Warsaw PhD Open Course: From Joins to Aggregates and Optimization Problems - Exam solutions

Djordje Zivanovic

February 10, 2019

1 Let $Q(\mathbf{A_1} \cup \mathbf{A_2} \cup \cdots \cup \mathbf{A_n})$ denote a join query on relations $R_1(\mathbf{A_1}) = R_1(A_1, A_2), \cdots R_n(\mathbf{A_n}) = R_n(A_n, A_1)$. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ denote a corresponding hypegraph to the join query, then $\mathcal{V} = \{A_1, .., A_n\}$ and $\mathcal{E} = \{\{A_1, A_2\}, ..., \{A_n, A_1\}\}$.

1.1 Let x_{R_i} denote the weight of an hyperedge (relation) R_i in the join query Q. In addition let $cover_i$ denote the set of relations corresponding to edges that are part of the cover for the variable i. The fractional edge cover number $\rho^*(Q)$ is the cost of an optimal solution to the linear program:

$$\text{minimize} \sum_{i \in [n]} x_{R_i} \tag{1}$$

$$\text{subject to} \sum_{rel_j \in cover_i} x_{rel_j} \geq 1, i \in [n] \tag{2}$$

$$x_{R_i} \geq 0, i \in [n]$$

By a definition of a cover, only edges incident to variable are part of its cover. Thus, in our case $cover_i = \{R_{i-1}, R_i\}$, where R_0 is R_n in the case of A_1. If we rewrite the inequalities 2 we will get:

$$\begin{aligned} x_{R_1} + x_{R_n} &\geq 1 \\ x_{R_1} + x_{R_2} &\geq 1 \\ &\vdots \\ x_{R_{n-1}} + x_{R_n} &\geq 1 \end{aligned}$$

By summing inequalities we get:

$$2 \left(\sum_{i \in [n]} x_{R_i} \right) \geq n$$

This means the minimized function $\rho^*(Q)$ cannot be less than $\frac{n}{2}$. This is achievable if we set $x_{R_i} = \frac{1}{2}, i \in [n]$. We can see that this solution satisfies all constraints in the linear program. Finally, $\rho^*(Q) = \frac{n}{2}$.
A width of a specific hypertree decomposition is defined as a maximum among all edge covers for each of the nodes of the hypertree. A hypertree width (htw) for the specific query Q is defined as a minimum width among all possible widths of hypertree decompositions of a query Q.
A fractional width of a specific hypertree decomposition is defined as a maximum among all fractional edge covers for each of the nodes of the hypertree. A fractional hypertree width (fhtw) for the specific query Q is defined as a minimum width among all possible widths of hypertree decompositions of a query Q. The only difference is that in the case of fractional hypertree width we use fractional edge covers instead of edge covers.
In the case of $n = 2$ $fhtw(Q) = htw(Q) = 1$. For $n >= 3$ the hypertree decomposition on the figure 1 is the optimal one. From picture we can figure out that $htw(Q) = 2$ in all cases, while $fhtw(Q) = 2$ for $n > 3$, but for $n = 3$ it is $\frac{3}{2}$.
The decomposition on the figure 1 is optimal because we have to "propagate" one variable through nodes of the hypertree in order to fulfill the condition that all hyperedges of hypergraph are contained in the nodes of hypertree. Adding more than three variables inside a node of hypertree would increase (fractional) width of a hypertree decomoposition because we would need to cover at least two variables that are not in the same relation. (Fractional) edge cover number for each of nodes of hypegraph inside nodes of hyperetree is 2 because $A_1 A_i A_{i-1}$ can be covered with $x_{R_i} = 1, x_{R_1} = 1$. The only exception to the previous case is $n = 3$ where fractional edge cover of $\frac{3}{2}$ is possible.

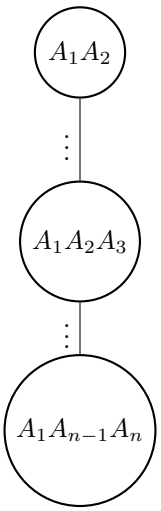

Figure 1: Hypertree decomposition that corresponds to fhtw

1.2 A size of listing representation is $\mathcal{O}(N^{\rho^*(Q)}) = \mathcal{O}(N^{\frac{n}{2}})$ and can be computed in time $\mathcal{O}(N^{\rho^*(Q)}) = \mathcal{O}(N^{\frac{n}{2}})$. Let $|D|$ represent a size the of training data set which is the number of tuples in the join of all relations $R_1, R_2, ..., R_n$. Let $\mathbf{X}$ represent training data set consisting of tuples of the join with an additional intercept column with values only 1. Let $\mathbf{y}$ represent labels of training data corresponding to tuples of join and let $\boldsymbol{\theta}$ be a column vector of training parameters for all features and a bias. In addition, let λ represent hyperparemeter of ℓ_2 regularizer. A loss function with added ℓ_2 regularizer is :

