SuperCon2019 認定問題と予選問題

1　ディープニューラルネットワーク

ニューラルネットワーク (NN) を用いた**ディープラーニング**による機械学習が大きな話題である。NN は生物の脳の神経回路を模した情報処理システムの総称で、**ニューロン**と呼ばれる素子が互いに情報を伝え合うことにより、情報を記憶したりデータを分類したりできる。特にディープラーニングに用いられる**ディープニューラルネットワーク (DNN)** は、たくさんの**層**で構成される NN である。各層にはいくつものニューロンが備えられている。

層は入力層・中間層・出力層に分けられ、ニューロン間の繋がり (**結合**という) に応じて、データが順に伝えられる。まず、**入力層**のニューロンに入力データがセットされる。このデータが二層目の各ニューロンに伝えられる際にあとで説明するような情報処理が行われる。それがさらに三層目のニューロンへの入力となり、さらにそのまた次の層へとデータが順に伝えられていき、その過程で情報が処理される。最後に**出力層**のニューロンが持つ値を読み取る。このように複雑な情報処理を経て、入力が出力に変換される。使い道としては、たとえば、入力層に動物の画像データを入れると、それがイヌかネコか[*1]が出力されるような「分類器」が考えられる。中間層の数が多いものが DNN と呼ばれ[*2]、現在盛んに使われている。

もちろん、画像に写っているものがイヌかネコかを判断できるようにするためには、NN にイヌとネコの画像をたくさん与えて特徴を覚えさせなくてはならない。この手続きを NN の**学習**という。学習で決めるものはニューロン間の結合である。うまく学習データを与えて結合を適切に調節すれば、学習に用いなかった写真データを与えてもそれがイヌかネコかを正しく判定できるようになる。このように、初めて与えられる入力データに対しても正しい出力をすることを**汎化**という。汎化のためには学習に使うデータの組の選び方が重要である。NN の学習方法としては**バックプロパゲーション法**などが知られている。

今回の問題では小規模な DNN を取り上げる。小規模というのは入出力とも数ビットで中間層もせいぜい数層という意味である。この程度であればバックプロパゲーション法を使うまでもなく、ちょっと工夫すれば、ニューロン間の結合を手当たり次第に作ってみる方法でも学習できる。

今回使う NN では各ニューロンの取りうる値は 0 または 1 のいずれかとする。第 $n+1$ 層の i 番

[*1] 総称して**ラベル**と呼ばれる

[*2] といっても、中間層が 3 層もあれば DNN と呼ばれるようである

目のニューロンの値を $x_i^{(n+1)}$ と書こう。これは第 n 層のニューロンが持つ値によって以下のように決まる。

$$x_i^{(n+1)} = \theta\left(\sum_j J_{ij}^{(n+1,n)} x_j^{(n)}\right) = \theta\left(J_{i1}^{(n+1,n)} x_1^{(n)} + J_{i2}^{(n+1,n)} x_2^{(n)} + \cdots + J_{iN}^{(n+1,n)} x_N^{(n)}\right)$$

ただし、第 n 層のニューロンの数を N とした。また、$\theta(y)$ は階段関数と呼ばれる関数で、$y > 0$ のときには 1、$y \leq 0$ のときには 0 を返す。さらに、問題を簡単にするために、結合 $J_{ij}^{(n+1,n)}$ は $+1$ または -1 のいずれかの値にしかならないものとしよう。

つまり、第 n 層のすべてのニューロンの値に結合 $J_{ij}^{(n+1,n)}$ を掛けて足し合わせ、それが正なら $x_i^{(n+1)}$ は 1、0 以下なら 0 となる。まず入力層 (第 1 層) のすべてのニューロンの値 (すべての $x_i^{(1)}$) を入力データに従ってそれぞれ 0 または 1 にセットし、次に上の操作を第 2 層のすべてのニューロンに対して行なって $x_i^{(2)}$ の値をすべて決める。第 2 層が済んだらそれをもとに第 3 層のすべてのニューロン $x_i^{(3)}$ の値を決める。このように層単位で上の操作を続けていくと、最後に出力層のニューロンの値が決まる。それが出力データである。

