Cogn Neurodyn (2010) 4:295–313
DOI 10.1007/s11571-010-9110-4

RESEARCH ARTICLE

Functional model of biological neural networks

James Ting-Ho Lo

Received: 18 September 2009 / Revised: 20 March 2010 / Accepted: 29 March 2010 / Published online: 20 April 2010

Abstract A functional model of biological neural networks, called temporal hierarchical probabilistic associative memory (THPAM), is proposed in this paper. THPAM comprises functional models of dendritic trees for encoding inputs to neurons, a first type of neuron for generating spike trains, a second type of neuron for generating graded signals to modulate neurons of the first type, supervised and unsupervised Hebbian learning mechanisms for easy learning and retrieving, an arrangement of dendritic trees for maximizing generalization, hardwiring for rotation-translation-scaling invariance, and feedback connections with different delay durations for neurons to make full use of present and past informations generated by neurons in the same and higher layers. These functional models and their processing operations have many functions of biological neural networks that have not been achieved by other models in the open literature and provide logically coherent answers to many long-standing neuroscientific questions. However, biological justifications of these functional models and their processing operations are required for THPAM to qualify as a macroscopic model (or low-order approximate) of biological neural networks.

Keywords Neuron model · Hebb learning · Spike train · Unsupervised learning · Dendritic tree model

Abbreviations

ECM	Expanded correlation matrix
FSI	Feature subvector index
GECM	General expanded correlation matrix
GOE	General orthogonal expansion
NXOR	Not-exclusive-or
OE	Orthogonal expansion
PU	Processing unit
PU(**n**)	Processing unit on feature subvector index **n**
RTS	Rotation, translation and scaling
SPD	Subjective probability distribution
THPAM	Temporal hierarchical probabilistic associative memory
XOR	Exclusive-or

J. T.-H. Lo (✉)
Department of Mathematics and Statistics, University of Maryland Baltimore County, Baltimore, MD 21250, USA
e-mail: jameslo@umbc.edu

Introduction

Biological neural networks are known to have such structures as hierarchical networks with feedbacks, neurons, denritic trees and synapses; and perform such functions as supervised and unsupervised Hebbian learning, storing knowledge in synapses, encoding information by dendritic trees, and detecting and recognizing spatial and temporal multiple/hierarchical causes. However, descriptions of these structures and functions are mostly fragmental and sometimes controversial in the literature on neuroscience (Arbib 2003; Dayan and Abbott 2001; Kandel et al. 2000; Koch 1999; Levitan and Kaczmarek 1993; Stuart et al. 2008) (Two examples related with this paper are logic gates vs. low-order polynomials in dendritic processing (Mel 1994), Hebbian vs. not Hebbian in learning (Mel 2002)) and on artificial neural networks (Bishop 2006; Dayan and Abbott 2001; Hawkins 2004; Hecht-Nielsen 2007; Hecht-Nielsen and McKenna 2003; Principe et al. 2000; Rieke et al. 1999; O'Reilly and Munakata 2000; Hassoun 1993; Hinton and Anderson 1989; Kohonen

1988). A single mathematical model that provides an integration of these structures and functions, and explains how the structures interact to perform the functions may shed some light to what processing operations might be required for each structure and function, suggest corresponding experiments to perform, and thereby enhance understanding of biological neural networks as systems whole. In fact, neuroscientists have long hypothesized a common cortical algorithm, and researchers on artificial neural networks have long searched for an ideal learning machine that learns and retrieves easily, detects and recognizes multiple temporal and spatial causes, and generalizes adequately on noisy, distorted, occluded, rotated, translated and scaled patterns. A common cortical algorithm and, more often than not, an ideal learning machine is intended to be a single mathematical model. To the best of this author's knowledge, the former was first mentioned by Vernon Mountcastle (1978) and the latter was first suggested by John von Neumann (1958).

This paper is intended to provide such a mathematical model. The model, called temporal hierarchical probabilistic associative memory (THPAM), comprises novel models of dendritic trees; neurons communicating with spike trains; a mechanism for unsupervised and supervised learning; a structure for detecting and recognizing noised, distorted and occluded patterns; hard-wiring for detecting and recognizing rotated, translated and scaled patterns; and feedback neural fibers for processing temporal data. Although biological justifications of these models have not been established, these models are logically coherent and integrate into the model, THPAM, of biological neural networks. Before the biological justifications are obained, THPAM can only be termed a functional model rather than a macroscopic model.

Derivation of THPAM is guided by the following four neurobiological postulates:

1. The biological neural networks are recurrent multilayer networks of neurons.
2. Most neurons output a spike train.
3. Knowledge is stored in the synapses between neurons.
4. Synaptic strengths are adjusted by a version of the Hebb rule of learning. (In his 1949 book, *The Organization of Behavior*, Donald Hebb posited: "When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell." A natural extension of this (alluded to by Hebb as the decay of unused connections) is to decrease the synaptic strength when the source and target neurons are not active at the same time.) [http://www.en.wikipedia.org/wiki/Hebbian_theory].

As a matter of fact, in the development of artificial neural networks, Postulates 1 and 3 led to multilayer perceptrons and recurrent multilayer perceptrons (Rieke et al. 1999; O'Reilly and Munakata 2000; Dayan and Abbott 2001; Hecht-Nielsen and McKenna 2003; Hawkins 2004; Hecht-Nielsen 2007; Principe et al. 2000; Bishop 2006; Haykin 2009), and Postulates 3 and 4 led to associative memories (Kohonen 1988; Hinton and Anderson 1989; Hassoun 1993). However, multilayer perceptrons exclude Postulates 2 and 4; and associative memories exclude Postulate 1. As useful as multilayer perceptrons and associative memories are in engineering, they have limited capabilities and offer little insight into the inner workings of biological neural networks.

The construction of a functional model of biological neural networks based on all the four postulates has broken the barriers confining the multilayer perceptrons and the associative memories. A first contribution of this paper lies in each of the following features of THPAM (temporal hierarchical probabilistic associative memory) that such existing models as the recurrent multilayer perceptron and associative memories do not have:

1. a recurrent multilayer network learning by the Hebb rule;
2. fully automated unsupervised and supervised Hebbian learning mechanisms (involving no differentiation, error backpropagation, optimization, iteration, cycling repeatedly through all learning data, or waiting for asymptotic behavior to emerge);
3. dendritic trees encoding inputs to neurons;
4. neurons communicating with spike trains carrying subjective probability distributions;
5. masking matrices facilitating recognition of corrupted, distorted, and occluded patterns; and
6. feedbacks with different delay durations for fully utilizing temporally and spatially associated information.

A second contribution of this paper lies in the integration of not only the above unique features but also the following additional features in a single model of biological neural networks:

1. detecting and recognizing multiple/hierarchical causes; and
2. hard-wired learning for detecting and recognizing rotated, translated and scaled patterns.

A third contribution of this paper is providing logically coherent answers jointly to the following long-standing questions by using a single functional model of biological neural networks:

1. What is the information that neurons communicate by spike trains?
2. How do spike trains carry this information?

3. In what form is this information stored in the synapses? How are the synapses updated to learn this information in supervised learning and unsupervised learning in accordance with the Hebb rule of learning?
4. How is this information stored in synapses retrieved and converted into spike trains?
5. How does a biological neural network generalize on corrupted, distorted or occluded input?
6. What enables a biological neural network to recognize rotated, translated or scaled patterns?
7. How do the spike generation and travel times affect the network processing?
8. What are the functions of dendritic nodes and trees? How are dendritic nodes connected into trees to perform their function?

However, we note that even if all its component models are biologically justified, THPAM is only a "first-order approximate" of biological neural networks, which is not intended to explain all the biological structures and phenomena observed in biological neural networks. Some biological structures or phenomena can undoubtedly be found that are seemingly or apparently missing in THPAM. Nevertheless, the 16 numbered list items above are like 16 pieces of a jigsaw puzzle. The fact that they fit together nicely into one piece whole for the first time indicates that THPAM is worth pursuing further as a candidate macroscopic model.

The components and processing operations in THPAM can be viewed as hypotheses about the macroscopic properties of biological neural networks. Some issues that need to be resolved to biologically justify or dismiss these hypotheses are mentioned in this paper. In recent years, we have seen rapid progress in technology for measuring dendritic, synaptic and neuronal quantities, and we expect to see more. It is hoped that those outstanding issues will soon be resolved in one way or another. (It may be appropriate to recall that when it was first published, the special theory of relativity was not more than a set of logically coherent mathematical results. The claims of bizarre space/time relativity and mass-energy conversion had not even been thought of, much less experimentally confirmed. Yet the theory led to experiments, and the bizarre space/time relativity and mass-energy conversion were eventual proven to be true.)

A current major research thrust on learning machines is the development of those with a deep architecture such as the convolutional networks (LeCun et al. 1989, 1998; Simard et al. 2003), deep belief networks (Hinton et al. 2006; Hinton and Salakhutdinov 2006) and deep Boltzmann machines (Salakhutdinov and Hinton 2009). Better versions and good understandings have been reported in (Bengio and LeCun 2007; Ranzato et al. 2007; Bengio et al. 2007; Desjardins and Bengio 2008; Erhan et al. 2010). The deep belief networks and their improved version, the deep Boltzmann machines, can learn without a supervisor by a ingenious technique called greedy layer-wise learning strategy. The convolutional networks capture the spatial topology of the input images and can recognize translated patterns very well. All these deep learning machines have good generalization capabilities for recognizing distorted, rotated and translated patterns. On the well-known and widely used "MNIST Database" of handwritten digits, the deep Boltzmann machine achieved an error rate of 0.95% in recognizing handwritten digits (without using training tricks such as supplementing the data set with lightly transformed versions of the training data) (Salakhutdinov and Hinton 2009). After about 20 years of evolution, the convolutional networks' latest version, "LeNet-6+ unsupervised training," achieved a recognition error of 0.39% on the same "MNIST Database" (Bengio and LeCun 2007). Performances of these deep learning machines are expected to continue improving even further, especially when feedback structures are added in them.

To appreciate these performances, we note that the error rate of human performance in recognizing handwritten numerals is 1.56% at about 1 digit per second (Wilkinson et al. 1992) and 0.91% (0.56% rejection and 0.35% error) in two rounds with no time limit in the second round (Geist et al. 1994). Nevertheless, none of the deep learning machines existing in the open literature has any of the first seven features listed above.

Several models of cortical circuits, which attempt to integrate neurobiological findings into a model of the cortex, have been reported (Martin 2002; Granger 2006; Grossberg 2007; George and Hawkins 2009). Martin (2002) states: "It is clear that we simply do not understand much of the detailed structure of cortical microcircuits or their relation to functions." (Granger 2006) provides a computational instruction set to establish a unified formalism for describing human faculties ranging from perception and learning to reasoning and language. Grossberg (2007) explains how laminar neocortical circuits, which embody two computational paradigms—complementary computing and laminar computing, give rise to biological intelligence. George and Hawkins (2009) describes how Bayesian belief propagation in a spatio-temporal hierarchical model can lead to a mathematical model for cortical circuits. The models of cortical circuits in (Granger 2006; George and Hawkins 2009) exhibit interesting pattern recognition capabilities in certain numerical examples. However, they have not been tested or compared with learning machines on those widely used databases. Granger (2006), Grossberg (2007) contain no numerical results. None of the models of cortical circuits has any of the first six features listed above.

A brief summary of the results on THPAM together with the organization of this paper follows: THPAM can be viewed as an organization of a biological neural network into a recurrent multilayer network of processing units (PUs). Section "A recurrent multilayer network of processing units" briefly describes this network and establishes notations of the inputs and outputs of PUs in the network.

The first two questions are answered in "Information carried by spike trains". It is argued that the ideal informations for neurons to communicate are the subjective probability distributions (SPDs) of the labels of patterns appearing in the receptive domains of neurons. Therefore, we hypothesize that the SPDs are said ideal informations to facilitate our derivation of THPAM. The resulting integrity of THPAM and impossibility to replace SPDs suggests that the hypothesis is likely to be valid. It is further argued that under the four postulates, the SPD is the average frequency of the spikes in the spike train.

A processing unit (PU) is a two-layer pattern recognizer, which learns and generates aforementioned SPDs. To achieve these, the PU uses not-exclusive-or (NXOR) logic gates to transform its input vectors into general orthogonal expansions (GOEs). GOEs are described in "Orthogonal expansion". The transformation by a large number of NXOR gates can be looked upon as a functional model of the dendritic trees of the neurons in the PU.

By a crude version of the Hebb rule, outer products of the GOEs (general orthogonal expansions) and their respective labels are accumulated to form general expansion correlation matrices (GECMs), which are the synaptic strengths stored in the PU (processing unit). GECMs are discussed in "Expansion correlation matrices". Simple multiplication of the GECMs and the GOE of the input pattern and simple manipulation of the resultant products yield the SPD of the label of the input pattern. This generation of SPDs together with an example is given in "Representations of probability distributions".

Each processing unit (PU) uses a masking matrix to automatically select a maximum number of components of its input feature subvector that matches those of a stored feature subvector and determine the SPD of the label involved. The masking matrix and how it works is described in "Masking matrices". The masking matrix can be viewed as mathematical idealization and organization of a large number of overlapped and nested dendritic trees.

As shown in Fig. 2, a processing unit (PU) comprises an Orthogonal Expander, a label SPD (subjective probability distribution) Estimator, a Spike Generator, a GECMs (general expansion correlation matrices) Adjuster, and a storage of $C(\mathbf{n})$ and $D(\mathbf{n})$. A PU has essentially two functions, retrieving "point estimates" of the label of a feature subvector $x_\tau(\mathbf{n})$ from the memory (i.e., GECMs) and learning a feature subvector and its label that is either provided from outside the PU (in supervised learning) or generated by itself (in unsupervised learning). The structural diagram of an example PU is shown in Fig. 3. Both supervised and unsupervised learning by the PU follow a crude version of the Hebb rule. Spike trains generated by each PU facilitate unsupervised learning. This simple novel modeling of the Hebbian unsupervised learning in biological neural networks is a major underpin of THPAM as a functional model of biological neural networks. The PU and its functions of retrieving and learning are explained in "Processing units and supervised and unsupervised learning".

The brain is known to be able to recognize rotated, translated and scaled patterns. To achieve this, each PU in THPAM learns a rotation, translation and scaling suite of its input feature subvector. Such suites are described in "Learning to recognize rotated, translated or scaled patterns". Some translations, rotation, compression and expansion of an example pattern are shown in Fig. 4.

Spike trains propagated among the PUs are one of the postulates leading to THPAM. In "Spike trains for each exogenous feature vector", the necessity of spike trains for the foregoing parts of THPAM to work properly is discussed. So is how the spike trains are feedbacked with delays. Typical feedback connections with delays are shown in Fig. 5.

A recurrent multilayer network of processing units

The temporal hierarchical probabilistic associative memory (THPAM) can be looked upon as an organization of a biological neural network into a recurrent hierarchical network of PUs (processing units). Each PU is a pattern recognizer that comprises dendritic trees, neurons of two types, synaptic weights, and a learning mechanism for updating these synaptic weights by a version of the Hebb rule in unsupervised or supervised learning.

