Scaling log-linear analysis to datasets with thousands of variables

François Petitjean Geoffrey I. Webb
Faculty of IT, Monash University, Melbourne, Australia – firstname.lastname@monash.edu

Abstract

Association discovery is a fundamental data mining task. The primary statistical approach to association discovery between variables is log-linear analysis. Classical approaches to log-linear analysis do not scale beyond about ten variables. We have recently shown that, if we ensure that the graph supporting the log-linear model is chordal, log-linear analysis can be applied to datasets with hundreds of variables without sacrificing the statistical soundness [21]. However, further scalability remained limited, because state-of-the-art techniques have to examine every edge at every step of the search. This paper makes the following contributions: 1) we prove that only a very small subset of edges has to be considered at each step of the search; 2) we demonstrate how to efficiently find this subset of edges and 3) we show how to efficiently keep track of the best edges to be subsequently added to the initial model. Our experiments, carried out on real datasets with up to 2000 variables, show that our contributions make it possible to gain about 4 orders of magnitude, making log-linear analysis of datasets with thousands of variables possible in seconds instead of days.

Keywords: Association discovery, statistical testing, high-dimensional data, chordal graphs

1 Introduction

Log-linear analysis (LLA) is the well established statistical technique for finding associations between discrete variables in data [11]. The general objective of LLA is to select a model that satisfactorily explains the observed frequencies of a given categorical dataset (*i.e.*, a model of the joint distribution). A general algorithm of LLA's forward selection process is given in Algorithm 1 of the supplementary file available at [19].

General approaches to LLA are exponential with respect to the number of variables, which make them impractical for datasets with more than a dozen variables [5, 21]. This is because the evaluation of the trade-off between the complexity of the model and its quality of fit to the data requires consideration of all the possible outcomes [5], *i.e.*, 2^M outcomes for M binary variables.

Recently, we have shown that this evaluation can be performed for high-dimensional data, and without approximation, if the model belongs to the class of *multiplicative* log-linear models [21, 20]. This corresponds to the Markov random fields for which the supporting graph is *chordal* [11].

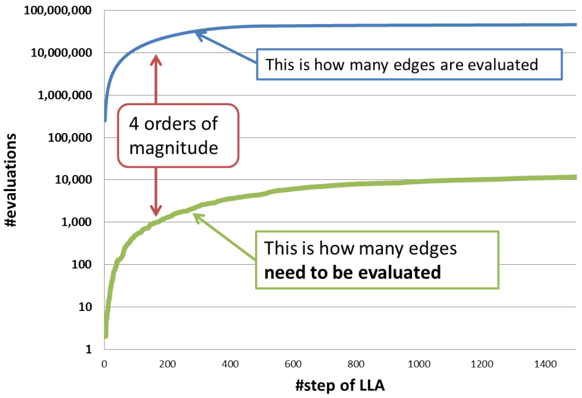

Figure 1: Study of the number of evaluations that are performed by state-of-the-art techniques (top), compared to the actual required number (bottom).

Although this discovery made it possible to perform LLA for datasets with hundred variables, tackling datasets with 1000+ variables is still out-of-reach, because of the general quadratic complexity of every step of LLA. At every step of the process, every possible edge has to be considered for addition to the current reference model $\mathcal{M}^\star$, and the number of possible edges is in the order of $O(M^2)$. For example, performing LLA with the state-of-the-art for high-dimensional data – Chordalysis [21] – requires more than 3 days of computation for a dataset with 700 variables (see our experiment on the `Protein` dataset in Section 5).

This paper is based on the idea that, **at every step, it is only necessary to re-consider a subset of edges** for addition to the successively refined models.

Let us motivate this idea with another real-world dataset representing 30,000 news articles described by 500 variables (see the description of dataset `ABC News` in Section 5 for more details). On the one hand, we recorded how many edges are examined by the LLA process. We report this number over the course of the LLA process in the top curve in Figure 1: the process examines the addition of more than 10,000,000 edges. On the other hand, we looked at how many edges actually lead to the same evaluation of the model between successive steps of LLA. We report the difference – *i.e.*, the number of times that edges need to be re-examined after the very first step – in the bottom curve of Figure 1: only about 10,000 edges' additions require re-examination.[1] This means that *the vast majority of the computation could be avoided if we knew which edges would lead to the same evaluation of the model*. This is, quite simply, the aim of the paper: showing how to exactly predict that an edge will need to be re-examined, and designing an algorithm that utilizes that knowledge to perform LLA several orders of magnitude faster than the current state-of-the-art methods.

Our experiments on real-world datasets with up to 2000 variables show that our algorithm, *Prioritized Chordalysis*, performs LLA about 4 orders of magnitude faster than state-of-the-art techniques, without making any additional assumption. *Prioritized Chordalysis* will is available open-source at http://sourceforge.net/p/chordalysis/.

This paper is organized as follows. In Section 2, we formalize the problem. In Section 3, we present our solution *Prioritized Chordalysis*, which enables the discovery of statistically sound multi-way correlations between the variables of datasets with thousands of variables. In section 4, we place this work in the context of related research. In Section 5, we conduct experiments that demonstrate the performance and relevance of our approach. Finally, we conclude this work in Section 6.

2 Definitions and problem statement

2.1 Log-linear models and log-linear analysis

Let $\mathcal{D}$ be a dataset of N samples over a set of M discrete variables $\mathcal{V} = \{V_1, \cdots, V_M\}$. Every variable V takes values in $\mathrm{Dom}(V)$. $\mathcal{D}$ is drawn from a probability distribution $p_{\mathcal{V}}$ over $\mathcal{V}$.

Log-linear models use a first-degree polynomial function to model the logarithm of the frequencies that can be observed in a contingency table.

