[計測自動制御学会論文集
Vol.42, No.8, 916/925 (2006)]

バッチ学習型競合連想ネットとその性質

黒木秀一*・西田健*・渕川康裕*

Batch Learning Competitive Associative Net and Its Properties

Shuichi KUROGI*, Takeshi NISHIDA* and Yasuhiro FUCHIKAWA*

So far, the competitive associative net called CAN2 has been developed to utilize the competitive and associative schemes for learning to achieve efficient piecewise linear approximation of nonlinear functions. Although the convensional online learning methods for the CAN2 have been shown effective, they basically are for infinite number of training data. Provided that only a finite number of training data are given, however, the batch learning scheme seems more suitable. We here present a batch learning method to learn a finite number of training data efficiently by means of combining competitive learning, associative learning and reinitialization using asymptotic optimality. Finally, we apply the present method to learning to approximate sevaral artificial benchmark functions and show that the batch CAN2 calculates faster and achieves smaller MSE (mean square error) than the conventional online CAN2, and it has several advantages superior to the SVR (support vector regression).

Key Words: competitie associative net, batch learning, piecewise linear approximation, comparative experiment to support vector regression

1. はじめに

競合連想ネット CAN2 (Competitive Associative Net 2) は競合ネット[1]と連想ネット[2]の機能を用いて非線形関数を学習し区分的線形関数として近似するニューラルネットであり[3]，非線形時変プラントの制御[4]，降水量推定[5]，非線形関数の学習問題[6]などへ応用され種々の側面からその有用性が示されている．特に2000年度電子情報通信学会総合大会シンポジウム・降水量推定コンテストでは CAN2 を用いた手法[5]が第2位の成績を得ており，その性能の高さが示された結果であると考える．このネットの特徴は訓練誤差 (訓練データに対する近似の二乗平均誤差) を最小化するために，勾配法に基づく競合学習により入力空間を区分し，再帰的最小二乗法に基づく連想学習により各領域での線形近似を最適化し，さらに漸近最適性の条件[6]を用いて勾配法の局所解問題に対処していることにある．この類似手法として，局所線形モデル[7]や区分的線形ニューラルネット[8]があるが，それらは訓練入力ベクトル集合と各区分領域の中心ベクトルとの距離測度を最小化するための手法である K 近傍法 (K-nearest neighbor) を用いて入力空間を区分するのに対し，CAN2 は訓練誤差自体を最小化するための競合学習により入力空間を区分する点に違いがある．また CAN2 はその連想部分をエキスパートとし，競合部分をゲートとする混合エキスパートモデル[9]としても捉えられる．この類似手法としてたとえば MARS (multivariate adaptive regression splines) モデル[10]は連続的な区分的線形近似を行なうのに対し，競合連想ネットは各区分領域で最適な線形近似を行なうために不連続な近似を行なう点などが異なる．

さて従来の CAN2 は訓練データを1個ずつ処理してネットのパラメタを少しずつ更新していくオンライン学習法を用いていた．しかし訓練データが有限個の場合には訓練データ全体を用いてネットのパラメタを一括して変更していくバッチ学習の枠組のほうがより効率的な学習法が構成できると考えられる．そこで著者らは CAN2 のバッチ学習法を考案し，いくつかの問題に適用した[11)~15)]．特に IJCNN2004 (International Joint Conference on Neural Networks) の時系列予測コンペティションでは第3位[14]，NIPS2004 (Neural Information Processing) の Evaluating Predictive Uncertainty Challenge の回帰部門では第1位 (regression winner)[15]に選ばれ，バッチ学習型 CAN2 の性能の高さが示された結果であると考えられる．

本稿ではこのバッチ学習型 CAN2 を提案し(注1)，その基本的な性質を示すことを目的とする．特にオンライン型との違い，および他手法のうちでも関数近似性能の高かった SVR (Support Vector Regression) との比較実験[6]を通して提案

* 九州工業大学工学部　北九州市戸畑区
* Faculty of Engineering, Kyushu Institute of Technology
(Received April 13, 2005)
(Revised March 13, 2006)

(注1) 本稿の一部は SCI2004 などで発表しているが，より詳細に論文誌に掲載するのは本稿が初めてである．

手法の性質を示す．以下，まず2でCAN2とそのバッチ学習法を示す．この学習法は従来のオンライン学習法と同様の手順を繰り返す手法であるが，各手順において有限個のデータから得られる情報を用いてより効果的に学習するように構成している．つぎに3でいくつかの非線形関数に対し，従来のオンライン学習法およびSVRとの比較数値実験を行ない，バッチ学習型CAN2の諸性質を示す．

2. バッチ学習型CAN2

2.1 CAN2による関数近似

まず k 次元ベクトル $\boldsymbol{x}_j \triangleq (x_{j1}, x_{j2}, \cdots, x_{jk})^T \in \mathbb{R}^{k\times 1}$ とスカラ値 $y_j \in \mathbb{R}$ を入出力とするシステム

$$y_j \triangleq f(\boldsymbol{x}_j) + d_j \tag{1}$$

を考える．ここで $j = 1, 2, \cdots$ は異なるデータを表わすための添字であり，$f(\boldsymbol{x}_j)$ は $\boldsymbol{x}_j$ の関数，d_j は平均値が0で分散が σ_d^2 の観測雑音とする．CAN2は荷重ベクトル $\boldsymbol{w}_i \triangleq (w_{i1}, \cdots, w_{ik})^T \in \mathbb{R}^{k\times 1}$ と連想行列 $\boldsymbol{M}_i \triangleq (M_{i0}, M_{i1}, \cdots, M_{ik}) \in \mathbb{R}^{1\times(k+1)}$ をもつユニットを N 個 $(i \in I = \{1, 2, \cdots, N\})$ 用いて

$$\widehat{y} \triangleq \widehat{y}_c \triangleq \boldsymbol{M}_c \widetilde{\boldsymbol{x}} \tag{2}$$

により，上記システムの観測雑音を除いた関数 $y = f(\boldsymbol{x})$ の近似を行なう．ここで $\boldsymbol{x} = (x_1, x_2, \cdots, x_k)^T \in \mathbb{R}^{k\times 1}$ は任意の入力ベクトル，$\widetilde{\boldsymbol{x}} \triangleq (1, \boldsymbol{x}^T)^T \in \mathbb{R}^{(k+1)\times 1}$ は上式(2)の線形近似のバイアス項を生成するために1を付加したベクトル，および

$$c \triangleq \underset{i\in I}{\operatorname{argmin}} \|\boldsymbol{x} - \boldsymbol{w}_i\| \tag{3}$$

は競合により選択されるユニットの番号を表わす．以上の関数近似は入力空間 $V = \mathbb{R}^k$ を N 個のボロノイ領域(またはディリクレ領域ともいう)

$$V_i \triangleq \{\boldsymbol{x} \mid i = \underset{l\in I}{\operatorname{argmin}} \|\boldsymbol{x} - \boldsymbol{w}_l\|\} \tag{4}$$

に分割して，関数 $y = f(\boldsymbol{x})$ の区分的線形近似を行なうことを意味する．

2.2 CAN2のバッチ学習法

2.2.1 CAN2の学習の目的と漸近最適条件

システム方程式(1)より得られる有限個の訓練データの集合 $D = \{(\boldsymbol{x}_j, y_j) \mid j \in J\}$ が与えられているとする．ここで $X = \{\boldsymbol{x}_j \mid j \in J\}$ は訓練入力(ベクトル)の集合，$Y = \{y_j = f(\boldsymbol{x}_j) + d_j \mid j \in J\}$ は訓練出力(スカラー)の集合，および $J = \{1, 2, \cdots, n\}$ は各訓練データの添字集合であり，訓練データの個数 n は有限である．CAN2の学習の目的は，この訓練データ集合 D を学習した後，システム方程式(1)への任意の入力 $\boldsymbol{x}$ に対して雑音のないシステム出力 $y = f(\boldsymbol{x})$ の推定値 $\widehat{y} \simeq f(\boldsymbol{x})$ を出力できるようになることである．そこでそのような学習を達成するには訓練データに対する近似誤差 $e(\boldsymbol{x}_j) \triangleq \widehat{y} - y_j$ の二乗の平均(期待値)を表わすエネルギー

$$E \triangleq \frac{1}{n}\sum_{i\in I}\sum_{\boldsymbol{x}\in X_i} \|e(\boldsymbol{x})\|^2 = \sum_{i\in I} E_i \tag{5}$$

