ニューラルネット

Report1

学籍番号 045713C:大城和也

提出日:平成 18 年 07 月 25 日 (火)

1　課題内容

サンプルプログラムを用いて、Ex-or の学習を行う.

- 逐次修正法
- 逐次修正法＋慣性項
- 一括修正法

について，それぞれ特徴を指摘せよ.

プログラムを変更して，隠れ層の unit 数を増やし，その影響を調べよう.

オリジナルの問題を設計し，問題設計の意図，入出力データ，学習に用いるニューラルネットの構造について議論せよ．また，可能であれば，その問題についてニューラルネットワークの学習実験を行い，結果を評価せよ.

2　報告事項

2.1　Ex-or 学習

2.1.1　逐次修正法

逐次修正法とはある与えられた値での出力結果を学習の元となる教師信号との誤差を逆伝搬して各層のユニットの誤差を計算し，重みの修正を行う．これを誤差が小さくなるまで行うことで学習するといったものだ.
流れとしては以下のようになる.

1. 入力パターンを提示し，それに対する出力結果を求める.
2. 出力結果と教師信号との誤差を逆伝播しながら，各層の各ユニットにおける誤差を求める.
3. その誤差をもとに結合荷重の修正量を計算し，結合荷重の修正を行う.
4. 以上の処理を全入力パターンに対して行い，さらに，何度も同じ処理を繰り返す.

2.1.2　逐次修正法＋慣性項

逐次修正法・一括修正法では,「二乗誤差 E の微分値」に比例した値だけによって結合荷重の修正量が決定する．一般に，このような勾配降下法では，修正量は微小な値が良いとされている．しかし，修正量を小さくすると，学習

速度が遅くなる．そこで，探索点の動きが振動しない程度に学習速度を上げる一つの方法として，慣性項を導入して結合荷重の修正を行う．

慣性項を用いた逐次修正法は前回の結合荷重が同じ方向なら大きくし，違う方向なら小さくするといった前回の変更量を考慮して荷重の変更を行う．そのため，最初は変更量を大きくし，徐々に小さくできるため，学習率をある程度大きくしても振動せずに収束できるのである．

以下の図 1，図 2 は学習の隠れ層を 7 に修正値を 15 に設定した時の慣性項を用いたときと用いないときの学習のグラフだ．慣性項がなかった場合，全然学習する事はない．しかし，慣性項があった場合は逆に学習が早くなっているのがわかる．

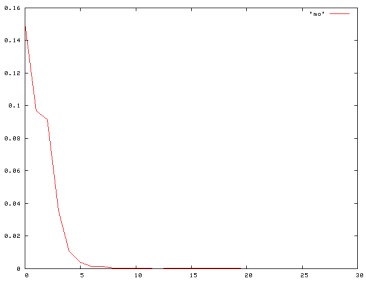

図 1: 慣性項有り

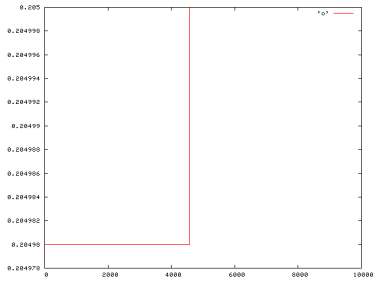

図 2: 慣性項無し

2.1.3 一括修正法

逐次修正法で各入力データごとにパラメータを修正していたのに対して，一括修正法では各入力データごとの修正量を他に記憶しておき，一通りの計算を終えた後に重みの修正を行う．各入力データごとの修正が必要ないために，逐次修正法と比べると高速な学習が可能となっている．

流れとしては以下のようになる．

1. 入力パターンを提示し，それに対する出力結果を求める．
2. 出力結果と教師信号との誤差を逆伝播し，結合荷重の修正量を求め蓄積する．
3. 以上のことを，全入力パターンに対して行い，その後にそれまでに蓄積された結合荷重の修正量をもとに，結合荷重の修正を行う．
4. これらの処理を誤差が小さくなるまで繰り返す．

