計算機科学実験及演習4 画像処理 レポート3

橘大佑
1029-31-6811
2019年度入学

2021/11/30

工学部情報学科計算機科学コース
1029-31-6811
橘 大佑

レポート 3

計算機科学実験及び演習 4
2021 年 11 月 30 日

課題内容

課題 2] のコードをベースに、3 層ニューラルネットワークのパラメータを学習するプログラムを作成した。

プログラムの説明

課題 2 のコードに加えて、mini_batch 関数を編集した。課題 2 との差分のみを以下で説明する。

- mini_batch 関数
  以下の手順を繰り返し行う。繰り返し回数は (画像の枚数)/(バッチサイズ) としたので、楽観的には全ての画像が教師データとして利用されるということになる。
  - ミニバッチを作成
  - ミニバッチに対する出力を順伝播によって計算
  - 損失関数の値 E_n を算出
  - 式 $\frac{\delta E_n}{\delta a_k} = \frac{y_k^{(2)} - y_k}{B}$ から $\frac{\delta E_n}{\delta a_k}$ を計算
  - 式 $\frac{\delta E_n}{\delta X} = W_2^T \frac{\delta E_n}{\delta a_k}, \frac{\delta E_n}{\delta W_2} = \frac{\delta E_n}{\delta a_k} X_2^T, \frac{\delta E_n}{\delta b} = rowSum(\frac{\delta E_n}{\delta a_k})$ から、3 つの値を計算
  - シグモイド関数の出力を y として、式 $\frac{\delta E_n}{\delta x} = \frac{\delta E_n}{\delta y}(1-y)y$ を $\frac{\delta E_n}{\delta x}$ の各要素に適用する。
  - 式 $\frac{\delta E_n}{\delta X} = W_1^T \frac{\delta E_n}{\delta a_k}, \frac{\delta E_n}{\delta W_1} = \frac{\delta E_n}{\delta a_k} X_1^T, \frac{\delta E_n}{\delta b} = rowSum(\frac{\delta E_n}{\delta a_k})$ から、3 つの値を計算
  - 学習率 η をとして、式 $W_1 = W_1 - \eta \frac{\delta E_n}{\delta W_1}, W_2 = W_2 - \eta \frac{\delta E_n}{\delta W_2}, b_1 = b_1 - \eta \frac{\delta E_n}{\delta b_1}, b_2 = b_2 - \eta \frac{\delta E_n}{\delta b_2}$ を用いてパラメータを更新する。

  また、各エポックの処理が終了する毎に、クロスエントロピー誤差を標準出力に出力するようにした。そして、学習した 4 つのパラメータを学習終了時に保存されるようにした。そして、学習開始時にそのデータを用いるか、それともランダムなパラメータから新たに学習するかを入力から受け取ることができるようにした。”Y” と入力すれば学習済みのパラメータを、それ以外の入力では一から学習するようになっている。

