計算機科学実験４レポート 必修課題4+発展課題A1

工学部情報学科３回生 計算機科学コース
1029310587 谷 綜一郎

2022 年 1 月 21 日

目 次

第1章　課題内容

1.1　概要

MNIST のテスト画像 1 枚を入力とし、3 層ニューラルネットワークを用いて、0〜9 の値のうち 1 つを出力するプログラムを作成せよ。

1.2　諸条件

- 重みパラメータ $W^{(1)}$、$W^{(2)}$、$\boldsymbol{b}^{(1)}$、$\boldsymbol{b}^{(2)}$ が課題 3 で計算された重みであること

  以外は課題 1 と同じ仕様である。重みパラメータについてはファイルから読み込むようにすること。

第2章　プログラム説明

2.1　概要

課題1で実装したプログラム program1.py を元として program4.py を実装した。それぞれ、追加・変更した部分について記述する。また、実装したプログラム program4.py の外部仕様と内部設計を記述する。

2.2　外部仕様

課題内容の通り、重みパラメータのファイルを読み込み、0〜9999 の値のうち 1 つを入力すると、その値に応じた画像に加えて、画像に描かれた 0〜9 の正しい値と認識した値を出力する。

2.3　内部設計

2.3.1　概要

課題1で実装した諸関数に、必要な関数を適宜追加・変更することで program4.py を実装した。グローバル変数については program1.py と変更は無いが、念のため記述した後、諸関数に関する内部設計を記述する。また、最終的に実行する関数 $main4()$ を、以下のプログラム 2.1 に示す。

プログラム 2.1: main4

```
1  def main4(w1, b1, w2, b2):
2    (i, img, ans) = preprocess()
3    max = propagation(img, w1, b1, w2, b2)
4    print("result: " + str(max))
5    print("answer: " + str(ans))
6    plt.imshow(X[i], cmap=cm.gray)
7    plt.show()
8    main4(w1, b1, w2, b2)
9
10 (w1, b1, w2, b2) = inputFileName()
11 main4(w1, b1, w2, b2)
```

このコード中の変数については以下のとおりである。

- i : 入力した数値（0〜9999 の整数値）
- img : 28 × 28 の画像データ（2 次元配列）
- ans : 正解ラベルデータ（0〜9 の整数値）

- w1：100 × 784 の二次元配列
- b1：100 × 1 の二次元配列
- w2：10 × 100 の二次元配列
- b2：10 × 1 の二次元配列
- max：y2 のうち最大の数値のインデックス（0〜9 の整数値）

2.3.2 グローバル変数

- X：10000 × 28 × 28 の画像テストデータ（3 次元配列）
- Y：10000 個の正解ラベルデータリスト（1 次元配列）
- ImageSize：MNIST の画像サイズに関する定数（ここでは 28）
- D：ImageSize の 2 乗で定義した定数（ここでは 784）
- ImageNum：画像テストデータの枚数（ここでは 10000）
- ClassNum：出力層におけるノード数（ここでは 10）
- MiddleNodeNum：中間層におけるノード数（ここでは 100）

2.3.3 諸関数

関数 inputNumber, inputLayer, combine, sigmoid, softmax に関しては、program1.py と全く同じであるが、念のため記述する。

- inputNumber

  具体的なコードは、以下のプログラム 2.2 に示すとおりである。上記の通り、この関数は program1.py の inputNumber と全く同じである。標準入力から 0〜9999 の数値を受け取り、適切でない場合は適宜エラーを出して、その数値を返す関数である。

プログラム 2.2: inputNumber

```
1  def inputNumber():
2      num = input("Please enter the number as you like. (0 ~ " + str(
           ImageNum - 1) + ")\n")
3      try:
4          num = int(num)
5      except:
6          print("Please enter the 'number'.")
7          return inputNumber()
8      if num < 0 or num > (ImageNum - 1):
9          print("Please enter '0 ~ " + str(ImageNum - 1) + "'.")
```

```
10          return inputNumber()
11      else:
12          return num
```

- inputFileName

具体的なコードは、以下のプログラム 2.3 に示すとおりである。これは今回の program4.py で追加された新しい関数であり、標準入力からファイルの名前を受け取り、適切でない場合は適宜エラーを出して、読み込んだファイル中の重みパラメータ $W^{(1)}$、$\boldsymbol{b}^{(1)}$、$W^{(2)}$、$\boldsymbol{b}^{(2)}$ の組を返す関数である。