を最小化する $\boldsymbol{w}_i$ と $\boldsymbol{M}_i$ $(i \in I)$ を求めればよいと考えられる．ここで $X_i = \{\boldsymbol{x} \in X \cap V_i\}$ はボロノイ領域 V_i に属する訓練入力ベクトルの集合であり，$E_i \triangleq (1/n)\sum_{\boldsymbol{x}\in X_i} \|e(\boldsymbol{x})\|^2$ である．この E の最小化問題は非線形問題であり，本手法ではつぎの3つの手順を繰り返す解法を適用する．すなわち(1)荷重ベクトルの更新，(2)連想行列の更新，および(3)再初期化，を訓練データ集合 D に対してそれぞれ1回ずつ行なうことを1回のバッチ学習とし，この学習を繰り返すことにより E を逐次的に最小化する手法を用いる．以下，各手順を説明する．

2.2.2 勾配法に基づく荷重ベクトルの更新

まず連想行列 $\boldsymbol{M}_i$ $(i \in I)$ の値が変化しないと仮定すると，以下のように勾配法の考え方により $\boldsymbol{w}_i$ を最適化することができる；今，ある荷重ベクトル $\boldsymbol{w}_i$ を含むボロノイ領域 V_i とその隣接領域 V_l との微小な幅 θ_W $(< 1$; 後述の実験では $\theta_W = 0.1$ とした$)$ をもつ境界領域

$$W_{il} = \{\boldsymbol{x} \mid \boldsymbol{x} \in X_i \cup X_l \quad \text{および} \quad \frac{\left|(2\boldsymbol{x} - \boldsymbol{w}_i - \boldsymbol{w}_l)^T(\boldsymbol{w}_i - \boldsymbol{w}_l)\right|}{\|\boldsymbol{w}_i - \boldsymbol{w}_l\|^2} \le \theta_W\} \tag{6}$$

にある訓練ベクトル $\boldsymbol{x} \in W_{il}$ に対して，$\boldsymbol{w}_i$ が微小量 $\Delta\boldsymbol{w}_i$ だけ変化することにより $\boldsymbol{x}$ が属する領域が V_i から V_l または V_l から V_i に移動するとき，エネルギー E は $s \triangleq \Delta\boldsymbol{w}_i^T(\boldsymbol{x} - \boldsymbol{w}_i)$ の符号で $(e_i^2(\boldsymbol{x}) - e_l^2(\boldsymbol{x}))/n$ だけ増加する．すなわち $\boldsymbol{x} \in V_l \cap W_{il}$ かつ $s > 0$ のとき(**Fig. 1** 参照)，$f(\boldsymbol{x})$ によるネットの近似誤差は第 l ユニットによる誤差 $e_l(\boldsymbol{x})$ $\left(\triangleq \boldsymbol{M}_l\widetilde{\boldsymbol{x}} - f(\boldsymbol{x})\right)$ から第 i ユニットによる誤差

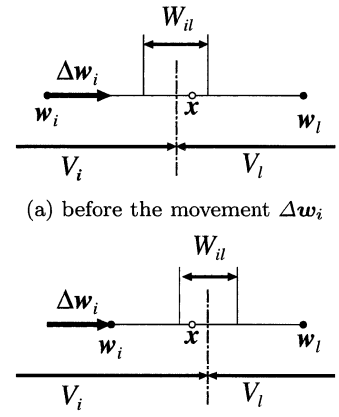


Fig. 1 Example of the effect of the movement $\Delta\boldsymbol{w}_i$. A training input vector $\boldsymbol{x}$ in (a) $V_l \cap W_{il}$ moves into (b) $V_i \cap W_{il}$ by the movement $\Delta\boldsymbol{w}_i$ when $s = \Delta\boldsymbol{w}_i^T(\boldsymbol{x} - \boldsymbol{w}_i) > 0$. Here, we suppose the input space is one-dimensional for simplicity.

$e_i(\boldsymbol{x})$ に変化し，逆に $\boldsymbol{x} \in V_i \cap W_{il}$ かつ $s < 0$ のときの誤差は $e_i(\boldsymbol{x})$ から $e_l(\boldsymbol{x})$ に変化する．一方，$\boldsymbol{x} \in V_l \cap W_{il}$ かつ $s < 0$，または $\boldsymbol{x} \in V_i \cap W_{il}$ かつ $s > 0$，のときは $\boldsymbol{x}$ が属する領域は変化せず，したがってエネルギー E も変化しない．これは $\Delta \boldsymbol{w}_i$ と E が不連続な関係をもつことを意味するが，今，W_{il} 内に訓練ベクトル $\boldsymbol{x}$ が一様に存在すると仮定すると，(6) 式より $V_i \cap W_{il}$ と $V_l \cap W_{il}$ は同面積なので W_{il} 内の半数の訓練ベクトルによりエネルギー E が変化し，$\Delta \boldsymbol{w}_i$ によるエネルギーの増加量 ΔE は

$$\Delta E \simeq \frac{1}{2n} \Delta \boldsymbol{w}_i^T \boldsymbol{\xi}_i \tag{7}$$

で近似できると考えられる．ここで

$$\boldsymbol{\xi}_i \triangleq \sum_{l \in A_i} \sum_{\boldsymbol{x} \in W_{il}} (e_i^2(\boldsymbol{x}) - e_l^2(\boldsymbol{x})) \frac{\boldsymbol{x} - \boldsymbol{w}_i}{||\boldsymbol{x} - \boldsymbol{w}_i||} \tag{8}$$

であり，A_i は V_i に隣接する領域 V_l の添字 l の集合である．したがって十分小さな学習係数 $\gamma\ (>0)$ を用いて $\Delta \boldsymbol{w}_i = \gamma \boldsymbol{\xi}_i$ とすると，$\Delta E = -(\gamma/2n)\|\boldsymbol{\xi}_i\|^2 < 0$ となり，エネルギー E の減少が期待できる．したがってすべての荷重ベクトル $\boldsymbol{w}_i\ (i \in I)$ について

$$\boldsymbol{w}_i := \boldsymbol{w}_i - \gamma \boldsymbol{\xi}_i \tag{9}$$

とする更新式が得られる．ここで := は右辺で計算した値を左辺に代入することを表わす．さらに学習係数は $\gamma = \gamma_0 D_x / D_\xi$ とし，γ_0 は 1 より十分小さな正定数 (後述の実験では $\gamma_0 = 0.001$ とした)，D_x は訓練入力ベクトル $\boldsymbol{x}_j$ の要素 x_{jl} の取りうる幅

$$D_x = \max_{l=1,\cdots,k} \left(\max_{\substack{j \in J \\ m \in J}} |x_{jl} - x_{ml}| \right) \tag{10}$$

および D_ξ は $\boldsymbol{\xi}_i$ の要素 ξ_{il} の最大値

$$D_\xi = \max_{l=1,\cdots,k} \max_{i \in I} \xi_{il} \tag{11}$$

である．すると (9) 式で最も大きく更新される要素 w_{il} は訓練入力ベクトルの要素が取りうる幅 D_x の γ_0 倍以下になるので，$\boldsymbol{w}_i$ が微小に更新されることを保証する．

以上のようにバッチ学習法ではすべての訓練データの情報として D_x と D_ξ を用いることができるが，訓練データが 1 個ずつ与えられる従来のオンライン学習法での学習係数[6] は $e_i(\boldsymbol{x})$ の取りうる範囲を予想して注意深く設定する必要がある．

2.2.3 線形最小二乗法による連想行列の更新

荷重ベクトル $\boldsymbol{w}_i\ (i \in I)$ が変化しないとすると，$E = \sum_{i \in I} E_i$ の最小化問題は各 i について

$$E_i = \frac{1}{n} \|\boldsymbol{M}_i \widetilde{\boldsymbol{X}}_i - \boldsymbol{Y}_i\|^2 \tag{12}$$

