Retrain or not retrain: Conformal test martingales for change-point detection

Vladimir Vovk, Ivan Petej, Ilia Nouretdinov,
Ernst Ahlberg, Lars Carlsson, and Alex Gammerman

практические выводы
теории вероятностей
могут быть обоснованы
в качестве следствий
гипотез о *предельной*
при данных ограничениях
сложности изучаемых явлений

On-line Compression Modelling Project (New Series)

Working Paper #32

First posted February 20, 2021. Last revised December 21, 2021.

Project web site:
http://alrw.net

Abstract

We argue for supplementing the process of training a prediction algorithm by setting up a scheme for detecting the moment when the distribution of the data changes and the algorithm needs to be retrained. Our proposed schemes are based on exchangeability martingales, i.e., processes that are martingales under any exchangeable distribution for the data. Our method, based on conformal prediction, is general and can be applied on top of any modern prediction algorithm. Its validity is guaranteed, and in this paper we make first steps in exploring its efficiency.

Contents

1 Introduction

The standard assumption in mainstream machine learning is that the observed data are IID (independent and identically distributed); we will refer to it as the *IID assumption*. Deviations from the IID assumption are known as dataset shift, and different kinds of dataset shift have become a popular topic of research (see, e.g., Quiñonero-Candela et al. 2009).

Testing the IID assumption has been a popular topic in statistics (see, e.g., Lehmann 2006, Chapter 7), but the mainstream work in statistics concentrates on the batch setting with each observation being a real number. In the context of deciding whether a prediction algorithm needs to be retrained, it is more important to process data online, so that at each point in time we have an idea of the degree to which the IID assumption has been discredited. It is also important that the observations are not just real numbers; in the context of machine learning the most important case is where each observation is a pair (x, y) consisting of a sample x (such as an image) and its label y. The existing work on detecting dataset shift in machine learning (see, e.g., Harel et al. 2014 and its literature review) does not have these shortcomings but does not test the IID assumption directly.

At this time the only existing method of testing the IID assumption online is based on the method of conformal prediction [Vovk et al., 2005]; see, e.g., Vovk [2021] for a recent review. As we explain in Section 2, conformal prediction allows us to construct *exchangeability martingales*, which can be used as tools for testing the IID assumption. In Section 3 we discuss an informal scheme that uses exchangeability martingales for deciding when a prediction algorithm needs to be retrained and illustrate it on the well-known Wine Quality dataset, concentrating on the simple case where there is only one change point where the distribution of the data changes.

In this paper we discuss three basic procedures for raising an alarm when the IID assumption appears to become violated. The simplest one is the *Ville procedure*, which raises an alarm when a given exchangeability martingale exceeds a given threshold. As we explain at the beginning of Section 4, the Ville procedure works well at the beginning of the testing process but then becomes less efficient. To remedy this drawback, we introduce conformal versions of the popular CUSUM and Shiryaev–Roberts procedures and illustrate their performance on the Wine Quality dataset, again concentrating on the case of one change point.

In Section 5 we state our proposed schemes for deciding when a prediction algorithm should be retrained. The two main schemes, which we call the variable and fixed schedules, are combinations of the three basic procedures. A short Section 6 concludes the main part of the paper.

In Appendix A we propose and discuss a slightly more advanced scheme. A naive implementation of this paper's methods often leads to numerical problems, and Appendix B discusses simple ways of avoiding them.

In this paper we repeatedly refer to the validity vs efficiency of our procedures. Validity refers to their behaviour when the IID assumption is satisfied

(the *ideal setting*, which we achieve by simulating IID p-values distributed uniformly on $[0,1]$); typically, it limits the probability or frequency of false alarms. Our procedures, being based on conformal prediction, satisfy various properties of validity automatically. Efficiency refers to their behaviour when the IID assumption is violated, such as raising an alarm soon after the change point, and achieving it is often an art.

2 Exchangeability martingales

In this section we will define conformal test martingales, which will be our key tool (see, e.g., Vovk et al. 2005 for further details). Let $z_1, z_2, \dots$ be an infinite sequence of observations (typically pairs $z_n = (x_n, y_n)$), elements of a measurable space $\mathbf{Z}$, our *observation space*. An *inductive conformity measure* is a measurable function A mapping any observation $z \in \mathbf{Z}$ to a real number $\alpha = A(z)$, the *conformity score* of z; the conformity score may also depend on some prior data (in a measurable manner). Given such an A, the *conformal p-value* computed from observations $(z_1, \dots, z_n) \in \mathbf{Z}^*$ is

$$p_n := \frac{|\{i \mid \alpha_i < \alpha_n\}| + \theta_n |\{i \mid \alpha_i = \alpha_n\}|}{n}, \tag{1}$$

where i ranges over $\{1, \dots, n\}$, $\alpha_1 = A(z_1), \dots, \alpha_n = A(z_n)$ are the conformity scores for $z_1, \dots, z_n$, and θ_n is a random number distributed uniformly on the interval $[0,1]$.

To state the property of validity of conformal p-values in a strong form, we need to relax the IID assumption. For any natural number N, a probability measure P on $\mathbf{Z}^N$ is *exchangeable* if it is invariant with respect to permutations in the following sense: for any measurable set $E \subseteq \mathbf{Z}^N$ and any permutation π of the set $\{1, \dots, N\}$,

$$P((z_1, \dots, z_n) \in E) = P\left((z_{\pi(1)}, \dots, z_{\pi(n)}) \in E\right).$$

The following property of validity is proved in, e.g., Vovk et al. [2005] (Theorem 8.2).

Proposition 1. *Suppose N is a natural number, observations $z_1, \dots, z_N$ are exchangeable (i.e., generated from an exchangeable probability measure), $\theta_1, \dots, \theta_N$ are IID, distributed uniformly on $[0,1]$, and independent of the observations. Then the p-values $p_1, \dots, p_N$ defined by (1) are IID and distributed uniformly on $[0,1]$.*

A probability measure P on $\mathbf{Z}^\infty$ (i.e., over the infinite sequences $(z_1, z_2, \dots)$) is *exchangeable* if, for any natural number N, its restriction to the first N observations $(z_1, \dots, z_N)$ is exchangeable. According to de Finetti's representation theorem (see, e.g., Schervish 1995, Theorem 1.49), assuming the observation space $\mathbf{Z}$ is a Borel space, every exchangeable probability measure is a mixture of *IID measures* (i.e., probability measures on the infinite sequences $\mathbf{Z}^\infty$ of observations under which the observations are IID). This makes the assumptions

of IID and exchangeability almost indistinguishable for $\mathbf{Z}^\infty$; however, the difference is essential for finite sequences of observations.

