IMS Collections
Markov Processes and Related Topics: A Festschrift for Thomas G. Kurtz
Vol. 4 (2008) 107–122

DOI: 10.1214/074921708000000336

On Detecting Fake Coin Flip Sequences

Michael A. Kouritzin,[1,*] **Fraser Newton,**[2] **Sterling Orsten,**[2]
and Daniel C. Wilson[3]

University of Alberta, Random Knowledge Inc., and Invidi Technologies Corporation

Abstract: Classification of data as true or fabricated has applications in fraud detection and verification of data samples. In this paper, we apply nonlinear filtering to a simplified fraud-detection problem: classifying coin flip sequences as either real or faked. On the way, we propose a method for generating Bernoulli variables with given marginal probabilities and pair-wise covariances. Finally, we present the empirical performance of the classification algorithm.

1. Introduction

The problem of classifying data samples as either true or fabricated is discussed at length in [4]. Applications of such classification algorithms include the verification of crucial empirical data and fraud detection in financial data [4]. In this paper, we develop an approach to a simplified fraud-detection problem: classifying coin flip sequences as either "faked" (i.e., generated by a person) or "real" (i.e., generated by perfect flipping of a true coin).

An algorithm for classifying coin flip sequences was developed by T. Varga (see Csörgő and Révész in [1], p. 97) . Varga had determined that real coin flip sequences of length two hundred almost always had a run of at least six heads or tails in a row; naive fakers rarely generated such long runs, feeling they appeared conspicuous, and instead favoured short runs. He used this algorithm to impress his students by classifying, at a glance, a coin flip sequence as real or fake. This illustrates an important property of the problem: people usually have strong misconceptions of what random behaviour entails.

To explore classification methods further, we challenged a set of undergraduate probability students to implement accurate and robust methods of classifying coin flip sequences. All methods had to be computationally efficient.

The Run Length Method: One can extend Varga's method to handle many different run lengths. For example, we can use the simple error metric $err = \sqrt{\sum_{k=0}^{n}(c_k - EX_k)^2}$, where n is the length of the flip sequence, c_k is the number of runs of length k found in the sequence, and EX_k is the expected number of runs of length k in a flip sequence of length n. We can then classify a sequence

[*] The author gratefully acknowledges support from NSERC through a Discovery Grant.

[1] Department of Mathematical and Statistical Sciences, University of Alberta, Edmonton, Alberta, Canada, T6G 2G1, e-mail: mkouritz@math.ualberta.ca; url: http://www.math.ualberta.ca.

[2] Random Knowledge Inc., 9932 81 Ave, Edmonton, Alberta, Canada, T6E 1W6, e-mail: fraser.newton@randomknowledge.net; sorsten@ualberta.ca; url: http://www.randomknowledge.net.

[3] Invidi Technologies Corporation, 2101, 10060 Jasper Ave NW, Edmonton, Alberta, Canada, T5J 3R8, e-mail: dan@invidi.com; url: http://www.invidi.com.

AMS 2000 subject classifications: Primary 60G35; secondary 62M99

Keywords and phrases: data authentication, nonlinear filtering, fake coin flip detection, simulating correlated variables

as fake if *err* is above a threshold, which is chosen so that 95% of real coin flip sequences are classified correctly. We call this approach the Run Length Method.

The Binomial Method: Another algorithm, which we call the Binomial Method, considers each flip to be a $(p = \frac{1}{2})$-Bernoulli random variable, and so the number of heads in a real coin flip sub-sequence of k flips is a $(k, p = \frac{1}{2})$-Binomial random variable. This algorithm simultaneously examines sub-sequences with $k = 3, 4, 5, 6$ of the 200 flips, and computes the error of a given flip sequence as the difference between the empirical average and expected number of heads in each sub-sequence. Based on this error metric, the sequence is then classified as real or fake.

The Frequency Method: This class challenge also led to the Frequency Method, which estimates $P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, \ldots, \zeta_{i-k} = m_{i-k})$ for all combinations of $m_j \in \{-1, 1\}$, where

$$\zeta_i = \begin{cases} 1 & : \text{ Flip } i \text{ is a head} \\ -1 & : \text{ Flip } i \text{ is a tail.} \end{cases}$$

This estimate is computed empirically using flip frequencies in sub-sequences of length k. A real coin flip sequence should give $P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, \ldots, \zeta_{i-k} = m_{i-k}) = 1/2$ due to its independence; fakers, however, tend to exhibit forms of dependence, skewing this quantity from 1/2.

The Static Probability and Covariance Method: In the Static Probability and Covariance Method, we assume that the marginal probability of a head at any given flip position and the pair-wise covariances between flips are constant quantities. For a given flip sequence, we estimate these quantities using flip frequencies, and classify the flip sequence as fake if the distance between these estimates and the expected quantities on a real coin flip sequence is greater than a predetermined threshold.

The Filtering Method: The aforementioned algorithms use the average properties of the entire flip sequence; as a result, a faker that deviates from the expected behaviour in one time period can compensate by later deviating from the expected behaviour in an opposite way, such that the deviant behaviour averages out. Indeed, this type of inhomogeneous behaviour was observed repeatedly in our faker data. Furthermore, as noted above, these algorithms are attempting to capture the properties of the real flip sequence, and assume that a faker will simply deviate enough from the expected properties to be detected. It would be more powerful to model the faker's behaviour also.

In our approach, we remedy these weaknesses by modeling the properties of both the real coin and the faker. To balance computational efficiency and power, we use marginal probabilities and pair-wise covariances between each flip and the flips that preceded it in time. Furthermore, we model the changes in the faker's strategy as he/she attempts to evade classification algorithms based on sample averages. The models of the real coin and the fakers are called *signals* in filtering terminology, and the coin flips in a sequence are the *observations*. Based on the observations, we provide real-time estimates of the likelihood of these competing signal models using our filtering algorithm.

This paper is organized as follows: in Section 2, we discuss how we obtained faked data and our observations of the properties of the faked data; in Section 3, we discuss algorithms for simulating flip sequences from marginal probabilities and pair-wise covariances; in Section 4, we present the problem in a particle filtering framework; in Section 5, we present empirical results of our filtering solution implementation; in Section 6, we compare our results to the alternative classification methods; in Section 7, we present our conclusions and future work.