プログラム 2.3: inputFileName

```
1   def inputFileName():
2       name = input("What's your file's name? ('.npy' isn't needed!) "+ "\n
            ")
3       try:
4           file = np.load(name + ".npy")
5       except:
6           print("No file!")
7           return inputFileName()
8       return file
```

- preprocess

具体的なコードの変更点は、以下のプログラム 2.4 に示すとおりである。program1.py と異なり、標準入力から数値 i を受け取る inputNumber を用いて、10000 枚のテストデータから画像を抽出して $image28$ としている。その正解ラベルを ans として、$i, image28, ans$ の組を返す関数である。

プログラム 2.4: preprocess

```
1   def preprocess():
2       i = inputNumber()
3       image28 = X[i]
4       ans = Y[i]
5       return i, image28, ans
```

- inputLayer

具体的なコードは、以下のプログラム 2.5 に示すとおりである。上記の通り、この関数は program1.py の inputLayer と全く同じである。inputLayer は、28×28 の二次元配列 $image$ を入力として、配列しなおすことで、784×1 の二次元配列 $image784$ を返す。

プログラム 2.5: inputLayer

```
1    def inputLayer(image):
2     image784 = image.reshape(D, 1)
3     return image784
```

- combine

具体的なコードは、以下のプログラム 2.6 に示すとおりである。上記の通り、この関数は program1.py の combine と全く同じである。combine は、$q \times\ r$ の二次元配列 $input$ と $p \times\ q$ の二次元配列 $weight$ と $p \times\ r$ の二次元配列 b を入力として、$weight$ と $input$ の行列積に b を加えたものを計算して、$p \times\ r$ の二次元配列を返す。

プログラム 2.6: combine

```
1    def combine(input, weight, b):
2     return np.dot(weight, input) + b
```

- sigmoid

具体的なコードは、以下のプログラム 2.7 に示すとおりである。上記の通り、この関数は program1.py の sigmoid と全く同じである。多次元配列 t を入力として、各要素に**シグモイド関数**を適用したような、t と同じ形の配列を返す。

プログラム 2.7: sigmoid

```
1    def sigmoid(t):
2     return 1/(1+np.exp(-t))
```

- softmax

具体的なコードは、以下のプログラム 2.8 に示すとおりである。上記の通り、この関数は program1.py の softmax と全く同じである。多次元配列 x を入力として、**ソフトマックス関数**を適用した後に、x と同じ形の配列を返す。

プログラム 2.8: softmax

```
1    def softmax(x):
2     y = np.exp(x - np.max(x))
3     f_x = y / np.sum(y)
4     return f_x
```

- propagation

具体的な新しいコードは、以下のプログラム 2.9 に示すとおりである。propagation は、program1.py の main1 をもとにした関数である。主な変更点はクロスエントロピー誤差 e を求めることと、複数枚を入力可能とするため、max を導出する際に axis を利用している。

プログラム 2.9: propagation

```
1    def propagation(img, w1, b1, w2, b2):
2     x = inputLayer(img)
3     y1 = sigmoid(combine(x, w1, b1))
4     a = combine(y1, w2, b2)
5     y2 = softmax(a)
6     max = np.argmax(y2, axis=0)
7     return max
```

このコードにある変数の説明は以下の通りである。

- x：784 × 1 の画像データ
- y1：100 × 1 の二次元配列
- a：10 × 1 の二次元配列
- y2：10 × 1 の二次元配列
- max：y2 のうち最大の数値のインデックス（0〜9 の整数値）

第3章　実行結果

実行結果コードを以下 3.1 に示す。また、実行結果画像のスクリーンショットを以下 3.1、 3.2 に示す。今回は program3.py で作成した重みパラメータのファイル temp50.py を読み込んだ。このファイルは 50 エポックで作成されたもので、テストデータに対する正答率は 95.87%である。

なお今回は使っていないが、10 エポックで作成した temp10.py というファイルの正答率は 93.92%で、自分の作ったプログラムでは、少なくとも 10 回よりは 50 回分エポックを回した方が良いことが分かった。

