計算機科学実験及演習4「画像認識」 課題2

工学部 情報学科 計算機科学コース 3回生
山田瑛平 学籍番号:1029282731

平成30年11月1日

課題内容

以下のようなプログラムを作成する.

- MNIST の学習画像 60000 枚からミニバッチとして B 枚取り出す.
- 各画像についてクロスエントロピー誤差を計算し, その平均を出力する.

プログラムの説明

当課題のプログラムは課題 1 のものをベースにしているため, 主として課題 1 と変更があった箇所について述べる.

定数

当プログラムでは定数を以下のように定義している.

```
SIZEX = 28
SIZEY = 28
PIC = 60000
MID1 = 5
MID2 = 10

BATCH_SIZE = 100
```

バッチサイズを新たに定義した変数 $BATCH\_SIZE$ に格納している. 今回はバッチサイズを 100 とした.

クロスエントロピー誤差関数

クロスエントロピー誤差関数 $crossentropy(t, y)$ は, *one-hotvector* 表記で表される正解のベクトル t と, 出力層のソフトマックス関数適用後の値を各要素に持つベクトル y を入力に取り, クロスエントロピー誤差を出力する関数である.

```
def crossentropy(t, y):
    delta = 1e-7
    return -np.sum(np.dot(t, np.log(y + delta)))
```

微小量 $delta$ を y に加算するのは, 関数 log の引数が 0 である場合に NAN が出力されるのを防ぐためである.

ミニバッチの選択

60000 枚の画像からバッチサイズの数だけ画像をランダムに取り出す工程が新たに必要となる. 以下はそれを実現するソースコードである.

```
num_arr = np.random.randint(0, PIC - 1, BATCH_SIZE)
```

.randint メソッドを用いて, 0 から 59999 までの数字の中からバッチサイズの数だけ (重複を許さずに) 値を取り出し, 配列 num_arr に格納している. ミニバッチから画像を取り出して, 各画像の列ベクトルをバッチサイズの数だけ横に並べた 784×100 (= バッチサイズ) 行列 X を作成する.

```
def picx(i):
    return X[i]
x_input = np.apply_along_axis(picx, 0, num_arr)
x_input = x_input.reshape((BATCH_SIZE, SIZEX * SIZEY))
```

中間層

入力画像の行列 X を 784×100 行列に拡張したため, W_1X が 5 (= 中間層の数)×100 行列となり, b_1 を 5×100 行列変更する必要が生じる. よって, b_1 を各列の値が全て同じになるような行列に変更する. この工程は以下のようなコードで表される.

```
y1 = np.c_[sigarr(w1 @ x_input.T + np.matlib.repmat(b1, 1, BATCH_SIZE))]
```

メソッド *.repmat* により元々のベクトル b_1 をバッチサイズの数だけ横方向にコピーしている. なお, b_2 についてもこれは同様である.

出力層

平均クロスエントロピー誤差の出力のため, ミニバッチ内の画像それぞれについての正解を *one-hotvector* 表記で表したベクトルを行ベクトルとし, そのベクトルを縦に積むことで完成する 100×10 行列 k を作成する. ここではあらかじめ空の 100×10 行列を作成した上で各画像について正解の箇所に 1 を代入していくことで実装している. 平均クロスエントロピー誤差は, 出力をソフトマックス関数に適用した 10×100 行列 $sofy2$ と k(を対角化した行列) とのアダマール積の要素の平均で表される.

```
k = np.zeros((BATCH_SIZE, MID2))
count = 0
for n in num_arr:
    k[count, Y[n]] = 1
    count += 1
entropy_average = (-1) * np.sum(k.T * np.log(sofy2)) / BATCH_SIZE

print(entropy_average)
```

実行結果

概ね 2.3 から 2.5 にかけての値が出力される. これは, 乱数で重みを決定しているため, 画像識別の精度が 0.1 程度であり, $\log 0.1$ が $-2.3025\cdots$ であることに起因していると考えられる.

工夫点および問題点

課題 1 のプログラムと同じく, 仕様変更可能性を高める工夫を行った. バッチサイズや中間層の数を容易に変更できるような設計になっているのはその顕著な例である.