Detection of local affinity patterns in big data

Andrea Marinoni, Paolo Gamba

Department of Electronics, University of Pavia, Italy

Abstract

Mining information in Big Data requires to design a new class of algorithms and methods so that the computational complexity load is acceptable and the informat ivity loss is avoided. Information theory-based methodologies can represent a valid option in this sense. In this paper, we introduce a novel method to efficiently detect local affinity patterns within Big Data sets. The proposed architecture, named PROMODE, is based on an undirected bipartite graph representation of the given dataset. Progressively spanning the graph, it is possible to detect all the affinities within the data. It is possible to prove how the proposed framework deliver a computational load that is less than that provided by the other algorithms in literature.

Introduction

Recently, wide-range innovation in information technology field allowed acquisition, collection and storing of a huge amount of data. Moreover, the records typically come from a broad spectrum of data sources and show high variability. The aforesaid properties summarize the fundamentals of the so-called "Big Data" paradigm.

Classic data mining, clustering and classification architectures that approach databases which fulfill those conditions require a pre-processing step s.t. a remarkable dimensionality reduction is performed. However, since very poor *a priori* informations are available for big datasets, standard feature selection and extraction algorithms can not efficiently work.

Therefore, benchmark data processing methods should thoroughly handle the whole amount of data s.t. no informativity would get lost. This requirement results in a very heavy load in terms of computational complexity that delivers consequent latency increase and large memory consumption. Thus, the quality of the considered analysis can be dramatically degraded [1].

The substantial trade-off between process quality and computational cost makes the need of more efficient methods for data processing become urgent. Hence, it is necessary to study, design and develop new data analysis schemes s.t. the complexity load can be strongly reduced without turning their effectiveness mild [1].

Information theory might actually help in that sense. Specifically, information theory-based methods can guarantee to keep the data informativity by using proper information rate metrics and managing the given relationships s.t. the data inference is optimized. Therefore, this kind of structures can provide efficient data analysis tools for feature selection process, high-performance classification algorithms and proper parametric modeling optimization.

Taking a close look to Big Data affinity mining field, information theory can deliver effective identification of affinity patterns while overcoming the issue caused by the lack of prior informations. In fact, the search of affinities over large datasets becomes quickly cumbersome as no efficient dimensionality reduction can be performed. Therefore, as previously mentioned, the quality of the standard affinity mining process can be jeopardized.

In this paper, a new information theory-based method for high-efficiency affinity mining in Big Data is introduced. Specifically, the proposed algorithm aims at delivering a thorough affinity analysis over a given dataset by means of a proper bipartite graph representation of the interactions among the samples. Moreover, this method provides top performance in terms of computational complexity as well.

The paper is organized as follows. Section *System model* introduces the elements of affinity mining and the definitions and notations that will be used throughout the whole paper. Section *Methods* provides the description of classic affinity mining algorithms and of the proposed method for detection of local affinity patterns. Further, Section *Performance results* shows the performance results, whilst Section *Conclusions* delivers the final discussion.

System model

Let us consider a dataset that is composed by P samples. Each sample is characterized by S attributes. Thus, the given dataset can be identified by a $P \times S$ matrix $\underline{\underline{H}} = \{\underline{H}_i\}_{i=1,\ldots,P}$, $\underline{H}_i = [H_{ij}]_{j=1,\ldots,S}$, $H_{ij} \in \Omega \subseteq \mathbf{R}$. A set of L samples shows an *affinity* over the j-th attribute if $\forall (k,l) \in \Lambda \subseteq \{1,\ldots,P\}$, $|\Lambda| = L \leq P$ it fulfills a proper condition that can be summarized as follows:

$$||H_{kj} - H_{lj}|| \leq \tau_j \tag{1}$$

where $\tau_j \in \mathbf{R}_{\geq 0}$ is a given threshold for the j-th attribute.

Thus, an *affinity criterion* can be identified by a set of thresholds $T = [\tau_j]_{j\in\Sigma}$, $\Sigma \subseteq \{1,\ldots,S\}$. Further, *affinity mining*'s goal is to find the samples that fulfill the conditions induced by a given *affinity criterion. Affinity mining* results in the identification of patterns of affinities that show up in some regions of the given dataset. Therefore, a *local affinity pattern* (LAP) is defined as a set Π of $|\Pi| \leq P$ samples that are affine over a set Σ of $|\Sigma| \leq S$ attributes according to a given criterion.

