NaiveBayes

December 20, 2016

1 Naive Bayes (the easy way)

We'll cheat by using sklearn.naive_bayes to train a spam classifier! Most of the code is just loading our training data into a pandas DataFrame that we can play with:

```
In [1]: import os
        import io
        import numpy
        from pandas import DataFrame
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.naive_bayes import MultinomialNB

        def readFiles(path):
            for root, dirnames, filenames in os.walk(path):
                for filename in filenames:
                    path = os.path.join(root, filename)

                    inBody = False
                    lines = []
                    f = io.open(path, 'r', encoding='latin1')
                    for line in f:
                        if inBody:
                            lines.append(line)
                        elif line == '\n':
                            inBody = True
                    f.close()
                    message = '\n'.join(lines)
                    yield path, message


        def dataFrameFromDirectory(path, classification):
            rows = []
            index = []
            for filename, message in readFiles(path):
                rows.append({'message': message, 'class': classification})
                index.append(filename)
```

```
            return DataFrame(rows, index=index)

        data = DataFrame({'message': [], 'class': []})

        data = data.append(dataFrameFromDirectory('e:/sundog-consult/Udemy/DataScience/emails/sp
        data = data.append(dataFrameFromDirectory('e:/sundog-consult/Udemy/DataScience/emails/ha
```

Let's have a look at that DataFrame:

```
In [2]: data.head()
```

```
Out[2]:                                                    class  \
        e:/sundog-consult/Udemy/DataScience/emails/spam...  spam
        e:/sundog-consult/Udemy/DataScience/emails/spam...  spam
        e:/sundog-consult/Udemy/DataScience/emails/spam...  spam
        e:/sundog-consult/Udemy/DataScience/emails/spam...  spam
        e:/sundog-consult/Udemy/DataScience/emails/spam...  spam


        e:/sundog-consult/Udemy/DataScience/emails/spam...  <!DOCTYPE HTML PUBLIC "-//W3C//DTD H
        e:/sundog-consult/Udemy/DataScience/emails/spam...  1) Fight The Risk of Cancer!\n\nhttp
        e:/sundog-consult/Udemy/DataScience/emails/spam...  1) Fight The Risk of Cancer!\n\nhttp
        e:/sundog-consult/Udemy/DataScience/emails/spam...  ##################################
        e:/sundog-consult/Udemy/DataScience/emails/spam...  I thought you might like these:\n\n1
```

Now we will use a CountVectorizer to split up each message into its list of words, and throw that into a MultinomialNB classifier. Call fit() and we've got a trained spam filter ready to go! It's just that easy.

```
In [3]: vectorizer = CountVectorizer()
        counts = vectorizer.fit_transform(data['message'].values)

        classifier = MultinomialNB()
        targets = data['class'].values
        classifier.fit(counts, targets)
```

```
Out[3]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Let's try it out:

```
In [4]: examples = ['Free Viagra now!!!', "Hi Bob, how about a game of golf tomorrow?"]
        example_counts = vectorizer.transform(examples)
        predictions = classifier.predict(example_counts)
        predictions
```

```
Out[4]: array(['spam', 'ham'],
              dtype='|S4')
```

1.1 Activity

Our data set is small, so our spam classifier isn't actually very good. Try running some different test emails through it and see if you get the results you expect.

If you really want to challenge yourself, try applying train/test to this spam classifier - see how well it can predict some subset of the ham and spam emails.

In [ ]: