Rk-means: Fast Clustering for Relational Data: Supplementary Material

Ryan R. Curtin
RelationalAI
ryan@ratml.org

Benjamin Moseley
Tepper School of Business
Carnegie Mellon University
moseleyb@andrew.cmu.edu

Hung Q. Ngo
RelationalAI
hung.q.ngo@relational.ai

XuanLong Nguyen
Department of Statistics
University of Michigan
xuanlong@umich.edu

Dan Olteanu
University of Oxford
dan.olteanu@cs.ox.ac.uk

Maximilian Schleich
University of Oxford
max.schleich@cs.ox.ac.uk

Abstract

This supplementary material contains details from Rk-means omitted from the main paper due to space constraints.

A Background on Database Queries and FAQs

Recent advancements in the database community have produced new classes of query plans and join algorithms Abo Khamis et al. (2017, 2016); Ngo (2018); Olteanu and Schleich (2016) for the efficient evaluation of general database queries. These general algorithms hinge on the expression of a database query as a *functional aggregate query*, or FAQ Abo Khamis et al. (2016).

Loosely speaking, an FAQ is a collection of *aggregations* (be they sum, max, min, etc.) over a number of functions known as *factors*[3], in the same sense as that used in graphical models. In particular, if there was only one aggregation (such as sum), then an FAQ is just a sum-product form typically used to compute the partition function. An FAQ is more general as it can involve many marginalization operators, one for each variable, and they can interleave in arbitrary way. Every relational database query can be expressed in this way. Consider the example query of Section 1: for this, the task of the database query evaluator is to compute `max(transactions.count)` for every tuple (i, s, t, p, y) that exists in the output. We can express this as a function:

$$\phi(i, s, t, p, y) = \max_c \max_i \max_s \psi_P(i, t, p)\psi_T(i, s, c)\psi_S(s, y). \quad (11)$$

In this we have three *factors* $\psi_P(\cdot)$, $\psi_T(\cdot)$, and $\psi_S(\cdot)$, which correspond to the `product`, `transactions`, and `store` tables, respectively. We define $\psi_P(i, t, p) = 1$ if the tuple (i, t, p) exists in the `product` table and 0 otherwise; we define $\psi_S(s, y)$ similarly. We define $\psi_T(i, s, c) = c$ if the tuple (i, s, c) exists in the `transactions` table and 0 otherwise. Thus, given any tuple (i, s, t, p, y), we can compute `max(transactions.count)`$= \phi(i, s, t, p, y)$.

In order to efficiently solve an FAQ (of which Equation (11) is but one example), the InsideOut algorithm of Abo Khamis et al. (2016) may be used; InsideOut is a variable elimination algorithm, inspired from variable elimination in graphical model, with several new twists. One twist is to adapt worst-case optimal join algorithms Ngo et al. (2018); Veldhuizen (2012) to speed up computations by exploiting sparsity in the data. Another twist is that the algorithm has to carefully pick a variable order to minimize the runtime, while at the same time respect the correctness and semantic of the query. Unlike in the case of computing a sum-product where the summation opeartors are commutative, in a FAQ the operators may not be commutative.

To characterize the runtime of this algorithm, we must first observe that each database query and thus FAQ corresponds to a hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$. The vertices $\mathcal{V}$ of this hypergraph correspond to the vari-

[3] A full formal definition of FAQs can be found in Abo Khamis et al. (2016), but is not required for our work here so we omit it.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

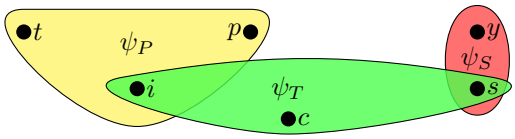

Figure 3: Example hypergraph $\mathcal{H}$ for the example query and FAQ in Equation 11.

ables of the FAQ expression; in our example, we have $\mathcal{V} = \{i, s, t, p, y, c\}$. The hyperedges $\mathcal{E}$, then, correspond to each factor $\psi_P(\cdot)$, $\psi_T(\cdot)$, and $\psi_S(\cdot)$—which in turn correspond to the tables in the database. This hypergraph $\mathcal{H}$ is shown in Figure 3.

Roughly, InsideOut proceeds by first selecting a variable ordering σ, reordering the FAQ accordingly, and then solving the inner subproblems repeatedly, in much the same way variable elimination works for inference in graphcal models Koller and Friedman (2009). The runtime of InsideOut is dependent on a notion of width of $\mathcal{H}$ called FAQ-width, or faqw$(\cdot)$. Fully describing this width is beyond the scope of this paper and we encourage readers to refer to Abo Khamis et al. (2016) for full details. The FAQ-width is a generalized version of *fractional hypertree width* of Grohe and Marx (2014) (denoted by fhtw). When the FAQ query does not have free variables, faqw = fhtw. Given some FAQ with hypergraph $\mathcal{H}$, via Section 4.3.4 of Abo Khamis et al. (2017), InsideOut runs in time $\tilde{O}(N^{\text{faqw}_{\mathcal{H}}(\sigma)} + Z)$, where we assume that the support of each factor[4] is no more than $O(N)$, and Z is the number of tuples in the output. As an example, the hypergraph of Figure 3 has $\text{faqw}_{\mathcal{H}}(\sigma) = 1$. Overall, InsideOut gives us the most efficient known way to evaluate problems that can be formulated as FAQs.

