Warsaw PhD Open Course: From Joins to Aggregates and Optimization Problems
– Exam Paper –

Lecturer: Dan Olteanu, University of Oxford

November 28, 2018

This exam has two questions. Each question is worth 50 marks, with a total of 100 marks achievable for the exam. The marks are to be mapped to a qualitative grading scheme as follows: *very good* means 65–100 marks; *good* means 45–64 marks; *pass* means 25–44 marks; and *fail* means 0–24 marks.

1 Learning Models over Databases (50 marks)

Assume a database with the binary relations

$$R_1(A_1, A_2), \ldots, R_{n-1}(A_{n-1}, A_n), R_n(A_n, A_1).$$

We would like to learn the following ridge linear regression model

$$f_{\boldsymbol{\theta}}(\mathbf{a}) = \sum_{i \in [n]} \langle \boldsymbol{\theta}_i, \mathbf{a}_i \rangle$$

over the training dataset defined by the natural join of the n relations. We consider that the variables $A_1, \ldots, A_{n-1}$ define the features and the variable A_n defines the label. In other words, the model f predicts A_n given the features defined by $A_1, \ldots, A_{n-1}$. The variables A_1, A_2, A_3 are categorical, while all other variables are continuous. The objective function used for training is composed of the square loss function and the ℓ_2 regularizer. Consider using batch gradient descent for learning as in the lecture.

Address the questions and tasks given in the following subsections.

1.1 Widths for the Join Query

Consider the the natural join J of the n relations, where each relation has size N.

- Give the fractional edge cover number of the join J. (5 marks)

- Give the hypertree width of the join J. (5 marks)

1.2 Out-of-Database Learning

Consider the out-of-database learning approach with batch gradient descent.

- Give the (data) complexity of learning the model f over the materialized training dataset as a function of the database size N, where you use t iterations for convergence.

(5 marks)

1.3 In-Database Learning

Consider the in-database learning approach that expresses the objective function and its gradient using the matrix $\mathbf{\Sigma}$ and the vector $\boldsymbol{c}$, as discussed in the lecture.

- Give the functional aggregate queries that define the entries in $\mathbf{\Sigma}$ and $\boldsymbol{c}$. (10 marks)

- Give the time complexities to compute each of these functional aggregate queries as well as the sizes of their results. (10 marks)

Optional question, more speculative:

- Is there a way to share computation across these functional aggregate queries? Detail your thoughts or show explicitly how to achieve this. (NO marks, just fun)

1.4 In-Database Learning under Functional Dependency

Consider the functional dependency: $A_2 \rightarrow A_1, A_3$.

- Show how to re-parameterize the objective function, i.e., the square loss and the ℓ_2 regularizer, under this dependency. (10 marks)

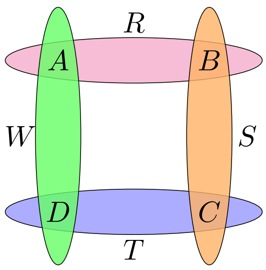
- What is the effect of this dependency on the time complexity of learning the model f inside the database? (5 marks)

2 Counting 4-cycles under Updates (50 marks)

Consider the following functional aggregate query φ that counts the number of cycles with one edge per each of the four factors R, S, T, and W:

$$\varphi = \sum_{a,b,c,d} R(a,b) \cdot S(b,c) \cdot T(c,d) \cdot W(d,a)$$

The hypergraph of the join part of φ is a 4-cycle, as depicted below:



The functions R, S, T, and W map pairs of values from a finite domain Dom to non-zero numbers in $\mathbb{Z}$. The query φ is also a function that maps the empty tuple to a non-zero number in $\mathbb{Z}$; it is empty in case the number of 4-cycles in zero.

Recall from the lecture that a single-tuple update $\delta R = \{(a_1, b_1) \rightarrow v\}$ to R is a function that maps the tuple (a_1, b_1) to a value $v \in \mathbb{Z}$. If $v < 0$, then δR is a delete, whereas if $v > 0$, then δR is an insert. Also recall that the size of the database is given by the number of tuples that are mapped by any of the four functions to non-zero numbers.

Consider the problem of incrementally maintaining φ under single-tuple updates to any of the four functions R, S, T, and W.

2.1 Lower Bound

Show that for any $\gamma > 0$, there is no algorithm that can maintain the 4-cycle count query φ with update time $\mathcal{O}(N^{1/2-\gamma})$, pre-processing time $\mathcal{O}(N^2)$, and answer time $\mathcal{O}(1)$ over a database of size N, unless the OuMv conjecture fails. (15 marks)

Hint: You may use a variation of the proof given in the lecture to show this lower bound.

2.2 Upper Bound

Give an algorithm that can maintain the 4-cycle count query φ with amortized sublinear update time, quadratic pre-processing time, and constant answer time.

Any function that is sublinear in the size N of the database would be sufficient for the amortized update time. It might not be possible to have this time $\mathcal{O}(N^{1/2})$ as for the case of the triangle count discussed in the lecture.

- Give the evaluation strategy for a single-tuple update to any of the four input functions, together with the pre-materialized views that are used. (15 marks)
- Give an analysis of the update time complexity for the proposed evaluation strategy. (15 marks)
- Give an analysis of the space and update time complexities for the views. (5 marks)

Hint: You may use a variation of the algorithm shown in the lecture for counting triangles, where the functions are partitioned on their join variables based on the skew of the join values.