Internship Report, Descriptive Clustering and Highlighting cells in OLAP query

Antoine Chédin
University of Tours, France
antoine.chedin@gmail.com

June 2019

Abstract

This internship report describe all the work I have currently did during my Master degree internship. The work take place in the context of the *Beyond Roll-Up's and Drill-Down's: An Intentional Analytics Model to Reinvent OLAP* research. This research define a new OLAP model for including user intentions as operator for data exploration, and a highlight algorithm for OLAP query results. The main challenges was to first find a parameter free algorithm to use along side the highlight algorithm defined in the research paper. And secondly to implement the algorithm on a platform in order to run it and to begin experimentation around it. In this report, I first describe and explain the research paper work in order to define the context of the work I did. Next, I present my parameter free clustering algorithm and the researches I did on knee detection that I need to automatically define the right number of cluster in a dataset. Then I provide detailed explanations about the highlight algorithm and how it was implemented. And finally I speak about the test platform I developed for testing both the clustering and the highlight algorithm.

1 Introduction

This paper is my internship report at the *University of Tours* for the last year of my Master degree in computer science. My internship has started in March and will finished in July. I mostly worked with Veronika Peralta and Patrick Marcel who supervised me during the internship. I have been working on a paper they were working on: *Beyond Roll-Up's and Drill-Down's: An Intentional Analytics Model to Reinvent OLAP* [1]. The purpose of this research is to define a new data model for online analytical processing (OLAP) called *Intentional Analytics Model.* To understand how it work let's do a quick reminder about OLAP.

The debate around artificial intelligence, especially in Machine Learning, and if data analysis can be fully automated is quite intense these days. In data warehousing community, there's a long tradition of having the decision maker at the center of the data analysis process. Whereas in automated algorithm application, data warehousing, since the beginning, has been all about facilitating the task of interactive exploration of a data space.

OLAP is an approach to answer multi-dimensional analytical queries with short computing time. Typical application of OLAP take place in business intelligence which also include relational databases management and data mining in order to provide historical, current and predictive view of business operations. Before OLAP, people would be working with relational queries and record sets returned by these queries [2]. This side of business intelligence was very DBMS-oriented where the main focus was to optimized databases for intense computing queries. But quickly both industry and scientific community understood that it is possible to simplified exploration by providing a simpler view of the data. Rather than working with a traditional database, user would work with cubes, providing an elegant simplification of data while hiding complexity of the underlying database. It is based on basic analytical operator such as roll-up, drill-down and slicing and dicing [3], providing easier way to navigate the multidimensional data space. OLAP is widely used in marketing, business process management, forecasting, financial reporting, budgeting and other similar field. Today the raise of artificial intelligence (AI) bring new way of automation that can be included in OLAP. Some industry have already integrated AI in their OLAP reporting tools which give the user new insight about the data and queries.

The main goal of [1] is to define a new data model for OLAP called *Intentional Analytics Model* by adding new operators along side the basic ones. These operators would be used by the user to give to system their intention about the query namely, describe, assess, explain, predict, and suggest.By giving their intention, the system would be able to personalize the query result, and would use data mining algorithm to find insights, data correlation and structure that are relevant for the user depending of their original intention. Table 1 come from [1] work and presents a list of different model that can be used for each intention.

For this report, we will only focus our work to the **description** intention. Research [1] describe an algorithm for highlighting cells in OLAP query result for description intention. The idea of this algorithm is to find **surprising** cell, given the previous query, and to highlight them to quickly catch user attention about these particular cells. This highlighting algorithm would helping user during their data space exploration, automatically finding surprising cells. Surprise can be defined in many different ways, it could be an outlier, a cluster of data with similar feature, the top-k cells, or more. In order to reach this goal, the algorithm rely on data mining to find cluster of cells, compute their surprise and highlight the cluster with the highest surprise. More details and concrete implementation are shown in section 3.2.

In this report we're going to work on an automatic descriptive clustering algorithm in section 2. This clustering is based on knee detection and we're also going to present two different approach to this problem. Then in section 3.1 we're going to make experimentation and test on the descriptive algorithm.

Name	Input signature	Output signature
	Model types for description	
Top-K	(Number of values, Name of measures)	(Rank)
Outlier	(Threshold, Name of measure)	(Outlierness)
Clustering	(Number of clusetes, Name of measure)	(Cluster_1, ..., Cluster_n, Representative)
Shrink	(Number of cells, Name of measure)	(Cell_1, ..., Cell_n)
Dominating Slice	(Name of measure)	(DomSlice_1, ..., DomSlice_n)
	Model types for assessment	
KPI	({Label rules}, Name of measure)	(Assessment)
Function-based Benchmark	{*function parameters*}	(Discrepancy)
	Model types for explanation	
Correlation	(Threshold name, Name of measure)	(Participation)
Regression	(Threshold, Name of measure)	(Discrepancy)
Decision Tree	({(Range, Label)}, {(Attributes)}, Name of measure)	(Label)
Statistical test	(Threshold, Name of measure)	(Discrepancy)
	Model types for prediction	
Auto-Regression	(Threshold, Name of measure)	(Discrepancy)
Time series Decomposition	({Threshold}, Name of measure)	(Trend, Seasonality, Noise)
	Model types for suggestion	
Content-based	(Number of queries)	(QueryId)
Collaborative	(Number of queries)	(QueryId)
Hybrid	(Number of queries)	(QueryId)

Table 1: A group of model types, organized per intention. The term *Name of measure* refers to the fact that data in cube come from *Measures* and the model is applied only on one of them

Section 3.2 will provides more explanations and an implementation of the highlighting algorithm. And finally section 4 will provides additional work on a concrete usage of both algorithm.