B Missing details from Section 3

B.1 Proposition 3.5

Proof of Prop. 3.5. As before the optimal transport plan from P^{in} to Q is such that each support point $s \in S$ is received by all $x \in \boldsymbol{X}$ nearest to s compared to other points in S. So,

$$\begin{aligned}
&\mathsf{W}_2^2(P^{\text{in}}, Q) + \Omega(M) \\
&= \sum_{j=1}^{m} \mathsf{W}_2^2(M_j, P_j^{\text{in}}) + \Omega(M) && (12)\\
&\le \alpha \sum_{j=1}^{m} (\mathsf{W}_2^2(M_j^*, P_j^{\text{in}}) + \Omega_j(M_j^*)) && (13)\\
&\le \alpha(\mathsf{W}_2^2(M^*, P^{\text{in}}) + \Omega(M^*)). && (14)
\end{aligned}$$

[4]Or in our case, the number of tuples in the table corresponding to that factor.

The second to last inequality is due to the α-approximation of (regularized) $\mathsf{wkmeans}_1$, and condition that $|\text{supp}(M_j)| \ge |\text{supp}(M_j^*)|$. The last inequality follows from Proposition 3.2 and the definition of Ω. By the triangle inequality of W_2, as before

$$\begin{aligned}
&\mathsf{W}_2(P^{\text{in}}, P) && (15)\\
&\le \mathsf{W}_2(P^{\text{in}}, Q) + \mathsf{W}_2(Q, P) && (16)\\
&\le \mathsf{W}_2(P^{\text{in}}, Q) + \sqrt{\gamma \mathsf{W}_2^2(Q, M^*) + \gamma\Omega(M^*) - \Omega(P)} && (17)\\
&\le \mathsf{W}_2(P^{\text{in}}, Q) + \big(2\gamma \mathsf{W}_2^2(P^{\text{in}}, Q) + 2\gamma \mathsf{W}_2^2(P^{\text{in}}, M^*) \\
&\qquad + \gamma\Omega(M^*) - \Omega(P)\big)^{\frac{1}{2}}. && (18)
\end{aligned}$$

Hence, by Cauchy-Schwarz and combining with (14) we obtain

$$\begin{aligned}
&\mathsf{W}_2^2(P^{\text{in}}, P) \\
&\le 2\Big\{(1 + 2\gamma)\mathsf{W}_2^2(P^{\text{in}}, Q) && (19)\\
&\qquad + 2\gamma \mathsf{W}_2^2(P^{\text{in}}, M^*) + \gamma\Omega(M^*) - \Omega(P)\Big\} && (20)\\
&\le (2\alpha + 4\gamma + 4\alpha\gamma)\mathsf{W}_2^2(P^{\text{in}}, M^*) \\
&\qquad + (2\alpha + 2\gamma + 4\alpha\gamma)\Omega(M^*) \\
&\qquad - (2 + 4\gamma)\Omega(M) - 2\Omega(P). && (21)
\end{aligned}$$

The conclusion is immediate by noting that Ω is a non-negative function. □

C Missing details from Section 4

C.1 Categorical variables

As we have mentioned, real-world relational database queries often involve a significant number of categorical variables, such as color, month, or city. The most common way to deal with categorical variables in practical settings is to one-hot encode them, whereby a categorical feature such as city is represented by an indicator vector

$$\boldsymbol{x}_{\text{city}} = \begin{bmatrix} \mathbf{1}_{\text{city}=c_1} & \mathbf{1}_{\text{city}=c_2} & \cdots \mathbf{1}_{\text{city}=c_L} \end{bmatrix} \qquad (22)$$

where $\{c_1, \ldots, c_L\}$ is the set of cities occuring in the data. The subspace associated with these indicator vectors is known as the *categorical subspace* of a categorical variable. This one-hot representation substantially increases the data matrix size via an increase in the *dimensionality* of the data. For example, a dataset of about 30 mostly categorical features with hundreds or thousands of categories for each feature will have its dimensionality exploded to the order of thousands with one-hot encoding.

The k-means subproblem within a categorical subspace is solvable efficiently *and* optimally, without one-hot encoding. This optimal solution can be computed in the same time it takes to find the number of points in each category, which is a vast improvement on either an optimal dynamic program or Lloyd's algorithm. Furthermore, it helps keep m as low as the number of database attributes in the query.

