Predicting the future of discrete sequences from fractal representations of the past

Peter Tiňo[*] **Georg Dorffner**[†]

Austrian Research Institute for Artificial Intelligence

Schottengasse 3

A-1010 Vienna, Austria

Phone: +43-1-5336112-15

Email: petert,georg@ai.univie.ac.at

Abstract

We propose a novel approach for building finite memory predictive models similar in spirit to variable memory length Markov models (VLMMs). The models are constructed by first transforming the n-block structure of the training sequence into a geometric structure of points in a unit hypercube, such that the longer is the common suffix shared by any two n-blocks, the closer lie their point representations. Such a transformation embodies a Markov assumption – n-blocks with long common suffixes are likely to produce similar continuations. Prediction contexts are found by detecting clusters in the geometric n-block representation of the training sequence via vector quantization. We compare our model with both the classical (fixed order) and variable memory length Markov models on five data sets with different memory and stochastic components. Fixed order Markov models (MMs) fail on three large data sets on which the advantage of allowing variable memory length can be exploited. On

[*]also with *the Department of Computer Science and Engineering, Slovak University of Technology, Ilkovicova 3, 812 19 Bratislava, Slovakia*

[†]also with *the Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, Freyung 6/2, A-1010 Vienna, Austria*

these data sets, our predictive models have a superior, or comparable performance to that of VLMMs, yet, their construction is fully automatic, which, is shown to be problematic in the case of VLMMs. On one data set, VLMMs are outperformed by the classical MMs. On this set, our models perform significantly better than MMs. On the remaining data set, classical MMs outperform the variable context length strategies.

Keywords: variable memory length Markov models, iterative function systems, fractal geometry, chaotic sequences, DNA sequences, volatility prediction

Running head: Predicting discrete sequences from fractal representations of the past

1 Introduction

Statistical modeling of complex sequences is a fundamental goal of machine learning due to its wide variety of applications (Ron, Singer, & Tishby, 1996): in genetics (Prum, Rodolphe, & deTurkheim, 1995), speech recognition (Nadas, 1984), finance (Bühlmann, 1998), or seismology (Brillinger, 1994).

One of the models for sequences generated by stationary sources, assuming no particular underlying mechanistic system, are Markov models (MMs) of finite order (Bühlmann, & Wyner, 1999). The only implicit assumption made is about the finite memory of the process. These statistical models define rich families of sequence distributions and give efficient procedures for both generating sequences and computing their probabilities. However, MMs can become very hard to estimate due to the familiar explosive increase in the number of free parameters (yielding highly variable estimates) when increasing the model order. Consequently, only low order MMs can be considered in practical applications.

Approaches proposed in the literature (Ron, Singer, & Tishby, 1996; Laird, & Saul, 1994; Nadas, 1984; Rissanen, 1983; Weinberger, Rissanen, & Feder, 1995; Willems, Shtarkov, & Tjalkens, 1995; Bühlmann, & Wyner, 1999) to overcome the curse of dimensionality in MMs share the same basic idea: instead of fixed order MMs consider variable memory length Markov models (VLMMs) with a "deep" memory just where it is really needed (Ron, Singer, & Tishby, 1994).

Prediction contexts of variable length in VLMMs are often represented as prediction suffix trees (PSTs) (Rissanen, 1983). The relevant prediction context is defined as the deepest node in the PST that can be reached from the root when reading the input stream in reversed order.

Prediction suffix trees can be constructed in a top-down (Ron, Singer, & Tishby, 1994; Ron, Singer, & Tishby, 1996; Weinberger, Rissanen, & Feder, 1995), or bottom-up (Guyon, & Pereira, 1995; Bühlmann, & Wyner, 1999) fashion. Both schemes strongly depend on the construction parameters regulating candidate context selection and growing/pruning decisions (Bühlmann, & Wyner, 1999; Guyon, & Pereira, 1995). The appropriate values

for those parameters are derived only under asymptotic considerations. In practical applications, the parameters must be set by the modeler, which can be, as we will see, quite inconvenient and problematic (see also (Bühlmann, 2000)). Bühlmann and Wyner (1999) suggest to optimize the construction parameters' values through minimization of model complexity measured, for example, by the Akaike information criterion (Akaike, 1974). In another study on VLMM model selection (Bühlmann, 2000), Bühlmann proposes a resampling strategy to estimate the asymptotic behavior of different risk functions. However, in practical applications, such a strategy may not be applicable, since fitting the individual VLMMs can be highly time-consuming.

We introduce finite-context predictive models similar in spirit to VLMMs. The key idea behind our approach is a geometric representation of candidate prediction contexts, where contexts with long common suffixes (i.e. contexts that are likely to produce similar continuations) are mapped close to each other, while contexts with different suffixes (and potentially different continuations) correspond to points lying far from each other. Selection of the appropriate prediction contexts is left to a vector quantizer. Dense areas in the spatial representation of potential prediction contexts correspond to contexts with long common suffixes and are given more attention by the vector quantizer.

The paper has the following organization:

In sections 2 and 3, we use the framework of finite memory sources to introduce our predictive models as well as the classical and variable memory length Markov models.

Section 4 contains a detailed comparison of the studied model classes on five data sets of different origin, representing a wide range of grammatical and statistical structure.

A discussion summarizes the empirical results and outlines directions in our current and future research.

2 Statistical modeling of complex sequences

We consider sequences $S = s_1 s_2 ...$ over a finite alphabet $\mathcal{A} = \{1, 2, ..., A\}$ (i.e. every symbol s_i is from $\mathcal{A}$) generated by stationary information sources (Khinchin, 1957). 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 $\hat{P}_n(w) > 0$. The set of all allowed n-blocks in S is denoted by $[S]_n$.

An information source (Khinchin, 1957; Weinberger, Rissanen, & Feder, 1995) over an alphabet $\mathcal{A} = \{1, 2, ..., A\}$ is defined by a family of probability measures P_n on n-blocks over $\mathcal{A}$, $n = 0, 1, 2, ...$ Consistent measures satisfy the marginality condition: for all[1] $s \in \mathcal{A}$, $w \in \mathcal{A}^n$, $n = 0, 1, 2...$,

$$\sum_{s \in \mathcal{A}} P_{n+1}(ws) = P_n(w).$$

In applications it is useful to consider probability functions P_n that are both consistent and easy to handle. This can be achieved, for example, by assuming a finite source memory of length at most L, and formulating 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* $c : \mathcal{A}^L \to \mathcal{C}$, 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 to estimate the probability distribution $P(s|w)$ from the data. On one hand such a finite memory model can be used for prediction. On the other hand, it can also be used as a sequence generator by initiating it with the first L-block and letting it produce a continuation according to the next-symbol distribution (1).

We now present two specific examples of finite memory learners and then introduce our novel approach for constructing finite memory sources from geometric representations of training sequences.

[1] $\mathcal{A}^0 = \{\Lambda\}$ and $P_0(\Lambda) = 1$, where Λ denotes the empty string.

2.1 Fixed-order Markov models

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. Formally, the context function $c : \mathcal{A}^L \to \mathcal{C}$ for Markov models (MMs) of order $n < L$ can be interpreted as a natural homomorphism $c : \mathcal{A}^L \to \mathcal{A}^L|_{\mathcal{E}}$ corresponding to an equivalence relation $\mathcal{E} \subseteq \mathcal{A}^L \times \mathcal{A}^L$ on L-blocks over $\mathcal{A}$: $(u, v) \in \mathcal{E}$, if the L-blocks u, v share the same suffix of length n. The factor set $\mathcal{A}^L|_{\mathcal{E}}$, i.e. the set of all equivalence classes on L-blocks $\mathcal{A}^L$ under the equivalence $\mathcal{E}$, consists of all n-blocks over $\mathcal{A}$,

$$\mathcal{A}^L|_{\mathcal{E}} = \mathcal{C} = \mathcal{A}^n.$$

As already mentioned in the introduction, 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 of probability distributions to be estimated rises by A^n leaving the learner with the problem to cope with a strong curse of dimensionality.

2.2 Variable length Markov models

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

$$\hat{P}(s|w) = \frac{\hat{P}_{n+1}(ws)}{\hat{P}_n(w)}.$$

If for a symbol $a \in \mathcal{A}$, such that $aw \in [S]_{n+1}$, the prediction probability of the next symbol s,

$$\hat{P}(s|aw) = \frac{\hat{P}_{n+2}(aws)}{\hat{P}_{n+1}(aw)},$$

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, one can 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 (Ron, Singer, & Tishby, 1994; Guyon, & Pereira, 1995),

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

- there exists a symbol $s \in \mathcal{A}$, such that (Ron, Singer, & Tishby, 1996)

$$\hat{P}(s|aw) \geq \frac{1}{A}(1+\epsilon_1)\epsilon_1 \quad \text{and} \quad \frac{\hat{P}(s|aw)}{\hat{P}(s|w)} > 1 + 3\epsilon_1. \tag{3}$$

The (small, positive) construction parameters ϵ_{KL}, ϵ_1 are supplied by the modeler. For other variants of decision criteria see (Weinberger, Rissanen, & Feder, 1995; Bühlmann, & Wyner, 1999).

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) (Ron, Singer, & Tishby, 1996; Rissanen, 1983). The edges of 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 PST are labeled by pairs $(s, \hat{P}(s|v))$, $s \in \mathcal{A}$, $v \in \mathcal{A}^+$, where v is a 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[2] (Ron, Singer, & Tishby, 1996; Ron, Singer, & Tishby, 1994; Guyon, & Pereira, 1995):

- 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:
  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) (or (3)) holds[3]
  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) (or (3)), 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} (ϵ_1) decrease, the size of the constructed PST increases.

Prediction suffix trees are usually constructed using a one-parameter scheme introduced in (Ron, Singer, & Tishby, 1994). 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 can 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. We empirically found the procedure with ratio-related parameters $\epsilon_{grow} = \rho\ \epsilon_{KL}$, $50 \leq \rho \leq 100$, to give the best results.

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

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

Variable memory length Markov models (VLMMs) are usually compactly described as stochastic machines (SMs). Briefly, SMs are like finite state machines except that the state transitions take place with probabilities prescribed by a distribution $T_{i,j,s}$. The generating process is started in an initial state and then, at any given time step, the machine is in some state i, and at the next time step moves to another state j outputting some symbol s, with the transition probability $T_{i,j,s}$.

The set $\mathcal{C}$ of prediction contexts encoded in a PST is the state set of the corresponding SM that contains the leaves of the PST plus contexts added so that the symbol driven state transition probabilities $T_{i,j,s}$ are properly defined (see (Ron, Singer, & Tishby, 1996; Ron, Singer, & Tishby, 1994; Guyon, & Pereira, 1995)). SMs representing VLMMs have suffix-free state sets Q and are known as probabilistic suffix automata (PSA) (Ron, Singer, & Tishby, 1996; Weinberger, Rissanen, & Feder, 1995). Although VLMMs can be emulated with the corresponding PSTs, PSA representations of VLMMs give higher processing speed. In PSA, the longest suffices are precomputed into states, whereas in PSTs the longest suffices must be dynamically determined (Guyon, & Pereira, 1995).

