Document_Classification

May 8, 2018

0.0.1 Text Classification

The ability of representing text features as numbers opens up the opportunity to run classification machine learning algorithms. Let's use subset of 20 newsgroups data to build a classification model and assess its accuracy.

```
In [1]: from sklearn.datasets import fetch_20newsgroups
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.preprocessing import Normalizer
        from sklearn import metrics
        import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans, MiniBatchKMeans
        import numpy as np
```

0.0.2 Load Data

```
In [2]: newsgroups_train = fetch_20newsgroups(subset='train')
        print(list(newsgroups_train.target_names))

        newsgroups_test = fetch_20newsgroups(subset='train')
```

```
['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.s
```

To keep it simple, let's filter only 3 topics. Assume that we do not know the topics, let's run clustering algorithm and examine the keywords of each clusters

```
In [3]: categories = ['alt.atheism', 'comp.graphics', 'rec.motorcycles', 'sci.space', 'talk.pol

        newsgroups_train = fetch_20newsgroups(subset='train', categories=categories,
                                              shuffle=True, random_state=2017, remove=('headers
        newsgroups_test = fetch_20newsgroups(subset='test', categories=categories,
                                             shuffle=True, random_state=2017, remove=('headers

        y_train = newsgroups_train.target
        y_test = newsgroups_test.target

        vectorizer = TfidfVectorizer(sublinear_tf=True, smooth_idf = True, max_df=0.5,  ngram_r
        X_train = vectorizer.fit_transform(newsgroups_train.data)
```

```
X_test = vectorizer.transform(newsgroups_test.data)

print("Train Dataset")
print("%d documents" % len(newsgroups_train.data))
print("%d categories" % len(newsgroups_train.target_names))
print("n_samples: %d, n_features: %d" % X_train.shape)

print("Test Dataset")
print("%d documents" % len(newsgroups_test.data))
print("%d categories" % len(newsgroups_test.target_names))
print("n_samples: %d, n_features: %d" % X_test.shape)
```

```
Train Dataset
2801 documents
5 categories
n_samples: 2801, n_features: 241036
Test Dataset
1864 documents
5 categories
n_samples: 1864, n_features: 241036
```

0.0.3 Naive Bayes Model

```
In [4]: from sklearn.naive_bayes import MultinomialNB
        from sklearn import metrics

        clf = MultinomialNB()
        clf = clf.fit(X_train, y_train)

        y_train_pred = clf.predict(X_train)
        y_test_pred = clf.predict(X_test)

        print 'Train accuracy_score: ', metrics.accuracy_score(y_train, y_train_pred)
        print 'Test accuracy_score: ',metrics.accuracy_score(newsgroups_test.target, y_test_pre

        print "Train Metrics: ", metrics.classification_report(y_train, y_train_pred)
        print "Test Metrics: ", metrics.classification_report(newsgroups_test.target, y_test_pr
```

```
Train accuracy_score:  0.976079971439
Test accuracy_score:  0.832081545064
Train Metrics:                precision    recall  f1-score   support

          0       1.00      0.97      0.98       480
          1       1.00      0.97      0.98       584
          2       0.91      1.00      0.95       598
          3       0.99      0.97      0.98       593
          4       1.00      0.97      0.99       546
```

```
avg / total       0.98      0.98      0.98      2801

Test Metrics:               precision    recall  f1-score   support

          0       0.91      0.62      0.74       319
          1       0.90      0.90      0.90       389
          2       0.81      0.90      0.86       398
          3       0.80      0.84      0.82       394
          4       0.78      0.86      0.82       364

avg / total       0.84      0.83      0.83      1864
```

Reference: Mastering machine learning using python in six-steps book