2. Fake Coin Experimentation

To obtain an understanding of how fakers evade detection, we acquired a large sample set of faked coin flip sequences by enlisting probability students as well as employees from two companies. They were tasked with producing fake sequences of 200 flips each of which mimicked the behaviour of a real coin as closely as possible. The participants discussed the different properties of a real coin amongst themselves so they became aware of what faked data would be harder to detect; it also led to several of the alternative methods discussed in Section 1.

We imposed the constraint that each sequence must be produced by a single individual in a single sitting for two reasons. First, this allows us to investigate how the strategy of a faker changes over time. Secondly, we believe that the most significant difference between the real coin and fakers is that fakers have a memory of the recent flips, and are somehow influenced by it. If fakers were allowed to combine flip sequences then each faker's memory of recent coin flips would only apply within sub-sequences; if fakers were allowed to generate the sequence over extended time periods, their memory of recent flips would be weakened or eliminated. These two factors could reduce the dependence between flips to insignificance.

Based on the data collected, we observed that two properties of real coin flip sequences proved difficult for fakers to maintain simultaneously: the marginal probability of each flip being a head should be 1/2 and the pair-wise covariances between all current and past coin flips should be 0. For example, a faker can maintain the marginal probability of a head at 1/2 by alternating between heads and tails; however, this will lead to a negative correlation between certain pairs of flips and a positive correlation between other pairs. On the other hand, a faker can maintain all pair-wise covariances at 0 by producing a flip sequence of all heads; however, this will lead to a bias in the marginal probability of a head.

We observed that, although fakers can produce sequences with nearly equal numbers of heads and tails without too much effort, they tend to accomplish this by attempting to undo what they have done: they balance out their recent flips with their current flips. However, this produces positive covariances between near pairs of flips, and negative covariances between far pairs of flips. More sophisticated fakers also use their current flips to try to undo positive and negative correlations observed in their past flip sequences. Under these circumstances, simple sample averages (i.e., measurements that average the properties of the entire flip sequence) would not distinguish fake sequences from real sequences, because the opposing properties in different sub-sequences cancel each other out.

In total, we obtained 345 fake coin flip sequences of 200 flips each. We incorporate the above observations into our solution in Section 4.

3. Fake Coin Simulation

In this section, we introduce an algorithm for simulating coin flip sequences, which is required by our approach to detecting faked coin flip sequences. This algorithm must be capable of adequately simulating real coin flip sequences; capable of adequately simulating faked coin flip sequences; and computationally feasible for use in the filtering algorithm, as discussed in Section 4. It must also be capable of producing a filtering weight. This means that we need to know the transition function of the flip sequence as a function of the corresponding marginal probabilities and lag covariances (see Equation (4.5) in Section 4).

We consider a real coin flip sequence to be a sequence of i.i.d. ($p = \frac{1}{2}$)-Bernoulli random variables. A fake coin flip sequence has either (time-varying) dependence between flips in the sequence, or periodic bias towards either heads or tails (i.e., $p \neq \frac{1}{2}$). The simulation algorithm will generate a sequence of flips of either type.

The most obvious and complete solution to this problem is to model coin flip sequences using the pmf of all possible flip sequences of length $n+1$. However, this method brings severe space demands in implementation for large n. Furthermore, each of the 2^{n+1} entries in the pmf are parameters that must be estimated by the particle filtering algorithm; as the number of parameters grows, the number of signals that must be created in the filtering algorithm must also grow so that the parameter space is adequately represented. For these reasons, we pursue an alternative simulation algorithm, described next.

3.1. Marginal Probabilities and Covariances

We now explore simulating the coin flip sequence using only covariances between pairs of flips and the marginal probabilities $P(\zeta_i = 1)$.

By using only pair-wise covariances and marginal probabilities, we have reduced the number of parameters from 2^{n+1} to only $n+1$ marginal probabilities and $\binom{n+1}{2}$ covariances. We obtain further gains by assuming that, since we are generating a sequence over time, covariances of equal lag are related (e.g., $\text{cov}(\zeta_j, \zeta_{j-l}) = \text{cov}(\zeta_i, \zeta_{i-l})$ for $j = i-1, \dots, i-n+1, l = 1, \dots, n$), which reduces the number of covariance parameters from $\binom{n+1}{2}$ to n.

Furthermore, we can model fakers in a more intuitive way than with a pmf; for example, we can now easily say that some fakers generate flips that are negatively correlated with the most recent flips, explaining the frequent $H \to T$ and $T \to H$ transitions. We can also represent pair-wise dependence with pair-wise covariances (i.e., if $\text{cov}(\zeta_i, \zeta_j) = 0$, then ζ_i, ζ_j are independent for Bernoulli random variables). Finally, this approach is supported by the empirical data gathered in Section 2.

Modeling only these attributes does have weaknesses. For example, pair-wise independence does not imply full independence, and so a faker who behaves independently of any specific flip in the recent history, but is instead dependent on some collection of recent flips, may not be adequately represented in this model. However, this is compensated for by the gains in implementation feasibility.

3.1.1. Methods of Simulating Correlated Binary Variables.

Herein, we discuss alternative methods of simulating binary variables using pair-wise covariances and marginal probabilities.

A good overview of methods of generating correlated binary variables is given in [6], which reviews those proposed in [2], [5], and [3]. As discussed in [6], these methods all require computationally-intense processes: [2] and [5] require solving nonlinear equations; [3] involves an iterative fitting procedure. [6] also proposes a method that does not require complicated numerical procedures, but the method cannot generate binary variables from negative correlations.

The above methods are not suited to our approach to the problem due to their limitations and computational requirements. As a result, we develop our own method in Section 3.1.2.

