Dealing with Small Data: On the Generalization of Context Trees

Ralf Eggeling EGGELING@INFORMATIK.UNI-HALLE.DE
Martin Luther University Halle-Wittenberg, Germany

Mikko Koivisto MIKKO.KOIVISTO@CS.HELSINKI.FI
Helsinki Institute for Information Technology, Department of Computer Science, University of Helsinki, Finland

Ivo Grosse GROSSE@INFORMATIK.UNI-HALLE.DE
Martin Luther University Halle-Wittenberg, Germany
German Center for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Leipzig, Germany

Abstract

Context trees (CT) are a widely used tool in machine learning for representing context-specific independences in conditional probability distributions. Parsimonious context trees (PCTs) are a recently proposed generalization of CTs that can enable statistically more efficient learning due to a higher structural flexibility, which is particularly useful for small-data settings. However, this comes at the cost of computationally expensive structure learning, which is feasible only for domains with small alphabets and tree depths. In this work, we investigate to which degree CTs can be generalized to increase statistical efficiency while still keeping the learning computationally feasible. Approaching this goal from two different angles, we (i) propose algorithmic improvements to the PCT learning algorithm, and (ii) study further generalizations of CTs, which are inspired by PCTs, but trade structural flexibility for computational efficiency. By empirical studies both on simulated and real-world data, we demonstrate that the synergy of combining of both orthogonal approaches yields a substantial breakthrough in obtaining statistically efficient and computationally feasible generalizations of CTs.

1. Introduction

Univariate conditional distributions play a central role in various multivariate probabilistic models such as Markov models, hidden Markov models, Bayesian networks, and general hierarchical graphical models. Ideally, each conditional distribution either involves only a few conditioning variables or we can assume the conditional distribution to take some simple form, e.g., a linear model. In practice, neither case may apply, and we encounter the curse of dimensionality: the representation size, i.e., the number of parameters, of the conditional distribution grows exponentially in the number of variables. Depending on the chosen statistical paradigm, such explosion easily leads to overfitting or otherwise poor use of the available data, which become scarce in relation to a multitude of parameters.

The concept of context-specific independence (Boutilier et al., 1996) provides an appealing approach to deal with the curse of dimensionality. Context-specific independence takes place when fixing some of the conditioning variables to certain states, called a *context*, the remaining variables provide no additional information about the response variable, that is, the response variable is independent of the rest given the context. Various structured models that enable learning a sufficient set of contexts from data have been investigated (Rissanen, 1983; Bryant, 1986; Quinlan, 1986).

Context trees (CTs) in particular enable computationally efficient learning of a sufficient set of contexts for categorical, linearly ordered random variables (Rissanen, 1983; Volf & Willems, 1994; Bühlmann & Wyner, 1999). CTs arise from organizing the full conditional probability distribution as a rooted tree, where each node at level ℓ of the tree is labeled by one of the possible states of the ℓth variable. If a node at level ℓ is a leaf of the tree, then the path from the root to the leaf corresponds to a context $x_1x_2\cdots x_\ell$ in the sense defined above: the conditional probability will be the same no matter what states are selected for the remaining variables. CTs can thus be learned efficiently by pruning subtrees that are not justified by observed data.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

Context trees have been extensively applied in variable-order Markov models where there is a natural ordering of the variables; see, e.g., the survey by Begleiter et al. (2004). But CTs appear also in more sophisticated models, such as permuted variable length Markov chains (Zhao et al., 2005) and variable-order Bayesian networks (Ben-Gal et al., 2005). The computational efficiency of CTs, however, comes at the cost of reduced statistical efficiency, stemming from the structural restrictions of CTs. Consider, for example, a case where fixing the first and the third conditioning variable to certain states x_1 and x_3 renders the response variable independent of all the remaining variables. To capture such context-specific independence, a CT has to include a distinct context $x_1x_2x_3$ per every possible state x_2 of the second variable, as "jumping" over variables is not allowed in CTs. In general, CTs may introduce an unnecessarily large number of contexts, which results in statistical inefficiency particularly when the data are scarce and the alphabet is not binary.

To address the shortcoming of CTs, Bourguignon & Robelin (2004) proposed *parsimonious context trees* (PCTs). PCTs generalize CTs by identifying a context with a selection of state *subsets* for the conditioning variables. Thus, in the above example, the multiple contexts $x_1x_2x_3$ with varying x_2 become represented by the single context $\{x_1\}S\{x_3\}$, where S stands for the state space of the second variable. See Figure 1 for a richer illustration of PCTs and the next section for a more formal definition. It has been observed that the increased flexibility of the model translates into improved prediction performance (Eggeling et al., 2013). PCTs have been applied in several domains, such as homogeneous parsimonious Markov models for modeling bacterial genomes (Bourguignon & Robelin, 2004), parsimonious higher-order Hidden Markov models for array-CGH analysis (Seifert et al., 2012), and inhomogeneous parsimonious Markov models for motif discovery in DNA sequences (Eggeling et al., 2014a).

Unfortunately, the existing algorithm for learning PCTs has an unfavorable computational complexity with respect to the alphabet size and the depth of the PCT (Bourguignon & Robelin, 2004). Hence, their applicability is currently limited to small-alphabet domains, with DNA sequence analysis being the most prominent instance. This is somewhat paradoxical, since generalizations of CTs that allow a higher structural flexibility are supposed to excel in statistical efficiency in particular when the data contain relatively long dependences over a large alphabet.