$$\begin{aligned} 2|D|\mathcal{L}(\mathbf{X},\mathbf{y},\theta) &= (\mathbf{y}-\mathbf{X}\theta)^2 + \lambda|D|||\theta||_2^2 \\ &= (\mathbf{y}-\mathbf{X}\theta)^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\theta) + \lambda|D|||\theta||_2^2 \\ &= \mathbf{y}^{\mathrm{T}}\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{X}\theta - \theta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \theta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta + \lambda|D||\theta||_2^2 \\ &= \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{y}^{\mathrm{T}}\mathbf{X}\theta + \theta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta + \lambda|D|\theta^{\mathrm{T}}\theta \end{aligned} \tag{3}$$

In the equation 3 we used the property that transpose of a scalar is the same scalar. All equations before that are simple vector multiplications and additions. Finally:

$$\mathcal{L}(\mathbf{X},\mathbf{y},\theta) = \frac{1}{2|D|}\mathbf{y}^{\mathrm{T}}\mathbf{y} - \frac{1}{|D|}\mathbf{y}^{\mathrm{T}}\mathbf{X}\theta + \frac{1}{2|D|}\theta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta + \frac{\lambda}{2}\theta^{\mathrm{T}}\theta$$

If we derive the equation 3 we get the gradient of the loss function for one point:

$$\begin{aligned} \nabla_\theta\mathcal{L}(\mathbf{X},\mathbf{y},\theta) &= 0 - \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta + \lambda\theta \\ &= \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta - \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \lambda\theta \end{aligned}$$

Here we used properties $\nabla_{\mathbf{x}}(\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{A}^{\mathrm{T}}\mathbf{x}$, $\nabla_{\mathbf{x}}(\mathbf{x}^{\mathrm{T}}\mathbf{x}) = 2\mathbf{x}$, $\nabla_{\mathbf{x}}(\mathbf{c}^{\mathrm{T}}\mathbf{x}) = \mathbf{c}$. In addition we used $(\mathbf{u}\mathbf{v})^{\mathrm{T}} = \mathbf{v}^{\mathrm{T}}\mathbf{u}^{\mathrm{T}}$.
A gradient descent formula is:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta\mathcal{L}(\mathbf{X},\mathbf{y},\theta)$$

where θ_t represents parameters in the t-th iteration (we start with θ_0) and η represents hyperparameter of gradient learning.
After joining relations we need in each step to calculate gradient. We can notice that $\mathbf{X}^{\mathrm{T}}\mathbf{y}$ can be cached and the complexity to do so is $\mathcal{O}(n \cdot N^{\frac{n}{2}})$. Similarly with $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, whose calculation takes $\mathcal{O}(n^2 \cdot N^{\frac{n}{2}})$. In each iteration we need to multiply $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ with θ which takes $\mathcal{O}(n^2)$ time. We will ignore additions of vectors because they are one $O(n)$. When we add all time complexities we get:

$$\mathcal{O}(N^{\frac{n}{2}} + n^2 \cdot N^{\frac{n}{2}} + n \cdot N^{\frac{n}{2}} + n^2 \cdot t)$$

If we refined the complexity and included the fact that categorical variables behave as the vectors whose length is the same as a length of the particular domain, the complexity would become:

$$\mathcal{O}(N^{\frac{n}{2}} + (n_d)^2 \cdot N^{\frac{n}{2}} + n_d \cdot N^{\frac{n}{2}} + n_d^2 \cdot t)$$

where $n_d = n - 3 + |Dom_{A_1}| + |Dom_{A_2}| + |Dom_{A_3}|$.
Memory complexity after the join is $\mathcal{O}(N^{\frac{n}{2}})$. $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ cached takes $\mathcal{O}(n^2)$ and $\mathbf{X}^{\mathrm{T}}\mathbf{y}$ takes $\mathcal{O}(n)$. When we add these complexities we get memory complexity at each iteration is: $\mathcal{O}(n^2)$. In addition, if we include in the result the fact that we have categorical values, memory complexity is $\mathcal{O}(n_d^2)$.