Log-linear analysis (LLA) is the general name given to methods that seek to select a statistically significant log-linear model from data. This corresponds to determining which of the u terms have to be part of the model. LLA classically uses hypothesis testing to decide if the current reference model (the *null* hypothesis) has to be replaced by a candidate model that is a variation of the reference model (the tested hypothesis). Statistical methods iteratively refine an initial model, for as long as the addition of terms results in a statistically significant improvement in the model's fit.

Replacing a current reference model $\mathcal{M}^\star$ by a candidate model $\mathcal{M}^c$ thus requires to assess a trade-off between quality and complexity, because:

- $\mathcal{M}^c$ is always going to fit the data better than $\mathcal{M}^\star$, because the two models are nested and $\mathcal{M}^c$ includes an additional terms;
- $\mathcal{M}^c$ is always going to be more complex than $\mathcal{M}^\star$, for the same reason.

Broadly speaking, the aim of the evaluation function (*evaluate_replacement* in Algorithm 1 of the supplementary file available at [19]) is to assess if the improvement in the quality of the model is significant enough to "justify" the increase in the complexity.

Let us now formalize this evaluation; we use $\mathrm{O}^A_{\mathbf{x}}$ (resp. $\mathrm{E}^A_{\mathbf{x}}$) to designate the observed (resp. expected from a model $\mathcal{M}$) frequencies for the configuration $\mathbf{x}$ with respect to the set of variables A. The improvement of the fit to the data of $\mathcal{M}^c$ compared to $\mathcal{M}^\star$ is estimated using the standard in statistics – the likelihood ratio test statistic [5, p. 97]:

$$\begin{aligned} {\mathrm{G}^2}_r = \mathrm{G}^2(\mathcal{M}^\star \text{ vs. } \mathcal{M}^c) &= \mathrm{G}^2(\mathcal{M}^\star) - \mathrm{G}^2(\mathcal{M}^c) \\ \text{with } \mathrm{G}^2(\mathcal{M}) &= 2 \cdot \sum_{\mathbf{x}\in\mathcal{V}} O^{\mathcal{V}}_{\mathbf{x}} \cdot \ln\left(\mathrm{O}^{\mathcal{V}}_{\mathbf{x}} / \mathrm{E}^{\mathcal{V}}_{\mathbf{x}}\right) \end{aligned}$$

Similarly, the difference in the complexity between the models is evaluated as the difference in the number of degrees of freedom $df_r = df(\mathcal{M}^\star) - df(\mathcal{M}^c)$ [5, p. 97].

The replacement of $\mathcal{M}^\star$ by $\mathcal{M}^c$ is then rejected with a significance level α if

$$\begin{aligned} &(2.1) & {\mathrm{G}^2}_r &> \chi^2(1-\alpha, df_r) \\ &(2.2) \Leftrightarrow & 1 - \int_{x=0}^{{\mathrm{G}^2}_r} \chi^2(x, df_r)dx &> \alpha \end{aligned}$$

The left term in Equation 2.2 is the p-value for the replacement of $\mathcal{M}^\star$ by $\mathcal{M}^c$.

2.2 LLA for high-dimensional data Statistical methods for LLA do not scale up beyond a dozen variables for the general class of log-linear models. This is because computing G^2, to evaluate the replacement of $\mathcal{M}^\star$, is exponential in the number of variables. This assessment indeed implies iteration over all possible combinations of values for all the variables. This assessment is clearly infeasible when the number of variables is high (*e.g.*, for 300 binary variables only, a single evaluation would require more operations than there are fundamental particles in the observable universe).

We have recently shown that the full LLA paradigm can still be used to analyze high-dimensional data if we focus on the subclass of log-linear models that are *decomposable* (or *multiplicative*) [21]. To the best of our knowledge, this is the only subclass of models for which the full statistical LLA paradigm (as described in Section 2.1) can apply, because decomposable models are

[1] Note that the remainder of this manuscript will make it clear how this graph can be generated.

the only log-linear models with closed-form maximum likelihood estimates [11].

In addition, decomposable models are not only practical but also a useful class of models. This is ensured by the fact that, for any non-decomposable log-linear model, there always exists a decomposable model that subsumes it and can hence exactly model any distribution that it models [21].

Definition 1. *[5] A log-linear model is* graphical *if, whenever the model contains all two-factor terms generated by a higher-order interaction, the model also contains the higher-order interaction.*

Property 2.1. *Being completely determined by its two-factor terms,* graphical *models can be represented by an undirected graph, where the vertices represent the variables and the edges represent the two-factor terms of the model.Note that graphical log-linear models are equivalent to Markov networks.*

Definition 2. *A graphical log-linear model is* decomposable *if the supporting graph is* chordal*, i.e., if the graph does not admit chord-less cycles of length strictly greater than three.*

Property 2.2. *Decomposable models are the only log-linear models that have closed-form maximum likelihood estimates (MLEs) [11]. Let $\mathcal{M}$ be a* decomposable *log-linear model with associated* chordal *graph $\mathcal{G} = (\mathcal{V}, E)$, its maximum likelihood estimates follow:*

$$\hat{p}_{\mathcal{M}}(\mathbf{x}) = \frac{\prod_{C \in \mathcal{C}} \hat{p}_C(\mathbf{x})}{\prod_{S \in \mathcal{S}} \hat{p}_S(\mathbf{x})} \tag{2.3}$$

where $\mathcal{C}$ (resp. $\mathcal{S}$) are the maximal cliques (resp. minimal separators) of $\mathcal{G}$, and $\hat{p}_A$ represents the marginal probability of $\hat{p}_{\mathcal{V}}$ over a set of variables A.

2.3 Why cannot current approaches tackle datasets with 1000+ of variables? As we have intuited in the introduction to this paper, the critical issue to scaling up LLA to datasets with thousands of variables lies in the number of times that every edge is examined for addition to the current reference model. This paper will show that only a very limited number of edges need to be re-examined at each step. Let us first motivate this intuition with a few examples; the remainder of this paper will demonstrate their validity.

