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 cryptography. In most real-world systems, these values come from a cryptographic pseudorandom number generator (PRNG), which in turn is seeded by an entropy source. The security of the entire cryptographic system then relies on the accuracy of the claimed amount of entropy provided by the source. If the entropy source provides less unpredictability than is expected, the security of the cryptographic mechanisms is undermined, as in[10, 5, 7]. For this reason, correctly estimating the amount of entropy available from a source is critical.
In this paper, we develop a set of tools for estimating entropy, based on mechansims that attempt to predict the next sample in a sequence based on all previous samples. These mechanisms are called *predictors*. We develop a framework for using predictors to estimate entropy, and test them experimentally against both simulated and real noise sources. For comparison, we subject the entropy estimates defined in the August 2012 draft of NIST Special Publication 800-90B[4] to the same tests, and compare their performance.

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. Typically, these values are generated by a cryptographic pseudorandom number generator (PRNG). A good cryptographic PRNG is capable of generating outputs which can stand in for truly random numbers for cryptographic applications. However, PRNGs are deterministic algorithms; their security ultimately depends on a truly unpredictable seed.

The seed comes from an *entropy source*. 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 seeding a cryptographic PRNG, the relevant measure is min-entropy, which corresponds to the difficulty of guessing or predicting the most-likely output of the entropy source. Correctly determining how much entropy is present in a sequence of outputs is critical–researchers have shown examples of deployed systems whose PRNG seeds were not sufficiently unpredictable, and as a result, cryptographic keys were compromised[10, 5, 7]. Estimating the amount of entropy that an entropy source 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 [2, 15]). Methods based on compression algorithms (e.g., see [11, 20]) use match length or frequency counting to approximate entropy. Hagerty and Draper provided a set of entropic statistics and bounds on the entropy in [9]. Lauradoux et al. [12] provided an entropy estimator for non-binary sources with an unknown probability distribution that converges towards Shannon entropy.

Draft NIST Special Publication (SP) 800-90B [4] 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 entropy assessment of the source is the minimum of the estimates obtained from the five estimators. This estimate is intended to be used for entropy sources that undergo validation testing in order to comply with Federal Information Processing Standard 140-2 [14].

It is important to note that good entropy estimation requires knowledge of the underlying nondeterministic process being used by the entropy source; statistical tests such as those referenced above and in this paper can only serve as a sanity check on that kind of estimate. Indeed, many of the experiments described in the remainder of this paper are based on using a non-cryptographic PRNG with a short seed to convincingly simulate an entropy source with some set of properties, in order to test the entropy estimators.

1.1 Entropy and Predictability

Shannon first investigated the relationship between the entropy and predictability of a sequence in 1951 [18], using the ability of humans to predict the next character in the text to estimate the entropy per character. In fact, predictability is exactly what we are interested in when considering the performance of an entropy source. An inadequate source of entropy allows the attacker to predict the PRNG's seed, and thus to predict all future pseduorandom numbers produced by the PRNG. More recently, in [1], there is a discussion of a method called a predictor, but details on how it works is missing.

1.2 Our Contributions

In this paper, we develop an alternative approach to estimating min-entropy, based on the ability of any of several models to successfully predict a source's outputs. These models are called *predictors*. Each predictor is an algorithm that attempts to predict each element in a sequence of samples, and updates its internal state based on the samples seen so far. A predictor that successfully predicts $\frac{1}{2}$ of the samples from a source demonstrates that the sequence elements can be predicted with probability at least $\frac{1}{2}$–at worst, an attacker could use the same algorithm to predict the sequence as is used by the predictor, but he might manage to find a better algorithm. That translates to a min-entropy of the source of at most 1 bit/sample, because $-\log_2(\frac{1}{2}) = 1$.

Our contributions include:

1. We introduce the use of *predictors* to estimate min-entropy of a noise source.
2. We describe a framework for using predictors for entropy estimation, including:
   (a) Entropy estimates from both global and local performance.
   (b) The use of ensemble predictors to combine similar predictors together.
   (c) The strategy of taking the minimum of all predictors' estimates.
3. We define a starting set of predictors which perform well in experiments, and which may be used directly for entropy estimation.
4. We describe a set of experiments to compare the performance of our predictors against that of the non-i.i.d. entropy estimators in the draft SP 800-90B[4], on two different kinds of data:
   (a) Data from simulated sources, whose correct entropy/sample is known.
   (b) Data from real-world hardware RNGs, whose correct entropy/sample is not known.
5. In both cases, our predictors performed very well, giving more accurate estimates than the 90B estimates where the correct entropy/sample was known, and comparable or lower estimates for the real-world sources.