3.1.2. Our Method of Simulating Correlated Binary Variables

The following proposition will enable us to simulate a coin flip sequence of length $N \in \mathbb{N}$ with given marginal probabilities and pair-wise covariances. It will also be instrumental in setting up our observation model to follow in Section 4.2. We assign conditional probabilities $P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, \ldots, \zeta_{i-n} = m_{i-n})$ such that we maintain the desired covariances and marginal probabilities as i increases:

Proposition 3.1. *Suppose that $N, l \in \mathbb{N}$, $\{p_i\}_{i=1}^N$ and $\{\beta_{i,j}^l\}_{i=1,j=1}^{N,l}$ satisfy $p_i \in (0,1)$, and all the numbers on the RHS of (3.1) are between* 0 *and* 1*. Form the conditional probabilities recursively, starting with $i = 1$, as*

$$\begin{aligned} P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-n} = m_{i-n}) \\ = p_i + \frac{\sum_{j=1}^n (m_{i-j} \times \beta_{i,j}^l)}{2^{n+1}\ P(\zeta_{i-1} = m_{i-1}, ..., \zeta_{i-n} = m_{i-n})} \end{aligned} \tag{3.1}$$

and

$$\begin{aligned} P(\zeta_i = -1 \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-n} = m_{i-n}) \\ = 1 - P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-n} = m_{i-n}) \end{aligned}$$

for each $m_{i-1}, ..., m_{i-n} \in \{-1, 1\}$, $i = 1, 2, 3, ..., N$, $n = l \wedge (i-1)$. Then, $\{\zeta_i\}_{i=1}^N$ are p_i-Bernoulli $\{-1, 1\}$ random variables with covariances $\beta_{k,j}^l = \operatorname{cov}(\zeta_k, \zeta_{k-j})$ for all $j = 1, ..., l \wedge (k-1)$ and $k = 1, 2, 3, ..., N$.

Remark: N is 200 in our experiments; l is the number of flips that we track in the recent history; n allows us to start-up when we have seen fewer than l flips (e.g., when dealing with the first flip, $n = 0$); $\{p_i\}_{i=1}^N$ and lag correlations $\{\beta_{i,j}^l\}_{i=1,j=1}^{N,l}$ are quantities that vary in time (e.g., p_i is the probability of flip i being a head).

Remark: If the right hand side of (3.1) is not between 0 and 1 for some $i \in \{1, 2, ..., N\}$ and some $m_{i-1}, ..., m_{i-n} \in \{-1, 1\}$, then we cannot use this algorithm to create $\{\zeta_i\}_{i=1}^N$ with the desired marginal probabilities and covariances. However, we will see in the sequel that this rarely happens when $\{p_i\}_{i=1}^N$ and $\{\beta_{i,j}^l\}_{i=1,j=1}^{N,l}$ are consistent with a Bernoulli sequence. Conversely, if the right hand side of (3.1) is always between 0 and 1, then we can let $\zeta_0, \zeta_{-1}, ..., \zeta_{1-l}$ be independent fair coin flips (so $\beta_{i,j}^l = 0$ for $j \geq i$) and find

$$\begin{aligned} P(\zeta_i = 1, \zeta_{i-1} = m_{i-1}, ..., \zeta_{i+1-l} = m_{i+1-l} \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-l} = m_{i-l}) \\ = p_i + \frac{\sum_{j=1}^l (m_{i-j} \times \beta_{i,j}^l)}{2^{l+1}\ P(\zeta_{i-1} = m_{i-1}, ..., \zeta_{i-l} = m_{i-l})} \end{aligned}$$

for all $i = 1, 2, ..., N$ and similarly for $\zeta_i = -1$. Therefore, we created an inhomogeneous l-step Markov chain, which will be used later as our observations and also guarantees existence of such $\{\zeta_i\}_{i=1}^N$.

Proof. By (3.1), one has that

$$
\begin{aligned}
(3.2)\qquad P(\zeta_i = 1) &= \sum_{m_{i-1},...,m_{i-n} \in \{-1,1\}} P(\zeta_i = 1, \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-n} = m_{i-n}) \\
&= p_i + \sum_{j=1}^{n} \sum_{m_{i-1},...,m_{i-n} \in \{-1,1\}} \frac{(m_{i-j} \times \beta^l_{i,j})}{2^{n+1}} \\
&= p_i + \sum_{j=1}^{n} \frac{2^{n-1}\beta^l_{i,j}}{2^{n+1}} \sum_{m_{i-j} \in \{-1,1\}} m_{i-j} \\
&= p_i,
\end{aligned}
$$

since $\sum_{m_{i-j} \in \{-1,1\}} m_{i-j} = 0$. Now, we take $k \in \{1, 2, ..., n\}$ and find by (3.1) and (3.2) that

$$
\begin{aligned}
(3.3)\quad & P(\zeta_i = 1, \zeta_{i-k} = 1) \\
&= \sum_{m_{i-1},...,m_{i-k+1},m_{i-k-1},...,m_{i-n}} P(\zeta_i = 1, \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-k+1} = m_{i-k+1}, \\
&\qquad\qquad \zeta_{i-k} = 1, \zeta_{i-k-1} = m_{i-k-1}, ..., \zeta_{i-n} = m_{i-n}) \\
&= p_i P(\zeta_{i-k} = 1) + \sum_{\substack{m_{i-1},...,m_{i-k+1},m_{i-k-1},...,m_{i-n} \in \{-1,1\} \\ m_{i-k}=1}} \frac{\sum_{j=1}^{n} (m_{i-j} \times \beta^l_{i,j})}{2^{n+1}} \\
&= p_i p_{i-k} + \sum_{j=1}^{n} \frac{\beta^l_{i,j}}{2^{n+1}} \sum_{\substack{m_{i-1},...,m_{i-k+1},m_{i-k-1},...,m_{i-n} \in \{-1,1\} \\ m_{i-k}=1}} m_{i-j} \\
&= p_i p_{i-k} + \sum_{\substack{j=1 \\ j \neq k}}^{n} \frac{\beta^l_{i,j}}{4} \sum_{m_{i-j} \in \{-1,1\}} m_{i-j} + 2^{n-1} \frac{\beta^l_{i,k}}{2^{n+1}} \\
&= p_i p_{i-k} + \frac{\beta^l_{i,k}}{4},
\end{aligned}
$$

and similarly,

$$(3.4)\qquad P(\zeta_i = -1, \zeta_{i-k} = -1) = (1 - p_i)(1 - p_{i-k}) + \frac{\beta^l_{i,k}}{4},$$

$$(3.5)\qquad P(\zeta_i = -1, \zeta_{i-k} = 1) = (1 - p_i)p_{i-k} - \frac{\beta^l_{i,k}}{4}, \text{ and}$$

$$(3.6)\qquad P(\zeta_i = 1, \zeta_{i-k} = -1) = p_i(1 - p_{i-k}) - \frac{\beta^l_{i,k}}{4}.$$

Therefore, taking expectations,

$$E[\zeta_i \zeta_{i-k}] = (1 - 2p_i)(1 - 2p_{i-k}) + \beta^l_{i,k}$$

and using $E[\zeta_i] = 2p_i - 1$, we find that $\mathrm{cov}(\zeta_i, \zeta_{i-k}) = \beta^l_{i,k}$. □

Thus, Proposition 3.1 gives us the desired pair-wise covariances and marginal probabilities. Moreover, if every $\beta^m_{i,j} = 0$, then Proposition 3.1 produces independent Bernoulli trials.