Intuition 1: disconnected components. Consider the model of a joint distribution over four variables (age – a, height – h, gender – g and cholesterol – c) illustrated in Figure 2(a). Starting with a model considering that the 4 variables are independent, the first step consists of finding which one of the 6 edges will result in the most statistically significant model. To this end, the addition of every single edge is evaluated. Let us assume that this model is the one including edge $\{a, h\}$, *i.e.*, including the correlation between age and height. The second step is then going to assess the addition of every single edge again. The p-value (see Equation 2.2) associated with the addition of edge $\{g, c\}$ is identical, regardless of it being added at the first or second step, because associated variables are not in the same connected components of the graph, and hence not “interacting” in the model; cholesterol and gender are independent of age and height ($c, g \perp\!\!\!\perp a, h$). As a result, this edge need not be evaluated again at the second step.

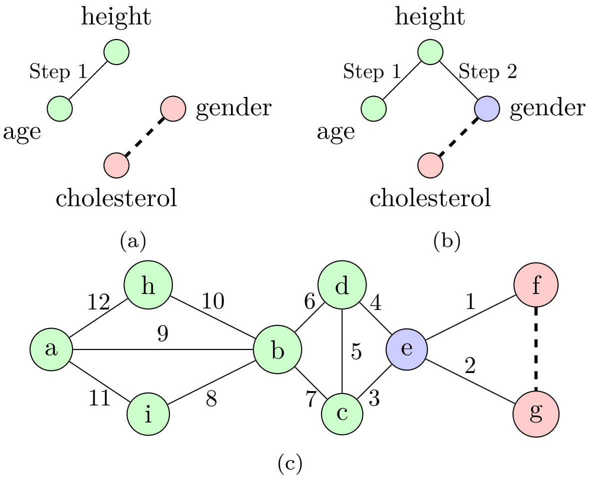

Figure 2: Illustrative cases where edges need not be re-evaluated between subsequent steps of LLA.

Intuition 2: empty minimal separator. Consider the model in Figure 2(b) that results from including the interaction $\{h, g\}$ at step 2. The third step is then going to examine the addition of every remaining edge again. The p-value associated with the addition of edge $\{g, c\}$ is identical, regardless of it being added at the first, second or third step, because adding $\{g, c\}$ will “explain” the same quantity of information in all three models; cholesterol is independent of age and height given gender ($c \perp\!\!\!\perp a, h \,|\, g$). We will see that this is due to an empty minimal separator between g and c: $S_{gc} = \emptyset$, *i.e.*, there is no vertex to remove from the graph to disconnect g from c. As a result, this edge need not be evaluated again at the second and third step.

Intuition 3: identical minimal separator. Consider the more elaborate model over 9 variables illustrated in Figure 2(c), where the numbers on the edges indicate the steps at which they were added. We show that from step 3, the addition of edge $\{f, g\}$ to any successively refined model need not be evaluated again and that the significance of adding $\{f, g\}$ will remain invariant. This is motivated by the fact that, from step 3 on-wards, removing the vertex e disconnects f from g

($S_{fg} = \{e\}$), leading to $f, g \perp\!\!\!\perp a, b, c, d, h, i \,|\, e$. In consequence, the last time that the addition of this edge needs to be evaluated is at step 3.

The next section will prove the validity of these intuitions. It is interesting to observe that being able to tell if an edge has to be re-evaluated is not sufficient, because the LLA process will still enumerate over all the edges at every step. This enumeration prevents LLA from scaling to datasets with thousands of variables, because there are $O(M^2)$ such edges for M variables. We will show that *Prioritized Chordalysis* can precisely identify the edges that have to be re-evaluated, and use this information to maintain a data structure that makes it possible avoid such enumeration.

3 Method – Prioritized Chordalysis

In this section, we introduce our method: *Prioritized Chordalysis*. We first lay its theoretical foundations by characterizing when an edge need or need not be re-examined between two steps of the LLA process. Then, we show how to use an advanced graph data structure – the clique-graph – to track the edges that require re-examination. We then show how to use a priority queue to iterate over the best modifications of the current reference model, in place of enumerating over all possible edges. Finally, we examine the complexity of our process and compare it to the state of the art.

3.1 What edges require re-examination? We have seen in Section 2.1 that computing the statistical significance (p-value) of replacing a current reference model $\mathcal{M}^\star$ by a candidate model $\mathcal{M}^c$ requires two elements: the difference in the fit ${\text{G}^2}_r$ and the difference in the complexity df_r. We now develop these elements for our target class of models, *i.e.*, decomposable models.

Definition 3. *[9, Definition 1] Let $\mathcal{G} = \{V, E\}$ be an undirected graph and two vertices $a, b \in V$. The set of vertices $S \subset V$ is an (a, b)-separator if removing S from $\mathcal{G}$ separates the vertices a and b into different connected components. If no proper subset of S is an (a, b)-separator, then S is a minimal (a, b)-separator, noted S_{ab}.*

We have moreover recently shown that:

Theorem 3.1. *[21, Theorem 1] If two decomposable models $\mathcal{M}^c \subset \mathcal{M}^\star$ differ only in one edge $\{a, b\}$, then:*

$$\begin{aligned} {\text{G}^2}_r &= 2 \cdot N\big(\text{H}(S_{ab} \cup \{a\}) \;+ \text{H}(S_{ab} \cup \{b\}) \\ &\quad - \text{H}\left(S_{ab} \cup \{a, b\}\right) - \text{H}(S_{ab})\big) \end{aligned} \tag{3.4}$$

where H(.) *denotes the entropy.*

Similarly for the assessment of the complexity, using Equation 2.3 and [5, Equation 10, pp. 97–98] we have:

$$\begin{aligned} df_r &= param\left(S_{ab} \cup \{a, b\}\right) + param(S_{ab}) \\ &\quad - param(S_{ab} \cup \{a\}) \; - param(S_{ab} \cup \{b\}) \end{aligned} \tag{3.5}$$

where $param(A) = -1 + \prod_{v \in A} |\,\text{Dom}(v)|$ and $|\,\text{Dom}(v)|$ the number of possible values for variable v.

