# コネクショニズムを応用した第二言語習得研究の試みⅢ
## ― 発話行為を想定した過去時制の学習 ―

石崎 貴士

**要旨**

石崎（2011）では、コネクショニズムを第二言語習得研究に応用した実習例として、同一の被験者が母語と第二言語で過去時制を学ぶシミュレーションを提案した。この実習例では、信号化した音素による入出力自体を軸とすることにより、母語と第二言語を共通のフォーマット上で扱うことを可能にしている。しかしながら、実際の発話行為を想定した場合、動詞の原形を表象する音韻の入力に対し、その過去形を表象する音韻を出力するという場面は考えにくい。そこで本研究では、意味表象を軸とするフォーマットを用いることにより、実際の発話行為を想定した母語と第二言語による過去時制の学習を模擬する実習例を提案する。今回のシミュレーションの結果、第二言語での学習は、いずれもバイリンガルの学習より早い段階で完了したが、母語による学習と比較した場合には、先に学習した母語によって学習完了の段階が早くなる促進効果と遅くなる干渉効果の両方が確認された。

### 1. 目的
### 1.1　先行研究に見られる問題点

コネクショニズムは、これまで主に心理学の領域で注目されてきたモデルであったため、解説書で取り上げられているコンピュータシミュレーションの実習例も心理学的な見地からのものが多く、それをそのまま第二言語習得研究に応用することはできない。例えば Plunkett & Elman（1997）や McLeod, Plunkett & Rolls（1998）で紹介されている

英語の過去時制の学習を模擬するシミュレーションも、人工的に 3 音素に統制して形成した動詞の語幹（英語の音韻規則に従うことで英語であると見なす）と英語の過去時制の標識となる接尾辞“-ed”の発音との対応関係を学習する、英語に特化された特定の言語に依存するものとなっている。

そこで石崎（2011）では、信号化した音素による入出力自体を軸として、母語と第二言語を共通のフォーマット上で扱うことができるシミュレーションの実習例を提案した。動詞の原形（語幹）を入力すると、それに対応する過去形が出力される形式は、先述のシミュレーションと同じだが、語幹は人工のものでなく実在する単語を用いており、出力についても、過去時制の標識となる接尾辞の発音のコード化は用いずに、入力と同様、直接音素で過去形を表象している。このように特定の言語に特化された簡易化や変換規則に頼ることなく、直接信号化された音素による入出力自体を軸とすることにより、英語であれ日本語であれ、共通のフォーマット上でシミュレーションを行うことが可能となった。

しかしながら、実際の人間による発話行為を想定した場合、このシミュレーションのように、動詞の原形を表象する音韻の入力に対し、その動詞の過去形を表象する音韻を出力するという場面は考えにくい。そこで本研究では、実際の発話行為を想定した、第二言語習得研究にも応用可能な新たなコンピュータシミュレーションの実習例を提案する。

### 1.2　発話行為を想定したシミュレーション

実際の発話行為を想定した場合、動詞の原形を表象する音韻が、まず頭に浮かんでから、その動詞の過去形が、それを表象する音韻の形で口から発せられるということは考えにくく、むしろ、現在形であるか過去形であるかの時制の判断も含め、頭の中に形成されたある種の概念が、発話という行為を通して音韻化されると考えられる。この概念については、特定の言語に依存せずに、発話を行う過程において個々の言語特有の規則を適用しながら音韻化されると考えることもできるが、コネクショニズムでは、情報の処理において予め定められた規則を適用すること自体を想定していないため（Rumelhart & McClelland 1986）、概念そのものの中に特定の言語に依存しない領域と個々の言語に特有の領域と

が共存すると見なすことにする。

具体的には、上述の概念を本論文では意味表象と位置づけ、この意味表象を、言語に依存しない共通の領域と言語によって異なる固有の領域、さらに動詞の時制を表象する領域という３つの下位範疇に分ける。実際の発話行為を想定したシミュレーションでは、ある動詞の持つ言語に依存しない共通のイメージをコード化したものと、その言語に特有の領域をコード化したもの、さらに当該の動詞の時制をコード化したものの 3 種類のコードを、その動詞の意味表象コードとして入力すると、音韻化された当該の動詞が、当該の言語、当該の時制で出力される。その際の言語的な処理については、特定の言語に固有の変換規則などを一切介在させることなく、神経細胞間の結合強度と個々の神経細胞内の閾値の調整のみで行う。このことは、母語話者が規則を意識することなく規則に適った言語使用を行っているという事実に基づいている。

## 2. 方法

### 2.1　意味表象を軸とした共通の変換フォーマット

第二言語習得研究に応用可能なシミュレーションを行うには、動詞の現在形と過去形を正しく表出するという同一の事象が、共通のフォーマット上でコード変換された母語と第二言語によって模擬されなければならない。今回のシミュレーションでは、そのような共通の変換フォーマットとして、入力については先述した意味表象を軸とするフォーマットを、出力については石崎（2011）で音素の信号化を行ったフォーマットを用いることにする。

入力のための共通変換フォーマットとして、本研究では意味表象を軸としたフォーマットを用いる。また、今回のシミュレーションで学習の対象とした動詞は、石崎（2011）で用いたものから抽出し（come、give、look、take、go、have、行く、来る、食べる、する）、それぞれに対し日本語（あるいは英語）の相当語を対応させた（行く/go、来る/come、見る/look、食べる/eat、とる/take、持つ/have、あげる/give、する/do）。これらのペアは完全に意味が一致しているわけではないが、重なる部分も存在する。そこで、まず、これらのペアから、そのような共通するイメージを抽出し（①「行く/go」、②「来る/come」、③「見る/look」、④

