# 博士学位論文

# 自己組織化する適応制御器の開発

平成 20 年 3 月

九州工業大学大学院生命体工学研究科

湊原　哲也

# 目次

# 略語一覧

| | |
|---|---|
| AANN | Auto-Associative Neural Network |
| 3 (5)L-AANN | 3 (5) Layer-AANN |
| ASSOM | Adaptive Subspace SOM |
| BMU | Best Matching Unit |
| BMM | Best Matching Module |
| BP | Back Propagation |
| CFC | Conventional Feedback Controller |
| EM | Expectation Maximization |
| ESOM | Evolving Self-Organizing Map |
| GTM | Generative Topographic Mapping |
| HMM | Hidden Markov Model |
| IDML | Inverse Dynamics Model Learning |
| MAE | Mean Absolute Error |
| MDS | Multi Dimensional Scaling |
| MLP | Multi-Layer Perceptron |
| mnSOM | modular network SOM |
| MMRL | Multiple Model Reinforcement Learning |
| MOSAIC | MOdular Selection And Identification for Control |
| MPFIM | Multiple Paired Forward-Inverse Models |
| (G) NG | (Growing) Neural Gas |
| NNC | Neural Network Controller |
| PCA | Principal Component Analysis |
| RBF | Radial Basis Function |
| RNN | Recurrent Neural Network |
| RP | Responsibility Predictor |
| SOAC | Self-Organizing Adaptive Controller |
| SOM | Self-Organizing Map |
| VQ | Vector Quantization |
| WTA | Winner-Takes-All |

# 記号一覧

変数の添字は下付きをデータに関するもの，上付きをユニットあるいはモジュールに関するもの，左肩をモジュール内の要素に関するものとする．

| SOM | |
|---|---|
| $\boldsymbol{x}_i$ | $i$-th data vector |
| $\boldsymbol{w}^k$ | $k$-th reference vector |
| $\mathcal{X}$ | input data set |
| $\mathcal{A}$ | lattice of SOM |
| $\boldsymbol{\xi}$ | coordinate vector in the map space |
| $k^*$ | index of the BMU |
| $\phi$ | neighborhood function (learning rate) |
| $\phi_i^k$ | $k$-th learning rate for $i$-th data |
| $\psi_i^k$ | normalized learning rate |
| $n$ | iteration number [step] |
| $\eta$ | learning coefficient |
| $\sigma(n)$ | neighborhood radius at step $n$ |
| $W$ | reference matrix (set of reference vectors) |
| $X$ | input matrix (set of input data) |
| $\Psi$ | matrix of learning rates |
| **Modular network** | |
| $\boldsymbol{x}_i$ | $i$-th vector |
| $\hat{\boldsymbol{y}}_i^k$ | output of $k$-th module for $i$-th input vector |
| $\hat{\boldsymbol{y}}_i$ | total output of the network |
| $\boldsymbol{w}^k$ | weight vector of $k$-th module |
| $E_i^k$ | error of $k$-th module for $i$-th input vector |
| $\bar{E}_i$ | total error of the network |
| $p_i^k$ | probability selected $k$-th module for $i$-th input vector |
| $T$ | annealing temperature |
| $\eta$ | learning coefficient |
| $\psi$ | learning rate |
| $H$ | set of neighborhoods |
| $C^r$ | average distance between neighborhoods |
| **mnSOM** | |
| $\boldsymbol{x}_{ij}$ | $j$-th input vector belonging to $i$-th class |
| $\boldsymbol{y}_{ij}$ | $j$-th output vector belonging to $i$-th class |
| $\mathcal{D}_i$ | data subset belonging to $i$-th class |
| $f_i$ | function of $i$-th class |
| $L^2(f,g)$ | distance between function $f$ and function $g$ |
| $p(\cdot)$ | probability density function (pdf) |
| $\tilde{\boldsymbol{y}}^k$ | output of $k$-th module |
| $E_i^k$ | ensemble average of $k$-th module for $i$-th class |
| $k_i^*$ | BMM for $i$-th class |
| $\psi_i^k$ | learning rate of $k$-th module for $i$-th class |
| $\boldsymbol{\xi}^k$ | coordinate vector of $k$-th module |

| MPFIM | |
|---|---|
| $x_t, \dot{x}_t, \ddot{x}_t$ | actual position, the velocity, and the acceleration at time $t$ |
| $\hat{x}_t, \hat{\dot{x}}_t, \hat{\ddot{x}}_t$ | desired position, the velocity, and the acceleration at time $t$ |
| $u$ | total control command |
| $u^k$ | control command of $k$-th inverse model |
| $u_{fb}$ | feedback control command |
| $u_{ff}$ | feedforward control command |
| $\sigma$ | standard deviation of Gaussian |
| ${}^f g(\cdot)$ | function of a forward model |
| ${}^i g(\cdot)$ | function of an inverse model |
| $\eta(\cdot)$ | function of an RP |
| ${}^f w$ | parameter of a forward model |
| ${}^i w$ | parameter of an inverse model |
| $\delta$ | parameter of an RP |
| $h(\cdot)$ | function of dynamics of a controlled object |
| $K_P, K_D, K_A$ | feedback gains of Proportion, Differential, and Acceleration |
| $\tilde{x}$ | predicted state |
| $l$ | prior probability |
| $\pi$ | likelihood |
| $\lambda$ | posterior probability |
| $\varepsilon$ | learning coefficient |
| **SOAC** | |
| $t$ | time [sec] |
| $n$ | iteration number [step] |
| $K$ | number of modules |
| $\boldsymbol{\xi}$ | coordinate vector in the map space |
| $\boldsymbol{x}$ | state vector |
| $\hat{\boldsymbol{x}}$ | desired state |
| $\tilde{\boldsymbol{x}}$ | predicted state |
| $\boldsymbol{x}_p$ | input vector to predictors |
| $*$ | index of the BMM |
| ${}^p f(\cdot)$ | function of a predictor |
| ${}^c f(\cdot)$ | function of a controller |
| ${}^p \boldsymbol{w}^k$ | weight vector of $k$-th predictor |
| ${}^c \boldsymbol{w}^k$ | weight vector of $k$-th controller |
| $\sigma(t)$ | neighborhood radius at time $t$ |
| $\sigma_0$ | initial neighborhood radius |
| $\sigma_\infty$ | final neighborhood radius |
| $\tau$ | time constant |
| $\psi$ | learning rate |
| $\phi$ | normalized learning rate (in the execution phase) |
| ${}^p \eta$ | learning coefficient of a predictor |
| ${}^c \eta$ | learning coefficient of an NNC |
| $\varepsilon$ | damping coefficient |

# 第 1 章

# 序論

## 1.1 はじめに

本研究は脳情報工学の観点から人間のような高い適応性と汎化性を持つ制御能力をロボットに持たせるにはどうすればよいかを考え，まずはその足がかりとなる基盤技術の開発を行うものである．

我々を取り巻く環境は常に変化している．そんな中人間は環境の変化に対して柔軟に対応できる能力を持ち，環境に応じた適切な行動を起こすことができる．しかも過去に経験したことのない未知の環境下においても過去の経験を踏まえてそれなりの対応ができる．これが人間が持つ適応性と汎化性である．一方，世の中には作業機械や作業ロボットなるものが存在する．これらは人間の指令には忠実に従うものの，人間から指令が与えられなければ何もできない．人間とは正反対である．人間によって作り出されたものだからとは言え，自ら環境に適応できる能力は皆無である．この違いはどこから生まれているのだろうか．もっと人間のように環境に適応できるロボットは作れないのだろうか．どうすれば作れるであろうか．

近年，ヒトや生物の仕組みに学びその機能を工学的に応用しようという試みの “脳情報工学” が確立されつつある．脳情報工学の目的は，脳内で行われている様々な情報処理の原理を明らかにしそれを工学的に応用することでヒトの役に立つモノを作る，あるいはそのような技術を確立することである．そうすると脳情報工学の観点から先の人間のような制御能力をロボットに持たせることは非常に有意義なものになるのではないだろうか．

例えばロボットアームを用いて物体を特定の軌道に沿って目標の位置まで操作することを考える．これが単に量産化された重さも大きさもすべて特定の物体を操作する課題で

あったならば何も問題は生じない．しかしここでは重さも大きさも形も異なるような物体が混在する状況を想定したい．このような条件下においても問題なく物体操作ができたら有意義ではないだろうか．物体のサイズが変わると慣性などが変化して必要となるトルク量も変化するから物体に合わせて制御器を切り替えなければならない．これを自動で，しかも少ない学習データ数から多数の物体について実現できたら非常に有効ではないであろうか．ここで従来の方法であれば物体ごとに（人間が）手動で制御器を切り替えるか，あるいは自動で制御器を切り替えたとしても物体が変わるたびにゲインを調整して物体に適応するまでに時間を要するであろうことは想像に難くない．

このような試みはアプローチは様々であるものの盛んに行われている．例えば Jacobs, Jordan ら [20, 21] は工学の立場から複数のタスクを 1 つのネットワークで表現するのではなく複数のエキスパートネットワークとゲーティングネットワークによって実現する “mixture of experts” と呼ばれる手法を提案した．Narendra ら [39, 40] は制御工学における適応制御の立場から様々な環境（制御対象）に適応可能な手法を提案した．また Gomi ら [16], Wolpert ら [57], Haruno ら [18, 19] は脳科学における計算論的なアプローチを行っている．これらの手法はそれぞれ一長一短を持つ．例えば Jacobs らの mixture of experts はモジュラーネットワークとしては最初の研究でありこれが元になってその後多数の研究成果が生まれた．しかし複数のエキスパートネットワークを 1 つのゲーティングネットワークで分割するのは実際には困難な場合がある．Narendra らの手法は制御工学の立場から安定性に関する議論がなされており安心して使える．しかし Narendra らのモデルはほとんど学習機能を持っていない．脳科学的なアプローチは脳内で行われているモデルに近づけることが狙いであり，脳機能解明のために必要な研究である．しかし工学的な応用として考えると実際には学習が困難な場合が多く実用性に乏しい面がある．

ところで，モジュラーネットワークのこれまでにない手法として Kohonen の自己組織化マップ（Self-Organizing Map : SOM）とモジュラーネットワークを融合したモジュラーネットワーク SOM （modular network SOM : mnSOM）と呼ばれる脳型情報処理機械が存在する．これは元々は制御器として開発されたものではないが，モジュールのデザインを目的に応じて変えることができるという特徴を持つ．すなわち，この mnSOM を制御系へと応用することでこれまでにない制御器が開発できるのではないかというのが本研究の着想点である．そこで本論文では，mnSOM を制御系へと応用した “自己組織化する適応制御器”（Self-Organizing Adaptive Controller : SOAC）を提案し，その有効性を検証する．

## 1.2 本研究の目的と意義

本研究の目的は人間のような高い適応性と汎化性を持つ制御器の実現への足がかりとして自己組織化する適応制御器（SOAC）のアルゴリズムを確立することである．ここで，“人間のような”とは人間の制御方法からヒントを得ることでこれまでにない制御手法の開発を目指す，という意味であり，生理学的な厳密さを追求するものではない．そこに拘ってしまっては実用性をスポイルすることになってしまうかもしれないからである．例えば，生理学的な厳密さを追求した結果，学習が非常に困難なアルゴリズムになってしまっては実用性がなくなってしまう．したがって，本研究においては厳密さよりも何度試行しても同じ結果を得ることができるような安定した学習アルゴリズムの確立を目指す．

また“高い適応性と汎化性を持つ制御器”とは具体的には以下の 2 点を実現する制御器であると位置づける．

1. 制御対象の特性の突然の変化に対応し得る制御器
2. できるだけ少ない標本数からの汎化的な制御能力を有する制御器

1. に関して，環境の変化は連続的に変化するものだけでなく突然変化することがあるために，このような制御器の構築は不可欠である．例えば，ロボットアーム等を用いて物体を操作することを考える．見た目も重さも全く異なる物体 A と物体 B があったとする．制御対象が A から B へ変われば制御対象に関する物理特性は一瞬のうちにしかも大幅に変化する．しかし対象を思い通りに操作するためには対象の物理特性に適した制御器を選ばなければならないが，一瞬のうちにゲインを調整することは非常に困難である．そこで，提案手法のようなモジュール構造を持つ学習システムが有効となる．学習によって獲得される性質の異なるモジュールを互いに切り替えあるいは組み合わせることによって複数の環境や制御対象に対して速やかな適応性を示すことが可能となる．

2. に関して，一般に獲得したい環境や制御対象の数が増えるほど必要な学習量は増加する．したがって，少ない標本から汎化的な制御能力を獲得できることは学習時間を削減できるという意味で有効である．提案手法は mnSOM を元にしていることで，少ない標本からそれらの内挿によって“中間的なシステム”を多数獲得できる．Jordan ら，Narendra ら，Wolpert らのような従来の方法では必要な環境や制御対象の数だけモジュールを用意しておきそれらについてのみしか学習・制御ができない点，提案手法の方が汎化性の高いシステム記述ができる．

## 1.3　本論文の構成

第 1 章は序論である．

第 2 章は本論文を読むために必要な基礎知識についての解説である．はじめの 2 節は自己組織化マップ，モジュラーネットワークについて述べており，これらの融合により 2.3 節のモジュラーネットワーク SOM が誕生した．2.4 節のフィードバック誤差学習と多重順逆対モデル（MPFIM）では，SOAC の制御器の学習則に導入されているフィードバック誤差学習について，他の類似手法との関係も織り交ぜながら解説している．多重順逆対モデルはフィードバック誤差学習をモジュラーネットワークへと拡張したモデルであり，提案手法と最も関係が深い．本研究において提案手法との比較にも用いられている手法であることから，フィードバック誤差学習と同様に解説の項を設けた．

第 3 章では自己組織化する適応制御器（SOAC）の提案を行う．はじめに 3.1 節で SOAC の目的と特徴について関連手法の紹介を交えながら解説する．3.2 節では SOAC の基本アイデアについて述べる．3.3 節で SOAC のアーキテクチャを示し，3.4 節で学習アルゴリズムを示す．そして 3.5 節では SOAC が表現する “パラメータマップ” についての解説とその用途について述べる．

第 4 章ではシミュレーションによって SOAC の有効性を示す．シミュレーション課題としては簡単なバネ・マス・ダンパ系を用いて物理特性の異なる操作物のアームトラッキングを行った．シミュレーションを通じて SOAC が従来の手法よりも優れた機能を有することを示す．また本章では新たに SOAC の追加学習についての考え方と具体例を示している．

第 5 章では SOAC が生成するマップを有効活用する方法についてパラメータの変化する倒立振子のシミュレーションを通じて述べる．マップの有効活用とは具体的には第 3 章で解説したパラメータマップを用いたモジュールのフィードフォワードの選択法とシステム解析の例を紹介する．

第 6 章は討論であり，本研究で示した SOAC の多手法との関連や特徴，課題，今後の展望などについて述べる．

そして第 7 章にてまとめる．

# 第 2 章

# 基礎知識

## 2.1 自己組織化マップ（SOM）

自己組織化とは生物のように他からの制御なしに自分自身の組織や構造を作り出すことであり，脳内における神経回路の構築は代表的な自己組織化の例である．自己組織化マップ（Self-Organizing Map： SOM）はこの脳内の神経回路の構築をモデル化したものであり，最初に Willshaw と von der Malsburg [56] によって提案された．その後 Amari[2] によって Willshaw と von der Malsburg のモデルを変更した手法が開発された．しかしこれらの生理学的なモデルは学習が難しく，完全に無作為に初期化された状態から理想的な写像を得ることが不可能であるとされている [3].

Kohonen は生理学的なモデルとしては理想化されているもののより強い自己組織化効果を含めるために学習アルゴリズムを変更した [27]．Kohonen のモデルはパラメータに依存しにくく，しかも望ましい結果を得やすいために扱いやすく工学的に有用なモデルとなっている．そのため Kohonen のモデルの応用例は幅広く，制御 [45, 38]，ロボット工学 [31]，画像処理 [9]，音源分離 [43]，経済 [6] など多岐に渡る．このため SOM のアルゴリズムと言えば一般に Kohonen のモデルを指し，本論文においても Kohonen のモデルを扱う．

SOM は人工ニューラルネットワークにおける教師なし学習の一種であり入力空間の位相的性質を保持したまま訓練データを低次元表現することができる．訓練データと写像先の空間（以降マップ空間と呼ぶ）の次元は異なって良いが通常は訓練データが高次元で，マップ空間は 1〜3 次元が使われる．マップ空間の次元を 2 とすれば平面の “地図” ができあがる．したがって SOM は次元削減によりデータを可視化する装置と見なすことがで

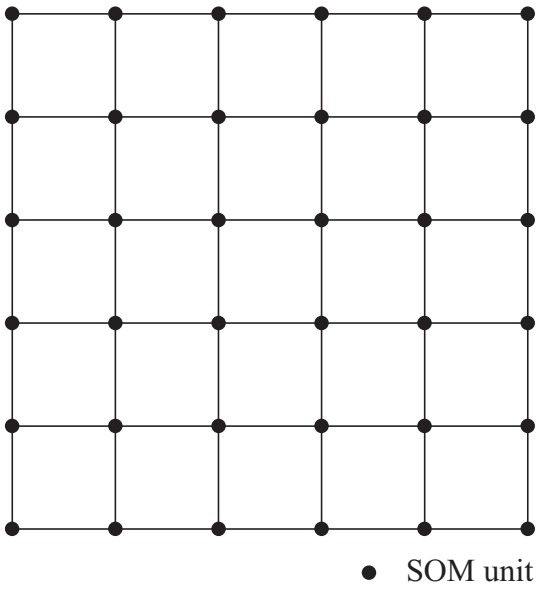



図 2.1　SOM のアーキテクチャ．マップが 2 次元でユニットが正方格子状に配置された場合．

きる．このようなデータ可視化は多次元距離尺度（Multi Dimensional Scaling：MDS）と呼ばれる統計的手法と密接に関わっている．サモンマップ [48] は MDS の代表的な手法である．

SOM のアーキテクチャを図 2.1 に示す．SOM は結合ベクトルを有するユニットが格子状に配置された構造を持ち，各ユニットにはラベルが割り振られていると同時にマップ空間に対して固定座標系が設けられている．したがって $k$-th ユニットに対して結合ベクトル $\boldsymbol{w}^k$ と $\boldsymbol{\xi}^k$ で表される座標値が存在する．$\xi$ は SOM の次元が 1 の場合にのみスカラ値をとり，それ以外の場合（2, 3）はベクトル値となる．以下で SOM の定式化を行う．

今，$d$ 次元の入力ベクトル $\boldsymbol{x}_i = [x_{i1}, ..., x_{id}]^T$ が $I$ 個からなる集合 $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_I\}$ を考える．したがって $\mathcal{X}$ は $d$ 次元入力空間内（$\mathcal{X} \subseteq \Re^d$）に存在する．また $\mathcal{A}$ を対応する入力ベクトルと次元が等しい結合ベクトル $\boldsymbol{w}^k = [w_1^k, ..., w_d^k]^T$ を持つ $K$ 個のユニットからなる格子であるとする．SOM は $\mathcal{X}$ を入力とし SOM のアルゴリズムにおける大きな特徴である (1) 競合作用と (2) 協調作用によって結合ベクトル $\boldsymbol{w}^k$ の学習が行われる．これら 2 つの作用について以下に述べる．

## (1) 競合作用

競合作用とは任意の入力ベクトル $\boldsymbol{x}_i$ に対し最も整合するユニット（Best Matching Unit：BMU）を決定すること（Winner-Takes-All：WTA）である．BMU を決定する

方法として少なくとも 2 つの可能性が存在する．1 つは入力ベクトルと結合ベクトルの内積が最大であるユニット $k^*$ を BMU と定める．

$$k^* = \arg\max_k \boldsymbol{x}_i^T \boldsymbol{w}^k \tag{2.1}$$

もう 1 つは入力ベクトルと結合ベクトル間のユークリッド距離が最小となるユニットを BMU と定める．

$$k^* = \arg\min_k \|\boldsymbol{x}_i - \boldsymbol{w}^k\| \tag{2.2}$$

内積による方法がより生理学的なモデルに近いために初期には内積型が用いられていた．Kohonen が最初に提案したモデルも内積型が用いられていた．$\boldsymbol{w}^k$ を結合ベクトルと表現したのはこのためである．しかし数学的にはユークリッド距離による方法が便利なために特に生理学的なモデルとして SOM を捉える必要がない場合にはユークリッド距離型がよく用いられる．本研究においてもユークリッド距離型を用いているため学習アルゴリズムはユークリッド距離型についてのみ説明する．ユークリッド距離を用いる場合はもはや “結合ベクトル” という呼び名はふさわしくなく，ベクトル量子化（Vector Quantization : VQ）の観点から “参照ベクトル” などと呼ばれる[*1]．

### (2) 協調作用

協調作用とはあるニューロンが発火することによりその影響を受けてその他のニューロンも発火しやすくなることであり，SOM における協調作用とは競合作用によって勝者となったユニット（BMU）だけでなくその近傍ユニットも学習する権利を獲得することである．すなわち BMU が最も学習量が大きく，マップ空間において BMU からの距離が離れるほど獲得できる学習量が小さくなる．

ユークリッド距離を用いた場合の参照ベクトルの学習則は次のように与えられる．

$$\Delta \boldsymbol{w}^k = \eta \phi^k(n)(\boldsymbol{x}_i - \boldsymbol{w}^k), k \in \mathcal{A} \tag{2.3}$$

ここで $\phi^k$ は近傍関数と呼ばれ，通常は次のようなガウス型近傍関数が用いられる．

$$\phi^k(n) = \exp\left(-\frac{\|\boldsymbol{\xi}^{k^*} - \boldsymbol{\xi}^k\|^2}{2\sigma(n)^2}\right) \tag{2.4}$$

[*1] 本論文においては以降参照ベクトルといえば SOM のユニットが持つ $\boldsymbol{w}^k$ のことを表し，結合ベクトルは線形ネットワークや多層パーセプトロンなどの階層型ネットワークにおける結合荷重を表すこととする．

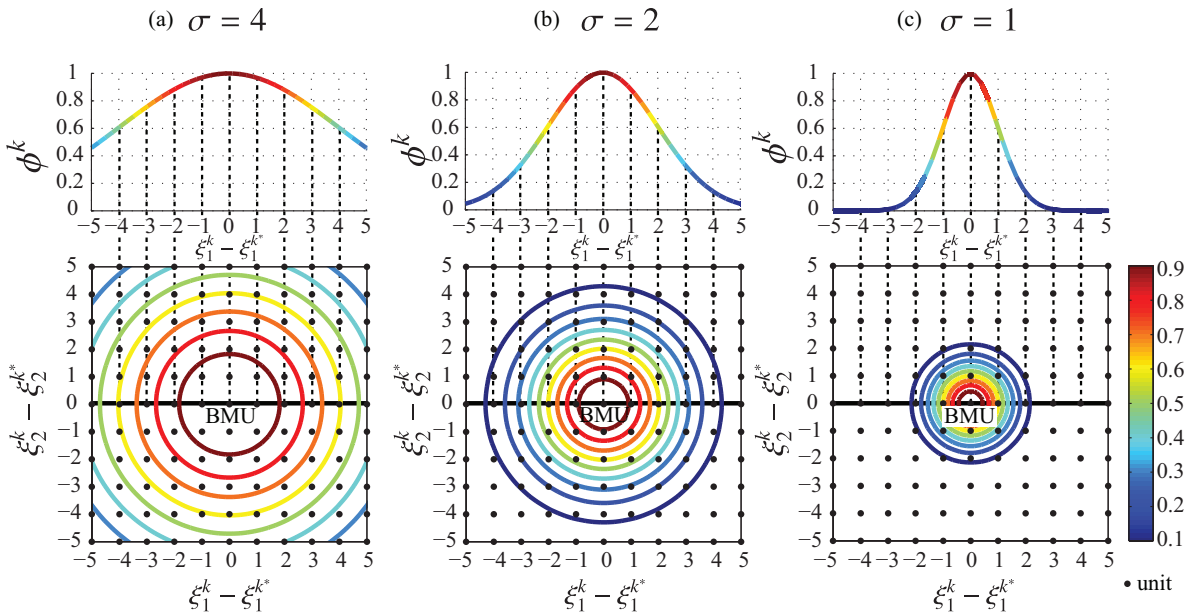



図 2.2　ガウス型近傍関数．(a) $\sigma = 4$．(b) $\sigma = 2$．(c) $\sigma = 1$．

ここで $\eta$ は学習係数で正の定数であり，$\sigma(n)$ は近傍半径の大きさを決めるパラメータである．$\sigma(n)$ は学習回数 $n$ とともに単調減少する（図 2.2 参照）．また学習を安定させるために $\sigma$ だけでなく $\eta$ も単調減少させることがある．

基本的には近傍関数の半径を小さくしながら (1) と (2) を繰り返すことによって SOM のマップが生成される．その結果入力空間内で近くに位置するユニットはマップ空間においても近い位置に存在するといった位相情報を保持したマップが生成される（図 2.3）．

SOM の学習則は入力データが 1 つ提示されるごとに参照ベクトルの修正を行う式 (2.3) で表されるオンライン型と入力データのセットが提示された後に参照ベクトルの修正を 1 度行うバッチ型が存在する（Kohonen はバッチマップと呼んでいる）．最初に提案されたのはオンライン型であり，バッチ型学習のアルゴリズムは後に幾つかの方法で理論的に導出された．例えば Luttrell[29, 30] は一般化ロイドアルゴリズムを用いて，Bishop[5] は期待値最大化（Expectation Maximization：EM）アルゴリズムを用いて Generative Topographic Mapping (GTM) としてバッチ型学習則を導出した（式 (2.5)）．式 (2.5) から分かるようにオンライン型に比べバッチ型の更新則には学習係数 $\eta$ が省略されており，次時刻における参照ベクトルの値は近傍関数により得られた学習率と入力データベクトルとの重み付き重心点で定まる．そのためバッチ型においては収束問題が生じない．したがってバッチ型はオンライン型よりも安定したマップの生成が可能であり，多くの場合でオンライン型よりもうまく機能する．特に事前にデータセットが得られる場合にはバッチ

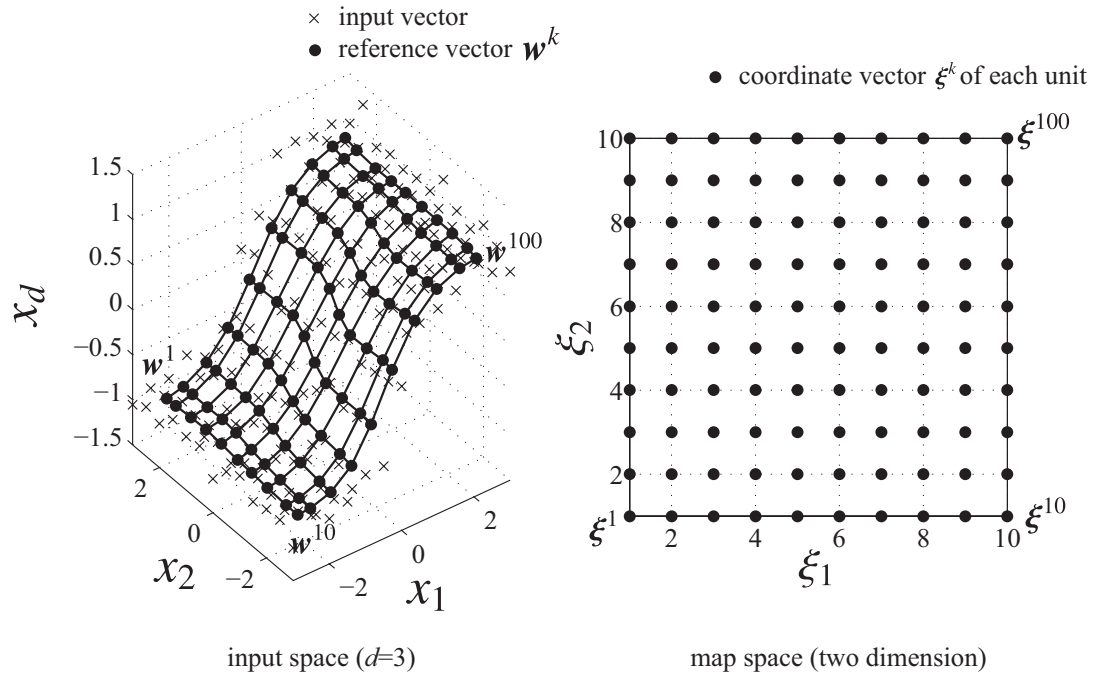



図 2.3　入力空間からマップ空間への写像.