As a result, the evaluation of the replacement of $\mathcal{M}^\star$ by $\mathcal{M}^c$ is a function of only four elements: the two vertices a and b that are newly linked in $\mathcal{M}^c$, the minimal separator S_{ab} of those vertices in $\mathcal{G}^\star$, and the dataset $\mathcal{D}$ for the computation of the marginal entropies. We can thus formulate the following theorem:

Theorem 3.2. *Let $\mathcal{M}_1^\star$ and $\mathcal{M}_2^\star$ be two reference models selected at different steps of LLA, $\mathcal{G}_1^\star = \{\mathcal{V}, E_1^\star\}$ and $\mathcal{G}_2^\star = \{\mathcal{V}, E_2^\star\}$ their associated graphs, and a, b two vertices such that $a, b \in \mathcal{V}, \{a, b\} \notin E_1^\star, E_2^\star$ (i.e., there is no edge between a and b in either models) and $\mathcal{G}_1^\star \cup \{a, b\}$ and $\mathcal{G}_2^\star \cup \{a, b\}$ are both chordal graphs (i.e., adding $\{a, b\}$ to either graphs keep them chordal). If S is a minimal (a, b)-separator in $\mathcal{G}^1$ and $\mathcal{G}^2$ ($S_{ab}^{\star_1} = S_{ab}^{\star_2}$), then the p-value associated with the addition of $\{a, b\}$ to $\mathcal{M}_1^\star$ is identical to the p-value associated with the addition of $\{a, b\}$ to $\mathcal{M}_2^\star$.*

Proof. Let $\mathcal{M}_1^{c_{ab}}$ (resp. $\mathcal{M}_2^{c_{ab}}$) be the candidate model considering the addition of edge (a, b) to $\mathcal{M}_1^\star$ (resp. $\mathcal{M}_2^\star$). If $S = S_{ab}^{\star_1} = S_{ab}^{\star_2}$, then Equations 3.4 and 3.5 directly give $\text{G}^2(\mathcal{M}_1^\star \text{ vs. } \mathcal{M}_1^{c_{ab}} = \text{G}^2(\mathcal{M}_2^\star \text{ vs. } \mathcal{M}_2^{c_{ab}})$ and $df(\mathcal{M}_1^\star \text{ vs. } \mathcal{M}_1^{c_{ab}}) = df(\mathcal{M}_2^\star \text{ vs. } \mathcal{M}_2^{c_{ab}})$. □

A direct consequence of this theorem is that the p-value associated with the addition of an edge only has to be re-evaluated between two steps of LLA if its minimal separator changes between these steps. The possible gain in computation then depends upon how frequently do minimal separators actually change between successive steps. This obviously depends on the underlying structure of the dataset. We can however bound the maximum number of edges that will change between two steps of the LLA process.

Theorem 3.3. *The number of edges that need to be re-examined after adding edge (a, b) to the current reference model is at most $2(M-1) - |N(a)| - |N(b)|$, where $N(x)$ designates the neighbours of x, i.e., only $O(M)$ edges require re-examination at every step.*

Proof. Adding (a, b) to a chordal graph results in the addition of only one maximal clique: $C_{ab} = S_{ab} \cup a \cup b$ [7, Section 3.2.1]. Any new edge added to the clique-graph has C_{ab} as one of its endpoints [7, Theorem 4.3] (note that we use the term “clique-graph” as defined in [9]). It results that any edge impacted by the addition of (a, b) has either the form (a, x) or (b, x) [7, Proof to Theorem 4.3]. Given that a can at most be connected to $M - 1$ vertices and that it is already connected to $|N(a)|$ of them, there are at most $M - 1 - |N(a)|$ edges of the form (a, x). Similar reasoning for b. □

This fundamental result establishes that, in the worst case scenario, only $O(M)$ edges have to be re-examined at each step. This strongly contrasts with the state-of-the-art techniques that require examination of all $O(M^2)$ possible edges.

3.2 How to select all edges that need to be re-examined? We have shown in the last subsection that an edge needs to be re-examined between two steps of LLA if and only if the associated minimal separator has changed between these two steps. The naive way to select all the edges that need to be re-examined at every new step would then be to iterate over all edges (a, b) and select those for which the minimal separator has changed. However, we have seen that iterating over all possible edges at every step of LLA is precisely the limiting factor to scale up to datasets with thousands of variables. Furthermore, even this naive selection would require prohibitive calculations, because finding all S_{ab} itself requires $O(|\mathcal{V}| + |E|)$ operations for chordal graphs [14].

In this sub-section, we show how both these problems can be solved using an advanced graph-theoretical data structure – the clique-graph [9]:

1. We demonstrate that, for all edges (a, b) that are considered for addition to successive reference models $\mathcal{M}^\star$, their minimal (a, b)-separators can be efficiently derived from the clique-graph.
2. We take an existing algorithm that aims at maintaining the clique-graph data structure when iteratively adding edges to the supporting graph [7], and show how to modify it to keep track of all minimal (a, b)-separators.

3.2.1 Minimal vertex separators align with edges of the clique-graph The clique-graph structure is an ideal base-structure for our task of keeping track of all minimal separators between the vertices.

