情報処理学会論文誌 Vol. 49 No. 6 2155–2163 (June 2008)

学習ベクトル量子化ニューラルネットワークの耐故障性向上のための学習法

蓑原 隆[†1]

学習ベクトル量子化（LVQ）ニューラルネットワークは，一般的なフィードフォワード型のニューラルネットワークが適していない大規模なパターン分類問題などの応用について研究されている．多くの場合，汎用の計算機上でニューロンの動作をシミュレートすることが行われるが，対象となる処理に実時間性が要求される場合には，専用のハードウェア化を行うことで高速化が期待できる．一方，ハードウェアの規模の増大につれて故障発生の可能性も高まり，高信頼性が要求される場合には，耐故障化が必要になる．従来のニューラルネットワークの耐故障化の研究の大部分はフィードフォワード型を対象としており，LVQ ニューラルネットワークに対する耐故障化はほとんど行われていない．本論文では，LVQ ニューラルネットワークの耐故障能力を評価する指標を提案するとともに，耐故障能力を向上させる学習方法として，境界強調とカップリング支援のアイデアを述べる．また，これらのアイデアを適用したシステムの評価をシミュレーションによって行う．

Training of Learning Vector Quantization Neural Networks for Improving their Fault Torelance

Takashi Minohara[†1]

The learning vector quantization (LVQ) is a model of neural networks, and it is used for complex pattern classifications in which typical feedforward networks don't give a good performance. Usually, conventional general-purpose serial computers are used for simulating LVQ, it is expected that special-purpose parallel neural-network processors will reduce execution time for real-time applications. However, increasing amount of hardware in relation to the number of neurons enlarges the probability of failure. Fault tolerance is required when the neural networks are used for critical application. Many methods for enhancing the fault tolerance of neural networks have been proposed, but most of them are for feedforward networks. There is scarcely any methods for fault tolerance of LVQ neural networks. In this paper, I have proposed a dependability measure for the LVQ neural networks, and then I have presented two idea, the border emphasis and the encouragement of coupling, to improve the learning algorithm for increasing dependability. The experiment result shows that the proposed algorithm trains networks so that they can achieve high dependability.

1. はじめに

近年，ニューラルネットワークはパターン認識，パターン分類，信号あるいは画像処理など様々な分野に応用され研究されている[1),2)]．多くの場合，ニューラルネットワークの処理は，従来の汎用計算機上でニューロンの動作をシミュレートすることによって行われているが，実時間性が要求される処理が対象となる場合には，専用のハードウェアを利用することで，処理の高速化が期待できる[3)–5)]．しかし，多数の処理ユニットを必要とするニューラルネットワークをハードウェア化したシステムにおいては，システムの一部に故障が発生する可能性も高まり，システムが重要な業務に使用される場合には，耐故障化が不可欠である．

ニューラルネットワークが多数のユニットからなる分散システムであることから，本質的に耐故障性を持つといわれることがあるが，特別な配慮を行わない場合，ユニット数を増加させても，必ずしも高い耐故障性は得られない[6)]．耐故障性を実現するためには，耐故障性を向上させる具体的な仕組みをシステムに与えなければならない．従来，多くのニューラルネットワークの耐故障化の試みが報告されている[7)–9)]が，その大部分は，誤差逆伝搬アルゴリズムを学習に使用するフィードフォワード型のネットワークを対象としている．複数のニューロンの出力を合成することで出力が定まるフィードフォワード型のネットワークが適していない大規模なパターン分類問題などに対して，自己組織化マップ（SOM）や学習ベクトル量子化（LVQ）ネットワークのようにネットワークの入力に対して最も活性化したニューロンが出力を決定する競合学習型のニューラルネットワークの有効性が報告されている[10)]が，ハードウェア化した場合の耐故障性については，Yasunaga らが SOM を WSI 上に実現した場合の故障の影響について評価している[11)]以外には，積極的に耐故障性を向上させようとする研究はほとんど行われていない．

本研究では，学習ベクトル量子化（LVQ）ネットワークについて，勝者がすべてを得るという特徴に適した耐故障性の評価方法を提案したうえで，ネットワークが持つ多数のニュー

†1 拓殖大学工学部情報工学科
Department of Computer Science, Takushoku University

ロンという冗長性を耐故障性の向上に生かすための学習方法を実現することを目的とする．以下，本研究が対象とする学習ベクトル量子化（LVQ）ネットワークについて概説し，対象とするハードウェアの構成と故障のモデルを定める．さらに，LVQ ネットワークに対する故障の影響を評価する指標について述べる．続いて，LVQ の耐故障能力を向上させるための学習アルゴリズムを提案し，提案したアルゴリズムの有効性を実験により評価する．

2. 学習ベクトル量子化ニューラルネットワーク（LVQ）とハードウェアモデル

2.1 LVQ

学習ベクトル量子化（LVQ）は，統計的な分類・認識手法で，入力空間 $\Re^n$ におけるクラス分けの領域を定義することを目的とする．**図 1** に LVQ ネットワークの構成を示す．

