Applying Neural Networks for Concept Drift Detection in Financial Markets

Bruno Silva [1] and **Nuno Marques** [2] and **Gisele Panosso** [3]

Abstract. Traditional stock market analysis is based on the assumption of a stationary market behavior. The recent financial crisis was an example of the inappropriateness of such assumption, namely by detecting the presence of much higher variations than what would normally be expected by traditional models. Data stream methods present an alternative for modeling the vast amounts of data arriving each day to a financial analyst. This paper discusses the use of a framework based on an artificial neural network that continuously monitors itself and allows the implementation on a multivariate financial non-stationary model of market behavior. An initial study is performed over ten years of the Dow Jones Industrial Average index (DJI), and shows empirical evidence of concept drift in the multivariate financial statistics used to describe the index data stream.

1 INTRODUCTION

Data streams are generated naturally within several domains. Network monitoring, web mining, telecommunications data management, stock-market analysis and sensor data processing are applications that have vast amounts of data arriving continuously. In such applications, the process may not be strictly stationary, i.e., the target concept may change over time. *Concept drift* means that the concept about which data is being collected may shift from time to time, each time after some minimum permanence [6].

In this paper we address the detection and analysis of concept drift in financial markets by employing a methodology based on Artificial Neural Networks (ANN). ANN are a set of biologically inspired algorithms and well-established data mining methods popular for technical market analysis and price predictions. We are currently undergoing a wider research on using ANN in Ubiquitous Data Mining. This work, in essence, is a real-world application of a mechanism to detect concept drift while processing data streams. The motivation for this approach in the financial field can be easily explained. Mathematical finance has made wide use of normal distributions in stock market analysis to maximize return rates, i.e., they assume stationary distributions, which are easier to understand and work well most of the times. However, this traditional approach neglects big heavy-tails, i.e.,huge asset losses, in the distributions and their potential risk evaluation [11, 12]. This is where the detection of drifting from this normal behavior is of critical importance to reduce investment risk in the presence of non-normal distribution of market events.

[1] DSI/ESTSetúbal, Instituto Politécnico de Setúbal, Portugal, email: bruno.silva@estsetubal.ips.pt
[2] CITI and Departamento de Informática, FCT, Universidade Nova de Lisboa, Portugal, email: nmm@fct.unl.pt
[3] ISEGI, Instituto Superior de Estatística e Gestão de Informação, Universidade Nova de Lisboa, Portugal, email: m2010147@isegi.unl.pt

The main contributions of this work are: (i) a drift detection method based on the output of *Adaptive Resonance Theory* (ART) networks [7] which produce *aggregations* (or *data synopsis* in some literature) of d-dimensional data streams. These fast aggregations compress a, possibly, high-rate stream while maintaining the intrinsic relations within the data. A fixed sequence of consecutive aggregations is then analyzed to infer concept drift in the underlying distribution – Section 2 ; (ii) an application of the previous scheme to the stock market, namely to the Dow Jones Industrial index (DJI), using a stream of with a chosen set of statistical and technical indicators. The detection of concept drift is performed over an incoming stream of these observations –Section 3.

These contributions adhere to the impositions of data stream models in [8], namely: the data points can only be accessed in the order in which they arrive; random access to data is not allowed; memory is assumed to be small relatively to the number of data points, thus only allowing a limited amount information to be stored. Therefore, all of the additional indicators are computed using sliding windows, thus only needing a small subset of data kept in memory. This is also true for the number of aggregations needed to compute the concept drift.

At the end of the paper, Section 4, discussion of the results are made together with final conclusions.

2 METHODOLOGY

The presented methodology for drift detection comprises two modules. The first module uses an ART network that receives the incoming stream and produces aggregations, or data synopsis, compressing the data and retaining the intrinsic relationships within the distribution (Section 2.1). This module feeds a second module that takes a fixed set of these aggregations and through simple computations produces an output that can be used to detect concept drift.

