Predictive Models for Min-Entropy Estimation

John Kelsey[†], Kerry A. McKay[†], and Meltem Sönmez Turan [†] [‡]

[†] National Institute of Standards and Technology, Gaithersburg, MD
[‡] Dakota Consulting Inc., Silver Spring, MD

Abstract. Random numbers are essential for generating cryptographic information, such as secret keys, nonces, random paddings, salts etc. Pseudorandom number generators are useful, but they still need truly random seeds generated by entropy sources in order to produce random numbers. Researchers have shown examples of deployed systems that did not have enough randomness in their entropy sources, and as a result, crypto keys were compromised. So how does one evaluate how much entropy is in a black-box entropy source? One approach to solving this problem is to use a series of statistical estimators, such as those presented in [8]. However, there are many assumptions that the statistical methods make that may not be true in the entropy source. In particular, the entropy source outputs may be dependent, and the distribution of random variables may change over time. To address these limitations, we propose alternative methods for estimating the entropy in each output from an entropy source based on concepts from machine learning and time series analysis. These methods, called predictors, estimate entropy by attempting to predict the next output in the sequence. We also present experimental results comparing the entropy estimates of predictors with those defined in the August 2012 draft of NIST SP 800-90B[3].

Keywords: Entropy estimation, Min-entropy, Random number generation

1 Introduction

Random numbers are essential for generating cryptographic information, such as secret keys, nonces, random paddings, salts etc. Pseudo random number generators (PRNGs) are capable of generating cryptographically strong outputs; however they still need truly random seeds generated by entropy sources in order to produce random numbers. Entropy sources include noise sources that extract randomness from physical elements; for example, many practical entropy sources are based on the timing variation from an unstable oscillator. To be useful, the entropy source's unpredictability must also be quantified – we need to know how many bits need to be drawn from the entropy source to produce a good seed. The unpredictability of the outputs of an entropy source is measured in terms of *entropy*. There are a number of different measures of entropy, such as Shannon entropy, Rényi entropy, and min-entropy. For cryptographic purposes such as seeding pseudorandom number generators (PRNGs), the relevant measure is min-entropy, which corresponds to the difficulty of guessing the most-likely output of the entropy source.

Researchers have shown examples of deployed systems that did not have enough entropy in their randomness sources, and as a result, cryptographic keys were compromised (e.g. see [9, 4, 6]). Estimating the amount of entropy that entropy sources provides is necessary to ensure that randomly generated numbers are sufficiently difficult to guess. However, entropy estimation is a very challenging problem when the distribution of the outputs is unknown and common assumptions (e.g. outputs are independent and identically distributed (i.i.d.)) cannot be made.

There are various approaches to entropy estimation. Plug-in estimators, also called maximum-likelihood estimators, apply the entropy function on empirical distributions (e.g., see [1, 14]). Methods based on compression algorithms (e.g., see [10, 19]) use match length or frequency counting to approximate entropy. Hagerty and Draper provided a set of entropic statistics and bounds on the entropy in [8]. Lauradoux et al. [11] provided an entropy estimator for non-binary sources with an unknown probability distribution that converges towards Shannon entropy.

Draft SP 800-90B [3] discusses procedures for evaluating how much entropy per sample can be obtained from an entropy source. It also provides a suite of five entropy estimators for sequences that do not satisfy the i.i.d. assumption. Each estimator takes a sequence of minimally processed samples from the underlying unpredictable process in the entropy source, and uses them to derive an entropy estimate. The minimum of the estimates obtained from the five estimators is used as the entropy assessment of a given source. This estimation is used for entropy sources that undergo validation testing in order to comply with FIPS 140 [13].

In this study, we present an alternative approach to estimating min-entropy using *predictors*. A predictor contains a model and a prediction function, and aims to predict the next sample value, based on previous observations. Entropy is a measure of unpredictability, so we can use the performance of a predictor to estimate the entropy of a source. The more accurate the predictor's predictions, the lower the entropy estimate. Estimates are calculated using global and local performance of the predictors, and the final entropy estimate is selected as the minimum of these two values. Shannon first published the relationship between the entropy and predictability of a sequence in 1951 [17], using the ability of humans to predict the next character in the text to estimate the entropy per character.

Predictors construct models from previous observations, which are then used to predict the next value in a sequence. A predictor that is a good fit for the behavior of a source will give a relatively accurate estimate; a predictor that is a poor fit will give an overestimate. This interacts well with the procedure of taking the minimum of many entropy estimates to get a final estimate. Applying a suite of many predictors, each tuned for a different kind of source, will not substantially degrade the quality of the final entropy estimates.

In this research, we introduce various categorical and numerical predictors, each of which makes predictions based on a different model, which builds models on-the-fly by observing the sequence of samples. These predictors use a variety of modeling and prediction strategies. Markov models and dictionaries are used to leverage dependencies between adjacent outputs. Another predictor attempts to discover periodicity by predicting the value that appeared at some fixed distance (lag) in the past. Changes to the source distribution over time (called *concept drift*) are also a problem. One method makes predictions based on the most common value in a fixed-size window of most recent outputs. Finally, numerical smoothing techniques are applied to find trends. These include computing a moving average over a given window size, and make predictions based on a first-order difference equation. Each predictor focuses on a relatively simple way for a source to exhibit some predictability. We also describe how predictors can be thought of as classifiers in machine learning.

