NLP and Machine Learning

We saw that we convert text document into a 'vector model' (bag-of-words)

The vector model allows us to perform mathematical analysis on documents - "which documents are similar to each other?"

Next question: can we construct machine learning models on document collections using the vector model?

Yes! We can construct classifiers.

NLP & ML

Consider again our news article data set.

We would like to construct a classifier that can correctly classifier political and science documents.

We will begin with our KNN algorithm (k nearest neighbors).

KNN - set up

```python
import pandas
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import cross_val_score
from re import sub
from nltk.stem import PorterStemmer


print("******** setup **********")
stemmer = PorterStemmer()
cats = ['talk.politics.misc', 'sci.space']
newsgroups_train = fetch_20newsgroups(subset='train',
                        remove=('headers', 'footers', 'quotes'),
                        categories=cats)
```

KNN - data prep

```
print("********* prepare data **********")
new_data = []
for i in range(len(newsgroups_train.data)):
    new_data.append(sub("[^a-zA-Z]", " ", newsgroups_train.data[i]))

lowercase_data = []
for i in range(len(new_data)):
    lowercase_data.append(new_data[i].lower())

stemmed_data = []
for i in range(len(lowercase_data)):
    words = lowercase_data[i].split()
    stemmed_words = []
    for w in words:
        stemmed_words.append(stemmer.stem(w))
    stemmed_data.append(" ".join(stemmed_words))
```

KNN - Vector Model & KNN Model

```python
print("******** setup vector model **********")
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer = "word", binary = True, min_df=2, stop_words='english')
docarray = vectorizer.fit_transform(stemmed_data).toarray()

print("******** model and XV **********")
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=4)

# do the 10-fold cross validation
scores = cross_val_score(model, docarray, newsgroups_train.target, cv=10)
print("Fold Accuracies: {}".format(scores))
print("XV Accuracy: {: 6.2f}".format(scores.mean()*100))
```

KNN - Vector Model & KNN Model

Here we introduced something new called the 'stop list'

A stop list defines a set of words that should be excluded from the vector model.

'Stop words' are words that usually do not add anything to the meaning or value of a document -- usually they just appear too frequently to be of any analytic use

Here is the beginning of a popular stop list defined for text mining tools in the R programming environment.

https://github.com/arc12/Text-Mining-Weak-Signals/wiki/Standard-set-of-english-stopwords

```
stoplist = [
"a",
"about",
"above",
"across",
"after",
"again",
"against",
"all",
"almost",
"alone",
"along",
"already",
"also",
"although",
...
]
```

KNN - Vector Model & KNN Model

```
******** setup **********
******** prepare data **********
******** setup vector model **********
******** model and XV **********
Fold Accuracies: [ 0.68224299  0.6635514   0.61682243  0.61320755  0.62264151  0.63809524
  0.59047619  0.66666667  0.62857143  0.6        ]
XV Accuracy:  63.22%
```

That's not very impressive - only a little bit better than random guessing.

Let's try out another model: Naive Bayes

Naive Bayes Model

"Standard" model for text processing

Fast to train, has no problems with very high dimensional data

Naive Bayes Model

NB is a classification technique based on Bayes' Theorem with an assumption of independence among predictors.

In simple terms, a NB classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Source: https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/

Naive Bayes Model
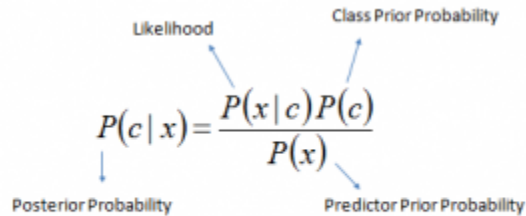
Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below, where

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
P(c) is the prior probability of class.
P(x|c) is the likelihood which is the probability of predictor given class.
P(x) is the prior probability of predictor.

Likelihood   Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c) P(c)}{P(x)}$$

Posterior Probability   Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Naive Bayes Model

Let's understand it using an example. Below isa training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Naive Bayes Model

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability than not playing.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Naive Bayes Model

Let's take our text classification and use a Naive Bayes classifier on it.

The setup and data prep is the same as in the case of the KNN classifier

Naive Bayes - Vector Model & NB Model

```python
print("******** setup vector model **********")
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer = "word", binary = True, min_df=2, stop_words='english')
docarray = vectorizer.fit_transform(stemmed_data).toarray()

print("******** model and XV **********")
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()

# do the 10-fold cross validation
scores = cross_val_score(model, docarray, newsgroups_train.target, cv=10)
print("Fold Accuracies: {}".format(scores))
print("XV Accuracy: {: 6.2f}".format(scores.mean()*100))
```

NB - Vector Model & NB Model

```
******** setup **********
******** prepare data **********
******** setup vector model **********
******** model and XV **********
Fold Accuracies: [ 0.94392523  0.93457944  0.95327103  0.96226415  0.93396226  0.94285714
  0.93333333  0.80952381  0.91428571  0.8952381 ]
XV Accuracy:  92.23%
```

Exercise

For this exercise you will build a classifier that can distinguish real news from fake news.

A training set for this is available here:

https://raw.githubusercontent.com/lutzhamel/fake-news/master/data/fake_or_real_news.csv

The fields you are interested in are 'text' and 'label' with the obvious interpretations.

Use the vector model and text preprocessing techniques from class to construct a training data set

Use that training data set to construct a Naive Bayes classifier.

Use 5 or 10-fold cross-validation to compute the accuracy of your classifier.

The data set contains a large number of articles (takes a long time to train), you can downsample this to something like a 1,000 articles or so in order to speed up training and evaluation (see pandas.DataFrame documentation for sampling)

You are free to pick your own team (max three members)

Extra Credit: Try the same thing but instead of 'text' use 'title' for your training text. How does a classifier built on this data set compare to the original classifier.