2010 12th International Workshop on Cellular Nanoscale
Networks and their Applications (CNNA)

Leaning Theory of Cellular Neural Networks based on Covariance Structural Analysis

Mamoru Tanaka
Sophia University

Hisashi Aomori
Tokyo University of Science

Yoshifumi Nishio
The University of Tokushima

Kumiko Oshima and Martin Hasler
Ecole Polytechnique Federale de Lausanne(EPFL)

***Abstract*—This paper describes a learning theory of the CNN based on the covariance structure analysis using new numerical integral methods. In general, a Cellular Neural Network (CNN) is defined as a local connected circuit which has continuous state variables $x \in R^n$. The importance is in that the piece-wise linear function of the CNN has a linear region $|x| \leq 1$ for $x \in \boldsymbol{x}$ because the learning method can be constructed only in linear state and measurement equations, and because the linear region can be quantized from the continuous variable x to the multi-level quantized variable$f(x)$ by each 1-bit $\Sigma\delta$ modulator which is corresponding to a spiking neuron model. That is, our purpose is to determine the weight parameters $\boldsymbol{\theta}$ in the connection matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{T}$ and e by the machine learning method for equilibrium points of the CNN states equation $\dot{x} = 0$. The covariance structure for the equilibrium point to the linear region will be constructed based on extended Chua's CNN theorem to have symmetric edges for $a_{ij} = a_{ji}$ and asymmetric one-way edge $a_{ij} \neq 0$ for $a_{ji} = 0$ for A-matrix $\boldsymbol{A} = [a_{ij}]$.**

I. Introduction

Recently, neural machine learning methods have been used as data mining to acquire the important information from massive amount of data and to predict future. That is, the data mining is to construct 'model' for classification, prediction, I/O mapping and association by using machine learning algorithm. However, the data mining has not been combined to a circuit model based on the CNN [2], because the CNN learning methods have not been proposed. Also, from circuit viewpoints, it is very important to construct various spiking neuron circuits because the connections will be dynamically on a large reconfigurable FPGA device. The circuit will be used in the field of neural prosthesis. Though various spiking neuron models [3] which are simplified Hodgkin-Huxley models have been proposed, the structure has not been built from the general learning. If each cell in the CNN can be changed to a $\Sigma\delta$ modulator [1], the CNN with learning will be a superior spiking neuron model to a brain computer. The problem is how we construct a generalized $\Sigma\delta$ CNN from data. This paper describes a machine learning of the generalized CNN based on the covariance structure analysis which includes factor analysis. The explicit or implicit numerical integral methods are used for the learning. In the latter case, at each learning iteration step for quasi-Newton method, the inverse Hessian matrix approximation for the matrix including second-order partial derivatives is updated by using Davidon-Fletcher-Powell (DFP) or Groyden-Fletcher-Goldfad-Shanno (BFGS) method. By using both of Backward Euler method and DFP(or BFGS) method, the solution for stiff systems in the parameter space can be obtained. That is, our purpose is to determine the weight parameters $\boldsymbol{\theta}$ in the connection matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{T}$ and $\boldsymbol{e}$ by the machine learning method for equilibrium points in linear region of the CNN states equation $\dot{\boldsymbol{x}} = 0$. The Backward Euler method, one of methods for solving a nonlinear differential equation, is used in CNN parameter space. The important point is that this method can solve stiff systems which have large difference among eigenvalues. The resulted parameters are used as weights on edges in the cellular signal flow graph (SFG) which works as a CNN circuit model with V-I transformation devices and as a prediction model for unknown input data. Our simulation for the model shows good result. This machine learning algorithm will be in SDP(Sophia Dynamics Program) which has been used as nonlinear circuit simulator based on Relaxation-based algorithm with Fast Automatic Differentiation. The SFG will be constructed from netlist of the SDP in future.

II. CNN State and Measurement Equations

The observed variable $\boldsymbol{y}$ is a visible information data which has been obtained from real human behavior, natural environment and so on. The mean $\boldsymbol{\mu}$ and the real covariance matrix $\boldsymbol{S} = E((\boldsymbol{y} - \boldsymbol{\mu})(\boldsymbol{y} - \boldsymbol{\mu})') \in \boldsymbol{R}^{l \times l}$ are calculated by using the observed variables $\boldsymbol{y}$.

