The Use of Domain Knowledge in Feature Construction for Financial Time Series Prediction

Pedro de Almeida[1] and Luís Torgo[2]

[1] Physics Department, Universidade da Beira Interior, 6200 Covilhã, Portugal
nop00997@mail.telepac.pt
[2] LIACC, Rua Campo Alegre, 823 – 2º, 4150 Porto, Portugal
ltorgo@liacc.up.pt

Abstract. Most of the existing data mining approaches to time series prediction use data preparation techniques involving an embed of the most recent values of the time series, following the traditional linear auto-regressive methodologies. However, in many time series prediction tasks the alternative approach that uses derivative features constructed from the raw data with the help of domain theories can produce significant prediction improvements. This is particularly noticeable when the available data includes multivariate information but the aim is still the prediction of one particular time series, a situation that occurs frequently in financial time series prediction. This paper presents a method of feature construction based on domain knowledge that uses multivariate time series information and improves the accuracy of next-day stock quotes prediction, when compared with the traditional embed of historical values extracted from the original data.

1 Introduction

Recently, several data mining techniques have been applied with success to time series prediction based on samples of historical data [1], [5], [20]. Most of these approaches use supervised machine learning techniques and a data preparation stage that produces a set of examples in a two-dimension, tabular, "standard form" [28]. Several approaches can be used to prepare this kind of tabular representation from the time series raw data. Choosing the most appropriate set of features for the time series problem in hand can have a significant impact on the overall accuracy of the prediction models. This is the main problem addressed in this paper, for the particular case of financial time series prediction.

1.1 Traditional Temporal Embed

The simplest and most common procedure of transforming the original time series data to produce a set of examples in tabular form is to use the last known values of the time series as features describing each example, and using the next value of the time series as the target value of the example. This auto-regressive data preparation

technique is usually called “time-delay embedding” [22], “tapped delay line” or “delay space embedding” [17].

To illustrate the basic temporal embedding, let us consider an univariate time series

$$X(t) = \ldots, x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, \ldots$$

from which the values up to x_t (that is, x_t, x_{t-1}, x_{t-2}, …) are known. Let us further consider as our objective the construction of a model to predict the next value of the same time series (x_{t+1}), using some supervised data mining process. Assuming that the next value of the time series depends at most on the k previous values, traditional time embed approaches will describe each example using k features, each taking one of the values of the k previous measurements of the time series (x_t, x_{t-1}, x_{t-2}, …, $x_{t-(k-1)}$). Each complete example will thus have the form

$$x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-(k-1)}, x_{t+1}$$

where x_{t+1} is the value of the target (dependent) variable.

This kind of data preparation forms the basis of the classical autoregressive time series prediction methods like AR [29] or ARMA [6], and is theoretically justified by the Takens theorem [24]. This theorem states that, with some restrictions, a number of $(2 \cdot N)+1$ past values is enough to reconstruct the model of a noiseless system with N dimensions. In the case of the described $X(t)$ time series, assuming absence of noise, and considering the series to be generated by a N-dimensional system, the features x_t, x_{t-1}, x_{t-2}, …, $x_{t-(2N)}$ would be enough for the extraction of the system model and for the prediction of the following values of the time series.

1.2 Limitations of the Temporal Embed

It should be remarked that many real-life systems suffer complex interactions with other systems and, even if they have an intrinsic linear behavior, tend to generate time series that present severe problems to comply with the Takens theorem restrictions. In particular, this theorem does not seem to apply to most financial time series and, particularly, to stock exchange time series.

In effect, the stock exchange time series are generated by extremely complex systems that involve the interaction of thousands or millions of independent agents (the investors) each capable of changing its behavior over short time frames according to a virtually limitless number of possible individual “states”, resulting in a system with a number of dimensions that, in practical terms, must be considered infinite [14]. This fact implies that the number of historical values needed to construct each example in a way conforming to the Takens theorem conditions would be unrealistic (the necessary data would not be available and would be unmanageable by the machine learning algorithms). Moreover, the system global behavior is not static over time (for instance, the number and individual characteristics of the intervening agents are not constant). This non-stationary behavior of the system means that “old” time series data is not truly representative of the current system (in fact, it was generated by a related but different system that no longer exists) and using them blindly to generate predictions for the current system can be misleading.

These characteristics of stock exchange time series result in the almost unanimous admittance that base data consisting solely of historical values of the time series being predicted do not contain enough information to explain more than a fraction of the variance in the time series. In fact, the Efficient Markets Theory [8], now somewhat discredited ([1], [11], [15]) but still accepted as basically correct by many economists, states that those data do not contain any information useful for the prediction of the future behavior of the system.

