Machine Learning
Lecture #7
(Building a Naive Bayes Text Classifier with scikit-learn)

Presented by

Assoc. Prof. Ghazal A. Fahmy

ghazal.Fahmy@nti.sci.eg

Naïve Bayes: A Little History

Reverend Bayes

Pierre-Simon Laplace

Bayes Theorem

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

Naïve Bayes: Advantages and Disavantages

Advantages

- Very simple to implement and fast
- Works well even when the feature independence assumption does not hold as in the case of text
- Deals well with data sets that have very large feature spaces

Disadvantages

- Does not work well with expressions that have a combination of words with unique meanings.

Naïve Bayes: The Equation

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Naïve Bayes: An example

Spam

I love song

Ham

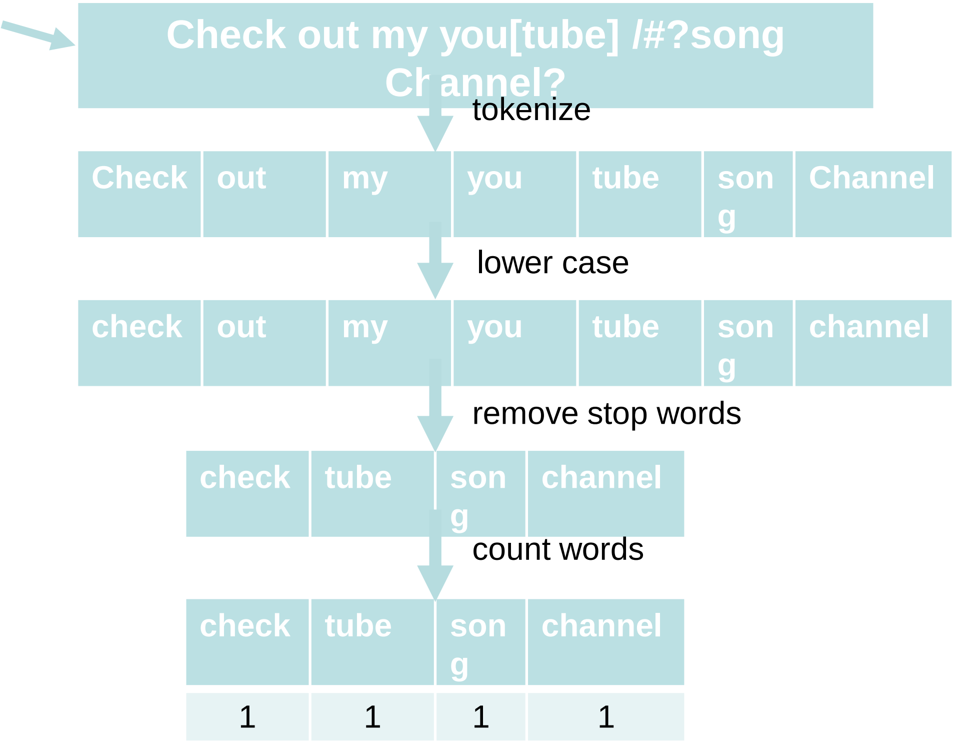
Naïve Bayes: An example

	comment	category
0	Check out my you[tube] /#?song Channel?	spam
1	I love song, please like my channel!!!	spam
2	I'm not a fan, :(	ham
3	This song is great, I love your channel	ham
4	love love love	ham

I love song ?

Naïve Bayes: An example

	comment	category
0	Check out my you[tube] /#?song Channel?	spam
1	I love song, please like my channel!!!	spam
2	I'm not a fan, :(	ham
3	This song is great, I love your channel	ham
4	love love love	ham



Naïve Bayes: The Equation

	channel	check	fan	great	like	love	song	tube	class
0	1	1	0	0	0	0	1	1	spam
1	1	0	0	0	1	1	1	0	spam
2	0	0	1	0	0	0	0	0	ham
3	1	0	0	1	0	1	1	0	ham
4	0	0	0	0	0	3	0	0	ham

I love song ?