「食べる/eat」、⑤「とる/take」、⑥「持つ/have」、⑦「あげる/give」、⑧「する/do」：これらは便宜的に番号を用いて記述しているが、実際には特定の言語に依存しないイメージを表している)、それらのイメージを３ビットでコード化する（具体的には、①を“１１１”、②を“１１０”、③を“１０１”、④を“１００”、⑤を“０１１”、⑥を“０１０”、⑦を“００１”、⑧を“０００”とする)。（表１を参照）

**表1　入力コード変換表【言語に共通するイメージ】**

| 日本語 | 英　語 | ｲﾒｰｼﾞ | ｺｰﾄﾞ |
|---|---|---|---|
| 行く | go | ① | １１１ |
| 来る | come | ② | １１０ |
| 見る | look | ③ | １０１ |
| 食べる | eat | ④ | １００ |
| とる | take | ⑤ | ０１１ |
| 持つ | have | ⑥ | ０１０ |
| あげる | give | ⑦ | ００１ |
| する | do | ⑧ | ０００ |

次に、ペアで共通するイメージからはみ出てしまう言語固有の要素について補完するため、１ビットを用いて当該の言語を特定する（日本語なら“０”、英語なら“１”とする)。さらに、時制を表象する要素についても、コネクショニズムでは発話を行う過程において個々の言語に特有の規則を適用しながら音韻化するとは考えにくいため、概念そのものの中に組み込む必要がある。そのため、１ビットを用いて当該の時制が現在形なら“０”、過去形なら“１”とする。以上、今回のシミュレーションでは、当該の入力信号が示す意味表象を、言語に共通するイメージに３ビット、言語固有の要素に１ビット、時制の表象に１ビット、計５ビットを用いて特定する。

一方、出力については、石崎（2011）で音素の信号化を行ったフォーマットを用いる。このフォーマットは直接音素を表象するため、特定の言語に依存することなく母語と第二言語で共通に利用できる。音素のコード化については６ビットで１つの音素を特定する。具体的には、最初

**表2 音素のコード変換表**

| 音素 | 文字コード | 子・母/有・無/長・短 | 構音の方法 | 構音の場所 |
|---|---|---|---|---|
| /p/ | p | 子音・無声（00） | 高（11） | 前（11） |
| /b/ | b | 子音・有声（01） | 高（11） | 前（11） |
| /t/ | t | 子音・無声（00） | 高（11） | 中後（01） |
| /d/ | d | 子音・有声（01） | 高（11） | 中後（01） |
| /k/ | k | 子音・無声（00） | 高（11） | 後（00） |
| /g/ | g | 子音・有声（01） | 高（11） | 後（00） |
| /f/ | f | 子音・無声（00） | 中（10） | 前（11） |
| /v/ | v | 子音・有声（01） | 中（10） | 前（11） |
| /θ/ | T | 子音・無声（00） | 中（10） | 中前（10） |
| /ð/ | H | 子音・有声（01） | 中（10） | 中前（10） |
| /s/ | s | 子音・無声（00） | 中（10） | 中後（01） |
| /z/ | z | 子音・有声（01） | 中（10） | 中後（01） |
| /ʃ/ | S | 子音・無声（00） | 中（10） | 後（00） |
| /ʒ/ | Z | 子音・有声（01） | 中（10） | 後（00） |
| /m/ | m | 子音・有声（01） | 鼻（00） | 前（11） |
| /n/ | n | 子音・有声（01） | 鼻（00） | 中後（01） |
| /ŋ/ | G | 子音・有声（01） | 鼻（00） | 後（00） |
| /w/ | w | 子音・有声（01） | 低（01） | 前（11） |
| /l/ | l | 子音・有声（01） | 低（01） | 中前（10） |
| /r/ | r | 子音・有声（01） | 低（01） | 中後（01） |
| /h/ | h | 子音・無声（00） | 低（01） | 後（00） |
| /j/ | y | 子音・有声（01） | 低（01） | 後（00） |
| /iː/ | E | 母音・長（11） | 高（11） | 前（11） |
| /ɪ/ | i | 母音・短（10） | 高（11） | 前（11） |
| /uː/ | U | 母音・長（11） | 高（11） | 後（00） |
| /ʊ/ | u | 母音・短（10） | 高（11） | 後（00） |
| /eɪ / | A | 母音・長（11） | 中（10） | 前（11） |
| /e/ | e | 母音・短（10） | 中（10） | 前（11） |
| /ɚː/ | ~ | 母音・長（11） | 中（10） | 中後（01） |
| /ə/ | ? | 母音・短（10） | 中（10） | 中後（01） |
| /oʊ/ | O | 母音・長（11） | 中（10） | 後（00） |
| /ʌ/ | a | 母音・短（10） | 中（10） | 後（00） |
| /æ/ | @ | 母音・短（10） | 低（01） | 前（11） |
| /aɪ/ | I | 母音・長（11） | 低（01） | 中前（10） |
| /aʊ/ | # | 母音・長（11） | 低（01） | 中後（01） |
| /ɔː/ | - | 母音・長（11） | 低（01） | 後（00） |
| /ɔ/ | o | 母音・短（10） | 低（01） | 後（00） |
| ϕ | * | （00） | （00） | （00） |