The problems that impair the applicability of Takens Theorem to very complex dynamic systems do not seem to be reduced by the use of multivariate information as basis data for time series modeling.

As an example, the original data we use in our experiments includes 7 values for each day and each stock. In effect, this corresponds to 7 time series that are available to be used as basic data for time series model construction. In this context, using all the 7 time series with an embed dimension of, say, 25 (obviously insufficient to include enough information to describe the system), would result in 175 features describing each example. Even if those features were discretized to adopt a maximum of 5 different values, that would create a "representation space" with the capacity of distinguishing 175^5 different cases. This kind of input dimension is obviously too large considering the number of training and testing examples available (10 years of daily data correspond to around 2400 records), and would result in a very sparsely populated space that would severely limit the efficiency of most machine learning algorithms [3], [23]. This problem is reinforced by the fact that due to the system complexity and the lack of sufficient information on the base data the system behaves as if each variable (dependent and independent) includes a strong noise component. Thus, since all the information present in the base data only explains a small proportion of the system variance and the information present in our 175 variables will explain an even smaller proportion of that variance, we can expect each of those variables to have a very small individual relevance to the desired prediction.

1.3 Alternatives to the Temporal Embed

In situations where heavy overfitting problems can be expected due to the sparsely populated representation space and when noisy data are associated with a large number of possible input variables with low individual value to knowledge extraction, it is frequently possible to obtain better results by combining several of the original variables [12], [16], [19]. The aim of developing such feature combinations is the creation of a reduced set of "derivative" variables having a greater discriminative power for the modeling task being considered.

One of the possible approaches to the development of a reduced set of derivative variables that contain most of the useful information present in the original data is the use of an automated method that searches for some combination of the original variables. The most used of such methods is "Principal Component Analysis" [4]. This method develops orthogonal linear combinations of the original variables and ranks them on the basis of their ability to "explain" the target variable variance. The main problem with this approach is that the original variables are replaced by the most significant linear combinations and so, the data mining algorithms will no longer be able to search for non-linear combinations of the original variables. This approach

is therefore particularly appropriate for "data reduction" when only linear prediction methods will be used, and on problems related to systems known at start to be basically linear (clearly not the case of stock exchange time series [14], [31]).

Another approach to feature construction consists in the "manual" development of a reduced set of derivative variables using domain knowledge. This approach is especially efficient in domains where there is an available (but incomplete, thus not allowing a deterministic prediction) body of domain theory relating the data to the system mechanics and behavior. In stock exchange prediction there is no sure domain knowledge and, considering only historical stock exchange information as base data, the available domain theories (related to "technical analysis" of stocks [7], [18]) are particularly uncertain [10]. However, the large number of known technical analysis indicators,[1] seems a good starting point to build derivative features. In effect, the highly uncertain applicability of these indicators to individual stocks, markets, and time frames, limits them as final global theories, but does not prevent their usefulness to construct input features for machine learning algorithms, since most of these algorithms are able of filtering out the features that prove to be least useful over the training data, and to adopt only the features that better fit the data.

2 Data Pre-processing in Financial Time Series Prediction

A review of the published works on financial time series prediction shows that, in spite of the limitations mentioned in Section 1.2, most works use either the basic version of temporal embed or very limited transformations of this technique.

As examples of the use of direct temporal embed, we can mention the use of a direct univariate embed of dimension two to predict particularly strong daily stock exchange quotes variations [20], or the use of a multivariate basic embed of daily stock quotes information as input data to a genetic algorithm used to conduct a direct search for trading criteria in [30].

As examples of basic transformations of temporal embed we can refer the approach used by [13], who employs the difference of the logarithms of the last values of several time series to predict the stock quotes of several Japanese companies, or the work of [15] where logarithmic transformations of the differences of the last two and three known values of exchange rate times series are used to predict the variation direction of those time series.

Interesting examples of more ambitious adaptations of the basic temporal embed can be found in two entries of the important time series prediction competition carried out in Santa Fe in 1992 [25]: Mozer tests several transformations involving different weights for the embedded values, in an univariate context related to the prediction of exchange rates between the US dollar and the Swiss franc [17] and, for the prediction of the same time series, Zang and Hutchinson try a univariate embed using non-consecutive past values for different prediction time frames [31].