Evaluating the quality of entropy estimates is challenging, because actual entropy per sample values for real-world entropy sources are not known. In order to evaluate our estimators, we employed three tools:

1. We generated datasets from *simulated sources* with a variety of distributions. All these simulated sources were constructed so that we would know their correct entropy per sample (because we would always know the probability distribution used to generate each output value).
2. We applied the tests to datasets from real-world hardware and software entropy sources. For these sources, we did not know the correct entropy per sample, but we were able to verify the plausibility of our current estimates.
3. We compared the results of our estimators on both these sets of datasets to results from the non-i.i.d. tests that appear in [3], and are described in detail in [8] and [7].

The remainder of the paper is organized as follows. Section 2 provides fundamental definitions about entropy. Section 3 describes the entropy estimation using predictors, and gives descriptions of several simple predictors. Section 4 provides experimental results, and the last section concludes the study with some future directions.

2 Preliminaries

Entropy sources, as defined in [3], are composed of three components; *noise sources* that extract randomness from physical phenomena, *health tests* that aim to ensure that entropy sources operate as expected, and (optional) *conditioning functions* that improve the statistical quality of the noise source outputs. The conditioning function is deterministic, hence does not increase the entropy of the outputs of the noise source. In this study, we assume that the samples obtained from a noise source consist of fixed-length bitstrings.

Let X be a discrete random variable taking values from the finite set $A = \{x_1, x_2, \ldots, x_k\}$, with $p_i = Pr(X = x_i), \forall x \in A$. The *min-entropy* of X is defined as $-\log_2(\max\{p_1, \ldots, p_k\})$.

If X has min-entropy h, then the probability of observing any particular value is no greater than 2^{-h}. The maximum possible value for min-entropy of an i.i.d random variable with k distinct values is $\log_2(k)$, which is attained when the random variable has a uniform probability distribution, i.e., $p_1 = p_2 = \ldots = p_k = \frac{1}{k}$.

A stochastic process $\{X_n\}_{n\in\mathbb{N}}$ that takes values from a finite set A is called a *first-order Markov chain*, if

$$Pr(X_{n+1} = i_{n+1}|X_n = i_n, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = Pr(X_{n+1} = i_{n+1}|X_n = i_n),$$

for all $n \in \mathbb{N}$ and all $i_0, i_1, \ldots, i_n, i_{n+1} \in A$. The initial probabilities of the chain are $p_i = \Pr(X_0 = i)$, whereas the transition probabilities p_{ij} are $\Pr(X_{n+1} = j|X_n = i)$. In a d-th order Markov Model, the transition probabilities satisfy

$$Pr(X_{n+1} = i_{n+1}|X_n = i_n, \ldots, X_1 = i_1) = Pr(X_{n+1} = i_{n+1}|X_n = i_n, \ldots, X_1 = i_{n-d}).$$

The min-entropy of a Markov chain with length L is defined as

$$H = -\log_2(\max_{i_1,\ldots,i_L} p_{i_1} \prod_{j=1}^{L} p_{i_j i_{j+1}}).$$

3 Entropy Estimation Using Predictors

A *predictor* contains a model that is updated as samples are processed sequentially. For each sample, the model offers a prediction, obtains the sample, and then updates its internal state, based on the observed sample value in order to improve its future predictions. The general strategy of the predictors is summarized in Fig. 1. It is important to note that predictors do not follow "traditional" supervised learning methods of training and evaluation. In particular, traditional methodologies contain disjoint *training* and *testing* sets. The training set is used to construct the model, possibly performing many passes over the data, and the testing data is used to evaluate the model but not update it. Predictors, on the other hand, use all observations to update the model. In other words, all observations are part of the training set. This allows the predictor to continually update its model, and remain in the training phase indefinitely. All observations are also part of the testing set, because a predictor is evaluated based on all predictions that were made since its initialization.

3.1 Predictor Performance and Entropy Bounds

There are two ways to broadly measure the performance of a predictor. The first one, which we call *global predictability*, considers the number of accurate predictions over a long period, whereas the second measure, called *local predictability*, considers the length of the longest run of correct predictions.

A predictive model that can predict each sample value with probability greater than a random guess, on average, will give the attacker a better than

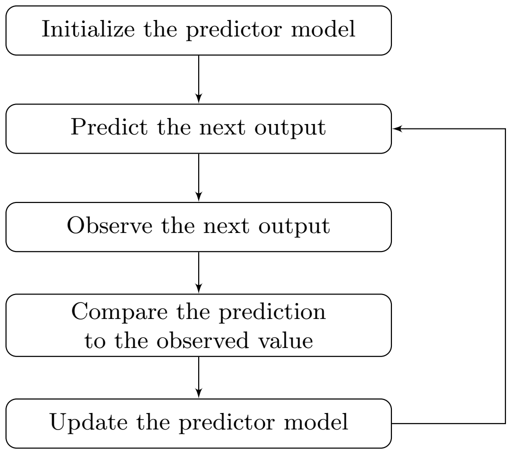
Initialize the predictor model
Predict the next output
Observe the next output
Compare the prediction to the observed value
Update the predictor model

Fig. 1: General strategy of predictors

expected chance to guess a PRNG seed each time one is generated. A predictive model that usually gives a much lower probability of success, but which occasionally gives a lengthy run of correct predictions in a row, gives the attacker a chance at guessing a specific PRNG seed very cheaply.