Spike trains propagating through biological neural networks are assumed to be sequences of unipolar binary numbers, 0's and 1's. A group of M spike trains can be viewed as a sequence of M-dimensional unipolar binary vectors, v_t, $t = 1, 2,\ldots,$ where $v_t = [v_{t1} \quad v_{t2} \quad \ldots \quad v_{tM}]'$. In this paper, we convert v_t, $t = 1, 2,\ldots,$ into a sequence of M-dimensional bipolar binary vectors, x_t, $t = 1, 2,\ldots,$ by $x_{tm} = 2v_{tm} - 1$, for $m = 1,\ldots, M$ and $t = 1, 2,\ldots.$ We will use x_t to simplify our description and discussion in this paper. Since x_t is only a mathematical representation of v_t, and x_{tm} can easily be converted back into v_{tm} by $v_{tm} = (x_{tm} + 1)/2$, we also call the components of x_t, $t = 1, 2,\ldots,$ spike trains with the understanding that they are mathematical representations of the "biological" spike trains, v_t, $t = 1, 2,\ldots.$

A vector input to THPAM is called an exogenous feature vector, and a vector input to a layer of PUs is called a feature vector. A feature vector input to a layer usually contains not only feedforwarded outputs from a preceding layer but also feedbacked outputs from the same or higher layers with a time delay. A feature vector may contain components from an exogenous feature vector. For simplicity, we assume that the exogenous feature vector is only input to layer 1 and is thus a subvector of a feature vector input to layer 1. These vectors over time form groups of spike trains.

A subvector of a feature vector that is input to a PU is called a feature subvector. Trace the feedforward connections backward from neurons of a PU to a subvector of the exogenous feature vector. This feature subvector is called the receptive domain of the PU. The collection of neurons in layer $l-1$ that have a feedforward connection to a neuron in a PU in layer l and the delay devices that hold a feedback for direct input to the same PU in layer l are called the immediate receptive domain of the PU.

The feature vector input to layer l at time or numbering t is denoted by x_t^{l-1}, and the output from the layer at t is denoted by $x\{y_t^l\}$, where y_t^l and $x\{y_t^l\}$ are specified in more detail later in this section. An exogenous feature vector is denoted by x_t^{ex}. It is a subvector of x_t^0, which may contain feedbacked components. For notational simplicity, the superscript $l-1$ in x_t^{l-1} and dependencies on $l-1$ or l in other symbols are sometimes suppressed in the following when no confusion is expected.

Let x_t, $t = 1, 2,\ldots$, denote a sequence of M-dimensional feature vectors $x_t = [x_{t1} \quad \ldots \quad x_{tM}]'$, whose components are ternary numbers. Let $\mathbf{n} = [\mathbf{n}_1 \quad \ldots \quad \mathbf{n}_k]'$ be a subvector $[1 \quad \ldots \quad M]'$ such that $\mathbf{n}_1 < \cdots < \mathbf{n}_k$. The subvector $x_t(\mathbf{n}) := [x_{t\mathbf{n}_1} \quad \ldots \quad x_{t\mathbf{n}_k}]'$ of x_t is a feature subvector of the feature vector x_t. $\mathbf{n}$ is called a feature subvector index (FSI), and $x_t(\mathbf{n})$ is said to be a feature subvector on the FSI $\mathbf{n}$ or have the FSI $\mathbf{n}$. Each PU is associated with a fixed FSI $\mathbf{n}$ and denoted by PU $(\mathbf{n})$. Using these notations, the sequence of subvectors of x_t, $t = 1, 2,\ldots$, that is input to PU($\mathbf{n}$) is $x_t(\mathbf{n})$, $t = 1, 2,\ldots$. The FSI $\mathbf{n}$ of a PU usually has subvectors, $\mathbf{n}(u)$, $u = 1,\ldots, U$, on which subvectors $x_t(\mathbf{n}(u))$ of $x_t(\mathbf{n})$ are separately processed by PU($\mathbf{n}$) at first. The subvectors, $\mathbf{n}(u)$, $u = 1,\ldots, U$, are not necessarily disjoint, but all inclusive in the sense that every component of $\mathbf{n}$ is included in at least one of the subvectors $\mathbf{n}(u)$. Moreover, the components of $\mathbf{n}(u)$ are usually randomly selected from those of $\mathbf{n}$.

The PUs in layer l have FSIs (feature subvector indices) denoted by $\mathbf{1}^l, \mathbf{2}^l,\ldots,\mathbf{N}^l$. Upon receiving a feature vector x_τ^{l-1} by layer l, the feature subvectors, $x_\tau^{l-1}(\mathbf{1}^l)$, $x_\tau^{l-1}(\mathbf{2}^l),\ldots,x_\tau^{l-1}(\mathbf{N}^l)$, are formed and processed by the PUs, PU$(\mathbf{1}^l)$, PU$(\mathbf{2}^l)$,…,PU$(\mathbf{N}^l)$, to compute $y_\tau^l(\mathbf{1}^l)$, $y_\tau^l(\mathbf{2}^l),\ldots,y_\tau^l(\mathbf{N}^l)$ first and then generate $x\{y_\tau^l(\mathbf{1}^l)\}$, $x\{y_\tau^l(\mathbf{2}^l)\},\ldots,x\{y_\tau^l(\mathbf{N}^l)\}$, respectively. Here $y_\tau^l(\mathbf{n}^l)$ denotes a representation of the subjective probability of the label $r_\tau^{l-1}(\mathbf{n}^l)$ of $x_\tau^{l-1}(\mathbf{n}^l)$, and $x\{y_\tau^l(\mathbf{n}^l)\}$ denotes the output of PU$(\mathbf{n}^l)$ based on $y_\tau^l(\mathbf{n}^l)$. These representations and outputs are grouped into the representation y_τ^l of subjective probabilities and the output vector $x\{y_\tau^l\}$ of layer l as follows:

$$y_\tau^l = [y_\tau^{l\prime}(\mathbf{1}^l) \quad y_\tau^{l\prime}(\mathbf{2}^l) \quad \ldots \quad y_\tau^{l\prime}(\mathbf{N}^l)]'$$

$$x\{y_\tau^l\} = [x'\{y_\tau^l(\mathbf{1}^l)\} \quad x'\{y_\tau^l(\mathbf{2}^l)\} \quad \ldots \quad x'\{y_\tau^l(\mathbf{N}^l)\}]'$$

The components of a feature vector x_τ^{l-1} input to layer l at time (or with numbering) τ comprise components of ternary vectors generated by PUs in layer $l-1$ and those generated at a previous time by PUs in the same layer l or PUs in higher layers with layer numberings $l+k$ for some positive integers k. The time delays may be of different durations.

Once an exogenous feature vector is received by THPAM, the PUs perform functions of retrieving and/or learning from layer to layer starting with layer 1, the lowest layer. After the PUs in the highest layer, layer L, complete performing their functions, THPAM is said to have completed one round of retrievings and/or learnings (or memory adjustments). For each exogenous feature vector, THPAM will continue to complete a certain number of rounds of retrievings and/or learnings.

We note that retrieving and learning by a PU are performed locally, meaning that only the feature subvector input to the PU and its label are involved in the processing by the PU. In "Orthogonal expansion, Expansion correlation matrices, Representations of probability distributions, 7, 8 and Learning to recognize rotated, translated or scaled patterns", the subscripts t and τ denote the time or numbering of a feature vector or subvector that is input to a layer or a PU, whereas, in "Spike trains for each exogenous feature vector", they denote the time or numbering of an exogenous vector that is input to THPAM.

Information carried by spike trains

Since the immediate receptive domain (defined in Section "A recurrent multilayer network of processing units") of a PU may be shared by more than one cause (or pattern) or may contain parts from more than one cause, and may contain corruption, distortion, occludion or noise from the PU's receptive domain (defined in Section "A recurrent multilayer network of processing units") or the sensor measurements, image pixels, or sound recordings that are transformed into the receptive domain; the PU's immediate receptive domain can completely be described or represented only by a probability distribution (or a relative frequency distribution). Therefore, probability distributions are the most desirable information for the PUs to communicate among them. Since probability distributions can be learned by the PU only from "experiences," they must be subjective probability distributions (SPDs). As will be seen in "Representations of probability distributions", SPDs of the labels of a PU's immediate receptive domain can be generated by the PU.

There are three possible ways spike trains can carry an SPD: (1) The SPD is carried by the shapes of the spikes. (2) The spike trains at an instant of time form a binary representation of the SPD. (3) The SPD is represented by the frequencies of spikes in spike trains.

Shapes of the spikes cannot be learned by the Hebb rule. Besides, SPDs output from a layer of PU are input to the next layer of PUs. In the process of learning, such SPDs for causes or patterns change. The learning and retrieving mechanisms of PUs must be able to tolerate such changes. It is not clear how changes in spike shapes can be tolerated. Hence, way (1) is ruled out for PUs in THPAM.

In way (2), each SPD is represented by a certain number of bits (or tets), the number depending on the level of accuracy required. The higher the accuracy level, the larger the dimensionality of the output vector of the processing unit.

Again, in the process of learning, the SPD for a certain cause changes. For the learning and retrieving mechanisms to tolerate changes in the codes for the SPDs, the codes must vary gradually as the SPD changes gradually. Such codes are known to consist of large numbers of bits, requiring a large dimensionality of the output vector of the PU. Furthermore, it is not clear how unsupervised learning can be performed with such binary codes by the Hebb rule. For instance, when feature subvectors (or their variants) that have not been learned are input to a PU, the SPDs output by the PU are the same, namely the uniform distribution, which is therefore represented by the same binary code and giving all such input vectors the same label in Hebbian learning, failing to learn to distinguish different feature subvectors without supervision.

Way (3) can be easily obtained by using a pseudo-random number generator to convert a subjective probability into a $+1$ spike with said subjective probability and a mathematical -1 spike (i.e., biological 0) otherwise. Using this representation, the rate of $+1$ spikes in a spike train is on the average the subjective probability generated by the PU that outputs the spike train. The dimensionality of this representation is the dimensionality of the label. At any instant of time, the spike trains output by the processing units in a layer form an image of $+1$, -1 and 0 (for simplicity in certain cases). Gradual change in the subjective probabilities changes the distribution of the ternary digits in the image gradually, which can be tolerated by the use of the masking matrices to be described in "Masking matrices. When a feature subvector (or a variant thereof) that has never been learned is input to a PU, a random label is assigned to the vector. Different new input feature subvectors are usually assigned different labels in unsupervised learning. Occasional coincidences of different feature subvectors assigned with the same label do not cause a problem, if the label is used as part of a feature subvector input to a higher-layer PU. For example, "loo" as a part of "look," "loot," and "loop" does not cause confusion.

Therefore, under the four postulates, the subjective probability of a component of a label being $+1$ is represented by the average spike rate of a spike train. It follows that if the dimensionality of the label is at most R, R neurons form a group whose R spike trains carry the SPD of the label.

Orthogonal expansion

As discussed in the preceding section, SPDs (subjective probability distributions) are the most desirable information for PUs to communicate among themselves. Can SPDs be learned and retrieved by PUs under the four postulates? Subjective probabilities are relative frequencies. We need to find out whether and how relative frequencies can be learned and retrieved.

Orthogonal expansion of ternary vectors from the coding theory (Slepian 1956) plays an important role in learning and retrieving of the relative frequencies. The following example motivates and explains the definition of orthogonal expansion of bipolar vectors.

Example 1 Given 2-dimensional bipolar vectors, $a = [a_1 \quad a_2]'$ and $b = [b_1 \quad b_2]'$, let

$$\breve{a} = [1 \quad a_1 \quad a_2 \quad a_2 a_1]'$$
$$\breve{b} = [1 \quad b_1 \quad b_2 \quad b_2 b_1]'$$

By simple algebra, $\breve{a}'\breve{b} = 1 + a_1 b_1 + a_2 b_2 + a_2 b_2 a_1 b_1 = (1 + a_1 b_1)(1 + a_2 b_2)$. It follows that $\breve{a}'\breve{b} = 1$, if $a = b$; and $\breve{a}'\breve{b} = 0$, if $a \neq b$. $\breve{a}$ and $\breve{b}$ are therefore called orthogonal expansions of a and b. Generalizing this idea of orthogonal expansion yields the following definition.

Definition Given an m-dimensional ternary vector $v = [v_1 \quad \ldots \quad v_m]'$, define $\breve{v}$ recursively by

$$\begin{aligned} \breve{v}(1) &= [1 \quad v_1]' \\ \breve{v}(1,\ldots,j+1) &= [\breve{v}'(1,\ldots,j) \quad v_{j+1}\breve{v}'(1,\ldots,j)]' \\ &\quad \text{for } j = 1,\ldots,m-1 \\ \breve{v} &= \breve{v}(1,\ldots,m) \end{aligned} \tag{1}$$

$\breve{v}$ is called the orthogonal expansion of v.

The above definition is justified by the following theorem.

Theorem 1 Let $a = [a_1 \quad \dots \quad a_m]'$ and $b = [b_1 \quad \dots \quad b_m]'$ be two m-dimensional ternary vectors. Then the inner product $\breve{a}'\breve{b}$ of their orthogonal expansions, $\breve{a}$ and $\breve{b}$, can be expressed as follows:

$$\breve{a}'\breve{b} = \prod_{j=1}^{m}(1 + a_j b_j) \tag{2}$$

which have the following properties:

1. If $a_k b_k = -1$ for some $k \in \{1,\dots, m\}$, then $\breve{a}'\breve{b} = 0$.
2. If $a_k b_k = 0$ for some $k \in \{1,\dots, m\}$, then $\breve{a}'\breve{b} = \prod_{j=1, j\neq k}^{m} (1 + a_j b_j)$.
3. If $\breve{a}'\breve{b} \neq 0$, then $\breve{a}'\breve{b} = 2^{a'b}$.
4. If a and b are bipolar vectors, then $\breve{a}'\breve{b} = 0$ if $a \neq b$; and $\breve{a}'\breve{b} = 2^m$ if $a = b$.

Proof Applying the recursive formula (1), we obtain

$$\begin{aligned}
&\breve{a}'(1,\dots,j+1)\breve{b}(1,\dots,j+1)\\
&\quad = [\breve{a}'(1,\dots,j) \quad a_{j+1}\breve{a}'(1,\dots,j)]\,[\breve{b}'(1,\dots,j) \quad b_{j+1}\breve{b}'(1,\dots,j)]'\\
&\quad = \breve{a}'(1,\dots,j)\breve{b}(1,\dots,j) + a_{j+1}b_{j+1}\breve{a}'(1,\dots,j)\breve{b}(1,\dots,j)\\
&\quad = \breve{a}'(1,\dots,j)\breve{b}(1,\dots,j)(1 + a_{j+1}b_{j+1})
\end{aligned}$$

The formula in (2) follows. The four properties above are easy consequences.

We remark that if some components of a are set equal to zero to obtain a vector c and the nonzero components of c are all equal to their corresponding components in b, then we still have $\breve{c}'\breve{b} \neq 0$. This property is used to construct masking matrices in "Masking matrices" for learning and recognizing corrupted, distorted and occluded patterns and for facilitating generalization on such patterns.