We will test the IID assumption by testing exchangeability. The idea is to gamble against the uniform distribution of the conformal p-values $(p_1, p_2, \dots) \in [0,1]^\infty$. A *betting martingale* is a measurable function $F : [0,1]^* \to [0,\infty]$ such that $F(\square) = 1$ ($\square$ being the empty sequence) and, for each sequence $(u_1, \dots, u_{n-1}) \in [0,1]^{n-1}$, $n \geq 1$,

$$\int_0^1 F(u_1, \dots, u_{n-1}, u) \, \mathrm{d}u = F(u_1, \dots, u_{n-1})$$

(with $(u_1, \dots, u_{n-1}) := \square$ for $n = 1$). *Conformal test martingales* are defined by

$$S_n = F(p_1, \dots, p_n), \quad n = 0, 1, \dots,$$

where $p_1, p_2, \dots$ are the p-values computed by (1), with $(\theta_1, \theta_2, \dots)$ distributed uniformly in $[0,1]^\infty$ and independent of the observations.

Conformal test martingales S_n are bona fide martingales in the sense of satisfying

$$\mathbb{E}(S_n \mid S_1, \dots, S_{n-1}) = S_{n-1} \tag{2}$$

for all $n \geq 1$, provided the observations are exchangeable, or, as we will say, they are *exchangeability martingales*; we only consider exchangeability martingales that are nonnegative and have 1 as their initial value. We interpret S_n as the amount of evidence found against our null hypothesis, the IID assumption, after seeing the first n observations. In betting terms, it is the capital of a tester who starts from 1 and gambles against the null hypothesis.

Remark 2. Even in the case of a finite horizon, where we have N observations $z_1, \dots, z_N$, we will have (2) for $n = 1, \dots, N$. The case of finite horizon is important for us since permuting a dataset ensures its exchangeability but not the IID assumption. And because of the exchangeability, all conformal test martingales constructed in this paper will always lose capital in the ideal setting.

By Ville's inequality (Ville 1939, p. 100, Shiryaev 2019, Theorem 7.3.1), for any constant $c > 1$,

$$\mathbb{P}(\exists n : S_n \geq c) \leq 1/c.$$

This is a property of validity for exchangeability martingales. If, for example, we raise an alarm when S_n exceeds the threshold of 100, the probability of ever raising a false alarm will not exceed 1%.

3 Ville procedure in action

In this section we discuss a possible informal scheme for deciding when to retrain a predictor. As an example, we consider the Wine Quality dataset [Cortez et al., 2009], available at the UCI Machine Learning repository [Dua and Graff, 2017]. The label is the score between 0 and 10 reflecting the wine quality, and there are eleven attributes (such as residual sugar and alcohol) that may be useful for

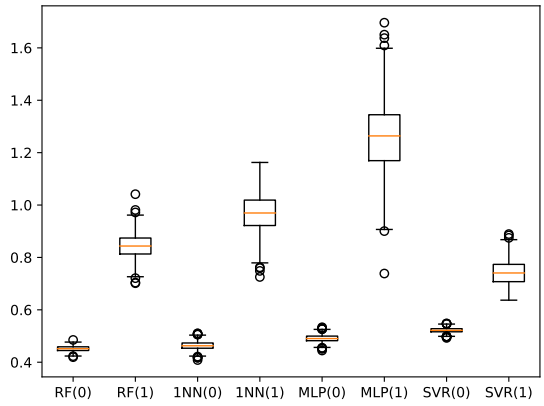

Figure 1: The accuracies of various prediction algorithms on the Wine Quality dataset, as described in the main text.

predicting the label. We consider the problem of predicting the label given the sample (consisting of the eleven attributes) as regression problem. The label is mostly between 4 and 7.

The dataset consists of two parts, 4898 white wines and 1599 red wines. We randomly choose a subset of 1599 white wines and refer to it as *test set 0*, and the remaining white wines (randomly permuted) will be our *training set*. All 1599 red wines form our *test set 1*; therefore we have two test sets of equal sizes.

We will be interested in two scenarios. In scenario 0 we train our prediction algorithm on the full training set and test the resulting model on test set 0. We can expect the quality of prediction to be good, since the training and test set are coming from the same distribution. If we normalize the data by using `StandardScaler` in `scikit-learn` [Pedregosa et al., 2011], we can achieve the test MAD (mean absolute deviation) of about 0.45. The best values achieved by the algorithms implemented in `scikit-learn` for the default values of the parameters are given in Figure 1, where RF stands for Random Forest, 1NN for 1-Nearest Neighbour, MLP for Multilayer Perceptron, and SVR for Support Vector Regression. The relevant boxplots are those marked with 0 in parentheses; the boxplots are over 1000 simulations and shown to give an idea of the dependence on the seed used for the random number generator (the seed affects the split into the training and test sets and may be used internally by the prediction algorithm, e.g., by Random Forest). The algorithms are ordered by their performance in scenario 0. In scenario 1 we test the same trained model on test set 1; since its distribution is different (from the very start of the test set), the resulting test MAD will be significantly worse, as indicated in Figure 1 by the boxplots marked with 1 in parentheses.

To detect a possible change point in the test set (which does not exist in test set 0 and is the very start in test set 1), the training set of 3299 white

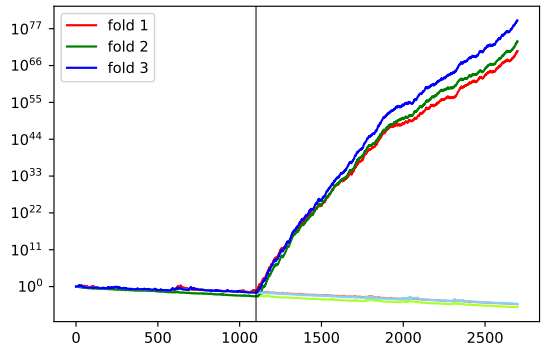

Figure 2: Paths of the three conformal test martingales (based on Random Forest, conformity measure $y - \hat{y}$, and Simple Jumper) applied to the Wine Quality dataset as described in the main text.

wines is randomly split into three *folds* of nearly equal sizes, 1100, 1100, and 1099. We use each fold in turn as the *calibration set* and the remaining folds as the *training set proper*. For each fold $k \in \{1, 2, 3\}$ we train a prediction algorithm on the training set proper and run an exchangeability martingale (based, in some way, on the resulting model) on the $1100 + 1599 = 2699$ observations $z'_1, \ldots, z'_{1100}, z''_1, \ldots, z''_{1599}$, where $z'_1, \ldots, z'_{1100}$ is the calibration set and $z''_1, \ldots, z''_{1599}$ is the test set (for one of the folds, 1100 should be replaced by 1099, but we will ignore this in our discussion). This way we obtain three paths, plots of the values of the exchangeability martingales vs time. We still have two scenarios: scenario 0 uses test set 0, and scenario 1 uses test set 1; thus we have 6 paths overall.