を最小化する線形問題となり，その解は $\boldsymbol{M}_i = \boldsymbol{Y}_i \widetilde{\boldsymbol{X}}_i^+$ となる．ここで $\widetilde{\boldsymbol{X}}_i^+$ は $\widetilde{\boldsymbol{X}}_i$ の一般化逆行列を表わし，$\widetilde{\boldsymbol{X}}_i \in \mathbb{R}^{(k+1) \times n_i}$ はすべての $\boldsymbol{x}_j \in X_i$ から生成した列ベクトル $\widetilde{\boldsymbol{x}}_j = (1, \boldsymbol{x}_j^T)^T$ を順に横に並べた行列，$\boldsymbol{Y}_i \in \mathbb{R}^{1 \times n_i}$ は $\widetilde{\boldsymbol{x}}_j$ に対応する $y_j \in Y_i$ を横に並べた行列，および n_i は X_i の要素数である．ここで n_i がベクトルの次元 k よりも小さいと，上述した解 $\boldsymbol{M}_i = \boldsymbol{Y}_i \widetilde{\boldsymbol{X}}_i^+$ は E_i を最小にする解のうち $\|\boldsymbol{M}_i\|$ を最小にするものになるが，この解は未知ベクトルに対する関数近似を行なうために一般には妥当であるとは限らない．この解決策として，まず $n_i = 0$ のユニットについては $\boldsymbol{w}_i$ と $\boldsymbol{M}_i$ の更新およびこのユニットによる関数近似は再初期化 (後述) されるまで行なわないこととする．さらにある定数 $n_{\min}$ に対して $1 \le n_i < n_{\min}$ の場合には $|X_i| = n_{\min}$ となるまで $\boldsymbol{w}_i$ の近傍の訓練データを補充する，すなわち，

$$X_i := \{\boldsymbol{x}_j \mid \|\boldsymbol{x}_j - \boldsymbol{w}_i\| < \min_{\boldsymbol{x}_l \in X - X_i} \|\boldsymbol{x}_l - \boldsymbol{w}_i\|, \ |X_i| = n_{\min}\} \tag{13}$$

となるように X_i を補充し (**Fig. 2** 参照)，この X_i を用いて $\boldsymbol{M}_i = \boldsymbol{Y}_i \widetilde{\boldsymbol{X}}_i^+$ を更新する．ここで以上の方法は V_i の外から $\boldsymbol{w}_i$ の球形近傍領域の訓練データを補充するので，$n_{\min}$ が大きすぎる場合や V_i の最適形状が球でない場合には不適切なデータを補充する恐れがあり，$n_{\min}$ は注意深く選択すべきである．しかし経験的には k の大きさにかかわらず，$n_{\min} = 3$ または 4 が多くの事例でよい性能を示したことを付記しておく．これは多くの関数近似問題においては比較的小数の訓練データしか与えられないので $n_{\min}$ を大きくしすぎるとあまりよい結果が得られないこと，および 3〜4 個の訓練データがあれば $\boldsymbol{M}_i = \boldsymbol{Y}_i \widetilde{\boldsymbol{X}}_i^+$ が $\|\boldsymbol{M}_i\|$ を最小にする機能により，予測誤差が発散しないような比較的良い解が得られるのではないかと考える．

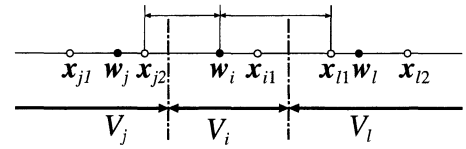


Fig. 2 Example of the augmentation of training vectors. For $n_{\min} = 2$, the original training dataset $X_i = \{x_{i1}\}$ in V_i is insuficient, and the training vector $\boldsymbol{x}_{j2}\ (\notin V_i)$ nearest to $\boldsymbol{w}_i$ is augmented so that $X_i := \{\boldsymbol{x}_{i1}, \boldsymbol{x}_{j2}\}$ and $|X_i| = n_{\min}$.

連想行列 $\boldsymbol{M}_i = \boldsymbol{Y}_i \widetilde{\boldsymbol{X}}_i^+$ は直接計算することもできるが，つぎの RLS (Recursive Least Square) 法により逐次的に求めることもできる．すなわちすべての $\boldsymbol{x} \in X_i$ と対応する $y \in Y_i$ について

$$\boldsymbol{M}_i := \boldsymbol{M}_i + \frac{(y - \boldsymbol{M}_i \widetilde{\boldsymbol{x}}) \widetilde{\boldsymbol{x}}^T \boldsymbol{\Psi}_i}{1 + \widetilde{\boldsymbol{x}}^T \boldsymbol{\Psi}_i \widetilde{\boldsymbol{x}}} \tag{14}$$

および

$$\boldsymbol{\Psi}_i := \boldsymbol{\Psi}_i - \frac{\boldsymbol{\Psi}_i \widetilde{\boldsymbol{x}} \widetilde{\boldsymbol{x}}^T \boldsymbol{\Psi}_i}{1 + \widetilde{\boldsymbol{x}}^T \boldsymbol{\Psi}_i \widetilde{\boldsymbol{x}}} \tag{15}$$

による更新を，すべての $i \in I$ について行なう．各行列の初期値は $\boldsymbol{M}_i = \boldsymbol{O}$ と $\boldsymbol{\Psi}_i = \boldsymbol{I}/\epsilon_0$ とし，$\boldsymbol{O}$ は零行列，$\boldsymbol{I}$ は

単位行列を表わし，ϵ_0 は小さな定数である (後述の実験では $\epsilon_0 = 10^{-4}$ とした). この初期化は，バッチ学習の繰り返しごとに行なうこともできるが，最初のバッチ学習時にのみ一度だけ行なうこともできる．いくつかの実験結果によると，連想行列を直接計算する場合や RLS 法でバッチ学習の繰り返しごとに初期化する場合はバッチ学習の繰り返しの途中で訓練データに対する二乗平均誤差が不安定になる場合があったが，最初のバッチ学習時に一度だけ初期化する RLS 法ではそのような不安定な現象は起こらなかったので，後述の実験では後者の方法を用いた．なお前者の場合に不安定になった要因として，バッチ学習の繰り返しの途中で，ある連想行列 $\boldsymbol{M}_i$ のひとつの要素の値が突然非常に大きくなり，学習の繰り返しとともに次第に発散していく現象を観察することができた．これはバッチ学習の繰り返しにより得られる X_i と Y_i のみを用いて計算した $\boldsymbol{M}_i$ は不安定になることがあり，後者の手法はこれを取り除く働きがあるものと考えられる．

以上の処理のうち (2) 式により訓練データを補充する処理は訓練データすべてを用いるバッチ学習の枠組でのみ行なうことができる処理である．逆にオンライン学習ではこの処理が不可能なため，訓練データが少ない領域での近似性能がバッチ学習より劣る可能性があることがわかる．

2.2.4　漸近最適条件に基づく再初期化

前述の荷重ベクトル $\boldsymbol{w}_i$ の更新で用いた勾配法には局所解問題が内在する．そこで荷重ベクトルが十分多い場合を仮定して得られる最適条件 (漸近最適条件という) を用いて荷重ベクトルを再初期化する学習法とその有効性が示されており [6]，本稿でもこの手法を利用する．以下，この条件を要約して示し，その使用法を説明する．まず漸近最適条件とは，荷重ベクトルが十分多い場合の荷重ベクトルの最適配置においては，各ボロノイ領域における観測雑音を除いた二乗誤差和が等しくなる，という条件であり，以下のように定式化される．まず (5) 式のエネルギーを連続化して

$$E = \sum_{i \in I} \int_{V_i} \|e_i(\boldsymbol{x})\|^2 p(\boldsymbol{x}) d\boldsymbol{x} \tag{16}$$

と表わす．ここで $p(\boldsymbol{x})$ は訓練ベクトル $\boldsymbol{x}$ の確率密度である．今，各 V_i は十分小さく，各 V_i 内で $p(\boldsymbol{x})$ は一定値 p_i で近似でき，$f(\boldsymbol{x})$ は 2 次関数で近似できるとする．するとネットのエネルギーは

$$\begin{aligned} E =& \sum_{i \in I} (C_i p_i v_i^{1+4/k} + \sigma_2^2 p_i v_i) \\ \geq& N^{-4/k} \|C(\boldsymbol{x}) p(\boldsymbol{x})\|_{\frac{1}{1+4/k}} + \sigma_d^2 \end{aligned} \tag{17}$$

となることが導かれる (詳しくは文献 6) 参照)．ここで $v_i = \int_{V_i} d\boldsymbol{x}$ は V_i の体積であり，$C_i \triangleq C(\boldsymbol{w}_i)$ は量子化係数と呼ばれ，関数 $f(\boldsymbol{x})$ の位置 $\boldsymbol{x} = \boldsymbol{w}_i$ における非線形性の強さを表わす．また $\|g(\boldsymbol{x})\|_\alpha = \left(\int_V |g(\boldsymbol{x})|^\alpha d\boldsymbol{x}\right)^{1/\alpha}$ であり，$\|C(\boldsymbol{x})p(\boldsymbol{x})\|_{\frac{1}{1+4/k}}$ は与えられた $f(\boldsymbol{x})$ と $p(\boldsymbol{x})$ に対し定数になる．さらに (17) 式の右辺は E の最小値を表わし，この式の等号が成立するのは

$$\alpha_i \triangleq C_i p_i v_i^{1+4/k} = \text{一定}\ (i = 1, 2, \cdots, N) \tag{18}$$

