# Level3

045713C:大城 和也
045709E:上原直久

提出日:平成 18 年 6 月 10 日 ()

# 1　文字認識プログラム

## 1.1　Level3.1-サンプルプログラムの動作

中間層を 10 に変更し，学習を行った後，eva1-1.txt と eva1-2.txt の認識ができるかを確認した．eva1-1.txt は適切に認識できていたが，eva1-2.txt は誤差が大きくて全く認識できていなかった．これは学習したものとは座標が大きく違うためだと考えられる．このようなものの認識させるアプローチについては Level3.5 にて述べる．

## 1.2　Level3.2-効率の良いパラメータの設定

効率のいいパラメータを探した結果，以下の表 1 のような値を選択し，大体の最適値とした．

表 1: 最適値

| パラメータ | 値 |
|---|---|
| ETA(学習係数) | 1.9 |
| ALPHA(慣性項) | 0.3 |
| 中間層のニューロン数 | 35 |

ニューラルネットワークにおける中間層のニューロンの数，学習係数や慣性係数などのパラメータにはまだ一般的な決定方法は知られていない．よって普通は適切な値を手探りで探すというプロセスを踏む．今回，私たちの班が出した方法は中間層のニューロンを変えながら for 文を使って ETA と ALPHA の組み合わせを探索した．つまるところ全探索のようなもので値を探すという手法を用いた．

誤差が 0.0001 以下になるまでにかかる学習回数の平均は 667 回 (10 回の平均) であった．実際は ALPHA の値を大きくする事でもっと早く収束できたが，その場合には違う誤差で収束することが多々あり，効率的とは考えられなかったため除外した．

## 1.3　Level3.3-学習曲線

以下の図 1〜図 4 はそれぞれ ETA(学習係数) と ALPHA(慣性項) を変化させた時の誤差の学習曲線である．図 1，図 2 は ETA は同一だが ALPHA の値を大きく変えており，図 3，図 4 は ALPHA は同じ値だが ETA を変化させている．

まず，図 1 と図 2 とを比べてみる．ALPHA が 0.9 である図 1 は所々で収束の様子を見せ，結局約 0.3 辺りで収束しているように見える．一方，ALPHA を 0.0 とした図 2 ではさきほどよりも幾分滑らかであり，そして大体 0(本来は 0.0001) に収束している事が分かる．これは ALPHA，つまりは慣性項が影響している．慣性項は前の値との慣例性を示す数値でこれが高いほど前回の数値を考慮して重みの変化を行うようになる．これにより，精度の高い学習を目指すのだがその値を大きくとりすぎると目標の誤差に辿りつくまでにはかなりの回数を重ねる必要がでてくると考えられる．

次に図 3 と図 4 を比較してみる．図 3 は ETA を 1.9 にしており，図 4 は ETA を 0.1 とおいている．図をみれば分かるように ETA が小さい時の方がグラフが滑らかであるが，ETA が大きいときよりは収束の速度は遅い．ETA は学習係数と呼ばれるものであり，これは重み変化の度合いのようなものである．よって値を大きくすると変化の割合が大きくなり早く収束するようになると考えられる．図**??**で 600 少し過ぎた所が切れているのは誤差の値が 0.0001 を下回って学習が終了しているためである．

図 5 は今までの考えを踏まえて組み合わせたパラメータで level3.2 で示したものである．グラフは 600 後半で終了しており，収束へのスピードも先ほどのグラフと比べて早い事が分かる．