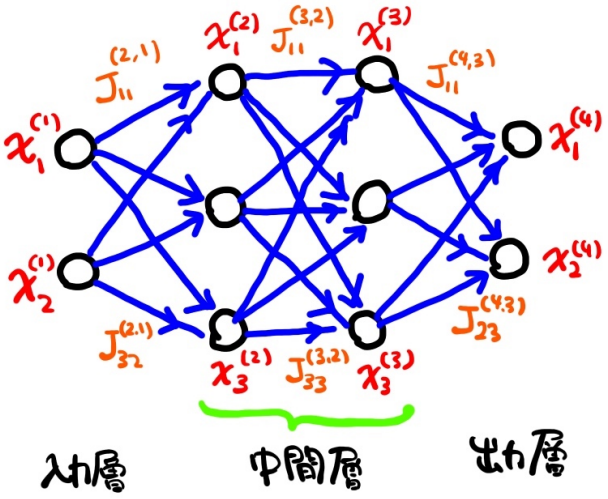

図 1　それぞれニューロン 2 個の入出力層とニューロン 3 個の中間層 2 層を持つ多層 NN の概念図。丸がニューロン、矢印が結合を表している

今回の問題では入力層のニューロンを 6 個、各中間層はニューロンを 7 個ずつ、そして出力層はニューロンを 4 個持つとする。つまり、6 ビットの情報を持つ入力データから 7 ビットの情報を経て、4 ビットの情報を持つ出力データが作られるわけである。中間層の数は問題によって異なる。この場合学習とは、入力 6 ビットと出力 4 ビットからなるデータの組をいくつか与えられたとき (これが**学習データ**である) に、それらを再現できるようにすべての結合 $J_{ij}^{(n+1,n)}$ の値を決定することを意味する。もちろん、可能な結合の組み合わせの数は学習データの数に比べて圧倒的に多い

のが普通なので ($J_{ij}^{(n+1,n)}$ が全部で m 個あれば可能な結合の組は 2^m 通りである)、同じ学習データを再現できる結合の組は多数ありうる。

2 問題

2.1 SuperCon 3 級認定問題

入力層と出力層だけで構成された (中間層を持たない)NN のすべての結合が問題として与えられる。そのとき、可能なすべての入力パターン (6 ビットなので 64 通りある) に対する出力を求めるプログラムを作れ。1 問を解くプログラムを作ればよい。

2.1.1 3 級認定問題詳細

結合は全部で 24 個ある (J_{11} から J_{46} まで。中間層がないので上付き添字は省略した)。SC_input 関数を実行すると、一次元グローバル整数配列 SC_J にこれが $J_{11}, J_{12}, \ldots, J_{16}, J_{21}, J_{22}, \ldots, J_{45}, J_{46}$ の順に格納される。各 J の値は 1 または-1 である。

ひとつの入力パターンは $x_1^{(1)}, \ldots, x_6^{(1)}$ までをそれぞれ 0 または 1 とすれば決まる。これを二進数と考えて、$I = x_6^{(1)} \times 2^5 + x_5^{(1)} \times 2^4 + x_4^{(1)} \times 2^3 + x_3^{(1)} \times 2^2 + x_2^{(1)} \times 2 + x_1^{(1)}$ と表すと (順番に注意)、$I = 0, 1, \ldots, 63$ ですべての入力パターンを尽くしている。そこで各入力パターンに対する出力 $x_1^{(2)}, \ldots, x_4^{(2)}$ を求める。入力パターンと同様に $O = x_4^{(2)} \times 2^3 + x_3^{(2)} \times 2^2 + x_2^{(2)} \times 2 + x_1^{(2)}$ と表せば O は 0 から 15 までのいずれかの整数となる。これを一次元グローバル整数配列 SC_ans に $I = n$ に対応する出力が n 番目の要素 SC_ans[n] となるように格納し (くれぐれも順番に注意すること)、最後に SC_output 関数を実行すると解答が出力される。

ヘッダーファイル sc3.h に必要な関数と配列が定義されているので、

```
#include "sc3.h"
```

として使うこと。定義されているものは以下である。

`void SC_input()` : **問題を標準入力から読み込む関数**
`void SC_output()` : **解答を標準出力に出力する関数**
`#define SC_NJIJ 24` : **結合の総数** (=24)
`int SC_J[SC_NJIJ]` : **問題の結合を格納する一次元整数配列**
`int SC_ans[64]` : **解答を格納する一次元整数配列**

2.1.2 3 級認定基準

問題は 10 問与えられ、1 問を解くプログラムを問題を変えて 10 回実行する。制限時間内にすべてに正解した場合に 3 級を認定する。制限時間は入出力を除く計算時間で 10 問合計 1 分とする。

2.2 SuperCon 2 級認定問題

中間層を 4 層持つ NN のすべての結合が問題として与えられる。そのとき、可能なすべての入力パターン (6 ビットなので 64 通りある) に対する出力を求めるプログラムを作れ。1 問を解くプログラムを作ればよい。