Expansion correlation matrices

In this section, it is shown how orthogonal expansions of subvectors of feature subvectors input to a PU are used to construct synaptic weights in the PU, and how such synaptic weights, in the form of matrices, are adjusted in the PU to learn feature subvectors.

Let the label of $x_t(\mathbf{n})$ be denoted by $r_t(\mathbf{n})$, which is an R-dimensional ternary vector. All subvectors, $x_t(\mathbf{n}(u))$, $u = 1,\dots, U$, of $x_t(\mathbf{n})$ share the same label $r_t(\mathbf{n})$. In supervised learning, $r_t(\mathbf{n})$ is provided from outside THPAM, and in unsupervised learning, $r_t(\mathbf{n})$ is generated internally in the PU itself.

The pairs $(x_t(\mathbf{n}(u)), r_t(\mathbf{n}))$, $t = 1, 2,\dots,$ are learned by the PU to form expansion correlation matrices (ECMs), $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$ on $\mathbf{n}(u)$. After the first T pairs are learned, these matrices are

$$D(\mathbf{n}(u)) = \Lambda \sum_{t=1}^{T} \Lambda^{T-t} r_t(\mathbf{n}) \breve{x}_t'(\mathbf{n}(u)) \tag{3}$$

$$C(\mathbf{n}(u)) = \Lambda \sum_{t=1}^{T} \Lambda^{T-t} \breve{x}_t'(\mathbf{n}(u)) \tag{4}$$

where $\breve{x}_t(\mathbf{n}(u))$ are orthogonal expansions of $x_t(\mathbf{n}(u))$, Λ is a scaling constant that is selected to keep all numbers involved in THPAM manageable, $\lambda^{T\text{-}t}I$ is a weight matrix, where I is the identity matrix, and $\lambda (0 < \lambda < 1)$ is a forgetting factor. Other matrix $W_t(\mathbf{n}(u), T)$ can be used as the weight matrix instead. Note that the matrix $C(\mathbf{n}(u))$ has only one row.

The ECMs, $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$, are adjusted as follows: If $r_t(\mathbf{n}) \neq 0$,

$$D(\mathbf{n}(u)) \leftarrow \Lambda D(\mathbf{n}(u)) + \Lambda r_t(\mathbf{n}) \breve{x}_t'(\mathbf{n}(u)) \tag{5}$$

$$C(\mathbf{n}(u)) \leftarrow \Lambda C(\mathbf{n}(u)) + \Lambda \breve{x}_t'(\mathbf{n}(u)) \tag{6}$$

If $r_t(\mathbf{n}) = 0$, then $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$ are unchanged. These update formulas are discussed in terms of supervised and unsupervised Hebbian learning in "Processing units and supervised and unsupervised learning". We note here that learning an input feature subvector using the above formulas is instantaneous. No differentiation, backpropagation, iteration, optimization, cycling repeatedly through training material or waiting for asymptotic convergence is required.

Orthogonal expansions (OEs) $\breve{x}_t(\mathbf{n}(u))$ and ECMs, $D(\mathbf{n}(u))$, $C(\mathbf{n}(u))$, $u = 1,\dots, U$, are assembled into a general orthogonal expansion (GOE) $\breve{x}_t(\mathbf{n})$ and general expansion correlation matrices (GECMs), $D(\mathbf{n})$ and $C(\mathbf{n})$, for PU($\mathbf{n}$) (the PU on the FSI $\mathbf{n}$) as follows:

$$\breve{x}_t(\mathbf{n}) = [\breve{x}_t'(\mathbf{n}(1)) \quad \breve{x}_t'(\mathbf{n}(2)) \quad \dots \quad \breve{x}_t'(\mathbf{n}(U))]' \tag{7}$$

$$D(\mathbf{n}) = [D(\mathbf{n}(1)) \quad D(\mathbf{n}(2)) \quad \dots \quad D(\mathbf{n}(U))] \tag{8}$$

$$C(\mathbf{n}) = [C(\mathbf{n}(1)) \quad C(\mathbf{n}(2)) \quad \dots \quad C(\mathbf{n}(U))] \tag{9}$$

Note that while $\dim \breve{x}_t(\mathbf{n}) = \sum_{u=1}^{U} 2^{\dim x_t(\mathbf{n}(u))}$, the dimensionality of the orthogonal expansion of $x_t(\mathbf{n})$ is $2^{\dim x_t(\mathbf{n})}$. The former can be made much smaller than the latter by setting $\dim x_t(\mathbf{n}(u))$ small. If the components of a subvector $\mathbf{n}(u)$ of the feature subvector index (FSI) $\mathbf{n}$ are selected from the FSI $\mathbf{n}$ at random, then the components of $x_t(\mathbf{n}(u))$ are a random sample of $x_t(\mathbf{n})$, and $x_t(\mathbf{n}(u))$ is a "lower-resolution" representation of $x_t(\mathbf{n})$. Hence, a sufficient number U of $x_t(\mathbf{n}(u))$, which may have common components, can sufficiently represent $x_t(\mathbf{n})$. Since subvectors $\mathbf{n}(u)$ are all inclusive, $x_t(\mathbf{n}) = x_\tau(\mathbf{n})$ if and only if $x_t(\mathbf{n}(u)) = x_\tau(\mathbf{n}(u))$ for $u = 1,\dots, U$. However, even if $x_t(\mathbf{n}) \neq x_\tau(\mathbf{n}), x_t(\mathbf{n}(u))$ may still be equal to $x_\tau(\mathbf{n}(u))$ for some values of u. Therefore, the use of the GOE $\breve{x}_t(\mathbf{n})$ has not only the advantage of having the much smaller dimensionality of $\breve{x}_t(\mathbf{n})$ and thereby the much smaller dimensionality of the GECMs, but also the advantage of helping enhance the generalization capability of the PU. This advantage is further discussed in "Masking matrices".

Note that the components of $\breve{x}_t(\mathbf{n}(u))$ are actually all the products that can be obtained from those of $x_t(\mathbf{n}(u))$. Each product is obtained by successive two-factor multiplications. For example,

$$1 = x_{ti}^2(\mathbf{n}(u))$$
$$x_{ti}(\mathbf{n}(u))x_{tj}(\mathbf{n}(u))x_{tk}(\mathbf{n}(u)) = x_{ti}(\mathbf{n}(u))\left[x_{tj}(\mathbf{n}(u))x_{tk}(\mathbf{n}(u))\right]$$

Because of commutativity and associativity of multiplication, the successive two-factor multiplication for a component of $x_t(\mathbf{n}(u))$ is not unique. Missing or repeating components in $\breve{x}_t(\mathbf{n}(u))$ in the GECMs, $D(\mathbf{n})$ and $C(\mathbf{n})$, or in the GOE of the input feature subvectors $\breve{x}_\tau(\mathbf{n}(u))$ cause only graceful degradation of subjective probability distribution representation $p_\tau(\mathbf{n})$.

Each two-factor multiplication can be looked upon as an NXOR operation on the two factors involed. Note that NXOR operations can be replaced with XOR operations without affecting the generation of subjective probability distribution representation $p_\tau(\mathbf{n})$. XOR gates were found in dendritic trees by Zador et al. (1992), Fromherz and Gaede (1993), and the existence of logic gates and low-order polynomials in dendritic trees were discussed in Mel (1994).

Representations of probability distributions

How the expansion correlation matrices are used to generate representations of SPDs (subjective probability distributions) is shown in this section. The following example illustrates the idea.

Example 2 Given two different feature subvectors, $u = [u_1 \quad u_2]'$ and $v = [v_1 \quad v_2]'$, which are 2-dimensional bipolar vectors. Then, $\breve{u}'\breve{u} = 4$, $\breve{u}'\breve{v} = \breve{v}'\breve{u} = 0$, and $\breve{v}'\breve{v} = 4$. Let a training data set consists of 8 copies of u with label $+1$ and 2 copies of u with label -1; and 3 copies of v with label $+1$ and 27 copies of v with label -1. This training data set is learned by a PU with $\Lambda = \Lambda = 1$ [in (3) and (4)] to form the GECMs (general expansion correlation matrices) with $U = 1$:

$$D = (8-2)\breve{u}' + (3-27)\breve{v}'$$
$$C = (8+2)\breve{u}' + (3+27)\breve{v}'$$

By simple algebra, $D\breve{u} = 6(4)$, $C\breve{u} = 10(4)$, $D\breve{v} = -24(4)$, $C\breve{v} = 30(4)$. It follows that $(D\breve{u} + C\breve{u})/(2C\breve{u})$ = 8/10 is the relative frequency that u has been learned with label $+1$ by the PU; and $1 - (D\breve{u} + C\breve{u})/(2C\breve{u}) = 2/10$ is the relative frequency that u has been learned with label -1 by the PU. Similary, $(D\breve{v} + C\breve{v})/(2C\breve{v}) = 3/30$ is the relative frequency that v has been learned with label $+1$; and $1 - (D\breve{v} + C\breve{v})/(2C\breve{v}) = 27/30$ is the relative frequency that v has been learned with label -1.

We now generalize the idea illustrated in Example 2 in the following. Let us first define the symbols $d_\tau(\mathbf{n}(u))$, $c_\tau(\mathbf{n}(u))$, $a_\tau(\mathbf{n}(u))$:

$$d_t(\mathbf{n}(u)) := D(\mathbf{n}(u))\breve{x}_t(\mathbf{n}(u)) \tag{10}$$

$$c_t(\mathbf{n}(u)) := C(\mathbf{n}(u))\breve{x}_t(\mathbf{n}(u)) \tag{11}$$

$$a_t(\mathbf{n}(u)) := \frac{1}{2}(\mathbf{I}C(\mathbf{n}(u)) + D(\mathbf{n}(u)))\breve{x}_t(\mathbf{n}(u)) \tag{12}$$

and the symbols $d_\tau(\mathbf{n})$, $c_\tau(\mathbf{n})$, $a_\tau(\mathbf{n})$:

$$\begin{aligned} d_\tau(\mathbf{n}) &:= D(\mathbf{n})\breve{x}_\tau(\mathbf{n}) = \sum_{u=1}^{U} D(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) \\ &= \sum_{u=1}^{U} d_\tau(\mathbf{n}(u)) \end{aligned} \tag{13}$$

$$c_\tau(\mathbf{n}) := C(\mathbf{n})\breve{x}_\tau(\mathbf{n}) = \sum_{u=1}^{U} C(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) = \sum_{u=1}^{U} c_\tau(\mathbf{n}(u)) \tag{14}$$

$$\begin{aligned} a_\tau(\mathbf{n}) &:= \frac{1}{2}\sum_{u=1}^{U}(\mathbf{I}C(\mathbf{n}(u)) + D(\mathbf{n}(u)))\breve{x}_\tau(\mathbf{n}(u)) \\ &= \sum_{u=1}^{U} a_\tau(\mathbf{n}(u)) \end{aligned} \tag{15}$$

where $\breve{x}_t(\mathbf{n})$ is a general orthogonal expansion (GOE) and $D(\mathbf{n})$ and $C(\mathbf{n})$ are general expansion correlation matrices (GECMs) for PU($\mathbf{n}$). It is easy to see that $a_\tau(\mathbf{n}(u)) = (\mathbf{I}c_\tau(\mathbf{n}(u)) + d_\tau(\mathbf{n}(u)))/2$, and $a_\tau(\mathbf{n}) = (\mathbf{I}c_\tau(\mathbf{n}) + d_\tau(\mathbf{n}))/2$.

As a special case, Example 2 shows that $a_{\tau i}(\mathbf{n})/c_\tau(\mathbf{n}) = (1 + d_{\tau i}(\mathbf{n})/c_\tau(\mathbf{n}))/2$ is an approximate of the subjective probability that the i-th component of the label of $a_\tau(\mathbf{n})$ is $+1$. The general case is examined in the following.

Assume that all $x_t(\mathbf{n})$ and $x_\tau(\mathbf{n})$ are bipolar binary vectors. By (11), (12), (3) and (4),

$$a_{\tau j}(\mathbf{n}(u)) = \Lambda \sum_{t \in G_{\tau j}(\mathbf{n}(u),+)} 2^{\dim \mathbf{n}(u)} \Lambda^{T-t}$$
$$c_\tau(\mathbf{n}(u)) = \Lambda \sum_{t \in G_\tau(\mathbf{n}(u))} 2^{\dim \mathbf{n}(u)} \Lambda^{T-t}$$

where

$$G_{\tau j}(\mathbf{n}(u),+) = \{t \in [0,T] | x_t(\mathbf{n}(u)) = x_\tau(\mathbf{n}(u)), r_{tj}(\mathbf{n}) = 1\}$$
$$G_\tau(\mathbf{n}(u)) = \{t \in [0,T] | x_t(\mathbf{n}(u)) = x_\tau(\mathbf{n}(u))\}$$

Assume further that $\dim \mathbf{n}(u)$, $u = 1,\ldots, U$ are all the same. Then if $c_\tau(\mathbf{n}) \neq 0$,

$$\frac{a_{\tau j}(\mathbf{n})}{c_\tau(\mathbf{n})} = \frac{\sum_{u=1}^{U} \sum_{t \in G_{\tau j}(\mathbf{n}(u),+)} \Lambda^{T-t}}{\sum_{u=1}^{U} \sum_{t \in G_\tau(\mathbf{n}(u))} \Lambda^{T-t}} \tag{16}$$

For example, if λ and U are set equal to 1, the above expression becomes

$$\frac{a_{\tau j}(\mathbf{n})}{c_\tau(\mathbf{n})} = \frac{|G_{\tau j}(\mathbf{n}(1), +)|}{|G_\tau(\mathbf{n}(1))|}$$

where $|G_{\tau j}(\mathbf{n}(1), +)|$ is the number of $x_t(\mathbf{n}(1))$'s with $r_{tj}(\mathbf{n}) = 1$ that have been learned and are equal to $x_\tau(\mathbf{n}(1))$, and $|G_\tau(\mathbf{n}(1))|$ is the number of $x_t(\mathbf{n}(1))$'s that have been learned and are equal to $x_\tau(\mathbf{n}(1))$. Hence, the ratio $a_{\tau j}(\mathbf{n})/c_\tau(\mathbf{n})$ is a relative frequence that the input feature subvector $x_\tau(\mathbf{n}(1))$ has a label with its j-th component $r_{tj}(\mathbf{n}) = 1$. The example also shows that if λ equal to 1, the memory, $D(\mathbf{n})$ and $C(\mathbf{n})$, never degrades. However, in this case, if learning continues, the memory can get saturated, causing memory "overflow."

The closer λ is to 1 and the smaller U is, the closer the above expression (16) approximates the subjective probability that the label $r_{\tau j}(\mathbf{n}) = 1$, based on the GECMs, $C(\mathbf{n})$ and $D(\mathbf{n}) = 2A(\mathbf{n}) - C(\mathbf{n})$, which are constructed with pairs $(x_t(\mathbf{n}(u)), r_t(\mathbf{n}))$, $t = 1, 2,\ldots, T$. Here $\mathbf{I} = [1 \ \ \ldots \ \ 1]'$ with R components. (note that $\mathbf{I}$ is not the identify matrix I.) The forgetting factor λ de-emphasizes past pairs gradually. It does not have to be applied each time a feature subvector $x_t(\mathbf{n})$ is learned by PU($\mathbf{n}$) as above. It can be applied once after a certain number, say 1,600 of feature subvectors are learned by the PU.