の 2 ビットで子音・無声音（００）、子音・有声音（０１）、短母音（１０）、長母音（１１）を表象し、次の 2 ビットを用いて構音の方法（鼻音（００）、高位（１１）、中位（１０）、低位（０１））を表象、さらに 2 ビットを用いて構音の場所（前方（１１）、中・前寄り（１０）、中・後寄り（０１）、後方（００））を表象する。また、当該の動詞が 6 音素に満たない場合にも対応できるよう、どの音素にも該当しない空（カラ）の音素コード（6 ビット全てが 0 となる“０００００００”）も設定している。（表 2 参照）

このようなフォーマットを用いることにより、母語および第二言語で動詞の現在形と過去形を正しく表出する学習を模擬することが可能となる。以下の節では、このフォーマットに基づいたシミュレーションを実行する際の具体的な設定方法と手続きについて解説する。

### 2.2　シミュレーションの設定

本研究では Plunkett らによって開発された“tlearn”を用いてシミュレーションを行う。tlearn は、Windows や Mac といった汎用性の高い OS 上で動作し、操作性にも優れているのみならず、インターネット上で公開されており、無料でダウンロードすることができる。また、tlearn については、Plunkett & Elman (1997) や McLeod, Plunkett & Rolls (1998) など、設定や操作の方法などを詳しく紹介した解説書も出版されている。この tlearn の登場により、コンピュータによるシミュレーションは、大分身近なものになったと言える（守 2002）。

この tlearn を用いてシミュレーションを実行するには、ネットワークの構成を設定する設定ファイル（configuration file）、ネットワークに提示する入力信号を設定するデータファイル（data file）、個々の入力信号に対する正しい出力を設定する教師信号ファイル（teach file）という 3 種類のファイルを作成しなければならない。

ネットワークの構成を設定する設定ファイル（ファイル拡張子は“.cf”）として、まず、今回のシミュレーションでは、入力層、隠れ層、出力層の三層から成るフィード･フォワードのネットワークを構成する。入力する意味表象を特定するために、言語共通のイメージに 3 ビット、言語固有の要素に 1 ビット、時制の表象に 1 ビットを要するので、入力

ユニットを５ビットに設定する。一方、出力される６つの音素を特定するには、１つの音素を特定するのに６ビットを要するので、出力ユニットとして 36 ビットが必要になる。また、隠れ層のユニット数については、石崎（2011）と同様、出力ユニットと同数の 36 ビットを設けることにした（設定ファイルでの項目ごとの設定の詳細については、図１を参照。）

```
NODES:
nodes = 72
inputs = 5
outputs = 36
output nodes are 37-72
CONNECTIONS:
groups = 0
1-36 from i1-i5
37-72 from 1-36
1-72 from 0
SPECIAL:
selected = 1-36
weight_limit = 1.00
```

**図1　設定ファイルでの入力内容**

設定ファイルは今回の全てのシミュレーションで共通のものを用いるが、入力信号を設定するデータファイル（ファイル拡張子は“.data”）は実施するシミュレーションによって異なる。まず、日本語を母語として習得する場合を想定したシミュレーションを実施するために、８つの動詞の日本語での現在形と過去形の意味を表象する日本語用の意味表象データファイルを作成する（表３参照）。また、英語を母語として習得する場合を想定したシミュレーションを実施するために、先程の動詞の英語での現在形と過去形の意味を表象する英語用の意味表象データファイルも作成する[1]（表４参照）。さらに、バイリンガルとして日本語と英語の

表3 データファイルコード【意味表象・日本語】

| 意味表象 | 言語共通 | | 言語固有 | 時制 |
|---|---|---|---|---|
| | ｲﾒｰｼﾞ | ｺｰﾄﾞ | 日(0)／英(1) | 現在(0)／過去(1) |
| 行く | ① | 1 1 1 | 0 | 0 |
| 来る | ② | 1 1 0 | 0 | 0 |
| 見る | ③ | 1 0 1 | 0 | 0 |
| 食べる | ④ | 1 0 0 | 0 | 0 |
| とる | ⑤ | 0 1 1 | 0 | 0 |
| 持つ | ⑥ | 0 1 0 | 0 | 0 |
| あげる | ⑦ | 0 0 1 | 0 | 0 |
| する | ⑧ | 0 0 0 | 0 | 0 |
| 行った | ① | 1 1 1 | 0 | 1 |
| 来た | ② | 1 1 0 | 0 | 1 |
| 見た | ③ | 1 0 1 | 0 | 1 |
| 食べた | ④ | 1 0 0 | 0 | 1 |
| とった | ⑤ | 0 1 1 | 0 | 1 |
| 持った | ⑥ | 0 1 0 | 0 | 1 |
| あげた | ⑦ | 0 0 1 | 0 | 1 |
| した | ⑧ | 0 0 0 | 0 | 1 |

表4 データファイルコード【意味表象・英語】

| 意味表象 | 言語共通 | | 言語固有 | 時制 |
|---|---|---|---|---|
| | ｲﾒｰｼﾞ | ｺｰﾄﾞ | 日(0)／英(1) | 現在(0)／過去(1) |
| go | ① | 1 1 1 | 1 | 0 |
| come | ② | 1 1 0 | 1 | 0 |
| look | ③ | 1 0 1 | 1 | 0 |
| eat | ④ | 1 0 0 | 1 | 0 |
| take | ⑤ | 0 1 1 | 1 | 0 |
| have | ⑥ | 0 1 0 | 1 | 0 |
| give | ⑦ | 0 0 1 | 1 | 0 |
| do | ⑧ | 0 0 0 | 1 | 0 |
| went | ① | 1 1 1 | 1 | 1 |
| came | ② | 1 1 0 | 1 | 1 |
| looked | ③ | 1 0 1 | 1 | 1 |
| ate | ④ | 1 0 0 | 1 | 1 |
| took | ⑤ | 0 1 1 | 1 | 1 |
| had | ⑥ | 0 1 0 | 1 | 1 |
| gave | ⑦ | 0 0 1 | 1 | 1 |
| did | ⑧ | 0 0 0 | 1 | 1 |