Although not so frequent, it is also possible to find some published works using more sophisticated variables derived from the base data. An early effort involving this approach can be found in [26], where a neural network with 61 inputs is used for the

[1] Descriptions can be found in a variety of sources, like, for instance, http://traders.com, or)

prediction of the next day value of the US dollar / German mark exchange rate. Forty five of the 61 variables used in this work correspond to an embed of known values of the times series that is being modeled, while the other 16 variables are developed from multivariate data that the authors believe to include relevant information for the prediction of the time series. Another work, involving more sophisticated derivative variables and more complete multivariate base data is described in [27]. In this work, the objective is again the prediction of the US dollar / German mark exchange rate, and the authors use 69 input variables for the neural network that produces the predictions, but none of those variables are the result of a direct embed of the time series being predicted. In fact, 12 of the 69 variables are "built" based on past values of the time series, using typical "technical analysis" transformations, and the other 57 variables reflect multivariate fundamental information exterior to the time series being predicted (using data associated to exchange rates between other currencies, interest rates, etc.). Obviously, in those two works, the large number of variables "feed" to the machine learning algorithms will tend to produce overfitting problems and, in effect, both works include as main objectives the presentation of new techniques to reduce overfitting problems in neural network algorithms.

3 A System for Stock Quotes Prediction

When the goal is the short term (up to a week) prediction of stock quotes, the relevance of fundamental information (both micro or macro-economic) tends to be small. In effect, even if this kind of information proves useful to predict a part of the long term variability of the stocks, the proportion of that ability with direct reflection on the variance of the next few days would be very small [11], [18]. Thus, it seems reasonable to believe that most of the short term variability of stock values is due to fluctuations related to the behavior of the investors, which technical analysis claims that can be inferred from past variations of the stock quotes and volumes. However, if an approach based on derived variables built with the help of domain knowledge from past values of stock exchange time series is to be tried, there are still important problems to address. These problems are related to the uncertain nature of the existing technical analysis "indicators", and to the vast number of existing indicators (the direct use of a large number of derivative variables of this kind could led to overfitting problems similar to those related to the use of a large direct embed [10]).

An approach based on derived variables would benefit from a practical way of representing domain knowledge (in the form of technical analysis indicators) and also from an efficient way of generating variables from that representation. Also, given the large quantity of possible derived "technical variables" it may be advantageous to perform some kind of automatic filtering of the less relevant features (in line with what would be needed if a large embed of the time series of interest was used as input).

To try to circumvent the problems associated with this approach, we have developed an integrated prediction system that includes a knowledge representation language that allows the direct description of most technical analysis indicators using pre-defined language elements, and generates automatically, from the raw data, the

features described by the user. The generated features are then automatically discretized (in our implementation, the original values of the features are discretized into five integer values, ranging from 1 to 5). The designed prediction system could use any machine learning algorithm with efficient behavior over noisy data. We chose to test this framework with a lazy learning k-nearest neighbor algorithm [5], and with a regression version of the PA3 rule induction algorithm [2]. The lazy learning algorithm uses a distance metric that involves a ranking of feature importance, while the rule induction algorithm does not need such ranking, but due to the domain characteristics mentioned before, also gains from a reduction of the number of features. As such, our system includes a feature ranking and selection step. The complete system architecture is shown in Figure 1.

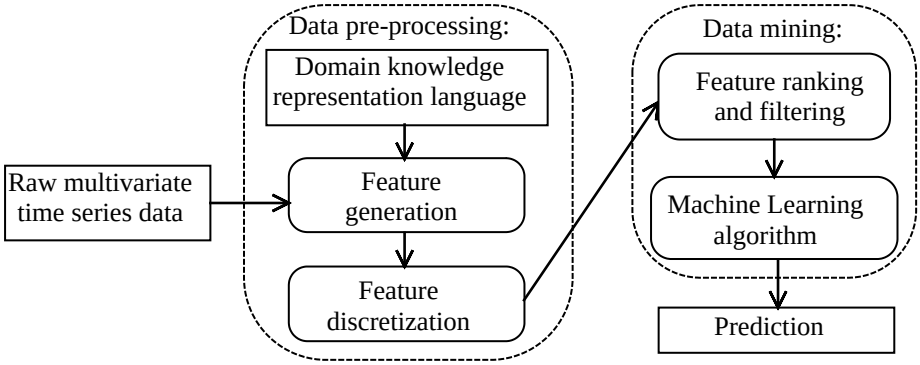

Fig. 1. The complete time-series prediction system

The domain knowledge representation language allows a domain expert to define technical analysis indicators using a simple but flexible representation. As examples, consider the following instructions of this language:

percent(clo(i),clo(i+1)); i=0..3
ratio(moa(vol;3;0),moa(vol;15;0))

The first of these instructions describes 4 features, each corresponding to the percent variation between the last four daily closing values of a stock and the respective previous closing value. The second instruction describes a single feature corresponding to the ratio between the moving average of the last 3 daily trading volumes of the stock and the moving average of the last 15 daily trading volumes[2].

