第 1 章

機械学習でμ's の声を識別する

hideo54

1.1　はじめに

こんにちは、hideo54 です。灘高校で高校 2 年生をしています。最近の口癖は「受験したくない」です。これまでずっと SunPro 会誌に記事を出してきましたが、もしかすると高校生で SunPro 会誌の記事を執筆するのはこれが最後かもしれない……?!

さて、僕は今年 1 月の NHK での再放送をきっかけとして今更ラブライブ！ にハマってしまったわけですが、この記事ではタイトル通り、μ's のメンバーの声を学習し、誰が歌っているのか判別できるようにした話をします。

この手の分野に対して全くの素人である僕がやる気になった動機として、学校の物理の授業で音について学習した際、先生が PC を持ち込んで、声の波形や、人によって波形が異なる様子を実演してくれたことで、なんとなくしか知らなかった、頑張ったら音を機械的に解析できそうだということが実感できたという出来事があります。そしてアニソンを聞いていて、歌っている人を識別できたら面白そうだなと思いました。「アニメ声」という言葉が存在するように、アニメの声は特徴的な声が多く、識別が容易そうな気がします。

数ある声優ユニットの中からμ's を選んだのには、以下の理由があります:

- 1 人 1 人の声が割りと異なるので識別が容易そう
- メンバー 1 人のみが歌っている「ラブライブ！ Solo Live!」というアルバムが各メンバーごとにあり、iTunes で視聴可能なので、サンプル音声の収集が楽そう

というわけで、次の節から、実際に解析を始めてみようと思います！　ただ、前述の通り僕はまだ初心者なので、読者の方にこの道のプロの方がいらっしゃれば、至らぬ点を見つけたらこの記事の終わりの方に書いてある連絡先の方に気軽に斧を投げていただければ幸いです。

また、以降の記事では、以下を利用しました:

- Python 3.5.2
- numpy (言わずと知れた計算お役立ちライブラリ)
- pylab (グラフ描画用に matplotlib を使用するためのモジュール)
- librosa (音声処理用)
- requests (サンプル集める用に API を叩く時に使用)
- ffmpeg (音声ファイル変換用)
- rpca_svs[*1] (Python 実装の歌声分離プログラム)
- scikit-learn (機械学習用)

詳しい話は使用箇所で述べています。
これらのインストールに必要な過程は省略します。

1.2　どうやんの?

高校物理で習うように、音 (音波) は波動の 1 つです。声の高さは振動数に、大きさは振れ幅に、特徴は波形に現れます。

たとえば、「あー」という声の波形を見てみましょう。ここでは、アニメ 1 期第 4 話「まきりんぱな」での、小泉花陽さんと西木野真姫さんの発声練習の部分から短時間分を切り抜いた声を見てみます。次のような Python コードを書きました:

リスト 1.1: 「あー」の波形

```
import wave
import numpy
import pylab

def dataFromWAV(filename):
    f = wave.open(filename, 'r')
    buf = f.readframes(f.getnframes())
    return numpy.frombuffer(buf, dtype='int16')

fig, (l,r) = pylab.subplots(ncols=2,figsize=(15,5))
l.plot(dataFromWAV('maki.wav'))
l.set_title('MAKI')
l.set_xlim(15000, 16000)
r.plot(dataFromWAV('hanayo.wav'))
r.set_title('HANAYO')
r.set_xlim(15000, 16000)
fig.savefig('a.png')
```

切り取りの都合で横軸を 15000 からにしてます。以上を実行して生成される画像が以下になります。

[*1] https://github.com/hiromu/rpca_svs

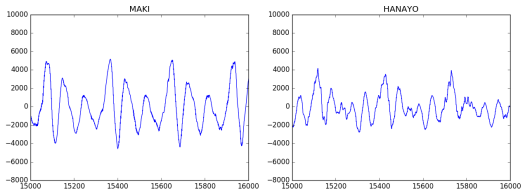

図 1.1: 「あー」の波形

確かに、波形が異なるものの、どちらも周期的な波であるということが確認できますね。(小泉花陽さんが小さめに歌ってる様子も振れ幅からわかります)

この波形 (=声の特徴) を扱いやすくするために、メル周波数ケプストラム係数 (MFCC) というものを使うと良いらしいです。波形をフーリエ変換することで得られるスペクトラム[*2]の対数値をさらにフーリエ変換して得られたケプストラムを、メル尺度を使って変換したものだとか。なるほどわからん。そこで、超お手軽に MFCC を計算できるライブラリ“librosa”を使ってみます。(これ使わずに numpy で一からやろうとしたけど挫折した。)