型を用いて学習を行うとよいであろう．さらにオンライン型に比べバッチ型は学習に要する時間（繰り返し回数）を大幅に削減できることも魅力的である．後述のモジュラーネットワーク SOM においても，これらの利点（学習の安定性，高速性）からバッチ型 SOM のアルゴリズムを基本としている．

$$\boldsymbol{w}^k = \sum_{i=1}^{I} \psi_i^k \boldsymbol{x}_i \tag{2.5}$$

$$\psi_i^k = \frac{\phi_i^k}{\sum_{i'=1}^{I} \phi_{i'}^k} \tag{2.6}$$

オンライン型の場合と異なりバッチ型では各データごとに学習率が存在することに注意する．またバッチ型の学習則は行列表記を用いて次のように簡潔に表すことができる．

$$W = X\Psi \tag{2.7}$$

ただし，

$$W = \{\boldsymbol{w}^1, ..., \boldsymbol{w}^k\} \tag{2.8}$$

$$X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_I\} \tag{2.9}$$

$$\Psi = \begin{bmatrix} \psi_1^1 & \cdots & \psi_1^K \\ \vdots & \ddots & \vdots \\ \psi_I^1 & \cdots & \psi_I^K \end{bmatrix}. \tag{2.10}$$

### 2.1.1 アルゴリズムの要約

#### オンライン型

1. BMU の決定

$$k^* = \arg\min_k \|\boldsymbol{x}_i - \boldsymbol{w}^k\| \tag{2.11}$$

2. 学習率の決定

$$\phi^k(n) = \exp\left(-\frac{\|\boldsymbol{\xi}^{k^*} - \boldsymbol{\xi}^k\|^2}{2\sigma(n)^2}\right) \tag{2.12}$$

3. 参照ベクトルの修正

$$\Delta\boldsymbol{w}^k = \eta\phi^k(n)(\boldsymbol{x}_i - \boldsymbol{w}^k), k \in \mathcal{A} \tag{2.13}$$

#### バッチ型

1. BMUs の決定

$$k_i^* = \arg\min_k \|\boldsymbol{x}_i - \boldsymbol{w}^k\| \tag{2.14}$$

2. 学習率の決定[*2]

$$\phi_i^k = \exp\left(-\frac{\|\boldsymbol{\xi}_i^{k^*} - \boldsymbol{\xi}^k\|^2}{2\sigma^2}\right) \tag{2.15}$$

$$\psi_i^k = \frac{\phi_i^k}{\sum_{i'=1}^I \phi_{i'}^k} \tag{2.16}$$

3. 参照ベクトルの修正

$$\boldsymbol{w}^k = \sum_{i=1}^I \psi_i^k \boldsymbol{x}_i \tag{2.17}$$

### 2.1.2 SOM の機能

本節では SOM における可視化，分類，内挿の 3 つの機能について取り上げる．特に可視化機能と内挿機能は本研究において重要な役割を果たす．それを踏まえてシミュレーションを交えながらこれらの機能について説明を行う．

[*2] 式 (2.15) の $\xi_i^{k^*}$ は正確に表記すれば $\xi^{k_i^*}$ となるが，見やすさを重視して表記を変えていることに注意する．

表 2.1　動物データ．

| | | dove | hen | duck | goose | owl | hawk | eagle | fox | dog | wolf | cat | tiger | lion | horse | zebra | cow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| is | small | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | medium | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | big | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| | nocturnal | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0.5 | 0 | 1 | 0.5 | 0.5 | 0 | 0 | 0 | 0 |
| | herbivorous | 0.5 | 0.5 | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| has | 2 legs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 legs | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | hair | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | hooves | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | mane | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| | feathers | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | stripes | 0 | 0 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| likes to | hunt | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | run | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| | fly | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | swim | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

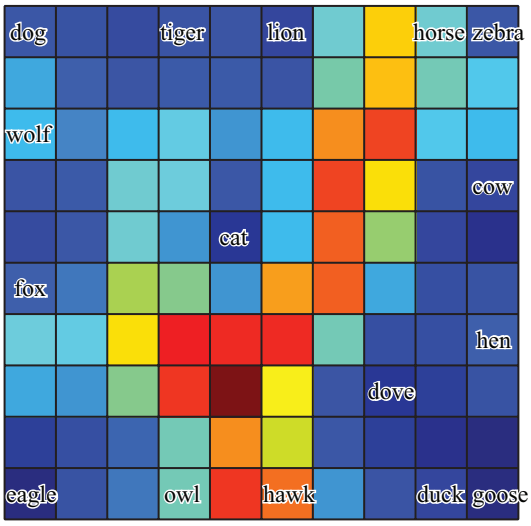



図 2.4　SOM による動物データの可視化．

## SOM の可視化機能

SOM は高次元のデータを低次元空間へと写像することでデータ間の類似度を視覚化できる．このことを確かめるために表 2.1 に示すような動物のデータを用いたシミュレーション例を紹介する．このデータは文献 [55] に記載されていたデータである[*3]．データ

[*3] ここでデータの一部に関して断っておく．表 2.1 中の “鷲”（eagle）の “飛ぶことが好きである” という特徴が 0 となっているが，これは明らかに誤植であると思われる（印字ミスではなくデータファイルその

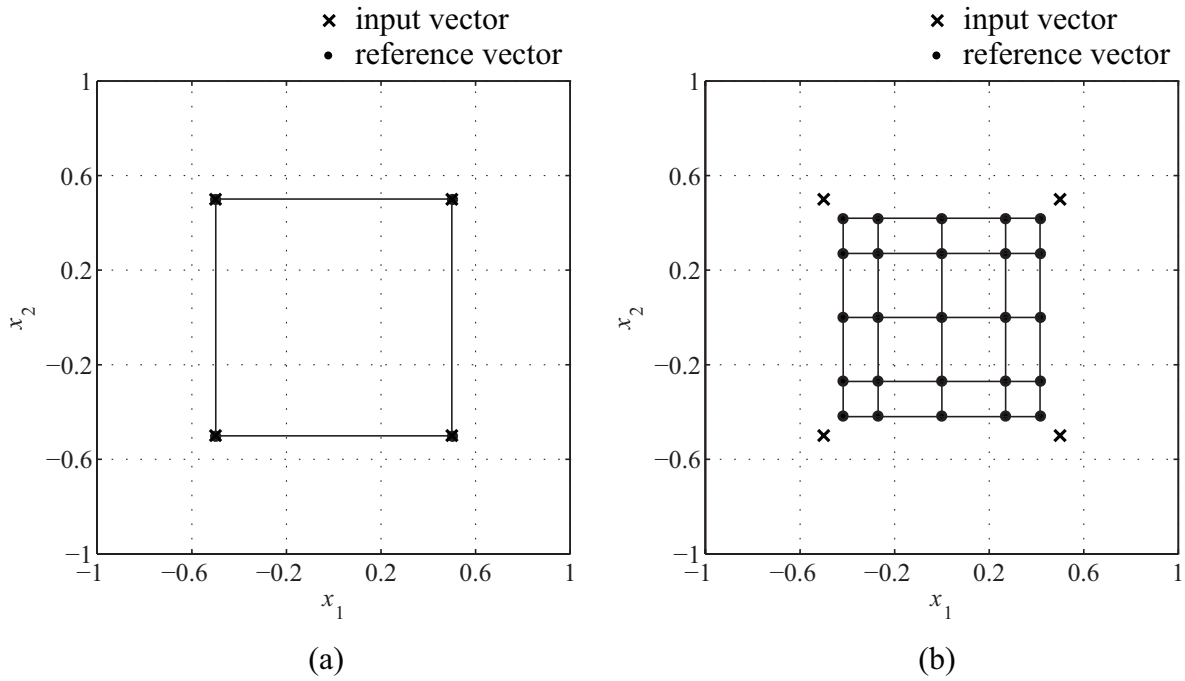



図 2.5　SOM の分類機能と内挿機能の例．(a) 分類機能．(b) 内挿機能．

は 16 種類の動物から成り，各動物は 16 個の特徴を有する（つまりデータ数 $I = 16$ ，入力データの次元 $d = 16$）．16 次元空間に存在するデータをプロットすることは普通できないために，SOM を用いたデータの視覚化が役立つ．

図 2.4 は実際に動物データに SOM を適用して得られたマップである．ここでは 100 個のユニット（マップ次元は 2）を用いた結果を示している．図中の色はあるユニットが表現している参照ベクトルと，それに隣接するユニットが表現している参照ベクトルとの間に生じる距離の平均を表している．式で表せば，

$$Cr^k = \frac{1}{|H|} \sum_{h \in H} \|\boldsymbol{w}^k - \boldsymbol{w}^h\| \tag{2.18}$$

となる．ここで $h$ は $k$-th ユニットに隣接しているユニットの番号を表し，$H$ は $h$ の集合である．この式の意味するところは $Cr^k$ の値が大きければ近傍ユニットとの距離が遠く，逆に小さければ近傍ユニットとの距離が近くなる．カラーとの対応は，距離が遠い状態が赤色で，距離が近い状態が青色となっている．したがって例えば梟（owl）と鷹（hawk）はマップ空間では 2 しか離れていないが，表現している参照ベクトルはさほど似ていない

ものが間違っている）．しかし文献 [55] と同じデータを扱うことに意義があると判断し，誤った特徴データのままシミュレーションを行った．ただし，データを正しくしても生成されるマップに違いはほとんど見られない．

ことを意味する．このような色付けは SOM を使う際にはよく使われており U-matrix などと呼ばれる．

改めて動物マップを見てみると，例えば犬（dog）と狼（wolf）はマップ空間上の近くにあるので似ている，逆に犬（dog）とガチョウ（goose）はマップ空間上の離れた位置にあるので似ていない，といった実際に我々の認識と合致した動物に関する地図が生成できているのが分かる．これが SOM の持つ可視化機能である．

以下ではデータを変えて次のような状況を考える．話を簡単にするために入力データの次元は $d = 2$ で SOM のマップ次元は 2 であるとする（通常は入力データ次元の方が高い）．ここで扱うデータはわずかに 4 点とし，この入力データは 2 次元平面上に正方形を描き，正方形の中心位置を原点としたときの座標値からなるとする（図 2.5 参照）．このデータに対し SOM を適用することを考える．

### SOM の分類機能

SOM の 2 つ目の機能として分類が挙げられる．これは通常入力データがいくつかのクラスタ（データの群れ）を形成している場合において，クラスタリングのために用いられる．例ではデータが 4 点あり，すべてのデータは離れて存在していることからそれぞれ独立したクラスであるとすれば，4 つのクラスタが存在すると見なせる．このデータに対し分類を目的として SOM を適用する．具体的には SOM のユニット数をデータ数と等しく $K = 4$ とし近傍半径を学習終了時に 0.1 まで落とす．このときの結果が図 2.5(a) である．この場合には SOM の各参照ベクトルが入力データベクトルのいずれかを表現し，かつデータの位相情報を保持したままマップが形成されているのが分かる．これが SOM の分類機能である．例ではデータ数が各クラスごとに 1 点しか存在しなかったが，例えばデータがもっとたくさんあり，各データの座標値を平均値として持つ 2 次元ガウス関数（分散は非常に小さいとする）にしたがって分布しているときには，各参照ベクトルが 1 つのクラスの平均ベクトルを表現する．

### SOM の内挿機能

SOM のもう 1 つの機能として内挿が挙げられる．データが十分にたくさんある場合には SOM はデータの分布によって分類器として機能したり，あるいは量子化器として働く．そうでない場合，すなわち得られたデータ数が非常に少ない場合には，データとデータの間を内挿し入力データの中間的なベクトルを参照ベクトルが表現する．これが SOM

の内挿機能である．SOM の内挿機能が働いた例が図 2.5(b) である．この場合には SOM のユニット数を多く設定（$K = 25$）し，近傍半径はある程度残す必要がある（最終値は $\sigma = 1.8$）．内挿機能は，本来はデータが連続的に分布しているが，何らかの原因によってそのごく一部しか学習に用いることができないような状況に有効である．この例で言えば，実はデータの母集団は正方形内に一様に分布しており，そこからたまたま得られた 4 点のデータだけを用いて学習を行った，という状況である．この時には新たに提示された未学習のパターンに対してそこそこ有効に働く．つまり内挿機能は未学習データに対する汎化性を向上させることがある．

これは筆者の考えであるが，SOM の分類機能よりも内挿機能の方が要求される場面は多い．それは単にデータ集合を複数のクラスに分類したいだけなら他にももっと有効な手法が多数存在するからである．例えばクラスタリングだけが目的であるならば一般化ロイドアルゴリズム（k-means 法とも呼ばれる）を使えばよい．したがって SOM の有意義な使い道はデータの視覚化と内挿であると思われる．なお，ここでは SOM に関する 3 つの機能を紹介したが，SOM が有する機能はこの限りではないことを断っておく．SOM には様々な使い道がありユーザによってその目的は異なるはずであるから機能を限定する必要はない．ここではあくまでも SOM の可視化機能と内挿機能が提案手法において重要な役割を果たすことからその前準備として対極にある分類機能との比較を例にとって解説したにすぎない．

## 2.2　モジュラーネットワーク

SOM は特徴空間（入力空間）を複数の参照ベクトルで表現する装置であり，違う表現を用いれば，特徴空間を分割し，分割された各領域を 1 つの参照ベクトルで担当する．このとき SOM は分割されたどの空間においてもデータベクトルの密度がほぼ同じになるように参照ベクトルを配置する．

モジュラーネットワークは入力空間をいくつかの部分空間に分割し，複数のモデルを用いてそれぞれのモデル（モジュール）がある部分空間を表現する手法であり，最初に Jacobs と Jordan らによって提案された [20]．Jacobs らの方法は “mixture of experts” と呼ばれ，いくつかのエキスパートネットワークと 1 つのゲーティングネットワークから構成される．エキスパートネットワークはある状況に特化した文字通りのエキスパートであり，あるクラスに属するデータ集合を最もよく表現したサブモデルとなっている．それに対しゲーティングネットワークは入力データがどのモデルに属するかのパターン認識

を行うネットワークであり，ゲーティングネットワークによって入力空間が分割される．Jordan[21] は後に mixture of experts に階層構造を導入した手法を提案している．

Jacobs らの mixturre of experts の元々の発想は，MLP のような単一の階層型ネットワークでは複数のサブタスクの獲得が困難であることにあった．例え学習が可能であったとしても学習の収束に時間が掛かったり，未学習データに対する汎化性が劣化することから，複数のネットワークを用いてタスクを分割することでこの問題を解決できると考えていた．このことは多くの課題について当てはめることができ，それ故これまでに種々のモジュラーネットワークが提案された [8, 11, 17, 18, 19, 45, 39, 40, 44, 47, 49, 50, 57]．ここですべての手法を網羅することはできないが，特に本研究に関わりの深い手法について以下で取り上げる．

古川は競合モジュラーネットワークを提案し，モデル化とクラスタリングを同時に行える方法を示した [11]．この手法がモジュラーネットワーク SOM の基になっている．したがってこの手法については次節で解説を行う．

Gomi ら [17] は mixture of experts を応用し物理特性の異なる制御対象の学習・制御を試みた．その結果，学習は可能であったものの非常に困難であったと述べている．

Narendra ら [39, 40] は複数モデルを制御に応用し，予測器と制御器を対にした手法を提案した．Narendra らの手法では予測誤差を最小としたモジュールと対になっている制御器を用いて制御が行われる．

Wolpert と Kawato[57] によって提案された多重順逆対モデル（Multiple Paired Forward-Inverse Models： MPFIM）においても Narendra らの手法と同様に順モデル（予測器）と逆モデル（制御器）を対にしたモジュールが用いられている．ただし Narendra らの手法では予測誤差を最小としたモジュールが選択されるのに対し，MPFIM では予測誤差を元に計算された “責任信号” を用いて制御信号だけでなく予測器と制御器の学習にも重み付けを行っている．MPFIM は既存の手法の中で提案手法に最も近い構造を持っていることから 2.5 節で解説を行う．

### 2.2.1 古川の競合モジュラーネットワーク

古川の競合モジュラーネットワークのアーキテクチャを図 2.6 に示す．ここでは各モジュールは 3 層からなる自己想起型ニューラルネットワーク（3 Layer Auto-Associative Neural Network： 3L-AANN）により構成される．3L-AANN は一種の MLP であるが，入力層と出力層のユニット数 $N_D$ は同じでかつ，隠れ層のユニット数 $N_H$ は入出力層よ

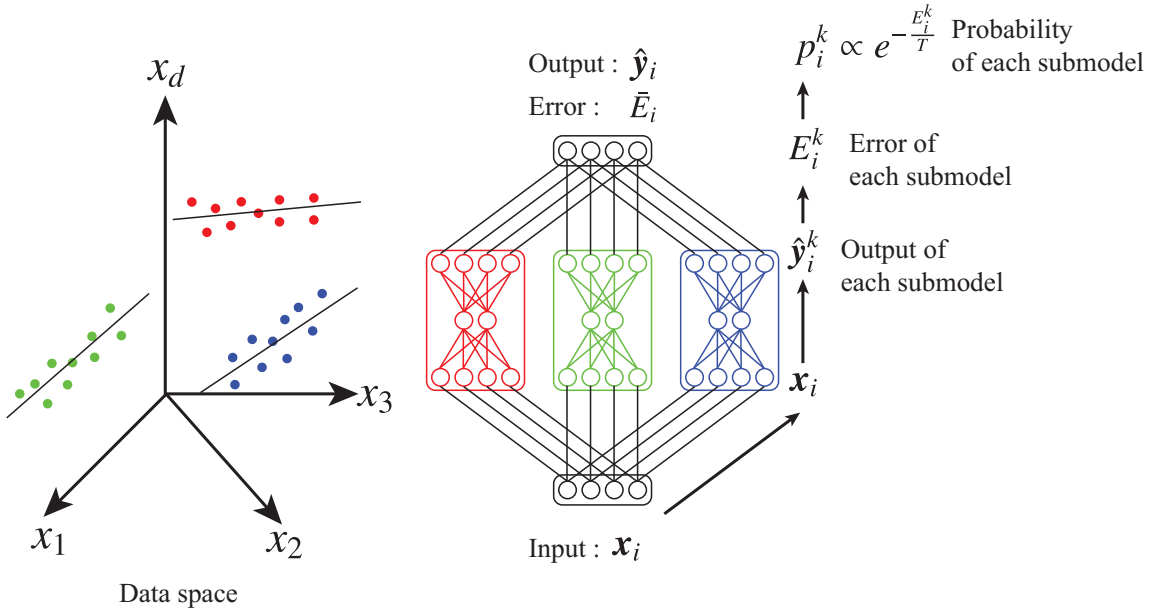



図 2.6　競合モジュラーネットワークのアーキテクチャ．

りも少ない（$N_H < N_D$）こと，さらに教師信号は入力データと等しい点が通常の MLP と異なる（したがって特別な教師信号が存在しないという意味では 3L-AANN は教師なし学習と見なすことができる）．3L-AANN の目的は入力データを出力で想起する，すなわち恒等写像を行うことである．ただし入力層の次元よりも隠れ層の次元の方が低いために隠れ層において入力データは低次元表現（圧縮）される．したがって出力層で復元したときには通常誤差が生じる．3L-AANN ではこの復元誤差が最小となるように結合荷重の更新を行う．このことから 3L-AANN はしばしば Auto-encoder とも呼ばれる．また 3L-AANN は主成分分析（Principal Component Analysis : PCA）と深く関わっており，例えば隠れ素子数を 2 に選ぶと PCA によって第 2 主成分まで求めそれを用いてデータをモデル化した場合に相当する．

古川の競合モジュラーネットワークの目的は与えられたデータ群を局所的なサブモデルの組み合わせで表現することを通して，最適なモデル化とクラス分類を同時に行うことである．古川はアヤメの分類課題（3 種のアヤメ）において教師信号を用いることなく 3 種のアヤメの分類とモデル化に成功した（シミュレーションでは 10 個のモジュールを用いて競合学習を行わせたところ，3 つのモジュールのみが生き残りの 7 つは淘汰されたとある）．以下競合モジュラーネットワークについて詳細に述べる．

今，$I$ 個の入力ベクトル集合 $\{\boldsymbol{x}_i\}(\boldsymbol{x}_i \in \Re^{N_D})$ があり，モジュラーネットワークのモジュール数を $K$ とする．次にデータ集合の中からデータ 1 個を取り出し，これを入力 $\boldsymbol{x}_i$

として各モジュールに与える．このとき各モジュールの出力 $\boldsymbol{y}_i^k$ と 3L-AANN の教師信号である入力ベクトル $\boldsymbol{x}_i$ との間に生じる復元誤差 $E_i^k$ は次式で表される．

$$E_i^k = \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{y}_i^k\|^2 \tag{2.19}$$

$\boldsymbol{x}_i$ に対して $k$-th モジュールが選ばれる確率 $p_i^k$ を次式で定義する．

$$p_i^k = \frac{\exp[-E_i^k/T]}{\sum_{k'}\exp[-E_i^{k'}/T]} \tag{2.20}$$

ここで $T$ はアニーリング温度であり，通常学習の初期には高い温度に設定し，学習が進むにつれて徐々に温度を下げていく．こうすることで学習の初期では各モジュールの選択確率がどれも均等になりどのデータについても平均的に学習する．その後学習が進むに連れ $T$ が小さくなりモジュール選択が偏る．モジュール選択が偏るとはすなわち，各モジュールは想起誤差が小さいデータについてはより学習が促され，一方想起誤差が大きいデータについては学習しなくなる．したがって最終的にはクラスタを形成しているサブクラスについて特化したモデル（いわゆるエキスパート）が形成される．

さて，ここまでで各モジュールの誤差関数とモジュールの選択確率が定義できた．次にネットワーク全体の評価関数を定義しよう．まずネットワーク全体の出力値を誤差の期待値をとって，

$$\hat{\boldsymbol{y}}_i = \sum_{k=1}^{K} p_i^k \boldsymbol{y}_i^k \tag{2.21}$$

と定義する．ただし，課題に応じては選択確率を最大とするモジュールを勝者モジュールとして次のようにすることも可能である．

$$\hat{\boldsymbol{y}}_i = \hat{\boldsymbol{y}}_i^k \tag{2.22}$$

$$k^* = \arg\max_k p_i^k \tag{2.23}$$

ネットワーク全体の誤差も同様に，誤差の期待値を用いて次式で表す．

$$\bar{E}_i = \sum_{k=1}^{K} p_i^k E_i^k \tag{2.24}$$

各モジュールの学習は上式のネットワーク全体の誤差が最小になるように行われる．3L-AANN は通常の誤差逆伝搬（BP）学習によって行われるので，$k$-th モジュールの学

習則は結合ベクトルを $\boldsymbol{w}^k$ として次のように表せる．

$$\Delta \boldsymbol{w}^k = -\eta \frac{\partial \bar{E}_i}{\partial \boldsymbol{w}^k} \tag{2.25}$$

$$= -\eta \left\{ p_i^k \frac{E_i^k}{\partial \boldsymbol{w}^k} + \sum_{k'=1}^{K} E_i^k \frac{\partial p_i^{k'}}{\partial \boldsymbol{w}^k} \right\} \tag{2.26}$$

$$= -\eta \psi_i^k \frac{\partial E_i^k}{\partial \boldsymbol{w}^k} \tag{2.27}$$

すなわち，上式は通常の BP 学習則に $\psi_i^k$ を乗じたものであり，$\psi_i^k$ は $i$-th データに対して $k$-th モジュールがどの程度学習をするかを決める学習率である．学習率 $\psi_i^k$ は次式で表される．

$$\psi_i^k = p_i^k \left\{ 1 + \frac{1}{T}(\bar{E}_i - E_i^k) \right\} \tag{2.28}$$

以上が競合モジュラーネットワークのアーキテクチャと学習アルゴリズムであり，基本的に $T$ を小さくしながら競合学習を行わせることによって，データの棲み分けが自己組織的に行われる．これが後述のモジュラーネットワーク SOM へと発展していく．

## 2.3　モジュラーネットワーク SOM（mnSOM）

### 2.3.1　はじめに

Kohonen の SOM は人工ニューラルネットワークにおける教師なし学習法の 1 つであり，高次元空間内のベクトルデータを 1〜3 次元程度の低次元空間へ位相を保存しながら写像することでデータ間の関係を視覚化することが可能である [27]．すなわち，SOM ではベクトルデータ間の距離（通常ユークリッド距離）に基づき入力データ空間において距離の近いデータはマップ空間の近い位置に，逆に入力データ空間において距離の遠いデータはマップ空間の遠い位置に配置された “地図” を生成できる．このようなデータの可視化は SOM における大きなの特徴の 1 つであり，通常我々が扱うデータは高次元となる場合が多いことから，SOM はデータ分類や，データ解析といったデータマイニングの一技術として特に有効である．

では，この可視化機能をもっと一般化し，マップしたい対象をベクトルデータだけに限定することなく，例えば静的システム群のマップ，動的システム群のマップ，関数群のマップなどを作ることができないであろうか．これができればシステムの分類や，システム解析などが可能になり，より面白い研究へと発展するのではないだろうか．残念なこと

に従来の SOM ではベクトルデータしか扱うことができないためにこのような，“機能の自己組織化マップ” の実現は困難であり，実現するには創意工夫が必要であった．

徳永，古川ら [52, 53, 54, 14] はモジュラーネットワーク SOM（modular network SOM : mnSOM）と呼ばれるモジュラーネットワークと Kohonen の SOM を融合した新しいニューラルネットワークを提案し，mnSOM によって SOM を一般化できることを示した．mnSOM の考え方は非常に単純明瞭であり，“SOM がベクトルデータしか扱えなかったのは SOM の基本ユニット（参照ベクトル）がベクトルユニットであったことに起因する．したがって SOM の基本ユニットとしてニューラルネットワークなどの機能モジュールを選べば他のデータタイプが扱えるようになる，” というものである．機能モジュールには，多層パーセプトロン（Multi-Layer Perceptron : MLP）や動径基底関数（Radial Basis Function : RBF），リカレントニューラルネットワーク（Recurrent Neural Network : RNN），SOM，ニューラルガス（Neural Gas : NG）[32] などが用いられる．

例えば，機能モジュールとして MLP を選べば（MLP を用いた mnSOM を MLP-mnSOM と略称する），各モジュールが 1 個の関数を表現できるためネットワーク全体として関数群の自己組織化マップが生成できる [52, 53]．これらの研究では mnSOM が単なる SOM の一拡張ではなく，一般化するものであることが理論的に示されている．すなわち SOM がデータベクトル空間における自己組織化マップを生成するのに対し，MLP-mnSOM は関数空間における自己組織化マップを生成するものであることが証明されている．また MLP の一種である自己想起型ニューラルネットワーク（AANN）を用いた研究では Kohonen ら [26] の適応部分空間 SOM（Adaptive Subspace SOM : ASSOM）を 5 層の AANN（5L-AANN）を用いることにより非線形適応部分空間 SOM（Non-Linear ASSOM : NL-ASSOM）へと発展させた [51]．ここで ASSOM とは一種の部分空間法であり，SOM の参照ベクトルを “参照部分空間” へ発展させたものと見なせる．通常の SOM では特徴ベクトル（入力ベクトル）間のユークリッド距離に従いマップが生成されるが，ASSOM では線形部分空間を表現するモデル間の距離に従いマップが生成される．したがって ASSOM は 3L-AANN-mnSOM（3 層の自己想起型ニューラルネットワークを機能モジュールとした mnSOM）と基本的に等価である．これを 5 層とすることでモデルの表現を線形部分空間から非線形な部分空間へと拡張したものが 5L-AANN-mnSOM であり，これによって非線形な ASSOM すなわち NL-ASSOM が実現可能となる．

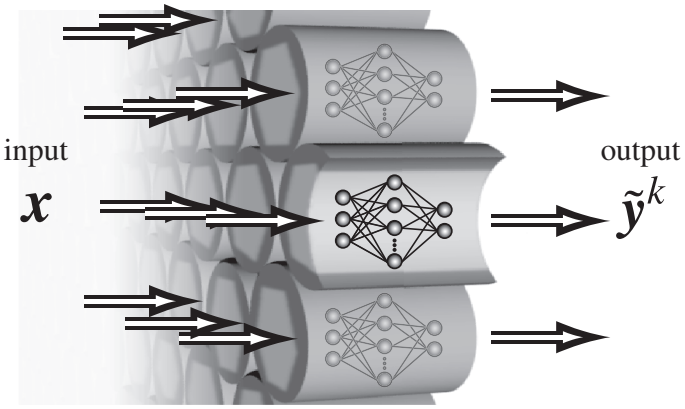



図 2.7　MLP-mnSOM のアーキテクチャ.

RNN-mnSOM（あるいは MLP-mnSOM）を用いた研究では，気象ダイナミクスや，物理系ダイナミクスといったダイナミカルシステム群の自己組織化マップを生成できることが示されている [14, 53, 41, 42]．ダイナミカルシステム群の自己組織化マップの生成は本研究に最も深く関係しており，MLP-mnSOM や RNN-mnSOM を制御系へ応用したものが本研究であると位置づけられる．また現在では多少形は変わっているもの mnSOM を原点として，機能モジュールを SOM や NG に置き換えた構造を持つ SOM×SOM（$\mathrm{SOM}^2$ とも記される）や NG×SOM が提案されている．これらの手法を用いて物体の形状認識課題への応用 [12] や手書き文字認識への応用 [13] ，顔画像分類への応用 [15] がなされている．

以上のように mnSOM は機能モジュールを変えることによって様々なデータタイプを扱うことができる．どの機能モジュールを用いるかはユーザの目的に依るため，データの本質を見極め目的に応じた適切な機能モジュールを選ぶことが必要である．適切な機能モジュールを選択することができれば応用の幅は今後も更に広がることであろう．本節では mnSOM の最も基本であり，かつ理解がしやすい MLP をモジュールとした mnSOM（MLP-mnSOM）を用いて mnSOM の一般的な枠組みとアーキテクチャ，アルゴリズムおよび具体例を紹介する．

### 2.3.2　mnSOM のアーキテクチャ

mnSOM のアーキテクチャを図 2.7 に示す．機能モジュールが MLP の場合，mnSOM は MLP が格子状に配置された構造を持つ.

以下で MLP-mnSOM が取り扱う問題設定について述べる．今，異なる入出力関係を持

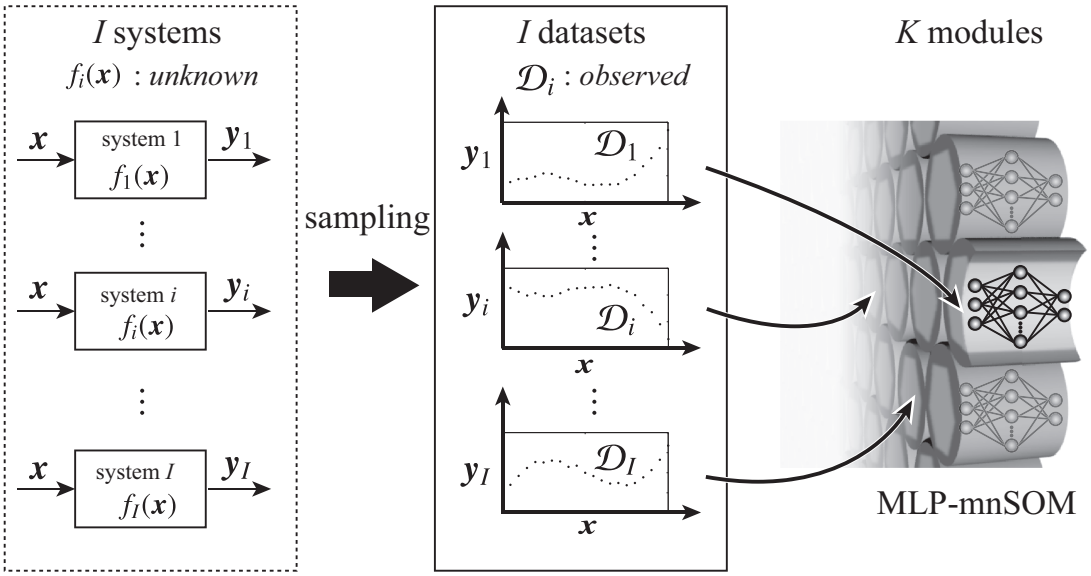



図 2.8　MLP-mnSOM の問題設定.

つシステムが $I$ 個あり，各システムから $J$ 個の入出力データが得られたとする．すなわち

$$\boldsymbol{y}_i = f_i(\boldsymbol{x}) \tag{2.29}$$
$$\mathcal{D}_i = \{(\boldsymbol{x}_{ij}, \boldsymbol{y}_{ij})\} \tag{2.30}$$
$$\boldsymbol{x}_{ij} = [x_{ij1}, ..., x_{ijd_i}]^T \tag{2.31}$$
$$\boldsymbol{y}_{ij} = [y_{ij1}, ..., y_{ijd_o}]^T. \tag{2.32}$$

ここで $f_i$ は $i$-th システムを記述する関数であり，$\mathcal{D}$ は $i$-th システムに属するデータの集合，$\boldsymbol{x}_{ij}$，$\boldsymbol{y}_{ij}$ はそれぞれ $i$-th システムの $j$-th 入力データベクトル（$d_i$ 次元）と出力データベクトル（$d_o$ 次元）である．観測できるのはシステムから得られた入出力データのみであり，$f_i$ を直接扱うことはできないものとする．ただし任意の入出力データはラベル付けがされておりどのデータがどのシステムから得られたものであるかは分かっているものとする（つまり $I$ 個のクラスが存在し，クラスごとに $J$ 個の入出力データが存在する状況を考える）.

このとき mnSOM への要求は 3 つ存在する.

1. 入出力データを元にシステムを記述している関数 $f_i$ を推定する.
2. 内挿によって中間的なシステムを生成する.
3. システムの類似度に基づき自己組織化マップを生成する.

1. は mnSOM のモジュールの中で入出力関係を最も良く近似した $I$ 個のモジュール（Best Matching Module： BMM）によって実現されることが期待される．2. は BMM

以外のモジュールが実現するシステムに期待される結果であり，SOM の内挿機能によって中間的なシステムを表現する．3. は mnSOM のモジュールはそれぞれ異なるシステムを表現しながらも似ているシステムはマップ空間上で近い位置に写像され，似ていないシステムは遠くの位置に写像されることが期待される（図 2.8 参照）．ここでシステムの類似度とは SOM が入力空間におけるユークリッド距離であったのに対し MLP-mnSOM ではシステム空間における距離,

$$L^2(f,g) = \int \|f(\boldsymbol{x}) - g(\boldsymbol{x})\|^2 p(\boldsymbol{x}) d\boldsymbol{x} \tag{2.33}$$

で定義される．ここで $p(\cdot)$ は確率密度関数である．したがって確率密度関数はシステムに依存することなく一定である必要があり，この仮定が成り立たない場合には mnSOM は正しく機能しないことがある．実課題においては確率密度関数が異なる場合が数多く存在するがその場合には mnSOM を用いる前にデータの規格化などの前処理が行われている [53]．しかしデータの規格化は各クラスが持つ重要な情報を損なう場合もあるために，いつもうまく機能するとは限らない.

また式 (2.33) において $f$ や $g$ の関数形を直接扱うことはできないために実際には出力データのアンサンブル平均に置き換えて類似度を評価する.

### 2.3.3　mnSOM のアルゴリズム

MLP-mnSOM のアルゴリズムについて説明する．mnSOM のアルゴリズムは評価過程，競合過程，協調過程，適応過程の 4 過程から構成される．評価過程では勝者モジュール（BMM）を決定するための誤差関数を評価する．競合過程では，評価過程で得られた誤差を元に BMM を決定する．協調過程では，各モジュールの学習率を算出する．最後に適応過程では，学習率を用いて MLP の結合荷重を修正する.

#### 評価過程（Evaluative process）

評価過程ではすべてのクラスとモジュールとの類似度を求めるために式 (2.33) を評価する．しかし実際には $f$ は未知であるから関数形を直接扱うことはできない．そのために式 (2.33) を出力データのアンサンブル平均に置き換えて類似度を算出する.

$$E_i^k = \frac{1}{2J} \sum_{j=1}^{J} \|\boldsymbol{y}_{ij} - \tilde{\boldsymbol{y}}_{ij}^k\|^2 \tag{2.34}$$

ここで $\tilde{\boldsymbol{y}}_{ij}^k$ は $i$-th クラスの $j$-th データに対する $k$-th モジュールの出力である.

### 競合過程（Competitive process）

競合過程ではすべてのクラスに対してそれぞれ勝者モジュール（BMM）を決める．

$$k_i^* = \arg\min_k E_i^k \tag{2.35}$$

### 協調過程（Cooperative process）

協調過程では各モジュールの学習率を求める．

$$\psi_i^k = \frac{\exp\left[-\|\xi_i^{k^*} - \xi^k\|^2/2\sigma(n)^2\right]}{\sum_{i'=1}^{I}\exp\left[-\|\xi_{i'}^{k^*} - \xi^k\|^2/2\sigma(n)^2\right]} \tag{2.36}$$

ここで $\sigma(n)$ は近傍関数の半径を表すパラメータであり初期には大きな値を持ち学習回数 $n$ の増加に伴い単調減少する．通常クラス数 $I$ よりもモジュール数が多い場合には近傍半径をゼロまで落とさずにある程度値を残しておくとよい．そのために $\sigma$ の初期値と最終値が設定可能な関数として次式が用いられる．

$$\sigma(n) = \sigma_\infty + (\sigma_0 - \sigma_\infty)\exp\left(-\frac{n}{\tau}\right) \tag{2.37}$$

ここで $\sigma_0$ は $n = 0$ における値，$\sigma_\infty$ は $n = \infty$ 時における値である．また $\tau$ は時定数で，$\sigma$ のスケジューリングを行うための係数である．

### 適応過程（Adaptive Process）

適応過程では協調過程で得られた学習率を元に誤差逆伝搬学習（Back Propagation : BP）学習を行う．

$$\Delta \boldsymbol{w}^k = -\eta \sum_{i=1}^{I} \psi_i^k \frac{\partial E_i^k}{\partial \boldsymbol{w}^k} \tag{2.38}$$

ここで $\boldsymbol{w}^k$ は $k$-th モジュールの結合荷重を表す．また $k$-th モジュールに対する誤差関数 $E^k$ は次のように表される．

$$E^k = \sum_{i=1}^{I} \psi_i^k E_i^k \tag{2.39}$$

すなわち各モジュールは式 (2.39) を最小化するように結合荷重の修正が行われる．仮に学習データが十分に存在し，式 (2.34) が近似的に式 (2.33) に等しくなるならば mnSOM

| class | $a$ | $b$ | $c$ |
|---|---|---|---|
| 1 | 0.2 | 0.2 | 0.2 |
| 2 | -0.2 | -0.2 | -0.2 |
| 3 | 0.4 | 0.0 | 0.0 |
| 4 | -0.4 | 0.0 | 0.0 |
| 5 | 0.0 | 0.0 | 0.4 |
| 6 | 0.0 | 0.0 | -0.4 |

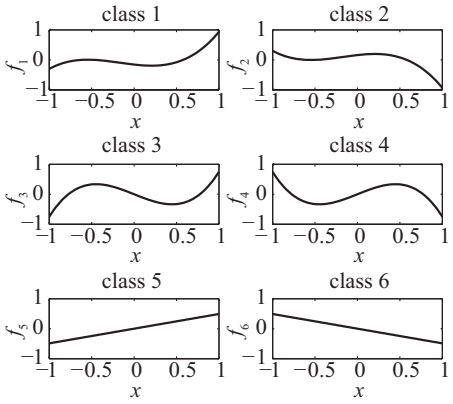



図 2.9　シミュレーションで用いた 3 次関数群.

