

# Estimating Information Theoretic Quantities of spike-trains using the Context Tree Weighting algorithm

Michael London<sup>1</sup>

Adi Schreiber<sup>2</sup>

Idan Segev<sup>1</sup>

## **Appendix of:**

### **“The information efficacy of a synapse”**

Michael London, Adi Schreiber, Micahel Häusser, Matthew E. Larkum & Idan Segev  
nature neuroscience volume 5 no 4 april 2002

<sup>1</sup>Department of Neurobiology Institute of Life Sciences  
and Center for Neural Computation,  
Hebrew University, Jerusalem 91904, Israel.

<sup>2</sup>Department of Computer Science,  
Hebrew University, Jerusalem 91904, Israel.

Correspondence please send to:

Michael London

Tel: 972-2-6586705 (work) Fax: 972-2-6586296 email: mikilon@lobster.ls.huji.ac.il

# 1 Background

In the following we present a method to estimate the entropy of a spike-trains and the mutual information ( $MI$ ) between two individual spike-trains. This is the first time this method has been used for the analysis of spike-trains. Most of the ideas presented are adapted from a series of works in the field of data compression (Willems et al., 1995). This document is self contained and provides all the required background to understand the method used in the paper by London et al. (2001). Many preliminary definitions are needed before describing the method, however for the benefit of readers who are acquainted with them, they will be presented at section 5 at the end of this document.

## 1.1 Spike-trains as Stochastic Processes and Information Sources

Assume we monitor the activity of an axon for a time duration  $T$ . The sequence of times of occurrence of spikes in this axon  $\{t_0, t_1, \dots, t_n\}$ , where  $0 \leq t_0 < t_1 \dots < t_n \leq T$ , is termed a **spike-train**. It is a finite and deterministic sequence. It is useful to consider the times of spike occurrence  $\{t_i\}$  with a certain degree of accuracy  $\Delta t$  which is called the **bin size** (MacKay and McCulloch, 1952; Rieke et al., 1997). For a small enough  $\Delta t$  (usually of the order of few  $ms$ ) the spike-train can be represented as a binary string  $\mathbf{x}$  of length  $T/\Delta t$  composed of 0's (when there was no spike during the bin) and 1's if there was (see Fig. 1)<sup>1</sup>. In the following we will assume that the spike-train is “binned” with a given bin size.

Information theory deals only with probabilistic entities and the analysis of spike-trains is justified by the understanding that during time period  $[0, T]$  many different sequences could have been realized. Thus the fact that only one particular sequence took place out of all possible sequences provides the stochastic context for the analysis within the probability framework.

We wish to give a probability measure to describe the process of spiking as a sequential process. This calls for the definition of a stochastic process. Basically a **stochastic process** is a sequence of random variables  $\{X_i\}_{i=1}^{\infty}$ . For the formal definition of a **stochastic process** and **information source** see Appendix on page 12.

**Example:** The simplest case of a stochastic process is an **i.i.d. process**, where all the variables  $\{X_i\}$  are independent and identically distributed.

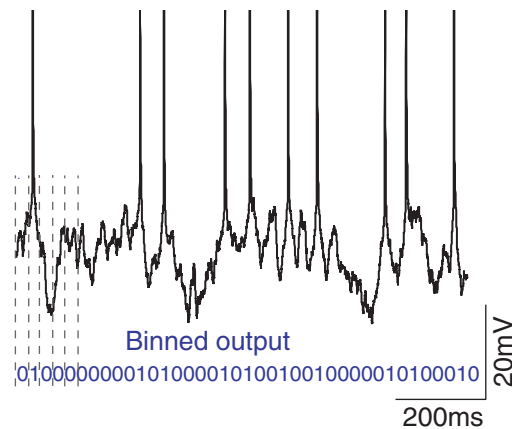


Figure 1: **Binning:** A 1s trace of membrane voltage is recorded from a layer V pyramidal neuron. Time is discretized into bins of  $\Delta t = 25ms$  (dashed vertical lines). Each bin has a value of 1 if a spike occurred within its time duration, otherwise its value is 0. The resulting binary string (of length  $T/\Delta t$ ) represents the spike-train and is used for entropy estimation.

<sup>1</sup>If the bin size is long such that two or more spikes may occur during a duration of one bin, then we would need to use more symbols to represent the event in each bin. This is referred as a larger alphabet. Note also that information-rate is affected by the bin size, see also Rieke et al. (1997).

Assume that the sample spike-train is produced by such a source. This means that at each bin, there is a fixed probability  $p$  to obtain a spike, and the probability of no spike in this bin is  $1 - p$ . This will be the case if the source is a Poisson Process, with a fixed rate, and a small  $\Delta t$ .

## 1.2 Definitions of Entropy and Entropy rate

Our estimation of mutual information relies on an estimation of entropy. We will describe entropy estimation first and then in section 3 we will use it to estimate mutual information.

**Definition 1.1** Let  $X$  be a discrete random variable taking the values  $a \in \mathcal{X}$  with probabilities  $p(a) = Pr\{X = a\}$ , then the **entropy**  $H(X)$  of  $X$  is defined as

$$H(X) = - \sum_{a \in \mathcal{X}} p(a) \log_2 p(a), \quad (1)$$

and is measured in *bits*. This definition can easily be generalized to  $n$ -tuples of random variables  $(X_1, X_2, \dots, X_n)$  by considering the joint probability distribution of the  $n$ -tuples (for more details see (Cover and Thomas, 1991)).

**Example:** If  $X$  is a binary variable that has the distribution

$$X = \begin{cases} 0 & : \text{ with probability } 1 - p, \\ 1 & : \text{ with probability } p. \end{cases} \quad (2)$$

then  $H(X) = -[p \cdot \log(p) + (1 - p) \cdot \log(1 - p)]$ . This entropy, which is a function of  $p$  alone, is sometimes denoted as  $H(p)$ . Note that for  $p = 1/2$  (i.e. a distribution of a fair coin)  $H(1/2) = 1$  bit.