3 Fractal prediction machines

We propose a novel approach for learning the statistical structure of symbolic sequences, which we call *fractal prediction machines* (FPMs). FPMs are similar in spirit to VLMMs, but derive a context function $c(w)$ in a more efficient way.

The main idea behind a FPM is to first transform the L-blocks appearing in the training sequence into points in a D-dimensional vector metric space $(\Re^D, d)$, so that the suffix structure of L-blocks is coded into a cluster structure in $(\Re^D, d)$. The equivalence relation $\mathcal{E}$ defining the context function is then constructed by vector-quantizing the geometric representations of allowed L-blocks. This way, we have a direct control over the number of predictive contexts and, at the same time, avoid using auxiliary construction parameters employed in the PST construction (see the last section).

3.1 Chaos game representations

The basis for the transformation of symbolic strings into points in $\Re^D$ is the so-called chaos game representation (CGR), originally introduced by Jeffrey (1990) to study DNA sequences (see also (Oliver, Galván, Garcia, & Roldan, 1993; Roldan, Galván, & Oliver, 1994; Li, 1997)). CGRs of symbolic sequences have been formally studied in (Tiňo, 1999) revealing the desired properties for our purposes.

The basis of the chaos game representation of sequences over an alphabet $\mathcal{A} = \{1, 2, ..., A\}$ is an iterative function system (IFS) (Barnsley, 1988) consisting of A affine contractive maps[4] $1, 2, ..., A$, acting on the D-dimensional unit hypercube[5] $X = [0, 1]^D$, $D = \lceil \log_2 A \rceil$:

$$i(x) = kx + (1 - k)t_i, \quad t_i \in \{0, 1\}^D, \quad t_i \neq t_j \text{ for } i \neq j. \tag{4}$$

The contraction coefficient of the maps $1, ..., A$, is $k \in (0, \frac{1}{2}]$.

The chaos game representation $CGR_k(S)$ of a sequence $S = s_1 s_2 ...$ over $\mathcal{A}$ is obtained as follows (Tiňo, 1999):

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

2. Plot the point $x_n = j(x_{n-1}),\ n \geq 1$, provided the n-th symbol s_n is j.

As an example, consider a sequence $S = 142...$ over the four-symbol alphabet $\mathcal{A} = \{1, 2, 3, 4\}$. Let the four affine maps on the unit square $[0, 1]^2$, 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

[4] 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.

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

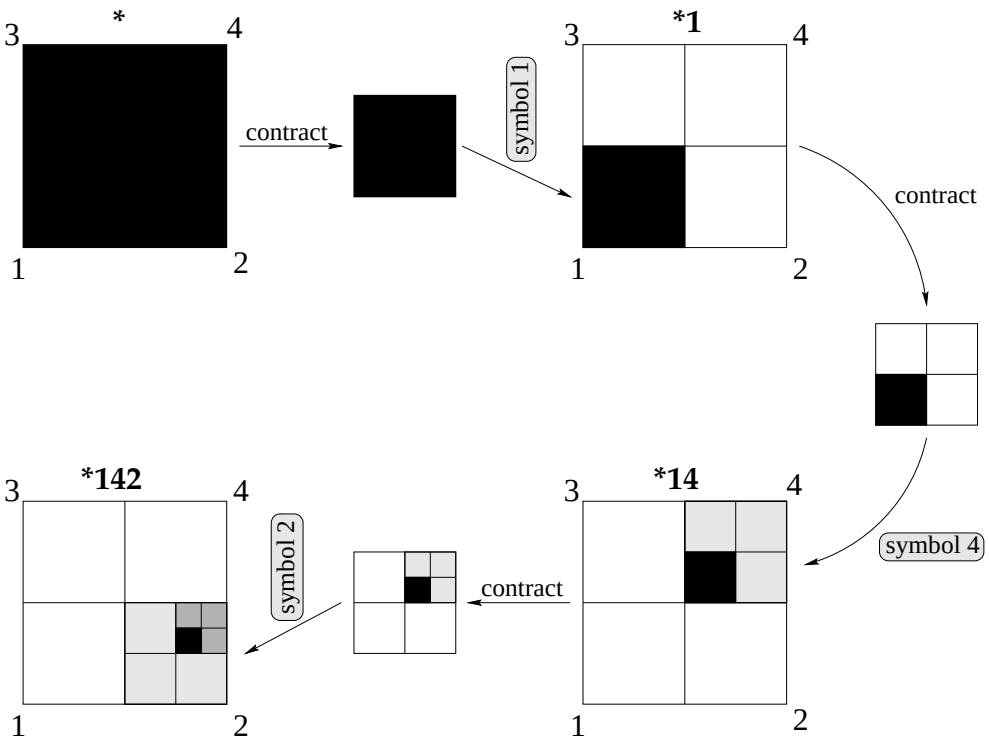

Figure 1: Illustration of the iterative function system behind the chaos sequence representations of symbolic streams. Each symbol 1, 2, 3 and 4 is associated with a unique corner of the black unit square at the top left. Upon seeing symbol 1, the unit square is contracted and shifted towards to corner 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.

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. 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 whole process can be iteratively repeated. Assume that the next symbol 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 yet another symbol, say, 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, etc...

Note that by iteratively making contractions and shifts, we effectively code the history of 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 to all strings ending with 142.

Two properties of the chaos game representation $CGR(S)$ of symbolic sequences S are of importance to us. First, if histories of the last symbols in two sequences S_1, S_2, are the same, i.e. if the sequences S_1, S_2 share a common suffix, the last points in the representations, $CGR(S_1)$, $CGR(S_2)$, lie close to each other. Second, the longer is the common suffix shared by S_1 and S_2, the smaller is the region containing the last points of $CGR(S_1)$, $CGR(S_2)$.

3.2 Deriving an appropriate context function

We slightly modify the concept of chaos game representations to compute a *chaos L-block representation* $CBR_{L,k}(S)$ of the sequence S. It is constructed by plotting only the last points of the chaos game representations $CGR_k(w)$ of allowed L-blocks $w \in [S]_L$. The representation of a single block, resulting in a single point, is defined by the map $\sigma : \mathcal{A}^L \to [0,1]^D$, from L-blocks $v_1 v_2 ... v_L$ over $\mathcal{A}$ to the unit hypercube,

$$\sigma(v_1 v_2 ... v_L) = v_L(v_{L-1}(...(v_2(v_1(x^*)))...)) = (v_L \circ v_{L-1} \circ ... \circ v_2 \circ v_1)(x^*), \qquad (5)$$

where $x^* = \{\frac{1}{2}\}^D$ is the center of the hypercube. The maps $v_1, ..., v_L$, corresponding to symbols appearing in L-blocks are defined in (4).

We thus obtain the (multi)set of points $CBR_{L,k}(S)$ in $\Re^D$ containing the geometric representations of allowed L-blocks in S. The set $CBR_{L,k}(S)$ codes the suffix structure in allowed L-blocks in the following sense (Tiňo, 1999): if $v \in \mathcal{A}^+$ is a suffix of length $|v|$ of a string $u = rv$, $r, u \in \mathcal{A}^+$, then $u(X) \subset v(X)$, where $v(X)$ is a D-dimensional hypercube of side length $k^{|v|}$. Hence, the longer is the common suffix shared by two L-blocks, the closer the L-blocks are mapped in the chaos L-block representation $CBR_{L,k}(S)$[6]. On the other hand, the Euclidean distance between points representing two L-blocks u, v, that have the same prefix of length $L-1$ and differ in the last symbol, is at least $1-k$.

Given this property, finding an appropriate context function can easily be done by performing vector quantization (VQ) on the chaos L-block representation $CBR_{L,k}(S)$ of the training sequence S. VQ in the metric space $(\Re^D, d)$, where d is the metric, positions in $\Re^D$ M codebook vectors (CVs), $b_1, ..., b_M$, each CV representing a subset of points from $CBR_{L,k}(S)$ that are closer to it (w.r.t. metric d) than to any other CV, so that the overall error of substituting CVs for points they represent is minimal. In other words, CVs tend to represent points in $CBR_{L,k}(S)$ lying close to each other (in metric d).

As the distance function d, we consider the L_1 distance

$$d_1(x, y) = \sum_{i=1}^{D} |x_i - y_i|, \tag{6}$$

or the L_2 (Euclidean) distance

$$d_2(x, y) = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2}, \tag{7}$$

where $x = (x_1, x_2, ..., x_D)$, $y = (y_1, y_2, ..., y_D) \in \Re^D$. Compared to the L_1 metric, the L_2 metric is less sensitive to smaller distances, while emphasizing the larger ones. Vector

[6] For k close to $\frac{1}{2}$, geometric representations of completely different L-blocks may lie close to each other. This happens, for example, for blocks 444...41 and 333...32 over the alphabet $\{1, 2, 3, 4\}$, geometrically represented through the iterative function system (4) acting on $[0, 1]^2$, with $t_1 = (0, 0)$, $t_2 = (1, 0)$, $t_3 = (0, 1)$ and $t_4 = (1, 1)$. As a remedy, one may lower the contraction ratio k. The issue of optimal contraction ratio with respect to a given training sequence and vector quantizer is also being currently investigated.

quantization in L_1 and L_2 metrics positions CVs in the median and the mean, respectively, of the set of points they represent.

Now, as with classical Markov models, we define the prediction context function $c : \mathcal{A}^L \to \mathcal{C}$ via an equivalence $\mathcal{E}$ on L-blocks over $\mathcal{A}$. This time, the equivalence $\mathcal{E}$ reads: two L-blocks u, v are in the same class if their images under the map σ (eq. (5)) are represented by the same codebook vector. In this case, the set of prediction contexts $\mathcal{C}$ can be identified with the set of codebook vectors $\{b_1, b_2, ..., b_M\}$. We refer to predictive models with such a context function as *fractal prediction machines* (FPMs)[7]. The prediction probabilities (1) are determined by

$$P(s|b_i) = \frac{N(i,s)}{\sum_{a\in\mathcal{A}} N(i,a)}, \quad s \in \mathcal{A}, \tag{8}$$

where $N(i,a)$ is the number of $(L+1)$-blocks ua, $u \in \mathcal{A}^L$, $a \in \mathcal{A}$, in the training sequence, such that the point $\sigma(u)$ (eq. (5)) is allocated to the codebook vector b_i.

