Multi–Directional Context Sets with Applications to Universal Denoising and Compression

Erik Ordentlich*, Marcelo J. Weinberger*, and Tsachy Weissman†[1],
*Hewlett-Packard Laboratories, Palo Alto, CA 94304, U.S.A., (eord,marcelo)@hpl.hp.com
†Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA, tsachy@stanford.edu

***Abstract*— The classical framework of context–tree models used in sequential decision problems such as compression and prediction is generalized to a setting in which the observations are multi–tracked or multi–directional, and for which it may be beneficial to consider contexts comprised of possibly differing numbers of symbols from each track or direction. Context set definitions, tree representations, and pruning algorithms are all extended from the classical uni–directional setting to the m–directional setting, with an emphasis on the case of $m = 2$. We provide a simple example suggesting that determining (pruning) the best m–directional context set for $m \geq 3$ is substantially more complex than in the case of $m = 2$. After briefly describing how the multi–directional framework can be applied to universal data compression, we focus on its application to universal denoising, where we pair the proposed framework with a new technique for estimating the loss of a denoising algorithm based only on noisy observations.**

I. Introduction

We first review the classical (uni–directional) context modeling framework [1], [2] used in data compression and other sequential decision problems. Let $\mathcal{X}$ be the data sequence alphabet and let $\mathcal{S} \subset \mathcal{X}^*$ denote a finite set of finite length strings of symbols comprising a context set, where $\mathcal{X}^*$ is the set of all finite length strings (including the empty string) over the alphabet $\mathcal{X}$. Let k be the length of the longest string in $\mathcal{S}$. For $\mathbf{s} \in \mathcal{S}$ define $\mathcal{P}(\mathbf{s}) = \{\mathbf{x} \in \mathcal{X}^k : \mathbf{x}^{|\mathbf{s}|} = \mathbf{s}\}$, where $\mathbf{x}^l$ is the l symbol prefix of $\mathbf{x}$ and $|\mathbf{s}|$ denotes the length of $\mathbf{s}$. A valid context set $\mathcal{S}$ must satisfy the following two properties: (1) $\cup_{\mathbf{s}\in\mathcal{S}}\mathcal{P}(\mathbf{s}) = \mathcal{X}^k$ (exhaustive) and (2) $\mathcal{P}(\mathbf{s}) \cap \mathcal{P}(\mathbf{s}') = \emptyset$ for any pair $\mathbf{s} \neq \mathbf{s}'$ (disjoint). Given a data sequence $\mathbf{x}^n = (x_1, x_2, \ldots, x_n)$, the subsequence of symbols associated with a context $\mathbf{s}$ in a well defined context set $\mathcal{S}$ consists of those symbols x_i whose preceding symbols satisfy $\mathbf{s} = \mathbf{x}_{i-|s|}^{i-1}$, where $\mathbf{x}_l^m = (x_l, x_{l+1}, \ldots, x_m)$. For each $\mathbf{s} \in \mathcal{S}$ let $\mathbf{x}(\mathbf{s})$ denote the subsequence of data symbols associated in this manner with $\mathbf{s}$. The "exhaustive" and "disjoint" properties guarantee that each x_i belongs to one and only one such subsequence. Any given $\mathbf{x}(\mathbf{s})$ may be empty, however. It is well known that the "exhaustive" and "disjoint" properties also imply that the set of strings in $\mathcal{S}$ can be represented as the leaves of a (context) tree having nodes $\mathbf{n} \in \mathcal{X}^*$ where each node $\mathbf{n}$ is either a leaf or has the $|\mathcal{X}|$ children $\{\mathbf{n}x : x \in \mathcal{X}\}$. Context tree models are extensively studied in [2].

In most applications of context models, the processing of a context–induced subsequence of data symbols results in a numerical loss, such as the ideal code length of a probability assignment procedure in compression, or the error rate in prediction. To each subsequence we associate a *weight*, given by the loss incurred by processing the subsequence. Assume that an application induces a weight function λ on sequences of symbols over $\mathcal{X}$.[2] Given λ, an individual data sequence $\mathbf{x}^n$, and a maximum context length, of interest is that valid context set $\mathcal{S} \subset \mathcal{X}^{k*}$ that minimizes $\sum_{\mathbf{s}\in\mathcal{S}} \lambda(\mathbf{x}(\mathbf{s}))$, where $\mathcal{X}^{k*}$ is the set of strings over $\mathcal{X}$ of length at most k. An efficient dynamic programming based algorithm that relies on the context tree representation of valid context sets is known for carrying out this computation [3]. Determining the best context set, in the above sense, is useful in training context based predictors and in universal compression schemes based on two part codes (see e.g. [4]).

In this work, we generalize the above classical context modeling framework to a setting in which the observations are multi–tracked, multi–sided, or multi–directional, and for which it may be beneficial to consider contexts comprised of possibly differing numbers of symbols from each track or direction. We describe applications of this framework to universal data compression, briefy, and to universal denoising, in greater depth. In the latter application we pair the framework with a new technique for estimating the loss of a denoising algorithm based only on noisy observations.