Consider a weighted k-means subproblem solved by $\mathsf{wkmeans}_1$ defined on a categorical subspace induced by a categorical feature K that has L categories. Then, the instance is of the form $(\boldsymbol{I}, v)$, where $\boldsymbol{I}$ is the collection of L indicator vectors $\mathbf{1}_e$, one for each element $e \in \mathsf{Dom}(K)$. (One can think of $\boldsymbol{I}$ as the identity matrix of order L.) Define the weight function v as

$$v(\mathbf{1}_e) = \sum_{\boldsymbol{x} \in \boldsymbol{X}, \boldsymbol{x}_K = e} w(\boldsymbol{x}). \tag{23}$$

For any set $F \subseteq \mathsf{Dom}(K)$, let $\boldsymbol{v}_F$ denote the vector $(v(\mathbf{1}_e))_{e \in F}$. Also, $\|\boldsymbol{v}_F\|_1$ and $\|\boldsymbol{v}_F\|_2$ denote the ℓ_1 and ℓ_2 norm, respectively. It is useful to rewrite the categorical weighted k-means problem:

Proposition C.1. *The categorical weighted k-means instance $(\boldsymbol{I}, v)$ admits the following optimization objective:*

$$\mathsf{OPT}(\boldsymbol{I}, v) = \|\boldsymbol{v}\|_1 - \max_{\mathcal{F}} \sum_{F \in \mathcal{F}} \frac{\|\boldsymbol{v}_F\|_2^2}{\|\boldsymbol{v}_F\|_1}, \tag{24}$$

where $\mathcal{F}$ ranges over all partitions of $\mathsf{Dom}(K)$ into k parts.

Proof. First, consider a subset $F \subseteq \mathsf{Dom}(K)$ of the categories; the centroid $\boldsymbol{\mu}$ of (weighted) indicator vectors $\mathbf{1}_e$, $e \in F$, can be written down explicitly:

$$\mu_e = \begin{cases} 0 & e \notin F \\ \frac{v_e}{\|\boldsymbol{v}_F\|_1} & v \in F, \end{cases} \tag{25}$$

The weighted sum of squared distances between $\mathbf{1}_e$ for all $e \in F$ to $\boldsymbol{\mu}$ is

$$\begin{aligned} &\sum_{e \in F} (\|\boldsymbol{\mu}\|_2^2 - \mu_e^2 + (\mu_e - 1)^2) v_e \\ &= \frac{\|\boldsymbol{v}_F\|_2^2}{\|\boldsymbol{v}_F\|_1} + \sum_{e \in F} ((\mu_e - 1)^2 - \mu_e^2) v_e \\ &= \frac{\|\boldsymbol{v}_F\|_2^2}{\|\boldsymbol{v}_F\|_1} + \sum_{e \in F} (-2\mu_e + 1) v_e \\ &= \|\boldsymbol{v}_F\|_1 - \|\boldsymbol{v}_F\|_2^2 / \|\boldsymbol{v}_F\|_1 . \end{aligned}$$

Thus, the weighted k-means objective takes the form

$$\min_{\mathcal{F}} \sum_{F \in \mathcal{F}} \left(\|\boldsymbol{v}_F\|_1 - \|\boldsymbol{v}_F\|_2^2 / \|\boldsymbol{v}_F\|_1 \right) \tag{26}$$

$$= \|\boldsymbol{v}\|_1 - \max_{\mathcal{F}} \sum_{F \in \mathcal{F}} \|\boldsymbol{v}_F\|_2^2 / \|\boldsymbol{v}_F\|_1 , \tag{27}$$

which concludes the proof. □

In (24), note that $\|\boldsymbol{v}\|_1$ is the total weight of input points; hence, we can equivalently solve the inner maximization problem. With the categorical weighted k-means objective in place, we can derive the optimal clustering. To do so, We next need the following elementary lemma.

Lemma C.2. *Suppose that $x, a_1, a_2, b_1, b_2 > 0$, $b_1^2 \geq a_1$, $b_2^2 \geq a_2$ and $x \geq \max\{a_1/b_1, a_2/b_2\}$. Then $x + \frac{a_1+a_2}{b_1+b_2} \geq \max\left\{ \frac{x^2+a_1}{x+b_1} + \frac{a_2}{b_2}, \frac{x^2+a_2}{x+b_2} + \frac{a_1}{b_1} \right\}$.*