**Definition 1.2 ((Cover and Thomas, 1991))** The **entropy rate** of a stationary and ergodic stochastic process<sup>2</sup>  $\mathbf{X} = \{X_i\}_{i=1}^{\infty}$  is defined by

$$H(\mathbf{X}) = \lim_{n \rightarrow \infty} \frac{1}{n} H(X_1, \dots, X_n) \quad (3)$$

## 2 Estimation of the entropy of a spike-train

In general there are three classes of methods to estimate entropy, a direct methods that follows the definition of entropy and the other two based on properties of sequences that are usually used by sophisticated compression algorithms.

1. The direct method: Here, one counts the frequency of the occurrence of words in a sequence to approximate the distribution of words in the source. This empirical distribution is then used in the expression of entropy rate (see Definition 1.2). This method can be greatly improved to account for biases in the estimators and for under-sampling problems (Treves and Panzeri, 1995; Strong et al., 1997; Rieke et al., 1997; Panzeri and Schultz, 2001; Schultz and Panzeri, 2001).

---

<sup>2</sup>For complete definitions of stationary and ergodicity see Feller (1961). These properties of the process ensure that the limit in the definition exists.

2. The second class of methods is based on compression algorithms by Ziv and Lempel (1977, 1978). These algorithms rely on theorems that state that a simple expression of the recurrence of words in a sequence, converges to the entropy rate of the source that produces the sequence. With a few modifications this expression can be used as an entropy rate estimator (Wyner, 1993).
3. The third class relies on another convergence theorem, by *Shannon, MacMillan & Brieman* (Cover and Thomas, 1991). Roughly speaking this theorem states that a simple expression of the probability of a source to produce a sequence converges to the entropy rate of the source with the length of the sequence. Thus, a good estimation of the probability of obtaining the sample sequence can be used to estimate the entropy rate. Some compression algorithms provide sophisticated probability estimation methods Willems et al. (1995); Cleary and Teahan (1997) that can be used for this purpose. The algorithm described in this document is an example of this class.

## 2.1 Estimation of the entropy of a spike-train: The direct method

We illustrate the direct way to estimate the entropy of a spike-train on the simplest case. Assume a sample spike-train  $\mathbf{x}$ , of length  $n$ , generated by an *i.i.d. process*  $\mathbf{X} = \{X_i\}_{i=1}^{\infty}$ . Although an i.i.d.-process is rarely relevant as a statistical model of a spike-train, it turns out that this is a very useful example. Note that for each  $i$ ,  $X_i$  is a binary random variable with a probability  $p$  have the value 1 and is independent of all other variables  $\{X_j\}_{j \neq i}$ . Then by independence we obtain (see Appendix on page 13 below), that the entropy of the process is equal to the entropy of each of the  $X_i$ 's in the sequence, which is a function of  $p$  alone:

$$\begin{aligned}
 H(\mathbf{X}) &= \lim_{n \rightarrow \infty} \frac{1}{n} H(X_1, \dots, X_n), \\
 &= \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n H(X_i), \quad \text{independence} \\
 &= \lim_{n \rightarrow \infty} \frac{1}{n} n H(X_1), \quad \text{identically distributed} \\
 &= H(X_1) = H(p).
 \end{aligned} \tag{4}$$

Because we do not know  $p$ , we want to estimate the entropy of the process, given spike-train  $\mathbf{x}$ . One possibility is to count the 1's in spike-train  $\mathbf{x}$  (denoted as  $N_{\mathbf{x}}(1)$ ), and use the following estimation:

$$\begin{aligned}
 \hat{H}_n^{Direct}(\mathbf{X}) &= H(\hat{p}_{\mathbf{x}}), \\
 &= H(N_{\mathbf{x}}(1)/n).
 \end{aligned} \tag{5}$$

In other words, estimate  $p$  from sequence  $\mathbf{x}$ , using  $\hat{p}_{\mathbf{x}} = (N_{\mathbf{x}}(1)/n)$ , the proportion of 1's in the sequence, and then use the formula for the entropy of a binary variable with  $\hat{p}_{\mathbf{x}}$ . In fact the quality of the estimation of the entropy depends on  $p$ , the length of sequence  $n$ , and on the properties of the estimator  $\hat{p}$ .

There is, however, an alternative approximation, which might seem cumbersome, but will become useful afterward. Assume that  $\mathbf{x}$  is a sequence generated by the i.i.d process with a parameter  $p$ , then

the probability of the source to generate  $\mathbf{x}$  of length  $n$  is:

$$P(\mathbf{x}|p) = p^{N_{\mathbf{x}}(1)}(1-p)^{n-N_{\mathbf{x}}(1)} \quad (6)$$

It follows that:

$$\begin{aligned} -\frac{1}{n} \log P(\mathbf{x}|p) &= -\frac{1}{n} \{N_{\mathbf{x}}(1) \log p + (n - N_{\mathbf{x}}(1)) \log (1 - p)\} \\ &\xrightarrow{n \rightarrow \infty} H(p), \quad \text{with probability 1} \end{aligned} \quad (7)$$

using that  $N_{\mathbf{x}}(1)/n \xrightarrow{n \rightarrow \infty} p$  with probability 1.

Consequently an alternative way to estimate the entropy of the source is to use an estimator  $\hat{P}(\mathbf{x})$  of the probability of the source to generate  $\mathbf{x}$  of length  $n$ , and use the following definition:

**Definition 2.1**

$$\hat{H}_n(\mathbf{x}) \triangleq -\frac{1}{n} \log \hat{P}(\mathbf{x}) \quad (8)$$

## 2.2 Estimation of the entropy of a spike-train: a Bayesian framework

Assume still that sample spike-train  $\mathbf{x}$ , of length  $n$ , is generated by an *i.i.d. process*  $\mathbf{X}$  with parameter  $p$ . Note that in the previous section we neglected that fact that the same spike-train  $\mathbf{x}$  might be generated by many different *i.i.d. processes*, each having different parameters (different  $p$ ). Because the entropy of the process is a function of  $p$ , each of these processes has a different entropy. This means that our estimation of the entropy of the sequence is somewhat arbitrary (see also Discussion).