2 Descriptive Clustering

As we saw, the highlight algorithm relies on data mining model for computing surprise and sectioning cells to highlight. In [1] research they propose a set of descriptive model (see Table 1, describe subsection) to cover kind of cell selection. The idea is to use them all and only keep the one that provides the best surprise. But all of these model comes with a major drawback, they highly depends of a strong parameter and so, are quite complicated to implement into an **automatic** system.

- Top-K relies on the parameter K which defines the size of the result set. This model is straight forward and entirely relies on this parameter.
- Outlier model often relies on a threshold parameter that defines boundaries between outliers and the rest of the data. This parameter is extremely important and even a small changes may produce different results.
- Clustering model like K-Mean or agglomerative clustering highly depends of a K parameter that will defines how many cluster the model have to produce. Again this parameter have a big impact on produced clusters

To resolve this problem, I propose descriptive clustering algorithm ($DesC$). It's a parameter free algorithm that capture sample distribution in a dataset returning multiple cluster. The main goal of $DesC$ is to describe the data through clustering. instead of using multiple data mining model (Top-K, outlier detection, clustering, etc) $DesC$ regroup dense linked sample together letting outlier and aberration in their own cluster. Note that the objective is NOT to guess the "real" number of cluster, like many research on automatic clustering algorithm. For example, in a dataset generated by two Gaussian with different center in the data space, one may say that the good result for an automatic clustering would be to find two cluster. That's not our case, our goal is to describe the data, in this way, if some point are alone is the data space, they have to be considered as outliers.

To achieve that I use a hierarchical clustering with a single-linkage criteria meaning that distance between cluster are measured by the two closest point of both cluster. Single linkage criteria allows to form dense cluster independent of its shape where other criteria would split up the cluster if it grew to long. But this is not enough. Indeed classical hierarchical clustering also requires a parameter to produce clusters. First each point are assigned to a unique cluster. In a dataset with n points it begin with n different cluster. The distance between two cluster during this phase is referred as the merging cost. Then the two clusters with the smallest merging cost are merge together and so on until the

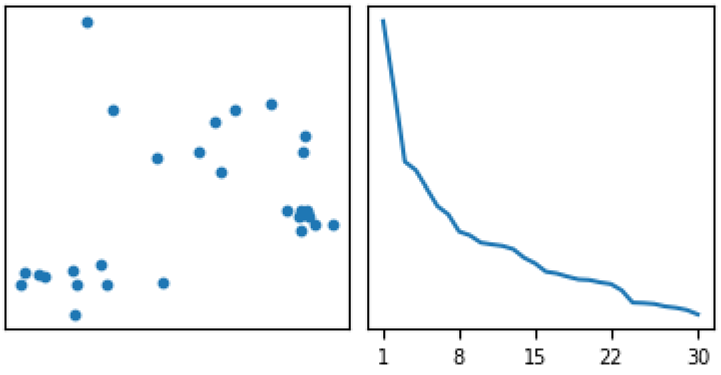


Figure 1: A simple merging cost evaluation graph along side the dataset that built it.

merging cost exceed a threshold parameter. Then the algorithm stop and return all clusters formed. Like for other descriptive algorithm, hierarchical clustering highly depends on this threshold parameter. To make our *DesC* completely autonomous, we need to automatically find the best threshold value for the dataset.

This can be performed by finding the knee of the evaluation graph of the algorithm. An evaluation graph is a two dimensional plot where the x-axis is the number of cluster produced and the y-axis the evaluation metric of a clustering considering x clusters. The y-axis values can be any metric such as similarity, error, silhouette score [4], completeness, merging cost, etc. For other clustering approaches like K-Means or spectral clustering, building the evaluation graph requires to run the clustering algorithm multiple times for each x, which is quite inefficient. But with hierarchical clustering algorithm, where each step consist in either merging clusters at each step, the evaluation graph with the merging cost as the evaluation metric, can be produce be running the algorithm only once. In hierarchical clustering, since merging cost constantly increase, evaluation graph often look like a L-shape curve with a more or less defined knee like in figure 1. Here we're making the assumption that the best merging cost threshold is at the curve knee, where the curve switch from a a sharp slope to a low decreasing line.

So to find the right threshold, we need to find the automatically find the knee in the evaluation graph of the hierarchical clustering. There exists a standard closed-form $K_f(x)$ [5] that defines the curvature of f at any point at any point as a function of its first and second derivative. The knee is the point with the maximum curvature of the function.

$$K_f(x) = \frac{f''(x)}{(1 + f'(x)^2)^{1.5}} \tag{1}$$

Unfortunately, while curvature is well defined for continuous functions, it's not the case for discrete dataset. We could try to determine the curvature by fitting a continuous function to the dataset we're using in order to find the point of maximum curvature, but this method is difficult, especially if the data is noisy. Moreover it may not be very efficient, since the curvature at any point of a function depend on the entire function, including points outside the bounds of the dataset. Thus, the point with maximum curvature may fall outside the dataset valid range. In conclusion, using the closed-form equation 1 as a direct basis for knee detection is not relevant.

We're now going to present two solutions from the literature, L-method [6] and Kneedle [5]. Both paper propose a new way to find the knee in a curve of discrete data.

2.1 L-method

L-method define in [6] was created specifically for that purpose finding the right number of Cluster/Segment in Hierarchical Clustering/Segmentation algorithms. It's a global approach, meaning that it's robust to outliers and discrete values. In order to find the knee, this method use a common property of the evaluation graph: the left part is a sharp slope and the right a flat line, both are approximately linear. If two line are fitted on the curve, one on the left part and the other on right, we're able the created a L shape and easily find the knee. To create these two line, one must find the pair of line that will most closely fit the curve.

