데이터 과학

L13.1: Naïve Bayes Practice

Kookmin University

SMS Spam Collection Dataset

- SMS 스팸 데이터
  - 문자메시지 데이터
  - spam과 ham으로 분류
  - http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/

```
ham     Go until jurong point, crazy.. Available c
ham     Ok lar... Joking wif u oni...
spam    Free entry in 2 a wkly comp to win FA Cup
ham     U dun say so early hor... U c already ther
ham     Nah I don't think he goes to usf, he lives
spam    FreeMsg Hey there darling it's been 3 week
ham     Even my brother is not like to speak with
ham     As per your request 'Melle Melle (Oru Minr
spam    WINNER!! As a valued network customer you
spam    Had your mobile 11 months or more? U R ent
```

모듈 불러오기

- 사용할 모듈 import 하기

```
import zipfile
import requests
from collections import Counter
import re
import math
```

데이터 나누기

- train, test 데이터 분리

```
train_size = int(0.8 * len(data))
test_size = len(data) - train_size

train = data[:train_size]
test = data[train_size:]
```

사전확률 구하기

- 스팸 메시지와 일반 메시지의 개수를 세어 사전확률 구하기

```
count = Counter([s for s, c in train])

prior_spam = count['spam'] / (count['spam'] + count['ham'])
prior_ham = count['ham'] / (count['spam'] + count['ham'])
```

Word Count

- 스팸 메시지들에서 각 단어가 등장한 횟수 세기
- 일반 메시지들에서 각 단어가 등장한 횟수 세기

```
spam_words = Counter(w.group() \
                     for s, c in train if s == 'spam' \
                     for w in re.finditer("[0-9a-zA-Z_]+", c.lower()))

ham_words = Counter(w.group() \
                     for s, c in train if s == 'ham' \
                     for w in re.finditer("[0-9a-zA-Z_]+", c.lower()))
```

Word Count

- 단어 집합 크기 구하기
- 스팸메일의 전체 단어 수 세기
- 일반메일의 전체 단어 수 세기

```
num_unique_words = len(set(w for s, c in train \
                           for w in re.finditer("[0-9a-z_]+", c.lower())))


num_spam_words = sum(spam_words.values())


num_ham_words = sum(ham_words.values())
```

P(S=T) = 0.4

단어	빈도	가능도
롤	■■	P(롤|S=T) = 2/21
시공	■■■■■■	P(시공|S=T) = 6/21
조아	■■■■	P(조아|S=T) = 4/21
ㄱㄱ	■	P(옵치|S=T) = 1/21
폭풍	■■■■■	P(폭풍|S=T) = 5/21
접속	■■■	P(접속|S=T) = 3/21

predict()

- naive bayes 방법에 따라 spam/ham 점수를 구하여 비교
- 라플라스 스무딩 적용
- 로그 합산

```
def predict(text, k=0.5):
    words = re.findall("[0-9a-z_]+", text.lower())

    denom_spam = num_spam_words + num_unique_words * k
    denom_ham = num_ham_words + num_unique_words * k

    spam_score = math.log(prior_spam)
    ham_score = math.log(prior_ham)

    for w in words:
        spam_score += math.log((spam_words[w] + k)/denom_spam)
        ham_score += math.log((ham_words[w] + k)/denom_ham)

    return spam_score > ham_score
```

P(S=T) = 0.4

단어	빈도	가능도
롤	■■	P(롤|S=T) = 2/21
시공	■■■■■■	P(시공|S=T) = 6/21
조아	■■■■	P(조아|S=T) = 4/21
ㄱㄱ	■	P(옵치|S=T) = 1/21
폭풍	■■■■■	P(폭풍|S=T) = 5/21
접속	■■■	P(접속|S=T) = 3/21

테스트 해보기

- 잘 맞히는지 확인해보기

```
true_positive = 0
true_negative = 0
false_positive = 0
false_negative = 0

for s, c in test:
    pred = predict(c)
    if pred and s == 'spam':
        true_positive += 1
    elif pred and s != 'spam':
        false_positive += 1
    elif not pred and s == 'spam':
        false_negative += 1
    else:
        true_negative += 1
```

		실제 정답	
		True	False
분류 결과	True	True Positive	False Positive
	False	False Negative	True Negative

테스트 해보기

- Precision, Recall, F1 Score, Accuracy

```
print("accuracy: ", \
(true_positive + true_negative) / (true_positive + false_positive + true_negative + false_negative))
print("precision: ", true_positive/(true_positive + false_positive))
print("recall: ", true_positive / (true_positive + false_negative))
print("f1: ", true_positive / (true_positive + (false_positive + false_negative)/2) )
```

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + F_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

accuracy: 0.9847533632286996
precision: 0.9923076923076923
recall: 0.8896551724137931
f1: 0.9381818181818182

Questions?