II. Bi–directional context sets: Definition and structure

For simplicity, we present our framework assuming two directions, a "left" and a "right." A discussion of the m–directional case for $m > 2$ appears at the end of this section. Formally, our setting involves a data sequence $\mathbf{x} = \{x_i : i \in \mathcal{I}\}$ where, for each i, left and right directional sequences $\mathbf{y}_\ell^{(i)}$ and $\mathbf{y}_r^{(i)}$ are available for forming contexts. In one example of such a setting, $\mathcal{I} = \{1, 2, 3, \ldots\}$ and $\mathbf{x}$ consists of two tracks so that $x_i = (x_{L,i}, x_{R,i})$ and the left and right directional sequences correspond to $\mathbf{y}_\ell^{(i)} = x_{L,i-1}, x_{L,i-2}, \ldots$ and $\mathbf{y}_r^{(i)} = x_{R,i-1}, x_{R,i-2}, \ldots$. In a digital image compression or processing setting, $\mathcal{I} = \{1, 2, 3, \ldots\} \times \{1, 2, 3, \ldots\}$, $x_{i,j}$ represents the pixel value in row i and column j of the image, and $\mathbf{y}_\ell^{(i,j)}$ may be set to $x_{i,j-1}, x_{i,j-2}, \ldots$ (the sequence

[1] This author is also with Hewlett–Packard Laboratories, Palo Alto, CA 94304, U.S.A.

[2] The weight function is often obtained by accumulation of instantaneous losses corresponding to the symbols in the sequence.

of pixel values appearing to the left in row i) while $\mathbf{y}_r^{(i,j)}$ may be set to $x_{i-1,j}, x_{i-2,j}, \ldots$ (the sequence of pixel values appearing above in column j). The denoising setting described in more detail in Section IV involves $\mathcal{I} = \{1, 2, 3, \ldots\}$ with $\mathbf{y}_\ell^{(i)} = x_{i-1}, x_{i-2}, \ldots$ and $\mathbf{y}_r^{(i)} = x_{i+1}, x_{i+2}, \ldots$.

Paralleling the classical case, a bi–directional context set $\mathcal{S} \subseteq \mathcal{X}^* \times \mathcal{X}^*$ is a finite set of ordered pairs of finite length strings over the observation alphabet specifying the bi–directional contexts. Let k be the length of the longest string in any pair in $\mathcal{S}$. For $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}$ define $\mathcal{P}(\mathbf{s}_\ell, \mathbf{s}_r) = \{(\mathbf{x}^k, \mathbf{y}^k) \in \mathcal{X}^k \times \mathcal{X}^k : \mathbf{x}^{|\mathbf{s}_\ell|} = \mathbf{s}_\ell, \mathbf{y}^{|\mathbf{s}_r|} = \mathbf{s}_r\}$, the pairs of strings of length k whose first and second components respectively have $\mathbf{s}_\ell$ and $\mathbf{s}_r$ as prefixes. For a bi–directional context set to be well defined or valid, the set $\mathcal{S}$ must satisfy the following generalizations of the "exhaustive" and "disjoint" conditions from the uni–directional case: (1) $\cup_{(\mathbf{s}_\ell, \mathbf{s}_r) \in S} \mathcal{P}(\mathbf{s}_\ell, \mathbf{s}_r) = \mathcal{X}^k \times \mathcal{X}^k$ (exhaustive) and (2) $\mathcal{P}(\mathbf{s}_\ell, \mathbf{s}_r) \cap \mathcal{P}(\mathbf{s}'_\ell, \mathbf{s}'_r) = \emptyset$ for any pair $(\mathbf{s}_\ell, \mathbf{s}_r) \neq (\mathbf{s}'_\ell, \mathbf{s}'_r)$ (disjoint).

Given a data sequence $\mathbf{x}$, the subset of symbols associated with a context pair $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}$ consists of those symbols x_i whose corresponding context formation sequences satisfy $\mathbf{s}_\ell = [\mathbf{y}_\ell^{(i)}]^{|\mathbf{s}_\ell|}$ and $\mathbf{s}_r = [\mathbf{y}_r^{(i)}]^{|\mathbf{s}_r|}$. As in the classical case, the new "exhaustive" and "disjoint" properties guarantee that each x_i belongs to one and only one such subset. For each $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}$, let $\mathbf{x}(\mathbf{s}_\ell, \mathbf{s}_r)$ denote the subset of data symbols associated in the above manner with $(\mathbf{s}_\ell, \mathbf{s}_r)$.

The new "exhaustive" and "disjoint" properties also lead to a tree based representation of a valid bi–directional context set $\mathcal{S}$. In the bi–directional case, the tree, which is defined in the next lemma and which we shall refer to as a bi–directional context tree, has a somewhat different structure from the uni-directional case, and is not necessarily unique for a given $\mathcal{S}$.

Lemma 2.1: The string pairs in a valid bi–directional context set $\mathcal{S}$ can be represented as the leaves of a rooted tree (bi–directional context tree) having nodes in $\mathcal{X}^* \times \mathcal{X}^*$ where the root node is the pair of empty strings $(\emptyset, \emptyset)$ and each node $n = (\mathbf{s}_\ell, \mathbf{s}_r)$ is either a leaf or the set of its children is either $\{(\mathbf{s}_\ell, \mathbf{s}_r x) : x \in \mathcal{X}\}$ or $\{(\mathbf{s}_\ell x, \mathbf{s}_r) : x \in \mathcal{X}\}$.

We note for future reference the easily seen fact that, conversely, the leaves of any bi–directional context tree, as defined in Lemma 2.1, determine a valid bi–directional context set. Additionally, the structure of a bi–directional context tree can be inferred from its nodes. In the sequel a bi–directional context tree will be represented using the set of its nodes.