The following algorithm is used to simulate the coin flip sequence.

Repeat for flips $\zeta_i, i = 1, \ldots, \infty$:

1. given the past $k = i-1$ flips, compute $P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-\min(k,n)} = m_{i-\min(k,n)})$ using Proposition 3.1.
2. set

$$\zeta_i = \begin{cases} 1 & : \text{ if } U \leq P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-\min(k,n)} = m_{i-\min(k,n)}) \\ -1 & : \text{ otherwise} \end{cases}$$

where U is a $[0,1]$ uniform random variable.

3.1.3. Simulation Results

Herein, we present the empirical results of the algorithm's ability to simulate a coin flip sequence with given marginal probabilities and pair-wise covariances. For testing purposes, we hold p_i and $\left\{\beta^l_{i,j}\right\}_{i=1,j=1}^{N,l}$ constant for $i = 1, \ldots, N$ and denote them as p and $\left\{\beta^l_j\right\}_{j=1}^{l}$, respectively. We use the following empirical methodology to measure error.

This process takes two inputs: N_c and N_f. N_c is the number of pair-wise covariances; N_f is the number of coin flips to be generated in this sequence. We repeat the following process for 1000 trials for each chosen input pair N_c and N_f. In order to sample the performance over all possible marginal probabilities and pair-wise covariances, we use randomly-generated marginal probabilities and pair-wise covariances in each trial.

1. Randomly generate a marginal probability p, uniformly distributed in [0,1], and pair-wise covariances $\left\{\beta^{N_c}_j\right\}_{j=1}^{N_c}$, each of which is uniformly distributed in [-1,1]. Reject them and try again if they cause Proposition 3.1 to produce a value out of bounds (further discussed in Section 3.2); otherwise, continue to the next step.
2. Use the algorithm to generate a coin flip sequence with N_f flips.
3. Estimate the marginal probability and pair-wise covariances, denoted $\bar{p}$ and $\left\{\bar{\beta}^{N_c}_j\right\}_{j=1}^{N_c}$ respectively, by estimating $P(\zeta_i = 1, \zeta_j = 1), P(\zeta_i = 1, \zeta_j = -1), P(\zeta_i = -1, \zeta_j = 1), P(\zeta_i = -1, \zeta_j = -1), P(\zeta_i = 1)$ using flip frequencies in the generated sequence and applying Equation (3.8).
4. Calculate the error on this trial as $err = \frac{(p-\bar{p})^2 + \sum_{j=1}^{N_c} (\beta^{N_c}_j - \bar{\beta}^{N_c}_j)^2}{N_c+1}$.

Empirical results are given in the table below. ME is the average of err (given above) after 1000 trials.

N_f	N_c	ME
100	2	0.0362142
200	2	0.0242806
500	2	0.0116569
1000	2	0.00454978
100	5	0.228708
200	5	0.207152
500	5	0.176541
1000	5	0.135316

As can be seen from the above results, ME increases with N_c; it is likely that this is due to the increased complexity of producing multiple pair-wise covariances when the covariances may influence the next flip in opposing ways. Also, ME decreases with N_f, indicating that, as expected, shorter sequence samples are likely to have higher error.

3.2. Algorithm Constraints

Herein, we investigate the limitations of Proposition 3.1 by determining the conditions that cause it to produce a value $x \notin [0,1]$. We begin with a theoretical constraint relating a particular $\beta^l_{i,k}$ to the marginal probabilities, then pursue more difficult multiple $\beta^l_{i,k}$ compatibility questions empirically.

Proposition 3.2. *Suppose $l \geq 1$ and $k < i \wedge (n+1)$. In the case of a single conditioning random variable, a necessary and sufficient condition for $\beta^l_{i,k}$ to be consistent with a proper right hand side of (3.1) is*

$$\beta^l_{i,k} \in [-4\, p_i p_{i-k} \vee -4(1-\ p_i)(1-\ p_{i-k}), 4\, p_{i-k}(1-\ p_i) \wedge 4(1-p_{i-k})p_i] \,. \tag{3.7}$$

Proof. We have by (3.3) that

$$\begin{aligned} & P(\zeta_i = 1 \mid \zeta_{i-k} = 1) \in [0,1] \\ \Leftrightarrow\ & \frac{4\, p_i p_{i-k} + \beta^l_{i,k}}{4\, p_{i-k}} \in [0,1] \\ \Leftrightarrow\ & 0 \leq 4\, p_i p_{i-k} + \beta^l_{i,k} \leq 4\, p_{i-k} \\ \Leftrightarrow\ & -4\, p_i p_{i-k} \leq \beta^l_{i,k} \leq 4\, p_{i-k} - 4p_i p_{i-k}\ . \end{aligned}$$

Similarly,

$$\begin{aligned} & P(\zeta_i = 1 \mid \zeta_{i-k} = -1) \in [0,1] \\ \Leftrightarrow\ & \frac{4\ (1-\ p_{i-k})p_i - \beta^l_{i,k}}{4(1-\ p_{i-k})} \in [0,1] \\ \Leftrightarrow\ & 0 \leq 4\ (1-\ p_{i-k})p_i - \beta^l_{i,k} \leq 4(1-\ p_{i-k}) \\ \Leftrightarrow\ & -4\ (1-\ p_{i-k})p_i \leq -\beta^l_{i,k} \leq 4(1-\ p_i)(1-\ p_{i-k}) \\ \Leftrightarrow\ & -4(1-\ p_i)(1-\ p_{i-k}) \leq \beta^l_{i,k} \leq 4\ (1-\ p_{i-k})p_i. \end{aligned}$$

Neither $P(\zeta_i = -1 \mid \zeta_{i-k} = 1)$ nor $P(\zeta_i = -1 \mid \zeta_{i-k} = -1)$ require additional constraints.