1.3 Guide to the Rest of the Paper

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 Section 4 gives descriptions of several simple predictors. Section 5 provides experimental results, and the last section concludes the study with some future directions.

2 Preliminaries

Entropy sources, as defined in [4], 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 are concerned only with the noise source outputs, and estimating the entropy/sample present in them. Also, in this study, we assume that the samples obtained from a noise source consist of fixed-length bitstrings, which can be considered either as symbols in an alphabet, or as integers.

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 of length L is defined as

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

The entropy per sample can be approximated by dividing H by L.

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.

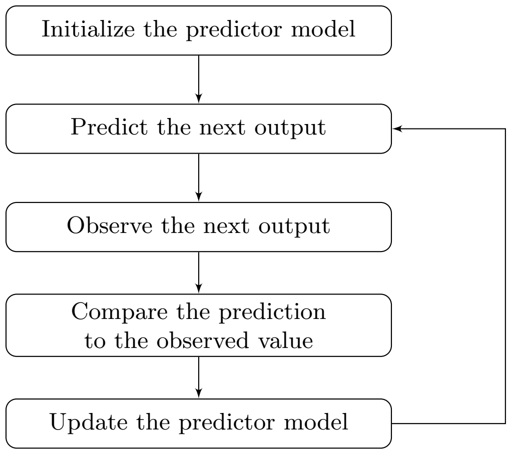
Initialize the predictor model
Predict the next output
Observe the next output
Compare the prediction to the observed value
Update the predictor model

Fig. 1: General strategy of predictors

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 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.

A predictor's performance can be expressed as a probability, and it provides a *lower bound* on the best performance an attacker could get predicting the source's outputs–an attacker will never do *worse* than the predictor (he could just reuse it directly), but he may do *better*. That lower bound on the probability of making a correct prediction gives an *upper bound* on the entropy/sample of the source–the more predictable a source is, the less entropy it has. This relationship is easy to see in the formula for min-entropy: If p^* is the probability of predicting the next sample using the best possible predictor, then $H = -\log_2(p^*)$ is the min-entropy of the next sample. As p^* gets closer to one, H gets closer to 0; as p^* gets closer to 0, H gets bigger.

Our predictors estimate entropy by first estimating the probability of successful prediction (both globally and locally) with a 95% upper bound, and then computing the min-entropy that would correspond to that success probability. The 95% upper bound ensures somewhat conservative estimates, and also matches the way entropy estimates are calculated by the estimators in 90B.

3.2 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

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

Our predictors 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

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

3.3 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 statistical tools used to analyze 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 $1-x+qp^r x^{r+1} = 0$ [6]. The root x can be efficiently approximated

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

using the recurrence relation $x_{i+1} = 1 + qp^r x_i^{r+1}$, as it converges on the root. In our experiments, ten iterations appear to be sufficient to converge on the right answer. 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

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

3.4 Deriving a Final Estimate

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

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

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 successfully predict samples from sources with a certain kind of behavior. Then, we generate a long output sequence from the source, 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 some predictable kind of behavior in the 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. By taking the minimum of all the predictors' estimates, we can guarantee that the predictor that was most effective at predicting the source's outputs determines the entropy estimate.

3.5 Underestimates and Overestimates

Predictors work in four steps:

1. Assume a probability model for the source.
2. Estimate the model's parameters from the input sequence on the fly.
3. Use these parameters to try to predict the still-unseen values in the input sequence.
4. Estimate the min-entropy of the source from the performance of these predictions.

This last step means that models that are a bad fit for the source can give big *overestimates*, but not big *underestimates* in entropy. A model that's a bad fit for the source will lead to inaccurate predictions (by definition), and thus will lead to a too-high estimate of the source's entropy. This distinguishes predictors from other entropy estimation mechanisms, such as those of [4], which give consistent, large underestimates for some source distributions.

Predictors can still give underestimates for the source's entropy, but these are based on successful predictions of the samples from the source. For example, a predictor which should be able to predict $\frac{1}{2}$ of the samples on average can have a run of unusually successful performance by chance, leading to an underestimate of entropy. Also, our predictors estimate a 95% upper bound on the success probability of the predictor from its performance, which also leads to underestimates on average. However, both of these effects are easy to bound, and for reasonable numbers of samples, they are small; by contrast, several of the 90B estimators are subject to huge systematic underestimates of entropy.