Let $\boldsymbol{x} \in \boldsymbol{R}^n$,$\boldsymbol{u} \in \boldsymbol{R}^m$ be the state(endogenous latent) and input(exogenous latent) vectors respectively, then the cellular structural equation is expressed by

$$\dot{\boldsymbol{x}} = -\boldsymbol{x} + \boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{T} \tag{1}$$

where $\boldsymbol{A} \in \boldsymbol{R}^{n \times n}$,$\boldsymbol{B} \in \boldsymbol{R}^{n \times m}$ are coefficient weight matrices which express connections between the state variables, and $\boldsymbol{T} \in \boldsymbol{R}^n$ is the dsturbance variables for the state variables. Each nonlinear function of $\boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{R}^n$ is a piece-wise linear function as

$$f(x) = \frac{1}{2}(|x+1| - |x-1|). \tag{2}$$

The cellular measurement equation should be used to express the casual and/or resulted relations between the observed (endogenous and/or exogenous) variables $\boldsymbol{y} \in \boldsymbol{R}^l$ and the

978-1-4244-6678-8/10/$26.00 ©2010 IEEE

latent variables $\boldsymbol{x}$ ($\boldsymbol{u}$). The measurement equation is given by

$$\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{C}\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{D}\boldsymbol{u} + \boldsymbol{e} \tag{3}$$

where $\boldsymbol{C} \in \boldsymbol{R}^{l \times n}$, $\boldsymbol{D} \in \boldsymbol{R}^{l \times m}$ are coefficient weight matrices which express connections between the latent variables $\boldsymbol{x}$,$\boldsymbol{u}$ and the observed variables $\boldsymbol{y}$ for its mean $\boldsymbol{\mu}$, and $\boldsymbol{e} \in \boldsymbol{R}^{l}$ is the error variables for the observed variables.

For example, the matrix $\boldsymbol{A}$ includes a weight element a_{ij} on the edge from a causal cell C_j to the corresponding resulted cell C_i . Generally, all matrices are sparse.

Our purpose is to make a general program which can determine the parameters of $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{T}$ and $\boldsymbol{e}$ by machine learning method for equilibrium points of the CNN states equation $\dot{\boldsymbol{x}} = 0$. It is very important that the model constructed by the proposed learning method becomes a CNN circuit because of high speed prediction and association. Also, it is very important that the multiplication in $\boldsymbol{A}\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{C}\boldsymbol{f}(\boldsymbol{x})$ becomes integer summation by the use of $\Sigma\delta$ modulations in each of which the nonlinear function $f(x) \in \boldsymbol{f}(\boldsymbol{x})$ can be constructed by the sum of local dynamical outputs from 1-bit quantizer (spiking neuron). The nonlinear function based on the $\Sigma\delta$ modulation type of CNN is described equivalently from theoretical viewpoints as the following multi-quantized function:

$$f(x) = \begin{cases} 1 & (x \geq \xi_0) \\ \dfrac{1}{\xi} g(x) & (-\xi_0 \leq x \leq \xi_0) \\ -1 & (x \leq -\xi_0) \end{cases} \tag{4}$$

where

$$g(x) = \begin{cases} \Delta([\dfrac{x}{\Delta} + \dfrac{1}{2}]) & (l \text{ is odd}) \\ \Delta([\dfrac{x}{\Delta}] + \dfrac{1}{2}) & (l \text{ is even}) \end{cases} \tag{5}$$

$$\Delta = \frac{2\xi}{l-1} \tag{6}$$

The parameters Δ and l are quantized step size and the number of quantized levels, respectively. The parameter ξ_0 is the value x to give the definition area of multi-steps function$\frac{1}{\xi}g(x)$ and the parameter ξ is the value x to satisfy $|f(x)| = 1$.

The important problems are how to learn the matrices such that we have equilibrium point in the linear region $|x| \leq 1(\xi = 1)$ corresponding to Chua's theorem $a_{ii} < 1$ ($a_{ii} = \epsilon$ or 0) for multi-level AD conversion region and how to construct the generalized CNN structure.