表 2.2　MLP-mnSOM のパラメータ.

| Parameters of MLP | |
|---|---|
| Number of input unit | 1 |
| Number of hidden units | 5 |
| Number of output unit | 1 |
| Learning coefficient $\eta$ | 0.05 |
| **Parameters of mnSOM** | |
| Map size | 100 ($10 \times 10$) |
| Initial value of neighborhood radius $\sigma_0$ | 10.0 |
| Final value of neighborhood radius $\sigma_\infty$ | 2.0 |
| Time constant $\tau$ | 50 |
| Iteration number | 300 |

の各モジュールの “教師関数” は $k$-th モジュールが実現する関数を $g^k$ として次式のように各クラスのシステムの重み付き内分点で表される.

$$g^k(\boldsymbol{x}) = \sum_{i=1}^{I} \psi_i^k f_i(\boldsymbol{x}) \tag{2.40}$$

以上の 4 過程を学習が定常状態になるまで繰り返すことでシステムの違いを反映した自己組織化マップが得られる.

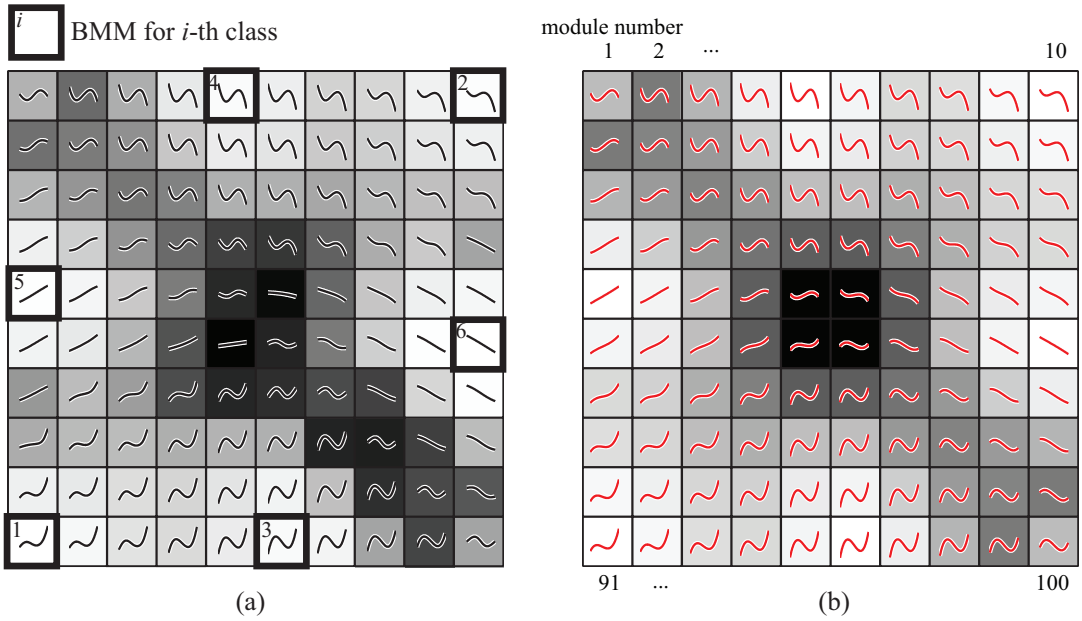



図 2.10　MLP-mnSOM による 3 次関数群のマップ．(a) 学習により得られた関数群．(b) 教師関数群．図中の濃淡は学習データに対するモジュールとその 8 近傍との出力の平均誤差を表しており，淡い色ほど近傍との性質が似ていることを意味する．

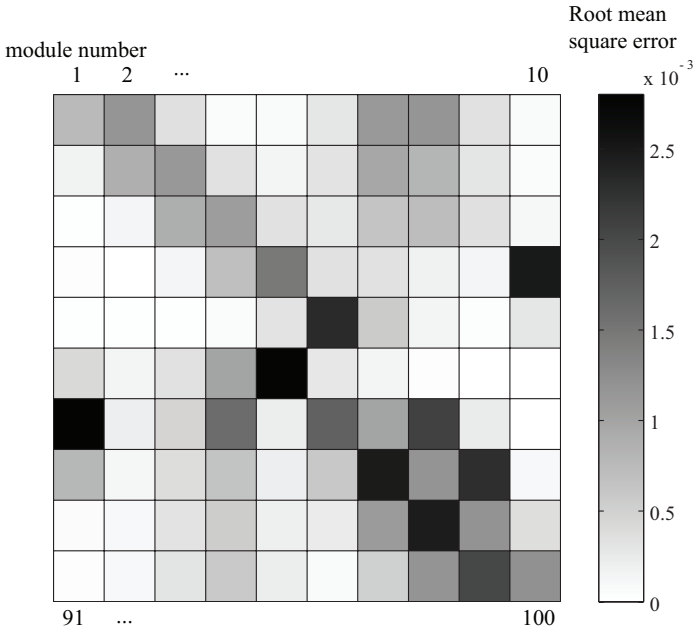



図 2.11　mnSOM の理想値との比較結果．格子は mnSOM のモジュールを表しており格子中の濃淡は平均二乗誤差の大きさを反映している．淡い色ほど誤差が小さいことを意味する．

### 2.3.4　mnSOM の例

mnSOM の具体例として 3 次関数群を用いた計算機シミュレーションを行う．3 次関数は次式で表され，各パラメータ $a, b, c$ を変えることによって異なる 6 個のクラス（$I = 6$）を実現した．シミュレーションで用いた 3 次関数群を図 2.9 に示す．

$$y_i = f_i(x) \tag{2.41}$$

$$f_i(x) = ax^3 + bx^2 + cx \tag{2.42}$$

mnSOM の学習に用いられる入出力データは入力 $x \in \{-1.0, -0.99, -0.98, ..., 0.99, 1.0\}$ に対応する出力 $y_i$ を式 (2.42) から得た．出力には観測ノイズは含まれないとした．したがって各クラスは 101 点の入出力データ（$J = 101$）から構成される．MLP-mnSOM による学習結果の例を図 2.10 に示す．またシミュレーションで用いたパラメータは表 2.2 に示す通りである．図 2.10 の (a) の格子中の波形は入力データに対する mnSOM の各モジュールの出力値をプロットしたものであり，(b) は式 (2.40) で得られたモジュールが表現すべき関数（教師関数）を表したものである．図中の濃淡は学習データに対するモジュールとその 8 近傍との平均誤差を表しており，淡い色ほど近傍と性質が似ていることを意味する．まず，クラス 1 とクラス 2，クラス 3 とクラス 4，クラス 5 とクラス 6 といったようにそれぞれ逆の位相を持つ関数に対する BMM は，関数空間における距離が遠いために，マップ空間においても対角の位置に配置された．BMM と BMM の間はそれらの関数の補間によって連続的な関数が実現された．このことから位相情報を保存したマップが生成できた．ただし，mnSOM が近似的に表現したモジュールの出力は理想値に比べて少なからず誤差を含んでいるために理想値のようにきれいな（対称な）濃淡画像は得られなかった．

次に各モジュールの出力 $\tilde{y}^k$ が教師信号 $\hat{y}^k$ をどのくらい近似できているかを評価するために，次式で表される二乗平均誤差を用いて両者の比較を行った．モジュール数が多いため数値ではなく濃淡値で二乗平均誤差を表している（図 2.11）．全モジュールの中で最も誤差が大きかったのは 61-th モジュールで，平均二乗誤差値は 0.0028 であった．このことより mnSOM の学習はおおむね成功していると言える．

$$\frac{1}{2J} \sum_{j=1}^{J} \|\hat{y}^k - \tilde{y}^k\|^2 \tag{2.43}$$

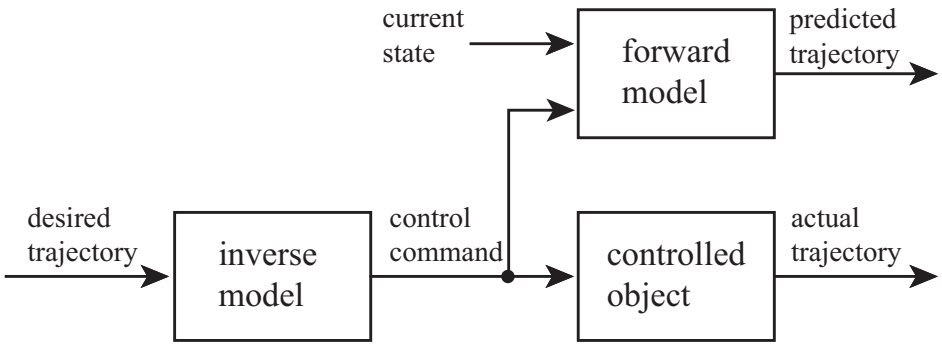



図 2.12　順モデルと逆モデル.

## 2.4　フィードバック誤差学習と多重順逆対モデル（MPFIM）

運動制御において，ヒトや生物が行えるような速くて滑らかな運動を行うためには脳内に制御対象や外界に関するモデル（内部モデル）が存在しなければならないことがこれまでの研究により分かっている [24]．また内部モデルは遺伝的に与えられるものではなく学習により獲得されることも分かっている．では内部モデルはどのように学習されているのだろうか．フィードバック誤差学習は内部モデル（特に逆モデル，後述）獲得のための教師ありの運動学習法の 1 つであり，Kawato によって提案された [23, 25]．この手法は本研究でも用いていることから特にフィードバック誤差学習に照準を絞って順を追って説明していくことにする．

まず，内部モデルと言っても制御対象と同じ入出力関係を持つ順モデル（正確には順ダイナミクスモデル）と逆の入出力関係を持つ逆モデル（正確には逆ダイナミクスモデル）がある．例えば簡単なバネ・マス・ダンパ系を考えてみる．質量 $M$ の制御対象に外部から力が加わると変位 $x$ が生じる．このとき制御対象は外部からの力を入力とし，その結果生じた変位が出力となる．このような制御対象の入出力方向と同じ入出力関係を持つモデルを順モデルと呼び，逆に変位 $x$ から力を求めるような制御対象の入出力方向と逆の入出力関係を持つモデルを逆モデルと呼ぶ．逆モデルの最も簡単な例は，制御対象を制御指令 $u(t)$ を変位 $x(t)$ に変換する汎関数（関数の関数）として $x = F(u)$ と表したとき，制御指令を $u = F^{-1}(x)$ と求める．そうすれば目標変位を $\hat{x}$ として $x = F(u) = F(F^{-1}(\hat{x})) = \hat{x}$ となり目標変位に一致する．順モデルと逆モデルの信号の流れを図示したものが図 2.12 である．

順モデルの役割は制御対象の状態を予測することであり，逆モデルの役割はフィード

フォワード制御を行うことである．したがって逆モデルの学習による獲得が速くて滑らかな運動を行うために重要であると考えられている．

ではこのような内部モデルはどのように学習が行われるのであろうか．まず，順モデルに関して，順モデルの信号の流れは制御対象と同じであるから順モデルは制御対象の状態を教師信号として通常の教師あり学習により獲得することができる．次に逆モデルに関してであるが，この場合には上記の順モデルの学習のような教師信号は存在しない．これは運動制御特有の問題であり，パターン認識の教師あり学習では生じない．この問題を解決するため直接逆モデリング，順逆モデリング，フィードバック誤差学習の 3 つがこれまでに提案されてきた．

Albus[1]，Kuperstein[28]，Miller[33]，Atkeson[4] らによって提案された方法を直接逆モデリングと呼ぶ．直接逆モデリングは単純に制御対象の入力と出力をひっくり返し，これを訓練データとして逆モデルの学習を行うというものである．直接逆モデリングは構造が単純で良い面もあるが，冗長性のある場合には対応できないし，ある特定の軌道を実現する保証がないという問題もある [24]．

Jordan と Rumelhart[21] によって提案された手法は順逆モデリングと呼ばれる．順逆モデリングは先に順モデルを獲得し，その後逆モデルの学習を行うという手法であり，冗長性のある場合にも対応ができる点で直接逆モデリングよりも優れた手法である．

Kawato[23, 25] によって提案されたフィードバック誤差学習は従来のフィードバック制御器（Conventional Feedback Controller： CFC）を用いて逆モデルを学習する手法であり，この方法も冗長性のある場合に対応できる．しかも必要となるのは CFC だけであり順逆モデリングのように順モデルの事前学習は必要なく学習と制御が同時に行える．

提案手法や後述の MPFIM にはフィードバック誤差学習が導入されているが，他の手法，特に順逆モデリングを導入することも可能である．ただし制御対象の物理特性などが変化した際にはフィードバック誤差学習では引き続き学習・適応ができるのに対し，順逆モデリングでは順モデルの再構築が必要となることを考慮するとフィードバック誤差学習の方が有効性が見込める．そこで，ここではフィードバック誤差学習に話を限り，フィードバック誤差学習について以下で詳細に述べる．

### 2.4.1　フィードバック誤差学習

以下で説明するフィードバック誤差学習は Kawato[23, 25] によって提案されたオリジナルのモデルであり，逆モデルがフィードフォワード制御器として機能する．

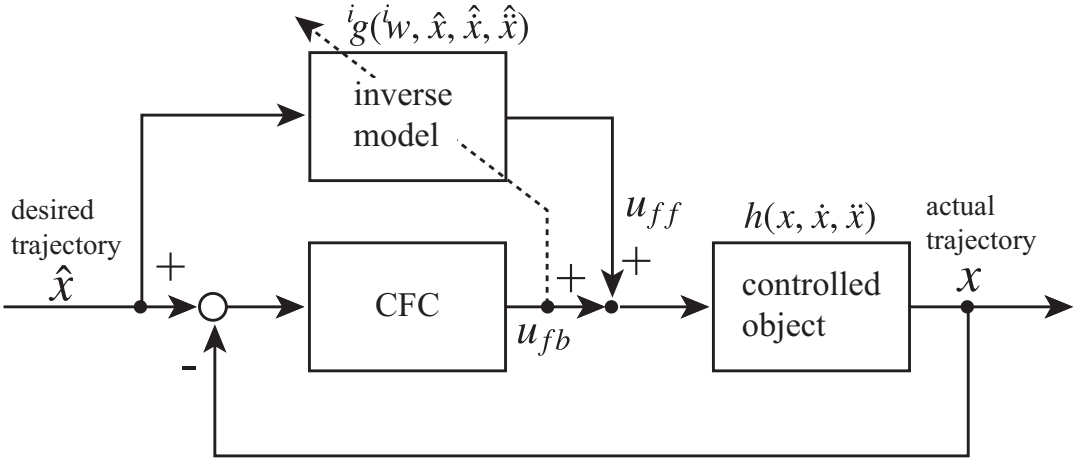



図 2.13　フィードバック誤差学習のアーキテクチャ．

フィードバック誤差学習のアーキテクチャを図 2.13 に示す．フィードバック誤差学習は CFC，逆モデル，制御対象から構成される．まず目標軌道 $\hat{x}$ が逆モデルと CFC の両者に与えられる．学習前には，制御は主に次式で与えられる CFC からの出力によって行われる．

$$u_{fb} = K_p(\hat{x} - x) + K_D(\hat{\dot{x}} - \dot{x}) + K_A(\hat{\ddot{x}} - \ddot{x}) \tag{2.44}$$

ここで $x, \dot{x}, \ddot{x}$ は実現された軌道，その速度と加速度を表し，$K_P, K_D, K_A$ はフィードバックゲインを表す．学習が進むにつれ逆モデルはフィードフォワード制御器として機能し始め，目標軌道 $\hat{x}$，その速度 $\hat{\dot{x}}$，その加速度 $\hat{\ddot{x}}$ から，そのときの結合荷重 ${}^{i}w$ に基づいて次式に従って制御指令を出力する．

$$u_{ff} = {}^{i}g({}^{i}w, \hat{x}, \hat{\dot{x}}, \hat{\ddot{x}}) \tag{2.45}$$

関数 ${}^{i}g$ はどんなニューラルネットワークが逆モデルに使われるかに依存して決まる．逆モデルとしては例えば MLP や RBF などの階層型ニューラルネットワークを用いることができる．

制御対象には CFC からの出力 $u_{fb}$ と逆モデルの出力 $u_{ff}$ の和が与えられる．

$$h(x, \dot{x}, \ddot{x}) = u_{fb} + u_{ff} \tag{2.46}$$

ただし $h$ は制御対象のダイナミクスを表す．逆モデルの学習は次式で表されるフィードバック誤差学習則により行われる．

$$dw/dt = \varepsilon(\partial u_{ff}/\partial {}^{i}w)^T u_{fb} \tag{2.47}$$

この式を Widrow-Hoff の教師あり学習則と比較すると，$u_{fb}$ が制御指令の誤差信号に相当することが分かる．制御信号の教師 $\hat{u}$ が与えられており，逆モデルの出力との二乗誤差 $\frac{1}{2}(\hat{u}-u_{ff})^T(\hat{u}-u_{ff})$ を ${}^{i}w$ の最急降下方向に減少させることを考えると，次式のような変化則が得られる．

$$dw/dt = \varepsilon(\partial u_{ff}/\partial {}^{i}w)^T(\hat{u}-u_{ff}) \tag{2.48}$$

式 (2.47) と (2.48) との比較により CFC の出力 $u_{fb}$ が $(\hat{u}-u_{ff})$ を近似する役割を果たしているのが分かる．

学習終了後に制御対象の逆モデルが獲得されると制御対象は目標軌道に等しく動く．これによって CFC のみでは不可能な素早い運動が可能になる．

なお，フィードバック誤差学習の安定性については Kawato の文献 [23] に示されている．また Miyamura[37] は適応制御の観点からフィードバック誤差学習の安定性を示している．フィードバック誤差学習の応用例として，Gomi ら [16] はフィードバック誤差学習がフィードフォワード制御器としてだけではなくフィードバック制御へも拡張できることを示している．またフィードバック誤差学習は産業用ロボットの制御へも応用がなされている [36]．さらに競合モジュラーネットワークへの応用もなされている [17].

### 2.4.2 多重順逆対モデル（MPFIM）

多重順逆対モデル（Multiple Paired Forward-Inverse Models : MPFIM），あるいは MOSAIC（MOdular Selection And Identification for Control）モデルは Wolpert と Kawato によって提案されたモジュラーネットワークの一種である [57]．このモデルはヒトや生物の素早い運動を行うためには内部モデルが必要であることに加え，ヒトや生物は多数の制御対象に対して同様に素早い運動を行うことができることから，脳内には複数の内部モデルが存在する，という予想の元に発案された．

Wolpert と Kawato の MPFIM は Gomi ら [17] のモジュラーネットワークと異なる．Gomi らの手法では mixture of experts を基にしており，エキスパートネットワークとして制御対象の逆モデルを用いており，逆モデルの学習はフィードバック誤差学習によりなされている．したがってモジュールの切り替えはゲーティングネットワークによって行われる．一方，Wolpert と Kawato の MPFIM ではモジュールが順モデルと逆モデルから構成されており，予測誤差の soft-max によってモジュールの棲み分けが行われる．逆モデルの学習には Gomi らと同様にフィードバック誤差学習を導入している．Haruno ら [18, 19] は同じ課題に対して Gomi らの競合モジュラーネットワークと MPFIM を用い

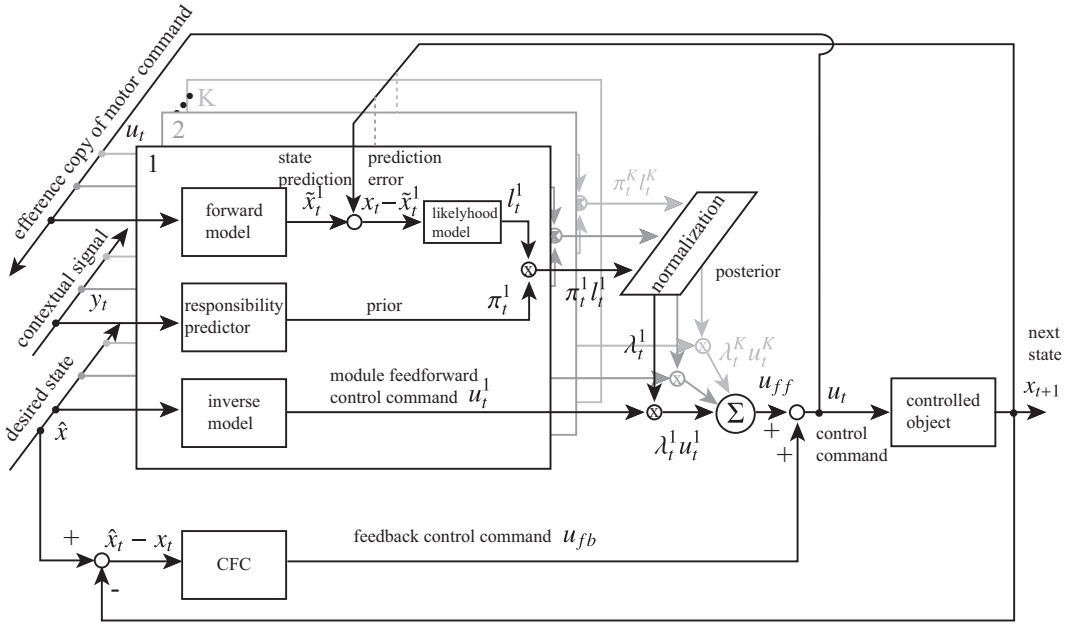



図 2.14　MPFIM のアーキテクチャ．

て実験を行ったところ MPFIM の方が学習が容易に行えたと述べている．

Doya，Samejima らは MPFIM を強化学習へと応用したモデル（Multiple Model Reinforcement Learning：MMRL）を提案している [8, 47]．

以下では Wolpert と Kawato の MPFIM のオリジナルのモデルである勾配法をベースとしたアルゴリズムについて解説する．MPFIM には隠れマルコフ（Hidden Markov Model：HMM）をベースとしたアルゴリズムもあるがこちらについては Haruno らの文献 [18] を参照されたい．

### 2.4.3　MPFIM のアーキテクチャ

MPFIM のアーキテクチャを図 2.14 に示す．MPFIM は $K$ 個のモジュールから構成され，各モジュールは 3 つの要素を持つ．最初の 2 つは順モデルと責任信号予測器（Responsibility Predictor：RP）であり，これらは責任信号を決定するために用いられる．責任信号とは各モジュールは制御量および学習量を決定する際に用いられる信号で，責任信号が大きいモジュールほどたくさんの制御信号を出力し，たくさん学習を行う権利を取得できる．順モデルは現在の状態，制御信号を入力とし次時刻における制御対象の状態を予測する．すなわち，

$$\tilde{x}_{t+1}^k = {}^f g({}^f w_t^k, x_t, u_t) \tag{2.49}$$

と表されるとする．ただし $w_t^k$ は制御対象の順ダイナミクスを近似する関数 ${}^f g$ のパラメータ（例えばニューラルネットワークの結合荷重）である．これらの予測値 $\tilde{x}_t^k$ と実際の制御対象の状態 $x_t$ とを比較することで各モジュールの尤度が算出される．尤度 $l_t^k$ は次式で与えられる．

$$l_t^k = P(x_t|{}^f w_t^k, u_t, k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-|x_t - \tilde{x}_t^k|^2/2\sigma^2} \tag{2.50}$$

ここで $\sigma$ は順モデルが含むノイズの標準偏差を表す．したがって予測誤差の小さいモジュールほど尤度が大きくなる．これら尤度を正規化するために soft-max 関数を導入する．

$$\frac{l_t^k}{\sum_{k'=1}^K l_t^{k'}} \tag{2.51}$$

これにより各モジュールの尤度は 0 から 1 の値を取るようになり，かつ尤度の合計値が 1 になる．

予測が正しかったかどうかの事後評価は制御信号を求める前に行うことができないため，MPFIM では制御対象に関する情報以外のセンサ信号（これを contextual signal と呼んでいる）を用いて事前評価を行っている．これが責任信号予測器（RP）である．入力を $y_t$ と表すと，RP は次式で与えられる．

$$\pi_t^k = \eta(\delta_t^k, y_t) \tag{2.52}$$

ここで $\delta_t^k$ は関数 $\eta$ のパラメータである．この事前確率は順モデルによって得られる尤度と結合することにより事後確率を生成する．

$$\lambda_t^k = \frac{\pi_t^k l_t^k}{\sum_{k'=1}^K \pi_t^{k'} l_t^{k'}} \tag{2.53}$$

この $\lambda_t^k$ を責任信号と呼ぶ．責任信号は 3 つの用途に用いられる．1 つ目は順モデルの学習に責任信号を重み付けている．2 つ目は逆モデルの出力に責任信号を重みづけている．そして 3 つ目は逆モデルの学習に責任信号を重み付けている．また責任信号は RP の教師信号としても用いられる．

MPFIM のモジュールに含まれている 3 つめの要素は目標軌道 $\hat{x}$ を追従するための制御信号を供給する逆モデルである．$k$-th 逆モデルは次式で表されるように目標軌道を入力とし制御信号を出力する．

$$u_t^k = {}^i g({}^i w_t^k, \hat{x}_t) \tag{2.54}$$

制御対象に与えられる制御信号は各逆モデルの出力を責任信号の重み付き線形和として次式で求まる．

$$u_t = \sum_{k=1}^{K} \lambda_t^k u_t^k = \sum_{k=1}^{K} \lambda_t^k \cdot {}^i g({}^i w_t^k, \hat{x}_t) \tag{2.55}$$

このとき逆モデルの学習則はフィードバック誤差学習則に責任信号を掛け合わせることで得られる．

$$\Delta {}^i w_t^k = \varepsilon \lambda_t^k \frac{d^i g^k}{d^i w_t^k} (\hat{u}_t - u_t) \simeq \varepsilon \frac{du_t^k}{dv_t^k} \lambda_t^k u_{fb} \tag{2.56}$$

また同じ責任信号を用いて順モデルの学習も行われる．

$$\Delta {}^f w_t^k = \varepsilon \lambda_t^k \frac{d^f g^k}{d^f w_t^k} (x_t - \tilde{x}_t^k) \tag{2.57}$$

### 2.4.4　MPFIM のアルゴリズムの要約

MPFIM の学習アルゴリズムを要約すると次のようになる．

1. 事前確率 $\pi_t^k$ を求める．

$$\pi_t^k = \eta(\delta_t^k, y_t) \tag{2.58}$$

2. 予測誤差に基づき尤度 $l_t^k$ を求める．

$$l_t^k = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-|x_t - \tilde{x}_t^k|^2 / 2\sigma^2} \tag{2.59}$$

3. 事後確率 $\lambda_t^k$ を求める．

$$\lambda_t^k = \frac{\pi_t^k l_t^k}{\sum_{k'=1}^{K} \pi_t^{k'} l_t^{k'}} \tag{2.60}$$

4. 制御信号 $u_t$ を求める．

$$u_t = \sum_{k=1}^{K} \lambda_t^k u_t^k \tag{2.61}$$

5. 結合荷重を修正する．

$$\Delta {}^i w_t^k = \varepsilon \frac{du_t^k}{dv_t^k} \lambda_t^k u_{fb} \tag{2.62}$$

$$\Delta {}^f w_t^k = \varepsilon \lambda_t^k \frac{d^f g^k}{dw_t^k} (x_t - \tilde{x}_t^k) \tag{2.63}$$

# 第 3 章

# 自己組織化適応制御器（SOAC）

## 3.1 はじめに

我々を取り巻く環境は時々刻々変化しており，環境に合った適切な行動を起こすことが要求される．運動制御においては，複数の制御対象が存在し，それらの物理特性に合わせた適切な制御が求められる．このような複数の制御対象や環境に対する制御アプローチとして定性的に 2 つの方法が考えられる．1 つは 1 個の複雑な制御器を用いて様々な状況に対応する方法であり，もう 1 つは数個の制御対象に対応できる単純な制御器を複数用いることによって様々な状況に対応することを目指したモジュール方式を用いる方法である．1 個の制御器による方法は回路規模が小さくなりやすくよい面もあるが，たった 1 つですべての環境に対応しなければならないため構成が複雑になる．また，すべての環境を 1 個の制御器で表現できない場合も考えられ，そのときには環境に適応するのに一時的に性能が大きく低下する．

モジュール構造を用いた最初の研究は Jacobs，Jordan らによって行われた [20]．彼らの手法は “mixture of experts” と呼ばれ，ある状況に特化したネットワーク（エキスパートネットワーク）と，どの状況にあるかを識別するネットワーク（ゲーティングネットワーク）から構成される．Jordan と Jacobs は後に mixture of experts を階層構造にした手法も提案している [21].