各ニューロン N_i は，コードブックベクトル $\mathbf{m}_i = [\mu_{i1}, \mu_{i2}, \cdots, \mu_{in}]^T \in \Re^n$ を持ち，入力ベクトル $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T \in \Re^n$ が分類されるいずれかのクラスに割り当てられている．入力ベクトル $\mathbf{x}$ は，全ニューロンに同時に与えられ，各ニューロンは自分のコードブックベクトルと入力ベクトルの距離をそれぞれ計算する．そうして入力ベクトルは計算結果のうち最も近いコードブックベクトルを持つニューロンの所属するクラスに分類される．

コードブックベクトル $\mathbf{m}_i$ の学習は，以下のように行われる．

まず，c を入力ベクトル $\mathbf{x}$ に最も近いコードブックベクトル $\mathbf{m}_i$ のインデックスとする．すなわち，

$$c = arg \min_i \{d(\mathbf{x}, \mathbf{m}_i)\} \tag{1}$$

ここで，$d(\mathbf{x}, \mathbf{m}_i)$ は，ユークリッド距離 $||\mathbf{x} - \mathbf{m}_i||$ とする．

次に $x(t)$ と $m_i(t)$ を，離散時間系列 $t = 0, 1, 2, \cdots$ に対応する入力ベクトルとコードブックベクトルの値とするとき，次の式によって，コードブックベクトルを更新する．

$$\mathbf{m}_c(t+1) = \mathbf{m}_c(t) + \alpha_c(t)[\mathbf{x}(t) - \mathbf{m}_c(t)]$$
$$\mathbf{x} \text{ と } \mathbf{m}_c \text{ が同一クラスの場合} \tag{2}$$
$$\mathbf{m}_c(t+1) = \mathbf{m}_c(t) - \alpha_c(t)[\mathbf{x}(t) - \mathbf{m}_c(t)]$$
$$\mathbf{x} \text{ と } \mathbf{m}_c \text{ が別クラスの場合} \tag{3}$$
$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t)$$
$$i \neq c \text{ の場合} \tag{4}$$

学習係数 $\alpha_i(t)$ は，$0 < \alpha_i(t) < 1$ の値を持ち，初期値から単調に減少させる（LVQ1 アルゴリズム）か，次の式を再帰的に適用することで最適化する（OLVQ1 アルゴリズム）．

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{1 + s(t)\alpha_c(t-1)} \tag{5}$$

ただし $s(t)$ は，分類が正しく行われたときに $s(t) = 1$ とし，そうでなければ，$s(t) = -1$ とする．

LVQ1 あるいは OLVQ1 アルゴリズムでは，入力ベクトルに最も近いコードブックベクトル（勝利者ベクトル）だけが更新されるが，次点となったコードブックも更新する LVQ3 と呼ばれる拡張も行われている．LVQ3 アルゴリズムでは，$\mathbf{m}_i$ と $\mathbf{m}_j$ を，入力ベクトル $\mathbf{x}$ についての勝利者と次点とすると，$\mathbf{x}$ と $\mathbf{m}_i$ が同じクラスに属し，$\mathbf{x}$ と $\mathbf{m}_j$ が異なったクラスに属する場合，次の式に従ってコードブックベクトルの値を更新する．

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)]$$
$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) - \alpha(t)[\mathbf{x}(t) - \mathbf{m}_j(t)] \tag{6}$$

また，もし，$\mathbf{x}$，$\mathbf{m}_i$ および $\mathbf{m}_j$ の 3 者が同じクラスに属する場合には，両方のコードブックベクトルを係数 ϵ によって入力ベクトルに近づける．

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \epsilon\alpha(t)[\mathbf{x}(t) - \mathbf{m}_i(t)]$$
$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \epsilon\alpha(t)[\mathbf{x}(t) - \mathbf{m}_j(t)] \tag{7}$$

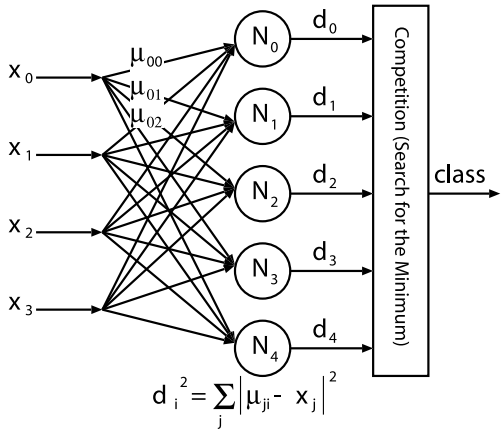

図 1　学習ベクトル量子化ニューラルネットワーク
Fig. 1　Learning Vectore Quantization neural networks.

2.2 ハードウェア構成と故障モデル

本研究では LVQ ニューラルネットワークとして**図 2** に示す SIMD 型のハードウェア構

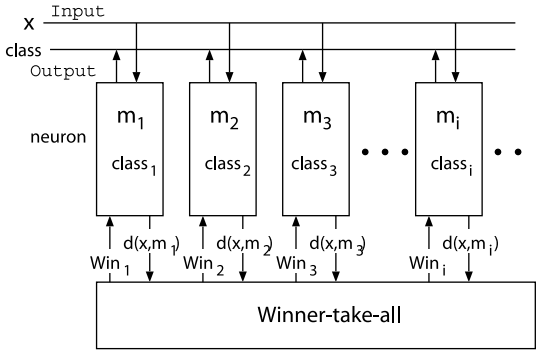

図 2　SIMD 型 LVQ ニーラルネットワークの構成
Fig. 2　Architecture for SIMD type LVQ neural networks.