III. Optimization

A. SDP Program: Minimization of Fit Function

Let $\boldsymbol{z}$ be the model standardized vector, then it is given by

$$\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{\mu}. \tag{7}$$

If the diagonal elements a_{ii} of the matrix $\boldsymbol{A}$ for CNN with Lyapunov energy function are ϵ,the attractor of dynamics for the CNN state equation will be in equilibrium point on the linear region and the equations are expressed for the $l \times l$, $n \times n$ or $(l+n) \times (l+n)$ unit matrix $\boldsymbol{I}$ as follows:

$$\begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{x} \end{pmatrix} =$$

$$\begin{pmatrix} \boldsymbol{0} & \boldsymbol{C} \\ \boldsymbol{0} & \boldsymbol{A} \end{pmatrix} \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{f}(\boldsymbol{x}) \end{pmatrix} + \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{D} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{B} \end{pmatrix} \begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{T} \\ \boldsymbol{u} \end{pmatrix} \tag{8}$$

The network is not symmetric. Therefore, for asymmetric (causal and resulted) connection, if $a_{ij} \neq 0$, we set $a_{ji} = 0$ for its equilibrium. There is no equilibrium problem for symmetric (covariance) connections according to Chua's theorem[2]. And if $a_{ii} = \epsilon \ll 1$, it seems that each state variable x is in the equilibrium point of linear region as $|x| \leq 1$ to become $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}$. This will be proved. In the linear region, if we define as

$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \end{pmatrix}$$

$$\boldsymbol{A}_0 = \left(\boldsymbol{I} - \begin{pmatrix} \boldsymbol{0} & \boldsymbol{C} \\ \boldsymbol{0} & \boldsymbol{A} \end{pmatrix} \right)^{-1}$$

$$\boldsymbol{B}_0 = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{D} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{B} \end{pmatrix},$$

it is derived that

$$\boldsymbol{z} = \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0 \begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{T} \\ \boldsymbol{u} \end{pmatrix}, \tag{9}$$

where $(\boldsymbol{e}, \boldsymbol{T}, \boldsymbol{u})'$ is an exogenous(source) vector.

Let $\boldsymbol{\Sigma} \in \boldsymbol{R}^{l \times l}$ be the covariance matrix $E(\boldsymbol{z}\boldsymbol{z}')$ for the state variables in linear region of the piece-wise linear function, then it is derived theoretically from the cellular structural and measurement equations as follow:

$$\boldsymbol{\Sigma} = \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}' \tag{10}$$

where $\boldsymbol{\Phi}_0$ defined as the covariance matrix of the vector $(\boldsymbol{e}, \boldsymbol{T}, \boldsymbol{u})'$ is a $(l+n+m) \times (l+n+m)$ diagonal block matrix by assumption as

$$\boldsymbol{\Phi}_0 = \begin{pmatrix} \boldsymbol{e}\boldsymbol{e}' & 0 & 0 \\ 0 & \boldsymbol{T}\boldsymbol{T}' & 0 \\ 0 & 0 & \boldsymbol{u}\boldsymbol{u}' \end{pmatrix}.$$

To use in (14), we define that $\boldsymbol{A}_! = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{C} \\ \boldsymbol{0} & \boldsymbol{A} \end{pmatrix}$.

By using the real covariance matrix $\boldsymbol{S} \in \boldsymbol{R}^{l \times l}$ and a parameter element vector $\boldsymbol{\theta} \in \boldsymbol{R}^{p}$, the fit function is defined as

$$f_c(\boldsymbol{\theta}) = \frac{1}{2} tr((\boldsymbol{S} - \boldsymbol{\Sigma})\boldsymbol{S}^{-1})^2. \tag{11}$$

where $tr(\mathbf{X})$ means the trace of a matrix. In order to minimize the fit function, the function $\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\partial f_c(\boldsymbol{\theta})}{\partial \theta_i}$ is changed to

$$\frac{\partial f_c(\boldsymbol{\theta})}{\partial \theta_i} = tr\left(\frac{\partial f_c(\boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})} \frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_i} \right). \tag{12}$$

In this equation, we derive that

$$\frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \quad = \quad \boldsymbol{S}^{-1}(\boldsymbol{\Sigma} - \boldsymbol{S})\boldsymbol{S}^{-1}.$$