モジュール構造を用いる場合，どのモジュールを用いるかのモジュール選択問題，および各モジュールをどのように学習するかのモジュール学習問題の 2 つの問題が生じる．まずモジュール選択問題に関して，mixture of experts ではゲーティングネットワークによるパターン識別により対応している．その他の方法として Narendra ら [39, 40] と

Wolpert ら [57] は予測器と制御器を対にし，予測誤差を元にモジュールを選択する方法をとっている．Narendra らのモデルでは予測誤差を最小としたモジュールと対になっている制御器を用いて制御を行っており，Wolpert らのモデルでは予測誤差の soft-max 関数による “責任信号” をもとに責任信号の大きさに比例して各制御器の出力の重み付けをとっている．ここで予測器の必要性は制御が実時間で行われなければならないことから生じる．複数ある制御器の中でどれが最も適切な制御器であるかを判断するには，すべての制御器の出力を 1 つ 1 つ順に制御対象に与えてみないと分からないが，それを実時間で行うことは不可能である．したがって実際には制御が行われる前に制御器の選択が行われなければならない．そこである特定の状況を表現している予測器とその状況に適した制御器を対にしておけば，予測誤差を最小としたモジュールを用いることで実時間での制御が可能となる．

次にモジュール学習問題に関してであるが，Narendra らのモデルでは，制御器の学習は行われずに事前に設計されたモジュールを用いている．Gomi ら [17] は mixture of experts に Kawato によって提案された教師あり運動学習モデルの 1 つであるフィードバック誤差学習 [23, 24, 25] を導入した．このモデルではモジュールの学習はフィードバック誤差学習によりなされている．彼らはこのモデルを用いて物理特性の異なる制御対象の逆モデル（制御器）の獲得を試みたが，学習は可能であったものの，非常に困難であったと述べている．Wolpert らの多重順逆対モデル（Multiple Paired Forward-Inverse Models : MPFIM）では予測誤差を元に計算された “責任信号” を用いて制御信号だけでなく予測器と制御器の学習にも重み付けを行っている．Wolpert らのモデルにもフィードバック誤差学習が用いられており，逆モデルの獲得はフィードバック誤差学習則に重み付けを行うことでなされている．したがって Wolpert らのモデルは予測器と制御器が同時に学習によって獲得される特徴を持つ．

これらの従来の手法に対し，本論文では，自己組織化する適応制御器（SOAC）を提案する．SOAC は Tokunaga ら [52, 53, 54] によって提案されたモジュラーネットワーク SOM（mnSOM）を制御系へと応用したものであり，mnSOM の機能モジュールとして予測器と制御器の対を導入したものである．したがってモジュール選択問題については各モジュールに予測器を持たせていることから予測誤差を基準に担当モジュールの切り替えが可能である．またモジュール学習問題については通常の mnSOM の学習則にフィードバック誤差学習を組み合わせることで可能にしている．mnSOM を制御系へ応用した例は他にもあるが，Nishida らの手法 [41, 42] とはモジュールの構造と学習アルゴリズムが

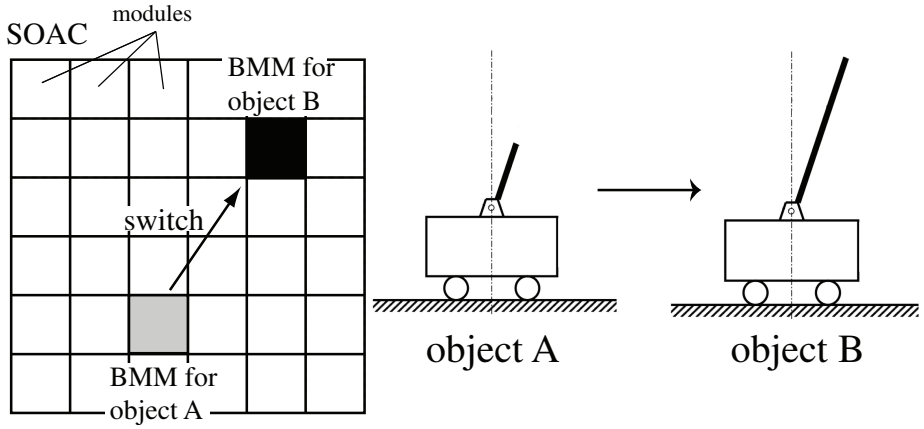



図 3.1 基本アイデア．SOAC の各モジュールの予測器は制御対象の状態を予測する．すべてのモジュールの中で予測誤差を最小としたモジュールを最適合モジュール（BMM）とする．BMM に含まれる制御器を用いて対象を制御する．制御中に制御対象の特性が変化すると，BMM も他のモジュールへ切り替わる．

異なる．

SOAC の目的は (1) 制御対象の特性の突然の変化に対応し得る制御器の構築 (2) できるだけ少ない標本数からの汎化的な制御能力の獲得 (3) 物理特性の変化の可視化である．

(1) は従来と同様の目的であり，様々な対象に対応可能な制御器の構築を目指すものである．これは mnSOM というモジュール構造を用いていることで可能である．(2) は従来のモジュール構造を持つ制御器ではあまり考慮されていなかった要素であるが，一般に獲得したい環境や制御対象の数が増えるほど必要な学習量は増加する．したがって，少ない標本から汎化的な制御能力を獲得できることは学習時間を削減できるという意味で有効である．SOAC は mnSOM を元にしていることで，少ない標本からそれらの内挿よって “中間的なシステム” を多数獲得できる．(3) は直接の目的ではないが，SOAC を用いることで結果として可能となる．これは mnSOM が有するシステムの可視化機能によって可能となる．

本章では，まず 3.2 節で SOAC の基本的な考え方について述べる．それから 3.3 節で SOAC のアーキテクチャを示す．さらに 3.4 節で予測器の一般的な学習アルゴリズムと制御器の学習アルゴリズムについて述べる．最後に制御対象のパラメータの一部を機能モジュールに含めたパラメータマップについての解説とその用途について述べる．

## 3.2 基本アイデア

SOAC は制御対象の次時刻の状態を予測する予測器と対象を制御する制御器の対を単位モジュールとし，それらが格子状に並んだ構造を持つ．SOAC の各モジュールは SOM のアルゴリズムに従って機能の分業・協調が行われ，学習により各モジュールはそれぞれ異なる環境あるいは制御対象に最適な制御系を形成する．ただし SOAC は SOM の特徴を引き継いでいるため似た特性を持つモジュールは近くに，逆に似ていないモジュールは遠くに配置される．ある制御対象が与えられたとき，制御対象を最も良く表現・制御するモジュールが最適合モジュール（Best Matching Module : BMM）として選ばれ，BMM に含まれる制御器を用いて対象が制御される．もし制御対象の特性が突然変化した場合は，それに伴い BMM も直ちに他のモジュールへと切り替わり，環境の変化に対して適応的な制御が期待される（図 3.1）．

## 3.3 アーキテクチャ

SOAC は予測器（predictor）と制御器（controller）を対とするモジュールを格子状に配置した構造を持つ（図 3.2）．$k$-th モジュールの予測器は，制御対象（controlled object）の現在の状態（current state）$\boldsymbol{x}(t)$ と制御信号（control signal）$\boldsymbol{u}(t)$ を入力とし，$\Delta t$ 秒後の制御対象の状態の予測値（predicted state）$\tilde{\boldsymbol{x}}^k(t+\Delta t)$[*1]を出力する．すなわち，

$$\tilde{\boldsymbol{x}}^k(t+\Delta t) = {}^{p}\boldsymbol{f}^k\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \tag{3.1}$$

と表されるとする．予測器はどのモジュールを使用するかの指標を与える．一方，$k$-th モジュールの制御器は，制御対象の現在の状態 $\boldsymbol{x}(t)$ と目標の状態（desired state）$\hat{\boldsymbol{x}}(t)$ を入力とし，制御信号 $\boldsymbol{u}^k(t)$ を出力する．

$$\boldsymbol{u}^k(t) = {}^{c}\boldsymbol{f}^k\left(\boldsymbol{x}(t), \hat{\boldsymbol{x}}(t)\right) \tag{3.2}$$

SOAC は，学習モードと実行モードの 2 つのモードを持つ．学習モードでは，すべての予測器と制御器[*2]を学習アルゴリズムにしたがって学習する．実行モードでは，学習の完了したモジュールを用いて実際に対象を制御する．

---

[*1] 本論文では，上付き文字は SOAC に関するインデックス（モジュール番号），下付き文字はデータに関するインデックス（クラス番号）を表す．

[*2] すべての予測器を予測器マップ（*predictor-map*），すべての制御器を制御器マップ（*controller-map*）と称する．

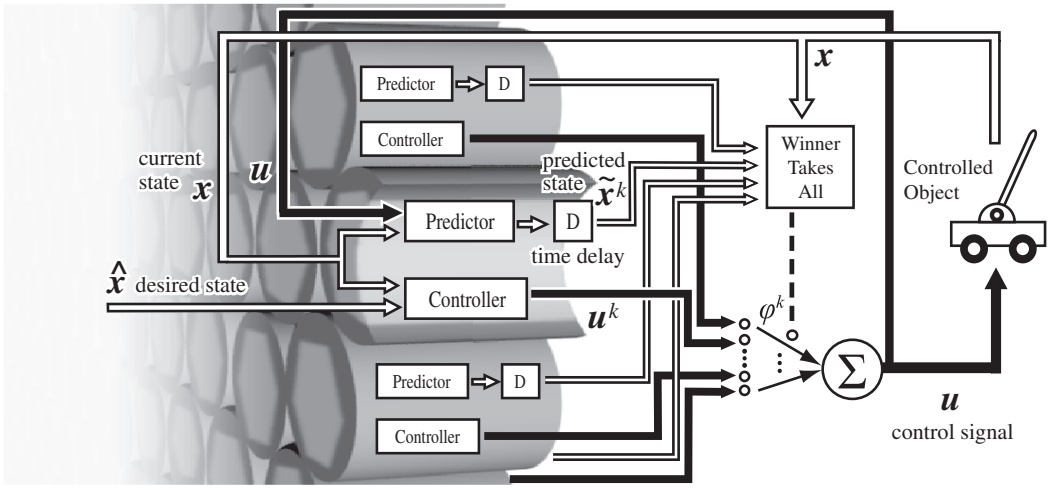



図 3.2 自己組織化適応制御器の基本構成図

## 3.4 学習アルゴリズム

### 3.4.1 学習モード（予測器）

本節では，予測器の学習について説明し，制御器の学習は次節で説明する．今，事前に $I$ 個の既知な制御対象があり，これらを学習に使用するものとする．したがって，これらを制御する制御器も $I$ 個用意する．よって $I$ 個の時系列データ $\{\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)\}(i = 1, ..., I)$ が得られる．

予測器の学習アルゴリズムは mnSOM のアルゴリズムと等しい [11, 52]．したがって，予測器のアルゴリズムは mnSOM と同様に，評価過程，競合過程，協調過程，適応過程の４つの過程からなる．ここで，予測器は MLP（Multi-Layer Perceptron）であると仮定し，重みベクトルを ${}^p\boldsymbol{w}^k$ とする．

#### 評価過程

評価過程では各予測器の出力と訓練データの誤差を $I$ 個すべてに対して求める．

$$ {}^pE_i^k = \frac{1}{T}\int_0^T \|\boldsymbol{x}_i(t) - \tilde{\boldsymbol{x}}_i^k(t)\|^2 dt \tag{3.3} $$

ここで，$\tilde{\boldsymbol{x}}_i^k(t)$ と ${}^pE_i^k$ はそれぞれ $i$-th 訓練データに対する $k$-th 予測器の出力と平均予測誤差である．また $T$ は時系列の長さを表す．

### 競合過程

予測誤差を求めた後，すべての訓練データについて最適合モジュール BMM （Best Matching Module : BMM）を決める．BMM は平均予測誤差を最小とするモジュールにより決定される．

$$*_i = \arg\min_k {}^pE_i^k \tag{3.4}$$

### 協調過程

近傍関数によって学習率 $\psi_i^k$ を決める．近傍関数をガウス関数とすると $\psi_i^k$ は次式のように表される．

$$\psi_i^k = \frac{\exp[-\|\boldsymbol{\xi}^k - \boldsymbol{\xi}_i^*\|^2/2\sigma^2]}{\sum_{i'=1}^{I}\exp[-\|\boldsymbol{\xi}^k - \boldsymbol{\xi}_{i'}^*\|^2/2\sigma^2]} \tag{3.5}$$

ここで，$\boldsymbol{\xi}^k, \boldsymbol{\xi}_i^*$ はそれぞれ $k$-th モジュールと $i$-th 訓練データに対する BMM のマップ空間における座標を表す．また $\sigma$ は近傍関数の半径を決めるパラメータで，学習回数の増加に伴い単調減少する関数が用いられる．ここでは $\sigma$ を学習回数を $n$ として次のように定義する．

$$\sigma(n) = \sigma_\infty + (\sigma_0 - \sigma_\infty)\exp\left(-\frac{n}{\tau}\right) \tag{3.6}$$

ただし $\sigma_0$ は近傍半径の初期値，$\sigma_\infty$ は近傍半径の最終値，$\tau$ は時定数である．

### 適応過程

協調過程で得られた学習率 $\psi_i^k$ を用いて予測器の結合加重を次式によって更新する．

$$\Delta^p\boldsymbol{w}^k = -{}^p\eta\sum_{i=1}^{I}\psi_i^k\frac{\partial^p E_i^k}{\partial^p\boldsymbol{w}^k} \tag{3.7}$$

ただし ${}^p\eta$ は学習係数を表す正の定数である．

これら 4 過程をネットワークが定常状態になるまで繰り返す．その結果，似た性質を持つモジュールがマップ空間上の近い位置に配置される．すなわち制御対象のダイナミクスの違いを反映した自己組織化マップが生成される．

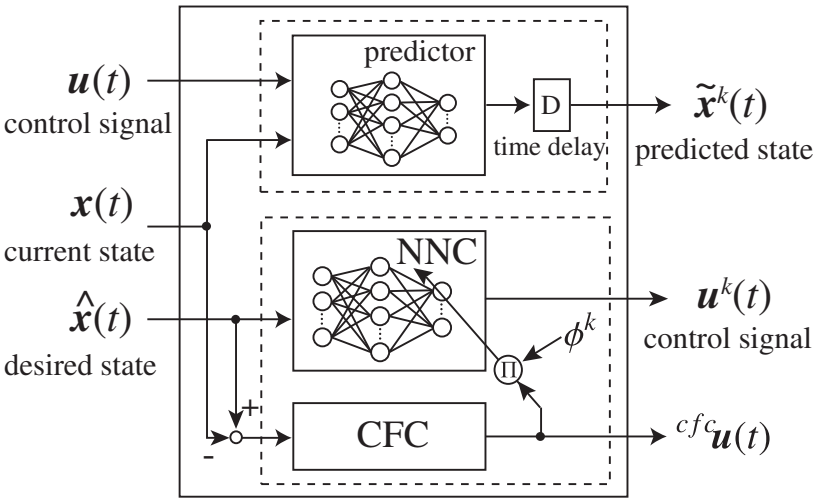



図 3.3 モジュールのブロック線図

### 3.4.2 学習モード（制御器）

SOAC の制御器として利用可能なものの 1 つとして Kawato によって提案されたフィードバック誤差学習（feedback-error-learning）がある [23, 25]．フィードバック誤差学習を用いることの利点は，従来型の線形フィードバック制御器（Conventional Feedback Controller：CFC）を用いて訓練することができ，事前に最適な制御器を決定する必要がないこと，したがって追加学習も可能になることである．ここではフィードバック誤差学習を用いた場合についてのみ説明する．

SOAC のモジュールとしてフィードバック誤差学習を導入したモデルを図 3.3 に示す．制御器は CFC とニューラルネットワーク制御器（Neural Network Controller：NNC）から構成され NNC は CFC と学習率の重み付き出力を誤差信号として教師あり学習を行う．制御器の学習は制御と同時に行われるため，学習則は次節で合わせて説明する．

### 3.4.3 実行モード

実行モードでは，まず制御対象の挙動 $\boldsymbol{x}(t)$ と各予測器が予測した挙動 $\tilde{\boldsymbol{x}}^k(t)$ との誤差を次式で定義する．

$$ {}^{p}e^k(t) = (1-\varepsilon){}^{p}e^k(t-\Delta t) + \varepsilon\|\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}^k(t)\|^2 \tag{3.8} $$

ここで ${}^{p}e^k(t)$ は指数減衰平均誤差である．すなわち実行モードでは，ごく近い過去から現在までの予測誤差の時間平均をとる．時間平均をとる区間は $0 < \varepsilon \leq 1$ で決まり，$\varepsilon$ の値

が小さいほど時間平均をとる区間は長くなる．$\varepsilon$ の大きさはモジュール切り替えの速さを決定づける．$\varepsilon$ が大きいほど速いモジュール選択が行えるが，制御対象に加わる外乱やノイズが大きい場合には $\varepsilon$ の値は小さめに設定する方がよい．

次に BMM を求める．BMM は ${}^{p}e^{k}(t)$ を最小とするモジュールであり添字を*とすれば，

$$*(t) = \arg\min_{k} {}^{p}e^{k}(t) \tag{3.9}$$

と表せる．BMM が決まったら各モジュールのいわゆる “責任信号” を次式で求める．

$$\phi^{k} = \frac{\exp[-\|\boldsymbol{\xi}^{k} - \boldsymbol{\xi}^{*}\|^{2}/2\sigma_{\infty}^{2}]}{\sum_{k'=1}^{K} \exp[-\|\boldsymbol{\xi}^{k'} - \boldsymbol{\xi}^{*}\|^{2}/2\sigma_{\infty}^{2}]} \tag{3.10}$$

ここで，$\sigma_{\infty}$ は近傍半径の最終値（学習終了時の値）を表す定数である．この式は，BMM が最も責任信号が大きく，マップ空間において BMM からの距離が遠いモジュールほど責任信号が小さくなることを意味する．

最終的に，制御対象へ与えられる制御信号は，各 NNC の出力と責任信号の積和に CFC の出力 ${}^{cfc}\boldsymbol{u}(t)$ を加えることで得られる[*3]．

$$\boldsymbol{u}(t) = \sum_{k=1}^{K} \phi^{k} \boldsymbol{u}^{k}(t) + {}^{cfc}\boldsymbol{u}(t) \tag{3.11}$$

$$\boldsymbol{u}^{k}(t) = {}^{c}f^{k}(\hat{\boldsymbol{x}}(t)) \tag{3.12}$$

$${}^{cfc}\boldsymbol{u}(t) = {}^{cfc}W(\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)). \tag{3.13}$$

ただし，${}^{cfc}W$ はフィードバック係数行列である．このとき，NNC の学習則は重みベクトルを ${}^{c}\boldsymbol{w}^{k}$ として，${}^{cfc}\boldsymbol{u}$，$\phi^{k}$ を用いて次のように表される．

$$\Delta {}^{c}\boldsymbol{w}^{k} = {}^{c}\eta \cdot \phi^{k} \frac{\partial {}^{c}f^{k}}{\partial {}^{c}\boldsymbol{w}^{k}} \cdot {}^{cfc}\boldsymbol{u} \tag{3.14}$$

## 3.5 パラメータマップ

これまで述べてきたように SOAC は予測器と制御器を対にしたモジュールを基本要素としており，予測器と制御器が自己組織的に構成される特徴を持つ．もちろん，このままでも機能するが，制御対象に関する情報（例．物理パラメータ）をモジュールの要素に追加することで後で説明する BMM のフィードフォワード選択や物理パラメータの推定などのシステム解析に利用できる．

[*3] 式 (3.11) の代わりに BMM の制御器の出力を制御入力とする方法も考えられる．

今，制御対象に関する物理パラメータ（例．長さや重さ）をベクトル $\boldsymbol{p}$ とし，$i$-th 訓練データに対するパラメータベクトルを $\boldsymbol{p}_i$ と表す．このとき各モジュールが表現するパラメータベクトル $\tilde{\boldsymbol{p}}^k$ は学習モードで最終的に得られた学習率 $\psi_i^k$ と $\boldsymbol{p}_i$ を用いて次式で定義される．

$$\tilde{\boldsymbol{p}}^k = \sum_{i=1}^{I} \psi_i^k \boldsymbol{p}_i \tag{3.15}$$

この $\tilde{\boldsymbol{p}}$ をパラメータマップ（*parameter-map*）と呼ぶことにする．パラメータマップは制御器マップと同様に予測誤差を元に計算された学習率を用いるために特別な学習は必要なく，予測器マップの生成に伴って自動的に生成される．

### 3.5.1 BMM のフィードフォワード選択

一般に式 (3.4) のような予測が正しかったかどうかの事後評価によって BMM を決定するフィードバックの選択法では時間遅れが生じる．そこで，パラメータマップを用いて見た目の情報から BMM を決定する，一種のフィードフォワードの選択法を導入することによってより素早く BMM の切り替えが可能となる．パラメータマップを用いたフィードフォワード選択は次式で表される．

$$*(0) = \arg\min_k \|\boldsymbol{p} - \tilde{\boldsymbol{p}}^k\|^2 \tag{3.16}$$

ただし，上式と式 (3.9) は距離尺度が異なるために厳密には同じマップにならないことに注意する必要がある．したがって初期段階での BMM はパラメータマップを用いて決定し，その後は予測誤差が最小となった真の BMM を用いて制御することが望ましい．

### 3.5.2 パラメータマップを用いたシステム解析

上述の BMM のフィードフォワード選択は制御対象に関する事前情報が得られる場合に事前情報を用いて BMM を決定することが可能であったが，パラメータマップのもう１つの使い方として制御対象のパラメータ推定が挙げられる．すなわち制御対象に関するパラメータが未知であるときに観測可能な情報から観測できないパラメータを推定する問題に適用できる．考え方は非常に単純であり，予測誤差を最小とした BMM と対になっているパラメータベクトル $\boldsymbol{p}^*$ を推定値とする．パラメータマップを用いたシステム解析については 5 章でシミュレーション結果を織り交ぜながら詳説する．

# 第 4 章

# 物理特性の異なる操作物の SOAC によるアームトラッキング

本章では SOAC の有効性を検証するため簡単なバネ・マス・ダンパ系を用いてシミュレーションを行う．具体的にはバネ係数，粘性摩擦係数が異なる複数の対象を用意しておき，それらを制御中に突然切り替える．もし SOAC が正しく機能していれば対象の変化に対して担当モジュールを適応的に切り替え，目標軌道を正しく追従するはずである．またシミュレーションを通して SOAC の内挿がもたらす汎化性と可視化機能，そして追加学習についても述べる．さらに，類似手法との性能比較として Wolpert らの多重順逆対モデル（Multiple Paired Forward-Inverse Models：MPFIM）を用いて両者の制御性能を比較した．

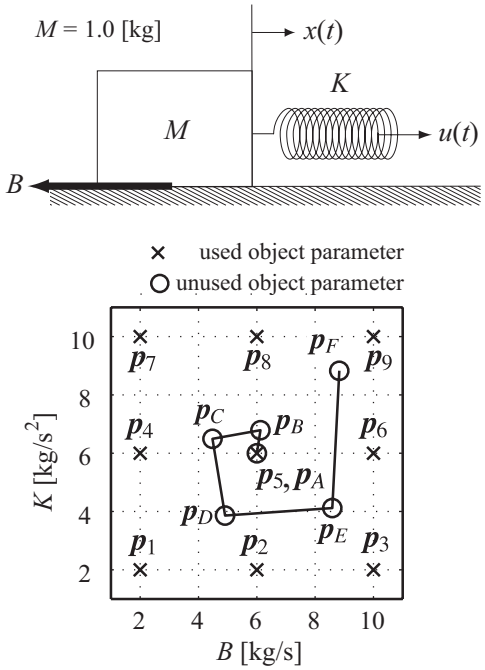
## 4.1 シミュレーションの概要

シミュレーションは以下の流れで行った．まず $I$ 個の制御対象から学習データを生成しそれらにラベルをつける．次に学習データを用いて予測器のオフライン学習を行う．予測器マップの学習が終わったら実行モードへ移行し制御器の学習をオンラインで行う．最後に制御器の学習が完了したらネットワークを固定し応答波形を調べる．

以下にシミュレーションについて詳細に述べる．

表 4.1　制御対象のパラメータ.

| | $B$ [kg/s] | $K$ [kg/s$^2$] | $M$ [kg] |
|---|---|---|---|
| $\boldsymbol{p}_1$ | 2.0 | 2.0 | 1.0 |
| $\boldsymbol{p}_2$ | 6.0 | 2.0 | 1.0 |
| $\boldsymbol{p}_3$ | 10.0 | 2.0 | 1.0 |
| $\boldsymbol{p}_4$ | 2.0 | 6.0 | 1.0 |
| $\boldsymbol{p}_5$ | 6.0 | 6.0 | 1.0 |
| $\boldsymbol{p}_6$ | 10.0 | 6.0 | 1.0 |
| $\boldsymbol{p}_7$ | 2.0 | 10.0 | 1.0 |
| $\boldsymbol{p}_8$ | 6.0 | 10.0 | 1.0 |
| $\boldsymbol{p}_9$ | 10.0 | 10.0 | 1.0 |
| $\boldsymbol{p}_A$ | 6.00 | 6.00 | 1.0 |
| $\boldsymbol{p}_B$ | 6.13 | 6.79 | 1.0 |
| $\boldsymbol{p}_C$ | 4.48 | 6.50 | 1.0 |
| $\boldsymbol{p}_D$ | 4.91 | 3.86 | 1.0 |
| $\boldsymbol{p}_E$ | 8.59 | 4.12 | 1.0 |
| $\boldsymbol{p}_F$ | 8.83 | 8.83 | 1.0 |




## 4.2　制御対象

シミュレーションで扱ったバネ・マス・ダンパ系は以下の方程式に従う.

$$M_i\ddot{x} + B_i\dot{x} + K_i x = u \tag{4.1}$$

ここで $x$ は変位 [m]，$u$ は外力 [N] であり，また $M_i, B_i, K_i$ はそれぞれ $i$-th クラスに対する質量 [kg]，粘性摩擦係数 [kg/s]，バネ定数 [kg/s$^2$] である．これらのパラメータを変えることによって異なる制御対象を表現している．各クラスに対するパラメータは表 4.1 に示す通りである．表の $\boldsymbol{p}_1$〜$\boldsymbol{p}_9$ は学習で用いたパラメータであり，その他の $\boldsymbol{p}_A$〜$\boldsymbol{p}_F$ がテスト用のパラメータである（ただし $\boldsymbol{p}_5$ と $\boldsymbol{p}_A$ は同じ値）．なおシミュレーションは 4 次の runge-kutta 法によって実現し，刻み幅 $h$=0.001 とした.

### 4.2.1　学習データの生成

予測器の学習で必要な学習データについて述べる．各予測器は現在の変位 $x$，速度 $\dot{x}$，外力 $u$ から加速度を出力するものとした（3 入力 1 出力）．学習データに与える入力はマップの正当性を保障するために，時系列データではなく $\{-0.1, -0.05, 0, 0.05, 0.1\}$ の 5

表 4.2 予測器の学習パラメータ.

| | |
|---|---|
| Number of classes | 9 |
| Map size | $9 \times 9$ (2 dimension) |
| Initial neighborhood radius $\sigma_0$ | 9.0 |
| Final neighborhood radius $\sigma_\infty$ | 1.5 |
| Time constant $\tau$ | 100 |
| Iteration number N | 1000 |
| Learning coefficient $\eta$ | 0.0001 |

種類の組み合わせで生成し，各入力ベクトルに対する加速度の値を教師信号とした．したがって学習データはクラス数 × データ数=$9 \times 5^{3}$=1125 である．

## 4.3 学習モード

### 4.3.1 予測器の構成

シミュレーションで扱う制御対象が線形モデルであるから予測器として線形ネットワークを用いた．予測器の入力は $\boldsymbol{x}_p = [x, \dot{x}, u]^T$，出力は加速度の予測値 $y = \tilde{\ddot{x}}$ とした．ここで $T$ は転置を表す．したがって $k$-th 予測器の結合荷重を ${}^p\boldsymbol{w}^k = [{}^pw_1^k, {}^pw_2^k, {}^pw_3^k]^T$ とすると予測値は次のように表される．

$$y^k = \boldsymbol{x}_p^T {}^p\boldsymbol{w}^k \tag{4.2}$$

予測器の学習パラメータを表 4.2 に示す．

### 4.3.2 結果

シミュレーション結果を図 4.1，図 4.2 に示す．図 4.2 は最終的に得られた予測器マップを示しており，図 4.1 はその過程を示している．また BMM が表現した制御対象に関するパラメータについては表 4.4 に数値でも示している．

学習の初期（$n$=0）ではランダムに生成された予測器が学習を重ねるにつれほころびがほどけ，学習 100 回目には四角形を形成した．さらに学習を繰り返すと四角形は次第に広がりを見せ最終的に学習で与えられた制御対象を BMM が表現し，それだけでなく他のモジュールが内挿によって “中間的な制御対象” を形成した（$n$=1000）.

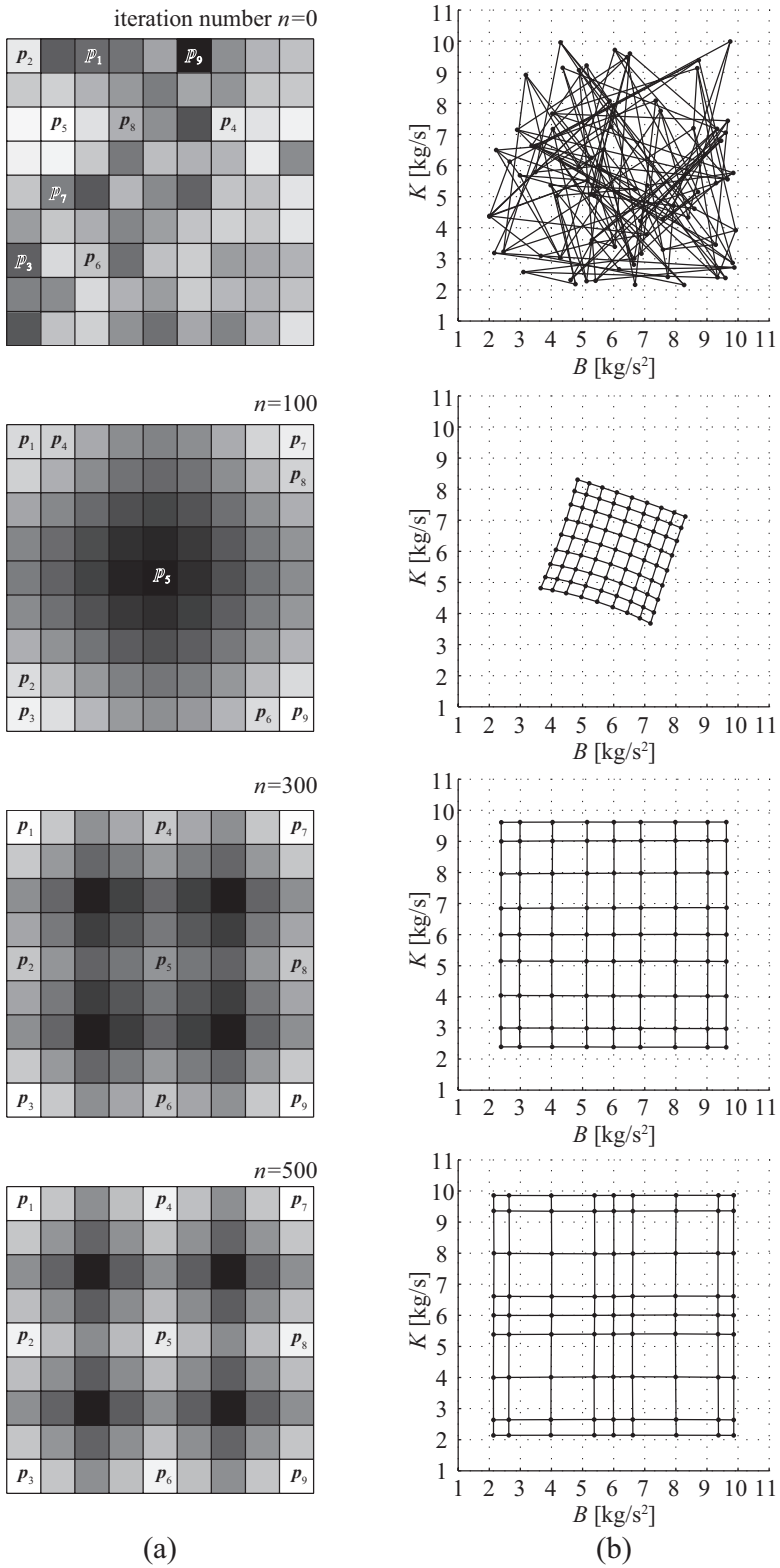



図 4.1　予測器マップの生成過程．(a) 予測器マップ．(b) 予測器の結合荷重から求めた制御対象のパラメータ推定値．上から学習回数 0，100，300，500 回のときの状態を表している．(a) の格子中の $\boldsymbol{p}_i(i = 1, ..., 9)$ は学習で使用した制御対象を表し，$\boldsymbol{p}_i$ の置かれているモジュールが $i$-th クラスに対する BMM を表している．

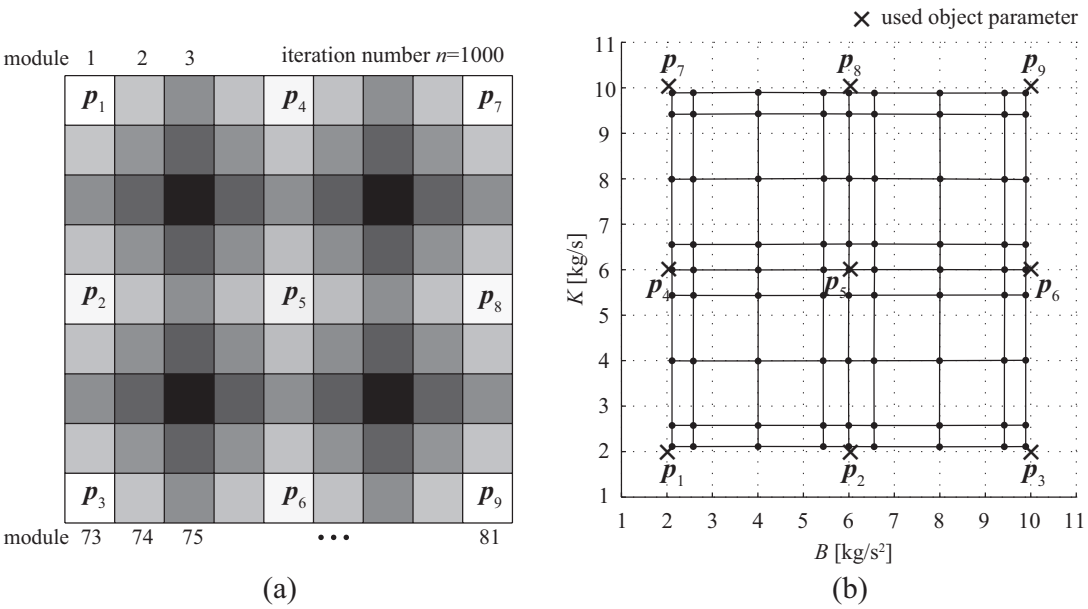



図 4.2 学習により得られた予測器マップ：(a) 予測器マップ．(b) 予測器の結合荷重から求めた制御対象のパラメータ推定値．

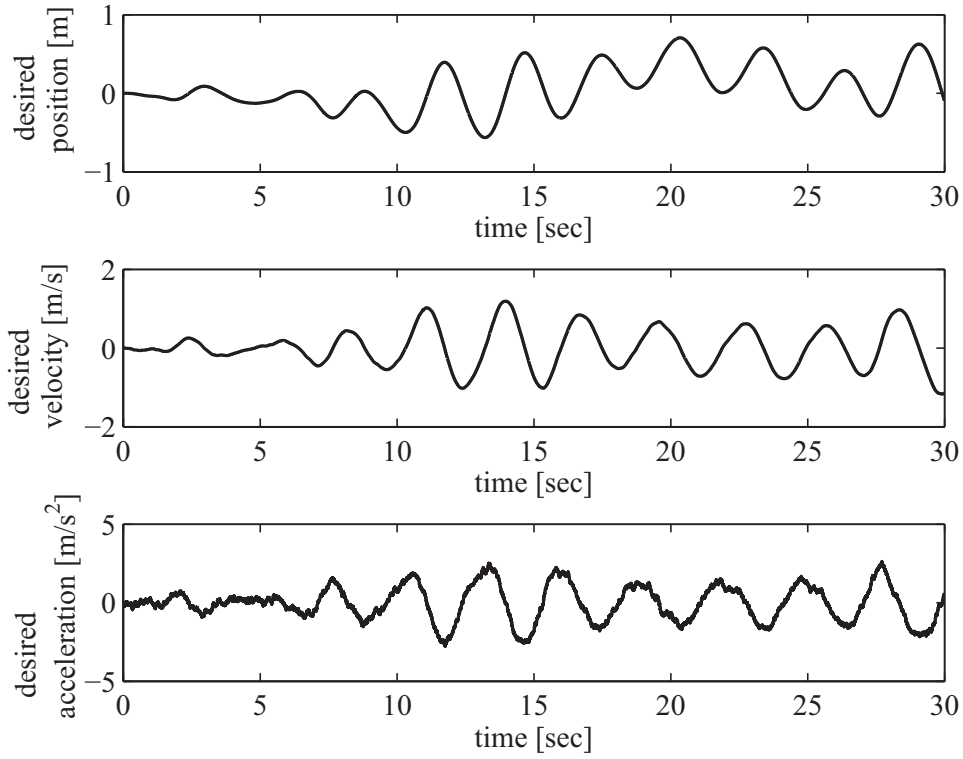



図 4.3 目標軌道の例．

表 4.3　制御器の学習パラメータ.

| | |
|---|---|
| Damping coefficient $\varepsilon$ | 1.0 |
| Neighborhood radius $\sigma_\infty$ | 1.5 |
| Learning coefficient $\eta$ | 0.01 |
| Learning time $t$ | 9000 [sec] |

## 4.4　実行モード

### 4.4.1　目標軌道の作成

目標軌道は Ornstein-Uhlenbeck 確率過程をもとに作成した．サンプリング間隔を 1 [msec] とし，30 [sec] のデータを作成した（30000 サンプル）．目標軌道の一例を図 4.3 に示す.

