Volatility Trading via Temporal Pattern Recognition in Quantized Financial Time Series

Peter Tiňo*

Neural Computing Research Group

Aston University

Aston Triangle

Birmingham, B4 7ET

United Kingdom

p.tino@aston.ac.uk

Christian Schittenkopf **Georg Dorffner**

Austrian Research Institute for Artificial Intelligence

Schottengasse 3

A-1010 Vienna

Austria

chris,georg@ai.univie.ac.at

Running title: Volatility Trading via pattern recognition on quantized sequences

*Corresponding author

Abstract

We investigate the potential of the analysis of noisy non-stationary time series by quantizing it into streams of discrete symbols and applying finite-memory symbolic predictors. Careful quantization can reduce the noise in the time series to make model estimation more amenable. We apply the quantization strategy in a realistic setting involving financial forecasting and trading. In particular, using historical data, we simulate the trading of straddles on the financial indexes DAX and FTSE 100 on a daily basis, based on predictions of the daily volatility differences in the underlying indexes. We propose a parametric, data-driven quantization scheme which transforms temporal patterns in the series of daily volatility changes into grammatical and statistical patterns in the corresponding symbolic streams. As symbolic predictors operating on the quantized streams we use the classical fixed-order Markov models, variable memory length Markov models and a novel variation of fractal-based predictors introduced in its original form in [1]. The fractal-based predictors are designed to efficiently use deep memory. We compare the symbolic models with continuous techniques such as time-delay neural networks with continuous and categorical outputs, and GARCH models. Our experiments strongly suggest that the robust information reduction achieved by quantizing the real-valued time series is highly beneficial. To deal with non-stationarity in financial daily time series, we propose two techniques that combine "sophisticated" models fitted on the training data with a fixed set of simple-minded symbolic predictors not using older (and potentially misleading) data in the training set. Experimental results show that by quantizing the volatility differences and then using symbolic predictive models, market makers can sometimes generate a statistically significant excess profit. We also mention some interesting observations regarding the memory structure in the studied series of daily volatility differences.

Keywords

Markov models, prediction suffix trees, fractal geometry, volatility, straddle, options.

1 Introduction

Many real-world time series, especially in economics, possess two important properties: **(1)** they have a large noise component masking the underlying patterns exploitable for prediction, and **(2)** they are, or can be, highly non-stationary. For a reliable estimation of pattern recognition or prediction models using data from such time series, due to property (1), large samples – i.e. time series values over a large time window ([2], [3]) – are needed. Due to property (2), however, the size of such samples cannot arbitrarily be enlarged.

We therefore investigate a class of models for prediction which aim at reducing the noise component while preserving the underlying predictable patterns in the stochastic process. This class of models starts with a careful *quantization* of the continuous-valued time series into a series of discrete symbols[1]. Subsequently, symbolic predictive models – most prominently, the family of Markov models – are used to extract the underlying statistical structure. We demonstrate the viability of such an approach in the domain of financial time series, namely on the application of predicting the volatility of indexes in the financial markets.

The idea of quantizing financial time series has already appeared in several studies [2], [3], [7], [8], [9], [10]. Papageorgiou built predictive models to determine the direction of change in high frequency Swiss franc/U.S. dollar exchange rate (XR) tick data [7] and studied the correlational structure of coupled time series of daily

[1]Quantizing real-valued time series into symbolic streams has been a well-understood and useful information reduction technique in symbolic dynamics. Under certain conditions, stochastic symbolic models of quantized chaotic time series represent, in a natural and compact way, the basic topological, metric and memory structure of the underlying real-valued trajectories (see e.g. [4] [5]). Analogous ideas in the context of stochastic real-valued time series were recently put forward by Bühlmann [6]. He introduces a new class of hybrid real-valued/symbolic models, the so-called quantized variable length Markov chains (QVLMCs), that describes a class of real-valued stochastic processes. QVLMCs are roughly Markov models with variable context length constructed on the quantized sequences with the next step distribution in $\Re$ defined as a mixture of local (say, Gaussian) densities. The densities correspond to the individual partition elements (symbols). The mixture weights are the next-symbol probabilities given by the symbolic Markov model. Bühlmann proves that the class of QVLMCs constitutes a good representational basis for stationary real-valued processes. In particular, the class of QVLMCs is weakly dense in the set of stationary $\Re$-valued processes.

XRs for five major currencies measured against the U.S. dollar [8]. In both cases the real-valued XR returns were quantized into 9 symbols. Papageorgiou predicts the directions of changes in Swiss franc/U.S. dollar XRs using a second-order Markov model (MM) and analyses the correlational structure in the five major XRs through a mixed memory MM [11]. Giles, Lawrence and Tsoi [2] [3] considered the same set of five major XRs and predicted the XR directional changes by applying recurrent neural networks to symbolic streams obtained by quantizing the historic real-valued directional change values using the self-organizing map [12].

Generally, it was found that discretization of financial time series can potentially effectively filter the data and reduce the noise. Even more importantly, the symbolic nature of the pre-processed data enables one to interpret the predictive models as lists of clear (and often intuitively appealing) rules [2] [3]. Yet, there are serious shortcomings in using such techniques:

- The determination of the number of quantization intervals (symbols) and their cut values is *ad hoc*. No strongly supported explanation is given in [7, 8] why 9 symbols with their particular quantization intervals were used. The authors of [2] [3] use up to seven symbols with the cut values determined by the self-organizing map without any attempt to set the quantization intervals to an "optimal"[2] configuration. Kohavi and Sahami [13] warn that naive discretization of continuous data can be potentially disastrous as critical information may be lost due to the formation of inappropriate quantization boundaries. Indeed, *discretization should be viewed as a form of knowledge discovery* revealing the critical values in the continuous domain.

- As already mentioned, due to (non)stationarity issues, it is a common practice to slide a working window through the available data, thereby substantially reducing the amount of training data for model fitting. In such situations, using many symbols can be potentially hazardous as the subsequence statistics is poorly determined.

- Due to the length constraints of the training sequence and the size of used alphabets, the Markov model order is usually set to 2 or 3, whereas only

[2] with respect to the performance measure

a small set of deeper prediction contexts may really be needed to achieve a satisfactory performance. In case of recurrent neural networks one runs across the well-known vanishing gradient effect [14] which reduces the network memory capacity.

We address these issues by

- transforming real-valued time series into symbolic sequences over 2 or 4 symbols. Quantization into 4 symbols is done in an intuitively appealing parametric way.

- testing the memory structure in the data. We use the classical fixed-order MMs alongside with the variable memory length Markov models (VLMM) [15]. The latter deal with the familiar explosive increase in the number of MM free parameters[3] by including *predictive contexts of variable length* with a deep memory *just where it is really needed*. We also propose a novel variation of the fractal-based predictive technique introduced in [1] [16]. The fractal-based predictors are able to efficiently explore long memory and were shown to outperform the classical MMs (and sometimes also VLMMs) on sequences with deep (albeit finite) memory structure [16].

Apte and Hong [9] addressed the issue of optimal alphabet size and cut values. They applied a minimal rule generation system R-MINI[4] to monthly S&P 500 data quantized by a special feature discretization subsystem. However, the features were quantized prior to the rule generation process without any reference to the final model's predictive behavior.

Bühlmann [10] models the extreme events of returns of the Dow Jones and volume of the NYSE given their previous histories. The original return and volume series are quantized into streams over 3 ordinal categories {*lower extreme, usual, upper extreme*} that are used to fit a hierarchy of generalized linear models viewed as sieve approximations to a finite state Markov chain. The fixed, pre-defined cut values for the 3 categories correspond to the 2.5% and 97.5% sample quantiles, so that the *lower* and *upper extreme* categories describe the extreme events with expected

[3]when increasing the model order

[4]the R-MINI rules are in disjunctive normal form

overall occurrence of about 5%. In our experiments, we adopt a similar strategy for a data-driven determination of the cut values in the four symbol quantization scheme.

Singh [17] [18] proposed an interesting time series forecasting system, called *Pattern Modeling and Recognition System* (PMRS), that incorporates the elements of forecasting methods on both quantized and original, real-valued time series. Forecasting in PMRS is done by making an analogy with important past patterns in the studied time-series that closely "match" the pattern characterizing recently seen values. A pattern is described on a crude level by a binary string consisting of signs of differences between consecutive values in a period of the time series. A more refined description of the pattern involves an information about the actual magnitudes of the differences. Two patterns "match", if they not only agree on the symbolic level, but also exhibit "similar" values on the real-valued level. The tolerance defining the desired level of similarity is set in a data-driven manner.

Van Biljon used hidden Markov models to predict daily closing prices in in the bull and bear markets from the Dow Jones Stock Market [19]. Hidden Markov models were also used by Gregoir and Lenglart in predicting turning points in business cycles [20].

An important aspect of the above-mentioned studies is that, apart from [9], they do not present their experimental parts in a realistic trading setting. In this paper, we decided to realistically simulate the trading of options based on predictions of future evolution of the volatility of the underlying.

Traditionally, option price forecasts are based on implied volatilities derived from an observed series of option prices[5]. Taking a different route, Noh, Engle and Kane [21] used a GARCH[6] model [22] to predict the volatility of the rate of return of an asset and then calculated their predictions of option prices based on the GARCH-predicted volatilities. The volatility change forecasts (volatility is going to increase, or decrease) can be interpreted as a buying or selling signal for a straddle[7]. This enables one to implement simple trading strategies to test the efficiency of option

[5]The basic assumption behind this approach is that the volatility must be reflected in option prices

[6]GARCH stands for generalized autoregressive conditional heteroskedasticity.

[7]A straddle is a couple of put (the right to sell) and call (the right to buy) options with the same time to maturity and the same strike price. See the appendix.

markets (e.g. S&P 500 index [21], or German Bund Future Options [23]). If the volatility decreases, we go short (straddle is sold), if it increases, we take a long position (straddle is bought). More details on options and option pricing can be found in the appendix.

Against the optimistic hope that such automatic trading strategies can lead to significant excess profits stands the pessimistic received wisdom (at least among academics) of the *efficient markets hypothesis* [24]. In its simplest form, this hypothesis states that asset prices follow a random walk and so apart from a possible constant expected appreciation (e.g. a risk-free return), the movement of an asset price is completely unpredictable from publicly available information. The ability of technical trading rules to generate an excess profit has been a controversial subject for many years. The evidence against, or for the efficient markets hypothesis still seems inconclusive [25].

In our technical trading, we quantize the time series of historic volatility changes into a symbolic sequence characterizing the original real-valued sequence only through a few distinct events (symbols) such as *sharp increase*, *small decrease*, etc... Instead of modeling the original real-valued trajectory, we look for a set of grammatical and probabilistic constraints characterizing its symbolic counterpart. The outputs of our predictive models give an indication of possible volatility changes and serve as buying/selling signals in simple strategies that trade straddles.

The paper is organized as follows: In the next two sections we describe symbolic, finite memory models that we use as predictors on symbolic streams. In particular, section 2 is devoted to fixed- and variable-order Markov models, and section 3 describes a fractal-based model that makes predictions with deep memory. In section 4 we specify the data used in our experiments and outline the strategy for trading straddles. Section 5 gives a detailed description of the experiments. The experimental results are presented and discussed in section 6. The conclusion summarizes the main findings of this study. We do not assume the reader is familiar with specifics of financial forecasting applications, such as trading options. Therefore, we provide in the appendix a brief introduction to options and option pricing.

