[숙제12] 단순 베이즈 분류기

언어와 컴퓨터

2018년 11월 30일 금요일 13시까지

- 소스코드 스크립트 `nb_class.py`, 데이터 `loglike.pkl` 및 보고서 `hw12_00000.pdf` 파일을 `hw12_00000.zip` 파일로 압축하여 제출하라.
- `nb_class.py` 파일은 17강 실습 코드를 수정하여 사용할 수 있다.
- 보고서는 문제 해결 방법, 코드 설명, 테스트 실행 결과 등을 포함하여 작성하라.

1 당시(唐詩)의 감정 분류

데이터 파일 `poems.txt`[1]는 중국 당나라 시대의 한시 160편에 사람이 붙인 감정 범주가 추가된 코퍼스이다. 각 행은 시의 ID(0—159), 감정 범주(1은 긍정, 0은 부정), 제목, 저자, 본문 다섯 가지 필드로 이루어져 있고, 각 필드는 탭(`\t`)으로 구분되어 있다. Excel 등 스프레드시트를 처리하는 프로그램에서 불러오면 아래와 같은 표의 형태로 표현된다.

0	1	虞世南	蝉	垂緌饮清露，流响出疏桐。居高声自远，非是藉秋风。
1	0	王绩	野望	东皋薄暮望，徙倚欲何依。树树皆秋色，山山唯落晖。
2	1	王绩	秋夜喜遇王处士	北场芸藿罢，东皋刈黍归。相逢秋月满，更值夜萤飞。
3	0	王梵志	吾富有钱时	吾富有钱时，妇儿看我好。吾若脱衣裳，与吾叠袍袄。吾出经

한문을 해독하지 못하는 사람도 통계와 프로그래밍으로 한시의 정서를 예측할 수 있다.
먼저 파이썬에서 아래와 같은 함수를 가져온다.

```
>>> import pickle
>>> from random import seed, shuffle
>>> from collections import Counter, defaultdict
>>> from scipy import log, argmax
```

1.0 코퍼스 준비

1.0.1 파일 읽기 (1점)

`poems.txt` 파일을 읽고 (시의 ID(정수), 감정(정수), 문서(글자의 리스트)) 꼴의 튜플로 이루어진 리스트 `data`를 만든다. 글자 하나를 단어 하나로 간주하며, ‘,’와 ‘。’ 등의 문장 부호도 단어에 포함시킨다.

```
>>> len(data)
160
>>> data[0]
(0, 1, ['垂', '緌', '饮', '清', '露', ',', ',', '流', '响', '出', '疏', '桐', '。', '居',
'高', '声', '自', '远', ',', ',', '非', '是', '藉', '秋', '风', '。'])
```

[1] `https://www.cl.uni-heidelberg.de/~hou/resources.mhtml`에 공개된 “Sentiment lexicon for classical Chinese poetry”의 자료를 가공하였다. 이 자료는 아래의 논문에서 발표된 것이다.

Hou, Y., & Frank, A. (2015). Analyzing sentiment in classical Chinese poetry. In *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)* (pp. 15-24). `http://www.aclweb.org/anthology/W15-3703`

1.0.2 순서 뒤섞기

리스트 data에서 항목들의 순서를 무작위로 뒤섞는다.

```
>>> seed(471)
>>> shuffle(data)
>>> data[0]
(97, 1, ['草', '树', '知', '春', '不', '久', '归', ',', ',', '百', '般', '红', '紫', '斗',
'芳', '菲', '。', '杨', '花', '榆', '荚', '无', '才', '思', ',', ',', '惟', '解', '漫', '天',
'作', '雪', '飞', '。'])
```

1.0.3 훈련 집합과 실험 집합으로 분할하기

코퍼스의 90%가 훈련 집합, 10%가 실험 집합이 되도록 분할한다.

```
>>> boundary = int(len(data) * 0.9)
>>> train = data[:boundary]
>>> test = data[boundary:]
```

1.1 훈련

1.1.1 logprior: 로그사전확률 (1점)

훈련 집합 전체의 문서 개수 Ndoc와 범주별 문서 개수 Nc를 구하고, 이를 이용하여 범주별 로그사전확률을 계산한다. 아래와 같이 $\log P(+) \approx -0.7357$, $\log P(-) \approx -0.6523$이 나오면 된다.

```
>>> Nc
Counter({0: 75, 1: 69})
>>> Ndoc
144
>>> logprior
{1: -0.7357067949787413, 0: -0.6523251860396901}
```

1.1.2 로그가능도

vocabulary: 어휘 목록 (1점) 먼저 훈련 집합에 출현한 어휘(type) 목록 V를 만든다. 만든 결과가 아래와 같으면 된다.

```
>>> len(vocabulary)
1403
>>> vocabulary[:14]
['草', '树', '知', '春', '不', '久', '归', ',', ',', '百', '般', '红', '紫', '斗', '芳']
```

bigdoc: 범주별 문서 통합 (1점) 범주별로 단어(token) 리스트를 값으로 가지는 딕셔너리 bigdoc을 만든다.

```
>>> bigdoc[1][:14]
['草', '树', '知', '春', '不', '久', '归', ',', ',', '百', '般', '红', '紫', '斗', '芳']
>>> bigdoc[0][:14]
['故', '乡', '杳', '无', '际', ',', ',', '日', '暮', '且', '孤', '征', '。', '川', '原']
```

긍정적인 문서에는 총 2941개, 부정적인 문서에는 총 3606개의 단어가 쓰였음을 확인할 수 있다.

```
>>> len(bigdoc[1]), len(bigdoc[0])
(2941, 3606)
```

counts: 범주별 단어 집계 및 평탄화 (2점) 범주별로 단어 빈도표(Counter 자료형)를 값으로 가지는 딕셔너리 counts를 만든다.