Proof of Lemma 2.1: Our proof is by induction on $m(\mathcal{S}) = \max_{(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}} |\mathbf{s}_\ell| + |\mathbf{s}_r|$, the maximum sum of the lengths of the context pair components. Only $\mathcal{S} = (\emptyset, \emptyset)$ satisfies the base case $m = 0$, and the lemma clearly holds for this case with a bi–directional context tree consisting only of the root–node/leaf $\{(\emptyset, \emptyset)\}$. Next, consider any valid $\mathcal{S}$ with $m(\mathcal{S}) = \tilde{m} \geq 1$ and assume, by induction, that the lemma holds for all valid $\mathcal{S}'$ with $m(\mathcal{S}') < \tilde{m}$. We show how it follows that the lemma also holds for $\mathcal{S}$.

First, note that $\mathcal{S}$ cannot contain $(\emptyset, \emptyset)$ since there must be at least one other pair $(\mathbf{s}_\ell, \mathbf{s}_r)$ in $\mathcal{S}$ and $\mathcal{P}(\emptyset, \emptyset) = \mathcal{X}^k \times \mathcal{X}^k$ could not be disjoint with $\mathcal{P}(\mathbf{s}_\ell, \mathbf{s}_r)$. Second, $\mathcal{S}$ cannot contain both a pair of the form $(\mathbf{s}_\ell, \emptyset)$ and $(\emptyset, \mathbf{s}_r)$ with $\mathbf{s}_\ell \neq \emptyset$ and $\mathbf{s}_r \neq \emptyset$ since then $\mathcal{P}(\mathbf{s}_\ell, \mathbf{s}_r)$ would be contained in both $\mathcal{P}(\mathbf{s}_\ell, \emptyset)$ and $\mathcal{P}(\emptyset, \mathbf{s}_r)$, again violating the "disjoint" condition in the definition of a valid $\mathcal{S}$. Thus, we are faced with two possibilities: Either all right contexts of each pair in $\mathcal{S}$ are non–empty strings, or all left contexts are non–empty strings. Assume, without loss of generality, that the former is the case. It is therefore possible to classify the pairs in $\mathcal{S}$ into $|\mathcal{X}|$ disjoint subsets of the form:

$$\mathcal{S}'_x = \{(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S} : \mathbf{s}_r = (x\mathbf{s}'), \mathbf{s}' \in \mathcal{X}^{k*}\},$$

for each $x \in \mathcal{X}$. The "exhaustive" property of $\mathcal{S}$ implies that $\mathcal{S}'_x$ is non–empty for each $x \in \mathcal{X}$. For each $\mathcal{S}'_x$ define

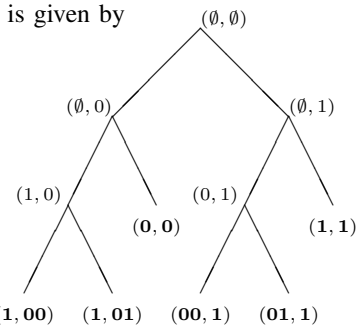
$$\mathcal{S}_x = \{(\mathbf{s}_\ell, \mathbf{s}_r) : (\mathbf{s}_\ell, x\mathbf{s}_r) \in \mathcal{S}'_x\}.$$

It follows that each $\mathcal{S}_x$ is non–empty (though may consist of the empty pair $(\emptyset, \emptyset)$). Additionally, since $\mathcal{S}$ is valid, it is not hard to see that each $\mathcal{S}_x$ must also be a valid context set. Since $m(\mathcal{S}_x) < m(\mathcal{S}) = \tilde{m}$, the induction hypothesis implies that associated with each $\mathcal{S}_x$ is a bi–directional context tree t_x. For each t_x, let t'_x denote the subtree rooted at $(\emptyset, x)$ obtained by changing each node $(\mathbf{s}_\ell, \mathbf{s}_r)$ of t_x to $(\mathbf{s}_\ell, x\mathbf{s}_r)$ while retaining the parent–child relationships of t_x.

Consider now the tree t consisting of a root node $(\emptyset, \emptyset)$ connected to the subtrees t'_x (i.e. the root–node's children consist of the root–nodes of t'_x, $x \in \mathcal{X}$). The properties of the subtrees t'_x then imply that t is a bi–directional context tree and that its leaves constitute $\cup_{x \in \mathcal{X}} \mathcal{S}'_x = \mathcal{S}$. Since $\tilde{m}$ and $\mathcal{S}$ were arbitrary, the lemma is proved by induction. □

The following is an example of a valid bi–directional context set and an associated bi–directional context tree, as guaranteed by Lemma 2.1. Note that the sets of strings formed from either the left components or the right components of the pairs in $\mathcal{S}$ fail to constitute valid uni–directional context sets.

Example 2.2: Let $\mathcal{X} = \{0, 1\}$ and $\mathcal{S} = \{(0, 0), (1, 00), (1, 01), (00, 1), (01, 1), (1, 1)\}$. An associated bi–directional context tree is given by



A bi–directional context tree can be interpreted as specifying, for a valid $\mathcal{S}$, a recursive sequence of splittings of the set $\mathcal{X}^k \times \mathcal{X}^k$ into the sets $\mathcal{P}(\mathbf{s}_\ell, \mathbf{s}_r)$ for $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}$. Lemma 2.1 thus shows that valid bi–directional context sets can be obtained as a sequence of splittings, where the three possible splittings are "not splitting" and splitting based on the value of the next left context string symbol or the value of the next right context string symbol.