2 Markov models

Consider sequences $S = s_1 s_2 ...$ over a finite alphabet $\mathcal{A} = \{1, 2, ..., A\}$, i.e. every symbol s_i is from $\mathcal{A}$. The sets of all sequences over $\mathcal{A}$ with a finite number of symbols and exactly n symbols are denoted by $\mathcal{A}^+$ and $\mathcal{A}^n$, respectively. By S_i^j, $i \leq j$, we denote the string $s_i s_{i+1} ... s_j$, with $S_i^i = s_i$. The (empirical) probability of finding an n-block $w \in \mathcal{A}^n$ in S is denoted by $\hat{P}_n(w)$. A string $w \in \mathcal{A}^n$ is said to be an allowed n-block in the sequence S, if $\hat{P}_n(w) > 0$. We denote the set of all allowed n-blocks in S by $[S]_n$.

Information sources over the alphabet $\mathcal{A}$ define a family P_n of probability measures on n-blocks over $\mathcal{A}$, $n = 0, 1, 2, ...$ Finite memory sources have a memory of length at most L, and formulate the conditional measures

$$P(s|w) = \frac{P_{L+1}(ws)}{P_L(w)}, \quad w \in \mathcal{A}^L,$$

using a so-called *context function* $c : \mathcal{A}^L \rightarrow \mathcal{C}$, from L-blocks over $\mathcal{A}$ to a (presumably small) finite set $\mathcal{C}$ of prediction contexts,

$$P(s|w) = P(s|c(w)). \tag{1}$$

The task of a learner is now to first find an appropriate context function $c(w)$ and then to estimate the probability distribution $P(s|w)$ from the data.

In classical *Markov models* (MMs) of (fixed) order $n \leq L$, for all L-blocks $w \in \mathcal{A}^L$, the relevant prediction context $c(w)$ is chosen a priori as the length-n suffix of w, i.e. $c(uv) = v$, $v \in \mathcal{A}^n$, $u \in \mathcal{A}^{L-n}$. In other words, for making a prediction about the next symbol, only the last n symbols are relevant. However, there is a price to pay for such an elegant and simple model formulation. For large suffix lengths n, the estimation of prediction probabilities $P(s|c(w))$ can become infeasible. By increasing the model order n the number free parameters to be estimated rises by A^n leaving the learner with the problem to cope with a strong curse of dimensionality.

The curse of dimensionality in classical Markov models has lead several authors to develop so-called *variable memory length Markov models* (VLMMs). The task of a VLMM is the estimation of an appropriate context function, giving rise to a potentially much smaller number of contexts considered. This is achieved by permitting the suffixes $c(w)$ of L-blocks $w \in \mathcal{A}^L$ to be of different lengths, depending on the particular L-block w. We briefly review strategies for selecting and representing the prediction contexts.

Suppose we are given a long training sequence S over $\mathcal{A}$. Let $w \in [S]_n$ be a potential prediction context of length $n < L$ used to predict the next symbol $s \in \mathcal{A}$ according to the empirical estimates $\hat{P}(s|w) = \hat{P}_{n+1}(ws)/\hat{P}_n(w)$. If for a symbol $a \in \mathcal{A}$, such that $aw \in [S]_{n+1}$, the prediction probability of the next symbol s, $\hat{P}(s|aw) = \hat{P}_{n+2}(aws)/\hat{P}_{n+1}(aw)$, with respect to the extended context aw differs "significantly" from $P(s|w)$, then adding the symbol $a \in \mathcal{A}$ in the past helps in the next-symbol predictions. Several decision criteria have been suggested in the literature. For example, extend the prediction context w with a symbol $a \in \mathcal{A}$, if the Kullback-Leibler divergence between the next-symbol distributions for the candidate prediction contexts w and aw, weighted by the prior distribution of the extended context aw, exceeds a given threshold [15] [26],

$$\hat{P}_{n+1}(aw) \sum_{s \in \mathcal{A}} \hat{P}(s|aw) \log_A \frac{\hat{P}(s|aw)}{\hat{P}(s|w)} \geq \epsilon_{KL}. \tag{2}$$

The (small, positive) construction parameter ϵ_{KL} is supplied by the modeler. For other variants of decision criteria see [27] [28] [6].

A natural representation of the set $\mathcal{C}$ of prediction contexts, together with the associated next-symbol probabilities, has the form of a prediction suffix tree (PST) [27] [29]. The edges of a PST are labeled by symbols from $\mathcal{A}$. From every internal node there is at most one outgoing edge labeled by each symbol. The nodes of a PST are labeled by pairs $(s, \hat{P}(s|v))$, $s \in \mathcal{A}$, $v \in \mathcal{A}^+$, where v is the string associated with the walk starting from that node and ending in the root of the tree. For each L-block $w = v_1 v_2 ... v_L \in \mathcal{A}^L$, the corresponding prediction context $c(w)$ is then the deepest node in the PST reached by taking a walk labeled by the reversed string, $w^R = v_L ... v_2 v_1$, starting in the root.

The algorithm for building PSTs has the following form[8] [15] [26] [27]:

- the initial PST is a single root node and the initial set of candidate contexts is $W = \{s \in \mathcal{A} | \ \hat{P}_1(s) > \epsilon_{grow}\}$.

- while $W \neq \emptyset$, do:

[8] ϵ_{grow} is a small positive construction parameter

1. pick any $v = aw \in W$, $a \in \mathcal{A}$, and remove it from W
2. add the context v to the PST by growing all the necessary nodes, provided the condition (2) holds[9]
3. provided $|v| < L$, then for every $s \in \mathcal{A}$, if $\hat{P}(sv) > \epsilon_{grow}$, add sv to W.

The depth of the resulting PST is at most L. The tree is grown from the root to the leaves. If a string v does not meet the criterion (2), it is not definitely ruled out, since its descendants are added to W in step 3. The idea is to keep a provision for the future descendents of v which might meet the selection criterion. In general, as the values of ϵ_{grow} and ϵ_{KL} decrease, the size of the constructed PST increases.

Prediction suffix trees are usually constructed using a one-parameter scheme introduced in [15]. This scheme varies only one parameter $\epsilon = \epsilon_{KL} = \epsilon_{grow}$. In this case, however, it can happen that for small values of ϵ, many low-probability subsequences are included as potential contexts in step 3 of the PST construction. The resulting PSTs are too specific and greatly overfit the training sequence. One may improve on that by fixing the growth parameter ϵ_{grow} to a small positive value and varying only the acceptance threshold parameter ϵ_{KL}. This usually removes the overfitting effect in larger PSTs. However, smaller PSTs, corresponding to larger values of ϵ_{KL}, often perform poorly, since the small fixed value of ϵ_{grow} results in considering unnecessarily specific contexts. In our previous work on modeling chaotic symbolic sequences [16], we empirically found that the procedure with ratio-related parameters $\epsilon_{grow} = \rho\ \epsilon_{KL}$, $\rho \approx 50$, give the best results. Although it is not clear that the ratio $\rho = 50$ is optimal for modeling the quantized financial time series used in this study, we nevertheless construct PSTs with both $\rho = 1$ and $\rho = 50$ to test whether the regularization that proved useful in modeling chaotic sequences can be helpful in financial forecasting.

3 Fractal classification machines

In this section, we introduce a novel variation on predictive models built on spatial representations of symbolic subsequences [1] [16]. In the basic construction step of such (finite memory) models, one first fixes a maximum memory length L, and

[9] $\hat{P}(s|\Lambda) = \hat{P}_1(s)$, Λ is the empty string.

then transforms (via a map T) subsequences of length L (L-blocks) into a spatial structure of points in a unit hypercube that has two important properties:

1. images of L-blocks sharing a long common suffix lie close to each other (i.e. form clusters),

2. the longer is the common suffix shared by the L-blocks, the smaller is the distance between their images.

We introduced such a map T, having the properties 1. and 2., in [1] [16] and [30]. The map T transforms L-blocks over the alphabet $\mathcal{A} = \{1, 2, ..., A\}$ into points in the D-dimensional unit hypercube[10] $X = [0,1]^D$, $D = \lceil \log_2 A \rceil$, via an iterative function system (IFS) [31]. The IFS consists of A affine contractive maps[11] $i : X \to X$, $i = 1, 2, ..., A$, acting on X

$$i(x) = kx + (1-k)t_i, \quad t_i \in \{0,1\}^D, \quad t_i \neq t_j \text{ for } i \neq j. \tag{3}$$

The contraction coefficient of the maps $1, ..., A$, is $k \in (0, \frac{1}{2}]$. In this study, we set $k = \frac{1}{2}$. The exact assignment $i \to t_i$ of symbols i to vertices t_i is not important, as long as the assignment is unique.

Given an L-block $u = u_1 u_2 ... u_L \in \mathcal{A}^L$, its spatial image, $T(u) \in X$, is found as follows:

1. Start in $x_0 = \left\{\frac{1}{2}\right\}^D$, the center of the hypercube X.

2. Sliding through the L-block u left-to-right, for $n = 1, 2, ..., L$, iteratively move to $x_n = i(x_{n-1})$, provided the n-th symbol u_n is i.

3. The image of the L-block u under the map T corresponds to the point we move to after observing the last symbol u_L, i.e. $T(u) = x_L$.

As an example, consider an L-block $u = ...142$ over a four-symbol alphabet $\mathcal{A} = \{1, 2, 3, 4\}$, the last three symbols of which are 142. Let the four affine maps

[10] for $x \in \Re$, $\lceil x \rceil$ is the smallest integer y, such that $y \geq x$

[11] To keep the notation simple, we slightly abuse mathematical notation and, depending on the context, regard the symbols $1, 2, ..., A$, as integers, or as referring to maps on X.

on the unit square $X = [0,1]^2$, corresponding to the symbols in $\mathcal{A}$, be defined as ($k = \frac{1}{2}$)

$$1(x) = \frac{1}{2}x + \frac{1}{2}(0,0), \quad 2(x) = \frac{1}{2}x + \frac{1}{2}(1,0),$$
$$3(x) = \frac{1}{2}x + \frac{1}{2}(0,1), \quad 4(x) = \frac{1}{2}x + \frac{1}{2}(1,1).$$

Here, symbols 1, 2, 3 and 4, are associated with the unit square corners $t_1 = (0,0)$, $t_2 = (1,0)$, $t_3 = (0,1)$ and $t_4 = (1,1)$, respectively. Each map $i(x)$, $i = 1,2,3,4$, first contracts the unit square $[0,1]^2$ into the subsquare $[0,\frac{1}{2}]^2$, and then shifts the subsquare towards the corresponding corner t_i of the unit square.