1.3 If we denote $\mathbf{\Sigma} = \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{X}, \mathbf{c} = \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{y}$, and elements of these tensors, respectively, $\sigma_{ij}, \mathbf{c}_i$ then FAQs for Σ are:

- When i-th feature and j-th feature are continuous values.

$$|D| \cdot \sigma_{ij} = \sum_{a_1 \in Dom(A_1)} \sum_{a_2 \in Dom(A_2)} \cdots \sum_{a_n \in Dom(A_n)} a_i \cdot a_j \prod_{k \in [n]} \mathbf{1}_{R_k(a_k, a_{k+1})}$$

- When i-th feature is a categorical value and j-th feature is a continuous value.

$$|D| \cdot \sigma_{ij}[a_i] = \sum_{a_1 \in Dom(A_1)} \sum_{a_2 \in Dom(A_2)} \cdots \sum_{a_{i-1} \in Dom(A_{i-1})} \sum_{a_{i+1} \in Dom(A_{1+1})} \cdots \sum_{a_n \in Dom(A_n)} a_j \prod_{k \in [n]} \mathbf{1}_{R_k(a_k, a_{k+1})}$$

- When i-th feature and j-th feature are categorical values.

$$|D| \cdot \sigma_{ij}[a_i, a_j] = \sum_{\substack{a_1 \in \\ Dom(A_1)}} \sum_{\substack{a_2 \in \\ Dom(A_2)}} \cdots \sum_{\substack{a_{i-1} \in \\ Dom(A_{i-1})}} \sum_{\substack{a_{i+1} \in \\ Dom(A_{i+1})}} \cdots \sum_{\substack{a_{j-1} \in \\ Dom(A_{j-1})}} \sum_{\substack{a_{j+1} \in \\ Dom(A_{j+1})}} \cdots \sum_{a_n \in Dom(A_n)} \prod_{k \in [n]} \mathbf{1}_{R_k(a_k, a_{k+1})}$$

If we calculate σ_{ij} we don't need to calculate σ_{ji} because Σ matrix is symmetric, therefore $\sigma_{ij} = \sigma_{ij}^{\mathrm{T}}$
For $\mathbf{c}$ FAQs:

- When i-th feature and label are continuous values:

$$|D| \cdot \mathbf{c}_i = \sum_{a_1 \in Dom(A_1)} \sum_{a_2 \in Dom(A_2)} \cdots \sum_{a_n \in Dom(A_n)} a_i \cdot a_n \prod_{k \in [n]} \mathbf{1}_{R_k(a_k, a_{k+1})}$$

- When i-th feature is a categorical value and label is a continuous value:

$$|D| \cdot \mathbf{c}_i[a_i] = \sum_{a_1 \in Dom(A_1)} \sum_{a_2 \in Dom(A_2)} \cdots \sum_{a_{i-1} \in Dom(A_{i-1})} \sum_{a_{i+1} \in Dom(A_{1+1})} \cdots \sum_{a_n \in Dom(A_n)} a_n \prod_{k \in [n]} \mathbf{1}_{R_k(a_k, a_{k+1})}$$

- When i-th feature and label are categorical values:

$$|D| \cdot \mathbf{c}_i[a_i, a_n] = \sum_{\substack{a_1 \in \\ Dom(A_1)}} \sum_{\substack{a_2 \in \\ Dom(A_2)}} \cdots \sum_{\substack{a_{i-1} \in \\ Dom(A_{i-1})}} \sum_{\substack{a_{i+1} \in \\ Dom(A_{i+1})}} \cdots \sum_{a_{n-1} \in Dom(A_{n-1})} \prod_{k \in [n]} \mathbf{1}_{R_k(a_k, a_{k+1})}$$

Just let us notice that $a_{n+1} = a_1$ for notational purposes, $i \in [n-1] \cup \{0\}$, where 0 represents intercept column feature. In cases where one of the features is intercept feature, there is no "skip" in sums over the intercept feature. Here we used $\mathbf{1}_E$ which is the Kronecker delta that evaulates to 1(0) whenever the event E (not) holds. $R_k(a_k, a_{k+1})$ evaluates to 1(0) whenever tuple (a_k, a_{k+1}) is a part of relation R_k. If we wanted FAQs per definition from the slides we would:

- set $\psi_k = \mathbf{1}_{R_k(a_k, a_{k+1})}$, except in the case of ψ_i and ψ_j where the expression above is multiplied by a_i, a_j.
- set $\oplus = \sum$
- set $\otimes = \prod$
- set free variables to corresponding categorical variables if there are any. For instance, in the case when there is one categorical variable a free variable would be that variable.

According to the proposition 3.4 from [1, page 7] complexity is:

$$\mathcal{O}\left(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot \sum_{i,j \in [n-1] \cup \{0\}} \left(N^{\mathrm{faqw}(i,j)} + |\sigma_{ij}|\right) \cdot \log N\right)$$

where $\text{faqw}(i,j) \leq \text{fhtw}(i,j) + c - 1$ according to proposition A.9 from [1], c is a maximum of categorical variables for any σ_{ij}, because they are free variables. In the case when we have only continuous variables faqw = fhtw the previous complexity is then

$$\mathcal{O}\left(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot \sum_{i,j \in \{4,5,\ldots,n-1\} \cup \{0\}} N^2 \cdot \log N\right) = \mathcal{O}\left(n^5 N^2 \cdot \log N\right)$$

