Naive Bayes Classification

d개의 독립변수가 주어져 있을 때 종속변수 y가 class k(k = 1,.., K)에 속할 확률은 Bayes rule에 의해,

$$P(y = k|x) = \frac{P(x|k)P(k)}{P(x)} = \frac{P(x|k)P(k)}{\sum_{k=1}^{K} P(x|k)P(k)}$$

$$x_j \sim N_d(\mu_k, \sigma^2) \qquad P(x|k) = \prod_{j=1}^{d} P(x_j|k),$$

Bayes' theorem을 이용한 분류 알고리즘

P(정상 메일 | 입력 테스트) = 입력 텍스트가 있을 때 정상 메일일 확률
P(스팸 메일 | 입력 테스트) = 입력 텍스트가 있을 때 스팸 메일일 확률

P(정상 메일 | 입력 테스트) = (P(입력 테스트 | 정상 메일) × P(정상 메일)) / P(입력 텍스트)
P(스팸 메일 | 입력 테스트) = (P(입력 테스트 | 스팸 메일) × P(스팸 메일)) / P(입력 텍스트)

Naive Bayes Classification

입력 테스트는 메일의 본문을 의미, 메일의 본문에 있는 단어가 3개임을 가정

P(정상 메일 | 입력 텍스트) = P(w1 | 정상 메일) × P(w2 | 정상 메일) × P(w3 | 정상 메일) × P(정상 메일)

P(스팸 메일 | 입력 텍스트) = P(w1 | 스팸 메일) × P(w2 | 스팸 메일) × P(w3 | 스팸 메일) × P(스팸 메일)

순서를 무시, 마치 Bag-of-words

Naive Bayes Classification - Spam Detection

-	메일로부터 토큰화 및 정제 된 단어들	분류
1	me free lottery	스팸 메일
2	free get free you	스팸 메일
3	you free scholarship	정상 메일
4	free to contact me	정상 메일
5	you won award	정상 메일
6	you ticket lottery	스팸 메일

P(정상 메일)
= P(스팸 메일)
= 총 메일 6개 중 3개
= 0.5

Naive Bayes Classification - Spam Detection

입력 텍스트 : you free lottery

P(정상 메일 | 입력 텍스트) = P(you | 정상 메일) × P(free | 정상 메일) × P(lottery | 정상 메일) × P(정상 메일)
P(스팸 메일 | 입력 텍스트) = P(you | 스팸 메일) × P(free | 스팸 메일) × P(lottery | 스팸 메일) × P(스팸 메일)

P(정상 메일 | 입력 텍스트) = 2/10 × 2/10 × 0/10 = 0
P(스팸 메일 | 입력 텍스트) = 2/10 × 3/10 × 2/10 = 0.012

Naive Bayes Classification - Spam Detection

Laplace Smoothing

단어 수가 많을수록, 문서에 단 한번도 등장하지 않은 단어가 있다면
-> 해당 단어의 확률값(우도)은 0이됨
-> 분모, 분자에 전부 수를 더하여 분자가 0이되는 것을 방지

P(정상 메일 | 입력 텍스트) = 2/10 × 2/10 × 0/10 = 0
P(스팸 메일 | 입력 텍스트) = 2/10 × 3/10 × 2/10 = 0.012

뉴스 그룹 데이터 분류하기 – 데이터 이해

총 20개의 category

```
#훈련용 샘플의 갯수
print (len(newsdata.data), len(newsdata.filenames), len(newsdata.target_names), len(newsdata.target))
```

```
11314 11314 20 11314
```

```
#20개의 카테고리 이름
print(newsdata.target_names)
```

```
['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space', 'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast', 'talk.politics.misc', 'talk.religion.misc']
```

뉴스 그룹 데이터 분류하기 - 토큰화

11,314 train dataset / 130,107 train dataset의 단어의 수

```
dtmvector = CountVectorizer()
tfidf_transformer = TfidfTransformer()
X_train_dtm = dtmvector.fit_transform(newsdata.data)
tfidfv = tfidf_transformer.fit_transform(X_train_dtm)
print(tfidfv.shape)
```

```
(11314, 130107)
```

뉴스 그룹 데이터 분류하기 – 적합, 예측

```
mod = MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
mod.fit(tfidfv, newsdata.target)
y_train_pred = mod.predict(tfidfv)
```

```
MultinomialNB()
```

```
newsdata_test = fetch_20newsgroups(subset='test', shuffle=True) #테스트 데이터 갖고오기
X_test_dtm = dtmvector.transform(newsdata_test.data) #테스트 데이터를 DTM으로 변환
tfidfv_test = tfidf_transformer.transform(X_test_dtm) #DTM을 TF-IDF 행렬로 변환

predicted = mod.predict(tfidfv_test) #테스트 데이터에 대한 예측
print("정확도:", accuracy_score(newsdata.target, y_train_pred))
print("정확도:", accuracy_score(newsdata_test.target, predicted)) #예측값과 실제값 비교
```

```
정확도: 0.9628498143892522
정확도: 0.7736960650504514
```

초모수 alpha

x_train ; tfidfv
y_train ;
newsdata.target

Test data에 대한
77%의 정확도

참고문헌

박유성 교수님 2020 Fall 통계적 머신러닝 강의안
딥러닝을 이용한 자연어 처리 입문(wikidocs)