2.1 Online Data Aggregation

One should point out that algorithms performing on data streams are expected to produce "only" approximated models [6], since the data cannot be revisited to refine the generated models. The aggregation module is responsible for the online summarization of the incoming stream and processes the stream in blocks of size S. For each S observations q representative prototypes of data are created, where $q \ll S$. This can be related to an incremental clustering process that is performed by an ART network. Each prototype is included in a tuple that stores other relevant information, such as the number of observations described by a particular prototype and the point in time that a particular prototype was last updated. These data structures were popularized in [1] and called *micro-clusters*.

Hence, we create q "weighted" prototypes of data stored in tuples $Q = \{\mathcal{M}_1, ..., \mathcal{M}_j, ..., \mathcal{M}_q\}$, each containing: a prototype of data P_j; the number of inputs patterns N_j assigned to that prototype and a *timestamp* T_j that contains the point in time that prototype was last accessed, hence $\mathcal{M}_j = \{P_j, N_j, T_j\}$. The prototype together with the number of inputs assigned to it (*weighting*) is important to preserve the input space density if one is interested in creating offline models of the distribution. The timestamp allows the creation of models from specific intervals in time.

ART [7] is a family of neural networks that develop stable recognition categories (clusters) by self-organization in response to arbitrary sequences of input patterns. Its fast commitment mechanism and capability of learning at moderate speed guarantees a high efficiency. The common algorithm used for clustering in any kind of ART network is closely related to the *k*-means algorithm. Both use single prototypes to internally represent and dynamically adapt clusters. The *k*-means algorithm clusters a given set of input patterns into *k* groups. The parameter *k* thus specifies the coarseness of the partition. In contrast, ART uses a minimum required similarity between patterns that are grouped within one cluster. The resulting number k of clusters then depends on the distances (in terms of the applied metric) between all input patterns, presented to the network during training. This similarity parameter is called *vigilance* ρ. *K*-means is a popular algorithm in clustering data streams, e.g., [4], but suffers from the problem that the initial *k* clusters have to be set either randomly or through other methods. This has a strong impact on the quality of the clustering process. On the other hand, ART networks do not suffer from this problem.

More formally, a data stream is a sequence of data items (observations) $x_1, ..., x_i, ..., x_n$ such that the items are read once in increasing order of the indexes i. If each observation contains a set of d-dimensional features, then a data stream is a sequence of $X_1^d, ..., X_i^d, ..., X_n^d$ vectors. We employ an ART2-A [3] network specially geared towards fast one-shot training, with an important modification given our goals: constrain the network on a maximum of q prototypes. It shares the basic processing of all ART networks, which is based on *competitive learning*. ART requires the same input pattern size for all patterns, i.e., the dimension d of the input space where the clusters regions shall be placed. Starting with an empty set of prototypes $P_1^d, ..., P_j^d, ..., P_q^d$ each input pattern X_i^d is compared to the j stored prototypes in a *search* stage, in a *winner-takes-all* fashion. If the degree of similarity between current input pattern and best fitting prototype W_J is at least as high as vigilance ρ, this prototype is chosen to represent the micro-cluster containing the input. Similarity between the input pattern i and a prototype j is given by Equation 1, where the distance is subtracted from one to get $S_{X_i,P_j} = 1$ if input and prototype are identical. The distance is normalized with the dimension d of an input vector. This keeps measurements of similarity independent of the number of features.

$$S_{X_i,P_j} = 1 - \sqrt{\frac{1}{d}\sum_{n=1}^{d}(X_i^n - P_j^n)^2} \tag{1}$$

The degree of similarity is limited to the range $[0, 1]$. If similarity between the input pattern and the best matching prototype does not fit into the vigilance interval $[\rho, 1]$, i.e., $S_{X_i,P_j} < \rho$, a new micro-cluster has to be created, where the current input is used as the prototype initialization. Otherwise, if one of the previously committed prototypes (micro-clusters) matches the input pattern well enough, it is adapted by shifting the prototype's values towards the values of the input by the update rule in Equation 2.