This complexity can be get by using InsideOut algorithm with no free variables:

$$\begin{aligned}
D| \cdot \sigma_{ij} = & \sum_{a_1 \in Dom(A_1)} \sum_{a_2 \in Dom(A_2)} \cdots \sum_{a_n \in Dom(A_n)} \prod_{k \in [n]} \psi_k(a_k, a_{k+1}) \\
= & \sum_{a_2 \in Dom(A_2)} \cdots \sum_{a_n \in Dom(A_n)} \prod_{k \in \{2,..,n\}} \psi_k(a_k, a_{k+1}) \cdot \underbrace{\left(\sum_{a_1 \in Dom(A_1)} \psi_1(a_1, a_2)\psi_n(a_1, a_n)\right)}_{\psi_{22}(a_2, a_n)} \\
= & \sum_{a_2 \in Dom(A_2)} \cdots \sum_{a_n \in Dom(A_n)} \psi_{22}(a_2, a_n) \prod_{k \in \{2,\ldots,n-1\}} \psi_k(a_k, a_{k+1}) \quad \mathcal{O}(N^2) \\
= & \sum_{a_3 \in Dom(A_3)} \cdots \sum_{a_n \in Dom(A_n)} \psi_{32}(a_3, a_n) \prod_{k \in \{3,\ldots,n-1\}} \psi_k(a_k, a_{k+1}) \quad \mathcal{O}(N^2) \\
\vdots & \\
= & \sum_{a_{n-1} \in Dom(A_{n-1})} \sum_{a_n \in Dom(A_n)} \psi_{n-12}(a_{n-1}, a_n)\psi_{n-1}(a_n, a_{n-1}) \quad \mathcal{O}(N^2) \\
= & \sum_{a_n \in Dom(A_n)} \psi_{n2} \quad \mathcal{O}(N)
\end{aligned}$$

where

$$\psi_{k2} = \sum_{a_{k-1} \in Dom(A_{k-1})} \psi_{k-1\,2}(a_{k-1}, a_n)\psi_{k-1}(a_{k-1}, a_k), k \in \{3, 4.., n\}$$

In each step where we calculated FAQ, we see that the calculation takes $O(N^2)$ because of the joining on a variable in each step and calculating sum, thus complexity is as it is stated in the proposition 3.4. We could have used indicator projection in each step to remove unnecessary domains from $\psi_{k2}, k \in \{2, 3, ..., n\}$. When we have categorical and continuous case the solution is similar, just in our calculations we would not sum over the free variable, moreover it will be on the top of variable order. Complexity in this case is:

$$\begin{aligned}
\mathcal{O}\left(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot \sum_{i \in \{1,2,3\} j \in \{4,5,n-1\} \cup \{0\}} \left(N^{\text{faqw}(i,j)} + |\sigma_{ij}|\right) \cdot \log N\right) &= \mathcal{O}\left(n^2 \cdot n \cdot 3n \left(N^2 + |\sigma_{ij}|\right) \log N\right) \\
&= \mathcal{O}\left(n^4 \left(N^2 + N\right)\right) \log N) \\
&= \mathcal{O}\left(n^4 N^2 \log N\right)
\end{aligned}$$

When we have two categorical variables we would join relations one by one leaving categorical variables on the top of variable order. At the end join process would be the same except in the case when $\sigma_{13}(a_1, a_3)\sigma_{31}(a_3, a_1)$, where we need to calculate sum over a_2 as the last. Complexity in this case is:

$$\begin{aligned}
\mathcal{O}\left(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot \sum_{i \in \{1,2,3\} j \in \{1,2,3\}} \left(N^{\text{faqw}(i,j)} + |\sigma_{ij}|\right) \cdot \log N\right) &= \mathcal{O}\left(n^2 \cdot n \cdot 9 \left(N^2 + |\sigma_{ij}|\right) \log N\right) \\
&= \mathcal{O}\left(n^3 \left(N^2 + N^2\right)\right) \log N) \\
&= \mathcal{O}\left(n^3 N^2 \log N\right)
\end{aligned}$$

Finally time complexity is:

$$\begin{aligned}
\mathcal{O}\left(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot \sum_{i,j \in [n-1] \cup \{0\}} \left(N^{\text{faqw}(i,j)} + |\sigma_{ij}|\right) \cdot \log N\right) &= \mathcal{O}\left(n^5 N^2 \cdot \log N \ + n^4 N^2 \cdot \log N + n^3 N^2 \log N\right) \\
&= \mathcal{O}\left(n^5 N^2 \cdot \log N\right)
\end{aligned}$$

$|\sigma_{ij}|$ is limited by $|N|$ or $|N|^2$ (depending on a number of free variables), but it can be as low as 1. The same is with the memory complexity where it can go up to $\mathcal{O}(n_d^2)$ because we need to list σ_{ij}. Although, it can be even $\mathcal{O}(n^2)$, depending on database.