Using

$$\begin{aligned} \operatorname{cov}(\zeta_i, \zeta_j) = 4[&P(\zeta_i = 1, \zeta_j = 1)P(\zeta_i = -1, \zeta_j = -1) \\ &-P(\zeta_i = 1, \zeta_j = -1)P(\zeta_i = -1, \zeta_j = 1)] \end{aligned} \tag{3.8}$$

(with $\zeta_i = \zeta_{i-k}$ on sets with maximal and minimal probability), we see that (3.7) is exactly the constraint required for $\beta^l_{i,k}$ to be a valid covariance. □

The next question to ponder is: Are there $\beta^l_{i,k}$ compatibility issues that limit the applicability of our algorithm (i.e., could covariances from a valid covariance function fail to be implementable by our algorithm)? The following simple example answers this question in the affirmative by producing values outside $[0,1]$ in Proposition 3.1:

m_3	m_2	m_1	$P(\zeta_3 = m_3, \zeta_2 = m_2, \zeta_1 = m_1)$
1	1	1	0.2
1	1	-1	0.25
1	-1	1	0.15
1	-1	-1	0.06
-1	1	1	0.11
-1	1	-1	0.225
-1	-1	1	0.004
-1	-1	-1	0.001

Using the above pmf and Proposition 3.1, we compute

$$\begin{aligned} P(\zeta_3 = 1) + \frac{(-1) \times \beta_{3,1}^2 + (1) \times \beta_{3,2}^2}{2^{2+1} P(\zeta_2 = -1, \zeta_1 = 1)} &= 0.66 + \frac{(-1) \times -0.2724 + (1) \times 0.17504}{8 \times 0.154} \\ &\approx 1.023. \end{aligned}$$

However, as we see below this is a very limited problem.

As described by [7], the bounds on the covariances become tighter when we have more constraints imposed by conditioning on more than one random variable. Therefore, we further pursue our investigation of the constraints on Proposition 3.1 empirically to determine if there are multiple $\beta_{i,k}^l$ compatibility questions. We use the following empirical methodology to determine if and when Proposition 3.1 produces a value outside of $[0, 1]$ when $\zeta_1, \zeta_2, \ldots, \zeta_n$ is a sample from a proper pmf.

1. Generate a random pmf pmf_1 of n binary random variables by generating each of the 2^n entries with a $[0, 1]$ uniform random variable; each entry is then normalized by the sum of the entries so that each entry is in $[0, 1]$ and the sum of the entries is 1.
2. Calculate the marginals and pair-wise covariances from pmf_1.
3. Using these marginals and covariances, use Proposition 3.1 to generate a pmf pmf_2.
4. Calculate the marginals and covariances from pmf_2, and compare them to the marginals and covariances computed from pmf_2; According to Proposition 3.1, the marginals and covariances should be equal. If they are not, then it can only be because (3.1) produced at least one number outside $[0, 1]$.

It should be noted that pmf_1 and pmf_2 need not be equal: Proposition 3.1 generates a pmf giving the desired marginals and covariances, but there are multiple pmf's capable of producing a set of desired marginals and covariances.

The empirical results of this methodology are given in the table below. N_v is the number of binary random variables used in the experiment; N_t is the number of trials of the experiment; N_{WB} is the number of trials in which the procedure of Proposition 3.1 produced a pmf (pmf_2) with all entries within the bounds of $[0, 1]$; $WB = N_{WB}/N_t$; C is the fraction of pmf_2's that had the expected p_i and $\beta_{i,k}^l$ when Proposition 3.1 produced a pmf_2 with all entries within bounds.

N_v	N_t	N_{WB}	WB	C
2	181040	181040	1.000000	1
3	181040	158633	0.876232	1
4	181040	155366	0.858186	1
5	181040	163441	0.902789	1
6	181020	174530	0.964148	1
7	181020	180137	0.995122	1
8	179000	178985	0.999916	1
9	179000	179000	1.000000	1
10	179000	179000	1.000000	1

The central conclusion that can be drawn from this methodology is that Proposition 3.1 has stricter constraints on $\beta^l_{i,k}$ than a pmf entails, since $WB \neq 1$ for some experiments. Also notice that for $N_v = 2$, $WB = 1$; this is because the constraints on $\beta^l_{i,k}$ imposed by Proposition 3.1 in the bivariate case are no stricter than those imposed by the pmf or the marginals p_i, as described by [7]. Furthermore, as $N_v > 2$ increases, $WB \to 1$; however, this is likely due to the empirical methodology of generating pmf's uniformly rather than indicating that the constraints on $\beta^l_{i,k}$ become looser as $N_v > 2$ increases. Finally, note that $C = 1$, which further supports the correctness of Proposition 3.1 in practice.

4. The Filtering Problem

Herein, we present the fake coin detection problem in a filtering algorithm framework. As noted previously, we require adequate and efficient models of the potential signals and the observations. In this problem, the signals are time-inhomogeneous marginal probabilities and covariances as well as an indicator of real coin or type of faker. The observations are the coin flip sequences. We begin by describing the signal and the observation.

4.1. Signal

In the sequel, we think of the marginal probabilities p_i and the covariances $\beta^l_{i,k}$ as being unknown or random and try to estimate them or, rather, determine which model they conform to. In filtering terminology, we call the probability and covariances the *signal*. In the simplest case, the signal does not vary in time but consists of random but static quantities.

4.1.1. Trivial Faker Signal

Here, we take the given parameters p_i and $\left\{\beta^l_{i,j}\right\}_{j=1}^l$ to be nearly time homogeneous. In particular, suppose that $\varepsilon \in (0, \frac{1}{2}]$ is small. Then, we suppose that

$$p_{k+1} = p_k + \varepsilon p_k(1-p_k)\xi^p_k \text{ and } \beta^l_{k+1,j} = \beta^l_{k,j} + \varepsilon(\beta^l_{k,j}+1)(1-\beta^l_{k,j})\xi^j_k. \tag{4.1}$$

Here, $\{\xi^p_k\}_{k=1}^\infty$ and $\left\{\xi^j_k\right\}_{j,k=1}^{l,\infty}$ are all independent $p = \frac{1}{2}$-Bernoulli $\{-1,1\}$ distributed random variables, independent of everything else.

