計算機科学実験4 画像処理
課題2 レポート

1029311501 坂井優斗

提出日：2021年12月6日

1　課題内容

課題 1 で作成したプログラムをミニバッチに対応できるように拡張する。また, 損失関数としてクロスエントロピー誤差を計算できるようにする。プログラムの仕様は以下の通りである。

- ミニバッチは B 枚とし, MNIST の学習画像 60000 枚の中から選ぶ。
- 標準出力にミニバッチ B 枚のクロスエントロピー誤差の平均値を出力する。
- バッチサイズ B は自由に決めて良い。$\rightarrow B = 100$ とする。
- ミニバッチを取り出す処理はランダムに行う。
- ニューラルネットワークの構造は前回の課題 1 と同様のものとする。

2　プログラムの説明

課題 1 から修正を加えたり, 追加したコードについて説明をまとめる。本レポートで触れられていないプログラム部分については課題 1 についてまとめたレポートを参照していただきたい。

2.1　グローバル変数

- train_images : 60000 枚の教師用画像が格納される配列
- train_labels : 教師用画像に描かれている数字が格納される配列
- test_images : 10000 枚の教師用画像が格納される配列
- test_labels : テスト用画像に描かれている数字が格納される配列
- batch_size = 100 : ミニバッチのサイズ

2.2　load_image

Code 1: load_image

```
1   def load_image():
2     global train_images, train_labels, test_images, test_labels
3     test_images = mnist.download_and_parse_mnist_file("t10k-images-idx3-ubyte.gz")
4     test_labels = mnist.download_and_parse_mnist_file("t10k-labels-idx1-ubyte.gz")
5     train_images = mnist.download_and_parse_mnist_file("train-images-idx3-ubyte.gz")
6     train_labels = mnist.download_and_parse_mnist_file("train-labels-idx1-ubyte.gz")
```

この関数は, MNIST の教師用画像及びテスト用画像とそのラベルを読み込む動作をまとめたものである。mnist.download_and_parse_mnist_file() 関数を使って, MNIST の画像データをロードし, 対応する変数に画像データとラベルを格納している。
なお, 課題 2 では MNIST のテスト用画像は使用しないが, 今後の課題で利用する必要が出てくるので, テスト用画像とそのラベルも読み込むようにしている。

2.3 loss_function

Code 2: loss_function

```
1   def loss_function(processed_images, labels): #cross_entropy_error
2     cross_entropy_error_list = []
3     for i in range(batch_size):
4       processed_image, label = processed_images[i], labels[i]
5       label_vector = np.zeros(output_node_size)
6       label_vector[label] = 1
7       cross_entropy_error = -(np.dot(label_vector, np.log(processed_image)))
8       cross_entropy_error_list.append(list(cross_entropy_error))
9     cross_entropy_error_mean = np.mean(cross_entropy_error_list)
10    return cross_entropy_error_mean
```

この関数は, 引数として受け取ったミニバッチとラベルに対する損失を計算して, 値を返す。正確に説明すると, 第一引数の processed_images は, ミニバッチをニューラルネットワークに通した後のデータである。

2 行目の cross_entropy_error_list は, バッチに含まれる各画像 1 枚 1 枚の損失を要素として保持する配列である。

one-hot-vector 表記されたラベルが $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, ニューラルネットワークの出力層の出力が $\boldsymbol{y'} = (y'_1, y'_2, \ldots, y'_n)$ と表されるとき, 交差エントロピー誤差 E は,

$$E = \sum_{i=1}^{n} -y_i \log y'_i \tag{1}$$

で計算される。

5, 6 行目でラベルを one-hot-vector 表記で表す操作を行い, 7 行目で (1) 式のように損失を計算している。そして, 8 行目で cross_entropy_error_list に計算した損失を加える。

バッチサイズ分の画像の損失を計算し終わったら, 最後に各画像の損失の平均をとり, これを本関数, すなわちミニバッチに対する損失関数の返り値とする。これが 9, 10 行目に当たる。

2.4 get_batch

Code 3: get_batch

```
1   def get_batch():
2     indexs = np.random.choice(len(train_images), size=batch_size, replace=False)
3     batchs, labels = train_images[indexs], train_labels[indexs]
4     return batchs, labels
```

これは, バッチを画像データからランダムに抽出して返す関数である。

2 行目にあるように, numpy の random.choice() 関数を利用して 0 ～ 59999 の整数のうち, バッチサイズ $B = 100$ 個の整数をランダムに選択して, indexs という変数に格納している。ちなみに, random.choice() 関数の第 3 引数 replace は重複を許してサンプリングを行うかどうかを指定する引数で, 今回は同一ミニバッチ内では異なる画像を利用した方が偏りがなくなると判断したため, replace=False としている。

3 行目で, 先ほど生成したランダムな整数列 indexs を配列のインデックスとしてアクセスすることで, 教師用画像データ, そしてそれに対応するラベルから B 個のデータをランダムに抽出している。

2.5 main 関数

Code 4: main 関数

```
1   def main():
2     load_image()
3     batchs, labels = get_batch()
4     processed_batchs = np.array(list(map(output_layer, map(inner_layer, map(input_layer,
        batchs)))))
5     cross_entropy_error = loss_function(processed_batchs, labels)
6     print(f"The␣mean␣of␣losses␣is␣{cross_entropy_error}.")
```

まず, 2 行目では load_image 関数で MNIST の画像を読み込んでいる。
次に, 3 行目で get_batch 関数を実行して教師用画像からバッチサイズ B 枚分の画像とラベルを抽出し, これを images, labels とする。そして, 4 行目で, 抽出したミニバッチを入力層, 中間層, 出力層の順で通して処理を行う。
最後に, 5 行目で loss_function の引数に先ほどのニューラルネットワークに通した画像データと, それに対応する正解ラベルの配列を与えて, 交差エントロピ誤差の平均を計算する。そして標準出力に出力する。

3 実行結果

コマンド `python task2.py` でスクリプトを実行する。
実行すると, 標準出力に”The mean of losses is 2.4922930523554396.”と出力された。これより, ランダムに抽出したミニバッチに対する損失 (交差エントロピー誤差) はおよそ 2.492 であることがわかった。また, 何度か実行してみると, 損失の値は”2.481”, ”2.506”が出力された。したがって, 重みとバイアス項が乱数であるときは, 損失の値が約 2.5 であることがわかる。

ターミナル

```
$ python task2.py
The mean of losses is 2.4922930523554396.
$ python task2.py
The mean of losses is 2.483123405993909.
$ python task2.py
The mean of losses is 2.5060122534873557.
```

4 工夫点

-

5 問題点

-