In this work, we aim at generalizing CTs for higher alphabets and depths in order to improve their statistical efficiency while keeping the computational effort in a practically feasible range. We approach this goal from two different angles. First, we investigate to which degree the existing algorithm for learning PCTs can by expedited by purely algorithmic improvements for storing and reusing intermediate results of computation. Second, we put forward alternative classes of generalized CTs that are inspired by and borrow features from PCTs, but trade statistical efficiency for computational efficiency with the goal of finding a good tradeoff between both extremes. We empirically show that significant speedups are obtained when the algorithmic improvements are applied to the new classes of generalized CTs and both the algorithmic and the modeling ideas thus act in concert. Furthermore, the speedups are particularly substantial when the data set is not only sparse but contains a *small* number of observations. We also study the prediction performance of the different types of generalized CTs, and find that, in general, the more structural flexibility, the better. In particular, the new classes of generalized CTS, which are less flexible but computationally more efficient than PCTs, can already yield a significant increase in statistical efficiency in relation to traditional CTs.

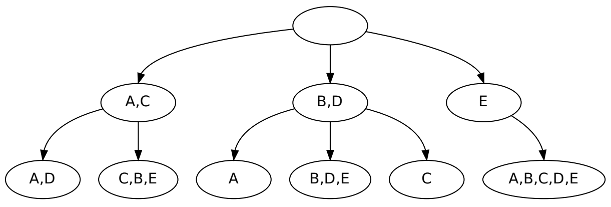

Figure 1. A parsimonious context tree (PCT) of depth 2 over a five-letter alphabet. The PCT encodes a partition of all 25 possible context words into the six contexts $\{\mathtt{AA}, \mathtt{AC}, \mathtt{DA}, \mathtt{DC}\}$, $\{\mathtt{CA}, \mathtt{CC}, \mathtt{BA}, \mathtt{BC}, \mathtt{EA}, \mathtt{EC}\}$, $\{\mathtt{AB}, \mathtt{AD}\}$, $\{\mathtt{BB}, \mathtt{BD}, \mathtt{DB}, \mathtt{DD}, \mathtt{EB}, \mathtt{ED}\}$, $\{\mathtt{CB}, \mathtt{CD}\}$, and $\{\mathtt{AE}, \mathtt{BE}, \mathtt{CE}, \mathtt{DE}, \mathtt{EE}\}$, each of which is assigned a distinct distribution of the response variable.

The next section briefly recaps the definition of PCTs and describes the dynamic programming (DP) algorithm of Bourguignon & Robelin (2004), to which we refer in the sequel as *basic DP*. In Section 3 we describe two new ideas for speeding up basic DP. In Section 4 we introduce alternative generalizations of CTs, which are inspired by PCTs but allow a higher computational efficiency. Section 5 provides a theoretical analysis of the expected computational gain of the proposed ideas. In Section 6 we evaluate the effect of the ideas on running time using both artificial and real data, and study the statistical efficiency of different CT generalizations for real data. Finally, we summarize the lessons learned by some concluding remarks (Section 7).

2. Parsimonious Context Trees and Basic Dynamic Programming

For simplicity of presentation, we will assume that the variables of interest take values from a common alphabet Σ of

σ symbols; generalization to the case where each variable has a dedicated state space is straightforward. A PCT of depth d over Σ is a rooted tree of depth d where each node is labeled by a non-empty subset of Σ, subject to the following constraint: for each inner node, the labels of the node's children form a set partition of Σ. Each node at level $\ell \leq d$ of the tree corresponds to a context $C \subseteq \Sigma^\ell$, namely a sequence of node labels $C_\ell \cdots C_1$ at levels $\ell, \dots, 1$ along the unique path from the node to the root (excluding the root). Note that we use a reversed order of indexing to write the "past" symbols on the left of the current symbol. We say that a word $x_d \cdots x_1$ and a context $C_\ell \cdots C_1$ *match* each other if $x_j \in C_j$ for all $j = 1, \dots, \ell$. A subtree of a PCT is *trivial* if all its nodes are labeled by Σ.

With each leaf of a PCT we associate a set of parameters θ_{Cx}, where x ranges over the symbols in Σ and C is the context that corresponds to the leaf. The parameter θ_{Cx} specifies the conditional probability that the symbol in a fixed position of the modeled sequence is x given that it is preceded by a context word in C. While several statistical models that make use of PCTs have been proposed (Bourguignon & Robelin, 2004; Seifert et al., 2012; Eggeling et al., 2013), we here assume without loss of generality that the data consist of a single long sequence. We thus consider a single conditional distribution and assume the availability of n *data samples*, each of which is a sequence of length $d+1$ over the alphabet Σ. The sufficient statistics for learning the parameters θ_{Cx} of a fixed PCT are simply the respective counts n_{Cx} defined as the number of data samples $x_d \cdots x_1 x_0$ in which $x_d \cdots x_1$ matches C and $x_0 = x$.

For measuring the goodness of fit of a PCT given the data, there is a large collection of possible scoring functions, which often assume the functional form of a penalized likelihood (Akaike, 1974; Schwarz, 1978; Silander et al., 2008). The only structural property we will need is that the score of a PCT factorizes into a product $\prod_C f(C)$, where C ranges over all leaves of the PCT and $f(C)$ is a *local* score that depends on the particular data subset of sequences that match C. Virtually all practically relevant scoring functions have this property.

The number of PCTs grows super-exponentially w.r.t. to depth and alphabet size. For example, there are 2.75×10^{19} PCTs for $\sigma = 4$ and $d = 3$ already. Hence, learning an optimal PCT by explicit enumeration of all possibilities is infeasible for all but the smallest instances.