成[12),13)] を対象とする．各ニューロンに相当するプロセッサはコードブックベクトル $\mathbf{m}$ と所属するクラスの情報を持ち，入力ベクトル $\mathbf{x}$ とコードブックベクトル $\mathbf{m}$ の距離を並列に計算する．各ニューロンで計算された距離は，最小値検出（Winner-take-all）回路によって比較され，各ニューロンに勝敗判定が通知される．勝敗判定の結果として勝者となったニューロンは自身が所属するクラスの情報を出力する．

図 2 の構成は，1) 複数のニューロンプロセッサ，2) SIMD 制御回路，3) 入出力バス，4) 最小値検出回路，の 4 つの部分からなる．このうち 2)〜4) の部分には冗長性がないため，この部分の耐故障化には古典的な多重化によって冗長性を導入する必要がある．このとき，2), 3) の部分については，ニューラルネットワークの対象とする問題の大きさに応じてニューロン数を増やしても回路規模は大きく変化しないと考えられる．また，4) の最小値検出回路は，バスアービタなどに用いられるプライオリティエンコーダの 1 種と考えられ，1 ニューロンあたりのゲート数はニューロン本体に比べて小さい規模で実現できる．たとえばビット直列ワード並列回路（**図 3**）を用いれば，ニューロンあたりのゲート数は非常に少ない．これらのことから，2)〜4) の部分については多重化による耐故障が適していると考えられる．

一方，ニューロンプロセッサは複数の同一ユニットの繰返しであり，システム自体として冗長性を有していると考えられる．したがって，個々のニューロンプロセッサについて，さらに多重構成にすることなしにシステムの耐故障性を実現することが望まれる．ここで，ニューロンプロセッサが故障によって誤った出力を出した場合の影響は，次の 2 つの現象として現れると考えられる．

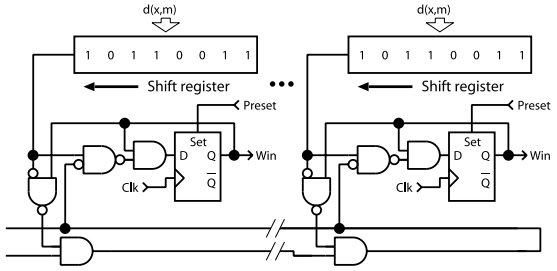

図 3　最小値検出回路
Fig. 3　Winner-take-all circuit for parallel search.

(1)　フォールスポジティブ：故障したニューロンが本来の勝者よりも小さい距離を出力することによって勝者として選ばれる．

(2)　フォールスネガティブ：故障したニューロンが本来の勝者であり，他のニューロンよりも大きい距離を出力することによって敗者になる．

今，最小値検出回路の多重化によって，本来の勝者を検出できるとすると，前者の故障の影響は排除できると考えられる．一方，後者の故障については，故障したニューロンがどのクラスに所属していたかを判別することが期待できないため，本来の勝者の次に小さい距離を出力した次点のニューロンのクラスをネットワークの出力とせざるをえない．本論文では，ニューロン外部の回路の耐故障構成によって救済できないフォールスネガティブのケースに注目し，これ以降，次の故障モデルのもとで議論を進める．

- 故障はニューロンに発生し，故障したニューロンは入力ベクトルに最も近いコードブックベクトルとして選ばれることがない．
- 入力ベクトルは無故障のニューロンのうち，最も近いコードブックベクトルと同じクラスに分類される．

3. LVQ における故障の影響の評価

LVQ における競合によって，入力ベクトルに最も近いコードブックベクトルを持つニューロンのみがネットワークの出力に関連する．したがって，ニューロンの故障によって誤った分類が行われるかどうかは，入力ベクトルに大きく依存する．すなわち，LVQ においては，従来行われているようにニューロンの故障率に対する性能劣化の程度を評価した場合，入力によって活性化されないニューロンが多ければ多いほど高い耐故障性を示すことになり，

本来のネットワークの性能を評価していることにつながらない．そこで，本論文では，入力ベクトルを考慮した耐故障の指標を提案する．

2.2 節の故障モデルにおいて，故障したニューロンが本来，入力ベクトルに最も近いコードブックベクトルを持っていたときのみ故障が活性化され，本来，次点であったニューロンの所属するクラスが故障したニューロンと異なるときのみ誤りが観測される．

今，ニューロン N_i の故障を F_i と表し，故障 F_i の事前確率を $P(F_i)$ とする．また，入力ベクトル $\mathbf{x}$ が与えられる確率密度関数を $p(\mathbf{x})$ とすると，誤り発生の期待値 E は次の式で与えられる．

$$E = \sum_i \int s(F_i, \mathbf{x}) P(F_i) p(\mathbf{x}) d\mathbf{x} \tag{8}$$

ただし，$s(F_i, \mathbf{x})$ は，入力ベクトル $\mathbf{x}$ による故障活性化を表し，次のように定義する．

$$s(F_i, \mathbf{x}) = 1 \quad \mathrm{N_i}\ \text{が}\ \mathbf{x}\ \text{に最も近く，かつ}$$
$$\mathrm{class(N_i)} \neq \text{class（次点ニューロン）} \tag{9}$$
$$s(F_i, \mathbf{x}) = 0 \quad \text{上記以外の場合} \tag{10}$$