Results for a specific conformity measure (based on Random Forest) and betting martingale (to be defined shortly) are shown in Figure 2. For each fold we have a conformal test martingale, and paths of these three martingales are shown using different colours, as indicated in the legend. We have two paths for each of the martingales: the one over the calibration set and test set 0 (with the part over test set 0 shown in lighter colours), and the other over the calibration set and test set 1. The behaviour of the three martingales in scenario 1 is similar, all achieve a high value, of the order of magnitude about 10^{70}, and start rising sharply soon after the change point (shown as the thin vertical line); the presence of the change point becomes obvious shortly after it happens.

The conformity measure used in Figure 2 is

$$\alpha_i := y_i - \hat{y}_i, \tag{3}$$

where $\hat{y}_i$ is the prediction for the label y_i of the sample x_i produced by the model found from the training set proper. (Intuitively, it is not obvious why (3) should be interpreted as a measure of conformity, but in conformal testing the difference between conformity and nonconformity often disappears; in particu-

Algorithm 1 Simple Jumper ($(p_1, p_2, \dots) \mapsto (S_1, S_2, \dots)$)

$C_{-1} := C_0 := C_1 := 1/3$
$C := 1$
for $n = 1, 2, \dots$:
    for $\epsilon \in \{-1, 0, 1\}$: $C_\epsilon := (1 - J)C_\epsilon + (J/3)C$
    for $\epsilon \in \{-1, 0, 1\}$: $C_\epsilon := C_\epsilon f_\epsilon(p_n)$
    $S_n := C := C_{-1} + C_0 + C_1$

lar, the Simple Jumper, to be defined momentarily, will work equally well with $-\alpha_i$ in place of α_i.)

To transform the p-values $p_1, p_2, \dots$ computed by (1) into a conformal test martingale we use the betting martingale

$$F(p_1, \dots, p_n) := \int \left(\prod_{i=1}^{n} f_{\epsilon_i}(p_i) \right) \mu(\mathrm{d}(\epsilon_0, \epsilon_1, \dots)), \tag{4}$$

where

$$f_\epsilon(p) := 1 + \epsilon(p - 0.5) \tag{5}$$

and μ is the following Markov chain with state space $\{-1, 0, 1\}$: the initial state is $\epsilon_0 = -1, 0, 1$ with equal probabilities, and the transition function prescribes maintaining the same state with probability $1 - J$ and, with probability J, choosing a random state from the state space $\{-1, 0, 1\}$. The intuition is that at each step i we are using one of the *betting functions* (5); f_{-1} corresponds to betting on small values of p_i, f_1 corresponds to betting on large values of p_i, and f_0 corresponds to not betting. Since $\int f_\epsilon = 1$ for all ϵ, (4) is really a martingale. For simplicity, in this paper we only consider f_ϵ that are linear functions, which implies $f_\epsilon(0.5) = 1$; (5) is a general expression for such functions. We always set $J := 0.01$. Therefore, we start from the uniform allocation of the initial capital to the states and usually continue betting in the same way as on the previous step. We will refer to this betting martingale as the *Simple Jumper* (it is a simplification of the Sleepy Jumper described in Vovk et al. 2005, Section 7.1). The pseudocode for the Simple Jumper applied to the p-values (1) is given as Algorithm 1.

Replacing the conformity measure (3) by its absolute value,

$$\alpha_i := |y_i - \hat{y}_i| , \tag{6}$$

leads to a slower growth, as illustrated in the left panel of Figure 3, and in this paper we concentrate on the signed version (3). In the case of fold 1 (the red line), we can see a pronounced phenomenon of "decay" setting in around observation 2000; as it were, the new distribution (corresponding to red wines) becomes a new normal, and the growth of the martingale stops.

In the case of Random Forest, there is an interesting alternative to the conformity measure (3). With the default values of the parameters in `scikit-learn`,

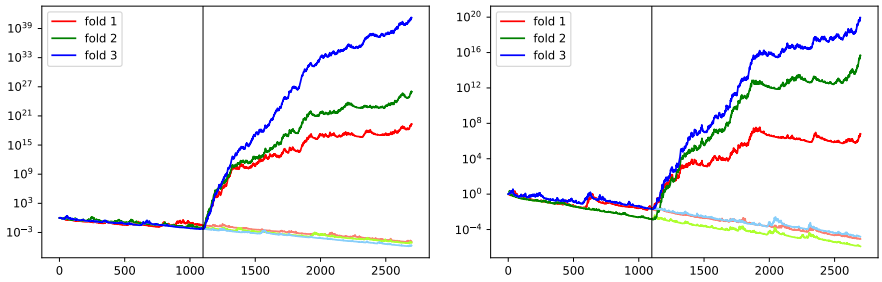

Figure 3: Analogues of Figure 2 with the conformity measure $y - \hat{y}$ replaced by $|y - \hat{y}|$ (left panel) or the PIT conformity measure $F_i(y_i)$ (right panel), with $\hat{y}_i$ and F_i produced by Random Forest.

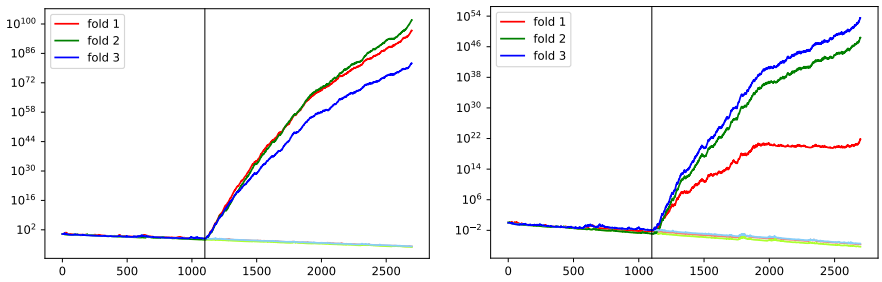

Figure 4: Analogues of Figure 2 with $\hat{y}_i$ produced by Multilayer Perceptron (left panel) or Support Vector Regression (right panel).

its prediction $\hat{y}_i$ is computed by averaging the predictions produced by 100 decision trees. The *PIT conformity measure* (where PIT stands for "probability integral transform") is $\alpha_i := F_i(y_i)$, where F_i is the empirical distribution function determined by the predictions $\hat{y}_i^j$ produced by the decision trees $j = 1, \dots, 100$. In other words,

$$\alpha_i := \left| \left\{ j \mid \hat{y}_i^j \le y_i \right\} \right| / 100.$$

The results are shown in the right panel of Figure 3; the growth of the conformal test martingales in scenario 1 becomes even weaker than for the conformity measure (6).