Definition 4. *[9, Definition 2] Let $\mathcal{G}$ be a chordal graph. The clique-graph $C(\mathcal{G}) = \{\mathcal{V}_c, E_c\}$ is defined by:*

- *$\mathcal{V}_c$ is the set of maximal cliques of $\mathcal{G}$*
- *(C_1, C_2) belongs to E_c iff $C_1 \cap C_2$ is a minimal (a, b)-separator for each $a \in C_1 \backslash C_2$ and each $b \in C_2 \backslash C_1$.*

We now formulate the theorem that is the base for tracking the minimal separators.

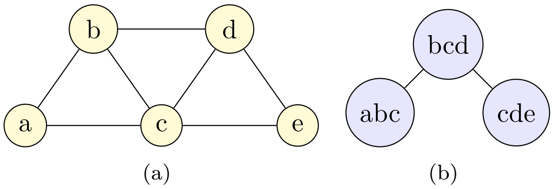

Figure 3: Example of chordal graph (a) and its associated clique-graph (b). Note that edge (abc, cde) is not part of the clique-graph because removing $abc \cap cde = c$ does *not* disconnect a from e.

Theorem 3.4. *If (a, b) can be added to a chordal graph $\mathcal{G}$ while maintaining its chordality, then $S_{ab} = C_a \cap C_b$ where $(C_a, C_b) \in E_c$, $a \in C_a$ and $b \in C_b$.*

Proof. If adding (a, b) maintains the chordality of $\mathcal{G}$, then $\exists (C_a, C_b)$ in $C(\mathcal{G})$ such as $a \in C_a$ and $b \in C_b$ [7, Lemma 3.1]. By Definition 4, if $(C_a, C_b) \in E_c$, then $C_a \cap C_b$ is a minimal (a, b)-separator. □

3.2.2 Efficiently tracking all minimal separators We have demonstrated in Theorem 3.4 that for all edges (a, b), the minimal (a, b)-separator S_{ab} can be obtained from edges (C_a, C_b) of the clique-graph, such as $a \in C_a$ and $b \in C_b$. A naive way of keeping track of all the minimal separators could thus be to iterate over the edges (C_1, C_2) of the clique-graph, and for each one of them, to iterate over all pairs of vertices $(x, y), x \in C_1 \backslash C_1 \cap C_2, y \in C_2 \backslash C_1 \cap C_2$ and memorize that $S_{xy} = C_1 \cap C_2$. This would however lead to a vast amount of unnecessary computations, because most of the structure of the clique-graph remains unchanged when adding an edge to the associated (normal) graph.

We here refine the state-of-the-art algorithm for the iterative update of clique-graphs [7] in order to keep track of all minimal vertex separators. Note that we only detail the modified parts of [7]'s algorithm.[2] The theoretical contribution of this part of the paper concerns $\mathcal{E}^f_\mathcal{M}$ – a boolean $(M \times M)$-matrix informing about the eligibility of any edge for addition to the current graph – and its iterative update:

1. We make $\mathcal{E}^f_\mathcal{M}$ a function that associates to any pair of vertices (a, b), its eligibility, its minimal separator S_{ab} and the clique-graph edge $(C_a, C_b) \in E_c$ such that $a \in C_a$, $b \in C_b$ and $C_a \cap C_b = S_{ab}$, *i.e.*, two nodes of the clique-graph allowing a to be connected to b in $\mathcal{G}$.
2. Following our Theorem 3.4, every time a new edge (C', C_{ab}) is added to the clique-graph as a result of adding (a, b), we set $\mathcal{E}^f_\mathcal{M}(x, a)$ to $(true, C' \cap C_{ab}, C', C_{ab})$ for all (x, a) such as $x \in C' \backslash C_{ab}$, $a \in C_{ab} \backslash C'$. Similarly for b.

[2] The reader can refer to the original paper and to our implementation available at [19] for more details.

3. Every time an edge (C_1, C_2) is deleted from $C(\mathcal{G})$ as a result of adding (a, b) to $\mathcal{G}$, and noting that such (C_1, C_2) will follow $C_1 \cap C_2 = S_{ab}$, we set $\mathcal{E}^f_{\mathcal{M}}(x, a)$ to $(false, \_, \_, \_)$ for all pairs (x, y) such that x (resp. y) is in the same connected component as a (resp. b) in $\mathcal{G} - S_{ab}$.

In addition, note that the scientific community has challenged the correctness of [7]'s algorithm, in particular for the case where $\mathcal{G}$ is made of several connected components [2], which leads to empty minimal separators. We attribute this to a few unfortunate typos present in [7], to the use of an imprecise vocabulary,[3] and to the absence of any available implementation of the algorithm. We have clarified, corrected and extended [7]'s algorithm. Our algorithm can easily been reversed back to the original algorithm by only considering the boolean values in $\mathcal{E}^f_{\mathcal{M}}(x, a)$. Note that the validity of our implementation has been carefully checked and tested over hundreds of experiments, where we verified that it led to the same results as algorithms which do not make use of the clique-graph [4].

3.3 Efficiently iterating over the best edges

This subsection describes the last component of our algorithm: how to prevent enumeration over all possible edges at every step.

At every step, the standard LLA framework considers all the possible modifications of the current reference model [5, Chapter 6]. This requires iteration over all $O(M^2)$ possible edges, which is the limiting factor to perform LLA for datasets with thousands of variables. As there are at most $\binom{M}{2}$ steps, state-of-the-art algorithms can *all* lead to the examination of $O(M^4)$ edges.

We propose to use a priority queue to store the edges that have to be successively considered for addition to the current reference model. We keep the edges ordered by their associated statistical significance. As we have seen in Section 3.1, if the minimal separator associated with an edge does not change from one step to another, neither does the statistical significance associated with this edge. This means that, at every step, the only edges that are going to change in the queue, are 1) the edges that are not eligible any more because they would not keep the graph chordal, 2) the edges that are newly eligible and 3) the edges that have had a change of minimal separator.