The features defined with the language are then automatically generated from the raw data and appended with the respective target value for each generated example, to form a set of examples in the tabular "normal form".

The resulting features can have very different discrete or continuous values. Discrete features can have ordered integer values (for instance, consider a feature that counts the number of times the closing value is higher than the opening value during

[2] The third parameter in the *moa* constructor specifies the number of days between the last value used in the moving average and the present reference example. This way, the expression *ratio(moa(vol;3;0),moa(vol;3;1))* relates two 3-day moving averages of the volumes (the first using the last 3 values and the second using the first 3 of the last 4 values).

the last 20 trading days), but can also have non-ordered values (e.g. a feature that represents the weekday of the last trading session). Continuous features can also have different ranges of values (consider, for example, a feature representing the difference between the closing and opening value of the last trading day, which can have negative or positive values, and a feature that represents the percent variation between the minimum and maximum values of the last trading day, that can only have positive values). Given these different characteristics of the features, in order to simplify the work of the data mining algorithms (many symbolic machine learning algorithms require discretized data), a feature discretization module is run before the set of examples is "feed" to the data mining algorithms. The feature discretization module we implemented consists of discretizes each feature into 5 values using a traditional approach [9] that is known to behave well over noisy data. This discretization technique works individually over each feature. It starts by analyzing the examples of the training set and divides the range of values of each feature into five intervals with the same number of values in each interval. Then it attributes a discrete value (an integer value ranging from 1 to 5) to the feature instances associated to each of those value intervals, using the same values to classify the feature over the training and testing sets of examples. This simple "non-supervised" (it does not look to the class values of the examples to decide the discretization intervals for each feature) approach to feature discretization produced robust results over our data, and we did not notice any significant accuracy differences with algorithms able to work with both the discretized and the non-discretized original feature values.

After the data preprocessing, our prediction system starts the core data mining step with a feature ranking and selection module. The ranking of each feature is useful for instance-based machine learning algorithms that integrate distance metrics that weight differently the features. The feature selection procedure aims to reduce the overfitting problems related to the difficult domain characteristics already described. In effect, some algorithms that use representation languages with larger descriptive power are able to fit almost perfectly the hypersurface defined by the training data (which is the case of our rule induction PA algorithm). These methods would have a tendency to find existing but non-meaningful statistic fluctuations on the training data if given enough irrelevant features.

To prevent this kind of overfitting, our feature ranking and filtering module combines the feature evaluations given by two very simple statistical relevance measures (the information gain and the Pearson's *r* correlation between each feature and the target values) that look to each feature individually, and by a more powerful feature relevance metric that evaluates the features in the context of the other features [12]. These measures are combined by direct averaging of the three independent feature rankings obtained from each measure, thus establishing the overall feature ranking. To select a number of features to be used by the machine learning algorithms, we simply retain the highest ranking features up to a certain number[3].

The lazy learning k-nearest neighbor algorithm we implemented and integrated within our prediction system is inspired in Bontempi's work [5]. It uses an orthogonal "Manhattan" distance metric that weights differently the features (which must be ranked by relevance before being used by the algorithm) and a kernel function that

---

[3] In this work we have used 10 features in all tests.

gives (linearly) greater weights to the training examples found to be nearest to the test example being predicted. After several tests over stock exchange time series, we opted to use a fixed neighborhood of 150 cases. This considerable number of neighbors seems to result in a good balance between bias and variance and has an important effect in the reduction of overfitting problems typically associated to the very noisy examples sets expected in stock exchange time series prediction.

The rule induction algorithm we also tried in our prediction system is a regression variant of a general-propose sequential-cover algorithm called PA3, which handles two-class problems. This algorithm was developed to be efficient over noisy data, and performed well when compared with other rule induction algorithms over several noisy data sets (including stock exchange time series prediction) [2].

PA3 main features include a rule evaluation function that integrates an explicit account for rule accuracy, coverage and simplicity, and a rule generalization step that is performed immediately after each rule is developed in an initial general-to-specific development phase.

PA3 induces an ordered list of "if…then…" rules. Each rule has the form "if <complex> then predict <class>", where <complex> is a conjunct of feature tests, the "selectors". In PA3, each selector implies testing a feature to see if its value is included in a specified range of values. Thus, each selector indicates the feature to be tested and the upper and lower limits of the range of values it has to be tested against. The postcondition of a PA3 rule is a single Boolean value that specifies the class that the rule predicts for the cases that satisfy the selectors. To produce a regression (numeric) prediction instead of a class prediction, we used the very simple approach of assigning as postcondition of each rule the mean value of the training examples covered by the rule.