となるときのみであり，これが漸近最適条件を表わす．なお，以下，α_i を量子化歪あるいは単に歪という．この条件を以下のように利用する．

まず第 i ユニットの二乗誤差和 $S_i = nE_i$ を計算すると，(5) 式，(16) 式，(17) 式および (18) 式より，

$$\begin{aligned} S_i &\triangleq \sum_{\boldsymbol{x} \in X_i} \|e(\boldsymbol{x})\|^2 \\ &\simeq n \int_{V_i} \|e(\boldsymbol{x})\|^2 p(\boldsymbol{x}) d\boldsymbol{x} \\ &= n\alpha_i + \sigma_d^2 n_i \end{aligned} \tag{19}$$

が得られる．ここで上式の近似は (5) 式と (16) 式における離散と連続の違いに基づく近似を表わす．いま荷重ベクトルが多数あり $f(\boldsymbol{x})$ が線形近似できる領域 V_i の存在を仮定すると，その領域での C_i は 0，したがって α_i は 0 となるので観測雑音の分散 σ_d^2 の推定値

$$\widehat{\sigma}_d^2 = \min\{S_i/n_i \mid i \in I \text{ および } n_i \geq n_\theta\} \tag{20}$$

が得られる．ここでしきい値 n_θ $(> k)$ は訓練データの個数 n_i が入力ベクトルの次元 k 以下のとき $\boldsymbol{M}$ の最適化により $S_i = 0$ となり σ_d^2 が正しく推定できなくなることを防ぐための定数である．以上により歪 α_i の推定値

$$\widehat{\alpha}_i := \frac{S_i - \widehat{\sigma}_d^2 n_i}{n} \tag{21}$$

が得られ，これらの $\widehat{\alpha}_i$ $(i \in I)$ が (18) 式を満たしているかどうかを判断するために，つぎの条件式を用いる．

$$\frac{\widehat{\alpha}_b}{\langle \widehat{\alpha}_i \rangle} \geq \theta_\alpha \quad \text{および} \quad \frac{H}{\ln(N)} \leq \theta_H \tag{22}$$

ここで $b \in I$ であり，θ_α (> 1) および θ_H (< 1) は正定数，$\langle \widehat{\alpha}_i \rangle$ は推定歪 $\widehat{\alpha}_i$ $(i \in I)$ の平均，H は次式で与えられるエントロピー

$$H \triangleq -\sum_{i \in I} \frac{\alpha_i}{\sum_{j \in I} \alpha_j} \ln \left(\frac{\alpha_i}{\sum_{j \in I} \alpha_j} \right) \tag{23}$$

である．条件式 (22) の左方の不等式はある推定歪 $\widehat{\alpha}_b$ が平均値よりかなり大きいとき成立し，右方の不等式はすべての $\widehat{\alpha}_i$ $(i \in I)$ があまり均一でないとき成立する．この連立不等式は安定した漸近最適性の判断を行なうためにベクトル量子化における漸近最適性の判断に用いられたものである (詳細は文献 16) を参照)．なお後述の実験では再初期化が適度に起こるしきい値として $\theta_\alpha = 5$ および $\theta_H = 0.75 + 0.15(N - 100)/400$ を用いた．ここでこの θ_H はユニット数が $N = 100$ と 500 のときそれぞれ $\theta_H = 0.75$ と 0.90 とすると良い結果が得られた経験則から導いた．さて条件式 (22) が満たされるとき，その左方の不等式を満たす大きい歪をもつ第 b ユニットの荷重ベクトル $\boldsymbol{w}_b$ の近くに小さな歪をもつユニットの荷重ベク

トルを移動することにより，より最適解に近づけることができると考えられる．すなわち，今，すべての $i \in I$ について j 番目に大きい $\widehat{\alpha}_i$ をもつユニットを第 $b(j)$ ユニットとするとき，j 番目に小さい第 $s(j) = b(N-j)$ ユニットを次式により再初期化する．

$$\boldsymbol{w}_{s(j)} := \boldsymbol{w}_{b(j)} + \theta_r(\boldsymbol{x}_{c(j)} - \boldsymbol{w}_{b(j)}) \tag{24}$$

$$\boldsymbol{M}_{s(j)} := \boldsymbol{M}_{b(j)} \tag{25}$$

ここで $\boldsymbol{x}_{c(j)}$ は $\boldsymbol{w}_{b(j)}$ に最も近い訓練ベクトルである．また新しい $\boldsymbol{w}_{s(j)}$ に対するボロノイ領域 $V_{s(j)}$ には少なくとも 1 つの訓練ベクトル $\boldsymbol{x}_{c(j)}$ が存在するように，再初期化の係数 θ_r は 1.9 とした．なお前節で述べた $n_i = 0$ となるユニットは $\widehat{\alpha}_i$ が最も小さいユニットとして優先的に再初期化する．

以上のようにバッチ学習では訓練データすべてを用いて歪の推定値 $\widehat{\alpha}_i$ を計算できるが，従来のオンライン学習では各領域の近似誤差をある時定数で漏れ積分して求めており，その時定数の決定に注意を要した．またバッチ学習では再初期化された荷重ベクトルのボロノイ領域に少なくとも 1 つの訓練ベクトルが存在するように構成できたが，オンライン学習では再初期化する荷重ベクトルを補充すべき領域の荷重ベクトルと同じベクトルに再初期化し，そのとき入力された訓練ベクトルを学習させていたため，再初期化されたボロノイ領域内に訓練ベクトルが存在せず近似性能が低下する可能性がある．

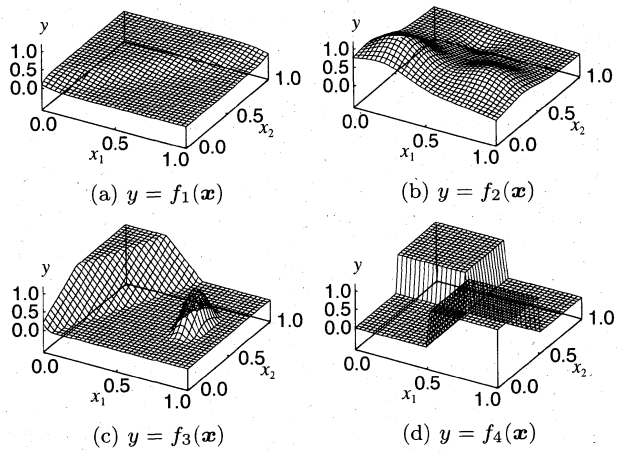
3. 数値実験

前述のように，提案手法はすでに IJCNN2004 や NIPS2004 のコンペティションで使用し，優れた成績を得ている．特に，NIPS2004 では stereopsis (両眼立体視のデータ)，gaze (計算機スクリーン上の注視データ)，outaouais (企業の秘密データ) という性質の異なる 3 つのデータセットに適用した．入力次元がそれぞれ 4，12，37 のベクトルからその関数の予測値 (予測分布の中心) と予測分布を求める課題であり，訓練データ数はそれぞれ 192，150，20,000，およびテストデータ数はそれぞれ 500，427，20,000 のデータセットが使用された．これらさまざまな入力次元やデータ数をもつデータセットに対し，提案手法の MSE (二乗平均予測誤差) はそれぞれ第 1 位，第 2 位，第 1 位の成績を得ることができた．このコンペティションでは最新の種々のニューラルネット手法が適用されており，そのうちでもバッチ学習型 CAN2 が優れていることが示された結果であると考えられる．以下では，視覚的にわかりやすい入力次元が 2 次元のデータセットに対する数値実験結果を示し，提案手法の基本的な性質を検討する．

3.1 準備

提案手法の学習性能をつぎの 4 つの関数 $f_i(\boldsymbol{x}) = f_i(x_1, x_2)$ $(i = 1, 2, 3, 4)$ について検討した (**Fig. 3** および文献 6) 参照)．

$$f_1(\boldsymbol{x}) = \frac{\tanh(9x_2 - 9x_1) + 1}{9}$$



(a) $y = f_1(\boldsymbol{x})$ (b) $y = f_2(\boldsymbol{x})$

(c) $y = f_3(\boldsymbol{x})$ (d) $y = f_4(\boldsymbol{x})$

Fig. 3 Benchmark functions

$$f_2(\boldsymbol{x}) = \frac{3}{4}\exp\left(-\frac{(9x_1-2)^2 + (9x_2-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x_1+1)^2}{49} - \frac{(9x_2+1)^2}{10}\right) + \frac{1}{2}\exp\left(-\frac{(9x_1-7)^2 + (9x_2-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x_1-4)^2 - (9x_2-7)^2\right)$$

$$f_3(\boldsymbol{x}) = \begin{cases} 1 & (x_2 - \xi > 1/2) \\ 2(x_2 - \xi) & (0 \le x_2 - \xi \le 1/2) \\ (\cos(4\pi r) + 1)/2 & (r \le 1/4) \\ 0 & (\text{そのほか}) \end{cases}$$

$$\xi = 2.1x_1 - 0.1;\ r = \sqrt{(\xi - 3/2)^2 + (x_2 - 1/2)^2}$$

$$f_4(\boldsymbol{x}) = \begin{cases} 0 & ((x_1 - 0.5)(x_2 - 0.5) \ge 0) \\ 1 & (\text{そのほか}) \end{cases}$$