This is illustrated in figure 1. Imagine we have already processed the first $L-3$ symbols $u_1 u_2 ... u_{L-3}$ in the L-block u, so that, starting in $x_0 = (\frac{1}{2}, \frac{1}{2})$, we ended up in a point $x_{L-3} = T(u_1 u_2 ... u_{L-3})$ in the unit square X. Under the map $1(x)$, the black unit square at the top left is contracted and then shifted, so that it fills the subsquare position associated with symbol 1. The shift vectors t_i are schematically shown as the corresponding symbols i appearing at the corners of the unit square. The point $x_{L-2} = T(u_1 u_2 ... u_{L-3} 1)$ lies somewhere in the black subsquare at the corner marked by symbol 1.

The whole process is iteratively repeated. The next symbol in u is 4. Again, the unit square is contracted into $[0,\frac{1}{2}]^2$, but this time the contracted subsquare is shifted to the upper right corner of the unit square. Upon seeing the last symbol in u, 2, the result of the previous step is contracted into $[0,\frac{1}{2}]^2$ and shifted to the lower right corner of the unit square. The points $x_{L-1} = T(u_1 u_2 ... u_{L-3} 14)$ and $x_L = T(u_1 u_2 ... u_{L-3} 142)$ lie inside the (increasingly shrinking) black subsquares in the corresponding plots.

Note that by iteratively making contractions and shifts, we effectively code the history of *recently* seen symbols into subsquares of $[0,1]^2$. Black subsquares inside unit squares in figure 1 correspond to seen strings schematically written on top of the squares. For example, the black square at the top left of figure 1 codes the state of total ignorance - every string over $\mathcal{A}$ could have been seen. The black subsquare inside the unit square labeled by *1 corresponds to all strings ending with symbol 1. The black "subsubsquare" in the unit square labeled by *14 lies in the subsquare corresponding to strings ending with 4 (shaded area) and codes all strings ending with 14. Likewise, the black region in the unit square labeled by *142 corresponds

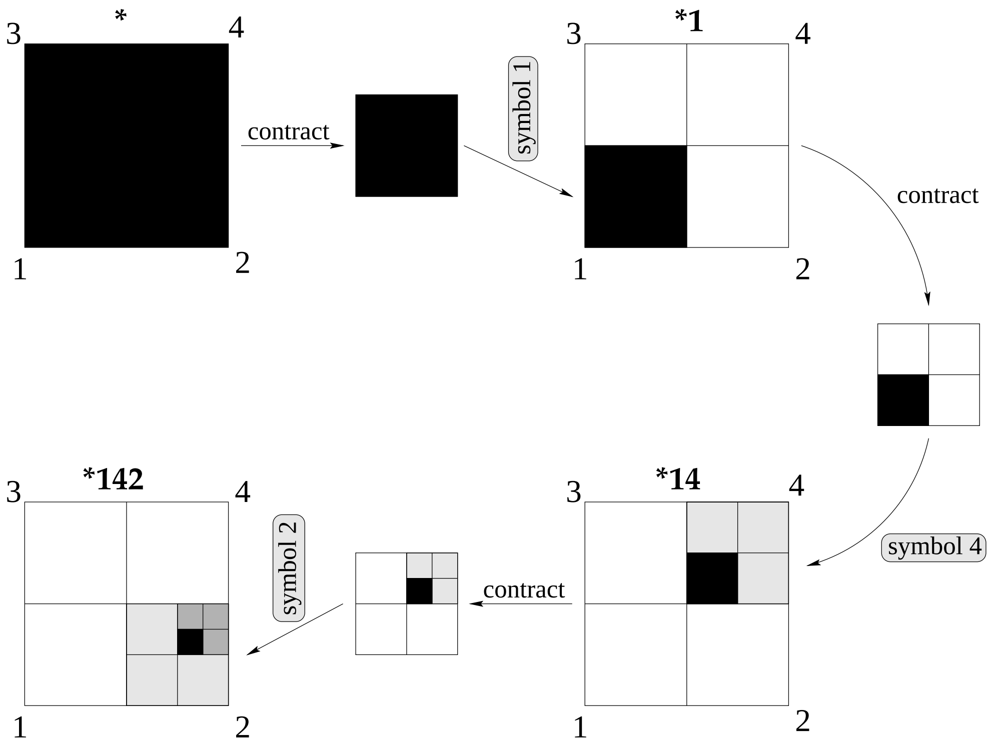

Figure 1: Illustration of the iterative function system behind the spatial representation of symbolic streams. Each symbol 1, 2, 3 and 4 is associated with a unique corner of the black unit square at the top left. Upon seeing symbol 1, the unit square is contracted and shifted towards to corner associated with symbol 1. This process is iteratively repeated as more new symbols arrive. Increasingly longer sequences are coded in the shrinking copies of the original black unit square. In each construction step, the resulting unit square is labeled by the suffix coded by the black subsquare.

to all strings ending with 142.

Two properties of such IFS-driven spatial representations of symbolic sequences are of importance to us. First, if histories of the last symbols in two L-blocks u, v, are the same, i.e. if the strings u and v share a common suffix, their representations $T(u)$ and $T(v)$ lie close to each other. Second, the longer is the common suffix shared by u and v, the smaller is the region containing the points $T(u)$ and $T(v)$. For a rigorous treatment of fractal representations of symbolic sequences driven by iterative function systems, see [30].

We now describe predictive models that we call *fractal classification machines* (FCM). These models are designed to efficiently use deep memory when inferring the next symbol. The inference is viewed as a classification task. The FCMs operate as follows:

1. Create a set $\mathcal{D}$ of training examples for a classifier operating on X by sliding a window of length L through the training sequence $S = s_1 s_2 ... s_N$

   (a) for each position $p = 1, 2, ..., N-L$, of the window, transform the L-block

   $$u = u_1 u_2 ... u_L = S_p^{p+L-1} = s_p s_{p+1} ... s_{p+L-1}$$

   appearing in the window into the point $T(u)$

   (b) the set $\mathcal{D}$ contains all couples $(T(u), s)$, such that $u \in [S]_L$ is an L-block in S, and s is the symbol following u in S. In other words, the set

   $$\mathcal{D} = \left\{ \left(T\left(S_i^{i+L-1} \right), s_{i+L} \right) \mid \; i = 1, 2, ..., N-L \right\}$$

   consists of labeled points $T(u)$ that are spatial representations of L-blocks $u \in [S]_L$ in S. The points are labeled with the corresponding next symbols in S.

2. Upon seeing a new history of L symbols, $v = v_1 v_2 ... v_L \in \mathcal{A}^L$, we make a guess about the next symbol by

   (a) mapping the L-block v into $T(v)$ and

   (b) predicting the next symbol as the class label for $T(v)$ returned by the K-Nearest-Neighbor (KNN) classifier [32] operating on the set $\mathcal{D}$. The KNN classifier finds K points $y_i \in X$, such that $(y_i, \kappa_i) \in \mathcal{D}$, $i = 1, 2, ..., K$, that are closest (in Euclidean distance) to $T(v)$. Then, it consults the corresponding labels $\kappa_1, \kappa_2, ..., \kappa_K \in \mathcal{A}$. The class assigned to $T(v)$ is the symbol appearing most often among the labels $\kappa_1, \kappa_2, ..., \kappa_K$.

The radius of the neighborhood of $T(v)$ in X involved in predicting the next symbol depends on the density of labeled points from $\mathcal{D}$ around $T(v)$ and on the parameter K determining the number of labeled samples from $\mathcal{D}$ we wish to consult. For fixed K, the neighborhoods around $T(v)$ will be small in dense regions that correspond to L-blocks sharing a long common suffix. In other words, if the last L symbols $v = v_1 v_2 ... v_L$ happen to share a deep suffix with a subset of L-blocks seen in the training sequence, then our decision about the next symbol will be based upon that subset, which effectively amounts to a prediction with a deep memory. If, on the other hand, the region around $T(v)$ is relatively sparse, then the effective memory involved in predicting the next symbol will be shorter. Also, the smaller is the parameter K, the smaller neighborhoods around $T(v)$ are considered, and hence the deeper is the memory used for prediction. In this sense, FCMs using relatively small values for the parameter[12] K correspond to variable memory length Markov models with a preference for deeper memory. For a different variant of finite-memory fractal-based predictive models constructed by vector quantizing the L-block representations in the set $\mathcal{D}$, see [1] [16] [33].

4 Data, volatility measure and trading strategy

We encourage the readers not familiar with option contracts and option pricing to consult the appendix prior to reading this section.

Our first data set consists of daily closing values of the DAX and daily closing prices of (European) call and put options on the DAX, from August 22, 1991 until June 9, 1998, which covers a period of 1700 trading days. In particular, the prices of the first in-the-money and the first out-of-the-money call and put option are available[13].

[12] in our experiments, $K \in \{3, 5, ..., 11\}$

[13] The at-the-money point is assumed to be the closing value of the DAX.

As a volatility measure, we chose the implied volatilities $\{V_t\}$ calculated from the observed option prices by inverting the Black-Scholes formula (see the appendix).

The basic trading strategy is to buy (sell) at-the-money straddles whenever volatility is predicted to increase (decrease). Since at-the-money straddles are approximately delta-neutral, there is no need to delta-hedge, and the strategy is thus a pure volatility trading strategy. In detail, every day, a constant amount of money is invested to buy (sell) the straddles, and on the next day, the straddles are sold (bought). If a model is uncertain about the sign of the volatility change (equal evidence for increase and decrease), the invested money is put into the bank at an annualized interest rate of 4%.

The second data set contains intra-day bid-ask prices of (American-style) options on the FTSE 100 between 29 May 1991 and 29 December 1995 (which covers a period of 1161 trading days). The option prices are recorded synchronously with the FTSE 100 and time-stamped to the nearest second. Since our trading strategy is set up on a daily basis, we have to fix a reference point in time on each trading day. This reference point is 3 pm on normal trading days and 12 pm on days where the stock exchange closes earlier. To trade straddles as in the DAX experiment, we extract the first quotes of call and put options with the same strike price that is as close as possible to the FTSE 100 at that time. For these options, which are roughly at-the-money, we calculate the average of bid and ask prices as a proxy of a reasonable option price. Then the prices of call and put options are added to obtain the straddle price. During this procedure, prices are carefully checked for outliers, recording errors, unusually large bid-ask spreads, and other sources of possible contamination. The final data set may thus be assumed to contain prices at which transactions could have been made at the market over the experimental time period.

The series of returns r_t of the financial indexes DAX and FTSE 100 obtained from daily index values x_t via

$$r_t = \log x_t - \log x_{t-1} \tag{4}$$

are shown in figure 2.