Those two algorithms where chosen to be integrated in our prediction system not only because they tend to be efficient over the noisy stock exchange time series data, but also because they use totally different algorithmic approaches. This is an important issue since we wanted to test the validity of our different data pre-processing methods. In effect, it is conceivable that some data preparation approaches lead to examples sets better suited for specific machine learning algorithms, but less efficient when used by others.

4 Experimental Testing

The main goal of this paper is to show that the use of domain knowledge to generate derivative features from the raw data leads to better predictive accuracy results than the traditional embed approach, within financial time series prediction problems. To test this hypothesis, we have carried out a set of three comparative experiments.

In our experiments we have used time series data concerning five of the more actively traded companies listed in the Portuguese BVLP stock exchange: "BCP", "Brisa", "Cimpor", "EDP" and "PT". The base data consists of 855 daily records for each of the five listed companies (from 25 November 1997 to 11 May 2001). Each of those daily records includes 7 base variables: The date of the day, the closing value

for the BVLP30 stock index, and, for the involved stock, the volume of traded stocks and the opening, maximum, minimum and closing values of the stock.

Regarding the methodology used to compare the different alternatives considered in this work we have used the following strategy. The available data for each company was split in four sections. The first 50 records were kept aside for the construction of the first processed example (thus allowing the use of embed dimensions of up to 50). The following 400 records were used to generate 400 training examples according to the three strategies that will be compared in this paper. The following 400 records were used to construct 400 testing examples. The last 5 basis records were kept aside to allow predictions up to 5 days ahead.

With respect to the method used to obtain predictions for the 400 test cases we have used the following strategy. Initially all learning algorithms were given access to the first 400 examples. With this set of examples a prediction is made for the first test case. After this prediction is obtained, this test example is added to the set of training cases leading to a new training set, now with 401 examples. This iterative train+test process continues until a prediction is obtained for all 400 test examples. This means that for instance the prediction for the last test example is obtained with a model that was learned using 799 training cases (the original 400 training examples, plus the first 399 test examples).

Given the goal of our comparative experiments, we have used a prediction horizon of one single day. Regarding the target variable we have used as the times series to predict we have chosen an average of the following day quotes of each stock. This average consists of the mean of the opening, maximum, minimum and closing values of the involved stock during the following day. We called this average the "reference value" of the stock. More precisely, we predict the percent change between the last known reference value and the following day reference value,

$$y(t) = \frac{Ref(t) - Ref(t-1)}{Ref(t-1)} \tag{1}$$

where $Ref(t) = \frac{Close(t) + Max(t) + Min(t) + Open(t)}{4}$.

The reason for the use of this "reference value" (instead of using, for example, the closing value of each day) is related to the nearly stochastic nature of these time series. In effect, this kind of time series behaves as if it had an important component of added white noise. Thus, every single observation of the time series is affected by a random amount of noise, which tends to "drown" the relatively small variations of the stock values attributable to the system "real" behavior ([8], [10]). The use of this reference value results in two main advantages when compared with the use of a single daily point value of the stock quotes (for instance the closing value). The first, and most important, is the reduction in the proportion of the variance that results from (unpredictable) random noise with relation to the variance resulting from the (hopefully predictable) fundamental subjacent system behavior. The second advantage is related to the usefulness of the predicted values for trading. To illustrate this latter advantage, let us suppose that we predict a 1% raise in the reference value for the next day. Based on that prediction, we could place a sell order with a (minimum) sell value equal to the predicted reference value. In this situation, we can

attain the sell value during the next day even if the predicted raise for the reference value falls short, since, in a typical day (one with variability in the stock value) the maximum value for the day must be higher than the reference value. This situation occurs if we base any (buy or sell) operation on the predicted reference value, since this value can be considered an average for the next day and, as such, the typical variability of the stock value "around" that average increases the probability of attaining the defined operation value, even if the prediction does not fully correspond to the actual reference value of that day.

