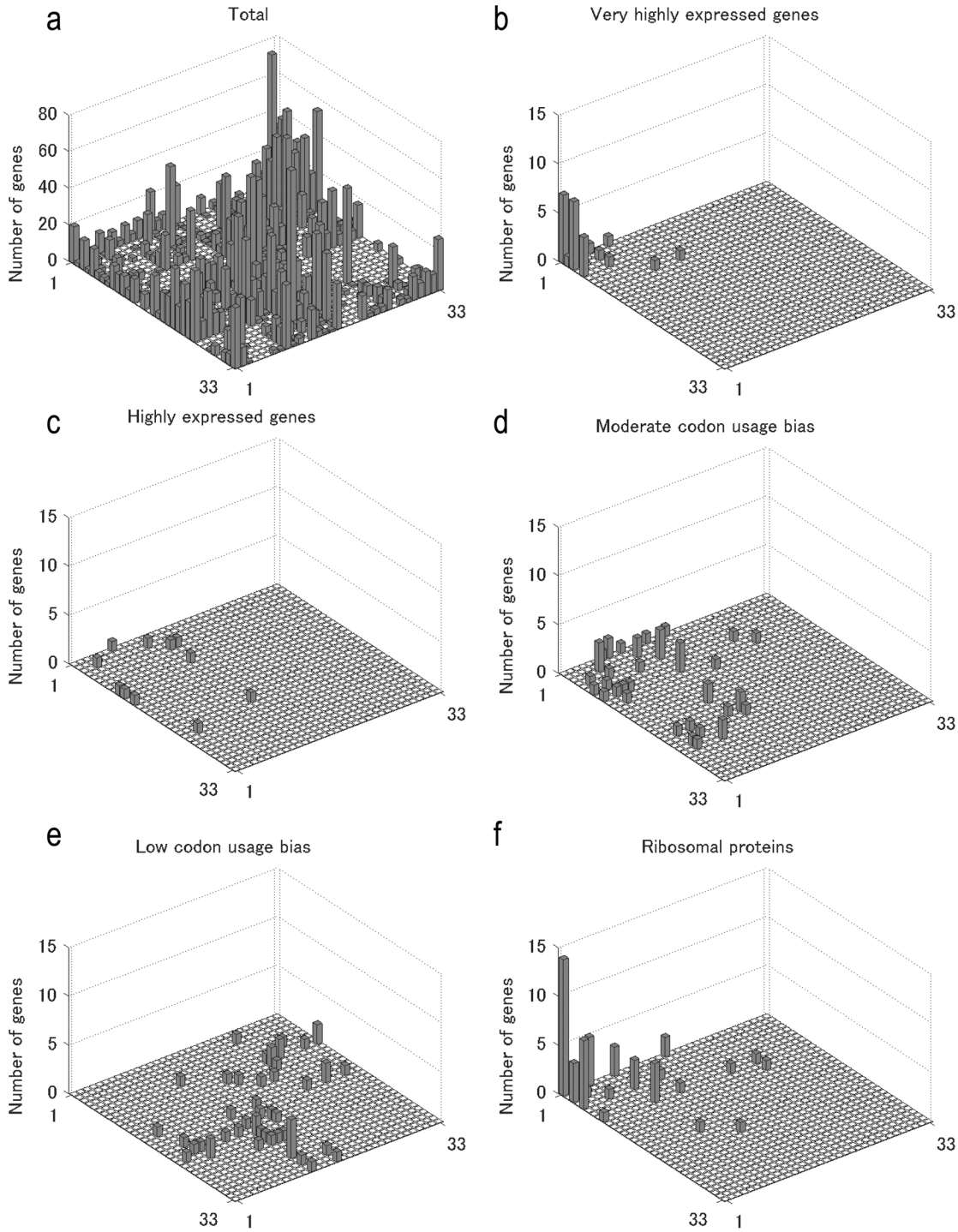

Fig. 2. Mapping results of codon frequencies of *E. coli* genes by the proposed algorithm. A lattice plane is a 33×33 map. (a) Third dimension indicating the number of genes mapped on each weight vector. Genes of four categories based on expression level: (b) very highly expressed genes, (c) highly expressed genes, (d) moderate codon bias, and (e) low codon usage bias [14]. (f) Genes encoding ribosomal proteins [15].