1.4 By properties of functional dependency $A_2 \to A_1A_3 \Leftrightarrow A_2 \to A_1, A_2 \to A_3$. This means if we one-hot encode each of categories from variables A_1, A_2, A_3 then $\mathbf{a}_{A_1} = \mathbf{R}_1\mathbf{a}_{A_2}$ and $\mathbf{a}_{A_3} = \mathbf{R}_3\mathbf{a}_{A_2}$ due to functional dependencies. Furthermore, if we concatenate particular matrices $\mathbf{R} = \begin{bmatrix}\mathbf{R}_1 \\ \mathbf{R}_3\end{bmatrix}$ and $\mathbf{a}_{A_1A_3} = \begin{bmatrix}\mathbf{a}_{A_1} \\ \mathbf{a}_{A_3}\end{bmatrix}$, we get

$$\mathbf{a}_{A_1A_3} = \mathbf{R}\mathbf{a}_{A_2} \tag{4}$$

where $\mathbf{a}_{A_2}, \mathbf{a}_{A_1}, \mathbf{a}_{A_3}$ are column vectors corresponding to tuples. Further, we will use notation $\mathbf{a}_i = \mathbf{a}_{A_i}$. Approximation of output based on $\boldsymbol{\theta}s$ is:

$$\begin{aligned}
\hat{y} &= \sum_{i\in[n-1]\setminus\{1,2,3\}\cup\{0\}} \langle\boldsymbol{\theta}_i, \mathbf{a}_i\rangle + \langle\boldsymbol{\theta}_1, \mathbf{a}_1\rangle + \langle\boldsymbol{\theta}_2, \mathbf{a}_2\rangle + \langle\boldsymbol{\theta}_3, \mathbf{a}_3\rangle && (5)\\
&= \sum_{i\in[n-1]\setminus\{1,2,3\}\cup\{0\}} \langle\boldsymbol{\theta}_i, \mathbf{a}_i\rangle + \langle\boldsymbol{\theta}_{13}, \mathbf{a}_{13}\rangle + \langle\boldsymbol{\theta}_2, \mathbf{a}_2\rangle && (6)\\
&= \sum_{i\in[n-1]\setminus\{1,2,3\}\cup\{0\}} \langle\boldsymbol{\theta}_i, \mathbf{a}_i\rangle + \langle\boldsymbol{\theta}_{13}, \mathbf{R}\mathbf{a}_2\rangle + \langle\boldsymbol{\theta}_2, \mathbf{a}_2\rangle && (7)\\
&= \sum_{i\in[n-1]\setminus\{1,2,3\}\cup\{0\}} \langle\boldsymbol{\theta}_i, \mathbf{a}_i\rangle + \boldsymbol{\theta}_{13}^{\mathrm{T}} \cdot (\mathbf{R}\mathbf{a}_2) + \boldsymbol{\theta}_2^{\mathrm{T}} \cdot \mathbf{a}_2 && (8)\\
&= \sum_{i\in[n-1]\setminus\{1,2,3\}\cup\{0\}} \langle\boldsymbol{\theta}_i, \mathbf{a}_i\rangle + (\boldsymbol{\theta}_{13}^{\mathrm{T}}\mathbf{R}) \cdot \mathbf{a}_2 + \boldsymbol{\theta}_2^{\mathrm{T}} \cdot \mathbf{a}_2 && (9)\\
&= \sum_{i\in[n-1]\setminus\{1,2,3\}\cup\{0\}} \langle\boldsymbol{\theta}_i, \mathbf{a}_i\rangle + (\boldsymbol{\theta}_{13}^{\mathrm{T}}\mathbf{R} + \boldsymbol{\theta}_2^{\mathrm{T}}) \cdot \mathbf{a}_2 && (10)\\
&= \sum_{i\in[n-1]\setminus\{1,2,3\}\cup\{0\}} \langle\boldsymbol{\theta}_i, \mathbf{a}_i\rangle + \langle\mathbf{R}^{\mathrm{T}}\boldsymbol{\theta}_{13} + \boldsymbol{\theta}_2, \mathbf{a}_2\rangle && (11)
\end{aligned}$$

