In-Database Learning with Sparse Tensors

Mahmoud Abo Khamis[1] Hung Q. Ngo[1] XuanLong Nguyen[2]
Dan Olteanu[3] Maximilian Schleich[3]

[1]LogicBlox, Inc. [2]University of Michigan [3]University of Oxford
{mahmoud.abokhamis,hung.ngo}@logicblox.com, xuanlong@umich.edu,
{dan.olteanu,max.schleich}@cs.ox.ac.uk

ABSTRACT

In-database analytics is of great practical importance as it avoids the costly repeated loop data scientists have to deal with on a daily basis: select features, export the data, convert data format, train models using an external tool, reimport the parameters. It is also a fertile ground of theoretically fundamental and challenging problems at the intersection of relational and statistical data models.

This paper introduces a unified framework for training and evaluating a class of statistical learning models inside a relational database. This class includes ridge linear regression, polynomial regression, factorization machines, and principal component analysis. We show that, by synergizing key tools from relational database theory such as schema information, query structure, recent advances in query evaluation algorithms, and from linear algebra such as various tensor and matrix operations, one can formulate in-database learning problems and design efficient algorithms to solve them.

The algorithms and models proposed in the paper have already been implemented inside the LogicBlox database engine and used in retail-planning and forecasting applications, with significant performance benefits over out-of-database solutions that require the costly data-export loop.

1. INTRODUCTION

Although both disciplines of databases and statistics occupy foundational roles for the emerging field of data science, they are largely seen as complementary. Most fundamental contributions made by statisticians and machine learning researchers are abstracted away from the underlying infrastructure for data management. However, there is undoubtedly clear value in tight integration of statistics and database models and techniques. A prime example of such a tight integration is provided by in-database analytics, which is receiving an increasing interest in both academia and industry [2, 32, 43]. This is motivated by the realization that in many practical cases data resides inside databases and bringing the analytics closer to the data saves nontrivial time usually spent on data import/export at the interface between database systems and statistical packages [28]. A complementary realization is that large chunks of statistical machine learning code can be expressed as relational queries and computed inside the database [17, 19, 33, 47]. In-database analytics problems naturally lend themselves to a systematic investigation using the toolbox of concepts and techniques developed by the database theorist, and by synergizing ideas from both relational and statistical data modeling. To solve optimization problems over relational data, one can exploit database schema information, functional dependencies, state-of-the-art query evaluation algorithms, and well-understood complexity analysis.

Our *conceptual contribution* in this paper is the introduction of a unified in-database framework for training and evaluating a class of statistical learning models. This class, commonly used in LogicBlox retail-planning and forecasting applications [9], includes ridge linear regression, polynomial regression, factorization machines, classification, and principal component analysis.

In such applications, the training dataset is the result of a feature extraction query over the database. Typical databases include weekly sales data, promotions, and product descriptions. A retailer would like to compute a parameterized model, which can predict, for instance, the additional demand generated for a given product due to promotion. As is prevalent in practical machine learning, the models are trained using a first-order optimization algorithm such as batch or stochastic gradient descent, in part because their convergence rates are dimension-free (for well-behaved objectives). This is a crucial property given the high-dimensionality of our problem as elaborated next.

The main *computational challenge* posed by in-database analytics is the large number of records and of features in the training dataset. There are two types of features: continuous (quantitative) such as price and revenue; and categorical (qualitative) such as colors, cities, and countries.[1] While continuous features allow for aggregation over their domains, categorical features cannot be aggregated together. To accommodate the latter, the state-of-the-art approach is to one-hot encode their active domain: each value in the active domain of an attribute is encoded by an indicator vector whose dimension is the size of the domain. For instance, the colors in the domain $\{\text{red}, \text{green}, \text{blue}\}$ can be represented by indicator vectors $[1, 0, 0]$ for red, $[0, 1, 0]$ for green, and $[0, 0, 1]$ for blue. The one-hot encoding amounts to a relational representation of the training dataset with one new attribute per distinct category of each categorical feature and with wide tuples whose values are mostly 0.

[1]Most of the features in our clients' datasets are categorical.

This entails huge redundancy due to the presence of the many 0 values. It also blurs the usual database-theory distinction between schema and data, since the schema can be as large as the input database.

Closely related to the computational challenge is a *cultural challenge*: the feasibility of in-database analytics is often called into question. In terms of pure algorithmic performance, why would an in-database optimization algorithm be more efficient than an out-of-database optimization implementation, given the plethora of tools and techniques widely available?

Our short answer to these challenges is that, for a large class of feature extraction queries, it is possible to train a model in time *sub-linear* in the output size of the feature extraction query! More concretely, our approach entails *three database-centric technical contributions*. First, we exploit join dependencies in the training dataset to asymptotically improve the per-iteration computation time of a gradient descent algorithm. Second, we exploit functional dependencies present in the database to reduce the dimensionality of the underlying optimization problem by only optimizing for those parameters that functionally determine the others and by subsequently recovering the latter based on their dependencies on the former. Third, we address the shortcomings of one-hot encoding by expressing the sum-product aggregates used to compute the gradient and point evaluation as functional aggregate queries (FAQs) [6]. The aggregates over continuous features are expressed as FAQs without free variables and their computation yields scalar values. In contrast, aggregates over categorical features originating from a set S of database attributes are expressed as FAQs with free (i.e., group-by) variables S. The tuples in the result of such FAQs are combinations of categorical values that occur in the database. The ensemble of FAQs defining the gradient form a *sparse tensor representation and computation solution* with lower space and time complexity than solutions based on one-hot encoding. In particular, the complexity of our end-to-end solution can be arbitrarily smaller than that of materializing the result of the feature extraction query.

Organization. The structure of the paper follows our contributions. Section 2 introduces our unified framework for in-database analytics. Section 3 introduces our sparse tensor representation and computation approach. Section 4 shows how to exploit functional dependencies to reduce the dimensionality for factorization machines and polynomial regression models.

Proofs of all theorems are in Appendix and the extended Technical Report of this paper [37]. The results presented in this paper form the foundation of an in-database analytics prototype extending the LogicBlox runtime engine. In experiments with real data, they show up to three orders of magnitude performance improvements over state-of-the-art competitors for polynomial regression models and factorization machines [37].

Related work. It has been recently acknowledged that database theory can effectively contribute to the arms race for in-database analytics [2]. Recent works highlight the potential of applying key database theory tools to this growing research of practical interest, e.g., the formal relational framework for classifier engineering [31] and in-database factorized learning of regression models with low data complexity [47].

Most related efforts in the (database and distributed) systems communities are on designing systems to support machine learning libraries or statistical packages on top of large-scale database architectures, e.g., MLLib [35] and DeepDist [36] on Spark [49], GLADE [44], TensorFlow [1], and SystemML [13,29]. This approach relies on the expensive data export/import at the interface between the machine learning library process and the database system process: The feature extraction query is computed inside a database system, its result exported and imported into the data format of a machine learning library, where the model is learned. This approach is very expensive and unnecessarily redundant due to the import/export step. Our approach differs from all these efforts in that it tightly integrates the analytics with the database query engine.

There are three lines of prior work closest to ours.

One line of work investigates the ability to express parts of analytical tasks within query languages. An important bulk of early work is on query languages with data mining, also called descriptive or backward-looking analytics, capabilities [15] and in-database data mining solutions, e.g., frequent itemsets [41] and association rule mining [8]. More recent work investigated how to (partly) express predictive (forward-looking) analytics, such as learning regression models and Naïve Bayes classification, together with the feature extraction query as a single optimized query with joins and sum-product aggregates [33,47]. MADlib [28] casts analytics as user-defined aggregate functions (UDAFs) that can be used in SQL queries and executed inside PostgreSQL. These UDAFs remain black boxes for the underlying query engine, which has to compute the feature extraction query and delegate the UDAF computation on top of the query result to the MADLib's specialized code.

A second line of work exploits join dependencies for efficient in-database analytics. Join dependencies form the basis of the theory of (generalized) hypertree decompositions [23] and factorized databases [40], with applications such as inference in probabilistic graphical models, CSP, SAT, and databases. In databases, they have been originally used as a tractability yardstick for Boolean conjunctive queries [23] and more recently for the computation and result representation of queries with free variables [40], with group-by aggregates [6,11], and with order-by clauses [11]. Our approach builds on earlier work that exploits join dependencies for learning linear regression models with continuous features [47]. Factorization machines [46] represent one practical regression model used for LogicBlox analytics and that we investigate in this paper. In contrast to polynomial regression models, factorization machines factorize the space of model parameters to better capture data correlations. We further this idea by also factorizing

the training dataset, which relies on join dependencies present in the data.

A third line of prior work uses functional dependencies (FDs) to avoid key-foreign key joins and reduce the number of features in Naïve Bayes classification and feature selection [34]. We consider the effect of FDs on the reparameterization of regression models, where a non-trivial development is on the effect of FDs on the model (non-linear) regularization function.

Several state-of-the-art machine learning systems use a sparse representation of the input data to avoid redundancy introduced by one-hot encoding [18, 45]. In our setting, however, such systems require an additional data transformation step after the result of the feature extraction query is exported. This additional step is time consuming and makes the use of such systems very inefficient in many practical applications. In statistics and machine learning, there is a rich literature on learning with sparse and/or multilinear structures [27]. Such methods complement our framework and it would be of interest to leverage some of these techniques to our schema-driven sparsity.

Finally, there is a large collection of gradient-based methods proposed in the optimization literature. The description of our approach assumes batch gradient descent (BGD), though our insights are applicable to other methods, including Quasi-Newton algorithms. The main rationale for our choice is simplicity and good statistical properties. When combined with backtracking line search (as we do in this paper) or second-order gradient estimation (as in Quasi-Newton methods), BGD is guaranteed to converge to a minimum with linear asymptotic convergence rate. A naïve computation of the gradient requires a full pass over the data, which can be inefficient in large-scale analytics. A popular alternative is stochastic gradient descent (SGD), which estimates the gradient with a randomly selected mini-batch of training samples. The convergence of SGD, however, is noisy, requires careful setting of hyperparameters, and does not achieve the linear asymptotic convergence rate of BGD [14]. In our setting, the entire BGD execution can be arbitrarily faster than one SGD iteration over the result of the feature extraction query.

2. PROBLEM FORMULATION

The goal of this section is to present a general optimization formulation encompassing a range of machine learning tasks, and then to lay out a versatile mathematical representation suitable for the in-database treatment of these tasks.

Notational Convention. Bold face letters, e.g., $\mathbf{x}$, $\boldsymbol{\theta}$, $\mathbf{x}_i$, $\boldsymbol{\theta}_j$, denote vectors or matrices, and normal face letters, e.g., x_i, θ_j, $\theta_i^{(j)}$, denote scalars. For any positive integer n, $[n]$ denotes the set $\{1, \ldots, n\}$. For any set S and positive integer k, $\binom{S}{k}$ denotes the collection of all k-subsets of S. We use the following matrix operations: $\otimes$ denotes the Kronecker/tensor product; $\circ$ the Hadamard product; $\star$ the Khatri-Rao product; and $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product of two matrices, which reduces to the vector inner product when the matrices have one column each.

Let S be a finite set and Dom be any domain, then $\mathbf{a}_S = (a_j)_{j \in S} \in \mathsf{Dom}^{|S|}$ is a *tuple* indexed by S, whose components are in Dom. If S and T are disjoint, and given tuples $\mathbf{a}_S$ and $\mathbf{a}_T$, then tuple $(\mathbf{a}_S, \mathbf{a}_T)$ is interpreted naturally as the tuple $\mathbf{a}_{S \cup T}$. The tuple $\mathbf{0}_S$ is the all-0 tuple indexed by S. If $S \subseteq G$, then the tuple $\mathbf{1}_{S|G}$ is the characteristic vector of the subset S, i.e., $\mathbf{1}_{S|G}(v) = 1$ if $v \in S$, and 0 if $v \in G - S$.

We make extensive use of basic concepts and results from matrix calculus summarized in Appendix A, where we also discuss a connection between tensor computation and the FAQ-framework [6].

Feature Extraction Query. We consider the setting where the training dataset D used as input to machine learning is the result of a natural join query Q, called *feature extraction query*, over a relational database I. Each tuple $(\mathbf{x}, y) \in D$ contains a scalar response (regressand) y and a tuple $\mathbf{x}$ encoding features (regressors).

We use standard notation for join hypergraphs. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ denote the hypergraph of the query Q, where $\mathcal{V}$ is the set of variables occurring in Q and $\mathcal{E}$ is the set of hyperedges with one hyperedge per set of variables in a relation symbol R in the body of Q. We denote by $V \subseteq \mathcal{V}$ the subset of variables selected as features, and let $n = |V|$. The features in V corresponding to qualitative attributes in I are called *categorical*, while those corresponding to quantitative attributes in I are *continuous*. Let N be the size of the largest input relation R in Q.