Fortunately, an exhaustive search can be avoided by dynamic programming (Bourguignon & Robelin, 2004). The key is the following recurrence: Define $f_d(C) = f(C)$ and

$$f_\ell(C_\ell \cdots C_1) = \max_{S_1 \sqcup \cdots \sqcup S_q = \Sigma} \left\{ \prod_{i=1}^{q} f_{\ell+1}(S_i C_\ell \cdots C_1) \right\}$$

for all $\ell = 0, 1, \dots, d-1$ and subsets $C_\ell, \dots, C_1 \subseteq \Sigma$, where "$\sqcup$" stands for disjoint union. It holds that f_0 equals the optimal score over all PCTs of depth d. The basic dynamic programming algorithm exploits this recurrence using a data structure called *extended PCT*. Each inner node of an extended PCT has $2^\sigma - 1$ children labeled by the non-empty subsets of the alphabet Σ. Hence, the extended PCT contains all leaves and inner nodes that may exist in a valid PCT of the same depth and alphabet. The algorithm prunes the extended PCT in a bottom-up traversal to a valid PCT by keeping, at each inner node, a set of child nodes that form an optimal partition of the alphabet, and by discarding the remaining child nodes in the extended PCT.

The time complexity of the basic DP algorithm is determined by the work needed at level $d-1$ for optimal partitioning of the alphabet for all possible contexts $C_{d-1} \cdots C_1$, and is given by

$$\mathcal{O}\big((2^\sigma - 1)^{d-1} B_\sigma \big) , \qquad (1)$$

where the Bell number B_σ is the number of partitions of an σ-element set (Rota, 1964). For example, for the four-letter DNA alphabet, $B_\sigma = 2^\sigma - 1 = 15$, and thus the running time scales as 15^d. For larger alphabets, B_σ rapidly grows much larger than $2^\sigma - 1$, and the computation of the optimal partitions of child nodes becomes the limiting factor. For small alphabets, however, the running time of the algorithm may be dominated by the computation of the $(2^\sigma - 1)^d$ local scores, one for each leaf of the extended PCT.

3. Enhanced Dynamic Programming

There are two major obstacles that render basic DP infeasible for learning optimal PCTs of larger depth and alphabet. First, the number of alphabet partitions, B_σ, grows super-exponentially in σ. Second, the sheer number of possible contexts, $(2^\sigma - 1)^d$ grows exponentially in d. We address the first challenge in Section 3.1 by giving a substantially faster algorithm for finding an optimal set partition. We address the second challenge in Section 3.2 by making use of the observation that in an optimal PCT, the subtrees of two nodes are identical if the nodes correspond to the same data subset. It should be noted that both ideas do not alter the set of admissible tree structures, thus the enhanced DP algorithm yields the same output as the basic algorithm.

3.1. Fast Alphabet Partitioning

The key to faster finding of an optimal set partition is to exploit the factorization of the objective function. For a finite set U, define $g(U)$ as the maximum of the product $f(S_1) \cdots f(S_q)$ over all set partitions $\{S_1, \dots, S_q\}$ of U, with the convention that $g(\emptyset) = 1$. Then, for non-empty U,

$$g(U) = \max_{\emptyset \subset T \subseteq U} \left\{ f(T) g(U \setminus T) \right\} .$$

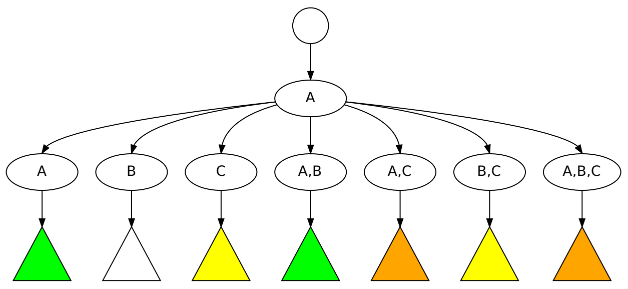

Figure 2. The memoization rule. Consider the shown part of the extended PCT and assume that the word $\ast$BA does not occur in any data word. Hence, the second layer node $\{\mathtt{A}\}$ represents the same data subset as its sibling node $\{\mathtt{A},\mathtt{B}\}$ and so the subtrees below both nodes are identical (green). The same applies for the subtrees $\{\mathtt{C}\}$ and $\{\mathtt{B},\mathtt{C}\}$ (yellow), and to the subtrees of $\{\mathtt{A},\mathtt{C}\}$ and $\{\mathtt{A},\mathtt{B},\mathtt{C}\}$ (orange).

This recurrence enables the computation of an optimal partition of alphabet Σ in each inner node in $|\{(T,U) : \emptyset \subset T \subseteq U \subseteq \Sigma\}| = 3^\sigma - 2^\sigma$ steps, yielding a significant improvement on the Bell number term for larger alphabets. The actual members of an optimal partition can be found by standard backtracking.

3.2. Pruning by Memoization

Consider two contexts $C_\ell \cdots C_1$ and $D_\ell \cdots D_1$ that correspond to two nodes of the extended PCT at level ℓ. Suppose the two contexts specify exactly the same data subset S. Then, in an optimal PCT containing the two nodes, the subtrees of the two nodes are identical, since the two fixed contexts do not impose any constraints on the possible subtrees. In terms of the associated scores, $f_\ell(C_\ell \cdots C_1) = f_\ell(D_\ell \cdots D_1)$. Hence, it suffices to compute the optimal score and the respective subtree only once and store them in an appropriate data structure. We use a hash map, with the data subset S, represented by the corresponding data point IDs, and the level ℓ as the key. Repeated computations are avoided by calling the values from the data structure. Figure 2 shows an example where the absence of one particular context word in the data leads to three applications of rule so that only 4 of 7 subtrees have to be explicitly computed.