The general multi–directional case. An m–directional context set consists of m–tuples of strings from $\mathcal{X}^*$. The notion of a well defined context set, expressed in terms of the "exhaustive" and "disjoint" properties from above, assuming the obvious generalization of $\mathcal{P}(\cdot)$, extends readily to $m > 2$. It seems natural to conjecture that Lemma 2.1 continues to hold with an m–directional context tree that is the obvious generalization of the $m = 2$ case defined in the lemma. Such a tree would have m–tuples of strings as nodes. Its root would be the null m–tuple and each node would be a leaf, or have as its children the m–tuples obtained by appending, in turn, all symbols in $\mathcal{X}$ to any single component of the m–tuple determining the node. Thus, for $m = 3$ and $\mathcal{X} = \{0, 1\}$, an internal node might be $(0, 00, 000)$ with possible child sets $\{(00, 00, 000), (01, 00, 000)\}$, $\{(0, 000, 000), (0, 001, 000)\}$, or $\{(0, 00, 0000), (0, 00, 0001)\}$.

The following simple example, however, establishes that for $m > 2$, any $\mathcal{X}$, and any k, the collection of well defined m–directional context sets is richer than what can be represented by a tree of the sort just described. Let $\mathcal{X} = \{0, 1\}$, $m = 3$, and consider the set of triples $\{(0, 0, \emptyset), (1, \emptyset, 0), (\emptyset, 1, 1), (1, 0, 1), (0, 1, 0)\}$. This set is a well defined 3–directional context set, as the "exhaustive" and "disjoint" conditions can be seen to hold. It turns out, however, that it cannot be represented as the leaves of a 3–directional context tree as described above. To see this, note that in such a tree the children of the root node are either $\{(\emptyset, \emptyset, 0), (\emptyset, \emptyset, 1)\}$, $\{(\emptyset, 0, \emptyset), (\emptyset, 1, \emptyset)\}$, or $\{(0, \emptyset, \emptyset), (1, \emptyset, \emptyset)\}$. Each case, however, splits at least one of the three subsets of string triples $\mathcal{P}(0, 0, \emptyset)$, $\mathcal{P}(1, \emptyset, 0)$, or $\mathcal{P}(\emptyset, 1, 1)$, respectively.

It is clear, therefore, that the above definition of an m–directional context tree for $m > 2$ must be augmented with a larger set of splittings at each node to allow for a full representation of all valid m–directional context sets. Moreover, it is desirable to find the smallest such set of splittings since the complexity of the corresponding pruning algorithm (as will be seen in the next section) is determined, to a large extent, by the size of the splitting set. The characterization of the minimal splitting set for any m and $\mathcal{X}$ is left for future work.

III. Bi–directional pruning

In [5], weighting and pruning algorithms are proposed for classes of context sets that are generated according to a variety of splitting rules. While the splitting rule relevant to bi–directional context sets described in Section II is not specifically considered in [5], it is straightforward to extend the algorithms in [5] for finding optimal context sets (and for weighting among these sets in compression applications) to this case. For completeness, we describe such an algorithm, based on dynamic programming, next. Given a sequence $\mathbf{x}^n$, a set of context formation sequences $\{(\mathbf{y}_\ell^{(i)}, \mathbf{y}_r^{(i)})\}$, a subsequence weight function λ, and a maximum context length k, let $L(\mathcal{S}) = \sum_{(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}} \lambda(\mathbf{x}(\mathbf{s}_\ell, \mathbf{s}_r))$. Of interest is $\mathcal{S}_{\text{opt}} = \arg\min_{\text{valid } \mathcal{S} \subseteq \mathcal{X}^k \times \mathcal{X}^k} L(\mathcal{S})$, where ties are broken according to a deterministic but arbitrary rule.

For any pair $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{X}^{k*} \times \mathcal{X}^{k*}$ we define the weight of $(\mathbf{s}_\ell, \mathbf{s}_r)$ as $w(\mathbf{s}_\ell, \mathbf{s}_r) = \lambda(\mathbf{x}(\mathbf{s}_\ell, \mathbf{s}_r))$, or the weight of the subsequence of data symbols whose left and right context formation sequences have prefixes equal to $(\mathbf{s}_\ell, \mathbf{s}_r)$. We set $w(\mathbf{s}_\ell, \mathbf{s}_r) = 0$ if this subsequence is empty. It is not difficult to see that for a valid context set $\mathcal{S}$, $L(\mathcal{S})$ is equal to the sum of the weights of the elements of $\mathcal{S}$ and, correspondingly, of the leaves of a representative bi–directional context tree.

For any bi–directional context tree T, as defined in Lemma 2.1, let $w(T)$ denote the weight of T defined as the sum of the weights of the leaves. Let $\mathcal{T}_{\text{opt}}$ be the set of bi–directional context trees with nodes in $\mathcal{X}^{k*} \times \mathcal{X}^{k*}$ having minimal weight. In general, this set will have cardinality greater than one, even when $\mathcal{S}_{\text{opt}}$ is unique, due to the multiple representations of the context set. Lemma 2.1 and the above then imply that $\mathcal{S}_{\text{opt}}$ corresponds to the leaves of an element of $\mathcal{T}_{\text{opt}}$, thereby reducing the problem of determining $\mathcal{S}_{\text{opt}}$ to the problem of determining an element of $\mathcal{T}_{\text{opt}}$.

