The Xerion dataset manger manual

浅川 伸一 Shin Asakawa asakawa@ieee.org

2018 年 12 月 23 日

1 はじめに

Xerion とは (Plaut, McClelland, Seidenberg, & Patterson, 1996) で用いられたデータセットの運用を目指した Python ライブラリです[*1]。**Xerion** は UNIX X window 上で動作するニューラルネットワークシミュレータです。この文章でオリジナルの **Xerion** で用いられたデータを Python で扱い易い形に整形したものです。ここではこのデータセットを xerion として公開しました。

1.1 インストール方法

https:/github.com/ShinAsakawa/wbai_aphasia からダウンロードしてください。インストールのオペレーションをいかに示します。

```
git clone https://github.com/ShinAsakawa/wbai_aphasia
cd wbai_aphasia
python setup.py install
```

データディレクトリとして，datadir と pkl_dir を指定します。前者はオリジナルデータの保存場所であり，後者は pickle ファイルを保存するディレクトリです。デフォルトでは ./data になっています。

1.2 簡単な使い方

必要最低限の使い方を示します。

```
1  import numpy
2  import wbai_aphasia as handson
3
4  from sklearn.neural_network import MLPRegressor
5
6  data = handson.xerion()
7  X = np.asarray(data.input, dtype=np.float32)
8  y = np.asarray(data.output, dtype=np.float32)
9
10 model = MLPRegressor()
11 model.fit(X,y)
12 model.score(X,y)
```

上記の例では，2 行目で本稿で説明しているデータセット管理ライブラリを読み込み，6 行目で data に代入しています。そのデータを 7, 8 行目で scikit-learn で使えるデータに変換しています。10 行目で scikit-learn の 3 層パーセプトロンモデルを呼び出し，11 行目で学習をさせ，12 行目で結果を表示させています。

結果を示すと以下のようになりました。

[*1] オリジナルデータ http://www.cnbc.cmu.edu/~plaut/xerion/xerion-3.1-nets-share.tar.gz

```
Out[2]:
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
Out[2]: 0.8349035046783984
```

1.3 データ構造

上の例でもある通り Python のクラス wbai_aphasia.xerion() を呼び出すことでデータを読み込むことができます。

データ数は, homograph が 13 存在します。データは wbai_ahasia.xerion().input と wbai_aphasia.xerion().output で定義されています。

データの次元はそれぞれ以下のようになります。

```
In [1]: import numpy as np
   ...: import wbai_aphasia.wbai_aphasia as handson
   ...: data = handson.xerion()

In [2]: print(data.input.shape, data.input.dtype)
   ...: print(data.output.shape, data.output.dtype)
(2998, 105) int16
(2998, 61) int16
```

上記から分かる通りデータは 2998 件あり，入力データ次元は 105 出力データ次元は 61 あります。それぞれのコーディング方法を以下に示します。

```
In [3]: for x in data.Orthography:
   ...:     print(x, data.Orthography[x], len(data.Orthography[x]))
   ...:
onset ['Y', 'S', 'P', 'T', 'K', 'Q', 'C', 'B', 'D', 'G', 'F', 'V', 'J', 'Z', 'L',
↪ 'M', 'N', 'R', 'W', 'H', 'CH', 'GH', 'GN', 'PH', 'PS', 'RH', 'SH', 'TH', 'TS',
↪ 'WH'] 30
vowel ['E', 'I', 'O', 'U', 'A', 'Y', 'AI', 'AU', 'AW', 'AY', 'EA', 'EE', 'EI',
↪ 'EU', 'EW', 'EY', 'IE', 'OA', 'OE', 'OI', 'OO', 'OU', 'OW', 'OY', 'UE', 'UI',
↪ 'UY'] 27
coda ['H', 'R', 'L', 'M', 'N', 'B', 'D', 'G', 'C', 'X', 'F', 'V', 'ʃ', 'S', 'Z',
↪ 'P', 'T', 'K', 'Q', 'BB', 'CH', 'CK', 'DD', 'DG', 'FF', 'GG', 'GH', 'GN', 'KS',
↪ 'LL', 'NG', 'NN', 'PH', 'PP', 'PS', 'RR', 'SH', 'SL', 'SS', 'TCH', 'TH', 'TS',
↪ 'TT', 'ZZ', 'U', 'E', 'ES', 'ED'] 48

In [4]: for x in data.Phonology:
   ...:     print(x, data.Phonology[x], len(data.Phonology[x]))
   ...:
onset ['s', 'S', 'C', 'z', 'Z', 'j', 'f', 'v', 'T', 'D', 'p', 'b', 't', 'd', 'k',
↪ 'g', 'm', 'n', 'h', 'I', 'r', 'w', 'y'] 23
vowel ['a', 'e', 'i', 'o', 'u', '@', '^', 'A', 'E', 'I', 'O', 'U', 'W', 'Y'] 14
coda ['r', 'I', 'm', 'n', 'N', 'b', 'g', 'd', 'ps', 'ks', 'ts', 's', 'z', 'f', 'v',
↪ 'p', 'k', 't', 'S', 'Z', 'T', 'D', 'C', 'j'] 24
```