Example 1. Consider the following natural join query Q that is a highly simplified version of a feature extraction query typically used at LogicBlox:

$$Q(\mathsf{sku}, \mathsf{store}, \mathsf{day}, \mathsf{color}, \mathsf{quarter}, \mathsf{city}, \mathsf{country}, \mathit{unitsSold}) \leftarrow R_1(\mathsf{sku}, \mathsf{store}, \mathsf{day}, \mathit{unitsSold}), R_2(\mathsf{sku}, \mathsf{color}), R_3(\mathsf{day}, \mathsf{quarter}), R_4(\mathsf{store}, \mathsf{city}), R_5(\mathsf{city}, \mathsf{country}).$$

Relation R_1 records the number of units of a given `sku` (stock keeping unit) sold at a `store` on a particular `day`. The retailer is a global business, so it has stores in different cities and countries. One objective is to predict the number of blue units to be sold next year in the Fall quarter in Berlin. The response is the continuous variable *unitsSold*, $\mathcal{V}$ is the set of all variables, and $V = \mathcal{V} - \{\mathit{unitsSold}, \mathsf{day}\}$, all of which are categorical.

2.1 Formulation with continuous features

Let $m \geq n$ be an integer. The feature map $h : \mathbb{R}^n \to \mathbb{R}^m$ transforms the raw input vector $\mathbf{x} \in \mathbb{R}^n$ into an m-vector of "monomial features" $h(\mathbf{x}) = (h_j(\mathbf{x}))_{j \in [m]}$. Each component h_j is a multivariate *monomial* designed to capture the *interactions* among dimensions of input $\mathbf{x}$. In particular, we write $h_j(\mathbf{x}) := \prod_{i \in [n]} x_i^{a_j(i)}$, where degree $a_j(i)$ represents the level of participation of input dimension i in the j-th monomial feature. Let p be the number of parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p) \in \mathbb{R}^p$, which produce the coefficients associated with features h via

parameter map $g : \mathbb{R}^p \to \mathbb{R}^m$, $g(\boldsymbol{\theta}) = (g_j(\boldsymbol{\theta}))_{j\in[m]}$. Each component g_j is a multivariate *polynomial* of $\boldsymbol{\theta}$.

A large number of machine learning tasks learn a functional quantity of the form $\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle$, where the parameters $\boldsymbol{\theta}$ are obtained by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ with

$$J(\boldsymbol{\theta}) = \sum_{(\mathbf{x},y)\in D} \mathcal{L}\left(\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle, y\right) + \Omega(\boldsymbol{\theta}). \quad (1)$$

$\mathcal{L}$ is a loss function (e.g., square loss) and Ω is a regularizer (such as ℓ_1- or ℓ_2-norm of $\boldsymbol{\theta}$). For square loss and ℓ_2-regularization, $J(\boldsymbol{\theta})$ becomes:

$$J(\boldsymbol{\theta}) = \frac{1}{2|D|}\sum_{(\mathbf{x},y)\in D} (\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle - y)^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2. \quad (2)$$

Example 2. The *ridge linear regression* (LR) model with response y and regressors $x_1, \ldots, x_n$ has $p = n+1$, parameters $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_n)$. For convenience, we set $x_0 = 1$ corresponding to the bias parameter θ_0. Then, $m = n+1$, $\mathbf{g}(\boldsymbol{\theta}) = \boldsymbol{\theta}$, and $h(\mathbf{x}) = \mathbf{x}$.

Example 3. The *degree-d polynomial regression* (PR^d) model with response y and regressors $x_0 = 1, x_1, \ldots, x_n$ has $p = m = 1+n+n^2+\cdots+n^d$ parameters $\boldsymbol{\theta} = (\theta_{\mathbf{a}})$, where $\mathbf{a} = (a_1, \ldots, a_n)$ is a tuple of non-negative integers such that $\|\mathbf{a}\|_1 \leq d$. In this case, $g(\boldsymbol{\theta}) = \boldsymbol{\theta}$, while the components of h are given by $h_{\mathbf{a}}(\mathbf{x}) = \prod_{i=1}^n x_i^{a_i}$.

Example 4. The *degree-2 rank-r factorization machines* (FaMa_r^2) model with regressors $x_0 = 1, x_1, \ldots, x_n$ and regressand y has parameters $\boldsymbol{\theta}$ consisting of θ_i for $i \in \{0, \ldots, n\}$ and $\theta_i^{(l)}$ for $i \in [n]$ and $l \in [r]$. Training FaMa_r^2 corresponds to minimizing the following $J(\boldsymbol{\theta})$:

$$\frac{1}{2|D|}\sum_{(\mathbf{x},y)\in D}\left(\sum_{i=0}^n \theta_i x_i + \sum_{\substack{\{i,j\}\in\binom{[n]}{2}\\ \ell\in[r]}} \theta_i^{(\ell)}\theta_j^{(\ell)} x_i x_j - y\right)^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2.$$

This loss function follows Equation (2) with $p = 1 + n + rn$, $m = 1 + n + \binom{n}{2}$, and the parameter maps

$$h_S(\mathbf{x}) = \prod_{i\in S} x_i, \text{ for } S \subseteq [n], |S| \leq 2$$

$$g_S(\boldsymbol{\theta}) = \begin{cases} \theta_0 & \text{when } |S| = 0 \\ \theta_i & \text{when } S = \{i\} \\ \sum_{\ell=1}^r \theta_i^{(\ell)}\theta_j^{(\ell)} & \text{when } S = \{i,j\}. \end{cases}$$

Example 5. *Classification methods* such as support vector machines (SVM), logistic regression and Adaboost also fall under the same optimization framework, but with different choices of loss $\mathcal{L}$ and regularizer Ω. Typically, $\Omega(\boldsymbol{\theta}) = \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$. Restricting to binary class labels $y \in \{\pm 1\}$, the loss function $\mathcal{L}(\gamma, y)$, where $\gamma := \langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle$, takes the form $\mathcal{L}(\gamma, y) = \max\{1 - y\gamma, 0\}$ for SVM, $\mathcal{L}(\gamma, y) = \log(1 + e^{-y\gamma})$ for logistic regression and $\mathcal{L}(\gamma, y) = e^{-y\gamma}$ for Adaboost.

Example 6. Various unsupervised learning techniques can be expressed as iterative optimization procedures according to which each iteration is reduced to an optimization problem of the generic form given above. For example, the *Principal Component Analysis (PCA)* requires solving the following optimization problem to obtain a principal component direction

$$\max_{\|\boldsymbol{\theta}\|=1} \boldsymbol{\theta}^\top \boldsymbol{\Sigma}\boldsymbol{\theta} = \max_{\boldsymbol{\theta}\in\mathbb{R}^p}\min_{\lambda\in\mathbb{R}} \boldsymbol{\theta}^\top\boldsymbol{\Sigma}\boldsymbol{\theta} + \lambda(\|\boldsymbol{\theta}\|^2 - 1),$$

where $\boldsymbol{\Sigma} := \frac{1}{|D|}\sum_{\mathbf{x}\in D}\mathbf{x}\mathbf{x}^\top$ is the (empirical) correlation matrix of the given data. Although there is no response/class label y, within each iteration of the above iteration, for a fixed λ, there is a loss function $\mathcal{L}$ acting on feature vector $h(\mathbf{x})$ and parameter vector $g(\boldsymbol{\theta})$, along with a regularizer Ω. Specifically, we have $h(\mathbf{x}) = \boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$, $g(\boldsymbol{\theta}) = \boldsymbol{\theta}\otimes\boldsymbol{\theta} \in \mathbb{R}^{p\times p}$, $\mathcal{L} = \langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle_F$, where the Frobenius inner product is now employed. In addition, $\Omega(\boldsymbol{\theta}) = \lambda(\|\boldsymbol{\theta}\|^2 - 1)$.

2.2 Categorical features

The active domain of a categorical feature/variable consists of a set of possible values or categories. For example, vietnam, england, and usa are possible categories of the categorical feature country. Categorical features constitute the vast majority (up to 99%) of features we see in LogicBlox's machine learning applications.

It is common practice to one-hot encode categorical variables [26]. Whereas a continuous variable such as salary is mapped to a scalar value x_{salary}, a categorical variable such as country is mapped to an indicator vector $\mathbf{x}_{\mathsf{country}}$ – a vector of binary values indicating the category that the variable takes on. For example, if the active domain of country consists of vietnam, england, and usa, then $\mathbf{x}_{\mathsf{country}} = [x_{\mathsf{vietnam}}, x_{\mathsf{england}}, x_{\mathsf{usa}}] \in \{0,1\}^3$. If a tuple in the training dataset has country = "england", then $\mathbf{x}_{\mathsf{country}} = [0,1,0]$ for that tuple.

In general, the feature vector $\mathbf{x}$ in a tuple $(\mathbf{x}, y) \in D$ has the form $\mathbf{x} = (\mathbf{x}_c)_{c\in V}$, where each component $\mathbf{x}_c$ is an indicator vector if c is a categorical variable and a scalar otherwise. Similarly, each component of the parameter vector $\boldsymbol{\theta}$ becomes a matrix (or a vector if the matrix has one column).

2.3 Tensor product representation

We accommodate a mix of continuous and categorical features in our problem formulation (2) by replacing arithmetic product by tensor product in the component functions of the parameter map g and the feature map h. Specifically, monomials h_j now take the form

$$h_j(\mathbf{x}) = \bigotimes_{f\in V} \mathbf{x}_f^{\otimes a_j(f)} \quad (3)$$

with degree vector $\mathbf{a}_j = (a_j(f))_{f\in V} \in \mathbb{N}^n$. For each $j \in [m]$, the set $V_j := \{f \in V \mid a_j(f) > 0\}$ consists of features that participate in the interaction captured by the (hyper-) monomial h_j. Let $C \subseteq V$ denote the set of categorical variables and $C_j := C \cap V_j$ the subset of categorical variables in V_j. For $f \in C_j$, h_j represents $\prod_{f\in C_j} |\pi_f(D)|$ many monomials, one for each combination of the categories, where $\pi_f(D)$ denotes the projection of D onto variable f. Due to one-hot encoding,

each element in the vector $\mathbf{x}_f$ for a categorical variable f is either 0 or 1, and $\mathbf{x}_f^{a_j(f)} = \mathbf{x}_f$ for $a_j(f) > 0$. Hence, h_j can be simplified as follows:

$$h_j(\mathbf{x}) = \prod_{f \in V_j - C_j} x_f^{a_j(f)} \cdot \bigotimes_{f \in C_j} \mathbf{x}_f. \tag{4}$$

Note that we use x_f instead of boldface $\mathbf{x}_f$ since each variable $f \in V_j - C_j$ is continuous.

Example 7. For illustration, consider a query that extracts tuples over schema (country, a, b, c, color) from the database, where country and color are categorical variables, while a, b, c are continuous variables. Moreover, there are two countries vietnam and england, and three colors red, green, and blue in the training dataset D. Consider three of the possible feature functions:

$$h_1(\mathbf{x}) = \mathbf{x}_{\text{country}} \otimes x_a^2 x_c \tag{5}$$
$$h_2(\mathbf{x}) = \mathbf{x}_{\text{country}} \otimes \mathbf{x}_{\text{color}} \otimes x_b \tag{6}$$
$$h_3(\mathbf{x}) = x_b x_c. \tag{7}$$

Under the one-hot encoding, the schema of the tuples becomes (vietnam, england, a, b, c, red, green, blue).

Equation (4) says that the functions h_1 and h_2 are actually encoding 8 functions:

$$\begin{aligned}
h_{1,\text{vietnam}}(\mathbf{x}) &= x_{\text{vietnam}} x_a^2 x_c \\
h_{1,\text{england}}(\mathbf{x}) &= x_{\text{england}} x_a^2 x_c \\
h_{2,\text{vietnam,red}}(\mathbf{x}) &= x_{\text{vietnam}} x_{\text{red}} x_b \\
h_{2,\text{vietnam,green}}(\mathbf{x}) &= x_{\text{vietnam}} x_{\text{green}} x_b \\
h_{2,\text{vietnam,blue}}(\mathbf{x}) &= x_{\text{vietnam}} x_{\text{blue}} x_b \\
h_{2,\text{england,red}}(\mathbf{x}) &= x_{\text{england}} x_{\text{red}} x_b \\
h_{2,\text{england,green}}(\mathbf{x}) &= x_{\text{england}} x_{\text{green}} x_b \\
h_{2,\text{england,blue}}(\mathbf{x}) &= x_{\text{england}} x_{\text{blue}} x_b.
\end{aligned}$$

We elaborate the tensor product representation for the considered learning models.

Example 8. In linear regression, parameter $\boldsymbol{\theta}$ is a vector of vectors: $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_n]$. Since our inner product is Frobenius, when computing $\langle \boldsymbol{\theta}, \mathbf{x} \rangle$ we should be multiplying, for example, θ_{usa} with x_{usa} correspondingly.

Example 9. In polynomial regression, the parameter $\boldsymbol{\theta}$ is a vector of tensors (i.e., high-dimensional matrices). Consider for instance the second order term $\theta_{ij} x_i x_j$. When both i and j are continuous, θ_{ij} is just a scalar. Now, suppose i is country and j is color. Then, the model has terms $\theta_{\text{vietnam,red}} x_{\text{vietnam}} x_{\text{red}}$, $\theta_{\text{usa,green}} x_{\text{usa}} x_{\text{green}}$, and so on. **All** these terms are captured by the Frobenius inner product $\langle \boldsymbol{\theta}_{ij}, \mathbf{x}_i \otimes \mathbf{x}_j \rangle$. The component $\boldsymbol{\theta}_{ij}$ is a matrix whose number of entries is the number of pairs (country, color) that appear together in some tuple in the training dataset. This number can be much less than the product of the numbers of countries and of colors in the input database.

