画像実験課題 2

1029323422 天野岳洋

2022 年 12 月 22 日

1　概要

課題 1 からコードを改良し, ミニバッチを入力可能とするように変更し, さらに, クロスエントロピー誤差を計算するプログラムを作成せよ.

2　要件定義

- MNIST の学習画像 60000 枚からランダムに B 枚をミニバッチとして取り出す.
- クロスエントロピー誤差の平均を標準出力に出力

3　実装の方針

ミニバッチとしてランダムな B 枚を取得するためには,0〜59999 からランダムな整数を B 個並べて配列として,MNIST の学習画像から取得すればよい. このミニバッチの形は (B, 28, 28) となっている. これを行列計算が簡単な (B, 28*28, 1) の形に変更してあげる. なぜ簡単になるのかというと, np.matmul の定義から (M, 28*28) と (B, 28*28, 1) の行列積が (B, M, 1) として取得できるからである. これは, ミニバッチのそれぞれ一枚ずつに対して, 重みをかけている作業を一回でできていることとなる. これで全結合層はクリアである.Sigmoid 関数は (B, M, 1) のそれぞれ一つずつに対して同じ関数を適用させているので, これもまた簡単に実装が可能である. そして後述する方法で Softmax を適用させればミニバッチへの拡張は完了である.

クロスエントロピー誤差に関しては何らかの方法で onehot-vector を作成し, クロスエントロピー誤差を計算し出力してやればよい. この方法は以降のセクションで詳細に述べる.

4　実装

各種準備

Listing 1　準備

```
1  import numpy as np
2  import mnist
3  #randseed
4  seed = 601
5  M = 100
6  C = 10
7  B = 100
8  np.random.seed(seed)
9  X = mnist.download_and_parse_mnist_file("train-images-idx3-ubyte.gz")
```

```
10  Y = mnist.download_and_parse_mnist_file("train-labels-idx1-ubyte.gz")
11  W1 = np.random.normal(loc = 0, scale = np.sqrt(1/img_size), size=(M,
        img_size))
12  b1 = np.random.normal(loc = 0, scale = np.sqrt(1/img_size), size=(M, 1))
13  W2 = np.random.normal(loc = 0, scale = np.sqrt(1/M), size=(C, M))
14  b2 = np.random.normal(loc = 0, scale = np.sqrt(1/M), size=(C, 1))
```

各種 import と変数準備である.B はミニバッチサイズである. そのほかの説明はレポート 1 で既に行った.

ランダムミニバッチ作成

ミニバッチの作成手続きを行うコードをいかに示す.

Listing 2　ミニバッチ

```
1  #pattern1
2  batch_random = np.random.randint(0, 60000, B)
3  #pattern2
4  batch_random = np.random.choice(np.arange(0, 60000), B, replace = False)
5  before_conv = np.array(X[batch_random])
6  answer = np.array(Y[batch_random])
```

実装としては二つのパターンが考えられる. 一つはミニバッチ内の重複を許すパターン. もう一つはミニバッチ内の重複を許さないパターン. 上のコードの pattern1 が許すパターンで pattern2 が許さないパターンである. このようにして取得した,before_conv, answer は (B, 28, 28), (B,) という形をしている.

onehot-vector

onehot-vector を作成する方法の説明である. コードは以下のとおりである.

Listing 3　onehot-vector

```
1  onehot = np.zeros((answer.size, 10))
2  onehot[np.arange(answer.size), answer] = 1
```

まず onehot-vector の形を確認する. これは, ミニバッチの一枚に対して, 9 個の 0 と 1 個の 1 が並ぶ形となる. まずこの形 (B, C) の形をした値がすべて 0 である行列を作る.

次に 2 行目で適切な場所を 1 に変えている. この作業を詳しく説明すると, まず answer.size は B に等しい. つまり 100 である. なので, np.arange(answer.size) は [0,1,2,3,..., 99] という形をしている. 次に,answer は (B,) であり, ミニバッチ一枚目の数字, 二枚目の数字... が並んだ [6, 9, 3...] という形をしている. これら二つを用いて指定した場所を 1 にしている. さらに詳しく説明すると, ミ

ニバッチ一枚目の六行目を 1 に, ミニバッチ二枚目の九行目を 1 に,... という作業をしていることとなる.

ソフトマックス手前まで

三層 NN の最後の層であるソフトマックス層までの実装方法について説明する. これは設計の方針で述べた通りの設計である.

Listing 4　伝播 1

```
img = before_conv.reshape((B, img_size, 1))
input1 = np.matmul(W1, img) + b1
output1 = 1/(1 + np.exp(-1 * input1))
input2 = np.matmul(W2, output1) + b2
```

一行目では (B, 28, 28) から (B, 764, 1) へと形を変更している. 次に全結合層では,np.matmul を用いることによって, ミニバッチに対して重みの行列積とバイアスの足し算を行っている. この説明を詳しく行いたいと思う. bias をひとまず無視する. この時すべき計算は次のとおりである.

$$Y_{ij1} = \sum_k w_{jk} * x_{ik1}$$

np.matmul では片方が三次元以上であるならば, これを行列のスタックとして扱うとレファレンスに書いてあるので, まさに上の計算をやるには np.matmul が適切である. ほかに, 次のやり方でもうまくいくはずである.

Listing 5　別解

```
Y = np.einsum("jk, ikl -> ijl", W, X)
```

シグモイド, 全結合層 2 の説明は省略する.

ソフトマックス

このセクションではソフトマックス関数の実装を行う. ソフトマックス関数では alpha が現れるので, 単純な設計ではうまくいかない.

Listing 6　SoftMax

```
alpha = np.repeat(input2.max(axis = 1), C, axis= 1).reshape(B, C, 1)
sumexp = np.repeat(np.sum(np.exp(input2 - alpha), axis=1), C, axis=1).
    reshape(B, C, 1)
output_last = np.exp(input2 - alpha) / sumexp
output_last = np.reshape(output_last, (B, C))
```

まずは alpha の生成から, alpha はミニバッチ内の各 10 の値から最大のものを取得するので, input2.max(axis = 1) が必要である. さらに alpha を用いた引き算が必要なので, これを 10 回

axis=1 の方向に repeat して reshape して input2 と形を同じにする. 以下の作業の簡略的なものは次のとおりである.

$$input = \begin{bmatrix}\begin{bmatrix}0.1\\0.7\\0.5\\0.4\end{bmatrix}\end{bmatrix}, alpha = \begin{bmatrix}\begin{bmatrix}0.7\\0.7\\0.5\\0.5\end{bmatrix}\end{bmatrix}$$

次に, この alpha を用いて sumexp を計算する. これもまた同様に, 同じ形にした状態で計算を行う. こうすることによって, ソフトマックスが計算できるようになった. 最後に onehot-vector の形が (B, C) であるのでこの形に変形しておく.

クロスエントロピー誤差

ここは定義通りであるので, 説明は省略する.

Listing 7　CrossEntropy

```
print((-1/B)* np.sum(onehot * np.log(output_last)))
```

5　実際の動作

Listing 8　Actual Move

```
2.4802906574896886
```

全く学習していないパラメータの場合, 予測値は確率が 1/10 であたるので, クロスエントロピーは次の値が妥当である $-\log\frac{1}{10} = 2.3025\ldots$ よっておそらく正しく計算できていると予測できる.