まず各関数 f_i $(i = 1, 2, 3, 4)$ に対して入力ベクトル $\boldsymbol{x}$ を単位正方領域 $S = \{\boldsymbol{x} = (x_1, x_2) \mid x_1, x_2 \in [0, 1]\}$ 内からランダムに選んで $f_i(\boldsymbol{x})$ を計算したあと，区間 $[-0.01, 0.01]$ 内の一様乱数による雑音を加えて訓練データ $y = f_i(\boldsymbol{x}) + d$ を生成し，データの個数 $n = j \times 10^3$ $(j = 1, 5, 10, 50)$ のデータ集合 $D(j; n)$ を作成した．つぎに各訓練データ集合をユニット数が $N = 100$，200，300，400 および 500 のネットにそれぞれバッチ学習させた．このときネットを評価するため訓練データに対するネットの出力の二乗平均誤差 E_{train}，およびテストデータとして格子点 $\boldsymbol{x} \in L(K) = \{(0.5 + i/32, 0.5 + j/32)^T \mid i, j = 0, \pm 1, \pm 2, \cdots, \pm(16 + K)\}$ とその関数値 $y = f_i(\boldsymbol{x})$ を用いて，ネットによる推定値 $\widehat{y}$ との二乗平均誤差 $E_{\text{test}}^{(K)}$ を求めた．ここで K は非負の整数であり，$K = 0$ のときの $L(K) = L(0)$ は訓練データを生成した単位正方領域内の点集合となり，$K \ge 1$ のときの $L(K)$ は学習した単位正方領域の外の点を含むこととなる．したがって $E_{\text{test}}^{(K)}$ は $K = 0$ のとき内挿能力，$K \ge 1$ のとき外挿能力，および両者を合わせて汎化能力，さらに E_{train} は学習能力を表わすと解釈できる．なお $L(0)$ は Fig. 3 で関数を描画するために用いた

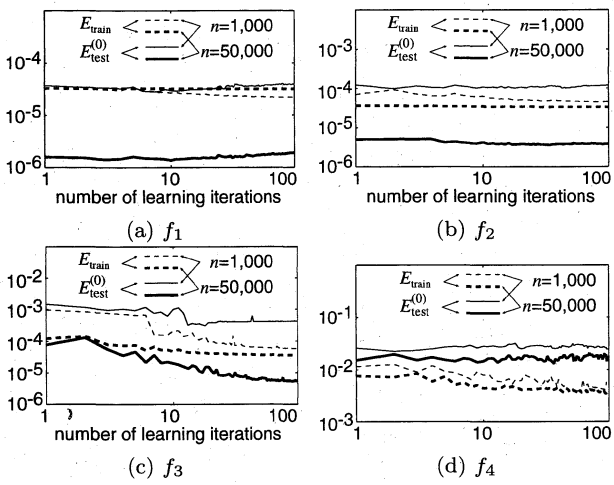


Fig. 4　E_{train} and $E^{(0)}_{\mathrm{test}}$ vs. the number of batch learning iterations

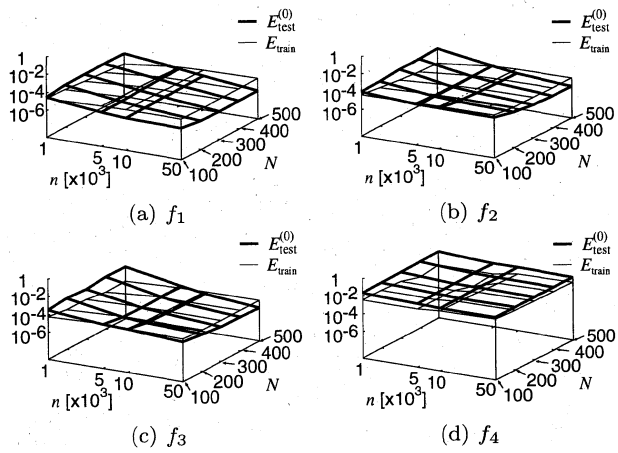


Fig. 5　E_{train} and $E^{(0)}_{\mathrm{test}}$ vs. n and N

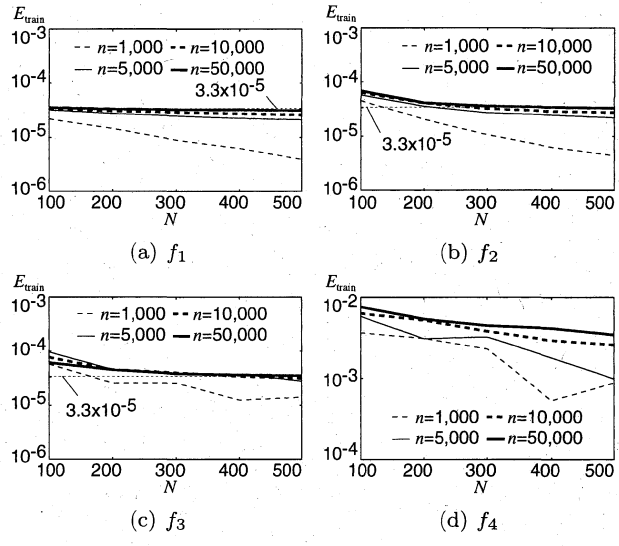


Fig. 6　E_{train} vs. N

33×33 の格子点の集合であり，この集合により生成されたテストデータ集合はネットが各関数を学習できたかどうかを判断するために妥当な集合であると考えられる．

なお以下の計算結果は，Intel(R) Pentium(R) III Mobile CPU 1200MHz の CPU，主メモリ 512M バイト，OS として Vine Linux 2.6，倍精度計算の C プログラムと gcc-2.95 のコンパイラを使用して得たものである．

3.2　バッチ学習回数に対する MSE の変化

まず各関数 f_i について $n = 1{,}000$ と 50,000 の訓練データ集合をそれぞれユニット数が $N = 100$ と 500 の CAN2 でバッチ学習させた場合の学習回数に対する E_{train} と $E^{(0)}_{\mathrm{test}}$ を **Fig. 4** に示す．この図からすべての関数の学習はほぼ 100 回のバッチ学習で収束していると考えられる．なお f_3 と f_4 の E_{train} と $E^{(0)}_{\mathrm{test}}$ には振動が見られるが，これらの多くは再初期化の影響である．すなわち再初期化が起こる場合は起こらない場合よりも近似誤差の変動が大きいことが多く，また再初期化の直後は近似誤差が大きくなることもある．ただし関数 f_4 については勾配法によるわずかな荷重変化によっても振動が起こった．これは f_4 が不連続関数であることに起因していると考えられる．

3.3　学習能力と内挿能力

バッチ学習を 100 回繰り返した後の CAN2 の訓練データ数 n とユニット数 N に対する E_{train} と $E^{(0)}_{\mathrm{test}}$ を **Fig. 5** に示す．本実験では訓練データ $y = f_i(\boldsymbol{x}) + d$ は雑音を含むがテストデータは雑音を含まないので，ネットが訓練データ中の雑音を除去した $y = f_i(\boldsymbol{x})$ のみをうまく学習するならば，$E^{(0)}_{\mathrm{test}}$ は E_{train} よりも小さくなる．この見地から同図を見ると関数 f_4 以外で訓練データ数 n が大きいときは $E^{(0)}_{\mathrm{test}}$ は E_{train} よりも小さくなり，雑音を除去する学習が行なわれているものと考えられる．