### 4.4.2　制御器の構成

SOAC の制御器として Kawato らのフィードバック誤差学習を用いた．制御対象が線形であることから予測器と同様に NNC も線形ネットワークとした．NNC は変位の目標値 $\hat{x}$，速度の目標値 $\hat{\dot{x}}$，加速度の目標値 $\hat{\ddot{x}}$ を入力とし，制御信号 ${}^{nnc}u^k$ を出力するものとした（3 入力 1 出力）．すなわち，

$$ {}^{nnc}u^k = \hat{\boldsymbol{x}}^T {}^c\boldsymbol{w}^k. \tag{4.3} $$

ここで $\hat{\boldsymbol{x}} = [\hat{x}, \hat{\dot{x}}, \hat{\ddot{x}}]^T$，${}^c\boldsymbol{w}^k = [{}^cw_1^k, {}^cw_2^k, {}^cw_3^k]^T$ である．

フィードバック誤差学習を行うために必要なフィードバック制御器は PDA 則を用いて ${}^{cfc}W = [k_x, k_{\dot{x}}, k_{\ddot{x}}] = [5, 10, 0.5]$ とした.

### 4.4.3　制御器のオンライン学習

学習モードで得られた予測器マップを用いて制御器の学習をオンラインで行った．学習中は $\boldsymbol{p}_1$〜$\boldsymbol{p}_9$ までの 9 種類の制御対象を 5 [sec] ごとにランダムに切り替えた．シミュレーション時間は $30 \times 300 = 9000$ [sec] とした（図 4.3 の目標軌道を繰り返し使用した）．制御器は 1 [msec] 間隔で制御信号を出力し，その間はゼロ次ホールドした．制御器の学習パラメータを表 4.3 に，学習結果を図 4.4 に示す．図 4.4 は各時刻における制御器

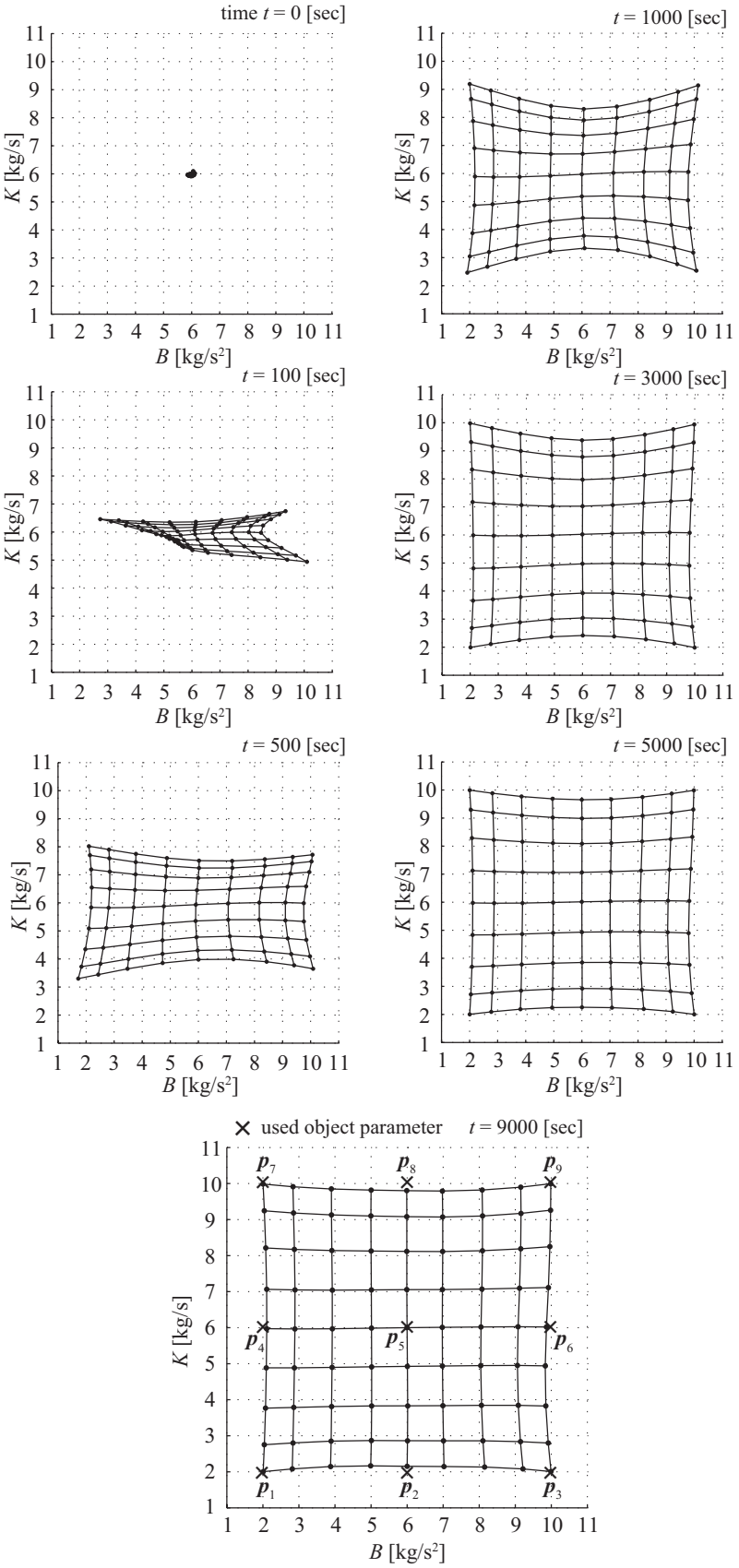



図 4.4 制御器マップの学習過程.

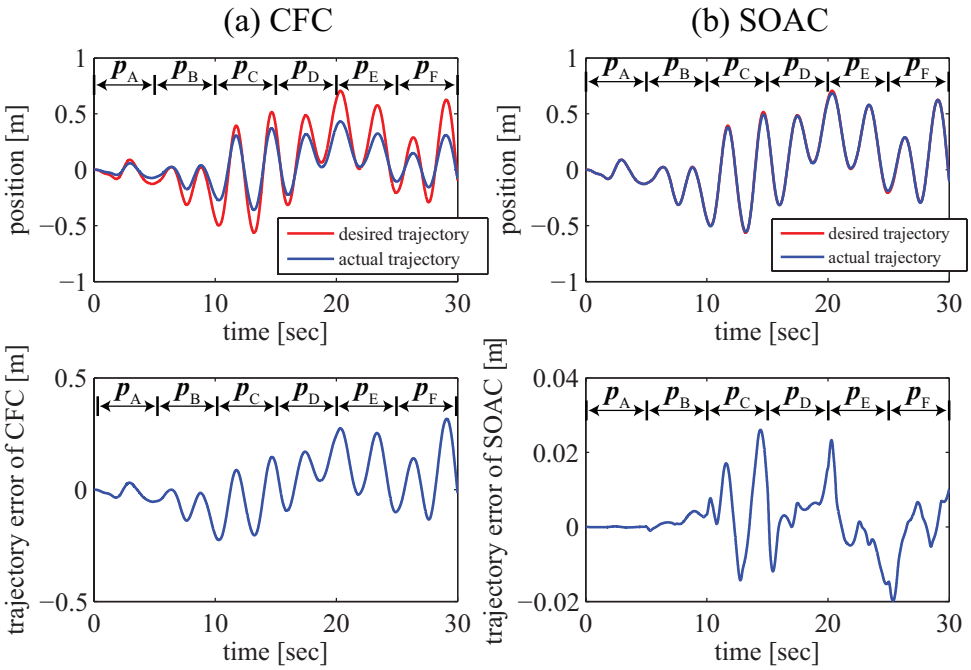



図 4.5　バネ・マス・ダンパ系の応答波形．(a) CFC の応答．(b) SOAC の応答．

表 4.4　学習により得られた制御対象に関するパラメータ．

| BMM | (for $i$-th class) | ${}^{p}B^{k}$ | ${}^{p}K^{k}$ | ${}^{p}M^{k}$ | ${}^{c}B^{k}$ | ${}^{c}K^{k}$ | ${}^{c}M^{k}$ |
|---|---|---|---|---|---|---|---|
| 1 | $(\boldsymbol{p}_1)$ | 2.1108 | 2.1118 | 1.0005 | 2.0084 | 2.0034 | 1.0002 |
| 37 | $(\boldsymbol{p}_2)$ | 5.9978 | 2.1115 | 0.9991 | 6.0000 | 2.1532 | 1.0181 |
| 73 | $(\boldsymbol{p}_3)$ | 9.8888 | 2.1112 | 1.0006 | 9.9929 | 2.0076 | 1.0109 |
| 5 | $(\boldsymbol{p}_4)$ | 2.1104 | 5.9990 | 1.0002 | 2.1069 | 5.9707 | 0.9895 |
| 41 | $(\boldsymbol{p}_5)$ | 6.0002 | 6.0017 | 1.0004 | 5.9979 | 6.0010 | 1.0011 |
| 77 | $(\boldsymbol{p}_6)$ | 9.8904 | 6.0015 | 1.0013 | 9.8641 | 6.0193 | 0.9957 |
| 9 | $(\boldsymbol{p}_7)$ | 2.1114 | 9.8882 | 0.9993 | 2.0062 | 9.9852 | 0.9931 |
| 45 | $(\boldsymbol{p}_8)$ | 5.9996 | 9.8898 | 1.0000 | 5.9896 | 9.8016 | 0.9741 |
| 81 | $(\boldsymbol{p}_9)$ | 9.8870 | 9.8891 | 0.9999 | 9.9856 | 9.9961 | 0.9922 |

マップの生成過程を示しており，図中の黒丸が各モジュールが表現した制御対象に関するパラメータを表している（正確には，各 NNC の結合荷重そのものの値を示しているわけではなく，結合荷重と学習率 $\phi^k$ との積和 $\sum_{k=1}^{K}\phi^k({}^{c}\boldsymbol{w}^k)$ で得られた値を示している）．学習の初期段階ではすべての制御器はほとんど同じ構造となっており，学習を重ねるにつれてマップが広がっていく様子が伺える．学習終了時点においては学習で与えられた制御

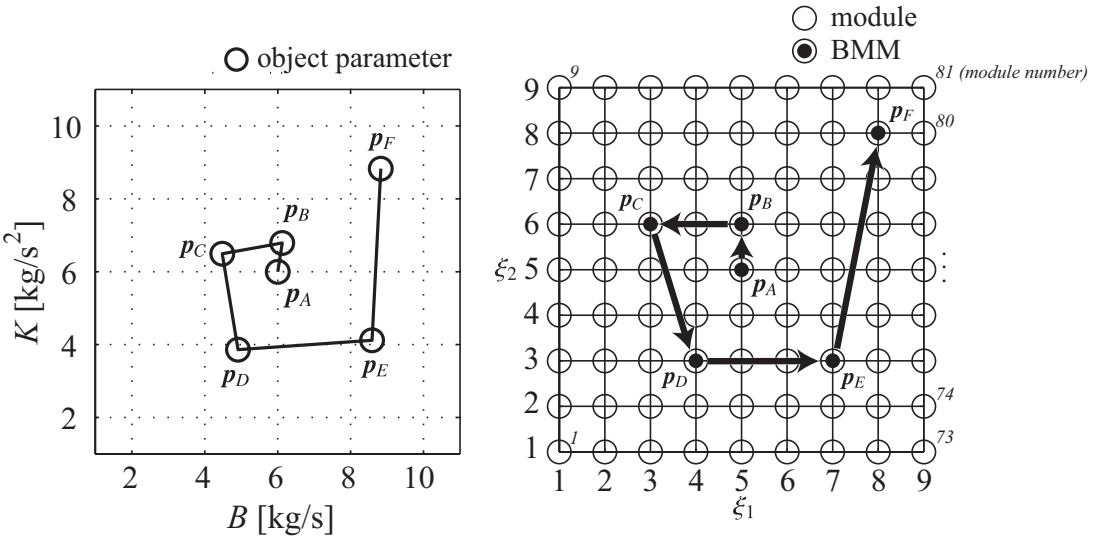



図 4.6 BMM の遷移図．与えられた制御対象（左）に対する BMM の遷移をマップ空間で表している（右）．

対象を表現しているだけでなく，学習では用いていない対象に関するパラメータも内挿によって表現できているのが分かる．

### 4.4.4 実行モードの結果

制御器の学習を行った後，ネットワークを固定し，$\boldsymbol{p}_A$〜$\boldsymbol{p}_F$ の 6 種類（このうち $\boldsymbol{p}_A$ は学習で用いた $\boldsymbol{p}_5$ に等しい）の制御対象を 5 [sec] ごとに切り替えた時のバネ・マス・ダンパ系に対する応答を調べた．ただし，BMM の瞬時的な切り替えを防ぐため $\varepsilon$=0.002 とした．結果を図 4.5 に示す．図の (a) は CFC のみを用いた通常のフィードバック系に対する応答を示しており，目標軌道を正しく追従できていないのが分かる．それに対し (b) は SOAC を用いた場合の応答を示しており，制御対象が突然変化する場合においても 30 [mm] 以下の誤差で目標を追従できた．

目標軌道を追従することはできたが，ここで次のような興味が湧く．それは SOAC は BMM をどのように切り替えているか，ということである．SOAC の特徴の 1 つである，似た特性を持つモジュールはマップ空間上の近くに配置されることを考えると，制御対象がゆっくりと変化する場合には SOAC の BMM はマップ空間上の近い位置で連続的に切り替わり，制御対象が突然大きく変化する場合は BMM は遠くのモジュールへと切り替わることが期待される．このことを確かめるため各々の制御対象に対する BMM の遷移を調べた．結果を図 4.6 に示す．この図は制御対象が提示されたときに最も BMM にな

表 4.5　MPFIM と SOAC の学習パラメータ.

| | MPFIM | SOAC |
|---|---|---|
| Number of classes | 9 | 9 |
| Damping coefficient $\varepsilon$ | 1.0 | 1.0 |
| Learning coefficient of predictor ${}^{p}\eta$ | 0.005 | 0.0001 |
| Learning coefficient of NNC ${}^{c}\eta$ | 0.005 | 0.05 |

る頻度の高かったモジュールを表示しており，期待どおりの結果を示した．すなわち似た特性を持つ $\boldsymbol{p}_A$ と $\boldsymbol{p}_B$ が順に提示されたときの BMM の変化はわずかに 1 モジュールであったのに対し，制御対象の特性の変化が大きくなるにつれそれに合わせて BMM の変化も次第に大きくなっていった．このときの BMM の遷移を線でつないだものと制御対象のパラメータをつないだものを見比べると両者は似たような形を形成し，SOAC が制御対象の特性の位相情報を保ったまま BMM を切り替えたことが分かった.

## 4.5　類似手法との比較

### 4.5.1　MPFIM について

Wolpert ら [57] によって提案された多重順逆対モデル（Multiple Paired Forward-Inverse Models）は順モデルと逆モデルを対にしたモジュールを基本要素としたモジュラーネットワークの 1 つであり，順モデル，逆モデルがそれぞれ SOAC の予測器と制御器に相当する．MPFIM では予測誤差の soft-max 関数により学習と制御信号の重み付けが行われる．すなわち予測誤差が小さいモジュールほど制御に大きく貢献するとともにたくさん学習する権利を取得できる．MPFIM の学習によって最終的に複数の環境あるいは制御対象を同定した順モデルとそれらに対応する逆モデルが生成される.

### 4.5.2　条件

MPFIM と SOAC の性能比較を行うためにバネ・マス・ダンパ系を用いてそれぞれ学習を行った．同じ条件下で比較を行うため両者の基本モジュールは同じもの（予測器は 4.3.1 節と，制御器は 4.4.2 節と同じもの）を使用した．また MPFIM における責任信号予測器（RP）と SOAC におけるパラメータマップは用いないことにした．さらにモジュー

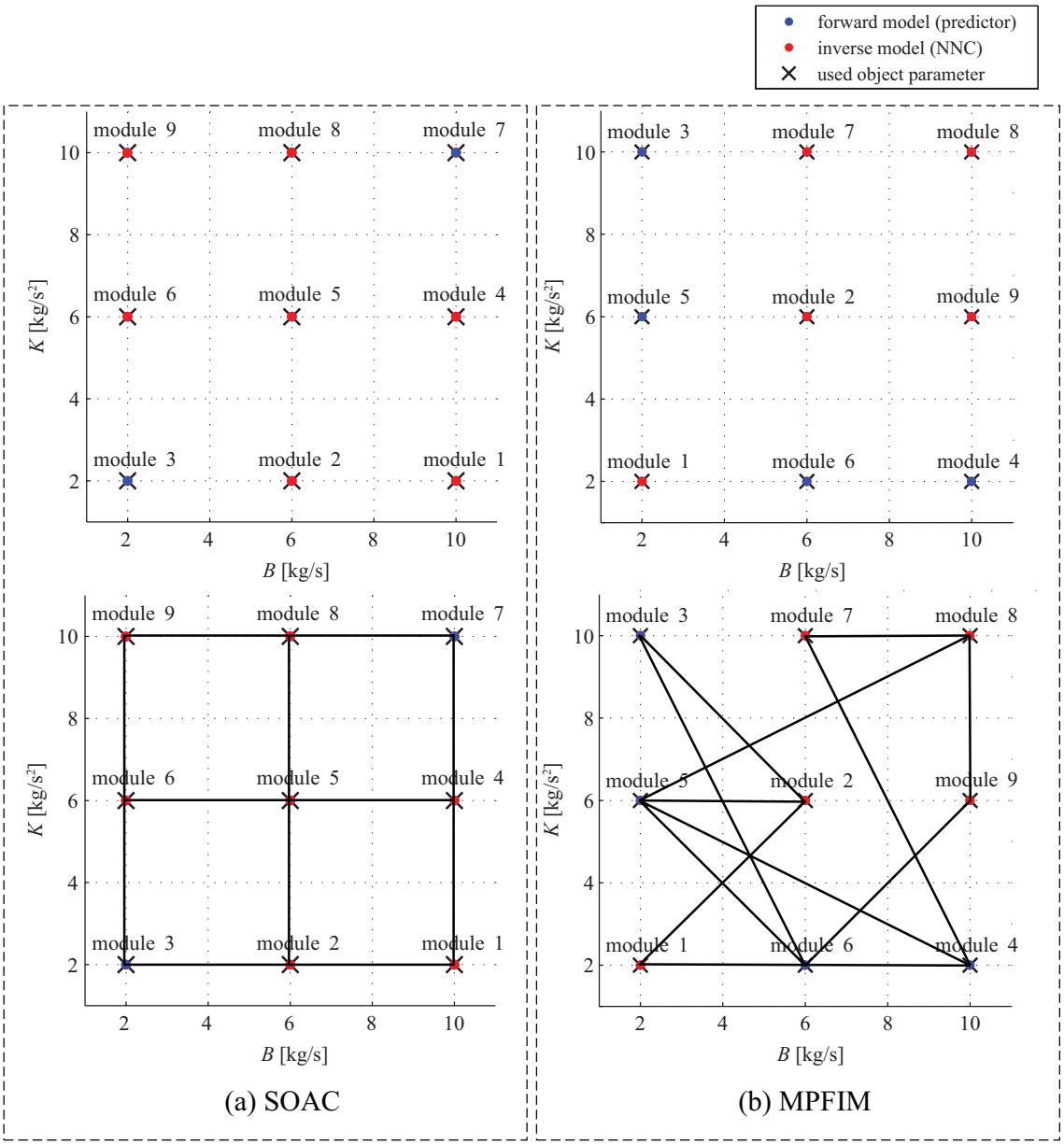



図 4.7　クラス数がモジュール数に等しい場合の SOAC と MPFIM の学習結果の例．(a) SOAC．(b) MPFIM.

ル数も同数に設定した（SOAC のマップ次元は 2 とした）．シミュレーションで用いた各学習パラメータを表 4.5 に示す．MPFIM スケーリングパラメータは学習中は手動で変化させ，制御応答を調べる時には固定した．一方 SOAC の最終近傍半径 $\sigma_\infty$ はマップのメッシュの細かさに合わせて設定した．学習時間は 9000 [sec] とした．

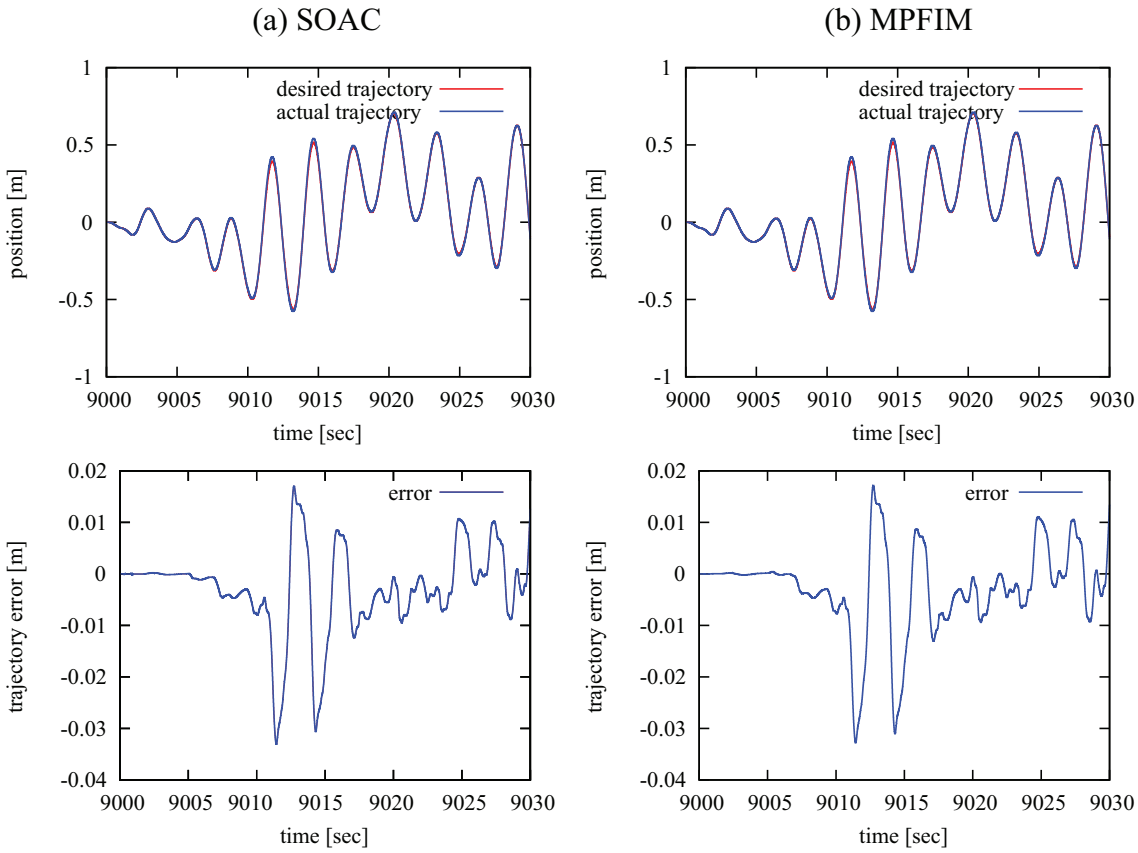



図 4.8　比較結果．(a) SOAC の応答．(b) MPFIM の応答．

### 4.5.3　クラス数とモジュール数が同じ場合の比較

クラス数とモジュール数が等しい場合，各モジュールがそれぞれの制御対象のついての順モデルと逆モデルを獲得することが期待される．この状況で SOAC と MPFIM の制御性能はどちらが優れているかを調べた．制御性能の指標として目標軌道と実軌道との平均絶対誤差（Mean Absolute Error：MAE）を用い，MAE 値が小さい方がより制御性能が高いとした．

クラス数とモジュール数が等しい場合（$K = 9$）における SOAC と MPFIM の学習結果の例を図 4.7 に，またそのときの応答波形を図 4.8 に示す．図 4.7 の結果より MPFIM の予測器と制御器が獲得した制御対象のパラメータはほぼ一致しており学習が正しく行えているのが分かる．また，どのモジュールがどの制御対象を表現しているかを調べてみたところ，SOAC は制御対象のパラメータとの位相を保ちながらモジュール生成が行われていたのに対し，MPFIM の場合，制御対象のパラメータとモジュール番号との間に規則

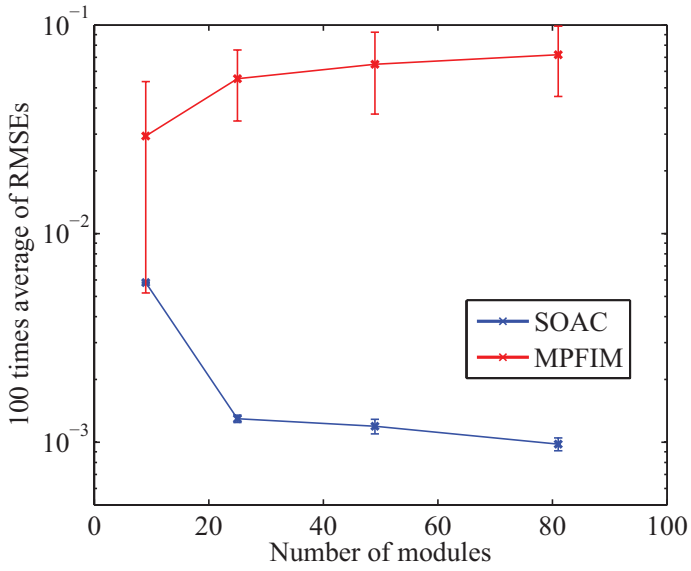



図 4.9　モジュール数の違いによる性能の変化．SOAC（青）と MPFIM（赤）それぞれの手法において，モジュール数を変化（9, 25, 49, 81）させたときの目標軌道と実軌道との平均絶対誤差（MAE）をネットワークの初期値を変え 100 回試行したときの平均と標準偏差を表している．学習時間は 9000 [sec] とした．

性は見られなかった．これは MPFIM では SOAC のような位相保存の機能を持ち合わせていないためであり，どのモジュールがどの対象を表現するかは試行の度に変化する．

図 4.8 の結果より得られた MAE は両者ともに 0.0058 であった．このことから，モジュール数がクラス数と等しい場合，学習が成功すれば SOAC と MPFIM の応答はほぼ等しくなり，同等の制御性能を有することが分かった．

上記の結果は学習成功時における比較結果であるが，MPFIM は SOAC に比べネットワークの初期値依存が大きく，実際には学習に失敗するあるいは収束までに非常に時間がかかるケースが多く見受けられた．そこでネットワークの初期状態を試行ごとに変えて 100 回学習を行ったときの MAE の平均をとったところ，SOAC が 0.0058 であったのに対し MPFIM は 0.0294 であった．したがって，学習成功時には同等の性能が得られたものの，学習の安定性の点では SOAC の方が優れていることが分かった．

### 4.5.4　モジュール数が異なる条件での比較

前節の結果からモジュール数がクラス数に等しい条件下においては両手法の制御性能は同程度であった．ではクラス数よりもモジュール数が増えた場合にはどうなるであろうか．SOAC は SOM の内挿機能を有するためモジュール数を増やすことにより中間的な

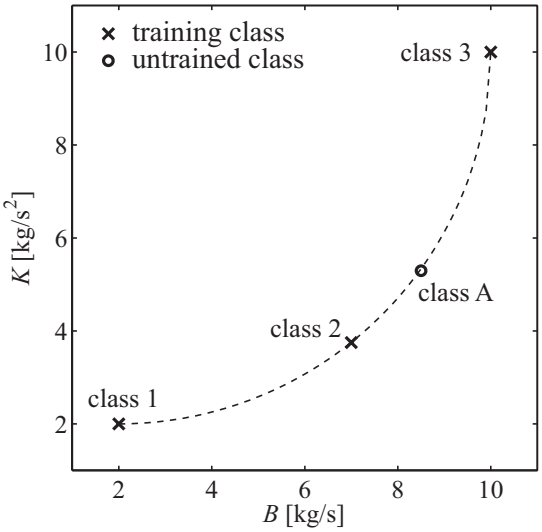



図 4.10　シミュレーションで用いた制御対象.