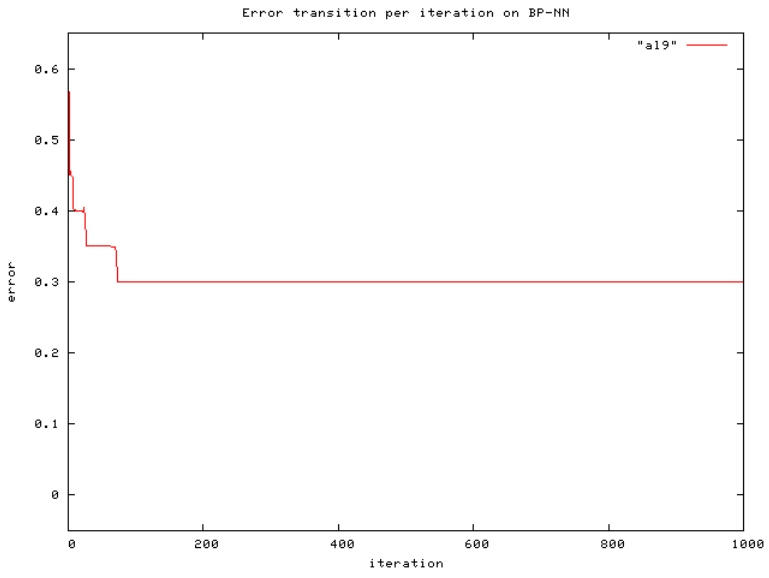


図 1: ETA(1.0), ALPHA(0.9) での学習曲線

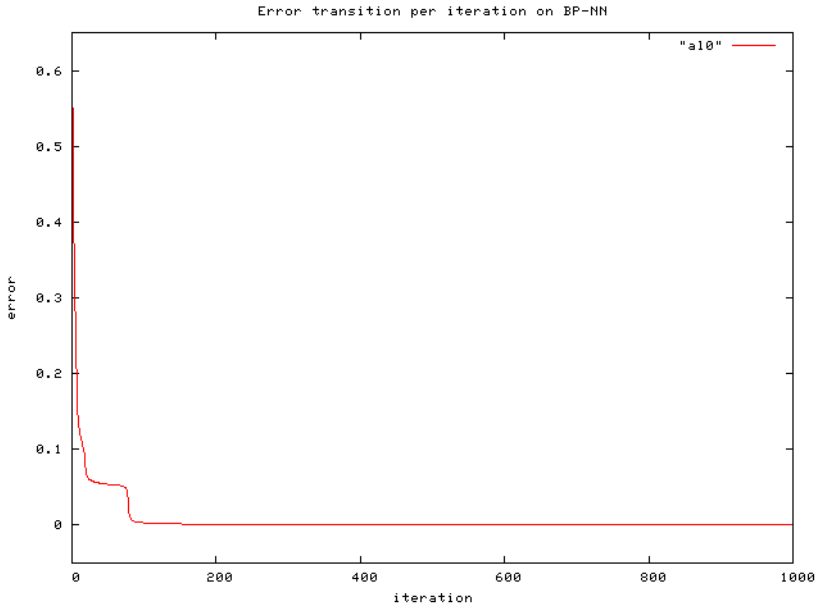


図 2: ETA(1.0), ALPHA(0.0) での学習曲線

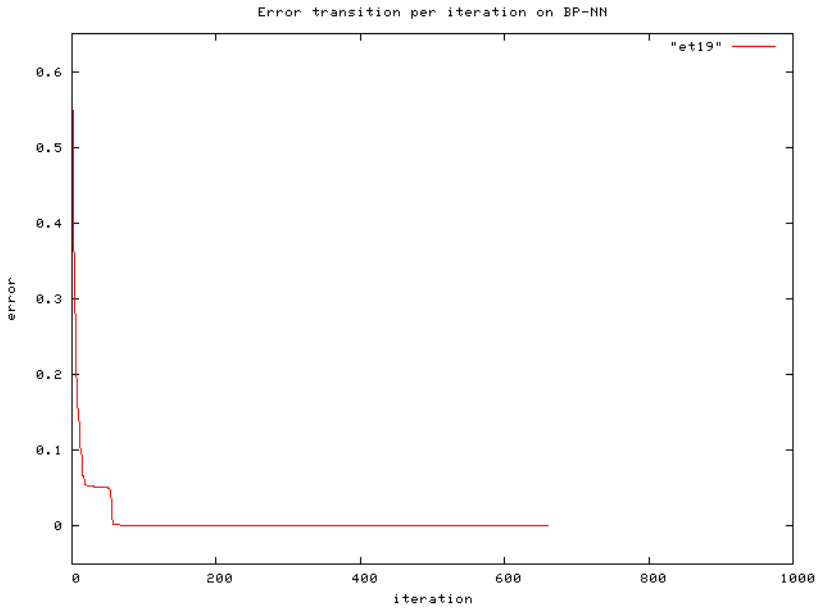


図 3: ETA(1.9), ALPHA(0.3) での学習曲線

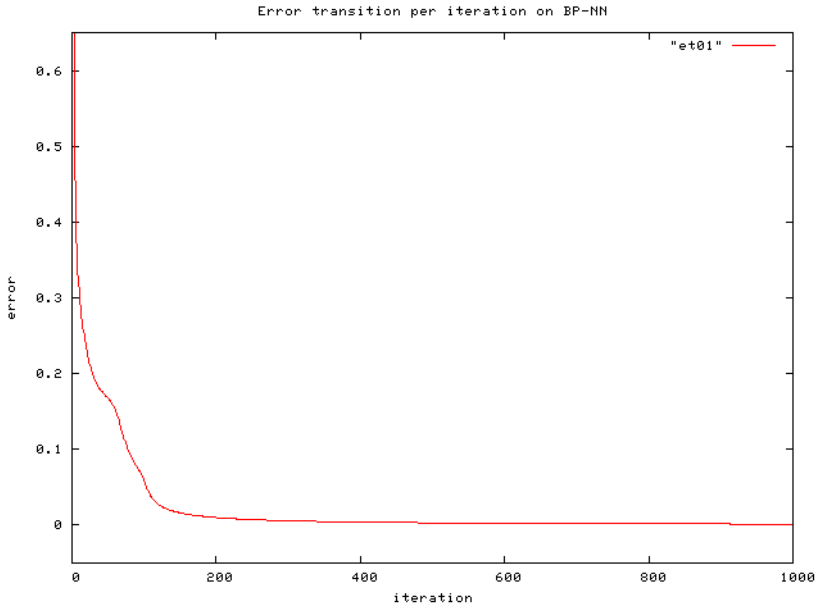


図 4: ETA(0.1), ALPHA(0.3) での学習曲線

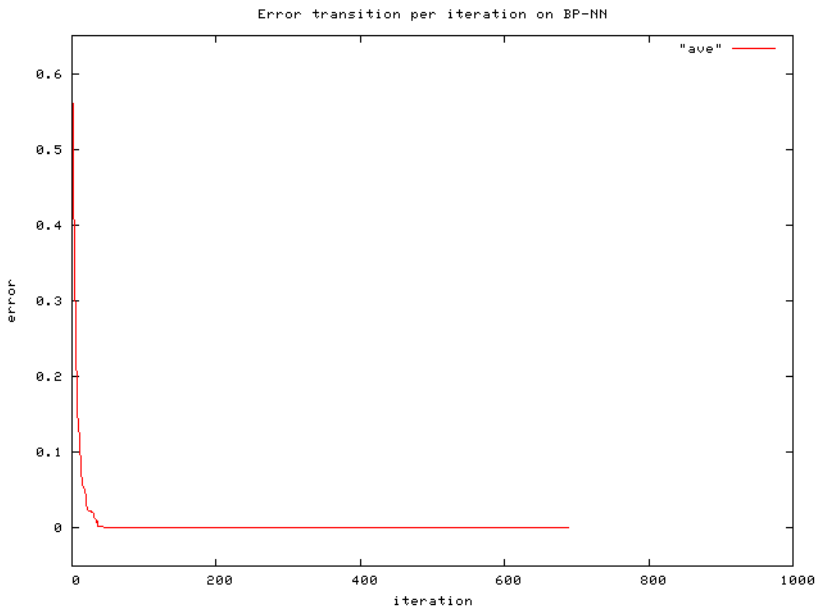


図 5: Level3.2 でのパラメータ設定での学習曲線

## 1.4 Level3.4-オリジナルデータの評価

### 1.4.1 学習用のデータ

以下に学習用のデータを示す．今回はlearn4.txtである4のデータを用いた．

4
0000110000
0001010000
0010010000
0100010000
1000010000
1111111111
0000010000
0000010000
0000010000
0000010000

### 1.4.2 オリジナル評価用データ

以下の4つがオリジナルで作成した4の評価用データである．それぞれの評価用データは学習用データとの違いが少ないものから並べている．data1は中心の縦が3つ欠けたもの．data2は左右と下が1つずつ欠け，さらに1つの値を交代している．data3は右の1列を左に移し替えたもので，data4は全てを反転したデータとなっている．

```
-------------------------------------------------------------------------
      4
      0000100000      0000110000      0000011000      1111001111
      0001000000      0001010000      0000101000      1110101111
      0010000000      0010010000      0001001000      1101101111
      0100010000      0100010000      0010001000      1011101111
      1000010000      0100010000      0100001000      0111101111
      1111111111      0111111110      1111111111      0000000000
      0000010000      0000010000      0000001000      1111101111
      0000010000      0000010000      0000001000      1111101111
      0000010000      0000010000      0000001000      1111101111
      0000010000      0000000000      0000001000      1111101111

        data1           data2           data3           data4
-------------------------------------------------------------------------
```

