Data Mining and Machine Learning
Course Work 2008-2009
The Stock Market Learnt From Websites

Eng. Fernando Cervigni Martinelli, M.S.c. student for Erasmus Mundus Masters in Computer Vision and Robotics (ViBot), Heriot-Watt University, University of Burgundy, Universitat de Girona

I. Introduction

Given that most of the information the society produces is in text format, the ability of mining meaningful data out of plain text is precious. The fields of application of text mining are numerous: security – e.g. surveillance of communication between suspicious people –, Customer Relationship Management (CRM), Online Media and Academic Research, to mention a few.

An important way of mining a text is assessing its bias, its impartiality. Such a study is called the "Sentiment Analysis" of a text. The idea can be summarised by the ability of saying, for instance, if a movie review is positive or negative.

The scope of this project falls, in a way, into the "Sentiment Analysis" field of Data Mining. The main objective is to find out whether the stock market has gone up or down by judging the text content of a set of reviews extracted from web pages.

The FTSE 100 index has been chosen as the focus of the current analysis for the simple fact that there is more information available about it than there is about any other individual share price. Therefore, richer analyses have been made, as one of the decisive factors for the success of a Data Mining/ Machine Learning project is the representativeness of the training and testing sets.

A. The definition of noise on a web page

The noise on the stock prices can be defined by the unexpected behaviour the market shows in some occasions. Even if all the economic indicators are positive, there could happen to be, for example, an important share holder that decided to sell hastily all his stocks, entailing a fall of the price of those stocks.

The noise on websites is one of the inconveniences and obstacles to mining data seeking to grasp the Stock Market's behaviour. Because of the fact that the language used on articles, written by humans, is predominantly subjective, there can be complex arrangements of ideas that can confuse an analytic data mining strategy. If words like "good", "high" and "up" are considered signs of a good stock market day, on a text like the one below, the words "good" and "high" are actually noise:

> *"In spite of the good weather and the high expectations of the investors, today's movement of the stock market was disappointing."*

Any part of the text that is not directly related to the actual result of the stock market can be deemed as noise, as states one of its definitions:

> *"Noise is the incomprehensibility resulting from irrelevant information or meaningless facts or remarks."*

It is important to notice that, in order to define noise on websites, it is necessary to define what the purpose of the Data Mining being applied is. What may be noise for one purpose of Data Mining can be the main source of information to others. The example of noisy paragraph above is considered in this specific context of mining websites in search for the stock market movement.

II. The Naive Bayes Classifier for data mining

A. Choice of the machine learning algorithm to learn the movement in the stock prices

The Naive Bayes Classifier has been chosen as the machine learning algorithm for this project for a

number of reasons. When compared to other methods, the Naive Bayes Classifier did not have some very relevant drawbacks the latter had.

Similarity metrics between texts could have been used. However, phrases such as "I do like this book" and "I don't like this book" are considered to be very similar by most commonly-used similarity measures. Indeed, the only differing token – the negation term – turns out to separate the two phrases into opposite semantic classes.

Decision trees, as they have been presented in the lectures, could have been prohibitively complex to implement. Some simplifications would be required so that only some words in the text and their frequencies would be used as decision nodes. The notion of information entropy would also need to be reformulated to enable the calculation of the best information gain.

The Naive Bayes Classifier and its variations, on the other hand, are very straightforward and suit fairly well the scope of this project. The Naive Bayes Classifier also seemed to have been the most popular choice amongst recent research studies in the field of text analysis.

All these factors led to the conclusion that the Naive Bayes Classifier would be the best reasonably-simple method to be implemented for this project.

B. Pre-processing of the web pages and its effect on the inductive bias of the chosen algorithm

The texts extracted from the websites, collected to form the training and testing sets, were manually copied and pasted to plain text files. Initially, though, the implementation of an automatic web crawler was considered.

This crawler would seek the meaningful text present on a list of websites – that is, all the text excluding html tags, other links, figures advertisements and other kinds of undesired information.

Nevertheless, it turned out that, since this was not the main focus of the project at hand, the time spent on the implementation of this crawler would not be worth it.