All the statements concerning a probability in this paper are statements concerning a subjective probability, and the word "subjective" is sometimes omitted. If $c_\tau(\mathbf{n}) \neq 0$, then $a_{\tau j}(\mathbf{n})/c_\tau(\mathbf{n})$ is approximately the probability $p_{\tau j}(\mathbf{n})$ that the j-th component $r_{\tau j}(\mathbf{n})$ of the label $r_\tau(\mathbf{n})$ of $x_\tau(\mathbf{n})$ is $+1$ based on $D(\mathbf{n})$ and $C(\mathbf{n})$. If $c_\tau(\mathbf{n}) = 0$, then we set $p_{\tau j}(\mathbf{n}) =$ 1/2. The vector

$$p_\tau(\mathbf{n}) = [p_{\tau 1}(\mathbf{n}) \quad p_{\tau 2}(\mathbf{n}) \quad \ldots \quad p_{\tau R}(\mathbf{n})]'$$

is a representation of a probability distribution of the label $r_\tau(\mathbf{n})$ of the feature subvector $x_\tau(\mathbf{n})$ input to PU($\mathbf{n}$). Since $D(\mathbf{n}) = 2A(\mathbf{n}) - \mathbf{I}C(\mathbf{n})$, if $c_\tau(\mathbf{n}) \neq 0$, the ratio $d_{\tau j}(\mathbf{n})/c_\tau(\mathbf{n})$ is equal to $2p_{\tau j}(\mathbf{n}) - 1$. If $c_\tau(\mathbf{n}) = 0$, set $2p_{\tau j}(\mathbf{n}) - 1 = 0$. Denote $2p_{\tau j}(\mathbf{n}) - 1$ by $y_{\tau j}(\mathbf{n})$. Then the vector $y_\tau(\mathbf{n}) = 2p_\tau(\mathbf{n}) - \mathbf{I}$ satisfies

$$\begin{aligned} y_\tau(\mathbf{n}) &= [2p_{\tau 1}(\mathbf{n}) - 1 \quad \ldots \quad 2p_{\tau R}(\mathbf{n}) - 1]' \\ &= [d_{\tau 1}(\mathbf{n})/c_\tau(\mathbf{n}) \quad d_{\tau 2}(\mathbf{n})/c_\tau(\mathbf{n}) \quad \ldots \quad d_{\tau R}(\mathbf{n})/c_\tau(\mathbf{n})]' \end{aligned}$$

and is also a representation of a probability distribution of the label $r_\tau(\mathbf{n})$ of the feature subvector $x_\tau(\mathbf{n})$. Here, $\mathbf{I} = [1 \quad 1 \quad \ldots \quad 1]'$.

A point estimate of the label $r_\tau(\mathbf{n})$ can be obtained by converting each component $y_{\tau j}(\mathbf{n})$ of $y_\tau(\mathbf{n})$ into a ternary number $x\{y_{\tau j}(\mathbf{n})\}$ by the following steps: For $k = 1, R$, set $p_{\tau j}(\mathbf{n}) = (y_{\tau j}(\mathbf{n}) + 1)/2$, and generate a pseudo-random number in accordance with the probability distribution of a random variable $v : P(v = 1) = p_{\tau j}(\mathbf{n})$ and $P(v = -1) = 1 - p_{\tau j}(\mathbf{n})$, and set $x\{y_{\tau j}(\mathbf{n})\}$ equal to the resultant pseudo-random number. Assemble $x\{y_{\tau j}(\mathbf{n})\}$, $j = 1,\ldots, R$, into $x\{y_\tau(\mathbf{n})\} = [x\{y_{\tau 1}(\mathbf{n})\} \quad x\{y_{\tau 2}(\mathbf{n})\} \quad \ldots \quad x\{y_{\tau R}(\mathbf{n})\}]'$, which is a point estimate of the label $r_\tau(\mathbf{n})$.

Masking matrices

Let a feature subvector that deviates from each of a group of feature subvectors that have been learned by the PU due to corruption, distortion or occlusion be presented to the PU. If the PU is able to automatically find the largest subvector of the presented subvector that matches at least one subvector among the group and generate the SPD of the label of the largest subvector, the PU is said to have a maximal generalization capability. This capability is achieved by the use of masking matrices described in this section.

Let a subvector $x_\tau(\mathbf{n}(u))$ be a slightly different (e.g., corrupted, distorted, occluded) version of $x_\xi(\mathbf{n}(u))$, which is one of the subvectors, $x_t(\mathbf{n}(u))$, $t = 1, 2,\ldots, T$, stored in ECMs, $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$, on $\mathbf{n}(u)$. Assume that $x_\tau(\mathbf{n}(u))$ is very different from other subvectors stored in the ECMs. Since $\breve{x}'_\xi(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) = 0$, the information stored in $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$ about the label $r_\xi(\mathbf{n})$ cannot be obtained from $d(\mathbf{n}(u)) = D(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u))$ and $c(\mathbf{n}(u)) = C(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u))$. This is viewed as failure of $d(\mathbf{n}(u))$ and $c(\mathbf{n}(u))$ to generalize. Because of property 2 in Theorem 1, if the corrupted, distorted and occluded components in $x_\tau(\mathbf{n}(u))$ are set equal to zero, then the information stored in the ECMs about the label $r_\xi(\mathbf{n})$ can be obtained in part from the remaining components of $x_\tau(\mathbf{n}(u))$. This observation motivated masking matrices.

Let us denote the vector $v = [v_1 \quad v_2 \quad \ldots \quad v_n]'$ with its i_1-th, i_2-th,..., and i_j-th components set equal to 0 by $v(i_1^-, i_2^-, \ldots, i_j^-)$, where $1 \leq i_1 < i_2 < \cdots < i_j \leq n$. For example, if $v = [1 \quad -1 \quad -1 \quad 1]'$, then $v(2^-, 4^-) = [1 \quad 0 \quad -1 \quad 0]'$. Denote the n-dimensional vector $[1 \quad 1 \quad \ldots \quad 1]'$ by $\mathbf{I}$ (not the identity matrix I) and denoting the orthogonal expansion of $v(i_1^-, i_2^-,\ldots, i_j^-)$ by $\breve{v}(i_1^-, i_2^-, \ldots, i_j^-)$. We note that $v(i_1^-, i_2^-,\ldots, i_j^-)$ = diag $(\mathbf{I}(i_1^-, i_2^-,\ldots, i_j^-))\, v$ and $\breve{v}(i_1^-, i_2^-, \ldots, i_j^-) = \text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{v}$, where $\breve{v}(i_1^-, i_2^-, \ldots, i_j^-)$ and $\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-)$ denote the orthogonal expansions of $v(i_1^-, i_2^-,\ldots, i_j^-)$ and $\mathbf{I}(i_1^-, i_2^-, \ldots, i_j^-)$ respectively (not the orthogonal expansions of v and $\mathbf{I}$ with their i_1-th, i_2-th, and i_j-th components set equal to 0).

Using these notations, a feature subvector $x(\mathbf{n}(u))$ with its i_1-th, i_2-th, and i_j-th components set equal to 0 is $x_t(\mathbf{n}(u))(i_1^-, i_2^-, \ldots, i_j^-)$, and the orthogonal expansion of $x_t(\mathbf{n}(u))(i_1^-, i_2^-, \ldots, i_j^-)$ is $\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_t(\mathbf{n}(u))$. Hence, the matrix $\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))$, as a matrix transformation, sets the i_1-th, i_2-th, and i_j-th components of $x_t(\mathbf{n}(u))$ equal to zero in transforming $\breve{x}_t(\mathbf{n}(u))$ (i.e., in $\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_t(\mathbf{n}(u))$).

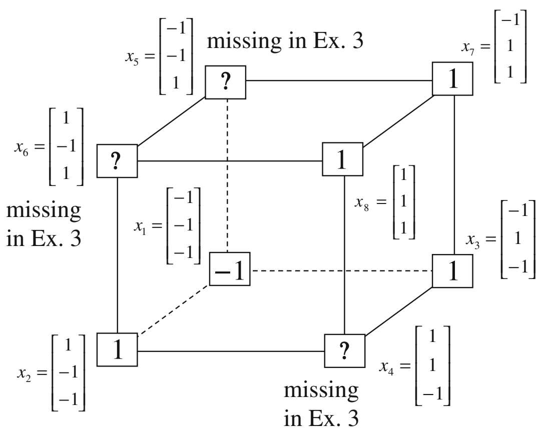

$x_5 = \begin{bmatrix}-1\\-1\\1\end{bmatrix}$
missing in Ex. 3
?
$x_7 = \begin{bmatrix}-1\\1\\1\end{bmatrix}$
1
$x_6 = \begin{bmatrix}1\\-1\\1\end{bmatrix}$
?
1
missing in Ex. 3
$x_1 = \begin{bmatrix}-1\\-1\\-1\end{bmatrix}$
$x_8 = \begin{bmatrix}1\\1\\1\end{bmatrix}$
$x_3 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}$
−1
1
$x_2 = \begin{bmatrix}1\\-1\\-1\end{bmatrix}$
1
?
$x_4 = \begin{bmatrix}1\\1\\-1\end{bmatrix}$
missing in Ex. 3

Fig. 1 Data for training and testing the PU (processing unit) in Example 3 and Example 4 are shown as the vertices of a cube. Their bipolar binary labels are the numbers or question marks for unknown labels in the squares at the vertices. x_4, x_5, x_6 are unavailable in the data set for Example 3. They are learned one by one without supervision (i.e., with labels generated by the PU) in Example 4

Two important properties of the matrix $\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))$ are the following:

1. If $\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_t(\mathbf{n}(u)) = \text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_\tau(\mathbf{n}(u))$, then $\breve{x}_t'(\mathbf{n}(u))\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_\tau(\mathbf{n}(u)) = 2^{\dim \mathbf{n}(u) - j}$.
2. If $\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_t(\mathbf{n}(u)) \neq \text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_\tau(\mathbf{n}(u))$, then $\breve{x}_t'(\mathbf{n}(u))\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))\breve{x}_\tau(\mathbf{n}(u)) = 0$.

The following example illustrates how such matrices $\text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-))$ can be used by a PU (processing unit) to generalize.

Example 3 Consider a cube shown in Fig. 1. The coordinate vectors of its eight vertices, x_t, $t = 1, 2, \ldots, 8$, and their corresponding labels, r_t, $t = 1, 2, \ldots, 8$, are shown at the vertices and in the squares, respectively, where the question marks indicate unknown labels. The training data consists of the pairs, (x_t, r_t), $t = 1, 2, 3, 7, 8$.

The pairs, $(\breve{x}_t', r_t)$, $t = 1, 2, 3, 7, 8$, are listed as rows in the following table:

$\breve{x}_t'$	1	x_{t1}	x_{t2}	$x_{t2}x_{t1}$	x_{t3}	$x_{t3}x_{t1}$	$x_{t3}x_{t2}$	$x_{t3}x_{t2}x_{t1}$	r_t
$\breve{x}_1'$	1	−1	−1	1	−1	1	1	−1	−1
$\breve{x}_2'$	1	1	−1	−1	−1	−1	1	1	1
$\breve{x}_3'$	1	−1	1	−1	−1	1	−1	1	1
$\breve{x}_7'$	1	−1	1	−1	1	−1	1	−1	1
$\breve{x}_8'$	1	1	1	1	1	1	1	1	1

Assume $U = 1$ and $\Lambda = \Lambda = 1$ in (5), (6), (3) and (4) in a PU (processing unit). The general expansion correlation matrices, D and C, of the PU is the following:

$$D = [3 \quad 1 \quad 3 \quad -3 \quad 1 \quad -1 \quad 1 \quad 3] \tag{17}$$

$$C = [5 \quad -1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 3 \quad 1] \tag{18}$$

Let

$$\mathbf{I}(1^-) = \begin{bmatrix}0\\1\\1\end{bmatrix}, \mathbf{I}(2^-) = \begin{bmatrix}1\\0\\1\end{bmatrix}, \mathbf{I}(3^-) = \begin{bmatrix}1\\1\\0\end{bmatrix}$$

Orthogonal expansion of them yields

$$\breve{\mathbf{I}}(1^-) = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0]'$$
$$\breve{\mathbf{I}}(2^-) = [1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0]'$$
$$\breve{\mathbf{I}}(3^-) = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]'$$

We introduce the following matrix

$$\begin{aligned} M &= I + 2^{-8}\text{diag}(\breve{\mathbf{I}}(1^-) + \breve{\mathbf{I}}(2^-) + \breve{\mathbf{I}}(3^-)) \\ &= I + 2^{-8}\text{diag}[3 \quad 2 \quad 2 \quad 1 \quad 2 \quad 1 \quad 1 \quad 0] \end{aligned} \tag{19}$$

where the weight 2^{-8} is selected to de-emphasize the effect of the second term above as compared with the first term. The orthogonal expansion of the three vertices of the cube in Fig. 1 that are not included in the training data are listed as follows:

$\breve{x}_t'$	1	x_{t1}	x_{t2}	$x_{t2}x_{t1}$	x_{t3}	$x_{t3}x_{t1}$	$x_{t3}x_{t2}$	$x_{t3}x_{t2}x_{t1}$
$\breve{x}_4'$	1	1	1	1	−1	−1	−1	−1
$\breve{x}_5'$	1	−1	−1	1	1	−1	−1	1
$\breve{x}_6'$	1	1	−1	−1	1	1	−1	−1

From the following examples,

$$\text{diag}(\breve{\mathbf{I}}(1^-))\breve{x}_4 = [1 \quad 0 \quad 1 \quad 0 \quad -1 \quad 0 \quad -1 \quad 0]'$$

$$\text{diag}(\breve{\mathbf{I}}(2^-))\breve{x}_6 = [1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0]'$$

$$\text{diag}(\breve{\mathbf{I}}(3^-))\breve{x}_5 = [1 \quad -1 \quad -1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]'$$

we see that $\text{diag}(\breve{\mathbf{I}}(k^-))$ sets the k-th component x_{tk} of $\breve{x}_t$ equal to 0 for $t = 1, \ldots, 8$, $k = 1, 2, 3$.

Simple matrix-vector multiplication yields $D\breve{x}_t = 0$ and $C\breve{x}_t = 0$ for $t = 4, 5, 6$. Hence no information is provided on x_t by $D\breve{x}_t$ and $C\breve{x}_t$ for $t = 4, 5, 6$. This shows that if x_t has not been learned, then no information on it is provided by the general expansion matrices. Recall that if $c_{\tau j}(\mathbf{n}) \neq 0$, the subjective probability $p_{\tau j}(\mathbf{n}) = (d_{\tau j}(\mathbf{n})/c_{\tau j}(\mathbf{n}) + 1)/2$, where $d_{\tau j}(\mathbf{n}) = D_{\tau j}(\mathbf{n})\breve{x}_\tau$ and $c_{\tau j}(\mathbf{n}) = C_{\tau j}(\mathbf{n})\breve{x}_\tau$. With M, we will use $d_{\tau j}(\mathbf{n}) = D_{\tau j}(\mathbf{n})M\breve{x}_\tau$ and $c_{\tau j}(\mathbf{n}) = C_{\tau j}(\mathbf{n})M\breve{x}_\tau$ instead.

