2023年度 計算機科学実験4
画像処理

提出日　2023年10月18日

京都大学
工学部情報学科計算機科学コース3年

締切日 2023/10/20

学籍番号　1029332664

氏名　村重寿弥

目 次

1　環境、課題内容

1.1　演習環境

本稿は計算機科学実験及び演習 4 画像処理における課題 1 報告書である。本実験はニューラルネットワークを用いた画像認識の演習を行う。演習を行う環境は以下の通りとする

- conda 23.7.4
- python 3.8.15
- mnist 0.2.2
- numpy 1.21.5
- matplotlib 3.5.3

1.2　課題内容

MNIST の画像 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する。つまり、順伝播のみを行い、その出力のみを行うプログラムを作成すれば良い

2　プログラムの説明

2.1　データの前処理

入力データである minst の要素は 28*28 の二次元配列である。今回はベクトルを入力としたいので、えられた 28*28 の二次元配列をベクトルに変換する。プログラムは以下のようになる。picture_size は画像データのサイズ、M は中間層のノード数、C は出力層のノード数 (分類クラス数) を表す。

Listing 1: データの前処理

```
import matplotlib.pyplot as plt
from pylab import cm

np.random.seed(0)

picture_size = 28 * 28

M = 50

i = int(input("の整数を入力してください0~9999:"))

C =10

test_data = X[i].reshape(picture_size)
```

2.2 入力層、全結合層 1

入力層では、上記のデータをそのまま受け取り出力するだけなので、特にコードは記述していない。中間層のノード数が $M = 50$ なので、全結合層 1 では 786 × M 行列 $\mathbf{W^1}$ と入力データ $\mathbf{x}$ との内積を取り、50 次元ベクトル $\mathbf{b^1}$ を加える。出力を $\mathbf{y}$ とすると、

$$\mathbf{y} = \mathbf{W^1}^T \cdot \mathbf{x} + \mathbf{b^1}$$

以下にプログラムのコードを示す

Listing 2: 入力、全結合 1

```
W_1 = np.random.normal(0, 1/np.sqrt(picture_size), (picture_size, M))

b_1 = np.random.normal(0, 1/np.sqrt(picture_size), M)

dot_1 = np.dot(np.array(W_1).T, test_data) + b_1
```

2.3 中間層

中間層では、全結合層 1 により、結合されたノードに活性化関数を作用させる。今回の活性化関数はシグモイド関数であり、ノード数は 50 である。シグモイド関数は以下の通りである。

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

以下にプログラムのコードを示す

Listing 3: 中間層

```
y_1 = 1 / (1 + np.exp(-dot_1))
```

2.4 全結合層 2

出力層のノード数が $C = 10$ なので、全結合層 2 では M × C 行列 $\mathbf{W^2}$ と中間層からの出力データ $\mathbf{x}$ との内積を取り、C 次元ベクトル $\mathbf{b^2}$ を加える。出力を $\mathbf{y}$ とすると、

$$\mathbf{y} = \mathbf{W^2}^T \cdot \mathbf{x} + \mathbf{b^2}$$

以下にプログラムのコードを示す

Listing 4: 全結合 2

```
W_2 = np.random.normal(0, 1/np.sqrt(picture_size), (M, C))

b_2 = np.random.normal(0, 1/np.sqrt(picture_size), C)

dot_2 = np.dot(np.array(W_2).T, y_1) + b_2
```

2.5 出力層

出力層では、全結合層 2 により、C 次元となったベクトル $\mathbf{x}$ を受け取り、これにソフトマックス関数を適用する。ソフトマックス関数は以下の通りである。

$$\mathrm{softmax}(\mathbf{z}) = \frac{e^{z_i - \alpha}}{\sum_{i=1}^{C} e^{z_i - \alpha}}$$

$$\alpha = max(\mathbf{z})$$

この出力 $z_i (i = 1, ..., C)$ は、入力画像がクラス i に属する尤度を表すので、softmax($\mathbf{z}$) の要素で最大となる要素のインデックスが認識結果として出力される。以下にプログラムのコードを示す。

Listing 5: 出力層

```
alpha = np.max(dot_2)

softmax_vector = np.exp(dot_2 - alpha) / np.sum(np.exp(dot_2 - alpha))

result = np.argmax(softmax_vector)

plt.imshow(X[i], cmap=cm.gray)
plt.show()
print("正解は:"+str(Y[i]))

print("予測結果は:"+str(result))
```

3 実行結果、工夫点、問題点

3.1 実行結果

何も学習させず、出鱈目なパラメータで順伝播を行って予測しただけなので、予測結果はほぼほぼ外れていた。

3.2 工夫点

ニューラルネットワークの内容については特に工夫を施していない。プログラムの実装方針に関して、大きなサイズの行列の演算が絡むので、for 文を利用しないよう心がけた。numpy ライブラリを用いると、行列の演算が容易を容易に行うことができ、行列の要素全体に関数を適用することも可能であった。

3.3 問題点

今回のモデルの問題点は一つのデータしか入力できない点、出鱈目なパラメータで予測を行ってしまっている点である。実際にニューラルネットワークで学習を進めていく際は 1 つのデータごとに学習しているとキリがないので、複数のデータをまとめて学習する。よって、複数のデータをまとめて入力できるようにする必要があるのと、逆伝播を実装し、実際にパラメータを調整していく必要がある。