We can let $X_k = \left(p_k, \beta^l_1(k), ..., \beta^l_l(k)\right)^T$ and find that

$$X_{k+1} = X_k + \varepsilon\sigma(X_k)\xi_k,$$

where σ and $\{\xi_k\}_{k=1}^{\infty}$ are defined implicitly through Equation (4.1).

For efficiency and robustness reasons, it is often better to run particles according to (4.1) even when p and $\left\{\beta_j^l\right\}_{j=1}^{l}$ are time homogeneous.

4.1.2. Random Sign Change Signal

In this section, we consider a more sophisticated faker.

The motivating idea is that a faker will try to undo what he/she has already done, as discussed in Section 2, which would be modeled as a sign change for covariances or switch of bias between heads and tails for the probability. We take $\left\{\beta_{0,j}^l\right\}_{j=1}^{l}$ to be small enough and p_0 to be close enough to $\frac{1}{2}$ that the probabilities given by Proposition 3.1 are in $[0,1]$. We let

$$\begin{aligned} p_{k+1} &= p_k + \rho_k^p(1-2p_k) + \varepsilon p_k(1-p_k)\xi_k^p \quad \text{and} \\ \beta_{k+1,j}^l &= \rho_{k,j}\beta_{k,j}^l + \varepsilon(\beta_{k,j}^l+1)(1-\beta_{k,j}^l)\xi_k^j \end{aligned} \tag{4.2}$$

where $\varepsilon > 0$ is small, $\{\rho_{k,j}\}, \{\rho_k^p\}$ are independent and $P(\rho_{k,j}=-1) = P(\rho_k^p = 1) = 1 - P(\rho_{k,j}=1) = 1 - P(\rho_k^p = 0) = \delta$ for some small number $\delta > 0$ and $\{\xi_k^p\}_{k=1}^{\infty}$, $\left\{\xi_k^j\right\}_{j,k=1}^{l,\infty}$ are all independent $\frac{1}{2}$-Bernoulli $\{-1,1\}$ distributed random variables, independent of everything else. This small δ captures the occasional switching of strategies by fakers over longer sequences of coin flips (e.g., switching between alternating heads and tails to long runs of heads and tails), which is why we keep δ small. Equation (4.1) models a faker who is being influenced by the recent history in a manner which may change slightly over time; Equation (4.2) introduces switching between multiple strategies in an attempt to deliberately "average-out" the evidence of any single strategy.

A variant of this model would be to wait a geometrically distributed time before changing a randomly selected subset of m of the covariances, where $1 <= m <= l$. To make things more precise, at this geometrically distributed time we let each covariance either switch sign or not with equal probability and we let the bias switch between heads and tails with some probability.

The faker procedure often produces a collection (of marginal probabilities and covariances) that are inconsistent with a covariance function. Moreover, it infrequently produces a valid collection that is incompatible with Proposition 3.1. Both situations are treated the same in the sequel by assigning an impossible observation sequence.

4.1.3. Real Coin Signal

A true coin's flips are i.i.d. $p = \frac{1}{2}$-Bernoulli, i.e., $p_0 = \frac{1}{2}$ and $\beta_{0,j}^l = 0$ for $j = 1, ..., l$ and

$$p_{k+1} = p_k \text{ and } \beta_{k+1,j}^l = \beta_{k,j}^l. \tag{4.3}$$

So our signal is

$$X_k = \begin{bmatrix} \theta \\ p_k \\ \left\{\beta_{k,j}^l\right\}_{j=1}^{l} \end{bmatrix},$$

where

$$\theta = \begin{cases} 1 & : \text{ trivial faker, with probability } \frac{1}{4}, \text{ for example} \\ 0 & : \text{ random sign change, with probability } \frac{1}{4}, \text{ for example} \\ -1 & : \text{ real coin, with probability } \frac{1}{2}, \text{ for example,} \end{cases}$$

and -1, 0, and +1 are arbitrarily assigned values for each of the above three cases. When $\theta = 1, 0, -1$, we evolve according to Equation (4.1), (4.2), and (4.3) respectively.

4.2. Observation

The observations are the real or fake coin flips described in the first section. They are modeled as a Markov chain according to the equations described above. In particular,

$$\begin{aligned} &P(\zeta_i = 1, \zeta_{i-1} = m_{i-1}, ..., \zeta_{i+1-l} = m_{i+1-l} \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-l} = m_{i-l}) \\ &\quad = p_i + \frac{\sum_{j=1}^{l}(m_{i-j} \times \beta_{i,j}^l)}{2^{l+1} P(\zeta_{i-1} = m_{i-1}, ..., \zeta_{i-l} = m_{i-l})} \end{aligned}$$

and so

$$\begin{aligned} &P(\zeta_i = -1, \zeta_{i-1} = m_{i-1}, ..., \zeta_{i+1-l} = m_{i+1-l} \mid \zeta_{i-1} = m_{i-1}, ..., \zeta_{i-l} = m_{i-l}) \\ &\quad = 1 - p_i - \frac{\sum_{j=1}^{l}(m_{i-j} \times \beta_{i,j}^l)}{2^{l+1} P(\zeta_{i-1} = m_{i-1}, ..., \zeta_{i-l} = m_{i-l})}, \end{aligned}$$

where we substitute the particular signal model of interest in for $\{p_i\}_{i=1}^{\infty}$ and $\{\beta_{i,j}^l\}_{j=1}^{l}$.

