Tutorial 9: Connectionist and Statistical Language Processing

Naive Bayes Classifiers

Date: 16.01.2002
Student's name:

1 The Task: Email Filtering

We will again use the Weka machine learning package in this tutorial. The files for the tutorial reside in:

```
/courses/connectionism.winter.01/tutorial9/
```

Our topic is spam filtering of email, i.e., the task of classifying an email into two categories: spam (i.e., unsolicited commercial email) and regular email. We will use a dataset called Spambase consisting of data extracted from 4601 email messages. Spam filtering can be performed well by a Naive Bayes classifier, as introduced in the last lecture (see Androutsopoulos et al. 2000 for an example).

Each instance in Spambase consists of 58 attributes. Most of the attributes represent the frequency of a given word or character in the email that corresponds to the instance. Here is a definition of the attributes:

(1) `word_freq_`*w*: 48 attributes describing the frequency of word *w*, defined as the percentage of words in the email that match *w*, i.e., 100 * (number of times the *w* appears in the email) / total number of words in email.
(2) `char_freq_`*c*: 6 attributes describing the frequency of a character *c*, defined in the same way as word frequency.
(3) `capital_run_length_average`: describing the average length of uninterrupted sequences of capital letters.
(4) `capital_run_length_longest`: describing the length of longest uninterrupted sequence of capital letters.
(5) `capital_run_length_total`: describing the total number of capital letters in the email.
(6) `spam_class`: the target attribute denoting whether the email was considered `spam` or `nospam`.

Note that these attribute definitions are slightly different from the ones we discussed in the lecture. We are now using the frequency of a word (or character) in the email, instead of its frequency in the whole dataset. The attributes dealing with capital letters have not been discussed in the lecture.

2 Exploring the Dataset

Browse the file `spambase.arff` in the directory `tutorial9` using an editor such as `emacs` (or simply display the file with `more`).

Have a look at the attribute definitions. The designers of this dataset obviously had to make a decision as to which word or character frequencies to include. What do you think the criterion

was? Which words or character do you expect to be particularly informative?

In the lecture, we defined a Bayes classifier for Spam as follows:

$$v_{\text{NB}} = \arg\max_{v_j \in V} P(v_j) \prod_k P(w_k|v_j) \tag{7}$$

where V is the set of target classes (`spam` or `nospam`), and $P(w_k|v_j)$ is probability that word w_k occurs in the email, given the email belongs to class v_j. The likelihood term was estimated as follows (disregarding smoothing):

$$P(w_k|v_j) = \frac{n_k}{n} \tag{8}$$

where n_k is the number of times word w_k occurs in emails with class v_j, and n is the number of words in emails with class v_j.

In the present dataset, the attributes are numeric, and they already represent frequencies (see above). Suggest a simple change to (8) that takes this into account.

A lot of the instances in the file `spambase.arff` have zero frequencies. Does this mean that the data is sparse? Does it constitute a problem based on what you have assumed about the estimation in the last question?

3 Frequency vs. Capitalization as Attributes

We will start by building a simple Bayes classifier that only takes capitalization into account, i.e., the attributes `capital_run_length_average`, `capital_run_length_longest`, and `capital_run_length_total`.

Start Weka, load the file `spambase.arff` and select these three attributes and the target `spam_class`. Now apply filtering to create a new working relation (see the last tutorial sheet for details). Then go to the `Classify` menu and select `NaiveBayes` as the classifier to work with. Train the classifier using a percentage split of 66%.

Report the precision and recall for this classifier. How does it compare to a frequency baseline (always assigning the most frequent class)?

Now build a minimal classifier that only consists of one of the three capitalization attributes into account. Report precision and recall for each of them. What does this tell us about the assumption that these three attributes are independent? (Recall that Naive Bayes assumes that all attributes are independent.)

Now build a classifier that uses all the frequency features (but no capitalization). What is its precision and recall? Do you get an improvement if you build a classifier that uses all features, i.e., frequency and capitalization? Are the independence assumptions of such a classifier met?

4 Analyzing the Errors

Our classifier on the spam dataset makes two sorts of errors: *false positives*, i.e., it wrongly classifies an email as spam even if it's no spam, and *false negatives*, i.e., it wrongly classifies an email a no spam even if it's spam.

Which type of error is more serious for a practical application of this classifier? Check the performance of the best Bayes classifier that you have constructed on this data. Compare the

number of false positives and false negatives that it yields.

How could you improve the classifier so that it tries to minimize the number of false positives? Try to come up with a way that only changes the dataset, and not the learning algorithm.

References

Androutsopoulos, I., J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. 2000. An evaluation of Naive Bayesian anti-spam filtering. In G. Potamias, V. Moustakis, and M. van Someren, eds., *Proceedings of the Workshop on Machine Learning in the New Information Age at the 11th European Conference on Machine Learning*, 9–17. Barcelona.