Example 10. Consider the FaMa_r^2 model from Example (4), but now with categorical variables. From the previous examples, we already know how to interpret the linear part $\sum_{i=0}^n \theta_i x_i$ of the model when features are categorical. Consider a term in the quadratic part such as $\sum_{\ell \in [r]} \theta_i^{(\ell)} \theta_j^{(\ell)} x_i x_j$. When i and j are categorical, the term becomes $\left\langle \sum_{\ell \in [r]} \boldsymbol{\theta}_i^{(\ell)} \otimes \boldsymbol{\theta}_j^{(\ell)}, \mathbf{x}_i \otimes \mathbf{x}_j \right\rangle$.

3. FACTORIZED OPTIMIZATION

In this section we introduce our solution to learning statistical models for the setting of square loss function $J(\boldsymbol{\theta})$ and ℓ_2-norm as in (2). We use a gradient-based optimization algorithm that employs the first-order gradient information to optimize the loss function $J(\boldsymbol{\theta})$. It repeatedly updates the parameters $\boldsymbol{\theta}$ by some step size α in the direction of the gradient $\nabla J(\boldsymbol{\theta})$ until convergence. To guarantee convergence, it uses backtracking line search to ensure that α is sufficiently small to decrease the loss for each step. Each update step requires two computations: (1) *Point evaluation*: Given $\boldsymbol{\theta}$, compute the scalar $J(\boldsymbol{\theta})$; and (2) *Gradient computation*: Given $\boldsymbol{\theta}$, compute the vector $\nabla J(\boldsymbol{\theta})$. In particular, we use the batch gradient descent (BGD) algorithm with the Armijo line search condition and the Barzilai-Borwein step size adjustment [12,20], as depicted in Algorithm 1. Quasi-Newton optimization algorithms (e.g., L-BFGS) and other common line search conditions are also applicable in our framework. We refer the reader to the excellent review article [22] for more details on fast implementations of the gradient-descent method.

Algorithm 1: BGD with Armijo line search.

```
θ ← a random point;
while not converged yet do
    α ← next step size;
    d ← ∇J(θ);
    while ( J(θ − αd) ≥ J(θ) − (α/2) ‖d‖_2^2 ) do
        α ← α/2          // line search;
    end
    θ ← θ − αd;
end
```

3.1 Continuous features

To illustrate the main idea, we first consider the case without categorical features. We rewrite (2) to factor out the data-dependent part of the point evaluation and gradient computation. Recall that, for $j \in [m]$, h_j denotes the jth component function of the vector-valued function h, and h_j is a multivariate monomial in $\mathbf{x}$.

Theorem 3.1. *Let $J(\boldsymbol{\theta})$ be the function in (2). Define the matrix $\boldsymbol{\Sigma} = (\sigma_{ij})_{i,j \in [m]}$, the vector $\mathbf{c} = (c_i)_{i \in [m]}$, and the scalar s_Y by*

$$\boldsymbol{\Sigma} = \frac{1}{|D|} \sum_{(\mathbf{x},y) \in D} h(\mathbf{x}) h(\mathbf{x})^\top \tag{8}$$

$$\boldsymbol{c} = \frac{1}{|D|} \sum_{(\mathbf{x},y) \in D} y \cdot h(\mathbf{x}) \tag{9}$$

$$s_Y = \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} y^2. \tag{10}$$

Then,

$$J(\boldsymbol{\theta}) = \frac{1}{2} g(\boldsymbol{\theta})^\top \boldsymbol{\Sigma} g(\boldsymbol{\theta}) - \langle g(\boldsymbol{\theta}), \mathbf{c}\rangle + \frac{s_Y}{2} + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \tag{11}$$

$$\boldsymbol{\nabla} J(\boldsymbol{\theta}) = \frac{\partial g(\boldsymbol{\theta})^\top}{\partial \boldsymbol{\theta}} \boldsymbol{\Sigma} g(\boldsymbol{\theta}) - \frac{\partial g(\boldsymbol{\theta})^\top}{\partial \boldsymbol{\theta}} \mathbf{c} + \lambda \boldsymbol{\theta}. \tag{12}$$

Note that $\frac{\partial g(\boldsymbol{\theta})^\top}{\partial \boldsymbol{\theta}}$ is a $p \times m$ matrix, and $\boldsymbol{\Sigma}$ is an $m \times m$ matrix. Statistically, $\boldsymbol{\Sigma}$ is related to the covariance matrix, $\mathbf{c}$ to the correlation between the response and the regressors, and s_Y to the empirical second moment of the response variable. Theorem 3.1 allows us to compute the two key steps of BGD *without* scanning through the data again, because the quantities $(\boldsymbol{\Sigma}, \mathbf{c}, s_Y)$ can be computed efficiently in a preprocessing step *inside the database* as aggregates over the query Q. We shall elaborate on this point further in Section 3.3.

When g is the identity function, i.e., the model is linear, as is the case in PR (and thus LR) model, (11) and (12) become particularly simple:

Corollary 3.2. *In a linear model (i.e., $g(\boldsymbol{\theta}) = \boldsymbol{\theta}$),*

$$J(\boldsymbol{\theta}) = \frac{1}{2} \boldsymbol{\theta}^\top \boldsymbol{\Sigma} \boldsymbol{\theta} - \langle \boldsymbol{\theta}, \mathbf{c}\rangle + \frac{s_Y}{2} + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 \tag{13}$$

$$\boldsymbol{\nabla} J(\boldsymbol{\theta}) = \boldsymbol{\Sigma} \boldsymbol{\theta} + \lambda \boldsymbol{\theta} - \mathbf{c}. \tag{14}$$

Let $\mathbf{d} = \boldsymbol{\nabla} J(\boldsymbol{\theta})$. Then,

$$\boldsymbol{\nabla} J(\boldsymbol{\theta} - \alpha \mathbf{d}) = (1-\alpha)\mathbf{d} - \alpha \boldsymbol{\Sigma} \mathbf{d}. \tag{15}$$

The Armijo condition $J(\boldsymbol{\theta} - \alpha \mathbf{d}) \geq J(\boldsymbol{\theta}) - \frac{\alpha}{2} \|\mathbf{d}\|_2^2$ becomes:

$$\alpha \boldsymbol{\theta}^\top \boldsymbol{\Sigma} \mathbf{d} - \frac{\alpha^2}{2} \mathbf{d}^\top \boldsymbol{\Sigma} \mathbf{d} - \alpha \langle \mathbf{c}, \mathbf{d}\rangle + \lambda \alpha \langle \boldsymbol{\theta}, \mathbf{d}\rangle \leq \frac{\alpha}{2} (\lambda \alpha + 1) \|\mathbf{d}\|_2^2. \tag{16}$$

The significance of (16) is as follows. In a typical iteration of BGD, we have to backtrack a few times (say t times) for each value of α. If we were to recompute $J(\boldsymbol{\theta} - \alpha \mathbf{d})$ using (13) each time, then the runtime of Armijo backtracking search is $O(tm^2)$, even after we have already computed $\mathbf{d}$ and $J(\boldsymbol{\theta})$. Now, using (16), we can compute in advance the following quantities (in this order): $\mathbf{d}$, $\|\boldsymbol{\theta}\|_2^2$, $\boldsymbol{\Sigma}\mathbf{d}$, $\langle \mathbf{c}, \mathbf{d}\rangle$, $\langle \boldsymbol{\theta}, \mathbf{d}\rangle$, $\mathbf{d}^\top \boldsymbol{\Sigma} \mathbf{d}$, $\boldsymbol{\theta}^\top \boldsymbol{\Sigma} \mathbf{d}$. Then, each check for inequality (16) can be done in $O(1)$-time, for a total of $O(m^2 + t)$-times. Once we have determined the step size α, (15) allows us to compute the next gradient (i.e., the next $\mathbf{d}$) in $O(m)$, because we have already computed $\boldsymbol{\Sigma}\mathbf{d}$ for line search.

To implement BGD, we need to compute four quantities efficiently: the covariance matrix $\boldsymbol{\Sigma}$ in (8), the correlation vector $\mathbf{c}$ in (9), point evaluation in (11), and the gradient in (12). The covariance matrix and the correlation vector only have to be computed once in a pre-processing step. The gradient is computed at every iteration, which includes several point evaluations as we perform line search.[2] We do not need to compute the second moment s_Y because optimizing $J(\boldsymbol{\theta})$ is the same as optimizing $J(\boldsymbol{\theta}) - s_Y$. Before describing how those four quantities can be computed efficiently, we discuss how we deal with categorical features.

3.2 Categorical features via sparse tensors

The more interesting, more common, and also considerably challenging situation is in the presence of categorical features. We next explain how we accommodate categorical features in the precomputation of $\boldsymbol{\Sigma}$ and $\mathbf{c}$.

Example 11. In Example 7, the matrix $\boldsymbol{\Sigma}$ is of size 8×8 instead of 3×3 after one-hot encoding. However, many of those entries are 0, for instance $(\forall (\mathbf{x}, y) \in D)$:

$$h_{1,\mathsf{vietnam}}(\mathbf{x}) h_{1,\mathsf{england}}(\mathbf{x}) = 0$$
$$h_{1,\mathsf{england}}(\mathbf{x}) h_{2,\mathsf{vietnam,blue}}(\mathbf{x}) = 0$$
$$h_{2,\mathsf{vietnam,blue}}(\mathbf{x}) h_{2,\mathsf{england,blue}}(\mathbf{x}) = 0$$
$$h_{2,\mathsf{vietnam,blue}}(\mathbf{x}) h_{2,\mathsf{vietnam,red}}(\mathbf{x}) = 0.$$

The reason is that the indicator variables x_{blue} and x_{england} act like selection clauses $x_{\mathsf{color}} = \mathsf{blue}$ and $x_{\mathsf{country}} = \mathsf{england}$. Thus, we can rewrite an entry σ_{ij} as an aggregate over a more selective query:

$$\sum_{(\mathbf{x},y)\in D} h_{1,\mathsf{vietnam}}(\mathbf{x}) h_{2,\mathsf{vietnam,red}}(\mathbf{x}) = \sum_{\phi} x_a^2 x_c x_b,$$

$\phi := ((\mathbf{x}, y) \in D \wedge x_{\mathsf{color}} = \mathsf{red} \wedge x_{\mathsf{country}} = \mathsf{vietnam})$.

Extrapolating straightforwardly, if we were to write $\boldsymbol{\Sigma}$ down in the one-hot encoded feature space, then the entries σ_{ij} under one-hot encoding got unrolled into many entries. More concretely, σ_{ij} is in fact a tensor $\boldsymbol{\sigma}_{ij}$ of dimension $\prod_{f\in C_i} |\pi_f(D)| \times \prod_{f\in C_j} |\pi_f(D)|$, because

$$\boldsymbol{\sigma}_{ij} = \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} h_i(\mathbf{x}) h_j(\mathbf{x})^\top. \tag{17}$$

Similarly, each component c_j of $\mathbf{c}$ defined in (9) is a tensor $\mathbf{c}_j$ of dimension $\prod_{f\in C_j} |\pi_f(D)|$, because $h_j(\mathbf{x})$ is a tensor in the categorical case. The following follows immediately.

Theorem 3.3. *Theorem 3.1 remains valid even when some features are categorical.*

Note that the outer product in (17) specifies the matrix layout of $\boldsymbol{\sigma}_{ij}$, and so $\boldsymbol{\Sigma}$ is a block matrix, each of whose blocks is $\boldsymbol{\sigma}_{ij}$. Furthermore, if we were to layout the tensor $\boldsymbol{\sigma}_{ij}$ as a vector, we can also write it as

$$\boldsymbol{\sigma}_{ij} = \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} h_i(\mathbf{x}) \otimes h_j(\mathbf{x}). \tag{18}$$

The previous example demonstrates that the dimensionalities of $\boldsymbol{\sigma}_{ij}$ and $\mathbf{c}_j$ can be very large. Fortunately, the tensors are very sparse, and a sparse representation of them can be computed with functional aggregate queries (in the FAQ-framework [6]) as shown in Proposition 3.4 below. We next illustrate the sparsity.

[2] In our implementation, each iteration typically involves 1-4 backtracking steps.

Example 12. Consider the query Q in Example 1, where the set of features is {sku, store, day, color, quarter, city, country} and *unitsSold* is the response variable. In this query $n = 7$, and thus for a PR_2 model we have $m = 1 + 7 + 7^2 = 57$ parameters. Consider two indices i and j to the component functions of g and h, where $i = (\mathsf{store}, \mathsf{city})$ and $j = (\mathsf{city})$. Suppose the query result states that the retailer has N_s stores in N_c countries. Then, the full dimensionality of the tensor $\boldsymbol{\sigma}_{ij}$ is $N_s \times N_c^2$, because by definition it was defined to be

$$\boldsymbol{\sigma}_{ij} := \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \underbrace{\mathbf{x}_{\mathsf{store}} \otimes \mathbf{x}_{\mathsf{city}}}_{h_i(\mathbf{x})} \otimes \underbrace{\mathbf{x}_{\mathsf{city}}}_{h_j(\mathbf{x})}. \tag{19}$$

Recall that $\mathbf{x}_{\mathsf{store}}$ and $\mathbf{x}_{\mathsf{city}}$ are both indicator vectors. The above tensor has the following straightforward interpretation: for every triple (s, c_1, c_2), where s is a store and c_1 and c_2 are cities, this triple entry of the tensor counts the number of data points $(\mathbf{x}, y) \in D$ for this particular combination of store and cities (divided by $1/|D|$). Most of these (s, c_1, c_2)-entries are 0. For example, if $c_1 \neq c_2$ then the count is zero. Thus, we can concentrate on computing entries of the form (s, c, c):

```
SELECT s, c, count(*) FROM D GROUP BY s, c;
```

Better yet, since store functionally determines city, the number of entries in the query output is bounded by N_s. Using relations to represent sparse tensor results in a massive amount of space saving.