An alternative way to estimate the entropy of sequence  $\mathbf{x}$  is to compute the average over all possible sources that could yield the given spike-train  $\mathbf{x}$ . We thus can average over all the possible sources each weighted by its probability to generate  $\mathbf{x}$ , given in Eq. (6). This results in the following estimation:

$$\begin{aligned} \hat{H}^{Bayes}(\mathbf{X}) &= \int_0^1 H(\theta) P(\mathbf{x}|\theta) d\theta \\ &= \int_0^1 H(\theta) \theta^{N_{\mathbf{x}}(1)} (1-\theta)^{n-N_{\mathbf{x}}(1)} d\theta \end{aligned} \quad (9)$$

From a Bayesian viewpoint, computing this expression is desirable; however, taken literally, it might be impracticable. First Eq. (9) might not be computable efficiently. Second, generalization for non-i.i.d. processes might be hard due to difficulties in evaluating  $P(\mathbf{x})$  for such processes. We thus take a slightly different route to overcome these generalization and efficiency difficulties.

The first step is to generalize the use of Definition 2.1. Instead of using  $\hat{P}(\mathbf{x})$  (the estimate of the probability of  $\mathbf{x}$  to be generated by the source), we can compute an estimator  $\tilde{P}(\mathbf{x})$ , for the average of  $P(\mathbf{x})$  over all possible i.i.d.-sources and use it in Definition 2.1. For an i.i.d process we use the **KT-estimator** (see below) to estimate this average probability. Then in the following steps, we generalize this idea for non-i.i.d. processes by both a method to compute  $\tilde{P}(\mathbf{x})$  for such processes, as well as a generalization of Eq. (7).

**Definition 2.2** ((Krichevsky and Trofimov, 1981)) *The Krichevsky-Trofimov estimated probability for a sequence  $\mathbf{x} \in \{0, 1\}^n$  with  $N_{\mathbf{x}}(0) \geq 0$  zeros and  $N_{\mathbf{x}}(1) \geq 0$  ones is defined as*

$$P_e(\mathbf{x}) \triangleq \int_0^1 \frac{1}{\pi \sqrt{\theta(1-\theta)}} (1-\theta)^{N_{\mathbf{x}}(0)} \theta^{N_{\mathbf{x}}(1)} d\theta, \quad (10)$$

This is a weighted average of the probability of string  $\mathbf{x}$  over all possible i.i.d sources that can realize it<sup>3</sup>. Because we assume that  $\mathbf{x}$  is generated by an i.i.d.-process, the estimator is dependent only on the number of ones and zeros in the sequence, and is independent of their order of appearance. We denote the KT-estimator of the sequence  $\mathbf{x}$  as  $P_e(a_{\mathbf{x}}, b_{\mathbf{x}})$  where  $a_{\mathbf{x}} = N_{\mathbf{x}}(0)$  and  $b_{\mathbf{x}} = N_{\mathbf{x}}(1)$ .

The KT-probability estimator can be computed sequentially (i.e. instead of evaluating the integral, one can use the formula in Lemma 2.1) presented below.

**Lemma 2.1** (Sequential computation (Willems et al., 1995)) *For the empty string  $P_e(\emptyset) = 1$ , and for all  $\mathbf{x} \in \{0, 1\}^n$  and  $a \in \{0, 1\}$*

$$P_e(\mathbf{x} \circ a) = \frac{N_{\mathbf{x}}(a) + \frac{1}{2}}{n + 1} \cdot P_e(\mathbf{x}). \quad (11)$$

where  $\mathbf{x} \circ a$  means concatenating the symbol  $a$  to the right side of the string  $\mathbf{x}$ .

**Example** Let  $\mathbf{x}$  be binary string 0010 of length 4, composed of 1 ones and 3 zeros. Then  $P_e(\mathbf{x}) = 5/128$ . If a 1 is now concatenated to  $\mathbf{x}$ , then we can update  $P_e(\mathbf{x} \circ 1) = (1\frac{1}{2}/5) \cdot (5/128) = 3/256$

Exchanging  $\hat{P}(x)$  in Definition 2.1, with  $P_e(\mathbf{x})$  we obtain the estimate for all possible i.i.d-process that might generate the sequence  $\mathbf{x}$ :

$$\hat{H}_n^{i.i.d}(\mathbf{x}) \triangleq -\frac{1}{n} \log P_e(\mathbf{x}) \quad (12)$$

### 2.3 Markov Processes

In the previous sections we analyzed the estimation of entropy of a spike-train under the assumption that it was generated by an i.i.d.-process with an unknown parameter. As stated above, this is not a very reasonable assumption. Neurons are assumed to have several kinds of short term memory, induced by physical properties such as the membrane time constant, synaptic time constants etc. It is desirable to extend the method to allow dependency between the bins in the spike-train. The most natural and simple generalization of an i.i.d.-process is a **Markov Process**. This generalization is formed by permitting any random variable  $X_{i+1}$  in the sequence to depend on the variable directly preceding it  $X_i$ . Given this preceding variable,  $X_{i+1}$  is conditionally independent of all other variables in the sequence.

**Definition 2.3** *A Markov process over a binary alphabet  $\{0, 1\}$  is a stochastic process, such that:*

$$P\{(X_1, \dots, X_n) = (x_1, \dots, x_n)\} = P(X_1 = x_1) \prod_{i=2}^n P(X_i = x_i | X_{i-1} = x_{i-1}), \quad (13)$$

for all  $n \geq 1$  and  $(x_1, \dots, x_n) \in \{0, 1\}^n$ .

---

<sup>3</sup>Note that  $P(\mathbf{x}|\theta)$  is weighed using the prior  $\frac{1}{\pi \sqrt{\theta(1-\theta)}}$ . It is beyond the scope of this appendix to justify this prior but it is possible to show that it is a **natural prior** for this case. More details can be found in Willems et al. (1995).

In the following we will use the short notation  $P(x_1, \dots, x_n)$  for  $P\{(X_1, \dots, X_n) = (x_1, \dots, x_n)\}$ .

**Higher order Markov processes** In the case of spike-trains we are interested in processes where the distribution of every random variable depends not only on the variable directly preceding it, but rather on the first  $D$  variables preceding it.

**Definition 2.4** A stochastic process over a binary alphabet  $\{0, 1\}$ , is said to have the **Markov property** if there exists a natural positive number  $D$  such that:

$$P(x_1, \dots, x_n) = P(x_1, \dots, x_D) \prod_{i=D+1}^n P(x_i | x_{i-1}, \dots, x_{i-D}), \quad (14)$$

for all  $n \geq D$  and  $(x_1, \dots, x_n) \in \{0, 1\}^n$ . The minimal number  $D$  such that Eq. (14) holds, is called the **order** of the process.