For the element ϕ_{ij} of the matrix $\boldsymbol{\Phi}_0$, we derive as

$$\begin{aligned}
\frac{\partial f_c}{\partial \phi_{ij}} &= tr\left(\frac{\partial f}{\partial \boldsymbol{\Sigma}} \frac{\partial \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}'}{\partial \phi_{ij}}\right) \\
&= tr\left(\frac{\partial f}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0(\boldsymbol{I}_{ij} + \boldsymbol{I}_{ji})\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}'\right) \\
&= tr\left(\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0(\boldsymbol{I}_{ij} + \boldsymbol{I}_{ji})\right) \\
&= \left(\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\right)_{ij} \times 2
\end{aligned}$$

where $\boldsymbol{I}_{ij}$ represents a matrix in which only (i, j)-element is 1 and others are 0. For the matrix $\boldsymbol{\Phi}_0$,

$$\frac{\partial f_c}{\partial \boldsymbol{\Phi}_0} = 2\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0.$$

For the element b_{ij}^0 of the matrix $\boldsymbol{B}_0$, we derive as

$$\begin{aligned}
\frac{\partial f_c}{\partial b_{ij}^0} &= tr\left(\frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \frac{\partial \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}'}{\partial b_{ij}^0}\right) \\
&= tr\left(\frac{\partial f}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0 \frac{\partial \boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'}{\partial b_{ij}^0} \boldsymbol{A}_0'\boldsymbol{E}'\right) \\
&= tr\left(\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0(I_{ij}\boldsymbol{\Phi}_0\boldsymbol{B}_0' + \boldsymbol{B}_0\boldsymbol{\Phi}_0 I_{ji}')\right) \\
&= \left(\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\right)_{ij} \times 2.
\end{aligned}$$

For the matirx $\boldsymbol{B}_0$,

$$\frac{\partial f_c}{\partial \boldsymbol{B}_0} = 2\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0. \tag{13}$$

For the element α_{ij} of the matrix $\boldsymbol{A}_!$, considering

$$\frac{d\boldsymbol{A}_0}{d\alpha_{ij}} = -\boldsymbol{A}_0(-\frac{d\boldsymbol{A}_!}{d\alpha_{ij}})\boldsymbol{A}_0, \quad tr\,(\boldsymbol{P}\boldsymbol{Q}) = tr\,(\boldsymbol{Q}\boldsymbol{P})\,,$$

we derive as

$$\begin{aligned}
\frac{\partial f_c}{\partial \alpha_{ij}} &= tr\left(\frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \frac{\partial \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0'\boldsymbol{E}'}{\partial \alpha_{ij}}\right) \\
&= tr\left(\frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}(\frac{\partial \boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0'}{\partial \alpha_{ij}})\boldsymbol{E}'\right) \\
&= tr(\frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}(\boldsymbol{A}_0 I_{ij}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0' \\
&\qquad + \boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0' I_{ji}\boldsymbol{A}_0')\boldsymbol{E}') \\
&= \left(\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0'\right)_{ij} \times 2.
\end{aligned}$$

For the matrix $\boldsymbol{A}_!$,

$$\frac{\partial f_c}{\partial \boldsymbol{A}_!} = 2\boldsymbol{A}_0'\boldsymbol{E}' \frac{\partial f_c}{\partial \boldsymbol{\Sigma}} \boldsymbol{E}\boldsymbol{A}_0\boldsymbol{B}_0\boldsymbol{\Phi}_0\boldsymbol{B}_0'\boldsymbol{A}_0'. \tag{14}$$

B. SDP Program: Nolinear Solution

The function (11) should be minimized such that the parameters at that time are determined. So we want to solve the equation:

$$\mathbf{g}(\boldsymbol{\theta}) = 0 \tag{15}$$

where $\mathbf{g}(\boldsymbol{\theta}) = \frac{\partial f_c}{\partial \boldsymbol{\theta}}$.

However, the convergence depends on initial value when the nonlinear equation (15) is solved by an iterative solution method. In order to escape the initial problem, we solve the following equation by using Backward Euler method.

$$\dot{\boldsymbol{\theta}} = -\mathbf{g}(\boldsymbol{\theta}) \tag{16}$$

The Backward Euler method is an implicit numerical integral method for solving stiff systems. This method approximates the solution at virtual time $t_{k+1} = t_k + h$ by solving the implicit equation:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - hg(\boldsymbol{\theta}_{k+1}) \tag{17}$$

where the gradient vector $\mathbf{g}(\boldsymbol{\theta}_k)$ is evaluated at $\boldsymbol{\theta}_k$.