Given any pair $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{X}^{k*} \times \mathcal{X}^{k*}$, a bi–directional context subtree rooted at $(\mathbf{s}_\ell, \mathbf{s}_r)$ has nodes in $\{(\mathbf{s}_\ell \mathbf{s}'_\ell, \mathbf{s}_r \mathbf{s}'_r) : \mathbf{s}'_\ell \in \mathcal{X}^{(k-|\mathbf{s}_\ell|)*}, \mathbf{s}'_r \in \mathcal{X}^{(k-|\mathbf{s}_r|)*}\}$, where, as in the definition of a full bi–directional context tree, a node $(\tilde{\mathbf{s}}_\ell, \tilde{\mathbf{s}}_r)$ is either a leaf or the set of its children is either $\{(\tilde{\mathbf{s}}_\ell, \tilde{\mathbf{s}}_r x) : x \in \mathcal{X}\}$ or $\{(\tilde{\mathbf{s}}_\ell x, \tilde{\mathbf{s}}_r) : x \in \mathcal{X}\}$. Let $\mathcal{T}(\mathbf{s}_\ell, \mathbf{s}_r)$ denote the set of bi–directional subtrees rooted at $(\mathbf{s}_\ell, \mathbf{s}_r)$. Extend the definition of the weight function $w(T)$ to bi–directional subtrees T in the obvious way and let $\mathcal{T}_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r)$ be the subset of bi–directional subtrees in $\mathcal{T}(\mathbf{s}_\ell, \mathbf{s}_r)$ having minimal weight. We then have the following principle of optimality.

Lemma 3.1: For any pair $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{X}^{k*} \times \mathcal{X}^{k*}$

$$\min_{T \in \mathcal{T}(\mathbf{s}_\ell, \mathbf{s}_r)} w(T) = \min \Bigg[w(\mathbf{s}_\ell, \mathbf{s}_r), \quad \sum_{x \in \mathcal{X}} \min_{T' \in \mathcal{T}(\mathbf{s}_\ell x, \mathbf{s}_r)} w(T'), \quad \sum_{x \in \mathcal{X}} \min_{T' \in \mathcal{T}(\mathbf{s}_\ell, \mathbf{s}_r x)} w(T') \Bigg], \quad (1)$$

where we take $\mathcal{T}(\tilde{\mathbf{s}}_\ell, \tilde{\mathbf{s}}_r)$ to be empty if $(\tilde{\mathbf{s}}_\ell, \tilde{\mathbf{s}}_r) \notin \mathcal{X}^{k*} \times \mathcal{X}^{k*}$ and the minimum of any function over an empty set to be infinity. If the minimum on the right hand side of (1) is achieved by the first term, then the tree $\{(\mathbf{s}_\ell, \mathbf{s}_r)\} \in \mathcal{T}_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r)$. Otherwise,

$$\{(\mathbf{s}_\ell, \mathbf{s}_r)\} \cup \bigcup_{x \in \mathcal{X}} T_x \quad \in \quad \mathcal{T}_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r)$$

where, if the minimum is achieved by the second term, T_x denotes any member of $\mathcal{T}_{\text{opt}}(\mathbf{s}_\ell x, \mathbf{s}_r)$, whereas if the minimum is achieved by the third term, T_x denotes any member of $\mathcal{T}_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r x)$.

Lemma 3.1 parallels a similar principle of optimality for the uni–directional case (where the minimum in (1) is over *two* values), and suggests the following dynamic programming algorithm for determining an element of $\mathcal{T}_{\text{opt}}$. We shall refer to Algorithm 3.2 as carrying out a bi–directional context pruning. Although the output of the algorithm is a bi–directional context

tree, it should be noticed that the data structure being pruned is *not* a tree.

Algorithm 3.2:

for each $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{X}^k \times \mathcal{X}^k$
  determine $w(\mathbf{s}_\ell, \mathbf{s}_r)$
  $T_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r) = \{(\mathbf{s}_\ell, \mathbf{s}_r)\}$
end
for $m = 2k-1$ to 0
  for each $(\mathbf{s}_\ell, \mathbf{s}_r)$ with $|\mathbf{s}_\ell| + |\mathbf{s}_r| = m$
    determine $w(\mathbf{s}_\ell, \mathbf{s}_r)$
    $N = w(\mathbf{s}_\ell, \mathbf{s}_r)$; $R = \sum_{x \in \mathcal{X}} w(T_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r x))$
    $L = \sum_{x \in \mathcal{X}} w(T_{\text{opt}}(\mathbf{s}_\ell x, \mathbf{s}_r))$
    $M = N$; $T_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r) = \{(\mathbf{s}_\ell, \mathbf{s}_r)\}$
    if $R < M$ **then**
      $M = R$; $T_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r) = \{(\mathbf{s}_\ell, \mathbf{s}_r)\} \cup \bigcup_{x \in \mathcal{X}} T_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r x)$
    if $L < M$ **then**
      $T_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r) = \{(\mathbf{s}_\ell, \mathbf{s}_r)\} \cup \bigcup_{x \in \mathcal{X}} T_{\text{opt}}(\mathbf{s}_\ell x, \mathbf{s}_r)$
  end
end

In the algorithm, $T_{\text{opt}}(\mathbf{s}_\ell, \mathbf{s}_r)$ is taken to be empty for $(\mathbf{s}_\ell, \mathbf{s}_r) \notin \mathcal{X}^{k*} \times \mathcal{X}^{k*}$ and the weight of an empty tree is taken to be infinity. The following theorem, which follows from Lemmas 2.1 and 3.1, establishes that Algorithm 3.2 carries out the desired computation.

Theorem 3.3: The tree $T_{\text{opt}}(\emptyset, \emptyset)$ generated by Algorithm 3.2 is an element of $\mathcal{T}_{\text{opt}}$ and its leaves constitute $\mathcal{S}_{\text{opt}}$.