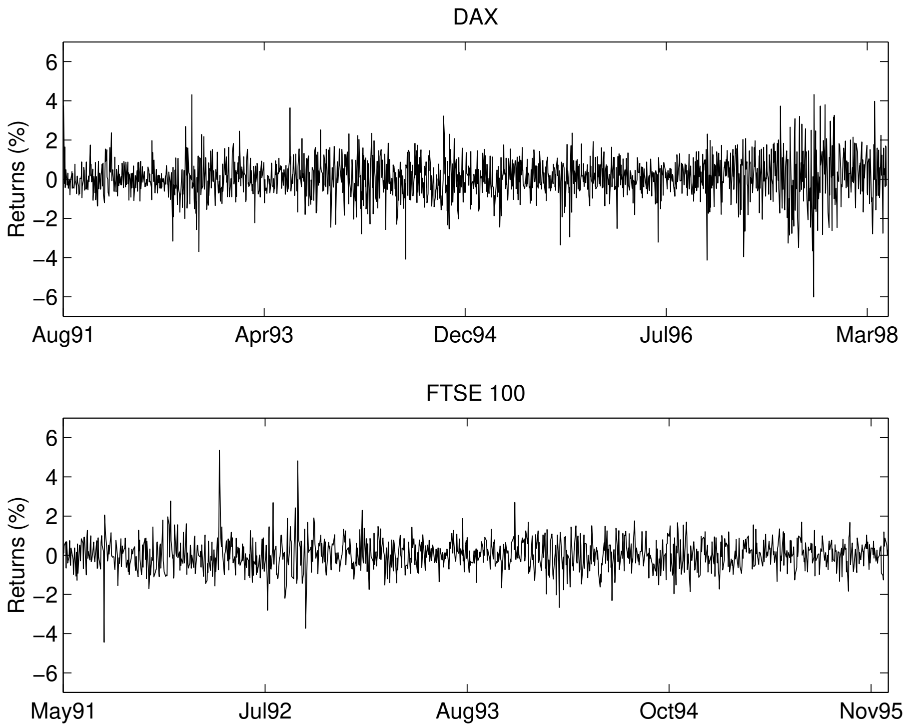

Figure 2: Series of daily returns of the financial indexes DAX and FTSE 100.

5 Experimental setup

Given the series $\{V_t\}$ of estimated daily volatilities, we create a new series $\{D_t\}$ of daily volatility differences $D_t = V_t - V_{t-1}$. On the basis of the series $\{D_t\}$ we construct, select and test predictive models used in our trading strategy.

As mentioned in the introduction, to deal with non-stationarity in $\{D_t\}$, we use the sliding window technique. At each position, the sliding window of length 630 contains the training set (the first 500 points - roughly two years), followed by a validation set (125 points - roughly six months), and a test set (5 points - one week). Predictive models are estimated on the training set. Within each model class, the best performing candidate with respect to profit is selected on the validation set, and finally, the profit of the selected model is determined on the test set. Then the time window is shifted by 5 days, predictive models are re-estimated, etc...

5.1 Quantizing real-valued time series into symbolic streams

Symbolic models operate on quantized versions of the real-valued series $\{D_t\}$ of daily volatility differences. In particular, we perform quantization into symbolic streams $\{Q_t\}$ and $\{R_t\}$ over two and four symbols, respectively. The sequence $\{Q_t\}$ over the binary alphabet $\mathcal{A} = \{1, 2\}$ is obtained from $\{D_t\}$ as follows:

$$Q_t = \begin{cases} 1 \text{ (down)}, & \text{if } D_t < 0 \\ 2 \text{ (up)}, & \text{otherwise.} \end{cases} \tag{5}$$

Quantization using four symbols is more involved, since we have to determine the positions of the cut values separating "normal" from "extremal" volatility differences D_t. The sequence $\{R_t\}$ over the alphabet $\mathcal{A} = \{1, 2, 3, 4\}$ is constructed as follows: for each position τ of the sliding window $\{D_t\}_{\tau}^{\tau+629}$

1. we quantize the training set $\{D_t\}_{\tau}^{\tau+499}$ into the symbolic stream $\{R_t\}_{\tau}^{\tau+499}$,

$$R_t = \begin{cases} 1 \text{ (extreme down)}, & \text{if } D_t < \theta_1 < 0 \\ 2 \text{ (normal down)}, & \text{if } \theta_1 \le D_t < 0 \\ 3 \text{ (normal up)}, & \text{if } 0 \le D_t < \theta_2 \\ 4 \text{ (extreme up)}, & \text{if } \theta_2 \le D_t. \end{cases} \tag{6}$$

   The parameters θ_1 and θ_2 correspond to $\mathcal{Q}$ percent and $(100 - \mathcal{Q})$ percent sample quantiles of the marginal empirical distributions $\hat{P}(\cdot|D < 0)$ and $\hat{P}(\cdot|D \ge 0)$, respectively, over the volatility differences D. The empirical distributions are calculated from the real-valued training set $\{D_t\}_{\tau}^{\tau+499}$. So $\hat{P}(\cdot|D < 0)$ describes the distribution of negative volatility differences in the training set, and the lower $\mathcal{Q}$% of volatility decreases are considered extremal. The upper $(100-\mathcal{Q})$% volatility decreases are viewed as normal. Analogically, $\hat{P}(\cdot|D \ge 0)$ captures the distribution of non-negative volatility differences in the training set. The upper $\mathcal{Q}$% of volatility increases are considered extremal. The lower $(100 - \mathcal{Q})$% volatility increases are viewed as normal. In our experiments, we use the quantile values $\mathcal{Q} \in \{10, 20, 30, ..., 90\}$.

2. the validation symbolic sequence $\{R_t\}_{\tau+500}^{\tau+624}$ is obtained, for each $\mathcal{Q} \in \{10, 20, 30, ..., 90\}$, by quantizing the real-valued validation set $\{D_t\}_{\tau+500}^{\tau+624}$ using the corresponding cut values θ_1, θ_2 determined on the training set.

3. for each predictive symbolic model, the optimal value $\mathcal{Q}^*$ of $\mathcal{Q} \in \{10, 20, 30, ..., 90\}$ is determined on the validation set:

   (a) both the training and validation sets are quantized using the same $\mathcal{Q}$

   (b) the model is fitted on the training set

   (c) $\mathcal{Q}^*$ is the value of $\mathcal{Q}$ which leads to the highest validation set profit.

4. the test set $\{D_t\}_{\tau+625}^{\tau+629}$ is then quantized into $\{R_t\}_{\tau+625}^{\tau+629}$ using the cut values θ_1, θ_2 (determined on the training set) that correspond to the quantile $\mathcal{Q}^*$.

5.2 Predictive models

To make predictions about the nature of the volatility move for the next day that will be used in our strategy for trading the straddles, we use the following predictive model classes:

- **MM(5)** – Markov models (MM) of order up to 5 (one week). This model class includes MMs of order 0,1,2,...,5 (see section 2). The model order for the test set is determined on the validation set.

- **MM(10)** – MMs up to order 10 (two weeks). The class includes MMs of order 0,1,2,...,10. As in the previous class, the model order is determined on the validation set.

- **PST(1)** – variable memory length Markov models (VLMM) with the associated prediction suffix trees (PST) constructed using the ratio-related parameter scheme $\epsilon_{grow} = \rho\ \epsilon_{KL}$, $\rho = 1$ (see section 2). The case of $\rho = 1$ is equivalent to the PST construction procedure introduced in [15]. Maximum memory depth is set to $L = 15$ (three weeks). We build PSTs of various sizes by varying the threshold parameter ϵ_{KL} as follows[14]: $\epsilon_{KL} \in \{0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$. The optimal value of ϵ_{KL} is determined on the validation set.

- **PST(50)** – regularized VLMMs with the associated PSTs constructed using the ratio coefficient $\rho = 50$. All the remaining construction details are the same as in the class *PST(1)*.

[14] these parameter values were experimentally found to give a reasonable range of PST sizes

- **FCM** – fractal classification machines using deep memory in an efficient way (see section 3). To test for deeper memory structures, we set the maximum memory depth to $L = 15$ (three weeks). The parameter K can take on values $K = 3, 5, ..., 11$. The optimal number of nearest neighbors, K, is determined on the validation set.

- **NN** – time-delay neural networks[15] with a single linear output unit, 3 hidden units and inputs corresponding to the last 1, 3, 6, or 10 points from the training set. The transfer function of the hidden units is the standard tangent hyperbolic $\nu(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$. The networks are trained (using conjugate gradient) on the real-valued series $\{D_t\}$ of daily volatility differences to predict the volatility difference for the next day. The optimal size of the network input is determined on the validation set.

- **NN-symb** – time-delay neural networks with categorical output. The architectures of neural networks in this class are the same as those in the previous class *NN*, except for the output layer. Here, we have two output units with the logistic sigmoid transfer function $\sigma(x) = \frac{1}{1+e^{-x}}$. The output units correspond to the symbols 1 and 2 used in the two-symbol quantization scheme. The networks are trained (again, using conjugate gradient) on the real-valued series $\{D_t\}$ as input to predict the *direction* of the volatility move for the next day, i.e. 1 (decrease), or 2 (increase).

- **GARCH** – GARCH(1,1) models [22] with a Gaussian distribution and a t-distribution. This model class was included as a benchmark, since it has been used in similar studies (e.g. [21] [23]). In contrast to the previous model classes, the GARCH models are not trained on the pre-computed series $\{D_t\}$ of differences of historical volatilities. Instead, the GARCH models try to reconstruct the volatilities from the series of returns $\{r_t\}$ (eq. (4)). Basic to these models is the notion that the series of returns $\{r_t\}$ can be decomposed into a predictable component μ_t and an unpredictable component e_t, which is assumed to be zero mean Gaussian (or t-distributed) noise of finite variance σ_t^2: $r_t = \mu_t + e_t$. The models are thus characterized by time-varying conditional

[15] time-delay neural networks can be considered as non-linear auto-regressive filters

variances σ_t^2 and are therefore well suited to explain volatility clusters typically present in the series of returns. The conditional mean is modeled as a linear function of the previous value: $\mu_t = ar_{t-1} + b$. For the GARCH(1,1) model, the conditional variance σ_t^2 is given by

$$\sigma_t^2 = a_0 + a_1 e_{t-1}^2 + a_2 \sigma_{t-1}^2. \quad (7)$$

At each sliding window position, the GARCH(1,1) models with a Gaussian and a t-distribution are fit, in the maximum likelihood framework, to the series of returns corresponding to the training set. The optimal form of the noise distribution (Gaussian vs. t-distribution) is then determined on the validation set. The GARCH model makes a prediction about the direction of the volatility change based on the sign of $\sigma_t^2 - \sigma_{t-1}^2$.

Given historical data, the models predict the sign of the next volatility change as follows:

- MMs and PSTs trained on sequences over
  - 2 symbols: predict *decrease* (*increase*) when the model predicts symbol 1 (2) with higher probability than symbol 2 (1); if the two probabilities coincide, output *don't know* (equal evidence for *decrease* and *increase*)
  - 4 symbols: predict *decrease* (*increase*) when the *sum* of the predictive probabilities for symbols 1 and 2 (3 and 4) is greater than that for symbols 3 and 4 (1 and 2); if the two sums of probabilities coincide, declare *don't know*
- FCMs on sequences over
  - 2 symbols: predict *decrease* (*increase*) when FCM classifies the spatial image $T(v)$ of the recent history of symbols v as class 1 (2); FCMs cannot make a *don't know* statement, since we use only odd values for the parameter K in the K-nearest neighbor classification
  - 4 symbols: predict *decrease* (*increase*) when, out of the K nearest neighbors of $T(v)$, the number of points labeled with 1 or 2 (3 or 4) is greater

than the number of points labeled by 3 or 4 (1 or 2)[16].