Since this equation(17) may be nonlinear, solving it in general requires an iterative solution method. An implicit method requires the solution of the nonlinear equation at each time step. For each step of the Backward Euler method, we use the quasi-Newton method.
Let

$$\mathbf{F}(\boldsymbol{\theta}_{k+1}) = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k + h\mathbf{g}(\boldsymbol{\theta}_{k+1}), \tag{18}$$

then the Newton method is described as

$${}^{(n+1)}\boldsymbol{\theta}_{k+1} = {}^{(n)}\boldsymbol{\theta}_{k+1} - (\frac{\partial \mathbf{F}({}^{(n)}\boldsymbol{\theta}_{k+1})}{\partial^{(n)}\boldsymbol{\theta}_{k+1}})^{-1}\mathbf{F}({}^{(n)}\boldsymbol{\theta}_{k+1}). \tag{19}$$

The inverse matrix computation of the Jacobi matrix $(\mathbf{I} + h\frac{\partial \mathbf{g}({}^{(n)}\boldsymbol{\theta}_{k+1})}{\partial^{(n)}\boldsymbol{\theta}_{k+1}})^{-1}$ is not available or expensive. Then the approximation technique is used. The Hessian matrix $\mathbf{H}(\boldsymbol{\theta}_n) \in \mathbf{R}^{p\times p}$ which is correspoding to the Jacobi matrix is a second-order partial derivatives of the function f_c with respect to $\boldsymbol{\theta}_n$. It is important to use the inverse of the Hessian matrix . However, since the Hessian leads to algorithmic and computational complexities, an approximation technique of the inverse Hessian is often used. We use Davidon-Fletcher-Powell (DFP) or Broyden-Fletcher-Goldfab-Shanno (BFGS) method which is one of quasi-Newton methods. The update formula is as follows:

DFP method:

$$\mathbf{H}_{n+1} = \mathbf{H}_n + \frac{\mathbf{r}\mathbf{r}'}{\mathbf{r}'\mathbf{d}} - \frac{\mathbf{H}_n'\mathbf{d}\mathbf{d}'\mathbf{H}_n}{\mathbf{d}'\mathbf{H}_n\mathbf{d}} \tag{20}$$

BFGS method:

$$\mathbf{H}_{n+1} = \mathbf{H}_n + (\mathbf{1} + \frac{\mathbf{d}'\mathbf{H_n}\mathbf{d}}{\mathbf{r}'\mathbf{d}})\frac{\mathbf{r}\mathbf{r}'}{\mathbf{r}'\mathbf{d}} - \frac{\mathbf{r}\mathbf{d}'\mathbf{H_n} + \mathbf{H_n}\mathbf{d}\mathbf{r}'}{\mathbf{d}'\mathbf{r}} \tag{21}$$

where

$$\boldsymbol{r} = -\alpha\boldsymbol{H}_n F(\boldsymbol{\theta}_{k+1}^{(n)}) \quad \boldsymbol{d} = F(\boldsymbol{\theta}_{k+1}^{(n+1)}) - F(\boldsymbol{\theta}_{k+1}^{(n)}).$$

IV. Circuit Model

A circuit model which is constructed by the learning algorithm becomes a general CNN circuit described as

State Equation

$$C\frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x}x_{ij}(t) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(t)$$
$$+ \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl} + T \tag{22}$$

Output

$$y_{ij}(t) = \frac{1}{2}(|x_{ij}+1| - |x_{ij}-1|) \tag{23}$$

where a cell $C(i,j)$ is placed in pixel $P(i,j)$ in 2D-plane.

Definition 1: A connection between a cell $C(k,l)$ and a cell $C(i,j)$ in the CNN is called symmetry for $A_1(i,j;k,l) = A_1(k,l;i,j)$ and one-directional asymmetry for that if $A_0(i,j;k,l) \neq 0$ then $A_0(k,l;i,j) = 0$.

For the type of each branch connection, we define each element of A-template by using a binary parameter Ω such that:

$$A_\Omega(i,j;k,l) = \begin{cases} 2A(i,j;k,l) & if \quad \Omega = 0 \\ A(i,j;k,l) & if \quad \Omega = 1 \end{cases} \tag{24}$$

where $\Omega = 0$ and $\Omega = 1$ mean asymmetrical and symmetrical connections respectively.