$$P_J^{(new)} = \eta \cdot X_i + (1 - \eta) \cdot P_J^{(old)} \tag{2}$$

The constant learning rate $\eta \in [0, 1]$ is chosen to prevent prototype P_J from moving too fast and therefore destabilizing the learning process. However, given our goals, i.e., to perform an adaptive vector quantization, we define η dynamically in such a way that the mean quantization error of inputs represented by a prototype is minimized. Equation 3 establishes the dynamic value of η, where N_J is the current number of assigned input patterns for prototype J. This way, it is expected that the prototypes converge to the mean of the assigned input patterns.

$$\eta = \frac{N_J}{N_J + 1} \tag{3}$$

This does not guarantee the convergence to local minimum, however, according to the adaptive vector quantization (AVQ) convergence theorem [2], AVQ can be viewed as a way to learn prototype vector patterns of real numbers; it can guarantee that average synaptic vectors converge to centroids exponentially quickly.

Another needed modification arises from the fact that ART networks, by design, form as much prototypes as needed based on the vigilance value. At the extremes, $\rho = 1$ causes each unique input to be encoded by a separate prototype, whereas $\rho = 0$ causes all inputs to be represented by a single prototype. Therefore, for decreasing values of ρ coarser prototypes are formed. However, to achieve exactly q prototypes solely on a manually tuned value of ρ is a very hard task, mainly due to the input space density, that can change over time, and is also different from application to application.

To overcome this, we make a modification to the ART2-A algorithm to impose a restriction on creating a maximum of q prototypes and dynamically adjusting the vigilance parameter. We start with $\rho = 1$ so that a new micro-cluster is assigned to each arriving input vector. After learning an input vector, a verification is made to check if $q = j + 1$, where j is the current number of stored micro-clusters. If this condition is met, then to keep only q we need to merge the *nearest pair* of micro-clusters. Let $T_{r,s} = min\{\|P_r - P_s\|^2 : r, s = 1, ..., q, r \neq s\}$ be the minimum Euclidean distance between prototypes stored in micro-clusters $\mathcal{M}_r$ and $\mathcal{M}_s$. We merge the two micro-clusters into one:

$$\mathcal{M}_{merge} = \{P_{merge}, N_r + N_s, max\{T_r, T_s\}\} \tag{4}$$

with the new prototype being a "weighted" average between the previous two:.

$$P_{merge} = \frac{N_r}{N_r + N_s}P_r + \frac{N_s}{N_r + N_s}P_s \tag{5}$$

With d-dimensional input vectors, Equation 1 defines a hypersphere around any stored prototype with radius $r = (1 - \rho) \cdot \sqrt{d}$. By solving this equation in respect to ρ, we update the vigilance parameter dynamically with Equation 6, hence $\rho^{(new)} < \rho^{(old)}$ and the radius, consequently, increases.

$$\rho^{(new)} = 1 - \frac{T_{r,s}}{\sqrt{d}} \tag{6}$$

Our experimental results show that this approach is effective in providing a summarization of the underlying distribution within the data streams. The inclusion of these results is out of the scope of this paper.

We must point out that the aggregation module produces more information that it is actually necessary for the concept drift detection,

namely the weighting of the prototypes and the timestamps. This module is an integrating part of a larger framework that also generates offline models of the incoming stream for specific points in time.

2.2 Detecting Concept Drift

Our method assumes that if the underlying distribution is stationary that the error-rate of the learning algorithm will decrease as the number of samples increases [5]. Hence, we compute the quantization error at each aggregation phase of the ART network and track the changes of these errors over time.

