Università degli studi di Padova
Corso di laurea magistrale in Ingegneria Informatica

Sistemi Intelligenti
Relazione del progetto

A spam classifier based on Bayesian networks

Alberto Franzin 1012883
Fabio Palese 1234567

Docenti:
Prof. Silvana Badaloni
Prof. Francesco Sambo

AA 2012-13

Contents

1 Introduction

This project has been developed as part of the course in Intelligent Systems, a.y. 2012/13, by Alberto Franzin and Fabio Palese.

The project consists in a spam classifier for emails based on Bayesian networks, a probabilistic model to represent conditional dependencies between random variables that can be used as a classifier in the field of supervised learning.

The code and the documentation of the project is available at `http://code.google.com/p/sist-int-2012project/`.

This documents is divided in four section: in the first one, we briefly introduce the theory behind Bayesian networks, in the second section we describe the working of project, then we show the results that we have obtained and finally we draw the conclusions.

2 Bayesian networks

2.1 The Bayes' theorem

The theorem stated by Thomas Bayes says that, given two events A and B, respectively with probabilities $P(A)$ and $P(B)$:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}.$$

We call $P(A)$ the *prior* probability of A, that is our initial, unconditioned knowledge about A, $P(B|A)$ is the *likelihood*, i.e. the function of the parameter having a certain value given the outcome, and $P(A|B)$ the *a posteriori probability of A given B*, that describes our modified knowledge about event A given that we know the event B has occurred.

While the Bayes' theorem is universally accepted, there are two possible interpretations: the frequentist approach, that relies on the knowledge (or the possibility of discovering) of the probability of an event

A typical application of Bayes' theorem is diagnostic. Given the results of a medical exam and the accuracy of the exam (percentage of false positives and false negatives), we can compute the probability of effectively having the disease.

2.1.1 Conditional independence

Given three events A, B and C, we say that A and B are *conditionally independent* if $P(A \cap B|C) = P(A|C)P(B|C)$, or, equivalently, $P(A|B \cap C) = P(A|B)$.

2.1.2 Explaining away

When an event has multiple causes, then we say that seeing one cause of the possible ones may "explain away" all the other causes, because it becomes less likely for them to happen.

For example, suppose that I am late for the lesson: I may be late because the train is late, or because I didn't hear the alarm. If the train is late, it becomes less likely that I haven't heard the alarm.

2.2 Bayesian networks

A Bayesian network is a probabilistic model that defines a probability distribution over a graph of variables. The nodes represent the events, and the (directed) arcs mean the probabilistic relationship between events. An arc going from node X to node Y with an associated probability p means that Y is conditionally dependent from X with probability p.

The name "Bayesian network" has been coined by Judea Pearl in 1985 (see Pearl), to underline three characteristics of this model:

1. it

2.2.1 The *naive* approach

Bayesian networks require a huge number of variables: because of the chain rule of probabilities, if a node has k incoming arcs, then it requires 2^k variables. Furthermore, it may difficult to compute the joint probability of variables.

The *naive* approach works under the strong assumption of independence among variables. It is called "naive" since this assumption is not always realistic: for example, if we meet, in a spam mail, the words "buy replica watches" (which already is a set of words) we expect to read some watch brands too. Anyway, while considering all the subset of words requires exponential time, the assumption of independence allows us to consider each word alone, pulling it out of context, and therefore it allows the computation to drop from exponential to linear time. Moreover, the calculation of the probabilities is now reduced to a product of the single probabilities of the variables. Maybe surprisingly, this approach works quite well in practice, being both fast and accurate, despite its "naivety".

2.3 Naive Bayes for spam classification

We describe now how to apply the theory above to a real problem, in this case the classification of a mail as a spam mail or a valid mail.

We define a mail as *spam* if it contains undesidered or illegal content. A valid mail is conversely defined as *ham*.

The naive approach that tells us to consider each words alone, allows to represent the document as a *bag of words*, which is a dictionary containing the words encountered in the mails, each one associated with its frequency.

In addiction to the words, often we can easily distinguish a spam email from a valid one by looking at it and recognizing some of these characteristics: bad grammar, bad syntax, excessive use of images, lots of links, and many others. Detecting these features may be useful to correctly classify an email.