We have used our prediction system to generate and compare several alternative ways of generating sets of examples for each of the 5 stocks considered in this paper. More specifically, we tested three data preprocessing approaches. In the first approach we have used a direct embed of the time series we are predicting. Namely, we developed 25 features that represent the percent variations between the last 25 consecutive pairs of values of the reference values time series (this implies using information associated to the last known 26 values of each reference values time series). The second approach compared in our study uses a similar direct embed strategy but now considering several of the time series available for each stock. In this alternative we have also used 25 features, which include the 5 last known percent changes of the reference values time series, and the 4 last known percent changes of the volume, opening, maximum, minimum and closing values time series[4]. Finally, in the third alternative data preprocessing approach, we developed 25 features based on domain knowledge. This knowledge was used both to generate and to select the final set of features. Since there are literally hundreds of technical indicators applicable to our base data[5], the development of this type of features requires some restricting choices. We have used a process to choose the final set of features where some of them were fixed from the start, while others went through an automatic feature selection process. Among the former we included the first 4 terms of the simple embed used in the first data preprocessing approach. Regarding the automatic selection of the other features, we used the knowledge representation language to describe and generate a set of features that seemed relevant, and then used a feature ranking process[6] to discard those with lowest evaluations. In order to increase the statistical significance of this selection process we have used the combined 2000 training examples (400 examples $\times$ 5 stocks).

It is interesting to observe which were the features that resulted from the process described above. It was rather surprising to find out that most of the retained features where not the "full" technical analysis indicators but "constructive elements" of those indicators. An example is the retention of simple ratios of moving averages, instead of

---

[4] The expressive power of our domain knowledge representation language allows very simple representations of these two first sets of features. The first is represented by the single expression *percent(ref(i),ref(i+1))* with *i=0..24*. The second is represented by 6 very similar expressions, one for each of the 6 base data time series involved.

[5] Many of them involving a choice of parameters and thus greatly increasing the number of possible variants.

[6] This process is similar to the one used in the data mining component of our system described in Section 3.

more complex variants of the Moving Average Convergence Divergence (MACD) indicator that where also considered as candidate features.

The final set of 25 chosen features can be grouped in the following families of features:

- One feature based on the date time series, stating the weekday of the example.
- Six features based on the reference values time series, four of them embeds of the differences, the other two relating moving averages of different lengths.
- Three features based on direct embeds of differences of the BVL30 time series.
- Two features based on the differential changes of the reference values and BVL30 time series, during two different time frames.
- Three features based on the volume of trade time series, one of them an embed of the differences, the other two relating moving averages of different lengths.
- Five features based on different relations of the opening, maximum, minimum, closing and reference values of the last known day.
- Five features also based on relations of the opening, maximum, minimum, closing and reference values, but using moving averages of different lengths.

Overall, among the 25 developed features, the oldest historical values used are reference values up to 20 days old and traded volumes up to 15 days old.

In order to evaluate the prediction performance of the three alternative ways of pre-processing the raw stock data, we used two very common measures [10], chosen for their relevance for trading criteria development. The first is the percent accuracy of the binary (rise or fall) predictions for the next day, defined as,

$$\%Acc = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} Loss(y_i, \hat{y}_i) \tag{2}$$

where, N_{test} is the number of test cases;
y_i is the truth target value of test case i;
$\hat{y}_i$ is the predicted value for that test case;

and

$$Loss(y, \hat{y}) = \begin{cases} 0 & \text{if } y \geq 0 \wedge \hat{y} \geq 0 \\ 0 & \text{if } y < 0 \wedge \hat{y} < 0 \\ 1 & \text{otherwise} \end{cases}$$

The second evaluation measure we have used is the average value of the return on the test examples, taken as positive when the prediction sign is correct and as negative when it is wrong[7]. We called this measure the Average Return on the Predicted Examples (ARPE), which can be defined as,

$$ARPE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left| Ref_{i+1} - Ref_i \right| sign(y_i \hat{y}_i) \tag{3}$$

where, N_{test} is the number of test cases;
y_i is the truth target value of test case i;
$\hat{y}_i$ is the predicted value for that test case;

---

[7] Notice that the target values of each example are the percent changes between the last known reference value and the reference value of the next day.

Ref_i is the reference value of test case i.

Tables 1 and 2 show the accuracy and ARPE results obtained in our experiments using the Lazy3 and PA6 algorithms over the three described data sets.

These tables also include the results obtained with two traditional benchmarks: The Naïve Prediction of Returns (NPR) and the buy-and-hold strategy[8]. The NPR merely predicts that the next change of value for the time series will be the same as the last known change of value. The results for this benchmark are presented in Table 1 as the percent accuracy of always predicting that *Ref(t+1)-Ref(t)* will have the same value as *Ref(t)-Ref(t-1)* and, as such, are directly comparable with the other values in Table 1. The results for the buy-and-hold benchmark are show in Table 2 as the average gain of the buy-and-hold strategy[9] over the 400 trading days involved in our tests, so that they are directly comparable with the other ARPE measure values presented.