2.2 隠れ層の効果

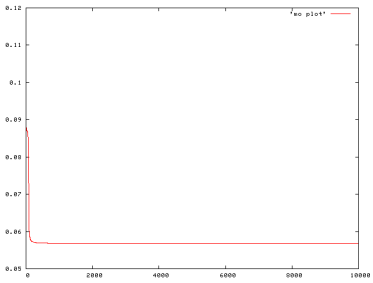

図 3: 隠れ層:1

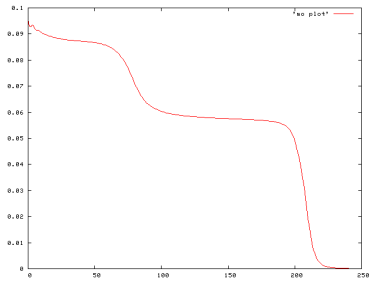

図 4: 隠れ層:2

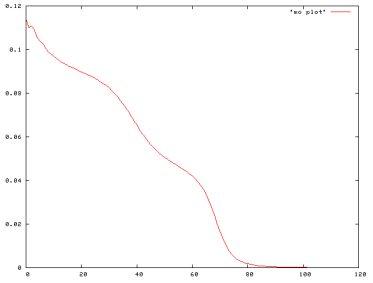

図 5: 隠れ層:7

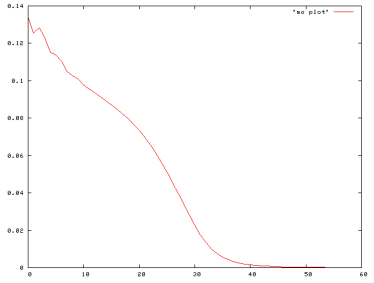

図 6: 隠れ層:17

上の図は ex_bp_mo.c のプログラムの隠れ層を変化させたものである．図を見ればわかるように隠れ層を増やす事によって収束に至る早さが早くなり，その収束のしかたも安定していく．隠れ層は増やす事はつまりその重みの変化が増え，複雑に学習できるという事である．図 1 のように隠れ層が一つのときに exor を学習できないのはこれが関係している．つまり，今回の場合は 2 つ以上の隠れ層のニューロンを準備すればよいが，さらに増やす事によって学習が高速に行えるのである．しかし，増やしすぎると計算時間が膨大になるので隠れ層の設定はある程度いい精度の値を手当たりで見つける必要が有る．

2.3 オリジナル問題

ニューラルネットが得意とする問題は数式計算などの厳密な処理を行う分野ではなく，認識や予測，分析などの曖昧な処理である．例で与えられたプログラムは学習を行う事によって，入力による出力の予測を行っているとも考えられる．

今回はオリジナル問題として，NOR の学習をさせたいと思う．そのために ex_bp_mo.c のプログラム中にある教師信号を以下のように変化させた．

```
/* NOR Problem */
i_lay[0][0]=OFF; i_lay[0][1]=OFF; i_lay[0][2]=ON; teach[0][0]=ON;
i_lay[1][0]=ON;  i_lay[1][1]=OFF; i_lay[1][2]=ON; teach[1][0]=OFF;
i_lay[2][0]=OFF; i_lay[2][1]=ON;  i_lay[2][2]=ON; teach[2][0]=OFF;
i_lay[3][0]=ON; i_lay[3][1]=ON;  i_lay[3][2]=ON; teach[3][0]=OFF;
```

その時のグラフは以下のようになった．

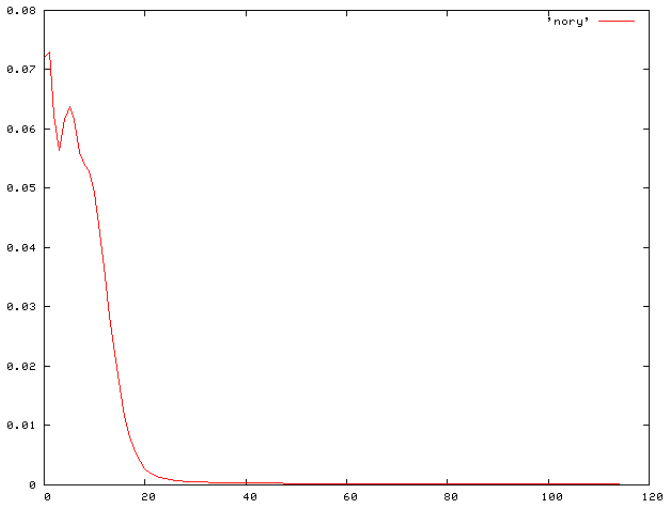

図 7: 学習曲線

ちゃんと学習している事が伺える．ちなみに，この問題は and や or の学習と同じく線形分離が可能なので隠れ層は 1 つでも学習が可能．実際，今回のプログラムでは隠れ層は 1 つとなっているが学習が行なえている．