多くの場合，入力ベクトルの確率密度関数 $p(\mathbf{x})$ をあらかじめ知ることはできず，トレーニングデータとして，いくつかのサンプルが与えられているにすぎない．しかし，もし，サンプルデータの集合 X が，入力ベクトル空間から $p(\mathbf{x})$ に従って無作為に独立に選ばれた m 個のサンプル（$\mathbf{x}_1, \cdots, \mathbf{x}_m$）からなるとすれば，誤り発生の期待値はサンプルデータを用いて次の式によって近似できる．

$$E(X) = \sum_i \sum_{j=1}^{m} s(F_i, \mathbf{x}_j) P(F_i) \tag{11}$$

ここで，故障の発生確率 $P(F_i)$ がすべてのニューロンについて等しいと仮定すると上式は次のように書き換えられる．

$$E(X)|_{P(F_i)=\lambda} = \lambda \sum_i \sum_{j=1}^{m} s(F_i, \mathbf{x}_j) \tag{12}$$

上式から本研究では，LVQ の耐故障性の指標として，以下の「誤り係数」を定義する．

表 1　アイリスデータについての誤り係数評価
Table 1　$CE(X)$ of the networks trained by the Iris plants database.

# of Neurons		8	9	10	11	12	13
Running Length of Training		400	600	800	1,000	1,200	1,400
Total Accuracy	(mean)	96.73	96.73	97.13	96.80	97.47	97.33
w/o Faults(%)	(s.d.)	0.36	0.38	0.35	0.42	0.24	0.23
$CE(X)$	(mean)	25.5	19.9	19.5	20.2	14.2	17.4
	(s.d.)	1.78	1.79	1.16	1.88	1.49	1.72

表 2　音声認識データについての誤り係数評価
Table 2　$CE(X)$ of the networks trained by the speech signal database.

# of Neurons		150	200	250	300	350	400
Running Length of Training		6,000	8,000	10,000	12,000	14,000	16,000
Total Accuracy	(mean)	92.88	93.73	94.32	94.81	95.10	95.31
w/o Faults(%)	(s.d.)	0.10	0.14	0.08	0.10	0.03	0.05
$CE(X)$	(mean)	201.1	183.9	181.0	175.9	168.6	162.3
	(s.d.)	4.92	2.34	2.85	4.80	2.12	3.44

$$CE(X) = \sum_i \sum_{j=1}^{m} s(F_i, \mathbf{x}_j) \tag{13}$$

サンプルデータが実際の入力の確率密度分布を反映しているとすれば，この誤り係数 $CE(X)$ が，実際の誤りの発生確率を反映することになると考えられる．

提案した誤り係数による LVQ ニューラルネットワークの耐故障性能の評価として，プログラムパッケージ “LVQ_PAK” [14),15)] の OLVQ1 アルゴリズムを用いて学習したネットワークについて $CE(X)$ を求めた．

学習に使用したデータは，アイリスデータ[16)] と音声認識データ[15)] の 2 種類で，前者は 4 つの属性を持つ 150 個のサンプルデータで 3 つのカテゴリに分類され，後者は 20 の属性を持つ 1,962 個のサンプルデータで 20 のカテゴリに分類される．ニューロン数を変えて $CE(X)$ を求めた結果を，それぞれ**表 1** と**表 2** に示す．この表から，通常の学習方法では，ニューロン数を増加させても必ずしも誤り係数の値が改善されないことが分かる．

4. LVQ の耐故障学習

ニューロンの故障は，他のどのコードブックベクトルよりもその故障ニューロンのコードブックベクトルに近い入力ベクトルが与えられたときに活性化され，次点であったニューロ

ンの所属するクラスが故障したニューロンと異なるときのみ誤りとして観測される．したがって，クラスの境界に近いニューロンの故障は，同一クラスのニューロンに囲まれたクラスの中央にあるニューロンよりも深刻であると考えられる．別の観点で見れば，同一のクラスに属するニューロンの密度が高い場合には，次点ニューロンとして同一クラスが選ばれる可能性が高くなると考えられることから，同一クラスに属するニューロンの密度が高い部分ほど故障の影響が少ないと考えられる．極端なケースとしてニューロンを 2 重化して完全に同じコードブックベクトルを持つニューロンが存在するとすれば，すべての単一故障はマスクされる．

4.1 境界強調

前述のことから，耐故障の観点からは，クラスの境界におけるニューロンのコードブックベクトルの密度が高いことが望ましい．しかし，LVQ は入力ベクトルの確率密度分布を近似するようにコードブックベクトルを学習することから，対象とする問題において，クラス領域の中央ほど同一クラスの入力の確率密度が高く周辺に向って裾野が広がっている場合には，コードブックベクトルの密度は耐故障のために期待するのと逆の傾向を持つと考えられる．