**表5 教師信号ファイルコード【音素・日本語】**

| 単語 | 音素 | 文字コード | 数値コード (1) | (2) | (3) | (4) | (5) | (6) |
|---|---|---|---|---|---|---|---|---|
| 【現在形】 | | | | | | | | |
| 行く | /ɪkʊ/ | iku*** | 101111 | 001100 | 101100 | 000000 | 000000 | 000000 |
| 来る | /kʊrʊ/ | kuru** | 001100 | 101100 | 010101 | 101100 | 000000 | 000000 |
| 見る | /mɪrʊ/ | miru** | 010011 | 101111 | 010101 | 101100 | 000000 | 000000 |
| 食べる | /tʌberʊ/ | taberu | 001101 | 101000 | 011111 | 101011 | 010101 | 101100 |
| とる | /tɔrʊ/ | toru** | 001101 | 100100 | 010101 | 101100 | 000000 | 000000 |
| 持つ | /mɔtsʊ/ | motsu* | 010011 | 100100 | 001101 | 001001 | 101100 | 000000 |
| あげる | /ʌgerʊ/ | ageru* | 101000 | 011100 | 101011 | 010101 | 101100 | 000000 |
| する | /sʊrʊ/ | suru** | 001001 | 101100 | 010101 | 101100 | 000000 | 000000 |
| 【過去形】 | | | | | | | | |
| 行った | /ɪttʌ/ | itta** | 101111 | 001101 | 001101 | 101000 | 000000 | 000000 |
| 来た | /kɪtʌ/ | kita** | 001100 | 101111 | 001101 | 101000 | 000000 | 000000 |
| 見た | /mɪtʌ/ | mita** | 010011 | 101111 | 001101 | 101000 | 000000 | 000000 |
| 食べた | /tʌbetʌ/ | tabeta | 001101 | 101000 | 011111 | 101011 | 001101 | 101000 |
| とった | /tɔttʌ/ | totta* | 001101 | 100100 | 001101 | 001101 | 101000 | 000000 |
| 持った | /mɔttʌ/ | motta* | 010011 | 100100 | 001101 | 001101 | 101000 | 000000 |
| あげた | /ʌgetʌ/ | ageta* | 101000 | 011100 | 101011 | 001101 | 101000 | 000000 |
| した | /ʃɪtʌ/ | Sita** | 001000 | 101111 | 001101 | 101000 | 000000 | 000000 |

両方を習得する場合を想定したシミュレーションを実施するために、日本語用と英語用を合わせたバイリンガル用の意味表象データファイルを作成する。この意味表象データファイルは、第二言語として英語（あるいは日本語）を習得する場合を想定したシミュレーションを実施する場合にも用いる。

個々の入力信号に対する正しい出力を設定する教師信号ファイル（ファイル拡張子は“.teach”）も、データファイルと同様に実施するシミュレーションごとに作成する。上述の意味表象データファイルに対応する日本語（あるいは英語）の動詞の現在形と過去形を表象する音素を、1音素6ビットで構成する数値コードに変換して作成する。常に6つの音素を出力するよう設定されているので、当該の動詞が6音素に満たない場合には、空（カラ）の音素コードを割り振って充当している。（日本語

表6 教師信号ファイルコード【音素・英語】

| 単語 | 音素 | 文字コード | 数値コード (1) | (2) | (3) | (4) | (5) | (6) |
|---|---|---|---|---|---|---|---|---|
| 【現在形】 | | | | | | | | |
| go | /g oʊ/ | g O * * * * | 011100 | 111000 | 000000 | 000000 | 000000 | 000000 |
| come | /k ʌ m/ | k a m * * * | 001100 | 101000 | 010011 | 000000 | 000000 | 000000 |
| look | /l ʊ k/ | l u k * * * | 010110 | 101100 | 001100 | 000000 | 000000 | 000000 |
| eat | /iː t/ | E t * * * * | 111111 | 001101 | 000000 | 000000 | 000000 | 000000 |
| take | /t eɪ k/ | t A k * * * | 001101 | 111011 | 001100 | 000000 | 000000 | 000000 |
| have | /h æ v/ | h @ v * * * | 000100 | 100111 | 011011 | 000000 | 000000 | 000000 |
| give | /g ɪ v/ | g i v * * * | 011100 | 101111 | 011011 | 000000 | 000000 | 000000 |
| do | /d uː/ | d U * * * * | 011101 | 111100 | 000000 | 000000 | 000000 | 000000 |
| 【過去形】 | | | | | | | | |
| went | /w e n t/ | w e n t * * | 010111 | 101011 | 010001 | 001101 | 000000 | 000000 |
| came | /k eɪ m/ | k A m * * * | 001100 | 111011 | 010011 | 000000 | 000000 | 000000 |
| looked | /l ʊ k t/ | l u k t * * | 010110 | 101100 | 001100 | 001101 | 000000 | 000000 |
| ate | /eɪ t/ | A t * * * * | 111011 | 001101 | 000000 | 000000 | 000000 | 000000 |
| took | /t ʊ k/ | t u k * * * | 001101 | 101100 | 001100 | 000000 | 000000 | 000000 |
| had | /h æ d/ | h @ d * * * | 000100 | 100111 | 011101 | 000000 | 000000 | 000000 |
| gave | /g eɪ v/ | g A v * * * | 011100 | 111011 | 011011 | 000000 | 000000 | 000000 |
| did | /d ɪ d/ | d i d * * * | 011101 | 101111 | 011101 | 000000 | 000000 | 000000 |