2.3.1 Algorithm

Data structures To represent the *bag of words*, we need a dictionary of `(word, stats)`. We need also to store the statistics for the various features detected, in an analogous data structure containing `(feature, stats)`. In both cases, the statistics have to distinguish between frequency in spam mails and frequency in ham mails. This is enough to contain all the informations we need to know about the training set.

The statistics of the mails in the validation set and in the test set will be stored in similar data structures, but their stats will contain only the number of times the words and the featured have been observed, since the network does not know the status of the mail when these operations are performed.

Training To train the system, we have to feed the network with the training set. The network will read the mails, extract the actual content and adjust the count of both words and features.

Validation The next step is to measure how good the results of the training are. To accomplish this, we provide the network other mails of which we already know the status, to later check the results. The network will read each mail, extract the stats and compare them to the overall statistics generated during the training step. The result of the classification will be the class that maximizes the probability for the mail to belong to that class.

Testing sf

3 Implementation

Here we describe our implementation of a Naive Bayes spam classifier. Further documentation is available with the package.

The project has been written in Python. It requires two external modules, `bs4` (`BeautifulSoup`) to parse a text and `Ply` to perform the lexical analysis of a text.

3.1 Structure

The package is composed as follows:

spam_bayes the module contianing the `__main__` class;

naive_bayes the module defining the Bayesian network;

trainer contains the trainer;

classifier applies the Bayesian logic to classify a mail;

config contains some general configurations, manages the settings defined by the user;

gen_stat

test_stat

lexer implementing the lexical analyzer to compute the statistics;

utils with some methods used in more classes.

3.1.1 Notes on implementation

eh?

4 Tests and results

4.1 Dataset

The dataset used is the SpamAssassin Public Corpus provided by SpamAssassin at `http://spamassassin.apache.org/publiccorpus/`, available for development purposes. It contains about 6400 mails in english language, collected from 2002 to 2005, with approximately the 31% of the mails being spam.

For testing, we have also used some of spam and ham mails received by us.

4.2 Modalities

4.3 Features of spam and ham mails

The following features has been selected as relevant and are tracked by the network:

- the number of words written in uppercase characters: since it is considered the equivalent of screaming, many spammers use this gross gimmick to gain attention;
- the number of urls found: we expect to find a lot of addresses of websites where to buy goods or where some criminal activity will be performed (phishing, stealing credentials, etc.);
- the number of email addresses: for the same reason of the previous feature. Furthermore, spammers operate on a large scale, so it is possible for a spam mail to have multiple recipients;
- the number of words shorter than a given threshold (given as a parameter): a trick to fool a system based on word classification is to hide the words. One of the most common and easier way of accomplish this, is to separate each letter with spaces. Since our token separation relies on whitespaces between words, a word like `s p a m`, which a human will easily understand as a single one, will be classified by the network as four different words. In a common, correct plain english text, the ratio of short words like `a` or `I` will be substantially lower, not to mention other random letters;
- the number of non-address words longer than a given threshold (parameter): similarly to the previous feature, a spam mail is likely to contain a number of very long, meaningless words much higher than a ham mail. Since there aren't many very long legit words[1], this is a reasonable feature to track;

[1] The average length of english words is slightly higher than 5 (`http://www.puchu.net/doc/Average_Word_Length`), while the majority of words appearing in a text is from 2 to 5 letters long. [2]

- the number of words in the form `username@host`: the tests have shown that the ham mails in the SpamAssassin Public Corpus contain many tokens of this type;
- the number of words in "title" format (first letter capital, remaining letter lowercase);
- the number of words in a mail.

Many other interesting features can also be used: for example, a heavy use of images as text replacement, the consistency of the email header, the consistency of the urls with the link they claim to go, the list of people the user have mailed before, the provenance of the mail from a known spammer or IP range, the correctness of words (by ckecking a dictionary), to name a few. All of these features require a semantic analysis of the mail, which goes beyond the purpose of this project, since no one of these features is directly connected to Bayesian networks. Nonetheless, all of these features can be used by a Bayesian network, just like the previous ones, and are very likely to increase the accuracy of the classification.

It should be noted that while some of these features are common for the majority of spam, some others are derived from our particular mail archive. Moreover, since many of the mails in the archive are ten years old, more recent spam mails may contain different features. The maintainers of the archive warn to use the archive only for development purposes, not in a live system.