The effectiveness of the memoization rule is data dependent. For small data sets but deep trees, the rule is expected to apply often, for then the number of contexts gets large while the number of distinct data subsets (matching long contexts) gets small. Likewise, the relative gain is expected be the higher, the larger the alphabet is. Also, the memoization rule is likely to apply more frequently on highly structured data than on random data.

The memoization rule relies on the assumption that the local score $f(C)$ depends on the context C only through the data subset S that matches the context, which is fulfilled for most scoring criteria (Akaike, 1974; Schwarz, 1978; Silander et al., 2008). A notable exception is the Bayesian marginal likelihood with context-dependent hyperparameters (Eggeling et al., 2013).

4. Alternative Context Tree Generalizations

While PCTs are a highly flexible generalization of CTs, it not obvious whether this high flexibility, which makes structure learning expensive, is actually necessary. For this reason, we propose alternative *generalized context trees* (GCTs) that borrow several, but not all features that are present in PCTs in comparison to CTs, with the aim of achieving a better tradeoff between statistical and computational efficiency than both CTs and PCTs permit to date.

We define a *k-generalized context tree* (k-GCT) as a PCT where every node labeled by more than k symbols has a trivial subtree. This is easily enforced by modifying the extended PCT accordingly into an *extended k-GCT*, as illustrated in Figure 3. This modification yields a guaranteed saving in the computations: at each level ℓ, the reduction in the number of nodes in the extended GCT is from $(2^\sigma - 1)^\ell$ to $\binom{\sigma}{k}_*^{\ell-1}(2^\sigma - 1)$, where $\binom{\sigma}{k}_* = \binom{\sigma}{1} + \binom{\sigma}{2} + \cdots + \binom{\sigma}{k}$. While k-GCTs sacrifice flexibility in favor of feasibility, they retain parsimonious features, such as multiple sibling nodes labeled with more than one symbol and non-trivial subtrees below such nodes.

A drawback of a k-GCT is that it cannot economically represent a "jump node" to indicate a non-informative context position. This is because no node labeled by the entire alphabet Σ is allowed to have a non-trivial subtree. We address this drawback by defining a *k^+-GCT* as a PCT where every node labeled with more than k, but less than σ, symbols has a trivial subtree. This modification only slightly expands the corresponding extended GCT. Yet, when we employ the memoization rule, k-GCTs and k^+-GCTs yield notably different computational gains (Section 6).

We note that the generalized CT variants do not directly enable computational savings in the alphabet partitioning problem. Namely, each of the $2^\sigma - 1$ possible subsets can be a member of an optimal partition, even if the other members are of size at most k. When $k = 2$ one could employ polynomial-time algorithms for finding a maximum-weight perfect matching (Edmonds, 1965), and thus reduce the asymptotic running time to within a polynomial factor of 2^σ; however, for $\sigma \leq 20$, our simpler algorithm is faster.

5. Theoretical Analysis

This section presents an analytic result concerning the savings due to memoization. For convenience, we consider

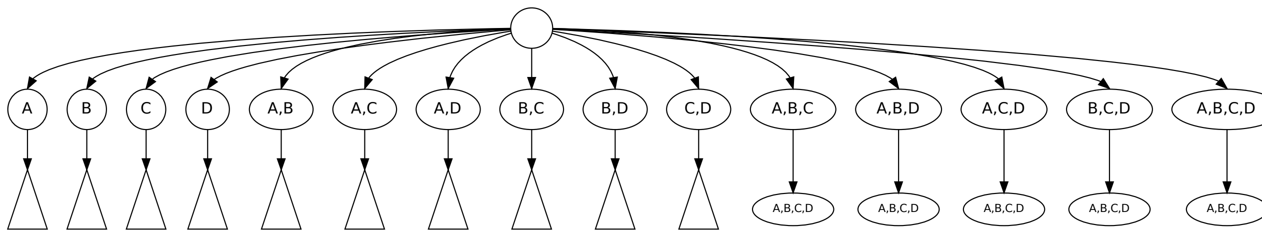

Figure 3. An extended 2-GCT over the alphabet $\{\texttt{A},\texttt{B},\texttt{C},\texttt{D}\}$, which is traversed and pruned within the DP algorithm in order to obtain a 2-GCT. Shown are the first two levels, whereas triangles indicate possibly deeper subtrees.

k-GCTs, which allow good control for the nodes in the extended k-GCT. Intuitively, we might expect the gain to be the larger, the smaller the bound k and the data size n are. Indeed, when we follow a path from the root to a leaf in the extended k-GCT, we may expect each node to reduce the associated data subset by a fraction of about σ/k. Consequently, at around depth $\log_{\sigma/k} n$, the size of the associated data subset is expected to be at most 1, at which point memoization allows us to avoid exploring the node's subtrees in all but n cases. In particular, with $k = \sigma/2$ the number of visited nodes of the extended k-GCT is expected to be about $(2^\sigma)^{\log_2 n} = n^\sigma$. Below we show that these intuitive expectations are valid, at least roughly and for random data; in Section 6 we observe the same for real-world data.

That said, we note that the above reasoning does not hold in the worst-case sense: it is possible to construct data sets where the number of distinct data subsets associated with the nodes in the extended k-GCT grows exponentially in n. We thus interpret our positive finding, the polynomial dependence on n, as a fortunate surprise, suggesting the worst-case view would be overly pessimistic in practice.

