ニューラルネットワークのオブジェクト化

Object-oriented design of neural network

発表者　大谷　将喜　　指導教員　坪井 一洋

１．はじめに

人間の脳の構造を模倣して作った情報処理機構がニューラルネットワークである. その構成要素であるニューロンは人間の脳に約 140 億個存在し, それぞれが複雑に相互結合している[1]. ニューロンには電位が存在し, この電位がある閾値を超えることで発火し, 結合する他のニューロンの電位を変化させる. このようなニューロンが多数存在し, ネットワークが構築される[2].

本研究では，オブジェクト指向モデリング[3]の考えに基づき, ニューラルネットワークのモデル設計およびシミュレーションを行うことを目的とする. 今回はニューラルネットワークを用いたパターン認識を例題として考える.

２．オブジェクト化の概要

2.1 分析・設計

ニューラルネットワークは複数の他のニューロンと関係性を持つことにより構築されている. ニューラルネットワークをオブジェクト化するにあたり, まずその構成要素であるニューロンのオブジェクト化を行う. このような観点から設計したクラス図を図１に示す.

今回の例題では，ニューロンはパターンの部分として, パターンはネットワークの部分として捉えることができるので, 集約の関係で表す. また，ニューロンは互いに結合するという関連を持つ.

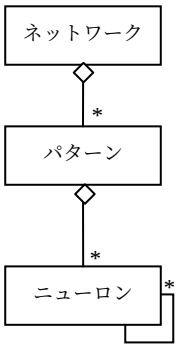

図１　クラス図

2.2 ニューロンクラス

ニューロンは自分自身が 1 or 0 のどちらをとるのか知る必要がある. 記憶パターンを読み込む時に, ニューロンはこの値を受け取る. その後, 必要な結合荷重を受け取り，自身の膜電位を更新する. 更新式は McCulloch-Pitts モデルを用いる[1]. 今回設計したニューロンクラスの概要を図 2 に示す.

ニューロン
・膜電位(u) ・入力(x) ・出力(y)
・初期の値を受け取る. ・自身を更新する. ・結合荷重を受け取る. ・y を出力 する.

図 2 ニューロンクラスの属性と操作

2.3 パターンクラス

複数のニューロンが集まってパターンを形成している. パターン内でネットワークエネルギーを計算し, その値を属性として持つ. ニューロン同士の結合の強度（結合荷重）をネットワーククラスから受け取り, ニューロンへ渡す必要がある. これらを包含するパターンクラスの概要を図 3 に示す.

パターン
・ニューロン ・パターン ・ネットワークエネルギー
・パターンを受け取る. ・ネットワークエネルギーを計算する. ・ニューロンを更新する. ・初期状態の値を渡す.

図 3 パターンクラスの属性と操作

2.4 ネットワーククラス

ネットワーククラスでは，記憶パターンを設定し, それらの結合荷重を計算する. また, 入力パターンと記憶パターンとのハミング距離を計算するメソッドが必要である. 入力パターンの更新回数をネットワーククラスで定義する. 概要を図 4 に示す.

ネットワーク
・パターン ・ハミング距離 ・結合荷重 ・更新回数
・パターンを設定する. ・結合荷重を計算する. ・ハミング距離を計算する.

図 4 パターンクラスの属性と操作

３．シミュレーション

3.1 問題設定

100 個のニューロン(10×10)で１つのパターンの場合で考える．C 言語での実行結果と，今回設計したクラスに基づいた java 言語での実行結果を比較する．比較は，入力パターンを更新し，その更新回数とどのようなパターンで平衡状態になるかで行う．

図 5 に示すように，記憶パターン A，B，C の 3 つを用意する．入力パターンと記憶パターンのハミング距離，ネットワークエネルギーを表 1 に示す．なお，C 言語，java 言語共に図 5 のパターンを用いてシミュレーションを行った．