用については表５を、英語用については表６をそれぞれ参照のこと。これら２つを合わせたものがバイリンガル用の教師信号ファイルとなる。）

### 2.3　シミュレーションの手続き

実際の発話行為を想定した今回のシミュレーションでは、意味表象を入力すると、音韻化された当該の動詞が、当該の言語、当該の時制で出力される。研究の大まかな流れとして、まず被験者を特定し、その被験者に適した学習率と慣性項を設定した後（これを初期状態とする）、母語として日本語（または英語）を習得する場合を想定したデータファイルを用いて、動詞８語の現在形と過去形の音韻表象を学習させる（母語の習得Ⅰ＆Ⅱ）。次に、ネットワークを初期状態に戻して、最初から日本語と英語の両方を習得するバイリンガルを想定したデータファイルを用い

て、先程の動詞の現在形と過去形の音韻表象を日本語と英語で学ばせる（バイリンガルの習得）。その後、今度は母語として日本語（または英語）の学習が完了した時点でのユニット間の結合強度を初期値とし、バイリンガルを想定したデータファイルを用いて、第二言語として英語（または日本語）を習得する場合を模擬した学習を行う（第二言語の習得Ⅰ＆Ⅱ）。最後に、これらのシミュレーションの結果を踏まえ、学習の過程にどのような違いが見られるかを考察する。

tlearn を用いた具体的な手順としては、まず、前節で設定したネットワークの設定ファイルと個々のシミュレーションに合ったデータファイルおよび教師信号ファイルを指定して、ネットワークのトレーニングを実行するためのプロジェクトファイルを作成する。また、トレーニング実行の際のオプション設定（Training Options）で、ユニット間の初期結合強度をランダムに割り振る乱数の種「ランダムシード（random seed）」の値を入力する。この値を統一することにより、特定の被験者を想定することができる。ランダムシードに対しては、適当な学習率（learning rate）と慣性項（momentum）も設定する。入力信号の提示順序についてはランダムであるが、提示回数を統制するため、必ず一巡しながら進んでいくよう、“Train Randomly”を選択して“With Replacement”のチェックを外す。“Use and Log RMS Error”を選択し、入力信号を一巡するごとに RMS エラーのログをとるよう、設定画面で当該の sweep 数を入力する。さらに、入力信号それぞれに対し 10 回の試行を行うごとに、ユニット間の結合強度をウェイトファイルとして記録していくよう“Dump weights”をチェックして当該の sweep 数を入力する。トレーニング実行後、テスティングオプション（Testing Options）の設定画面で、これらのウェイトファイルを指定し、どの段階で全ての意味表象に対し正しい出力ができるようになったのかを、アウトプットの翻訳機能を活用しながら特定する。以上の作業をシミュレーションごとに繰り返す。（この他に、第二言語の習得の場合は、母語のシミュレーションで判明した学習完了時点でのウェイトファイルを“Load Weights File”にチェックを入れて指定する。こうすることで、その被験者が母語での学習を完了した状態を想定できる。）

## 3. シミュレーションの結果

### 3.1　母語の習得Ⅰ（母語：日本語）

ここでは、日本語を母語として習得する場合を想定したシミュレーションとして、日本語の動詞８語の現在形と過去形それぞれの意味表象に対応する音韻表象を学習させる。本研究で行われるシミュレーションは、全て同一の被験者によってなされることを想定しているので、ランダムシード、学習率、慣性項の値を統一する（ランダムシード：9、学習率：0.5、慣性項：0.8）。入力信号を一巡するごとに RMS エラーのログをとるよう当該の sweep 数を入力する（Log error every 16 sweeps）。さらに、入力信号それぞれに対し 10 回の試行を行うごとにウェイトファイルを記録していくよう当該の sweep 数を入力する（Dump weights every 160 sweeps）。

ネットワークに、これら日本語の動詞８語の現在形と過去形それぞれの意味表象を 20,000 回ずつ提示してトレーニングを行ったところ（Training Sweeps: 320,000）、図２のエラー曲線が示すように、ある時点から急に RMS エラーの値が小さくなり、学習が成立している様子が見られた。そこで、10 試行ごとのウェイトファイルを指定して、どの段階で全ての意味表象に対し、正しい音韻表象が出力できるようになったのかを特定したところ 220,000 sweeps からであることがわかった。つまり、16 ある入力信号のそれぞれが 13,750 回の試行を経た段階から全ての意味表象に対し、日本語で正しい音韻表象が出力できるようになったと言える。

### 3.2　母語の習得Ⅱ（母語：英語）

ここでは、英語を母語として習得する場合を想定したシミュレーションとして、英語の動詞８語の現在形と過去形それぞれの意味表象に対応する音韻表象を学習させる。本研究で行われるシミュレーションは、全て同一の被験者によってなされることを想定しているので、ランダムシード、学習率、慣性項の値を統一する（ランダムシード：9、学習率：0.5、慣性項：0.8）。入力信号を一巡するごとに RMS エラーのログをとるよう当該の sweep 数を入力する（Log error every 16 sweeps）。さらに、入力信号それぞれに対し 10 回の試行を行うごとにウェイトファイ