This means that the strategy of applying a large set of very different predictors to the sequence of noise source samples, and taking the minimum estimate, is workable. If five predictors whose underlying models are very bad fits for the noise source are used alongside one predictor whose underlying model fits the source's behavior well, the predictor that fits well will determine the entropy estimate–predictors that are a bad fit will never give large systematic underestimates. Further, this means that it's reasonable to include many different predictors–adding one more predictor seldom does any harm, and occasionally will make the entropy estimate much more accurate.

4 A Concrete Set of Predictors for Entropy Estimation

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[4], produce discrete values from a fixed alphabet, represented as bitstrings. Therefore, we consider predictors as a solution to a classification problem, rather than a regression problem. However, we can still build predictors for numerical data (samples whose integer values are meaningful as integers), as well.

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. However, for noise sources where few assumptions about the underlying probability distribution can be made, it may be difficult to construct sophisticated learners.

4.1 Ensemble Predictors

Many of the predictors described below are *ensemble predictors*. An ensemble predictor is constructed using two or more subpredictors. At each point in the sequence of samples being processed, the ensemble predictor keeps track of which subpredictor has made the most correct predictions so far–that subpredictor's current prediction is used as the ensemble predictor's current prediction. Note that the ensemble predictor's final entropy estimate is based on the success of its own predictions. It is possible in principle for one of the subpredictors to have a higher final probability of successful prediction than the ensemble predictor.

Our ensemble predictors combine many similar subpredictors with slightly different model parameters–they allow us to construct a single predictor that in effect has many different choices for those model parameters at once, and which will choose the best set of parameters based on the data. A major benefit of using an ensemble predictor is that, for many sources, a collection of very similar predictors differing only in a few parameter choices would give very similar predictions. If we simply applied all these closely-related predictors directly and then took the minimum, we would introduce a systematic underestimate when all the closely-related predictors got similar results. By combining the similar predictors into a single ensemble predictor, we avoid this issue–adding more predictors with different parameters to the ensemble predictor will not introduce a bias toward underestimates.

4.2 Categorical Data Predictors

Here, we describe 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) 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 is a tie for most common value in the window the window, the value that has occurred most recently is used. We expect the MCW predictor 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 remembers the most recent N values seen, and predicts the one that appeared N samples back in the sequence, where N is a parameter to the predictor. We expect the single lag predictor to perform well on sources with strong periodic behavior, if N is close to the period. In our experiments, this predictor is used inside the *Lag* ensemble predictor.

Markov Model with Counting (MMC) remembers every N-sample string that has been seen so far, and keeps counts for each value that followed each N-sample string. N is a parameter for this predictor. We expect the MMC predictor to perform well on data from any 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 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, it keeps track of the most common value that immediately followed the string. (Note that even after the dictionary reaches maximum size, the counts continue to be updated.) Whenever the most recently seen samples match an entry in its dictionary, the predictor finds the longest such match, and predicts that the next sample will be the value that most often has followed this string so far. This predictor is based loosely on the LZ78 family of compression algorithms[17]. We expect the LZ78Y predictor to perform well 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) contains 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. This ensemble predictor is parameterized by the window sizes of its subpredictors w, where $w \in \{63, 255, 1023, 4095\}$ for our experiments. We expect MultiMCW to perform well in cases where the most common value changes over time, but not too quickly. The wide range of window sizes is intended to give the predictor a good chance of tracking well with many different rates of change of the most common value.

Lag Predictor contains d subpredictors, one for each lag $i \in \{1, \dots, d\}$, for a maximum depth d. This ensemble predictor is parameterized by d. We expect Lag to perform well on sources that exhibit periodic behavior, where the period is somewhere between 1 and d.

Multi Markov Model with Counting Predictor (MultiMMC) contains multiple MMC predictors. D is a parameter to the ensemble predictor, and specifies the number of MMC subpredictors, where each MMC sub predictor is parameterized by $N \in \{1, \dots, D\}$. We expect MultiMMC to perform well on sources which can be modeled reasonably well by a Markov model of depth $\{1, \dots, D\}$.