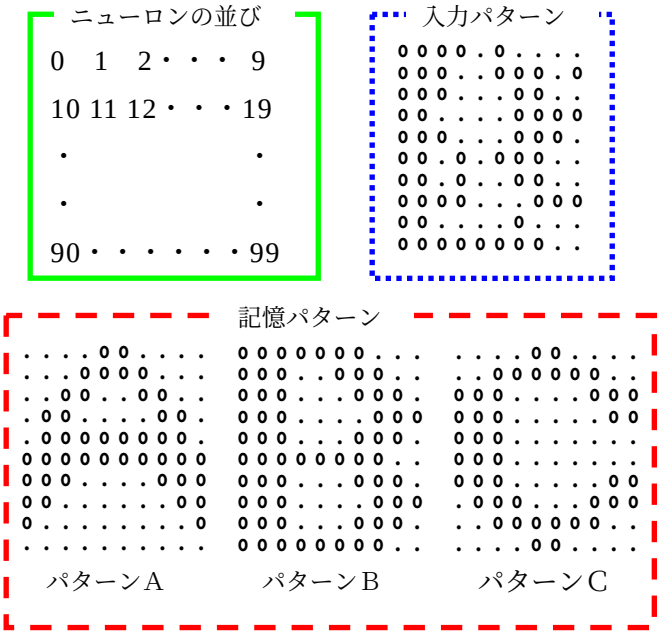

図 5 記憶パターンと入力パターン

表１．ハミング距離とネットワークエネルギー

パターン	ハミング距離	ネットワークエネルギー
A	46	-1198
B	15	-2044
C	44	-1302

3.2 ランダム更新

図 5 の入力パターンのニューロンの更新をランダムで行う．100 個あるニューロンの中から，ランダムに１つを選び膜電位の更新をする．この過程を１回の更新とする．また，今回のハミング距離は入力パターンとパターン B の距離とする．結果を図 6 に示す．

図 6 の結果では，C 言語と java 言語共に同じ記憶パターン B で平衡状態になった．この場合 java 言語のほうが更新回数が少なくなっているが，ランダムで行うため，ずれが生じる．しかし，50 回の試行による更新回数の平均をとった結果，表２のようになり，C 言語と変わらない更新回数で平衡状態になった．この結果から，今回実装した java プログラムの動作確認ができた．

表２．java・C 言語の平均更新回数(50 回)

	平均更新回数
java 言語	約 325 回
C 言語	約 332 回

3.3 正順更新と逆順更新

100 個あるニューロンを 0→99 番目まで１個ずつ順番に更新させる(正順更新)．また，逆に 99→0 番目の順番で更新させる(逆順更新)．結果を図 6 に示す．

図 6 の結果から，ランダムに更新するのではなく，正順，逆順更新共に記憶パターン B を想起することができた．更新する方向はどちらの方向でも変化は見られなかった．これは，入力パターンの状態に左右されるので，入力パターンによっては，差が生じると考えられる．

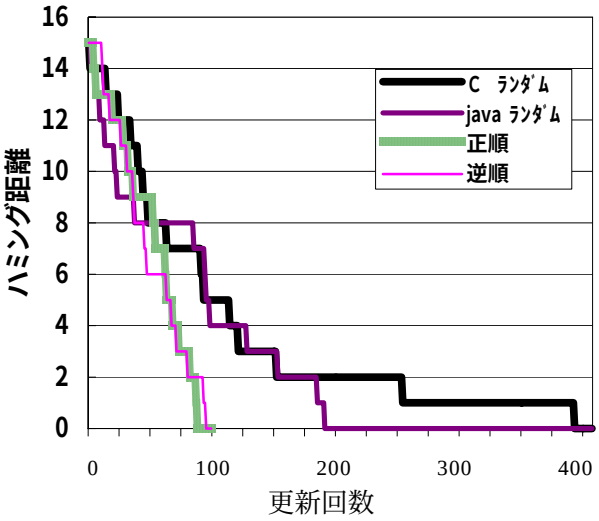

図 6 入力パターンのハミング距離の変化

４．まとめと今後の課題

パターン認識のためのニューラルネットワークを例題として，オブジェクト指向モデリングを用いたオブジェクト設計を行った．また，その設計に基づき java 言語で実装し，シミュレーションを行った．

今回のシミュレーションで，java 言語によって実装したプログラムと C 言語でのプログラムで同様の結果を得ることができた．

今回行ったオブジェクト化により，例えば java 言語のマルチスレッド機能を利用した並行処理を利用することができ，より現実に近いニューラルネットワークモデルを構築することができると思われる．

[参考文献]

[1]中野　馨：ニューロコンピュータの基礎
コロナ社（2003）
[2]O.Hoshino：Enhanced Sound Perception by Widespread-Onset Neuronal Responses in Auditory Cortex, Neural Comp.,19(2007)
[3]磯田　定宏：オブジェクト指向モデリング，
コロナ社（1998）