A process with the Markov property and order  $D$  will be termed a  **$D$ -order Markov process**. The class of all such processes are termed **finite order Markov processes**. In the following sections we consider only Markov processes that are time invariant (i.e. the conditional probability  $P(x_i | x_{i-1}, \dots, x_{i-D})$  does not depend on  $i$ ) and are ergodic (For complete definitions of these properties see Feller (1961)). We assume that sequence  $\mathbf{x}$  is generated by a source from this class, and want to extend our method to estimate the entropy of the source.

The following theorem extends Eq. (7) to non-i.i.d processes, and allows us to continue using Definition 2.1 by developing a method to estimate  $\tilde{P}(\mathbf{x})$  (i.e. the average probability to obtain  $\mathbf{x}$  over all possible sources) in the case of a Markov process as well.

**Theorem 2.1 (Shannon, MacMillan & Brieman (Cover and Thomas, 1991))** For a stationary and ergodic process  $\{X_i\}_{i=1}^{\infty}$

$$-\frac{1}{n} \log P \{(X_1, X_2, \dots, X_n) = (x_1, x_2, \dots, x_n)\} \xrightarrow{n \rightarrow \infty} H, \quad \text{with probability 1} \quad (15)$$

where  $H$  is the **entropy rate** of the process.

## 2.4 Representation of a Markov Source as Suffix Trees

A  **$D$ -order Markov process** over a binary alphabet can be represented as a **Suffix Tree Source**. A Suffix Tree over a binary alphabet, is a binary tree (see Fig. 2), where each node is either a leaf, or has two daughter nodes. Any left edge in the tree is labeled “0” and each right edge “1” (or vice versa). Such a tree is also called a **Model**. Each path from the root of the tree to one of the leaves defines a binary string  $s$  which will be denoted the **suffix**. The collection of all suffixes defines the **Suffix Set**  $\mathcal{S}$ , and is an equivalent representation of the Suffix Tree. To obtain a **Suffix Tree Source**, we assign to each leaf on the Suffix Tree a probability  $\theta_s$  (where  $s$  is the path from the root to this leaf). We will denote the set of all these parameters  $\Theta_{\mathcal{S}} = \{\theta_s : s \in \mathcal{S}\}$ .

This source is identical to a Markov process. Assuming the depth of the tree is  $D$ , and given the past  $D$  symbols produced by the source (the history of a string), the tree combined with parameters  $\theta_s$  determines the probability of the next upcoming symbol. Assume we have a binary

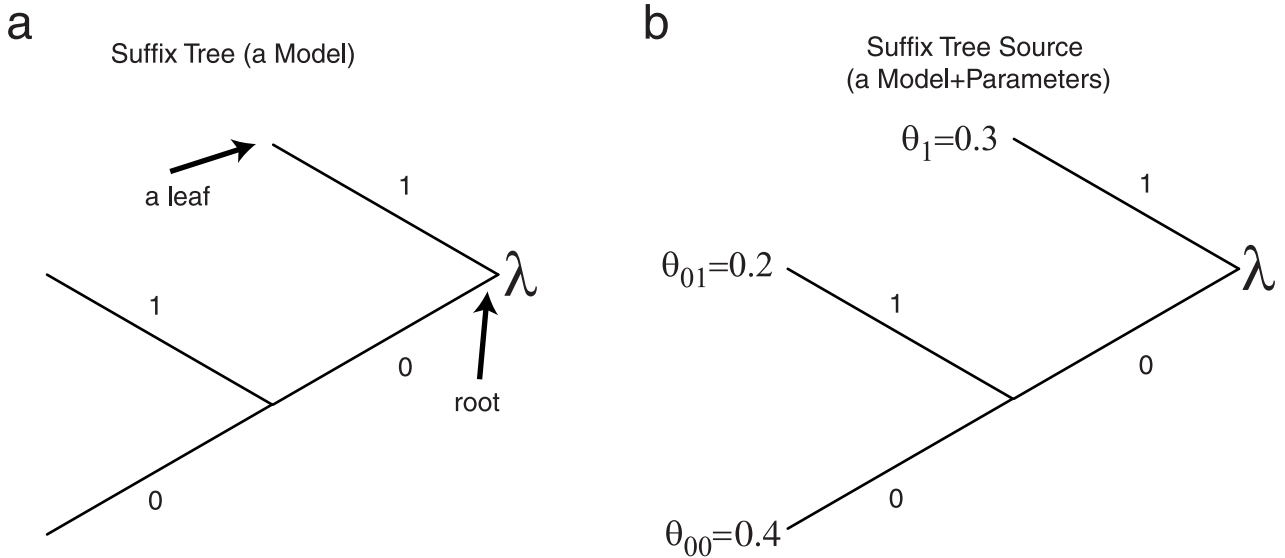


Figure 2: **An example of a Suffix Tree and a Suffix Tree Source.** (a) A binary tree construct a model provided that each node is either a leaf or a father of two nodes. Each edge is labeled as either "1" or "0". Each leaf on the tree is associated with a suffix  $s$  which is composed of the labels on the edges on the path from the root to the leaf. The model in (a) is equivalent to the Suffix Set  $\mathcal{S} = \{1, 01, 00\}$ . (b) Associating a probability  $\theta_s$  for each leaf  $s$  in the Suffix Tree results with a Suffix Tree Source. This means that if the previous symbols in the sequence were  $\dots, 10$  then the next symbol is generated with the probability  $\theta_{01} = 0.2$  (note that the sequence is read backward in order to find the appropriate leaf). Because many combinations of probabilities are possible, many sources can share the same model.

string  $\{x_{-\infty}, \dots, x_{-1}, x_0\}$ , based on this string and the Suffix Tree Source we obtain the probability of  $P(X_1 = 1)$  in the following way: we traverse the tree from the root, according to  $\{\dots, x_{-1}, x_0\}$ , starting with an edge labeled with the value  $x_0$ , then the edge labeled with the value  $x_{-1}$  and so on, until we get to a leaf. The parameter  $\theta_s$  in this leaf gives us the desired probability. The next symbol is now produced with this probability, and we can now look for at the new suffix  $\{\dots, x_{-1}, x_0, x_1\}$  in order to obtain the probability for  $x_2$ , and so on. It is easy to see that each Markov process of order  $\leq D$  can be represented by such a Suffix Tree Source with depth  $D$ .