3.3 FPM construction

To summarize what was described above, fractal prediction machines are constructed as follows:

1. calculate the chaos L-block representation $CBR_{L,k}(S)$ of the training sequence $S = s_1 s_2 ... s_m$ containing point representations $\sigma(w) \in \Re^D$ (eq. (5)) of all allowed L-blocks $w \in [S]_L$ in S

2. partition the hypercube $[0,1]^D$ into M regions $V_1, ..., V_M$, by running a vector quantizer on the set $CBR_{L,k}(S)$. The regions V_i, $i = 1, ..., M$, in the metric space

[7] We note that FPMs depend on cluster density in the geometric L-block representations, that is controlled by the contraction parameter k (see eq. (4)). Smaller k's yield more dense clusters. Furthermore, quantization of the geometric representations is controlled by the magnification factor (Ritter, & Schulten, 1986; Bauer, Der, & Herrmann, 1996) of the used vector quantization scheme. The magnification factor relates, under asymptotic considerations, the frequency of codebook vectors in the quantized region with the frequency of L-block representations in that region. One can find a formal relationship among the contraction factor k, magnification factor of the vector quantizer and the dynamics of the FPM context transitions. This and other related issues are currently under investigation.

$(\Re^D, d)$, are the Voronoi compartments (Aurenhammer, 1991) of the codebook vectors $b_1, ..., b_M$,

$$V_i = \{x \in [0,1]^D |\ d(x, b_i) = \min_j d(x, b_j)\}.$$

All points in V_i are allocated[8] to the codebook vector b_i.

3. set the counters $N(i, a)$, $i = 1, ..., M$, $a = 1, ..., A$, to zero

4. for $1 \leq t \leq m - L$

   - code the L-block S_t^{t+L-1} by a point $\sigma\left(S_t^{t+L-1}\right)$
   - if $\sigma\left(S_t^{t+L-1}\right) \in V_i$, increment the counter $N(i, s_{t+L})$ by one

5. with each prediction context (codebook vector) $b_1, ..., b_M$, associate the next symbol probabilities

$$P(s|b_i) = \frac{N(i,s)}{\sum_{a \in \mathcal{A}} N(i,a)}, \quad s \in \mathcal{A}.$$

4 Experiments

We compared the fractal prediction machines (FPMs) with both the classical and variable memory length Markov models referred to as MM and VLMM (or PST, for prediction suffix tree), respectively. The experiments were performed on five data sets of various origin and different levels of subsequence distribution structure. These five data sets comprise the following:

- two classical symbolic sequences studied previously, namely DNA sequences and text sequences from the bible,
- two sequences obtained by quantizing chaotic time series, which have been well-studied and have a known deep and complex structure: quantized Laser data and the Feigenbaum sequence,

[8] Ties as events of measure zero (points land on the border between the compartments) are broken according to index order

- one sequence derived from quantizing a time series from a real world stochastic process, namely the historical Dow Jones industrial average.

By choosing these data sets we aim to demonstrate where and when FPMs can outperform the classical fixed-order and the more flexible variable-order Markov models. At the same time, we demonstrate the feasibility of transforming continuous time series into symbolic streams and subsequently using MMs, VLMMs and FPMs to learn about their structure.

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. Crutchfield & Young, 1990; Katok, & Hausselblatt, 1995).

Analogous ideas in the context of stochastic real-valued time series were recently put forward by Bühlmann (1999). 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 VLMMs constructed on the quantized sequences with the next step distribution in $\Re$ defined as a mixture of local (say, Gaussian) densities corresponding to the individual partition elements (symbols). The mixture weights correspond to the next-symbol probabilities given by the symbolic model (VLMM). Bühlmann (1999) proves two key results. First, 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. Second, given an appropriate partition function into symbols, finding the optimal QVLMC in the maximum likelihood setting can be achieved exclusively by finding the optimal underlying VLMM on the symbolic level. Hence, modeling quantized time series is of great importance. We found the quantization approach very effective in our recent study on financial time series modeling (Tiňo et al., 2000a). See also (Bühlmann, 1998; Giles, Lawrence, & Tsoi, 1997; Papageorgiou, 1998).

4.1 Experimental setup

In all experiments we constructed FPMs using a contraction coefficient $k = \frac{1}{2}$ (see eq. (4)) and K-means clustering (MacQueen, 1967; Buhmann, 1995), in both L_1 and L_2 norms, as a vector quantization tool. PSTs representing VLMMs were constructed using the Kullback-Leibler criterion (eq. (2)).

4.2 DNA – coding vs. non-coding regions

4.2.1 Data and methods

The DNA alphabet consists of four symbols A, C, T and G that, for our purposes, correspond to symbols 1, 2, 3 and 4, respectively. In the first experiment, we classified DNA sequences into coding and non-coding classes. In contrast to non-coding sequences, coding DNA strands contain protein coding genes. Locating the coding genes is a necessary step before any further DNA analysis. For each model class, the classification module consists of two models – a coding expert built on the coding sequences and a non-coding expert built on the non-coding ones. Upon presentation of an unseen DNA sequence, the classification module makes its decision based on the probabilities assigned to the sequence by the two experts.

In DNA sequences, almost all short subsequences are allowed, with a rather uniform subsequence distribution. Among the models studied in this paper, fixed order Markov models should perform well in this experiment.

We collected a large data set of vertebrate DNA sequences[9] used to test gene structure prediction programs (Burset & Guigó, 1996). From the data set, we extracted a portion of 880 coding sequences as the coding training set and a different portion of 880 coding sequences as the coding test set. The same applies to the non-coding sequences. So both the training and test sets consisted of 880 coding and 880 non-coding sequences. The length of sequences ranged from 100 to 20 000.

Maximal memory depth was set to $L = 7 \cdot 3 = 21$ (to account for the triplet structure of

[9] `http://www1.imim.es/GeneIdentification/Evaluation/Index.html`

the coding genes). For each model class and model size, we built two different models, one for the coding regions (constructed on the coding training set), and one for the intergenic regions (constructed on the non-coding training set). We tested the model performance by calculating the normalized negative log-likelihood (NNL) of the two models on each of the test sequences. The model pair classifies a test sequence as coding if the NNL achieved by the coding expert is lower than that of the non-coding expert. Otherwise, the sequence is classified as non-coding.

The likelihood can be calculated as follows. Denote the empirical n-block frequency counts in S by $\hat{P}_n$. Let $\mathcal{M}$ be a finite memory source built on S. The probability that the model $\mathcal{M}$, initiated with the first L-block S_1^L, assigns to the continuation S_{L+1}^m is

$$P_{\mathcal{M}}\left(S_{L+1}^m | S_1^L\right) = \prod_{i=L+1}^{m} P\left(s_i | c\left(S_{i-L}^{i-1}\right)\right) \tag{9}$$

and the likelihood of the sequence S with respect to the model $\mathcal{M}$ is determined as

$$P_{\mathcal{M}}(S) = \hat{P}_L\left(S_1^L\right) P_{\mathcal{M}}\left(S_{L+1}^m | S_1^L\right). \tag{10}$$

The normalized negative log-likelihood[10] is calculated by

$$NNL_{\mathcal{M}}(S) = \frac{-\log_A P_{\mathcal{M}}(S)}{m}. \tag{11}$$

Normalized negative log-likelihood measures the amount of "statistical surprise" induced by the model (Ron, Singer, & Tishby, 1996).

4.2.2 Results

The classification results are summarized in the contingency table containing four items: *true positives* (TP) – the number of coding sequences correctly classified as coding, *true negatives* (TN) – the number of non-coding sequences correctly classified as non-coding, *false positives* (FP) – the number of non-coding sequences incorrectly classified as coding, and *false negatives* (FN) – the number of coding sequences incorrectly classified as non-coding.

[10] base of the logarithm is the number of symbols A in the alphabet $\mathcal{A}$

From the contingency table, four performance measures were calculated:

hit rate (HR) – proportion of correctly classified sequences

$$HR = \frac{TP + TN}{TP + TN + FP + FN},$$

sensitivity (Sen) – proportion of coding sequences correctly classified as coding

$$Sen = \frac{TP}{TP + FN},$$

specificity (Sp) – proportion of non-coding sequences correctly classified as non-coding

$$Sp = \frac{TN}{TN + FP},$$

and *correlation coefficient* (CC) – Pearson product-moment correlation coefficient in the particular case of two binary variables (Burset & Guigó, 1996)

$$CC = \frac{TP \cdot TN - FN \cdot FP}{\sqrt{(TP + FN) \cdot (TN + FP) \cdot (TP + FP) \cdot (TN + FN)}}.$$

CC is an alternative measure of overall prediction accuracy: $CC = 1$ corresponds to perfect prediction, $CC = 0$ is expected for a random prediction.

Classification results are summarized in tables 1 and 2. In this experiment, FPMs perform worse than VLMMs, but VLMMs never achieve the performance of classical MMs. We used McNemar's test (Everitt, 1977) (on 5% level) to test for significance in the model performance differences. PSTs built with the fixed growth strategy ($\epsilon_{grow} = 0.001$) perform always significantly better than FPMs of comparable size. Since the size of PSTs is controlled only indirectly through the construction parameters, the PST experts in coding/non-coding pairs have only approximately the same size. MMs significantly outperform both the L_1 norm and L_2 norm based FPMs, and PSTs built using the one-parameter and ratio $\epsilon_{grow} = 50\ \epsilon_{KL}$ schemes.

Classical MMs are difficult to beat in this experiment, because the suffix structure in the DNA strands is rather uniform. In figure 2 we show geometric representations of L-blocks of both the coding and non-coding training sequences. Compared with geometric representations of L-blocks in the laser or Feigenbaum sequences (figures 6, 8), there is

Table 1: Classification results of FPMs in the DNA experiment. Models were used to classify unseen strings of DNA into coding (positive class) and non-coding (negative class) sequences. Hit rate, sensitivity and specificity are given in percentages. Column **Signif** collects significance results of McNemar's test (on 5% level) applied to pairs of classifiers with comparable number of free parameters: $*$ and $+$ mean that the classifier is significantly worse than the corresponding Markov model and fixed-growth-PST based classifier, respectively; – marks no significance; dots appear where the model pair of the corresponding size does not exist.

model	**# contexts**	**Hit rate**	**Sensitivity**	**Specificity**	**Corr. coef.**	**Signif**
FPM–L_1	1	63.7	66.9	60.4	0.274	$\cdot\ \cdot$
	4	67.7	59.5	75.8	0.358	$*\ \cdot$
	16	76.5	80.6	72.5	0.532	$*\ \cdot$
	64	82.1	82.3	81.9	0.642	$*\ +$
	256	84.9	80.3	89.5	0.701	$*\ \cdot$
	500	85.5	79.9	91.9	0.715	$\cdot\ +$
	750	85.1	76.9	93.4	0.713	$\cdot\ +$
	1024	83.8	74.2	93.4	0.689	$*\ +$
FPM–L_2	4	72.5	74.1	70.9	0.450	$-\ \cdot$
	16	76.5	81.8	71.1	0.533	$*\ \cdot$
	64	81.8	80.7	82.3	0.636	$*\ +$
	256	85.2	80.7	89.6	0.706	$*\ \cdot$
	500	85.0	78.4	91.5	0.705	$\cdot\ +$
	750	84.7	76.9	92.6	0.704	$\cdot\ +$
	1024	84.5	74.5	94.5	0.705	$*\ +$

