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
  - RNN.py
  - report.pdf
  - report.tex
  - 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
    - RNN.py
    - report.pdf
    - report.tex
    - task.pdf
- Lec4 へ移動

  step1: `cd ./Lec4`

2 課題

2.1 データセットの準備

RNN を用いて音声分類を行う．今回用いる TI-46 Word[*1]データセットは，500 個の音声信号データを含む．これらは，5 人の発話者 (speaker) が “zero” から “nine” までの 10 個の数字単語 (digit) を 10 回ずつ発話 (utterance) したときの音声信号に対応する．音声分類では，音声信号を周波数成分ごとの強度に分解し，スペクトログラム[*2]やコクリアグラム[*3]に変換してから機械学習モデルを適用することが多い．ここでは，TI-46 Word の音声信号を変換したコクリアグラムのデータを利用する．以下の方法等でダウンロードする．

- ブラウザ
  `https://github.com/dsiufl/Reservoir-Computing` にアクセスして緑色の Code ボタンを押し，zip ファイルをダウンロードする．これを解凍して，フォルダ `Lyon_decimation_128` を作業フォルダ直下に移動する．
- コマンドプロンプト
  `> git clone https://github.com/dsiufl/Reservoir-Computing.git`
  フォルダ `Reservoir-Computing` ができるので，その中のフォルダ `Lyon_decimation_128` を作業フォルダ直下に移動する．

各ファイル名は `s(発話者番号)_u(発話番号)_d(発話数字).mat` となっている．フォルダ内に全部で 500 ファイルあることを確認する．コマンドプロンプトで

```
> ls | wc -w
```

とすれば，そのフォルダ内のファイル数をカウントできる．

プログラム内では，1 から 10 の発話番号のうち，一部を訓練用 (`utterance_train`)，残りをテスト用 (`utterance_test`) とする．例えば，発話番号 1 から 7 を訓練用，8 から 10 をテスト用とするとき，訓練データ数は 350，テストデータ数は 150 となる．

2.2 プログラム作成

以下のプログラムを作成せよ．ただし, RNN.py にコードを保存すること．

1. RNN の順伝播を考える．まず，スライド 8 page の $\boldsymbol{z}'_t (t = 0, \ldots, T)$ を並べた $q \times (T+1)$ 行列

$$Z' = \begin{bmatrix} \boldsymbol{z}'_0 & \boldsymbol{z}'_1 & \cdots & \boldsymbol{z}'_T \end{bmatrix}$$

   を変数 `Z_prime` として，また，後の誤差逆伝播で使用する $q \times T$ 行列

$$\nabla f = \begin{bmatrix} \nabla f(\boldsymbol{u}_1) & \ldots & \nabla f(\boldsymbol{u}_T) \end{bmatrix}$$

   を変数 `nabla_f` として作成せよ．ただし，スライドにある通り，$\boldsymbol{u}_t = W^{\mathrm{in}} \boldsymbol{x}_t + W \boldsymbol{z}'_{t-1}$ である．それぞれ，1 列ずつ関数 `forward(x, z_prev, W_in, W, actfunc)` で生成する（$\boldsymbol{z}'_0 = \boldsymbol{0}$ であるため，Z'

[*1] Linguistic Data Consortium (`https://catalog.ldc.upenn.edu/LDC93S9`)
[*2] `https://ja.wikipedia.org/wiki/スペクトログラム`
[*3] `https://en.wikipedia.org/wiki/Computational_auditory_scene_analysis`

と ∇f は一列ずつずれることに注意)．関数は forward は $\boldsymbol{z}'_t$ と $\nabla f(\boldsymbol{u}_t)$ の二つを返り値として返すとする．活性化関数 f として，sigmoid は RNN.py に記載済みなのでこれを用いればよい（これまでのものと同じ)．訓練とテストで同様の処理を 2 箇所作成する必要があるので注意．ヒント: 行列を表現する 2 次元の np.array 変数から特定の行や列だけにアクセスする方法について以下に例を示す:

```
In [1]: z = np.zeros((3,4))

In [2]: z # 3x4 の 0 行列を生成
Out[2]:
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])

In [3]: z[:,1] = np.array([1,2,3]) # (0 から数えて)1 列目に [1 2 3] を代入

In [4]: z
Out[4]:
array([[0., 1., 0., 0.],
       [0., 2., 0., 0.],
       [0., 3., 0., 0.]])

In [5]: z[1,:]
Out[5]: array([0., 2., 0., 0.]) # (0 から数えて)1 行目を表示
```