Suppose now that we have a spike-train  $\mathbf{x}$  that is generated by such a Suffix Tree Source, and suppose further that we know this source (i.e. we know both the model and the parameters of this source). We can compute the probability  $P(\mathbf{x})$  simply by the following decomposition (see 14 above):

$$P(x_1, \dots, x_n | x_0, \dots, x_{1-D}) = \prod_{i=1}^n P(x_i | x_{i-1}, \dots, x_{i-D}) \quad (16)$$

where we obtain  $P(x_i | x_{i-1}, \dots, x_{i-D}) = \theta_{s_i}$  if  $x_i = 1$ , and  $(1 - \theta_{s_i})$  otherwise.  $\theta_{s_i}$  is the probability at the leaf when traversing the tree according to  $\{x_{i-1}, \dots, x_{i-D}\}$  (note that  $s_i$  might be a substring of  $\{x_{i-1}, \dots, x_{i-D}\}$ ). In other words, we are sliding a window of length  $D$  through the sequence  $\mathbf{x}$ , each time using this window to find a suffix on the suffix tree, in order to obtain the probability of the next symbol. Multiplying all these probabilities together yields the overall probability to obtain the whole sequence  $\mathbf{x}$  from the source represented by the suffix tree and the parameter set  $\Theta_{\mathcal{S}}$ .

However, in reality we do not know the model ( $\mathcal{S}$ ), or the parameter set ( $\Theta_{\mathcal{S}}$ ). Our goal is thus to

estimate  $P(\mathbf{x})$ , without this information. We present the solution in two steps. First we assume that we do know the model (i.e. the tree) but we do not know  $\theta_s$  for any of the leaves. Then we abandon this assumption and only assume that we know a bound on the order of the source.

## 2.5 Estimation of the entropy rate for a known model

Given a model  $\mathcal{S}$ , there are many sources that can be represented by it. There is a distinct source for every assignment of  $\Theta_{\mathcal{S}}$  to its leaves. The class of all stationary and ergodic sources that can be represented by  $\mathcal{S}$  is called the *model class* of  $\mathcal{S}$ . Let  $\mathbf{x}$  be a sequence produced by some source; assume we know the model of the source but we do not know the parameters. We wish to estimate the entropy of the source by generalizing the estimate for the i.i.d. case. In other words, we want an estimate that will consider all the possible sources in the model class of  $\mathcal{S}$ .

Let  $\mathcal{S}$  be a model, and let  $\mathbf{x}$  be a sequence generated by a source having  $\mathcal{S}$  a a model. We can rewrite Eq. (16) to obtain:

$$P(\mathbf{x}|x_0, \dots, x_{1-D}) = \prod_{i=1}^n P(x_i|s_i) \quad (17)$$

where  $s_i$  is the appropriate suffix in  $\mathcal{S}$  (a path on the tree from the root to one of the leaves) that precedes  $x_i$  in  $\mathbf{x}$ .

The key point in this section is to realize that given the suffix  $s \in \mathcal{S}$ , all the  $x_i$ 's in the sequence  $\mathbf{x}$  that immediately follow  $s$ , are drawn from the same probability distribution (binary distribution with a parameter  $\theta_s$ ). Moreover, let us denote

$$\mathbf{x}_s \triangleq \{x_i : 1 \leq i \leq n, s_i = s\},$$

i.e. the substring of  $\mathbf{x}$  that is composed of all the  $x_i$ 's in  $\mathbf{x}$  that immediately follow  $s$  (have  $s$  as a suffix). Using this notation we notice that

$$\begin{aligned} P(\mathbf{x}_s) &= \prod_{x_i \in \mathbf{x}_s} P(x_i|s) \\ &= (\theta_s)^{a_{\mathbf{x}_s}} (1 - \theta_s)^{b_{\mathbf{x}_s}} \end{aligned} \quad (18)$$

(recall that  $a_{\mathbf{x}_s}$  is the number of 1's in  $\mathbf{x}_s$ ), which suggests that the KT-estimator  $P_e(\mathbf{x}_s)$  can be used to estimate  $P(\mathbf{x}_s)$  over all possible  $\theta_s$ 's just as was done for the i.i.d process. It is easy to see from Eq. (17) that:

$$P(\mathbf{x}|x_0, \dots, x_{1-D}) = \prod_{s \in \mathcal{S}} P(\mathbf{x}_s) \quad (19)$$

which lets us estimate  $P(\mathbf{x})$  of the true source having the model  $\mathcal{S}$  by using  $P_e(\mathbf{x}_s)$  for all  $s \in \mathcal{S}$ :

$$\tilde{P}(\mathbf{x}|x_0, \dots, x_{1-D}) = \prod_{s \in \mathcal{S}} P_e(\mathbf{x}_s) \quad (20)$$

**Example** Let  $\mathbf{x} = 001101001$  be a binary string produced by a Suffix Tree Source of order 1 (i.e. by a tree with two leaves the first denoted as 0 with the associated parameter  $\theta_0$  and the second leaf 1 with  $\theta_1$ ). Assume also that  $x_0 = 1$ , (this is the relevant history of the sequence that we need to start parsing

it). We can now define two strings  $\mathbf{x}_0 = \{x_2, x_3, x_6, x_8, x_9\} = 01101$  which is composed of all the  $x_i$ 's following the suffix 0, and the complementary string  $\mathbf{x}_1 = \{x_1, x_4, x_5, x_7\} = 0100$ . Following Eq. (18) we use the *KT-estimator*, and compute  $P_e(\mathbf{x}_0)$  and  $P_e(\mathbf{x}_1)$  as estimators of  $P(\mathbf{x}_0)$  and  $P(\mathbf{x}_1)$ . Then we simply estimate  $\hat{P}(\mathbf{x}) = P_e(\mathbf{x}_0) \cdot P_e(\mathbf{x}_1) = 3/256 \cdot 5/128 = 15/32768$ . We can now substitute  $\tilde{P}(\mathbf{x})$  for  $\hat{P}(\mathbf{x})$  in Eq. (8) to obtain

$$\hat{H}_n(\mathbf{x}) = -\frac{1}{n} \log \tilde{P}(\mathbf{x}) = -\frac{1}{9} \log \frac{15}{32768} = 1.232 \text{ bits/symbol} \quad (21)$$

This is the estimate of the entropy of the string  $\mathbf{x}$  provided that we know the model (i.e. the structure of the tree of the source). The knowledge of the model is needed for parsing the string. Note the similarity to the case of an i.i.d.-process, where the approximation is actually an average over all sources in the *model class*. In other words we average over all the Tree Sources that can be represented on the given tree, and only their parameters  $\theta_s$  on the leaves are different. It is important to note as well that since  $P_e$  can be computed sequentially (Lemma 2.1), we can compute  $\tilde{P}(\mathbf{x})$  in one path through the string  $\mathbf{x}$ .