Table 2: Classification results of MMs and PSTs in the DNA experiment. PSTs constructed using the one-parameter, fixed growth parameter $\epsilon_{grow} = 0.001$, and ratio $\epsilon_{grow} = 50\ \epsilon_{KL}$ schemes are identified by `PST`, `PST-fg`, and `PST(50)`, respectively. Sizes of PST based classifiers are shown as (S_1, S_2), where S_1 and S_2 are the sizes of the coding and non-coding PST experts, respectively. For other details, see caption to the previous table.

model	**# contexts**	**Hit rate**	**Sensitivity**	**Specificity**	**Corr. coef.**	**Signif**
PST	(54,31)	85.3	83.9	86.8	0.707	$* +$
	(910,760)	83.2	72.8	94.6	0.712	$* +$
PST–fg	(56,30)	86.3	84.9	87.9	0.728	–
	(520,410)	87.4	79.9	95.0	0.758	$\cdot$
	(860,840)	88.1	80.4	95.7	0.770	–
PST(50)	(52,32)	86.4	85.2	87.6	0.729	$- -$
	(920,533)	84.9	74.2	95.7	0.716	$* +$
MM	4	73.2	75.8	70.6	0.464	$\cdot$
	16	84.4	86.0	82.8	0.689	$\cdot$
	64	87.1	85.3	88.9	0.742	–
	256	90.0	86.1	94.0	0.803	$\cdot$
	1024	86.9	76.9	96.8	0.752	–

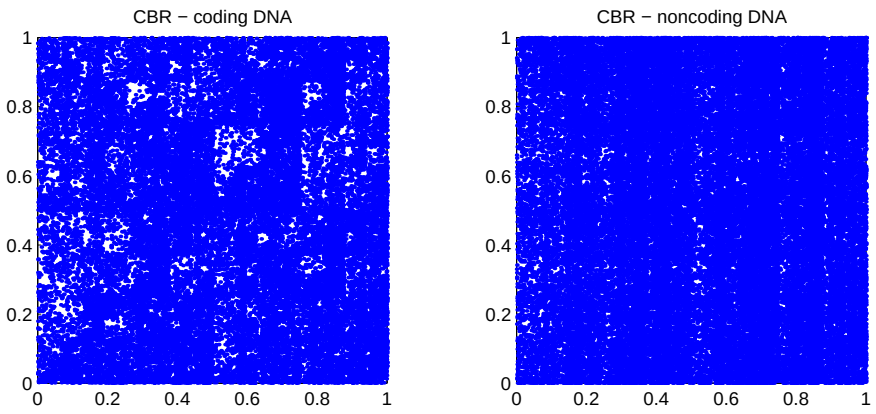

Figure 2: Geometric chaos block representations (CBR) of L-blocks in the DNA coding (left) and non-coding (right) training sequences.

almost no structure in the DNA L-blocks and both the L_1 and L_2 norm vector quantizers place the codebook on an approximately uniform grid similar to that formed by MMs. FPMs constructed on a perfectly uniform square grid mimic the corresponding MM. Poorer performance of FPMs is caused by deviations of the codebooks from regular grids.

The distribution of allowed blocks in the DNA sequences is more flat than that found in the chaotic laser sequence (section 4.4), but more subtle than the special self-similar Feigenbaum subsequence metric structure (section 4.5). Therefore, for small construction parameter values, the one-parameter and ratio PST construction schemes are prone to overfitting and the best PST results are achieved by the fixed growth parameter $\epsilon_{grow} = 0.001$ construction.

4.3 The Bible

4.3.1 Data and methods

In the second experiment, we tested our model on the experiments of Ron, Singer and Tishby with language data from the Bible (Ron, Singer, & Tishby, 1996). The alphabet was English letters and the blank character (27 symbols). They trained classical MMs and a VLMM on the books of the Bible except for the book of Genesis. Then the models were evaluated on the basis of normalized negative log-likelihood (eq. (11)) on an unseen portion of 236 characters from the book of Genesis. When constructing PST, Ron, Singer and Tishby set the maximal memory depth to $L = 30$. They built a PST with about 3000 nodes.

We compared likelihood results of our model with those obtained by Ron, Singer and Tishby for MMs and VLMMs. The training and test sets were the same as in (Ron, Singer, & Tishby, 1996). As with the VLMM, we set the maximal memory length to $L = 30$. FPMs were constructed by vector quantizing (in both L_1 and L_2 norms) a 5-dimensional[11] geometric representation of 30-blocks appearing in the training set.

4.3.2 Results

NNL results on the test set are shown in figure 3.

Both the PST and FPMs clearly outperform the MMs. FPMs appear to perform slightly better than the PST. Unfortunately, we were not able to further expand this experiment by giving results for various PST sizes and construction schemes. The training sequence contains approximately $3.4\ 10^6$ symbols from an alphabet of 27 characters. On a 2x-Ultrasparc workstation, all the FPM experiments were finished within a few days. We could not reproduce the PST reported in Ron, Singer and Tishby (1996) and (1994). The PST construction procedures worked extremely slow (recall that the maximal memory depth was set to $L = 30$, and the alphabet has 27 symbols), or resulted in small PSTs. Even after 3 months of computation we were not able to find suitable parameters that

[11] alphabet has 27 symbols

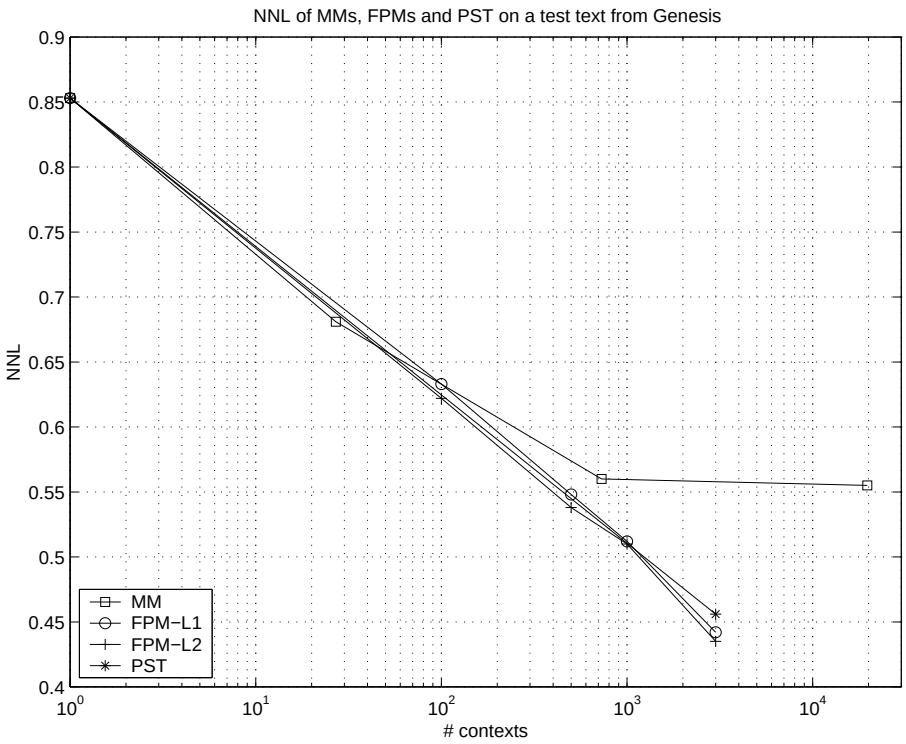

Figure 3: Normalized negative log likelihoods (NNL) achieved by finite context sources on an unseen text from the book of Genesis. The test sequence is the same as that used by Ron, Singer and Tishby (1996). The MM and PST results are reproduced from (Ron, Singer, & Tishby, 1996).

would yield a series of PSTs of size 500–3000. In this respect, the speed and self-organizing character of FPM construction proved to be of great advantage.

4.4 Laser in a chaotic regime

4.4.1 Data and methods

In the third experiment, we trained the models on a sequence of quantized activity changes of a laser in a chaotic regime. Deterministic chaotic dynamical systems usually organize their behavior around chaotic attractors containing regions of different levels of instability (sensitivity to small perturbations in initial conditions), measured e.g by the local Lyapunov exponents. Periods of relatively predictable behavior are followed by periods of unpredictable development (due to finite precision of our measuring devices and computing machines). By quantizing a chaotic trajectory into a symbolic stream (each symbol corresponds to a region of the state space where the system evolves), a technique well-known in symbolic dynamics, we obtain a rough picture about the basic topological, metric and memory structure of the trajectories (see e.g. Katok, & Hausselblatt, 1995). Relatively predictable subsequences having various levels of memory structure are followed by highly unpredictable events usually requiring a deep memory. For example, in this experiment, the chaotic laser produces periods of oscillations with increasing amplitude, followed by sudden, difficult to predict, activity collapses (see figure 4). To model such sequences with the simple class of stochastic models studied in this paper – finite context sources – we need to vary the memory depth with respect to the context. This is exactly the thing variable memory length models should be good at.

The data set was a long sequence[12] $\{D_t\}$ of 10 000 differences between the successive activations of a real laser in a chaotic regime. The sequence $\{D_t\}$ was quantized into a symbolic stream $S = \{s_t\}$ over four symbols corresponding to low and high positive/negative laser activity change:

[12] taken from `http://www.cs.colorado.edu/~andreas/Time-Series/SantaFe.html`

$$s_t = \begin{cases} 1 \text{ (normal up)}, & \text{if } 0 \le D_t < \theta_2 \\ 2 \text{ (extreme up)}, & \text{if } \theta_2 \le D_t \\ 3 \text{ (normal down)}, & \text{if } \theta_1 \le D_t < 0 \\ 4 \text{ (extreme down)}, & \text{if } D_t < \theta_1, \end{cases} \tag{12}$$

where the parameters θ_1 and θ_2 correspond to $\mathcal{Q}$ percent and $(100 - \mathcal{Q})$ percent sample quantiles, respectively. The number of positive differences is approximately the same as that of the negative differences. So, the upper (lower) $2\mathcal{Q}\%$ of all laser activation increases (decreases) in the sample are considered extremal, and the lower (upper) $(100 - 2\mathcal{Q})\%$ of laser activation increases (decreases) are viewed as normal. The quantile $\mathcal{Q}$ was set to 10%. Figure 4 shows a portion of the first 1000 laser activations, together with a histogram of the differences between the successive activations. Dotted vertical lines show the cut values θ_1 and θ_2 corresponding to the 10% and 90% quantiles, respectively.

The first 8000 symbols and the remaining 2000 symbols from the laser symbolic sequence S formed the training and test sequences, respectively. After constructing the finite-context sources MMs, VLMMs and FPMs on the training sequence (maximal memory depth was set to $L = 20$), we evaluated the normalized negative log-likelihood (NNL) (see eq.(11)) of the test sequence with respect to the fitted models.

4.4.2 Results

The results are shown in figure 5.

Classical MMs of order up to 5 are outperformed by FPMs with comparable number of contexts. There is almost no difference between the performances of FPMs constructed using the L_1-norm and L_2-norm based procedures[13].