It is important to understand that a predictor's performance provides an *upper-bound* on the min-entropy of the sequence. If a predictor successfully predicts $\frac{7}{8}$ of the values in a sequence, this provides solid evidence that this sequence can, in fact, be correctly predicted $\frac{7}{8}$ of the time. Other predictors might provide an even higher probability of success, but at the very least, we now have a lower-bound on the success probability that can be attained in predicting the values in this sequence. That *lower-bound* on the maximum probability gives us an *upper-bound* on the min-entropy of the sequence.

Global Predictability A predictor's global accuracy p_{acc} is the probability that the predictor will correctly predict a given sample from a noise source over a long sequence of samples. Accuracy is simply the percentage of predictions that were correct. Let c denote the count of correct predictions, and n be the number of predictions made. For a given predictor, a straightforward estimate of its accuracy is

$$\hat{p}_{acc} = \frac{c}{n}. \tag{1}$$

We can also compute a confidence interval on $\hat{p}_{acc}$, making the assumption that the success probability of each prediction is independent of the correctness of other predictions. The upper bound of the $(1-\alpha)\%$ confidence interval on $\hat{p}_{acc}$, denoted as $\tilde{p}_{acc}$ is calculated as;.

$$\tilde{p}_{acc} = \begin{cases} 1 - \left(\frac{\alpha}{2}\right)^{\frac{1}{n}}, & \text{if } \hat{p}_{acc} = 0, \\ 1, & \text{if } \hat{p}_{acc} = 1, \\ \hat{p}_{acc} + t_\alpha \sqrt{\frac{\hat{p}_{acc}(1-\hat{p}_{acc})}{n-1}}, & \text{otherwise,} \end{cases} \tag{2}$$

where t_α refers to the upper $\alpha/2$ tail of Student's t-distribution with n-1 degrees of freedom[1]. Note that if $\hat{p}_{acc}$ is 1 or 0, computing a confidence interval using the Student's t-distribution is not valid. In these cases, confidence intervals are calculated using the binomial distribution.

The global min-entropy estimate for this predictor, $\hat{H}_{acc}$, is derived from $\tilde{p}_{acc}$ using

$$\hat{H}_{global} = -\log_2(\tilde{p}_{acc}). \tag{3}$$

Local Predictability A second method to measure the performance of a predictor uses the length of the longest run of correct predictions. This estimate is valuable mainly when the source falls into a state of very predictable outputs for a short time. Should this happen, the estimated min-entropy per sample will be lower.

This entropy estimate can be obtained using recurrent events. Let r be one greater than the longest run of correct predictions (e.g., if the longest run has length 3, then $r = 4$). Then the probability that there is no run of length r, is calculated as

$$\alpha = \frac{1 - px}{(r + 1 - rx)q} \cdot \frac{1}{x^{n+1}}, \tag{4}$$

where $q = 1 - p$, n is the number of predictions and x is the real positive root of the polynomial $1 - x + qp^r x^{r+1} = 0$ [5]. The root x can be efficiently approximated using the recurrence relation $x_{i+1} = 1 + qp^r x_i^{r+1}$, as it converges on the root. In practice, ten iterations appears sufficient. Note that there may be two real positive roots, one of which is $\frac{1}{p}$. This root is generally considered extraneous, and the recurrence relation will converge on it if and only if it is the only positive root. To find the min-entropy estimate, denoted as $\hat{H}_{local}$, using local predictability, first perform a binary search to solve for p, then the apply following equation.

$$\hat{H}_{local} = -\log_2 p \tag{5}$$

Deriving a Final Estimate The final entropy estimate for a predictor is the minimum of the global and the local entropy estimates, i.e.,

$$\hat{H} = \min(\hat{H}_{global}, \hat{H}_{local}) \tag{6}$$

[1] t_α can be approximated by the $1 - \frac{1}{2}\alpha$ percentile of a standard normal distribution, when $n \geq 30$.

Entropy Estimation Using Multiple Predictors In order to estimate the entropy of a given entropy source, we first select a set of predictors where each predictor is designed to catch a particular type of behavior. Then, we generate a long output sequence and evaluate the accuracy of each predictor, which provides an entropy estimate. After obtaining the estimates from the predictors, the final entropy estimate of the source is taken as the minimum of all the estimates. If there is an undesirable behavior in an entropy source, but no predictor is applied that can detect that behavior, then the entropy estimate will be overly generous. Because of this, it is important that a set of predictors that use different approaches be applied, and the lowest entropy estimate is taken as the final entropy estimate.

Many entropy estimators (such as the ones in [8]) work by using the observed data to fit the parameters to some probability model, and then using the resulting probability model to produce an entropy estimate. There is one obvious way this can fail: if the source's actual behavior is a very bad fit for the probability model, those estimated parameters are meaningless. This can lead to large underestimates or overestimates of the actual entropy coming from the source. On some simulated data, such as normally-distributed data with relatively high entropy per sample, the tests from [3] give extreme underestimates for this reason (See Fig. 3).