ダイナミクスを有するモジュールが生成されることが期待される．それに伴い未学習パラメータに対する汎化性が向上し，制御性能が上がることが予想される．また MPFIM においても，モジュール数が多い方が表現可能な能力が向上することが見込まれるためこちらもやはり制御性能が上がることが期待される．

このことを確かめるためモジュール数を 9,25,49,81 と変化させたときの制御性能の変化を調べた．制御性能はネットワークの初期状態を変え 100 回試行を行ったときの MAE の平均と標準偏差を調べることで評価した．結果を図 4.9 に示す．

SOAC に関して，モジュール数が増えるに伴い，MAE の値が下がり，制御性能が向上した．しかも標準偏差の値から分かるように学習結果にばらつきはほとんどなかった．このことから高い制御性能が安定して得られていることが分かった．一方，MPFIM はモジュール数が増えるにつれ MAE の値が上がり制御性能は逆に低下する結果となった．また MPFIM では初期値依存が大きく，試行ごとによる性能のばらつきが大きかった．

## 4.6　追加学習による適応性の向上

本節では新たに SOAC に追加学習の考えを導入する．追加学習とは実行モードにおいてもネットワークを固定せず学習を続けることであり，その必要性は SOM の内挿機能により生成された “中間的なモジュール” が，一般に “中間的な制御対象” に一致しないことから生じる．すなわち，

$$h(x;\theta_1+\theta_2) \neq h(x;\theta_1)+h(x;\theta_2). \tag{4.4}$$

表 4.6 モデル誤差.

| | SOAC | | |
|---|---|---|---|
| | $\sum_{i=1}^{3} {}^{m}E_i^*/3$ | ${}^{m}E_A^*$ | $\sum_{i=1}^{3} {}^{m}E_i^*/3 + {}^{m}E_A^*$ |
| First level | 0.3120 | 1.5766 | 1.8886 |
| Second level | 0.3120 | 1.1208 | 1.4328 |
| Third level | 0.0179 | 0.0265 | 0.0444 |
| | MPFIM | | |
| | $\sum_{i=1}^{3} {}^{m}E_i^*/3$ | ${}^{m}E_A^*$ | $\sum_{i=1}^{3} {}^{m}E_i^*/3 + {}^{m}E_A^*$ |
| First level | 0.0057 | 3.0513 | 3.0570 |
| Second level | 0.0057 | 1.1489 | 1.1546 |
| Third level | 5.5436 | 1.9253 | 7.4689 |

ここで $h$ は制御対象を記述するダイナミクス，$x$ は状態変数，$\theta$ は制御対象に関するパラメータを表す．したがって中間的なモジュールを中間的な制御対象へ適応させるためには追加学習による “微調整” が必要となる．内挿が十分ではないとは言えモジュールは解の近くに存在しているので残りの微小な誤差を修正する程度で済む．したがって無作為な状態から学習を行う場合とは状況が異なる．ここでは引き続きバネ・マス・ダンパ系を用いて追加学習について詳細に述べる．

まず SOAC の $k$-th 予測器が表現したモデル $\tilde{\boldsymbol{o}}^k$ と制御対象 $\boldsymbol{o}_i$ との間に生じるモデル誤差を定義する．モデル誤差が小さいほど制御対象に適応できていることを表す．ただしここでモデル誤差は順モデル（つまり予測器が表現したモデル）と制御対象との間に生じる誤差についてのみ議論する（逆モデルについては触れない）．バネ・マス・ダンパ系は $B, K, M$ の 3 つのパラメータによって記述されるため $\boldsymbol{o}_i = [B_i, K_i, M_i]$ としてモデル誤差（Model Error）をユークリッド距離を用いて次式のように定義する．

$$ {}^{m}E_i^k := \|\boldsymbol{o}_i - \tilde{\boldsymbol{o}}^k\| \tag{4.5}$$

次に SOAC の学習に用いるクラスとして図 4.10 に示す 3 つのクラス（1, 2, 3）と学習終了後，実行モード中に突然提示される未学習のクラス A を用意した．ここでパラメータ $M$ はクラスに依存せず一定（$M = 1$）とし，$B$ と $K$ については $(B-2)^2 + (K-10)^2 = 8^2, 2 \leq B, K \leq 10$ の関係を満たすように分布しているものとした．

シミュレーションは SOAC と MPFIM の両手法を用いて行った．モジュール数は SOAC が 5 個（マップ次元は 1），MPFIM は学習用のクラス数と同数の 3 とした．ここで SOAC に比べ MPFIM の方がモジュール数が少ないが，これは 4.5.4 節の結果を踏ま

えて設定したものであり，基本的には MPFIM ではクラス数よりも多いモジュール数は扱えないことを念頭に置いている．

図 4.11 はシミュレーション結果であり，SOAC と MPFIM の結果を 3 つの段階に分けて示してある．第 1 段階は SOAC の学習モードで得られた結果（(a)）と MPFIM の学習結果（(b)）を示している．第 2 段階は学習で得られた結果を用いて，第 1 勝者（予測誤差を最小としたモジュール）の予測誤差と第 2 勝者（予測誤差が 2 番目に小さかったモジュール）の予測誤差を元に線形内分をとったときの結果を表している（(c), (d)）．したがってこの場合には勝者モジュールをそのまま使うのではなく，既存のモジュールの組み合わせでモデルを新たに生成している．第 3 段階は追加学習を行った結果を表している（(e), (f)）．また，各段階におけるモデル誤差を表 4.6 にまとめた．

第 1 段階において，SOAC は 1-st，3-rd，5-th モジュールがそれぞれクラス 1，2，3 の勝者モジュールとなり，他のモジュールがそれらの中間的なモデルを表現した．一方，MPFIM は 1-st モジュールがクラス 1 に対する勝者モジュール，2-nd モジュールがクラス 3 に対する勝者モジュール，3-rd モジュールがクラス 2 に対する勝者となった．SOAC はバッチ SOM の性質により各クラスと対応する勝者モジュールは完全には一致していない．そのために MPFIM に比べて学習用のクラスに対してモデル誤差が大きくなっているのが表 4.6 から分かる．未学習のクラス A に対し，SOAC では内挿により得られた 4-th モジュールが勝者となり，MPFIM ではモジュールが 3 つしかないためにクラス 2 の勝者と同じく 3-rd モジュールが勝者となった．この場合，内挿による中間的なモデルを獲得できている SOAC の方が MPFIM よりも少ない誤差でモデル近似できた．

第 2 段階においても第 1 段階と同様に学習で得られたモジュールを用いるが，ここではモジュールとモジュールの間を予測誤差の soft-max による線形内分により連続的な内挿を試みている．具体的には，予測誤差を最小とした第 1 勝者（1st winner）と予測誤差が 2 番目に小さかった第 2 勝者（2nd winner）との間の内分を次式のように定めた．

$$e^* = (1-\varepsilon)e^* + \varepsilon\|\boldsymbol{x} - \tilde{\boldsymbol{x}}^*\| \tag{4.6}$$

$$e^{**} = (1-\varepsilon)e^{**} + \varepsilon\|\boldsymbol{x} - \tilde{\boldsymbol{x}}^{**}\| \tag{4.7}$$

$$\boldsymbol{o}^{ip} = \frac{e^{**}\boldsymbol{o}^* + e^*\boldsymbol{o}^{**}}{e^* + e^{**}} \tag{4.8}$$

ここで，$(\cdot)^*, (\cdot)^{**}$ はそれぞれ第 1 勝者，第 2 勝者を表す．線形内分を用いることによりモデル誤差が両手法において幾分か小さくなった．特に MPFIM の場合はモジュールが 3 つしか存在していないことから線形内分を取ることで未学習のクラス A に対するモデル誤差が第 1 段階に比べ 1/2 以下になった．しかし，線形内分を取る方法は第 1 勝者と

第 2 勝者間の直線上におけるオブジェクトについてのみ精密な近似ができるが，それ以外のオブジェクトを正確に表現することはできない（ただし第 3 勝者まで用いればその 3 点を通る三角形の内側については表現可能）．したがって本シミュレーションのようにオブジェクトが非線形に分布する場合には適応できない．このような場合において第 3 段階のような追加学習が有効となる．

第 3 段階では，学習終了後にネットワークを固定するのではなく，実行モードにおいても学習を行い続ける追加学習を導入している．言い換えると学習モードで得られたモデルを初期値として実行モードを行っている．ここで用いた追加学習の更新則は次のように表される．

$$\Delta^{p}\boldsymbol{w}^{*} = -{}^{p}\eta \frac{\partial}{\partial^{p}\boldsymbol{w}^{*}} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}^{*}\|^{2} \tag{4.9}$$

式 (4.9) は勝者モジュールの予測器の出力と制御対象の状態に誤差がある場合に結合荷重の修正を行う．したがって追加学習を行うのは勝者モジュールだけであり，他のモジュールについては結合荷重の修正は行われない．

図 4.11(e), (f) は学習で用いた 3 つのクラスと新たに未学習のオブジェクトが与えられ合計 4 つのクラスで実行モードを行ったときの結果を表している．図 4.11(e) より，SOAC においては内挿により得られた 4-th モジュールが追加学習によって学習モードでは使用されていなかったクラス A に適応しているのが分かる．またそれだけでなくバッチ SOM の性質からわずかにモデル誤差が含まれていた 1-st，3-rd，5-th モジュールについても追加学習によってより高い精度でモデル近似ができている．ところが MPFIM の場合，モジュール数が元々 3 個であった状態に 4 個の制御対象が与えられたことで学習が不安定になりモデル誤差が著しく増加した（表 4.6）．図 4.11(f) は学習終了時刻における各モジュールの状態を表している．追加学習中各モジュールは絶えず変化しており，一定値に収束することはなかった．これはモジュール数よりもクラス数が増えてしまったことでモジュール分割が正しく行えなかったことから生じたものと考えられる．このような問題は SOAC の場合においても起こりうるが，少なくともモジュール数が多くても SOAC では問題なく学習できることを考慮すると，1 つでも制御対象が増えると学習が行えなくなる MPFIM よりも SOAC の方がより多くの状況に対応できる，すなわち高い適応性を有していると言える．

## 4.7 考察

### 4.7.1 予測器マップと制御器マップの整合性

予測器と制御器は対の関係にあることから，予測器マップと制御器マップは一致することが望まれる．バネ・マス・ダンパ系であれば予測器が表現する ${}^{p}B, {}^{p}K, {}^{p}M$ がそれぞれ制御器が表現する ${}^{c}B, {}^{c}K, {}^{c}M$ に等しくなることが望まれる．しかし，シミュレーションで得られた予測器マップと制御器マップを重ねると両者は必ずしも一致していない（図 4.12）．特に内挿により表現された箇所でその違いが顕著に現れている．考えられる原因としては，まず予測器マップが SOM のバッチアルゴリズムを基本にしているのに対し制御器マップはオンラインアルゴリズムに従って形成されること．さらに，各予測器は単一でひとつの制御対象のモデルを表現しているのに対し，各制御器はすべてのモジュールの重み付き線形和でひとつのモデルを表現していることが考えられる．

この問題を解決する（すなわち予測器マップと制御器マップを一致させる）ために最も有効な手段は，予測器を “制御対象” と見なし，それぞれの “制御対象” に対してフィードバック誤差学習を行うことである．こうすることで各予測器に対して最適な制御器を学習により獲得することができる．ただし，この方法は制御対象がモジュール数（シミュレーションでは 81）存在する状況に相当するため確実な方法ではあるものの膨大な学習時間を要する．

図 4.13 は SOAC の学習モードで得られた予測器を制御対象と見なして各モジュールに対しフィードバック学習を行うことにより得られた NNC を表している．図 4.12 の結果に比べ，予測器と対応する制御器の整合性が飛躍的に改善されているのが分かる．

### 4.7.2 “中間的な制御器” とは何か？

目指すところは当然学習で与えていないパラメータを持つ制御対象に相当する順モデルと逆モデルを獲得することであるが，ここで議論したい “中間的な制御器” とは SOAC のアルゴリズムにしたがって自己組織的に生成された制御器のうち，特に BMM になれなかったモジュールは何を表現しているのか，ということである．予測器マップと制御器マップが必ずしも一致しないという事実を考慮すると，中間的な制御器とは mnSOM で獲得したい中間的なシステム（ここでは予測器）とは異なっていると言わざるを得ない．

では中間的な制御器は何を実現しているのか．話を簡単にして考えてみることにする．

具体的にはシミュレーションで扱ったバネ・マス・ダンパ系の中でクラス 1 とクラス 2 だけを用いて中間的な制御器を獲得することを考える．ただしモジュール数は 5 個とする．

予測器マップと制御器マップが $\sigma_\infty$ の値の変化に対してどのようなマップを形成するかを調べるために，$\sigma_\infty$ を 2.0, 1.5, 1.0 と変えたときの両マップを調べた．結果を図 4.14 に示す．

予測器は $\sigma_\infty$ が大きい時には内挿が細かく行われるが制御対象を忠実には再現できず，$\sigma_\infty$ が小さくなるにつれ内挿が荒くなり制御対象を表現するといったバッチ型 SOM の動作にしたがいマップが生成されている．制御器も同じように振る舞うことを期待するわけだが，実際にはそのようにはなっていない．第 1 に（$\sigma_\infty = 1.0$ の場合には多少崩れているが）基本的には $\sigma_\infty$ の値に依らずクラス 1 とクラス 2 を忠実に表現している．第 2 に $\sigma_\infty$ を変えても内挿の細かさに変化があまり見られないあるいは内挿そのものが正しく行えてない．また図には示していないが $\sigma_\infty$ の値を 1.0 よりも小さくすると学習そのものが成功しないという結果が得られている．

近傍半径を小さくすると内挿が適切に行われないことと，クラス 1 とクラス 2 のモデル獲得には成功していることから，中間的な制御器とは近傍との重み付けで逆モデルを形成しようとした結果得られたものであり，mnSOM が獲得する中間的なシステムとは異なっていることが分かる．言い換えると，制御器は CFC の出力が誤差信号として与えられるため，学習で与えられた制御対象（クラス 1 とクラス 2）に対する逆モデルさえ表現できれば誤差は生じなくなる．ここで BMM のみを用いた制御則であれば BMM が対応する制御対象に関する逆モデルを獲得し，他のモジュールは何を表現しても誤差関数には全く影響を与えない．しかし近傍との重み付けによる制御則を用いていることにより，クラス 1 に対する BMM の近傍モジュールはクラス 1 の影響を大きく受けつつもクラス 2 の影響も多少受ける，といったようにして中間的な逆モデル（制御器）を生成したものと考えられる．

### 4.7.3 MPFIM との比較

SOAC と MPFIM との本質的な違いは近傍関数の定義の仕方であり，SOAC が SOM の近傍関数に従っているのに対し，MPFIM はエントロピー最大化 [46] の考えに基づいている．そのために SOAC ではデータを補間するような “中間的なシステム” を得ることができるが，MPFIM ではそのような機能は備わっていない．これはエントロピー最大化に基づく手法がデータの中心（クラスタの中心）を表現するようなアルゴリズムになって

いるためであり，データが存在していない領域を表現することは基本的にできないためである．

また学習アルゴリズムの違いとして，MPFIM では予測器と制御器の学習は同時に行われるのに対し，SOAC では予測器の学習を行った後に制御器の学習を行っている．SOAC の学習を 2 段階に分けているのは，予測器の学習に mnSOM の学習アルゴリズムの基本となっている一括型学習則を導入していることに依る．一括型学習はオフラインで学習が行われるのに対し，フィードバック誤差学習では制御器の学習は制御と同時にオンラインで行われなければならない．このため予測器の学習後制御器の学習を行っている．ここで，予測器の学習に一括型学習を用いているのは，より安定した学習結果を得るためであり，逐次型学習が行えないことを意味するわけではない．mnSOM には逐次型学習則も存在する [54].

一方，MPFIM では予測器と制御器は同時に学習できる．この点は現在の SOAC のアルゴリズムでは実行不可であるため魅力的である．しかし 4.5 節で得られた結果から分かるように MPFIM は学習が難しい上に基本的にクラス数と同数のモジュールを用いた場合にしかうまく機能しないという問題がある．これらは同時学習以上に重要な問題であるから同時学習ができなくても学習が安定している上にモジュール数が多い場合でも学習可能である SOAC の方が有効であると考えられる．ちなみに予測器の学習後に制御器の学習を行っていることが両者間の性能の優劣を決しているわけではなく，本質的な違いは前述した近傍関数にある．なぜなら MPFIM の学習を予測器が学習した後に制御器を学習するという 2 段階学習にしても学習の収束までにかかる時間が多少短くなること以外に性能的な向上はほとんど生じない．図 4.15 は 9 個のクラス（本章で扱った制御対象と同じもの），25 個のモジュールを用いて SOAC，MPFIM 両手法において 2 段階学習を行ったときの結果の一例である．まず SOAC の結果を見ると，予測器，制御器ともに学習用のオブジェクトだけでなくそれらを内挿した中間的なオブジェクトを生成しているのが分かる．この結果は初期値を変えて試行を重ねても安定して得られる．次に MPFIM の結果を見てみる．図 4.15(c) は予測器が表現しているオブジェクトの遷移を表しており，無作為に配置された予測器が学習とともにある値へと収束している様子が伺える．MPFIM の場合予測器の学習結果は，ある値へと収束はするものの SOAC のようにいつも同じ結果（ただしどのオブジェクトをどのモジュールが表現するかは試行によって変わる）となるわけではなく，学習用のオブジェクトに相当するモジュール以外はどのオブジェクトを表現するかは試行の度に異なった（ただし学習用のオブジェクトをつないで形成される長

方形の内側のどこかに収束することは共通していた）．図 4.15(d) は制御器が表現しているオブジェクトの遷移を表しており，こちらは学習を重ねてもある値へ収束することはなかった．図は 9000 [sec] 学習後までの様子を示しているが，このあと学習を続けると制御器の結合荷重はどんどん増幅し，最終的には発散した．このことはクラス数よりもモジュール数が多い場合には常に起こった．この原因ははっきりとは分かっていないがクラス数とモジュール数が等しい場合には（収束までにかかる時間は試行ごとに大きく異なるものの）制御器の結合荷重は学習用のオブジェクトに相当する値に収束していたことから余分なモジュールが学習に不具合を引き起こしていることに違いない．

文献 [18, 19] で MPFIM の未学習データに対する汎化性に関する議論がなされているが，これは学習の完了したモジュールの出力の線形内分によって未学習データに対応できるという意味であり，未学習データを表現したモジュールができるわけではない．上でも述べたようにエントロピー最大化による方法よりも SOM の近傍関数の方が内挿の機能が優れているために SOAC の方が MPFIM よりも高い汎化性を持つと考えられる．これはシミュレーション結果からも伺うことができる．

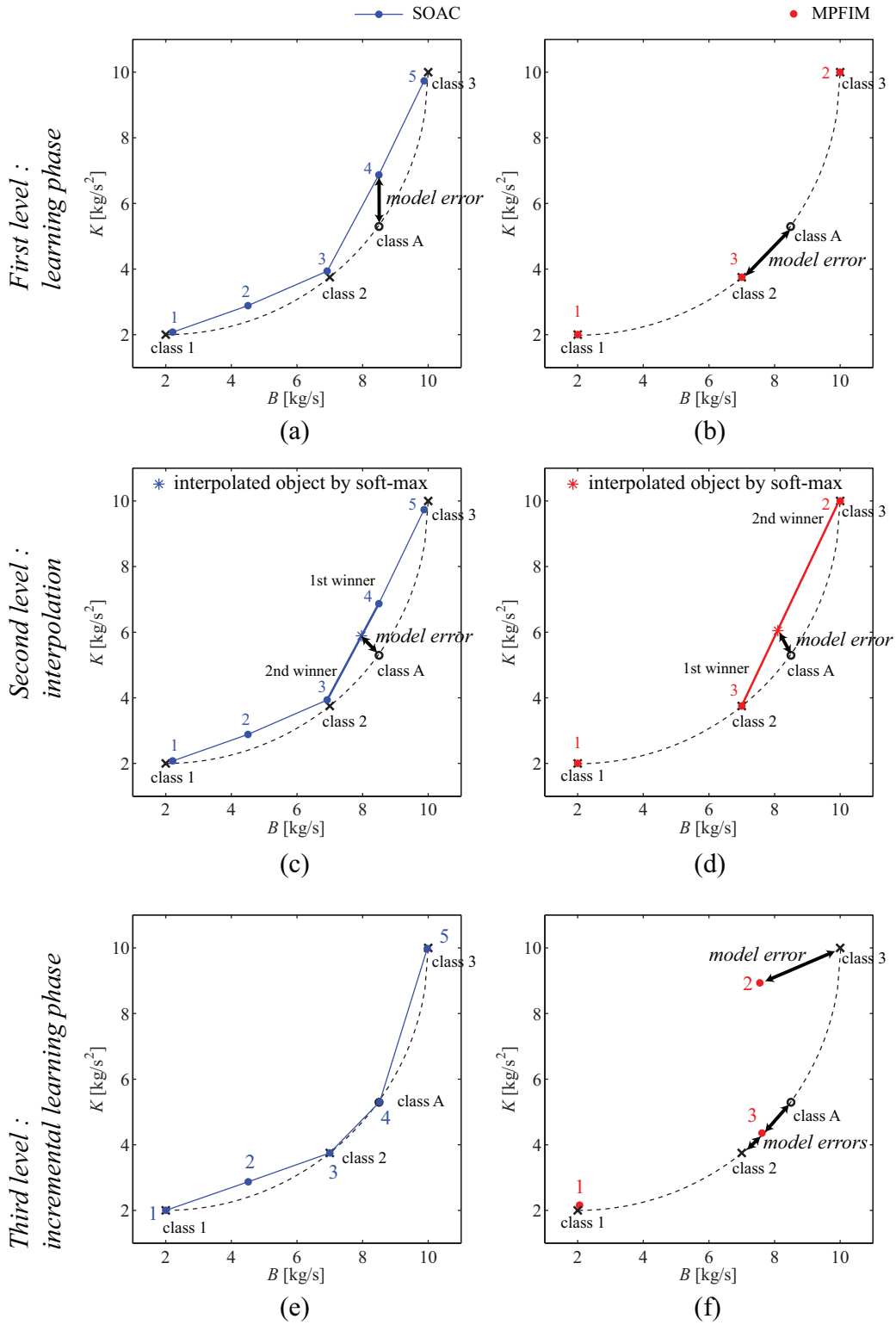



図 4.11　シミュレーション結果．SOAC と MPFIM の結果について 3 段階に分けて表示している．(a) SOAC の学習モードの結果．(b) MPFIM の学習結果．(c) 線形内分により内挿を行った結果（SOAC）．(d) 線形内分により内挿を行った結果（MPFIM）．(e) 追加学習を行った結果（SOAC）．(f) 追加学習を行った結果（MPFIM）．

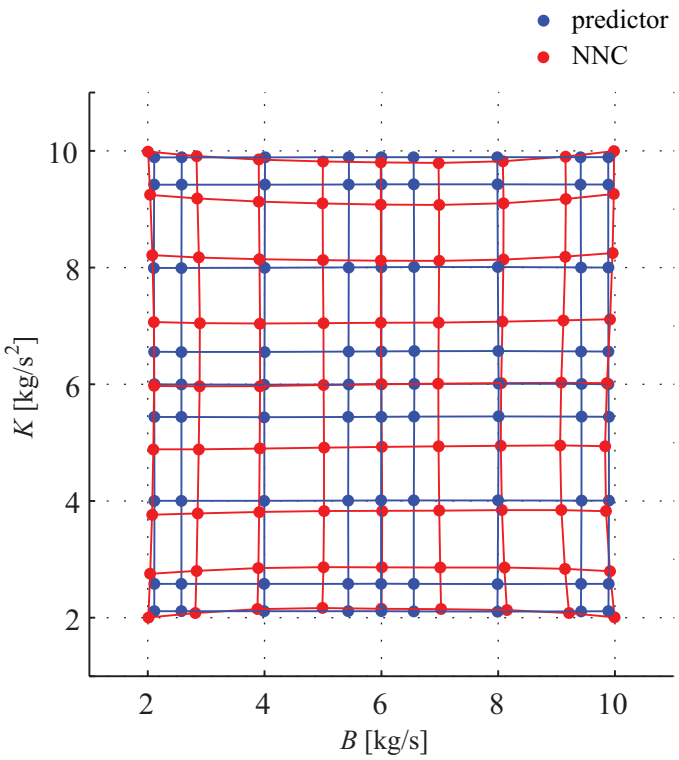



図 4.12　予測器マップと制御器マップの整合性.

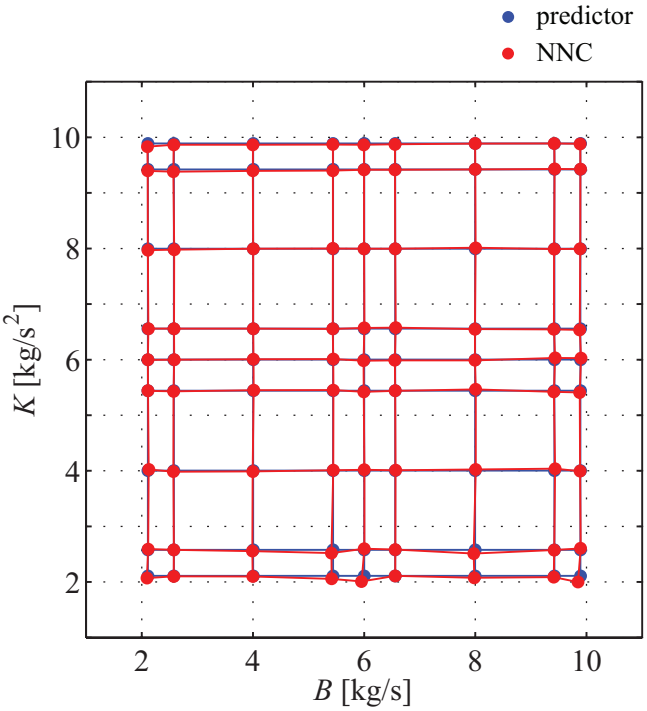
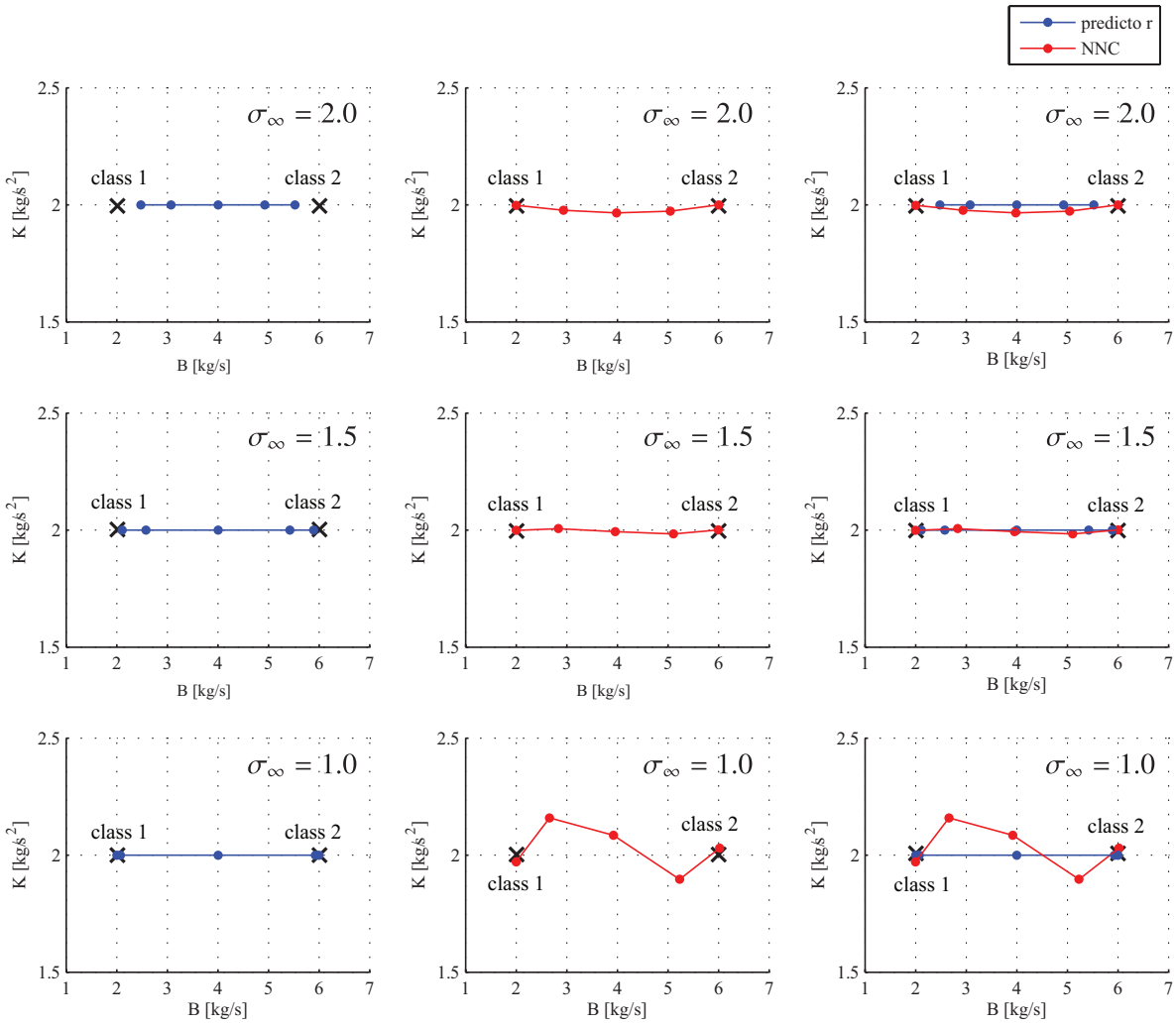



図 4.13　各予測器にフィードバック誤差学習を適用することで得られた制御器マップ.




(a) predictor　　(b) NNC　　(c) predictor and NNC

図 4.14　近傍半径の違いによる予測器と制御器の変化．(a)，(b)，(c) はそれぞれ $\sigma_\infty = \{2.0, 1.5, 1.0\}$ における (a) 予測器，(b)NNC，(c) 予測器と NNC を表している．

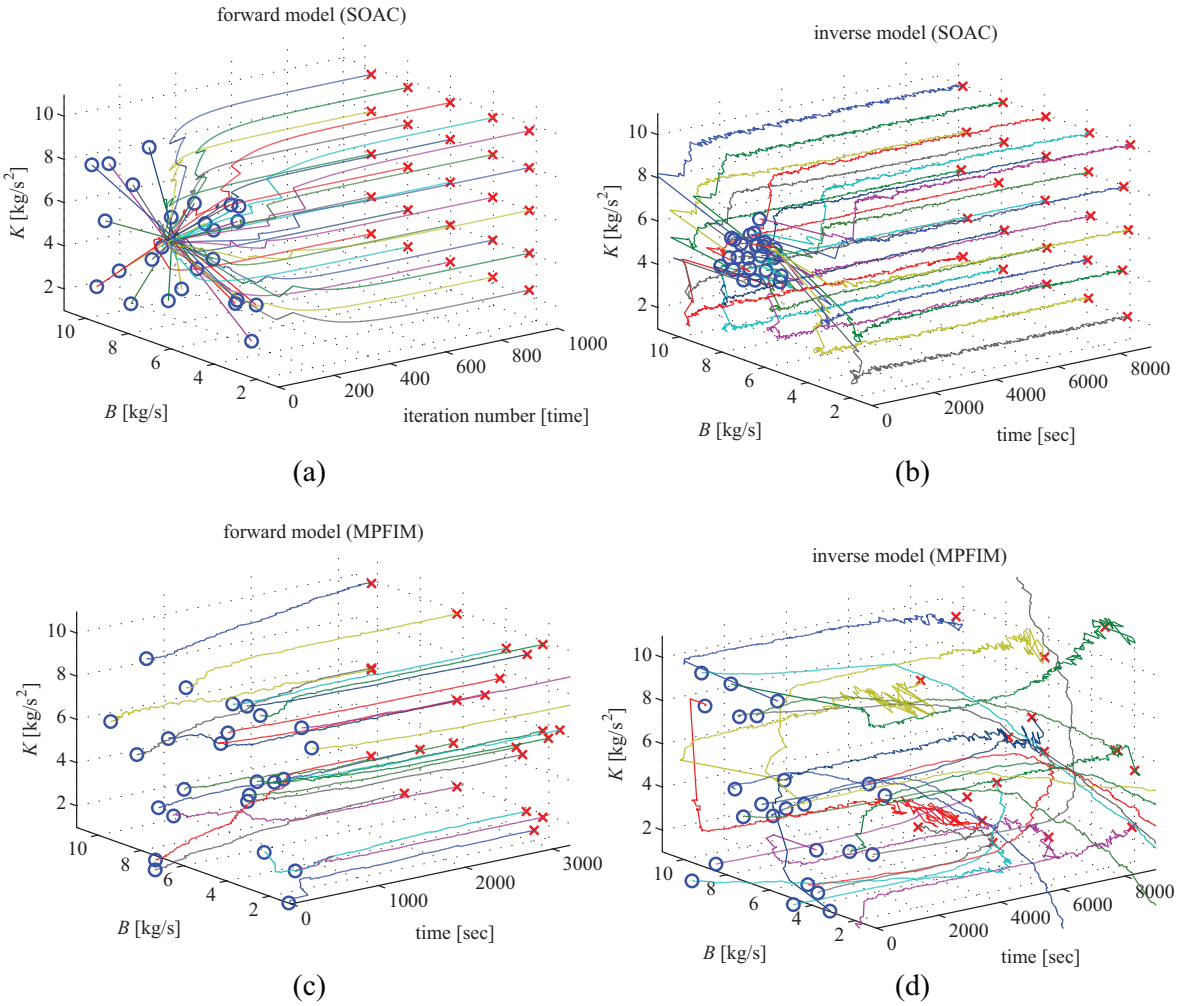



図 4.15 2 段階学習を行った場合のモジュールの生成過程．SOAC だけでなく MPFIM においても順モデルの学習後に逆モデルの学習を行った結果を表している．(a) SOAC の順モデルの生成過程．(b) MPFIM の順モデルの生成過程 (c) SOAC の逆モデルの生成過程．(d) MPFIM の逆モデルの生成過程．

# 第 5 章

# パラメータの変化する倒立振子の SOAC による制御

## 5.1 はじめに

本章では SOAC が持つ可視化機能とパラメータマップの有効性を倒立振子の例を用いて示す．パラメータマップとは SOAC の機能モジュールの要素に制御対象に関する一部のパラメータ（倒立振子では振子の長さや重さの情報など）を持たせることで SOAC の新たな機能を提供するものであり，これによって例えば見た目情報を用いたより速いモジュール選択法やパラメータ推定などのシステム解析に用いることができる．

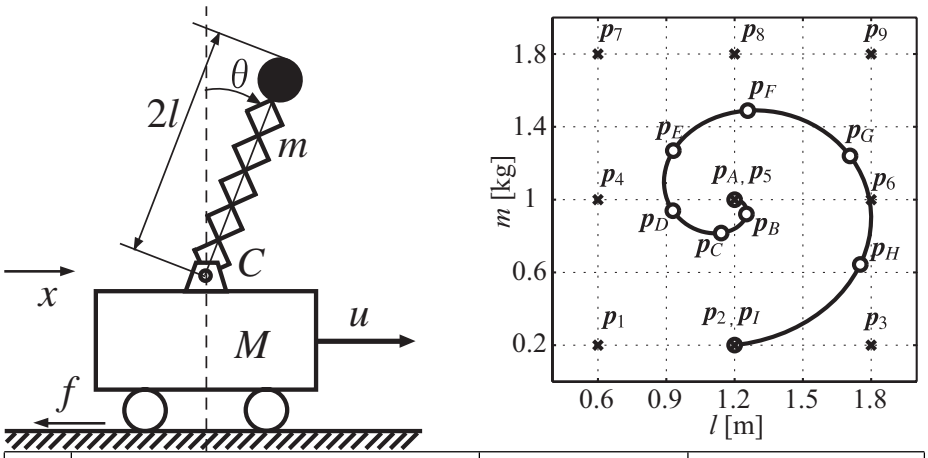
## 5.2 制御対象

本章で扱う倒立振子は 1 入力 2 出力のもので台車の移動によって振子の倒立状態を維持する．倒立振子の運動方程式は次式で表される．

$$(M+m)\ddot{x} + ml\cos\theta \cdot \ddot{\theta} - ml\dot{\theta}^2 \sin\theta + f\dot{x} = a \cdot u \tag{5.1}$$

$$ml\cos\theta \cdot \ddot{x} + (I+ml^2)\ddot{\theta} - mlg\sin\theta + C\dot{\theta} = 0 \tag{5.2}$$