The conformity measure (3) can be applied to any regressor. Figure 4 (left and right panels) and the left panel of Figure 5 are the analogues of Figure 2 for Multilayer Perceptron, Support Vector Regression, and 1-Nearest Neighbour. Notice that our martingales detect lack of exchangeability best in situations where it matters most: according to Figure 1, the accuracy of Multilayer Perceptron suffers most of the dataset shift, and according to the left panel of

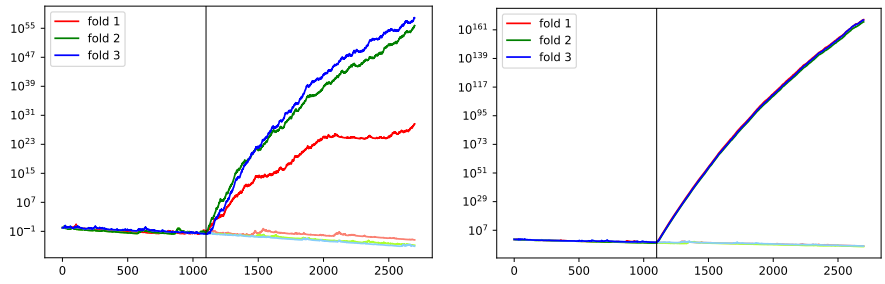

Figure 5: Left panel: Analogue of Figure 2 with $\hat{y}_i$ produced by 1-Nearest Neighbour. Right panel: Paths of the three conformal test martingales based on the nearest-distance conformity measure (ignoring the labels and their predictions).

Figure 4, the conformal test martingales based on this algorithm achieve the fastest growth. In some cases (as for fold 1 in the right panel of Figure 4) the phenomenon of decay is even more pronounced than in the left panel of Figure 3.

The procedure described in this section may be used for deciding when to retrain: e.g., we may decide to retrain when one of the three martingales exceeds the threshold 100. In this case, the probability of ever raising a false alarm never exceeds 3%.

Detecting covariate shift

The conformity measures that we have used so far in this section were functions of the true labels and predictions. If the underlying algorithm is robust to moderate covariate shift, the resulting testing procedures will not detect deviations from exchangeability under such covariate shift. It makes sense since no retraining is required in this case.

The right panel of Figure 5 shows the results for the conformity score α of an observation (x, y) (in the calibration or test set) computed as the Euclidean distance from x to the nearest sample in the training set proper. This conformity measure completely ignores the labels but still achieves spectacular values for conformal test martingales based on it. The two panels of Figure 5 show the paths of conformal test martingales based on the Nearest Neighbour idea, but the left one is restricted to evaluating the quality of predictions, and the restriction shows in a slower growth.

4 CUSUM and Shiryaev–Roberts procedures for change detection

A well-known disadvantage of the Ville procedure is that it becomes less and less efficient as time passes and the value of the martingale goes down, which in-

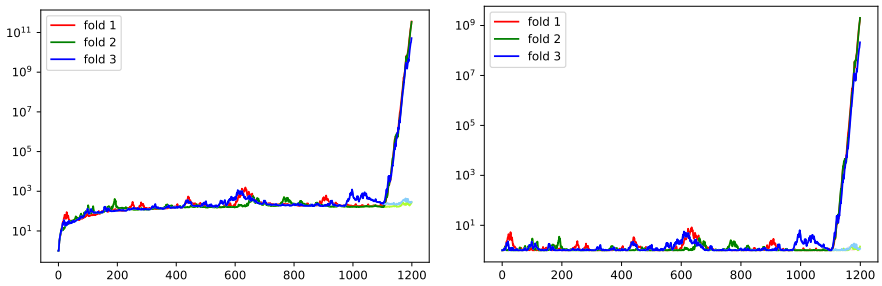

Figure 6: The Shiryaev–Roberts (left panel) and CUSUM (right panel) statistics over the first 1200 observations of the combined calibration and test sets for Multilayer Perceptron and the conformity measure $y - \hat{y}$.

evitably happens in the absence of change points: cf. scenario 0 (lighter colours) in Figures 2–5. It may take a long time to recover the lost capital and so to detect the change point. We can say that, whereas the Ville procedure may be suitable during the first stages of the testing process (while our capital is still not negligible), it is less suitable for later stages.

Let S be an exchangeability martingale that never takes value 0 (such are all martingales considered in the previous section). The *CUSUM procedure* [Page, 1954] raises an alarm at the time

$$\tau := \min\{n \mid \gamma_n \geq c\}, \text{ where } \gamma_n := \max_{i=0,\dots,n-1} \frac{S_n}{S_i} \tag{7}$$

and $c > 1$ is the parameter of the procedure. The Shiryaev–Roberts procedure [Shiryaev, 1963, Roberts, 1966] modifies this by replacing the maximum with a sum:

$$\sigma := \min\{n \mid \psi_n \geq c\}, \text{ where } \psi_n := \sum_{i=0}^{n-1} \frac{S_n}{S_i}. \tag{8}$$

In both (7) and (8), $\min \emptyset := \infty$. The maxima γ_n in (7) (with $\gamma_0 := 0$) are called the *CUSUM statistics*, and the sums ψ_n (with $\psi_0 := 0$) in (8) are called the *Shiryaev–Roberts statistics*.

It is known that, under the null hypothesis (the IID assumption in this context), $\mathbb{E}(\sigma) \geq c$ (see, e.g., Vovk 2021, Proposition 4.2); moreover, $\mathbb{E}(\sigma)$ equals c if we ignore the possibility of the Shiryaev–Roberts statistics overshooting the threshold c. Since $\tau \geq \sigma$, we also have $\mathbb{E}(\tau) \geq c$. It is also shown in Vovk [2021] (Proposition 4.4) that, when the Shiryaev–Roberts procedure is applied repeatedly, the relative frequency of false alarms will not exceed $1/c$ in the long run (this statement is informative only when $\mathbb{E}(\sigma) = \infty$, since otherwise it follows from $\mathbb{E}(\sigma) \geq c$ and Kolmogorov's strong law of large numbers).