The input of the Naive Bayes learner has been, then, the text contained in the main body of some websites that talked about the index FTSE 100. However, before adding the words present on the input text to its vocabulary and calculating its frequencies, three pre-processing operations have been performed:

- All tokens that contained non alphabetical symbols – e.g. numbers, punctuation, "%", "$", "&" and such symbols – have been ignored;
- From the remaining words, all those who belonged to a "stop list" – that is, words that were considered to be meaningless to our application, and thus regarded as noise – were also ignored;
- The words that appeared less or more than given frequency thresholds on the training set have similarly been ignored. This strategy has been incorporated as a heuristic suggestion commonly applied in text analysis and cited on the course book.

Inductive learning classifiers are able to classify unseen instances only because they have an implicit bias for selecting one consistent hypothesis over another. On the other hand, an inductive bias may be the cause of classification errors when it is inconsistent with the target concepts.

To analyse whether or not the inductive bias of the learner has been affected, the definition of the inductive bias of a Naive Bayes Classifier itself must be clarified.

In the case of the Naive Bayes Classifier, the main inductive bias is made quite explicit through the a priori probability term that appears in the classifying equation. The a priori probability is an inductive bias as it assumes that the frequency of the target values is not going to change from the training to the testing dataset. This means that if the market goes up on 60% of the days of the training set, there is a bias towards the prediction that the market is going up on the testing days.

Another inductive bias, illustrated in equation 1 below, corresponds to the "naive" assumption that the conditional probability of a word given a target

concept is independent of the conditional probabilities of other words given that concept.

$$P(a_1, a_2 .... a_n \mid v_j) = \prod_i P(a_i \mid v_j) \qquad \text{Eq. 1}$$

This is obviously not true, since it is much more likely to find for instance the word "stock" near the word "market" than anywhere else in the text. This bias simplifies tremendously the problem of calculating the words frequencies/probabilities, fortunately not affecting substantially the performance of the Naive Bayes Classifier.

Since these two biases are not directly correlated with the pre-processing operations described earlier on, they have not been affected by them. Notwithstanding, each one of the pre-processing operations has added one corresponding bias:

- ***Exclusion of non-alphabetical symbols:*** this adds the bias that implies non-alphabetical symbols are not correlated to the movement of the stock market;
- ***Stop list exclusion:*** this operation generates a bias that implies that none of the words on the list are important for the prediction of the stock market movement;
- ***The low and high frequency cut-off:*** cutting off words that appear only a few or many times on the training examples generates the bias that those words are irrelevant for the prediction.

None of these biases achieve complete accuracy, but they may improve the performance of the classifier as more weight is given to the remaining and potentially more useful words in the vocabulary.

III. Implementation and Training

In order to implement the Naive Bayes Classifier with the objective of binary classifying texts, python has been chosen as the programming language. This was due to the fact that python is a high-level language that handles effortlessly many of the aspects necessary for text analysis.

C. Algorithm implementation and learning

A Naive Bayes Classifier python class has been created to implement the Naive Bayes Classifier. This class has as attributes one list containing the words provided as input text examples of the "up" target concept as well as another one for the "down" target concept.

In practice, the Naive Bayes Classifier learns through the calling of a class method called "learn()". The "learn()" method has as input a comma separated value file with a list of names of plain text files containing information concerning the FTSE index for a specific day. The second input is the name of the "stop list" file, which contains the undesired words, which are to be filtered during the learning procedure. The definition of the method is as follows:

NaiveBayesClassifier.learn(training_file.csv, stoplist_file.txt).

Along with the name of each text file appearing in the *.csv* training file, comes a binary number – a label that indicates whether the market has gone up or down on that day.

After the application of the pre-processing heuristics on the input training texts, each word is added to the vocabulary of the learner. When the day is labelled as "up" the words are added to the "up" target concept vocabulary and similarly for days labelled as "down".

The repetition of words is dealt with in an elegant way since the dictionary tool in python makes it easy to keep track of the frequency of each word of the vocabulary list.

The output of the learning process is two word lists, one for each target concept, containing all the post-processed words and their respective incidence on the training set texts.

The "intelligence" of the classifier is all stored in these two lists. With these probabilities – each word related to each target concept, "up" or "down" – we use equation 2 below to classify an unseen text as "up" or "down".

$$v = \max_{vj \in V} P(v_j) \prod_i P(a_i \mid v_j) \qquad \text{Eq. 2}$$

$$P(a_i \mid v_j) = \frac{n_c + 1}{n + |Vocabulary|} \quad \text{Eq. 3}$$

Where:

- $P(v_j)$ is the a priori probability of the target concept j;
- n_c is the frequency of word a_i on the list of words of target concept j;
- n is the number of different words on the list mentioned above;
- $|Vocabulary|$ is the sum of the number of different words on the lists of both target concepts;
- i varies from 1 to the number of words present in the text being classified.