### 1.4.3 評価結果

以下がそれぞれを評価した結果である．

```
++++++++++++++++++++++++++++++++++++++++++++++++++ data1_result +++++++++
CHECK filename data/data1.txt
EVA o[0] = 0.00269, correct[0] = 0.0
EVA o[1] = 0.00103, correct[1] = 0.0
EVA o[2] = 0.00086, correct[2] = 0.0
```

```
EVA o[3] = 0.00028, correct[3] = 0.0
EVA o[4] = 0.97218, correct[4] = 1.0
EVA o[5] = 0.00019, correct[5] = 0.0
EVA o[6] = 0.03799, correct[6] = 0.0
EVA o[7] = 0.00147, correct[7] = 0.0
EVA o[8] = 0.00100, correct[8] = 0.0
EVA o[9] = 0.00185, correct[9] = 0.0
EVA sum_error = 0.07516
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++++ data2_result +++++++++
CHECK filename data/data2.txt
EVA o[0] = 0.00153, correct[0] = 0.0
EVA o[1] = 0.02373, correct[1] = 0.0
EVA o[2] = 0.00359, correct[2] = 0.0
EVA o[3] = 0.00007, correct[3] = 0.0
EVA o[4] = 0.94470, correct[4] = 1.0
EVA o[5] = 0.00094, correct[5] = 0.0
EVA o[6] = 0.00379, correct[6] = 0.0
EVA o[7] = 0.01347, correct[7] = 0.0
EVA o[8] = 0.00171, correct[8] = 0.0
EVA o[9] = 0.01541, correct[9] = 0.0
EVA sum_error = 0.11954
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++++ data3_result +++++++++
CHECK filename data/data3.txt
EVA o[0] = 0.00099, correct[0] = 0.0
EVA o[1] = 0.00020, correct[1] = 0.0
EVA o[2] = 0.00272, correct[2] = 0.0
EVA o[3] = 0.00127, correct[3] = 0.0
EVA o[4] = 0.45781, correct[4] = 1.0
EVA o[5] = 0.00046, correct[5] = 0.0
EVA o[6] = 0.09912, correct[6] = 0.0
EVA o[7] = 0.00306, correct[7] = 0.0
EVA o[8] = 0.01236, correct[8] = 0.0
EVA o[9] = 0.20276, correct[9] = 0.0
EVA sum_error = 0.86513
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++++ data4_result +++++++++
CHECK filename data/data4.txt
EVA o[0] = 0.06185, correct[0] = 0.0
EVA o[1] = 0.00062, correct[1] = 0.0
EVA o[2] = 0.03595, correct[2] = 0.0
EVA o[3] = 0.10192, correct[3] = 0.0
EVA o[4] = 0.00001, correct[4] = 1.0
EVA o[5] = 0.00816, correct[5] = 0.0
EVA o[6] = 0.00088, correct[6] = 0.0
EVA o[7] = 0.00602, correct[7] = 0.0
EVA o[8] = 0.04372, correct[8] = 0.0
EVA o[9] = 0.12988, correct[9] = 0.0
EVA sum_error = 1.38900
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