- NNs predict *decrease* (*increase*) provided the output is negative (positive). Otherwise, declare *don't know*. Since prior to training, the network weights are randomly initiated, we stabilize the network classification decisions by training a committee of 10 networks (for each input size - 1, 3, 6 and 10). Given an input, each member of the committee makes its prediction (*decrease*, *increase*, or *don't know*). The overall output of the committee is then based on the majority vote.

- NNs with categorical output (NN-symb) predict *decrease* (*increase*), when the activation of the output unit corresponding to symbol 1 (2) is greater than that of the other unit. If the activations are the same, declare *don't know*. The committee technique described for the class *NN* is used also in this class.

5.3 Taking the non-stationarity seriously

At each position of the time-window, the models are fitted to data that start $2\frac{1}{2}$ years before their actual use for volatility predictions on the test set. This can be dangerous, especially when sudden large "stationarity breaks" are present in the data (see [34]). On the other hand, as discussed in the introduction, using a shorter sliding window can lead to undesirable overfitting effects.

Our idea is to use yet another model class, **Simple**, that is a small collection of simple, *fixed* predictors requiring no training on the training set. A suitable candidate model to be applied on the test set is selected on the validation set. The class *Simple* avoids using the training data since they form the oldest part in each sliding window. Of course, the price to pay is the fixed nature of the models in *Simple*. However, in financial prediction tasks, simple, short memory models often outperform more sophisticated predictors. Our choice of models for the class *Simple* is a collection of four simple-minded predictors operating on the series of volatility differences quantized using the two-symbol scheme: *always predict 1 (decrease)*,

[16]Note that in this case, FCM may predict an increase, even though the largest number of neighbors belong to, say, class 1. What is important are the *summed* class counts for *down* and *extreme down* (*up* and *extreme up*).

always predict 2 (increase), *copy the last symbol* and *reverse the last symbol* (i.e. predict the other symbol).

We hope to build a more powerful prediction strategy by combing the model classes $\mathcal{M}$ summarized in section 5.2 with the class *Simple*. At each position of the time-window and for each model class $\mathcal{M}$, we switch between the winner candidate of $\mathcal{M}$ and the winner from *Simple* in two ways:

1. select between the two candidates based on the *mean* profit achieved on the validation set. We denote this technique by **Comb-Val**. This may not always work, since the consecutive validation sets are highly overlapping and a good validation set profit can be generated on an older part of the set, while the more recent profits can be negative. Similarly, a bad validation set performance may be caused by negative profits on an older part of the validation set, while the model performance on the more recent days may be quite good. What matters is the *slope* of average validation set profits corresponding to recent time-window positions. The slope tells us whether in the history of recent mean validation set profits there is a tendency towards improving the model performance, or not. Moreover, the magnitude of the slope reflects the strength of such a tendency. Therefore we

2. compute the *mean* (per-day) validation set profits of the winner candidates from $\mathcal{M}$ over the last 8 time-window positions (2 months of test data). We do the same for the *Simple* model class. In both cases, we compute the slope in the recent history of 8 mean validation set profits by regressing the last 8 mean validation set profits using a linear regression. Note that if one of the more recent validation set profits is unusually high (e.g. due to some abrupt change in the market conditions), the linear regression will yield a steep slope, but the "outlier" character of such a profit will be reflected in a broad confidence interval for the slope. Therefore, we determine the 95% confidence interval for the slopes and consider only the slope corresponding to the lower boundary of the confidence interval. If both slopes are positive, we select the candidate with the higher slope; if they are negative, we choose between the winner from $\mathcal{M}$ and *Simple* based in the current mean validation set profit (strategy *Comb-Val*). If the slopes are of different sign, use the candidate corresponding to the

positive slope. We denote this method by **Comb-LR**.

5.4 Performance measures

For each model class, we evaluate the overall test set profit by concatenating the test set profits for all time-window positions. All profits are expressed in percent of money we invest each day; 0 means no profit. A big advantage of this approach is that all results are scalable and readily interpretable for any amount of money we wish to invest. The first quantity we report is the mean of the series of concatenated test set profits, i.e. the mean profit per-day before transaction costs.

Many studies in the literature on automatic trading report standard deviations and t-test-related significance results for the series of daily profits. There are two difficulties with such reports.

1. The unconditional distribution of daily profits is usually far from Gaussian. In particular, it is fat-tailed, i.e. it has excess kurtosis. In this case, the application of t-tests is not justified. On the other hand, using standard non-parametric tests based on ordering the joint series, like the Wilcoxon or Mann and Whitney U-tests [35], may not be economically meaningful. Consider comparing two series of profits that are equal up to the last profit, which is 0 in the first series and a very large loss of money in the second series. The non-parametric test would still claim that the two series are not significantly different.

2. Standard deviations computed from the series of daily profits provide an estimate of the risk involved in investing your money, *if you wanted to trade just for one day.* This is not what a reasonable trader would do. He, or she, may be interested in assessing the risk involved in trading for some fixed, longer period of time.

Therefore, we partition the time series of daily profits into non-overlapping blocks of length ℓ and compute a new series of average daily profits achieved within each block. Since the daily test set profits are virtually uncorrelated[17], if the block length

[17]at the 95% confidence level

ℓ is long enough, by the central limit theorem, the average block-profits[18] are approximately Gaussian-distributed[19]. Hence we can subject the series of average block-profits to t- and paired t-tests. The significance level for both tests is set to 95%. The standard deviation computed from the series of average block-profits estimates the risk involved in trading over a period of ℓ days. The mean remains unchanged, since the average of the average block-profits is the average profit per-day. There is an upper bound on the block length ℓ, too. Large ℓ means smaller number of blocks in the partition of the daily profits and hence smaller number of average block-profits for computation of the standard deviation and significance tests. Considering the lengths of the series of daily profits in the DAX and FTSE experiments, we set the block length ℓ to 60 (3 months) and 40 (2 months), respectively.

To account for transaction costs,(TC) we report the maximum amount[20] that we can subtract from each daily profit, so that the average block-profits still have a significantly positive mean (under the t-test).

5.5 Summary of the experimental setup

The overall picture of our experimental setup is shown in figure 3. A sliding window contains a training, a validation and a test sequence. Sequences of daily volatility differences appear in the sliding window in the (original) real-valued form, or as symbolic streams produced in the quantization step. At each sliding window position, and for each model class $\mathcal{M}$, the models from $\mathcal{M}$ are trained on the training sequence. Then the models are used to predict the signs of volatility differences on the validation set. These predictions are used as trading signals in our trading strategy thereby producing a series of validation set profits. Based on the average validation set profit, we select the candidate of the model class $\mathcal{M}$ to be used on the test set[21]. The selected candidate is then used to predict the signs of volatility

[18] the average profits per-day computed within the blocks

[19] The central limit theorem is formulated under asymptotical considerations: when the number of i.i.d. random variables approaches infinity, the distribution of their average is Gaussian. However, the distributions of the averages approach the Gaussian distribution very fast, as the number of variables increases (40 random variables are usually sufficient).

[20] expressed, as in the case of daily profits, in percent of money we invest every day

[21] when quantization into 4 symbols is used to pre-process the data, the optimal quantization quantile is also determined on the validation set

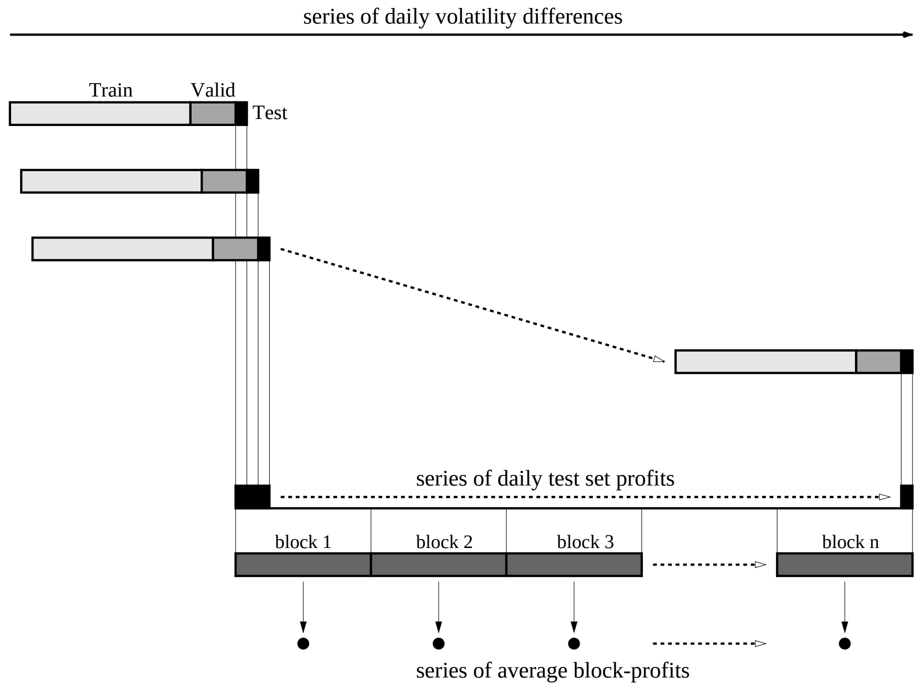

Figure 3: An illustration of the experimental setup used in the DAX and FTSE 100 experiments.

differences on the test set, which are in turn plugged into the trading strategy to produce the test set profits. Also, at each position of the sliding window and for each model class $\mathcal{M}$, we save the average profit achieved on the validation set by the selected candidate from $\mathcal{M}$. The series of average validation set profits are used in methods *Comb-Val* and *Comb-LR* that combine simple strategies from *Simple* with the more complex model classes studied in this paper.

After the sliding window reaches its final position, for each model class, we concatenate all the daily test set profits into a single profit series. This series is then partitioned into several non-overlapping blocks of constant length. For each block we compute the average profit per-day. Finally, we report various statistics computed on thus obtained series of average block-profits.

6 Results and discussion

Tables 1, 2 and 3 present results for both the DAX and FTSE 100 experiments. The performances of symbolic models operating on the binary sequences and sequences over the four-symbol alphabet are shown in tables 1 and 2, respectively. The results of the models operating on the real-valued sequences are reported in table 3. Shown are the average profits per-day (before transaction costs) achieved by the base model classes and the two associated combination strategies importing simple methods into the more sophisticated base model classes (see section 5.3). Standard deviations reflect the variations among the average block-profits.

Furthermore, we show significance results obtained by running paired t-tests on the average block-profits. In particular, we compare each predictor with the class *Simple*, and we also compare the combined strategies (switching between the base model class and *Simple*) with the base model class: $(-)$ means that the model class performance is significantly worse than that of *Simple*, $(+)$ appears where a combined strategy significantly outperforms the base model class, $(*)$ indicates that the realized profit is significantly higher than the profit of the simple strategy.

Finally, we report the highest transaction costs (TC) that can be taken out from daily profits, so that the average block-profits still have a significantly positive mean (under the t-test).

To test the usefulness of our notion of volatility for automatic trading strategies, we report profits for the hypothetic *always correct predictor*, (**ACP**), i.e. the predictor that always knows in advance the sign of the next volatility difference.

Several observations can be made based on the experimental results.

