課題 2 レポート

羽路 悠斗

2021 年 12 月 21 日

1　課題内容

1.1　ミニバッチ対応

課題 1 のコードをベースにミニバッチを入力可能とするように改良せよ。

- MNIST の学習画像 60000 枚の中からランダムに B 枚をミニバッチとして取り出すこと。
- バッチサイズ B は自由に決めて良い (100 程度がちょうど良い)。
- ミニバッチを取り出す処理はランダムに行う。

1.2　クロスエントロピー誤差の計算

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。

- クロスエントロピー誤差の平均を標準出力に出力すること。

2　作成したプログラムの説明

2.1　ミニバッチ対応

2.1.1　設計方針

各層の入力と出力のインターフェースを、axis=-1 をバッチサイズとして拡張する。すなわち、行列の各列が 1 画像を表す。

2.1.2　パラメータ

パラメータとして、バッチサイズを追加する。重みとバイアスは、バッチ内の全ての画像に同じ値を利用するので構造の変更はない。

2.1.3　前処理

ミニバッチとして取り出す画像を決める。(バッチサイズ、高さ、幅) の形で画像を、(バッチサイズ、クラス数) の形で one-hot vector 表記でラベルを取り出す。

ソースコード 1　ex2.py

```
1    def preprocessing():
2      idx = random.randint(0, len(train_y), batch_size)
3      tr_x = train_x[idx]
4      l = [[1 if i == label else 0 for i in range(10)] for label in train_y[idx]]
5      tr_y = np.zeros((len(l), len(l[0])))
6      tr_y[:] = l
7      return tr_x, tr_y # tr_x is (bs, 28, 28) tr_y is one-hot-encoding (bs, 10)
```

2.2　クロスエントロピー誤差の計算

2.2.1　設計方針

出力層の出力はインターフェースに従って、(クラス数、バッチサイズ) の形で one-hot vector 表記で与えられる。定義に従い、クラスとバッチに沿って加算すれば良い。

2.2.2　クロスエントロピー誤差

クロスエントロピー誤差は次の式で計算される。

$$E_n = \frac{1}{B} \sum_{i \in \text{ミニバッチ}} \sum_{k=1}^{C} -y_{i,k} \log y_{i,k}^{(2)}$$

これをコードにすると次のようになる。

ソースコード 2　ex2.py

```
1    def cross_entropy(true_vec ,pred_vec):
2      return np.sum(np.sum(-1 * true_vec * np.log(pred_vec), axis=0)) / batch_size
```

3　実行結果

実際にミニバッチを取り出して、クロスエントロピー誤差を標準出力に出力するプログラムの、実行と実行結果は次の通りである。

ソースコード 3　ex2.py

```
1      def model():
2        tr_x, tr_y = preprocessing()
3        return cross_entropy(tr_y.T, softmax(dense_layer2(sigmoid(dense_layer1(
             input_layer(tr_x))))))
```

ソースコード 4　ex2.py

```
1      if __name__ == '__main__':
2        stdout = model()
3        print(f'cross␣entropy␣of␣a␣batch␣is␣{stdout}')
```

実行結果

```
cross entropy of a batch is 2.6056293937236092
```

4　工夫点

各層の入力と出力のインターフェースを、課題 1 のインターフェースの axis=-1(今回の場合は行列の列となる) にバッチサイズを追加して統一することで、課題 1 のコードをほぼそのまま利用することができた。また、各層でノードが縦に並んでおり右に伝搬していくという教科書の図と同じイメージが持てるという利点もある。

課題 1 と同じく、for 文を一度も使わずに行列演算を駆使したことで、高速化できた。

5　問題点

上で for 文を一度も使わずと述べたが、前処理での one-hot vector 表記への変換にはリスト内包表記を用いている。numpy をうまく使えば高速化できるかもしれない。