[13] In this experiment, we also tried other vector quantization techniques like the classical Kohonen self-organizing feature maps (SOFM) (Kohonen, 1990), SOFM with the star topology of neuron field (Tiňo, & Šajda, 1995), dynamic cell structures (Bruske, & Sommer, 1995) or deterministic annealing based hierarchical clustering (Rose, Gurewitz, & Fox, 1990). We got model performances comparable to those of the models obtained via the K-means clustering. Clustering via deterministic annealing took enormous time without any apparent improvement in the resulting predictive models.

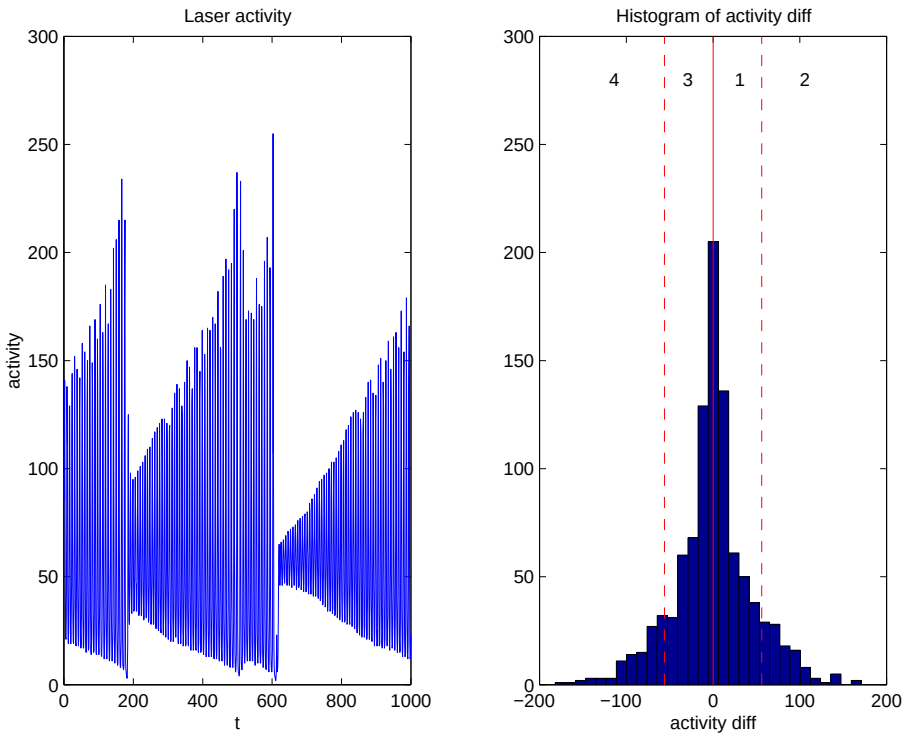

Figure 4: Left – the first 1000 activations of a laser in a chaotic regime. Right – histogram of the differences between the successive activations. Dotted vertical lines show the cut values θ_1 and θ_2 corresponding to the $\mathcal{Q}\%$ and $(100-\mathcal{Q})\%$ quantiles, respectively ($\mathcal{Q} = 10$). Symbols corresponding to quantization regions appear on top of the figure.

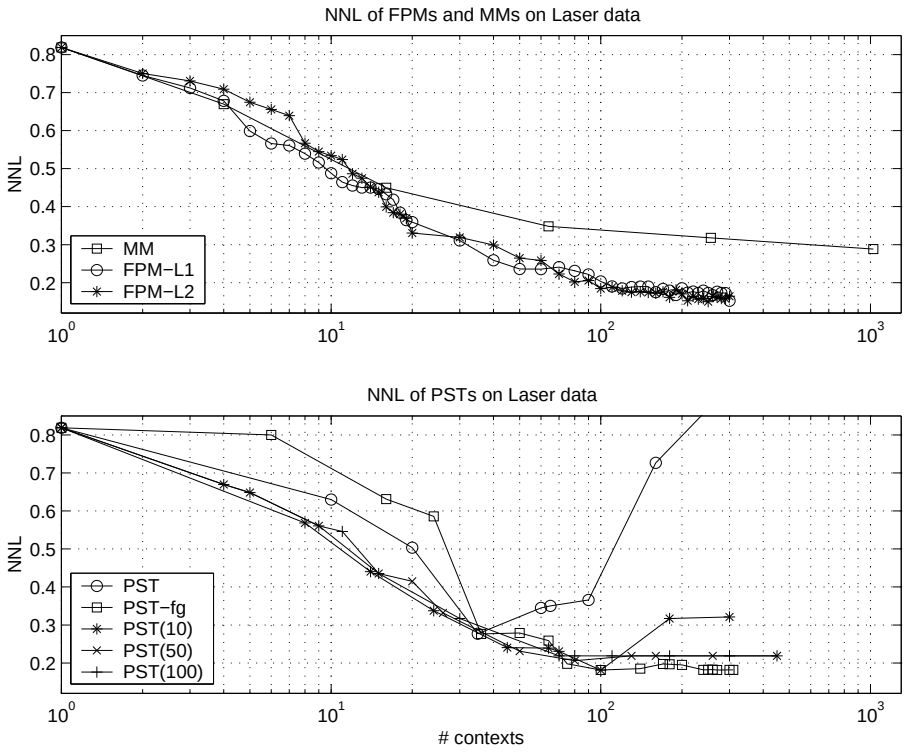

Figure 5: Normalized negative log-likelihoods (NNL) of the laser test sequence with respect to finite-context sources built on the laser training sequence. Markov models are indicated by `MM`. FPMs corresponding to the $L1$-norm and L_2-norm based constructions are indicated by `FPM-L1` and `FPM-L2`, respectively. PSTs constructed using the one-parameter scheme, PSTs build with fixed growth parameter ϵ_{grow}, and PSTs constructed with ratio-related growth and threshold parameters $\epsilon_{grow} = \rho\epsilon_{KL}$, $\rho = 10, 50, 100$, are indicated by `PST`, `PST-fg`, and `PST(`ρ`)`, respectively.

As discussed in section 2.2, small values of $\epsilon = \epsilon_{KL} = \epsilon_{grow}$ in the one-parameter PST construction scheme lead to including low-probability subsequences as potential prediction PST contexts. This results in PSTs greatly overfitting the training sequence (line indicated by `PST` in figure 5). The line indicated by `PST-fg` traces the NNLs achieved by the fixed growth parameter $\epsilon_{grow} = 0.001$ PST construction scheme. Only the acceptance threshold parameter ϵ_{KL} is varied. While the overfitting effect in larger PSTs has disappeared, smaller PSTs (corresponding to larger values of ϵ_{KL}) perform poorly, since the fixed small value of ϵ_{grow} resulted in considering unnecessarily specific contexts. Finally, we show the results for the procedure constructing PSTs with ratio-related parameters $\epsilon_{grow} = \rho\ \epsilon_{KL}$, $\rho = 10, 50, 100$ (lines indicated by `PST(`ρ`)`). For small ϵ_{KL}, the ratio value of 10 is still too low to prevent the overfitting effect. PSTs constructed with ratios $\rho = 50$ and $\rho = 100$ achieve performances comparable to those of FPMs.

This experiment demonstrates that VLMM construction can be highly dependent on construction parameters and that using the one-parameter scheme of Ron, Singer, & Tishby, (1994) may result in too specific models strongly overfitting the training sequence. FPMs, on the other hand, are constructed by simply enlarging the codebook in the vector quantization phase and show no deterioration in performance when increasing the number of prediction contexts[14].

To illustrate the difference between the fixed-order and variable-context-length Markov models, we plot in figure 6 the geometric representations $\sigma\left(S_t^{t+L-1}\right)$, $t = 1, 2, ..., m-L+1$, of L-blocks appearing in the training sequence $S = s_1 s_2 ... s_m$, (see eq. (5)), together with geometric representations $\sigma(w)$ of prediction contexts $w \in \mathcal{C}$ found in the MM, PST and FPMs of comparable size (approximately 256 contexts). Geometric representations of the training sequence L-blocks are shown as dots in the upper left part of figure 6.

Geometric representations of prediction contexts of the 4th-order MM, shown as circles in the lower right, blindly cover the unit square $[0, 1]^2$, regardless of the actual L-block distribution in the training sequence.

Prediction contexts of the VLMM (PST constructed with ratio related parameters

[14] at least up to 300 contexts

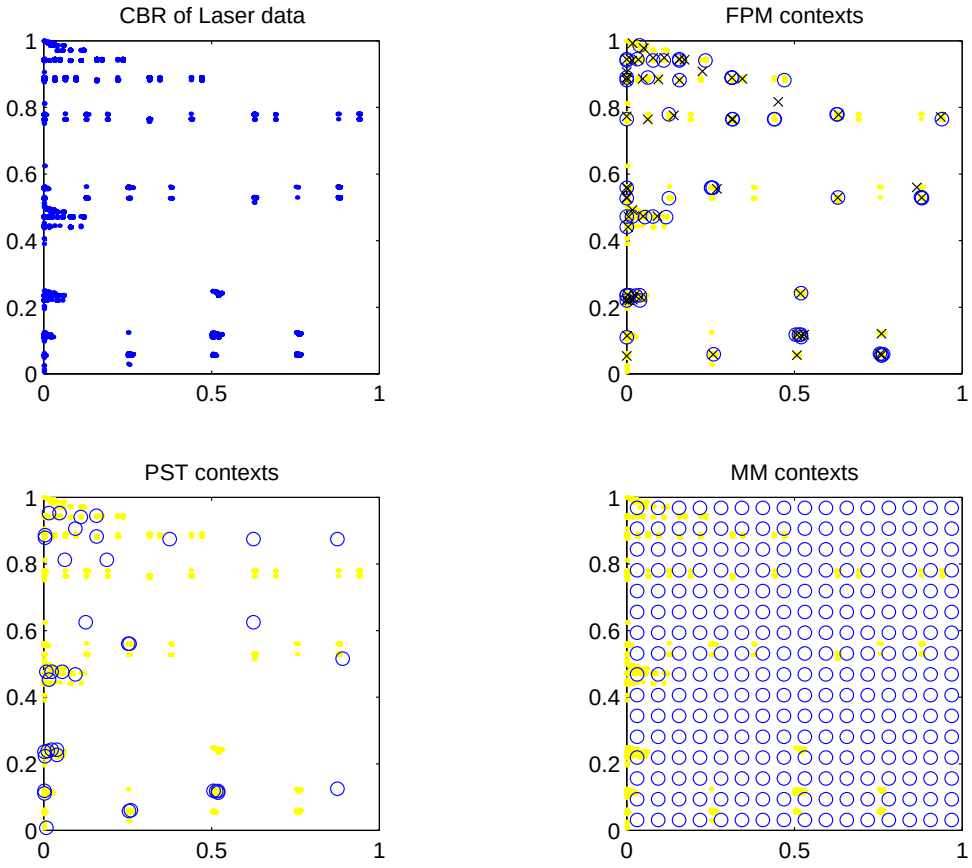

Figure 6: Chaos block representations (CBR) of L-blocks in the laser training sequence (upper left), prediction contexts of FPMs (upper right), PST (lower left), and MM (lower right). Chaos block representations of prediction contexts are shown as circles, except for contexts of the L_2-norm constructed FPM (shown as crosses).