## 2.6 Entropy estimation for an unknown model: The Context Tree Weighting Method

The final step we need in order to obtain an approximation  $\tilde{P}(\mathbf{x})$  is to drop the assumption that the model  $\mathcal{S}$  is known, and average over all possible models having orders less than  $D$ . Note that the number of such models is exponential in  $D$ . However, the *CTW* algorithm (Willems et al., 1995), “magically” provides a method to compute the average over all these models via a relatively simple procedure. To do this we introduce the concept of a *context tree* and then the procedure of *weighting* which together are combined into the final method of the Context Tree Weighting algorithm.

Note first that all possible model with an order less than  $D$  is a sub-tree of the full tree of depth  $D$  (see Fig. 3). This full tree serves as a skeleton for the context tree. The basic difference between the Suffix Tree we considered previously and the Context Tree is that in the Context Tree each node is a possible suffix, and not only the leaves. Suppose we scan the string  $\mathbf{x}$  until the  $i^{\text{th}}$  position, and we now have the (history) sequence  $\{x_{i-1}, \dots, x_{i-D}\}$ ; then each of the following might be a suffix in the true underlying unknown model:  $\{x_{i-1}\}, \{x_{i-1}, x_{i-2}\}, \dots, \{x_{i-1}, \dots, x_{i-D}\}$ . Note that each of these suffixes are on the same path in the Context Tree, and all but the last are

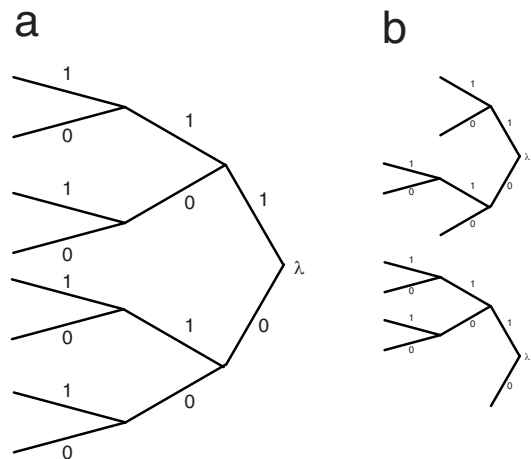


Figure 3: **An example of a Context Tree.** (a) A context tree of order 3 is the full binary tree of depth 3. Each internal node or leaf in the tree is associated with a context  $s$  which is the path from the root to this node. (b) All Suffix Trees with order  $\leq 3$  are sub-models of the context tree. Two such examples are shown. The Context Tree Weighting method averages over all such possible models, and for each model  $\mathcal{S}$  over all sources that in the model class of  $\mathcal{S}$  (i.e. all sources that share  $\mathcal{S}$  as their suffix tree, but have different parameter set  $\Theta_{\mathcal{S}}$ ).

represented by an internal node. Because we do not know for sure which of these strings  $s$  is a true suffix (i.e. a suffix in the true model), we term them “contexts” (hence the name Context Tree). We want to assign an estimator of probability associated with each context  $s$ , which we denote as  $P_w^s(\mathbf{x})$ . Note that  $P_w^s(\mathbf{x})$  is **not** an estimator of a parameter associated with the context, but rather a generalization of the KT-estimator  $P_e(\mathbf{x}_s)$  for the case that we are not sure that  $s$  is indeed a suffix in the true source.

Each internal node  $s$  on the Context Tree is either a leaf or an internal node on the true model tree. If it is a leaf we should treat it as before. We should count the zeros following this context in the string  $\mathbf{x}$  ( $a_{\mathbf{x}_s}$ ), and the ones following it ( $b_{\mathbf{x}_s}$ ). We can then use the KT-estimator to estimate  $P(\mathbf{x}_s)$ , just as we did before.

What if this node is not a leaf in the true model? Let’s suppose that it is a father of two leaves in the unknown true model. Following the example computed at the end of the previous section, it is clear that in this case, the probability assigned to this node should be the product of the probabilities assigned to the two leaves - its two daughter branches. Actually this is true not only for those nodes which are father of two leaves, but to any internal node. The two daughter branches of a node  $s$  are 1s and 0s. Thus in the case of an internal node in the true model we should assign the probability:  $P_w^{0s}(\mathbf{x}) \cdot P_w^{1s}(\mathbf{x})$

Because we cannot differentiate between the two possibilities (a leaf or an internal node) we take the average between them. This leads us to the following recursive fundamental formula:

$$\begin{aligned} P_w^s(\mathbf{x}) &\triangleq P_e(\mathbf{x}_s) \quad \text{if } \text{length}(s) = D, \\ P_w^s(\mathbf{x}) &\triangleq \frac{1}{2}P_e(\mathbf{x}_s) + \frac{1}{2}P_w^{0s}(\mathbf{x}) \cdot P_w^{1s}(\mathbf{x}) \quad \text{if } \text{length}(s) < D. \end{aligned} \quad (22)$$

We can thus construct a Context Tree, and apply Eq. (22) to evaluate  $P_w^s(\mathbf{x})$  for each node and leaf  $s$  on this tree. Eventually, after computing all these probabilities, at the root of the tree, we obtain  $P_w^\lambda(\mathbf{x})$ . This is the estimated probability that we will use in Eq. (8) to obtain the approximated entropy  $\hat{H}(\mathbf{x})$ .