We have shown in Section 3.2 that such changes are all associated to the addition and deletion of edges in the clique-graph: adding a clique-graph edge enables new edges (or change their minimal separator) while removing a clique-graph edge disables edges.

To keep the explanation simple, and because we will see that this does not change the overall complexity, we consider a priority queue based on a heap data structure, with retrieval and removal of the min in $O(1)$, and insertion/deletion of an element in $O(\log n)$.

We now prove that, even in the worst case, our solution exhibits a far better complexity than state-of-the-art methods.

Initialization. At the start, all pairs of edges are sorted and added to the queue, which requires $O(M^2 \log(M))$ operations.

Edge addition. Any new clique-graph edge has C_{ab} as its endpoint (see proof to Theorem 3.3). In consequence any edge impacted by the addition of (a, b) has either the form (x, a) or (x, b). As a (and b) cannot be connected to more than $M - 1$ vertices, at every step of LLA, at most $2(M - 1)$ edges might be added to the queue; resulting in a quasi-linear complexity with the number of variables for each of the $O(M^2)$ possible steps, thus $O(M^3 \log M)$.

Edge deletion. Any edge that is removed from the priority queue has to have been added to it. As there are at most $O(M^2 \log M + M^3 \log M)$ such additions, there will also be at most $O(M^3 \log M)$ such deletions.

Overall. For k steps performed, our algorithm thus requires only $O(kM \log M)$ operations; every step of LLA exhibits a quasi-linear complexity with the number of variables. This starkly contrasts with the quadratic $O(kM^2)$ complexity of state-of-the-art algorithms [7, 2, 21, 20]. Our experiments will show that this difference makes it possible to gain efficiency by several orders of magnitude and allows us to perform LLA for datasets with thousands of variables.

4 Related research

Researchers have investigated the learning of graphical log-linear models from high-dimensional data.

A first approach builds log-linear models on subsets of variables – for which the classical LLA scales up – and then to combine these sub-models [25, 6]. However, because they make strong assumptions about the independence between the variables, they often inaccurately discover associations between variables (see for example Section 5.2 in [6]), and thus do not align with the required low false-discovery rate of LLA.

A second approach uses ℓ_1-regularizers. This makes the search possible in very high dimensional spaces, by biasing the search towards models for which many parameters are zero. Different configurations have been studied: performing a logistic regression for every variable independently [24], focusing on a reduced subset of

[3] An example is the use of "connected" which can be interpreted as the presence of a direct edge between two vertices, or as the existence of a path connecting these vertices.

features [15] or finding a set of variables that best divides the graph [10]. Because these methods aim mostly at being predictive (as opposed to explanatory), and because they focus on local substructures, they often result in false discoveries (see for example the precision trend depicted in [24] – Section 6) and thus cannot be considered as LLA methods.

A third approach evaluates the trade-off quality/complexity of the models in order to ensure that only associations, for which there is enough evidence, are included in the model. We have showed with Chordalysis that LLA can be correctly performed to datasets with hundreds of variables for the class of decomposable models – based on χ^2 goodness-of-fit tests in [21] and on information theory in [20]. However, as we have explained in this paper (and as our experiments will demonstrate), these methods cannot tackle datasets with thousands of variables.

5 Experimental evaluation

We have shown in Section 3 that Prioritized Chordalyis dominates the state of the art in terms of algorithmic complexity. This section seeks to demonstrate its computational superiority on real-world datasets. Note that this section does not seek to further assess the relevance of χ^2 goodness-of-fit tests for LLA, because it has long been accepted by statisticians (see [5, 21]). Rather, our experiments seek to demonstrate that we can achieve further scalability without sacrificing the statistical soundness of LLA.

To this end, we consider four successively refined algorithms for LLA, starting from the current state of the art for high-dimensional data [21] and progressively incorporating the contributions of this paper:

Version 1 We start with Chordalysis: the first method that can perform LLA on high-dimensional data [21].

Version 2 We integrate the clique-graph update algorithm from [7] into Version 1.

Version 3 We add to *Version 2* the ability to keep track of the minimal (a, b)-separators.

Version 4 – Prioritized Chordalysis We add to *Version 3* the ability to keep track of the best edges to be successively added in a priority queue.

On the need for a variety of real-world datasets. As we have demonstrated in Section 3.3, the worst-case complexity only depends on the number of variables. This is a consequence of the number of edges depending on the number of vertices. However, the number of edges to be discovered from data depends on the actual dependencies that can be found in data. If the data is drawn from a probability distribution where all variables are actually independent, then the process will quickly finish. In contrast, real-world datasets often exhibit numerous high-order correlations, leading to more computation time.

In addition, the quantity of data has also a significant impact on the computation time. This can seem counter-intuitive because the scoring of an edge depends on four entropies only (see Theorem 3.1), and each entropy can be naively computed with a quasi-linear complexity with the size of the dataset. However, increased data quantity allows more edges to be identified as statistically significant and will thus often lead to a very significant increase in the computation time. This is well exemplified by the toss of a coin and the associated decision: if we toss the coin 100 times and we observe 51 heads and 49 tails, we cannot state that the coin is unbalanced. However, it we toss it 100,000 times and 51,000 heads and 49,000 tails, and while this is the same proportion of heads/tails, statistical tests tell us that we can confidently state that it is very unlikely that the coin is balanced. This phenomenon is similar to the one observed with the learning of decision trees: larger quantities of data will tend to create deeper trees.

This is why we use a broad range of real-world datasets, with both various number of variables and various quantities of data:

Mushroom the classical mushroom dataset, 22 variables, 8k examples [3].

EPESE epidemiological study of the elderly, 25 variables, 14k examples [22].