In the equation 5 we used formula for prediction of output based on input using learned parameters. In the equation 6 we concatenated parameters and input tuples into column vectors. In the equation 7 we changed variable $\mathbf{a}_{13}$ using equation 4. In the equation 8 we used the fact that $\langle\mathbf{a}, \mathbf{b}\rangle = \mathbf{a}^{\mathrm{T}}\mathbf{b}$. In the equation 9 we used an associativity of matrix multiplication. In the equation 10 we used an distributivity over matrix addition. In the we used the fact $\langle\mathbf{a}, \mathbf{b}\rangle = \mathbf{a}^{\mathrm{T}}\mathbf{b}$ and the fact $(\mathbf{a} \cdot \mathbf{b})^{\mathrm{T}} = \mathbf{b}^{\mathrm{T}} \cdot \mathbf{a}^{\mathrm{T}}$
Let $\boldsymbol{\gamma}_2 = \mathbf{R}^{\mathrm{T}}\boldsymbol{\theta}_{13} + \boldsymbol{\theta}_2$ and let $\boldsymbol{\gamma}_j = \boldsymbol{\theta}_j, j \in \{0, 4, ..., n-1\}$. $\boldsymbol{\theta}_1, \boldsymbol{\theta}_3$ won't appear in the square loss function because they don't appear in $\hat{y}$ as we see in the equation . They still appear in ℓ_2 regularizer. The square loss for each input (tuple in join) is:

$$\left(\sum_{i\in[n-1]\setminus\{1,3\}\cup\{0\}} \langle\boldsymbol{\gamma}_i, \mathbf{a}_i\rangle - \mathbf{a}_n\right)^2$$

In the ℓ_2 regularizer we have:

$$\ell_2 = \frac{\lambda}{2} \sum_{j\in\{0,4,5,...,n\}} ||\boldsymbol{\gamma}_j||_2^2 + ||\boldsymbol{\theta}_{13}||_2^2 + ||\boldsymbol{\gamma}_2 - \mathbf{R}^{\mathrm{T}}\boldsymbol{\theta}_{13}||_2^2$$

By optimizing out (we reach optimal value) $\boldsymbol{\theta}_{13}$ from regularizer we get

$$\begin{aligned}
\frac{1}{\lambda}\frac{\partial\ell_2}{\partial\boldsymbol{\theta}_{13}} &= 0 + \frac{\partial\boldsymbol{\theta}_{13}^{\mathrm{T}}\boldsymbol{\theta}_{13}}{\partial\boldsymbol{\theta}_{13}} + \frac{\partial(\boldsymbol{\gamma}_2 - \mathbf{R}^{\mathrm{T}}\boldsymbol{\theta}_{13})^{\mathrm{T}}(\boldsymbol{\gamma}_2 - \mathbf{R}^{\mathrm{T}}\boldsymbol{\theta}_{13})}{2\partial\boldsymbol{\theta}_{13}} && (12)\\
&= \frac{\partial\boldsymbol{\theta}_{13}^{\mathrm{T}}\boldsymbol{\theta}_{13}}{2\partial\boldsymbol{\theta}_{13}} + \frac{\partial\boldsymbol{\gamma}_2^{\mathrm{T}}\boldsymbol{\gamma}_2}{2\partial\boldsymbol{\theta}_{13}} - 2\frac{\partial(\boldsymbol{\theta}_{13}^{\mathrm{T}}\mathbf{R}\boldsymbol{\gamma}_2)}{2\partial\boldsymbol{\theta}_{13}} + \frac{\partial(\boldsymbol{\theta}_{13}^{\mathrm{T}}\mathbf{R}\mathbf{R}^{\mathrm{T}}\boldsymbol{\theta}_{13})}{2\partial\boldsymbol{\theta}_{13}} && (13)\\
&= \boldsymbol{\theta}_{13} + 0 - \mathbf{R}\boldsymbol{\gamma}_2 + \mathbf{R}\mathbf{R}^{\mathrm{T}}\boldsymbol{\theta}_{13} && (14)\\
&= (\mathbf{I}_{13} + \mathbf{R}\mathbf{R}^{\mathrm{T}})\boldsymbol{\theta}_{13} + \mathbf{R}\boldsymbol{\gamma}_2 = 0 && (15)
\end{aligned}$$

In the equation 12 we rewrote expressions for norms using rule $|\mathbf{v}|_2^2 = \mathbf{v}^{\mathrm{T}}\mathbf{v}$. In the equation 13 we used simple matrix operations (multiply and add). In the equation 14 we used properties $\nabla_{\mathbf{x}}(\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{A}^{\mathrm{T}}\mathbf{x}$, $\nabla_{\mathbf{x}}(\mathbf{x}^{\mathrm{T}}\mathbf{x}) = 2\mathbf{x}$, $\nabla_{\mathbf{x}}(\mathbf{c}^{\mathrm{T}}\mathbf{x}) = \mathbf{c}$. By further solving the equation 15 we get:

$$\boldsymbol{\theta}_{13} = (\mathbf{I}_{13} + \mathbf{R}\mathbf{R}^{\mathrm{T}})^{-1}\mathbf{R}\boldsymbol{\gamma}_2$$