ここで，$x$ を台車の位置 [m]，$\dot{x}$ を速度 [m/s]，$\ddot{x}$ を加速度 [m/s$^2$]，$\theta$ を振子の角度 [rad]，$\dot{\theta}$ を角速度 [rad/s]，$\ddot{\theta}$ を角加速度 [rad/s$^2$] とする．状態変数は $\boldsymbol{x} = [x, \theta, \dot{x}, \dot{\theta}]^T$ とし，制御変数は $u$ とした．なお，$M$ は台車の質量 [kg]，$m$ は振子の質量 [kg]，$l$ は振子の重心までの距離 [m]，$f$ は台車の粘性摩擦係数 [kg/s]，$C$ は振子の粘性摩擦係数 [kgm$^2$/s]，$g$ は重力加速度 [m/s$^2$]，$a$ は信号を力に変換するゲイン [N/V]，$I$ は慣性モーメントで，




| | | learning phase | execution phase |
|---|---|---|---|
| $\boldsymbol{p}_i$ | variable parameter<br>$\boldsymbol{p}_i = [l_i[\mathrm{m}], m_i[\mathrm{kg}]]$<br>$l_i$ : length to the mass center<br>$m_i$ : pendulum mass | $\boldsymbol{p}_1 = [0.6, 0.2]$<br>$\boldsymbol{p}_2 = [1.2, 0.2]$<br>$\boldsymbol{p}_3 = [1.8, 0.2]$<br>$\boldsymbol{p}_4 = [0.6, 1.0]$<br>$\boldsymbol{p}_5 = [1.2, 1.0]$<br>$\boldsymbol{p}_6 = [1.8, 1.0]$<br>$\boldsymbol{p}_7 = [0.6, 1.8]$<br>$\boldsymbol{p}_8 = [1.2, 1.8]$<br>$\boldsymbol{p}_9 = [1.8, 1.8]$ | $\boldsymbol{p}_A = [1.20, 1.00]$<br>$\boldsymbol{p}_B = [1.25, 0.92]$<br>$\boldsymbol{p}_C = [1.14, 1.82]$<br>$\boldsymbol{p}_D = [0.93, 0.94]$<br>$\boldsymbol{p}_E = [0.93, 1.23]$<br>$\boldsymbol{p}_F = [1.26, 1.49]$<br>$\boldsymbol{p}_G = [1.71, 1.24]$<br>$\boldsymbol{p}_H = [1.75, 0.64]$<br>$\boldsymbol{p}_I = [1.20, 0.20]$ |
| $M$ | cart mass | 5.0 [kg] | |
| $C$ | friction coefficient of pendulum | $4.0 \times 10^{-4}$ [kgm$^2$/s] | |
| $f$ | friction coefficient of cart | 10.0 [kg/s] | |
| $g$ | gravity acceleration | 9.8 [m/s$^2$] | |
| $a$ | gain | 25 [N/V] | |

図 5.1　振子の長さおよび重さが変化する倒立振子．シミュレーションで用いたパラメータを表に示す．

$I = ml^2/3$ [kgm$^2$] である．

本シミュレーションでは振子の長さ，および重さを可変パラメータとし，その他のパラメータについては固定値を用いた．したがって長さ，重さを切り替えることで異なる振子を表現している．シミュレーションで用いた倒立振子のパラメータを図 5.1 に示す．なおシミュレーションは 4 次の runge-kutta 法によって実現し，刻み幅 $h$=0.01 とした．

### 5.2.1　学習データの生成

予測器の学習で必要な学習データについて述べる．各予測器は $\boldsymbol{x}(t)$ と $u(t)$ から次時刻における状態 $\boldsymbol{x}(t + \Delta t)$ を出力するものとした（5 入力 4 出力）．ただし $\boldsymbol{x}$ は倒立振子のモデルから得られた値を用いており，線形化したモデルから取得しているわけではない．

表 5.1 予測器の学習パラメータ.

| | |
|---|---|
| Number of classes | 9 |
| Map size | $9 \times 9$ (2 dimension) |
| Initial neighborhood radius $\sigma_0$ | 10.0 |
| Final neighborhood radius $\sigma_\infty$ | 1.8 |
| Time constant $\tau$ | 100 |
| Iteration number N | 1000 |

学習データに与える入力はマップの正当性を保障するために，すなわち入力の確率分布を等しくするために時系列データではなく $\{-0.1, 0, 0.1\}$ の 3 種類の組み合わせで生成し，各入力ベクトルに対する $\Delta t$ [sec] 後の状態値を教師信号とした．したがって学習データはクラス数 $\times$ データ数=$9 \times 3^5$=2187 である．

## 5.3 学習モード

### 5.3.1 予測器の構成

倒立振子は厳密には非線形システムであるが，実際には平衡点（$\theta = 0$）付近では線形化可能であり，線形なシステムとして見なすこともできる．ここでは倒立振子の平衡点付近を予測器に学習させるものし，予測器に線形ネットワークを用いた．予測器は状態変数 $\boldsymbol{x}(t)$ と制御信号 $u(t)$ を入力とし，$\Delta t$ [sec] 後の状態 $\boldsymbol{x}(t + \Delta t)$ を出力するものとした（5 入力 4 出力）．ここで $\Delta t = 0.01$ [sec] とした．予測器の学習パラメータを表 5.1 に示す．本シミュレーションではより安定した学習を行うために，重み付き最小二乗法を用いた．したがって表には予測器の学習係数 ${}^p\eta$ が省略されている．

### 5.3.2 結果

学習により得られた予測器マップを図 5.2 に示す．図中の各格子は SOAC の予測器を表しており，格子内の波形は予測器が実現したシステムの閉ループ系に対する振子の角度のインパルス応答を示している．倒立振子は不安定系であるから SOAC が同定したシステムそのもののインパルス応答はとれないために，CFC を含んだフィードバック系のインパルス応答を示している．波形は同じ CFC に対する応答を表しており，波形の違いが

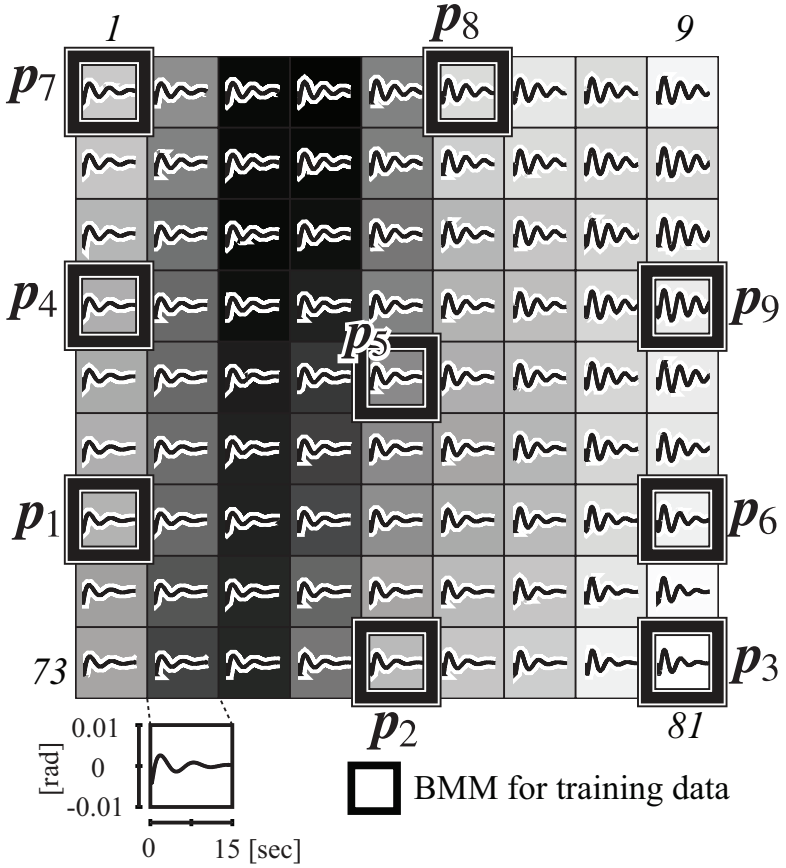



図 5.2　学習により得られた予測器マップ．図中の波形は CFC を含んだ閉ループ系に対する各モジュールのインパルス応答を表す．ここで CFC のフィードバック係数は ${}^{cf}W = [k_x, k_\theta, k_{\dot{x}}, k_{\dot{\theta}}] = [-0.25, -5.64, -0.67, -2.03]$ とした．また図中の濃淡は訓練データに対する各モジュールの出力とその 8 近傍との平均誤差を表しており，淡い色ほど近傍との特性が似ていることを意味する．

システムの違いを表現している．波形を見比べると振子振動の減衰の遅い特性を持つモジュールはマップの右上に集まり，逆に振子振動の減衰の早いシステムがマップ空間上の左下に集まるといったシステムの違いを反映したマップが形成されているのが分かる．

## 5.4　実行モード

### 5.4.1　制御器の構成

ここで扱う制御器は予測器が同定したシステムに対して最適な制御器となるようなものを用意した．具体的には線形二次レギュレータを用いて次式を最小化するようなフィード

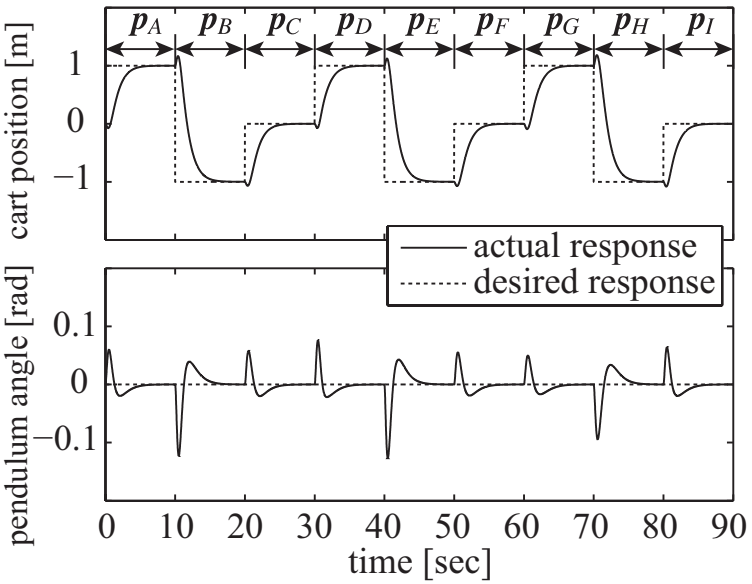



図 5.3 倒立振子の応答波形例．倒立振子の長さと重さが 5 [sec] ごとに変化する．

バック係数を定めた．

$$\int_0^\infty (\boldsymbol{x}^T Q \boldsymbol{x} + u^T R u) dt \tag{5.3}$$

線形二次レギュレータによるフィードバック係数は次式で与えられる．

$${}^{cfc}W^k = R^{-1} B^k P^k \tag{5.4}$$

ここで $P^k$ は $(4 \times 4)$ の対称行列で次の Riccati 型行列方程式の唯一な正定値の解である．

$$(A^k)^T P^k + P^k A^k + Q - P^k B^k R^{-1} (B^k)^T P^k = 0 \tag{5.5}$$

$A^k$, $B^k$ はそれぞれ $k$-th 予測器の結合荷重から推定したシステム行列と制御ベクトルを表す．また，$Q \in \Re^{4\times4}$ を単位行列とし，$R = 1$ とした．

### 5.4.2 結果

実行モードにより得られた倒立振子の応答波形の例を図 5.3 に，そのときの BMM の遷移の様子を図 5.4 に示す．制御器は解析的に得られたものであるが，パラメータの変化する倒立振子を制御し続けるためには適切なモジュール選択が行われなければならない．図 5.4 を見るとパラメータの変化に合わせ位相情報を保ったままモジュール選択が行われた．すなわち，ゆっくりとパラメータが変化する場合には担当モジュールもゆっくり変化（高々 1 モジュールの移動）し，倒立振子のパラメータが大きく変化する場合には担当モ

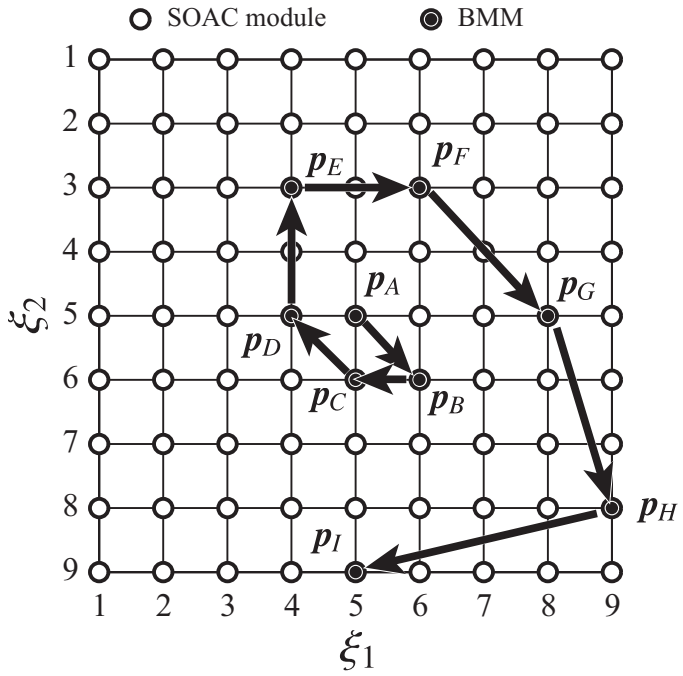



図 5.4　パラメータの変化する倒立振子の可視化.

ジュールも遠くのモジュールへと切り替わった．その結果安定した制御を維持できた（図 5.3）.

## 5.5　パラメータマップによるフィードフォワード選択

3.5.1 節でも述べたように，予測が正しかったかどうかに基づく BMM の決定法では一般に時間が遅れが生じる．したがってより速いモジュール選択を行うためにはフィードフォワードの選択法が有効となる．特に SOAC においてはパラメータマップと呼ばれる制御対象に関する情報（倒立振子では長さと重さ）の自己組織化マップを作ることができるため，パラメータマップを用いて事前に BMM を決定することができる.

予測器の学習で得られた学習率を元に生成されたパラメータマップを図 5.5 に示す．ここでパラメータマップは予測器の学習により得られた学習率 $\psi_i^k$ とパラメータベクトル $\boldsymbol{p}_i$ との積和によって得られたものであり，$\boldsymbol{p}_i$ を用いた通常の SOM により得られた結果ではないことに注意する．つまり誤差は次時刻における制御対象の状態とその予測値の間で求められたものであり，$\boldsymbol{p}_i$ とパラメータベクトル $\tilde{\boldsymbol{p}}^k$ とのユークリッド距離で計算しているわけではない.

まず，図 5.5 のパラメータマップを用いて BMM のフィードフォワード選択を行った.

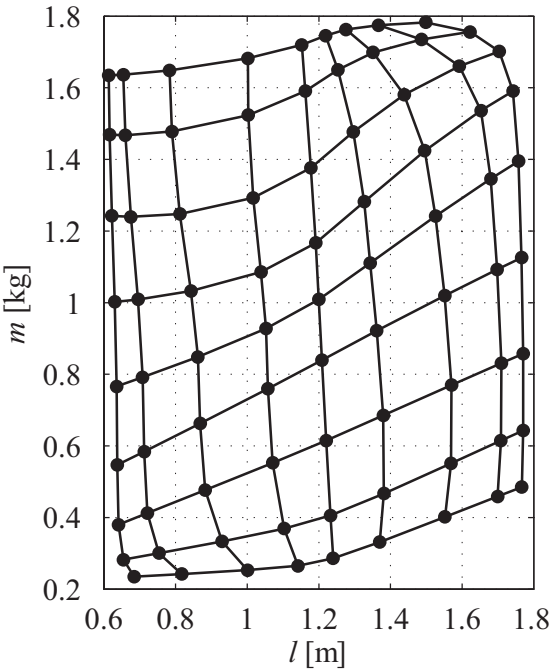



図 5.5 長さと重さに関するパラメータマップ.

その結果図 5.6 のようになった．これを通常の予測誤差に基づく BMM の結果と比較すると，全く同じ結果にはならなかったものの，BMM のずれは高々 1 モジュールであり，フィードフォワード選択の有効性が確認できた．またフィードフォワード選択による方法においても振子は倒れることなく安定した制御を維持できた.

次にフィードフォワード選択を導入することによってどの程度 BMM の切り替えが速く行われるかを確かめるために次のような実験を行った.

今長さと重さが異なる 2 つの振子があり，それぞれの長さと重さは $\boldsymbol{p}_1 = [l_1, m_1] = [1.8, 0.2]$，$\boldsymbol{p}_2 = [l_2, m_2] = [0.6, 1.8]$ であるとする．最初に $\boldsymbol{p}_1$ が提示され，10 [sec] 後に $\boldsymbol{p}_2$ に切り替わる状況を考える．このときに予測誤差に基づく通常のフィードバックの BMM 選択法とパラメータマップを用いたフィードフォワードの BMM 選択を用いた場合に BMM が両手法でどのように遷移するかを調べた．ただしフィードフォワード選択法では振子の長さと重さが何かしらの方法で推定することができるとし，ここではその推定に要する時間は考えない（したがって時間遅れがない理想の状況を考える）こととした．結果を図 5.7 に示す．フィードフォワード選択を用いた場合，時間遅れが生じないために BMM が瞬時に切り替わった．一方予測誤差に基づく方法では振子が切り替わってから BMM が安定するまでに約 0.9 [sec] の遅れが生じ，収束するまでの間，BMM はマップ空間を点々と移動した．その結果，系が不安定になることこそなかったもののフィード

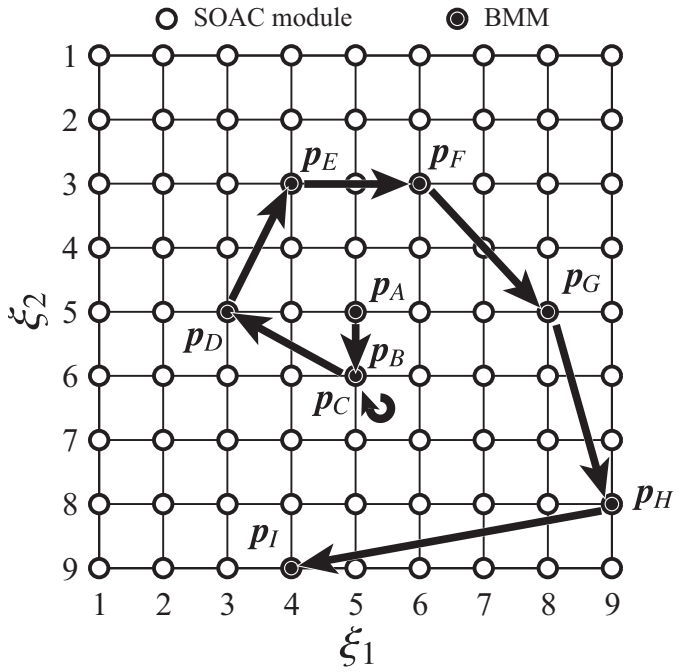



図 5.6　パラメータマップを用いたフィードフォワード選択によるパラメータ変化の可視化.

フォワード選択の場合に比べ応答が若干遅くなった（図 5.8）．この結果は $\varepsilon = 0.001$ のときの結果を表しており，$\varepsilon$ を大きくすると BMM の切り替わりに要する時間は 0.9 [sec] よりも短くなったが BMM の決定が安定せずに BMM が終始ふらつく結果となった．また両手法の距離尺度が異なるために BMM は同じ位置にはならず 1 モジュールずれて選択された.

以上の結果からフィードフォワード選択を用いた方法は切り替えが速く，BMM がふらつくこともないために予測誤差に基づく方法よりも一見有効そうに見える．しかし両者は優劣を決めるものではなく，組み合わせて用いるべきである．予測誤差に基づく方法だけでは BMM の切り替わりが遅く応答が遅れてしまう場合がある一方，見た目で BMM を決定する方法だけでは見た目と実際が異なる場合には誤った制御を行ってしまう可能性がある．そこで始めは見た目で BMM を決定しておき，そのあと予測誤差を用いた方法に切り替える方法を採用するとよい．例えば，簡単なステップ関数を用いて次のように両手法を切り替える.

$$P(k_{ff}^*) = 1 - U(t_c - t_s) = \begin{cases} 1 & (t_c < t_s) \\ 0 & (t_c \geq t_s) \end{cases} \tag{5.6}$$

$$P(k_{fb}^*) = U(t_c - t_s) = \begin{cases} 0 & (t_c < t_s) \\ 1 & (t_c \geq t_s) \end{cases} \tag{5.7}$$

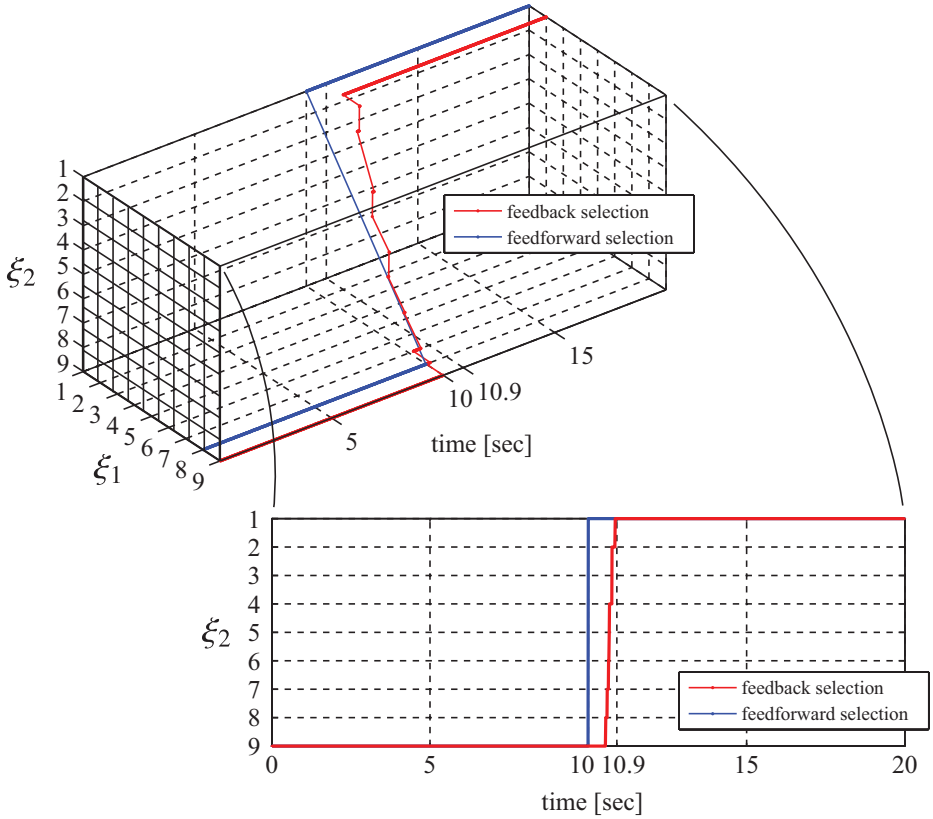



図 5.7　予測誤差に基づいた BMM のフィードバック選択と見た目のパラメータの誤差に基づいた BMM のフィードフォワード選択による BMM の遷移.

ここで $P(k^*_{ff})$ はフィードフォワード選択によって選ばれた勝者モジュール $k^*_{ff}$ が採用される確率，$P(k^*_{fb})$ は予測誤差に基づくフィードバック選択によって選ばれた勝者モジュール $k^*_{fb}$ が採用される確率を表すとする．また $t_c$ は制御対象が切り替わった瞬間からの経過時間，$t_s$ はフィードフォワード選択とフィードバック選択を切り替える時刻を表す．したがって制御対象が切り替わってから $t_s$ [sec] はフィードフォワードによる選択法が適用され，その後予測誤差に基づいたフィードバック選択法が適用される．$t_s$ の設定は状況によりけりであるが，ノイズや外乱が大きく予測誤差の平均をとるパラメータ $\varepsilon$ を小さく設定せざるを得ない場合等には $t_s$ を大きくするといったように $\varepsilon$ の値によって変更するとよい．

上記の方法を用いて実際にシミュレーションを行った結果を図 5.9 に示す．ここで扱った制御対象は先ほどと同様に $\boldsymbol{p}_1 = [l_1, m_1] = [1.8, 0.2]$，$\boldsymbol{p}_2 = [l_2, m_2] = [0.6, 1.8]$ とした．ただし，今回は $\boldsymbol{p}_2$ の重さを誤って 0.8[kg] と推定してしまった状況を考える．これは金属でできた振子の表面に木目の塗料が塗られており，一見軽そうに見えるけれども実

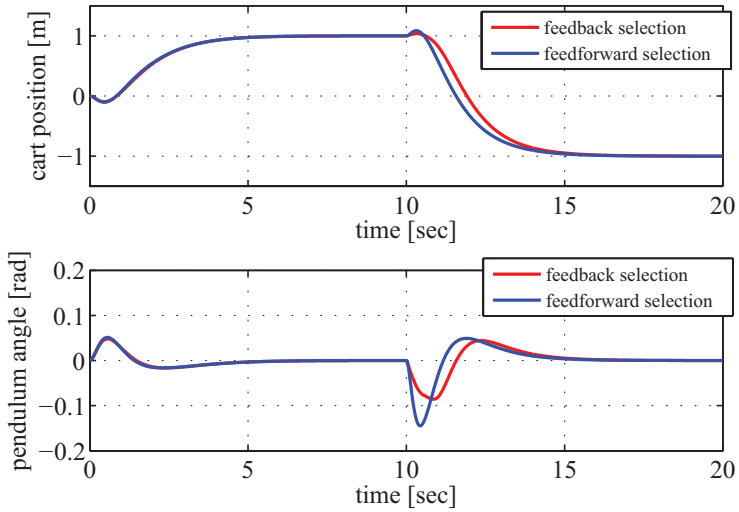



図 5.8　予測誤差に基づいた BMM のフィードバック選択と見た目のパラメータの誤差に基づいた BMM のフィードフォワード選択による応答の変化.

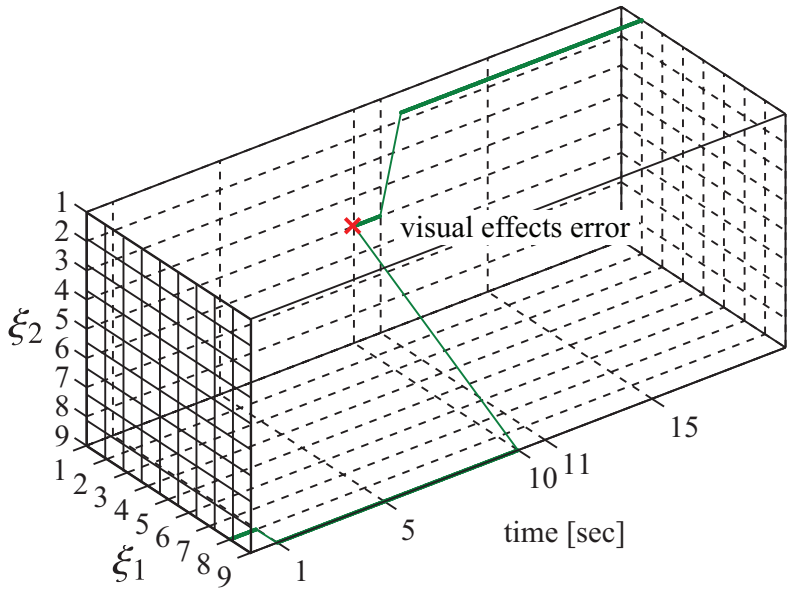



図 5.9　BMM のフィードフォワード選択とフィードバック選択の融合による BMM の遷移図.

際には重い，といった状況に相当する．式 (5.7) における切り替え時間は $t_s = 1$ とした．図 5.9 を見ると分かるように，最初の 1 [sec] は見た目誤差に基づく方法により BMM が決定され，その後予測誤差に基づき BMM が隣のモジュールへと切り替わった．続いて開始 10 [sec] 後に振子が突然他のものに切り替わると，重さの推定を誤ってしまい見た目誤差によって間違った BMM 選択が行われてしまったが，その 1 [sec] 後には予測誤差に

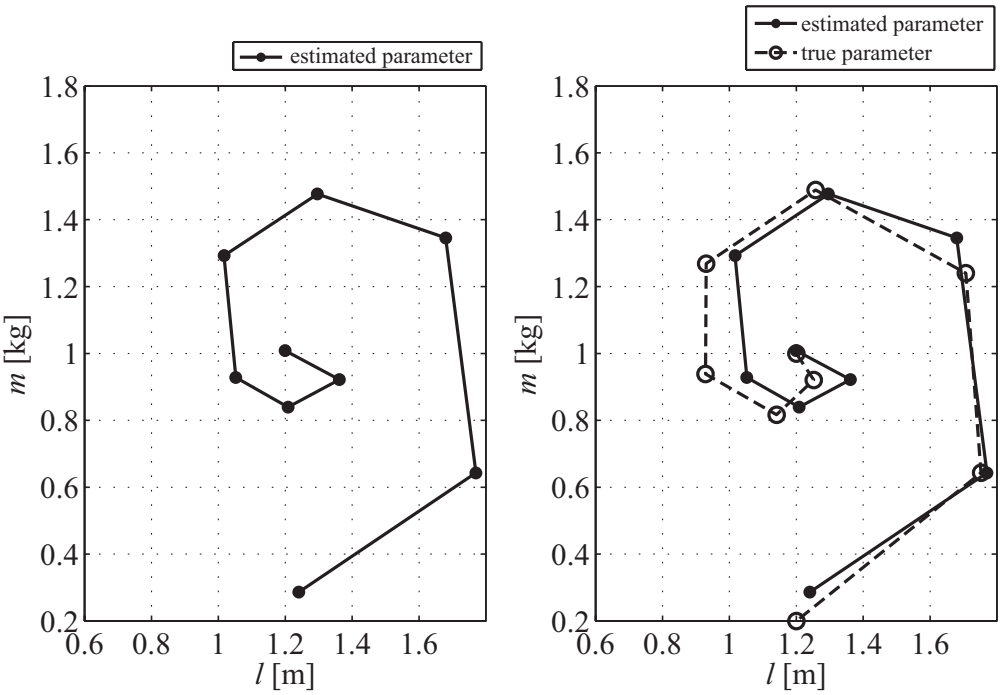



図 5.10　パラメータマップを用いた倒立振子のパラメータ推定．図中の黒丸はパラメータマップから推定した倒立振子の長さと重さのパラメータ，白丸は実際のパラメータを表している．

基づき正しい BMM へと移動した．

## 5.6　パラメータマップを用いたシステム解析例

本節では倒立振子の長さ，重さに関するパラメータマップを用いたシステム解析を行う．具体的には台車の位置，振子の角度などのセンサ情報から，倒立振子の長さと重さを推定する問題を考える．

今, 学習では用いていない長さと重さを持つ振子 ($\boldsymbol{p} = [1.60, 1.50]$) を用意し, SOAC を用いて制御することを考える．このときに振子の長さと重さの情報は与えずに，位置，速度などのセンサ情報から予測誤差に基づき BMM を決定する．実際に選ばれた BMM が持つパラメータベクトル $\tilde{\boldsymbol{p}}^*$ を調べたところ, 振子のパラメータ推定値は $\tilde{\boldsymbol{p}}^* = [1.65, 1.54]$ となり，このときの実際のパラメータ $\boldsymbol{p}$ とのユークリッド距離（誤差）はわずかに $\|\boldsymbol{p} - \tilde{\boldsymbol{p}}^*\| = 0.064$ であった．また同様の推定を 50 種類のパラメータ（訓練データで用いたパラメータで最大の長さと重さ，および最小の長さと重さを頂点に持つ長方形内からランダムに選んだ）に対して行ったところ平均推定誤差は 0.1052 であった．また図 5.4 で

得られた結果にパラメータマップを適用したところ，パラメータ推定値は図 5.10 のようになった．これらのことからパラメータ推定の有効性が確認できた．

## 5.7 考察

### 5.7.1 多入出力系の扱い

SOAC はその構造上多入出力系についても容易に扱うことができる（倒立振子は 1 入力 2 出力系）が，多入出力系を扱う場合には各要素のスケーリングをどのようにするかの問題が常につきまとう．これは SOAC の問題というよりむしろ SOM のアルゴリズムに内在する問題で，SOM で生成されるマップは入力データのスケーリングを変えることによって簡単に異なる結果を生む（異なるマップが生成される）．mnSOM においては入力データと教師データに正規化を施すことによってこの問題に対応しているが，この方法は単にマップを生成することを目的とした場合には有効であるものの，実際に（SOAC のように）生成されたマップを使う必要がある場合不都合が生じることがある．例えば，訓練データを正規化することができても，制御している過程で時々刻々獲得されるデータを訓練データのようにオフラインで正規化することはできないし，時間平均をとって近似的に正規化を行おうにも平均や分散が時間的に変化する（非定常過程）場合もある．

仮にデータのスケーリングが可能であったとすると，次にどのようなスケールで正規化を行うかの問題が生じてくる．平均 0，分散 1 にすることが常に正しいかは分からないし，またクラスごとの統計を元に正規化を行うか，それとも全クラスの統計を元に正規化を行うかで生成されるマップも変わってくる．

シミュレーションでは上記のような問題があることは認識しながらも特別なスケーリングは行わっていない（ただし数値のオーダーが極端に異なることがないような単位を選んで用いてはいる）．そのため妥当な結果を得るためには制御対象のパラメータを慎重に選ぶ必要があり，訓練データに用いた倒立振子の長さと重さの幅が異なっているのはそのためである．また，倒立振子において長さの変化に対するダイナミクスの変化と，重さの変化に対するダイナミクスの変化は対等ではなく，長さの変化に対する感度の方が敏感であることから長さの幅を小さく重さの幅を大きくとることで BMM がマップ全体に広がるように配慮している．

