MYONGJI UNIVERSITY

Artificial Intelligence

Lecture 6. Text Mining Part 2

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail : jchun@mju.ac.kr
Lecture Note : https://lms.mju.ac.kr

Agenda

- Text Mining 개요
- 영문 텍스트 분석
- 한글 텍스트 처리 기법
- WordCloud

Probability Recap

- Conditional probabilities

	TV	Books	
female	1	2	1+2=3
male	4	3	4+3=7
	1+4=5	2+3=5	3+7=10 or 5+5=10

- The probability a randomly sampled person in this group will be female is *P(female) = 3/10 = .3*
- Joint probability

  P(female, books) = 2/10 = .2

  $P(x, y) = P(x|y)\,P(y)$

  *P(female, books) = P(female|books) P(books) = 2/5 * 5/10 = .2*

  $P(x|y) \geq P(x, y)$

  $P(x, y) = P(x)\,P(y)$ if *x* and *y* are statistically independent

Bayesian Classification

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

where

$p(c_j \mid d)$ = probability of instance d being in class c_j,

$p(d \mid c_j)$ = probability of generating instance d given class c_j,

$p(c_j)$ = probability of occurrence of class c_j, and

$p(d)$ = probability of instance d occurring

Naïve Bayesian Classifiers

- Bayesian classifiers require
  - computation of $p(d \mid c_j)$
  - precomputation of $p(c_j)$
  - $p(d)$ can be ignored since it is the same for all classes

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d \mid c_j) = p(d_1 \mid c_j) * p(d_2 \mid c_j) * \ldots * p(d_n \mid c_j)$$

  - Each of the $p(d_i \mid c_j)$ can be estimated from the training instances

Naïve Bayes with Numpy

다음과 같은 training data가 주어졌을때,

X_1	X_2	Class
-3	7	3
1	5	3
1	2	3
-2	0	3
2	3	4
-4	0	3
-1	1	3
1	1	4
-2	2	3
2	7	4
-4	1	4
-2	7	4

Unseen data [X_1 = 1, X_2 = 0], [X_1 = -4, X_2 = 0],[X_1 = -8, X_2 = -16]은 각각 어떤 class (class 3 혹은 class 4)에 속하지는 Naïve Bayesian을 사용하여 예측하시오.

$$P(X_1, X_2 \mid Class = 3) = P(X_1 \mid Class = 3) \times P(X_2 \mid Class = 3)$$

$$P(X_1, X_2 \mid Class = 4) = P(X_1 \mid Class = 4) \times P(X_2 \mid Class = 4)$$

각각을 Gaussian distribution을 사용하여 estimation 한 후에, P(Class = 3) = 7/12 P(Class = 4) = 5/12를 곱하여 클래스를 예측!

Naïve Bayes with Numpy

```
from sklearn.naive_bayes import GaussianNB
import numpy as np

# assigning training data
x = np.array([[-3,7],[1,5],[1,2],[-2,0],[2,3],[-4,0],[-1,1],[1,1],[-2,2],[2,7],[-4,1],[-2,7]])
# assinging class label as class 3 & 4
y = np.array([3, 3, 3, 3, 4, 3, 3, 4, 3, 4, 4, 4])

# Create a Naive Bayesian Classifier (using Gaussian distribution)
model = GaussianNB()

# Train the model
model.fit(x, y)

# predict the output
predicted = model.predict([[1,0],[-4,0],[-8, -16]])
print(predicted)
```

```
[3 3 4]
```

Naïve Bayes for Text Classification

- 다음과 같은 training text data가 주어졌을때,

Doc_id	apple	brain	cat	glass	Class
1	2	0	0	1	Neg
2	0	3	0	0	Pos
3	0	0	1	1	Neg
4	1	0	0	0	Neg
5	0	2	0	0	Pos

- 다음 document X는 어느 클래스에 속하는지 예측?

Doc_id	apple	brain	cat	glass	Class
X	1	0	1	0	?

Naïve Bayes for Text Classification

- $P(a = 1, b = 0, c = 1, g = 0 \mid Neg) = P(a = 1 \mid Neg) \times P(b = 0 \mid Neg) \times P(c = 1 \mid Neg) \times P(g = 0 \mid Neg)$

- $P(a = 1, b = 0, c = 1, g = 0 \mid Pos) = P(a = 1 \mid Pos) \times P(b = 0 \mid Pos) \times P(c = 1 \mid Pos) \times P(g = 0 \mid Pos)$

- 각각을 text classification에 적합한 Mutlinomial distribution을 사용하여 estimation 한 후에, P(Neg) = 3/5 P(Pos) = 2/5를 곱하여 클래스를 예측!

Accuracy 측정

- sklearn.metrics accuracy_score 사용하여 accuracy 측정이 가능

```
In [3]: import numpy as np
        from sklearn.metrics import accuracy_score
        y_pred = [0, 2, 1, 3]
        y_true = [0, 1, 2, 3]
        accuracy_score(y_true, y_pred)

Out[3]: 0.5
```

Movie Review 자동 분류

- Naïve Bayesian with Multinomial distribution estimation

```
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import load_files
import numpy as np

reviews_train = load_files("/Users/jonghoonchun/Downloads/aclImdb/train")
reviews_test = load_files("/Users/jonghoonchun/Downloads/aclImdb/test")

text_train, y_train = reviews_train.data, reviews_train.target
text_test, y_test = reviews_test.data, reviews_test.target

text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]

# 영화 리뷰 학습 데이터와 테스트 데이터에 대한  BOW 생성
# 주의: 테스트 데이터에 대한 BOW 생성 시, 학습 데이터에서 생성한 어휘사전을 사용해야 함
vect = CountVectorizer().fit(text_train)
X_train = vect.transform(text_train)
X_test = vect.transform(text_test)

# GaussianNB는 sparse matrix를 입력으로 받지 못하므로 주의할 것!
nb = MultinomialNB()
nb.fit(X_train, y_train)
pre = nb.predict(X_test)

ac_score = metrics.accuracy_score(y_test, pre)
print("정답률 =", ac_score)
```

```
정답률 = 0.81432
```

