Spam Filtering

Naive Bayesian Classification 실습

Bayes's Theorem

P(A | B)를 알고 있을때 P(B | A)를 구할 수 있다

P(A)는 사건 A가 일어날 확률
P(B)는 사건 B가 일어날 확률
P(A | B)는 사건 A가 일어났을 때 사건 B가 일어날 확률
P(B | A)는 사건 B가 일어났을 때 사건 A가 일어날 확률

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

예. 한 학교의 학생이 남학생인 확률이 P(A)라고 하고, 학생이 키가 170이 넘는 확률을 P(B)라고 했을때, 남학생 중에서, 키가 170이 넘는 확률은 B의 조건부 확률이 되며 P(B|A)로 표현 한다.

앞의 남학생인 확률 P(A)와 키가 170이상인 확률 P(B)를 알고, 남학생중에서 키가 170인 확률 P(B|A)를 알면, 키가 170인 학생중에, 남학생인 확률 P(A|B)를 알 수 있다

Naive Bayesian Classification

매개 변수, x,y가 있을때, 분류 1에 속할 확률이 p1(x,y)이고, 분류 2에 속할 확률이 p2(x,y)일때,

p1(x,y) > p2(x,y) 이면, 이 값은 분류 1에 속함
p1(x,y) < p2(x,y) 이면, 이 값은 분류 2에 속함

즉, 분류하는 대상의 **각 분류별 확률을 측정하여, 그 확률이 큰 쪽으로 분류.**

예를 들어, 이메일에 대해서 분류가 스팸과 스팸이 아닌 분류가 있을때, 이메일에 들어가 있는 단어들 w1,…,wn 매개 변수 (“쇼핑”,” 비아그라”,”보험”,….)에 대해서, 해당 이메일이 스팸일 확률과 스팸이 아닌 확률을 측정하여, 확률이 높은 쪽으로 판단

Naive Bayesian Classification 예

영화	단어	분류
1	fun,couple,love,love	Comedy
2	fast,furious,shoot	Action
3	Couple,fly,fast,fun,fun	Comedy
4	Furious,shoot,shoot,fun	Action
5	Fly,fast,shoot,love	Action

어떤 문서에 “fun,furious,fast” 라는 3개의 단어만 있을 때, 이 문서는 Comedy인가? Action인가 ?

영화가 Comedy일 확률: P(Comedy | Words) = P(Words | Comedy) X P(Comedy) / P(Words) -> A
영화가 Action 확률: P(Action | Words) = P(Words | Action) X P(Action) / P(Words) -> B
A > B라면 Comedy로 분류하고, A < B라면 Action으로 분류
A, B 확률을 구할때 분모에 P(Words)가 들어가는데 **A, B의대소만 비교 하기 때문에 P(Words) 무시**
A = P(Words | Comedy) X P(Comedy)
B = P(Words | Action) X P(Action)

영화	단어	분류
1	fun,couple,love,love	Comedy
2	fast,furious,shoot	Action
3	Couple,fly,fast,fun,fun	Comedy
4	Furious,shoot,shoot,fun	Action
5	Fly,fast,shoot,love	Action

- 각 단어의 빈도 수
  Count (fast,comedy) = 1
  Count(furious,comedy) = 0
  Count(fun,comedy) = 3
  Count(fast,action)= 2
  Count(furious,action)=2
  Count(furious,action) = 1 (action 중 fast 라는 단어가 나오는 횟수)
- P(Words | Comedy)는 Comedy 영화 중, 지정한 단어가 나타나는 확률
  P(fast,furious,fun | Comedy)로 표현 가능
  P(fast | Comedy) * P(furious | Comedy) * P(fun | Comedy)로 표현 가능
  Comedy 영화에 나오는 총 단어의 개수 : 9
  P(fast | Comedy) * P(furious | Comedy) * P(fun | Comedy) = (1/9) * (0/9) * (3/9)
  전체 영화 5편중에서 2편이 Comedy이기때문에, P(Comedy) = ⅖
- A = P(Comedy Words) = ((1/9) * (0/9) * (3/9)) * 2/5 = 0
- B = P(Action Words) = ((2/11) * (2/11)*(1/11)) * 3/5 = 0.0018 (동일한 방법으로 계산)
- A < B 이기 때문에 해당 문서는 Action으로 분류