It is worth to note that setting the criteria thresholds does not necessarily imply prior informations. Indeed, each element in a criterion T is a parameter that can be properly tuned according to the desired analysis. E.g., if we define T s.t. $\tau_j = 0 \ \forall j \in J \subseteq \{1, \dots, S\}$, affinity mining delivers the LAPs composed by the samples that show the same exact value over their attributes identified by the indices in J.

Furthermore, the elements of $\underline{\underline{H}}$ can be subject to data cleansing, i.e., inaccurate or irrelevant records w.r.t. a given analysis can be filtered out by the dataset. This pre-analysis step can be performed by setting to a given value the elements of $\underline{\underline{H}}$ that have been cleaned, e.g., $H_{ij} = 0$ if H_{ij} is not relevant. It is important to note that data cleansing do not lead to sparse datasets. In fact, this step does not require specific *a priori* knowledge. Thus, to prevent informativity losses, this process just removes the incorrect data and takes out of the analysis the elements that occur as outliers w.r.t. given distributions that can be properly tuned. Therefore, data cleansing can not be considered as an instance of dimensionality reduction or feature extraction [2].

Throughout this paper, we assume that the datasets have been properly cleaned, i.e., if $H_{ij} = 0$ the i-th sample does not show a relevant record over the j-th attribute. Moreover, without losing generality, we assume that the affinity criteria rely on the local exact match condition, i.e., samples k and l are affine over the j-th attribute if $||H_{kj} - H_{lj}|| = 0$. The methods in Section are based on these assumptions.

Methods

Background

As previously mentioned, several methods have been developed to face affinity mining problems. When performing an affinity search, classic algorithms typically require to draw the transaction graph induced by the given database [3]. In other terms, given a set of P samples, it is possible to progressively outline a depth-P graph where each instance at depth-p identifies one of the $\binom{P}{p}$ combinations of different p samples that can be laid out. Thus, it is possible to arrange the aforesaid graph in a depth-P tree-like way. Fig. 1 depicts a transaction graph for a dataset counting $P = 4$ samples. It is worth to note that the definition of sample combination differs w.r.t. the application that is considered. Specifically, when affinity mining is performed, a sample combination in the aforesaid graph represents a LAP.

In order to provide a computationally efficient affinity search, frequent subgraph mining algorithms (FSMAs) [3] are widely used. These methods take advantage of all possible prior knowledge s.t. the transaction graph can be properly pruned. This step is performed depending on the frequency parameter, which can be properly set according to the considered problem. Fig. 1 reports what happens to the graph in Fig. 1 when FSMAs apply to it, assuming that sample A is marked as infrequent. Basically, if sample A does not comply with the required frequency condition, the whole subgraph generated from the A-node (red in Fig. 1) can be peeled out of the analysis. Therefore, the computational costs for this search are less than those required by the thorough analysis over the whole transaction graph.

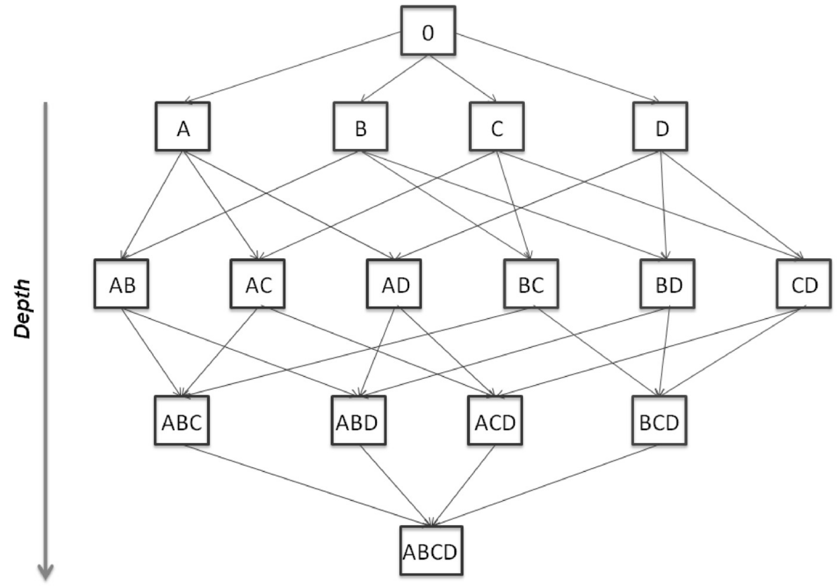

Figure 1: *Transaction tree for a dataset which counts $P = 4$ samples, namely A, B, C and D. 0 represents the root of the tree. The red branches and nodes are filtered out of the search by frequent subgraph mining algorithms (FMSAs) if sample A is marked as infrequent.*

FSMAs actually provide good trade-off in terms of computational complexity and mining performance. However, when poor or no *a priori* informations are available, FSMAs can not deliver efficient results. Further, if the aforementioned pruning step is performed when only inadequate knowledge of the dataset and/or of the desired outcomes is supplied, informativity loss can occur.