### 1.4.4 考察

評価結果のそれぞれの sum_error の値は $0.07516 < 0.11954 < 0.86513 < 1.369$，つまり $data1 < data2 < data3 < data4$ と昇順になっている．さきほども述べた通り，今回は data1 から順に違いが大きくなるようになっている．そのため，結果の誤差も後になるほど大きくなったと考えられ，その点ではちゃんと学習していると言えよう．

では次に各パラメータを見てみる．data1 では EVA o[4] の値が 0.97218 となっておりこの程度の違いならしっかりと認識されていることがわかる．次

に data2 では EVA o[4] の値が 0.94470，data1 より劣るものの認識率は高い．一方，data3 では辛うじて o[4] の値が一番高くなっているがその値は 0.45781 と低い，この data3 は学習用のデータを 1 つずらしただけだがこのような結果となっている．これより，今回の学習プログラムはその座標に座標の評価が大切であり，その形のパターンなどを見て判断してない事が分かる．このことは data4 を見ても言える事である．data4 はビット反転を行ったデータを使っているため，o[4] の値は 0.00001 と最も低くなっており，全ての値に対して違う値をとったため極端に低い値が出力されたと考えられる．

## 1.5 Level3-5 情報が欠落している文字の認識率を高めるための工夫

### 1.5.1 アフィン変換を用いた文字座標の変換

文字の大きさや位置がずれて認識ができなくなってしまうのは，文字の情報をマスが 0(白) か 1(黒) がどうかで判断しているためである．このため，文字の位置 (座標) がずれると，本来 1 であるマスが 0 になっているために認識できないことがある (図 6 参照).

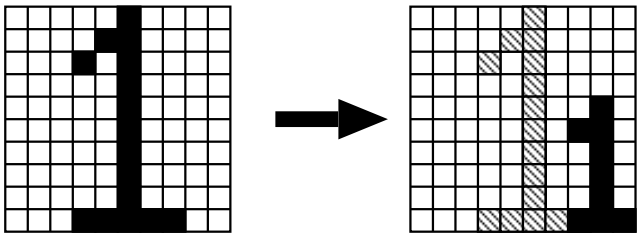

図 6: 位置が違う場合

そこで，入力された文字をアフィン変換する．アフィン変換とは，図形を変形させないで平行移動や回転，拡大縮小等を用いて変換する方法である．これで，入力した文字をある特定の大きさの領域に合わせて変換を行えばよい．

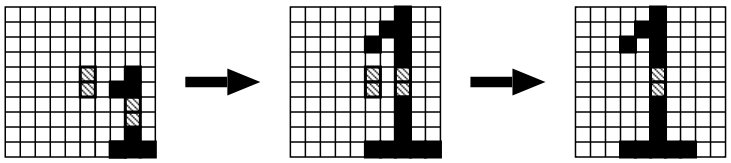

図 7: 重心を合わしていく様子

図 7 を用いて説明する．図 6 の矢印の左側をサンプルデータとして学習させた場合，領域の中央の斜線マスがサンプルデータの重心となる．次に，アフィン変換を用いて読み込む数字を拡大する．あとは，その読み込む数字の重心を学習したサンプルデータの重心に合わせればよい (平行移動)．

このアフィン変換は，文字の書式が同じ場合，サイズや位置の違いなどを変換できるが，書体が変わると認識できないと考えられる．また，このようにサイズや位置を調整することを正規化という．

### 1.5.2 一般的な文字認識

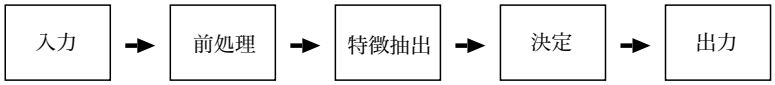


図 8: パターン認識の過程

一般的な文字認識は，図 8 に示したフローチャートで行われる．

前処理は，文字を読み込むとき，スキャナ等で読み込むときにインクの染みや紙のしわなどの文字認識に邪魔になるノイズを取り除く作業である．

正規化は，位置や文字の大きさを認識しやすいように統一する作業である．正規化には，線形正規化と非線形正規 化がある。線形正規化は、単純な拡大縮小を行って大きさを合わせるのに対して非線形正規化は，部分的に拡大縮小をしする．非線形正規化は，手書き文字認識に有効で，人の文字の癖をある程度吸収できる．アフィン変換は線形正規化となる．

次に特徴抽出は，文字の特徴を抽出する．特徴の抽出方法としては，パターンマッチング法，ゾンデ法，ストロークアナリシス法などがある．その方法を簡単に示すと以下の通りになる．

**パターンマッチング法**

予め読もうとする文字の様々なパターンを用意して同じ文字で共通の特徴を見つけ，それを学習させることで，特定の文字を識別する．

**ゾンデ法**

ある固定された出発点からゾンデ (図 9) と呼ばれる触手みたいなものを何本か伸ばし，文字線との衝突状況を文字の特徴として認識する．

**ストロークアナリシス法**

文字を構成する線から基本的な線を集めて，その基本的な線がどういう組み合わせで構成されているかによって文字を認識する．

最後に特徴量を用いて文字を決定する．

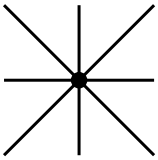

図 9: ゾンデの例

## 参考文献


[1] ”http://pooh.fukushima-nct.ac.jp/nakaolab/sotuken/moji/MOJI_TOP.HTML”
可変ゾンデ法による文字認識アルゴリズム

[2] ”http://metalbrain.piroo.com/”
Metal Brain

[3] ”http://cis.k.hosei.ac.jp/w̃akahara/patrec_2.pdf”
第 2 回講義 「パターン認識 」

[4] ”www.hokkaido-iri.go.jp/book/reports/291/0302TB0120.pdf”
ニューラルネットワーク