Распознавание фонем при помощи классификатора на основе скрытых марковских моделей

Кузьмин А.А.
kuzAleksAleks@gmail.com

6 декабря 2016 г.

Введение

Основная цель данной работы — дать общее понимание того как стохастические модели на основе скрытых марковских моделей (СММ) используются для решения задачи классификации речевого сигнала.
Работа включет в себя:

1. подготовку данных,
2. расчет параметров моделей,
3. классификацию тестовых сигналов,
4. оценку точности классификации.

В качестве языка программирование предлагается использовать язык программирования Python. Этот язык широко используется как для обработки речевых сигналов в частности, так и для анализа данных в целом.

Программного обеспечение

- Python >= 2.6 `https://www.python.org/`
- NumPy >= 1.9.3 `http://www.numpy.org/`

- SciPy >= 0.16.0 https://www.scipy.org/
- scikit-learn >= 0.16 http://scikit-learn.org/stable/
- hmmlearn
  http://hmmlearn.readthedocs.io/en/latest/index.html
  https://github.com/hmmlearn/hmmlearn

Методические указания

Подготовка данных

В качестве данных в данной работы выступают последовательности векторов признаков сигналов. Сигналы соответствуют фонемам русского языка: “а”, “и”, “е”, “м”, “н”, “о”, “р”, “с”, “ы”. Сами данные находятся в файле corpus.json:

```
{
  "a": [
      [
          [
              16.0750675201416,
              -14.44073486328125,
              -13.535823822021484,
              -20.60512351989746,
              10.367831230163574,
              11.35115909576416,
              -16.2695369720459,
```

Каждый сигнал представляет из себя упорядоченную во времени последовательность векторов признаков. В зависимости от длительности звучания фонемы последовательность может быть различной длинны. Каждый вектор — это 13 кепстральных коэффициентов, распределенных по МЭЛ шкале.

Для того что бы считать данные нужно выполнить следующие команды:

```
import json
with open('path/to/corpus.json') as f:
```

```
    corpus = json.load(f)
print corpus.keys()  # ['a', 'i', 'm', 'o', 'n', 's', 'r', 'y', 'je']
```

Каждому ключу словаря corpus соответствует набор последовательностей векторов для соответствующей фонемы.
Например, выборка для фонемы "a" состоит из 224 последовательностей. Первая последовательность состоит из 10 векторов признаков, а размерность каждого вектора равна 13:

```
print len(corpus['a'])        # 224
print len(corpus['a'][0])     # 10
print len(corpus['a'][0][0])  # 13
```

Расчет параметров моделей

CMM с проинициализированными значениями параметров может быть создана следующим образом:

```
from hmmlearn import hmm

model = hmm.GaussianHMM(n_components=3, covariance_type='diag')
```

n_components задает количество состояний в модели, a covarience_type — вид матрицы ковариации: диагональный ('diag') или общий ' full ').

Вызов метода fit позволит применить алгоритм Баума-Уэлча для расчета параметров модели:

```
O = corpus['a'][0]
model.fit(O)
```

Если алгоритм сойдется, эти значения параметров будут являются точкой максимума функции правдоподобия $P(O|\lambda)$, где O — данная обучающей последовательности наблюдений.

В том случае, когда обучающая выборка состоит из нескольких последовательностей, перед вызовом метода fit их необходимо объединить в одну длинную последовательность. Список, в который входят длины соответствующих исходных последовательностей передается в качестве второго аргумента:

```
import numpy as np

O = np.concatenate([corpus['a'][0], corpus['a'][1]])
lengths = [len(corpus['a'][0]), len(corpus['a'][1])]

model.fit(O, lengths)
```

Распознавание

Предположим, что есть обученные модели для фонемы Ph_1, Ph_2. Фонема, к которой относится неизвестный сигнал X, определяется по следующей формуле:

$$Ph = \begin{cases} Ph_1 & \text{if} \quad P(X|Ph_1)P(Ph_1) > P(X|Ph_2)P(Ph_2), \\ Ph_2 & \text{if} \quad P(X|Ph_1)P(Ph_1) < P(X|Ph_2)P(Ph_2) \end{cases}$$

В данной лабораторной работе мы будем предполагать, что появление любой фонемы равновероятно: $P(Ph_i) = P(Ph_j)$.

Как известно, значение любой вероятности принадлежит замкнутому интервалу $[0, 1]$. Что бы избежать проблем, связанных с выходом значений переменных за границы допустимые типом данных double, будут рассчитываться логарифмы вероятностей. И тогда, критерий распознавания фонемы принимает следующий вид:

$$Ph = \begin{cases} Ph_1 & \text{if} \quad \ln P(X|Ph_1) > \ln P(X|Ph_2), \\ Ph_2 & \text{if} \quad \ln P(X|Ph_1) < \ln P(X|Ph_2) \end{cases}$$

Для того, что бы вычислить логарифм вероятности появления последовательности векторов X, при данной модели λ, вызывается метод score:

```
    X = corpus['a'][2]
    print model.score(X)  # -267.95158196241908
```

Оценка точности распознавания

Теперь нужно оценить точность созданного классификатора. Для этого необходимо разделить данные на две непересекающиеся подвыборки: *обучающую* и *тестирующую* в соотношении примерно 80 к 20.

Например, выборка для фонемы “а” состоит из 224 последовательностей. Это значит, что *обучающая* подвыборка будет включать в себя $\lfloor 224 * 0.8 \rfloor = 179$ последовательностей, а *тестирующая*: $224 - 179 = 45$

Точность распознавания *acc*, будет оцениваться так:

$$acc = \frac{n}{N}$$

, где n — количества точных распознаваний, а N — количество всех распознаваний. Последнее совпадает с количеством последовательностей в *тестирующей* подвыборке.