Proof. It suffices to establish $x + \frac{a_1+a_2}{b_1+b_2} \geq \frac{x^2+a_1}{x+b_1} + \frac{a_2}{b_2}$, or equivalently

$$x - \frac{x^2 + a_1}{x + b_1} \geq \frac{a_2}{b_2} - \frac{a_1 + a_2}{b_1 + b_2},$$

which can be simplified as

$$x(b_1 + b_2 + a_1/b_1 - a_2/b_2) \geq a_1 b_2/b_1 + a_2 b_1/b_2. \tag{28}$$

To verify this inequality, consider two cases. If $a_1/b_1 \geq a_2/b_2$, then $LHS \geq x(b_1 + b_2) \geq (a_2/b_2)b_1 + (a_1/b_1)b_2$. On the other hand, if $a_2/b_2 > a_1/b_1$. Since $b_2 - a_2/b_2 \geq 0$,

$$\begin{aligned} LHS &\geq (a_2/b_2)(b_1 + b_2 + a_1/b_1 - a_2/b_2) \\ &= a_2 b_1/b_2 + a_2 + a_1 a_2/(b_1 b_2) - a_2^2/b_2^2 \\ &= a_2 b_1/b_2 + a_1 b_2/b_1 \\ &\quad + (b_2 - a_2/b_2)(a_2/b_2 - a_1/b_1) \\ &\geq a_2 b_1/b_2 + a_1 b_2/b_1. \end{aligned}$$

Thus the proof is complete. □

Then, the optimal solution to the categorical k-means instance is an immediate consequence:

Corollary C.3. *Let $(e_1, \ldots, e_L)$ be a permutation of $\mathsf{Dom}(K)$ such that $v_{e_1} \geq v_{e_2} \geq \ldots \geq v_{e_L}$. Then for any $k \geq 2$ and any k-partition $\mathcal{F}$ of $\mathsf{Dom}(K)$, there holds*

$$v_{e_1} + \ldots + v_{e_{k-1}} + \frac{\sum_{i=k}^{L} v_i^2}{\sum_{i=k}^{L} v_i} \geq \sum_{F \in \mathcal{F}} \frac{\|\boldsymbol{v}_F\|_2^2}{\|\boldsymbol{v}_F\|_1}.$$

Proof. We prove the claim by induction on k. Let $F \in \mathcal{F}$ be the set containing the element $\{e_1\}$. If there is only one element in F then we apply the induction hypothesis on the remaining terms. Otherwise, F contains at least two elements. Let $G \in \mathcal{F}$ be an arbitrary element of $\mathcal{F}$ where $G \neq F$. Define $\mathcal{F}'$ to

be the partition obtained from $\mathcal{F}$ by replacing (F, G) with $(\{e_1\}, F \cup G - \{e_1\})$. Then, Lemma C.2 can be applied to get

$$\sum_{F \in \mathcal{F}} \frac{\|\boldsymbol{v}_F\|_2^2}{\|\boldsymbol{v}_F\|_1} \leq \sum_{F \in \mathcal{F}'} \frac{\|\boldsymbol{v}_F\|_2^2}{\|\boldsymbol{v}_F\|_1}.$$

Induction on the tail $k-1$ terms completes the proof. □

Theorem 4.1 follows trivially from the above corollary. Corollary C.3 and the objective for k-means on a single attribute in the equation of Proposition C.1 establishes precisely the structure of the optimal solution for data consisting of a single categorical variable.

C.2 Reducing the coreset size with FDs

Next, we address the second call to $\mathsf{wkmeans}_2$: its runtime is dependent on the size of the grid $\boldsymbol{G}$, which can be up to $O(k^m)$, where m is the number of features from the input. Databases often contain *functional dependencies* (FDs), which we can exploit to reduce the size of $\boldsymbol{G}$. An FD is a dimension whose value depends entirely on the value of another dimension. For example, for a retailer dataset that includes geographic information, one might encounter features such as storeID, zip, city, state, and country. Here, storeID functionally determines zip, which determines city, which in turns determines state, leading to country. This common structure is known as an *FD-chain*, and appears often in real-world FEQs.

If we were to apply Rk-means without exploiting the FDs, the features storeID, zip, city, state, and country would contribute a factor of k^5 to the grid size. However, by using the functional dependency structure of the database, we show that only a factor of $5k$ is contributed to the grid size, because most of the k^5 grid points $\boldsymbol{g}$ have $w_{\mathsf{grid}}(\boldsymbol{g}) = 0$ as defined in (3). More generally, whenever there is an *FD chain* of (simple) functional dependencies including p features, their overall contribution to the grid size is a factor of $O(kp)$ instead of $O(k^p)$, and the grid points with non-zero weights can be computed efficiently in time $O(kp)$.

Lemma C.4. *Suppose all d input features are categorical and form an FD-chain. Then, the total number of grid points $\boldsymbol{g} \in \boldsymbol{G}$ with non-zero w_{grid} weight is at most $d(k-1)+1$.*

Proof. Suppose the features are $K_1, \ldots, K_d$, where K_i functionally determine K_{i+1}, and $\mathsf{Dom}(K_i) = \{e_1^i, e_2^i, \cdots, e_{n_i}^i\}$. Without loss of generality, we also assume that the elements in $\mathsf{Dom}(K_i)$ are sorted in descending order of weights:

$$w(\mathbf{1}_{e_1^i}) \geq w(\mathbf{1}_{e_2^i}) \geq \cdots \geq w(\mathbf{1}_{e_{n_i}^i}). \tag{29}$$

From Corollary C.3, we know the set $\boldsymbol{C}_i$ of k centroids of each of the categorical subspace for K_i: there is a centroid $\boldsymbol{\mu}_j^i = \mathbf{1}_{e_j^i}$ for each $j \in [k-1]$, and then a centroid $\boldsymbol{\mu}_k^i$ of the rest of the indicator vectors. The elements e_j^i for $j \in [k-1]$ shall be called "heavy" elements, and the rest are "light" elements.