Finally ℓ_2 regularizer is:

$$\frac{\lambda}{2}\left(\sum_{j\in\{0,4,5,\ldots,n-1\}}||\boldsymbol{\gamma}_j||_2^2+\langle(\mathbf{I}_2+\mathbf{R}^{\mathrm{T}}\mathbf{R})^{-1}\boldsymbol{\gamma}_2,\boldsymbol{\gamma}_2\rangle\right)$$

The final derivation is the same as in [1, page 21 C.1 appendix] just in indices of parameters instead of city is 2 and instead of country is 13.
The reparametrized loss function is:

$$\frac{1}{2|D|}\sum_{\mathbf{a}\in D}\left(\sum_{i\in[n-1]\setminus\{1,3\}\cup\{0\}}\langle\boldsymbol{\gamma}_i,\mathbf{a}_i\rangle-\mathbf{a}_n\right)^2+\frac{\lambda}{2}\left(\sum_{j\in\{0,4,5,\ldots,n-1\}}||\boldsymbol{\gamma}_j||_2^2+\langle(\mathbf{I}_2+\mathbf{R}^{\mathrm{T}}\mathbf{R})^{-1}\boldsymbol{\gamma}_2,\boldsymbol{\gamma}_2\rangle\right)$$

We have reduced of categorical variables from three to one, which reduces n_d to $n_w = n-3+|Dom_{A_2}|$ in the case of listing representation

$$\mathcal{O}(N^{\frac{n}{2}}+(n_w)^2\cdot N^{\frac{n}{2}}+n_w\cdot N^{\frac{n}{2}}+n_w^2\cdot t)$$

In the case of FAQs we have removed the most time consuming FAQs, more precisely in the case of 2 categorical variables, we just need to evaluate FAQ for one variable. Thus, if sizes of domains are big in comparison to database size, constant in complexity is reduced drastically.

2.

2.1 The OuMv problem is:

- Input an $n\times n$ Boolean matrix $\mathbf{M}$ and n pairs $(\mathbf{u}_1,\mathbf{v}_1),\ldots,(\mathbf{u}_n,\mathbf{v}_n)$ of Boolean column-vectors of size n arriving one after the other.
- Goal is after seeing each pair $(\mathbf{u}_r,\mathbf{v}_r)$ to output $\mathbf{u}_r^{\mathrm{T}}\mathbf{M}\mathbf{v}_r$.

Conjecture 1. *For any $\gamma>0$, there is no algorithm that solves the OuMv problem in time $\mathcal{O}(n^{3-\gamma})$.*

Let us assume there is an algorithm $\mathcal{A}$ which maintains 4-cycle detection with update time $\mathcal{O}(N^{\frac{1}{2}-\gamma})$, pre-processing time $\mathcal{O}(N^2)$, and answer time $\mathcal{O}(1)$ over a database of size N, unless the OuMv conjecture fails. The algorithm $\mathcal{B}$ using algorithm $\mathcal{A}$ which solves OuMv in subcubic time is:

(1) Insert at most n^2 tuples in R such that $R(A,B)=\{(i,j)|\mathbf{M}(i,j)=1\}$ and in T such that $T(C,D)=\{(j,i)|\mathbf{M}(i,j)=1\}$;
(2) In each round $r\in[n]$:
  (2.1) Delete all tuples in in S,W;
  (2.2) Insert at most $2n$ tuples into S,W $S(B,C)=\{(j,j)|\mathbf{v}_r(j)=1\}$ $W(D,A)=\{(i,i)|\mathbf{u}_r(i)=1\}$;
  (2.3) Check $R\bowtie S\bowtie T\bowtie W\neq\emptyset$. This holds only iff $\mathbf{u}_r^{\mathrm{T}}\mathbf{M}\mathbf{v}_r=1$ $\exists i,j\in[n]$ with $\mathbf{u}_r(i)=1,\mathbf{M}(i,j)=1,$ and $\mathbf{v}_r(j)=1$. Time analysis:

There are at most n^2 tuples inserted at the step (1) and $2n$ at the step 2, thus the size of database is at most $O(n^2)$, because tuples from the step 2 are deleted in each round. A complexity of the step (1) of algorithm $\mathcal{B}$ is:

$$\mathcal{O}(n^2\cdot(n^2)^{\frac{1}{2}-\gamma})=\mathcal{O}(n^2\cdot n^{1-2\gamma})=\mathcal{O}(n^{3-2\gamma})$$

Here we calculated complexity of inserting n^2 tuples to the database. A complexity of the step (2) of algorithm $\mathcal{B}$ is:

$$\mathcal{O}(4n\cdot(n^2)^{\frac{1}{2}-\gamma}+1)=\mathcal{O}(4n^{2-2\gamma}+1)=\mathcal{O}(n^{2-2\gamma})$$

The analysis of the step (2) takes into consideration that there are at most $2n$ tuples to be deleted and inserted from S,W ($\mathbf{u}_r$ and $\mathbf{v}_r$ are filled with ones). Time for n rounds in the step (2) is:

$$\mathcal{O}(n\cdot n^{2-2\gamma})=\mathcal{O}(n^{3-2\gamma})$$

Summing complexities of steps (1) and 2) we get that an overall time complexity is $\mathcal{O}(n^{3-2\gamma})$, which is in contradiction to the claim from the conjecture 1.