Considering a common evaluation graph where the x-axis values varies from 1 to n. Let's L_k and R_k be the left and right sequence of data point partitioned at $x = k$. That is L_k has all point from 1 to k and left from $k+1$ to n. Equation 2 defines the total root mean squared error of both line when their partition is at $x = k$

$$RMSE_k = \frac{k}{n} \times RMSE(L_k) + \frac{n-k}{n} \times RMSE(R_k) \tag{2}$$

where $RMSE(L_k)$ is the root mean squared error of the left fitted line (same goes for $RMSE(R_k)$). The goal is to get the k value with the minimal fitting error.

$$k_{min} = argmin(RMSE_k) \tag{3}$$

Limitations

However L-method present some limitation. Indeed, since $RMSE_k$ is computed by weighting both line by the number of point used for the fitting, it may give wrong result when it's applied on a dataset with a high number of sample in the right part of the curve. In such case, the few point on the left become statically irrelevant. This method work better when there an equivalent number of point on both side of the knee.

This issue is identified in paper [6], and they propose an improvement to their algorithm in order to fix it. The idea is to recursively trim a portion of the right part of the graph until the knee found before and after the cut is the same.

This solution may work in the original paper, it seems that it brings new kind of issues. For example, with certain kind of graph with a very smooth knee, it happened during our experimentation that the trimming remove every point of the dataset, not knowing when to stop. There is still some improving needed to this solution.

There is also an issue on very small dataset (less than 10 points) that was not identified in the original paper. It happens that evaluation graph don't form a L-shape curve (like in figure 2). In such case, L-method may fit on what I call an "anti-knee". This issue could be avoided by adding a constraint to the algorithm in order to force the two line to nit form an anti-knee (by forcing the angle between the left and the right line to be less than 180 degree for example).

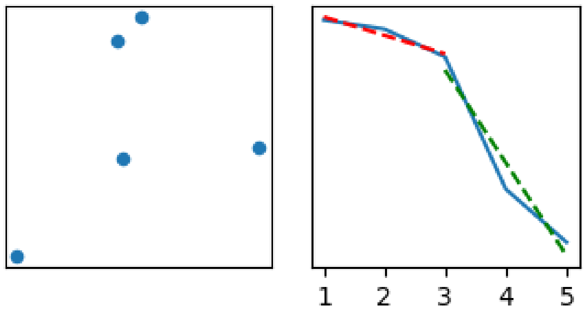


Figure 2: A small dataset with 5 point and it's evaluation graph on the right. The red and the green dashed lines represent the L-method best fit for this evaluation curve. A correct knee detection should probably detect a knee at $x = 4$. However, L-method will returned a detected knee at $x = 3$

2.2 Kneedle

Kneedle [5] is a global and discrete approach to knee detection. It's based on the notion that the point of maximum curvature in a curve is the point that is the farthest of the diagonal line formed by the points (x_0, y_0) and (x_n, y_n) with n the number of point in the dataset. Using this diagonal line is very interesting because it allows to keep the global behavior of the curve during the analysis. For example, using a fitted line instead may cut off end point due to a higher concentration of point in the middle of the curve.

In the original paper [5], they use the perpendicular distance to the diagonal line, and select the point with the highest distance. In our work I find that using the perpendicular distance, which is the "normal" way of computing distance between a point and a line, is not required at all and is equivalent to using an θ-distance, a distance computed by measuring the length of the segment between a point and a point on the line in such a way that the segment and the line from

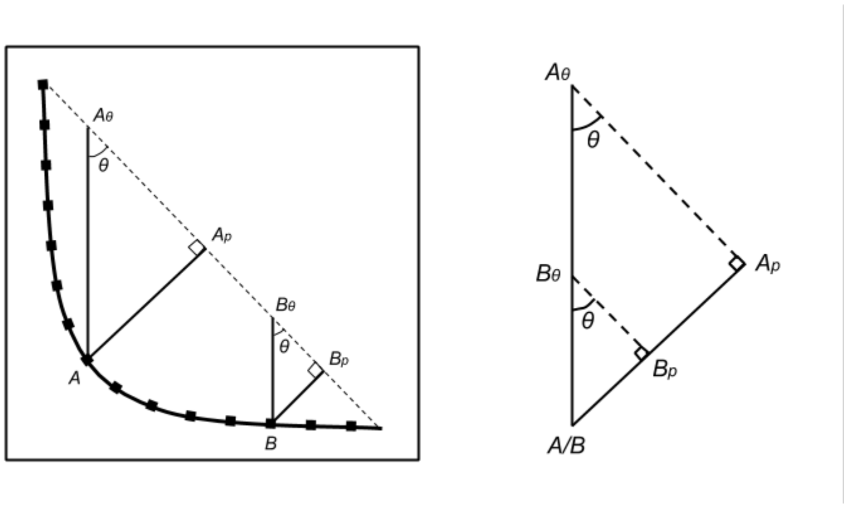


Figure 3: Graphical demonstration with the Intercept theorem on a simple evaluation curve

an θ angle. This small modification will allow us to improve the algorithm later on.