In many applications, such as two-part codes with Krichevskii–Trofimov estimation [4] and the denoising application of Section IV, $w(\mathbf{s}_\ell, \mathbf{s}_r)$ is a relatively simple function of the vector of counts

$$\mathbf{m}(\mathbf{x}^n, \mathbf{s}_\ell, \mathbf{s}_r)[x] = \sum_{i: x_i \in \mathbf{x}(\mathbf{s}_\ell, \mathbf{s}_r)} 1(x_i = x), \qquad (2)$$

for all $x \in \mathcal{X}$. In these cases, the sequence $\mathbf{x}^n$ need only be processed to determine the counts of the contexts of maximal length, as done in the first **for** loop. The counts for the shorter contexts can then be determined as the sum of the counts of either the left children or right children. Specifically, for $(\mathbf{s}_\ell, \mathbf{s}_r)$ with $|\mathbf{s}_\ell| + |\mathbf{s}_r| < 2k$, $\mathbf{m}(\mathbf{x}^n, \mathbf{s}_\ell, \mathbf{s}_r) = \sum_{x' \in \mathcal{X}} \mathbf{m}(\mathbf{x}^n, \mathbf{s}_\ell, \mathbf{s}_r x') = \sum_{x' \in \mathcal{X}} \mathbf{m}(\mathbf{x}^n, \mathbf{s}_\ell x', \mathbf{s}_r)$. Finally, we note that the complexity of Algorithm 3.2 can be reduced by restricting processing only to those contexts that actually occur in the sequence.

IV. Application to denoising

The bi–directional context set framework described in Sections II and III can be applied to context–based denoising, in particular to enhance the DUDE algorithm proposed in [6]. In the (semi-stochastic) universal denoising setting, we consider an individual sequence $\mathbf{x}^n$, which is corrupted by a discrete memoryless channel with known transition probability matrix $\mathbf{\Pi}$. For simplicity, we will assume that the input and output alphabets coincide, so that $\mathbf{\Pi} = \{\Pi(x,z)\}_{x,z \in \mathcal{X}}$. It is also assumed that $\mathbf{\Pi}$ is invertible. A (noisy) sequence $\mathbf{z}^n$ is observed at the output of the channel, and the goal is to denoise $\mathbf{z}^n$ without knowledge of the (clean) sequence $\mathbf{x}^n$, to obtain a sequence $\hat{\mathbf{x}}^n \in \mathcal{X}^n$, where a given loss function $\Lambda : \mathcal{X}^2 \to [0, \infty)$, represented by the matrix $\mathbf{\Lambda} = \{\Lambda(x,z)\}_{x,z \in \mathcal{X}}$, determines the loss incurred by estimating each symbol x_i with the symbol $\hat{x}_i$, $1 \le i \le n$. The cumulative loss is determined by adding the instantaneous losses over time. The denoiser is allowed to observe the entire sequence $\mathbf{z}^n$ before starting to make its decisions. A family of denoisers, parameterized by a nonnegative integer parameter k, is proposed in [6]. For a given k, the denoiser output $\hat{x}_i^*(z_i)$ at time i for a noisy input z_i, $k+1 \le i \le n-k$, is given by

$$\hat{x}_i^*(z_i) = \arg\min_{\hat{x} \in \mathcal{X}} \mathbf{m}(\mathbf{z}^n, z_{i-k}^{i-1}, z_{i+1}^{i+k}) \mathbf{\Pi}^{-1} [\boldsymbol{\lambda}_{\hat{x}} \odot \boldsymbol{\pi}_{z_i}] \qquad (3)$$

where, similar to (2), $\mathbf{m}(\mathbf{z}^n, z_{i-k}^{i-1}, z_{i+1}^{i+k})$ is a $|\mathcal{X}|$-dimensional row vector whose β-th component, $\beta \in \mathcal{X}$, is the number of appearances of the string $z_{i-k}^{i-1} \beta z_{i+1}^{i+k}$ in $\mathbf{z}^n$, $\boldsymbol{\lambda}_a$ denotes the a-th column of $\mathbf{\Lambda}$, $\boldsymbol{\pi}_a$ denotes the a-th column of $\mathbf{\Pi}$, and for two vectors $\mathbf{u}$ and $\mathbf{v}$ with the same dimensions, $\mathbf{u} \odot \mathbf{v}$ denotes the vector obtained through componentwise multiplication. Thus, the decision made at time i by the denoiser of Eq. (3) upon observing a (noisy) sequence $\mathbf{z}^n$ is viewed as a function of the noisy symbol z_i to be corrected. This function depends on the occurrences of a *left context* z_{i-k}^{i-1}, denoted $\mathbf{s}_\ell^{(i)}$, and a *right context* z_{i+1}^{i+k}, denoted $\mathbf{s}_r^{(i)}$, in $\mathbf{z}^n$ (as well as on the parameters $\mathbf{\Pi}$ and $\mathbf{\Lambda}$ of the system).