Listing 1: mini_batch

```
def mini_batch():
  global W_2, W_1, b_1, b_2
  i = input('Do you reuse parameter files?(Y/N) ')
  if i == 'Y':
    print('Yes')
    W_1 = np.load('W_1.npy')
    W_2 = np.load('W_2.npy')
    b_1 = np.load('b_1.npy')
    b_2 = np.load('b_2.npy')
  else:
    print('No')

  for e in range(epoch_size):
```

```
    epoch_error = 0
    for j in range(iters_number):

      random_list = np.array(random.sample(range(60000),B))
      error = 0

      dE_da = np.empty((C, 0), dtype=np.float32)
      X_2 = np.empty((M, 0), dtype=np.float32)
      X_1 = np.empty((d, 0), dtype=np.float32)

      for i in range(B):
        index = random_list[i]
        output_l = np.array(fully_connected_layer_2(fully_connected_layer_1(
            input_layer(index))))
        after_sigmoid = np.array(fully_connected_layer_1(input_layer(index))).
            reshape(M,1)
        before_sigmoid = np.array(input_layer(index)).reshape(d,1)
        y = Y[index]
        one_hot = np.array([0] * C)
        one_hot[y] = 1
        #new_func = np.frompyfunc(log_fun, 1, 1)
        #z = new_func(output_l)
        #error -= np.dot(z, one_hot)

        error -= math.log(output_l[np.argmax(one_hot)])
        new_l = np.array(output_l - one_hot).reshape(C,1)
        dE_da = np.append(dE_da, new_l, axis=1)
        X_2 = np.append(X_2, after_sigmoid, axis=1)
        X_1 = np.append(X_1, before_sigmoid, axis=1)
      E = error/B
      epoch_error += E

      dE_dX = np.dot(W_2.T, dE_da)
      dE_dW_2 = np.dot(dE_da, X_2.T)
      dE_db_2 = np.sum(dE_da, axis=1)

      dE_da2 = dE_dX * (1-X_2) * X_2

      dE_dX = np.dot(W_1.T, dE_da2)
      dE_dW_1 = np.dot(dE_da2, X_1.T)
      dE_db_1 = np.sum(dE_da2, axis=1)

      W_1 = W_1 - learning_rate * dE_dW_1
      W_2 = W_2 - learning_rate * dE_dW_2
      b_1 = b_1 - learning_rate * dE_db_1
      b_2 = b_2 - learning_rate * dE_db_2
    print(e+1)
    print(epoch_error/iters_number)
```

```
60      plt.plot(e,epoch_error/iters_number,"b",marker='.')
61      plt.title('Cross-Entropy Error',fontsize=15)
62      plt.xlabel('epoch',fontsize=10)
63      plt.ylabel('cross-entropy error',fontsize=10)
64
65    np.save('W_1',W_1)
66    np.save('W_2',W_2)
67    np.save('b_1',b_1)
68    np.save('b_2',b_2)
69    print("Parameter files are saved.")
70    plt.show()
```

実行結果

以下が実行したときの出力である。

```
1
0.49165225622647624
2
0.2031574896329253
3
0.15415935528554164
4
0.1313546749481394
...
17
0.04281335536638935
18
0.04109429087650737
19
0.03551777292748371
20
0.037029892308096515
```

上は学習済みのパラメータを利用した場合である。エントロピー誤差は 0.49 から 0.03 まで減少しているのが分かる。また、これをプロットしたものが図 1である。

工学部情報学科計算機科学コース
1029-31-6811
橘 大佑

レポート 3

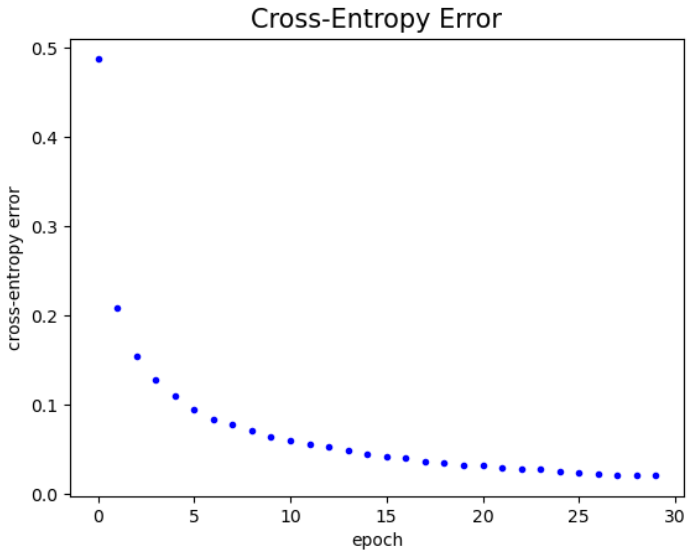

図 1: クロスエントロピー誤差の推移

工夫点

課題 2 では 2 つの配列の要素ごとに計算する必要があったが、配列 y_k は 10 個の要素のうち、i つだけが 1 で、そのほかの 9 個は 0 であるため、1 となる要素のインデックスを取り出し、それを用いてベクトルの計算を行った。

問題点

バッチサイズ (B) だけ計算するのに for 文を用い、しかも繰り返し操作でも for 文を用いたため、for 文の入れ子構造ができてしまい、かなり学習に時間がかかった。