Type of S	Type of T	Type of W	Time	Evaluation from left to right
L	L	L	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_c S_L(b',c) \sum_d T_L(c,d) W_L(d,a')$
L	L	H	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_c S_L(b',c) \sum_d T_L(c,d) W_H(d,a')$
L	H	LL	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_c S_L(b',c) \sum_d W_{LL}(d,a') T_H(c,d)$
L	H	LH	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_L T_H W_{LH}}(b',a')$
L	H	HL	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_c S_L(b',c) \sum_d W_{HL}(d,a') T_H(c,d)$
L	H	HH	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_L T_H W_{HH}}(b',a')$
H	LL	LL	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_d W_{LL}(d,a') \sum_c T_{LL}(c,d) S_H(b',c)$
H	LH	LL	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_{LH} W_{LL}}(b',a')$
H	L	LH	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_L W_{LH}}(b',a')$
H	LL	HL	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_d W_{HL}(d,a') \sum_c T_{LL}(c,d) S_H(b',c)$
H	LH	HL	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_{LH} W_{HL}}(b',a')$
H	L	HH	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_L W_{HH}}(b',a')$
H	HL	LL	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_d W_{LL}(d,a') \sum_c T_{HL}(c,d) S_H(b',c)$
H	HH	LL	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_{HH} W_{LL}}(b',a')$
H	H	LH, HH	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_H W_{LL,HH}}(b',a')$
H	HL	HL	$\mathcal{O}(\|D\|^{2\varepsilon})$	$\delta R_*(a',b') \cdot \sum_d W_{HL}(d,a') \sum_c T_{HL}(c,d) S_H(b',c)$
H	HH	HL	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_{HH} W_{HL}}(b',a')$
H	H	HH	$\mathcal{O}(1)$	$\delta R_*(a',b') \cdot V_{S_H T_H W_{HH}}(b',a')$

Table 2: Table of evaluation strategies for update of 4-cycle count

2.2 Let us call the first variables in relations R, S, T, W variables A, B, C, D, respectively . Similarly, let us call the second variables in relations R, S, T, W: B, C, D, A, respectively. Let us partition each relation in 4 partitions based on whether the variable is light or heavy in that relation. In case of a relation, we have 4 possible partitioning. If type of the second variable is not stated, it will include both heavy and light values. We will use the same strategy used for triangle query with auxiliary views and updates of a count by iterating different type of partitions. We will only give an update strategy for cycle count when relation $R(A, B)$ is updated, other relations are updated analogously.
In the table we have complexities for different types of partitions of variables, along with update strategies for the cycle count. The column evaluation strategy will be explained on an example of the first row: Iterate through c values in relation S_L corresponding to updated b', for each of those c values iterate through corresponding d in T_L, for each of those d values iterate though a' (check if exists a tuple (d, a')) in W_L, for each of the iterated values add product of tuples from corresponding relations to the cycle count. In this case we would add for each iterated values a product: $\delta R_*(a', b') \cdot S_L(b', c) T_L(c, d) W_L(d, a')$. Since we have at most $|D|^{\varepsilon}$ c values corresponding to b' in S_L, and $|D|^{\varepsilon}$ d values for each of c values in T_L we have at most $O|D|^{2\varepsilon}$ possible iterations. After we calculated sum of all products, we add this to the cycle count number. Also, we have to update relation R.
In some cases, where previous logic does not work because a number of iterations is at least linear, we have to keep an auxiliary view of cycle counts to which tuple (a, b) belongs (excluding multiple of numbers of (a, b)). Updating auxiliary view in this case when (a, b) is not needed. Although, we need to update cycle count which is updated by adding $\delta R_*(a', b')V(...)$. In addition, we might need to update other views, using the same logic as in the update to non-views. Calculation of initial views/cycle count is $\mathcal{O}(N^2)$ because we need to do a join over a database, which is at most $\mathcal{O}(N^2)$, and list all possible combinations of (a, b) which is at most $\mathcal{O}(N^2)$.

References

[1] M. Abo Khamis, H. Q. Ngo, X. Nguyen, D. Olteanu, and M. Schleich, "In-database learning with sparse tensors," in *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. ACM, 2018, pp. 325–340.