2.2.1 2 級認定問題詳細

結合は全部で $(42 + 3 \times 49 + 28)$ 個ある ($J_{11}^{(2,1)}$ から $J_{47}^{(6,5)}$ まで)。SC_input 関数を実行すると、一次元グローバル整数配列 SC_J にこれが

$$J_{11}^{(2,1)}, J_{12}^{(2,1)}, \ldots, J_{16}^{(2,1)}, J_{21}^{(2,1)}, \ldots J_{76}^{(2,1)}, J_{11}^{(3,2)}, J_{12}^{(3,2)}, \ldots, J_{46}^{(6,5)}, J_{47}^{(6,5)}$$

の順に格納される (ややこしいので、順番に注意)。各 J の値は 1 または-1 である。

ひとつの入力パターンは $x_1^{(1)}, \ldots, x_6^{(1)}$ までをそれぞれ 0 または 1 とすれば決まる。これを二進数と考えて、$I = x_6^{(1)} \times 2^5 + x_5^{(1)} \times 2^4 + x_4^{(1)} \times 2^3 + x_3^{(1)} \times 2^2 + x_2^{(1)} \times 2 + x_1^{(1)}$ と表すと (順番に注意)、$I = 0, 1, \ldots, 63$ ですべての入力パターンを尽くしている。そこで各入力パターンに対する出力 $x_1^{(2)}, \ldots, x_4^{(2)}$ を求める。入力パターンと同様に $O = x_4^{(2)} \times 2^3 + x_3^{(2)} \times 2^2 + x_2^{(2)} \times 2 + x_1^{(2)}$ と表せば O は 0 から 15 までのいずれかの整数となる。これを一次元グローバル整数配列 SC_ans に $I = n$ に対応する出力が n 番目の要素 SC_ans[n] となるように格納し (くれぐれも順番に注意すること)、最後に SC_output 関数を実行すると解答が出力される。

ヘッダーファイル sc2.h に必要な関数と配列が定義されているので、

```
#include "sc2.h"
```

として使うこと。定義されているものは以下である。

`void SC_input()` : **問題を標準入力から読み込む関数**
`void SC_output()` : **解答を標準出力に出力する関数**
`#define SC_NJIJ 217` : **結合の総数** (=217)
`int SC_J[SC_NJIJ]` : **問題の結合を格納する一次元整数配列**
`int SC_ans[64]` : **解答を格納する一次元整数配列**

2.2.2 2 級認定基準

問題は 10 問与えられ、1 問を解くプログラムを問題を変えて 10 回実行する。制限時間内にすべてに正解した場合に 2 級を認定する。制限時間は入出力を除く計算時間で 10 問合計 1 分とする。

2.3 SuperCon 1 級認定問題兼予選問題

中間層を 4 層持つ NN に対して、学習データとして入出力パターンの組が 10 組与えられる。それらの学習データすべてを再現できる結合の組を見つけるプログラムを作れ。同じ学習データを再現できる結合の組は多数ありうるが、そのうちひとつだけ見つければよい。3 級 2 級と異なり、問題は Q 問 (Q は 10 または 100) 与えられるので、それらを続けて解くプログラムを作成する。

2.3.1 1 級認定問題詳細

ひとつの入力パターンは $x_1^{(1)}, \ldots, x_6^{(1)}$ までをそれぞれ 0 または 1 とすれば決まる。これを二進数と考えて、$I = x_6^{(1)} \times 2^5 + x_5^{(1)} \times 2^4 + x_4^{(1)} \times 2^3 + x_3^{(1)} \times 2^2 + x_2^{(1)} \times 2 + x_1^{(1)}$ と表すと (順番に注意)、I は $0, 1, \ldots, 63$ のいずれかである。また、同様に出力を $O = x_4^{(6)} \times 2^3 + x_3^{(6)} \times 2^2 + x_2^{(6)} \times 2 + x_1^{(6)}$ とすれば、O は 0 から 15 までのいずれかの整数となる。

`SC_input` 関数を実行すると、10 個の I と対応する O の組が上で説明した整数として、二次元グローバル整数配列 `SC_prob[][]` に格納される。第 n 問目は `SC_prob[n-1][0]` **から** `SC_prob[n-1][19]` までに格納され、その順番は

$$I_1, O_1, I_2, O_2, \ldots, I_{10}, O_{10}$$

である。問題の総数は変数 `SC_n` に格納される。

各問題ごとに、この I と O の組すべて (10 組) を再現する結合を求める。結合は全部で $(42 + 3 \times 49 + 28)$ 個ある ($J_{11}^{(2,1)}$ から $J_{47}^{(6,5)}$ まで。各 J は 1 か-1 のいずれかでなくてはならない)。求めた結合を一次元グローバル整数配列 `SC_J` に

$$J_{11}^{(2,1)}, J_{12}^{(2,1)}, \ldots, J_{16}^{(2,1)}, J_{21}^{(2,1)}, \ldots J_{76}^{(2,1)}, J_{11}^{(3,2)}, J_{12}^{(3,2)}, \ldots, J_{46}^{(6,5)}, J_{47}^{(6,5)}$$