Figure 6 shows the evolution of the Shiryaev–Roberts and CUSUM statistics over the calibration set and the first 100 observations of the test set (in scenarios

c. measure	Ville	CUSUM	SR
$y - \hat{y}$, RF	70 [56, 87]	68 [57, 85]	64 [52, 80]
$y - \hat{y}$, 1NN	92 [69, 138]	93 [70, 137]	86 [65, 126]
$y - \hat{y}$, MLP	55 [44, 72]	55 [45, 72]	51 [42, 67]
$y - \hat{y}$, SVR	156 [100, 294]	156 [102, 297]	142 [94, 270]
$\lvert y - \hat{y}\rvert$, RF	222 [131, 538]	221 [130, 501]	203 [119, 450]
$\lvert y - \hat{y}\rvert$, 1NN	192 [114, 416]	188 [115, 412]	172 [106, 342]
$\lvert y - \hat{y}\rvert$, MLP	88 [62, 143]	88 [62, 142]	81 [58, 130]
$\lvert y - \hat{y}\rvert$, SVR	720 [300, ∞]	721 [303, ∞]	562 [272, ∞]
$F(y)$, RF	199 [133, 375]	196 [135, 373]	177 [123, 333]
ND	29 [26, 32]	29 [27, 31]	27 [25, 29]

Table 1: The median delay and the interquartile intervals for the delay for different conformity measures and different procedures for change-point detection, as described in the main text.

1 and 0) for the conformity measure $y - \hat{y}$, $\hat{y}$ being computed by Multilayer Perceptron. The CUSUM statistic is shown in the form $\max(\gamma_n, 1)$ (since the values $\gamma_n < 1$ are less interesting). Both statistics will raise an alarm in scenario 1 for c up to 10^9.

Let us now compare the performance of different prediction algorithms and conformity measures for change-point detection more systematically, still concentrating on the Wine Quality dataset. The betting martingale is, as before, the Simple Jumper (with $J := 0.01$). Our results will be summarized in Table 1.

We randomly choose a subset of 1000 white wines as training set, a disjoint subset of 1000 white wines as calibration set, and a subset of 1000 red wines, all three subsets randomly ordered. We train various prediction algorithms, labelled with the same abbreviations as in Figure 1, on the training set, use the conformity measure given in the column "c. measure" to obtain a conformal test martingale, as described earlier, and run the Ville, CUSUM, and Shiryaev–Roberts (SR) procedures with the thresholds $c = 10^2, 10^4, 10^6$, respectively, on the calibration set continued by the test set. The alarm is raised (i.e., the threshold is exceeded) on the test set, in the vast majority of cases, and we define the *delay* as the ordinal number of the observation in the test set at which the alarm happens. Table 1 reports the median delay accompanied by the interquartile intervals for the delays (i.e., the intervals whose end-points are the lower and upper quartiles) for ten different conformity measures (already described earlier) and 1000 simulations; ND stands for the nearest distance conformity measure discussed at the end of Section 3. A delay of ∞ refers to the case where no alarm is ever raised. One striking feature is how much the conformity measures $|y - \hat{y}|$ lose as compared with $y - \hat{y}$. The nearest distance conformity measure (detecting covariate shift) is quickest in raising alarms.

5 Our proposed procedures

In this section we propose two procedures for deciding when to retrain, which we call the fixed training schedule and the variable training schedule. They are based on exchangeability martingales as described in the previous sections. Both schedules use the Ville procedure at the beginning, but then apply different strategies for deciding when to retrain. For simplicity, instead of free parameters we will often use specific numbers (such as splitting the dataset into 3 folds; we do not claim that 3 is always the best possible number of folds). Fix a prediction algorithm (such as Random Forest) and a conformity measure (such as $y - \hat{y}$). Training the algorithm on a dataset then gives both a predictor (trained model) and an exchangeability martingale (conformal test martingale).

Variable training schedule

Train the prediction algorithm on the full training set, and start running the resulting predictor on the stream of test observations. Decide on the target lifespan of the predictor, say C (this is a way of setting the sensitivity of our procedure).

1. As the first step of the testing component, split the training set into 3 approximately equal folds, 1, 2, and 3.
2. For each $k \in \{1, 2, 3\}$:
   - Train the prediction algorithm on the folds different from k; the resulting predictor gives rise to a conformal test martingale S^k, as described earlier.
   - Start running the conformal test martingale S^k on fold k (randomly permuted) and then on the stream of test observations in chronological order. Run the Shiryaev–Roberts statistic $\psi_n^k := \sum_{i=0}^{n-1} S_n^k / S_i^k$ on top of S^k.
3. When two out of the three martingales S^k raise an alarm at level 100, $S_n^k \geq 100$, retrain (i.e., retrain when $\left|\{k \mid S_n^k \geq 100\}\right| \geq 2$).
4. When two out of the three Shiryaev–Roberts statistics ψ^k raise an alarm at level C, $\psi_n^k \geq C$, retrain.

Formally, the time of retraining when using the variable schedule is

$$\min\left\{n \mid \left|\{k \mid S_n^k \geq 100\}\right| \geq 2 \text{ or } \left|\{k \mid \psi_n^k \geq C\}\right| \geq 2\right\}.$$

The variable training schedule is based on the fact that, in the ideal setting (in which the IID assumption holds), $\mathbb{E}(\sigma) \geq C$ (where σ is defined in (8)) and $\mathbb{E}(\sigma) \approx C$ when overshoots do not play a big role. Suppose the target lifespan

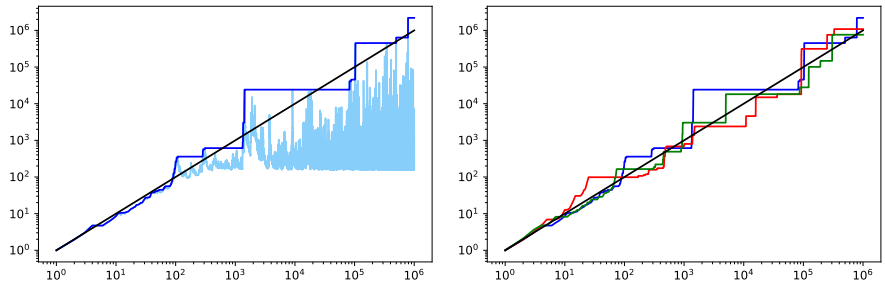

Figure 7: Left panel: Behaviour of the Shiryaev–Roberts statistic in the ideal setting (of simulated p-values). The path of the statistic itself is shown in light blue and the path of its maximum process in dark blue. Right panel: Behaviour of the maximum process of the Shiryaev–Roberts statistic for three seeds of the random number generator in the ideal setting.