1.4 意味 semantics とのリンク

言語情報処理を考える場合の入出力について考える。入力は，視覚 (画像や動画などの場合と文字や記号などの 2 つに大別される)，聴覚 (音声，音楽)，運動感覚がある。一方，出力は，音声，運動 (書字，描画，指差し，視線の移動

など) がある。これらの入出力を分類アルゴリズムを通して学習可能であると考えるのが一般的である。入出力のそれぞれをモダリティ (modality，様相，様式) というが生理学における感覚入力の別に加えて，細分類がなされることもある。たとえば，視覚における，明るさ，色，線分の方位，聴覚における周波数帯域などは別モダリティと考えられる場合もある。各モダリティは独立した入力装置と見なさる場合が多い。

2 非単語

2.1 非単語の読み

スペルから発音を得る方法はいくつかありますが，一般的に用いられている `nltk` パッケージの中にも `cmudict` があります。

以下の関数を呼び出すことによって，非単語の読みについての近似的な発音を得ることが出来ます。

```
from nltk.corpus import cmudict
```

https://stackoverflow.com/questions/33666557/get-phonemes-from-any-word-in-python-nltk-or-other-modul

```
import nltk
from functools import lru_cache
from itertools import product as iterprod

try:
    arpabet = nltk.corpus.cmudict.dict()
except LookupError:
    nltk.download('cmudict')
    arpabet = nltk.corpus.cmudict.dict()

@lru_cache()
def wordbreak(s):
    s = s.lower()
    if s in arpabet:
        return arpabet[s]
    middle = len(s)/2
    partition = sorted(list(range(len(s))), key=lambda x: (x-middle)**2-x)
    for i in partition:
        pre, suf = (s[:i], s[i:])
        if pre in arpabet and wordbreak(suf) is not None:
            return [x+y for x,y in iterprod(arpabet[pre], wordbreak(suf))]
    return None

for word in not_in_cmu:
print(word, wordbreak(word))
```

2.2 非単語の意味表象

レーベンシュタイン距離 (別名編集距離) Levenshtein (1965/1966) を用いる方法がある。ただし，レーベンシュタイン距離が人間の類似度と一致しているわけではありませんが，近似としてはあり得るかも知れません。

3 意味

https://fasttext.cc/docs/en/pretrained-vectors.html Bojanowski, Grave, Joulin, & Mikolov (2016); Joulin et al. (2017); Joulin, Grave, Bojanowski, & Mikolov (2016)

```
word2vec_file = 'wiki.en.vec'
semantics = {}
n = 0
with codecs.open(word2vec_file, 'r') as f:
    for line in f:
        if n > 0:
            buff = line.strip().split(' ')
            word = buff[0]
            if word in orthography:
                semantics[word] = np.asarray([float(x) for x in
                ↪ buff[1:]],dtype=np.float32)
        n += 1
```

4 まとめ

昨今の深層学習，機械学習のモデルは人間に比肩する性能を示すようになってきた。本稿では，このような情勢に鑑み，神経心理学モデルを機械学習モデルを用いて構築する試みを考えた。

本稿の試みのように，神経心理学モデルに対して機械学習モデルを適用することにより，診断補助，治療計画立案，自立支援，評価手法の確立と提案，などこの分野に対する貢献が可能だろうと考えている。これにより，患者とその家族，脳神経外科医師，言語聴覚治療士，作業療法士，理学療法士，神経心理学者，高次脳機能障害に関心を持つ機械学習関係者との相互交流が促進され，これらの領域が活性化することが期待できる。本来，同じ目的を持つ領域を相互に結びつけることで，あらたな発展が関係するすべての領域にとって刺激的で良好な関係を構築できるのではないかと期待している。

引用文献

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1705.00823*.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jëgou, H., & Mikolov, T. (2017). FASTTEXT.ZIP: Compressing text classification models. In Y. Bengio & Y. LeCun (Eds.), *The proceedings of International Conference on Learning Representations (ICLR)*. Toulon, France.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals (D. A. Nauk, Trans.). *Soviet Physics Doklady*, *10*, 707–710. (Original work published 1965)

Plaut, D., McClelland, J. L., Seidenberg, M. S., & Patterson, K. (1996). Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review*, *103*, 56–115.

Seidenberg, M. S., & McClelland, J. L. (1989). A distributed, developmenta model of word recognition and naming. *Psychological Review*, *96*, 523–568.