3.3 Efficient precomputation of Σ and c

We employ two orthogonal ideas to compute the quantities (8) and (9) efficiently. First, our FAQ [6] and FDB [47] frameworks are designed to compute such aggregates over feature extraction queries, which are wider than traditional OLAP queries. Let $|\boldsymbol{\sigma}_{ij}|$ denote the *size* (i.e., number of tuples) of the sparse representation of the $\boldsymbol{\sigma}_{ij}$ tensor. Let $\mathsf{faqw}(i, j)$ denote the FAQ-*width* of the FAQ-query that expresses the aggregate $\boldsymbol{\sigma}_{ij}$ over the feature extraction query Q[3]; fhtw the fractional hypertree width of Q; and ρ^* the fractional edge cover number of Q.[4] Let I be the input database and $D = Q(I)$. Let N be the size of the largest input relation R in Q. Our precomputation time can be bounded as follows.

Proposition 3.4. *The tensors $\boldsymbol{\sigma}_{ij}$ and $\mathbf{c}_j$ can be sparsely represented by FAQ-queries with group-by variables $C_i \cup C_j$ and C_j, respectively. They can be computed in time*

$$O\left(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot \sum_{i,j\in[m]} (N^{\mathsf{faqw}(i,j)} + |\boldsymbol{\sigma}_{ij}|) \cdot \log N\right).$$

In case all features in D are continuous, $C_j = \emptyset$ for all $j \in [m]$, and $\mathsf{faqw}(i, j)$ is exactly the fractional hypertree width of Q [6]. Then, the overall runtime becomes $O(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot m^2 \cdot N^{\mathsf{fhtw}} \cdot \log N)$. When some features are categorical, we can also bound the width $\mathsf{faqw}(i, j)$ and tensor size, as stated by the next proposition.

Proposition 3.5. *Let $c = \max_{i,j} |C_i \cup C_j|$. Then, $\mathsf{faqw}(i, j) \leq \mathsf{fhtw} + c - 1$ and $|\boldsymbol{\sigma}_{ij}| \leq \min\{|D|, N^c\}$, $\forall i, j \in [m]$. For any query Q with $\rho^* > \mathsf{fhtw} + c - 1$, there are infinitely many database instances for which*

$$\lim_{N\to\infty} \frac{|D|}{\sum_{i,j\in[m]} (N^{\mathsf{faqw}(i,j)} + |\boldsymbol{\sigma}_{ij}|) \log N} = \infty. \tag{20}$$

Our precomputation step takes strictly sub-output-size runtime for infinitely many queries and database instances. If we were to compute $\boldsymbol{\sigma}_{ij}$ on a training dataset with categorical variables one-hot encoded, then the complexity would raise to $O(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot m^2 \cdot N^{\mathsf{fhtw}+2d} \log N)$, where d is the degree of the model.

Second, we exploit the observation that in the computation of $\boldsymbol{\Sigma}$ many *distinct* tensors $\boldsymbol{\sigma}_{ij}$ have *identical* sparse representations. For instance, the tensor $\boldsymbol{\sigma}_{ij}$ from Example 12 corresponding to $i = (\mathsf{store}, \mathsf{city})$ and $j = (\mathsf{city})$ has the same sparse representation as any of the following tensors: $(i, j) \in \{((\mathsf{city}, \mathsf{city}), \mathsf{store}), ((\mathsf{store}, \mathsf{store}), \mathsf{city}), ((\mathsf{store}, \mathsf{city}), \mathsf{store}), \ldots\}$. There are 12 tensors sharing this particular sparse representation. This is because store and city are categorical features and taking any power of the binary values in their indicator vectors does not change these values. Furthermore, any of the two features can be in i and/or j.

3.4 Point evaluation and gradient computation

To compute point evaluation and the gradient efficiently, we also introduce two ideas. First, we employ a sparse representation of tensors in the *parameter space*. We need to evaluate the component functions of g, which are polynomial. In the FaMa_r^2 example, for instance, we evaluate expressions of the form

$$g_{\mathsf{store},\ \mathsf{city}}(\boldsymbol{\theta}) = \sum_{\ell=1}^{r} \boldsymbol{\theta}_{\mathsf{store}}^{(\ell)} \otimes \boldsymbol{\theta}_{\mathsf{city}}^{(\ell)}. \tag{21}$$

The result is a 2-way tensor whose CP-decomposition (a sum of rank-1 tensors) is already given by (21)! There is no point in materializing the result of $g_{\mathsf{store},\ \mathsf{city}}(\boldsymbol{\theta})$ and we instead keep it as is. Assuming N_c distinct cities and N_s distinct stores in the training dataset D, if we were to materialize the tensor, then we would end up with an $\Omega(N_c N_s)$-sized result for absolutely no gain in computational and space complexity, while the space complexity of the CP-decomposition is only $O(N_c + N_s)$. This is a prime example of factorization of the parameter space.

Second, we explain how to evaluate (11) and (12) with our sparse tensor representations. There are two aspects of our solution worth spelling out: (1) how to multiply two tensors, e.g., $\boldsymbol{\sigma}_{ij}$ and $g_j(\boldsymbol{\theta})$, and (2) how to exploit that some tensors have the same representation to speed up the point evaluation and gradient computation.

To answer question (1), we need to know the intrinsic dimension of the tensor $\boldsymbol{\sigma}_{ij}$. In order to compute $\boldsymbol{\Sigma} g(\boldsymbol{\theta})$ in Example 12, we need to multiply $\boldsymbol{\sigma}_{ij}$ with

[3] We show in the proof of Proposition 3.4 how to express $\boldsymbol{\sigma}_{ij}$ and $\mathbf{c}_j$ as FAQ-queries.

[4] Due to space limitation, these width notions are defined in Appendix A.4.

$g_j(\boldsymbol{\theta})$ for $i = (\mathsf{store}, \mathsf{city})$ and $j = (\mathsf{city})$. In a linear model, $g_j(\boldsymbol{\theta}) = \boldsymbol{\theta}_j = \boldsymbol{\theta}_{\mathsf{city}}$. In this case, when computing $\boldsymbol{\sigma}_{ij}\boldsymbol{\theta}_{\mathsf{city}}$ we marginalize away one city dimension of the tensor, while keeping the other two dimensions store, city. This is captured by the following query:

```
SELECT store, city, sum(σ_{i,j}.val * θ_j.val)
FROM σ_{i,j}, θ_j WHERE σ_{i,j}.city = θ_j.city
GROUP BY store, city;
```

where the tensors $\boldsymbol{\sigma}_{i,j}$ and $\boldsymbol{\theta}_j$ map (store, city) and respectively (city) to aggregate values. In words, $\boldsymbol{\sigma}_{ij} g_j(\boldsymbol{\theta})$ is computed by a group-by aggregate query where the group-by variables are precisely the variables in C_i.

For the second question, we use the CP-decomposition of the parameter space as discussed earlier. Suppose now we are looking at the $\boldsymbol{\sigma}_{ij}$ tensor where $i = (\mathsf{city})$ and $j = (\mathsf{store}, \mathsf{city})$. Note that this tensor has the identical representation as the above tensor, but it is a *different* tensor. In a FaMa^2_r model, we would want to multiply this tensor with the component function $g_j(\boldsymbol{\theta})$ defined in (21) above. We do so by multiplying it with each of the terms $\boldsymbol{\theta}^{(\ell)}_{\mathsf{store}} \otimes \boldsymbol{\theta}^{(\ell)}_{\mathsf{city}}$, one by one for $\ell = 1, \ldots, r$, and then add up the result. Multiplying the tensor $\boldsymbol{\sigma}_{ij}$ with the first term $\boldsymbol{\theta}^{(1)}_{\mathsf{store}} \otimes \boldsymbol{\theta}^{(1)}_{\mathsf{city}}$ corresponds precisely to the following query:

```
SELECT city, sum(σ_{i,j}.val * θ^(1)_store.val * θ^(1)_city.val)
FROM σ_{i,j}, θ^(1)_store, θ^(1)_city
WHERE σ_{i,j}.city = θ^(1)_city.city AND
      σ_{i,j}.store = θ^(1)_store.store
GROUP BY city;
```

where the tensors $\boldsymbol{\sigma}_{i,j}$, $\boldsymbol{\theta}^{(1)}_{\mathsf{city}}$, and $\boldsymbol{\theta}^{(1)}_{\mathsf{store}}$ map (store, city), (city), and respectively (store) to aggregate values.

Finally, to answer question (2), note that for the same column j (i.e., the same component function $g_j(\boldsymbol{\theta})$), there can be multiple tensors $\boldsymbol{\sigma}_{ij}$ which have identical sparse representations. (This holds especially in models of degree > 1.) In such cases, we have queries with identical from-where blocks but different select-group-by clauses, because the tensors have different group-by variables. Nevertheless, all such queries can share computation as we can compute the from-where clause once for all of them and then scan this result to compute each specific tensor. This analysis gives rise to the following straightforward (and conservative) estimates.

For each $j \in [m]$, let d_j denote the degree and t_j denote the number of terms in the polynomial g_j (a component function of g). Recall that p is the number of parameters.

Proposition 3.6. *Point evaluation* (11) *and gradient computation* (12) *can be computed in time* $O(\sum_{i,j\in[m]} t_i t_j d_i d_j |\boldsymbol{\sigma}_{ij}|)$, *and respectively* $O(p \sum_{i,j\in[m]} t_i t_j d_i d_j |\boldsymbol{\sigma}_{ij}|)$.

For example, in a PR^d model, the point evaluation time is $O(d^2 \sum_{i,j\in[m]} |\boldsymbol{\sigma}_{ij}|)$, and gradient computation time becomes $O(n^d \sum_{i,j\in[m]} |\boldsymbol{\sigma}_{ij}|)$. In the FaMa^d_r model, the corresponding runtimes are $O(r^2 d^2 \sum_{i,j\in[m]} |\boldsymbol{\sigma}_{ij}|)$ and $O(nr^3 d^2 \sum_{i,j\in[m]} |\boldsymbol{\sigma}_{ij}|)$, respectively.

Overall, there are a couple of remarkable facts regarding the overall runtime of our approach. Without loss of generality, suppose the number of iterations of BGD is bounded. (This bound is typically dimension-free, dependent on the Lipschiz constant of J.) Then, from Proposition 3.5, there are infinitely many queries for which the overall runtime of BGD is unboundedly better than the output size. First, our approach is faster than even the data-export step of the "use an external tool to train model" approach. Second, it is often well-agreed upon that SGD is "faster" than BGD. However, a single iteration of SGD requires iterating through all data tuples, which takes time at least the output size. In particular, by training the model using BGD in the factorized form, BGD can be unboundedly faster than a single iteration of SGD.

4. FD-AWARE OPTIMIZATION

In this section, we show how to exploit functional dependencies among variables to reduce the dimensionality of the optimization problem by eliminating functionally determined variables and re-parameterizing the model. We compute the quantities $(\boldsymbol{\Sigma}, \mathbf{c})$ on the subset of features that are not functionally determined, and then solve the lower-dimensional optimization problem. Finally, we show how to recover the parameters in the original space in closed form. Exploiting functional dependencies drastically reduces the computation time for precomputation and each gradient iteration step.

4.1 Introduction to the main ideas

Consider a query Q with categorical variables country and city. For simplicity, assume that there are only two countries "vietnam" and "england", and 5 cities "saigon", "hanoi", "oxford", "leeds", and "bristol". Under one-hot encoding, the corresponding features are encoded as indicators x_{vietnam}, x_{england}, x_{saigon}, x_{hanoi}, x_{oxford}, x_{leeds}, x_{bristol}. Since city $\rightarrow$ country is an FD, for a given tuple $\mathbf{x}$ in the training dataset, the following hold:

$$x_{\mathsf{vietnam}} = x_{\mathsf{saigon}} + x_{\mathsf{hanoi}} \tag{22}$$

$$x_{\mathsf{england}} = x_{\mathsf{oxford}} + x_{\mathsf{leeds}} + x_{\mathsf{bristol}}. \tag{23}$$

The first identity states that if a tuple in the query result has "vietnam" as the value for country ($x_{\mathsf{vietnam}} = 1$), then its city's value can only be either "saigon" or "hanoi", i.e., $[x_{\mathsf{saigon}}, x_{\mathsf{hanoi}}]$ is either $[1, 0]$ or $[0, 1]$, respectively. The second identity is explained similarly.