is $C = 10^6$. The left panel of Figure 7 shows the behaviour of the SR statistic ψ_n (see (8)) and its *maximum process*

$$\psi_n^* := \max_{i \in \{0,\dots,n\}} \psi_i \tag{9}$$

in the *ideal setting*, where the p-values are independent and uniformly distributed on $[0, 1]$ (as they are under the IID assumption). The black line is the compensator n of the submartingale ψ_n, in the sense of $\psi_n - n$ being a martingale. The typical behaviour of ψ_n is illustrated by the light blue line in Figure 7 (left panel), which is very different from its expectation, the black line. The right panel of Figure 7 shows three paths of the maximum process ψ_n^*.

Figure 8 gives the histogram for the times of the alarm, $\min\{n \mid \psi_n \geq 10^6\}$, over 10^5 simulations in the ideal setting. In numbers, rounded to the nearest 10^4: the mean time to the alarm is 1.125×10^6, which exceeds 10^6 because of overshoots, with the large standard deviation of 1.123×10^6; the median is 0.781×10^6, and the interquartile interval is $[0.320 \times 10^6, 1.561 \times 10^6]$. These figure and numbers illustrate the property of validity of the variable training schedule: the expected lifespan of the trained predictor is indeed around $C = 10^6$. However, the variability of the lifespan, even in the ideal setting, may be a problem in some applications, if retraining is a complicated process that needs to be planned in advance.

Fixed training schedule

In the fixed training schedule we decide in advance when we would like to retrain our predictor, and change our plans only when we have significant evidence that the distribution of the data has changed (presumably, this will be a rare event, and so retraining will typically happen at the prespecified time). Since we do not

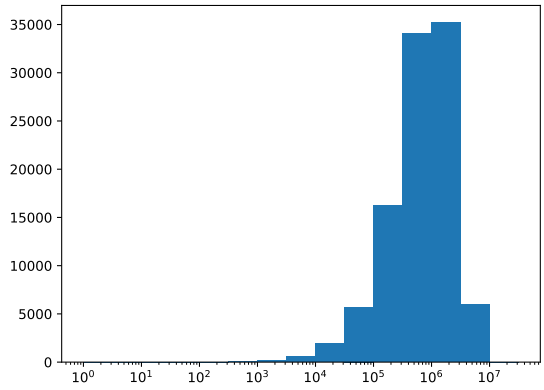

Figure 8: The histogram for the time of alarm for the Shiryaev–Roberts statistic for the target lifespan 10^6 and 10^5 simulations in the ideal setting (of simulated p-values).

f	alarms	99.9% confidence interval
3.5×10^5	969	$[0.87\%, 1.08\%]$
3.6×10^5	939	$[0.84\%, 1.04\%]$
3.7×10^5	905	$[0.81\%, 1.01\%]$
3.8×10^5	866	$[0.77\%, 0.97\%]$
4×10^5	820	$[0.73\%, 0.92\%]$
5×10^5	635	$[0.56\%, 0.72\%]$

Table 2: The confidence intervals, based on 10^5 simulations, for the probability of false alarm in the fixed training schedule for lifespan $C = 10^6$ and various choices of the threshold f.

have any theoretical property of the form $\mathbb{E}(\tau) \approx C$ (the inequality $\mathbb{E}(\tau) \geq C$ still holds, of course, but it is very conservative), our property of validity for the fixed training schedule will be computational.

The fixed training schedule also starts from the target lifespan of the predictor, C.

1. Split the training set into 3 approximately equal folds, 1, 2, and 3.
2. For each $k \in \{1, 2, 3\}$:
   - Train the prediction algorithm on the folds different from k getting a conformal test martingale S^k.
   - Start running the conformal test martingale S^k on fold k (randomly permuted) and then on the stream of test observations in chronological order. Run the CUSUM statistic $\gamma_n^k := \max_{i<n} S_n^k / S_i^k$ on top of

each S^k.

3. When two out of the three martingales S^k raise an alarm at level 100, $S_n^k \geq 100$, retrain.

4. When two out of the three CUSUM statistics γ^k raise an alarm at level $f = f(C)$, $\gamma_n^k \geq f$, retrain.

The value $f = f(C)$ should be chosen in such a way that the probability of one CUSUM statistic reaching level f should not exceed 1% in the ideal setting. Let us consider, for concreteness, $C = 10^6$. One possibility is to set f to the 99th percentile, over a large number K of simulations, of the empirical distribution function of the maximum attained by the CUSUM statistic over the random path of the Simple Jumper between 0 and 10^6 in the ideal setting. Setting $K := 10^5$, we obtain, for seed 0 of the NumPy random number generator, 3.4798×10^5 as the 99th percentile. To ensure the validity of f, we have computed the exact confidence intervals [Clopper and Pearson, 1934] for several round values for f at confidence level 99.9% (to allow for multiple hypothesis testing, as we are looking at several candidates for $f = f(C)$). For each of those f, we computed the number of the paths of γ_n that trigger an alarm (which is a false alarm, since we are in the ideal setting) at level f over $n = 1, \dots, 10^6$; these numbers are given in the column "alarms" in Table 2. The confidence intervals are computed using the R package `binom` [Dorai-Raj, 2014]. For example, according to Table 2, we can set $f := 4 \times 10^5$, since the corresponding confidence interval is a subset of $[0, 1\%]$. (It is sometimes argued that the Clopper–Pearson confidence intervals are too conservative and less conservative approximate intervals are desirable, but in our current context there is no need to sacrifice exact validity since the number of simulations is under our control.)

The overall probability of the fixed training schedule raising a false alarm is at most 3%. This follows from the Ville component 3 raising a false alarm with probability at most 1.5% and the CUSUM component 4 raising a false alarm with probability at most 1.5%.

Practical aspects

For both schedules, the predictions provided to the users of our prediction algorithm should be computed from the full training set, of course. The predictions computed from two out of the three folds should only be used for monitoring the validity of the IID assumption.

We could marginally improve the sensitivity of our procedures (for both variable and fixed training schedules) by resetting the conformal test martingales S^k, $k \in \{1, 2, 3\}$, to 1 when they leave fold k and enter the test set, since we know that the exchangeability assumption holds for fold k.