The main advantage of the process of weighting through the context tree is contained in the following equation (Willems et al., 1995), which states that the  $P_w^\lambda(\mathbf{x})$  is an average probability, where this average is computed over all possible sub-models with orders less than  $D$  and for each model over all possible sources (i.e. possible assignment of parameters  $\theta_s$  to the model).

$$P_w(\mathbf{x}) = \sum_{S \in \mathcal{C}_D} 2^{-\Gamma_D(S)} P_e^S(\mathbf{x}), \quad (23)$$

where  $\mathcal{C}_D$  is the class of all tree models with a finite order  $D$ , and  $\Gamma_D(S)$  is the number of bits needed to encode the model.

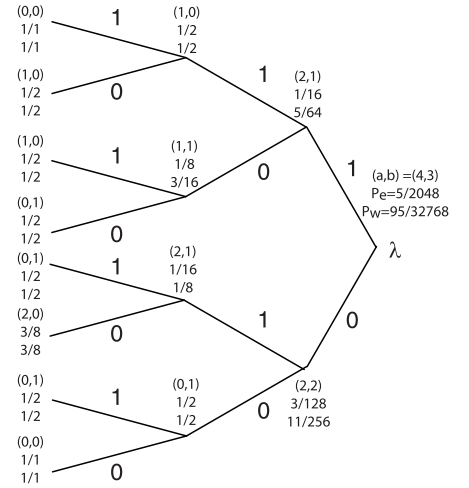


Figure 4: **An example of a Weighted Context Tree.** A context tree of order 3 for the sequence  $\mathbf{x} = 0110100$  with the history  $\dots 010$ .

## 2.7 Summary of entropy estimation procedure and implementation notes

We can summarize the process of estimating the entropy rate of a spike train  $\mathbf{x}$  assuming depth parameter  $D$  in the following steps:

1. Construct a Context Tree of depth  $D$ .
2. For each node in the tree (a context  $s$ ), count the number  $a_s$  of zeros following it in  $\mathbf{x}$ , and the number  $b_s$  of ones following  $s$ .
3. For each node recursively compute  $P_w^s(a_{\mathbf{x}_s}, b_{\mathbf{x}_s})$  using Eq. (22).
4. Substitute  $P_w^\lambda(\mathbf{x})$  (the probability at the root of the tree) in Eq. (8) to obtain the estimated entropy.

A few points for implementation. First, in order to obtain a good estimate one should use infinite-precision arithmetic in the calculations of  $P_e$  and  $P_w$ . This can be done with several algorithms. In this study we used the LEDA (<http://www.mpi-sb.mpg.de/LEDA/>) library for this purpose. Second, importantly, at step 1 there is no need to construct a full Context Tree of depth  $D$  (which has  $2^D$  leaves). Simply add to the tree only contexts that were encountered in  $\mathbf{x}$ . This ensures that the memory used is linear in the length of the input string, while Eq. (23) ensure that the average is still correct. Lastly Lemma 2.1, makes it possible to compute everything on the fly in one path. This means that practically, one scans the spike-train  $\mathbf{x}$ , considering a bin at a time, and at each such step, updates the counts,  $P_e$ , and  $P_w$  for all the current contexts which are not longer than  $D$ . If a specific context is not currently present in the tree, it is added.

## 3 Estimation of Mutual Information

**Definition 3.1** (*Mutual Information for binary variables*) Let  $X$  and  $Y$  be binary random variables taking the values  $a \in \{0, 1\}$ , then the **mutual information**  $I(X; Y)$  of  $X$  and  $Y$  is defined as

$$I(X; Y) = H(X) - H(X|Y) = H(X) - \sum_{a \in \{0,1\}} P(Y = a) \sum_{b \in \{0,1\}} P(X = b|Y = a) \log_2 P(X = b|Y = a) \quad (24)$$

and is measured in *bits*. Using this definition we can also define the mutual information between information sources.

**Definition 3.2** ((Ash, 1967)) The **mutual information** between two stationary and ergodic stochastic processes  $\mathbf{X} = \{X_i\}_{i=1}^\infty$  and  $\mathbf{Y} = \{Y_i\}_{i=1}^\infty$  is defined by

$$I(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) - \lim_{n \rightarrow \infty} \frac{1}{n} H(X_n | X_{n-1}, \dots, X_1, Y_n, \dots, Y_1). \quad (25)$$

where  $H(\mathbf{X})$  is the entropy rate of  $\mathbf{X}$ .

Given two individual spike-trains we would like to estimate the *MI* using a procedure that is similar to what we used for entropy estimation. The basic idea is simple,  $H(\mathbf{X})$  is estimated as in

the previous section. For the term  $-\lim_{n \rightarrow \infty} \frac{1}{n} H(X_n | X_{n-1}, \dots, X_1, Y_n, \dots, Y_1)$  we use an extension of the Context Tree Weighting procedure to obtain  $P_w^\lambda(\mathbf{x}|\mathbf{y})$  and substitute it in Eq. (8) to estimate the conditional entropy.

Consider the case of a source that produces the next symbol  $x_i$  in  $\mathbf{x}$  with a probability that depends on the previous  $D$  symbols in  $\mathbf{x}$  but also on the previous  $D$  symbols in  $\mathbf{y}$ . In order to estimate the conditional entropy we use an ordinary context tree by combining the two sequences  $\{x_{i-1}, \dots, x_{i-D}\}$  and  $\{y_{i-1}, \dots, y_{i-D}\}$  into one interleaved sequence  $\{x_{i-1}, y_{i-1}, \dots, x_{i-D}, y_{i-D}\}$ . The process of estimation of the conditional probability  $P(\mathbf{x}|\mathbf{y})$  becomes almost identical to the one we used to estimate  $H(\mathbf{X})$ . The only differences are (i) Use depth of  $2 \cdot D$  (ii) Use the interleaved sequence of  $\mathbf{x}$  and  $\mathbf{y}$  (iii) Advance the index in steps of 2 such that the counts are taken only for the  $x_i$ 's.

