課題 1

浦川樹

2021 年 12 月 17 日

1　課題内容

キーボードから 0〜9999 の整数を入力 i として受け取る．MNIST のテストデータ 10000 枚の画像のうち i 番目の画像を入力として，乱数で決定された重みをもつニューラルネットワークを順伝播させ，0〜9 の整数を標準出力に出力する．

2　順伝播 (ネットワーク関数)

順伝播の数学的な定義を述べる．後ほどの誤差逆伝搬のレポートでわかりやすいように実験資料とはすこし違う記号を用いている．入力ベクトルの次元を d，中間層のユニット数を M，クラス数を C とする．まず入力変数 $x_i(i = 1, \ldots, d)$ の線形和を計算する．

$$a_j^{(1)} = \sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)} = \sum_{i=0}^{d} w_{ji}^{(1)} x_i (j = 1, \ldots, M)$$

ここで x_0 は $x_0 = 1$ として固定した便宜上の入力変数である．次にこれを活性化関数 $h(\cdot)$ で変換し，

$$y_j^{(1)} = h(a_j^{(1)})$$

を得る．そしてこれらの線形和を再び計算する．

$$a_k^{(2)} = \sum_{j=1}^{M} w_{kj}^{(2)} y_j^{(1)} + w_{k0}^{(2)} = \sum_{j=0}^{M} w_{kj}^{(2)} y_j^{(1)} (k = 1, \ldots, C)$$

ここで $y_0^{(1)}$ は $y_0^{(1)} = 1$ として固定した便宜上の入力変数である．活性化関数 $\sigma(\cdot)$ で得られた線形和を変換することで出力 $y_k^{(2)}$ を得る．

$$y_k^{(2)} = \sigma(a_k^{(2)})$$

今回の課題では活性化関数 $h(\cdot)$ にシグモイド関数

$$h(t) = \frac{1}{1 + \exp(-t)}$$

活性化関数 $\sigma(a_k^{(2)})$ にソフトマックス関数

$$\sigma(a_l^{(2)}) = \frac{\exp(a_l^{(2)} - \alpha)}{\sum_{k=1}^{C} \exp(a_k^{(2)} - \alpha)}$$

$$\alpha = \max a_k^{(2)}$$

を用いる．また以降線形和の重み $w_{ji}^{(1)}(j = 1, \ldots, M, i = 0, \ldots, d)$ を (j, i) 成分とする $M \times (d + 1)$ 行列を $W^{(1)}$，$w_{kj}^{(2)}(k = 1, \ldots, C, j = 0, \ldots, M)$ を (k, j) 成分とする $C \times (M + 1)$ 行列を $W^{(2)}$ と表す．

$$W^{(1)} = \begin{pmatrix} w_{10}^{(1)} & w_{11}^{(1)} & \cdots & w_{1d}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & \cdots & w_{2d}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M0}^{(1)} & w_{M1}^{(1)} & \cdots & w_{Md}^{(1)} \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} w_{10}^{(2)} & w_{11}^{(2)} & \cdots & w_{1M}^{(2)} \\ w_{20}^{(2)} & w_{21}^{(2)} & \cdots & w_{2M}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{C0}^{(2)} & w_{C1}^{(2)} & \cdots & w_{CM}^{(2)} \end{pmatrix}$$

3 プログラムの仕様

3.1 プログラムの構成

課題 1 の実装はニューラルネットワークが実装されている ireco パッケージの neural モジュールと課題 1 の本体の処理が記述されている kadai1.py からなっている (ireco というパッケージ名は思いつきでつけたもので深い意味はないです)．neural モジュールには順伝播を行う関数 predict() が実装されている．kadai1.py では入力データの処理などを行った後 predict() 関数を用いて順伝播を行っている．

3.2 入力の処理と重みの初期化:kadai1.py

kadai1.py において標準入力で与えられた数字に対応する画像データを 784 次元のベクトルに変換する．また乱数によって $W^{(1)}$，$W^{(2)}$ を決定する．

3.3 順伝播の実装:ireco.neural.predict()

neural モジュールに順伝播を行う predict() を実装した．predict() は，各行が一つの入力ベクトルである行列 X と重み $W^{(1)}$，$W^{(2)}$ を入力として受け取り，行列 X の各行のベクトルごとに順伝播を行う関数である．predict() 内部で線形和を計算する lsum()，シグモイド関数を計算する sigmoid()，ソフトマックス関数を計算する softmax() を用いて順伝播を計算している．

$$\mathrm{lsum}(x, W) = \begin{pmatrix} \sum_{m=1}^{q} w_{1m} x_m + w_{10} \\ \vdots \\ \sum_{m=1}^{q} w_{pm} x_m + w_{p0} \end{pmatrix}$$

lsum() は上のような計算を行う．ここで $x = {}^t(x_1, \ldots, x_q)$，$W = (w_{lm})$ である．

$$\mathrm{sigmoid}(x) = \begin{pmatrix} h(x_1) \\ \vdots \\ h(x_q) \end{pmatrix}$$

sigmoid() は上のような計算を行う．ここで $x = {}^t(x_1, \ldots, x_q)$，$h(t)$ は 2 における 1 次元のシグモイド関数である．

$$\mathrm{softmax}(a) = \begin{pmatrix} \sigma(a_1) \\ \vdots \\ \sigma(a_C) \end{pmatrix}$$

softmax() は上のような計算を行う．ここで $a = {}^t(a_1, \ldots, a_C)$，$\sigma(\cdot)$ は 2 におけるソフトマックス関数である．predict() は入力ベクトルに lsum()，sigmoid()，lsum()，softmax() を順に用いることで，2 の計算を行っている．

4 実行結果

次のような結果が得られた．

```
Enter 0~9999: 4
1
Enter 0~9999: 85
9
```

5 工夫点

5.1 モジュール化

今回の実験では後ほどの課題でコードを再利用することになると考えたため，今後追加される部分も含めニューラルネットワークの実装部分はすべて neural モジュールに統一して記述することにした．各課題の実行の本体は/tests ディレクトリ配下に作り，各課題の実行プログラムで neural モジュールを利用するほうが全体の見通しが良くなると考えた．

5.2 ベクトル行列演算

Numpy が提供するベクトル行列演算を活用した実装を行った．順伝播の計算の本体である lsum()，sigmoid()，softmax() はすべてベクトル同士の演算で実装している．for ループを用いた記述をしていないため，関数の定義がすっきりとしている．

6 問題点

6.1 モジュール化に関わる問題点

今後 neural モジュールに機能を追加していくことになるが，機能追加前と互換性を持たせる必要がある．あるいは機能追加後にそれ以前の各課題の実行プログラムをすべて変更し，neural モジュールの変更に対応する必要がある．どちらにせよ少し煩わしい作業となる．

6.2 関数への予期しない入力

predict() 関数は 3 つの ndarray を引数に取る．predict() 関数では入力の ndarray が想定している ndarray の shape と等しいかどうかの検査をおこなっていない．そのためもしかしたら想定しない入力に対しても実行時エラーとならずに意味のない値を返してしまうかもしれない．