計算機科学実験４レポート 必修課題2

工学部情報学科3回生 計算機科学コース
1029310587 谷 綜一郎

2021 年 12 月 23 日

目 次

第1章　課題内容

1.1　概要

[課題 1] のコードをベースに、ミニバッチ（＝複数枚の画像）を入力可能とするように改良し、さらにクロスエントロピー誤差を計算するプログラムを作成せよ。

1.2　諸条件

- MNIST の学習画像 60000 枚の中からランダムに B 枚をミニバッチとして取り出すこと。
- クロスエントロピー誤差の平均を標準出力に出力すること。
- バッチサイズ B は自由に決めて良い（100 程度がちょうどよい）。
- ミニバッチを取り出す処理はランダムに行う。

第2章　プログラム説明

2.1　概要

まず、ニューラルネットワークの基本的な学習アルゴリズムを示した後、損失関数・ミニバッチについて説明する。そして、課題1で実装したプログラム program1.py に対して、追加・変更した部分について記述する。最後に、実装したプログラム program2.py の外部仕様と内部設計を記述する。

2.2　基本的な学習アルゴリズム

教師データを用いて、重み $W^{(1)}$、$W^{(2)}$、$\boldsymbol{b}^{(1)}$、$\boldsymbol{b}^{(2)}$ を学習する手法を解説する。基本的な学習アルゴリズムを以下に示す。

1. 適当な値で重みを初期化（課題 1 で実装済み）
2. 定められた繰り返し回数に達するまで 3〜5 を繰り返す
3. 教師データに対する出力を計算（課題 1 で実装済み）
4. 3 の結果と正解との誤差を計算（**損失関数**）
5. **誤差逆伝播法**により重みを修正

2.3　損失関数

教師データにおいて $\boldsymbol{x}$ に対する正解 y はクラスラベル（0〜9 までの値）なのに対し、ニューラルネットワークの出力層の出力は C 次元のベクトル $\boldsymbol{y}^{(2)}$ である。そこで、教師データの正解を正解ラベルを 1、残りを 0 とした C 次元ベクトルで表記することを考える。例えば、ある入力に対する正解が $y = 3$ である場合、これを 10 次元ベクトル (0, 0, 0, 1, 0, 0, 0, 0, 0, 0) で表記する。この表記法を **one − hotvector** 表記と呼ぶ。

多クラス識別における損失関数として、次式で表されるクロスエントロピー誤差がよく用いられる。

$$E = \sum_{k=1}^{C} -y_k \log y_k^{(2)}$$

ここで、y_k は one-hot vector 表記における正解の k 番目の要素、$y_k^{(2)}$ は出力層における k 番目の要素の出力である。

2.4 ミニバッチ

教師データを用いて学習を行う際、全ての教師データに対して損失関数の値を計算するのは計算量の観点から効率的ではない。そこで、教師データの中からランダムに一部のデータを取り出し、そのデータに対してクロスエントロピー誤差を計算してその平均を全ての教師データに対する誤差の近似として用いる方法がとられる。これを**ミニバッチ学習**と呼び、一度に取り出すデータの個数を**バッチサイズ**と呼ぶ。

$$E_n = \frac{1}{B} \sum_{i \in \text{ミニバッチ}} \sum_{k=1}^{C} -y_{i,k} \log y_{i,k}^{(2)}$$

B はバッチサイズ、$y_{i,k}$ と $y_{i,k}^{(2)}$ はそれぞれ i 番目のデータにおける y_k と $y_k^{(2)}$ である。

2.5 外部仕様

課題説明の通り、実行するとクロスエントロピー誤差を出力する。なお、プログラム内のランダムシードを変更することで、違う数値が出力される。

2.6 内部設計

2.6.1 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 program2.py を実装した。グローバル変数について説明した後に、各ステップに分けて、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 $main2()$ を、以下のプログラム 2.1 に示す。

プログラム 2.1: main2()

```
1  def main2():
2    (img, ans) = preprocess()
3    (w1, b1, w2, b2) = preWB()
4    (max, e) = propagation(img, ans, w1, b1, w2, b2)
5    #print(max)
6    print(e)
```

このコード中の変数については以下のとおりである。

- img：100 × 28 × 28 の画像データ（3 次元配列）
- ans：正解ラベルデータ（0〜9 の整数値 100 個）
- 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 100 個）
- e：クロスエントロピー誤差

2.6.2 グローバル変数

- X：60000 × 28 × 28 の画像データ（3 次元配列）
- Y：60000 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：画像データの枚数（ここでは 60000）
- 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）
- BatchSize：バッチサイズ（ここでは 100）

2.6.3 変更した関数

- preprocess

  具体的なコードの変更点は、以下のプログラム 2.2 に示すとおりである。標準入力から数値を受け取る inputNumber を用いることなく、60000 枚の学習データからバッチサイズ分のデータを重複することなくランダムに抽出して *image* としている。

プログラム 2.2: preprocess

```
1  def preprocess():
2   i = np.random.choice(ImageNum, BatchSize, replace=False)
3   image = X[i]
4   answer = Y[i]
5   return image, answer
```

- inputLayer

  具体的なコードの変更点は、以下のプログラム 2.3 に示すとおりである。複数枚の画像を入力可能とするために、reshape を利用して配列の形を変更している。ここで望ましい配列の形を得るために .T を利用している。

プログラム 2.3: inputLayer

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

- softmax

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。先と同様に、複数枚の画像を入力可能とするために、axis を利用して望ましい配列を得ている。

プログラム 2.4: softmax

```
1  def softmax(x):
2   y = np.exp(x - np.max(x, axis=0))
3   f_x = y / np.sum(y, axis=0)
4   return f_x
```

2.6.4 追加した関数

- crossEntropy

具体的な新しいコードは、以下のプログラム 2.5 に示すとおりである。crossEntropy は、正解ラベルデータ *answer* と出力層のデータ *out* を入力として、onehot vector を作成して、クロスエントロピーを出力する。

プログラム 2.5: crossEntropy

```
1  def crossEntropy(answer, out):
2   onehot = np.identity(10)[answer]
3   e = np.dot(-onehot, np.log2(out))
4   return (np.sum(np.diag(e)))/BatchSize
```

- propagation

具体的な新しいコードは、以下のプログラム 2.6 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 2.6: propagation

```
1  def propagation(img, ans, 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       e = crossEntropy(ans, y2)
8       return max, e
```

このコードにある変数の説明は以下の通りである。

- x：784 × 100 の画像データ
- y1：100 × 100 の二次元配列
- a：10 × 100 の二次元配列
- y2：10 × 100 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 100 個）
- e：クロスエントロピー誤差

第3章　実行結果

実行結果を以下 3.1 3.2 に示す。なお、この結果はそれぞれランダムシードを 0,1 に固定したものである。

プログラム 3.1: 実行結果 (0)

```
>python program2.py
3.425236208381143
```

プログラム 3.2: 実行結果 (1)

```
>python program2.py
3.8846408079430397
```

第4章　工夫点・問題点

全体

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。そこで、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。また、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べき部分だと感じた。