の順に格納し (ややこしいので、順番に注意)、`SC_output` 関数を実行すると解答が出力される (1 問解くごとに `SC_output` を実行すること)。

ヘッダーファイル `sc1.h` に必要な関数と配列が定義されているので、

```
#include "sc1.h"
```

として使うこと。定義されているものは以下である (乱数については、あとで説明する)。

`void SC_input()` : **問題を標準入力から読み込む関数**
`void SC_output()` : **解答を標準出力に出力する関数**
`void init_genrand(unsigned long s)` : **メルセンヌ・ツイスター乱数の初期化関数**
`int genrand_int31()` : **メルセンヌ・ツイスターにより** 31 **ビット整数乱数を発生する関数**
`#define RAND_MAX 2147483647` : **乱数の最大値**

`#define SC_NJIJ 217` : **結合の数**（=217）

`#define SC_NDATA 20` : **学習データの数の** 2 **倍**（=20）

`int SC_n;` **問題の総数**

`int SC_prob[24]` : **問題を格納する一次元整数配列**

`int SC_J[SC_NJIJ]` : **解答の結合を格納する一次元整数配列**

2.3.2 1 級認定基準

問題は 10 問与えられる。制限時間内にすべてに正解した場合に 1 級を認定する。制限時間は入出力を除いて 10 問合計 5 分とする。

2.3.3 本選出場チーム選考基準

本選出場チームの選考は東西に分けて行う。100 問を出題し、制限時間 (入出力を除いて合計 5 分) 内に正解できた数が多い順に、原則として東西各上位 10 チームを本選出場チームとする。ただし、正解数が同じ場合は計算時間が短いほうを上位とする。なお、全問正解を求めていないので、本選には出場できても 1 級には認定されない場合もありうる。

2.3.4 補足解説と解法のヒント

作り得る結合の種類は 2^{217} 通りと膨大にあるのに対し、入出力の組み合わせ (64 通りの入力に対し 16 通りの出力の組み合わせ) はたかだか 2^{10} 通りである。したがって、平均すると各入出力の組み合わせごとに**正解**の結合は 2^{207} 通りある。しかし、実際にはこの正解数には大きなばらつきがあり、それが学習の難易度のばらつきとなる。たとえば、結合をランダムに作った NN の入出力を使って学習データを作ると、おうおうにして非常に易しい (正解を求めやすい) データができる。これはランダム NN が作るデータは「珍しくない」ことを意味する。逆に学習データのほうをランダムに作ると、学習できるものもできないものも生成されるが、総じて学習が難しめになりがちである。

今回は学習データのほうをランダムに作り、その中から我々のテストプログラム (下で説明する貪欲法を用いた) で数秒以内に学習が完了したものだけを問題として使う。つまり、各問とも、少なくともひとつ正解が存在し、ひどく易しくはないがひどく難しくもないことが確認されている。そこで、問題の性質を理解できるように、そのようにして作ったサンプル問題を 50 問提供するので、それらを解けるようなプログラムを作ってもらいたい。[*3][*4]

[*3] 現実に NN を応用する場では何か意味のある学習データを使うが、今回のデータはランダムである。ただし、我々は正解となる NN をひとつ知っている。いわば、ある「教師機械」があって、これで各入力データに対する出力データ（ラベル）を生成していると考えてよい。皆さんには入出力データだけが提示されるので、それをもとに「教師機械」の振る舞いを模倣する「生徒機械」を自前でつくるというのが課題である。ただしこの問題では、教師機械がどんな構造をしているかは開示されている。現実の問題ではそれすらわからないのが普通である。

[*4] この問題では 64 通りの可能な入力のうち、10 通りの入力に対する応答を学習する。残りの 54 通りの入力は NN が「知らない入力」であり、これらに対しても適切に応答できるかどうかというのが汎化の問題である。

結合の組み合わせが膨大なので、総当たりで試すのは不可能である。最も簡単なのはランダムな (乱数で作った) 結合からスタートして、ランダムに結合を変化させてみて正解に近づけばその変化を採用するという**貪欲法**だろう。正解への近さをどのように測るかは工夫のしどころである。正解にたどり着けずに止まってしまう可能性もあるので、貪欲法を適当な回数行なっても正解が得られないときにはランダムな結合を作り直す必要があるだろう。この打ち切り回数はサンプル問題を使って試行錯誤して、適切な回数を決定しなくてはならない。もちろん貪欲法以外の手法を工夫してもかまわない。