プログラム 3.1: 実行結果

```
>python program4.py
What's your file's name? ('.npy' isn't needed!)
temp50
Please enter the number as you like. (0 ~ 9999)
1
result: [2]
answer: 2
Please enter the number as you like. (0 ~ 9999)
111
result: [7]
answer: 7
...
```

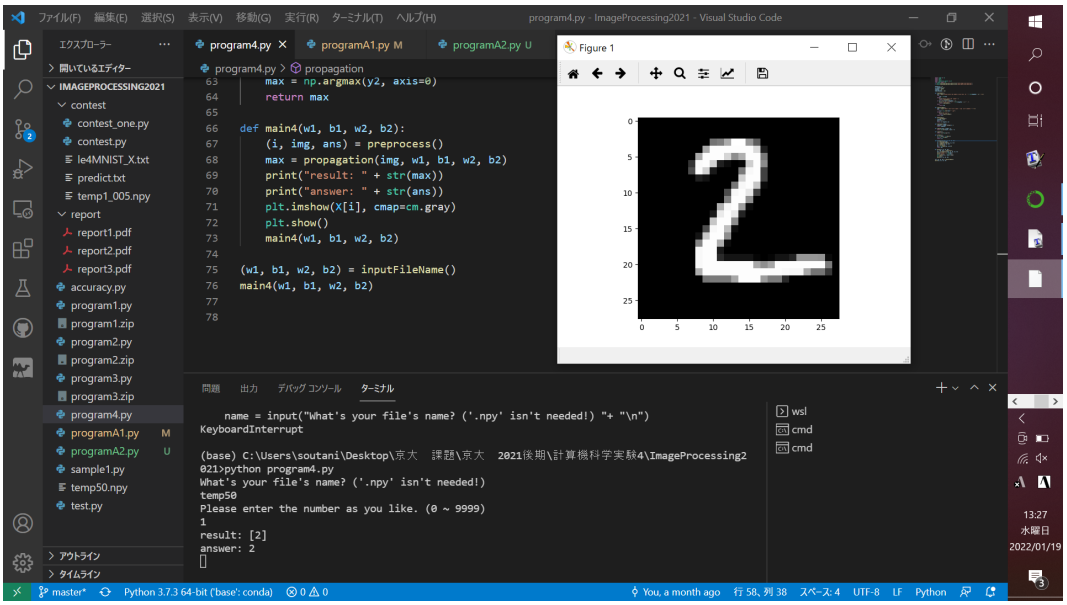

図 3.1: 実行結果 1

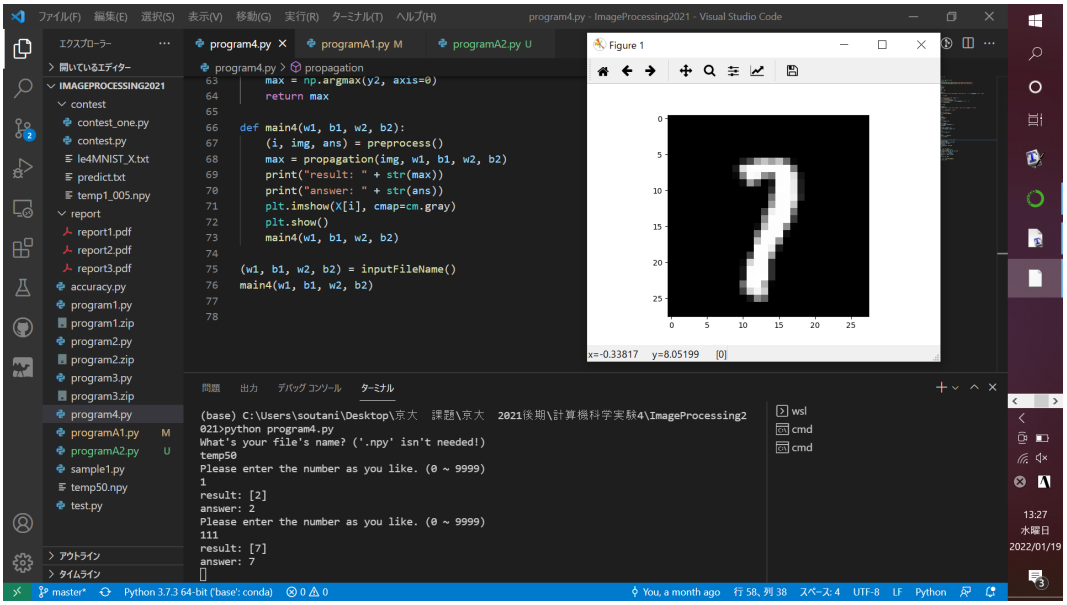
program4.py - ImageProcessing2021 - Visual Studio Code
Figure 1
(base) C:\Users\soutani\Desktop\京大　課題\京大　2021後期\計算機科学実験4\ImageProcessing2021>python program4.py
What's your file's name? ('.npy' isn't needed!)
temp50
Please enter the number as you like. (0 ~ 9999)
1
result: [2]
answer: 2
Please enter the number as you like. (0 ~ 9999)
111
result: [7]
answer: 7