4.3 Numerical Predictors

We now describe predictors that assume that the samples are numerical–that is, that the integer values of the samples have some numerical meaning. Numerical models generally represent continuous data, whereas outputs of the entropy sources are discrete. This raises an issue that did not exist for categorical predictors: 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 compute the average of the last w values seen, where w is a parameter to the predictor. Note that the MA predictor always rounds its prediction to the nearest integer value it has seen so far in the sequence–on a binary sequence, an average of 0.7 leads to a prediction of a 1. We expect MA to be most successful when there is some periodic component to the source's behavior with period close to w. In our experiments, MA is used inside the *MultiMA* ensemble predictor.

First Difference (D1) Equation Predictor constructs a difference equation on the two most 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. We expect D1 to be most successful when the source's behavior can be well-described by a first-order difference equation.

Ensemble Methods for Numerical Data Our single ensemble predictor for numerical data works in the same way as the ensemble predictors for categorical data work–it keeps track of which subpredictor has made the most correct predictions, and uses that one to make the next prediction of its own. Once again, the entropy estimate that results is based on the success of the ensemble predictor's predictions, not necessarily that of its most successful subpredictor.

MultiMA Predictor contains multiple MA predictors, and is parameterized by the window sizes $w \in \{16, 32, 64, 128, 256, 512, 1024\}$ in our experiments.

5 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[2] data. We have also compared our results to the entropy estimators presented in SP 800-90B[4]. Our predictors in these experiments

[2] Any mention of commercial products or organizations is for informational purposes only; it is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology nor is it intended to imply that the products identified are necessarily the best available for the purpose.

compute an $\alpha = 0.05$ upper-bound on the estimated probability from which a min-entropy estimate is computed. This matches the approach and value of α used in the 90B estimates.

5.1 NIST Entropy Estimation Suite

Draft NIST SP 800-90B [4] includes five estimators, which were originally specified in [9, 8]. 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.

5.2 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.

Eighty 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.

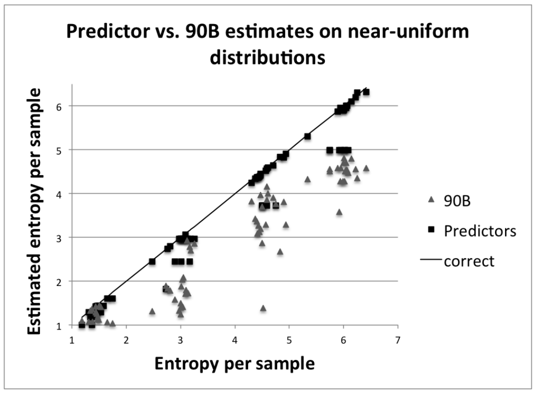

Fig. 2: Comparison of the lowest predictor entropy estimate, the lowest 90B entropy estimate, and the true entropy from 80 simulated sources with near-uniform distributions.

Figure 2 shows the results of the lowest estimate given by the 90B estimators[3] and the lowest estimate given by the predictors presented in this work, applied 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[13], 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.

Figure 4 shows that 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.

Figures 5 and 6 show that 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

[3] 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.

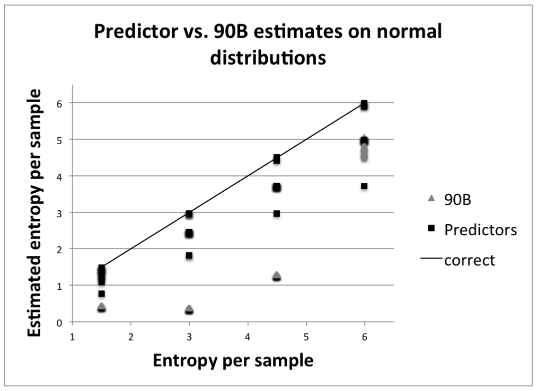

Fig. 3: Comparison of the lowest predictor entropy estimate, the lowest 90B entropy estimate, and the true entropy from 80 simulated sources with normal distributions.

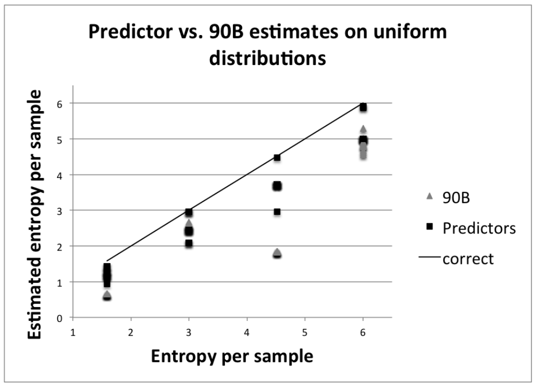

Fig. 4: Comparison of lowest predictor entropy estimate, lowest 90B entropy estimate, and the true entropy from 80 simulated sources with uniform distributions.