Predictors work in an entirely different way. A predictor sitll fits parameters to a model, but then uses that model to make *predictions*. The entropy estimate is thus directly linked to the practical security question: *how difficult will it be for an attacker to predict these outputs?* The existence of a predictor that can consistently predict half the outputs from the source is proof that the source cannot provide more than one bit of entropy per sample. A predictor must produce a prediction for every sample in the sequence[2]. The entropy assessment obtained from a predictor is entirely determined by the predictor's performance on a sequence of noise source outputs. Further, we can make assumptions about the distribution of the prediction results. In particular, the distribution is binomial (since predictions are either correct or incorrect), and the variables are i.i.d.

A predictor that employs an apparently inappropriate model, but that generates accurate predictions, is not really inappropriate – it's just another case where a messy real-world system is usefully modeled by a simple mathematical construct, and it is a tool that an attacker could use to predict source outputs in practice.

Predictors can give underestimates only by getting unusually lucky in their predictions–a kind of error that both decreases as we examine more samples, and that can be analyzed and bounded using probability theory.

This means that applying a large set of very different predictors to a sequence of noise source outputs, and then taking the minimum estimate, is a workable strategy. Inappropriate models will not substantially lower the final entropy esti-

[2] Some predictors produce a null prediction (represented as None in our Python code) very early in the sequence; this is just a prediction that is guaranteed to be incorrect.

mate. The entropy estimate will be lowered somewhat by many equally accurate predictors (for example, if there are ten predictors, then there are ten opportunities for one of the predictors to get lucky and give an unusually low estimate). Each predictor's performance on a given sequence from a noise source can be seen as a random variate whose mean is the correct entropy per sample for this noise source), but the probability of this happening can be bounded.

3.2 Categorical and Numerical Predictors

In this section, we present a set of predictors for categorical and numerical data that are designed to characterize the behaviors of the noise sources. Entropy sources, defined in SP 800-90B[3], only contain discrete values. Therefore, we consider predictors as a solution to a classification problem, rather than a regression problem. However, this does not preclude one from constructing a predictor from numerical data.

Some of these predictors are constructed using windowing or ensemble methods to accommodate concept drift. Ensemble methods use multiple classifiers that run simultaneously and obtain the final prediction by a weighted vote.

Predictors can be constructed using existing methods from online and stream classification, but do not need to be complex. Classifiers are often designed to be domain specific. For noise sources where few assumptions about the underlying probability distribution can be made, it may be difficult to construct sophisticated learners.

Categorical Data Predictors This part describes several predictors that assume that the samples represent categorical data, i.e., all samples have no numerical meaning or ordering, and serve as labels only.

Most Common in Window (MCW) is a predictor that maintains a sliding window of the most recently observed w samples, where w is a parameter which can be varied in different instances of the predictor. Its prediction is the most common value that has occurred in that window. If there are multiple values that have occurred the highest number of times in the window, the value that has occurred most recently is used as the prediction. This predictor is expected to perform well in cases where there is a clear most-common value, but that value varies over time. For example, a source whose most common value slowly changes due to environmental changes, such as operating temperature, might be approximated well by this predictor. In our experiments, this predictor is used inside the ensemble predictor MultiMCW.

Single Lag Predictor is a predictor that remembers the most recent N values seen, and predicts the one that appeared N samples back in the sequence. N is a parameter to the predictor. The single lag predictor is expected to perform well when applied to data that has strong periodic behavior. In our experiments, this predictor is used inside the Lag ensemble predictor.

Markov Model with Counting (MMC) is a predictor that remembers every N-sample that has been seen so far, and keeps counts for each value that followed each N-sample sequence. N is a parameter for this predictor. The MMC predictor is expected to perform well on data from a Markov source with order N, and on any real-world process that can be accurately modeled by an Nth-order Markov model. In our experiments, this predictor is used inside the MultiMMC ensemble predictor.

LZ78Y is a predictor that is based loosely on the LZ78 family of compression algorithms [16]. The predictor keeps track of all observed strings of samples up to a maximum length of 32 until its dictionary reaches maximum size. For each such string, the predictor keeps track of the counts of what values followed the string. (Note that even after the dictionary reaches maximum size, the counts continue to be updated.) The LZ78Y predictor is expected to perform well on data from Markov sources and on the sort of data that would be efficiently compressed by LZ78-like compression algorithms. The LZ78Y predictor is used directly in our experiments.

Ensemble Predictors for Categorical Data
We make use of three ensemble predictors based on categorical data. Each of the three predictors contains many very similar subpredictors, keeps track of which subpredictor has performed the best on the sequence so far, and uses that as the source of its predictions. This minimizes the error in estimates introduced by having many distinct predictors with very similar performance – the ensemble predictor's performance is measured on its predictions, not on the performance of the predictions of any one of its subpredictors.

Multi Most Common in Window Predictor (MultiMCW) comprises several MCW subpredictors, each of which maintains a window of the most recently observed w samples, and predicts the value that has appeared most often in that w-sample window. If a subpredictor encounters a tie, the most common sample value that has appeared most recently is used as the prediction. The MCW subpredictor with the highest score is used for predicting the next sample. This ensemble predictor is parameterized by the window size w, where $w \in \{63, 255, 1023, 4095\}$. This predictor was designed for cases where the source transitions over time between different most-common values, but still remains relatively stationary over reasonable lengths of the sequence.