	Lazy3 algorithm			PA6 algorithm			NRP
	Data Set 1	Data Set 2	Data Set 3	Data Set 1	Data Set 2	Data Set 3	
BCP	57.50	62.75	69.00	56.50	56.50	62.75	65.50
Brisa	54.75	62.00	66.50	57.75	63.25	63.75	56.75
Cimpor	59.75	63.00	69.25	56.00	57.00	64.50	58.50
EDP	58.50	65.75	72.25	58.00	67.75	72.50	59.00
PT	57.50	61.75	68.50	57.50	63.50	67.75	57.50
Average	57.60	63.05	69.10	57.15	61.60	66.25	59.45

Table 1. Percent accuracy over the test examples

	Lazy3 algorithm			PA6 algorithm			Buy-and-hold
	Data Set 1	Data Set 2	Data Set 3	Data Set 1	Data Set 2	Data Set 3	
BCP	0.134	0.253	0.286	0.193	0.215	0.215	0.002
Brisa	0.176	0.286	0.394	0.228	0.312	0.333	0.099
Cimpor	0.266	0.404	0.498	0.192	0.386	0.447	0.141
EDP	0.263	0.515	0.578	0.302	0.531	0.591	0.010
PT	0.494	0.762	1.045	0.482	0.922	1.007	0.113
Average	0.267	0.444	0.560	0.279	0.473	0.519	0.073

Table 2. Average Return on the Predicted Examples (ARPE)

The results shown on these tables seem to validate our hypothesis concerning the use of domain knowledge to generate the training examples. In effect, considering the results over the 5 stocks, we can observe that the models obtained with data set 3 consistently achieve better accuracy results than the others, independently of the algorithm used to obtain the model. When comparing the results obtained with data sets 1 and 2, we also observe worse results with the simple embed (with a single exception in the BCP stock using the PA6 algorithm). If we consider the ARPE

[8] An extended discussion of these benchmarks can be found in [10] and [21].

[9] This strategy consists of buying at the price of the last known training case, Ref_{50+400}, and selling at the price of the last known test case value $Ref_{50+400+400}$.

evaluation measure the conclusions are similar. An interesting point to notice is the fact that all the algorithm / data sets combinations produce positive results (accuracy results greater than 50% and ARPE results greater than 0%).

Analyzing the benchmark results in isolation, we notice that the NPR accuracy results are considerably above the 50% average values that could be expected. This is due to the high auto-correlation (at lag 1) of the five time series in analysis (which also helps to explain the prediction ability of data set 1, as used in our system)[10]. Comparing our results with those achieved by the benchmarks, we notice that all the algorithm / data sets combinations achieve considerably better ARPE values than the buy-and-hold benchmark. On the other hand, the NPR accuracy results are globally higher than the results of the simple embed used in data set 1, although they are worse than the results of data set 2 and, particularly, of data set 3.

In order to assert the statistical significance of results of these experiments, we have carried out one-sided paired *t* tests over the 400 test examples of each stock, using the accuracy results previously obtained. The accuracy differences between models obtained with data set 1 and data set 3 were observed to be highly significant. In effect, with the Lazy 3 algorithm all the differences are statistically significant with a 99.5% confidence level, except for BCP where the level was 97.5%. Regarding the PA6 algorithm, the improvements obtained with data set 3 are somewhat less marked, but still significant with 99.5% confidence for Cimpor, EDP, and PT, with 97.5% confidence for BCP, and with 95% for Brisa.

5 Conclusions

This paper described an approach to financial time series prediction whose main distinctive feature is the use of domain knowledge to generate the training cases that form the basis of model construction. With this purpose we have developed a domain knowledge representation language that allows the time series expert to easily express his knowledge. The description obtained with the help of this language is then used by our prediction system to generate a training sample from the raw time series data.

We have compared this knowledge intensive approach with the traditional method of embedding the last time series values. With this purpose we have carried out a series of comparative experiments using time series data from five actively traded Portuguese companies. The results of our experiments with theses stocks provide strong empirical evidence that a data preprocessing approach based on a direct embed of the time series has large limitations, and that the use of features constructed from the available raw data with the help of domain knowledge is advantageous.

Regarding futures developments of this research, we intend to extend the experimental evaluation to prediction horizons larger than the next day, as well as further refine the important feature selection phase.

---

[10] An analysis of those time series showed no significant auto-correlation at other lags, limiting the potential effectiveness of more powerful prediction techniques based in auto-correlation.