Here's the demonstration. First let's D be the diagonal line formed by the points (x_0, y_0) and (x_n, y_n) with n the number of point in the evaluation curve. Let's take two point A and B on the evaluation curve and two point A_p and B_p on line D in such a way that both line (A, A_p) and (B, B_p) are perpendicular to D. Points A and B must be choose in a way that $d(A, A_p) > d(B, B_p)$ with $d()$ the length of the line. Now that A and B are set, we find points A_θ and B_θ on line D in such a way that $A\widehat{A_\theta}A_p = B\widehat{B_\theta}B_p = \theta$. We now have two triangles, (A, A_θ, A_p) and (B, B_θ, B_p), with the same shape, allowing us to use the Intercept theorem. Equation 4 show that if $d(A, A_p) > d(B, B_p)$ then $d(A, A_\theta) > d(B, B_\theta)$

$$\begin{aligned}
&\begin{cases} \dfrac{d(A, A_p)}{d(B, B_p)} = \dfrac{d(A, A_\theta)}{d(B, B_\theta)} \\ d(A, A_p) > d(B, B_p) \end{cases} \\
\iff\ & \frac{d(A, A_p)}{d(B, B_p)} > 1 \\
\iff\ & \frac{d(A, A_\theta)}{d(B, B_\theta)} > 1 \\
\iff\ & d(A, A_\theta) > d(B, B_\theta)
\end{aligned} \tag{4}$$

That's mean order of magnitude are kept for each point of the curve. Therefore, using perpendicular distance or θ-distance are strictly equivalent when looking for the knee.

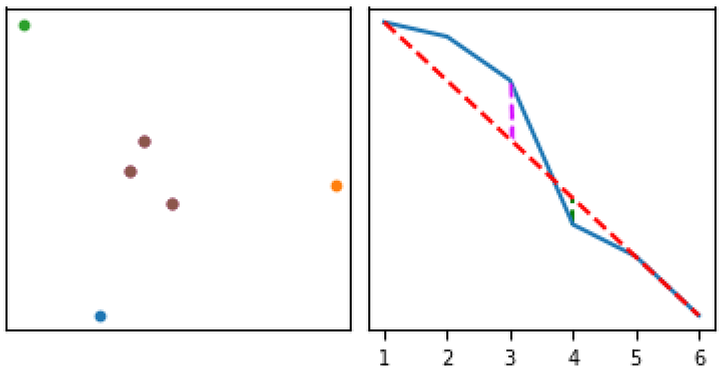


Figure 4: A simple evaluation graph. In light purple the knee detected before the improvement and in green, the new knee found after the improvement, which gives us better a outcome.

To find the knee point in the evaluation curve, if l is the diagonal line formed by the first and the last point of the evaluation graph is defined by the next function l:

$$l(x) = a.x + b \text{ with } a = \frac{y_n - y_0}{x_n - x_0} \text{ and } b = (y_0 - a.x_0) \tag{5}$$

we can then pose the equation 6 to compute the vertical distance v between a point k of the evaluation graph and the diagonal line

$$v_k = |l(k) - y_k| \tag{6}$$

The number of cluster k returned by the method is the k that maximize v_k

Improvements for small dataset:
Kneedle method show good results overall. However, like for section 2.1 with L-method, it happen that evaluation graph doesn't look like a L-shape. It happen sometimes that the evaluation graph is "above" the diagonal line often resulting into a wrong predicted knee. Whereas with L-method it's complicated to detect these cases, there's a very simple solution to that problem with Kneedle. We just need to take equation 6 and remove the absolute function from it. So when the evaluation graph point is higher than the diagonal line, v_k become negative allowing the algorithm to find the correct knee.

$$v_k = l(k) - y_k \tag{7}$$

This small modification allows Kneedle to give good result in regardless the dataset size or it's distribution (not like L-method which is dependent of both). Figure 4 showcase an example on how the result is improved.

3 Proposal

3.1 Finding the knee

3.1.1 Dataset generation

To test our methods and compare their results, we've created some toy dataset specifically for the experiment. Multiple dataset size were chosen (6, 30 and 300 samples). If the dataset were generated using a uniform random function, dataset generated wouldn't have been very interesting to visualize. In order to create "interesting" dataset they were created following some rules.

- First column was generated with the `make_circles(noise, factor)` function from *Scikit-Learn* Python package[1]. This function randomly generate a dataset with a large circle of point, containing a smaller circle of point. Noise parameter, which is the standard deviation of Gaussian noise added to the data, was set to 1.5 and factor parameter, which is the scale factor between the inner and outer circle, was set to 0.2.
- Second column was created using three normal distribution with different center, and parameters for each one.
- Third column was generated with the `make_moons(noise)` function, also from the *Scikit-Learn* Python package[1]. This function randomly generate a dataset with two interleaving half circles. Noise parameter, which is the standard deviation of Gaussian noise added to the data, was set to 1.5.

Figure 5 shows the different dataset that were generated and used for testing. Note that all dataset have been normalized with a `MinMaxScaler` between 0 and 10 for both axis. This can be set to any value since our method isn't affected by the value range of the dataset. This values has been chosen because they were convenient to display. The two method shown in section 2.1 (L-method) and 2.2 (Kneedle) were tested to see how they perform. Figure 6, 7 and 8 show algorithms clustering result along side their evaluation graph for each dataset.