1. The profits gained by the hypothetic *always correct predictor* (ACP in table 1) show that, theoretically, predicting daily volatility changes (with volatilities estimated by implied volatilities) provides a good basis for automatic trading strategies buying/selling straddles. We also tried another popular measure of volatility - the average of the past squared returns with exponentially declining weights - also known as RiskMetrics™ [36] [37]. Such volatility measures were advocated as being well-suited for the purpose of trading options on financial indexes by Figlewski [38] (see also [39]). Interestingly enough, compared with implied volatilities, we got almost the same (DAX experiment) or less favorable (FTSE 100 experiment) results when using the RiskMetrics™ notion of volatility.

2. In general, profits generated by automatic trading of straddles are higher in the FTSE 100 experiment than in the DAX experiment (see tables 1, 2 and

3), suggesting that the option market for the DAX seems to be more efficient than that for the FTSE 100.

3. As typical in financial prediction tasks, simple-minded predictors in the class *Simple* are difficult to beat by more complicated models and should definitely be considered in studies like this one.

4. In general, the binary quantization scheme leads to better results than the more refined four-symbol scheme (see tables 1 and 2). This is primarily due to the short lengths of the training sequences - a burden typical of daily financial data. We are highly sceptical about using more than four symbols in the quantization scheme. In applications like this one, we advise to start with two symbols, and if desired, carefully and in a controlled way, add further quantization levels.

5. On average, the binary Markov models (MMs) (see the rows corresponding to MMs in table 1) outperform the continuous-valued neural networks and GARCH models (table 3). This confirms, in a realistic trading setting, the previous findings [2], [3], [7], [8], [9], [10] [33] that quantizing real-valued time series in the financial domain may be beneficial.

6. Comparing "pure" model performances without combing with the class *Simple*, the highest per-day profits are obtained by the binary Markov models (MMs) with memory depth up to one week (class *MM(5)*). Higher order MMs (from the class *MM(10)*) are prone to overfitting (each of our training sets contained only 500 items).

7. Apart from the 4-symbol scheme in the DAX experiment, prediction suffix trees (PST) do not outperform the fixed-order MMs (see tables 1 and 2). This illustrates the phenomenon we addressed in [1] [16], that there are many *practical* problems in fitting variable memory length Markov models (VLMM). One-parameter construction schemes, like that presented in [15] and used here (see section 2), operate only on small subsets of potential VLMMs. Especially when trained on relatively short sequences generated by processes with a rather shallow memory, VLMMs tend to overestimate the data structure. Even the regularization procedure using $\epsilon_{grow} = 50\epsilon_{KL}$ (see section 2), which

we found useful in other contexts [16], did not improve the VLMM profits. Simply stated, VLMMs are "too sophisticated" for financial forecasts of the kind studied in this paper.

8. Neural networks tend to overestimate the structure in the training data. Forcing them to predict only the principal trends (increase/decrease) in the series of daily volatility differences leads to better profits (compare entries in table 3 corresponding to NN and NN-symb). However, detailed values of volatility differences presented at networks' inputs are not necessary and can actually be misleading. Our results indicate that nothing is lost by using simple Markov models fed by the quantized inputs (MM(5) entry in table 1).

9. In most cases, the combination strategies *Comb-LR* and *Comb-Val* work well, increasing the mean and/or decreasing the variance of the block profits. Sometimes, a combination of the base model class with the *Simple* class achieves better results than those achieved by the two model classes themselves (e.g. *Comb-LR* technique using binary Markov models in the DAX experiment, see table 1). As an illustration, we show in figure 4 cumulative profits (expressed in percentage of invested money) in the DAX experiment, achieved by the class *MM(10)* of Markov models up to order 10, the class *Simple*, and the combination *Comb-LR* of *MM(10)* and *Simple*. In general, none of the combination methods *Comb-LR*, *Comb-Val*, seems to outperform the other one.

10. By their nature, fractal classification machines (FCMs) are *forced* to use deep memory. Maximum memory depth is set to 15, which would be inconceivable with classical Markov models (MMs), given the short length of the training sequences. Inevitably, FCMs suffer more from overfitting than MMs, where for each sliding window position, the order can be selected between 0 and 5 (10), depending on the validation set performance. PSTs are also constructed with maximum memory depth 15, but the individual prediction trees can be very shallow, depending on the construction parameters' values.

    We used FCMs to test how good the combination strategies are in switching between the *long*-memory FCM predictions and the *short*-memory predictions of *Simple*. In the DAX and FTSE 100 experiments, under the binary quantiza-

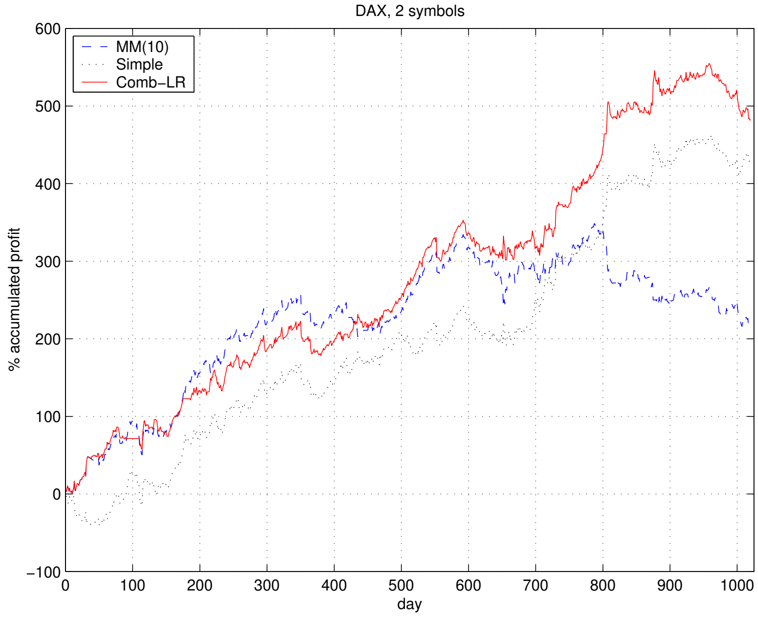

Figure 4: Cumulative profits gained by the class *MM(10)* of Markov models of maximal order 10, the class *Simple*, and the combination *Comb-LR* of *MM(10)* and *Simple* in the DAX experiment. The profits are expressed in percentage of invested money.

tion scheme, the combination schemes *Comb-Val* and *Comb-LR*, respectively, work very well (see the Comb-Val and Comb-LR FCM entries in table 1 for DAX and FTSE 100 experiments, respectively). This indicates that a profitable series of volatility predictions can be obtained by switching between the two extremes, i.e. long-memory and short-memory regimes. In combination with *Simple*, FCMs can be very powerful. Hence, to make profitable volatility predictions, we do not have to use models with a wide spectrum of memory depths. It is sufficient to switch between shallow memory models (such as those in the class *Simple*) and models like FCMs that *use deep memory in an efficient way.*

11. It may be interesting to analyze the data by plotting and comparing characteristics of the selected candidates from the predictive model classes used in the technical trading. For example, in figure 5 corresponding to the DAX experiment, we show the evolution of memory depths in the classes *MM(5)* and *MM(10)* of binary Markov models of maximal order 5 and 10, respectively

(upper graph). For each trading day, we show the order of the Markov model that was selected on the corresponding validation set as the candidate from the class *MM(5)* (*MM(10)*)[22]. Models from the class *MM(10)* tend to use either a shallow (order 0–1) or a rather deep (order 8–10) memory. Very few intermediate memory lengths are used. Note the sharp changes from low- or no-memory regimes to deep memory contexts in the plot for the model class *MM(10)*.

The history of active strategies from the class *Simple* (*always predict Up* (U), *always predict Down* (D), *copy the last symbol* (C) and *reverse the last symbol* (R)) is shown in the lower graph of figure 5. For each trading day, the active strategy was selected based on the validation set performance. Note that there seems to be a correlation between memory depths selected in the class *MM(5)* and strategies picked up from the class *Simple*. Roughly, in periods dominated by MM orders 1 and higher (days 1–70 and 700–970), strategies *always predict Up* and *reverse the last symbol* are selected. In periods dominated by memoryless MMs[23] (order 0), the strategies *always predict Down* and *copy the last symbol* seem to be preferred. Such an analysis is beyond the scope of this paper, but it is an interesting direction for the future research.

12. Schmitt and Kaehler[24] [39] distinguish three groups of investors: market makers, other registered traders (traders) and non-members of the exchange, with transaction costs[25] per straddle of 0.1%, 0.5% and 1%, respectively. To be on the safe side, we double the transaction costs, i.e. we assume that market makers, traders and non-members pay 0.2%, 1% and 2%, respectively, of their

[22]The MM order determines how far into the past the model looks in order to predict the nature of the next volatility move. MM of order 0 simply predicts the most frequent symbol in the training set. MM of order 1 looks back one day and based on the nature of the most recent volatility change determines the next volatility move. The predictive probabilities (conditional on what happened in the previous day) are calculated on the training set. MM of order 2 considers the last 2 volatility changes, etc.

[23]MMs that just count the numbers of *Ups* and *Downs* in the training set

[24]Schmitt and Kaehler traded straddles on the DAX index at the German Futures and Options Exchange (Deutsche Terminbörse - DTB). The transaction cost estimates reported in [39] and used here are based on the information provided by DTB. We assume similar transaction costs for trading straddles on the FTSE 100 at LIFFE.

[25]expressed in percent of the straddle price

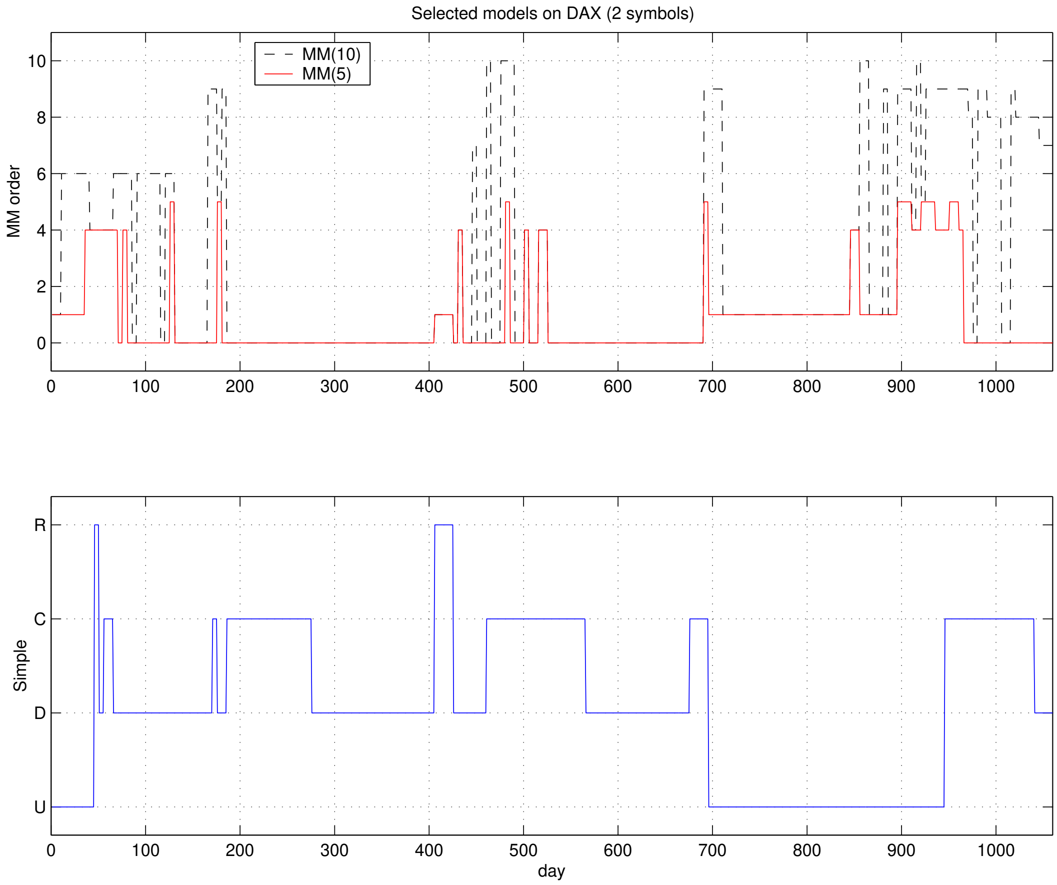

Figure 5: Characteristics of representatives of the binary model classes *MM(5)*, *MM(10)*, and *Simple* in the DAX experiment. For trading day we plot the MM order (upper graph), and the selected strategy from *Simple* (lower graph). The strategies *always Up*, *always Down*, *reverse last*, *copy last* are denoted by U, D, R and C, respectively.