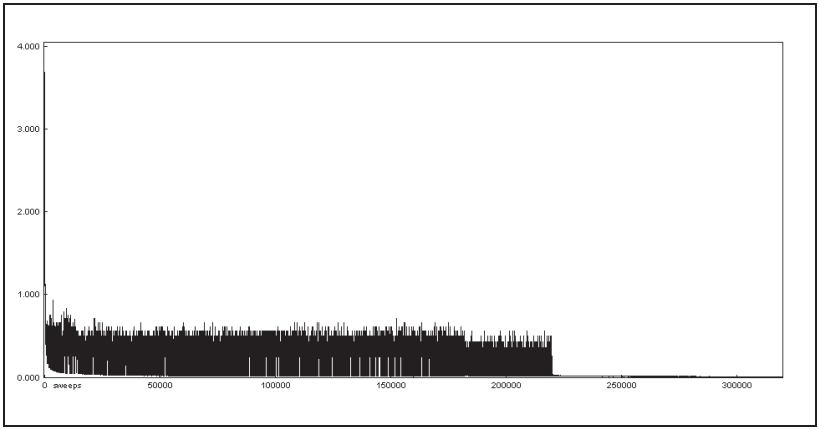


図2 母語としての日本語の学習（RMS エラー曲線）

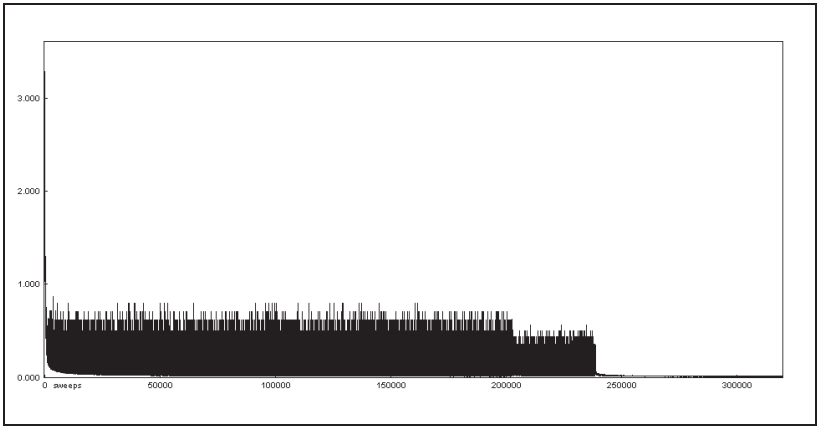


図3 母語としての英語の学習（RMS エラー曲線）

ルを記録していくよう当該のsweep数を入力する（Dump weights every 160 sweeps）。

ネットワークに、これら英語の動詞８語の現在形と過去形それぞれの意味表象を20,000回ずつ提示してトレーニングを行ったところ（Training Sweeps: 320,000）、図３のエラー曲線が示すように、ある時点から急にRMS エラーの値が小さくなり、学習が成立している様子が見られた。そこで、10 試行ごとのウェイトファイルを指定して、どの段階で全ての意味表象に対し、正しい音韻表象が出力できるようになったのかを特定したところ 238,720 sweeps からであることがわかった。つまり、入力信号のそれぞれが 14,920 回の試行を経た段階から全ての意味表象に対し、英語で正しい音韻表象が出力できるようになったと言える。

### 3.3　バイリンガルの習得

ここでは、日本語と英語の両方を母語として習得する場合を想定したシミュレーションとして、動詞８語の現在形と過去形それぞれの意味表象に対応する音韻表象を日本語と英語で学習させる。本研究で行われるシミュレーションは、全て同一の被験者によってなされることを想定しているので、ランダムシード、学習率、慣性項の値を統一する（ランダムシード：9、学習率：0.5、慣性項：0.8）。入力信号を一巡するごとにRMS エラーのログをとるよう当該の sweep 数を入力する（Log error every 32 sweeps）。さらに、入力信号それぞれに対し 10 回の試行を行うごとにウェイトファイルを記録していくよう当該の sweep 数を入力する（Dump weights every 320 sweeps）。

ネットワークに、これらの動詞８語の現在形と過去形それぞれの意味表象を日本語と英語で 20,000 回ずつ提示してトレーニングを行ったところ（Training Sweeps: 640,000）、図４のエラー曲線が示すように、ある時点から急に RMS エラーの値が小さくなり、学習が成立している様子が見られた。そこで、10 試行ごとのウェイトファイルを指定して、どの段階で全ての意味表象に対し、正しい音韻表象が出力できるようになったのかを特定したところ 621,760 sweeps からであることがわかった。つまり、各入力信号 19,430 回の試行を経た段階から全ての意味表象に対し、日本語と英語で正しい音韻表象が出力できるようになったと言える。