We use a queue $\mathcal{B}$ of b aggregation results, such that $\mathcal{B} = \{Q_l, Q_{l-1}, ..., Q_{l-b+1}\}$, where Q_l is the last aggregation obtained. For each Q_l that arrives, we compute the average Euclidean distance between each prototype P_i in Q_l and the closest one in $\mathcal{B}_{l-1} = \{Q_{l-1}, ..., Q_{l-b+1}\}$. Equation 7 formalizes this Average Quantization Error (AQE) computation for the l^{th} aggregation, where $\| \cdot \|^2$ is the Euclidean distance and q is the number of prototypes in Q_l by definition. This computes the error of the last aggregation in "quantifying" previous aggregations in a particular point in time.

$$AQE(l) = \frac{1}{q} \sum_{i=1}^{q} min(\quad \| P_i - P_j \|^2, \quad \forall P_j \in \mathcal{B}_{l-1} \quad) \quad (7)$$

By repeating this procedure over time, we obtain a series of errors that stabilizes and/or decreases when the underlying distribution is stationary and presents increases on this curve when the underlying distribution is changing, i.e., concept drift is occurring. This series of errors is the drift curve.

Larger values of b are used to detect *abrupt* changes in the underlying distribution, whereas to detect *gradual* concept drift a lower value should be adopted. We exemplify the automatic concept drift detection in this drift curve using a moving average in Section 3.2.

3 APPLICATION TO DOW JONES INDUSTRIAL

We present an application of the previous methodology to the stock market, namely to the Dow Jones Industrial index (DJI). Instead of using daily prices of several stocks that compose the DJI, our approach to this problem uses the DJI daily index values themselves and other computed statistical and technical indicators, which are explained in Section 3.1. We make extensive use of moving averages, as they reduce the short term volatility of time series and retain information from previous market events; another statistical indicator is the Hurst index [9], defined as a function to uncover changes in the direction of the trend of a set of values in time. We believe that these indicators, together with the index value, can provide a multi-variate insight to hidden and subtle changes in the normality of financial events and be used to assess the risk of investment at any point in time, thus lowering exposure to risk.

This application makes use of data gathered from the period comprised between the 1^{st} of January of 2001 to the 31^{st} of December of 2011, in a total of 2767 observations.

3.1 Variable Selection and Generated Data Stream

The data gathered was composed by a set of technical variables including different index values for one trading day like Open, Close, High and Low values. From these we chose the lowest daily price (PX LOW) because it provides better insight to the risk of a fall. Other available technical indicator was the trading Volume.

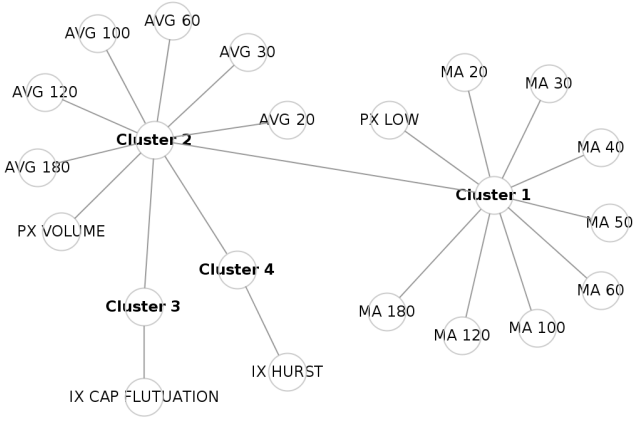

Figure 1. Hierarchical clustering of variables produced by VARCLUS.

In terms of statistical indicators, we initially considered a large number of them, like moving averages (MA) from 20 to 180 trading days, relative numbers, i.e., the DJI index value divided by moving averages (AVG), price fluctuation and Hurst index. However, it was important to reduce the number of variables because redundant variables can reduce the model efficiency. For this purpose we performed an analysis with the VARCLUS procedure (SAS/STAT).The VARCLUS procedure can be used as a variable-reduction method. The VARCLUS procedure divides a set of numeric variables into disjoint or hierarchical clusters through principal component analysis. All variables were treated as equally important. VARCLUS created an output was used by the TREE procedure to draw a tree diagram of hierarchical clusters (SAS/STAT®9.1 User's Guide p. 4797). The tree diagram is depicted in Figure 1. We can observe in the hierarchical clustering that the price variables and moving averages are correlated, so it was only chosen PX LOW of *Cluster 1*. In *Cluster 2* all variables were selected because, although they are correlated, they measure different characteristics. In the case of relative numbers different averages were selected because it is interesting to see the differences between the analysis of short, medium and long term. Finally in *Cluster 3* and *Cluster 4* just Hurst index and price fluctuation appeared, because they are not correlated with any other variable, so these variables were included in the final data set.