Not all observations on which the trained predictor is run are necessarily included in the test stream. In general, we have a training set, a test stream, and an exploitation stream.

We should also be careful about including observations in the test stream in order not to violate exchangeability for irrelevant reasons. For example, new test observations can be added in randomly shuffled batches of reasonable sizes.

6 Conclusion

In this paper we have discussed using conformal prediction for testing exchangeability (this is the only known way of constructing non-trivial exchangeability martingales) and then for deciding when a prediction algorithm should be retrained. We have not discussed the process of retraining, which is an interesting direction of research. A natural question is: which part of the available data should be used for retraining? One possible approach is to use an exchangeability martingale (trained on recent data) backwards: starting from the recent data, move into the past until the martingale detects loss of exchangeability.

This paper only scratched the surface of various specific kinds of dataset shift, such as concept shift and covariate shift. Those kinds will require adapting the methods proposed in this paper and developing new ones.

Acknowledgments

Thanks to Emily Hector for useful comments and to the three anonymous referees of the conference version of this paper for their suggestions for improving the presentation. This research was partially supported by Amazon and Stena Line.

References

C. J. Clopper and Egon S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26:404–413, 1934.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47:547–553, 2009.

Sundar Dorai-Raj. Binomial confidence intervals for several parameterizations. Version 1.1-1, 2014.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Maayan Harel, Koby Crammer, Ran El-Yaniv, and Shie Mannor. Concept drift detection through resampling. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the Twenty Ninth International Conference on Machine Learning. Proceedings of Machine Learning Research*, volume 32, pages 1009–1017, 2014.

Erich L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Springer, New York, revised first edition, 2006.

Ewan S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence, editors. *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA, 2009.

S. W. Roberts. A comparison of some control chart procedures. *Technometrics*, 8:411–430, 1966.

Mark J. Schervish. *Theory of Statistics*. Springer, New York, 1995.

Albert N. Shiryaev. On optimum methods in quickest detection problems. *Theory of Probability and Its Applications*, 8:22–46, 1963.

Albert N. Shiryaev. *Probability-2*. Springer, New York, third edition, 2019.

Jean Ville. *Etude critique de la notion de collectif*. Gauthier-Villars, Paris, 1939.

Vladimir Vovk. Testing randomness online. *Statistical Science*, 36:595–611, 2021.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.

A Middlegame

A weakness of both variable and fixed training schedules is that, in detecting deviations from exchangeability, they concentrate on the opening and the endgame of the testing process, to use a chess metaphor. The Ville procedure quickly loses its efficiency as the tester's capital S_n becomes very small, and the CUSUM or Shiryaev–Roberts procedures are not efficient closer to the beginning of the test stream since the thresholds they use are comparable with the target lifespan of the predictor. There is a danger that in the middlegame neither opening nor endgame procedures work well.

In this appendix we will discuss a procedure intermediate between the opening Ville procedure and the endgame CUSUM procedure (we will concentrate on the fixed training schedule); in fact, the intermediate ("middlegame") procedure will be also a CUSUM procedure, but the horizontal barrier implicitly

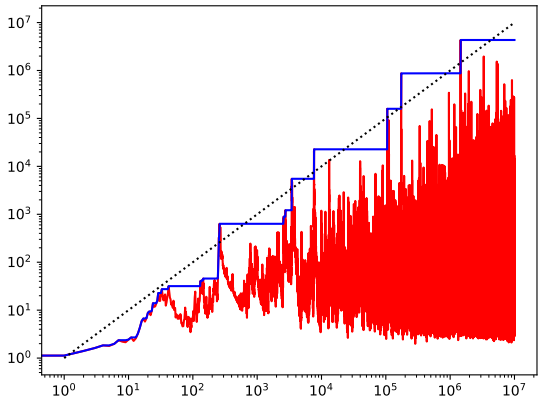

Figure 9: The maximum of 100 CUSUM paths in red and its maximum process in blue in the ideal setting.

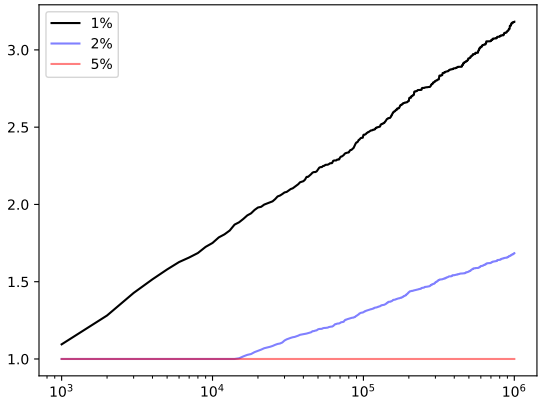

Figure 10: The slopes in the middlegame in the ideal setting, as described in the main text.

used in (7) and Section 5 will be replaced by a more complex barrier. Figure 9 shows in red the maximum of 100 simulated CUSUM paths, and it suggests that a reasonable barrier is a straight line with slope 1 in the loglog representation; in the original (x, y)-axes the barrier has the equation $y = cx$ (a straight line passing through the origin). The blue line in Figure 9 is defined as in (9).

Figure 10 summarizes the empirical performance of various slopes for such barriers. A path of the CUSUM statistic γ_n over $n = 1, \ldots, N$ will trigger an alarm for a barrier $y = cx$ if $\gamma_n \geq cn$ for some $n \leq N$. Let us generate a large number K ($K = 10^5$ in the case of Figure 10 and Table 3) of paths of the

c	alarms	99.9% confidence interval
3.2	988	$[0.89\%, 1.10\%]$
3.3	958	$[0.86\%, 1.06\%]$
3.4	930	$[0.83\%, 1.03\%]$
3.5	901	$[0.81\%, 1.00\%]$
4	793	$[0.70\%, 0.89\%]$
5	622	$[0.54\%, 0.71\%]$

Table 3: The analogue of Table 2 for the middlegame.

CUSUM statistic in the ideal setting. For each N, let c_N be the number such that 1% of the K paths trigger an alarm (a false one) for the barrier $y = c_N x$. (The definite article "the" in "the number" is almost justified for large K.) The black line in Figure 10 plots c_N vs N for $N \in \{1000, 2000, \dots, 10^6\}$; it looks like a straight line. The blue line, corresponding to the 2% frequency of false alarms, also looks straight, except that it cannot go under $y = 1$. This allows us to estimate the right slope c for a given target lifespan of our predictor.