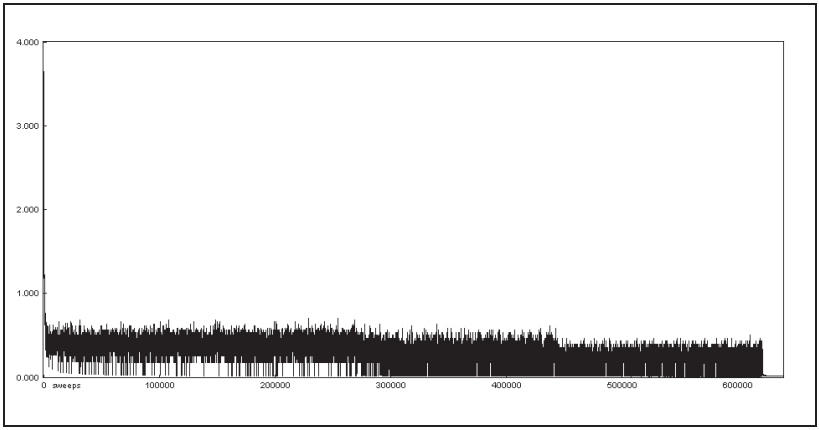


**図4　バイリンガルの学習（RMS エラー曲線）**

### 3.4　第二言語の習得Ⅰ（母語：日本語、第二言語：英語）

このシミュレーションでは、母語として日本語を習得した後に第二言語として英語を習得する場合を想定し、動詞８語の現在形と過去形それぞれの意味表象に対応する音韻表象を日本語で正しく出力できるようになった後、英語でも正しく出力できるようになることを目指す。具体的には、日本語での学習が完了した時点でのユニット間の結合強度を初期値として、バイリンガルの習得の際に用いたデータファイルを使ってトレーニングを行う。この場合も、本研究で行われるシミュレーションは全て同一の被験者によってなされることを想定しているので、ランダムシード、学習率、慣性項の値を統一する（ランダムシード：9、学習率：0.5、慣性項：0.8）。入力信号を一巡するごとに RMS エラーのログをとるよう当該の sweep 数を入力する（Log error every 32 sweeps）。さらに、入力信号それぞれに対し 10 回の試行を行うごとにウェイトファイルを記録していくよう当該の sweep 数を入力する（Dump weights every 320 sweeps）。

トレーニング実施の際のオプション設定で“Load Weights File”を選

択し、母語として日本語での学習が完了した時点でのユニット間の結合強度（220,000 sweeps のウェイトファイル）を指定した後、ネットワークに、当該の動詞８語の現在形と過去形それぞれの意味表象を日本語と英語で 20,000 回ずつ提示してトレーニングを行ったところ（Training Sweeps: 640,000）、図５のエラー曲線が示すように、ある時点から急に RMS エラーの値が小さくなり、学習が成立している様子が見られた。そこで、10 試行ごとのウェイトファイルを指定して、どの段階で全ての意味表象に対し、正しい音韻表象が出力できるようになったのかを特定したところ延べ 597,920 sweeps（正味 377,920 sweeps）からであることがわかった。つまり、各入力信号 11,810 回の試行を経た段階から全ての意味表象に対し、日本語と英語で正しい音韻表象が出力できるようになったと言える。

### 3.5　第二言語の習得Ⅱ（母語：英語、第二言語：日本語）

このシミュレーションでは、母語として英語を習得した後に第二言語として日本語を習得する場合を想定し、動詞８語の現在形と過去形それぞれの意味表象に対応する音韻表象を英語で正しく出力できるようになった後、日本語でも正しく出力できるようになることを目指す。具体的には、英語での学習が完了した時点でのユニット間の結合強度を初期値として、バイリンガルの習得の際に用いたデータファイルを使ってトレーニングを行う。この場合も、本研究で行われるシミュレーションは全て同一の被験者によってなされることを想定しているので、ランダムシード、学習率、慣性項の値を統一する（ランダムシード：9、学習率：0.5、慣性項：0.8）。入力信号を一巡するごとに RMS エラーのログをとるよう当該の sweep 数を入力する（Log error every 32 sweeps）。さらに、入力信号それぞれに対し 10 回の試行を行うごとにウェイトファイルを記録していくよう当該の sweep 数を入力する（Dump weights every 320 sweeps）。

トレーニング実施の際のオプション設定で “Load Weights File” を選択し、母語として英語での学習が完了した時点でのユニット間の結合強度（238,720 sweeps のウェイトファイル）を指定した後、ネットワークに、当該の動詞８語の現在形と過去形それぞれの意味表象を日本語と英