Now, consider an input vector $\boldsymbol{x} = (x_1, \ldots, x_d)$ where $x_i \in \mathsf{Dom}(K_i)$. Under one-hot encoding, this vector is mapped to a vector of indicator vectors $\mathbf{1}_{\boldsymbol{x}} := (\mathbf{1}_{x_1}, \cdots, \mathbf{1}_{x_d})$. We need to answer the question: which grid point in $\boldsymbol{G} = \boldsymbol{C}_1 \times \cdots \times \boldsymbol{C}_d$ is $\mathbf{1}_{\boldsymbol{x}}$ closest to? Since the ℓ_2^2-distance is decomposable into component sum, we can determine the closest grid point by looking at the closest centroid in $\boldsymbol{C}_i$ for $\mathbf{1}_{x_i}$, for each $i \in [d]$.

If $x_i \in \{e_1^i, \ldots, e_{k-1}^i\}$, then the corresponding one-hot-encoded version $\mathbf{1}_{x_i}$ *is* itself one of the centroids in $\boldsymbol{C}_i$, and thus it is its own closest centroid. Otherwise, the closest centroid to $\mathbf{1}_{x_i}$ is $\boldsymbol{\mu}_k^i$, because $\left\|\mathbf{1}_{x_i} - \boldsymbol{\mu}_k^i\right\|^2 < 2$, and $\left\|\mathbf{1}_{x_i} - \boldsymbol{\mu}_j^i\right\|^2 = 2$ for every $j \in [k-1]$.

Let $\boldsymbol{\mu}^i(x_i) \in \boldsymbol{C}_i$ denote the closest centroid in $\boldsymbol{C}_i$ to $\mathbf{1}_{\boldsymbol{x}_i}$. The closest grid point to $\mathbf{1}_{\boldsymbol{x}}$ is completely determined: $\boldsymbol{g} = (\boldsymbol{\mu}^1(x_1), \cdots, \boldsymbol{\mu}^d(x_d))$. Furthermore, let $i \in [d]$ denote the smallest index such that x_i is heavy. Then, we can write $\boldsymbol{g}$ as

$$\boldsymbol{g} = (\boldsymbol{\mu}_k^1, \cdots, \boldsymbol{\mu}_k^{i-1}, \mathbf{1}_{x_i}, \boldsymbol{\mu}^{i+1}(x_{i+1}), \cdots, \boldsymbol{\mu}^d(x_d)) \tag{30}$$

Note that once x_i is fixed, due to the FD-chain the *entire* suffix $(\mathbf{1}_{x_i}, \boldsymbol{\mu}^{i+1}(x_{i+1}), \cdots, \boldsymbol{\mu}^d(x_d)$ of $\boldsymbol{g}$ is determined. Hence, the number of different $\boldsymbol{g}$s can only be at most $d(k-1)+1$: there are $d+1$ choices for i (from 0 to d), and $k-1$ choices for x_i if $i > 0$. □

Theorem 4.2 follows trivially from the above lemma, because the ℓ_2^2-distance is the sum over the ℓ_2^2-distances of the subspaces.

C.3 Analysis of Step 4 of Rk-means

Here we discuss the optimization and acceleration of Step 4 of the Rk-means implementation as described in Section 4. Recall that the categorical subspace k-means problem is solved trivially using Theorem 4.1, where we sort all the weights, and the heaviest $k-1$ elements form their own centroid, while the remaining vectors are clustered together (the "light cluster").

If S_j is a categorical subspace corresponding to a categorical variable K where $\mathsf{Dom}(K) = \{e_1, \ldots, e_L\}$. Without loss of generality, assume $w(\mathbf{1}_{e_1}) \geq \cdots \geq w(\mathbf{1}_{e_L})$, then the centroid of the light cluster is an

L-dimensional vector $\boldsymbol{c} = (s_e)_{e \in \mathsf{Dom}(K)}$

$$s_{e_i} := \begin{cases} 0 & i \in [k-1] \\ \frac{w(\mathbf{1}_{e_i})}{\sum_{j=k}^{L} w(\mathbf{1}_{e_j})} & i \geq k \end{cases} \tag{31}$$

This encoding is sound and space-inefficient.