本来の LVQ の目的はクラス領域の定義であり，その意味ではクラス境界に隣接するコードブックベクトルのみが重要である．入力ベクトルの確率密度分布を近似することは本来の目的からは必ずしも必要ではない．そこで，本研究では，LVQ の耐故障性能を向上させる学習方法として「境界強調」を提案する．境界強調では，学習時の入力ベクトルの分布をサンプルデータの分布から変更し，境界に近いサンプルの出現確率を上昇させることで，学習結果としての境界付近のコードブックの密度を高くする．

境界強調のためには，サンプルデータが境界に近いかどうかを判断しなければならない．しかし，どのサンプルデータが境界に近いのかはあらかじめ与えられてはいない．そこで，学習時において，サンプルデータに対する勝利者ニューロンと次点ニューロンのクラスが異なった場合に，そのサンプルデータは境界に近いと判断して学習データとして与える回数を増加させることにする．

具体的には次の手順に従って学習を行う．

(1)　すべてのサンプルデータを 1 回ずつ学習させ，その際に勝利者ニューロンと次点ニューロンが異なったデータを保存する．
(2)　保存したデータについて追加の学習を行う．
(3)　上記の繰返し．

4.2 カップリング支援

本研究の故障モデルの下では，すべての入力について勝利者ニューロンと次点ニューロンのクラスが一致すれば単一故障はマスクされる．この条件を満足するにはすべてのニューロンを 2 重化すればよい．もし，必要な数の 2 倍のニューロンがあるとすれば，すべてのニューロンを 2 重化は次の学習方法によって実現できる．

(1)　ニューロンを 2 つずつペアと考え同一の値で初期化する．
(2)　ニューロンの更新を 2 つずつ組で行う．すなわち，入力ベクトルに最も近いニューロン，および次点のニューロンはそれぞれ 2 つずつ存在するはずなので，それら 2 組をそれぞれ更新する．

ニューロンの 2 重化により単一故障はマスクされるが，必要なハードウェア量も競合制御部の構成しだいでは，2 倍以上になる．一般に，クラス境界から離れたコードブックベクトルを持つニューロンのように，故障に対して敏感でないニューロンが存在することから，必ずしもすべてのニューロンを 2 重化する必要はなく，故障に敏感なクラス境界付近のニューロンが 2 重化されることが望ましい．

残念ながら先に述べたようにどのデータが境界に近いのかをあらかじめ知ることはできない．そこで，境界に近いデータほど 2 重化に近づくような学習方法を考えることにする．

式 (7) に示した LVQ3 アルゴリズムでは，勝利者と次点のニューロンが同一クラスに所属する場合，両方とも入力ベクトルに近づけていた．この学習規則は，コードブックベクトルを集める効果があると考えられる．この効果に注目し，完全な 2 重化を行う代わりに，耐故障化が必要なクラス境界付近では，LVQ3 アルゴリズムの ϵ を大きくすることで，「カップリングを支援」することにする．

どのサンプルデータについてカップリングを支援するかどうかは，再びサンプルデータに対する競合の結果を利用し，勝利者，次点が同一クラスで，第 3 位が別クラスに属する場合に，ϵ を大きくすることにより，勝利者と次点を入力ベクトルに近づけることにする．

5. 実験による評価

提案した 2 つのアイデアを LVQ3 アルゴリズムの改良としてインプリメントした．それぞれ次のように呼ぶことにする．

FTLVQ3.1　LVQ3 + 境界強調
FTLVQ3.2　LVQ3 + 2 重化支援
FTLVQ3.3　LVQ3 + 境界強調 + 2 重化支援

表 3　アイリスデータに対する誤り係数および分類精度
Table 3　Coefficient of error and accuracy (%) of the classification for the iris plants database.

# of Neurons		8	9	10	11	12	13
OLVQ1	CE(X) mean(s.d.)	25.5 (1.78)	19.9 (1.79)	19.5 (1.16)	20.2 (1.88)	14.2 (1.49)	17.4 (1.72)
	accuracy mean(s.d.)	96.73 (0.36)	96.73 (0.38)	97.13 (0.35)	96.80 (0.43)	97.47 (0.25)	97.33 (0.23)
LVQ3	CE(X) mean(s.d.)	13.2 (0.83)	12.3 (0.55)	9.3 (1.07)	8.5 (1.12)	3.4 (0.53)	4.8 (0.80)
	accuracy mean(s.d.)	97.67 (0.11)	97.73 (0.14)	97.47 (0.13)	97.80 (0.16)	97.60 (0.10)	97.80 (0.17)
FTLVQ3.1	CE(X) mean(s.d.)	9.2 (0.61)	9.3 (0.40)	8.3 (0.58)	6.9 (0.73)	4.6 (0.38)	6.7 (0.93)
	accuracy mean(s.d.)	97.53 (0.19)	98.00 (0.00)	98.00 (0.00)	98.00 (0.19)	98.07 (0.15)	98.20 (0.10)
FTLVQ3.2	CE(X) mean(s.d.)	1.6 (0.32)	2.6 (0.45)	1.9 (0.50)	0.9 (0.30)	0.8 (0.19)	0.6 (0.25)
	accuracy mean(s.d.)	94.60 (0.93)	97.67 (0.11)	97.60 (0.34)	97.07 (0.30)	97.93 (0.06)	97.53 (0.10)
FTLVQ3.3	CE(X) mean(s.d.)	4.1 (0.52)	3.3 (0.60)	3.5 (0.75)	2.9 (1.17)	1.0 (0.64)	1.1 (0.50)
	accuracy mean(s.d.)	97.20 (0.53)	97.93 (0.06)	98.00 (0.94)	97.47 (0.32)	97.87 (0.25)	98.00 (0.13)

表 4　音声認識データに対する誤り係数および分類精度
Table 4　Coefficient of error and accuracy (%) of the classification for the speech signal database.