なお、ひとつの問題に対する正解は膨大にあるため、例題の答えをすべて列挙することはできないし、ひとつくらい答えの例を挙げても意味がないだろう。そこで、例題には正解例を付けない。入力から出力を求めるプログラムは 2 級認定問題の解答と同じものが使えるので (どのみち、1 級問題を解くプログラムの中で必要になる)、解答をチェックするプログラムも用意して、自力でチェックすること。

2.3.5 乱数について

乱数が必要になる可能性が高いので、ヘッダー `sc1.h` に乱数生成関数を含めてある。これはメルセンヌ・ツイスターと呼ばれる生成方法に基づいており、開発者である松本眞氏のウェブサイト (`http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/mt.html`) で提供されている関数を少し改変したものである (都合により、より速い SFMT ではなく、オリジナルのメルセンヌ・ツイスターを用いている)。メルセンヌ・ツイスターは C 言語あるいは C++ 言語に標準で用意されている乱数よりも少しだけ計算時間がかかるようだが、「乱数の質」は圧倒的によいことが知られており、現実の科学技術計算に広く用いられている。今回は公平を期すために、乱数を必要とするプログラムでは必ず `sc1.h` に含まれるメルセンヌ・ツイスターを使うこととする。

使い方は、まず使用前に初期化関数 `void init_genrand(unsigned long s)` を実行する。`s` に適当な整数を入れればよく、この値が変わると異なる乱数の系列が生成されるが、今の場合は最初に一度実行するだけでよい。31 ビット整数乱数の生成には `int genrand_int31()` 関数を引数なしで使う。この関数が呼ばれるたびに異なる整数乱数がひとつ生成される。乱数の値は 0 から `RAND_MAX` までの整数である。また、0 から 1 まで (1 は含まない) の double 型乱数が必要なら、`double genrand_real2()` 関数を使う。

3 言語と実行環境その他の注意

プログラムは ANSI 準拠の C 言語 (C99) または C++ 言語 (C++14) で書く。また、プログラムは分割せずひとつのファイルにまとめること (ヘッダーファイルは提出不要)。独自のヘッダーファイルやライブラリーは使用できないので、プログラム開発に統合開発環境を使用する参加者はくれぐれも注意すること。

すべての級で、`SC_input` 関数は必ずプログラムの最初の実行文であること。つまり、`SC_input` の前に実行文を書いてはならない。入出力は `SC_input` と `SC_output` のみで行ない、他の出力を

プログラムに含めてはならない。また、変数 SC_StartTime と SC_EndTime の値はプログラム中で変えてはならない。なお、ヘッダーファイルは改変してはならない。これらの注意に反するプログラムは失格とする。

審査は 2.5GHz の Intel Xeon CPU を搭載した LINUX マシン上で実行する。コンパイラは gcc バージョン 4.8.2 を使い、プログラムが C 言語で書かれている場合は

```
gcc -O -lm -std=c99 プログラム名.c
```

C++ 言語で書かれている場合は

```
g++ -O -lm -std=c++14 プログラム名.cpp
```

とコンパイルする。実行時に問題は標準入力から読まれ、解答は標準出力に出力される。

計算時間は time.h に含まれる clock() 関数で測定する。入力終了後に測定を開始し、出力直前までの所要時間を測る。1 級については、出力に要した時間を差し引くことにより、計算時間を求める。

なお、問題データと解答のサンプルをウェブサイトに掲載する。

4 提出方法

ひとつの級にだけ応募してもよいし、複数の級に応募してもよい。作成したプログラムはファイル名を 1,2,3 級それぞれ、sc1.c、sc2.c、sc3.c とし (C++ の場合は拡張子を cpp とする。たとえば sc1.cpp)、メールの添付ファイルとして提出する。メールには必ず学校名、全員の名前と学年、連絡先メールアドレスを明記すること。級認定のみの応募はひとりでも構わないが、SuperCon 予選に応募する場合には 2 ないし 3 名のチームを作り、チーム名も明記すること (予選通過後のチーム構成変更は認められないので注意)。1 級応募者は何も書かれていなければ予選参加とみなすので、級認定のみ希望の場合はその旨を明記すること。また、予選応募の場合はアルゴリズム上の工夫などを記したレポートを同時に添付すること。

提出および問い合わせのメールアドレスは SuperCon ウェブサイトを参照のこと。