For our analysis we assume the n data words are independent draws from the uniform distribution on Σ^d. Our result concerns k-*contexts*, defined as contexts $C_\ell \cdots C_1$ with $|C_j| \leq k$ for all j. In particular, we consider k-contexts of fixed length ℓ, that is, nodes of the extended k-GCT at level ℓ. The following theorem bounds the expected number of distinct data subsets that are associated with the nodes (a proof is given in the Supplement). That bound corresponds to the number of nodes at level ℓ for which the memoization rule does not apply, and thus to the amount of work needed at level ℓ.

Theorem 1 *The expected number of data subsets matched by k-contexts of length ℓ is at most*

$$\binom{n}{r_k - 1}_* + \binom{n}{r_k}\left(1 + \left(\frac{k}{\sigma}\right)^\ell\right)^n,$$

where $r_k = \left\lceil \ln\binom{\sigma}{k} \big/ \ln\frac{\sigma}{k} \right\rceil \leq \left\lceil k\left(1 + \frac{1}{\ln\frac{\sigma}{k}}\right)\right\rceil$.

This result shows that for large enough ℓ, the work needed is, in essence, bounded from above by n^{r_k}. To be more precise, a length ℓ is large enough if $(k/\sigma)^\ell \leq 1/n$, equivalently $\ell \geq \ell_0 = \log_{\sigma/k} n$. For k-contexts that are shorter than ℓ_0, the theorem does not give a good bound. However, the plain number of such k-contexts is less than $\binom{\sigma}{k}_*^{\ell_0} \approx n^{r_k}$. For example, if $k = \sigma/2$, we obtain $r_k = \sigma$, and the bound in the theorem is at most n^σ for $\ell \geq \ell_0 = \log_2 n$, while the plain number of k-contexts of length ℓ_0 is, likewise, at most n^σ, agreeing with our intuitive expectation.

In summary, we have that, in expectation, the number of nodes in the k-GCT that are expanded for score evaluation scales as $O(n^{r_k})$, where $k < r_k \leq 2k$ for $k \leq \sigma/2$.. Because this does not yet include the nodes labeled by more than k symbols, the total number of visited nodes is bounded by $O(2^\sigma n^{r_k})$.

6. Case Studies

In this section, we empirically investigate the performance of the presented ideas. We have implemented the presented algorithms in Java based on the Jstacs framework (Grau et al., 2012) and conducted the experiments on a server with 2.4 GHz cores.

6.1. Running Time on Synthetic Data

Our first study concerns the running time of the enhanced DP algorithm compared to the basic DP algorithm for learning optimal k-GCTs, k^+-GCTs, and original PCTs. To enable comparison to the theoretical bounds, we varied the alphabet size σ and depth d, and for each (σ, d) we generated 100 sequences of length $100 + d$ independently uniformly at random, each sequence resulting in a set of $n = 100$ data words of length $d + 1$. For each configuration of the algorithm we measured its median running time (see Supplement), and we discuss selected key results in the following.

We observe that for learning original PCTs the two algorithmic ideas leading to the enhanced DP algorithm yield

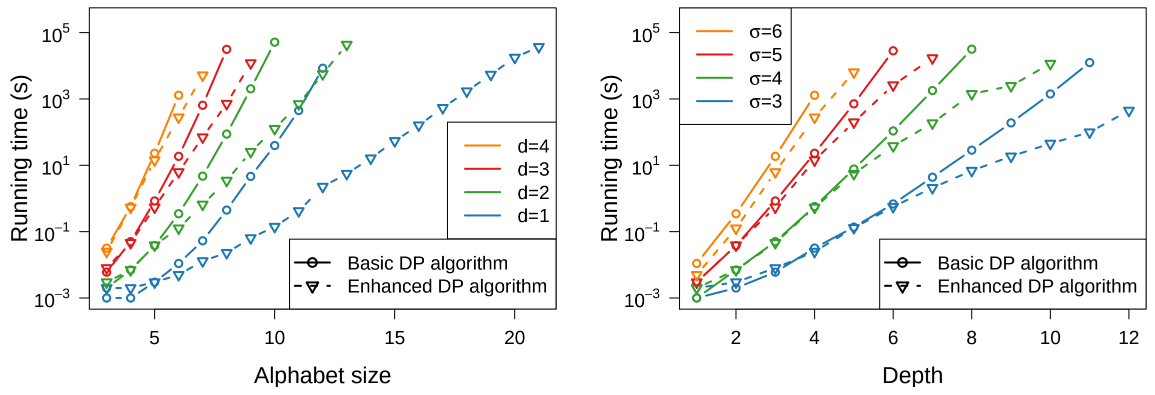

Figure 4. The time requirement of the basic and the enhanced DP algorithm for learning original PCTs for varying alphabet size σ and depth d from synthetic data sets of size 100.