Internet demographic information on internet users, 70 variables, 10k examples [12].

CoIL2000 insurance customer management, 86 variables, 6k examples [23].

MITFace face recognition dataset, discretized to 4 bins using equal frequency, 362 variables, 31k examples [17].

Finance stock performance of the companies listed in the S&P500 over 20 years of trading, 500 variables.

Protein Multiple alignment of the Serpin family of proteins, 750 variables, 212 proteins [13].

Orphamine Frequency of occurrence of 1260 symptoms for 2600 rare diseases, 1260 variables, 2600 examples [18].

ABC Use of the 500 most interesting words in all the news articles about Melbourne published by the Australian Broadcasting Network (ABC), 500 variables, 35k examples.

NYT Use of the 2000 most interesting words in 10% of the articles published by the New York Times from 1987 to 2007, 2000 variables, 180k examples [8].

Where licensing restrictions permit us to do so, we have made these datasets available at [19].

Figure 4 presents the computation time required to perform LLA for every version of the algorithm on these real-world datasets. Note that the graphs associated with each dataset are provided at [19]. These results confirm the superiority of our method. Prioritized Chordalysis is the fastest method for all datasets. Moreover, for all datasets with more than 100 variables (from MIT Face), Prioritized Chordalysis performs LLA with about 4 orders of magnitude faster than the state of the art. For example, for the ABC dataset – which comprises 500 variables – Prioritized Chordalysis performs LLA in 27 seconds while Chordalysis (Version 1) requires 39 hours (to obtain exactly the same result); this is more than a 5200x speedup.

This is a major result that makes it possible to tackle datasets with *thousands* of variables. For such datasets, our experiments indeed show that Prioritized Chordalysis makes it possible to perform LLA in seconds or minutes, when the state of the art requires days. For example, for the NYT dataset – which comprises 2000 variables – Prioritized Chordalysis performs LLA in only 3 minutes while Version 1 could not provide any result in 10 days of computation.

Furthermore, we can observe that all the successive elements that we have introduced in this paper play a major role in making LLA scalable to very high-dimensional datasets. Each of the contributions that we have made in this paper – from providing a complete and correct clique-graph-update algorithm, to keeping track of the minimal separators in order to maintain the possible modifications in a priority queue – gains one to two orders of magnitude, depending on the dimensionality of the dataset, amount of available evidence, and complexity of the underlying joint distribution.

Finally, we examine the scalability of Prioritized Chordalysis, on a dataset with increasing number of variables. The **NYT** dataset is a good test bed for this task because 1) it is our biggest dataset with 180,000 instances and 2000 variables and 2) its variables are ordered (occurrence frequency of every word), which makes its study possible with an increasing number of variables; the most frequent words first.

Figure 5 presents the results of this experiment. We can observe that our proposed algorithm, Prioritized Chordalysis, greatly dominates all other methods. Moreover, we can see that the magnitude of the improvement of Prioritized Chordalysis actually increases over time, *i.e.*, the functions get farther apart as the number of variables increases.

Interestingly, we can also see that when the number of variables increases, Version 2 tends to perform as fast as Version 3. This is because the time required to find the minimal separator of every edge from the clique graph (Version 2) becomes negligible relative to maintaining the structure of the clique graph.

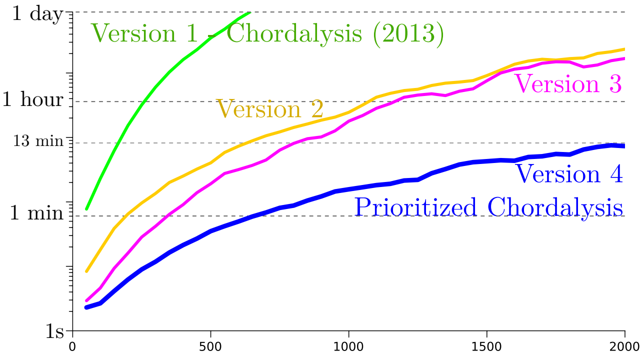

Figure 5: Comparison of the computation time required to perform LLA with regard to the number of variables used – dataset NYT. We limited the discovery to the first 100 edges to limit the computation time.

In consequence, tracking the minimal separators (Version 3) tend to provide only marginal improvement over Version 2. Note however that Version 4 (Prioritized Chordalysis) requires to keep track of the minimal separators to maintain the priority queue; this element is thus necessary to obtain the exhibited improvement.

6 Conclusions and future work

Log-linear analysis (LLA) is the statistically established method to select a statistically significant model of the joint distribution for a categorical dataset. In 2013, we made LLA possible for datasets with more than a dozen variables [21]. However, scalability to datasets with thousands of variables was prevented because the addition of every edge had to be re-examined at every step of LLA.

This paper showed that a very small subset of edges has to be re-examined at each step of LLA, and demonstrated how to identify the set of such edges. Backed by these theoretical contributions, we then showed a priority queue can keep track of the best edge to be added to the current reference model, thus allowing the reference model to be successively refined without iterating over all the possible edges. Finally, we have demonstrated that our algorithm will, in the worst-case scenario, examine $O(kM \log M)$ modifications while the state of the art previously required $O(kM^2)$ such examinations. Our experiments demonstrate that this improvement in worst case complexity translates into dramatic increases in scalability on real-world data.

Our prioritized algorithm also applies to other decomposable evaluation metrics such as the KL-divergence [16] and MDL/MML scores [1, 20], which further broadens the interest for our algorithm.