$\epsilon_{grow} = 50\ \epsilon_{KL}$) are suffixes of the allowed L-blocks, and so geometric representations of the prediction contexts concentrate on the areas inhabited by representations of the allowed L-blocks (see section 3.1). The context selection criteria favor prediction contexts whose probability exceeds the "acceptance" threshold ϵ_{grow} and whose next-symbol probabilities do not significantly differ from those of the extended contexts. The result (lower left of figure 6) is a sort of "conditional" vector quantization of geometric representations of the training sequence L-blocks, whose aim is to cover the set of "accepted" allowed blocks with a set of prediction contexts, taking into account the associated next-symbol probabilities.

FPM contexts, shown in the upper right of figure 6, correspond to codebooks constructed by vector quantization in the L_1 (circles) and L_2 (crosses) norms.

4.5 Feigenbaum sequence

4.5.1 Data and methods

In the fourth experiment, we applied the models to the Feigenbaum binary sequence with a very strict topological and metric organization of allowed subsequences (see e.g. (Katok, & Hausselblatt, 1995)). The sequence was obtained by quantizing the time series resulting from the well-known logistic equation in the chaotic regime with respect to the sign of the iterands (1–negative, 2–non-negative). Highly specialized, very deep prediction contexts are needed to model this sequence. Classical Markov models cannot succeed and the full power of admitting a limited number of variable length contexts can be exploited.

The sequence is well-studied in symbolic dynamics and has a number of interesting properties. First, the topological structure of the sequence (i.e. the structure of allowed n-blocks, not regarding their probabilities) can only be described using a context sensitive tool – a restricted indexed context-free grammar (Crutchfield, & Young, 1990). Second, for each block length $n = 1, 2, ...$, the distribution of n-blocks is either uniform, or has just two probability levels. Third, the n-block distributions are organized in a self-similar fashion (Freund, Ebeling, & Rateitschak, 1996). The transition between the ranked distributions for block lengths $2^g \rightarrow 2^{g+1}$, $3 \cdot 2^{g-1} \rightarrow 3 \cdot 2^g$, $g \geq 1$, is achieved by rescaling the horizontal

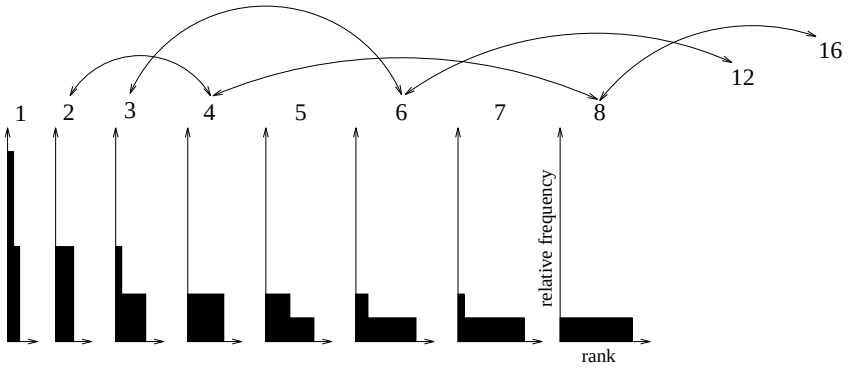

Figure 7: Plots of self-similar rank-ordered block distributions of the Feigenbaum sequence for different block lengths (indicated by the numbers above the plots). The self similarity relates block distributions for block lengths $2^g \to 2^{g+1}$, $3 \cdot 2^{g-1} \to 3 \cdot 2^g$, $g \geq 1$ (connected by arrows).

and vertical axis by a factor 2 and $\frac{1}{2}$, respectively. Plots of the Feigenbaum sequence n-block distributions, $n = 1, 2, ..., 8$, can be seen in figure 7. Numbers above the plots indicate the corresponding block lengths. The arrows connect distributions with the $(2, \frac{1}{2})$-scaling self-similarity relationship.

The sequence can be specified by the composition rule

$$a_0 = 2, \quad a_1 = 21, \quad a_{n+1} = a_n a_{n-1} a_{n-1}. \tag{13}$$

We chose to work with the Feigenbaum sequence, because increasingly accurate modeling of the sequence with finite memory models requires a selective mechanism for deep prediction contexts.

We created a large portion of the Feigenbaum sequence and trained a series of classical MMs, variable memory length MMs (VLMMs), and fractal prediction machines (FPMs) on the first 260 000 symbols. The following 200 000 symbols formed a test set. Maximum memory length L for VLMMs and FPMs was set to 30.

4.5.2 Results

Due to the special metric structure of the Feigenbaum sequence, where for each block length n, the n-block distribution is either uniform, or has just two probability levels,

the issues concerning the growth parameter ϵ_{grow} in the PST construction, prominent in the previous experiment, are not relevant. Therefore we report just VLMM results corresponding to the one-parameter PST construction scheme.

Nevertheless, constructing a series of increasingly complex VLMMs by varying the construction parameter appeared to be a troublesome task. Unlike in the previous experiment, the PST construction procedure did not work "smoothly" with varying the construction parameter. We experienced a highly non-regular behavior with intervals of parameter values yielding unchanged PSTs, and tiny regions in parameter space corresponding to a large spectrum of PST sizes. Therefore, it was impossible to simply iteratively change the parameters by a small amount and save the resulting PSTs (as done in the previous experiment). Instead, one had to spent a fair amount of time to find the "critical" parameter values.

In contrast, a fully automatic construction of FPMs involved sliding a window of length $L = 30$ through the training set; for each window position, mapping the L-block w appearing in the window to the point $\sigma(w)$ (eq. (5)), vector-quantizing (in both L_1 and L_2 norms) the resulting set of points (up to 30 codebook vectors). After the quantization step, we computed predictive probabilities according to eq. (8).

Figure 8 is analogous to figure 6 from the previous experiment. One dimensional[15] geometric representations of the training sequence L-blocks form very dense, well-separated clusters. In this case, vector quantization in L_1 and L_2 norms gives almost identical codebooks and so both the L_1 and L_2 norm based FPM constructions yielded the same results. Variable-context-length models quickly grasp the structure in allowed L-blocks. The rigid fixed-order MMs, instead of specializing on deeper contexts, spare their resources to cover the missing subsequences.

Normalized negative log-likelihoods (NNL) (eq.(11)) of the test set computed using the fitted models exhibited a step-like increasing tendency shown in Table 3. We also investigated the ability of the models to reproduce the n-block distribution found in the training and test sets. This was done by letting the models generate sequences of length

[15]Alphabet $\mathcal{A} = \{1, 2\}$ has two symbols

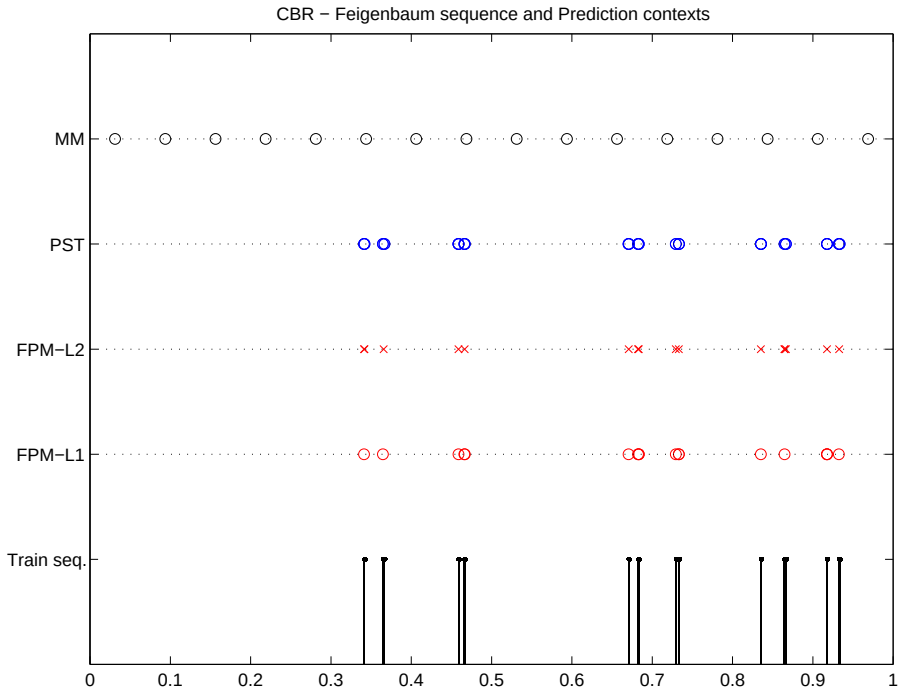

Figure 8: One-dimensional chaos block representations of L-blocks in the binary Feigenbaum training sequence (bottom). Shown are also geometric representations of the prediction contexts of FPMs, PST, and MM with approximately 16 prediction contexts. Representations of prediction contexts are shown as circles, except for contexts of the L_2-norm constructed FPM (shown as crosses).

Table 3: Normalized negative log-likelihoods (NNL) on the Feigenbaum test set.

model	**# contexts**	**NNL**	**captured block distribution**
FPM	2–4	0.6666	1–3
	5–7	0.3333	1–6
	8–22	0.1666	1–12
	23–	0.0833	1–24
PST	2–4	0.6666	1–3
	5	0.3333	1–6
	11	0.1666	1–12
	23	0.0833	1–24
MM	2,4,8,16,32	0.6666	1–3

equal to the length of the training sequence and for each block length $n = 1, 2, ..., 30$, computing the L_1 distance between the n-block distribution of the training and model-generated sequences. The n-block distributions on the test and training sets were virtually the same for $n = 1, 2, ...30$. In Table 3 we show block lengths for which the L_1 distance does not exceed a small threshold Δ. We set $\Delta = 0.005$, since in this experiment, either the L_1 distance was less 0.005, or exceeded 0.005 by a large amount.

The classical MM totally fails in this experiment, since the context length 5 is far too small to enable the MM to mimic the complicated subsequence structure in the Feigenbaum sequence. FPMs and VLMMs quickly learn to explore a limited number of deep prediction contexts and perform comparatively well.

An explanation of the step-like behavior in the log-likelihood and n-block modeling behavior of VLMMs and FPMs is out of the scope of this paper. For a detailed analysis, see (Tiňo, Dorffner, & Schittenkopf, 2000). We briefly mention, however, that by combining the knowledge about the topological and metric structures of the Feigenbaum sequence (e.g. (Freund, Ebeling, & Rateitschak, 1996)) with a careful analysis of the models, one can show why and when an inclusion of a prediction context leads to an abrupt improve-

ment in the modeling performance. In fact, we show that VLMMs and FPMs constitute increasingly better approximations to the infinite self-similar Feigenbaum machine known in symbolic dynamics (Crutchfield, & Young, 1990).