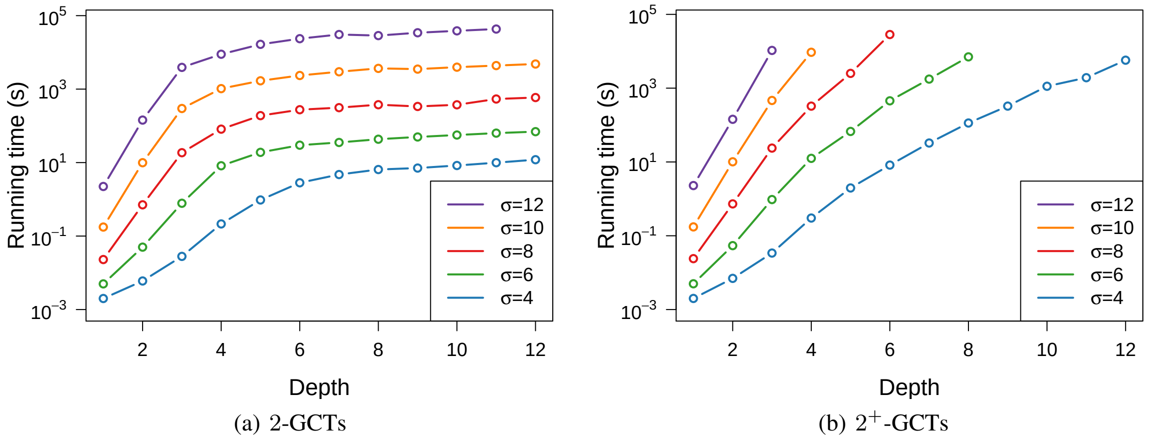

Figure 5. The time requirement of the enhanced DP algorithm for learning generalized CTs for varying alphabet size σ and depth d from synthetic data sets of size 100.

somewhat orthogonal advancements (Figure 4). For increasing alphabet sizes but shallow PCTs, the speedup is substantial due to the faster alphabet partitioning. Similarly, for increasing depths but small alphabets, the speedup is significant due to frequent memoization at the deeper levels in the extended PCT. However, when the depth exceeds 3 it is no longer feasible to have large enough alphabet size to benefit from fast alphabet partitioning. And vice versa, when the alphabet size exceed 5 it is no longer feasible to go sufficiently deep to benefit significantly from the memoization rule.

The picture changes when we consider k-GCTs and k^+-GCTs, which we here investigate for the case of $k = 2$ (Figure 5). For 2-GCTs we observe dramatic improvements in the running time, since now we can go deep enough to trigger off massive memoization. The observed gain is in good agreement with our analytic bounds: the running times grow very little when the depth exceeds $\log_{\sigma/2} 100$. For 2^+-GCTs the improvements are less dramatic, but still substantial in relation to the unrestricted PCTs. While the feature of allowing subtrees below completely fused nodes, thus skipping a position in the context, may be very useful for finding a sparse model for the problem at hand (Eggeling et al., 2013), it increases the complexity of the learning problem to a large extent.

6.2. Efficiency of Memoization on Real Data

While the study on synthetic data provides an insight in the general effect of the different techniques on running time, there remain some open questions that can be addressed on real data, only. We have seen on synthetic data that the potential for memoization is immense, but it needs to be investigated to which degree memoization applies on real data. To shed light on this question, we here investigate two

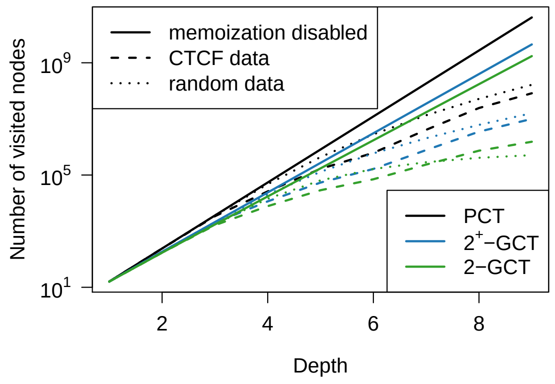

Figure 6. Effect of memoization on the CTCF data. Shown are the number of visited nodes in the extended PCT, 2-GCT, and 2^+-GCT of varying depth. For comparison, shown are also the number of nodes in the complete extended PCT, 2-GCT, and 2^+-GCT (i.e., memoization disabled), and the median number of visited nodes on the random data sets (cf. Section 6.1).

types of data: DNA binding sites and protein sequences. Rather than measuring the running times of the different configurations of the algorithm, we count the number of visited nodes in the extended GCTs to enable as direct comparison among different data sets as possible.

First, we study DNA binding sites, i.e. symbolic sequences over a four-letter alphabet, which is the most prominent application of PCTs to date. The focus is here on binding sites of the human insulator protein CTCF, for which strong statistical dependencies have been recently unveiled using PCTs (Eggeling et al., 2014a). We extract a data set of CTCF binding sites, which consists of 908 DNA sequences, from the Jaspar database (Sandelin et al., 2004), and plot the number of visited nodes for learning the different PCT variants under memoization (Figure 6). We observe that in almost all cases the memoization rule applies more frequently on the CTCF data than on synthetic data (cf. Section 6.1) The only exception are deep 2-GCTs ($d > 7$), where in the random data sets the data get "uniformly" scarce as soon as we get sufficiently deep. While this is not the case for the CTCF data, as the number of visited nodes steadily continues to increase, the savings due to the memoization rule are nevertheless immense: for $d = 9$ the number of visited nodes reduces by more than three orders of magnitude.

Since DNA binding sites are highly regular and limited to $\sigma = 4$, they may be not fully representative for other types of data. We thus additionally investigate the effect of memoization on protein sequences, which are typically described using the 20-letter amino acid alphabet. However, for many applications it is common to reduce this alphabet to smaller sizes (Li et al., 2003; Peterson et al., 2009; Bacardit et al., 2009). In this study we use the alphabet reduction method of Li et al. (2003), since it offers for each possible reduced alphabet size an optimal clustering of amino acids into groups, and study several well-known proteins of different size and functionality, extracted from protein sequence database UniProt (The UniProt Consortium, 2013).

We display the efficacy of the memoization rule in terms of visited nodes in the extended 2-GCT in Table 1 (for PCTs and 2^+-GCTs see Supplement). We observe that the percentage of visited nodes correlates with the length of the protein, which is in good agreement with our analytic observations. For alphabet size $\sigma = 11$ and depth $d = 5$, the number of nodes is reduced by one to two orders of magnitude. For smaller alphabet sizes and larger depths the savings gradually increase, in some cases up to a factor as large as 10^5. In order to demonstrate the data-management required for memoization has not a negative effect itself, we also measures the running times of all experiments (Supplement), and observe them to be in agreement with the number of visited nodes.