Thus, in such conditions, a thorough search throughout the whole transaction tree provided by depth-first search (DFS)-based methods is necessary to prevent any important information leakage. On the other hand, DFS is not computationally efficient at all. Several instances of optimized DFS algorithm have been proposed in literature, such as iterative deepening DFS or lexicographic ordered DFS. However, these algorithms work well in specific mining, i.e., when some prior knowledge is available.

Another approach for affinity mining can be based on minimum-cost path search over transaction graph using Dijkstra algorithm [4]. Specifically, Dijkstra algorithm finds the shortest (i.e., minimum-cost, according to a given metric) path starting from a given graph vertex. Hence, tuning the cost parameter w.r.t. the desired affinity mining over the given dataset, it is possible to find the LAPs starting from a given sample node. Therefore, Dijkstra algorithm, as DFS, does not require prior knowledge to work. However, it gets computationally efficient as the search dataset size gets smaller.

The aforementioned issues make the design of a new affinity mining method that can efficiently (both in terms of search performance and computational costs) work over datasets with no *a priori* informations become necessary. The following Section introduce the method we propose.

Progressive molecule detection (PROMODE)

Let $\underline{\underline{H}}$ be a dataset which consists of P samples that show S attributes. Let us assume $\underline{\underline{H}}$ to be properly data cleaned, as in Section . Thus, it is possible to draw an undirected bipartite graph that identify the given dataset. Specifically, two classes of nodes, named *p-nodes* and *s-nodes*, represent each sample and attribute in $\underline{\underline{H}}$, respectively. In this paper, we alternatively use the p-nodes - s-nodes and sample-attribute notations. The edge that links the i-th p-node to the j-th s-node identifies the dataset element H_{ij}. Hence, if H_{ij} has been considered not relevant by data cleansing, no edge connects p-node i to s-node j. Otherwise, the edge shows a *weight* H_{ij}. Further, let us define the *degree* of the i-th p-node d_{p_i} as the number of relevant attributes that the i-th sample shows, i.e., $d_{p_i} = \sum_{j=1}^{S} \chi(H_{ij})$, where $\chi(z) = 1 \leftrightarrow z > 0$. Analogously, the *degree* of the j-th s-node d_{s_j} can be defined as the number of samples that show relevant data over the j-th attribute, i.e., $d_{s_j} = \sum_{i=1}^{P} \chi(H_{ij})$.

E.g., let us consider a dataset $\underline{\underline{H'}} = \left\{H'_{ij}\right\}_{(i,j)\in\{1,\ldots,P\}\times\{1,\ldots,S\}}$, $P = 6$, $S = 11$, $H'_{ij} \in \{0,1,2\}$, where $H'_{ij} = 0$ if the j-th attribute of the i-th sample has been called as not relevant. $\underline{\underline{H'}}$ is reported in Fig. 2: p-nodes are shown as red squares, whilst s-nodes are identified by blue circles. Taking into account the quantities that have been previously introduced, it is possible to note that in this dataset $d_{p_1} = 5$ while $d_{p_4} = 8$: on the other hand, $d_{s_1} = 2$ while $d_{s_3} = 4$. Further, Fig. 2 shows the bipartite graph that is induced by $\underline{\underline{H'}}$. It is possible to notice that the width of the edges changes according to their weights. Specifically, the edges that have a weight set to 2 show a width that is thicker than that showed by the edges whose weight is 1.

From this new bipartite graph representation of a dataset, in order to efficiently identify the LAPs, we propose to use a different approach w.r.t. those introduced in Section . Basically, we aim at producing a thorough LAP detection analysis for each p-node of the bipartite graph. Since the bipartite graph can be seen as a polymer of two atoms (p-nodes and s-nodes), it is possible to identify by analogy LAPs as molecules in this structure. Therefore, we provide a progressive molecule detection algorithm, which is named PROMODE.

$$\underline{\underline{H'}} = \begin{bmatrix} 2 & 2 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 2 & 2 & 1 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 1 \end{bmatrix}$$