The full code about generating dataset and the test are available on *Github*[2]

3.1.2 Results

About the knee detection, both solution gives similar good results like in figure 7, but L-method still present some drawbacks. First L-method is way much longer to execute than Kneedle. And the bigger the dataset is, the worst it become, whereas Kneedle show a linear computation time increase. Second, there's the shifting knee issue that we saw in section 2.1. We can observe it in figure 8 on the top right graph. The right knee seems to be located at $x = 25$ but the method returned a knee at $x = 62$. And like the previous issue, it only get worse as the size of the dataset grow, making knee detection on big dataset

[1] `https://scikit-learn.org/stable/index.html`
[2] `https://github.com/antoinechedin/descriptive-clustering`

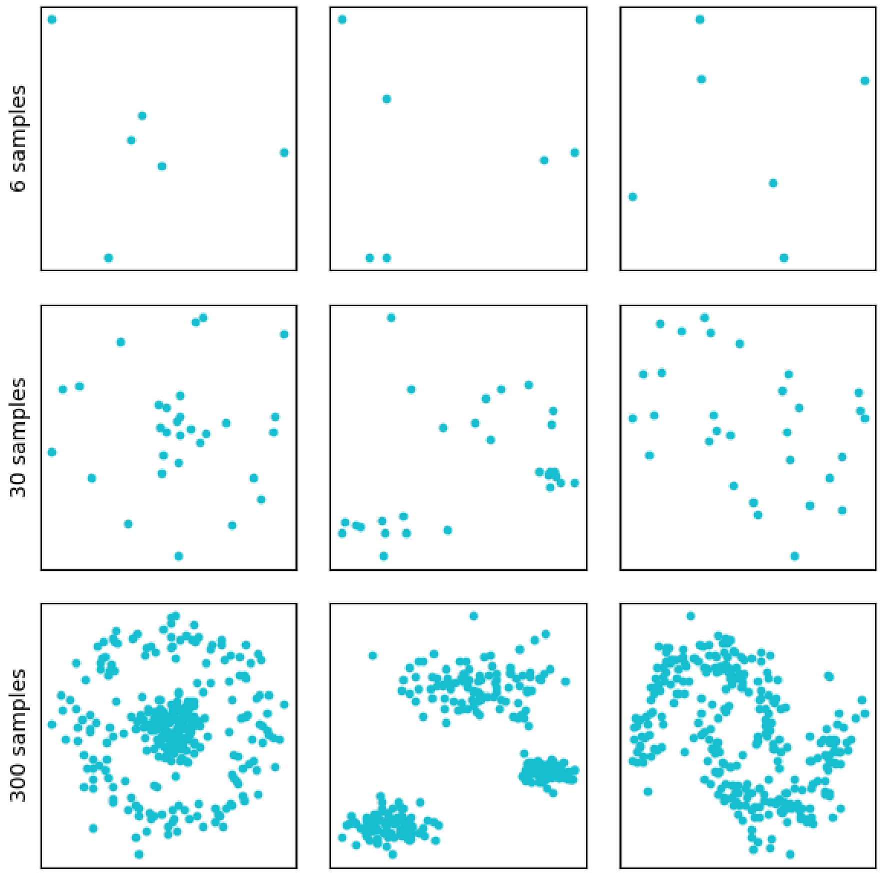


Figure 5: The 9 dataset generated for testing *DesC*

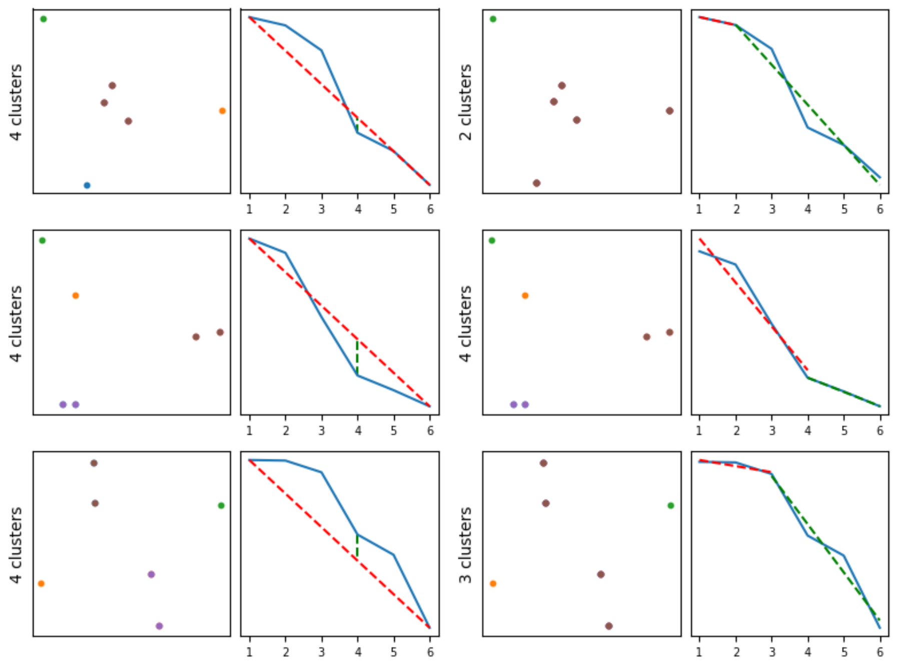


Figure 6: Kneedle and L-method results on the 5 samples dataset. Kneedle is more consistent at finding the "right" amount of cluster. Whereas L-method struggle in top right and bottom right, fall into the anti-knee issue.

