計算機科学実験及演習 4 画像処理 レポート A2

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 A2

計算機科学実験及び演習 4
2021 年 11 月 30 日

課題内容

活性化関数として Dropout を用いて、テストと学習に分けてプログラムを作成した。

プログラムの説明

学習時は中間層のノードをランダムに選び、その出力を無視した。今回 50 個のノードのうちの 3 割が無視されるように設定した。テスト時は全てのノードを無視することなく、代わりに出力を 0.7 倍したものを出力として用いた。既存のパラメーターを用いてテストするか、一から学習を行うかどうかはコマンドで入力を受け取り、”TEST”と入力された場合にはテストを、”LEARN”と入力された場合は学習を行うようにした。上の 2 種類以外の文字列が入力された場合は再び入力を促すように設定した。

- first_question 関数
  while を用いて、正しい入力を受け取るまで、入力を促すようにした。正しい入力がされた場合は別のメソッド実行し、while ループから抜け出すようにした。

Listing 1: first_question

```
def first_question():
  while(True):
    i = input('Learn by Dropout algorithm, or test accuracy?(LEARN or TEST) ')
    if i == 'LEARN':
      mini_batch()
      final_input()
      break
    elif i == 'TEST':
      show_accuracy()
      break
    else:
      print("Try Again.")
```

- fully_connected_layer_1_TEST 関数
  学習ではなく既存のパラメーターを用いてテストを行う際に用いる関数である。1 次元配列を受け取って、重み付けしてから dropout を適用している。単に出力全体に $(1-\eta)$ をかけているだけである。

Listing 2: fully_connected_layer_1_TEST

```
def fully_connected_layer_1_TEST(input_vector):
  global W_1, b_1
  W_1 = np.load('W_1.npy')
  b_1 = np.load('b_1.npy')
  y_1 = (1-rho) * (np.dot(W_1, input_vector) + b_1)
  return y_1
```

- fully_connected_layer_1 関数
  テストではなく学習を行う際に用いる関数である。1 次元配列を受け取って、重み付けしてから dropout を適用している。今回は ρ 割の確率で出力を 0 にし、残りの $1-\rho$ に関してはそのままでなにもしないという操作であった。これを実現するために、以下の方法を用いた。

- ノードの数 (M 個) だけ、0 から 1 の範囲で乱数の配列を作成する。
- 上で作った配列の要素についてそれぞれ ρ より大きいかどうかを計算し、True または False からなる配列にする。
- 上の配列と元の配列を掛け合わせることで、3 割の割合で要素を 0 にし、残り $1-\rho$ 割はそのままとなる配列をつくるとができた。

Listing 3: fully_connected_layer_1

```
def fully_connected_layer_1(input_vector):
  global W_1, b_1, true_or_false
  ratio = np.random.rand(M)
  true_or_false = ratio > rho
  y_1 = true_or_false *((np.dot(W_1, input_vector) + b_1))
  return y_1
```

- show_accuracy 関数
  テスト終了時に正しい数字が出力されたかどうかを数え、正答率を出力する。パラメーターを用いてテストを行う際は 60000 枚の画像データを用いたため、実行にかなり時間がかかるので、今 60000 枚の画像のうち何枚目なのかを 1 行ごとに出力していると見にくいので、上書きして表示するようにした。また、その結果を円グラフでも表示するようにした。

Listing 4: show_accuracy

```
def show_accuracy():
  true_cnt = 0
  false_cnt = 0
  pro_size = 60000
  for i in range(pro_size):
    estimated_ans = output_answer(fully_connected_layer_2_TEST(
       fully_connected_layer_1_TEST(input_layer(i))))
    real_ans = Y[i]
    print("\r"+str(i+1),end="/60000...")
    time.sleep(0.00)
    if estimated_ans == real_ans:
      true_cnt += 1
    else:
      false_cnt += 1
  print("")
  print("Test is done.")
  print("Hit : ", true_cnt)
  print("Miss :", false_cnt)
  print("Accuracy is ", true_cnt/(true_cnt+false_cnt),".")
```

実行結果

以下がテストを実行したときの出力である。

レポート A2

```
Learn by Dropout algorithm, or test accuracy?(LEARN or TEST) TEST
60000/60000...
Test is done.
Hit :  23434
Miss : 36566
Accuracy is  0.3905666666666667 .
```

Accuracy の通り、40 パーセントの確率で推定値と正しい値は一致する。Dropout を使わない場合は 90 パーセント程度であったので、Dropout では正答率はあまり高くなかった。
以下が学習した際の出力である。

