2023年度 計算機科学実験4
画像処理 課題2

提出日　2023年10月26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 2023/10/27

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 2 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

課題 1 で作成した 3NN を改良し、複数枚の画像データ (ミニバッチ) を入力可能にし、クロスエントロピー誤差を求めるプログラムを作成する。

2　プログラムの説明

2.1　データの前処理

以下、バッチ数を B と表す。入力データは B × 28 × 28 の 3 次元配列である。課題 1 で構成した 3NN モデルはベクトルの入力を想定しているので、B 個の 28 × 28 行列を全て 784 次元ベクトルに直す。プログラムは以下のようになる。

Listing 1: データの前処理

```
B = 100

random_elements = random.sample(range(10000), B)

minibatch_index = np.array(random_elements)

minibatch_list = X[minibatch_index]
minibatch_vector_list = []
for x in minibatch_list:
    y = x.reshape(picture_size)
    minibatch_vector_list.append(y)

minibatch_vector_list = np.array(minibatch_vector_list).T / 255
```

2.2 正解データ、one_hot ベクトル

クロスエントロピーを計算するのに必要な one_hot ベクトルを用意しておく。one_hot ベクトルは、10 次元のベクトルで、正解ラベルのインデックスが 1、それ以外が 0 となるようなベクトルである。以下のように、正解ラベルから one_hot ベクトルを生成する。

Listing 2: 正解データと one_hot

```
correct_labels = Y[minibatch_index]

one_hot = np.array([np.eye(10)[y] for y in correct_labels]).T
```

2.3 順伝播の演算

課題 2 からはニューラルネットワークの順伝播を実行するクラスを用いて、各処理はクラス内に関数を定義するなどして行う。
このクラスをインスタンス化する時に、引数に、全結合層のパラメータ、ニューラルネットへの入力データ、その入力データの正解ラベルを one_hot 表記にしたものを指定する。以下では、順伝播の、最後にソフトマックス関数を適用する直前までを記したものである。

Listing 3: 順伝播

```
class Neuralnet:
    def __init__(self, W_1, b_1, W_2, b_2, minibatch_vector_list, one_hot):

        self.one_hot = one_hot

        self.dot_1_list = np.dot(np.array(W_1).T, minibatch_vector_list)
        + b_1[:, np.newaxis]

        self.sigmoid = 1 / (1 + np.exp(-self.dot_1_list))


        self.dot_2_list = np.dot(np.array(W_2).T, self.sigmoid) + b_2[:, np.newaxis]

        self.dot_2_alpha_list = self.dot_2_list - np.max(self.dot_2_list, axis=0)
```

2.4 クロスエントロピー計算

上記の順伝播の過程で行なった演算結果は、以下の関数、softmax_list が返す。その値の自然対数をとり、one_hot ベクトルとの内積を取ると、クロスエントロピーが計算される。バッチ数分のクロスエントロピーを計算し、その平均をとる。

Listing 4: クロスエントロピー計算

```
    def onehot_vector(self):
        return self.one_hot
```

```
    def logistic(self):
        return self.sigmoid

    def softmax_list(self):

        return np.exp(np.array(self.dot_2_alpha_list))/np.sum(np.exp(np.array
        (self.dot_2_alpha_list)), axis=0)

    def closs_entropy(self):

        log_before = np.exp(np.array(self.dot_2_alpha_list)) /
        np.sum(np.exp(np.array(self.dot_2_alpha_list)), axis=0)
        softmax_vector_list = -np.log(log_before)
        result = np.sum(self.one_hot * softmax_vector_list, axis=0)
        return np.sum(result) / B
```

2.5 ラベルの予測

課題 1 と同様のことをバッチ数分だけ行なっている。学習は全く行なっていないので、結果はほとんどランダムである。

Listing 5: ラベルの予測

```
def expectations(self):
        softmaxs = np.exp(np.array(self.dot_2_alpha_list))/
        np.sum(np.exp(np.array(self.dot_2_alpha_list)), axis=0)
        return np.argmax(softmaxs, axis=0)
```

3 実行結果

バッチ平均 (100) のクロスエントロピーは 2.3388948672708336 となった。ランダムのシードを変更して同じバッチ数で同様のことを行なってもクロスエントロピーの値は大きく変わらなかった。また、正解ラベルに対する一致率を計算すると毎回 10-12 %程度に収まり、完全にランダムで予測が進んでいることがわかった。

4 工夫点、問題点

4.1 工夫点

課題 1 と同様、極力行列演算で for 文を用いず、処理が遅くならないようにした。

4.2 問題点

課題 1 と同様に学習を行なっていないので、クロスエントロピーは 2.3 程度から微動だにしない。逆伝播により誤差を伝えることで、学習を進めていく必要がある。また、今回ニューラルネットというクラスを作成してしまっているが、層が深くなるといちいち書き直す量が増えてしまうので非効率であるという点である。次回課題では層ごとのクラス分けを行い実行していく。