profit as transaction costs. The experimental profits suggest that the option market for the DAX is not efficient from the point of view of market makers, but it is efficient from the standpoint of registered traders and non-members. The option market for the FTSE 100 seems to provide the market makers and registered traders with an opportunity to generate (in an automatic way) an excess profit. The market is efficient for non-members of the exchange.

We stress that we provide a rather *pessimistic* assessment of profits. Consider the following example: we buy a straddle on day t; according to our strategy we will sell it the next day; but on day $t+1$ we may decide to buy the same straddle again. In this case it would be better to keep the straddle until we decide, based on the volatility prediction, to sell it. This way we could avoid having to pay the transaction costs every day. Also, traders like us, that trade on a daily basis and are willing to invest potentially large amounts of money[26], usually negotiate at the stock exchange a special discount rate for the transaction costs.

13. It may seem, as suggested by one of the anonymous reviewers, that the length of the test period for each time-window position (5 days) is too short and may result in unstable overall trading results. However, the stability of the results is not lost because the training sets in neighboring time-window positions are *highly overlapping*. This means that there will be little difference between deterministically constructed models (classes MM, PST, FCM) built in neighboring time-window positions. There is a potential for instability in the class NN, since the neural networks are randomly initiated, but the model construction instability is decreased by using *committees* of networks as opposed to predicting with a single neural network.

    To support our arguments, we run a new set of experiments with time-windows containing test sets of length 20 (1 month) and 40 (2 months). Note that by prolonging the test period, the number of experimental time-window positions is reduced (see figure 3). The predictive model classes considered in the new experiments were the Markov Models (MM) and neural networks (NN). Com-

[26] The actual invested amount of money is not specified in the experiments, since we report our profits in *percentages* of the invested money.

pared with our original (5-day-test-set) experiments, standard deviations of the block-profits are roughly the same. In fact they are slightly higher in the DAX experiment (0.6 – 0.8) and in a wider range in the FTSE 100 experiment (0.6 – 1.2). In the DAX experiment, the mean profits significantly decreased with increasing test set length. This pulled the highest affordable transaction costs (TC) to the negative range, even for the compound models using the class *Simple*. In the FTSE 100 experiment, the mean profits were in the same range as in our original experiments.

7 Conclusion

We considered a realistic trading setting, where straddles are traded based on predictions of differences of daily volatilities of the underlying. When dealing with daily financial time series, typically, one has to cope with relatively short training sets. In two independent and controlled experiments, we showed that in such cases, quantization of real-valued sequences can help to analyze patterns underlying the evolution of the system, which could otherwise be masked by large amounts of noise and/or few dominant outliers. We proposed a data-driven parametric quantization scheme. Using fewer symbols (quantization levels) is preferable. In general, symbolic Markov models (MMs) were able to achieve larger profits than the models operating on real-valued sequences. The model order of MMs was selected on an independent validation set.

To deal with non-stationarity in the daily data we used the sliding window technique and proposed to add a special class of fixed, simple symbolic predictive models. Such models avoid using older, and potentially misleading parts of the sliding window devoted to the training data. We presented two techniques for incorporating these simple predictors into more sophisticated models that are fitted on the training set. This approach proved to be of great benefit for most of the studied model classes.

We also studied the memory structure in the time series of daily volatility differences using a novel variation on our own fractal-based symbolic predictive models introduced in [1] [16]. Such models use deep memory in an efficient way. Two memory regimes, characterized by deep and shallow memory, seem to dominate the studied financial time series.

Our experiments show that market makers involved in trading options on the DAX and the FTSE 100 can generate, in an automatic way, an excess profit. However, for non-members of the exchange the option markets on the DAX and the FTSE 100 are efficient. The option market for the DAX seems more efficient than that for the FTSE 100.

Except for the GARCH models, the predictive model classes used in this paper have a bounded (although potentially long) memory. We are currently investigating the trading potential of models with theoretically unbounded memory, such as recurrent neural networks.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) within the research project "Adaptive Information Systems and Modeling in Economics and Management Science" (SFB 010). The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Science and Transport. Many thanks go to R. Tompkins for valuable discussions on trading options and for suggesting a regime-shifting model for predicting the daily volatility differences. We are thankful to the anonymous reviewers for many useful suggestions that helped to improve the paper presentation.

References

[1] Tiňo P, and Dorffner G. Building predictive models from fractal representations of symbolic sequences. In Advances in Neural Information Processing Systems 12. MIT Press, 2000, pp. 645-651.

[2] Giles CL, Lawrence S, and Tsoi AC. Rule inference for financial prediction using recurrent neural networks. In Proceedings of the conference on Computational Intelligence for Financial Engineering, New York City, NY. 1997, pp. 253-259.

[3] Giles CL, Lawrence S, Tsoi AC. Noisy time series prediction using a recurrent neural network and grammatical inference. Machine Learning, 2000, accepted.

[4] Crutchfield JP, and Young C, Computation at the onset of chaos. In: Zurek WH (ed). Complexity, Entropy, and the physics of Information, SFI Studies in the Sciences of Complexity, vol 8. Addison-Wesley, 1990, pp. 223-269.

[5] Katok A, and Hasselblatt B. Introduction to the modern theory of dynamical systems. Cambridge University Press, 1995.

[6] Buhlmann P, Wyner AJ. Variable length markov chains. Annals of Statistics 1999; 27: 480-513.

[7] Papageorgiou CP. High frequency time series analysis and prediction using markov models. In Proceedings of the conference on Computational Intelligence for Financial Engineering, New York City, NY. 1997, pp. 182-185.

[8] Papageorgiou CP. Mixed memory markov models for time series analysis. In Proceedings of the conference on Computational Intelligence for Financial Engineering, New York City, NY. 1998, pp. 165-170.

[9] Apte C, and Hong SJ. Predicting equity returns from securities data. In Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds). Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1994, pp. 541-560.

[10] Buhlmann P. Extreme events from return-volume process: a discretization approach for complexity reduction. Applied Financial Economics 1998; (8): 267-278.

[11] Saul LK, and M.I. Jordan MI. Mixed memory markov models. In Proceedings of the 6th International Workshop on Artificial Intelligence and Statistics, Fort Lauderdale, Florida. 1998.

[12] Kohonen T. The self–organizing map. Proceedings of the IEEE 1990; 78(9): 1464-1479.

[13] Kohavi R, and Sahami M. Error-based and entropy-based discretization of continuous features. In Simondis E, Han J, Fayyad U (eds). Proceedings of the Second International Conference on Knowledge Discovery in Databases. AAAI Press, 1996, pp. 114-119.

[14] Bengio Y, Simard P,, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 1994; 5(2): 157-166.

[15] Ron P, Singer Y, and Tishby N. The power of amnesia In Advances in Neural Information Processing Systems 6. Morgan Kaufmann, 1994, pp. 176-183.

[16] Tiňo P, Dorffner G. Predicting the future of discrete sequences from fractal representations of the past. Machine Learning 2001, in print.

[17] Singh S. A long memory pattern modeling and recognition system for financial forecasting. Pattern Analysis and Applications 1999; 2(3): 264-273.

[18] Singh S. Pattern modeling in time-series forecasting. Cybernetics and Systems – An International Journal 2000, 31(1): 49-66.

[19] van Biljon D. Predicting stock market prices with hidden markov models. M.S. thesis, Department of Computer Science, University of Stellenbosch, Stellenbosch, South Africa. 1999.

[20] Gregoir S, and Lenglart F. Measuring the probability of a business cycle turning point by using a multivariate qualitative hidden markov model. Tech. Rep. DP 9848, Economics Research Library, University of Minnesota. 1988.

[21] Noh J, Engle RF, Kane A. Forecasting volatility and option prices of the s&p 500 index. Journal of Derivatives 1994 (3): 17-30.

[22] Bollerslev T. A generalized autoregressive conditional heteroscedasticity. Journal of Econometrics 1986; 31: 307-327.

[23] Dockner EJ, Strobl G, and Lessing A. Volatility forecasts and the profitability of option trading strategies. Tech. Rep., University of Vienna, Austria. 1988.

[24] White H. Economic prediction using neural networks: the case of ibm daily stock returns. In Proceedings of the 2nd Annual IEEE Conference on Neural Networks II. 1988, pp. 451-458.

[25] LeBaron B. Technical trading rule profitability and foreign exchange intervention. Journal of International Economics 1999; 49: 125-143.

[26] Guyon I, and Pereira F. Design of a linguistic postprocessor using variable memory length markov models. In International Conference on Document Analysis and Recognition, Monreal, Canada. IEEE Computer Society Press, 1995, pp. 454-457.

[27] Ron D, Singer Y, Tishby N. The power of amnesia. Machine Learning 1996; 25: 117-150.

[28] Weinberger MJ, Rissanen JJ, Feder M. A universal finite memory source. IEEE Transactions on Information Theory 1995; 41(3): 643-652.

[29] Rissanen J. A universal data compression system. IEEE Transactions on Information Theory 1983; 29(5): 656-664.

[30] Tiňo P. Spatial representation of symbolic sequences through iterative function system. IEEE Systems, Man, and Cybernetics, part A: Systems and Humans 1999; 29(4): 386-392.

[31] Barnsley MF. Fractals everywhere. Academic Press, New York, 1988.

[32] Duda RO and Hart PE. Pattern Classification and Scene Analysis. Wiley, New York, NY, 1973.

[33] Tiňo P, Schittenkopf C, Dorffner G, and Dockner EJ. A symbolic dynamics approach to volatility prediction. In Abu-Mostafa YS, LeBaron B, Lo AW, Weigend AS (eds). Computational Finance. MIT Press, Cambridge, MA, 2000, pp. 137-151

[34] Pesaran MH, and Timmermann A. Model instability and choice of observation window. Tech. Rep. 99–19, Cambridge University, Cambridge, UK, 1999.