For example, if the target lifespan is $N = 10^6$, we can see from Figure 10 that $c_N \approx 3.2$ for 1% (more precise values are 3.183 for 1% and 1.685 for 2%). To find a suitable barrier, we need a confidence interval for the probability of a false alarm at a suitable confidence level that is completely inside $[0, 1\%]$. According to Table 3, we can take the barrier $y = 4x$ (for the confidence level 99.9%).

Let us see how our middlegame strategy compares with the strategies that we proposed earlier for the opening and endgame. Suppose the target lifespan of the predictor is 10^6, and we follow the fixed training schedule (the comparison is easier in this case). If we use 4×10^5 as the alarm threshold in the endgame (see Table 2) and $4n$ as the alarm threshold at step n in the middlegame (see Table 3), the former rule dominates the latter if $n > 10^5$ (meaning that the former rule triggers an alarm whenever the latter does), while the latter rule dominates the former if $n < 10^5$. We can say that, with these rules, the middlegame ends and the endgame starts at step 10^5.

To compare the rules used in the opening and the middlegame, we need to understand how the tester's capital S_n evolves during these stages of the testing process. The left panel of Figure 11 gives three typical paths (those corresponding to the first three seeds for the NumPy random number generator) of the Simple Jumper's capital S_n in the ideal setting. On the log-scale for the capital, they look linear and very close. The right panel of Figure 11 is the histogram of the final values S_{10^6} of the Simple Jumper over 10^5 simulations. In numbers, the median final capital is $10^{-1720.0}$, and the interquartile interval is $[10^{-1731.6}, 10^{-1708.1}]$. Since the Simple Jumper is a martingale, the true expectation of its final value is 1, but this fact is not visible in the histogram at all (we need many more simulations for it to become visible). If the change point is at step n, after which the Simple Jumper S starts a quick growth, the

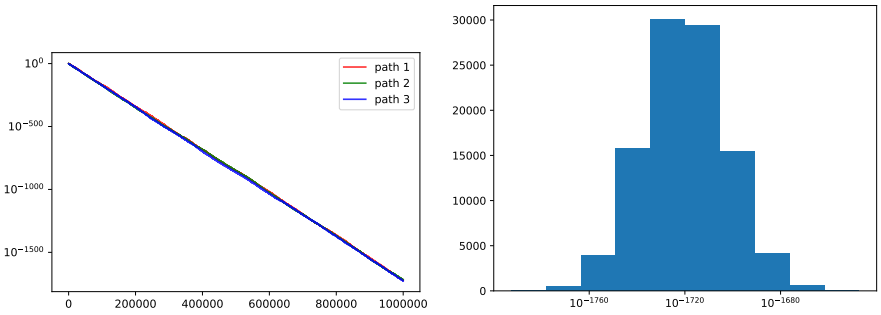

Figure 11: Left panel: Three typical paths of the Simple Jumper's capital in the ideal setting. Right panel: The histogram of the final values of the Simple Jumper's capital based on 10^5 simulations in the ideal setting.

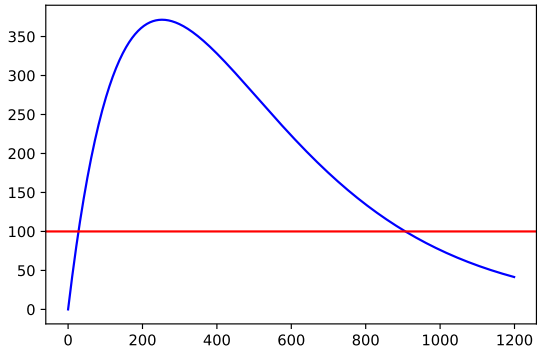

Figure 12: Opening vs middlegame.

opening Ville procedure triggers an alarm when $S_n \geq 100$, whereas the middlegame CUSUM triggers an alarm around the time when $S_n/10^{-0.00172n} \geq 4n$. Solving numerically the equation $4n \times 10^{-0.00172n} = 100$, we obtain 906.7 as its second solution (see Figure 12, in which the blue line is the left-hand side of the equation as function of n, and the red line is the right-hand side). Therefore, we can regard the boundary between the opening and the middlegame to be approximately 10^3; in the middlegame defined this way the rule of triggering an alarm when $\gamma_n \geq 4n$ can be said to dominate the Ville rule of triggering an alarm when $S_n \geq 100$.

Remark 3. In principle we can also use the Shiryaev–Roberts procedure in the middlegame, and even in the fixed-schedule endgame. In this paper, however, we concentrate on the more popular and intuitive CUSUM procedure.

Remark 4. On the other hand, we can simplify the testing process by including

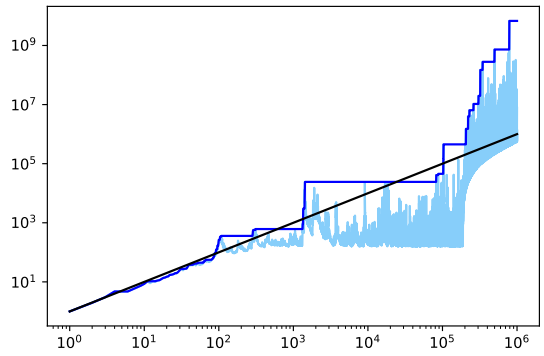

Figure 13: The Shiryaev–Roberts statistic, implemented naively, over the first 10^6 observations.

only the middlegame and discarding the opening Ville and endgame procedures altogether.

B Avoiding numerical problems

If the Shiryaev–Roberts statistic ψ_n is implemented on a modern computer directly using the formula in (8), we will obtain Figure 13 instead of the left panel of Figure 7. The behaviour of the Shiryaev–Roberts statistic abruptly changes shortly before the 200,000th observation. This happens because of a numerical underflow. Up to that point the value S_n of the martingale has been exponentially decreasing, down to a small multiple of $2^{-1074} \approx 5 \times 10^{-324}$, the smallest positive number representable as double-precision floating-point number (cf. Figure 11). After that point the value S_n of the martingale cannot decrease further substantially and keeps fluctuating in that small region.

To get rid of the underflow, we can use the recursion

$$\psi_n = \frac{S_n}{S_{n-1}}(\psi_{n-1} + 1)$$

with occasional rescaling of S_n (in our code we rescale S_n, setting it to 1, every 10,000th observation). This results in Figure 7. Similar precautions need to be taken for the CUSUM statistic as well, in which case the recursion is

$$\gamma_n = \frac{S_n}{S_{n-1}} \max(\gamma_{n-1}, 1).$$