以上の結果をさらに定量的に検討すると，まず E_{train} には最適値が存在すると考えられる．すなわち訓練データ $y = f_i(\boldsymbol{x}) + d$ 中の雑音 d の標本分散 σ_d^2 は作成したすべての訓練データ集合についてほぼ理論値 $3.3 \times 10^{-5} \simeq \int_0^{0.01} x^2 dx / 0.01$ になっていたので，ネットが雑音を含まない真の関数 f_i を学習している場合の E_{train} はこの σ_d^2 の値になる．さらに E_{train} がこの値より小さくなりすぎるとネットは訓練データを過学習していると考えられる．一方，$E^{(0)}_{\mathrm{test}}$ が σ_d^2 より小さくなっていれば雑音を除去した学習が行なわれていると考えられる．そこで Fig. 5 の E_{train} と $E^{(0)}_{\mathrm{test}}$ をユニット数 N に対して再描画したものを **Fig. 6** と **Fig. 7** に示す．まず Fig. 6 から f_4 以外および $n = 1{,}000$ 以外の E_{train} はその最適値である $\sigma_d^2 = 3.3 \times 10^{-5}$ の近傍にあり，良好な学習能力が得られていると考えられる．さらに Fig. 7 から 3.3×10^{-5} より小さな $E^{(0)}_{\mathrm{test}}$ は f_4 以外および n と N が十分大きい場合に実現されている．これらの結果は N を大きくするとより多くの区分領域を用いてより複雑な非線形関数を近似できるが，データ数 n が少ないと各区分領域の学習が良好に行なえない場合が生じることから理解できる．また f_4 の $E^{(0)}_{\mathrm{test}}$ が小さくできなかった場合の検討は後述する (3.7 参照)．

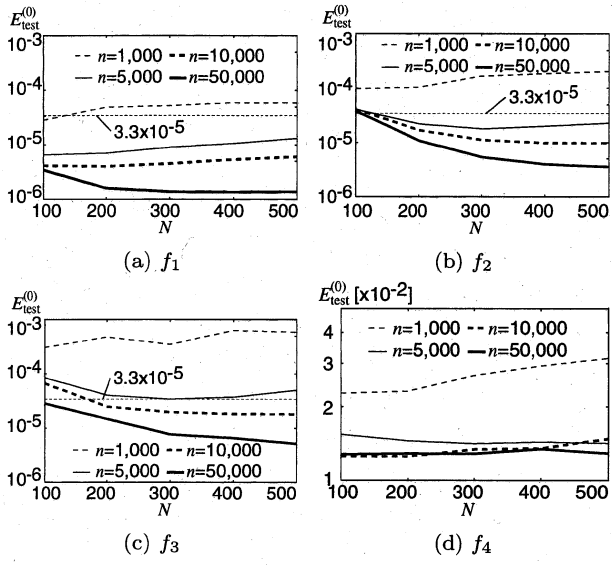


Fig. 7 $E^{(0)}_{\text{test}}$ vs. N

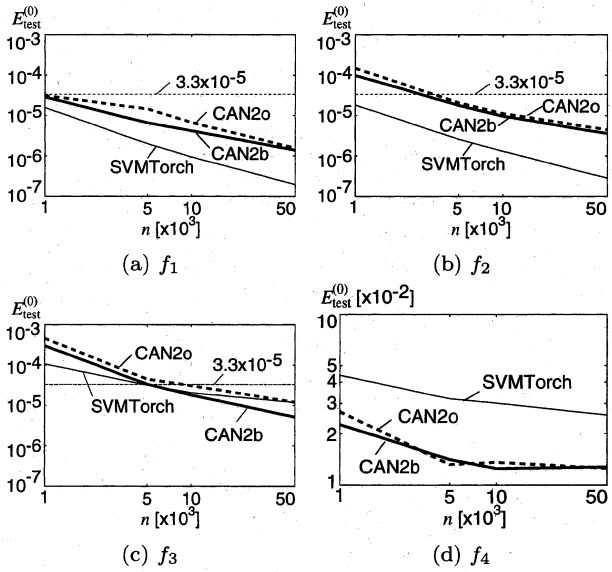


Fig. 8 $E^{(0)}_{\text{test}}$ vs. n

3.4 他手法の内挿能力との比較

Fig. 7 で各関数 f_i と各 n に対して最小の $E^{(0)}_{\text{test}}$ を抽出したものを **Fig. 8** の CAN2b の実線で表わす．比較のため，従来のオンライン学習型 CAN2 の結果を同図の CAN2o の破線で示す．さらに文献 4) において BPN (Backpropagation Net) や RBFN (Radial Basis Function Net) よりも良い結果を示した SVR (Support Vector Regression) の一手法である SVMTorch[18] の結果を同図の細い実線で表わす．なお SVMTorch を用いた理由とそのパラメータ探索の方法は付録 A を参照のこと．またカーネル関数は RBF (Radial Basis Function) すなわちガウス関数 $K(\boldsymbol{x},\boldsymbol{y}) = \exp(-\|\boldsymbol{x}-\boldsymbol{y}\|/r^2)$ を用いた．

Fig. 8 においてまず CAN2b はほとんどすべての訓練データ集合に対して CAN2o より小さな $E^{(0)}_{\text{test}}$ を実現している．

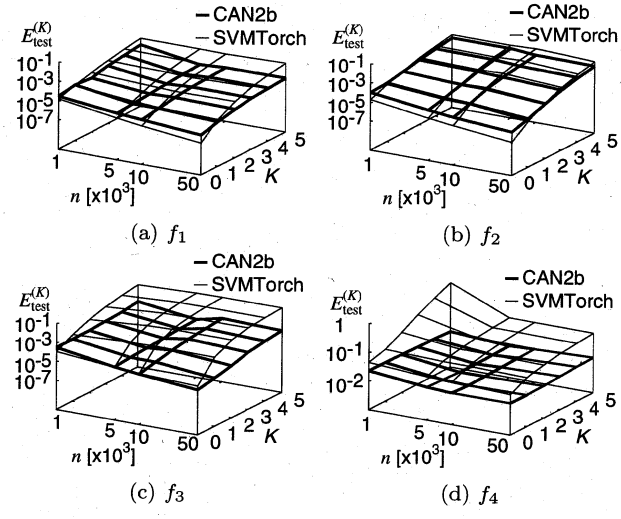


Fig. 9 $E^{(K)}_{\text{test}}$ vs. n and K

なお f_4 で $n = 5{,}000$ のときはその例外になっているが，ほかの場合と比べるとその差はほとんどないとみることができる．CAN2b が CAN2o より $E^{(0)}_{\text{test}}$ を小さくできた主な要因は，連想行列の計算においてボロノイ領域内の訓練データ数が少ない場合の処理を施したこと (2.2.3 参照) などが考えられる．

つぎに SVMTorch は f_1 と f_2 のすべての n についておよび f_3 の $n = 1{,}000$ のとき，CAN2b よりも小さな $E^{(0)}_{\text{test}}$ を実現できたことがわかる．これは CAN2b が k 次元空間上のある 1 点の関数値を雑音を除去して近似するためには $(k+1)$ 個よりも多くの訓練データが必要なのに対し SVMTorch はガウスカーネル関数の機能により f_1 や f_2 のような滑らかな関数を少数の訓練データを学習して内挿する能力に優れていることが原因であると考えられる．

逆に f_3 の $n \geq 5{,}000$ および f_4 のすべての n に対して CAN2b のほうが SVMTorch よりも内挿誤差が小さいのは，f_3 や f_4 は区分的に線形な関数部分をもち，その部分は滑らかに変化するカーネル関数の和で近似するよりも区分的線形関数として近似するほうが誤差を小さくできることが原因であると考えられる．

3.5 訓練データ数に対する外挿能力

前節で SVMTorch はガウスカーネル関数を用いて CAN2b よりも優れた内挿能力を示す場合があることがわかったが，逆にこのカーネル関数は $K \geq 1$ に対する $E^{(K)}_{\text{test}}$ すなわち外挿能力を制約すると考えられる．そこで $K = 0 \sim 5$ の $E^{(K)}_{\text{test}}$ を CAN2b と SVMTorch について計算した結果を **Fig. 9** に示す．この図より SVMTorch の f_1, f_2, f_3 に対する $E^{(K)}_{\text{test}}$ は $K = 0$ では CAN2b より小さいが，$K = 1$ で CAN2b と同程度になり，$K \geq 2$ では CAN2b のほうが小さくなっていることがわかる．また f_4 ではすべての場合について CAN2b のほうが小さくなっている．この結果をより詳細に検討するため $n = 5{,}000$ についての CAN2b と SVMTorch の出力 $\widehat{y}$ の誤差 $y - \widehat{y} = f_i(\boldsymbol{x}) - \widehat{y}$ を $K = 5$ に対応する領域，すなわ