How do we express the identities such as (22) and (23) in a formal manner in terms of the input vectors $\mathbf{x}_{\mathsf{city}}$ and $\mathbf{x}_{\mathsf{country}}$? We can extract in a preprocessing step from the database a relation of the form $R(\mathsf{city}, \mathsf{country})$ with city as primary key. Let N_{city} and N_{country} be the number of cities and countries, respectively. The predicate $R(\mathsf{city}, \mathsf{country})$ is the sparse representation of a matrix $\mathbf{R}$ of size $N_{\mathsf{country}} \times N_{\mathsf{city}}$, such that if $\mathbf{x}_{\mathsf{city}}$ is an indicator vector for saigon, then $\mathbf{R}\mathbf{x}_{\mathsf{city}}$ is an indicator

for vietnam. In this language, the above identities are written as $\mathbf{x}_{\text{country}} = \mathbf{R}\mathbf{x}_{\text{city}}$. For example, in the above particular example $N_{\text{city}} = 5$, $N_{\text{country}} = 2$, and

$$\mathbf{R} = \begin{array}{ccccc} \text{saigon} & \text{hanoi} & \text{oxford} & \text{leeds} & \text{bristol} \\ \hline 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array} \begin{array}{l} \\ \text{vietnam} \\ \text{england} \end{array}$$

This relationship suggests a natural idea: replace any occurrence of statistics $\mathbf{x}_{\text{country}}$ by its functionally determining quantity $\mathbf{x}_{\text{city}}$. Since these quantities are present only in the loss function $\mathcal{L}$ via inner products $\langle g(\mathbf{x}), h(\boldsymbol{\theta})\rangle$, such replacements result in a (typically) linear reparameterization of the loss. What happens next is less obvious, due to the presence of nonlinear penalty function Ω. Depending on the specific structure of FDs and the choice of Ω, it turns out that many parameters associated with redundant statistics, which do not affect the loss $\mathcal{L}$, can be optimized out directly with respect to the transformed Ω penalty.

The remainder of this subsection is a gentle introduction of our idea in the presence of *one* simple FD in the LR model. Consider a query Q in which city and country are two of the categorical features and functionally determine one another via a matrix $\mathbf{R}$ such that $\mathbf{R}\mathbf{x}_{\text{city}} = \mathbf{x}_{\text{country}}$ for all $\mathbf{x} = (\cdots, \mathbf{x}_{\text{city}}, \mathbf{x}_{\text{country}}, \cdots) \in D$. We exploit this fact to "eliminate" $\mathbf{x}_{\text{country}}$ as follows.

$$\begin{aligned}
&\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle = \langle \boldsymbol{\theta}, \mathbf{x}\rangle \\
&= \sum_{j \notin \{\text{city},\text{country}\}} \langle \boldsymbol{\theta}_j, \mathbf{x}_j\rangle + \langle \boldsymbol{\theta}_{\text{city}}, \mathbf{x}_{\text{city}}\rangle + \langle \boldsymbol{\theta}_{\text{country}}, \mathbf{x}_{\text{country}}\rangle \\
&= \sum_{j \notin \{\text{city},\text{country}\}} \langle \boldsymbol{\theta}_j, \mathbf{x}_j\rangle + \langle \boldsymbol{\theta}_{\text{city}}, \mathbf{x}_{\text{city}}\rangle + \langle \boldsymbol{\theta}_{\text{country}}, \mathbf{R}\mathbf{x}_{\text{city}}\rangle \\
&= \sum_{j \notin \{\text{city},\text{country}\}} \langle \boldsymbol{\theta}_j, \mathbf{x}_j\rangle + \left\langle \underbrace{\boldsymbol{\theta}_{\text{city}} + \mathbf{R}^\top \boldsymbol{\theta}_{\text{country}}}_{\boldsymbol{\gamma}_{\text{city}}}, \mathbf{x}_{\text{city}} \right\rangle .
\end{aligned}$$

Reparameterize the model by defining $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_j)_{j \in V - \{\text{country}\}}$, and two functions $\overline{g} : \mathbb{R}^{n-1} \to \mathbb{R}^{n-1}$, $\overline{h} : \mathbb{R}^n \to \mathbb{R}^{n-1}$:

$$\boldsymbol{\gamma}_j = \begin{cases} \boldsymbol{\theta}_j & j \neq \text{city} \\ \boldsymbol{\theta}_{\text{city}} + \mathbf{R}^\top \boldsymbol{\theta}_{\text{country}} & j = \text{city}. \end{cases} \tag{24}$$

$$\overline{g}(\boldsymbol{\gamma}) = \boldsymbol{\gamma} \tag{25}$$

$$\overline{h}_j(\mathbf{x}) = \mathbf{x}_j, \ j \neq \text{city}. \tag{26}$$

(There is no $\boldsymbol{\gamma}_{\text{country}}$.) Reparameterize $J(\boldsymbol{\theta})$ by

$$\begin{aligned}
J(\boldsymbol{\theta}) &= \frac{1}{2|D|} \sum_{(\mathbf{x},y)\in D} (\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle - y)^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 \\
&= \frac{1}{2|D|} \sum_{(\mathbf{x},y)\in D} (\langle \overline{g}(\boldsymbol{\gamma}), \overline{h}(\mathbf{x})\rangle - y)^2 + \\
&\frac{\lambda}{2} \left(\sum_{j \neq \text{city}} \|\boldsymbol{\gamma}_j\|_2^2 + \|\boldsymbol{\gamma}_{\text{city}} - \mathbf{R}^\top \boldsymbol{\theta}_{\text{country}}\|_2^2 + \|\boldsymbol{\theta}_{\text{country}}\|_2^2 \right).
\end{aligned}$$

Note how $\boldsymbol{\theta}_{\text{country}}$ has disappeared from the loss term, but it still remains in the penalty term. We now "optimize out" $\boldsymbol{\theta}_{\text{country}}$ by observing that

$$\frac{1}{\lambda} \frac{\partial J}{\partial \boldsymbol{\theta}_{\text{country}}} = \mathbf{R}(\mathbf{R}^\top \boldsymbol{\theta}_{\text{country}} - \boldsymbol{\gamma}_{\text{city}}) + \boldsymbol{\theta}_{\text{country}} \tag{27}$$

By setting (27) to 0 we obtain $\boldsymbol{\theta}_{\text{country}}$ in terms of $\boldsymbol{\gamma}_{\text{city}}$: $\boldsymbol{\theta}_{\text{country}} = (\mathbf{I}_{\text{country}} + \mathbf{R}\mathbf{R}^\top)^{-1}\mathbf{R}\boldsymbol{\gamma}_{\text{city}}$ and equivalently as $\mathbf{R}(\mathbf{I}_{\text{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\text{city}}$, where $\mathbf{I}_{\text{country}}$ is the order-N_{country} identity matrix and similarly for $\mathbf{I}_{\text{city}}$. (See Appendix C.1.) J can thus be expressed completely in terms of $\boldsymbol{\gamma}$, and its gradient with respect to $\boldsymbol{\gamma}$ is also available:

$$\begin{aligned}
J(\boldsymbol{\theta}) &= \frac{1}{2|D|} \sum_{(\mathbf{x},y)\in D} (\langle \overline{g}(\boldsymbol{\gamma}), \overline{h}(\mathbf{x})\rangle - y)^2 + \\
&\frac{\lambda}{2} \left(\sum_{j \neq \text{city}} \|\boldsymbol{\gamma}_j\|_2^2 + \left\langle (\mathbf{I}_{\text{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\text{city}}, \boldsymbol{\gamma}_{\text{city}} \right\rangle \right),
\end{aligned}$$

$$\frac{1}{2}\frac{\partial \|\boldsymbol{\theta}\|_2^2}{\partial \boldsymbol{\gamma}_j} = \begin{cases} \boldsymbol{\gamma}_j & j \neq \text{city} \\ \left(\mathbf{I}_{\text{city}} + \mathbf{R}^\top\mathbf{R}\right)^{-1} \boldsymbol{\gamma}_{\text{city}} & j = \text{city}. \end{cases}$$

The gradient of the loss term is computed using the matrix $\overline{\boldsymbol{\Sigma}}$ and the vector $\overline{\mathbf{c}}$ with respect to the pair $(\overline{g}, \overline{h})$ of reduced dimensionality. The matrix $(\mathbf{I}_{\text{city}} + \mathbf{R}^\top\mathbf{R})$ is a rank-N_{country} update to the identity matrix $\mathbf{I}_{\text{city}}$, strictly positive definite and thus invertible. The inverse can be obtained using database aggregate queries; for numerical stability, one may compute its Cholesky decomposition which can *also* be computed using aggregate queries. These "linear algebra via aggregate queries" computations are possible because our matrices admit a database interpretation. (See Section 4.5 below.)

4.2 Functional dependencies (FDs)

Composite FDs lead to more complex identities. For instance, the FD (guest, hotel, date) $\to$ room leads to the identity $x_{\text{room}} = \sum x_{\text{guest}} x_{\text{hotel}} x_{\text{date}}$. Let R be a relation on attributes guest, hotel, date, and room, encoding this dependency, i.e., R has a compound key (guest, hotel, date). Then, corresponding to R there is a matrix $\mathbf{R}$ of dimension $N_{\text{room}} \times N_{\text{guest}} \cdot N_{\text{hotel}} \cdot N_{\text{date}}$ for which $\mathbf{x}_{\text{room}} = \mathbf{R}(\mathbf{x}_{\text{guest}} \otimes \mathbf{x}_{\text{hotel}} \otimes \mathbf{x}_{\text{date}})$. Our results can be extended to the case of composite FDs, yet with a great notational burden; for the sake of clarity, we only state the results on simple FDs (FDs whose left-hand side is one variable).

Definition 1. To formalize general FD structures, consider a query Q in which there are k disjoint groups $G_1, \ldots, G_k$ of features, among other features. The ith group is $G_i = \{f_i\} \cup S_i$, where f_i is a feature, S_i a set of features, and $f_i \to S_i$ is an FD. We shall refer to these as *groups of simple FDs*. (See also Figure 1.)

Example 13. In a typical feature extraction query for LogicBlox's retailer customers, we have $k = 3$ groups (in addition to other features): the first group contains week $\to$ month $\to$ quarter $\to$ year, and thus $f_1 =$ week and $S_1 =$ { month, quarter, year }. In the second group, $f_2 =$ sku and $S_2 =$ { type, color, size, ...} (a rather large group). In the third group $f_3 =$ store and $S_3 =$ { city, country, region, continent }.

For each feature $c \in S_i$, let $\mathbf{R}_c$ denote the matrix for which $\mathbf{x}_c = \mathbf{R}_c \mathbf{x}_{f_i}$. For the sake of brevity, we also define a matrix $\mathbf{R}_{f_i} = \mathbf{I}_{f_i}$ (the identity matrix of dimension equal to the active domain size of attribute f_i); this way the equality $\mathbf{R}_c \mathbf{x}_{f_i} = \mathbf{x}_c$ holds for every $c \in G_i$.

The linear relationship holds even if the variables are not categorical. For example, consider the FD $\mathsf{sku} \to \mathsf{price}$ (assuming every stock-keeping unit has a fixed sale-price). The relationship is modeled with a $1 \times N_{\mathsf{sku}}$ matrix $\mathbf{R}$, where the entry corresponding to a sku is its price. Then, $\mathbf{R}\mathbf{x}_{\mathsf{sku}} = x_{\mathsf{price}}$ for any indicator vector $\mathbf{x}_{\mathsf{sku}}$.

Definition 2 (FD-reduced pairs of functions)**.** Given a pair of functions g and h in our problem setting. Recall that C_j's are defined in Section 2.3, while S_k's are given in Definition 1. Define

$$K := \{j \in [m] \mid C_j \cap (S_1 \cup \cdots \cup S_k) \neq \emptyset\}$$

(In words, K is the set of component functions of h containing at least one functionally determined variable.)

The group of simple FDs induces an *FD-reduced pair* of functions $\overline{g} : \mathbb{R}^{p-|K|} \to \mathbb{R}^{m-|K|}$ and $\overline{h} : \mathbb{R}^n \to \mathbb{R}^{m-|K|}$, which are specified as follows: The component functions of $\overline{h}$ are obtained from the component functions of h by removing all component functions h_j for $j \in K$. Similarly, $\overline{g}$ is obtained from g by removing all component functions g_j for which $j \in K$. Naturally, define the covariance matrix $\overline{\mathbf{\Sigma}}$ and the correlation vector $\overline{\mathbf{c}}$ as in (8) and (9), but with respect to $\overline{h}$.

In the next two sections, we generalize the above technique to speedup the training of PR^d and FaMa under an arbitrary collection of simple FDs.