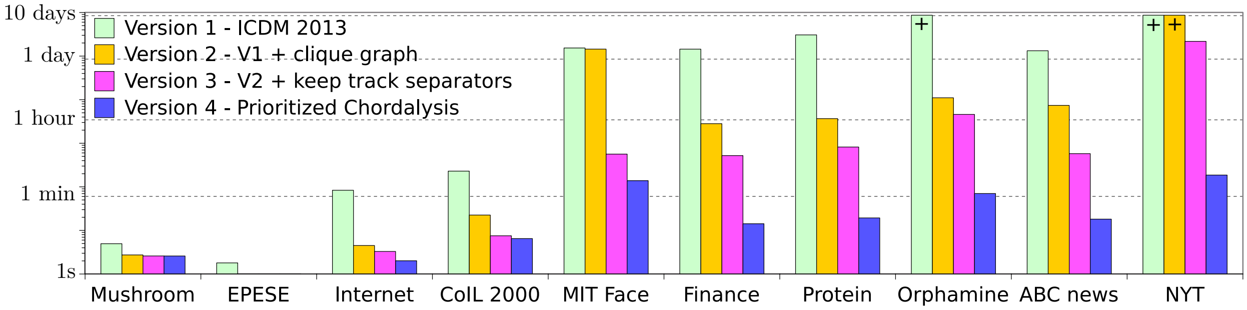

Figure 4: Comparison of the computation time required to perform LLA on various real-world datasets. "+" indicates that the computation did not finish within 10 days of computation.

7 Acknowledgement

This research has been supported by the Australian Research Council under grant DP120100553 and by the Air Force Office of Scientific Research, Asian Office of Aerospace Research under contract FA2386-15-1-4017. We would like to thank Wray Buntine for his comments and help with the news' datasets, and Jinjing Li for providing the curated ABC news collection.

References

[1] S. Altmueller and R.M. Haralick, *Approximating high dimensional probability distributions*, in IEEE Int. Conf. on Pattern Recognition, 2004, pp. 299–302.

[2] ———, *Practical aspects of efficient forward selection in decomposable graphical models*, in IEEE Int. Conf. on Tools with Artificial Intelligence, 2004, pp. 710–715.

[3] K. Bache and M. Lichman, *UCI machine learning repository*. http://archive.ics.uci.edu/ml, 2013.

[4] A. Berry and R. Pogorelcnik, *A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph*, Information Processing Letters, 111 (2011), pp. 508–511.

[5] R. Christensen, *Log-Linear Models and Logistic Regression Second Edition*, Springer, 1997.

[6] C. Dahinden, M. Kalisch, and P. Bühlmann, *Decomposition and model selection for large contingency tables*, Biometrical Journal, 52 (2010), pp. 233–252.

[7] A. Deshpande, M. Garofalakis, and M.I. Jordan, *Efficient stepwise selection in decomposable models*, in Uncertainty in Artificial Intel., 2001, pp. 128–135.

[8] Sandhaus E., *The New York Times Corpus*. https://catalog.ldc.upenn.edu/LDC2008T19, 2008.

[9] P. Galinier, M. Habib, and C. Paul, *Chordal graphs and their clique graphs*, in Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, Springer, 1995, pp. 358–371.

[10] V. Gogate, W.A. Webb, and P. Domingos, *Learning Efficient Markov Networks*, in Advances in Neural Information Processing Systems, 2010, pp. 748–756.

[11] Shelby J Haberman, *The analysis of frequency data*, University of Chicago Press, 1974.

[12] S. Hettich and S.D. Bay, *UCI KDD archive*, 1999.

[13] J.A. Irving, R.N. Pike, A.M. Lesk, and J.C. Whisstock, *Phylogeny of the serpin superfamily: Implications of patterns of amino acid conservation for structure and function*, Genome Research, 10 (2000), pp. 1845–1864.

[14] P.S. Kumar and C.E.V. Madhavan, *Minimal vertex separators of chordal graphs*, Discrete Applied Mathematics, 89 (1998), pp. 155–168.

[15] S.-I. Lee, V. Ganapathi, and D. Koller, *Efficient Structure Learning of Markov Networks using ℓ_1-Regularization*, in Advances in neural Information processing systems, 2006, pp. 817–824.

[16] F.M. Malvestuto, *Approximating discrete probability distributions with decomposable models*, IEEE Transactions on Systems, Man and Cybernetics, 21 (1991), pp. 1287–1294.

[17] MIT Center For Biological and Computation Learning, *CBCL Face Database #1*. http://www.ai.mit.edu/projects/cbcl, 2000.

[18] Orphanet, *An online database of rare diseases and orphan drugs*. http://www.orpha.net, 2014.

[19] F. Petitjean, *Supporting website*. http://www.francois-petitjean.com/Research/SDM2015/, 2014.

[20] F. Petitjean, L. Allison, G.I. Webb, and A.E. Nicholson, *A statistically efficient and scalable method for log-linear analysis of high-dimensional data*, in IEEE Int. Conf. on Data Mining, 2014, pp. 480–489.

[21] F. Petitjean, G.I. Webb, and A.E. Nicholson, *Scaling log-linear analysis to high-dimensional data*, in IEEE Int. Conf. on Data Mining, 2013, pp. 597–606.

[22] J.O. Taylor, R.B. Wallace, A.M. Ostfeld, and D.G. Blazer, *Established Populations for Epidemiologic Studies of the Elderly, 1981-1993*. http://dx.doi.org/10.3886/ICPSR09915, 1998.

[23] P. van der Putten and M. van Someren, *A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000*, Machine Learning, 57 (2004), pp. 177–195.

[24] M.J. Wainwright, P. Ravikumar, and J.D. Lafferty, *High-dimensional graphical model selection using ℓ_1-regularized logistic regression*, in Advances in Neural Information Processing Systems, 2007, pp. 1465–1472.

[25] X. Wu, D. Barbará, and Y. Ye, *Screening and interpreting multi-item associations based on log-linear modeling*, in Int. Conf. on Knowledge Discovery and Data Mining, 2003, pp. 276–285.