2. RNN の逆伝播を考える．スライド 9 page を参考に $\boldsymbol{\delta}^{out}$ と $\boldsymbol{\delta}_t (t = 1, \ldots, T)$ を作成せよ．$\boldsymbol{\delta}_t (t = 1, \ldots, T)$ は $q \times T$ 行列として

$$\Delta = \begin{bmatrix} \boldsymbol{\delta}_1 & \ldots & \boldsymbol{\delta}_T \end{bmatrix}$$

を持つ変数 delta として作成する．関数 backward(W, W_out, delta, delta_out, derivative) を用いて，$t = T$ から $t = 1$ まで逆順に計算する（python の添え字では $T - 1$ から 0 になることに注意)．この関数は $t = T$ の場合とそれ以外の両方を同じ関数で計算するようになっている．呼び出し元の引数の渡し方と，スライド 9 page の式をよく見て，関数 backward を作成せよ．

3. スライド 9 page を整理して以下のように表記する

$$\frac{\partial E}{\partial W^{\text{out}}} = \boldsymbol{\delta}^{out} \boldsymbol{z}_T^\top = \boldsymbol{\delta}^{\text{out}} \begin{bmatrix} 1 \\ \boldsymbol{z}'_T \end{bmatrix}^\top$$

$$\frac{\partial E}{\partial W^{\text{in}}} = \sum_{t=1}^{T} \boldsymbol{\delta} \boldsymbol{x}_t^\top = \begin{bmatrix} \boldsymbol{\delta}_1 & \ldots & \boldsymbol{\delta}_T \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_T^\top \end{bmatrix} = \Delta X$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \boldsymbol{\delta}_t \boldsymbol{z}_{t-1}'^\top = \Delta \tilde{Z}'^\top$$

ただし，$\tilde{Z}'$ は Z' から最後の列のみ除いた $q \times T$ 行列とし，また，ここでの $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T]^\top$ は定数項を含んだ $T \times (d+1)$ 行列である．それぞれを変数 dEdW_out, dEdW_in, dEdW に作成せよ．ここまでで，一通りの処理が完成するので，誤差の推移や ConfusionMatrix の図を確認すること（この段階では，ConfusionMatrix を見るとうまく分類できてない可能性が高いので, 出力されていることだけ確認できればよい)．

4. RNN.py の初期状態では単純な確率勾配降下法になっているが，前回作成した adam に変更し，結果の変化を確認せよ（注: 誤差推移は多少ガタつくこともある）．RNN は計算が重いので，最初は訓練用発話番号リスト `utterance_train` の要素数や `epoch` を少なくしておき，増加させていくとよい．
5. (任意課題) こちらの課題は任意とする．提出があれば加点するが，なくても減点等はしない．作成した RNN.py の様々な設定を自由に変えて，結果を考察せよ．解析結果のレポートを作成し, pdf ファイルで提出せよ．tex や word など何を使用して作成してもよいが，大まかにでよいので report.pdf のような体裁で整えること（report.pdf を生成した tex も含まれているので，使いたければこれを使ってもよい）．なお, レポートには少なくとも**次の 3 つの節は必ず含めること**:
   1. 実験設定
   2. 結果
   3. 考察

   また，以下のことに留意すること．
   - 設定した中間層の数や, 中間層ごとのユニット数, 各層で用いた活性化関数などの**実験設定を正確に記述すること．**
   - **解析結果の図や表を用いること** (例えば, 誤差関数の推移や confusion matrix など).
   - 解析結果に対する**考察を述べること**.
   - 口語表現は使用しない．

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.
- Lec4 の項目に RNN.py をアップロードする.
- 任意提出課題を行った場合はレポートの PDF もアップロードする.

7/4(金) の 17:00 を提出期限とします.