## 4 Discussion

The subject of entropy estimation from sequences is not a trivial issue and has lately received attention from practical and theoretical perspectives. Certain issues are not usually discussed and deserve to be mentioned. In general the *entropy* of an individual sequence is not well defined, and the same goes for the *mutual information* ( $MI$ ) between a pair of sequences. The main reason for this ill-definition is that a given sequence is deterministic and final, while entropy and  $MI$  are quantities of probabilistic nature. A finite sequence can be generated by many information sources, each having a different entropy rate. This means that the entropy of the source can only be estimated with certain degree of accuracy within a certain confidence limits. For example given a sequence, a strict lower bound on the entropy of the source cannot be established. The direct method of estimating the entropy rate neglects this issue and implicitly uses a maximum likelihood estimator. Roughly speaking, this means that it estimates the entropy of the source that is most likely to generate the sequence. If the spike-train is long enough the difference between the entropies of the possible sources that produced it becomes negligible, and the probability of error becomes small, leading to a reasonably good estimation.

However, the main problem that entropy estimation presents in practice is the problem of under-sampling (Treves and Panzeri, 1995; Strong et al., 1997; Panzeri and Schultz, 2001; Schultz and Panzeri, 2001). This problem becomes crucial when for Mutual Information estimation, because for any given context the counts given that context are usually very small. Using maximum-likelihood estimators for short sequences has a potential tendency to over-fit the data, and thus negatively affecting the robustness of the estimation. The motivation for the method presented here is that a Bayesian framework might lead to a better generalization ability. Moreover, the essence of a successful compression algorithm is the ability to generalize from short sequences. Comparison of these two approaches would require a separate study.

## 5 Appendix

**Stochastic process and Information source** A *stochastic process* (Cover and Thomas, 1991, p.60) over the binary alphabet  $\{0, 1\}$  is an indexed sequence of binary random variables  $\{X_i\}_{i=1}^\infty$ .

The stochastic process is characterized by a family of joint probability mass functions  $\Pr\{(X_1, X_2, \dots, X_n) = (x_1, x_2, \dots, x_n)\}$ ,  $(x_1, x_2, \dots, x_n) \in \{0, 1\}^n$  for all finite  $n = 1, 2, \dots$ .

In the case of spike-trains the random variable  $X_1$  represents the first bin  $[0, \Delta t)$  such that  $X_1 = 1$  if there was a spike during the bin and  $X_1 = 0$  if there was not. We treat this process as though it were stochastic with a given probability for each of these two possible events.

An equivalent representation is that of an information source. An *information source* over a binary alphabet is defined by a family of joint probability mass functions  $P^n(\mathbf{x})$  for all  $n = 0, 1, \dots$ . Each distribution  $P^n$  is defined on the set  $\{0, 1\}^n$  of all strings  $\mathbf{x} = x_1, \dots, x_n$  of length  $n$ . An information source must satisfy the marginality condition

$$P^n(\mathbf{x}) = \sum_{a \in \{0, 1\}} P^{n+1}(\mathbf{x} \circ a),$$

for every  $n$  and  $\mathbf{x}$ . Here  $\mathbf{x} \circ a = x_1, x_2, \dots, x_n, a$  is the concatenation of  $\mathbf{x}$  and  $a$  and  $P^0(\emptyset) = 1$  where  $\emptyset$  is the empty string. Every stochastic process defines a unique information source and vice versa.

A stochastic process is said to be *stationary* if the joint distribution of any subset of the sequence of random variables is invariant with respect to shifts in the time index, i.e.,

$$\Pr\{(X_1, X_2, \dots, X_n) = \mathbf{x}\} = \Pr\{(X_{l+1}, X_{l+2}, \dots, X_{l+n}) = \mathbf{x}\}, \quad (26)$$

for every  $l$  and for all  $\mathbf{x} \in \{0, 1\}^n$ .

**conditional entropy** The *conditional entropy* of  $X_n$  given  $X_{n-1}, \dots, X_1$  is defined by

$$H(X_n | X_{n-1}, \dots, X_1) = - \sum_{s \in \mathcal{X}^{n-1}} P(s) \sum_{a \in \mathcal{X}} P(a|s) \log P(a|s). \quad (27)$$

**Lemma 5.1 (Chain rules (Cover and Thomas, 1991))**

$$H(X_1, X_2) = H(X_1) + H(X_2 | X_1) \quad (28)$$

Note that if  $X_2$  is independent of  $X_1$ , then  $H(X_2 | X_1) = H(X_2)$  and thus for independent variables we obtain  $H(X_1, X_2) = H(X_1) + H(X_2)$

## References

- Ash, R. B. (1967). *Information Theory*. Interscience tracks in pure and applied mathematics. John Wiley & Sons, New York, NY, USA.
- Cleary, J. G. and Teahan, W. J. (1997). Unbounded length contexts for PPM. *Computer Journal*, 40(2/3):67–74.
- Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA.
- Feller, W. (1961). *An introduction to probability theory and its applications*. John Wiley, New York.
- Krichevsky, R. and Trofimov, V. (1981). The performance of universal encoding.
- London, M., Shribman, A., Häusser, M., Larkum, M., and Segev, I. (2001). Synaptic information efficacy: Bridging the cleft between biophysics and function. *Submitted to Nat. Neuro.*
- MacKay, D. and McCulloch, W. S. (1952). The limiting information capacity of a neuronal link. *Bull. Math. Biophys.*
- Panzeri, S. and Schultz, S. (2001). A unified approach to the study of temporal, correlational, and rate coding. *Neural Comput.*, 13(6):1311–1349.
- Rieke, F., Warland, D., de Ruyter van Steveninck, R., and Bialek, W. (1997). *Spikes: Exploring the neural code*. MIT Press, Cambridge, MA USA.
- Schultz, S. R. and Panzeri, S. (2001). Temporal correlations and neural spike train entropy. *Phys. Rev. Lett.*, 86(25):5823–5826.
- Strong, P. S., Koberle, R., de Ruyter, R. R., and Bialek, W. (1997). Entropy and information in neuronal spike trains. *Phys. Rev. Lett.*, 80:197.
- Treves, A. and Panzeri, S. (1995). The upward bias in measures of information derived from limited data samples. *Neural Computation*, 7(2):399–407.
- Willems, F. M. J., Shtarkov, Y. M., and Tjalkens, T. (1995). The context-tree weighting method: basic properties. *IEEE Trans. Info. Theory*, pages 653–664.
- Wyner, A. J. (1993). String matching theorems and applications to data compression and statistics.
- Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343.
- Ziv, J. and Lempel, A. (1978). Compression of individual sequences by variable rate coding.