4.4 The design parameters

The Bayesian network requires some parameters to be tuned: for example, the "spamicity" threshold (how much the normalized probability of being spam should be to be really considered as spam), or the "weight" of the feature stats with respect to the word stats when computing the final probability. Different parameter configurations lead to obtain different accuracy values, so we have to choose the best one.

We'll now present the results obtained. Since there are multiple degrees of freedom, we show how the accuracy varies when trying to change one single parameter at time, all the other ones being set to their optimal value.

Size of the training set For finding the best size of the training set, we have begun with few mails, then we have tried to increase the size of the training set until the accuracy has started to decrease (the *early stopping* technique).

Size of training set		
Spam	Ham	Accuracy
10	10	0.1
20	20	0.2
50	50	0.3
100	100	0.4

As we see, only 50 spam and 50 ham mails are enough to have best accuracy. Given that we have over six thousand of mails in the archive, it is a very small size.

"Spamicity" threshold Another very important parameter is what is sometimes called the "spamicity" threshold, namely the value that the normalized probability $\frac{p_{spam}}{p_{spam}+p_{ham}}$ must reach for the mail to be classified as spam. This is crucial: a low threshold will bring in many false positives (good mails will be marked as spam), while a too high threshold will classify some spam mails as ham, thus having a lot of false negatives. Anyway, the choice of the threshold to be 0.5 may not be obvious, since we may want to be more tolerant with words we have never met before, or we way consider the chance of false positive too bad for us, and therefore lift the threshold to ensure this fact, allowing at the same time some spam to pass through the filter.

Threshold	False positives	False negatives	Accuracy
0.2	10	0	0.1
0.4	10	0	0.1
0.5	10	0	0.1
0.6	10	0	0.1
0.7	10	0	0.1
0.8	10	0	0.1
0.9	10	0	0.1
0.95	10	0	0.1

The highest accuracy is obtaines with the "trivial" threshold of 0.5. However, if we aim to minimize false positives or false negatives, we may prefer other thresholds.

Feature/words stats proportion Another problem is to determine how much weight should we assign to the features statistics, and how much to the word statistics.

The final formula for spam is $p_{spam} = W \times p_{fs} + (1 - W) \times p_{ws}$, where p_{spam} is the probability of the mail being spam (yet to be normalized), p_{fs} is the probability of being spam computed using the statistics of the features, p_{ws} is the probability of being spam computed using the statistics of the words, and W is the parameter we want to adjust. The formula for ham is the dual.

A value close to 1 will lead to a classification based solely on the features, thus almost ignoring the actual content of the mail. A low value will instead disregard the features, and therefore many useful and common characteristics of spam mails.

Threshold	Accuracy
10	0.1
20	0.2
50	0.3
100	0.4

4.5 Analysis of results

Here we show and analyze some interesting results observed. First of all, here are the feature statistics computed.

	Occurrences	
Feature	# in spam	# in ham
10	10	0.1
20	20	0.2
50	50	0.3
100	100	0.4

Length of words Clearly, "very short" and "very long" words (out of our thresholds) are mostly common in spam mails.

Most common words It's interesting also to look at the statistics of the single words. We have already discussed about how the distribution of unusually long words. Now, let's take a look at the most common (and untrivial) spam and ham words.

5 Conclusions

We have seen that the Naive Bayes approach requires a surprisingly small size for the training set, with about a hundred of mails giving the highest accuracy, a tiny fraction of the corpus available with the data set. When the size of the training set grows, the accuracy degrades rather quickly.

Note that a more complex system, better if integrated into a mail server or client, can surely achieve higher accuracy. We have already named a few features that can be added to the classifier to improve its performance. Anyway, we have shown that a very simple and "naive" Bayesian classifier can work pretty well with little data and a handful of key features.

References

[1] J. Pearl. *Bayesian Networks: A Model of Self-Activated: Memory for Evidential Reasoning*. Report. Computer Science Department, University of California, 1985. URL http://ftp.cs.ucla.edu/tech-report/198_-reports/850017.pdf.

[2] Bengt Sigurd, Mats Eeg-Olofsson, and Joost Van Weijer. Word length, sentence length and frequency – zipf revisited. *Studia Linguistica*, 58(1):37–52, 2004. ISSN 1467-9582. doi: 10.1111/j.0039-3193.2004.00109.x. URL http://person2.sol.lu.se/JoostVanDeWeijer/Texts/studling.pdf.