Lag Predictor is an ensemble predictor that contains d subpredictors, one for each lag $i \in \{1, \ldots, d\}$, for a maximum depth d. Each of these subpredictors predicts the value at time t as $x_t = x_{t-i}$. The lag subpredictor with the highest score is used for predicting the next sample.

Multi Markov Model with Counting Predictor (MultiMMC) is another form of ensemble classifier, comprising multiple MMC predictors. In particular, the parameter D specifies the number of MMC subpredictors, where each MMC sub predictor is parameterized by $N \in \{1, \ldots, D\}$. This predictor is expected to be most successful when the source has a limited amount of memory, so that the next output produced by the source is related to its most recent k outputs, but not strongly related to more distant outputs.

Numerical Predictors We now describe predictors that assume that the samples are numerical, thus that numerical relationships between them can be inferred. Numerical models generally represent continuous data, whereas outputs of the entropy sources are discrete. An issue that occurs in this class of predictors that did not exist for categorical predictors is that the outputs from a numerical model may not exist in the output alphabet of the source. Because of this discrepancy in data types, the numerical predictors are constructed from two parts:

1. A numerical model and numerical prediction function, and
2. A *grid* that remembers all values seen so far and rounds all predictions to the nearest value that has been seen so far.

Moving Average (MA) Predictors are a family of predictors, parameterized by w, that compute the average of the last w values seen. An MA predictor uses the output value observed so far that is closest to that average as the prediction. This predictor is expected to be most successful when the source has a symmetric distribution that moves relatively slowly over time. The MA predictor is used in our experiments inside the MultiMA ensemble predictor.

First Difference (D1) Equation Predictor constructs a difference equation on a few recent sample values, and uses it to predict the next value. For example, if the difference between the previous value x_{t-1} and the one before that x_{t-2} was δ, then the predictor computes $x_{t-1} + \delta$, and then uses the output value seen so far that is closest to that value as the prediction. This predictor is expected to be most successful when the rate of change of the values output by the source is relatively stable.

MultiMA Predictor is an ensemble predictor that keeps track of multiple MA predictors, and uses the MA predictor that has been most successful so far to generate its predictions.

4 Experimental Results

To determine whether simple predictive models were effective for the purpose of min-entropy estimation, we have applied the predictors presented above to simulated and real-world[3] data. We have also compared our results to the entropy estimators presented in SP 800-90B[3].

[3] Any mention of commercial products within NIST web pages is for information only; it does not imply recommendation or endorsement by NIST.

4.1 NIST Entropy Estimation Suite

Draft NIST SP 800-90B [3] includes five estimators, which were originally specified in [8, 7]. These estimators are suitable for sources that do not necessarily satisfy the i.i.d. assumption.

- *Collision test* computes entropy based on the mean time for a repeated sample value.
- *Partial collection test* computes entropy based on the number of distinct sample values observed in segments of the outputs.
- *Markov test* estimates entropy by modeling the noise source outputs as a first-order Markov model.
- *Compression test* computes entropy based on how much the noise source outputs can be compressed.
- *Frequency test* computes entropy based on the number of occurrences of the most-likely value.

We refer to the estimators as the 90B estimators.

4.2 Simulated Data

Datasets of simulated sequences were produced using the following distribution families:

- *Discrete uniform distribution:* This is an i.i.d. source in which the samples are equally-likely.
- *Discrete near-uniform distribution:* This is an i.i.d source where all samples but one are equally-likely; the remaining sample has a higher probability than the rest.
- *Normal distribution rounded to integers:* This is an i.i.d. source where samples are drawn from a normal distribution and rounded to integer values.
- *Time-varying normal distribution rounded to integers:* This is a non-i.i.d. source where samples are drawn from a normal distribution and rounded to integer values, but the mean of the distribution moves along a sine curve to simulate a time-varying signal.
- *Markov Model:* This is a non-i.i.d. source where samples are generated using a kth-order Markov model.

80 simulated sources were created in each of the classes listed above. A sequence of 100,000 samples was generated from each simulated source, and estimates for min-entropy were obtained from the predictors and 90B estimators for each sequence. For each source, the correct min-entropy was derived from the known probability distribution.

Figure 2 shows the results of the lowest estimate given by the 90B estimators[4] and the lowest estimate given by the predictors presented in this work, applied

[4] The implementation of the SP 800-90B estimators is slightly modified by removing the restriction that the output space is $[0, ..., 2^b - 1]$, where b is the maximum number of bits required to represent output values.

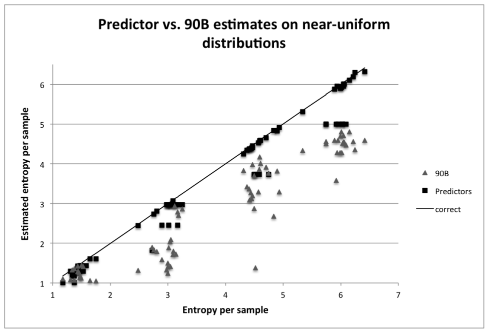

Fig. 2: Comparison of the lowest predictor entropy estimate, the lowest 90B entropy estimate, and the true entropy from 80 simulated sources with near-uniform distributions.