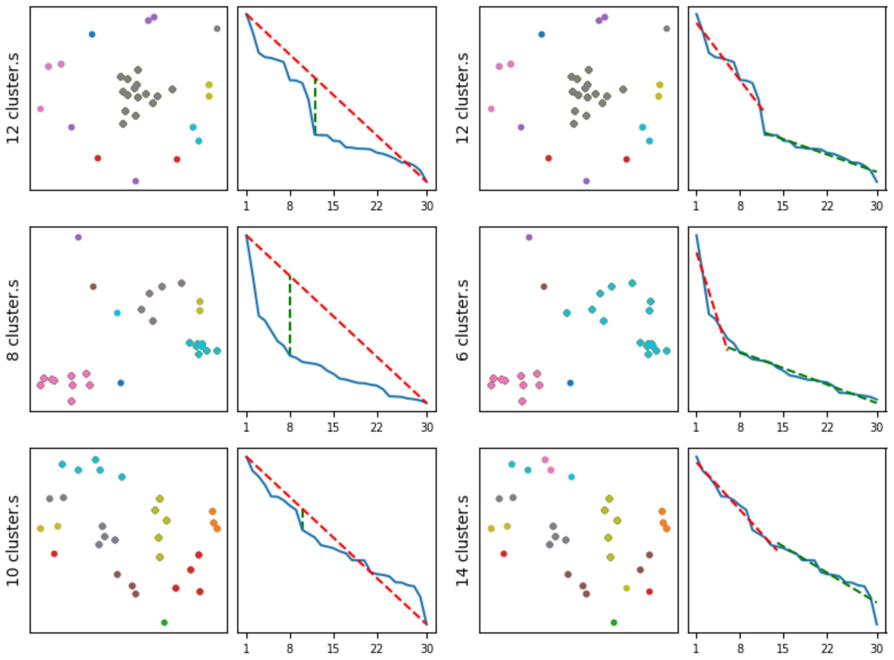


Figure 7: Kneedle and L-method results on the 20 samples dataset. Both method shows very similar good results, and consistently find the knee in the evaluation graphs

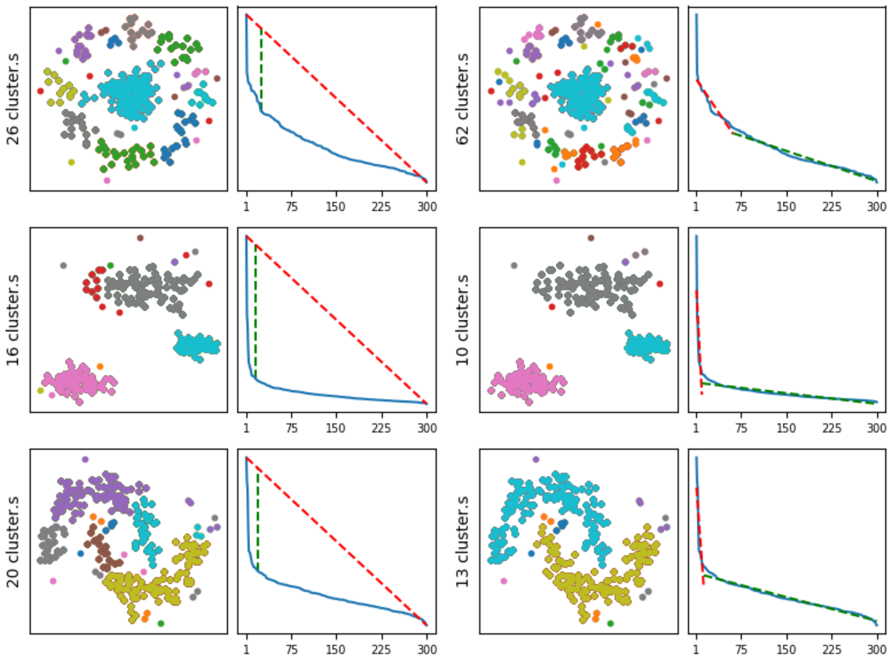


Figure 8: Kneedle and L-method results on the 50 samples dataset. Same conclusion as figure 7, except for L-method in top right graph where the knee is shift to the right due to the long right part (see section 2.1)

very inconsistent. And finally, L-method tends to find anti-knee in small dataset like the ones in figure 6. In top right and bottom right graph, L-method didn't detected the right knee whereas Kneedle did.

For all those reasons, Kneedle is the solution we're going to use for knee detection for the rest of this paper since, it's quicker and have a better consistency.

About *DesC*, the different tests in figure 6, 7 and 8 describe well how *DesC* behave in general. Sample that are alone in the data space are consistently put in a cluster alone and samples very similar feature are put together in a cluster.

3.2 Highlight algorithm

As said in section 1, research [1] has defines a theoretical algorithm to highlight cells in an OLAP query result.

The main goal of this algorithm is to highlight interesting cells to the user so they won't miss them. In the context of "describing" the data (which is the context we decided to focus in this report) they define interestingness by three concept: proxies, significance score and surprise score. Since this algorithm work with sequence of query result, the proxies relation must represent the relationship between the cells of the actual query result and the previous one. This relation may be produced following the cube structure and OLAP basic operators, roll-up's, drill-down's, etc. The significance score is very important as it must characterizes each cell with an objective importance. Finally, the surprise score is the core function of the algorithm. Surprise exploit the fact that cells between the current query result and the previous one are linked by proxies. Thus, they can use it to compare the prior and new belief in the query sequence. The more interesting the comparison is, the higher is the surprise score.

Algorithm 1 unveils as follows. First significance score are computed for each cells of C^O and C^N using the $significance$ function.Then, surprise score are computed for each cells of C^N by contrasting their significance score with the score.s int their relatives in C^O (thanks to $proxies$). Finally, a global surprise score is computed for each models, by aggregating the surprise score of the cells participating to the model component their in.

In order to use this algorithm, one must define how proxies, significance and surprise are computed. Fortunately, research [1] also propose a simple implementation of the core function in order to use the algorithm. We developed a concrete implementation in *Java* of the algorithm, following the advises of research [1].