[35] Mann HB, Whitney DR. On a test of whether one of two random variables is stochastically larger than the other. Annals Mathematical Statistics 1047; 18.

[36] Morgan Guaranty Trust. RiskMetrics - Technical Document. Morgan Guaranty Trust Company Global Research, New York, 3rd edition, 1995.

[37] Boudoukh J, Richardson M, Whitelaw RF. Investigation of a class of volatility estimators. Journal of Derivatives 1997; 1: 63-71.

[38] Figlewski S. Forecasting volatility. Financial Markets, Institutions & Instruments 1997; 6(1): 1-88.

[39] Schmitt C, and Kaehler J. Delta-neutral volatility trading with intra-day prices: an application to options on the Dax. Tech. Rep. No. 96-25, Zentrum für Europäische Wirtschaftsforschung, 1996.

[40] Black F, Scholes M. The pricing of options and corporate liabilities. Journal of Political Economy 1973; 81: 637-654.

[41] Tompkins R. Options explained, Macmillan Press, 1994.

A Option Pricing

A holder of a European *call option* on an underlying asset S (e.g. a stock) has the right, but not the obligation, to *buy* the *underlying* (asset) at a specified price K (the *strike price*) at a specified time T in the future (at maturity). The strike price K and the maturity time T of the option are fixed when the option contract is signed at time $t \leq T$.

If the price S_T of the underlying at the maturity time T is larger than the agreed strike price K, the option holder will exercise his right to buy the underlying, thereby realizing a profit of $S_T - K > 0$. If $S_T < K$, his option is worthless. Therefore, the option payoff at the maturity time T is given by $\max\{S_T - K, 0\}$.

The fair price $\mathcal{C}$ of a call option depends on the assumptions about the stochastic process driving the price S_t of the underlying asset S till maturity, i.e. for $t \leq T$. Black and Scholes [40] pioneered the field of option pricing by deriving a closed-form expression for the fair price $\mathcal{C}$ of a European call option. They assumed that the price process S_t of the underlying asset S follows a geometric Brownian motion with volatility σ. Although real market dynamics are much more complicated, their famous formula

$$\begin{aligned} \mathcal{C} &= S_t N(d_1) - K e^{-r(T-t)} N(d_1 - \sigma\sqrt{T-t}), && (8) \\ d_1 &= \frac{\ln \frac{S_t}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}, && (9) \\ N(d) &= \int_{-\infty}^{d} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx, && (10) \end{aligned}$$

where r denotes the riskless interest rate, is still applied by many traders at financial markets around the world. All variables in the Black-Scholes formula are explicitly known except for the volatility σ which must be estimated from the market data.

A European *put option* gives the option holder the right to *sell* the underlying asset S at maturity time T for a specified (strike) price K, thereby yielding a profit $\max\{K - S_T, 0\}$. The option is thus worthless, if the asset price at maturity, S_T, is higher than the strike price K. In the Black-Scholes world, the fair price $\mathcal{P}$ of a put option is given by

$$\mathcal{P} = \mathcal{C} + K e^{-r(T-t)} - S_t, \tag{11}$$

where $\mathcal{C}$ is the price of the call option with the same strike price K and the same maturity time T.

Depending on the value S_t of the underlying asset S at time $t < T$ and the agreed strike price K, a *call* option is called *in-the-money* and *out-of-the-money*, if $S_t > K$ and $S_t < K$, respectively. A *put* option is called *in-the-money* and *out-of-the-money*, if $S_t < K$ and $S_t > K$, respectively. An option is called *at-the-money*, if $S_t = K$.

A combination of a call option and a put option with the same strike price K and maturity time T is called a *straddle* on the underlying asset S, and its price is given by $\mathcal{S} = \mathcal{C} + \mathcal{P}$. If a straddle is *at-the-money*, which roughly means that $S_t = K$, it is *delta-neutral*, i.e., small price changes ΔS_t of the underlying asset S have a negligible effect on the straddle price $\mathcal{S}$. In other words, the owner of the straddle is *not* exposed to market risk (with respect to price fluctuations). However, the straddle price *does* depend on changes in the volatility σ in prices S_t of the underlying asset S (the larger σ, the larger $\mathcal{S}$). Indeed, the sensitivity of straddle prices with respect to volatility σ is maximal for at-the-money straddles. Therefore, buying (selling) at-the-money straddles if volatility is expected to increase (decrease), is a *volatility trading strategy* which is mainly independent of the direction of price changes (see [41] for more details).

One of the popular approaches to estimating the (inherently unobservable) volatility σ from historical data of the underlying asset prices S_t is to keep track of the prices of options on the underlying asset S that appeared in the past and then invert an option pricing model (such as the Black-Scholes model given by eqs. (8) – (10)) to calculate the volatility parameter σ. Volatility estimated in this manner is known as *implied volatility*. The volatility measure used in this study is the implied volatility estimated by inversion of the Black-Scholes model. In the Black-Scholes world, the volatility σ is assumed to be constant and therefore one should obtain the same value of the implied volatility for options with different strike prices. In reality, however, volatility is not constant but often has the shape of a "smile", if plotted against the strike prices. One way to deal with this inconsistency is to average the implied volatilities over a range of strike prices. For the DAX, the average value of the implied volatilities of the first in-the-money and the first out-of-the money call and put option is calculated. For the FTSE 100, the average value of the implied volatilities of the call and put option closest to the at-the-money point is used. So the average value of 4 (2) implied volatilities is calculated for the DAX (FTSE 100) every day.

The definition of an *American* option is similar to the definition of a European option. The only difference is that the owner has the right to exercise his option whenever he or she wants, i.e., he or she can exercise the option *before* maturity time T. The pricing of American options is more difficult but one can use the prices of the corresponding European options as a rough approximation.

Figure legends

Figure 1: Series of daily returns of the financial indexes DAX and FTSE 100.

Figure 2: An illustration of the experimental setup used in the DAX and FTSE 100 experiments.

Figure 3: Cumulative profits gained by the class *MM(10)* of Markov models of maximal order 10, the class *Simple*, and the combination *Comb-LR* of *MM(10)* and *Simple*. The profits are expressed in percentage of invested money.

Figure 4: Characteristics of representatives of the binary model classes *MM(5)*, *MM(10)*, and *Simple* in the DAX experiment. For trading day we plot the MM order (upper graph), and the selected strategy from *Simple* (lower graph). The strategies *always Up*, *always Down*, *reverse last*, *copy last* are denoted by U, D, R and C, respectively.

Table 1: Profits of the symbolic models operating on the binary sequences in the DAX and FTSE 100 experiments. Combined strategies are referred to by the employed selection method. Combined strategies using simply the validation set performances and those using the slopes of linear regression on the past validation set performances are referred to as `Comb-Val` and `Comb-LR`, respectively. For comparison, we also report profits that would be made by the always-correct-predictor, `ACP`, perfectly predicting all the volatility changes. To make a more realistic assessment of the achieved profits, we also report the maximal transaction costs (TC) that one can subtract from each daily profit, so that when subjected to the t-test (p=0.05), the average block-profits are still significantly positive.

	DAX			FTSE 100		
model class	% profit per-day Mean	Std.	highest TC for signif. pos. mean	% profit per-day Mean	Std.	highest TC for signif. pos. mean
ACP	1.310	0.652	1.034	2.706	1.109	2.132
Simple	0.423	0.521	0.21	1.676	1.222	1.05
MM(5)	0.262	0.603	0.01	1.551	0.833	1.12
Comb-LR	0.497	0.485	0.30	1.611	0.884	1.16
Comb-Val	0.430	0.458	0.24	1.551	0.833	1.12
MM(10)	0.208	0.668	–	1.490	0.894	1.03
Comb-LR	0.473	0.613	0.22	1.550	0.942	1.07
Comb-Val	0.425	0.578	0.19	1.489	0.894	1.03
PST(1)	0.013^-	0.584	–	0.703^-	1.312	0.03
Comb-LR	0.307^-	0.535	0.08	1.604^+	1.235	0.97
Comb-Val	0.449^+	0.509	0.23	1.640^+	1.195	1.03
PST(50)	-0.013^-	0.568	–	1.493	0.878	1.04
Comb-LR	0.394^+	0.479	0.19	1.405	0.826	0.98
Comb-Val	0.421^+	0.509	0.20	1.485	0.876	1.04
FCM	-0.243^-	0.451	–	0.568^-	0.918	0.10
Comb-LR	0.371^+	0.417	0.19	1.644^+	0.642	1.30
Comb-Val	0.471^+	0.404	0.30	1.538^+	1.055	1.00

Table 2: Profits and affordable transaction costs (TC) of the symbolic models operating on the sequences over the four-symbol alphabet in the DAX and FTSE 100 experiments (implied volatility).

	DAX			FTSE 100		
model class	% profit per-day Mean	Std.	highest TC for signif. pos. mean	% profit per-day Mean	Std.	highest TC for signif. pos. mean
MM(5)	0.043^{-}	0.679	–	1.381	1.357	0.68
Comb-LR	0.335	0.604	0.08	0.959^{-}	1.094	0.40
Comb-Val	0.311	0.592	0.07	1.381	1.357	0.68
PST(1)	0.172	0.671	–	0.541^{-}	0.592	0.24
Comb-LR	0.507	0.528	0.28	0.967^{-}	0.903	0.50
Comb-Val	0.537	0.583	0.29	1.138^{-}	1.126	0.56
PST(50)	-0.008^{-}	0.736	–	1.012^{-}	0.958	0.52
Comb-LR	0.275^{-}	0.597	0.02	1.122	1.135	0.54
Comb-Val	0.307^{-}	0.616	0.04	1.044	0.944	0.56
FCM	-0.205^{-}	0.490	–	0.385^{-}	1.233	–
Comb-LR	0.351^{+}	0.499	0.14	1.038^{-+}	1.013	0.52
Comb-Val	0.414^{+}	0.514	0.19	1.125^{-+}	0.942	0.64

Table 3: Profits and affordable transaction costs (TC) of the models operating on the real-valued sequences in the DAX and FTSE 100 experiments (implied volatility).

	DAX			FTSE 100		
model class	% profit per-day Mean	Std.	highest TC for signif. pos. mean	% profit per-day Mean	Std.	highest TC for signif. pos. mean
NN	0.0326^{-}	0.576	–	1.331	1.095	0.77
Comb-LR	0.397	0.599	0.15	1.562	1.121	0.99
Comb-Val	0.405^{+}	0.554	0.18	1.432	1.131	0.85
NN-symb	0.123	0.773	–	1.541	0.777	1.14
Comb-LR	0.410	0.574	0.17	1.470	0.973	0.97
Comb-Val	0.375	0.550	0.15	1.429	1.033	0.90
GARCH	$0.079^{(-)}$	0.420	–	0.092	1.394	–
Comb-LR	0.408	0.671	0.13	0.116	1.056	–
Comb-Val	$0.437^{(+)}$	0.684	0.15	0.175	1.123	–