Naïve Bayes: The Equation

	channel	check	fan	great	like	love	song	tube	class
0	1	1	0	0	0	0	1	1	spam
1	1	0	0	0	1	1	1	0	spam
2	0	0	1	0	0	0	0	0	ham
3	1	0	0	1	0	1	1	0	ham
4	0	0	0	0	0	3	0	0	ham

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

P(**Spam**|I love song) = P (**Spam**) x P(I love song|**Spam**) / P(I love song)

P(**Ham**|I love song) = P (**Ham**) x P(I love song|**Ham**) / P(I love song)

Naïve Bayes: The Equation

Calculating the apriori:

$$= \frac{\boxed{P(y)} P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

	channel	check	fan	great	like	love	song	tube	class
0	1	1	0	0	0	0	1	1	spam
1	1	0	0	0	1	1	1	0	spam
2	0	0	1	0	0	0	0	0	ham
3	1	0	0	1	0	1	1	0	ham
4	0	0	0	0	0	3	0	0	ham

P(**Spam**) = $\frac{\text{\# of Spam comments}}{\text{\# total comments}} = \frac{2}{5}$

P(**Ham**) = $\frac{\text{\# of Ham comments}}{\text{\# total comments}} = \frac{3}{5}$

Naïve Bayes: The Equation

Calculating conditional probability

	channel	check	fan	great	like	love	song	tube	class
0	1	1	0	0	0	0	1	1	spam
1	1	0	0	0	1	1	1	0	spam
2	0	0	1	0	0	0	0	0	ham
3	1	0	0	1	0	1	1	0	ham
4	0	0	0	0	0	3	0	0	ham

$$= \frac{P(y)\boxed{P(x_1, \ldots x_n \mid y)}}{P(x_1, \ldots, x_n)}$$

P(love|**Spam**) x P(song|**Spam**) $= \frac{1}{8} \times \frac{2}{8}$

$= \frac{\mathbf{1}}{\mathbf{32}}$

P(love|**Ham**) x P(song|**Ham**) $= \frac{4}{8} \times \frac{1}{8}$

$= \frac{\mathbf{1}}{\mathbf{16}}$

Naïve Bayes: The Equation

Calculating conditional probability

	channel	check	fan	great	like	love	song	tube	class
0	1	1	0	0	0	0	1	1	spam
1	1	0	0	0	1	1	1	0	spam
2	0	0	1	0	0	0	0	0	ham
3	1	0	0	1	0	1	1	0	ham
4	0	0	0	0	0	3	0	0	ham

$$\boxed{= \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}}$$

P(**Spam**|love song) $= \frac{1}{5} \times \frac{1}{32}$

$=$ **0.006**

I love song **ham**

P(**Ham**|love song) $= \frac{4}{5} \times \frac{1}{16}$

$=$ **0.05** ✓

Naïve Bayes: and now for the Code

Loading the Dataset

```
In [1]: # Import necessary modules
        import pandas as pd
        import numpy as np
        import glob
```

```
In [2]: # Import the data set
        path = r'C:\Users\obiamOneDrive\YouTube-Spam-Collection-v1'
        allFiles = glob.glob(path + "/*.csv")
        frame = pd.DataFrame()
        list_ = []
        for file_ in allFiles:
            df = pd.read_csv(file_,index_col=None, header=0)
            list_.append(df)
        frame = pd.concat(list_)
```

Loading the Dataset

```
In [3]: frame.head()
```

Out[3]:

	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
0	LZQPQhLyRh80UYxNuaDWhIGQYNQ96IuCg-AYWqNPjpU	Julius NM	2013-11-07T06:20:48	Huh, anyway check out this you[tube] channel: ...	1
1	LZQPQhLyRh_C2cTtd9MvFRJedxydaVW-2sNg5Diuo4A	adam riyati	2013-11-07T12:37:15	Hey guys check out my new channel and our firs...	1
2	LZQPQhLyRh9MSZYnf8djyk0gEF9BHDPYrrK-qCczIY8	Evgeny Murashkin	2013-11-08T17:34:21	just for test I have to say murdev.com	1
3	z13jhp0bxqncu512g22wvzkasxmvvzjaz04	ElNino Melendez	2013-11-09T08:28:43	me shaking my sexy ass on my channel enjoy ^_^	1
4	z13fwbwp1oujthgqj04chlngpvzmtt3r3dw	GsMega	2013-11-10T16:05:38	watch?v=vtaRGgvGtWQ Check this out .	1

```
In [4]: frame.shape
```