4.3 Polynomial regression under FDs

Recall the PR^d-model formulated in Example 3. Consider the set A_V of all tuples $\mathbf{a}_V = (a_w)_{w \in V} \in \mathbb{N}^V$ of non-negative integers such that $\|\mathbf{a}_V\|_1 \leq d$. For any $(\mathbf{x}, y) \in D$ and $\mathbf{a} \in A_V$, define $\mathbf{x}^{\otimes \mathbf{a}} := \bigotimes_{v \in V} \mathbf{x}_v^{\otimes a_v}$. In the PR^d model we have $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\mathbf{a}})_{\|\mathbf{a}\|_1 \leq d}$, $g(\boldsymbol{\theta}) = \boldsymbol{\theta}$, and $h_{\mathbf{a}}(\mathbf{x}) = \mathbf{x}^{\otimes \mathbf{a}}$. If a feature, say $v \in V$, is non-categorical, then $\mathbf{x}_v^{\otimes a_v} = x_v^{a_v}$. If we knew $x_v \in \{0, 1\}$, then $x_v^{a_v} = x_v$ and thus there is no need to have terms for which $a_v > 1$. A similar situation occurs when v is a categorical variable. To see this, let us consider a simple query where $V = \{b, c, w, t\}$, and all four variables are categorical. Suppose the PR^d model has a term corresponding to $\mathbf{a} = (a_b, a_c, a_w, a_t) = (0, 2, 0, 1)$. The term of $\langle \boldsymbol{\theta}, h(\mathbf{x}) \rangle$ indexed by tuple $\mathbf{a}$ is of the form

$$\left\langle \boldsymbol{\theta}_{\mathbf{a}}, \mathbf{x}_c^{\otimes 2} \otimes \mathbf{x}_t \right\rangle = \left\langle \boldsymbol{\theta}_{\mathbf{a}}, \mathbf{x}_c \otimes \mathbf{x}_c \otimes \mathbf{x}_t \right\rangle.$$

For the dimensionality to match up, $\boldsymbol{\theta}_{\mathbf{a}}$ is a 3rd-order tensor, say indexed by (i, j, k). The above expression can be simplified as

$$\sum_i \sum_j \sum_k \boldsymbol{\theta}_{\mathbf{a}}(i, j, k) \cdot \mathbf{x}_c(i) \cdot \mathbf{x}_c(j) \cdot \mathbf{x}_t(k) = \sum_j \sum_k \boldsymbol{\theta}_{\mathbf{a}}(j, j, k) \mathbf{x}_c(j) \mathbf{x}_t(k),$$

where the equality holds due to the fact that $\mathbf{x}_c(j)$ is idempotent. In particular, we only need the entries indexed by (j, j, k) of $\boldsymbol{\theta}_{\mathbf{a}}$. Equivalently, we write:

$$\langle \boldsymbol{\theta}_{\mathbf{a}}, \mathbf{x}_c \otimes \mathbf{x}_c \otimes \mathbf{x}_t \rangle = \left\langle ((\mathbf{I}_c \star \mathbf{I}_c)^\top \otimes \mathbf{I}_t) \boldsymbol{\theta}_{\mathbf{a}}, \mathbf{x}_c \otimes \mathbf{x}_t \right\rangle.$$

Multiplying on the left by the matrix $(\mathbf{I}_c \star \mathbf{I}_c)^\top \otimes \mathbf{I}_t$ has precisely the same effect as selecting out only entries $\boldsymbol{\theta}_{\mathbf{a}}(j, j, k)$ from the tensor $\boldsymbol{\theta}_{\mathbf{a}}$. More generally, in the PR^d model we can assume that all the indices $\mathbf{a}_V = (a_v)_{v \in V}$ satisfy the condition that $a_v \in \{0, 1\}$ whenever v is categorical. (This is in addition to the degree requirement that $\|\mathbf{a}_V\|_1 \leq d$.)

Given k groups of FDs represented by $G_1, \ldots, G_k$, let $G = \bigcup_{i=1}^k G_i$, $S = \bigcup_{i=1}^k S_i$, $\overline{G} = V - G$, $\overline{S} = V - S$, and $F = \{f_1, \ldots, f_k\}$. For every non-empty subset $T \subseteq [k]$, define $F_T := \{f_i \mid i \in T\}$. Given a natural number $q < d$, and a non-empty set $T \subseteq [k]$ with size $|T| \leq d - q$, define the collection

$$\mathcal{U}(T, q) := \{U \mid U \subseteq G \quad \wedge U \cap G_i \neq \emptyset, \forall i \in T \quad \wedge \quad U \cap G_i = \emptyset, \forall i \notin T \quad \wedge \quad |U| \leq d - q\}. \tag{28}$$

For every tuple $\mathbf{a}_{\overline{G}} \in \mathbb{N}^{\overline{G}}$ with $\|\mathbf{a}_{\overline{G}}\|_1 = q < d$, $i \in T$, and every $U \in \mathcal{U}(T, q)$, define the following matrices, which play the same role as $\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top \mathbf{R}$ in Section 4.1:

$$\mathbf{B}_{T,q,i} = \sum_{U \in \mathcal{U}(T,q)} \left(\left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right]^\top \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \right), \tag{29}$$

$$\mathbf{R}_{\mathbf{a}_{\overline{G}}, U} = \bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \bigstar_{c \in U \cap G_i} \mathbf{R}_c. \tag{30}$$

The following theorem reparameterizes $J(\boldsymbol{\theta})$ for PR^d ($d \geq 1$) to become $\overline{J}(\boldsymbol{\gamma})$. While $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\mathbf{a}})$ is a vector indexed by tuples $\mathbf{a} = \mathbf{a}_V \in \mathbb{N}^V$, the new parameters $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_{\mathbf{b}})$ are indexed by integer tuples $\mathbf{b} = \mathbf{b}_{\overline{S}} \in \mathbb{N}^{\overline{S}}$.

Theorem 4.1. *Consider the PR^d-model with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\mathbf{a}_V})_{\|\mathbf{a}_V\|_1 \leq d}$, and k groups of simple FDs $G_i = \{f_i\} \cup S_i$, $i \in [k]$. Define the reparameterization:*

$$\boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}} = \begin{cases} \boldsymbol{\theta}_{(\mathbf{b}_{\overline{G}}, \mathbf{0}_G)} & \mathbf{b}_F = \mathbf{0}_F \\ \displaystyle\sum_{U \in \mathcal{U}(T,q)} \mathbf{R}_{\mathbf{b}_{\overline{G}}, U}^\top \boldsymbol{\theta}_{(\mathbf{b}_{\overline{G}}, \mathbf{1}_{U|G})} & \substack{T = \{j \mid j \in F, b_{f_j} = 1\}, \\ q = \|\mathbf{b}_{\overline{G}}\|_1} \end{cases}.$$

Then, minimizing $J(\boldsymbol{\theta})$ is equivalent to minimizing the function

$$\overline{J}(\boldsymbol{\gamma}) = \frac{1}{2} \boldsymbol{\gamma}^\top \overline{\mathbf{\Sigma}} \boldsymbol{\gamma} - \langle \boldsymbol{\gamma}, \overline{c} \rangle + \frac{\lambda}{2} \Omega(\boldsymbol{\gamma}), \tag{31}$$

where

$$\Omega(\boldsymbol{\gamma}) = \sum_{\substack{\|\mathbf{b}_{\overline{S}}\|_1 \leq d \\ \|\mathbf{b}_F\|_1 = 0}} \left\| \boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}} \right\|_2^2 + \sum_{\substack{\|\mathbf{b}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{0 < |T| \leq d - q} \left\langle \left(\bigotimes_{\substack{w \in \overline{G} \\ b_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \mathbf{B}_{T,q,i}^{-1} \right) \boldsymbol{\gamma}_{(\mathbf{b}_{\overline{G}}, \mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{b}_{\overline{G}}, \mathbf{1}_{F_T|F})} \right\rangle$$

The proof of this theorem (Appendix C.2) is technically involved. $\overline{J}$ is defined above with respect to the FD-reduced pair of functions $\overline{g}, \overline{h}$ and a reduced parameter space of $\boldsymbol{\gamma}$. Its gradient is simple to compute, since

$$\frac{1}{2}\frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}}} = \begin{cases} \boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}} & \text{when } \mathbf{b}_F = \mathbf{0}_F, \\ \left(\bigotimes_{\substack{w \in \overline{G} \\ b_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \mathbf{B}_{T,q,i}^{-1} \right) \boldsymbol{\gamma}_{(\mathbf{b}_{\overline{G}}, \mathbf{1}_{F_T \mid F})} & \\ \text{when } T = \{j \mid {}^{j \in F,}_{b_j = 1}\}, q = \|\mathbf{b}_{\overline{G}}\|_1 . \end{cases} \tag{32}$$

Moreover, once a minimizer $\boldsymbol{\gamma}$ of $\overline{J}$ is obtained, we can compute a minimizer $\boldsymbol{\theta}$ of J by setting

$$\boldsymbol{\theta}_{\mathbf{a}_V} = \begin{cases} \boldsymbol{\gamma}_{\mathbf{a}_{\overline{S}}}, & \text{when } \|\mathbf{a}_G\|_1 = 0 \\ \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \mathbf{B}_{T,q,i}^{-1} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T \mid F})}, & \\ \quad \text{when} \quad \|\mathbf{a}_G\|_1 > 0, \|\mathbf{a}_{\overline{G}}\|_1 = q, & \\ \quad T = \{i \mid \exists c \in G_i, a_c > 0\} & \\ \quad \text{and } U = \{c \mid a_c > 0, c \in G\}. & \end{cases} \tag{33}$$

Theorem 4.1 might be a bit difficult to grasp at first glance due to its generality. To give the reader a sense of how the theorem is applied in specific instances, sections C.4 and C.5 present two specializations of the theorem for (ridge) linear regression (PR^1), and degree-2 polynomial regression (PR^2).

4.4 Factorization machines under FDs

We now turn our attention to FaMa_r^2.

Theorem 4.2. *Consider the* FaMa *model of degree 2, rank r, parameters, $\boldsymbol{\theta} = (\boldsymbol{\theta}_i, (\boldsymbol{\theta}_i^{(\ell)})_{\ell \in [r]})_{i \in V}$ and k groups of simple FDs $G_i = \{f_i\} \cup S_i$, $i \in [k]$. Let $G = \cup_{i \in [k]} G_i$,*

$$\boldsymbol{\beta}_{f_i} := \sum_{\ell=1}^{r} \sum_{\{c,t\} \in \binom{G_i}{2}} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \circ \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)}, \ i \in [k] \tag{34}$$

and the following reparameterization:

$$\boldsymbol{\gamma}_w = \begin{cases} \boldsymbol{\theta}_w & w \notin \bigcup_{i=1}^k G_i \\ \boldsymbol{\theta}_{f_i} + \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c + \boldsymbol{\beta}_{f_i} & w = f_i, i \in [k]. \end{cases}$$

$$\boldsymbol{\gamma}_w^{(\ell)} = \begin{cases} \boldsymbol{\theta}_w^{(\ell)} & w \notin F \\ \boldsymbol{\theta}_{f_i}^{(\ell)} + \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} & w = f_i, i \in [k]. \end{cases}$$

Then, minimizing $J(\boldsymbol{\theta})$ is equivalent to minimizing the function $\overline{J}(\boldsymbol{\gamma}) := \frac{1}{2}\overline{g}(\boldsymbol{\gamma})^\top \overline{\boldsymbol{\Sigma}} \overline{g}(\boldsymbol{\gamma}) - \langle \overline{g}(\boldsymbol{\gamma}), \overline{\mathbf{c}} \rangle + \frac{\lambda}{2}\Omega(\boldsymbol{\gamma})$, where

$$\Omega(\boldsymbol{\gamma}) := \sum_{w \notin G} \|\boldsymbol{\gamma}_w\|_2^2 + \sum_{i=1}^{k} \left\langle \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), (\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) \right\rangle + \sum_{\substack{\ell \in [r] \\ w \notin F}} \left\| \boldsymbol{\gamma}_w^{(\ell)} \right\|_2^2 + \sum_{\substack{i \in [k] \\ \ell \in [r]}} \left\| \boldsymbol{\gamma}_{f_i}^{(\ell)} - \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\gamma}_c^{(\ell)} \right\|_2^2 . \tag{35}$$

In order to optimize $\overline{J}$ with respect to $\boldsymbol{\gamma}$, the following proposition provides a closed form formulae for the relevant gradient.

Proposition 4.3. *The gradient of $\Omega(\boldsymbol{\gamma})$ defined in (35) can be computed by computing $\boldsymbol{\delta}_i^{(\ell)} = \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\gamma}_c^{(\ell)}$, and*

$$\boldsymbol{\beta}_{f_i} = \sum_{\ell=1}^{r} \left[\left(\boldsymbol{\gamma}_{f_i}^{(\ell)} - \frac{1}{2}\boldsymbol{\delta}_i^{(\ell)} \right) \circ \boldsymbol{\delta}_i^{(\ell)} - \frac{1}{2} \sum_{t \in S_i} \mathbf{R}_t^\top (\boldsymbol{\gamma}_t^{(\ell)} \circ \boldsymbol{\gamma}_t^{(\ell)}) \right]$$

Then,

$$\frac{1}{2}\frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_w} = \begin{cases} \boldsymbol{\gamma}_w, & w \notin G \\ \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) & w = f_i, i \in [k]. \end{cases} \tag{36}$$

$$\frac{1}{2}\frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_w^{(\ell)}} = \begin{cases} \boldsymbol{\gamma}_w^{(\ell)}, \ \textit{for } w \notin G, \ell \in [r] \\ \boldsymbol{\gamma}_{f_i}^{(\ell)} - \boldsymbol{\delta}_i^{(\ell)} - \frac{1}{2}\boldsymbol{\delta}_i^{(\ell)} \circ \frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_{f_i}}, \ \textit{for } {}^{w=f_i,}_{\ell \in [r]} \\ \boldsymbol{\gamma}_w^{(\ell)} - \mathbf{R}_w \left[\boldsymbol{\gamma}_{f_i}^{(\ell)} \circ \frac{1}{2}\frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_{f_i}} + \frac{1}{2}\frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_{f_i}^{(\ell)}} \right], \\ \textit{when } w \in S_i, \ell \in [r]. \end{cases} \tag{37}$$

Suppose that the minimizer $\boldsymbol{\gamma}$ of $\overline{J}$ has been obtained, then a minimizer $\boldsymbol{\theta}$ of J is available in closed form:

$$\boldsymbol{\theta}_w = \begin{cases} \boldsymbol{\gamma}_w & w \in V \setminus G \\ \mathbf{R}_t \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), & \forall t \in G_i, i \in [k]. \end{cases}$$

$$\boldsymbol{\theta}_w^{(\ell)} = \begin{cases} \boldsymbol{\gamma}_w^{(\ell)}, & \forall w \notin F, \ell \in [r]. \\ \boldsymbol{\gamma}_w^{(\ell)} - \delta_i^{(\ell)}, & w = f_i, \ell \in [r]. \end{cases}$$

This section shows that our technique applies to a non-linear model too. It should be obvious that a similar reparameterization works for FaMa_r^d for any $d \geq 1$. There is some asymmetry in the reparameterization of 1st-order parameters $\boldsymbol{\theta}_i$ and 2nd-order parameters $\boldsymbol{\theta}_i^{(\ell)}$ in Theorem 4.2, because we can solve a system of linear equation with matrix inverses, but we don't have closed form solutions for quadratic equations.