These results confirm that the memoization idea excels on real data, and particularly so in combination with generalized CTs that make compromises w.r.t. the structural flexibility allowed. This is due to the synergy effect discussed in Section 6.1. In addition, memoization may also be useful for the original PCT for small alphabets and highly regular data, as it is the case for the CTCF data.

6.3. Predictive Performance

While it has been previously demonstrated that PCTs can be superior to CTs in terms of statistical efficiency (Eggeling et al., 2013), it remains to be investigated whether the same statement also holds for less flexible GCTs as proposed in this work.

In order to shed light on this issue, we compare CTs, 2-GCTs, 2^+-GCTs, and PCTs in terms of their predictive performance for the task of modeling DNA binding sites. For learning inhomogeneous PMMs, which make a position-specific use of context trees, we use the best reported learning method in Eggeling et al. (2014b), that is, BIC (Schwarz, 1978) as structure score and fsNML (Silander et al., 2009) as parameter estimation method. As data sets, we use the CEBP data set of Eggeling et al. (2013), for which PCTs have been demonstrated to predict better than CTs, and four additional data sets from the JASPAR database (Sandelin et al., 2004), namely DAF-12 from *C. elegans*, BZR1 and PIL5 from *A. thaliana*, and human NR2C2. For all data sets and all structural variants, we compare the prediction performance using leave-one-out cross validation (Table 2). In order to further test statistical significance of differences, we also perform Wilcoxon signed rank tests (Wilcoxon, 1945) on the population log predictive probabilities ($\alpha = 0.05$).

Table 1. Percentage of the number of visited nodes in extended 2-GCTs on protein sequences of size n under memoization in relation to the maximal number when memoization is disabled (third column).

σ	d	Memoization disabled	Random $n = 100$	Insulin $n = 110$	RuBisCO $n = 479$	Myoglobin $n = 154$	Actin $n = 377$	HG α $n = 142$	GFP $n = 238$
11	5	1.27×10^9	0.576%	2.053%	35.020%	4.247%	22.125%	3.539%	9.520%
10	6	2.82×10^{10}	0.056%	0.076%	1.226%	0.131%	0.765%	0.111%	0.301%
7	7	1.40×10^{10}	0.018%	0.022%	0.432%	0.044%	0.273%	0.036%	0.105%
6	8	3.97×10^{10}	0.004%	0.005%	0.109%	0.010%	0.067%	0.009%	0.025%
5	9	4.12×10^{10}	0.002%	0.003%	0.076%	0.006%	0.046%	0.005%	0.016%

For three data sets (CEBP, DAF-12, PIL5), we observe that the possibility to skip a certain context position by using a "jump node", which is allowed in PCTs and 2^+-GCTs but forbidden in 2-GCTs and traditional CTs, significantly increases prediction performance. In the case of DAF-12, 2^+-GCTs are even significantly better than the fully flexible PCTs. While being surprising at first glance, such a phenomenon can occur if the optimal model w.r.t. to test data is already included in the less flexible model class. For BZR1 and N2RC2, even the 2-GCTs are already significantly better than the traditional CTs. In case of BZR1, there is not even a significant difference between the 2-GCT and the more flexible classes of tree structures.

These results show that even a slight generalization of structural flexibility can yield a significant increase in prediction performance. However, considering all five data sets, we may conclude that the most substantial improvement in relation to a CT in total is gained by allowing "jump node". Since PCTs are in not a single case significantly better than 2^+-GCTs, we may speculate that their full flexibility is not actually necessary, but the latter provide a better tradeoff between statistical efficiency and computational complexity. Nevertheless, there may still be cases, in which even the optimization of 2^+-GCTs is computationally too demanding, and here 2-GCTs offer the opportunity to outperform traditional context trees.

7. Concluding Remarks

This work was motivated by the observation that while CTs are statistically inefficient in comparison to PCTs in *small-data* settings, the latter entail an unfavorable time complexity for structure learning. To address this issue, we have contributed ideas to improve the state-of-the art from two different perspectives. On the one hand, we proposed two algorithmic enhancement to the PCT learning algorithm, namely (i) faster alphabet partitioning in each node of the PCT, and (ii) memoization of already computed subtrees in extended PCT. On the other hand, we proposed alternative generalizations of CTs, trading structural flexibility for computational efficiency, while keeping particular merits of PCTs, such as the capability of "jumping" over putative unimportant context positions.

Table 2. Comparison of CTs and generalizations w.r.t. predictive performance for modeling DNA binding sites ($d = 7$, all values $d < 7$ are shown in Supplement). Table entries show negative mean log predictive probabilities from a leave-one-out cross validation experiment.The rank of each method is shown in brackets. Two methods obtain the same rank when the difference between them does not pass a significance test.

Data set	CT	2-GCT	2^+-GCT	PCT
CEBP	14.31 (3)	14.23 (3)	12.75 (1)	12.57 (1)
DAF12	9.98 (3)	9.93 (3)	9.58 (1)	9.68 (2)
BZR1	12.12 (4)	11.98 (1)	11.93 (1)	11.93 (1)
PIL5	11.89 (3)	11.82 (3)	11.77 (1)	11.66 (1)
N2RC2	9.63 (4)	9.59 (3)	9.56 (1)	9.54 (1)

Our main finding—supported by analytic results as well as empirical evidence on synthetic and real data—is that both approaches in combination now let us take a significant computational advantage of the scarcity of the data. The purely algorithmic enhancements yield a considerable speed-up by two to four orders of magnitude for learning original PCTs. In addition, there is a substantial synergy effect when they are combined with less flexible CT generalizations as proposed in this work, yielding further speed-ups by several orders of magnitude. While less flexibility may entail the danger of sacrificing statistical efficiency, we have seen that 2^+-GCT are as competitive as fully flexible PCTs w.r.t. to prediction performance.