今後，特に非線形な多入出力系を扱う際にはスケーリングの問題はより顕著に現れることが予想され，どのようにスケーリングを行うかを議論する必要性が生じてくるであ

ろう．

### 5.7.2 パラメータマップの有効性

制御対象のパラメータを SOAC の機能モジュールの一部に組み込むことで，BMM のフィードフォワード選択法とパラメータ解析の 1 つとして制御対象のパラメータを推定することが可能であることが分かった．複数のモデルを用意してそれらの単純な切り替えで制御を行う従来の手法に比べて，提案手法を用いることで今制御対象がどういった状況にあるかを把握しながら制御できる．このことは危険予測などにおいて重要なことであるから SOAC を用いたパラメータ推定は有効であると考えられる．

### 5.7.3 フィードバック系に対する学習制御器

本章で扱った制御器は，予測器が同定したシステムに対する最適な制御器を最適レギュレータを用いて設計したものであった．これは 4 章で扱ったバネ・マス・ダンパ系と異なり，倒立振子が不安定系であり一般にフィードフォワード制御では制御できない課題であることに依る．すなわち 3 章で説明したアルゴリズムをそのまま適用できないため NNC を含めずフィードバック制御器のみで制御を行わせている．

GOMI ら [16] は，IDML（Inverse Dynamics Model Learning）と呼ばれるフィードバック誤差学習を閉ループ系に適用する手法を提案した．IDML を用いることにより閉ループ系に対しても逆モデルが獲得できることが示されている．SOAC のモジュールとして IDML を導入することで倒立振子系に応用することも可能であり，実際に適用を試みた例もある [34, 35]．この場合，ノイズの多い環境下において，通常のフィードバック制御器のみの場合よりも安定した制御が可能であった．しかし，台車つきの倒立振子の場合，振子が傾きすぎると物理的に振子を倒立状態に持って行くことが不可能となるため，訓練データの生成が困難でそれゆえ逆モデルの獲得までには至っていない．

# 第 6 章

# 討論

## 6.1 他の手法との関連

SOAC は (1) 環境の突然の変化に対応できること (2) 少ない標本からの汎化的な能力を目指して開発されたものである．序論でも取り上げたように，Jacobs と Jordan らの “mixture of experts” や Narendra らのモデル，そして Wolpert らの MPFIM はそれぞれアプローチは異なるものの (1) を目指したモデルであり，MPFIM に限っては (2) の汎化性の高いモデル構築も目指している．まず，mixture of experts と SOAC ではアーキテクチャもアルゴリズムも異なる．Narendra らの手法と Wolpert らの手法は予測器と制御器を対にしたモジュールを用いている点は SOAC と同じであるが主旨が異なる．Narendra らの手法は，工学的に有効な制御器の実現へ向けて，学習機能はほとんど持たせずに固定することで安定性に関する議論がなされている．Wolpert らの MPFIM は生物の複数のタスクに対する素早い運動を説明するための小脳のモデルとして提案されたもので，厳密な安定性は多少犠牲にしても生理学的に有効なモデル構築を目指している．そのために予測器と制御器の同時学習に拘り，各データがどの制御対象から得られたかというラベルも設けていない．汎化性に関してはモジュールの出力の線形内分をとることで向上を狙っている．SOAC も予測器と制御器を基本モジュールとして用いている点は同じであるが，mnSOM の学習アルゴリズムに基づいているため学習アルゴリズムが異なる．そのため基本的にはラベル付きの学習方法であり，予測器と制御器の獲得は予測器を学習後に制御器の学習という形をとっている．こうすることでより安定した学習が可能となっている．また SOAC では中間的な予測器と制御器の獲得が期待されるが MPFIM にそのような機能は備わっていない．

## 6.2　SOAC の安定性

ここでは SOAC の安定性に関して 3 つの見解から議論したい．3 つの見解とはすなわち，学習の安定性，制御の安定性，構造の安定性である．

### 学習の安定性

本研究によって SOAC はネットワークの初期値に依存しにくく，学習結果が安定して得られることが分かった．これに対し類似手法である MPFIM ではモジュール分割が正しく行えるかどうかはネットワークの初期値に大きく依存し，学習結果の安定性が乏しかった．この原因についてはまだはっきりと分かっていないが，SOAC ではマップ空間内で近傍半径を決めることが可能であるのに対し，MPFIM ではデータ空間内で近傍半径を決めなければならないため，入力データの変化に敏感に反応してしまい，それが元で学習が不安定になるのではないかと考えている．

### 制御の安定性

フィードバック誤差学習の安定性に関する議論がなされていることを考えると，モジュールが正しく選択されれば安定した制御が可能である．つまり制御の安定性に関して問題となるのはモジュールが切り替わってから収束するまでの間と，正しいモジュール選択が行われるかどうかの 2 点である．予測が正しく行えなかった場合には正しい制御が行えないことは明白であるので，特にモジュールが切り替わり始めてから収束するまでの間が議論の対象となる．SOAC は見た目誤差に基づくフィードフォワード選択と予測誤差のフィードバック選択を組み合わせることで素早いモジュール切り替えができる．そのため，予測誤差に基づくモジュール選択で問題となるモジュール選択の時間遅れを帳消しにしてくれるために制御の安定性に関する信頼性は高い．しかし，真の意味で制御の安定性を議論するためには理論的な導出が必要となる．これについては今後の重要な課題となるだろう．

### 構造の安定性

ここでの構造とは SOAC が持つモジュール構造を指しており，構造の安定性とはモジュールが多数存在する SOAC では，一部のモジュールがたとえ失われても（破損して

も）その制御機能は失われない，ことを意味している．SOAC では通常，学習する制御対象の数よりも多数のモジュールを扱う．しかも各モジュールはそれぞれ異なる制御器を実現しながらも隣り合うモジュールは似た制御器を形成している．したがって，例えあるモジュールが何かしらの原因によって破損したとしても近傍のモジュールは同じような制御能力を持っているからそれらを用いることで引き続き安定した制御を維持できる．

## 6.3 SOAC の課題と対策

本研究は mnSOM の制御系への応用の観点から行ったものである．したがってできるだけ mnSOM の枠組みに沿うように SOAC のアルゴリズムを書き起こした．そのためにいくつかの考慮すべき課題が残っている．本節では SOAC の現在のアルゴリズムに残された課題を明らかにし，その対応について考える．

### 多要素からなるモジュールの扱い

SOAC のモジュールは予測器と制御器から構成される（さらにパラメータベクトルが加わる場合もある）．この場合には当然予測器と制御器 2 つのネットワークの学習が必要となる．そして学習の結果，予測器と制御器は “対” になることが求められる．ここでの対とは構造上のことではなく，予測器を順モデルと言い換えれば，順モデルの逆モデルを制御器が表現することを指す．では，現在の学習アルゴリズムではどうなっているか，改めて振り返る．

SOAC は mnSOM と同様に SOM におけるバッチ型学習を基本としている．予測器については教師信号を含む学習用データを得ることは容易であり，各クラスについて事前に得ることができる．そのためオフライン学習が可能であり，バッチ型学習が適用できる，すなわち mnSOM のアルゴリズムがそのまま適用できる．これは過去の研究成果からも明らかである [52, 53]．問題は制御器の獲得であるが，現在は 2 種類の方法が存在する．1 つは 3 章で示した学習アルゴリズムによる方法であり，フィードバック誤差学習に予測誤差から算出された学習率を重みづけることで各制御器の学習をオンラインで行う．もう 1 つは 5 章で用いたような，各予測器に対する最適な制御器をそれぞれ最適制御などの制御理論に基づく方法によって解析的に求める方法である．後者は予測器にとって最適な制御器を実現するが，前者の方法では必ずしもそうならないことは 4.7.1 節で既に示した．

本研究において予測器と制御器の “微小” なずれは致命的な問題とはならなかったが，

予測器マップと制御器マップを完全に一致させるためには予測器に最適な制御器となるように解析的な方法を用いるか，あるいは各予測器にそれぞれフィードバック誤差学習を適用するなどの工夫（4.7.1 節参照）が必要となる．また Jordan と Rumelhart の順逆モデリング [22] では予測器の学習後に制御器の学習が行われる．したがって順逆モデリングを SOAC の学習則に導入すると有効かもしれない．いずれにせよ，多要素からなるモジュールに mnSOM のアルゴリズムを適用する場合には十分吟味する必要があるだろう．

### 同時学習

mnSOM（のバッチ型）は学習結果が安定して得られるために有効な手法であるが，制御系に応用するにあたって問題となるのがモジュールのオンライン学習である．フィードバック誤差学習では制御器の学習はオンラインで行われるが，フィードバック誤差学習に重み付けされる学習率は予測誤差を用いてなされるので制御器を学習により得るためには予測器の学習も行われなければならない．したがって，予測器と制御器をオンライン学習させるためには MPFIM のような同時学習が必要となるのである．予測器を先に学習してその後制御器をオンラインで学習する形態が悪いとは思わないが，同時学習によって完全にオンラインで予測器と制御器が学習できれば，工学的なモデルとしてだけでなく生理学的なモデルとしての有効性も高まると思われる（例え生理学的に有効なモデルの構築を目指していないとしても）．そのために必要となるのは SOAC の学習アルゴリズムをオンライン型に修正することであるが，これはラベル付き（どの制御対象であるか分かっている）であればそれほど難しくない．基本的には mnSOM のオンライン学習 [54] を SOAC のアルゴリズムに適用するだけで良い．ただし，これはアルゴリズムの変更が難しくないという意味であり学習が簡単であるかは別の話である．しかし，ラベルもなくて予測器と制御器を同時に学習するのは，特に SOAC のように多数のモジュールを扱う場合には，容易ではないがこれができるようになればさらに有効性は高まるであろう．

## 6.4　SOAC の一般化

SOAC は mnSOM を制御系に応用したものであり，mnSOM の機能モジュールとして順モデルと逆モデルの対（順逆モデル対）を採用したものである．Tokunaga[52] が述べているように mnSOM では機能モジュールとして MLP，RBF，RNN，SOM，NG など様々なタイプのニューラルネットワークを扱うことができる．このことは順逆モデル対に

も当てはめることができ，SOMやNGなどを順モデルあるいは逆モデルとして採用することでより精度の高い時系列予測や制御を行うことができると予想される [32, 45].

またSOACについて違う言い方をすれば，順逆対モデルを要素とするモジュラーネットワークにSOMの近傍関係を取り入れたものである．ここでSOMの代わりにエントロピー最大化の考えを導入したモデルがMPFIMであると解釈すれば（ただし，厳密には同じではなくMPFIMではラベルなし学習であるのに対しmnSOMの本質はラベルあり学習である），この自然な拡張としてSOMの代わりにNGの近傍関係を導入することもできるし，あるいは進化型SOM（Evolving SOM : ESOM）[7] などを導入することもできるだろう．Furukawa[12, 13] は既にSOMの代わりにNGを導入したモデルを提案している．

このように様々な自己組織化アルゴリズムが考えられる状況で，どの自己組織化アルゴリズムを用いるのが適当であろうか．その答えは常にユーザの目的や制御対象の特性に依存する．本論文で提案したSOACにはSOMの近傍関係が用いられているが，これはSOMの内挿機能に着目し，少ない標本数からの高い汎化性を目指したためであるし，言わばその副産物としてSOMを用いたことでパラメータ変化の可視化を行うことが可能になった．データ数が少ない場合や，可視化が行いたい場合にはSOMは有効であるが，制御性能の観点で言えば，制御対象の自由度が高くなると（SOMを用いた）SOACの性能は劣化するだろう．それはSOMは低次元多様体を表現する装置であるから，対象の（本質的な）次元が高くなれば表現し切れなくなることに依る．この場合にはSOMよりNGを用いた方が性能は向上するだろう．また，獲得したい制御対象や環境を動的に変化させたい場合もあるだろう．通常のSOMやNGではユニット数は固定されているためそれ以上のデータ表現はできないが，GNG（Growing Neural Gas）[10] やESOMなどの成長型あるいは自己進化型のアルゴリズムを用いればその問題に対応できるだろう．このように目的や課題によって自己組織化アルゴリズムを変えることにより，より一般化されたSOACの開発が可能となる．

### 6.4.1 教師あり学習から強化学習へ

mnSOMやSOACを単純に教師あり学習ということはできないが，教師あり学習，教師なし学習，そして強化学習のどれに属するかと言えば，一番近いのは教師あり学習であろう（正確には教師あり学習と教師なし学習どちらの要素も含まれている [54]）．教師あり学習で求められることは教師に倣うことと，未学習データに対する汎化性を高めること

であり，SOAC は教師あり学習システムの中でも特に汎化性の高いシステムであると言える．したがって物体操作において目標軌道が与えられると SOAC は操作物体の特性が途中で変わるような状況においても目標軌道に沿った運動を可能にする．逆に言えば，教師あり学習の枠組みでは目標軌道が与えられなければ先の物体操作は行えない．これを目標軌道が与えられないような状況においても対応できるように拡張できないだろうか．

そこで，次なる展開として教師あり学習から強化学習への発展が考えられる．例えば，物体 A，B があり，これを地点 1 から地点 2 まで動かすといった物体操作を考える．これまでの SOAC の枠組みであれば，地点 1 から地点 2 までの軌道（経路）は外から与えられる．したがって SOAC への要求は物体の種類（A，B）に関わらず同じ目標軌道（経路）に沿って物体を動かすことだけである．これが強化学習になると状況は次のように変化する．地点 1 から地点 2 へ物体を動かすというタスクは同じであるが，どのような軌道に沿って物体を移動するかの情報は与えられない．つまりここにはどの軌道に沿って物体を操作するかを自身で決定するという「自律性」が新たに要求される．問題は当然難しくなるが，その分適用範囲，可能性が広がる．

今後，SOAC の強化学習版を実現するときに MMRL（Multiple Model Reinforcement Learning）が参考になるだろう．MMRL とは MPFIM を強化学習へと拡張したモデルである．基本的なアーキテクチャは MPFIM と同じであり，制御器の構成と学習則が異なる．すなわちフィードバック誤差学習による教師あり学習を強化学習コントローラを用いた強化学習へと拡張したものである．鮫島ら [47]（あるいは Doya ら [8]）は MMRL を用いて単振子の振り上げ課題を行っている．

SOAC は MPFIM に類似した手法であるので構成上は SOAC のモジュールを強化学習コントローラに変える事で強化学習版の SOAC を作ることが可能であると考えられる．その場合に冗長な数のモジュールが必要かどうか，必要であるとすると中間的なモジュールが何を表現しているか，あるいは生成されたマップが意味をなすかどうかなど議論しなければならない事も多い．しかし，その一方で SOAC がそうであったように SOM の近傍関数を導入することで学習結果が安定して得られる，あるいは試行回数が少なくてすむなどの性能面の向上が見込める．そういう期待も込めて強化学習版 SOAC の開発に期待したい．

# 第 7 章

# 総括

本研究では，人間のような高い適応性と汎化性を持つ制御器の実現への足がかりとして，自己組織化する適応制御器（SOAC）の提案を行った．SOAC は予測器と制御器を対をモジュールとしたモジュラーネットワークであり，その学習アルゴリズムにはモジュラーネットワーク SOM（mnSOM）とフィードバック誤差学習を導入した．

SOAC の有効性を確かめるために，物理特性の異なる制御対象のアームトラッキングを行った．その結果 SOAC は突然の制御対象の変化に対し速やかな適応性を示し，目標軌道を追従し続けることに成功した．また，提案手法の性能は従来手法（MPFIM）に比べて高い制御性能を発揮し，未学習データに対する汎化性が優れていることが分かった．また同時に学習の安定性の点においても従来手法よりも優れていた．

さらに，SOAC を長さと重さのパラメータが変化する倒立振子系へ応用したところ，パラメータの変化に合わせ担当モジュールを適応的に切り替え安定した制御を実現できただけでなく，SOAC が生成したマップを用いてパラメータ変化の可視化に成功した．また SOAC を用いたフィードフォワードのモジュール選択方法とシステム解析の例として倒立振子のパラメータ推定を行い，有効性が確認された．

これらのことから提案手法の有効性が確認でき，本研究の具体的な目的であった

1. 制御対象の特性の突然の変化に対応し得る制御器の実現
2. できるだけ少ない標本数からの汎化的な制御能力を有する制御器の実現

を達成できた．今後の課題としては，SOAC の安定性に関する理論的な導出，2 段階学習を用いないオンライン学習へ向けたアルゴリズムの修正と同時学習の実現，非線形性の強い課題や多入出力系への応用，より一般化された自己組織化する適応制御器の開発，強化

表 7.1　SOAC と MPFIM の比較.

| | MPFIM | SOAC |
|---|---|---|
| 学習と制御 | 同時に学習 | 予測器の学習後に制御 |
| 必要な記憶容量 | 少 | 多 |
| 冗長な数の<br>モジュールの学習 | × | ○ |
| 追加学習 | × | ○ |
| ラベル | なし | あり |
| 学習の安定性 | 低 | 高 |
| 未学習データに<br>対する汎化性 | ○（モジュールの<br>線形内分により表現） | ○（内挿されたモジュールそのもの,<br>あるいはモジュールの線形内分により表現） |
| 可視化 | × | ○ |

学習への応用などが挙げられる.

最後に提案手法と MPFIM を比較したのが表 7.1 である．MPFIM では学習と制御は同時に行われるのに対し SOAC では予測器の学習後に制御を行う．また MPFIM では基本的に学習で用いられるクラスと同数のモジュール数についてのみ学習可能であり，モジュールが増えた場合には一般的に学習が収束しない．一方 SOAC ではたくさんのモジュールを用いることが前提であり SOM の内挿機能によって中間的なモジュールが生成される．そのため MPFIM に比べると必要な記憶容量が多い．しかし冗長なモジュールは追加学習が可能であり，内挿が不十分であった場合でも適応が可能である．MPFIM で追加学習を行うことはアルゴリズム的には可能であるが，その場合それまで表現していたオブジェクトを忘れて別のモデルを表現することになるため追加学習というよりは再学習の要素が濃い．MPFIM では学習データにラベルが付けられておらず SOAC よりも難しい課題をこなそうとしている．そのために学習結果が安定しないことも多い．SOAC の学習は mnSOM を元にしているために結果が非常に安定している．未学習データに対する汎化性については MPFIM は予測誤差を元にモジュール出力の線形内分をとることで向上を狙っている．SOAC では未学習データに相当するオブジェクトを内挿により得られた中間的なモジュールが表現することを期待している．またそれに加えて線形内分をとる方法を導入することも可能である．最後に SOAC は制御対象の変化を可視化する機能を持つが MPFIM ではそのような機能はない.

# 謝辞

本研究の遂行にあたり以下の方々にお世話になりました．ここに感謝の意を表します．

九州工業大学大学院生命体工学研究科脳情報専攻の古川徹生教授は私が研究に関して途方に暮れていた時に救いの手を差し伸べてくださいました．それが本研究となり本論文となりました．古川教授には本当に感謝の念が絶えません．心よりお礼申し上げます．また古川教授は私に海外で研究をする機会を与えてくださり，普段体験できない貴重な経験をさせていただきました．この場を借りて心より感謝いたします．

同専攻の安井湘三元教授には私が学部 4 年の時から博士後期課程 1 年までの 4 年間公私にわたってご指導，ご鞭撻を賜りました．また退官されて以降も貴重な助言を頂きました．特に英作文のスキルは安井元教授の存在なくして上達は見込めせんでしたし，論文の英文校閲でもお世話になりました．安井元教授には本当に感謝の念が絶えません．心よりお礼申し上げます．

同専攻・石川眞澄教授，宮本弘之准教授には本研究の質を高めるための貴重な助言を頂きました．厚くお礼申し上げます．石川教授には SOAC と MPFIM の差異をより顕著に，そしてより効果的に魅せるためのアイデアをいただき，論文の有効性をさらに高めることができました．宮本准教授には引用文献の誤りを指摘していただいたことで，論文の信頼性を損なわずに済みました．また研究成果をより良く示すためのアイデアをくださいました．

同専攻・徳永憲洋 COE 講師をはじめ古川研究室のメンバーおよび関係者の方々に心より感謝いたします．徳永講師が提案された mnSOM が本研究の元となったこともさることながら，論文の執筆に関して的確なコメントをいただけたことが後の論文採録につながりました．古川研究室博士後期課程の金子宗司氏，大谷誠氏の存在があったおかげで私は研究に集中することができました．博士後期課程の両氏の支えがなければ大所帯である古川研究室での学位取得は成し得なかったと思います．また，一緒くたにしてしまい大変申し訳ないのですが，後輩にあたる古川研究室の学生達には単に研究の話だけに留まらず

様々な面で支えていただきました．同様に彼らの存在があったからこそ幾多の試練を乗り越えられました．改めて心より感謝いたします．

英国サウサンプトン大学の Sandor M. Veres 教授は SOAC を人工衛星の姿勢制御に応用するアイデアを提供してくださり，共同研究として私の同大学滞在を認めてくださいました．残念ながら私の学位取得のため共同研究については中断を余儀なくされましたが，英国滞在中の経験は私にとって非常に貴重な財産になりました．これもひとえに Veres 教授が共同研究を快く承諾してくださったこと，サウサンプトン大学への滞在を許可してくださったこと，そして何より親切に私の世話をしてくださったことの賜物と思っております．改めまして心よりお礼申し上げます．

また，長い長い学生生活を支え続けてくれた両親，辛い時に心の支えとなってくれた友人達に心より感謝いたします．

なお本研究の一部は 21 世紀 COE プログラム「生物とロボットが織りなす脳情報工学の世界」（拠点番号 J19）および文部科学省科学研究費（No. 17500193）の支援の下に実施されました．

# 参考文献

[1] J.S. Albus, “A new approach to manipulator control : The cerebellar model articulation controller (CMAC),” Transactions of the AMSE. Journal of Dynamic Systems, Measurement, and Control, vol.97, 220–227, 1975.

[2] S. Amari, “Topographic organization of nerve fields,” Bulletin of Mathematical Biology, vol.42, no.3, pp.339–364, 1980.

[3] S. Amari, “Field theory of self-organizing neural nets,” IEEE Trans. Systems, Man and Cybernetics, vol.13, no.5, pp.741–748, 1983.

[4] C.G. Atkeson, and D.J. Reinkensmeyer, “Using associative content-addressable memories to control robots,” Proc. IEEE Conference on Decision and Control, pp.792–797, Austin, Texas, Dece., 1988.

[5] C.M. Bishop, M. Svensén, and C.K.I. Williams, “GTM: The generative topographic mapping,” Neural Computation, vol.10, no.1, pp.215–234, 1998.

[6] G. Deboeck, T. Kohonen 著，徳高平蔵，田中雅博 監訳，“金融・経済問題における可視化情報探索,” シュプリンガー・フェアラーク東京，1999.

[7] D. Deng, and N. Kasabov, “On-line pattern analysis by evolving self-organizing maps,” Neurocomputing, vol.51, pp.87–103, 2003.

[8] K. Doya, K. Samejima, K. Katagiri, and M. Kawato, “Multiple model-based reinforcement learning,” Neural Computation, vol.14, no.6, pp.1347–1369, 2002.

[9] M. Egmont-Petersen, D. de Ridder, and H. Handels, “Image processing with neural networks – a review,” Pattern Recognition, vol.35, pp.2279–2301, 2002.

[10] B. Fritzke, “A growing neural gas network learns topologies,” Advances in Neural Information Processing Systems, vol.7, pp.625–632, 1995.

[11] 古川徹生, “データのクラス振り分けとクラス別モデルの同時推定法,” 神経回路学会誌, vol.9, no.2, pp.92–102, 2002.

[12] T. Furukawa, “SOM of SOMs: Self-organizing map which maps a group of self-organizing maps,” Lecture Notes in computer Science, vol.3696, pp.391–396, 2005.

[13] T. Furukawa, “SOM of SOMs : An Extension of SOM from ‘ Map ’ to ‘ Homotopy ’ ,” Lecture Notes in Computer Science (Edited book of 13th International Conference of Neural Information Processing (ICONIP2006)) , vol.4232, pp.950–957, 2006.

[14] T. Furukawa, K. Tokunaga, S. Kaneko, K. Kimotsuki, and S. Yasui, “Generalized self-organizing maps (mnSOM) for dealing with dynamical systems,” Proc. International Symposium on Nonlinear Theory and its Applications, pp.231–234, Fukuoka, Japan, Nove.–Dece. 2004.

[15] T. Furukawa, “Self-Organizing Homotopy Network,” Proc. Workshop on Self-Organizing Maps (WSOM 2007), Germany, 2007.

[16] H. Gomi, and M. Kawato, “Neural network control for a closed-loop system using feedback-error-learning,” Neural Networks, vol.6, no.7, pp.933-946, 1993.

[17] H. Gomi, and M. Kawato, “Recognition of Manipulated Objects by Motor Learning With Modular Architecture Networks,” Neural Networks, vol.6, no.4, pp.485–497,1993.

[18] M. Haruno, D.M. Wolpert, and M. Kawato, “MOSAIC model for motor learning and control,” Neural Computation, vol.13, pp.2201–2220, 2001.

[19] M. Haruno, D.M. Wolpert, and M. Kawato, “Multiple Paired Forward-Inverse Models for Human Motor Learning and Control,” Advances in neural information processing systems, vol.11, pp.31–37, 1999.

[20] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive mixtures of local experts," Neural Computation, vol.3, pp.79–87, 1991.

[21] M.I. Jordan, and R.A. Jacobs, "Hierarchical mixture of experts and the EM algorithm," Neural Computation, vol.6, pp.181–214, 1994.

[22] M.I. Jordan, and D.E. Rumelhart, "Forward models: Supervised learning with a distal teacher," Cognitive Science, vol.16, pp.307–354, 1992.

[23] M. Kawato, "Feedback-error-learning neural network for supervised motor learning," Advanced Neural Computers, In Eckmiller R (Ed.), Elsevier, North-Holland, pp.365–372, 1990.

[24] 川人光男，"脳の計算理論," 産業図書，1996.

[25] M. Kawato, K. Furukawa, and R. Suzuki, "A hierarchical neural network model for the control and learning of voluntary movements," Biological Cybernetics, vol.56, pp.1–17, 1987.

[26] T. Kohonen, S. Kaski, and H. Lappalainen, "Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM," Neural Computation, vol.9, no.6, pp.1321–1344, 1997.

[27] T. Kohonen, "self-organizing maps," Springer-Verlag, 2001.

[28] M. Kuperstein, "Neural model of adaptive hand-eye coordination for single posture," Science, vol.239, 1308–1311, 1988.

[29] S.P. Luttrell, "Self-organization: A derivation from first principles of a class of learning algorithms," Proc. IEEE Int. Joint Conf. on Neural Networks (IJCNN89), Part I, pp.495–498, IEEE Press, 1989.

[30] S.P. Luttrell, "Derivation of a class of training algorithms," IEEE Trans. Neural Networks, vol.1, no.2, pp.229–232, 1990.

[31] T.M. Martinetz, H.J. Ritter, and K.J. Schulten, "Three-dimensional neural net for learning visuomotor coordination of a robot arm," IEEE Trans. Neural Net-

works, vol.1, no.1, pp.131–136, 1990.

[32] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten, ""Neural-Gas" Network for Vector Quantization and its Application to Time-Series Prediction," IEEE Trans. Neural Networks, vol.4, no.4, pp.558–569, 1993.

[33] T.W. Miller, F.H. Glanz, and L.G. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," International Journal of Robotics Research," vol.6, no.2, pp.84–98, 1987.

[34] T. Minatohara, and T. Furukawa, "Self-Organizing Adaptive Controllers: Application to the Inverted Pendulum ," Proc. Workshop on Self-Organizing Maps, pp.41–48 , France, 2005.

[35] 湊原哲也，古川徹生，"modular network SOM による自己組織化適応制御器：倒立振子への応用," 電子情報通信学会技術研究報告，vol.105, no.130, pp.49–54, 2005.

[36] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," Neural Networks, vol.1, pp.251–265, 1988.

[37] A. Miyamura, and H. Kimura, "Stability of feedback error learning scheme," Systems & Control Letters, vol.45, pp.303–316, 2002.

[38] M.A. Motter, and J.C. Principe, "Predictive multiple model switching control with the self-organizing map," International Journal of Robust and Nonlinear Control, vol.12, no.11, pp.1029–1051, 2002.

[39] K.S. Narendra, J. Balakrishnan, and M.K. Ciliz, "Adaptation and learning using multiple models, switching, and tuning," IEEE Control Systems Magazine, vol.15, no.3, pp.37–51, 1995.

[40] K.S. Narendra, and J. Balakrishnan, "Adaptive control using multiple models," IEEE Trans. Automatic Control, vol.42, no.2, pp.171–187, 1997.

[41] S. Nishida, K. Ishii, and T. Furukawa, "An Online Adaptation Control System Using mnSOM," Lecture Notes in Computer Science (Edited book of 13th Inter-

national Conference of Neural Information Processing (ICONIP2006)), vol.4232, pp.935–942, 2006.

[42] 西田周平, 石井和男, 古川徹生, “水中ロボットにおける自己組織的行動獲得システム -第一報:自己組織化マップを用いた運動制御システムの提案-,” 日本船舶海洋工学会論文集 第 3 号, pp.205–213, 2006.

[43] P. Pajunen, A. Hyvärinen, and J. Karhunen, “Nonlinear blind source separation by self-organizing maps,” Proc. International Conference on Neural Information Processing (ICONIP’96), vol.2 pp.1207–1210, 1996.

[44] K. Pawelzik, J. Kohlmorgen, and K.-R. Müller, “Annealed competition of experts for a segmentation and classification of switching dynamics,” Neural Computation, vol.8, no.2, pp.340–356, 1996.

[45] J.C. Principe, L. Wang, and M.A. Motter, “Local Dynamic Modeling with Self-Organizing Maps and Applications to Nonlinear System Identification and Control,” Proc. IEEE, vol.86, no.11, pp.2240–2258, 1998.

[46] K. Rose, E. Gurewitz, and G.C. Fox,“ Statistical mechanics and phase transitions in clustering, ” Physical Review Letters, vol.65, no.8, pp.945–948, 1990.

[47] 鮫島和行, 片桐憲一, 銅谷賢治, 川人光男, “複数の予測モデルを用いた強化学習による非線形制御, ” 電子情報通信学会論文誌, vol.J84-D-II, no.9, pp.2092–2106, 2001.

[48] J.W. Sammon, “A Nonlinear Mapping for Data Structure Analysis,” IEEE Trans. Computers, vol. 18, no.5, pp.401–409, 1969.

[49] 鈴木敏, 安藤宏志, “2 次元射影像からの 3 次元物体の認識と類別 -モジュール構造を用いた教師なし学習モデル - ,” 電子情報通信学会論文誌, vol.J79-D-II, no.7, pp.1291–1300, 1996.

[50] S. Suzuki, and H. Ando, “A modular network scheme for unsupervised 3D object recognition,” Neurocomputing, vol.31, pp.15–28, 2000.

[51] K. Tokunaga, and T. Furukawa, “Nonlinear ASSOM constituted of autoassociative neural modules,” Proc. Workshop on Self-Organizing Maps, pp.637–644,

2005.

[52] K. Tokunaga, T. Furukawa, and S. Yasui, “Modular Network SOM: Self-Organizing Maps in Function Space,” Neural Information Processing – Letters and Reviews, vol.9, pp.15–22, 2005.

[53] 徳永憲洋，肝付謙二，安井湘三，古川徹生，“関数空間形 SOM,” 神経回路学会誌, vol.12，no.1，pp.39–51，2005.

[54] 徳永憲洋，古川徹生，“一般化自己組織化マップ〜 教師あり学習と教師なし学習の融合 〜,” 電子情報通信学会技術研究報告, vol.35, pp.75–80, 2006.

[55] 徳高平蔵，岸田悟，藤村喜久郎，“自己組織化マップの応用ー多次元情報の 2 次元可視化ー,” 海文堂，1999.

[56] D.J. Willshaw, and C. von der Malsburg, “How patterned neural connections can be set up by self-organization,” Proc. Roy. Soc. Lond. B, vol.194, pp.431–445, 1976.

[57] D.M. Wolpert, and M. Kawato, “Multiple paired forward and inverse models for motor control,” Neural Networks, vol.11, pp.1317–1329, 1998.

# 研究業績リスト

## I. 学術論文

1. 湊原哲也，古川徹生，“適応性と汎化性を考慮した自己組織化適応制御器，” 電子情報通信学会誌，2007（*accepted*）.

## II. 国際会議

**（口頭発表・査読あり）**

1. T. Minatohara, T. Furukawa, “Self-Organizing Adaptive Controllers: Application to the Inverted Pendulum ,” Proc. Workshop on Self-Organizing Maps, pp.41–48 , France, 2005.

**（ポスター発表・査読あり）**

1. T. Minatohara, T. Furukawa, “A proposal of self-organizing adaptive controller (SOAC),” Proc. International Conference on Brain-inspired Information Technology, Japan, 2005.

**（ポスター発表・査読なし）**

1. T. Minatohara, T. Furukawa, “An adaptive controller based on modular network SOM,” Proc. Postech-Kyutech Joint Workshop on Neuroinformatics, Korea, 2005.

## III. 国内学会

**（口頭発表・査読なし）**

1. 湊原哲也，古川徹生，“modular network SOM による自己組織化適応制御器：倒立振子への応用,” 電子情報通信学会技術研究報告，vol.105, no.130, pp.49–54, 2005.