Remember also that Step 4 clusters the coreset $\boldsymbol{G}$ using a modified version of Lloyd's weighted k-means that exploits the structure of $\boldsymbol{G}$ and sparse representation of categorical values. We show how to improve the distance computation $\|\boldsymbol{c}_j - \boldsymbol{\mu}_j\|^2$ for sub-space S_j, where $\boldsymbol{c}_j$ and $\boldsymbol{\mu}_j$ are the j-th components of a grid point and respectively of a centroid for $\boldsymbol{G}$. Since this subspace corresponds to a categorical variable K with, say, L_j categories, it is mapped into L_j sub-dimensions. Let $\boldsymbol{c}_j = [s_1, \ldots, s_{L_j}]$ and $\boldsymbol{\mu}_j = (t_1, \ldots, t_{L_j})$. Using the explicit one-hot encoding of its categories, we would need $O(L_j)$ time to compute $\|\boldsymbol{c}_j - \boldsymbol{\mu}_j\|^2 = \sum_{\ell \in [L_j]} (s_\ell - t_\ell)^2$. We can instead achieve $O(1)$ time as shown next. There are k distinct values for $\boldsymbol{c}_j$ by our coreset construction, each represented by a vector of size L_j with one non-zero entry for $k-1$ of them and $L_j - k + 1$ non-zero entries for one of them.

If $\boldsymbol{c}_j = \mathbf{1}_e$ is an indicator vector for some element $e \in K$ (e is one of the $k-1$ heavy categories), then

$$\|\boldsymbol{c}_j - \boldsymbol{\mu}_j\|^2 = \|\mathbf{1}_e - \boldsymbol{\mu}_j\|^2 = 1 - 2t_e + \|\boldsymbol{\mu}_j\|^2 . \tag{32}$$

If $\boldsymbol{c}_j$ is a light cluster centroid,

$$\|\boldsymbol{c}_j - \boldsymbol{\mu}_j\|^2 = \|\boldsymbol{c}_j\|^2 + \|\boldsymbol{\mu}_j\|^2 - 2\langle \boldsymbol{c}_j, \boldsymbol{\mu}_j \rangle . \tag{33}$$

In (32), by pre-computing $\|\boldsymbol{\mu}_j\|^2$ we only spend $O(1)$-time per heavy element e. In (33), by also pre-computing $\|\boldsymbol{c}_j\|^2$ and $\langle \boldsymbol{c}_j, \boldsymbol{\mu}_j \rangle$, and by noticing that $\boldsymbol{c}_j$ is $(L_j - k + 1)$-sparse, we spend $O(L_j - k)$-time here. Overall, we spend time $O(L_j)$ for computing $\|\boldsymbol{c}_j - \boldsymbol{\mu}_j\|^2$ per categorical dimension, modulo the pre-computation time.

Step 4 thus requires $O(|\boldsymbol{G}|mk + \sum_{j \in [m]} L_j k) = O(|\boldsymbol{G}|mk + Dkm)$ per iteration, whereas a generic approach would take time $O(\sum_{j \in [m]} |\boldsymbol{G}|kL_j) = O(|\boldsymbol{G}|Dkm)$. Our modified weighted k-means algorithm thus saves a factor proportional to the total domain sizes of the categorical variables, which may be as large as D.

C.4 Theorem 4.3

Proof of Theorem 4.3. Let N denote the maximum number of tuples in any input relation of the FEQ, $|\boldsymbol{X}|$ the number of tuples in the data matrix, fhtw the *fractional hypertree width* of the FEQ t the number of iterations of Lloyd's algorithm, d denote the number of features pre-one-hot encoding, r number of input relations to the FEQ, D the real dimensionality of the problem after one-hot-encoding.

We analyze the time complexity for each of the four steps of the Rk-means algorithm.

Step 1 projects $\boldsymbol{X}$ into each subspace S_j and compute the total weight of each projected point:

$$\forall j \in [d] : w_j(\boldsymbol{x}_{S_j}) := \sum_{\boldsymbol{x}_{[d] \setminus \{S_j\}}} \prod_{F \in \mathcal{E}} R_F(\boldsymbol{x}_F) \tag{34}$$

Each of the d FAQs (34) in Step 1 can be computed in time $\tilde{O}(rd^2 N^{\mathsf{fhtw}})$ using $\mathsf{InsideOut}$, as we have reviewed in Section A.

In Step 2, the optimal clustering in each dimension takes time $\tilde{O}(L_j)$ for each categorical variable j (whose domain size is L_j, and $O(kN^2)$ for each continuous variable, with an overall runtime of $O(kdN^2)$.

Step 3 constructs $\boldsymbol{G}$, whose size is bounded by $|\boldsymbol{X}|$ and by the FD result of Theorem 4.2. In practice, this number can be much smaller since we skip the data points whose weights are zero. To perform this step we construct a tree decomposition of FEQ with with equal fhtw (this step is data-independent, only dependent on the size of FEQ). Then, from each value x_j of an input variable X_j, we determine its centroid $c(x_j)$ which was computed in step 2. By conditioning on combinations of $(c_1, \ldots, c_j)$, we can compute w_{grid} one for each combiation in $\tilde{O}(dN^{\mathsf{fhtw}})$-time, for a total run time of $\tilde{O}(rd|\boldsymbol{G}|N^{\mathsf{fhtw}})$.