References

1. Abu-Mostafa, Y., LeBaron, B., Lo, A. and Weigend, A. (eds.): Proceedings of the Sixth International Conference on Computational Finance, CF99. MIT Press (1999)
2. Almeida, P. and Bento, C.: Sequential Cover Rule Induction with PA3. Proceedings of the 10th International Conference on Computing and Information (ICCI'2000), Kuwait. Springer-Verlag (2001)
3. Bellman, R.; Adaptative Control Processes: A Guided Tour. Princeton University Press, (1961)
4. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
5. Bontempi, G.: Local Learning Techniques for Modeling, Prediction and Control. Ph.D. Dissertation, Université Libre de Bruxelles, Belgium (1999)
6. Box, G., and Jenkins, G.: Time Series Analysis, Forecasting and Control. Holden-Day (1976)
7. Demark, T.: The New Science of Technical Analysis. John Wiley & Sons (1994)
8. Fama, E.: Efficient Capital Markets: A review of Theory and Empirical Work. Journal of Finance, 25 (1970) 383-417
9. Hall, M.: Correlation-Based Feature Selection for Machine Learning. Ph.D. Dissertation, Department of Computer Science, University of Waikato (1999)
10. Hellstrom, T.: Data Snooping in the Stock Market. Theory of Stochastic Process 5(21) (1999)
11. Herbst, A.: Analyzing and Forecasting Futures Prices. John Wiley & Sons (1992)
12. Hong, S.: Use of Contextual Information for Feature Ranking and Discretization. IEEE Transactions on Knowledge and Data Engineering, 9(5) (1997) 718-730
13. Hutchinson, J.: A Radial Basis Function Approach to Financial Time Series Analysis. Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (1994)
14. Kustrin, D.: Forecasting Financial Time series with Correlation Matrix Memories for Tactical Asset Allocation. Ph.D. Dissertation, Department of Computer Science, University of York, UK (1998)
15. Lawrence, S., Tsoi, A. and Giles C.: Noisy Time Series Prediction using Symbolic Representation and Recurrent Neural Network Grammatical Inference. Technical Report UMIACS-TR-96-27 and CS-TR-3625, Institute for Advanced Computer Studies, University of Maryland, MD (1996)
16. Michalski, R.: A Theory and Methodology of Inductive Learning. In Michalski, R., Carbonell, J., and Mitchell, T., (eds): Machine Learning: An Artificial Intelligence Approach, Vol. 1. Morgan Kaufmann (1983)
17. Mozer, M.: Neural Net Architectures for Temporal Sequence Processing. In: Weigend, A. and Gershenfeld, N. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)
18. Murphy, J.: Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. Prentice Hall (1999)
19. Murthy, S., Kasif, S. and Salzberg, S.: A System for Induction of Oblique Decision Trees. Journal of Artificial Intelligence Research, 2 (1994) 1-32
20. Povinelli, R.: Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events. Ph.D. Dissertation, Marquette University, Milwaukee, Wisconsin (1999)

21. Refenes, A.: Testing Strategies and Metrics. In Refenes, A. (ed.): Neural Networks in the Capital Markets. John Wiley & Sons (1995)
22. Sauer, T., Yorke, J. and Casdagli, M.: Embedology. Journal of Statistical Physics 65 (1991) 579-616
23. Scott, D.; Multivariate Density Estimation. John Wiley & Sons (1992)
24. Takens, F.: Detecting Strange Attractors in Turbulence. In Rand, D. and Young, L. (eds.), Lecture Notes in Mathematics, Vol. 898. Springer (1981) 366-381
25. Weigend, A. and Gershenfeld, N. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)
26. Weigend, A., Huberman, B. and Rumelhart, D.: Predicting Sunspots and Exchange Rates with Connectionist Networks. In Casdagli, M. and Eubank, S. (eds.): Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity. Addison-Wesley (1992)
27. Weigend, A., Zimmermann, H. and Neuneier, R.: Clearning. In Refenes, P., Abu-Mostafa, Y., Moody, J. and Weigend, A. (eds.): Neural Networks in Financial Engineering (Proceedings of NNCM'95). World Scientific (1996)
28. Weiss, S. and Indurkhya, N.: Predictive Data Mining: A Practical Guide. Morgan Kaufmann (1998)
29. Yule, G.: On a Method of Investigating Periodicities in Disturbed Series with Special Reference to Wolfer's Sunspot Numbers. Phil. Trans. Royal Society, Series A, 226 (1927)
30. Yuret, D. and Maza, M.: A Genetic Algorithm System for Predicting de OEX. Technical Analysis of Stocks and Commodities, 12(6) (1994) 255-259
31. Zang, X. and Hutchinson, J.: Simple Architectures on Fast Machines: Practical Issues in Nonlinear Time Series Prediction. In Weigend, A. and Gershenfeld, N. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)