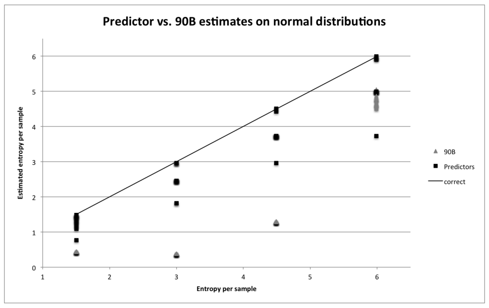

Fig. 3: Comparison of the lowest predictor entropy estimate, the lowest 90B entropy estimate, and the true entropy from 80 simulated sources with normal distributions.

to simulated sources with near-uniform distributions. Near-uniform distributions are particularly interesting because the majority of the 90B estimators try to fit the data to a distribution in that family. Thus, one would expect the 90B estimators to work quite well. However, the plot shows that this is not always the case – there are several points where the 90B methods give massive underestimates.

Figure 3 shows results for the simulated sources with normal distributions. For this class of simulated source, the 90B estimators are prone to large underestimates. In most cases, the minimum estimate is the result of the partial collection estimator, although the compression and collision estimates are quite low as well. The results of the partial collection and collision estimates are highly dependent on the segment size, and it is unclear whether the current strategy for selecting the segment size is optimal. The compression estimator, based on Maurer's universal statistic[12], does not contain the corrective factor $c(L, K)$ that is used to reduce the standard deviation to account for dependencies between variables, and this is likely a factor in the low estimates.

Plots depicting the results of the lowest 90B and predictor estimates for the remaining simulated distribution families are located in Appendix A.

Table 1: Error measures for the lowest 90B and predictor estimates by simulated source class.

Simulated data class	90B MSE	Predictor MSE	90B MPE	Predictor MPE
Uniform	2.4196	0.5031	37.9762	17.4796
Near-uniform	1.4136	0.1544	26.6566	6.4899
Normal	4.9680	0.4686	62.6330	14.1492
Time-varying normal	3.0706	0.2564	54.1453	3.1706
Markov	0.9973	0.8294	6.4339	-11.7939

None of the 90B or predictor estimates were overestimates for the uniform sources, which is to be expected. Overall, underestimates given by the predictors were smaller than those given by the 90B estimators.

The predictors did give a number of overestimates when applied to the Markov and time-varying normal sources, particularly as the true min-entropy increases. This suggests that the predictors, with the parameters used in these experiments, were unable to accurately model these sources. The 90B estimators gave both significant overestimates and underestimates for the Markov sources, and tended towards large underestimates for the time-varying normal sources.

While it can be useful to look at the trends, it is often more informative to compare the errors. Table 1 shows the mean squared error (MSE) of the lowest 90B estimate and the lowest predictor estimate over 80 sequences from each class of simulated sources. For all five classes, the MSE was lower for the predictors than it was for the 90B estimators. This suggests that the predictors are better estimators; however, the MSE does not tell the entire story. Because of the nature of the problem, underestimates are preferred to overestimates, and MSE does not capture the sign of the error. To capture this, the mean percentage error (MPE) is provided in Table 1 as well.

The MPE values show that the average errors from the 90B and predictor estimates have the same sign, except in the case of the Markov sources.

4.3 Real-World Data

Results were also obtained using random number generators deployed in the real world. The true entropy per sample for these sources is unknown, so no error can be computed for the estimators. However, the estimates from the predictors presented here can still be compared to the 90B estimates, based on the knowledge that underestimates from predictors have theoretical bounds. The estimates of the real world sources are presented in Table 2.

RDTSC Three sequences were generated using the the last bit returned by calls to RDTSC, which returns the number of clock cycles since system startup. *RDTSC1* has an output alphabet of $\{0, 1\}$, *RDTSC4* has an output alphabet of $\{0, \ldots, 15\}$, and *RDTSC8* has an output alphabet of $\{0, \ldots, 255\}$. These sequences are processed. In particular, Von Neumann unbiasing was applied to the raw sequence generated by the repeated calls to RDTSC.

The lag predictor gives the lowest estimate for *RDTSC1*, the MultiMMC predictor gives the lowest estimate for *RDTSC4*, and the compression estimate gives the lowest estimate for *RDTSC8*. In *RDTSC1*, the lag predictor provides an estimate 0.205 below that of the 90B collision estimate, suggesting that there was periodicity that the 90B estimators were unable to detect. The predictors did not achieve significant gains over uninformed guessing when applied to *RDTSC8*, with the LZ78Y estimator performing particularly poorly on this sequence.

RANDOM.ORG [15] is a service that provides random numbers based on atmospheric noise. It allows the user to specify the minimum and maximum values that are output. The sequence used here consisted of bytes.

The predictors did not achieve significant gains over uninformed guessing when applied to this sequence, with the LZ78Y estimator performing particularly poorly on this sequence. One would expect that this is because of the cryptographic processing; the entropy estimates should be close to eight bits of entropy per sample. However, the 90B estimates are between 5.1830 and 5.6662. Although we cannot prove it, we suspect that this discrepancy comes from the inaccuracy of the estimators, rather than a weakness of the source.

Ubld.it The final two real-world sequences in this paper come from a TrueRNG device by Ubld.it [18]. The *Ubld.it1* sequence contained bits, and the *Ubld.it8* is the byte interpretation of the *Ubld.it1* bit sequence. The outputs of this dataset have been cryptographically processed and should resemble an ideal uniform distribution.