リスト 1.2: librosa を使った MFCC の算出

```
import librosa
import numpy

def getMfcc(filename):
    y, sr = librosa.load(filename)
    return librosa.feature.mfcc(y=y, sr=sr)

maki = getMfcc('maki.wav')
```

超簡単ですね。とりあえず、様々な曲から MFCC を得るため、次の節でひとまずサンプルを集めます。

1.3 各メンバーの声を集める

前述の通り、各メンバーがソロで歌っている「ラブライブ！ Solo Live!」というアルバムがあります。そのうち全アルバムに共通している曲 (全員のソロが用意されている曲) は以下の 11 曲です:

- もぎゅっと“love”で接近中！
- 愛してるばんざーい！
- Wonderful Rush
- Oh,Love&Peace!
- 僕らは今のなかで
- WILD STARS
- きっと青春が聞こえる
- 輝夜の城で踊りたい
- Wonder zone
- No brand girls
- START:DASH!!

これらの試聴用音声 (30 秒[*3]) を、iTunes Search API[*4]を利用して落としてきます。

リスト 1.3: 各 Solo live!のダウンロード

```
import os
import requests

artists = ['HONOKA', 'ELI', 'KOTORI', 'UMI', 'RIN',
    'MAKI', 'NOZOMI', 'HANAYO', 'NICO']
songs = [
    ['もぎゅっと"love"で接近中!',
        'もぎゅっとloveで接近中'],
    ['愛してるばんざーい!', '愛してるばんざーい'],
    ['Wonderful Rush', 'Wonderful-Rush'],
    ['Oh,Love&Peace!', 'Oh-Love-and-Peace'],
    ['僕らは今のなかで', '僕らは今のなかで'],
    ['WILD STARS', 'WILD-STARS'],
    ['きっと青春が聞こえる', 'きっと青春が聞こえる'],
    ['輝夜の城で踊りたい', '輝夜の城で踊りたい'],
    ['Wonder zone', 'Wonder-zone'],
    ['No brand girls', 'No-brand-girls'],
    ['START:DASH!!', 'START-DASH']
]

def download(url, filename):
    res = requests.get(url, timeout=10)
    with open(filename, 'wb') as f:
        f.write(res.content)

timeouts = []

for artist in artists:
    os.mkdir(artist)
    for song in songs:
        print('Downloading: %s(%s Mix)'
            % (song[0], artist))
        baseurl = 'https://itunes.apple.com/search'
        params = {
            'term': '%s(%s Mix)'
                % (song[0], artist),
            'country': 'JP',
            'media': 'music',
            'entry': 'song'
        }
        res = requests.get(baseurl, params=params)
        result = res.json()['results'][0]
        previewUrl = result['previewUrl']
        try:
            download(previewUrl,
                '%s/%s.m4a' % (artist, song[1]))
        except requests.exceptions.ReadTimeout:
            print('Timeout: %s(%s Mix)'
                % (song[0], artist))
            timeouts.append({
                'url': previewUrl,
                'song': song,
                'artist': artist
            })

for timeout in timeouts:
    url = timeout['url']
    original_title = timeout['song'][0]
    short_title = timeout['song'][1]
    artist = timeout['artist']
    print('Downloading again: %s(%s Mix)'
        % (original_title, artist))
    try:
        song = download(url, '%s/%s.m4a'
            % (artist, short_title))
    except requests.exceptions.ReadTimeout:
        print('Timeout again, download manually: '
            + timeout['url'])
```

よくわからないのですが、たまにタイムアウトするので、タイムアウトした分はもう 1 度落とすようにしています。それでも無理なものは数分後くらいにタイムアウ

[*2] ちなみに、「はじめに」で述べた、学校の物理の授業で見せてもらったのはこれです。

[*3] iTunes Store だと 90 秒の試聴ができるんですが、公開されている API だと 30 秒しか落とせないそうです。(Apple 社員の人が stack overflow で言ってた http://stackoverflow.com/a/14620405 。) 悲しいなあ。

[*4] https://affiliate.itunes.apple.com/resources/documentation/itunes_search_api_jp/

トが解消されたら適当に curl とかで拾ってやります。

また、ファイル名に!や", :, スペースなどが入っていると扱いづらくなるので、適宜手動で代わりのファイル名 (songs の各配列の 2 番目) を与えました。各メンバーごとにディレクトリを作成し、その中にファイルを突っ込むようにしています。