4.5 Linear algebra with database queries

To apply the above results, we need to solve a couple of computational primitives. The first primitive is to compute the matrix inverse $\mathbf{B}_{T,q}^{-1}$ (or rather, the product of the inverse with another vector). This task can be done in one of two ways: we either explicitly compute the inverse, or compute the Cholesky decomposition of the matrix $\mathbf{B}_{T,q}$. We next explain how both of these tasks can be done using database queries.

Maintaining the matrix inverse with rank-1 updates. Using Sherman-Morrison-Woodbury formula ((47) in Appendix), we can incrementally compute the inverse of the matrix $\mathbf{I} + \sum_{c \in G_i} \mathbf{R}_c^\top \mathbf{R}_c$ as follows. Let $S \subset G_i$ be some subset and suppose we have already computed the inverse for $\mathbf{M}_S = \mathbf{I} + \sum_{s \in S} \mathbf{R}_s^\top \mathbf{R}_s$. We now explain how we can compute the inverse for $\mathbf{M}_{S \cup \{c\}} = \mathbf{I} + \sum_{s \in S \cup \{c\}} \mathbf{R}_s^\top \mathbf{R}_s$. For concreteness, let's say $\mathbf{R}_c$

maps city to country. For each country country, let $\mathbf{e}_{\mathsf{country}}$ denote the 01-vector where there is a 1 for each city the country has. For example, $\mathbf{e}_{\mathsf{cuba}} = [1\ 1\ 0\ 0\ 0]^\top$. Then, $\mathbf{R}_c^\top \mathbf{R}_c = \sum_{\mathsf{country}} \mathbf{e}_{\mathsf{country}} \mathbf{e}_{\mathsf{country}}^\top$. And thus, starting with $\mathbf{M}_S$, we apply the Sherman-Morrison-Woodbury formula for each country, such as:

$$(\mathbf{M}+\mathbf{e}_{\mathsf{cuba}}\mathbf{e}_{\mathsf{cuba}}^\top)^{-1} = \mathbf{M}^{-1} - \frac{\mathbf{M}^{-1}\mathbf{e}_{\mathsf{cuba}}\mathbf{e}_{\mathsf{cuba}}^\top\mathbf{M}^{-1}}{1+\mathbf{e}_{\mathsf{cuba}}^\top\mathbf{M}^{-1}\mathbf{e}_{\mathsf{cuba}}}. \quad (38)$$

This update can be done with database aggregate queries, because $\mathbf{e}_{\mathsf{cuba}}^\top\mathbf{M}^{-1}\mathbf{e}_{\mathsf{cuba}}$ is a sum of entries (i,j) in $\mathbf{M}^{-1}$ where both i and j are cities in cuba; $\mathbf{v} = \mathbf{M}^{-1}\mathbf{e}_{\mathsf{cuba}}$ is the sum of columns of $\mathbf{M}^{-1}$ corresponding to cuba; and $\mathbf{M}^{-1}\mathbf{e}_{\mathsf{cuba}}\mathbf{e}_{\mathsf{cuba}}^\top\mathbf{M}^{-1}$ is exactly $\mathbf{v}\mathbf{v}^\top$.

Overall, each update (38) can be done in $O(N_{\mathsf{city}}^2)$-time, for an overall runtime of $O(N_{\mathsf{city}}^2 N_{\mathsf{country}})$. This runtime should be contrasted with Gaussian-elimination-based inverse computation time, which is $O(N_{\mathsf{city}}^3)$. When the FDs form a chain, the blocks are nested inside one another, and thus each update is even cheaper as we do not have to access all N_{city}^2 entries.

Maintaining a Cholesky decomposition with rank-k update. Maintaining a matrix inverse can be numerically unstable. It would be best to compute a Cholesky decomposition of the matrix, since this strategy is numerically more stable. There are known rank-1 update algorithms [16, 21], using strategies similar to the inverse rank-1 update above. A further common computational primitive is to multiply a tensor product with a vector, such as in $(\mathbf{B}_i^{-1} \otimes \mathbf{B}_j^{-1})\boldsymbol{\gamma}_{f_i f_j}$ (also expressible as aggregate queries, cf. Appendix A.3).

4.6 Discussion

The prevalence of FDs presents fresh new challenges from both computational and statistical viewpoints. On the one hand, a reasonable and well-worn rule of thumb in statistics dictates that one should always eliminate features that are functionally dependent on others, because this helps reduce both computation and model's complexity, which in turn leads to reduced generalization error (as also noted in [34]). On the other hand, the statistical effectiveness of such a rule is difficult to gauge when the nature of dependence goes beyond linearity. In such scenarios, it might be desirable to keep some redundant variables, but only if they help construct simpler forms of regression/classification functions, leading to improved approximation ability for the model class.

It is, however, difficult to know a priori which redundant features lead to simple functions. Therefore, the problem of dimensionality reduction cannot be divorced from the model class under consideration. While this remains unsolved in general, in this work we restricted ourselves to specific classes of learning models, the complexity of which may still be varied through regularization via (non-linear) penalties. Within a regularized parametric model class, we introduced dimensionality reduction techniques (variable elimination and reparameterization) that may not fundamentally change the model's capacity. The reduction in the number of parameters may still help reduce the variance of parameter estimates, leading to improved generalization error guarantees. We pursued in this section a more tangible benefit that lies in improved computational efficiency.

There is substantial runtime saving brought by the reparameterizations from Theorems 4.1 and 4.2. In the setting without exploiting FDs, the runtime of our approach consists of the precomputation time stated in Proposition 3.4, and the per-iteration time stated in Theorem 3.6; this time might be multiplied by the number of backtracking steps depending on the specific iteration. By exploiting the FDs, the quantities in Propositions 3.4 and 3.6 are computed with respect to the smaller dimensional matrix $\overline{\boldsymbol{\Sigma}}$. Let $q = |V - \bigcup_{i\in[k]} S_i|$, then the dimensionality of $\overline{\boldsymbol{\Sigma}}$ for PR^d is $\Theta(q^d) \times \Theta(q^d)$, much smaller than that of $\boldsymbol{\Sigma}$, which is $\Theta(n^d) \times \Theta(n^d)$. (See Example 13.) Last but not least, Proposition C.1 in the appendix provides the corresponding version of Corollary 3.2 with respect to $\overline{\boldsymbol{\Sigma}}$.

5. OPEN PROBLEMS

Our in-database learning framework raises open questions on statistics, algorithm design, and optimization. We next sketch a few representatives.

One research direction is to extend the class of statistical models to train efficiently inside the database beyond those considered in this paper. Our formulation (1) captures a common class of regression models (such as PR and FaMa) and classification models (such as logistic and SVM), which is done by changing the loss function $\mathcal{L}$. It remains open how to extend our formulation to capture latent variable models.

The aggregates defining $\boldsymbol{\Sigma}$, $\mathbf{c}$, point evaluation, and gradient computation are "multi-output" queries. They deserve a systematic investigation, from formulation to evaluation and complexity analysis. In practice, one often reserves a fragment of the training data for model validation. It is an interesting question to incorporate this data partitioning requirement into our framework.

Understanding how to adapt further optimization algorithms, such as coordinate descent or stochastic gradient, to our in-database framework is an important research direction. Furthermore, our FD-aware optimization is specific to the ℓ_2-norm in the penalty term. We would also like to understand the effect of other norms, e.g., ℓ_1, on model reparameterization under FDs.

Finally, we conjecture that the cost function may be easier to optimize with respect to the reduced set of parameters that are not functionally determined: As redundant variables are eliminated or optimized out, the cost function's Hessian with respect to reduced parameters becomes less ill-conditioned, resulting in faster convergence behavior for gradient-based optimization techniques. The impact of FD-based dimensionality reduction, from both computational and statistical standpoints, have not been extensively studied for learning (nonlinear) models with categorical variables, which are precisely the kind discussed in our framework.

6. REFERENCES

[1] M. Abadi and et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[2] S. Abiteboul and et al. Research directions for principles of data management (Dagstuhl Perspectives Workshop 16151). *CoRR*, abs/1701.09007, 2017.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[4] M. Abo Khamis, H. Q. Ngo, C. Ré, and A. Rudra. Joins via geometric resolutions: Worst-case and beyond. In *PODS*, pages 213–228, 2015.

[5] M. Abo Khamis, H. Q. Ngo, and A. Rudra. FAQ: questions asked frequently. *CoRR*, abs/1504.04044, 2015.

[6] M. Abo Khamis, H. Q. Ngo, and A. Rudra. FAQ: Questions asked frequently. In *PODS*, pages 13–28, 2016.

[7] I. Adler. *Width functions for hypertree decompositions*. 2006. Ph.D. Dissertation, Albert-Ludwigs-Universität Freiburg. 2006.

[8] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Advances in knowledge discovery and data mining. chapter Fast Discovery of Association Rules, pages 307–328. 1996.

[9] M. Aref, B. ten Cate, T. J. Green, B. Kimelfeld, D. Olteanu, E. Pasalic, T. L. Veldhuizen, and G. Washburn. Design and implementation of the LogicBlox system. In *SIGMOD*, pages 1371–1382, 2015.

[10] A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. In *FOCS*, pages 739–748, 2008.

[11] N. Bakibayev, T. Kociský, D. Olteanu, and J. Závodný. Aggregation and ordering in factorised databases. *PVLDB*, 6(14):1990–2001, 2013.

[12] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA J. Numer. Anal.*, 8(1):141–148, 1988.

[13] M. Boehm, S. Tatikonda, B. Reinwald, P. Sen, Y. Tian, D. Burdick, and S. Vaithyanathan. Hybrid parallelization strategies for large-scale machine learning in SystemML. *PVLDB*, 7(7):553–564, 2014.

[14] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade (2nd ed)*, pages 421–436, 2012.

[15] J.-F. Boulicaut and C. Masson. *Data Mining Query Languages*, pages 715–726. 2005.

[16] T. A. Davis and W. W. Hager. Multiple-rank modifications of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 22(4):997–1013, 2001.

[17] T. Elgamal, S. Luo, M. Boehm, A. V. Evfimievski, S. Tatikonda, B. Reinwald, and P. Sen. SPOOF: sum-product optimization and operator fusion for large-scale machine learning. In *CIDR*, 2017.

[18] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.

[19] X. Feng, A. Kumar, B. Recht, and C. Ré. Towards a unified architecture for in-rdbms analytics. In *SIGMOD*, pages 325–336, 2012.

[20] R. Fletcher. On the Barzilai-Borwein method. In *Optimization and control with applications*, volume 96 of *Appl. Optim.*, pages 235–256. 2005.

[21] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974.

[22] T. Goldstein, C. Studer, and R. G. Baraniuk. A field guide to forward-backward splitting with a FASTA implementation. *CoRR*, abs/1411.3406, 2014.

[23] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. In *PODS*, pages 21–32, 1999.

[24] M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *SODA*, pages 289–298, 2006.

[25] W. W. Hager. Updating the inverse of a matrix. *SIAM Rev.*, 31(2):221–239, 1989.

[26] D. Harris and S. Harris. *Digital Design and Computer Architecture*. 2nd edition, 2012.

[27] T. Hastie, R. Tibshrani, and M. J. Wainwright. *Statistical Learning with Sparsity: The Lasso and generalizations*. CRC Press, 2015.

[28] J. M. Hellerstein and et al. The MADlib analytics library or MAD skills, the SQL. *PVLDB*, 5(12):1700–1711, 2012.

[29] B. Huang, M. Boehm, Y. Tian, B. Reinwald, S. Tatikonda, and F. R. Reiss. Resource elasticity for large-scale machine learning. In *SIGMOD*, pages 137–152, 2015.

[30] C. G. Khatri and C. R. Rao. Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhy S̄er. A*, 30:167–180, 1968.

[31] B. Kimelfeld and C. Ré. A relational framework for classifier engineering. In *PODS*, pages 5–20, 2017.