Step 4 – as analyzed in Section C.3 – clusters $\boldsymbol{G}$ in time $O((|\boldsymbol{G}| + D)kmt)$, where t is the number of iterations of k-means used in Step 4. The most expensive computation is due to the one-dimensional clustering for the continuous variables and the computation of the coreset.

To compare the total runtime with $|\boldsymbol{X}|$, we only need to note that $|\boldsymbol{X}|$ can be as large as N^{ρ^*}, where $\rho*$ is the *fractional edge covering number* of the FEQ's hypergraph Ngo (2018). Depending on the query, ρ^* is always at least 1, and can be as large as the number of features d. Furthermore, there are classes of queries where fhtw is bounded by a constant, yet ρ^* is unbounded Marx (2013). This means, for classes of FEQs where $\rho^* > \max\{\mathsf{fhtw}, 2\}$ the ratio between $|\boldsymbol{X}|$ and Rk-means's runtime will be $\tilde{O}mega(N^{\rho^* - \max\{\mathsf{fhtw}, 2\}}/t)$, which is unbounded. □

For reference, we compare the asymptotic runtime of Rk-means to the standard implementation of Lloyd's algorithm. The standard implementation contains two steps: (1) compute the one-hot-encoded data matrix

$\boldsymbol{X}$, and (2) run Lloyd's algorithm on $\boldsymbol{X}$. The first step, materializing $\boldsymbol{X}$, takes time $\tilde{O}(rd^2N^{\mathsf{fhtw}} + D|\boldsymbol{X}|)$. The second step, running Lloyd algorithm, takes time $\tilde{O}(tkD|\boldsymbol{X}|)$, as is well known. Thus, the standard approach takes time $\tilde{O}(rd^2N^{\mathsf{fhtw}} + tkD|\boldsymbol{X}|)$.

D Missing details from Section 5

We provide a more detailed description of the three datasets introduced in Section 5.

Retailer has five relations: *Inventory* stores the number of inventory units for each date, location, and stock keeping unit (sku); *Location* keeps for each store: its zipcode, the distance to the closest competitors, and the type of the store; *Census* provides 14 attributes that describe the demographics of a given zipcode, including population size or average household income; *Weather* stores statistics about the weather condition for each date and store, including the temperature and whether it rained; *Items* keeps track of the price, category, subcategory, and category cluster of each sku.

Favorita has six relations: *Sales* stores the number of units sold for items for a given date and store, and an indicator whether or not the unit was on promotion at this time; *Items* provides additional information about the skus, such as the item class and price; *Stores* keeps additional information on stores, like the city they are located it; *Transactions* stores the number of transaction for each date and store; *Oil* provides the oil price for each date; and *Holiday* indicates whether a given day is a holiday. The original dataset gave the `units_sold` attribute with a precision of three decimals places. This resulted in a very many distinct values for this attribute, which has a significant impact on the Step 2 of the Rk-means algorithm. We decreased the precision for this attribute to two decimal places, which decreases the number of distinct values by a factor of four. This modification has no effect on the final clusters or their accuracy.

Yelp has five relations: *Review* gives the review rating that a user gave to a business and the date of the review; *User* provides information about the users, including how many reviews the made, when they join, and how many fans they have; *Business* provides information about the businesses that are reviewed, such as their location and average rating; *Category* provide information about the categories, i.e. Restaurant, and respectively attributes of the business, *Attributes* is an aggregated relation, which stores the number of attributes (i.e., open late) that have been assigned to a business. A business can be categorized in many ways, which is the main reason why the size of the join is significantly larger than the underlying relations.

Bibliography

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN 0-201-53771-0. URL `http://webdam.inria.fr/Alice/`.

Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. FAQ: questions asked frequently. In *PODS*, pages 13–28, 2016.

Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. Juggling functions inside a database. *SIGMOD Rec.*, 46(1):6–13, 2017.

Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Ac/dc: In-database learning thunderstruck. In *2nd Workshop on Data Mgt for End-To-End ML*, DEEM'18, pages 8:1–8:10, 2018a.

Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. In-database learning with sparse tensors. In *PODS*, pages 325–340, 2018b.

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *FOCS*, pages 61–72, 2017.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, page 1027–1035, 2007.

Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k -means clustering via lightweight coresets. In *SIGKDD*, pages 1119–1127, 2018.

Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *PVLDB*, 5(7):622–633, 2012.

Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 1995–2003, 2013.

Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F. Yang. Clustering high dimensional dynamic data streams. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 576–585, 2017.

Field Cady. *The Data Science Handbook*. John Wiley & Sons, 2017.

Ryan R. Curtin, Marcus Edel, Mikhail Lozhnikov, Yannis Mentekidis, Sumedh Ghaisas, and Shangtong

Zhang. mlpack 3: a fast, flexible machine learning library. *J. Open Source Software*, 3:726, 2018.