Hence, the complete set of features in the data stream is the following:

PX LOW: Minimum daily price;
PX VOLUME: Volume of daily business;
IX HURST: Hurst index computed for 30 days;
IX CAP FLUTUATION: PX LOW(t)/ PX LOW (t 1). This variable represents price fluctuation for one day interval;
AVG 20: PX LOW / 20-day moving average. This variable represents the relative number of current price divided by the 20-day Moving Average. This shows whether the current price is cheap, average value, expensive or really expensive. The same applies to the next indicators but within other time frames;

AVG 30: PX LOW / 30-day moving average;
AVG 60: PX LOW / 60-day moving average;
AVG 100: PX LOW / 100-day moving average;
AVG 120: PX LOW / 120-day moving average;
AVG 180: PX LOW / 180-day moving average;

The dataset is depicted in Figure 2, where the behavior of all variables can be seem. This data is our data stream. The stream comprises 10 features, e.g., a 10-dimensional stream.

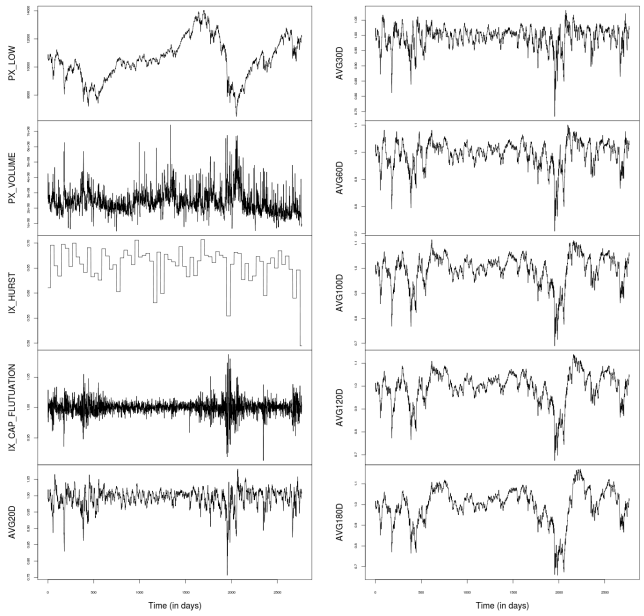

Figure 2. Variables of the data stream used in the presented application. It comprises technical and statistical indicators (description in text).

3.2 Concept Drift in the Dow Jones Industrial

The methodology presented in Section 2 was applied to the above data. It is converted into a data stream by taking the data input order as the order of the streaming. All features were previously normalized to the range $[0, 1]$ so they have equal importance in the Euclidean distances used to process them. The largest moving average indicator computed was over 180 days. Therefore, only after the 180^{th} observation can the stream be presented to the algorithm.

However, since we are dealing with financial time-series, it is important to retain the time dependency of the sequence of observations. Therefore, in this application, we use a sliding-window of 100 trading days, i.e., approximately a trimester of trading as input to each aggregation phase. Note that a year of trading has approximately 260 days. This means that the stream is processed in blocks of 100 observations that are kept in a queue. For each new observation that arrives the oldest in the queue is discarded and the new one added. The parameterization used was the following:

Block size: $S = 100$;
Number of micro-clusters: $q = 10$;
Concept drift buffer size: $b = 15$

The result of the procedure of Section 2.2 applied to the data stream is presented in Figure 3. Each point of the series corresponds to the error of the model for a particular trading day, thus providing possible indications of drifting. It can be seen an overall shape of a curve that indicates the drift over time. Since this drift is being computed for every trading day, the "noise" around the curve is considered normal since it is affected by the daily volatility of the index values.