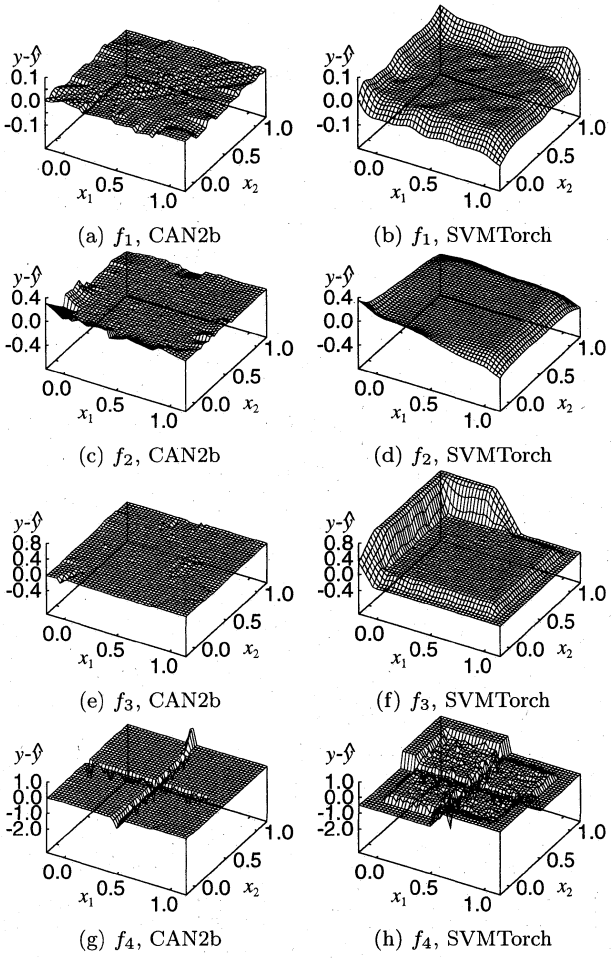


Fig. 10 Extrapolation error of CAN2b (left) and SVMTorch (right)

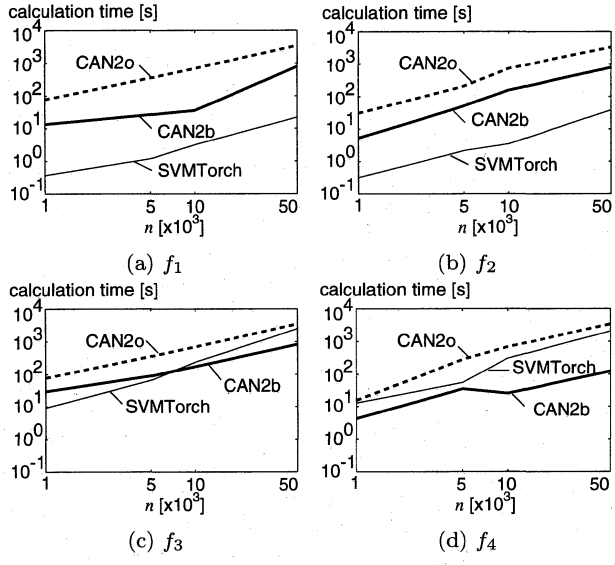


Fig. 11 Computational cost

ち $x_1, x_2 \in [-0.15625, 1.15625]$ の領域について **Fig. 10** に示す．図よりCAN2bの誤差は訓練領域である $x_1, x_2 \in [0, 1]$ の外でもそれほど大きくなく，CAN2bの区分的線形近似が訓練領域外でも良好に機能していることがわかる．これに対し，SVMTorchは訓練領域から離れるにつれて大きな誤差が出ているが，これはSVMTorchの出力は訓練領域から離れるに従い訓練データの関数値の平均値に近づいていくことから理解できる．またSVMTorchの誤差の大きさは訓練領域から離れるにつれて急速に増加しているが，これはこの場合のRBFの幅 r が f_1，f_2，f_3，f_4 でそれぞれ0.1432，0.1680，0.05612，0.0160とかなり小さかったことから理解できる．

なお外挿能力は学習すべき関数の性質に大きく依存するので，一般的な性質として記述することは困難であるが，少なくとも本稿で扱った関数についてはバッチ学習型CAN2は良い性能を示しているといえる．さらに前述のNIPS2004でのコンペティションのgazeデータセットにおいても外挿能力が要求され，バッチ学習型CAN2は良い成績を残している．

3.6 計算時間

各訓練データに対して最小の $E_{\text{test}}^{(0)}$ を与えるパラメータを用いたときの学習に要した計算時間を **Fig. 11** に示す．図よりCAN2bはCAN2oよりも平均して約10倍程度少ない時間で計算できていることがわかり，また最も時間がかかった場合でも約15分であった．これはバッチ学習法はオンライン学習で行なう共通の処理を一括して行なうように工夫しているので一般に短時間で学習できることを反映している．またSVMTorchは関数 f_1 と f_2 のすべての n についておよび f_3 の $n = 1{,}000$ の場合についてCAN2bよりも短い時間で計算できているが，関数 f_3 と f_4 ではCAN2bよりも長く時間がかかっており $n = 50{,}000$ のとき約40分の計算時間がかかった．

なおCAN2bとCAN2oのパラメータ探索はユニット数 $N = 100, 200, 300, 400, 500$ 以外は固定して行なったので各訓練データ集合に対してパラメータ探索を含めた計算時間はFig. 11の5倍程度と見積もることができるが，SVMTorchはパラメータ r を付録Aの段階的な探索法で4〜5段，各段で約8回ずつの探索を行なったのでFig. 11の約32〜40倍の時間が r の探索に必要であったといえる．また不感帯 $\epsilon = 0.01$ と0.001の選択も含めるとさらにその2倍，すなわち約70倍要したことになる．また最適値以外のパラメータ値を用いて80分以上かかって得た結果が最適でないためFig. 11には含めていない場合もある．ただし実際には計算時間を計測するこの数値実験以外は複数の性能の異なる計算機を並列して使用した．

3.7 関数 f_4 とテストデータ

関数 f_4 について，再びFig. 8を見て，CAN2b，CAN2o，SVMTorchのいずれの手法でも $E_{\text{test}}^{(0)}$ の値がほかの関数に比べて数桁大きな値になっていることについて考える．まずこの結果はネットや学習法によるのではなく，この関数 f_4 に特有の性質，より具体的にはこの関数の不連続性に起因するものであると考えられる．さらにテストデータセット $E_{\text{test}}^{(0)}$ を生成した格子点集合 $L(0) = \{(0.5 + i/32, 0.5 +$

$j/32)^T \mid i,j = 0,\pm1,\pm2,\cdots,\pm16\} = \{i/32, j/32)^T \mid i,j = 0,1,2\cdots,32\}$ は f_4 の関数値が不連続な直線 $x_1 = 0.5$ と $x_2 = 0.5$ 上の点を含んでいるので，学習結果の評価をするのに適していない可能性がある．そこで新たに $L_{33}(0) = \{i/33, j/33)^T \mid i,j = 0,1,2,\cdots,33\}$ という格子点集合を用いてテストデータを作り，再実験した．その結果 $n = 50{,}000$ の訓練データに対し，ユニット数 $N = 500$ で $E^{(0)}_{\text{test}} = 1.1 \times 10^{-4}$ が得られ，さらに $N = 800$ とすると雑音の分散 $\sigma_d^2 = 3.3 \times 10^{-5}$ よりも小さな値 2.5×10^{-5} が得られた．一方，CAN2o では $N = 500$ で 1.4×10^{-3} および $N = 800$ で 7.2×10^{-4}，またこのテストデータを用いてパラメータを再探索した SVMTorch では 4.5×10^{-3} までしか小さくすることができなかった．以上の結果は CAN2b は CAN2o や SVMTorch よりも f_4 のような不連続関数を学習する能力にも優れていることを示していると考えられる．

4. おわりに

本稿ではバッチ学習型 CAN2 の学習法について検討しその諸性質を明らかにした．まずバッチ学習法は従来のオンライン学習に以下のような処理を付加することにより性能向上を図っていることを示した．すなわちバッチ学習法は荷重ベクトルを勾配法により更新する際には訓練データの変動範囲に応じた学習係数の値を用いることにより荷重ベクトルの変化量が微小量になるようにしており，また連想行列を更新する際には再初期化された荷重ベクトルのボロノイ領域内の訓練データの個数が小さいときにその周囲から訓練データを補充する手法を用いていることを示した．さらに漸近最適条件を用いて再初期化する際には，訓練データのすべてを用いて歪の推定値が計算できることなど，バッチ学習法の有用性を示した．最後にオンライン学習法および SVR との比較数値実験を行ないバッチ学習型 CAN2 の諸性質と有効性を示した．本稿では入力次元が 2 次のいくつかの関数を学習する際の諸性質を数値実験を用いて検討したにすぎないが，本学習法がより高次の入力次元をもつ関数についても有効に適用可能であることは前述した IJCNN2004 や NIPS2004 のコンペティションで好成績を修めたことから理解できると考える．なお，滑らかな関数の内挿能力は SVMTorch よりも劣ることが判明したが，今後，その原因をさらに追究し，対処法を検討したいと考えている．また非線形時変の温度制御系への応用[11]においてもバッチ学習法によるいくつかの性能の改善が得られているが，今後，さらに良好な性能を得るための検討も行なっていく計画である．