これで、各メンバーにつき 10 曲の 30 秒試聴用音声がダウンロードされました。ありがたいことに試聴部分は曲ごとにすべて統一されています。

ただこの楽曲ファイルには、メンバーの声だけでなく BGM が多分に含まれているので、精度の高い学習が難しそうです。Instrumental 版 (歌声なし版) が提供されていれば歌声のみを抽出できるものの、残念ながら提供されていません……。そこで hiromu に相談したところ、Matlab で書かれた音声分離プログラム[*5]の Python 実装版を作ってくれた[*1](!) ので、それを使ってほぼ音声のみのファイルを生成します。

まず m4a を wav に変換するために ffmpeg を使いたいので、この過程はシェルスクリプトを使用することにします。

リスト 1.4: wav に変換し音声分離

```
members=(HONOKA ELI KOTORI UMI RIN MAKI NOZOMI
HANAYO NICO)
songs=(
    'もぎゅっとloveで接近中'
    '愛してるばんざーい'
    'Wonderful-Rush'
    'Oh-Love-and-Peace'
    '僕らは今のなかで'
    'WILD-STARS'
    'きっと青春が聞こえる'
    '輝夜の城で踊りたい'
    'Wonder-zone'
    'No-brand-girls'
    'START-DASH'
)

for member in "${members[@]}"
do
    for song in "${songs[@]}"
    do
        ffmpeg -i "$member/$song.m4a" \
            "$member/$song.wav"
        python separation.py "$member/$song.wav" \
            "$member/$song-voice.wav" /dev/null
    done
done
for member in "${members[@]}"
do
    ffmpeg -i
        "$member/もぎゅっとloveで接近中-voice.wav" \
        -ss 15 -t 15 \
        "$member/もぎゅっとloveで接近中-short-voice.
wav"
done
```

ついでに色々やっているので、少し説明:

- まず ffmpeg を用いて m4a を wav に変換
- rpca_svs プログラム (separation.py) を用いてほぼ音声のみを含むファイルを生成
- 「もぎゅっと“love”で接近中！」のみ、試聴曲 30 秒のうち半分は歌ってないので、後半 15 秒のみにカット

分離された音声を試しに 1 つ聞いてみるとわかるのですが、結構いい感じに音声だけ出ていて良い。この RPCA プログラムは統計的処理に基づいて自動で音声分離を行っているらしいのですが、楽曲ファイル 1 つのみを渡すだけでこうやって高精度の分離ができるのは素晴らしいなあと思いました。

これで無事 90 曲分の音声データを手に入れることができました。お次はいよいよ学習です。

1.4 学習させてみる

集めた 99 曲のうち“No brand girls”を除いた 90 曲の MFCC を、scikit-learn を使って、SVM(サポートベクトルマシン) を用いて学習します。[*6] その後、テストデータに含まれていない“No brand girls”の各メンバーのソロ曲を与え、返ってきた MFCC の各要素が誰にあたるかを予測させ、最も多かった予測を歌い手の予想とします。

まず、各サンプルの MFCC の値を求めます。

リスト 1.5: 各 MFCC 値の算出

```
import scipy.io.wavfile as wav
import librosa
from sklearn.svm import SVC
import numpy

def getMfcc(filename):
    y, sr = librosa.load(filename)
    return librosa.feature.mfcc(y=y, sr=sr)

artists = ['HONOKA', 'ELI', 'KOTORI', 'UMI', 'RIN',
'MAKI', 'NOZOMI', 'HANAYO', 'NICO']
songs = [
    'もぎゅっとloveで接近中-short',
    '愛してるばんざーい',
    'Wonderful-Rush',
    'Oh-Love-and-Peace',
    '僕らは今のなかで',
    'WILD-STARS',
    'きっと青春が聞こえる',
    '輝夜の城で踊りたい',
    'Wonder-zone',
    'START-DASH'
]

song_training = []
artist_training = []
for artist in artists:
    print('Reading data of %s...' % artist)
    for song in songs:
        mfcc = getMfcc('%s/%s-voice.wav'
            % (artist, song))
        song_training.append(mfcc.T)
        label = numpy.full((mfcc.shape[1], ),
            artists.index(artist), dtype=numpy.int)
        artist_training.append(label)
song_training = numpy.concatenate(song_training)
artist_training = numpy.concatenate(artist_training)
```