図 3.2: 実行結果 2

第4章　発展課題A1

4.1　課題内容

活性化関数としてシグモイド関数の代わりに、以下に説明する ReLU 関数を実装する。課題 3 で作成した program3.py をもとにして、グローバル変数や関数を追加することで、programA1.py を作成した。

4.2　ReLU 関数

活性化関数としてシグモイド関数の他に次式で挙げる ReLU もよく用いられる。

$$a(t) = \begin{cases} t & (t > 0) \\ 0 & (t \leq 0) \end{cases}$$

また、ReLU の微分は以下で与えられる。

$$a(t)' = \begin{cases} 1 & (t > 0) \\ 0 & (t \leq 0) \end{cases}$$

4.3　プログラム説明

4.3.1　グローバル変数

具体的なグローバル関数に関するコードを以下 4.1 に示す。また、そのグローバル変数の説明も下に記述する。

プログラム 4.1: global

```
1    X = mnist.download_and_parse_mnist_file("train-images-idx3-ubyte.gz")/255
2    ActiveFanction = 1
3    \# 0:sigmoid, 1:ReLU
```

- X : 60000 × 28 × 28 の画像学習データ（3 次元配列）を正規化したもの
- ActiveFunction : 活性化関数（0 ＝シグモイド関数、1 ＝ ReLU）

4.3.2 追加・変更した部分

- relu

  具体的な新しいコードは、以下のプログラム 4.2 に示すとおりである。各要素を 0 と比較して、大きい方を取り出すことで、ReLU 関数を表現した。

プログラム 4.2: relu

```
1  def relu(t):
2   return np.maximum(0.0, t)
```

- revRelu

  具体的な新しいコードは、以下のプログラム 4.3 に示すとおりである。ReLU を適用した後の各要素の符号を取り出すことで、ReLU 関数の導関数を表現した。

プログラム 4.3: revrelu

```
1  def revRelu(y, enY):
2   return enY*np.sign(relu(y))
```

- propagation

  具体的な新しいコードは、以下のプログラム 4.4 に示すとおりである。propagation は、program3.py のものとほぼ変わらない関数である。主な変更点は、グローバル変数 ActiveFunction を用いて、シグモイド関数と ReLU 関数を使い分けている点である。

プログラム 4.4: propagation

```
1  def propagation(img, ans, w1, b1, w2, b2):
2   x = inputLayer(img)
3   if ActiveFanction == 0:
4       y1 = sigmoid(combine(x, w1, b1))
5   else:
6       y1 = relu(combine(x, w1, b1))
7   a = combine(y1, w2, b2)
8   y2 = softmax(a)
9   max = np.argmax(y2, axis=0)
10  (onehot, e) = crossEntropy(ans, y2)
11  prop = (x, y1, y2, onehot, e, max)
12  return prop
```

- backPropagation

具体的な新しいコードは、以下のプログラム 4.5 に示すとおりである。backpropagation も、先と同様に、program3.py のものとほぼ変わらない関数である。主な変更点は、グローバル変数 ActiveFunction を用いて、シグモイド関数と ReLU 関数の導関数を使い分けている点である。

プログラム 4.5: backpropagation

```
1   def backPropagation(w1, b1, w2, b2, prop):
2    (x, y1, y2, onehot, e, max) = prop
3    enY2 = revEntropy(y2, onehot)
4    (enX2, enW2, enB2) = revCombine(w2, y1, enY2)
5    if ActiveFanction == 0:
6        enY1 = revSigmoid(y1, enX2)
7    else:
8        enY1 = revRelu(y1, enX2)
9    (enX1, enW1, enB1) = revCombine(w1, x, enY1)
10   new = renew(w1, b1, w2, b2, enW1, enW2, enB1, enB2)
11   return new, max, e
```