4.6 Financial data

4.6.1 Data and methods

The final data set consisted of quantized daily volatility changes of the Dow Jones Industrial Average (DJIA) from Feb. 1 1918 until April 1 1997, transformed into a time series of returns $r_t = \log x_{t+1} - \log x_t$. Predictive models were used to predict the direction of volatility move for the next day. In (Tiňo et al., 2000a) we show that the quantization, symbol based approach to volatility prediction can outperform the more traditional econometric models of the ARCH and GARCH families (Bollerslev, 1986). Financial time series are known to be highly stochastic with a relatively shallow memory structure (Jaditz, & Sayers, 1993). In addition, to account for stationarity, financial time series of daily values are usually kept short. In this case, it is difficult to beat the low-order classical MMs. One can perform better than MMs only by developing a few deeper specialized contexts, but that, on the other hand, can easily lead to overfitting.

We considered the squared return r_t^2 a volatility estimate for day t. Volatility change forecasts (volatility is going to increase or decrease) based on historical returns can be interpreted as a buying or selling signal (in an option market) for a straddle (see e.g. (Noh, Engle, & Kane, 1994)). If the volatility decreases, we go short (straddle is sold), if it increases, we take a long position (straddle is bought). In this respect, the quality of a volatility model can be measured by the percentage of correctly predicted directions of daily volatility differences.

The series of returns $\{r_t\}$ was transformed into a series $\{D_t\}$ of differences between the successive squared returns $D_t = r_{t+1}^2 - r_t^2$. We then partitioned the series $\{D_t\}$ of daily volatility moves into 13 non-overlapping intervals, each containing 1700 values (spanning approximately $6\frac{1}{2}$ years). Each interval was further partitioned into the training set (the

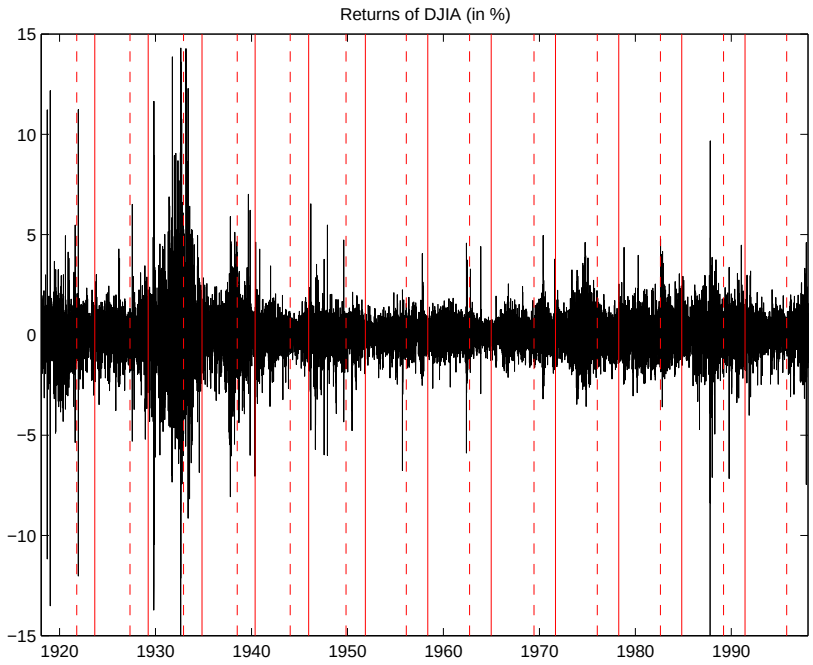

Figure 9: Series of returns (in percent) of the DJIA from February 1918 till April 1997. The solid vertical lines indicate division into intervals, the dotted vertical line within each interval indicates the split between the training and validation sets. The first 600 values from the training set of an interval forms a test set for the previous interval.

first 1100 values) and the validation set (the remaining 600 values). The series of returns of the DJIA can be seen in figure 9. The solid vertical lines indicate division into the intervals, the dotted vertical line within each interval indicates the split between the training and validation sets. For each interval $I = 1, 2, ..., 12$, predictive models were trained on the training set, candidate models were selected on the validation set, and the selected candidate models were tested on the test set formed by the first 600 values from the training set of the next interval. This way, we got 12 partially overlapping epochs of the series $\{D_t\}$, each containing 1100+600+600=2300 values (spanning approximately 9 years). Training sets of the 12 epochs do not overlap. The same applies to the test and validation sets.

In each epoch, we transformed the training series $\{D_t\}$ of daily volatility differences into

a sequence over four symbols via the quantile technique used in the laser data experiment (see eq. (12)). Given a quantile $\mathcal{Q}$, the validation and test sets were quantized using the cut values determined for $\mathcal{Q}$ on the training set.

Maximum memory length L for VLMMs and FPMs was set to 10 (two weeks). We trained classical MMs, PSTs and FPMs with various numbers of prediction contexts (up to 256) and extremal event quantiles $\mathcal{Q} \in \{10, 20, ..., 40\}$. For each model class, the model size and the quantile $\mathcal{Q}$ to be used on the test set were selected according to the validation set performance. Performance of the models was quantified as the percentage of correct guesses of the volatility change direction for the next day. If the next symbol was 1 or 2 (3 or 4) and the sum of conditional next symbol probabilities for 1 and 2 (3 and 4) given by a model was greater than 0.5, the model guess was considered correct.

4.6.2 Results

For all 12 epochs, test set performances of the models selected on the validation sets are shown in figure 10.

We subjected the differences in model performances across the 12 epochs to the parametric t- and non-parametric Wilcoxon paired significance tests. The results of significance tests are summarized in table 4. Both tests reveal that the FPMs significantly outperform VLMMs. The L_2 norm based FPMs perform significantly better than MMs. Both tests also suggest that MMs significantly outperform PSTs constructed with one-parameter scheme. Restricting to t-test, MMs appear to be significantly better than any PST scheme.

This experiment illustrates the *practical* problems in fitting VLMMs. Training sequences in this experiment are relatively short (1100 symbols – approximately $4\frac{1}{2}$ years). Considering stationarity issues, they can hardly be made substantially larger. In addition, financial time series are known to be highly stochastic with a relatively shallow memory structure (Jaditz, & Sayers, 1993). All PST construction schemes develop too specialized prediction contexts, even for small PSTs. In this case, the use of validation set strategy does not completely prevent PSTs from overestimating the memory structure in the data.

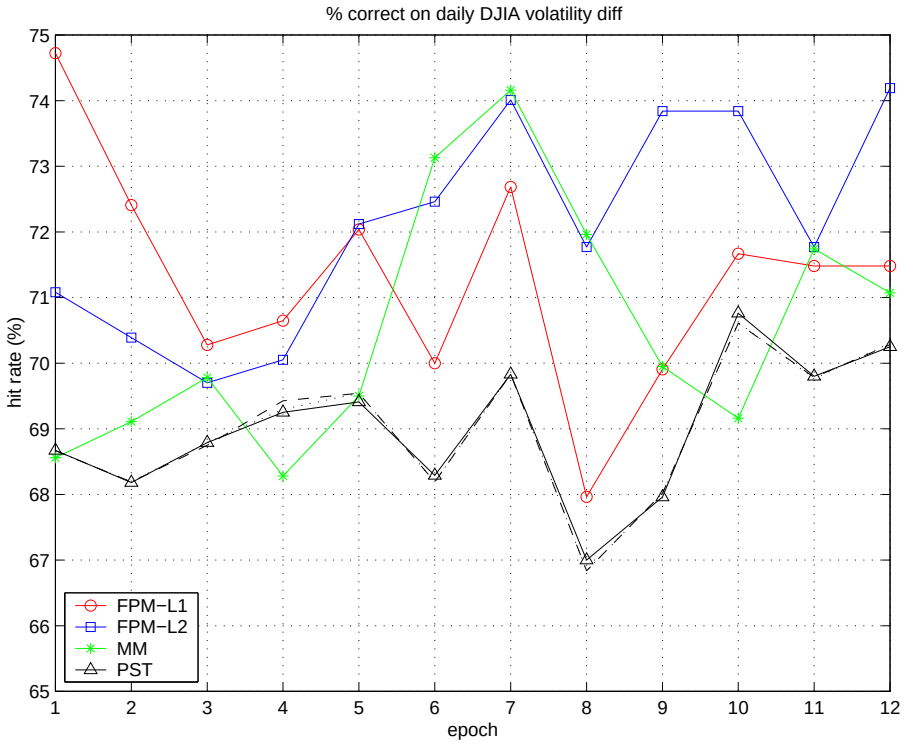

Figure 10: Prediction performance (hit rates) of MMs, PSTs and FPMs on 12 epochs of the quantized daily volatility moves of the DJIA. FPMs constructed through the L_1 and L_2 norm procedures are indicated by `FPM-L1` and `FPM-L2`, respectively. The performances of PSTs constructed via the one-parameter (solid line), fixed growth parameter $\epsilon_{grow} = 0.001$ (dashed line), and ratio $\epsilon_{grow} = 50\ \epsilon_{KL}$ (dotted line) schemes are almost identical.

Table 4: Tests for significance in model performances across 12 epochs in the DJIA experiment. Item (i, j) reports results of the test whether the model corresponding to the row i significantly outperforms the model associated with the column j. Significance suggested by the (parametric) t- and (non-parametric) Wilcoxon paired tests is marked with $*$ and $+$, respectively. A double star (plus) means a significance on 1% level, a single star (plus) corresponds to a significance on 5% level, – means no significance. PSTs constructed using the one-parameter, fixed growth parameter $\epsilon_{grow} = 0.001$, and ratio $\epsilon_{grow} = 50\ \epsilon_{KL}$ schemes are denoted by PST, PST–fg, and PST(50), respectively.

model	FPM–L_1	FPM–L_2	PST	PST–fg	PST(50)	MM
FPM–L_1	–	–	$**++$	$**++$	$**++$	–
FPM–L_2	–	–	$**++$	$**++$	$**++$	$*+$
PST	–	–	–	–	–	–
PST–fg	–	–	–	–	–	–
PST(50)	–	–	–	–	–	–
MM	–	–	$*+$	$*$	$*$	–

5 Discussion

In four of the five experiments, fractal prediction machines (FPMs) performed at least as well as variable memory length Markov models (VLMMs). The only exception is the DNA sequence, where due to the uniform distribution of contexts, classical Markov models (MMs) are favored. In this case, FPMs performed worse than VLMMs.

In the remaining four cases FPMs outperformed classical MMs, and showed a decisive advantage over VLMMs with respect to model performance and/or ease of construction:

- in the case of the bible text, inhibitive computational demands of the VLMM were revealed. In contrast, FPMs could efficiently be estimated on the same data set with a variety of numbers of contexts,
- in the case of quantized Laser data, the experiments pointed to a severe parameter-dependency of the estimation algorithm for VLMMs, whereas FPMs proved to be robust and effective,
- in the case of the Feigenbaum sequence, FPMs achieved the same level of performance (measured by negative log likelihood) as the VLMM, but the construction of FPMs was much less troublesome.
- in the case of quantized financial data, FPMs significantly outperformed VLMMs, mainly due to the availability of only short training sequences rendering the estimation of a VLMM difficult.