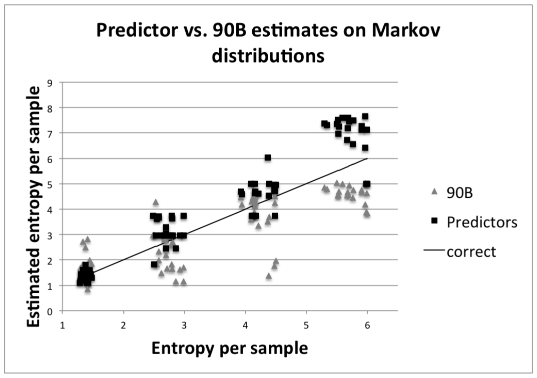

Fig. 5: Comparison of lowest predictor entropy estimate, lowest 90B entropy estimate, and the true entropy from 80 simulated sources with Markov distributions.

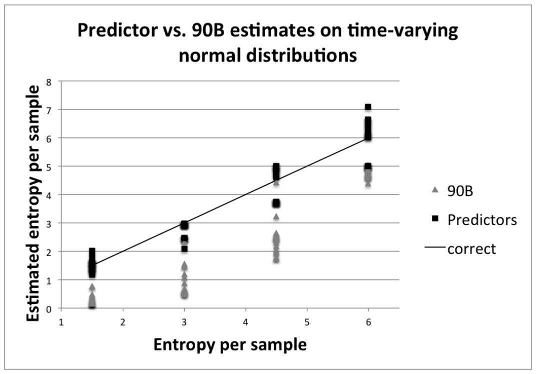

Fig. 6: Comparison of lowest predictor entropy estimate, lowest 90B entropy estimate, and the true entropy from 80 simulated sources with time-varying normal distributions.

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

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.

5.3 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 $\{0, \dots, 15\}$, and *RDTSC8* has an output alphabet of $\{0, \dots, 255\}$. 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 [16] 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 [19]. The *Ubld.it1* sequence contained bits, and the *Ubld.it8* is the byte interpretation of the *Ubld.it1* bit sequence.

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.

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

5.4 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 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.

6 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 [4].

Barak and Halevi [3] 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.

6.1 Future Work

This work shows the usefulness of a number of simple, generic predictors for entropy estimation. The predictor framework is very closely related to the classification problem in machine learning. While the predictors presented in this work are simple, more sophisticated learning techniques may be used to construct more complex predictors. 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. Also, in future work, we hope to adapt the predictor framework for real-time health testing of noise sources.

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.

7 Acknowledgments

We would like to thank Stefan Lucks for his suggestion to a performance metric that considered runs of correct predictions. We would also like to thank Tim Hall for his implementations of the entropy estimates in [9], and John Deneker, Tim Hall, and Sonu Sankur for providing samples from real-world noise sources for testing.

References

1. Cryptography Research Inc., Evaluation of VIA C3 Nehemiah Random Number Generator. Tech. rep. (Revision Dated: February 27,2003), `http://www.cryptography.com/public/pdf/VIA_rng.pdf`
2. 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`
3. 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`
4. 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
5. 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)
6. Feller, W.: An Introduction to Probability Theory and its Applications, vol. One, chap. 13. John Wiliey and Sons, Inc. (1950)
7. 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`
8. 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
9. 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
10. 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)
11. 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)
12. 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`
13. 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`
14. FIPS PUB 140-2, Security Requirements for Cryptographic Modules (2002), U.S. Department of Commerce/National Institute of Standards and Technology
15. Paninski, L.: Estimation of entropy and mutual information. Neural Comput. 15(6), 1191–1253 (Jun 2003), `http://dx.doi.org/10.1162/089976603321780272`
16. RANDOM.ORG: `https://www.random.org/`
17. Sayood, K.: Introduction to Data Compression, chap. 5. Morgan Kaufmann, third edn. (2006)
18. Shannon, C.: Prediction and entropy of printed english. Bell System Technical Journal 30, 50–64 (January 1951), https://archive.org/details/bstj30-1-50
19. ubld.it: TrueRNG, `http://ubld.it/products/truerng-hardware-random-number-generator/`

20. 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)