All of the 4299 genes, with/without categorical attributes, were distributed across the feature map [Fig. 2(a)]. At the left corner, "very highly expressed genes" were clustered [Fig. 2(b)], and "highly expressed genes" were mapped near the left corner [Fig. 2(c)]. Genes with "moderate codon usage biases" were mapped on the left half of the map, surrounding but clearly avoiding the weight vectors around left corner: the cluster of "very highly expressed genes" [Fig. 2(e)]. Genes with "low codon usage biases" were scattered in the middle of the map, avoiding four corners: the relatively large parts of the left, bottom, and right sides and narrower part around the top corner of the map [Fig. 2(f)]. These results are supporting the former works referring to the theory that a very high bias on a codon usage is seen in the highly expressed genes and a rather low bias in other genes [14].

The mapping results showed that 14 out of 20 input vectors mapped on coordinate (1, 1) were the "ribosomal protein" encoding genes. "Very highly expressed genes" were most frequently mapped on this weight vectors as well. This is consistent to the widely known fact that many "ribosomal proteins" are expressed highly. Genes mapped

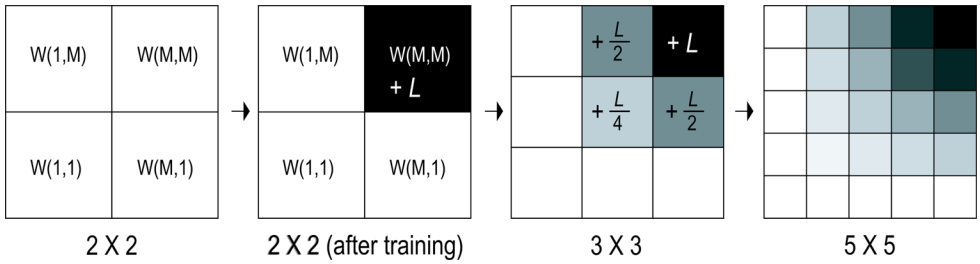

Fig. 3. Transmission of a learning effect through subdivisions from 2×2 to 5×5. Squares denote weight vectors. Only $W_{(M,M)}$ is trained just one time before the subdivisions. L denotes a variation of $W_{(M,M)}$ by the training. The magnitudes of the transmitted learning effects are expressed in the darkness of the squares.

on the left corner were not the majority of the entire *E. coli* genes [Fig. 2(a)]. The majority of genes were mapped around where the genes with "low codon bias" were clustered indicating the majority of the genes may have low codon bias.

IV. Discussion

The proposed algorithm does not search all weight vectors for the winner vectors. This characteristic of the approach avoids a problem associated with the basic SOM with a large feature map, in which two similar input vectors that belong to the same cluster are mapped on the distant weight vectors. The basic SOM compares all weight vectors for an input vector. When the size of a feature map is large, it occurs frequently that the two weight vectors at distant coordinates on the feature map have accidentally very close values. In that case, two similar input vectors would be mapped on the distant two weight vectors, and end up forming two separate clusters on the feature map. This problem scarcely occurs with a small map. Even when the final feature map size is large, this problem will not likely occur in the proposed algorithm. The proposed algorithm starts with a 2×2 map as shown in Fig. 2(a). After learning with the 2×2 map, the values of four weight vectors are very different as is the case of the basic SOM with a small feature map. Then, the map size is gradually multiplied by the subdividing method. The values of the four vectors on the vertices of the feature map do not become closer by learning, because the proposed algorithm does not train neighboring vectors of a winner vector. After all the learning iterations are finished, every input vector is mapped on the winner vector found by the binary search. At the beginning of the binary search, a temporary winner vector is found among the four weight vectors on the vertices of the feature map. Because the weight vectors on the vertices are supposedly different for the aforementioned reason, the same weight vector would be selected as a temporary winner vector for the two similar input vectors. The search space is reduced to a quarter that includes the temporary winner vector by the subdividing method. The values of the four vectors on the vertices of the quarter are different also, because the new weight vectors are assigned the average values of the neighboring weight vectors when the map is subdivided 2×2 to 3×3. The details of this reason are described in the next paragraph. The same temporary winner vector for the two similar input vectors is found. Thus the two similar input vectors would be mapped on the weight vectors at the close distant coordinates on the feature map.