It is shown in [6] that for every underlying sequence $\mathbf{x}^n$, the above denoiser is guaranteed to attain asymptotically, with probability one, the performance of the *best* k_n-th order sliding-window denoiser, tuned to $\mathbf{x}^n$ and to the observed noisy sequence $\mathbf{z}^n$, provided k_n grows sufficiently slowly with n. While these results provide asymptotic guidance on the choice of k for universal denoising, they refer to a sequence of problems, and shed little light on how k ought to be selected upon observation of a specific sequence $\mathbf{z}^n$. We would like to select the "best" value of k given $\mathbf{z}^n$, but our goal cannot be to determine the denoiser in the family that minimizes the loss, as this loss depends on the unobserved sequence $\mathbf{x}^n$. Instead, we propose in this paper to minimize an *estimate* of the actual loss, that depends on $\mathbf{z}^n$ only. This minimization will be performed over a family of denoisers larger than the one specified in Eq. (3). In particular, we will minimize the loss estimate over all bi–directional context sets of the type introduced in Section II (up to a preset maximal context length) for the decision rule of Eq. (3), where z_{i-k}^{i-1} and z_{i+1}^{i+k} are replaced by a left context $\mathbf{s}_\ell^{(i)}$ and a right context $\mathbf{s}_r^{(i)}$, respectively, which are determined by the context set and by corresponding left and right directional subsequences $\mathbf{y}_\ell^{(i)} = \{z_{i-1}, z_{i-2}, \ldots\}$ and $\mathbf{y}_r^{(i)} = \{z_{i+1}, z_{i+2}, \ldots\}$. The vector of counts $\mathbf{m}(\mathbf{z}^n, \mathbf{s}_\ell^{(i)}, \mathbf{s}_r^{(i)})$ is as defined in Eq. (2).

Our loss estimate is motivated by Theorem 4.1 below. Let $L(x_j^m, \mathbf{z}^n) = \sum_{i=j}^m \Lambda(x_i, \hat{x}_i(z_i))$ denote the cumulative loss incurred between (and including) locations j and m by a denoiser $\{\hat{x}_i(\cdot)\}$ (where the denoising functions $\hat{x}_i(\cdot)$ may depend on $\mathbf{z}^n$, as in the denoiser of Eq. (3)), upon observing $\mathbf{z}^n$, when the underlying clean sequence is x_j^m. Consider the

cumulative loss estimate

$$\hat{L}(\mathbf{z}^n, j, m) = \sum_{i=j}^{m} \sum_{x \in \mathcal{X}} \Pi^{-T}(x, z_i) \sum_{z \in \mathcal{X}} \Lambda(x, \hat{x}_i(z)) \Pi(x, z) \tag{4}$$

where $\hat{x}_i(z)$ denotes the output of the denoiser at time i when the symbol z_i is replaced by the symbol z in $\mathbf{z}^n$ (therefore, for a denoising function that depends on $\mathbf{z}^n$, the value of z affects the function itself, and not just its argument). Notice that the estimate of Eq. (4) depends on the observed sequence $\mathbf{z}^n$, but not on the unobserved sequence $\mathbf{x}^n$.

Theorem 4.1: For all $j > 0$, $m \geq j$, $n \geq m$, and $\mathbf{x}^n \in \mathcal{X}^n$, every denoiser satisfies $EL(x_j^m, \mathbf{Z}^n) = E\hat{L}(\mathbf{Z}^n, j, m)$.

The theorem is proved by noting in (4), that the expectation of $\Pi^{-T}(x, Z_i)$ is $\mathbf{1}(x = x_i)$, that $\sum_{z \in \mathcal{X}} \Lambda(x, \hat{x}_i(z)) \Pi(x, z)$ is $E(\Lambda(x, \hat{x}_i(Z_i)) | Z_1^{i-1}, Z_{i+1}^n)$, and that these two factors are independent. A result analogous to Theorem 4.1 was presented in [7, Lemma 4.1] for the causal case, in which the denoiser (filter) must make its decision at location i without access to z_{i+1}^n. The theorem states that the *observable* $\hat{L}(\mathbf{z}^n, j, m)$ is an *unbiased estimate* of $EL(x_j^m, \mathbf{z}^n)$. This property motivates the use of $\hat{L}(\mathbf{z}^n, k+1, n-k)$ as the cumulative loss to be minimized over all possible bi–directional context sets with context length upper-bounded by k, for the denoiser (3).

Now, to perform this minimization using the context pruning algorithm presented in Section III, we need to decompose the loss estimate given in Eq. (4) into a sum of contributions from each context pair $(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}$ (namely, the weights $w(\mathbf{s}_\ell, \mathbf{s}_r)$). Letting $\mathcal{I}_{(\mathbf{s}_\ell, \mathbf{s}_r)} = \{i : k+1 \leq i \leq n-k, (\mathbf{s}_\ell^{(i)}, \mathbf{s}_r^{(i)}) = (\mathbf{s}_\ell, \mathbf{s}_r)\}$, (4) becomes $\hat{L}(\mathbf{z}^n, k+1, n-k) = \sum_{(\mathbf{s}_\ell, \mathbf{s}_r) \in \mathcal{S}} w(\mathbf{s}_\ell, \mathbf{s}_r)$, where

$$w(\mathbf{s}_\ell, \mathbf{s}_r) = \sum_{i \in \mathcal{I}_{(\mathbf{s}_\ell, \mathbf{s}_r)}} \sum_{x \in \mathcal{X}} \Pi^{-T}(x, z_i) \sum_{z \in \mathcal{X}} \Pi(x, z) \Lambda(x, g^*_{z_1^{i-1} z z_{i+1}^n, (\mathbf{s}_\ell, \mathbf{s}_r)}(z)). \tag{5}$$

While the weights specified in (5) allow the use of the dynamic programming scheme of Section III to determine a minimizing set $\mathcal{S}_{\text{opt}}$, notice that they may depend on the value of symbols z_i such that $i \notin \mathcal{I}_{(\mathbf{s}_\ell, \mathbf{s}_r)}$. This dependency is due to the fact that the function $g^*_{z_1^{i-1} z z_{i+1}^n, (\mathbf{s}_\ell, \mathbf{s}_r)}(\cdot)$ depends on the vector $\mathbf{m}(z_1^{i-1} z z_{i+1}^n, \mathbf{s}_\ell, \mathbf{s}_r)$, whose components may be affected by appearances of the context pair $(\mathbf{s}_\ell, \mathbf{s}_r)$ in the sequence $z_1^{i-1} z z_{i+1}^n$ at locations other than those specified by $\mathcal{I}_{(\mathbf{s}_\ell, \mathbf{s}_r)}$. As discussed in Section III, such a dependency reduces the efficiency of the pruning algorithm's weight computation.