문장 부호를 제외하면 긍정적인 시와 부정적인 시에서 人과 春이 공통적으로 자주 쓰였다.

```
>>> counts[1].most_common(5)
[(', ', 211), ('。', 210), ('山', 30), ('人', 26), ('春', 24)]
>>> counts[0].most_common(5)
[(', ', 260), ('。', 239), ('人', 39), ('不', 37), ('春', 32)]
```

긍정적인 문서에는 총 964종, 부정적인 문서에는 총 1011종의 어휘가 쓰였다.

```
>>> len(counts[1]), len(counts[0])
(964, 1011)
```

想과 같이 긍정적인 시에만 쓰인 단어가 있으며, 哀와 같이 부정적인 시에만 쓰인 단어가 있다.

```
>>> counts[1]['想']
1
```

```
>>> counts[0]['想']
0
```

```
>>> counts[1]['哀']
0
```

```
>>> counts[0]['哀']
1
```

이렇게 한 범주에만 나타나는 단어라도 다른 범주에서 양의 확률을 가져야 하므로, V에 속하는 모든 어휘의 빈도에 1을 더하여 counts를 업데이트한다.

앞에서 구한 빈도가 모두 1씩 증가했음을 확인할 수 있다.

```
>>> counts[1].most_common(5)
[(', ', 212), ('。', 211), ('山', 31), ('人', 27), ('春', 25)]
>>> counts[0]['想']
1
>>> counts[1]['哀']
1
```

긍정 범주와 부정 범주의 단어 빈도표의 크기가 1403으로 같아졌다.

```
>>> len(counts[1]), len(counts[0])
(1403, 1403)
```

마침내 모든 $w \in V$에 대하여 $P(w|+)$와 $P(w|-)$를 계산할 수 있게 된다.

loglikelihood: 로그가능도 (2점) loglikelihood는 V의 각 단어 w를 키로 하고, w의 범주별 로그가능도 딕셔너리를 값으로 하는 딕셔너리이다. 이 딕셔너리를 만들어서 피클 파일 loglike.pkl로 저장하자.

```
>>> f = open('loglike.pkl', 'wb')
>>> pickle.dump(loglikelihood, f)
>>> f.close()
```

loglikelihood의 값을 1403개 단어에 대하여 모두 확인할 수는 없으므로, 몇 개를 골라 값을 살펴보자.

$w =$乡에서는 $\log P(w|+) < \log P(w|-)$로 나타난다. 고향에 대한 그리움이 슬픔으로 표현된 것 같다.

```
>>> loglikelihood['乡']
{1: -8.37655086161377, 0: -6.321766996021397}
```

$w =$坐일 때는 $\log P(w|+) > \log P(w|-)$이다. 편안히 앉아 있으면 기분이 좋은 모양이다.

```
>>> loglikelihood['坐']
{1: -6.179326284277552, 0: -8.518991573357617}
```

$w =$醉의 경우 $\log P(w|+)$와 $\log P(w|-)$이 거의 같다. 시인들은 기쁠 때나 슬플 때나 항상 술에 취한다.

```
>>> loglikelihood['醉']
{1: -6.990256500493881, 0: -6.909553660923517}
```

이제 다른 단어에 대한 로그가능도를 스스로 찾아보자. $w =$愁(근심 '수')의 $\log P(w|+)$와 $\log P(w|-)$ 값은 각각 얼마인가? $w =$秀(빼어날 '수')의 $\log P(w|+)$와 $\log P(w|-)$ 값은 각각 얼마인가?

1.2 실험

1.2.1 `results`: 결과 (1점)

`test`는 (시ID, 감정 범주(정답), 본문) 튜플로 이루어진 리스트이다. 훈련 단계에서 구한 로그사전확률 $P(c)$와 로그가능도 $P(w|c)$를 사용하여 각 본문 $testdoc$에 대한 확률 $P(c|testdoc)$를 계산하고, 이 값이 최대가 될 때의 c를 $testdoc$의 감정으로 예측한다. 이 결과를 {시ID: (정답, 예측)} 꼴의 딕셔너리 `results`로 저장한다.

시 111와 144는 정답과 예측이 일치하며, 14는 긍정적인 시인데 부정적인 시로 잘못 예측되었음을 확인할 수 있다.

```
>>> results[111]
(1, 1)
```

```
>>> results[144]
(0, 0)
```

```
>>> results[14]
(1, 0)
```

대화형 인터프리터에서 `results` 전체의 값을 확인하라.

1.3 평가

1.3.1 정확도 (1점)

`results`를 사용하여 분류기의 정확도 `accuracy`를 계산해 보면 80%가 넘는다. 한문을 해독하지 못해도 적절한 데이터를 받으면 한시의 정서를 잘 예측하는 분류기를 만들 수 있다.

```
>>> accuracy > 0.8
True
```

`accuracy`의 실제 값은 얼마인가?

1.3.2 정밀도 및 재현율 (추가 +2점)

여력이 되면 긍정 범주에 대한 정밀도 `precision_pos`와 재현율 `recall_pos`, 부정 범주에 대한 정밀도 `precision_neg`와 재현율 `recall_neg`을 구해 보자.