The binary search method reduces the computational costs and can work only when it is combined with the subdividing method. Without the subdividing method, the binary search would fail in finding out the correct winner vectors. In the subdividing method, the new weight vectors, which are created in the course of subdividing of the map, are assigned the average values of the neighboring weight vectors. This makes the feature map with weight vectors arranged in continuous change. This means that the four weight vectors on the vertices of the feature map or the search space can represent one of the quarter space of the feature map or the search space that includes the vertex itself. The continuous change of the weight vectors created by the subdividing method assures the effective working of the binary search.

The proposed algorithm eliminates the time-consuming parameter tuning in neighborhood function that has a critical effect on the performance of SOM. In the neighborhood function, there is a term of the distance from a winner vector that concerns the learning rates. Generally, the learning rates for the vectors close to the winner are set high and those for vectors far from the winner are set low. This variation of learning rates set in the neighborhood function is effectively replaced by the subdividing method in proposed algorithm as discussed in the next paragraph.

Fig. 3 shows transmissions of a learning effect, assuming that only one vector $W_{(M,M)}$ is trained into $W_{(M,M)} + L$ just one time. Each square denotes a weight vector and L denotes the variation of $W_{(M,M)}$ by the training. The learning effect L is represented by black color in Fig. 3. In the course of a subdivision from 2×2 map to 3×3, the learning effect L is transmitted to the newly emerging neighbor vectors as defined in (1)–(5): $L/2$ is transmitted to the just adjacent two vectors out of five new vectors and $L/4$ to the center vector and none to the other two far vectors, accordingly. Magnitude of the transmitted learning effect is expressed by the darkness of the square in Fig. 3. When the map is subdivided into 5×5, it is observed that the closer to $W_{(M,M)}$, the more strongly the weight vectors are affected by the learning effect. In a real application of the proposed algorithm, training is iterated $T(p)$ times at each subdivision stage. In the course of that, each weight vector is affected by two factors : the training in T times' learning and/or the transmission occuring during $(p-1)$ times' subdivisions. Though the values of weight vectors fluctuate multiple times in the course of training and transmissions, all the effects on weight vectors can be simply treated as a vectorial sum. Thus, the proposed algorithm can work effectively without the neighborhood function.

V. Conclusion

In this letter, a new SOM algorithm with computational cost $O(\log_2 M)$ is proposed. The proposed algorithm eliminates the time-consuming parameter tuning in neighborhood function in SOM applications. The effectiveness of the proposed algorithm was justified by simulation of computational costs. The results of the computation time were examined according to the orders of computational costs. The mapping results of the codon frequencies of *E. coli* K12 genes with 33×33 feature map showed the proposed algorithm working effectively.

References

[1] T. Kohonen, *Self-Organizing Maps*, 2nd ed. New York: Springer-Verlag, 1997.

[2] I. Lapidot (Voitovetsky), H. Guterman, and A. Cohen, "Unsupervised speaker recognition based on competition between self-organizing maps," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 877–887, Jul. 2002.

[3] A. Plebe and A. M. Anile, "A neural-network-based approach to the double traveling salesman problem," *Neural Comput.*, pp. 437–471, Feb. 2001.
[4] S. Kaski, J. Sinkkonen, and J. Peltonen, "Bankruptcy analysis with self-organizing maps in learning metrics," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 936–947, Jul. 2001.
[5] C.-H. Chang, P. Xu, R. Xiao, and T. Srikanthan, "New adaptive color quantization method based on self-organizing maps," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 237–249, Jan. 2005.
[6] G. A. Barreto and A. F. R. Araújo, "Identification and control of dynamical systems using the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1244–1259, Sep. 2004.
[7] G. Dong and M. Xie, "Color clustering and learning for image segmentation based on neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 925–936, Jul. 2005.
[8] H.-C. Wang, J. Badger, P. Kearney, and M. Li, "Analysis of codon usage patterns of bacterial genomes using the self-organizing map," *Mol. Biol. Evol.*, pp. 792–800, May 2001.
[9] M. Sultan, D. A. Wigle, C. A. Cumbaa, M. Maziarz, J. Glasgow, M. S. Tsao, and I. Jurisica, "Binary tree-structured vector quantization approach to clustering visualizing microarray data," *Bioinf.*, vol. 18, no. 1, pp. S111–S119, 2002.
[10] R. T. Gill, E. Katsuoulakis, W. Schmitt, G. T. Oldenburg, J. Misra, and G. Stephanopoulos, "Genome-wide dynamic transcriptional profiling of the light-to-dark transition in synechocystis sp. strain PCC 6803," *J. Bacteriol.*, pp. 3671–3681, Jul. 2002.
[11] P. Koikkalainen and E. Oja, "Self-organizing hierarchical feature maps," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 1990, pp. 279–284.
[12] K. K. Truong, "Multilayer Kohonen image codebooks with a logarithmic search complexity," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, May 1991, pp. 2789–2792.
[13] P. Xu and C.-H. Chang, "Self-organizing topological tree," in *Proc. Int. Symp. Circuits Syst. (ISCAS)*, 2004, pp. 732–735.
[14] P. M. Sharp and W.-H. Li, "Codon usage in regulatory genes in Escherichia coli does not reflect selection for 'rare' codons," *Nucleic Acids Res.*, vol. 14, no. 19, pp. 7737–7749, 1986.
[15] M. Riley, "Functions of the gene products of *Escherichia coli*," *Microbiol. Rev.*, vol. 57, no. 4, pp. 862–952, Dec. 1993.