本研究の一部は文部科学省科学研究費，基盤研究 (B) 16300070 の援助を受けました．ここに感謝致します．

参考文献

1) A. C. Ahalt, A. K. Krishnamurthy, P. Chen and D. E. Melton: Competitive learning algorithms for vector quantization, Neural Networks, **3**, 277/290 (1990)
2) T. Kohonen: Associative Memory, Springer Verlag (1977)
3) S. Kurogi and S. Ren: Competitive associative networks for function approximation and control of plants, Proc. NOLTA'97, 775/778 (1997)
4) 黒木，西田：複数のモデルの学習と切り替えを行う競合連想ネットを用いる適応予測制御，計測自動制御学会論文集，**37**-3, 203/212 (2001)
5) 黒木，藤，寺田：競合連想ネットを用いる降水量推定，信学総大，**SD-1**, 260/261 (2001)
6) 黒木秀一：競合連想ネットの漸近最適性と非線形関数の逐次学習への応用，電子情報通信学会論文誌 D-II，**J86-D-II**-2, 184/194 (2003)
7) J. D. Farmer and J. J. Sidorowich: Predicting chaotic time series, Phys. Rev. Lett., **59**, 845/848 (1987)
8) H. Chandrasekaran and M. T. Manry: Convergent design of a piecewise linear neural network, Proceedings of IJCNN1999, **2**, 1339/1344 (1999)
9) R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton: Adaptive mixtures of local experts, Neural Computation, 3, 79/87 (1991)
10) J. H. Friedman: Multivariate adaptive regression splines, Ann Stat, **19**, 1/50 (1991)
11) S. Kurogi, N. Araki, H. Miyamoto, Y. Fuchikawa, T. Nishida, M. Mimata and K. Itoh: Temperature Control of RCA cleaning solutions using batch learning competitive associative net, Proceedings of SCI2004, **V**, 18/23 (2004)
12) S. Kurogi, T. Ueno and M. Sawa: A batch learning method for competitive associative net and its application to function approximation, Proceedings of SCI2004, **V**, 24/28 (2004)
13) S. Kurogi, T. Ueno and M. Sawa: Batch learning competitive associative net and its application to time series prediction, Proceedings of IJCNN2004, CD-ROM (2004)
14) CATS benchmark: *http://www.cis.hut.fi/~lendasse/competition/results.html*
15) Evaluating Predictive Uncertainty Challenge: *http://predict.kyb.tuebingen.mpg.de/pages/home.php*
16) 西田，黒木：再初期化法を用いた適応ベクトル量子化，電子情報通信学会論文誌 D-II，**J84-D-II**-7, 1503/1511 (2001)
17) T. Joachims: Making large-scale suppoert vector machine learning practical, in Advances in Kernel Methods, ed. B. Schölkopf, C. J. C. Burges, and A. J. Smola, 169/184, The MIT Press (1999). Software available at *http://svmlight.joachims.org/*
18) R. Collobert and S. Bengio: SVMTorch — Support vector machines for large-scale regression problems, Journal of Machine Learning Research, **1**-1, 143/160 (2001). Software available at *ftp.idiap.ch/pub/learning/SVMTorch.tgz*
19) Stefan Rüping, mySVM-Manual, Universität Dortmund: Lehrstuhl Informatik VIII (2000). Software available at *http://wwwai.cs.uni-dortmund.de/SOFTWARE/MYSVM/*

《付　　録》

A. SVR の諸手法とパラメータ探索

CAN2 と比較する SVR の手法として，$\text{SVM}^{\text{light}}$ [17]，SVMTorch [18] および mySVM [19]，について試行したが，各手法は最適化アルゴリズムや収束条件の定義の違いなどにより，同じパラメータ値を用いても異なる結果が得られた．たとえば関数 f_3 の $n = 50{,}000$ の訓練データ集合の学習に対して mySVM, $\text{SVM}^{\text{light}}$, SVMTorch のパラメータを粗い最適化により求め $E^{(0)}_{\text{test}}$ の値をそれぞれ 4.1×10^{-5}，1.6×10^{-5}，

Table A.1 A course-to-fine method to search the width r of RBF for SVR

ステップ 1. 第 $i=1$ 段の刻み幅を $\Delta_1=0.1$ とし，$r=j\Delta_1$ $(j=1,2,\cdots)$ に対する $E^{(0)}_{\text{test}}$ を計算する．$E^{(0)}_{\text{test}}$ の細かな変動は無視しながら 4〜5 点程度の範囲で $E^{(0)}_{\text{test}}$ が増加し始めたならばこの段の探索を終了し，最小の $E^{(0)}_{\text{test}}$ を与えた r を $r=r^*_i$ とする．$i:=i+1$ とおき，ステップ 2 にいく．

ステップ 2. 刻み幅を $\Delta_i=0.1/5^{i-1}$ とし，前段での最適値 $r=r^*_{i-1}$ を中心とした 8 点 $r=r^*_{i-1}+j\Delta_i$ $(j=\pm1,\pm2,\pm3,\pm4)$ について $E^{(0)}_{\text{test}}$ を計算し，その最小値を与えた r を $r=r^*_i$ とする．前段の最小値 $E^{(0)}_{\text{test}}(r^*_{i-1})$ に対するこの段の最小値 $E^{(0)}_{\text{test}}(r^*_i)$ の減少率 $1-E^{(0)}_{\text{test}}(r^*_i)/E^{(0)}_{\text{test}}(r^*_{i-1})$ が 5%未満ならこの探索を終了する．そうでなければ $i:=i+1$ とおき，ステップ 2 を繰り返す．

1.8×10^{-5} とすることができたが，そのときの学習に要した時間はそれぞれおよそ 307 分，129 分，40 分であった．各訓練データに対して粗い最適化により各手法が実現できた最小の $E^{(0)}_{\text{test}}$ の大小関係はデータ集合により異なったが，計算時間は常に SVMTorch が最も短かった．そこで本稿では SVMTorch でパラメータ値をより詳細に最適化した結果を示す．各パラメータは以下のように設定し探索した：SVR のマージンと訓練誤差のトレードオフ係数 C は 1,000 とし(注 2)，不感帯の幅 ϵ は 0.01 と 0.001 について調べた．収束判定条件の誤差 (以下，ϵ_T と表記する) は $\epsilon_T=0.0001$ としたが，最適化繰り返し回数は 100,000 回で終了させた．これは関数 f_3 と f_4 では 1,000,000 回でも終了しなかったが，そこで打ち切ったときの $E^{(0)}_{\text{test}}$ は 100,000 回で打ち切ったときとほとんど同じであったからである．カーネルとして RBF すなわち $K(\boldsymbol{x},\boldsymbol{y})=\exp(-||\boldsymbol{x}-\boldsymbol{y}||/r^2)$ を使用し，その幅 r は **Table A.1** の手順により刻み幅を段階的に小さくする手法で探索した．

[著　者　紹　介]

黒　木　秀　一（正会員）

昭和 55 年九州工業大学工学部電気工学科卒．昭和 60 年東工大大学院博士課程修了．同年より九州工業大学制御助手を経て平成 3 年同大・助教授．工博．主にニューラルネットの研究に従事．日本神経回路学会，電子情報通信学会などの会員．

西　田　　健（正会員）

平成 10 年九州工業大学工学部設計生産工学科卒．平 14 年同大学院博士後期課程修了．同年より九州工業大学・制御・助手．工博．主にニューラルネットによるパターン認識の研究に従事．日本神経回路学会，電子情報通信学会などの会員．

渕　川　康　裕（学生会員）

平成 13 年九州工業大学工学部機械知能工学科卒，平成 15 年同大大学院博上前期課程修了．現在同大大学院博士後期課程に在学中．主にニューラルネットによるパターン認識の研究に従事．日本神経回路学会，電子情報通信学会などの学生会員．

(注 2) 係数 C を大きくすると訓練誤差は小さくなるが，大きすぎると過学習が起こり近似関数は滑らかでなくなる．SVMTorch のデフォールト値 $C=100$ では訓練誤差は小さくできず，$C=1{,}000$ として小さな訓練誤差が得られるようになった．さらに汎化誤差を小さくするために Table A.1 による r の最適化を行ない Fig. 10 のような滑らかさの近似誤差が得られた．