Assume that x_1 is input to the PU with the above D and C. By matrix multiplication,

$$\begin{aligned} DM\breve{x}_1 &= -8 + 2^{-8}(9-2-6-3-2-1+1) \\ &= -8 + 2^{-8}(12) = -7.9531 \\ CM\breve{x}_1 &= 8 + 2^{-8}(15+2-2-1+2+1+3) \\ &= 8 + 2^{-8}(20) = 8.0781 \end{aligned}$$

Then the subjective probability that the label of x_4 is 1 is $(DM\breve{x}_1/(CM\breve{x}_1)+1)/2 = 0.0077$, and the subjective probability that the label of x_4 is -1 is 0.9923. Note that x_1 with a label of -1 has been learned. The subjective probability that the label of x_4 is -1 should be 1. The use of M causes a very small amount of error to the subjective probability, which can be adjusted by changing the weight, 2^{-8}.

Assume that x_4 is input to the PU with the above D and C. By matrix multiplication,

$$\begin{aligned} DM\breve{x}_4 &= 0 + 2^{-8}(9+2+6-3-2+1-1) = 2^{-8}(12) \\ CM\breve{x}_4 &= 0 + 2^{-8}(15-2+2-1+2-1-3) = 2^{-8}(12) \end{aligned}$$

Then the subjective probability that the label of x_4 is 1 is $(DM\breve{x}_4/(CM\breve{x}_4)+1)/2 = 1$. From Fig. 1, we see that all the three vertices neighboring x_4 have been learned and have a label of 1. It is a good generalization that a label of 1 is assigned to x_4.

Assume that x_6 is presented to the same PU. By matrix multiplication,

$$DM\breve{x}_6 = 0 + 2^{-8}(9+2-6+3+2-1-1) = 2^{-8}(8) \tag{20}$$

$$CM\breve{x}_6 = 0 + 2^{-8}(15-2-2+1-2+1-3) = 2^{-8}(8) \tag{21}$$

Then the subjective probability that the label of x_6 is 1 is $(DM\breve{x}_6/(CM\breve{x}_6)+1)/2 = 1$. From Fig. 1, we see that only two vertices neighboring x_4 have been learned, and they both have a label of 1. It is a good generalization that a label of 1 is assigned to x_6.

Assume that x_5 is input to the same PU. By matrix multiplication,

$$\begin{aligned} DM\breve{x}_5 &= 0 + 2^{-8}(9-2-6-3+2+1-1) = 2^{-8}(0) \\ CM\breve{x}_5 &= 0 + 2^{-8}(15+2-2-1-2-1-3) = 2^{-8}(8) \end{aligned}$$

Then the subjective probability that the label of x_5 is 1 is $(DM\breve{x}_5/(CM\breve{x}_5)+1)/2 = 1/2$. From Fig. 1, we see that only two vertices neighboring x_4 have been learned, and one of them has a label of 1, and the other has a label of -1. No generalization is possible. A label of 1 is assigned to x_6 with a subjective probability of 1/2 and that a label of -1 is assigned to x_6 with equal subjective probability.

In the general case, we combine all such matrices $\text{diag}(\mathbf{I}(i_1^-, i_2^-, \ldots, i_j^-))$ that set less than or equal to a selected positive integer $J(\mathbf{n}(u))$ of components of $x_t(\mathbf{n}(u))$ equal to zero into the following masking matrix

$$\begin{aligned} M(\mathbf{n}(u)) =& I + \sum_{j=1}^{J(\mathbf{n}(u))} \sum_{i_j=j}^{\dim \mathbf{n}(u)} \\ & \cdots \sum_{i_2=2}^{i_3-1} \sum_{i_1=1}^{i_2-1} 2^{-8j} 2^j \text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-, \ldots, i_j^-)) \end{aligned} \tag{22}$$

where 2^j is used to compensate for the factor 2^{-j} in $2^{\dim \mathbf{n}(u)-j}$ in the important property stated above, and 2^{-8j} is an example weight selected to differentiate between different levels j of maskings. What this weight really is in biological neural networks needs to be found by biological experiments. So is the positive integer $J(\mathbf{n}(u))$.

Let us denote $M(\mathbf{n}(u))$ by M here for abbreviation. Note that for $k = 1,\ldots, R$, we have the following:

- If $C(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) \neq 0$, then

$$\begin{aligned} D_k(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) &\approx D_k(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u)) \\ C(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) &\approx C(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u)) \end{aligned}$$

- If $C(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) = 0$, but $C(\mathbf{n}(u))\sum_{i_1=1}^{\dim \mathbf{n}(u)} \text{diag}(\breve{\mathbf{I}}(i_1^-))\breve{x}_\tau(\mathbf{n}(u)) \neq 0$, then

$$D_k(\mathbf{n}(u)) \sum_{i_1=1}^{\dim \mathbf{n}(u)} \text{diag}(\breve{\mathbf{I}}(i_1^-))\breve{x}_\tau(\mathbf{n}(u)) \approx D_k(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u))$$

$$C(\mathbf{n}(u)) \sum_{i_1=1}^{\dim \mathbf{n}(u)} \text{diag}(\breve{\mathbf{I}}(i_1^-))\breve{x}_\tau(\mathbf{n}(u)) \approx C(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u))$$

- If $C(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) = 0$, $C(\mathbf{n}(u))\sum_{i_1=1}^{\dim \mathbf{n}(u)} \text{diag}(\breve{\mathbf{I}}(i_1^-))\breve{x}_\tau(\mathbf{n}(u)) = 0$, but $C(\mathbf{n}(u))\sum_{i_2=2}^{\dim \mathbf{n}(u)}\sum_{i_1=1}^{i_2-1} \text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-))\breve{x}_\tau(\mathbf{n}(u)) \neq 0$, then

$$\begin{aligned} & D_k(\mathbf{n}(u)) \sum_{i_2=2}^{\dim \mathbf{n}(u)} \sum_{i_1=1}^{i_2-1} \text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-))\breve{x}_\tau(\mathbf{n}(u)) \\ & \approx D_k(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u)) \\ & C(\mathbf{n}(u)) \sum_{i_2=2}^{\dim \mathbf{n}(u)} \sum_{i_1=1}^{i_2-1} \text{diag}(\breve{\mathbf{I}}(i_1^-, i_2^-))\breve{x}_\tau(\mathbf{n}(u)) \\ & \approx C(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u)) \end{aligned}$$

Continuing in this manner, it is seen that $D_k(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u))$ and $C(\mathbf{n}(u))M\breve{x}_\tau(\mathbf{n}(u))$ always use the greatest number of uncorrupted, undistorted or unoccluded components of $x_\tau(\mathbf{n}(u))$ in estimating $d_{\tau k}(\mathbf{n}(u))$, $c_\tau(\mathbf{n}(u))$, and $a_{\tau k}(\mathbf{n}(u))$.

Corresponding to $\breve{x}_t(\mathbf{n})$, $D(\mathbf{n})$ and $C(\mathbf{n})$ defined in (7), (8) and (9), a general masking matrix is defined as follows:

$$M(\mathbf{n}) = \text{diag}[M(\mathbf{n}(1)) \quad M(\mathbf{n}(2)) \quad \ldots \quad M(\mathbf{n}(U))] \tag{23}$$

where the right side is a matrix with $M(\mathbf{n}(u))$, $u = 1, 2, U$, as diagonal blocks and zero elsewhere.

If the masking matrix $M(\mathbf{n}(u))$ is used, the symbols $a_\tau(\mathbf{n}(u))$, $c_\tau(\mathbf{n}(u))$, $d_\tau(\mathbf{n}(u))$ are defined as follows:

$$d_\tau(\mathbf{n}(u)) := D(\mathbf{n}(u))M(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) \tag{24}$$

$$c_\tau(\mathbf{n}(u)) := C(\mathbf{n}(u))M(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) \tag{25}$$

$$a_\tau(\mathbf{n}(u)) := \frac{1}{2}(\mathbf{I}c_\tau(\mathbf{n}(u)) + d_\tau(\mathbf{n}(u))) \tag{26}$$

With the masking matrix $M(\mathbf{n})$, the symbols $a_\tau(\mathbf{n})$, $b_\tau(\mathbf{n})$, $c_\tau(\mathbf{n})$, $d_\tau(\mathbf{n})$ are in turn defined as follows:

$$d_\tau(\mathbf{n}) := D(\mathbf{n})M(\mathbf{n})\breve{x}_\tau(\mathbf{n}) = \sum_{u=1}^{U} D(\mathbf{n}(u))M(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) \tag{27}$$

$$c_\tau(\mathbf{n}) := C(\mathbf{n})M(\mathbf{n})\breve{x}_\tau(\mathbf{n}) = \sum_{u=1}^{U} C(\mathbf{n}(u))M(\mathbf{n}(u))\breve{x}_\tau(\mathbf{n}(u)) \tag{28}$$

$$a_\tau(\mathbf{n}) := \frac{1}{2}(c_\tau(\mathbf{n}) + d_\tau(\mathbf{n})) \tag{29}$$

where $\breve{x}_t(\mathbf{n})$ is a general orthogonal expansion (GOE) and $D(\mathbf{n})$ and $C(\mathbf{n})$ are general expansion correlation matrices (GECMs) for PU$(\mathbf{n})$. It follows that

$$d_\tau(\mathbf{n}) = \sum_{u=1}^{U} d_\tau(\mathbf{n}(u)) \tag{30}$$

$$c_\tau(\mathbf{n}) = \sum_{u=1}^{U} c_\tau(\mathbf{n}(u)) \tag{31}$$

$$a_\tau(\mathbf{n}) = \sum_{u=1}^{U} a_\tau(\mathbf{n}(u)) \tag{32}$$

It is easy to see that $d_\tau(\mathbf{n}(u)) = 2a_\tau(\mathbf{n}(u)) - c_\tau(\mathbf{n}(u))$, and $d_\tau(\mathbf{n}) = 2a_\tau(\mathbf{n}) - c_\tau(\mathbf{n})$. If $c_\tau(\mathbf{n}) = 0$, then we set $d_\tau(\mathbf{n})/c_\tau(\mathbf{n}) = 0$. If $c_\tau(\mathbf{n}) \neq 0$, then $d_{\tau k}(\mathbf{n})/c_\tau(\mathbf{n}) = 2p_{\tau k}(\mathbf{n}) - 1$, where $p_{\tau k}(\mathbf{n})$ is the probability that the k-th component $r_{\tau k}(\mathbf{n})$ of the label $r_\tau(\mathbf{n})$ of $x_\tau(\mathbf{n})$ is $+1$ based on $D(\mathbf{n})$ and $C(\mathbf{n})$. It follows that

$$\begin{aligned} &2p_\tau(\mathbf{n}) - \mathbf{I} \\ &= [d_{\tau 1}(\mathbf{n})/c_{\tau 1}(\mathbf{n}) \quad d_{\tau 2}(\mathbf{n})/c_{\tau 2}(\mathbf{n}) \quad \ldots \quad d_{\tau R}(\mathbf{n})/c_{\tau R}(\mathbf{n})]' \end{aligned}$$

is a representation of a probability distribution of the label $r_\tau(\mathbf{n})$ of $x_\tau(\mathbf{n})$.

It is mentioned in "Expansion correlation matrices" that selecting sufficiently small subvectors $\mathbf{n}(u)$, $u = 1,\ldots, U$, has the advantage of making $\dim \breve{x}_t(\mathbf{n}) = \sum_{u=1}^{U} 2^{\dim x_t(\mathbf{n}(u))}$ sufficiently small. The formula (22) shows that selecting sufficiently small subvectors $\mathbf{n}(u)$, $u = 1,\ldots, U$, also has the advantage of making the number of terms in the formula sufficiently small. The use of subvectors $\mathbf{n}(u)$, $u = 1,\ldots, U$, has another way to help enhancing the generalization capability of PU$(\mathbf{n})$: If the number of corrupted, distorted or occluded components of a subvector $x_\tau(\mathbf{n}(u))$ of $x_\tau(\mathbf{n})$ exceeds $J(\mathbf{n}(u))$, then $x_\tau(\mathbf{n}(u))$ does not contribute to $y_\tau(\mathbf{n})$ or the output $x\{y_\tau(\mathbf{n})\}$ of PU$(\mathbf{n})$. This eliminates the effect of a subvector $x_\tau(\mathbf{n}(u))$ that contains too many errors and allows PU$(\mathbf{n})$ to produce a better estimate of the subjective probability distribution of a label on better subvectors of $x_\tau(\mathbf{n})$.

If some terms in (22) are missing, PU$(\mathbf{n})$ suffers only graceful degradation of its generalization capability. We hypothesize that a masking matrix is a mathematical idealization and organization of a large number of nested and overlapped dendritic trees.

Processing units and supervised and unsupervised learning

We are ready to assemble a PU (processing unit) and see how supervised and unsupervised learning are performed. A processing unit, PU$(\mathbf{n})$, on a feature subvector index $\mathbf{n}$, is shown in Fig. 2. It has essentially two functions, retrieving a "point estimate" of the label of a feature subvector $x_\tau(\mathbf{n})$ from the memory (i.e., GECMs) and learning a feature subvector and its label that is either provided from outside the PU (in supervised learning) or generated by itself (in unsupervised learning). PU$(\mathbf{n})$ comprises an Orthogonal Expander, a label SPD (subjective probability distribution) Estimator, a Spike Generator, a GECM (general expansion correlation matrix) Adjuster, and a storage of the GECMs, $C(\mathbf{n})$ and $D(\mathbf{n})$. The Orthogonal Expander models dendritic trees with NXORs (or XORs) as tree nodes, $C(\mathbf{n})$ and $D(\mathbf{n})$ model the synaptic weights in a biological neural network, and the label SPD Estimator and Spike Generator jointly model R neurons of one type and 1 neuron of another type in the same layer of a biological neural network. These two types of neuron will be described below.

During retrieving, a feature subvector $x_\tau(\mathbf{n})$ on the FSI (feature subvector index) $\mathbf{n}$ is first expanded into a GOE (general orthogonal expansion) $\breve{x}_\tau(\mathbf{n})$ by the Orthogonal Expander. $\breve{x}_\tau(\mathbf{n})$ is then processed by the SPD (subjective probability distribution) Estimator, using the GECMs (general expansion correlation matrices), $C(\mathbf{n})$ and $D(\mathbf{n})$, to obtain a representation $y_\tau(\mathbf{n})$ of an SPD of the label of the feature subvector $x_\tau(\mathbf{n})$. The Spike Generator converts $y_\tau(\mathbf{n})$ into a ternary vector $x\{y_\tau(\mathbf{n})\}$, which is the output of the PU. This process of generating $y_\tau(\mathbf{n})$ and $x\{y_\tau(\mathbf{n})\}$ by PU$(\mathbf{n})$ is called retrieval of a label of the feature subvector $x_\tau(\mathbf{n})$ by PU$(\mathbf{n})$.