To overcome this problem, we will use an *approximate* set of weights. Notice that $\mathbf{m}(z_1^{i-1} z z_{i+1}^n, \mathbf{s}_\ell, \mathbf{s}_r)$ differs from $\mathbf{m}(\mathbf{z}^n, \mathbf{s}_\ell, \mathbf{s}_r)$ in two possible ways when $z \neq z_i$. First, replacing z_i with z increases the z-th component by 1 and decreases the z_i-th component by 1. Second, such a replacement may induce new appearances (and cancel actual appearances) of the context pair $(\mathbf{s}_\ell, \mathbf{s}_r)$ in the vicinity of location i. The first situation occurs whenever $z \neq z_i$, but it is not problematic as it depends only on the subsequence $\mathbf{z}(\mathbf{s}_\ell, \mathbf{s}_r)$. The second situation is the problematic one, but in practice it will rarely occur, as it requires that the context pair in question overlap with itself over significant portions. Thus, it will usually yield a second order contribution to the weight, and we will disregard it. This approximation yields a new set of weights,

$$\tilde{w}(\mathbf{s}_\ell, \mathbf{s}_r) = \sum_{\beta \in \mathcal{X}} \mathbf{m}(\mathbf{z}^n, \mathbf{s}_\ell, \mathbf{s}_r)[\beta] \sum_{x \in \mathcal{X}} \Pi^{-T}(x, \beta) \sum_{z \in \mathcal{X}} \Pi(x, z) \Lambda(x, g^{z \setminus \beta}_{\mathbf{z}^n, (\mathbf{s}_\ell, \mathbf{s}_r)}(z)) \tag{6}$$

where the denoising function $g^{z \setminus \beta}_{\mathbf{z}^n, (\mathbf{s}_\ell, \mathbf{s}_r)}(\cdot)$ is defined as $g^*_{\mathbf{z}^n, (\mathbf{s}_\ell, \mathbf{s}_r)}$, but with the vector $\mathbf{m}(\mathbf{z}^n, \mathbf{s}_\ell, \mathbf{s}_r)$ replaced with $\mathbf{m}(\mathbf{z}^n, \mathbf{s}_\ell, \mathbf{s}_r) + \mathbf{1}_{z \setminus \beta}$, $\mathbf{1}_{z \setminus \beta}$ denoting a vector with z-th component equal to 1, β-th component equal to -1, and whose all other components are 0, in case $\beta \neq z$, or the all-zero vector otherwise. Clearly, $\tilde{w}(\mathbf{s}_\ell, \mathbf{s}_r)$ depends on $\mathbf{z}^n$ only through $\mathbf{m}(\mathbf{z}^n, \mathbf{s}_\ell, \mathbf{s}_r)$, and can be efficiently employed for bi–directional context pruning.

The proposed algorithm has been applied to text denoising, improving over the number of errors after denoising reported in [6] by approximately 20%. In [6], a fixed context length of $k = 2$ was used. In addition, experiments performed over binary data (generated by a first order Markov source) corrupted by a binary symmetric channel show that the loss estimate $\hat{L}$ of Eq. (4) is indeed very close to the actual loss (with the difference being usually less than 1%) for any denoiser in the family defined by Eq. (3), yielding indeed the best choice of k.

We remark that a different approach for denoising with variable length bi–directional contexts, based on a stochastic modeling of the noisy data, has been proposed recently in [8].

ACKNOWLEDGMENT

We thank Giovanni Motta, Gadiel Seroussi, and Sergio Verdú for useful discussions.

REFERENCES

[1] J. Rissanen, "A universal data compression system," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 656–664, Sept. 1983.

[2] M. J. Weinberger, J. Rissanen, and M. Feder, "A universal finite memory source," *IEEE Trans. Inform. Theory*, vol. IT-41, pp. 643–652, May 1995.

[3] R. Nohre, *Some Topics in Descriptive Complexity*. PhD thesis, Department of Computer Science, The Technical University of Linkoping, Sweden, 1994.

[4] A. Martín, G. Seroussi, and M. J. Weinberger, "Linear time universal coding and time reversal of tree sources via FSM closure," *IEEE Trans. Inform. Theory*, vol. IT-50, pp. 1442–1468, July 2004.

[5] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "Context weighting for general finite–context sources," *IEEE Trans. Inform. Theory*, vol. IT-42, pp. 1514–1520, Sept. 1996.

[6] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdú, and M. Weinberger, "Universal discrete denoising: Known channel," *IEEE Trans. Inform. Theory*, vol. IT-51, pp. 5–28, Jan. 2005.

[7] E. Ordentlich, T. Weissman, M. J. Weinberger, A. Baruch-Somekh, and N. Merhav, "Discrete universal filtering through incremental parsing," *Proceedings of the 2004 Data Compression Conference* (DCC'04), pp. 352–361, Mar. 2004.

[8] J. Yu and S. Verdú, "Schemes for bi-directional modeling of discrete stationary sources," 2005 Conference on Information Sciences and Systems, The Johns Hopkins University, Baltimore, Md., Mar. 16, 2005.