C.J. Date, H. Darwen, and N. Lorentzos. *Temporal Data & The Relational Model*. Elsevier, 2002.

E. del Barrio, J. A. Cuesta-Albertos, C. Matran, and A. Mayo-Iscar. Robust clustering tools based on optimal transportation. *Statistics and Computing*, pages 1–22, 2017.

Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using mapreduce. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 681–689, 2011.

B. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley & Sons, 2011.

Favorita Corp. Corporacion Favorita Grocery Sales Forecasting: Can you accurately predict sales for a large grocery chain? `https://www.kaggle.com/c/favorita-grocery-sales-forecasting/`, 2017.

S. Graf and H. Luschgy. *Foundations of quantization for probability distributions*. Springer-Verlag, New York, 2000.

Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. *ACM Trans. Algorithms*, 11 (1):4:1–4:20, 2014.

Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. Knowl. Data Eng.*, 15(3):515–528, 2003.

S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *SODA*, 2004.

Sariel Har-Peled and Soham Mazumdar. Coresets for k-means and k-median clustering and their applications. *CoRR*, abs/1810.12826, 2018. URL `http://arxiv.org/abs/1810.12826`.

J. A. Hartigan. *Clustering algorithms*. Wiley, New York, 1975.

N. Ho, X. Nguyen, M. Yurochkin, H. H. Bui, V. Huynh, and D. Phung. Multilevel clustering via Wasserstein means. In *ICML*, pages 1501–1509, 2017.

Z. Huang. Extensions to the k-means algorithm for clustering large data sets of categorical values. *Data mining and Knowledge discovery*, 2:283–304, 1998.

L. Kaufman and P. J. Roussew. *Finding Groups in Data - An Introduction to Cluster Analysis*. Wiley & Sons, 1990.

Mahmoud Abo Khamis, Ryan Curtin, Benjamin Moseley, Hung Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. On functional aggregate queries with additive inequalities. In *PODS*, 2019.

Daphne Koller and Nir Friedman. *Probabilistic graphical models*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2009. Principles and techniques.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):42:1–42:51, 2013.

Adam Meyerson, Liadan O'Callaghan, and Serge A. Plotkin. A k-median algorithm with running time independent of data size. *Machine Learning*, 56(1-3): 61–87, 2004.

Hung Q. Ngo. Worst-case optimal join algorithms: Techniques, results, and open problems. In *PODS*, pages 111–124, 2018.

Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms. *Journal of the ACM (JACM)*, 65(3):1–40, 2018.

Dan Olteanu and Maximilian Schleich. Factorized databases. *SIGMOD Rec.*, 45(2):5–16, 2016.

Carlos Ordonez. Integrating k-means clustering with a relational DBMS using SQL. *IEEE Trans. Knowl. Data Eng.*, 18(2):188–201, 2006.

Carlos Ordonez and Edward Omiecinski. Efficient disk-based k-means clustering for relational databases. *IEEE Trans. Knowl. Data Eng.*, 16(8):909–921, 2004.

D. Pollard. Quantization and the method of k-means. *IEEE Trans. Inf. Theory*, 28(2):199–204, 1982.

C. Sanderson and R.R. Curtin. A user-friendly hybrid sparse matrix class in C++. In *Proceedings of the 2018 International Congrees on Mathematical Software (ICMS)*, pages 422–430. Springer, 2018. ISBN 978-3-319-96418-8.

Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. Learning linear regression models over factorized joins. In *SIGMOD*, pages 3–18, 2016.

Maximilian Schleich, Dan Olteanu, Mahmoud Abo Khamis, Hung Q. Ngo, and XuanLong Nguyen. A layered aggregate engine for analytics workloads. In *SIGMOD*, pages 1642–1659, 2019.

Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 802–813, 2018.

W. Sun, J. Wang, and Y. Fang. Regularized k-means clustering of high-dimensional data and its asymptotic consistency. *Electronic Journal of Statistics*, 9: 148–167, 2012.

Mikkel Thorup. Quick k-median, k-center, and facility location for sparse graphs. In *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, pages 249–260, 2001.

Todd L. Veldhuizen. Leapfrog Triejoin: A simple, worst-case optimal join algorithm. *arXiv preprint arXiv:1210.0481*, 2012.

Cédric Villani. *Optimal transport*, volume 338 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 2009.

Haizhou Wang and Mingzhou Song. Ckmeans.1d.dp: Optimal k-means clustering in one dimension by dynamic programming. *The R Journal*, 3(2):29, 2011.

D. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105:713–726, 2010.

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus F. M. Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, 2008.

J. Ye, P. Wu, J. Wang, and J. Li. Fast discrete distribution clustering using barycenter with sparse support. *IEEE Trans. Signal Proc.*, 65(9):2317–2332, 2017.

Yelp. Yelp dataset challenge, `https://www.yelp.com/dataset/challenge/`, 2017.