The SPD Estimator and Spike Generator may be viewed as R neurons of one type and one neuron of another type

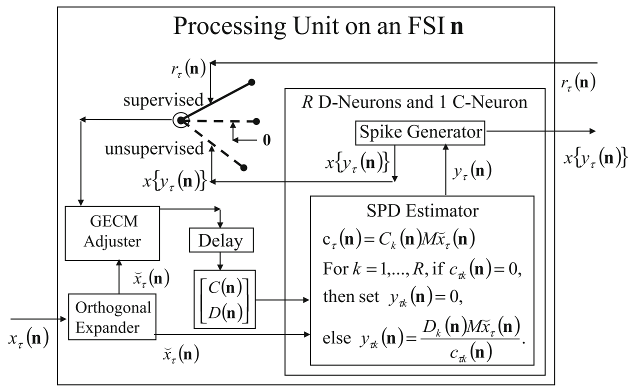

Processing Unit on an FSI $\mathbf{n}$
$r_\tau(\mathbf{n})$
supervised
unsupervised
0
$x\{y_\tau(\mathbf{n})\}$
R D-Neurons and 1 C-Neuron
Spike Generator
$y_\tau(\mathbf{n})$
SPD Estimator
$c_\tau(\mathbf{n}) = C_k(\mathbf{n})M\breve{x}_\tau(\mathbf{n})$
For $k = 1,\ldots,R$, if $c_{\tau k}(\mathbf{n}) = 0$,
then set $y_{\tau k}(\mathbf{n}) = 0$,
else $y_{\tau k}(\mathbf{n}) = \frac{D_k(\mathbf{n})M\breve{x}_\tau(\mathbf{n})}{c_{\tau k}(\mathbf{n})}$.
GECM Adjuster
Delay
$\begin{bmatrix} C(\mathbf{n}) \\ D(\mathbf{n}) \end{bmatrix}$
Orthogonal Expander
$\breve{x}_\tau(\mathbf{n})$
$x_\tau(\mathbf{n})$

Fig. 2 A processing unit, PU($\mathbf{n}$), with a feature subvector index $\mathbf{n}$, comprising an Orthogonal Expander, a label SPD (subjective probability distribution) Estimator, a Spike Generator, a GECM (general expansion correlation matrix) Adjuster, and a storage of $C(\mathbf{n})$ and $D(\mathbf{n})$. PU($\mathbf{n}$) has essentially two functions, retrieving a "point estimate" or a sequence of "point estimates" (i.e., spike trains) of the label of a feature subvector $x_\tau(\mathbf{n})$ from the memory, GECMs, and learning a feature subvector and its label that is either provided from outside the PU (in supervised learning) or generated by the PU itself (in unsupervised learning)

that jointly output a "point estimate" $x\{y_\tau(\mathbf{n})\}$ of the label of $x_\tau(\mathbf{n})$. The former type is called D-neurons and the latter C-neuron. The C-neuron does a simple multiplication

$$c_\tau(\mathbf{n}) = C(\mathbf{n})M\breve{x}_\tau(\mathbf{n})$$

For $j = 1,\ldots, R$, D-neuron j performs the following tasks:

1. Input $c(\mathbf{n})$. If $c_\tau(\mathbf{n}) = 0$, set $y_{\tau j}(\mathbf{n}) = 0$; else compute $d_j(\mathbf{n}) = D_j(\mathbf{n})M\breve{x}_\tau(\mathbf{n})$ and set $y_{\tau j}(\mathbf{n})$ equal to $d_{\tau j}(\mathbf{n})/c_\tau(\mathbf{n})$.
2. Compute the subjective probability $p_{\tau j}(\mathbf{n}) = (y_{\tau j}(\mathbf{n}) + 1)/2$ that the j-th component of the label of $x_\tau(\mathbf{n})$ is $+1$.
3. Generate a pseudo-random number in accordance with the probability distribution of a random variable v: $P(v = 1) = p_{\tau j}(\mathbf{n})$ and $P(v = -1) = 1 - p_{\tau j}(\mathbf{n})$, and set $x\{y_{\tau j}(\mathbf{n})\}$ equal to the resultant pseudo-random number. This is a point estimate of the j-th component of the label of $x_\tau(\mathbf{n})$.

Figure 2 provides a "flow chart" of the general PU. A structural diagram of an example PU is shown in Fig. 3, where the PU is that of Example 3. Input to the PU is the feature subvector $x_\tau = [x_{\tau 1} \quad x_{\tau 2} \quad x_{\tau 3}]'$. An dendritic tree encode x_τ into the orthogonal expansion $\breve{x}_\tau$, whose components are multiplied by the synapses, denoted by $\otimes$, and the resultant multiples are distributed to the D-neuron and C-neuron. In learning, the general expansion correlation matrices, D and C, are incremented by $r_\tau \breve{x}_\tau' \Lambda$ and $\breve{x}_\tau' \Lambda$, respectively. In supervised learning of D, r_τ is provided from outside the PU. In unsupervised learning of D, r_τ is set equal to $x\{y_\tau\}$, which is generated by D-neurons. C is the accumulation of $\breve{x}_t' \Lambda$. A possible way to perform this accumulation by the Hebb rule is for the C-neuron to have a second output that is always equal to the constant 1.

If a label $r_\tau(\mathbf{n}) \neq 0$ of $x_\tau(\mathbf{n})$ from outside the PU is available for learning, and learning $x_\tau(\mathbf{n})$ and $r_\tau(\mathbf{n})$ is wanted, supervised learning is performed by the PU. In supervised learning, the class label $r_\tau(\mathbf{n}) \neq 0$ is received through a lever represented by a thick solid line with a solid dot at its end in Fig. 2 by the GECM Adjuster, which receives also $\breve{x}_\tau(\mathbf{n})$ from the Orthogonal Expander and adjusts ECMs by formulas (5)–(6) and assembles the resultant ECMs, $C(\mathbf{n}(u))$ and $D(\mathbf{n}(u))$, $u = 1,\ldots,U$, into general ECMs, $C(\mathbf{n})$ and $D(\mathbf{n})$, by (8) and (9).

These $C(\mathbf{n})$ and $D(\mathbf{n})$ are then stored, after a one-numbering delay (or a unit-time delay), in the storage, from which they are sent to the SPD Estimator.

If a label $r_\tau(\mathbf{n})$ of $x_\tau(\mathbf{n})$ from outside the PU is unavailable but learning $x_\tau(\mathbf{n})$ is wanted, unsupervised learning is performed by the PU. In this case, the lever in Fig. 2 should be in the unsupervised training position represented by the lower dashed line with a solid dot at its end in Fig. 2. The feature subvector $x_\tau(\mathbf{n})$ is first processed by the Orthogonal Expander, SPD Estimator, and Spike Generator as in performing retrieval described above. The resultant bipolar vector $x\{y_\tau(\mathbf{n})\}$, which is a point estimate of the lable of $x_\tau(\mathbf{n})$ is received, through the lever in the unsupervised training position, and used by the GECM Adjuster as the label $r_\tau(\mathbf{n})$ of $x_\tau(\mathbf{n})$. The GECM Adjuster receives $\breve{x}_\tau(\mathbf{n})$ also and adjusts GECMs, $C(\mathbf{n})$ and $D(\mathbf{n})$, using the update formulas, (5)–(6), in the same way as in supervised learning.

Let us now see how a "vocabulary" is created by the PU through unsupervised learning: If a feature subvector $x_\tau(\mathbf{n})$ or a slightly different version of it has not been learned by PU($\mathbf{n}$), and $C(\mathbf{n})M\breve{x}_\tau(\mathbf{n}) = 0$; then $y_\tau(\mathbf{n}) = 0$ and $p_\tau(\mathbf{n}) = (1/2)\mathbf{I}$, where $\mathbf{I} = [1 \quad 1 \quad \ldots \quad 1]'$. The SPD Estimator and Spike Generator uses this probability vector to generate a purely random label $r_\tau(\mathbf{n}) = x\{y_\tau(\mathbf{n})\}$. Once this $x_\tau(\mathbf{n})$ has been learned and stored in $C(\mathbf{n})$ and $D(\mathbf{n})$, if $x_\tau(\mathbf{n})$ is input to PU($\mathbf{n}$) and to be learned without supervision for the second time, then $x\{y_\tau(\mathbf{n})\} = r_\tau(\mathbf{n})$ and one more copy of the pair $(x_\tau(\mathbf{n}), r_\tau(\mathbf{n}))$ is included in $C(\mathbf{n})$ and $D(\mathbf{n})$.

If a feature subvector $x_\tau(\mathbf{n})$ or a slightly different version of it has been learned by PU($\mathbf{n}$) with different labels for different numbers of times, then $y_\tau(\mathbf{n}) \neq 0$ and $p_\tau(\mathbf{n}) \neq (1/2)\mathbf{I}$. For example, assume that two labels, $r_\tau^1(\mathbf{n})$ and $r_\tau^2(\mathbf{n})$ of the same feature subvector $x_\tau(\mathbf{n})$ have been learned with relative frequencies, 0.7 and 0.3, respectively. Since these two labels may have common components, the point estimate of the label resembles $r_\tau^1(\mathbf{n})$ with a probability of higher that 70% and resembles $r_\tau^2(\mathbf{n})$ with a probability of greater than 30%. To learn this probability, a number of such point estimates need to be learned. This is one of the reasons for each PU to generate multiple spikes for each exogenous feature vector, which is to be discussed in "Spike trains for each exogenous feature vector".

If no learning is to be performed by PU($\mathbf{n}$), the lever represented by a thick solid line with a solid dot in Fig. 2 is placed in the neutral position, through which 0 is sent as the label $r_\tau(\mathbf{n})$ of $x_\tau(\mathbf{n})$ to the GECM Adjuster, which then keeps $C(\mathbf{n})$ and $D(\mathbf{n})$ unchanged. Here is a condition for setting $r_\tau(\mathbf{n}) = 0$ to skip learning (supervised or unsupervies): If $y_\tau(\mathbf{n})$ generated by a PU's estimation means in retrieving is a bipolar vector or sufficiently close to a bipolar vector by some criterion, which indicates that the input feature subvector $x_\tau(\mathbf{n})$ is adequately learned, then the lever is placed in the middle position and no learning is performed. This avoids "saturating" the expansion correlation matrices with too many copies of one feature subvector and its label.

We note that a well-known unsupervised learning method based on a kind of Hebb rule is the Oja learning algorithm that generates the principal components of the input vectors (Oja 1982). Oja's method gets the principal components only asymptotically and the principal components must taper down fast enough, which is true only if the input vectors do not have too many major features.

We use the PU of Example 3 to illustrate unsupervised learning in the following example.

Example 4 In this example, the PU in Example 3 with D, C, M in (17), (18), (19) will learn x_6 and then x_5 without supervision.

Recall that

$\breve{x}'_t$	1	x_{t1}	x_{t2}	$x_{t2}x_{t1}$	x_{t3}	$x_{t3}x_{t1}$	$x_{t3}x_{t2}$	$x_{t3}x_{t2}x_{t1}$
$\breve{x}'_6$	1	1	−1	−1	1	1	−1	−1
$\breve{x}'_5$	1	−1	−1	1	1	−1	−1	1

and

$$DM\breve{x}_6 = 0 + 2^{-8}(9 + 2 - 6 + 3 + 2 - 1 - 1) = 2^{-8}(8)$$

$$CM\breve{x}_6 = 0 + 2^{-8}(15 - 2 - 2 + 1 - 2 + 1 - 3) = 2^{-8}(8)$$

and the subjective probability that the label of x_6 is 1 is $(DM\breve{x}_6/(CM\breve{x}_6) + 1)/2 = 1$. Hence, $y_6 = DM\breve{x}_6/(CM\breve{x}_6) = 1$ and thus the spike $x\{y_6\}$ generated by the PU is 1. To learn x_6 without supervision, the GECM adjuster in Fig. 2 (i.e., Hebbian learning mechanism in Fig. 3) uses this spike as r_6 in (5) and (6) and updates D and C in (17) and (18) into

$$D = [4 \quad 2 \quad 2 \quad -4 \quad 2 \quad 0 \quad 0 \quad 2] \tag{33}$$

$$C = [6 \quad 0 \quad 0 \quad -2 \quad 0 \quad 2 \quad 2 \quad 0] \tag{34}$$

To learn x_5, the SPD Estimator in Fig. 2 (i.e., D- and C-neurons in Fig. 3) first processes it to obtain

$$\begin{aligned} DM\breve{x}_5 &= 0 + 2^{-8}(12 - 4 - 4 - 4 + 4 + 0 + 0 + 0) \\ &= 2^{-8}(4) \end{aligned} \tag{35}$$

$$\begin{aligned} CM\breve{x}_5 &= 0 + 2^{-8}(18 + 0 + 0 - 2 + 0 - 2 - 2 + 0) \\ &= 2^{-8}(12) \end{aligned} \tag{36}$$

$y_5 = DM\breve{x}_5/(CM\breve{x}_5) = 1/3$. The Spike Generator then uses the subjective probability $p_5 = (y_5 + 1)/3 = 2/3$ to output the spike $+1$ with probability 2/3 and -1 with probability 1/3. The resultant spike is then used as r_5 in (5) and (6) to updates D and C in (35) and (36) into

$$D = [5 \quad 1 \quad 1 \quad -3 \quad 3 \quad -1 \quad -1 \quad 3]$$
$$C = [7 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1]$$

or

$$D = [3 \quad 3 \quad 3 \quad -5 \quad 1 \quad 1 \quad 1 \quad 1]$$
$$C = [7 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1]$$

depending on whether $r_5 = 1$ or $r_5 = -1$, respectively. For D and C to learn the subjective probability $p_5 = 2/3$, the feature subvector x_5 needs to be learned a number of times. This is one of the motivations of PUs generating spike trains for each exogenous feature vector input to the THPAM. Generating spike trains in a THPAM are discussed in "Spike trains for each exogenous feature vector".

Learning to recognize rotated, translated or scaled patterns

In this section, we describe a method for PUs (processing units) to learn to recognize rotated, translated and scaled patterns. The method can be modified for PUs to learn to recognize translated and scaled temporal patterns such as speech and music. Since the method is valid for both supervised and unsupervised learning, labels $r_t(\mathbf{n})$ to be referred to may be provided from outside THPAM in supervised learning or generated by the PUs in unsupervised learning. It is assumed in this section that feature vectors are arrays of ternary pixels.

Locations of ternary pixels in an array are assumed to be dense relative to the locations of the pixels selected as components of a feature subvector $x_t(\mathbf{n})$ input to a PU. We identify the FSI (feature subvector index) $\mathbf{n}$ of a feature subvector with the locations of the pixels in $x_t(\mathbf{n})$. In other words, the components of $\mathbf{n}$ are also the numberings of the locations of the pixels included as components of $x_t(\mathbf{n})$.