In summary, the experiments demonstrate that fractal prediction machines are an efficient and viable candidate for learning the statistical structure of symbolic sequences, whenever the classical Markov models are not appropriate due to the existence of deep structure involving only a few relevant contexts.

One of the main advantages of our approach is the self-organizing character of constructing a series of fractal-based predictive models, fractal prediction machines (FPMs), of increasing size. Vector quantization covers the geometric L-block representations of

the training sequence with increasingly large codebooks in a natural and self-organized manner. Predictive models constructed on such codebooks can be compared through a model selection criterion, e.g. validation set performance.

Constructing a series of increasingly large models to enter the model selection phase is an important issue that has attained little attention in the VLMM literature. In practical applications with larger alphabets and very long sequences, constructing a set of potential candidate VLMMs can take a prohibitively long time. Indeed, results in the VLMM literature are usually presented only for a few fitted models, stressing the memory requirement advantage of VLMMs over the classical MMs. Little is said about whether a particular model was selected from a set of potential candidates, or how difficult it was to arrive at the presented solution (see, for example (Ron, Singer, & Tishby, 1996; Ron, Singer, & Tishby, 1994)).

Guyon and Pereira (1995) study two ways of constructing increasingly complex VLMMs: by increasing the source memory L with other construction parameters kept fixed, or by fixing a (long enough) source memory L and gradually changing a single parameter, while keeping all the other construction parameters fixed. The latter scheme was experimentally shown to yield a superior performance (Guyon, & Pereira, 1995). It should be noted, that while Guyon and Pereira (1995) do construct a series of increasingly complex VLMMs on a very large set (AP news corpus, containing about 10^8 characters), they do so by setting the maximal memory depth to $L = 5$. Such a shallow memory construction[16] enabled the authors to construct a series of VLMMs in a realistic time. Larger memory lengths L would lead to an exponential increase in PST construction time.

In addition, as mentioned in section 4.5, the construction parameters' selection is a non-intuitive task that may require a lot of interactive steps. In this respect, the FPM construction is more intuitive (the number of codebook vectors directly corresponds to the number of predictive contexts), easier to automate (growth of predictive models is directed by the codebook growth in the self-organizing quantization algorithms) and often faster.

[16] compare with memory depth of $L = 30$ in the Ron, Singer and Tishby (1996) Bible experiment

Moreover, we illustrated in the laser data experiment, that considering different VLMM parameter selection strategies can lead to completely different learning scenarios. In contrast, the simple FPM construction is free of such defects. Interestingly enough, it gives similar results for both the L_1 and L_2 norm FPM algorithms[17].

Variable memory length strategies work better than the classical fixed order Markov models when there is a significant suffix structure in long allowed blocks of the training sequence, not explainable by considering some pre-defined, (relatively) small suffix length. Natural language, as demonstrated in the Bible experiment, is an example of such a situation. Another example is provided by the Feigenbaum sequence.

DNA sequences stand at the opposite end, with a rather uniform suffix structure. In this case, it is difficult to outperform the classical MMs. Better performance might be achieved with specialized models, incorporating some a-priori knowledge, e.g. gene structure expressed in a hidden Markov model topology (Krogh, 1997), or similarity searches with respect to known amino acid sequences (Burset & Guigó, 1996).

However, allowing for a variable memory length is a double-edged sword. Especially on shorter sequences (relative to the alphabet size), the variable memory length model construction often specializes on overly deep prediction contexts, even for small model sizes. As shown in the Dow Jones Industrial Average experiment, in this case, model selection strategies cannot eliminate the overlearning effects.

Is is only fair to note that even though the FPMs emerge from our experiments as potentially interesting and favorable alternatives to VLMMs, so far, they lack a sound theoretical background comparable to that supporting the use of VLMMs (Ron, Singer, & Tishby, 1996; Weinberger, Rissanen, & Feder, 1995; Bühlmann, & Wyner, 1999). Proceeding in this direction, we have theoretically analyzed the multifractal properties of the basis for our predictive models' construction – the geometric L-block representation (Tiňo, 1999), and found a relationship among the chaos block representation contraction factor, magnification factor of the vector quantizer and the dynamics of the FPM context

[17] We thank one of the anonymous reviewers for suggesting to use also the L_1-norm FPM construction scheme

transitions.

Acknowledgements

We wish to thank Owen Kelly for helpful comments on variable length Markov models and anonymous reviewers for many helpful suggestions. 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.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 716–723.

Aurenhammer, F. (1991). Voronoi Diagrams - Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 3, 345–405.

Barnsley, M.F. (1988). *Fractals everywhere*. New York: Academic Press.

Bauer, H.U., Der, R., & Herrmann, M. (1996). Controlling the magnification factor of self-organizing feature maps. *Neural Computation*, 8, 757–771.

Bollerslev, T. (1986). A generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, 31, 307–327.

Brillinger, D.R. (1994). Examples of scientific problems and data analysis in demography, neurophysiology and seismology. *J.Comp. and Graph. Statistics*, 3, 1–22.

Bruske, J., & Sommer, G. (1995). Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, 4, 845–865.

Burset, M., & Guigó, R. (1996) Evaluation of gene structure prediction programs. *Genomics*, 34, 353–357.

Bühlmann, P. (1998). Extreme events from return-volume process: a discretization approach for complexity reduction. *Applied Financial Economics*, 8, 267-278.

Bühlmann, P. (1999). Dynamic adaptive partitioning for nonlinear time series. *Biometrika*, 86, 555-571.

Bühlmann, P. & Wyner, A.J. (1999). Variable length Markov chains. *Annals of Statistics*, 27, 480–513.

Bühlmann, P. (2000). Model selection for variable length Markov chains and tuning the context algorithm. *Annals of the Institute of Statistical Mathematics*, in press.

Buhmann, J.M. (1995). Learning and data clustering. In Arbib, M. (Eds.), *Handbook of Brain Theory and Neural Networks*. Bradford Books, MIT Press.

Crutchfield, J.P., & Young, K. (1990). Computation at the onset of chaos. In Zurek, W.H. (Eds.), *Complexity, Entropy, and the physics of Information, SFI Studies in the Sciences of Complexity* (pp. 223-269). Reading, MA: Addison-Wesley.

Everitt, B.S. (1977). *The analysis of contingency tables*. London: Chapman and Hall.

Freund, J., Ebeling, W., & Rateitschak, K. (1996). Self-similar sequences and universal scaling of dynamical entropies. *Physical Review E*, 5, 5561–5566.

Giles, C.L., Lawrence, S., & Tsoi, A.C. (1997). Rule inference for Financial Prediction using Recurrent Neural Networks. *Proceedings of the conference on Computational Intelligence for Financial Engineering* (pp. 253–259). New York City, NY.

Guyon, I., & Pereira, F. (1995). Design of a linguistic postprocessor using variable memory length Markov models. In *Proceedings of International Conference on Document Analysis and Recognition* (pp. 454–457). Montreal, Canada: IEEE Computer Society Press.

Jaditz, T., & Sayers, C.L. (1993). Is chaos generic in economic data? *Int. Journal of Bifurcation and Chaos*, 3, 745–755.

Jeffrey, J. (1990). Chaos game representation of gene structure. *Nucleic Acids Research*, 8, 2163–2170.

Katok, A., & Hausselblatt, B. (1995). *Introduction to the Modern Theory of Dynamical Systems*. Cambridge, UK: Cambridge University Press.

Khinchin, A.I. (1957). *Mathematical Foundations of Information Theory*. New York: Dover Publications.

Kohonen, T. (1990). The self–organizing map. *Proceedings of the IEEE*, 9, 1464–1479.

Krogh, A. (1997). Two methods for improving performance of a HMM and their application for gene finding. *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology* (pp. 179–186). Menlo Park, CA: AAAI Press.

Laird, P., & Saul, R. (1994). Discrete sequence prediction and its applications. *Machine Learning*, 15, 43–68.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297), Berkeley, CA: University of California Press.

Nadas, A. (1984). Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Trans. on ASSP*, 4, 859–861.

Noh, J., Engle, R.F., & Kane, A. (1994). Forecasting volatility and option prices of the s&p 500 index. *Journal of Derivatives*, 1, 17–30.

Papageorgiou, C.P. (1998). Mixed Memory Markov Models for Time Series Analysis. *Proceedings of the conference on Computational Intelligence for Financial Engineering* (pp. 165–183). New York City, NY.

Prum, B., Rodolphe, F., & deTurckheim, E. (1995). Finding words with unexpected frequencies in deoxyribonucleic acid sequences. *Journal of Royal Statistical Society*, B 57, 205–220.

Rissanen, J. (1983). A universal data compression system. *IEEE Trans. Inform. Theory*, 5, 656–664.

Ritter, H., & Schulten, K. (1986). On the stationary state of the kohonen's self-organizing sensory mapping. *Biol. Cybern.*, 54, 99–106.

Ron, D., Singer, Y., & Tishby, N. (1994). The power of amnesia. In *Advances in Neural Information Processing Systems 6* (pp. 176–183). Morgan Kaufmann.

Ron, D., Singer, Y., & Tishby, N. (1996). The power of amnesia. *Machine Learning*, 25, 117–150.

Rose, K., Gurewitz, E., & Fox, G.C. (1990). Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 8, 945–948.

Tiňo, P., & Šajda, J. (1995). Learning and extracting initial mealy machines with a modular neural network model. *Neural Computation*, 4, 822–844.

Tiňo, P., & Dorffner, G. (1998). Recurrent neural networks with iterated function systems dynamics. *International ICSC/IFAC Symposium on Neural Computation* (pp. 526–532).

Tiňo, P., & Köteles, M. (1999). Extracting finite state representations from recurrent neural networks trained on chaotic symbolic sequences. *IEEE Transactions on Neural Networks*, 10(2), 284–302.

Tiňo, P. (1999). Spatial representation of symbolic sequences through iterative function systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 4, 386-392.

Tiňo, P., Dorffner, G., & Schittenkopf, Ch. (2000). Understanding State Space Organization in Recurrent Neural Networks with Iterative Function Systems Dynamics. In S. Wermter & R. Sun (Eds.), *Hybrid Neural Symbolic Integration* (pp. 256–270). Heidelberg: Springer Verlag.

Tiňo, P., Schittenkopf, Ch., Dorffner, G., & Dockner, E.J. (2000a). A Symbolic dynamics approach to volatility prediction. To appear in Y.S. Abu-Mostafa, B. LeBaron, A.W. Lo & A.S. Weigend (Eds.), *Computational Finance*. Cambridge, MA: MIT Press.

Weinberger, M.J., Rissanen, J.J., & Feder, M. (1995). A universal finite memory source. *IEEE Transactions on Information Theory*, 3, 643–652.

Willems, F.M.J., Shtarkov, Y.M., & Tjalkens, T.J. (1995). The context tree weighting method: basic properties. *IEEE Trans.Info.Theory*, 3, 653–664.