The presented ideas may enable to replace CTs by generalized variants in complex models such as permuted variable length Markov chains (Zhao et al., 2005) or variable-order Bayesian networks (Ben-Gal et al., 2005), and to determine whether k-GCTs, k^+-GCTs, or fully flexible PCTs are the structural class of choice is certainly a question that needs to be investigated for each application domain anew. While such studies and model extensions were previously computationally infeasible, the results of this work allow to benefit from the statistical efficiency of generalized context trees in a broad variety of applications.

Acknowledgements

The authors thank the anonymous reviewers for valuable suggestions to improve the presentation. This work was supported by the Academy of Finland, Grant 276864 "Supple Exponential Algorithms" (M.K.) and Deutsche Forschungsgemeinschaft, Grant GR 3526/1 (R.E., I.G.).

References

Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19: 716–723, 1974.

Bacardit, J., Stout, M., Hirst, J.D., Valencia, A., Smith, R.E., and Krasnogor, N. Automated alphabet reduction for protein datasets. *BMC Bioinformatics*, 10:6, 2009.

Begleiter, R., El-Yaniv, R., and Yona, G. On prediction using variable order Markov models. *J. Artif. Intell. Res.*, 22:385–421, 2004.

Ben-Gal, I., Shani, A., Gohr, A., Grau, J., Arviv, S., Shmilovici, A., Posch, S., and Grosse, I. Identification of transcription factor binding sites with variable-order Bayesian networks. *Bioinformatics*, 21:2657–2666, 2005.

Bourguignon, P.-Y. and Robelin, D. Modèles de Markov parcimonieux. In *Proc. JOBIM*, 2004.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. Context-specific independence in Bayesian networks. In *Proc. UAI*, pp. 115–123, 1996.

Bryant, Randal E. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35:677–691, 1986.

Bühlmann, P. and Wyner, A.J. Variable length Markov chains. *Annals of Statistics*, 27:480–513, 1999.

Edmonds, J. Paths, trees, and flowers. *Canad. J. Math.*, 17: 449–467, 1965.

Eggeling, R., Gohr, A., Bourguignon, P.-Y., Wingender, E., and Grosse, I. Inhomogeneous parsimonious Markov models. In *Proc. ECMLPKDD*, volume 1, pp. 321–336. Springer, 2013.

Eggeling, R., Gohr, A., Keilwagen, J., Mohr, M., Posch, S., Smith, A.D., and Grosse, I. On the value of intra-motif dependencies of human insulator protein CTCF. *PLoS ONE*, 9:e85629, 2014a.

Eggeling, R., Roos, T., Myllymäki, P., and Grosse, I. Robust learning of inhomogeneous PMMs. In *Proc. AISTATS*, volume 33 of *JMLR: W&CP*, pp. 229–237, 2014b.

Grau, J., Keilwagen, J., Gohr, A., Haldemann, B., Posch, S., and Grosse, I. Jstacs: A Java framework for statistical analysis and classification of biological sequences. *J. Mach. Learn. Res.*, 13:1967–1971, 2012.

Li, T., Fan, K., Wang, J., and Wang, W. Reduction of protein sequence complexity by residue grouping. *Protein Engineering*, 16:323–330, 2003.

Peterson, E.L., Kondev, J., Theriot, J.A., and Phillips, R. Reduced amino acid alphabets exhibit an improved sensitivity and selectivity in fold assignment. *Bioinformatics*, 25:1356–1362, 2009.

Quinlan, J. Ross. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

Rissanen, J. A universal data compression system. *IEEE Trans. Inform. Theory*, 29:656–664, 1983.

Rota, G.C. The number of partitions of a set. *Amer. Math. Monthly*, 71:498–504, 1964.

Sandelin, A., Alkema, W., Engström, P., Wasserman, W.W., and Lenhard, B. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Research*, 32:D91–D94, 2004.

Schwarz, G. E. Estimating the dimension of a model. *Annals of Statistics*, 2:461–464, 1978.

Seifert, M., Gohr, A., Strickert, M., and Grosse, I. Parsimonious higher-order hidden Markov models for improved array-CGH analysis with applications to Arabidopsis thaliana. *PLoS Computational Biology*, 8:e1002286, 2012.

Silander, T., Roos, T., Kontkanen, P., and Myllymäki, P. Factorized NML criterion for learning Bayesian network structures. In *Proc. PGM*, pp. 257–264, 2008.

Silander, T., Roos, T., and Myllymäki, P. Locally minimax optimal predictive modeling with Bayesian networks. In *Proc. AISTATS*, volume 5 of *JMLR: W&CP*, pp. 504–511, 2009.

The UniProt Consortium. Update on activities at the Universal Protein Resource (UniProt) in 2013. *Nucleic Acids Research*, 41:D43–D47, 2013.

Volf, P. and Willems, F. Context maximizing: Finding MDL decision trees. In *Proc. 15th Symp. Inform. Theory Benelux*, pp. 192–200, May 1994.

Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Zhao, X., Huang, H., and Speed, T.P. Finding short DNA motifs using permuted Markov models. *Journal of Computational Biology*, 12:894–906, 2005.