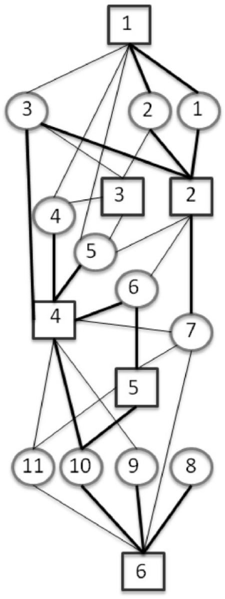

Figure 2: *Bipartite graph representation of dataset* $\underline{\underline{H'}}$

Thus, let us assume we want to find the LAPs that involve the i-th p-node. First, we work on cutting all the edges that surely do not lead to a LAP. In this step, we observe that if the links of a given s-node to two p-nodes are not affine, then that s-node can not be involved for sure in a LAP. Thus, if $\forall j = 1, \ldots, S$ each H_{kj} is not affine to H_{ij}, then the k-th p-node can not be involved in any LAP entailing p-node i. Further, we work on every possible combination of the surviving p-nodes so that it is possible to sieve the LAPs. Specifically, the s-nodes which identify no affinity attributes w.r.t. the given combination of p-nodes involving p-node i are filtered out. Moreover, it is important to note that it is possible to reduce the computational costs by taking track of the cut branches. Finally, we progressively iterate this procedure until all the p-nodes have been analyzed.

Performance results

As it has been introduced in the previous Section, Dijkstra-based method, DFS-based algorithm and PROMODE can provide exhaustive search of LAPs over a given dataset with no prior informations. However, the computational cost required by each procedure is very different. Thus, in this Section, we deliver an estimate of the computational complexity implied by each algorithm by computing the number of total required operations over a P-samples S-attributes dataset.

Specifically, when we consider Dijkstra-based methods, we have to first compare each couple of samples. Then, we must work on every possible combinations of these pairs to find the minimum-affinity-cost path among the samples in the induced graph. Hence, $\mathcal{C}_{\texttt{DIJ}}$, i.e., the number of total operations required by a Dijkstra-based method, can be written as follows:

$$\mathcal{C}_{\texttt{DIJ}} = S \cdot \sum_{j=2}^{\binom{P}{2}} \binom{\binom{P}{2}}{j} \tag{2}$$

On the other hand, when considering a DFS-based procedure, we must draw the transaction tree of all possible combinations of samples up to depth P in order not to avoid any LAP from the detection. Therefore, if we define $\mathcal{C}_{\texttt{DFS}}$ as the number of total operations required by a DFS-based method, we can write the following equation:

$$\mathcal{C}_{\texttt{DFS}} = S \cdot \sum_{j=2}^{P} \binom{P}{j} \tag{3}$$

Apparently, $\mathcal{C}_{\texttt{DIJ}} > \mathcal{C}_{\texttt{DFS}}$, i.e., DFS-based algorithms should outperform Dijkstra-based methods in terms of computational complexity for exhaustive LAP detection. However, $\mathcal{C}_{\texttt{DFS}}$ identifies a very large amount of required operations, which directly converts into latency and memory consumption. PROMODE procedure can overcome this issue. In fact, by means of the analysis over the bipartite graph representation, we reduce the computational cost by carefully selecting the edges which can lead to LAPs. Thus, we can define $\mathcal{C}_{\texttt{PMD}}$, which is the number of required operations by PROMODE architecture, as follows:

$$\mathcal{C}_{\texttt{PMD}} \leq \sum_{i=1}^{P} d_{p_i} \cdot \left[(P-1) + \sum_{j=1}^{d_{m_i}} \binom{d_{m_i}}{j} \right] \tag{4}$$

where d_{p_i} is number of relevant attributes that the i-th sample shows and d_{m_i} identifies the number of p-nodes which are somehow affine to the i-th p-node. Hence, it is possible to prove that $\sup(\mathcal{C}_{\texttt{PMD}}) < \mathcal{C}_{\texttt{DFS}}$.

Conclusions

In this paper, a novel architecture for searching of local affinity patterns in Big Data is introduced. The aforementioned method, named PROMODE, aims at detecting all the affinities in a given dataset taking advantage of a proper representation of the analyzed database in terms of undirected bipartite graph. It has been shown how PROMODE actually outperforms the other algorithms in literature in terms of computational complexity. Thus, PROMODE might represent a valid option for Big Data applications, in order to provide efficient mining with low computational cost while preventing any information loss.

References

[1] D. Lazer, R. Kennedy, G. King, and A. Vespignani. The parable of google flu: traps in big data analysis. *Science*, 343:1203–1205, March 2014.

[2] S. Wu. A review on coarse warranty data and analysis. *Reliability Engineering and System*, 114:1–11, 2013.

[3] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 0:1–31, 2004.

[4] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.