- $\rho = 0.0$

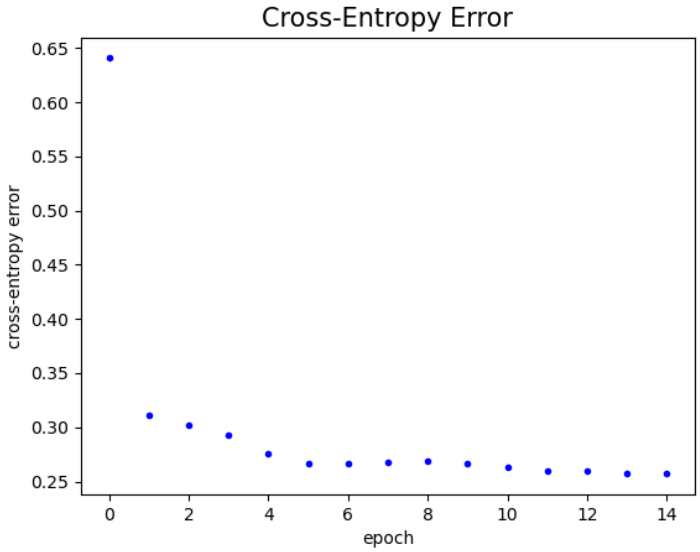

図 1: ドロップアウトを用いた際のクロスエントロピー誤差の推移 ($\rho = 0.0$)

図 1から、エントロピー誤差が最初は 0.63 程度であったのが、15 回の試行後には 0.26 程度まで減少しているのが分かる。2 から 15 回目のエポックの計算ではエントロピー誤差は減少していない。

- $\rho = 0.1$

```
Learn by Dropout algorithm, or test accuracy?(LEARN or TEST) LEARN
Do you reuse parameter files?(Y/N) N
No
1
0.6866242950249714
```

```
2
0.3444378252829289
3
0.32611476630673164
...
13
0.32568463038873235
14
0.328844464664752
15
0.31870147274343946
```

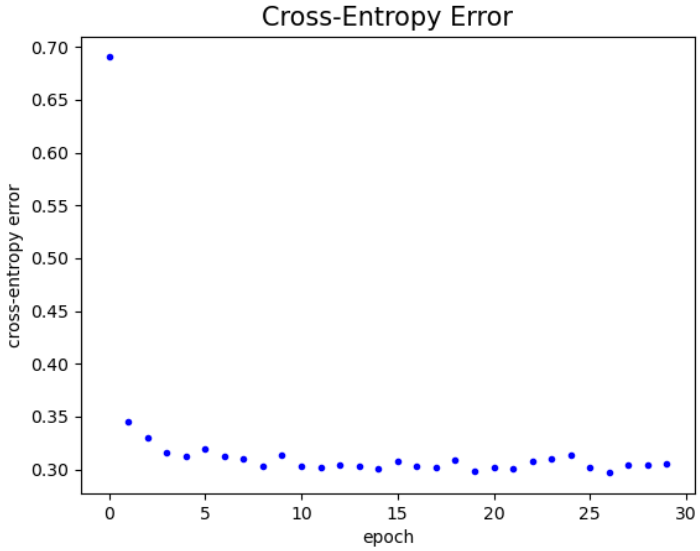

図 2: ドロップアウトを用いた際のクロスエントロピー誤差の推移 ($\rho = 0.1$)

図 2から、エントロピー誤差が最初は 0.68 程度であったのが、30 回の試行後には 0.3 程度まで減少しているのが分かる。5 から 30 回目のエポックの計算ではエントロピー誤差は減少していない。

- $\rho = 0.3$

レポート A2

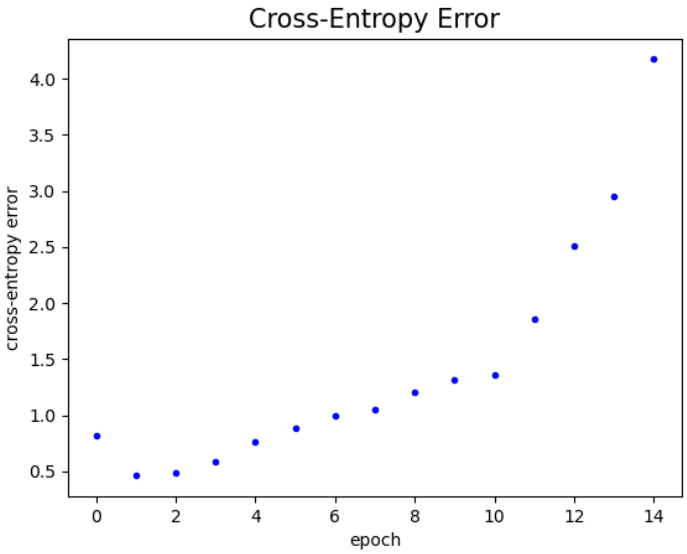

図 3: ドロップアウトを用いた際のクロスエントロピー誤差の推移 ($\rho = 0.3$)

図 3から分かるように ρ が大きすぎると、エントロピー誤差が発散していることが分かる。

工夫点

- 学習係数 η の設定
  学習係数 η が大きすぎる (0.01 程度) と、log 関数の中身が 0 未満になり、エラーが表示された。また、一度収束した後に発散することが分かった。逆に η が小さすぎると、学習が進まないので、適度な学習係数を設定することが重要である。

- ρ の設定
  過学習を防止するための Dropout であるが、ρ を大きくとると、逆にエントロピー誤差が発散することが分かったので、0.1 程度に設定した。

問題点

- 計算量
  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。具体的な計算量は (繰り返し回数)×(バッチサイズ)×(エポックの回数)=120000(教師データのサイズ) となる。

- エントロピー誤差の振動
  図 2から分かるように、2 回目まではエントロピー誤差は単調減少しているが、そこから 15 回目までは振動するだけで誤差は小さくなっていない。