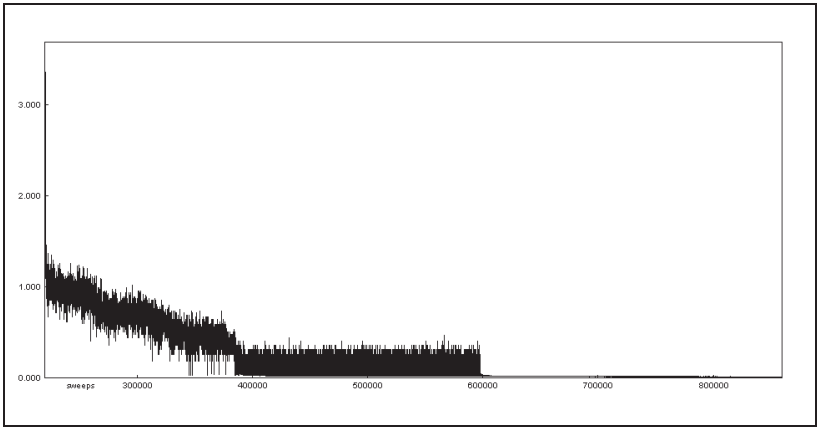


図5 第二言語としての英語の学習（RMS エラー曲線）

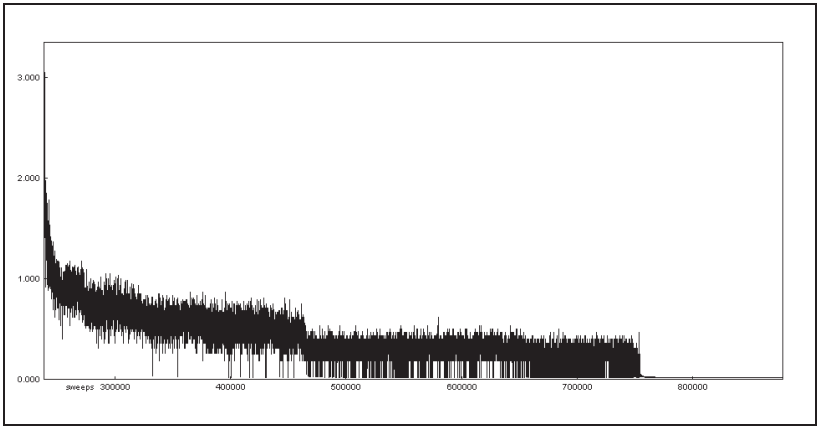


図6 第二言語としての日本語の学習（RMS エラー曲線）

語で 20,000 回ずつ提示してトレーニングを行ったところ（Training Sweeps: 640,000）、図 6 のエラー曲線が示すように、ある時点から急に RMS エラーの値が小さくなり、学習が成立している様子が見られた。そこで、10 試行ごとのウェイトファイルを指定して、どの段階で全ての意味表象に対し、正しい音韻表象が出力できるようになったのかを特定したところ延べ 754,880 sweeps（正味 516,160 sweeps）からであることがわかった。つまり、各入力信号 16,130 回の試行を経た段階から全ての意味表象に対し、日本語と英語で正しい音韻表象が出力できるようになったと言える。

## 4．考察

本研究では、意味表象を軸とした共通のフォーマットを用いることによって、実際の発話行為を想定した母語と第二言語による過去時制の学習を模擬した。ここでは、同一の被験者を想定して実施した、母語として日本語や英語を学ぶ場合と、バイリンガルとして日本語と英語の両方を学ぶ場合、また、第二言語として英語や日本語を学ぶ場合のシミュレーションの結果を比較、考察する。

まず、母語として日本語を学ぶ場合については、各入力信号 13,750 回の試行を経た段階から 16 ある全ての意味表象に対し日本語で正しい音韻表象が出力できるようになり、母語として英語を学ぶ場合については、各入力信号 14,920 回の試行を経た段階から全ての意味表象に対し英語で正しい音韻表象が出力できるようになった。つまり、母語として見た場合には、英語よりも日本語の方が習得しやすい可能性が示唆されたが、その差はわずかであった。

また､バイリンガルとして日本語と英語の両方を学ぶ場合については、日本語と英語で正しい音韻表象が出力できるようになるのに各入力信号 19,430 回の試行を要した。日本語・英語とも母語として学習する場合に比べ、学習完了の段階が遅くなっているが、これは一度に二つの言語を学習するので母語の学習よりも負担が大きくなったためと考えられる。

一方、母語として日本語の学習が完了した後に第二言語として英語を学習する場合は、各入力信号 11,810 回の試行を経た段階から、英語を母語として学習した後に日本語を第二言語として学習する場合は、各入力

信号 16,130 回の試行を経た段階から、日本語と英語で正しい音韻表象が出力できるようになっている。いずれの場合も、バイリンガルの学習（19,430 回）より学習完了の段階が早くなっているが、これは、すでに母語として一つの言語については学習を完了しているためであると考えられる。しかしながら、母語による学習と比較すると、第二言語として英語を学ぶ場合（11,810 回）については、母語として英語を学ぶ場合（14,920 回）よりも学習完了の段階が早くなっているので、すでに母語として学習している日本語が促進効果の役割を果たしていると言えるが、第二言語として日本語を学ぶ場合（16,130 回）については、母語として日本語を学ぶ場合（13,750 回）よりも学習完了の段階が遅くなっているので、すでに母語として学習している英語がむしろ干渉効果の役割を果たしていると言える。すでに学習している母語によって、果たす役割が変わってしまうという結果は大変興味深い。

今回のシミュレーションは、極端に単純化された意味表象のフォーマットを限定された動詞の範囲内で適用して実施したものであり、本研究の結果のみで一般的な結論を語るのは早計かもしれない。今後は、より精緻な定義づけに基づいた意味表象のフォーマットを適用するなど、様々な観点からのシミュレーションを積み上げていくことが望まれる。

## 注

1　今回のシミュレーションでは、英語の動詞の現在形について、主語が三人称単数の場合に“-(e)s”を付加する語形の変化は、主語を特定できないため学習の対象から除外している。そのため現在形は、原形（語幹）と同じ形になっている。

## 参考文献

石崎　貴士. 2011. コネクショニズムを応用した第二言語習得研究の試み：第二言語による過去時制の学習. 山形大学地域教育文化学部英語教育講座『山形英語研究』第 12 号. 1-23.

守 一雄. 2002. コネクショニストモデルシミュレータ *tlearn* を使った心理学実験実習課題―対称性の学習における中間ユニットの数と学習率の効果―. 信州大学教育学部附属教育実践総合センター紀要『教育実践研究』No.3. 171-180.

McLeod, P., Plunkett, K. & Rolls, E. T. 1998. *Introduction to Connectionist Modeling of Cognitive Processes*. Oxford: Oxford University Press.（深谷 澄男（監訳）2005.『認知過程のコネクショニスト・モデル』北樹出版.）

Plunkett, K. & Elman, J. L. 1997. *Exercises in Rethinking Innateness*. Cambridge, MA: MIT Press.

Rumelhart, D. E. & McClelland, J. L. 1986. On learning the past tenses of English verbs. In D. E. Rumelhart, J. L.McClelland & the PDP Research Group, Eds. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 2, pp. 216-271. Cambridge, MA: MIT Press.