# of Neurons		150	200	250	300	350	400
OLVQ1	CE(X) mean(s.d.)	201.1 (4.92)	183.9 (2.34)	181.0 (2.85)	175.9 (4.80)	168.6 (2.12)	162.3 (3.44)
	accuracy mean(s.d.)	92.88 (0.10)	93.73 (0.14)	94.32 (0.08)	94.81 (0.10)	95.10 (0.03)	95.10 (0.05)
LVQ3	CE(X) mean(s.d.)	88.0 (4.28)	61.5 (2.41)	47.9 (2.71)	42.4 (2.40)	32.7 (0.65)	31.0 (1.85)
	accuracy mean(s.d.)	94.63 (0.12)	94.99 (0.11)	95.14 (0.10)	95.30 (0.10)	95.30 (0.09)	95.39 (0.08)
FTLVQ3.1	CE(X) mean(s.d.)	107.9 (5.19)	61.0 (3.53)	40.4 (4.04)	26.3 (2.92)	16.8 (1.46)	11.3 (0.92)
	accuracy mean(s.d.)	95.85 (0.06)	96.12 (0.06)	96.12 (0.12)	96.06 (0.10)	96.04 (0.11)	96.02 (0.08)
FTLVQ3.2	CE(X) mean(s.d.)	61.6 (2.02)	49.2 (2.93)	44.9 (1.66)	46.1 (2.61)	39.5 (2.66)	36.9 (1.93)
	accuracy mean(s.d.)	93.92 (0.14)	94.62 (0.08)	95.06 (0.05)	95.51 (0.08)	95.57 (0.09)	95.78 (0.08)
FTLVQ3.3	CE(X) mean(s.d.)	63.2 (4.87)	45.2 (5.09)	30.9 (2.72)	26.1 (2.68)	17.1 (2.63)	13.9 (1.07)
	accuracy mean(s.d.)	95.47 (0.15)	96.02 (0.16)	96.22 (0.14)	96.57 (0.10)	96.41 (0.11)	96.42 (0.05)

なお，FTLVQ3.3 アルゴリズムの詳細を付録に付す．

これらのアルゴリズムの耐故障性を，3 章で用いたものと同じデータを使って評価した．実験にあたって各クラスに属するニューロンの数がほぼ等しくなるようにニューロンを割り当て，各ニューロンが属するクラスのサンプルデータから選んだ値によってニューロンのコードブックベクトルを初期化した．

学習パラメータとして学習率の初期値は $\alpha = 0.05$ とし，$\epsilon = 0.1$ とした．また，FTLVQ3.2 と FTLVQ3.3 において，2 重化支援の条件が満された場合は，ϵ を 1.0 に増加させてコードブックベクトルの更新を行った．

各サンプルデータセットについて学習後のニューラルネットワークの誤り係数 $CE(X)$ を求めた結果を**表 3** および**表 4** に，ニューロン数の変化に対する誤り係数の変化をプロットしたものを**図 4**，**図 5** に示す．これらの結果から，比較的ニューロン数が少い場合には 2 重化支援（FTLVQ3.2）が良い性能を示すことが分かる．一方，境界強調（FTLVQ3.1）は問題の規模が小さいアイリスデータに対しては効果が出ていないが，音声認識データにおいてニューロン数が比較的多い場合に，比較した方法のなかで最善の性能を示している．また，2 つの耐故障性能向上手法の組合せ（FTLVQ3.3）は相補的に働き，ニューロン数の少い場合，多い場合の両方で性能の改善が得られている．

表 3 および表 4 には，無故障時のパターン分類の精度（accuracy）もあわせて示した．これらの結果から，耐故障学習によってパターン分類の精度が低下していないことが分かる．

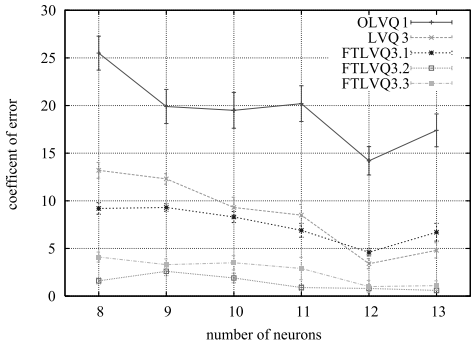

図 4　アイリスデータに対する誤り係数
Fig. 4　Coefficient of error for the iris plants database.