- Cube old C^O and Cube new C^N are represented by a sequence of MDX queries over a cube. For a sequence of n queries, there's $n-1$ pair of queries that can fulfill the role of C^O and C^N.
- In order to test our descriptive clustering algorithm, and for the sake of simplicity, the set of model M only contain one element, a *DesC* model.

Algorithm 1 Highlight cell selection

```
**DATA:**
cube $C^O$
cube $C^N$
set of models $M = \{M^1, \ldots, M^k\}$
with their components $MC = \{MC_1^1, \ldots, MC_m^k\}$
$n : m$ relationship proxies between the cells of $C^O$ and $C^N$
function *significance* for computing the significance of a cell
function $\mathcal{D}$ for surprise characterization
function $\mathcal{A}^C$ for computing component scores
function $\mathcal{A}^M$ for computing model scores
**RESULT:**
a component among $\{MC_1^1, \ldots, MC_m^k\}$

for all cells $c^O$ in $C^O$ do
  compute $c^O.significance$
end for
for all cells $c^N$ in $C^N$ do
  compute $c^N.significance$
end for
for all cells $c^N$ in $C^N$ do
  compute $c^N.surprise = \mathcal{D}(c^N.significance, c^N.proxies.significance)$
end for
for all component $MC_j^i$ in $MC$ do
  compute $MC_j^i.surprise = \mathcal{A}^C_{c^N \in MC_j^i}(c^N.surprise)$
end for
for all model $M^i$ in $M$ do
  compute $MC^i.surprise = \mathcal{A}^M_{MC_j^i \in M^i}(MC_j^i.surprise)$
end for
Return the component maximizing $MC_j^i.surprise$
```

The set of model component MC is determined by the number of cluster discovered by the $DesC$ model. So if k cluster are found by the model, there will be k model component. Each component show which point belong or not to the cluster it represent.

- Proxies relation are created between cells from C^O and C^N if they're ancestor or descendant. That's mean, C^N is either a drill-down or a roll-up of C^O.
- Significance score sig for each cell is computed via z-score. The goal here is to give high significance score to cells with an high deviation from the mean value of the dataset. We assume that the more the point has an high significance score the more likely it will be highlighted.
- Surprise score is computed by simply subtracting the significance score of a cell from C^N, by the mean significance score of all of its proxy cell in C^O. If a cell don't have any proxies relationship with the previous cube, then the cell is mapped to he entire previous cube.
- Finally, the aggregation function $\mathcal{A}^C$ is a simple average function. That's mean a model component surprise is defined by the average value of all the surprise score of all cell that compose the model component. The model component with the higher surprise score s highlighted. We don't need to define $\mathcal{A}^M$ since we're only using one model instead of a set, so we don't need a function to compare surprise between all the model.