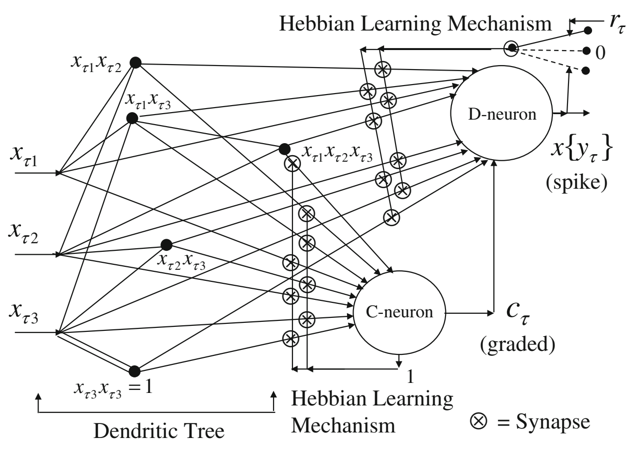


Fig. 3 The structural diagram of the PU (processing unit) in Example 3 and Example 4. The dendritic tree is the orthogonal expansion of the input feature subvector x_τ. The tree nodes are NXORs. A Hebbian learning mechanism for the D-neuron can perform supervised or unsupervised depending on whether the label r_τ is provided from outside the PU or is the output $x\{y_\tau\}$ of the D-neuron. A "pseudo-Hebbian" learning mechanism for the C-neuron performs only unsupervised learning and always uses 1 in so doing. While the D-neuron output spike trains, the C-neuron generates graded signals to modulate the D-neuron

Imagine a thin rubber disk with small holes at the locations of the pixels of the feature subvector with the FSI **n**. We translate the disk in some directions (e.g., 0, 15, 30, 45,…, 330, 345 degrees) a number of steps (e.g., 0, 1, 2,…), rotate the disk clockwise and counterclockwise a number (e.g., 0, 1, 2,…) of angles (e.g., 0, 5, 10, 15 degrees) at each translation, and expand and compress the rubber disk uniformly for a number of times (e.g., 0, 1, 2,…) at each translation for some percentages (e.g., 0, 5, 10%,…), to obtain other feature subvector indices of the same dimensionality as **n**. Note that in using the rubber disk to determine an FSI, if a hole in the rubber disk contains more than one pixel in the image, the one nearest to the center of the hole is included in the FSI.

Figure 4 shows examples of rotation, translations and scalings in an RTS (rotation, translation and scaling) suite of a feature subvector index **n**, which is shown in Fig. 4a. The components of **n** are the numberings (of a feature subvector) shown in the small circles within the retangular box. The cross without arrow heads indicate the orientation and position of **n**. Figure 4b shows a translation to the right. Figure 4c shows a rotation of the translation in Fig. 4b. Figure 4d and e show a compression and an expansion of the translation in Fig. 4b. Five examples of translations of **n** are shown in Fig. 4f.

Let $\Omega(\mathbf{n}) = \{\Omega(i), i = 1,\ldots,|\Omega(\mathbf{n})|\}$ be a set of FSIs $\omega(i)$ identified with such rotations, translations, and scalings of **n** including **n**. $\Omega(\mathbf{n})$ is called a rotation/translation/scaling (RTS) suite of **n**, and $|\Omega(\mathbf{n})|$ denotes the number of elements in $\Omega(\mathbf{n})$. Notice the digit 0 in the parentheses (e.g., 0, 1, 2,…) in the last paragraph. It indicates a rotation, a translation, or a scaling that is the feature subvector itself.

Although $\omega(i)$ is a rotation, translation, or scaling of **n**, this dependence on **n** is not indicated in the symbol $\omega(i)$ for notational simplicity. As **n** is rotated, translated or scaled into $\omega(i)$, $\mathbf{n}(u)$ as a subvector of **n** is rotated, translated or scaled into a subvector of $\omega(i)$. This subvector of $\omega(i)$ is denoted by $\mathbf{n}(u,\Omega(i))$. The set $\{\mathbf{n}(u,\Omega(i)), i = 1,\ldots,|\Omega(\mathbf{n})|\}$ of such subvectors of $\Omega(i), i = 1,\ldots,|\Omega(\mathbf{n})|$, is denoted by $\Omega(\mathbf{n}(u))$ and called a rotation/translation/scaling (RTS) suite of $\mathbf{n}(u)$. Note that $|\Omega(\mathbf{n}(u))| = |\Omega(\mathbf{n})|$. The set $\{x_t(\mathbf{n}(u,\Omega(i))), i = 1,\ldots,|\Omega(\mathbf{n})|\}$, which is also denoted by $\{x_t(\mathbf{n}(u,\Omega)), \Omega \in \Omega(\mathbf{n})\}$, is called the rotation/translation/scaling (RTS) suite of $x_t(\mathbf{n}(u))$ on $\Omega(\mathbf{n}(u))$. In

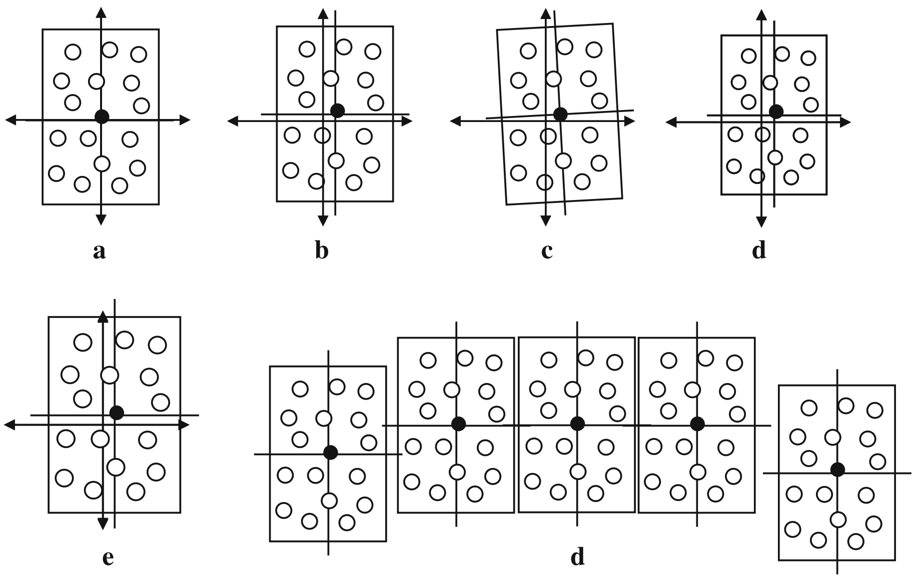


Fig. 4 A receptive domain is shown in (**a**), and a translation in (**b**). A rotation, compression and expansion of the translation are shown in (**c**), (**d**), (**e**). (**f**) shows five receptive domains that are translations of one another

generating and summing orthogonal expansions on an RTS suite $\Omega(\mathbf{n}(u))$, elements in the RTS suite of $x_t(\mathbf{n}(u))$ on $\Omega(\mathbf{n}(u))$ first go through orthogonal expansion. The resultant orthogonal expansions $\breve{x}_t(\mathbf{n}(u,\Omega(i)))$ are then added up to form the sum $\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}_t(\mathbf{n}(u,\Omega))$ on the RTS suite $\Omega(\mathbf{n}(u))$ of $\mathbf{n}(u)$.

In both the supervised learning and unsupervised learning, the subvectors, $x_t(\mathbf{n}(u,\Omega))$, $\Omega\in\Omega(\mathbf{n})$, on $\Omega(\mathbf{n}(u))$ are assigned the label $r_t(\mathbf{n})$ of $x_t(\mathbf{n})$. ECMs (expansion correlation matrices), $C(\mathbf{n}(u))$ and $D(\mathbf{n}(u))$, on $\Omega(\mathbf{n}(u))$ are defined by

$$C(\mathbf{n}(u))=\Lambda\sum_{t=1}^{T}\Lambda^{T-t}\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}'_t(\mathbf{n}(u,\Omega)) \tag{37}$$

$$D(\mathbf{n}(u))=\Lambda\sum_{t=1}^{T}\Lambda^{T-t}r_t(\mathbf{n})\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}'_t(\mathbf{n}(u,\Omega)) \tag{38}$$

$C(\mathbf{n}(u))$ and $D(\mathbf{n}(u))$ can be adjusted to learn a pair $(x_t, r_t(\mathbf{n}))$, where λ is a forgetting factor, and Λ is a scaling constant. If $r_\tau(\mathbf{n})\neq 0$, $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$ are replaced respectively with $\Lambda D(\mathbf{n}(u))+\Lambda r_\tau(\mathbf{n})\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}'_t(\mathbf{n}(u,\Omega))$ and $\Lambda C(\mathbf{n}(u))+\Lambda\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}'_t(\mathbf{n}(u,\Omega))$. If $r_\tau(\mathbf{n})=0$, then $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$ are unchanged.

Sums $\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}_t(\mathbf{n}(u,\Omega))$ of orthogonal expansions (OEs), and ECMs, $D(\mathbf{n}(u))$ and $C(\mathbf{n}(u))$, $u = 1,\ldots, U$, are respectively assembled into a general orthogonal expansion (GOE) $\breve{x}_t(\mathbf{n})$ and general expansion correlation matrices (GECMs), $D(\mathbf{n})$ and $C(\mathbf{n})$, for PU($\mathbf{n}$) (the PU on the feature vector $\mathbf{n}$) as follows:

$$\breve{x}'_t(\mathbf{n},\Omega)=\left[\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}'_t(\mathbf{n}(1,\Omega))\quad\ldots\quad\sum_{\Omega\in\Omega(\mathbf{n})}\breve{x}'_t(\mathbf{n}(2,\Omega))\right] \tag{39}$$

$$D(\mathbf{n})=[\,D(\mathbf{n}(1))\quad D(\mathbf{n}(2))\quad\ldots\quad D(\mathbf{n}(U))\,] \tag{40}$$

$$C(\mathbf{n})=[\,C(\mathbf{n}(1))\quad C(\mathbf{n}(2))\quad\ldots\quad C(\mathbf{n}(U))\,] \tag{41}$$

If these are used in PU($\mathbf{n}$), Fig. 2 should be modified: $\breve{x}_t(\mathbf{n})$ should be replaced with $\breve{x}_t(\mathbf{n},\Omega)$. $D(\mathbf{n})$ and $C(\mathbf{n})$ in Fig. 2 should denote those $D(\mathbf{n})$ and $C(\mathbf{n})$ above. Note that the input feature subvector $\breve{x}_t(\mathbf{n})$ in Fig. 2 is not adequate. It should be replaced with $\breve{x}_t(\Omega(\mathbf{n}))$ to provide the the RTS suite of $x_t(\mathbf{n}(u))$ on $\Omega(\mathbf{n}(u))$ for $u = 1,\ldots, U$.

Spike trains for each exogenous feature vector

Recall that a ternary vector $x\{y_\tau(\mathbf{n})\}$ output from a processing unit, PU($\mathbf{n}$), is obtained by converting a representation $y_\tau(\mathbf{n})$ of a probability distribution of a label $r_\tau(\mathbf{n})$ of a feature subvector $x_\tau(\mathbf{n})$. The spike generator in PU($\mathbf{n}$) uses a pseudo-random number generator to do the conversion. If some components of $y_\tau(\mathbf{n})$ are greater than -1 and less than 1, then the corresponding components of$x\{y_\tau(\mathbf{n})\}$ generated by the pseudo-random number generator contain uncertainty (i.e., pseudo-randomness), which reflects probabilistic information contained in $y_\tau(\mathbf{n})$.

In retrieving, when a PU receives a feature subvector with such components with uncertainty, it uses masking matrices or general masking matrices to suppress or "filter out" those components that make the received feature subvector inconsistent with those stored in its ECMs or GECMs and to find a match between the received feature subvector and feature subvectors stored in those ECMs or GECMs. Masking matrices are described in "Masking matrices".

However, there is a chance for pseudo-random number generators to generate a ternary vector $x\{y_\tau(\mathbf{n})\}$ that is an outlier for the probability distribution $y_\tau(\mathbf{n})$. As $x\{y_\tau(\mathbf{n})\}$ is used as a label in unsupervised learning in PU($\mathbf{n}$) and is feedforwarded or feedbacked as inputs to PUs, such an outlier may have undesirable effects on learning and retrieving of THPAM in spite of masking matrices. To minimize such undesirable effects and to represent the subjective probabilities involved in the PUs, we let a THPAM complete a certain number of rounds of retrieving and learning for each exogenous feature vector x_τ^{ex} so that many versions of $x\{y_\tau(\mathbf{n})\}$ are generated and learned by each PU for the same x_τ^{ex}.

The subscript t or τ in $x_\tau(\mathbf{n})$, $y_\tau(\mathbf{n})$, and $x\{y_\tau(\mathbf{n})\}$ denote the time or numbering of the quantities going through PU($\mathbf{n}$) in Sections "A recurrent multilayer network of processing units" and "Expansion correlation matrices to Learning to recognize rotated, translated or scaled patterns". In the rest of this section, assume that each exogenous feature vector is presented to THPAM for one unit of time, and that during this one unit of time, there are ζ spikes in each spike train. Here, the subscript t or τ denotes only the time the exogenous feature vector x_t^{ex} or x_τ^{ex} arrives at the input terminals of THPAM. For each exogenous feature vector x_τ^{ex}, ζ rounds of retrieving and learning are performed by THPAM at times, $\tau + i/\zeta$,, $i = 0, 1, \zeta -1$. Consequently, PU($\mathbf{n}$) generates a sequence of ternary vectors denoted by $x\{y_{\tau+i/\zeta}(\mathbf{n})\}$, $i = 0, 1,\ldots, \zeta -1$, for each exogenous feature vector x_τ^{ex}. This sequence consists of R spike trains, each having ζ spikes each of $1/\zeta$ unit of time.

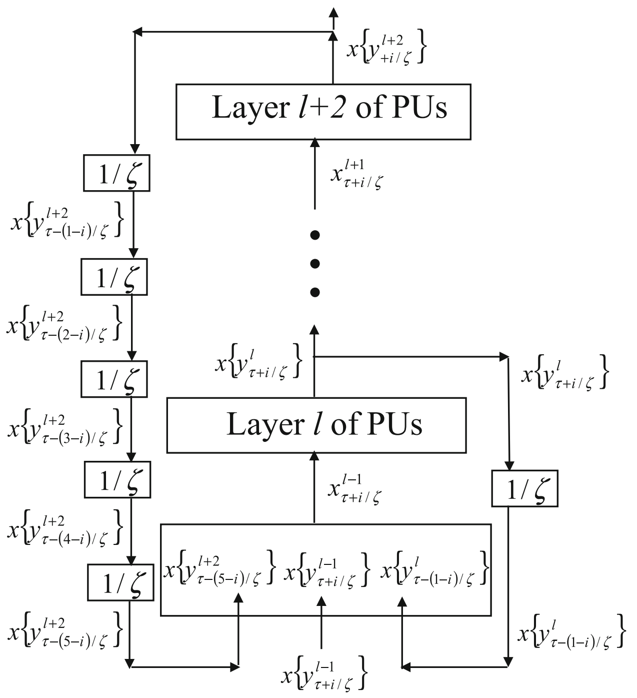
A feedback connection from layer $l + k$ to layer l for $k \geq 0$ must have at least one delay device to ensure stability. Each delay device holds a spike for $1/\zeta$ unit of time before it is allowed to pass. Causes in patterns, temporal or spatial, usually form a hierarchy. Example 1: Phonemes, words, phrases, sentences, and paragraphs in speech. Example 2: Notes, intervals, melodic phrases, and songs in music. Example 3: Bananas, apples, peaches, salt shaker, pepper shaker, Tabasco, fruit basket, condiment tray, table, refrigerator, water sink, and kitchen in a house. Note that although Example 3 is a spatial hierarchy, when one looks around in the kitchen, the images received by the person's retina form a temporal hierarchy.