Out[4]: (1956, 5)

Sub-setting

In [5]:
```
df = frame.iloc[:,3:5]
df.head()
```

Out[5]:

	CONTENT	CLASS
0	Huh, anyway check out this you[tube] channel: ...	1
1	Hey guys check out my new channel and our firs...	1
2	just for test I have to say murdev.com	1
3	me shaking my sexy ass on my channel enjoy ^_^	1
4	watch?v=vtaRGgvGtWQ Check this out .	1

In [6]:
```
df.shape
```

Out[6]:
```
(1956, 2)
```

Train/test split

```
In [7]: # Import the necessary modules
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.model_selection import train_test_split
```

```
In [8]: # Create a series to store the comments:x
        x = df['CONTENT']
        # Create a series to store the labels: y
        y = df['CLASS']
        # Create training and test sets
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state=53)
```

```
In [9]: print(x_train.shape,
        x_test.shape,
        y_train.shape,
        y_test.shape)

        (1369,) (587,) (1369,) (587,)
```

Feature extraction: Bag of words approach

```
In [10]: # Initialize a CountVectorizer object: count_vectorizer
         count_vectorizer = CountVectorizer(stop_words="english")

         # Transform the training data using only the 'text' column values: count_train
         # Creates a Document Term Matrix
         count_train = count_vectorizer.fit_transform(x_train.values)
         # Transform the test data: tf_test
         count_test = count_vectorizer.transform(x_test.values)
```

Bag of words approach-Training

In [11]:

```python
# Import the necessary modules
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

# Instantiate a Multinomial Naive Bayes classifier: nb_classifier
nb_classifier = MultinomialNB()
```

```python
# Fit the classifier to the training data
nb_classifier.fit(count_train, y_train)
```

Bag of Words approach-Testing and Evaluatio

In [12]:

```
# Create the predicted tags: pred
pred = nb_classifier.predict(count_test)

# Calculate the accuracy score: score
score = metrics.accuracy_score(y_test, pred)
print(score)
```

```
0.9114139693356048
```

In [13]:

```
# Calculate the confusion matrix: cm
cm = metrics.confusion_matrix(y_test, pred,labels=[1, 0])
print(cm)
```

```
[[289   9]
 [ 43 246]]
```

Feature Extraction: TF-IDF Approach

```
In [14]: # Import TfidfVectorizer
         from sklearn.feature_extraction.text import TfidfVectorizer

         # Initialize a TfidfVectorizer object: tfidf_vectorizer
         tfidf_vectorizer = TfidfVectorizer(stop_words="english", max_df=0.7)

         # Transform the training data: tfidf_train
         tfidf_train = tfidf_vectorizer.fit_transform(x_train.values)

         # Transform the test data: tfidf_test
         tfidf_test = tfidf_vectorizer.transform(x_test.values)
```

TF-IDF Approach: Training

```
In [11]: # Import the necessary modules
         from sklearn import metrics
         from sklearn.naive_bayes import MultinomialNB

         # Instantiate a Multinomial Naive Bayes classifier: nb_classifier
         nb_classifier = MultinomialNB()

         # Fit the classifier to the training data
         nb_classifier.fit(count_train, y_train)
```

TF-IDF Approach: Testing and Evaluation

In [16]:
```
# Create the predicted tags: pred
pred = nb_classifier.predict(tfidf_test)

# Calculate the accuracy score: score
score = metrics.accuracy_score(y_test, pred)
print(score)
```

```
0.914821124361585
```

In [17]:
```
# Calculate the confusion matrix: cm
cm = metrics.confusion_matrix(y_test, pred, labels=[1, 0])
print(cm)
```

```
[[289   9]
 [ 41 248]]
```

Question ?