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

Djordje Zivanovic

February 7, 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.

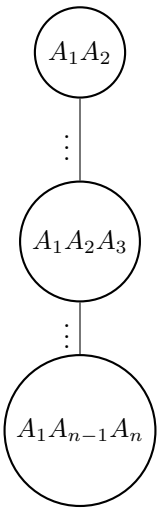

Figure 1: Hypertree decomposition that corresponds to fhtw

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 θ 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 [1, page 7] complexity

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

where faqw $\leq$ fhtw $+ c - 1$ where c is maximum of categorical variables for any σ_{ij}. In the case when we have only continuous variables faqw = fhtw the previous complexity is then

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

For categorical case we would join relations one by one leaving categorical variables (relations to whom they belong) the last with each join taking the most $O(N^2)$ and keeping aggregates (in this case count). By doing this we can get final join in $O(n \cdot N^2 \cdot \log N)$ (which is what the slides claim).[1]

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 $\mathbf{R} = \begin{bmatrix}\mathbf{R}_1 \\ \mathbf{R}_2\end{bmatrix}$ and $\mathbf{a}_{A_1A_3} = \begin{bmatrix}\mathbf{a}_{A_1} \\ \mathbf{a}_{A_3}\end{bmatrix}$, we get $\mathbf{a}_{A_1A_3} = \mathbf{R}\mathbf{a}_{A_2}$. Approximation of output based on θs is:

$$\begin{aligned}
\hat{y} &= \sum_{i \in [n-1] \setminus \{1,2,3\} \cup \{0\}} \langle \theta_i, \mathbf{x}_i \rangle + \langle \theta_1, \mathbf{x}_1 \rangle + \langle \theta_2, \mathbf{x}_2 \rangle + \langle \theta_3, \mathbf{x}_3 \rangle && (4)\\
&= \sum_{i \in [n-1] \setminus \{1,2,3\} \cup \{0\}} \langle \theta_i, \mathbf{x}_i \rangle + \langle \theta_{13}, \mathbf{x}_{13} \rangle + \langle \theta_2, \mathbf{x}_2 \rangle\rangle && (5)\\
&= \sum_{i \in [n-1] \setminus \{1,2,3\} \cup \{0\}} \langle \theta_i, \mathbf{x}_i \rangle + \langle \theta_{13}, \mathbf{R}\mathbf{x}_2 \rangle + \langle \theta_2, \mathbf{x}_2 \rangle\rangle && (6)\\
&= \sum_{i \in [n-1] \setminus \{1,2,3\} \cup \{0\}} \langle \theta_i, \mathbf{x}_i \rangle + \langle \mathbf{R}^{\mathrm{T}}\theta_{13} + \theta_2, \mathbf{x}_2 \rangle\rangle && (7)\\
& && (8)
\end{aligned}$$

Let $\gamma_2 = \mathbf{R}^{\mathrm{T}}\theta_{13} + \theta_2$ and let $\gamma_j = \theta_j, j \in \{0, 4, ..., n-1\}$. Then γ_j won't appear in the loss function because they don't appear in $\hat{y}$ as we see in the equation 7. They still appear in l2 regularizer. In the l2 regularizer we have:

$$\frac{\lambda}{2} \sum_{j \in \{0,4,5,\ldots,n\}} ||\gamma_j||_2^2 + ||\theta_{13}||_2^2 + ||\gamma_2 - \mathbf{R}^{\mathrm{T}}\theta_{13}||$$

By optimizing out (we reach optimal value) θ_{13} from regularizer we get

$$\frac{1}{\lambda}\frac{\partial l2}{\partial \theta_{13}} = \mathbf{R}(\mathbf{R}^{\mathrm{T}}\theta_{13} - \gamma_2) + \theta_{13} = 0$$

which by further solving we get:

$$\theta_{13} = (\mathbf{I} + \mathbf{R}\mathbf{R}^{\mathrm{T}})^{-1}\mathbf{R}\gamma_2$$

Finally l2 regularizer is:

$$\frac{\lambda}{2} \sum_{j \in \{0,4,5,\ldots,n-1\}} ||\gamma_j||_2^2 + \langle (\mathbf{I} + \mathbf{R}^{\mathrm{T}}\mathbf{R})^{-1}, \gamma_2 \rangle$$

(the full derivation in the "Learning Models over Relational Data using Sparse Tensors and Functional Dependencies"). 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 $n_d = n + |Dom_{A_1}| + |Dom_{A_2}| + |Dom_{A_3}|$. We have reduced unnecessary operations from three categorical variables to one, which reduces n_d to $n_w = n + |Dom_{A_2}|$

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

2.

2.1 The OuMv problem is:

- Input an $n \times n$ Boolean matrix $\mathbf{M}$ and n pairs $(\mathbf{u}_1, \mathbf{v}_1), ..., (\mathbf{u}_n, \mathbf{v}_n)$ of Boolean column-vectors of size n arriving one after the other.

[1] Further explanations omitted due time constraints

- 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.

2.2

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.