Movie Review 자동 분류

- Multinomial distribution smoothing (α값 tuning)

```
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import load_files
import numpy as np

reviews_train = load_files("/Users/jonghoonchun/Downloads/aclImdb/train")
reviews_test = load_files("/Users/jonghoonchun/Downloads/aclImdb/test")

text_train, y_train = reviews_train.data, reviews_train.target
text_test, y_test = reviews_test.data, reviews_test.target

text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]

# 영화 리뷰 학습 데이터와 테스트 데이터에 대한  BOW 생성
# 주의: 테스트 데이터에 대한 BOW 생성 시, 학습 데이터에서 생성한 어휘사전을 사용해야 함
vect = CountVectorizer().fit(text_train)
X_train = vect.transform(text_train)
X_test = vect.transform(text_test)

# GaussianNB는 sparse matrix를 입력으로 받지 못하므로 주의할 것!
# alpha값을 0.2로 설정
nb = MultinomialNB(alpha = 0.2)
nb.fit(X_train, y_train)
pre = nb.predict(X_test)

ac_score = metrics.accuracy_score(y_test, pre)
print("정답률 =", ac_score)
```

```
정답률 = 0.80676
```

Default alpha값은 1!

Movie Review 자동 분류

- Multinomial distribution smoothing (α값 tuning)

```
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import load_files
import numpy as np

reviews_train = load_files("/Users/jonghoonchun/Downloads/aclImdb/train")
reviews_test = load_files("/Users/jonghoonchun/Downloads/aclImdb/test")

text_train, y_train = reviews_train.data, reviews_train.target
text_test, y_test = reviews_test.data, reviews_test.target

text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]

# 영화 리뷰 학습 데이터와 테스트 데이터에 대한  BOW 생성
# 주의: 테스트 데이터에 대한 BOW 생성 시, 학습 데이터에서 생성한 어휘사전을 사용해야 함
vect = CountVectorizer().fit(text_train)
X_train = vect.transform(text_train)
X_test = vect.transform(text_test)

# GaussianNB는 sparse matrix를 입력으로 받지 못하므로 주의할 것!
# alpha값을 0.7로 설정
nb = MultinomialNB(alpha = 0.7)
nb.fit(X_train, y_train)
pre = nb.predict(X_test)

ac_score = metrics.accuracy_score(y_test, pre)
print("정답률 =", ac_score)
```

```
정답률 = 0.81188
```

Movie Review 자동 분류

- Multinomial distribution smoothing (α값 tuning)

```
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.datasets import load_files
import numpy as np

reviews_train = load_files("/Users/jonghoonchun/Downloads/aclImdb/train")
reviews_test = load_files("/Users/jonghoonchun/Downloads/aclImdb/test")

text_train, y_train = reviews_train.data, reviews_train.target
text_test, y_test = reviews_test.data, reviews_test.target

text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]

# 영화 리뷰 학습 데이터와 테스트 데이터에 대한  BOW 생성
# 주의: 테스트 데이터에 대한 BOW 생성 시, 학습 데이터에서 생성한 어휘사전을 사용해야 함
vect = CountVectorizer().fit(text_train)
X_train = vect.transform(text_train)
X_test = vect.transform(text_test)

# GaussianNB는 sparse matrix를 입력으로 받지 못하므로 주의할 것!
# alpha값을 1.8로 설정
nb = MultinomialNB(alpha = 1.8)
nb.fit(X_train, y_train)
pre = nb.predict(X_test)

ac_score = metrics.accuracy_score(y_test, pre)
print("정답률 =", ac_score)
```

```
정답률 = 0.8164
```

TF-IDF 사용 자동 분류

- TfidfVectorizor 사용

```
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.datasets import load_files
import numpy as np

reviews_train = load_files("/Users/jonghoonchun/Downloads/aclImdb/train")
reviews_test = load_files("/Users/jonghoonchun/Downloads/aclImdb/test")

text_train, y_train = reviews_train.data, reviews_train.target
text_test, y_test = reviews_test.data, reviews_test.target

text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]

# tf-idf vector 방식 사용
vect = TfidfVectorizer().fit(text_train)
X_train = vect.transform(text_train)
X_test = vect.transform(text_test)

nb = MultinomialNB()
nb.fit(X_train, y_train)
pre = nb.predict(X_test)

ac_score = metrics.accuracy_score(y_test, pre)
print("정답률 =", ac_score)
```

```
정답률 = 0.83024
```

TF-IDF 사용 자동 분류

- Multinomial distribution smoothing (α값 tuning)

```
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.datasets import load_files
import numpy as np

reviews_train = load_files("/Users/jonghoonchun/Downloads/aclImdb/train")
reviews_test = load_files("/Users/jonghoonchun/Downloads/aclImdb/test")

text_train, y_train = reviews_train.data, reviews_train.target
text_test, y_test = reviews_test.data, reviews_test.target

text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]

# tf-idf vector 방식 사용
vect = TfidfVectorizer().fit(text_train)
X_train = vect.transform(text_train)
X_test = vect.transform(text_test)

# alpha값을 1.8로 설정
nb = MultinomialNB(alpha = 1.8)
nb.fit(X_train, y_train)
pre = nb.predict(X_test)

ac_score = metrics.accuracy_score(y_test, pre)
print("정답률 =", ac_score)
```

```
정답률 = 0.83172
```

END