Design Details

Erricos Michaelides

July 2, 2017

1 Data Acquisition

Twitter provides a RESTful API for accessing tweets. It grants almost unrestricted access (by allowing 450 requests every 15 minutes). The first script was used to check the timeline of the account targeted ('premierleague'). After fetching the raw tweets I kept the sections relevant to our search from the raw data. The part that was most interesting was the text.

2 Simple Naive Bayes

The acquired data is passed to the Model training class to create our chosen classifier: Naive Bayes. This is a probabilistic classifier based on the Bayes' rule. Suppose that two random variables A and B are given with probability density $P(B \mid A)$ and $P(B)$ respectively. Then the posterior distribution is obtained by:

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

where **B** is the data and **A** are the unknown parameters. In other words Bayesian inference derives the posterior probability as a consequence of two antecedents, a prior probability and a likelihood function derived from a probability model for the data to be observed[5]. This likelihood function describes our belief that **B** would be the outcome if we knew **A** to be true. Inference about the parameters is then obtained from the posterior distribution.

- The *prior probability* is the distribution of the parameters before any data is observed, $P(\mathbf{A})$.
- The *likelihood function* is defined as: $f(b_1, \ldots, b_n \mid \mathbf{A}) = \prod_{i=1}^{n} f(b_i \mid \mathbf{A})$
- The *posterior probability*, $P(\mathbf{A} \mid \mathbf{B})$, is the distribution of the parameters after taking into account the observed data. This is determined by Bayes' rule, which forms the heart of Bayesian inference:

$$posterior \propto likelihood \times prior.$$

Naive Bayesian classification is called naive because it assumes class conditional independence. That is, the effect of an attribute value on a given class is independent of the values of the other attributes. In essence this means, that the appearance of a word in a tweet is independent of the appearance of any other word. Although it is very unlikely that words in a tweet are not related to each other, the assumption simplifies the model as it reduces computational costs. For our first iteration of the classifier we only used a simple bag of words stripping off extra information mentioned before: hastags, links, references and timestamps. The class assigned to a tweet is the one with maximum value for the following:

$$c_{MAP} = argmax_c P(c) \prod_i P(w_i \mid c), \tag{1}$$

where $P(\mathbf{c})$ is the class prior probability. It is multiplied by the product of all probabilities $P(w_i \mid c)$ of words in the tweet found in the given class' training set.

Our class prior is defined as:

$$P(c_j) = \frac{tweetcount(C = c_j)}{N_{tweets}}.$$

In other words it is equal to the count of tweets in that class over the total number of tweets in the training set.

The likelihood is defined as the product mentioned earlier. Each member of the product looks like:

$$P(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum\limits_{w \in V} count(w, c_j)}.$$

In order to accommodate for words in the tweet not seen in the training set, we apply the Laplace correction to each member of the product. Had this not been done such a case would yield to likelihood 0. V is the vocabulary of all words seen in the training set.

$$P(w_i \mid c) = \frac{count(w_i, c_j) + 1}{\left(\sum\limits_{w \in V} count(w, c_j)\right) + |V|}.$$

These equations were applied to a training set containing 407 hand labelled tweets. 150 of these were identified as Live (classID = 0) and 257 as past (classID = 1). Two files were then created containing the words seen in each class, along with the count of times they were seen and their probability yielded by the equations above. In this way when a new tweet is to be classified the process is faster. The only operation made is to check for its words in each class probability file and copy the probability values there. If a word in the tweet does not occur in the file, the Laplace correction is used as stated above. As the product of these probabilities would be infinitesimal instead of (1) we use:

$$c_{MAP} = \mathrm{a}rgmax_c(P(c) + \sum_i log(P(w_i \mid c))).$$

The tweet is assigned to the class that maximises the above equation. Classifying all test tweets, we obtain accuracy of:

(89.6%, 10.4%), correctly vs incorrectly classified tweets.