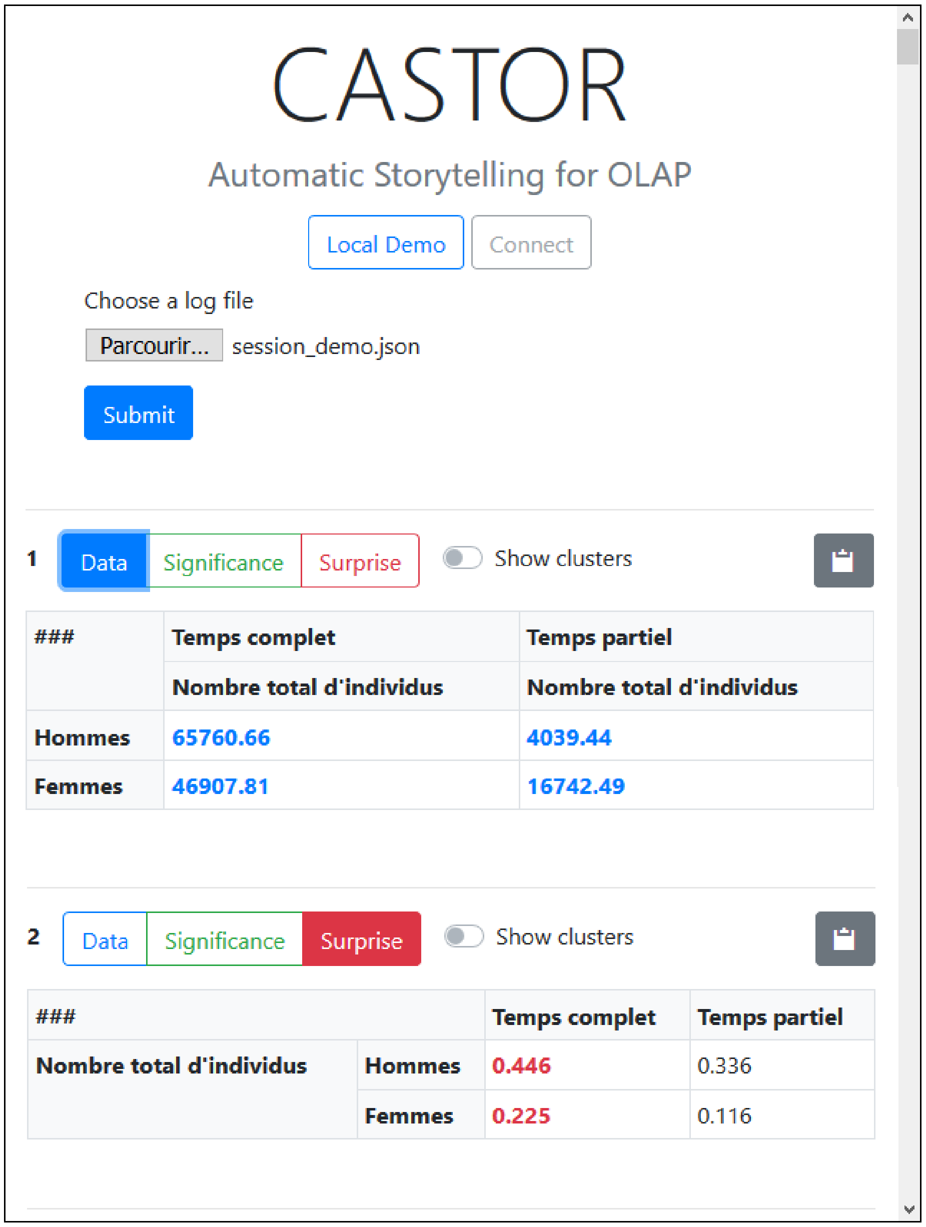
4 Application

Until now, all test we did was performed on generated dataset. They are friendly because it's easy to get them, but often don't represent real use case, where algorithm may behave differently due to data distribution.

That's why we're going to introduce a new dataset *DOPAN*. This dataset is described in [7] and consist of a navigation traces collected in the context of a French project on energy vulnerability. These traces are OLAP sessions regrouping a sequence of MDX request over a cube using *Saiku*[3]. The goal of this project was to ask volunteer student of a Master degree in business intelligence to explore data warehouse and cubes to answer high-level information needs defined by their lecturer. The sessions was then analysed by an expert who annotated each session about what was the user intention about their exploration, and how well they performed.

This dataset is well suited for test the highlight algorithm and *DesC*. The query sequences can be highlighted and be compared with the user intention to see how they match. Each MDX request returns dataset that can be analysed with *DesC*. If the data are split up by measure[4] a single query may even produce multiple 1-dimensional dataset.

[3]https://www.meteorite.bi/products/saiku
[4]measure in term of cube measure


CASTOR

Automatic Storytelling for OLAP

Local Demo
Connect
Choose a log file
Parcourir...
session_demo.json
Submit

1

Data Significance Surprise Show clusters

###	Temps complet	Temps partiel
	Nombre total d'individus	Nombre total d'individus
Hommes	**65760.66**	**4039.44**
Femmes	**46907.81**	**16742.49**

2

Data Significance Surprise Show clusters

###		Temps complet	Temps partiel
Nombre total d'individus	Hommes	**0.446**	0.336
	Femmes	**0.225**	0.116

Figure 9: Current version of *CASTOR* platform. Cells with values in bold are the one highlighted by the highlight algorithm

Instead of retrieving the dataset and preforming static tests, I developed web platform *CASTOR* (Automatic storytelling for OLAP) for evaluating *Intentional Analytics Model*. Build in *javascript*, this platform provide a tool to run and customize *Intentional Analytics Model* and see how it perform. For now most of the feature are still work in progress, but the goal is to allow the user to select, query sequences, significance function, surprise function, proxies function and the set of model to use to have control over the whole process. For now, only one significance, surprise, proxies and model function are implemented, and there's only the query sequences from the *DOPAN* dataset.

CASTOR also provide visualisation tools for a better understanding of each algorithm step and to propose a more friendly way of approaching the data. The different visualisation are:

- Raw data.
- Significance score.
- Surprise score.
- Clustering.

Figure 9 show a first version of the *CASTOR* platform In the future, we will also be able to visualize data and the evaluation graph (if *DesC* clustering is involve), proxies relationship between query result and a step-by-step view of the highlight algorithm.

5 Conclusion

During this internship I worked a lot with Veronika Peralta and Patrick Marcel on OLAP and clustering problem. The main challenge was to find a clustering algorithm with no parameter able of describe a query result on an OLAP system. The second one was to implement algorithm 1 on a real use case in order to test and see its performance, limitations and quality of results.

My main contributions are:

- doing researches on knee detection in literature, and making a survey on two method that I personally found interesting and appropriate for my problem.
- Developing a descriptive clustering algorithm (*DesC*) using both personal knowledge and literature paper.
- Proposing improvements to existing algorithm, and implement them to increase robustness of my descriptive clustering algorithm.
- Implementing algorithm 1 defined in [1] to test storytelling in OLAP and more precisely in the new *Intentional Analytics Model*.

- Developing a web platform *CASTOR* to provide an online and user friendly test environment for *DesC* and algorithm 1.

As future work, I would begin to perform more detail test on *DesC* and more specifically on performance. We found that L-method is slower than Kneedle but I didn't made any precise measure on it. I also didn't had time to test *DesC* on the *DOPAN* dataset. These tests might might reveal flaws that will need to be tweak in order to improve the *DesC* robustness.

Also, for now, only 1 dimensional dataset have been considered. It may be worth trying to explore the query result as a multidimensional data space, with each measure corresponding to a dimension.

References

[1] Panos Vassiliadis, Patrick Marcel, and Stefano Rizzi. Beyond roll-up's and drill-down's: An intentional analytics model to reinvent olap.

[2] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and subtotals. *CoRR*, abs/cs/0701155, 2007.

[3] Sunita Sarawagi. Explaining differences in multidimensional aggregates. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 42–53, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[4] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

[5] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171, June 2011.

[6] S. Salvador and P. Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 576–584, Nov 2004.

[7] Mahfoud Djedaini, Nicolas Labroche, Patrick Marcel, and Verónika Peralta. Detecting user focus in olap analyses. In Mārīte Kirikova, Kjetil Nørvåg, and George A. Papadopoulos, editors, *Advances in Databases and Information Systems*, pages 105–119, Cham, 2017. Springer International Publishing.