Laplace Smoothing

나이브베이시안은 Training Data에 없는 새로운 단어가 나왔을 때 확률이 0이 되는 문제가 있다
이를 방지하기 위해 도수에 +1을 해줌으로써 확률이 0이 되는 것을 방지

$$\hat{P}(x|c)=\frac{count(x,c)+1}{\Sigma_{x\in V}(count(x,c)+1)}=\frac{count(x,c)+1}{(\Sigma_{x\in V}count(x,c))+|V|}$$

|V|는 전체 단어의 수가 아니라 유일한 단어의 수-7. Laplace Smoothing을 적용하여 다시 계산한 각각의

$$P(comedy|d)=P(fast|comedy)\bullet P(furious|comedy)\bullet P(fun|comedy)\bullet P(comedy)$$

$$=\frac{1+1}{9+7}\bullet\frac{0+1}{9+7}\bullet\frac{3+1}{9+7}\bullet\frac{2}{5}$$

$$=\frac{2}{16}\bullet\frac{1}{16}\bullet\frac{4}{16}\bullet\frac{2}{5}$$

$$=0.00078$$

$$P(action|d)=P(fast|action)\bullet P(furious|action)\bullet P(fun|action)\bullet P(action)$$

$$=\frac{2+1}{11+7}\bullet\frac{2+1}{11+7}\bullet\frac{1+1}{11+7}\bullet\frac{3}{5}$$

$$=\frac{3}{18}\bullet\frac{3}{18}\bullet\frac{2}{18}\bullet\frac{3}{5}$$

$$=0.0018$$

확률 0이 되는 문제는 해결되었으며 action 부류의 확률이 높기 때문에 해당 문서는 여전히 action으로 분류

Log를 이용한 언더 플로우 방지

P(words|comedy)나 P(words|action)은 각 단어의 확률의 곱인데, 항목이 많은 경우 소숫점 아래로 계속 내려가서, 구분이 어려울 정도까지 값이 작게 나올 수 있다. 이를 해결 하기 위해서 로그 (log)를 사용.

log(a*b) = log (a) + log(b)와 같기 때문에,
P(comedy|words) = P(words|comedy)*P(comedy) 양쪽 공식에 모두 log를
log(P(comedy|words)) = log(P(words|comedy)*P(comedy))

log(P(words|comedy)*P(comedy))
= log(P(fun|comedy)*P(furios|comedy)*…*P(Comedy))
= log(P(fun|comedy))+log(P(furios|comedy)+…+log(P(Comedy))

http://bcho.tistory.com/1010

Spam filtering

메일 제목으로 [광고, 중요] 카테고리 분류

오른쪽 프로그램 수행 결과

결과 = 광고
[('광고', -19.942524744665512), ('중요', -20.544606748320554)]

```
from bayes import BayesianFilter
bf = BayesianFilter()
# 텍스트 학습
bf.fit("파격 세일 - 오늘까지만 30% 할인", "광고")
bf.fit("쿠폰 선물 & 무료 배송", "광고")
bf.fit("현데계 백화점 세일", "광고")
bf.fit("봄과 함께 찾아온 따뜻한 신제품 소식", "광고")
bf.fit("인기 제품 기간 한정 세일", "광고")
bf.fit("오늘 일정 확인", "중요")
bf.fit("프로젝트 진행 상황 보고","중요")
bf.fit("계약 잘 부탁드립니다","중요")
bf.fit("회의 일정이 등록되었습니다.","중요")
bf.fit("오늘 일정이 없습니다.","중요")


# 예측
pre, scorelist = bf.predict("재고 정리 할인, 무료 배송")
print("결과 =", pre)
print(scorelist)
```

Spam filtering: 전처리

1. text를 단어로 분리하여 리스트에 저장: split()
2. 단어와 카테고리 빈도 저장: inc_category(), inc_word()
3. 학습 데이터 추가: fit()

word_dict

{'광고': {'파격': 1, '세일': 3, '-': 1, '오늘까지만': 1, '30%': 1, '할인': 1, '쿠폰': 1, '선물': 1, '&': 1, '무료': 1, '배송': 1, '현데계': 1, '백화점': 1, '봄과': 1, '함께': 1, '찾아온': 1, '따뜻한': 1, '신제품': 1, '소식': 1, '인기': 1, '제품': 1, '기간': 1, '한정': 1}, '중요': {'오늘': 2, '일정': 1, '확인': 1, '프로젝트': 1, '진행': 1, '상황': 1, '보고': 1, '계약': 1, '잘': 1, '부탁드립니다': 1, '회의': 1, '일정이': 2, '등록되었습니다.': 1, '없습니다.': 1}}

category_dict

{'광고': 5, '중요': 5}

```
import math, sys
class BayesianFilter:
    def __init__(self):
        self.words = set() # 출현한 단어 기록
        self.word_dict = {} # 카테고리마다의 출현 횟수 기록
        self.category_dict = {} # 카테고리 출현 횟수 기록
    # 단어 tokenization --- (※1)
    def split(self, text):
        return text.split()
    # 단어와 카테고리의 출현 횟수 세기 --- (※2)
    def inc_word(self, word, category):
        # 단어를 카테고리에 추가하기
        if not category in self.word_dict:
            self.word_dict[category] = {}
        if not word in self.word_dict[category]:
            self.word_dict[category][word] = 0
        self.word_dict[category][word] += 1
        self.words.add(word)
    def inc_category(self, category):
        # 카테고리별 (광고/중요) 단어 갯수 추가
        if not category in self.category_dict:
            self.category_dict[category] = 0
        self.category_dict[category] += 1
    def fit(self, text, category):
        word_list = self.split(text)
        for word in word_list:
            self.inc_word(word, category)
        self.inc_category(category)
```