Definition 2: if any connection is $\Omega = 0$ or $\Omega = 1$, the CNN is called a data-mining CNN.

All Cell have different templates in the data-mining CNN. The weights depends on the learning for the structure determined by a data-mining analyzer. The structure is determined to satisfy the conditions of the definitions. The data-mining CNN must give a stable state in linear region [-1,1] of the piece-wise linear function in parallelism because it is a circuit model for the data mining. The dynamics will be converged to an equilibrium point according to the following theorem.

Theorem 1: The data-mining CNN energy function which is defined as

$$E(t) = -\sum_{(i,j)}\sum_{(k,l)}\{\Omega\frac{1}{2} + (1-\Omega)\}A_\Omega(i,j;k,l)y_{ij}(t)y_{kl}(t)$$
$$+\frac{1}{2R_x}\sum_{(i,j)} y_{ij}(t)^2$$
$$-\sum_{(i,j)}\sum_{(k,l)}B(i,j;k,l)y_{ij}(t)u_{kl} - \sum_{(i,j)} Ty_{ij}(t). \tag{25}$$

Taking it consideration as

$$\frac{dE(t)}{dt} =$$
$$-\sum_{(i,j)}\sum_{(k,l)}\{\Omega + (1-\Omega)\frac{1}{2}\}A_\Omega(i,j;k,l)\frac{dy_{ij}}{dx_{ij}}\frac{dx_{ij}(t)}{dt}y_{kl}(t)$$
$$+\frac{1}{R_x}\sum_{(i,j)}\frac{dy_{ij}}{dx_{ij}}\frac{dx_{ij}(t)}{dt}y_{ij}(t)$$
$$-\sum_{(i,j)}\sum_{(k,l)}B(i,j;k,l)\frac{dy_{ij}}{dx_{ij}}\frac{dx_{ij}(t)}{dt}u_{kl}(t) - \sum_{(i,j)} T\frac{dy_{ij}}{dx_{ij}}\frac{dx_{ij}(t)}{dt}, \tag{26}$$

we can change the A-element $A_\Omega(i,j;k,l)$ to the original A-element. Then we can prove the Lyapunov stability as

$$\frac{dE(t)}{dt} = -\sum_{(i,j)}\frac{dy_{ij}}{dx_{ij}}\frac{dx_{ij}(t)}{dt}$$
$$\cdot[\sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(t) - \frac{1}{R_x}y_{ij}(t)$$
$$+ \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl} + T]$$
$$= -\sum_{|x_{ij}|<1}\frac{dx_{ij}(t)}{dt}\cdot$$
$$\cdot[\sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(t) - \frac{1}{R_x}x_{ij}(t)$$
$$+ \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl} + T \quad] \tag{27}$$

$$\frac{dE(t)}{dt} = -\sum_{|x_{ij}|<1} C\left[\frac{dx_{ij}(t)}{dt}\right]^2 \leq 0 \tag{28}$$

Thorem 2: If

$$A(i,j;i,j) < \frac{1}{R_x} \tag{29}$$

then, each cell converges to the linear region:

The proof should be done such as that in the paper[2].

V. Conclusion

A novel cellular learning analysis method of covariance structure was proposed. It is very important that we build a structure such that the degree of freedom $\frac{l(l+1)}{2}$ - ζ in which ζ is the number of variables to be obtained must be positive.

Acknowledgment

This research by corporation of Sophia, Tokushima and EPFL Universities is supported by the fund of Open Research Center Project from MEXT of Japanese Government (2007-2011).

References

[1] H. Aomori, T. Otake, N. Takahashi, and M. Tanaka," Sigma-detal cellular neural network for 2D modulation," Neural Networks, vol. 21, no. 2?3, pp. 349-357, Mar. 2008.

[2] L.O.Chua and L.Yang, "Cellular neural networks:thorery," IEEE Trans. Circuits Syst., vol, no.10, pp. 1257-1272, Oct. 1988.

[3] E.M. Izhikevich,"Which model to use for cortical spiking neurons?," IEEE Trans. on Neural Networks, vol. 15, pp. 1063-1070,2004.