次に、これを学習させた上で、“No brand girls”の各ソロバージョンを与え、誰が歌っているのか推測させます。

[*5] https://github.com/posenhuang/singingvoiceseparationrpca

[*6] scikit-learn の各機能については、http://qiita.com/ynakayama/items/9c5867b6947aa41e9229 が参考になりました。初心者の身としては、インターネット上に資料が多いというのも SVM にした理由の 1 つです。

当初、以下のようなコードを書いていました:

リスト 1.6: 学習と予測

```
svc = SVC()
svc.fit(song_training, artist_training)
print('Learning Done')

for artist in artists:
    mfcc = getMfcc('%s/No-brand-girls-voice.wav'
        % artist)
    prediction = svc.predict(mfcc.T)
    counts = numpy.bincount(prediction)
    result = artists[numpy.argmax(counts)]
    original_title = 'No brand girls(%s Mix)' %
artist
    print('%s recognized as sung by %s.'
        % (original_title, result))
```

学習は、各パラメータの値をデフォルトのままにしていました。また、推測は、曲について得られた MFCC の各要素についてメンバーの推測を立て、最も多い推測をその曲の推測とするようにしていました。

その結果、全ての推測が園田海未さんになってしまいました。そこで Grid Search を使ってパラメータ最適化を行おうかと思ったのですが、hiromu に相談したところ、重い SVM を何度も繰り返して適切なパラメータ値を探索する Grid Search をすると時間がかかりそうだということと、gamma 値を調整すると良い感じになりそうだということを教わりました。さらに推測方法は、曲自体のバイアスを打ち消すため、全員の平均を求めておいて、各曲について推定結果と平均との差が最大になるものを選ぶ方法を取ると良いと教わり、コードで返してくれました。hiromu がいなかったらこの原稿は絶対落ちていましたね……。

以上の要領で色々調整をして最も正答率が高くなったコードが以下のものです。

リスト 1.7: 調整後の学習と予測

```
clf = SVC(C=1, gamma=1e-4)
clf.fit(song_training, artist_training)
print('Learning Done')

counts = []

for artist in artists:
    mfcc = getMfcc('%s/No-brand-girls-voice.wav'
        % artist)
    prediction = clf.predict(mfcc.T)
    counts.append(numpy.bincount(prediction))

counts = numpy.vstack(counts)

for artist, count in zip(artists, counts):
    result = artists[numpy.argmax(
        count-count.mean(axis=0)
    )]
    original_title = 'No brand girls(%s Mix)' %
artist
    print('%s recognized as sung by %s.'
        % (original_title, result))
```

この結果は表 1.1 のようになりました。

全完……！　やったね！

gamma 値ごとの正答率は表 1.2 のようになりました。

ちなみに、1e-5 の時に唯一当たらなかった推測は、星空凛さんを東條希さんとした間違いでした。気持ちはわからんでもない……？

表 1.1: No brand girls 歌手推測結果 (敬称略)

正答	推測結果
高坂穂乃果	高坂穂乃果
絢瀬絵里	絢瀬絵里
南ことり	南ことり
園田海未	園田海未
星空凛	東條希
西木野真姫	西木野真姫
東條希	東條希
小泉花陽	小泉花陽
矢澤にこ	矢澤にこ

表 1.2: gamma 値調整による正答率

gamma 値	正答率
0.05	1/9
1e-4	9/9
1e-5	8/9
1e-6	6/9

声のみから機械学習によって話者を推測するというネタはいくつかあるようですが、楽曲のみを利用しても、前述の音声分離を用いて手軽に話者推測ができたのはすごいなあと思いました。

1.5　おわりに

実は、題材を自分で決めて機械学習するのはこれが初めてでした。圧倒的経験の少なさ故苦労が多かった……。相談に乗ってくれた hiromu には激感謝。hiromu はこの道のプロというイメージがありますが、これに限らず中学の時から今まで、hiromu に締切前に助けてもらったことばかりでした。頼もしい。

慣れていないことを締め切りのある原稿のネタにするんじゃない、という教訓も得られたものの、好きなことを題材にしたということもあって、結構楽しめたかなあと思います。機械学習ムズいけど楽しいので、勉強していきたいです。(現状ライブラリを叩いているだけに等しいのでやがて理論の理解も深めていきたい)

期末試験の都合上、設定されていた最終締切の時点で 1 字も書いてなかったものの、それでも 12/18 という超ギリギリまで待ってくれた会長の hakatashi には本当に感謝しています……！

ここまで読んでいただきありがとうございました！

連絡先

斧は以下にお投げください:

- Email: contact@hideo54.com
- Twitter: @hideo54