The difference between the lowest 90B and predictor estimates for the *Ubld.it1* sequence was only 0.0071, which is not a significant difference. The results for *Ubld.it8* are similar to those of the *RANDOM.ORG* and *RDTSC8* datasets – the predictors did not achieve significant gains over uninformed guessing, and the LZ78Y estimator gave an impossibly high result.

Across Datasets It is also informative to look at results across the real-world datasets, particularly when looking at bytes. For byte sequences, the 90B estimates are between five and six bits of entropy per sample, with the collision and compression estimators providing the lowest estimates. The LZ78Y predictor, on the other hand, provided impossible results of over 11 bits of entropy per sample. This indicates that the models constructed by the LZ78Y predictor are not good fits for these bytes sequences.

4.4 General Discussion

It is interesting that in both the simulated and real-world datasets, the 90B estimators seem prone to greater underestimation as the sequence sample size

Table 2: Entropy estimates for real world sources. The lowest entropy estimate for each source is shown in bold font.

Estimator	RDTSC1	RDTSC4	RDTSC8	RANDOM.ORG	Ubld.it1	Ubld.it8
Collision	0.9125	3.8052	5.3240	**5.1830**	0.9447	**5.2771**
Compression	0.9178	3.6601	**5.3134**	5.1926	0.9285	5.5081
Frequency	0.9952	3.9577	5.8666	5.6662	0.8068	5.8660
Markov	0.9983	3.9582	5.7858	5.3829	0.8291	5.7229
Partial Collection	0.9258	3.7505	5.3574	5.5250	0.9407	5.8238
D1	0.9616	3.9986	7.9619	7.9126	0.8734	7.9489
Lag	**0.7075**	3.9883	7.9546	7.9237	**0.7997**	7.9862
LZ78Y	0.9079	3.9989	11.9615	11.5924	**0.7997**	11.8375
MultiMA	0.9079	3.6458	7.9594	7.8508	0.8073	7.9441
MultiMCW	0.9079	3.9888	7.9381	7.9744	0.8072	7.9544
MultiMMC	0.9079	**3.6457**	7.9663	7.9237	0.8072	7.9880

goes from bits to bytes. There are two limiting factors as sample sizes increase. First, the near-uniform distribution only contains two probability levels (p and q, where $p > q$), and any source distribution with more than two levels seems to cause p to increase, and therefore, the entropy decreases. Second, the Markov estimate "maps down" the sequence so that only six bits are used to construct the first-order model. Therefore, estimates from the set of 90B estimators are capped at six bits of entropy per sample.

5 Conclusions

In this work, we attempted to estimate the min-entropy of entropy sources using predictive models, and show that even simplistic learners are capable of estimating entropy. We have also compared results from our simplistic learners with those of the entropy estimation suite provided in [3].

Barak and Halevi [2] criticize the approach of estimating the entropy from the point of an attacker, by just testing the outputs. We agree that the entropy estimation of a noise source should be done by analyzing the physical properties of the source, constructing a model of its behavior, and using that to determine how much unpredictability is expected from the source. However, there are still a number of places where external tests of entropy estimates are very useful:

For the designer The best efforts of the designer to understand the behavior of his noise source may not be fully successful. An independent test of the unpredictability of the source can help the designer recognize these errors.

For an evaluator A testing lab or independent evaluator trying to decide how much entropy per sample a source provides will have limited time and expertise to understand and verify the designer's analysis of his design. Entropy tests are very useful as a way for the evaluator to double-check the claims of the designer.

For the user A developer making use of one or more noise sources can sensibly use an entropy estimation tool to verify any assumptions made by the designer.

Predictors are well-suited to providing a sanity-check on the entropy estimation done by the designer of a source based on some kind of deeper analysis, because they give an upper-bound estimate, which is very unlikely to be much below the correct entropy per sample. If the designer's model indicates that a source gives h bits of entropy per sample, and a predictor consistently estimates that it has much less than h bits/sample, this is strong evidence that the designer's estimate is wrong.

Additionally, a designer who has a good model for his noise source can turn it into a predictor, and get an entropy estimate based on that model in a straightforward way. He can then evaluate the entropy of his source based on the minimum of these simple, general-purpose predictors and his own more carefully tailored one.

5.1 Future Work

This work shows the usefulness of a number of simple, generic predictors for entropy estimation. In future work, we will adapt mainstream classification algorithms and data stream mining algorithms to fit the predictor framework, and examine their effectiveness as generic predictors.

Our hope is that this work inspires additional research in two different directions:

1. We hope that experts on the physical properties of specific noise sources will use the predictor framework to design better predictors that capture the behavior of those sources more precisely than our generic predictors.
2. We hope that experts from the machine learning community will bring more sophisticated machine-learning tools to bear on the practical problem of the entropy estimation of noise sources.

