知能プログラミング演習 I 演習課題

1 準備

1.1 自前の環境の場合

- Moodle から課題を各自任意のフォルダにダウンロードし, 展開したフォルダの中に以下のものがすべて入っていることを確認
  - auto_encoder.py
  - task.pdf

1.2 CSE の場合

- まだ演習用のフォルダを作っていない人は DLL のフォルダを作成
  - ホームディレクトリに演習用のディレクトリを作成

    step1: `mkdir -p DLL`
- 作業ディレクトリ DLL に移動

  step1: `cd ./DLL`
- 今日の課題を DLL にダウンロードして展開
  - 展開したフォルダの中に, 以下のものがすべて入っていることを確認
    - auto_encoder.py
    - task.pdf
- Lec3 へ移動

  step1: `cd ./Lec3`

2 課題

手書き文字 (28× 28 ピクセル) のに対するデノイジングオートエンコーダを adam を用いて実装する．auto_encoder.py にコードを保存すること．このコードでは，$d = 784(= 28 \times 28)$ の入力画像を $q = 64$ 次元の中間層ユニットに圧縮してから復元する．

1. 二乗誤差関数を `ErrorFunction(x, y)` として作成せよ．なお, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ に対して, 二乗誤差関数は以下で定義される．

$$E(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 = \frac{1}{2}\sum_{i=1}^{d}(y_i - x_i)^2$$

   以下はヒントである (自分で適当な np.array 変数を作って確かめること):
   - np.array 同士の引き算は要素ごとに行われる
   - np.array の変数に対して`**2` とすると各要素の二乗が計算される
   - np.sum を使うと np.array の和が計算できる

   `ErrorFunction(x, y)` が正しく定義されていれば (定義した状態で, コンソール上で `auto_encoder.py` を実行した後は)，例えば以下のような結果をコンソールから確かめることができるはずである:

```
In [1]: x = np.array([1,2,3])

In [2]: y = np.array([4,5,6])

In [3]: ErrorFunction(x,y)
Out[3]: 13.5
```

2. スライド 9page を参考に，元データ `x_train[i, :]` にノイズを加え，デノイジングオートエンコーダの順伝播を作成せよ（訓練とテストで 2 箇所あるため注意すること）．加えるノイズは平均 0 標準偏差 0.05 の正規分布によって生成するものとする．例えば，以下のようにすると平均 0 標準偏差 0.1 の長さ 5 の乱数ベクトルを作ることができる．

```
In [1]: np.random.normal(0,0.1,5)
Out[1]: array([ 0.00670631,  0.18137909, -0.01784399,  0.00315327, -0.06428307])
```

   順伝播計算に使う関数 `forward` は（正しく作れていれば）前回 DNN.py で作成したものをそのまま使うことができる．入力層から中間層の活性化関数 f はシグモイドとする（関数がすでに記載済み）．今回は最終層の活性化関数は恒等関数を考えるため，ネットワークの出力は直接 $\tilde{\boldsymbol{x}} = W_1\boldsymbol{z}_1$ となることに注意せよ（`np.dot` を使う）．`ErrorFunction` のコメントを外す（訓練とテスト，2 箇所）と初期の誤差が error.pdf に表示されるようになるので確認しておくこと．
3. スライド 10page を参考に逆伝播を完成させ，まずは，これまでと同じ確率勾配降下法によるデノイジングオートエンコーダーを完成させよ．確率勾配降下法によるパラメータ更新はコメントとしてすでに記載されているので，逆伝播を完成させたらコメントを外すだけでよい．関数 `backword` は（正しく作れていれば）前回 DNN.py で作成したものをそのまま使うことができる．完成すると error.pdf に誤差の推移が保存されるので，減少していくことを確認すること．
4. スライド 13page を参考に，Adam によるパラメータの更新則を関数 `adam` として作成せよ．以下はヒ

ントである．

- 出力は, 更新されたパラメータ ($W_0^{(t)}$ または $W_1^{(t)}$) および, 勾配の情報 (m_t, v_t) である．
- t はすでに定義され適切なタイミングで increment されているのでそのまま使えばよい
- m や v は層ごとに存在する（W_0 と W_1 それぞれに m と v がある）
- np.array の各要素をある a 乗するには，np.array の変数に対して`**a` と後ろにつければよい
- `np.sqrt` に np.array 変数を渡せば，要素毎の平方根が得られる

3. で単純な確率勾配降下法を使った場合と比べて誤差の減少の度合いの違いを確認すること．

5. auto_encode.py の先頭付近で定義されている `use_small_data` を False にして，データ全体を使って実行し，以下を確認せよ．
   - error.pdf を見て誤差が減少することを確認すること．確率的勾配降下と adam で違いを比較せよ．
   - original.pdf と reconstruct.pdf に元画像と，対応する復元画像が保存されるので確認すること（テスト出力の計算後に plot 用の変数を保存する部分があるのでコメントを外して有効化しておくこと）．また適当に，データを変えて結果を見てみること（auto_encoder.py の下部「元画像と復元画像の保存」付近の変数 j を変える）．
   - W_0 のヒートマップが W0.pdf に，W_1 のヒートマップが W1.pdf に保存される．どのようなパラメータが学習されているかを確認し，どう解釈できるのか考えてみること．

3 課題の提出

Moodle を使ってファイルを提出してください. 提出方法は以下の通りです.

- Moodle にログインし, 知能プログラミング演習のページへ移動.
- Lec3 の項目に, auto_encoder.py をアップロードする.

6/27(金) の 17:00 を提出期限とします.