To obtain a "clean" curve we apply a convolution filter along this drift series of the same size as b, i.e., 15 days. An alarm scheme is created through the generation of an empirical moving average of 60 days performed over the drift series. The cleaned drift curve and its moving average are depicted in Figure 4a).

We then compare the differences between the drift series and its moving average obtaining a line that oscillates around zero. We call this line the *drift trend*, shown in Figure 4b). Whenever the drift series has values lower than its moving average we are in a descending trend. This is reflected in the drift trend with values lower than zero. Whenever the moving average is crossed by the drift series it signals a shift in the trend and the drift trend crosses zero. This reasoning to detect trends is also very popular in financial technical analysis. In this context, the 60 trading days moving average reflects the intuitive notion of long-term "decreasing" or "increasing" trend of the drift.

All plots in Figure 4 are aligned in time for easy comparison. Figure 4c) shows the time series of PX LOW, i.e., the DJI index, that we compare to the detection of drift performed.

4 DISCUSSION AND CONCLUSIONS

Based on experiments we found that a tenth of prototypes relative to the number of observations are sufficient in most applications to represent them adequately, hence, $q = 10$. Usage of higher values of q did not improve the results with the additional problem of increased computational time. Additionally, since we are both interested in abrupt and gradual drift detection we used a moderate sized buffer of aggregations ($b = 15$) to compute the series of quantification errors. During our experiments we found that this value was appropriate for the established goals.

By inspecting Figure 4 and comparing the *drift trend* with the behavior of the DJI index we can make two important observations: (i)

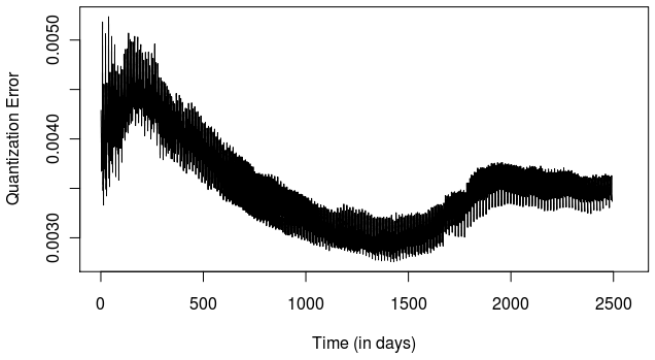

Figure 3. Concept drift series obtained through the methodology in Section 2.2 computed for each trading day.