4.3.3 実行結果

実行結果コードを以下 4.6 に示す。今回作成した重みパラメータのファイルを temp50_A1.py として保存した。

プログラム 4.6: 実行結果 A1

```
1   >python programA1.py
2   Do you have a file? y/n n
3   1_epoc : 1.7872492954157413
4   1_result: 72\%
5   2_epoc : 0.7901488770228029
6   2_result: 86\%
7   ..
8   49_epoc : 0.18718473767956997
9   49_result: 96\%
10  50_epoc : 0.18936977303204677
11  50_result: 96\%
12  Do you save file? y/n y
13  Please enter the file's name. ('.npy' isn't needed!) temp50_A1
14  Saved.
```

第5章　工夫点・問題点

5.1　program.1py

- 数値入力の際のエラー対応などの機能を追加した。
- 実行する度に同じ結果が出力されるように乱数シードを設定する際に、標準入力で得られる i を用いた。
- 画像テストデータに対して、0〜9 のうち 1 つの数値を出力するだけでなく、比較として、正解ラベルの数値と選ばれた画像データも出力するようにした。
- ３層ニューラルネットワークのステップや関数ごとに、細かく分割してプログラムを実装して、見た目などの可読性も考慮して記述した。

5.2　program2.py

- ミニバッチ学習を利用する際に、複数枚の画像を入力とするために、配列の形に注意して実装した。
- 3 次元以上の配列の形を変えるとき、望ましいように配置されてるか確認しなければならなかった点に苦労した。
- ３層ニューラルネットワークのステップや関数ごとに、細かく分割してプログラムを実装して、見た目などの可読性も考慮して記述した。
- 出力される数値をプログラム内のランダムシードの値を変える必要がある点は改良する余地があると感じた。

5.3　program3.py

- 課題 1 と同様に、適切でない入力にはエラーメッセージを返すことで、プログラムが終了しないようにした。
- 当初、「(img, ans) = preprocess()」を j ループではなく i ループに記述していたため、以下の 5.1 に示すようにクロスエントロピー誤差の増減に違和感があった。当初のままでは、教師データのごく一部を何周も学習する仕様になっていたことが原因と考えられたので、教師データをのべ全て学習できるように j ループに移動した。

プログラム 5.1: 当初の出力結果

```
..
13_epoc : 0.2950703574867874
13_result: 96\%
14_epoc : 0.27851366277419637
14_result: 96\%
15_epoc : 0.22615694616498117
15_result: 96\%
16_epoc : 0.26319333088649616
16_result: 97\%
17_epoc : 0.19232569045082357
17_result: 96\%
18_epoc : 0.1742842994219058
18_result: 98\%
19_epoc : 0.18105245388450647
19_result: 97\%
20_epoc : 0.27358853288502
20_result: 94\%
..
```

5.4 program4.py

- 課題 1 と同様に、比較として、正解ラベルの数値と選ばれた画像データも出力するようにした。またループを利用することで、表示された画像を閉じるともう一度数値入力を促される仕様になっている。

5.5 programA1.py

- 当初、クロスエントロピー誤差の計算の際に log 演算で 0 除算が行われているというエラーが出ていた。そこで、学習用の画像データをダウンロードする際に 255 でわって正規化することで、不適切な計算が行われないようにした。

- 上記で保存した tem50_A1.npy は 50 エポックで作成されたもので、テストデータに対する正答率は 44.34%であった。これはシグモイド関数を利用したものより著しく低いため、何かしらのバグが存在しているのではないかと考えている。しかし、バグの原因発見には至っていない。

5.6 全体

- デバッグの際に不都合が生じないよう、ctrl+C を入力しない限りはプログラムが終了しないように記述した。例えば、ファイル名や数値の標準入力に不適切な入力がされた場合にはエラーメッセージが返されてもう一度入力が促される。あるいは、program4.py において、関数 main4() をループさせることで、手書きの画像表示を消した瞬間に、プログラムが終了せず、数値の入力が促されるように記述した。

- 可読性を高めるために、比較的多くの部分を関数化して整理した。そのため、どのプログラムも main の部分は簡潔に記述することが出来たように思う。