A Cross-Layer Adaptation Scheme for Improving IEEE 802.11e QoS by Learning

Chiapin Wang, Po-Chiang Lin, and Tsungnan Lin

***Abstract*—In this letter, we propose a cross-layer adaptation scheme which improves IEEE 802.11e quality of service (QoS) by online adapting multidimensional medium access control (MAC)-layer parameters depending on the application-layer QoS requirements and physical layer (PHY) channel conditions. Our solution is based on an optimization approach which utilizes neural networks (NNs) to learn the cross-layer function. Simulations results demonstrate the effectiveness of our adaptation scheme.**

***Index Terms*—Adaptive algorithm, IEEE 802.11e wireless local area networks (WLAN), neural networks (NNs), quality of service (QoS).**

I. Introduction

With the popularity of IEEE 802.11-based wireless local area networks (WLAN) which are capable of providing high data-rate wireless access, the demands of multimedia services are increasing for mobile users. To support quality of service (QoS) for multimedia applications in the contention-based part of 802.11 medium access control (MAC), the IEEE 802.11 standardization committee just finished a service differentiation scheme, called enhanced distributed coordination function (EDCF) [1]. It grants the higher class traffics such as voice and video traffic to access the wireless medium early in most cases by differentiating interframe space (IFS) and backoff parameters at MAC layer with up to eight priorities, which are also known as traffic categories (TC) [2]. Although this mechanism can improve QoS of real-time traffic, the performance obtained is not optimal since the fixed EDCF parameters cannot be adaptive to the variation of communication circumstances such as traffic characteristics and load conditions.

There have been several works about improving IEEE 802.11e QoS [2]–[4] by optimizing EDCF parameters based on traffic types or load situations. Xiao [3] developed an analytical model of 802.11 EDCF and proposed a backoff-based priority scheme for real-time services. Tinnirello *et al.* [4] used simulations to investigate the behavior of differentiating EDCF parameters under various conditions of traffic loads and proposed a differentiation scheme based on a joined use of minimum contention window and IFS. However, most of these works provide solutions with the assumption of ideal channel conditions or homogeneous link qualities among the participating hosts which is impractical in realistic wireless environments. The transmission qualities of hosts, e.g., bit-error rates (BER), actually are unequal at most of the times even with a link adaptation mechanism applied on 802.11 physical layer (PHY) [1] due to limited modulation and coding schemes (MCS) available. Under heterogeneous channel conditions, the EDCF parameters determined with these schemes may be no more effective to provide differentiated QoS optimally. For example, we consider a simple transmission scenario of one real-time traffic flow and one best-effort flow in the network. In case the two flows are with

Manuscript received May 29, 2006; revised July 1, 2006. This work was supported in part by Yulong Corporation under Grant 95-S-C01A and by Taiwan National Science Council under Grants 95-2219-E-002-018 and 95-2221-E-002-190.

C. Wang and P.-C. Lin are with Graduate Institute of Communication Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C.

T. Lin is with Graduate Institute of Communication Engineering and the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: tsungnan@ntu.edu.tw).

Digital Object Identifier 10.1109/TNN.2006.883014