We define the observations as the most recent l coin flips: $Y_k = (\zeta_k, \zeta_{k-1}, ..., \zeta_{k-l})^T$, so

$$\begin{aligned} Y_k &= \begin{bmatrix} m_k \\ m_{k-1} \\ \vdots \\ m_{k-l} \end{bmatrix} \\ \to Y_{k+1} &= \begin{cases} \begin{bmatrix} 1 \\ m_k \\ \vdots \\ m_{k+1-l} \end{bmatrix} & \begin{array}{l} \text{with probability} \\ p_{Y_k \to Y_{k+1}}(X_k) = \\ p_k + \frac{\sum_{j=1}^{l}(m_{k-j} \times \beta_{k,j}^l)}{2^{l+1}\ P(\zeta_{k-1} = m_{k-1}, ..., \zeta_{k-l} = m_{k-l})} \end{array} \\ \begin{bmatrix} -1 \\ m_k \\ \vdots \\ m_{k+1-l} \end{bmatrix} & \begin{array}{l} \text{with probability} \\ 1 - p_k - \frac{\sum_{j=1}^{l}(m_{k-j} \times \beta_{k,j}^l)}{2^{l+1}\ P(\zeta_{k-1} = m_{k-1}, ..., \zeta_{k-l} = m_{k-l})} \end{array} \end{cases} . \end{aligned}$$

Note that, as discussed in Section 3.2, $p_k + \frac{\sum_{j=1}^{l}(m_{k-j} \times \beta_{k,j}^l)}{2^{l+1}\ P(\zeta_{k-1} = m_{k-1}, ..., \zeta_{k-l} = m_{k-l})}$ is capable of producing a value outside of $[0, 1]$ (i.e., the signal marginal probabilities and pair-wise covariances are incompatible with Proposition 3.1). In this case, we transition to a cemetery state 0 that receives no weight and cannot be revived:

$$Y_k = \begin{bmatrix} m_k \\ m_{k-1} \\ \vdots \\ m_{k-l} \end{bmatrix} \to Y_{k+1} = \begin{bmatrix} 0 \\ m_k \\ \vdots \\ m_{k+1-l} \end{bmatrix}$$

$$\to Y_{k+2} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ m_{k+2-l} \end{bmatrix} \to \cdots \to Y_\infty = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The above two paragraphs give $p_{Y_{k-1}\to Y_k}(X_k)$ as required in the sequel.

In the sequel, we will also be interested in the information, $\mathcal{F}_k^Y \overset{\circ}{=} \sigma\{Y_1, ..., Y_k\}$, generated by the observations up to time k, as well as $\mathcal{F}_k^{XY} \overset{\circ}{=} \sigma\{X_1, ..., X_k; Y_1, ..., Y_k\}$, given by the signal-observation pair up to time k. As in classical filtering, the observation model is not perfect, prompting questions like: "Is there a robust nonlinear filter for this model?"

4.3. Particle Filter Approach

Developments analogous to classical nonlinear filtering theory, to be published elsewhere, establish that we only need to approximate

$$\mu_j(dx) \doteq \frac{\bar{E}[1_{X_j \in dx}\eta_j \mid \mathcal{F}_j^Y]}{\bar{E}[\eta_j \mid \mathcal{F}_j^Y]}. \tag{4.4}$$

Here $\bar{E}$ is the expectation with respect to a reference probability measure $\bar{P}$, where X is independent of the observations and the observations are a vector obtained by storing the n most recent real coin flips, and

$$\eta_j = \prod_{k=0}^{j-1} 2p_{Y_k\to Y_{k+1}}(X_k)$$

is the weighting function. Now, suppose that we introduce independent signal particles $\{X_k^i, k = 1, 2, ...\}_{i=1}^{\infty}$, each with the same law as the signal, and define the weights

$$\eta_j^i = \prod_{k=0}^{j-1} 2p_{Y_k\to Y_{k+1}}(X_k^i). \tag{4.5}$$

Then, it follows by deFinnetti's theorem and the law of large numbers that

$$\frac{1}{N}\sum_{i=1}^{N} \eta_j^i \delta_{X_j^i}(dx) \Rightarrow \mu_j(dx).$$

Moreover, we can apply re-sampling to improve performance. Finally, we convert back to the original measure

$$P(X_j \in dx \mid \mathcal{F}_j^Y) = \frac{\mu_j(dx)}{\mu_j(1)}$$

so

$$\begin{bmatrix} \widehat{p}_j \\ \widehat{\beta}_{j,1} \\ \vdots \\ \widehat{\beta}_{j,l} \end{bmatrix} = E(X_j \mid \mathcal{F}_j^Y) = \frac{\int x\mu_j(dx)}{\mu_j(1)} \approx \frac{\frac{1}{N}\sum_{i=1}^N \eta_j^i X_j^i}{\frac{1}{N}\sum_{i=1}^N \eta_j^i}.$$

5. Filtering Results

Herein, we present the filter's classification performance on coin flip sequences. Performance P is calculated as the number of correct classifications divided by the number of trials N_t; we use an equal number of real and fake coin flip sequences. We also present the performance in relation to the number of particles N_p that were initialized and the number of variables N_v that were tracked; the number of seconds S required on a computer is also given. Recall that if we have N_v variables, we have $N_v - 1$ pair-wise covariances. For each N_t, N_v, N_p combination presented in the table below, the results using the best performing parameters are given.

N_t	N_v	N_p	S	P
690	2	200	189.691	0.87971
690	2	5000	1100.15	0.87971
690	2	1000	340.456	0.885507
690	6	200	222.646	0.897101
690	6	25000	8299.74	0.897101
690	4	200	206.154	0.898551
690	6	1000	466.005	0.901449
690	6	5000	1729.94	0.901449
690	4	1000	402.127	0.905797
690	4	125000	32399.4	0.905797
690	4	5000	1402.79	0.907246
690	4	25000	6585.35	0.910145

A number of conclusions can be drawn from these results. First, we examine the effect of increasing N_p. Notice that as N_p increases, holding N_v, N_t constant, S increases as expected, since there are more particles that must be updated for each observation and evolved. Furthermore, the experiments show that the performance is relatively independent of the number of particles, provided that at least 200 are used.

Secondly, we examine the effect of increasing N_v. As N_v increases, holding N_p and N_t constant, S increases as expected since calculating Proposition 3.1 has a time complexity in $O(n^2)$, and the procedure of Proposition 3.1 is used to update the particle weights with each observation. However, as N_v increases, holding N_p and N_t constant, P does not consistently increase; in fact, the best observed performance is achieved with $N_v = 4$. This may be because most fakers are strongly influenced by their very recent past, but are weakly influenced by their distant past. By attempting to represent the distant past with a high N_v, we are increasing the complexity of the signal and the difficulty of the estimation task, but there may be very little potential performance gain.