In the case of this work, the a priori probabilities have been 43% for "up" and 57% for "down".

D. Justification of the training set size

The training set used in this work was comprised of daily reports on the FTSE 100 index during 3 months: August, September and October – November has been spared to compose the testing set – **and the specific report of July the 8th, my birthday**. These reports have been extracted from three different specialised websites in finances and stock markets[1].

August has been included on purpose because the crisis had not yet completely surfaced. It is possible that this contributed to the training set with a slightly different and complementary vocabulary than the one used since the crisis has begun, although this is just a hypothesis.

The total number of words in the "up" list without the pre-processing operations was 18,364 whereas the number in the "down" list was 23,030. This has been considered a satisfactory amount of words when compared to other research examples of text analysis and experiments found on the related literature.

[1] http://ftse-100-news.newslib.com, www.thisismoney.co.uk and http://myfinances.co.uk

IV. Results on Testing Set

A. Performance of the algorithm on the test set

As stated in the previous section, the month of November has been used as the test set. There is no restriction that requires the training set to be composed of days preceding the ones on the testing set.

Although random days could have been picked out of the four months to compose the testing set, the chronological organisation prepares the context for the challenge of predicting the movement of the next day[2].

In order to test the learner and the influence of the pre-processing operations on the accuracy of the final predictions, a testing procedure with varying parameters has been set up.

To first measure the influence of the "stop list", two learners were set to predict the movement of November, one having been trained using the "stop list" and the other completely ignoring it. Table 1 shows the percentage error obtained for each of the testing configurations. The first number between parentheses shows what proportion of the error is actually composed of false positives – days where the market went "up" but the classifier labels as "down" – and the second shows the proportion of false negatives.

	Stop list	
	With	Without
Error (%)	20 (0, 20)	35(5, 30)

Table 1 - Results with and without the stop list

When using the "stop list", the first column shows that the Naive Bayes Classifier correctly labels 16 out of the 20 working days of November. The performance was 75% worse when the learner was trained without the list.

Another test sequence has been effectuated to assess the influence of the exclusion of words that appeared less than a minimum number of times. Table xx below shows the results in the same way as

[2] If one tries to predict tomorrow's movement, they must accept to have only past examples on the training set.

Table 1, but for different minimum frequency cut-off thresholds.

	Minimum frequency cut-off threshold		
	1	2	3
Error (%)	35 (0, 35)	35(0, 35)	40(0, 40)

Table 2 - Results for different minimum frequency thresholds

It is interesting to notice how the elimination of the low frequency words has biased the learner toward optimistic market predictions. Before using any low-frequency cut-off filter, the error was limited to 20%, which has increased to 35 for a threshold of 1 word and then to 40 for a threshold of 3 words. This shows that in this specific training set chosen for this work, the words that appear few times tend to help classify the unseen text as "down".

Tests were also conducted excluding the most frequent words. Not surprisingly, words like "up" and "down" where among the ones eliminated by this filter, since the other commonly frequent words like "the" and "of" where on the "stop list" and had already been ignored. This filter turned out to worsen considerably the performance of the learner.

The bias generated by eliminating the least and the most frequent words was, in this case – unlike in some examples in the literature – inaccurate.

B. Effect of the noise on stock prices and web pages on the learning performance of the algorithm

The focus of the Naive Bayes Learner described in this project was websites talking about the FTSE 100. However, since the FTSE is a weighted composition of the 100 largest UK registered companies, all reports about it are permeated by comments on the performance of many of these individual shares.

It is evident that there is a strong correlation between the performance of each share composing the FTSE 100 index and its movement. Nevertheless, regardless of the movement of the FTSE 100 index, there are almost always shares going up and down.

On most of the collected reports about the FTSE 100, there were also comments on the shares that went down and up. These can be seen as noise as they are not accurately representative of the overall index.

It goes without saying that the performance of the classifier would have been much better if all the comments on individual shares had been removed, leaving only specific comments on the FTSE 100 index. This was not done because it would have been a human-touch biased pre-processing. The way the information was collected for this work, it could similarly be done by an automatic crawler for a dataset a thousand times larger.

To summarise, the answer to whether the noise on the websites affect the performance of the learner is clearly yes.