The higher a layer in THPAM is, the higher in the hierarchy the causes the PUs in the layer treat, and the more time it takes for the causes to form and be recognized by the PUs. Therefore, the number of delay devices on a feedback connection is a monotone increasing function of k. This requirement is consistent with the workings in a biological neural network in the cortex. Note that it takes time (1) for PUs to process feature subvector, (2) for spikes to travel along feedforward connections from a layer to the next layer, and (3) for spikes to travel along feedback connections from a layer to the same or lower-numbered layer. Note also that the times taken for (1) and (2) can be ignored in the feedforward connections, because the subscripts of the input vector $x_{\tau+i/\zeta}^{l-1}$ and output vector $x\{y_{\tau+i/\zeta}^{l}\}$ of all layers are the same. However, a feedback $x\{y_{\tau+i/\zeta-j/\zeta}^{l+k}\}$ from layer $l+k$ to layer l for inclusion in $x_{\tau+i/\zeta}^{l-1}$ must have a delay j that is proportional to the sum of the times taken for (1), (2) and (3) from the input terminals of layer l back to the same input terminals.

For illustration, two examples are given in the following:

Example 5 Let us set the number N_d of delay devices on the feedback connection from layer $l+k$ to layer l, and the number ζ of learning and retriving rounds to be equal to $1+2k$ and 8 respectively, and consider the feedback connections for $k = 0$, 2, 3 and 7. Figure 5 shows layer $l+2$ and layer l in the THPAM. The feature subvector $x_{\tau+i/\zeta}^{l-1}$ input to layer l consists $x\{y_{\tau+i/\zeta}^{l-1}\}$ feedforwarded from layer $l-1$ and feedbacks from layer l and layers above it. Only the feedback $x\{y_{\tau-(1-i)/\zeta}^{l}\}$ from layer l and the feedback $x\{y_{\tau-(5-i)/\zeta}^{l+2}\}$ from layer $l+2$ are shown in the figure. The boxes containing $1/\zeta$ are delay devises with delay duration $1/\zeta$.

- On the feedback connection from layer l to layer l ($k = 0$): There is 1 delay device on the connection. At the time τ, the exogenous feature vector x_{τ}^{ex} arrives, the feedback $x\{y_{\tau-1/\zeta}^{l}\}$ is the last output from layer l in response to the preceding exogenous feature vector $x_{\tau-1}^{ex}$. At time $\tau + i/\zeta$ for $i = 1,\ldots, 7$, the feedback is $x\{y_{\tau+(i-1)/\zeta}^{l}\}$, which is an output from layer l in response to the same exogenous feature vector x_{τ}^{ex}. This feedback connection is shown on the right side of Fig. 5.
- On the feedback connection from layer $l+2$ to layer l ($k = 2$): There are five delay devices on the connection. At time τ, the exogenous feature vector x_{τ}^{ex} arrives at the input terminals of THPAM, and the five delay devices on the feedback connection holds the 5 feedbacks, $x\{y_{\tau-5/\zeta}^{l+3}\}$, $x\{y_{\tau-4/\zeta}^{l+3}\},\ldots,x\{y_{\tau-1/\zeta}^{l+3}\}$, which are outputs from layer $l+2$ in response to the preceding exogenous feature vector $x_{\tau-1}^{ex}$. During the presence of x_{τ}^{ex}, these five feedbacks are respectively included in the first five feature vectors, x_{τ}^{l-1}, $x_{\tau+1/\zeta}^{l-1},\ldots,x_{\tau+4/\zeta}^{l-1}$, input to layer l. The next 3 inputs, $x_{\tau+5/\zeta}^{l-1}$, $x_{\tau+6/\zeta}^{l-1}$, $x_{\tau+7/\zeta}^{l-1}$, to layer l include respectively the feedbacks, $x\{y_{\tau}^{l+3}\}$, $x\{y_{\tau+1/\zeta}^{l+3}\}$, $x\{y_{\tau+2/\zeta}^{l+3}\}$, output from layer $l+3$ in response to x_{τ}^{ex}. This feedback connection is shown on the right side of Fig. 5.
- On the feedback connection from layer $l+3$ to layer l ($k = 3$): There are 7 delay devices on the connection. At time τ, the exogenous feature vector x_{τ}^{ex} arrives at the input terminals of THPAM, and the 7 delay devices on the feedback connection holds the 7 feedbacks, $x\{y_{\tau-7/\zeta}^{l+3}\},\ldots,x\{y_{\tau-6/\zeta}^{l+3}\},\ldots,x\{y_{\tau-1/\zeta}^{l+3}\}$, which are outputs from layer $l+3$ in response to the preceding exogenous feature vector $x_{\tau-1}^{ex}$. During the presence of x_{τ}^{ex}, these 7 feedbacks are respectively included in the first 7 feature vectors, x_{τ}^{l-1}, $x_{\tau+1/\zeta}^{l-1}$, $x_{\tau+6/\zeta}^{l-1}$, input to layer l. The eighth input $x_{\tau+7/\zeta}^{l-1}$ to layer l includes the feedback $x\{y_{\tau}^{l+3}\}$ output from layer $l+3$ in response to x_{τ}^{ex}.
- On the feedback connection from layer $l+3$ to layer l ($k = 7$): There are 15 delay devices on the connection. At time τ, the exogenous feature vector x_{τ}^{ex} arrives at the input terminals of THPAM, and the 15 delay devices on the feedback connection holds the 15



$x\{y_{+i/\zeta}^{l+2}\}$
Layer l+2 of PUs
$1/\zeta$
$x_{\tau+i/\zeta}^{l+1}$
$x\{y_{\tau-(1-i)/\zeta}^{l+2}\}$
$1/\zeta$
$x\{y_{\tau-(2-i)/\zeta}^{l+2}\}$
$x\{y_{\tau+i/\zeta}^{l}\}$
$x\{y_{\tau+i/\zeta}^{l}\}$
$1/\zeta$
Layer l of PUs
$x\{y_{\tau-(3-i)/\zeta}^{l+2}\}$
$1/\zeta$
$x_{\tau+i/\zeta}^{l-1}$
$1/\zeta$
$x\{y_{\tau-(4-i)/\zeta}^{l+2}\}$
$1/\zeta$
$x\{y_{\tau-(5-i)/\zeta}^{l+2}\}$ $x\{y_{\tau+i/\zeta}^{l-1}\}$ $x\{y_{\tau-(1-i)/\zeta}^{l}\}$
$x\{y_{\tau-(5-i)/\zeta}^{l+2}\}$
$x\{y_{\tau-(1-i)/\zeta}^{l}\}$
$x\{y_{\tau+i/\zeta}^{l-1}\}$

Fig. 5 Layer l and layer $l+2$ of an example THPAM with feedback connections from layer l to layer l and from layer $l+2$ to layer l. For each exogenous feature vector x_{τ}^{ex}, ζ rounds of retrieving and learning are performed by each PU in THPAM at times, $\tau + i/\zeta$,, $i = 0, 1,\ldots, \zeta - 1$. The outputs of a PU form R spike trains

feedbacks, $x\{y^{l+3}_{\tau-15/\zeta}\},\ldots,x\{y^{l+3}_{\tau-14/\zeta}\},\ldots,x\{y^{l+3}_{\tau-1/\zeta}\}$. The first 7 of them are outputs from layer $l+7$ in response to the exogenous feature vector $x^{ex}_{\tau-2}$. The next 1 of them, $x\{y^{l+3}_{\tau-8/\zeta}\}$, is the first output from layer $l+7$ in response to the exogenous feature vector $x^{ex}_{\tau-1}$. During the presence of x^{ex}_{τ}, these 8 feedbacks are respectively included in the 8 feature vectors, $x^{l-1}_{\tau}, x^{l-1}_{\tau+1/\zeta},\ldots,x^{l-1}_{\tau+8/\zeta}$, input to layer l in response to x^{ex}_{τ}.

During the presence of the exogenous feature vector x^{ex}_{τ}, the feedbacks, output from layer $l+k$ in response to x^{ex}_{τ}, provide spatial associative information; and the feedbacks, output from layer $l+k$ in response to $x^{ex}_{\tau-1}$ provide less spatial and more temporal associative information. The further back that feedbacks are from, the less spatial and more temporal associative information is used in processing the current x^{ex}_{τ}. Of course, if an exogenous feature vector is presented to THPAM for a large number of time units, all the feedbacks are actually from the same exogenous feature vector, and spatial associative information is thoroughly utilized by the use of the feedback connections.

Conclusion

The temporal hierarchical probabilistic associative memory (THPAM), proposed in this paper, is the only single mathematical model of biological neural networks that has all the eight features and answers coherently all the eight questions listed in the introductory section "Introduction". John von Neumann said: "We require exquisite numerical precision over many logical steps to achieve what brains accomplish in very few short steps" in his well-known 1958 book, *The Computer and the Brain* (von Neumann 1958). Showing that it is possible to achieve so many functions of biological neural networks in a few short logical steps by a single functional model is a small but perhaps significant step towards unraveling the brain.

THPAM's mathematical structures, functions and their processing operations are hypothesized to be low-order approximates of those of biological neural networks. The integration of them, THPAM, is hypothesized to be a low-order approximate of the biological neural networks themselves. These hypotheses have been under examination. Insight into the inner workings and interactions of the components of a biological neural network is expected to be gained through the examination.

The work reported in this paper points to three research directions:

1. Examine the components and processing operations of THPAM as biological hypotheses. If possible, justify these hypotheses to establish THPAM as a macroscopic model or low-order approximate of biological neuronal models.
2. Expand and modify THPAM into functional models of the visual, auditory, somatosensory and (premotor, primary and supplementary) motor cortices.
3. Test and apply THPAM to such applications as face detection and recognition, radiograph reading, baggage examination, financial time series prediction, video monitoring, text understanding, prostheses, etc.

References

Arbib MA (2003) The handbook of brain theory and neural networks, 2nd edn. The MIT Press, Cambridge

Bengio Y, Lamblin P, Popovici D, Larochelle H (2007) Greedy layer-wise training of deep networks. In: Advances in neural information processing systems. The handbook of brain theory and neural networks, MIT Press, Cambridge

Bengio Y, LeCun Y (2007) Scaling learning algorithms towards AI. In: Bottou et al. (eds) Large-scale kernel machines. The MIT Press, Cambridge

Bishop CM (2006) Pattern recognition and machine learning. Springer Science, New York

Dayan P, Abbott LF (2001) Theoretical neuroscience: computational and mathematical modeling of neural systems. The MIT Press, Cambridge

Desjardins G, Bengio Y (2008) Empirical evaluation of convolutional rbms for vision. Technical Report 1327, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal

Erhan D, Bengio Y, Courville A, Manzagol P, Vincent P, Bengio S (2010) Why does unsupervised pre-training help deep learning? J Mach Learn Res Appear

Fromherz P, Gaede V (1993) Exclusive-or function of single arborized neuron. Biol Cybern 69:337–334

Geist J, Wilkinson R, Janet S, Grother P, Hammond B, Larsen N, Klear R, Matsko M, Burges C, Creecy R, Hull J, Vogl, Wilson C (1994) The second census optical charater recognition systems conference. Technical Report NIST 5452, National Institute of Standards and Technology, May 1994

George D, Hawkins J (2009) Towards a mathematical theory of cortical micro-circuits. PLoS Comput Biol 5–10:1–26

Granger R (2006) Engines of the brain: the computational instruction set of human cognition. AI Mag 27:15–31

Grossberg S (2007) Towards a unified theory of neocortex: laminar cortical circuits for vision and cognition. Prog Brain Res 165:79–104

Hassoun MH (1993) Associative neural memories, theory and implementation. Oxford University Press, New York

Hawkins J (2004) On intelligence. Henry Holt and Company, New York

Haykin S (2009) Neural networks and learning machine, 3rd edn. Pretice Hall, Upper Saddle River

Hecht-Nielsen R (2007) Confabulation theory. Springer, New York

Hecht-Nielsen R, McKenna T (2003) Computational models for neuroscience. Springer, New York

Hinton GE, Anderson JA (1989) Parallel models of associative memory. Lawrence Erlbaum Associates, Hllsdale

Hinton GE, Osindero S, Teh Y (2006) A fast learning algorithm for deep belief nets. Neural Comput 313:504–507

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 18:1527–1554

Kandel ER, Schwartz JH, Jessell TM (2000) Principles of neural science. McGraw-Hill, New York

Koch C (1999) Biophysics of computation. Oxford University Press, Oxford

Kohonen T (1988) Self-organization and associative memory. Springer, New York

LeCun Y, Bose B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1–4:541–551

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. In: Proceedings of The IEEE 86(11):2278–2324

Levitan IB, Kaczmarek LK (1993) Simulated annealing and boltzmann machines. Wiley, London

Martin KAC (2002) Microcircuits in visual cortex. Curr Opinion Neurobiol 12–4:418–425

Mel BW (1994) Information processing in dendritic trees. Neural Comput 6:1031–1085

Mel BW (2002) Have we been hebbing down the wrong path? Neuron 34:275–288

Mountcastle VB (1978) An organizing principle for cerebral function: the unit model and the distributed system. In: Edelman GM, Mountcastle VB (eds) The mindful brain. MIT Press, Cambridge

Oja E (1982) A simplified neuron nodel as a principal component analyzer. J Math Biol 15:267–273

O'Reilly R, Munakata Y (2000) Computational explorations in cognitive neuroscience. The MIT Press, Cambridge

Principe JC, Euliano NR, Lefebvre WC (2000) Neural and adaptive systems: fundamentals through simulations. Wiley, New York

Ranzato M, Huang F, Boureau Y, LeCun Y (2007) Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: Proceedings of computer vision and pattern recognition conference (CVPR 2007). IEEE Press, New York

Rieke F, Warland D, de Ruyter R, van Steveninck, Bialek W (1999) Spikes: exploring the neural code. The MIT Press, Cambridge

Salakhutdinov R, Hinton G (2009) Deep Boltzmann machines. In: Proceedings of the 12th international conference on artificial intelligence and statistics (AISTATS) volume 5. Clearwater Beach, Florida, pp 448–455

Simard P, Steinkraus D, Platt JC (2003) Best practices for convolutional neural networks. In: Proceedings of the seventh international conference on document analysis and recognition 2:958–962

Slepian D (1956) A class of binary signaling alphabets. Bell Syst Tech J 35:203

Stuart G, Nelson S, Hausser M (2008) Dendrites, 2nd edn. Oxford University Press, New York

von Neumann J (1958) The computer and the brain. Yale University Press, New Haven

Wilkinson R, Geist J, Janet S, Grother P, Burges C, Creecy R, Hammond B, Hull J, Larsen N, Vogl, Wilson C (1992) The first census optical charater recognition systems conference. Technical Report NIST 4912, National Institute of Standards and Technology, August 1992

Zador AM, Clairborne BJ, Brown TH (1992) Nonlinear pattern separation in single hippocampal neurons with active dendritic membrane. In: Moody J, Lippmann R (eds) Advances in neural information processing systems, vol. 4. Morgan Kaufmann, San Mateo, pp 51–58