References

1. Antos, A., Kontoyiannis, I.: Convergence properties of functional estimates for discrete distributions. Random Struct. Algorithms 19(3-4), 163–193 (Oct 2001), `http://dx.doi.org/10.1002/rsa.10019`
2. Barak, B., Halevi, S.: A model and architecture for pseudo-random generation with applications to /dev/random. In: Proceedings of the 12th ACM Conference on Computer and Communications Security. pp. 203–212. CCS '05, ACM, New York, NY, USA (2005), `http://doi.acm.org/10.1145/1102120.1102148`
3. Barker, E., Kelsey, J.: NIST Draft Special Publication 800-90 B: Recommendation for the Entropy Sources Used for Random Bit Generation (August 2012), http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf
4. Dorrendorf, L., Gutterman, Z., Pinkas, B.: Cryptanalysis of the random number generator of the windows operating system. ACM Trans. Inf. Syst. Secur. 13(1), 10:1–10:32 (Nov 2009)

5. Feller, W.: An Introduction to Probability Theory and its Applications, vol. One, chap. 13. John Wiliey and Sons, Inc. (1950)
6. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the linux random number generator. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy. pp. 371–385. SP '06, IEEE Computer Society, Washington, DC, USA (2006), `http://dx.doi.org/10.1109/SP.2006.5`
7. Hagerty, P.: Presentation of non-iid tests. In: NIST Random Bit Generation Workshop (2012), http://csrc.nist.gov/groups/ST/rbg_workshop_2012/hagerty.pdf
8. Hagerty, P., Draper, T.: Entropy bounds and statistical tests. In: NIST Random Bit Generation Workshop (2012), http://csrc.nist.gov/groups/ST/rbg_workshop_2012/hagerty_entropy_paper.pdf
9. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your Ps and Qs: Detection of widespread weak keys in network devices. In: Proceedings of the 21st USENIX Security Symposium (Aug 2012)
10. Kontoyiannis, I., Algoet, P., Suhov, Y.M., Wyner, A.: Nonparametric entropy estimation for stationary processes and random fields, with applications to English text. IEEE Trans. Inform. Theory 44, 1319–1327 (1998)
11. Lauradoux, C., Ponge, J., Roeck, A.: Online Entropy Estimation for Non-Binary Sources and Applications on iPhone. Research Report RR-7663, INRIA (Jun 2011), `https://hal.inria.fr/inria-00604857`
12. Maurer, U.M.: A universal statistical test for random bit generators. J. Cryptology 5(2), 89–105 (1992), `http://dx.doi.org/10.1007/BF00193563`
13. FIPS PUB 140-2, Security Requirements for Cryptographic Modules (2002), U.S. Department of Commerce/National Institute of Standards and Technology
14. Paninski, L.: Estimation of entropy and mutual information. Neural Comput. 15(6), 1191–1253 (Jun 2003), `http://dx.doi.org/10.1162/089976603321780272`
15. RANDOM.ORG: `https://www.random.org/`
16. Sayood, K.: Introduction to Data Compression, chap. 5. Morgan Kaufmann, third edn. (2006)
17. Shannon, C.: Prediction and entropy of printed english. Bell System Technical Journal 30, 50–64 (January 1951), https://archive.org/details/bstj30-1-50
18. ubld.it: TrueRNG, `http://ubld.it/products/truerng-hardware-random-number-generator/`
19. Wyner, A.D., Ziv, J.: Some asymptotic properties of the entropy of a stationary ergodic data source with applications to data compression. IEEE Transactions on Information Theory 35(6), 1250–1258 (1989)

A Additional Figures

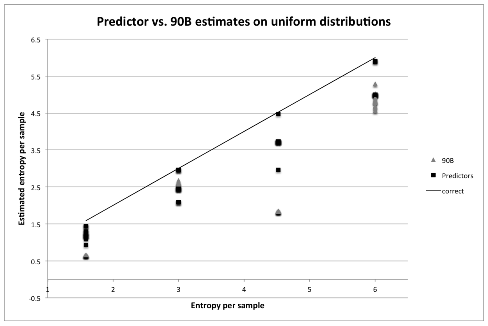

Fig. 4: Comparison of lowest predictor entropy estimate, lowest 90B entropy estimate, and the true entropy from 80 simulated sources with uniform distributions.

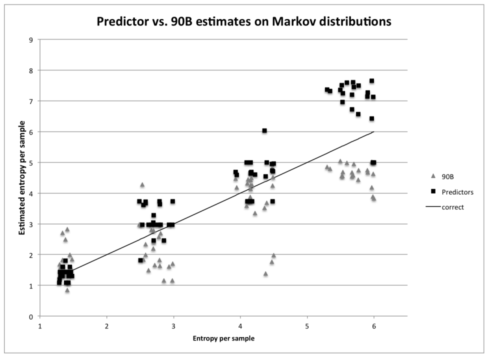

Fig. 5: Comparison of lowest predictor entropy estimate, lowest 90B entropy estimate, and the true entropy from 80 simulated sources with Markov distributions.

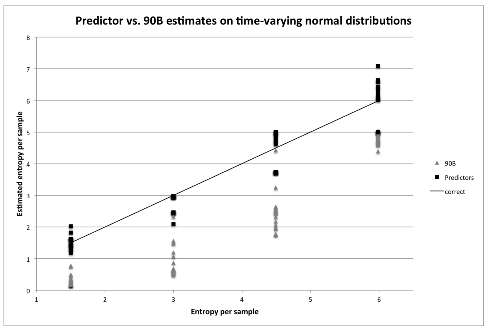

Fig. 6: Comparison of lowest predictor entropy estimate, lowest 90B entropy estimate, and the true entropy from 80 simulated sources with time-varying normal distributions.