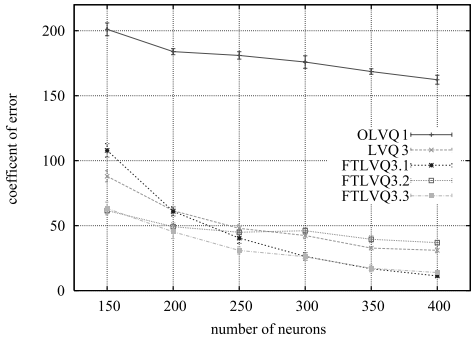

図 5　音声認識データに対する誤り係数
Fig. 5　Coefficient of error for the speech signal database.

6. おわりに

本論文では，LVQ ニューラルネットワークをハードウェア化する際の耐故障性の指標として，学習サンプル集合に基づく誤り発生の期待値である誤り係数 $CE(X)$ を提案した．また，LVQ ネットワークの耐故障性を向上させるための学習方法として「境界強調」と「2 重化支援」のアイデアを示し，それらをインプリメントした学習アルゴリズムの改良について実験による評価を行った．実験結果から提案したアイデアは LVQ ニューラルネットワークの耐故障性の向上に効果があることが確認された．

本論文では，LVQ ネットワークのハードウェア構成として，SIMD 型の並列処理モデルを対象とし，多数の処理ユニットとして冗長性を持つニューロン以外の部分を冗長化して耐故障化しても救済できないニューロンの故障に限定して議論を行った．具体的な冗長アーキテクチャの設計に基づく全体的な耐故障化の検討は今後の課題である．

提案した学習アルゴリズムは，学習サンプルの出現確率を，勝者と 2 位，3 位のニューロンの所属するクラスの違いによって調整し，2 位となったニューロンにも学習を行う．ハードウェア上でオンラインの学習を行うには，2 位，3 位のニューロンのクラスを判別する必要があるが，最小値判定回路では，勝者が故意に大きい値を距離として出力することで，勝負に参加しないことができることから，複数回の判定を行い，勝者が抜けていくことで 2 位，3 位の判定も可能である．具体的なアルゴリズムについては，全体的なハードウェアの設計とともに今後の課題とする．

参 考 文 献

1) Chowdhury, F.N., Wahi, P., Raina, R. and Kaminedi, S.: A Survey of Neural Networks Applications in Automatic Control, *33rd Southeastern Symposium on System Theory*, pp.349–353 (2001).
2) Yasdi, R.: A Literature Survey on Applications of Neural Networks for Human-Computer Interaction, *Neural Computing & Applications*, Vol.9, No.4, pp.245–248 (2000).
3) Danese, G., Leporati, F. and Ramat, S.: A Parallel Neural Processor for Real-Time Application, *IEEE Micro*, Vol.22, No.3, pp.20–31 (2002).
4) Yasunaga, M., Masuda, N., Yagyu, M., Asai, M., Shibata, K., Ooyama, M., Yamada, M., Sakaguchi, T. and Hashimoto, M.: A Self-Learning Digital Neural Network Using Wafer-Scale LSI, *IEEE Trans. Solid-State Circuits*, Vol.28, No.2, pp.106–113 (1993).
5) Morie, T., Fujita, O. and Amemiya, Y.: Analog VLSI Implementation of Adaptive Algorithms by an Extended Hebbian Synapse Circuit, *IEICE Trans. Electronics*, Vol.E75-C, No.3, pp.303–311 (1992).
6) Martin, D.E. and Damper, R.I.: Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application, *IEEE Trans. Neural Networks*, Vol.4, No.5, pp.788–793 (1993).
7) Tan, Y. and Nanya, T.: Fault-Tolerant Back-Propagation Model and Its Generalization Ability, *Digest IJCNN*, pp.1373–1378 (1991).
8) Phatak, D.S.: Fault Tolerant Artificial Neural Networks, *5th Dual Use Technolo-*

gies and Applications Conference, pp.193–198 (1995).
9) Hammadi, N.C., Ohmameuda, T., Kaneko, K. and Ito, H.: Fault Tolerant Constructive Algorithm for Feedforward Neural Networks, *PRFTS'97*, pp.215–220 (1997).
10) Duda, R.O., Hart, P.E. and Stork, D.G.: *Pattern Classification*, 2nd edition, John Wiley & Sons, Inc. (2001).
11) Yasunaga, M., Hachiya, I., Moki, K. and Kim, J.H.: Fault-Tolerant Self-Organizing Map Implemented by Wafer-Scale Integration, *IEEE Trans. Very Large Scale Integration*, Vol.6, No.2, pp.257–265 (1998).
12) Sato, Y., Shibata, K., Asai, M., Ohki, M., Sugie, M., Sakaguchi, T., Hashimoto, M. and Kuwabara, Y.: Development of a high-performance general purpose neurocomputer coposed of 512 digital neurons, *Internationa Joint Conference on Neural Networks*, pp.1967–1970 (1993).
13) 高柳　博, 落合辰男, 宮下　浩, 茂木啓次, 岡橋卓夫, 桑原良博, 佐藤裕二：高速ニューロコンピュータシステム，日立マイコン技報，Vol.8, No.1, pp.2–9 (1994).
14) Kohonen, T.: *Self-Organizing Maps*, Springer-Verlag (1995).
15) LVQ Programming Team of the Helsinki University of Technology: LVQ_PAK: The Learning Vector Quantization Program Package (1995). ftp://cochlea.hut.fi/pub/lvq_pak/
16) Fisher, R.A.: The use of multiple measurements in taxonomic problems, *Annual Eugenics*, Vol.Part II, No.7, pp.179–188 (1936).