On the other hand, according to the way the noise on stock prices was described on section I, it does not affect the performance of the learner, being transparent to it. The noise generated by an important shareholder that influences the market is already taken into account by the learner through the label of that day. It does not generate confusion between the words on the websites and the movement of the FTSE 100 index.

V. The Challenge

A. The prediction of the stock price the next day

A first and straightforward strategy to try to predict the movement of the next day has been to train the learner with the results of the next day. In other words, the training days have all been labelled with the movement of the following day.

The same label-shift procedure has been applied for the testing days, so that it was possible to measure the accuracy of the Naive Bayes learner when trying to predict the next day movement on all days of November. Interestingly, the error obtained when classifying the testing days was of only 35%. This can be only a mere coincidence – instead of 35% we could have obtained 65% error – but this can also mean that there are possibly some words on each daily report that somehow indicate a

meaningful feeling, or expectation, for the following day.

The error probability of a random classifier for data with 50:50% a priori probabilities is 50%. However, because of the reduction in the information entropy, this error approaches zero as the a priori probabilities are unbalanced towards 100:0%. In fact, the random error for a priori probabilities *a:(1-a)* can be demonstrated to be *2.a.(1-a)*. The 35% error obtained is well below the expected 49% (*2.0,43.0,57*) for a random classifier.

Whether there are really trustworthy hints for the future performance of the stock market or this finding has been a simple coincidence, only exhaustive testing with wider databases can tell.

Nevertheless, some other modifications on the previous approach might be applied to help predicting the movement of the next day:

- ***Training up until the preceding day***

Instead of proceeding like above, where a number of days are used for the learning and then a number of days are tested, in the case of a real prediction attempt, all the days up until the date should be used as the testing set. This would help the learner to be as up-to-date on the market as possible.

- ***An alternative way of calculating the a priori probability***

It is evident that the market shows long and mid-term trends. It is largely on this assumption that the technical stock market analysis is based upon. Bearing this in mind, it may be adequate to weight more the movement of recent days, in detriment of old days, for the calculation of the a priori probabilities.

One suggestion of such calculation is illustrated in equation 3 below:

$$P'(v_j) = \sum_{i}^{N\_train\_days} e^{-(N\_train\_days-i).cte}.C_j(i) \qquad \text{Eq.3}$$

Where:

- *N_train_days* is the number of days in the training set;
- $C_j(i)$ is the label for the target concept $j-1$ if the day is labelled as the target concept *j*, 0 otherwise;
- *cte* is a constant that controls the importance of the past – during a crisis one might decrease the importance of the past.

The exponential weights can be graphically visualised in Figure 1 below, for *cte* = 0.05 and *N_train_days* = 20.

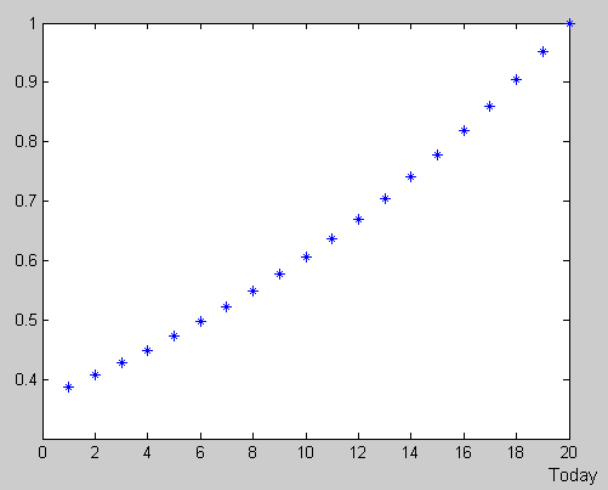

Figure 1 - Proposed weighting for a priori probability

It is important to notice that after calculating $P'(v_{Down})$ and $P'(v_{Up})$, these two numbers need to be normalised to add up to one, like happens with usual a priori probabilities – although, for the classification comparison, this normalisation has no effect.

VI. Conclusion

This report has been very useful in order for the students to have the opportunity to apply and observe a machine learning algorithm working. It has been rather satisfying to obtain such good results from the implementation of what seemed to be theories distant from practical use.

Since this is a very practical subject, it would be really helpful to have ongoing practical exercises or laboratories to allow the students to experiment on-the-go, rather than only on the last weeks of the semester. Besides, this could also be a way of taking the entire concentrated assessment role off of this project and distributing it among other activities.