Spam filtering

4. 단어 리스트 words가 카테고리에 속할 확률 계산. 확률값이 너무 작은 경우 downflow가 발생할 수 있어서 log 사용
6. 카테고리에 단어 word가 출현하는 확률 계산. 사전에 없는 단어가 나오면 확률이 0이 되므로 1을 더함

```
  # 카테고리 내부의 단어 출현 횟수 구하기
   def get_word_count(self, word, category):
      if word in self.word_dict[category]:
         return self.word_dict[category][word]
      else:
         return 0
   # 카테고리의 확률 계산
   def category_prob(self, category):
      sum_categories = sum(self.category_dict.values())
      category_v = self.category_dict[category]
      return category_v / sum_categories
   # 카테고리에 단어 word가 출현하는 확률 계산 --- (※6)
   def word_prob(self, word, category):
      n = self.get_word_count(word, category) + 1 # ---(※6a)
      d = sum(self.word_dict[category].values()) +
len(self.words)
      return n / d

   # words 단어가 카테고리에 속할 확률 계산--- (※4)
   def score(self, words, category):
      score = math.log(self.category_prob(category))
      for word in words:
         score += math.log(self.word_prob(word, category))
      return score
```

실습 1: text가 속할 카테고리 예측하는 predict()함수 작성

5. 주어진 텍스트의 카테고리 점수를 계산하고 가장 높은 카테고리 리턴
   - 주어진 text를 단어로 split하여 words라는 리스트로 만든다
   - words가 각 카테고리 (중요, 광고)에 속할 확률을 score() 함수를 이용하여 계산하고 score_list 리스트에 저장
   - 확률이 높은 카테고리를 best_category 변수에 저장

```
# 예측
pre, scorelist = bf.predict("재고 정리 할인, 무료 배송")
print("결과 =", pre)
print(scorelist)
```

결과 = 광고
[('광고', -19.942524744665512), ('중요', -20.544606748320554)]

```
    # 예측 --- (※5)
def predict(self, text):



    return best_category, score_list
```

실습 2: 모든 문장을 한꺼번에 받아 처리하도록 코드 변경

모든 문장을 받아 한번에 학습하는 sfit() 함수를 작성하세요. __init__ 에 정의된 인스턴스 변수를 수정하거나 추가/삭제 가능.
nltk.ConditionalFreqDist()을 사용 가능.

결과 = 광고
[('광고', -19.744073805941674), ('중요', -20.792442912225134)]
{'광고': 25, '중요': 16}

```
train = (("파격 세일 - 오늘까지만 30% 할인", "광고"),
("쿠폰 선물 & 무료 배송", "광고"),
("현데계 백화점 세일", "광고"),
("봄과 함께 찾아온 따뜻한 신제품 소식", "광고"),
("인기 제품 기간 한정 세일", "광고"),
("오늘 일정 확인", "중요"),
("프로젝트 진행 상황 보고","중요"),
("계약 잘 부탁드립니다","중요"),
("회의 일정이 등록되었습니다.","중요"),
("오늘 일정이 없습니다.","중요"))
bf = BayesianFilter()
bf.sfit( train) # sentence fit
# 예측
pre, scorelist = bf.predict("재고 정리 할인, 무료 배송")
print("결과 =", pre)
print(scorelist)
print(bf.categories)
```

실습 3: brown corpus의 카테고리에 해당하는 문장 판별

```
>>> from nltk.corpus import brown
>>> print(brown.categories())
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']

>>> genre_sents = [(s, genre)
            for genre in ['news', 'romance']
            for s in brown.sents(categories=genre)]
print(genre_sents)
[(['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place', '.'], 'news'), (['``', 'I', 'suppose', 'I', 'can', 'never', 'expect', 'to', 'call', 'you', "''", 'General', "''", 'after', 'that', 'Washington', 'episode', "''", '.'], 'romance'), (['``', "I'm", 'afraid', 'not', "''", '.'], 'romance')]
```

실습 4: brown corpus의 다른 장르에 해당하는 문장 판별

Brown corpus의 sents() 함수는 문장을 단어 리스트로 저장함. 단어 리스트 형태의 문장을 처리하는 lfit() 함수를 작성하시오.

```
bf = BayesianFilter1()
bf.lfit( genre_sents) # sentence fit
# 예측
pre, scorelist = bf.predict("Good luck to you all the best")
print("결과 =", pre)
print(scorelist)
print(bf.categories)

[실행 결과]
결과 = romance
[('news', -51.24489975442419), ('romance', -47.46630612000449)]
{'news': 100554, 'romance': 70022}
```