付録 FTLVQ3.3 アルゴリズム

$mode \Leftarrow$ "*normal*" ; $j \Leftarrow 1$; $k \Leftarrow 0$; $l \Leftarrow 1$
while ($l \leq l_{runlength}$)
loop: /* The selection of the input vector */
　if (mode = "normal")
　　if ($j <= N_{sampledata}$)
　　$\mathbf{x} \Leftarrow \mathbf{x}_j$
　　else
　　$mode \Leftarrow$ "*additional*"
　　goto loop
　else
　　if ($k >= 1$)
　　$\mathbf{x} \Leftarrow \mathbf{y}_k$
　　else
　　$mode \Leftarrow$ "*normal*" ; $j \Leftarrow 1$
　　goto loop
/* The calculation of the distance between
codebook vectors and the input vector */
for ($i \Leftarrow 1$; $j \leq N_{neuron}$; $i \Leftarrow i + 1$)
　$d_i \Leftarrow ||\mathbf{m}_i - \mathbf{x}||$
/* The search for codebook vectors which
are the closest, the 2nd, and the 3rd place. */
if ($d_1 < d_2$)
　if ($d_2 < d_3$)
　　$c \Leftarrow 1$; $r \Leftarrow 2$; $t \Leftarrow 3$
　else if ($d_1 < d_3$)
　　$c \Leftarrow 1$; $r \Leftarrow 3$; $t \Leftarrow 2$
　else
　　$c \Leftarrow 3$; $r \Leftarrow 1$; $t \Leftarrow 2$
else if ($d_1 < d_3$)
　$c \Leftarrow 2$; $r \Leftarrow 1$; $t \Leftarrow 3$
else if ($d_2 < d_3$)
　$c \Leftarrow 2$; $r \Leftarrow 3$; $t \Leftarrow 1$
else
　$c \Leftarrow 3$; $r \Leftarrow 2$; $t \Leftarrow 1$
for ($i \Leftarrow 4$; $i \leq N_{neuron}$; $i \Leftarrow i + 1$)
　if ($d_i < d_c$)
　　$t \Leftarrow r$; $r \Leftarrow c$; $c \Leftarrow i$
　else if ($d_i < d_r$)
　　$t \Leftarrow r$; $r \Leftarrow i$
　else if ($d_i < d_t$)
　　$t \Leftarrow i$
/* The update of the codebook vectors */

if (class($\mathbf{m}_c$) $\neq$ class($\mathbf{m}_r$))
　if (mode = “normal”)
　　$\mathbf{y}_k \Leftarrow \mathbf{x}$; $k \Leftarrow k+1$
　if (class($\mathbf{m}_c$) = class($\mathbf{x}$))
　　$\mathbf{m}_c \Leftarrow \mathbf{m}_c + \alpha(l)(\mathbf{x} - \mathbf{m}_c)$
　　$\mathbf{m}_r \Leftarrow \mathbf{m}_r - \alpha(l)(\mathbf{x} - \mathbf{m}_r)$
　if (class($\mathbf{m}_r$) = class($\mathbf{x}$))
　　$\mathbf{m}_c \Leftarrow \mathbf{m}_c + \alpha(l)(\mathbf{x} - \mathbf{m}_c)$
　　$\mathbf{m}_r \Leftarrow \mathbf{m}_r - \alpha(l)(\mathbf{x} - \mathbf{m}_r)$
else if (class($\mathbf{m}_c$) = class($\mathbf{x}$))
　if (class($\mathbf{m}_t$) $\neq$ class($\mathbf{x}$))
　　$\mathbf{m}_c \Leftarrow \mathbf{m}_c + \alpha(l)(\mathbf{x} - \mathbf{m}_c)$
　　$\mathbf{m}_r \Leftarrow \mathbf{m}_r + \alpha(l)(\mathbf{x} - \mathbf{m}_r)$
　else
　　$\mathbf{m}_c \Leftarrow \mathbf{m}_c + \epsilon \cdot \alpha(l)(\mathbf{x} - \mathbf{m}_c)$
　　$\mathbf{m}_r \Leftarrow \mathbf{m}_r + \epsilon \cdot \alpha(l)(\mathbf{x} - \mathbf{m}_r)$
$l \Leftarrow l+1$
if (mode = “normal”)
　$j \Leftarrow j+1$
else
　$k \Leftarrow k-1$

(平成 19 年 10 月 8 日受付)
(平成 20 年 3 月 4 日採録)

蓑原　隆（正会員）

1991 年東京工業大学大学院理工学研究科情報工学専攻博士課程修了．工学博士．同年同大学工学部情報工学科助手．1994 年拓殖大学工学部情報工学科専任講師，1998 年同学科助教授，2007 年同学科教授．ディペンダブルコンピューティングの研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE，ACM 各会員．