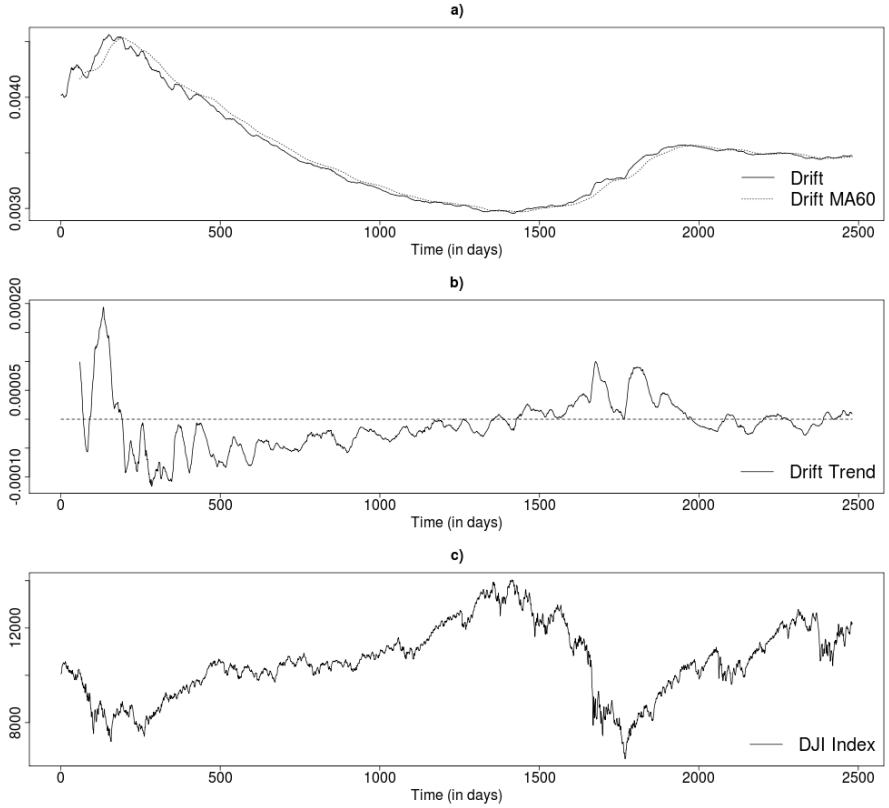

Figure 4. a) Cleaned drift curve and its moving average. b) The trend drift curve is used to automatically detect drifting. c) The DJI index time series (PX LOW variable).

the drift trend crossed zero before the market crash of 2008 (around day 1500). It appears that the concept that was being learned changed sometime before the crash occurred. (ii) it may be reasonable to assume that in periods of normality the long-term tendency of these indexes is upwards. One of such periods is after the recovery of the 2002 market crash, i.e., the dot-com bubble, until the other crash of 2008 (approximately between days 300 and 1300). During such period it is interesting to see that the drift trend was always below zero.

In the present work we have shown a methodology to detect concept drift in financial markets. We intend to apply this same methodology to intra-day trading as soon as it is possible, thus reinforcing the need to efficient processing of large volumes of data. The proposed methodology applied over a data stream comprised of carefully chosen technical and statistical indicators seems promising in detecting changes in markets events ahead of time that can reduce the exposure to risk.

The characterization of the drifts, i.e., trying to understand what is really changing in the markets through inspection of hidden changes in the indicators is reserved for future work. Work is under way in this subject and we are using Self-Organizing Maps [10] to produce different mappings of the variables for particular segments in time, namely ones where the market seems to exhibit a stable behavior and comparing with others where it does not. This segments are obtained by segmenting time with the concept drift detection. As another immediate future work we will apply this methodology to other indexes and perform the same study.

REFERENCES

[1] C.C. Aggarwal, J. Han, J. Wang, and P.S. Yu, 'A framework for clustering evolving data streams', in *Proceedings of the 29th International Conference on Very Large Databases*, volume 29, pp. 81–92. Morgan Kaufmann Publishers Inc., (2003).

[2] K. Bart, *Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence*, Prentice-Hall of India, 1997.

[3] G.A. Carpenter, S. Grossberg, and D.B. Rosen, 'Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition', *Neural networks*, **4**(4), 493–504, (1991).

[4] F. Farnstrom, J. Lewis, and C. Elkan, 'Scalability for clustering algorithms revisited', in *ACM SIGKDD Explorations Newsletter*, volume 2, pp. 51–57. ACM, (2000).

[5] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, 'Learning with drift detection', *Advances in Artificial Intelligence–SBIA 2004*, 66–112, (2004).

[6] Joao Gama, *Knowledge discovery from data streams*, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2010.

[7] S. Grossberg, 'Adaptive pattern classification and universal recording: II. Feedback, expectation, olfaction, illusions', *Biological Cybernetics*, **23**, (1976).

[8] Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan, 'External memory algorithms', chapter Computing on data streams, 107–118, American Mathematical Society, Boston, MA, USA, (1999).

[9] H.E. Hurst, RP Black, and YM Simaika, *Long-term storage: An experimental study*, Constable, 1965.

[10] T. Kohonen, 'Self-organized formation of topologically correct feature maps', *Biological cybernetics*, **43**(1), 59–69, (1982).

[11] B. Mandelbrot, R.L. Hudson, and E. Grunwald, 'The (mis) behaviour of markets', *The Mathematical Intelligencer*, **27**(3), 77–79, (2005).

[12] N.N. Taleb, 'Errors, robustness, and the fourth quadrant', *International Journal of Forecasting*, **25**(4), 744–759, (2009).