NLP Lab Session 8 (c)

Creating data and using Weka for classification

In the NLTK, we have been using the Naïve Bayes classification algorithm, which has reasonable performance on many classification problems. However, on text classification problems, it is also often the case that the Support Vector Machines (SVM) algorithms have the best classification performance.

For this, we can use the Weka software
package, which is an open software application that collects together many data mining algorithms, including classification algorithms. Weka is particularly useful for classification experiments, for example, to compare different types of features (and less useful for creating a classifier for unlabeled data).

To prepare data for Weka, we will use the NLTK to create feature sets as usual, but then I have written a function that will write the feature sets to a file that can be used for classification testing in Weka. One file format that Weka can use is a csv (comma separated values) file where the first line should contain the names of the features (which Weka calls attributes) separated by commas and the remaining lines have one line per training example, with the feature values separated by commas.

Note that you should not allow any feature name or value containing quotes, apostrophes, or commas. Also, Weka treats the class label (for example, 'pos' or 'neg') as one of the features, and it is customarily the last one.

In order to demonstrate this function, let's load the movie reviews in NLTK.

```
>>> from nltk.corpus import movie_reviews
>>> import random

>>> documents = [(list(movie_reviews.words(fileid)), category)
       for category in movie_reviews.categories()
       for fileid in movie_reviews.fileids(category)]
```

Since the documents are in order by label, we mix them up for later separation into training and test sets.

```
>>> random.shuffle(documents)
```

We need to define the set of words that will be used for features. This is essentially all the words in the entire document collection, except that we will limit it to the 2000 most frequent words.

```
>>> all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
>>> word_features = all_words.keys()[:2000]
```

We give the same feature definitions as before for all words, except that we make sure that no word of length 1 is used. This eliminates the punctuation, including quotes, apostrophes and commas.

```
>>> def document_features(document):
        document_words = set(document)
        features = {}
        for word in word_features:
            if len(word) > 1:
                features['contains(%s)' % word] = (word in document_words)
        return features
```

Define the feature sets for the documents.

```
>>> featuresets = [(document_features(d), c) for (d,c) in documents]
```

Here is a function definition that takes featuresets and a path to where you want to write the csv file, and converts the featuresets to be a weka input file in csv format.

```
>>> def wekawrite(featuresets, outpath):
    # take featuresets defined in the nltk and convert them to weka
    # input csv file  and write the file to the outpath location
    # open outpath for writing - it should include the name of the csv file
    f = open(outpath, 'w')
    # get the feature names from the feature dictionary in the first featureset
    featurenames = featuresets[0][0].keys()
    # create the first line of the file as comma separated feature names
    #  with the word class as the last feature name
    featurenameline = ''
    for featurename in featurenames:
       featurenameline += featurename + ', '
    featurenameline += 'class'
    # write this as the first line in the csv file
    f.write(featurenameline)
    f.write('\n')
    # convert each feature set to a line in the file
    # with comma separated feature values, for booleans just write the
    # words true and false
    for featureset in featuresets:
       featureline = ''
       for key in featurenames:
          featureline += str(featureset[0][key]) + ', '
       featureline += featureset[1]
    # write each feature set values to the file
    f.write(featureline)
    f.write('\n')
    f.close()
```

Now we need to make a path to the output file (which will be weka input). Include the name of the file and the file extension .csv. (On my Mac, it's `outpath = "/Users/arw/Documents/Data/AAAdocs/movies.csv"`)

```
>>> outpath = <put your path here for the lab>
```

Now we can call the function:

```
>>> wekawrite(featuresets, outpath)
```

And we can go to the file, open it either in Excel or in NotePad++ to see the contents.

Now we can classify in Weka; go to the Weka document for instructions.