6. Comparative Results

In this section, we present the classification performance of alternative methods, including methods discussed in Section 1.

The first method, introduced in Section 1 and described in more detail in Section 3.1.3, assumes p_i and $\left\{\beta^l_{i,j}\right\}_{i=1,j=1}^{\infty,l}$ are static quantities and statically estimates the marginal probability and pair-wise covariances (denoted $\bar{p}$ and $\bar{\beta}^l_j$, respectively) using flip frequencies in the coin flip sequence. The distance between the estimates of $\bar{p}$ and $\bar{\beta}^l_j$ and the real coin ($p = \frac{1}{2}$ and $\beta^l_j = 0$ for $j = 1, \ldots, l \wedge (k-1)$ and $k = 1, 2, 3, \ldots$) is used as an error metric; a threshold on the error metric is then empirically determined such that 95% of real coin flip sequences fall within the threshold. For a given sequence, if the error metric falls within the threshold, then the sequence is real; otherwise it is fake. This yields a performance of $P = 0.894203$ on the same dataset used in Section 5.

The second method, developed and implemented by applied probability students Kevin Swersky and Mark Zschocke, extends Varga's method by empirically determining the expected number of runs of lengths $4, 5, 6, 7$ in a real sequence of 200 coin flips. A threshold on the error metric is then empirically determined such that 95% of real coin flip sequences fall within the threshold. The classification algorithm for each sequence is then as follows: if the error metric is within the threshold, then the sequence is real; otherwise, it is fake. This yields a performance of $P = 0.885507$ on the same dataset used in Section 5.

The third method we applied increases its performance by employing a weighted combination of alternative methods. It examines the following properties of the sequence: the number of heads in sub-sequences of length $k = 3, 4, 5, 6$, which would follow the $(k, p = \frac{1}{2})$-Binomial distribution in a real coin flip sequence; a frequency-based estimate of $P(\zeta_i = 1 \mid \zeta_{i-1} = m_{i-1}, \ldots, \zeta_{i-k} = m_{i-k})$, which would show independence in a real coin flip sequence; a frequency-based estimate of the pmf for sub-sequences of varying lengths k, which would have approximately equal entries in a real coin flip sequence; a count of the number of runs of length 1, which tend to be over-represented in faked flip sequences; and a count of the number of switches between heads and tails, which also tend to be over-represented in faked flip sequences. This yields a performance of $P = 0.904348$ on the same dataset used in Section 5.

The fourth method we applied, which we use as a performance baseline for other classifiers, is an implementation of Varga's original method. This method classifies a flip sequence as real if and only if it contains at least one run of at least six heads or tails in a row; otherwise the flip sequence is classified as fake. This yields a performance of $P = 0.672464$ on the same dataset used in Section 5.

7. Conclusions and Future work

This paper's work focused on two areas: the development of a method of simulating correlated binary variables; and the application of nonlinear filtering to the problem of detecting fabricated data.

Proposition 3.1 gives conditional probabilities for binary variables with given marginal probabilities and pair-wise covariances with a reasonable time-complexity ($O(n^2)$). Furthermore, our analysis in Section 3.2 showed Proposition 3.1 to have no more constraints than those imposed by the pmf in the bivariate case. However, we also found that the algorithm has (slightly) stricter constraints than those imposed by the pmf in the k-variate case when $k > 2$.

We will further pursue development of this algorithm in two ways: first, we will further develop analytic forms for bounds on Proposition 3.1 by extending Proposition 3.2 to the k-variate case; secondly, we will extend Proposition 3.1 to include higher-order terms, as described in [7], so that the bounds on pair-wise covariances can be made looser when applying Proposition 3.1 to the k-variate case.

As the empirical results show, our initial application of nonlinear filtering resulted in an algorithm that beat alternative (in one case very elaborate) methods described in Section 6, even when the alternative method examines far more properties of the coin flip sequence. Furthermore, our algorithm provides a real-time estimate of the likelihood of a faker at any point in the coin flip sequence; the alternative methods are empirically tuned to the properties of a sequence of 200 flips, and thus only provide a classification of the coin flip sequence once all 200 flips have been observed. However, our performance appears to plateau, or at least suffer from diminishing returns, as the number of covariances and particles are increased. Overall, the results are promising and encourage future development, especially on the signal model.

The classification algorithm can be further developed by extending the faker models and by attempting new methods of modeling the signal. Faker models can be extended by improving the existing faker strategy models; for example, we could extend the evolution of the covariances with a finite state machine. New signal models will largely be motivated by the weakness of the current signal model: an inability to capture the full dependence of each flip on the recent history. One alternative is to use the full pmf, which would contain the full probability information about a coin flip sequence; for a recent history of length $n + 1$, we will have 2^{n+1} parameters. Another alternative is to directly use the conditional probability of the current flip given the recent past, which gives all dependence between the present and the past and the marginal probability of the current flip; for a history of length $n + 1$, there will be 2^n parameters. These signal model alternatives are much less computationally efficient, but they may offer significant performance gains.

References

[1] Csörgő, M. and Révész, P. (1981). *Strong Approximations in Probability and Statistics.* Academic Press.

[2] Emrich, L. J. and Piedmonte, M. R. (1991). A method for generating high-dimensional multivariate binary variates *The American Statistician* **45** (4) 302–304.

[3] Gange, S. J. (1995). Generating multivariate categorical variates using the iterative proportional fitting algorithm. *The American Statistician* **49** (2).

[4] Hill, T. P. (1999). The difficulty of faking data. *Chance Magazine* **12** (3) 27–31.

[5] Lee, A. J. (1993). Generating random binary deviates having fixed marginal distributions and specified degrees of association. *The American Statistician* **47** (3).

[6] Park, C. G., Park, T. and Shin, D. W. (1996). A simple method for generating correlated binary variates. *The American Statistician* **50** (4).

[7] Prentice, R. L. (1998). Correlated binary regression with covariates specific to each binary observation. *Biometrics* **44** (4) 1033–1048.