[32] A. Kumar, M. Boehm, and J. Yang. Data management in machine learning: Challenges, techniques, and systems. In *SIGMOD*, pages 1717–1722, 2017.

[33] A. Kumar, J. F. Naughton, and J. M. Patel. Learning generalized linear models over normalized data. In *SIGMOD*, pages 1969–1984, 2015.

[34] A. Kumar, J. F. Naughton, J. M. Patel, and X. Zhu. To join or not to join?: Thinking twice about joins before feature selection. In *SIGMOD*, pages 19–34, 2016.

[35] X. Meng and et al. Mllib: Machine learning in apache spark. *J. Mach. Learn. Res.*, 17(1):1235–1241, 2016.

[36] D. Neumann. Lightning-fast deep learning on Spark via parallel stochastic gradient updates, www.deepdist.com, 2015.

[37] H. Ngo, X. Nguyen, D. Olteanu, and M. Schleich. In-Database Learning with Sparse Tensors. *CoRR*, https://arxiv.org/pdf/1703.04780v1, 2017.

[38] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. Worst-case optimal join algorithms. In *PODS*, pages 37–48, 2012.

[39] H. Q. Ngo, C. Ré, and A. Rudra. Skew strikes back: New developments in the theory of join algorithms. In *SIGMOD Rec.*, pages 5–16, 2013.

[40] D. Olteanu and J. Závodný. Size bounds for factorised representations of query results. *TODS*, 40(1):2, 2015.

[41] J. Pei, J. Han, and L. V. Lakshmanan. Mining frequent itemsets with convertible constraints. In *ICDE*, pages 433–442, 2001.

[42] K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. Version 20121115.

[43] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich. Data management challenges in production machine learning. In *SIGMOD*, pages 1723–1726, 2017.

[44] C. Qin and F. Rusu. Speculative approximations for terascale distributed gradient descent optimization. In *DanaC*, pages 1:1–1:10, 2015.

[45] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, 2012.

[46] S. Rendle. Scaling factorization machines to relational data. *PVLDB*, 6(5):337–348, 2013.

[47] M. Schleich, D. Olteanu, and R. Ciucanu. Learning linear regression models over factorized joins. In *SIGMOD*, pages 3–18, 2016.

[48] T. L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In *ICDT*, pages 96–106, 2014.

[49] M. Zaharia and et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 15–28, 2012.

APPENDIX

A. MATRIX CALCULUS

A.1 Basics

We list here common identities we often use in the paper; for more details see the Matrix Cookbook [42]. We use *denominator layout* for differentiation, i.e., the gradient is a column vector. Let $\mathbf{A}$ be a matrix, and $\mathbf{u}, \mathbf{v}, \mathbf{x}, \mathbf{b}$ be vectors, where $\mathbf{A}$ and $\mathbf{b}$ are independent of $\mathbf{x}$, and $\mathbf{u}$ and $\mathbf{v}$ are functions of $\mathbf{x}$ then

$$\frac{\partial \langle \mathbf{b}, \mathbf{x} \rangle}{\partial \mathbf{x}} = \mathbf{b} \tag{39}$$

$$\frac{\partial \mathbf{x}^\top \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x} \tag{40}$$

$$\frac{\partial \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2}{\partial \mathbf{x}} = 2\mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{41}$$

$$\frac{\partial \mathbf{u}^\top \mathbf{v}}{\partial \mathbf{x}} = \frac{\partial \mathbf{u}^\top}{\partial \mathbf{x}}\mathbf{v} + \frac{\partial \mathbf{v}^\top}{\partial \mathbf{x}}\mathbf{u} \tag{42}$$

$$\frac{\partial (\mathbf{B}\mathbf{x} + \mathbf{b})^\top \mathbf{C}(\mathbf{D}\mathbf{x} + \mathbf{d})}{\partial \mathbf{x}} = \mathbf{B}^\top \mathbf{C}(\mathbf{D}\mathbf{x} + \mathbf{d}) + \mathbf{D}^\top \mathbf{C}^\top (\mathbf{B}\mathbf{x} + \mathbf{b}). \tag{43}$$

The following matrix inversion formulas will be useful. See [25] and references thereof for details.

Proposition A.1. *We have*

$$(\mathbf{B} + \mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}\mathbf{B}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{B}^{-1}. \tag{44}$$

whenever all dimensions match up and inverses on the right hand side exist. In particular, the following holds when $\mathbf{C} = (1)$, $\mathbf{U} = \mathbf{1}$, $\mathbf{V} = \mathbf{1}^\top$, *and* $\mathbf{J}$ *is the all-*1 *matrix:*

$$(\mathbf{B} + \mathbf{J})^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{1}(1 + \mathbf{1}^\top \mathbf{B}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top \mathbf{B}^{-1}. \tag{45}$$

Another special case is

$$(\mathbf{A} + \mathbf{U}^\top \mathbf{U})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}^\top(\mathbf{I} + \mathbf{U}\mathbf{A}^{-1}\mathbf{U}^\top)^{-1}\mathbf{U}\mathbf{A}^{-1}. \tag{46}$$

An even more special case is the Sherman-Morrison formula, where $\mathbf{U}^\top$ *is just a vector* $\mathbf{u}$. *The matrix* $\mathbf{A} + \mathbf{u}\mathbf{u}^\top$ *is typically called a rank-*1 *update of* $\mathbf{A}$*:*

$$(\mathbf{A} + \mathbf{u}\mathbf{u}^\top)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{u}^\top \mathbf{A}^{-1}}{1 + \mathbf{u}^\top \mathbf{A}^{-1}\mathbf{u}}. \tag{47}$$

A.2 The Product Cookbook: Tensor product, Kronecker product, and Khatri-Rao product

Next, we discuss some identities regarding tensors. We use $\otimes$ to denote the *tensor product*. When taking tensor product of two matrices, this is called the *Kronecker* product, which is *not* the same as the outer product for matrices, even though the two are isomorphic maps. If $\mathbf{A} = (a_{ij})$ is an $m \times n$ matrix and $\mathbf{B} = (b_{k\ell})$ is a $p \times q$ matrix, then the tensor product $\mathbf{A} \otimes \mathbf{B}$ is an $mp \times nq$ matrix whose $((i,k),(j,\ell))$ entry is $a_{ij}b_{k\ell}$. In particular, if $\mathbf{x} = (x_i)_{i\in[m]}$ is an m-dimensional vector and $\mathbf{y} = (y_j)_{j\in[p]}$ is an p-dimensional vector, then $\mathbf{x} \otimes \mathbf{y}$ is an mp-dimensional vector whose (i,j) entry is $x_i y_j$; this is *not* an $m \times p$ matrix as in the case of the outer product. This layout is the correct layout from the definition of the tensor (Kronecker) product. If $\mathbf{A}$ is matrix, then $\mathbf{A}^{\otimes k}$ denote the tensor power $\underbrace{\mathbf{A} \otimes \cdots \otimes \mathbf{A}}_{k \text{ times}}$. More generally,

Definition 3 (Tensor product)**.** Let $\mathbf{A}$ be a tensor of order r (i.e., a function $\psi_A(X_1, \ldots, X_r)$) and $\mathbf{B}$ be a tensor of order s (i.e., a function $\psi_B(Y_1, \ldots, Y_s)$), then the tensor product $\mathbf{A} \otimes \mathbf{B}$ is the multilinear function

$$\psi(X_1, \ldots, X_r, Y_1, \ldots, Y_s) = \psi_A(X_1, \ldots, X_r)\psi_B(Y_1, \ldots, Y_s).$$

(A matrix is a tensor of order 2.)

Definition 4 (Khatri-Rao product)**.** Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices each with n columns. We use $\mathbf{A} \star \mathbf{B}$ to denote the matrix with n columns, where the jth column of $\mathbf{A} \star \mathbf{B}$ is the tensor product of the jth column of $\mathbf{A}$ with the jth columns of $\mathbf{B}$. The operator $\star$ is a (special case of) the *Khatri-Rao* product [30], where we partition the input matrices into blocks of one column each. More elaborately, if $\mathbf{A}$ has columns $\mathbf{a}_1, \ldots, \mathbf{a}_n$, and $\mathbf{B}$ has columns

$\mathbf{b}_1, \ldots, \mathbf{b}_n$,then one can visualize the $\star$ operator as follows:

$$\mathbf{A} \star \mathbf{B} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & \cdots & | \end{bmatrix} \star \begin{bmatrix} | & | & \cdots & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ | & | & \cdots & | \end{bmatrix} = \begin{bmatrix} | & | & \cdots & | \\ | & | & \cdots & | \\ \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_n \otimes \mathbf{b}_n \\ | & | & \cdots & | \\ | & | & \cdots & | \end{bmatrix}.$$

(Note $\mathbf{A}$ and $\mathbf{B}$ do not need to have the same number of rows.)

Definition 5 (Hadamard product)**.** Let $\mathbf{A} = (a_{ij})$ and $\mathbf{B} = (b_{ij})$ be two $m \times n$ matrices, then the Hadamard product $\mathbf{A} \circ \mathbf{B}$ is an $m \times n$ matrix, where each i, j element is given by $(\mathbf{A} \circ \mathbf{B})_{ij} = a_{ij} b_{ij}$.

Proposition A.2. *We have (if the dimensionalities match up correctly):*

$$(\mathbf{AB} \otimes \mathbf{CD}) = (\mathbf{A} \otimes \mathbf{C})(\mathbf{B} \otimes \mathbf{D}) \tag{48}$$

$$(\mathbf{A} \otimes \mathbf{B})^\top = (\mathbf{A}^\top \otimes \mathbf{B}^\top) \tag{49}$$

$$\langle \mathbf{x}, \mathbf{By} \rangle = \langle \mathbf{B}^\top \mathbf{x}, \mathbf{y} \rangle \tag{50}$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = (\mathbf{A}^{-1} \otimes \mathbf{B}^{-1}) \qquad \textit{if both are square matrices} \tag{51}$$

$$\langle \mathbf{A} \otimes \mathbf{B}, \mathbf{RX} \otimes \mathbf{SY} \rangle = \langle \mathbf{R}^\top \mathbf{A} \otimes \mathbf{S}^\top \mathbf{B}, \mathbf{X} \otimes \mathbf{Y} \rangle. \tag{52}$$

If $\mathbf{x}$ *is a standard* n*-dimensional unit vector,* $\mathbf{A}$ *and* $\mathbf{B}$ *are two matrices with* n *columns each, and* $\mathbf{a}$ *and* $\mathbf{b}$ *are two* n*-dimensional vectors, then*

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{x} \otimes \mathbf{x}) = (\mathbf{A} \star \mathbf{B})\mathbf{x} \tag{53}$$

$$\langle \mathbf{a} \otimes \mathbf{b}, \mathbf{x} \otimes \mathbf{x} \rangle = \langle \mathbf{a} \circ \mathbf{b}, \mathbf{x} \rangle. \tag{54}$$

Let $\mathbf{x}$ *be a standard* n*-dimensional unit vector,* $\mathbf{A}_1, \ldots, \mathbf{A}_k$ *be* k *matrices with* n *columns each. Then,*

$$(\bigotimes_{i=1}^{k} \mathbf{A}_i)(\mathbf{x}^{\otimes k}) = (\bigstar_{i=1}^{k} \mathbf{A}_i)\mathbf{x}. \tag{55}$$

We note in passing that the first five identities are very useful in our dimension reduction techniques by exploiting functional dependencies, while (53)(54)(55) are instrumental in achieving computational reduction in our handling of categorical features.

Proof. The identities (48), (49), (50), and (51) can be found in the Matrix Cookbook [42]. Identity (52) follows from (48) and (49). To see (53), note that

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{x} \otimes \mathbf{x}) = \mathbf{Ax} \otimes \mathbf{Bx} = (\mathbf{A} \star \mathbf{B})\mathbf{x},$$

where the last equality follows due to the following reasoning. Suppose $x_j = 1$ for some j, then $\mathbf{Ax} = \mathbf{a}_j$ and $\mathbf{Bx} = \mathbf{b}_j$, where $\mathbf{a}_j$ and $\mathbf{b}_j$ are the jth columns of $\mathbf{A}$ and $\mathbf{B}$, respectively. Thus,

$$\mathbf{Ax} \otimes \mathbf{Bx} = \mathbf{a}_j \otimes \mathbf{b}_j = (\mathbf{A} \star \mathbf{B})_j = (\mathbf{A} \star \mathbf{B})\mathbf{x}.$$

Identities (54) and (55) are proved similarly, where (55) is a trivial generalization of (53). □

A.3 Tensor computation, FAQ-expression, and the InsideOut algorithm

Quite often we need to compute a product of the form $(\mathbf{A} \otimes \mathbf{B})\mathbf{C}$, where $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ are tensors, provided that their dimensionalities match up. For example, suppose $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{B}$ a $p \times q$ matrix, and $\mathbf{C}$ a $nq \times 1$ matrix (i.e. a vector). The result is a $mp \times 1$ tensor. The brute-force way of computing $(\mathbf{A} \otimes \mathbf{B})\mathbf{C}$ is to compute $\mathbf{A} \otimes \mathbf{B}$ first, taking $\Theta(mnpq)$-time, and then multiply the result with $\mathbf{C}$, for an overall runtime of $\Theta(mnpq)$. The brute-force algorithm is a horribly inefficient algorithm.

The better way to compute $(\mathbf{A} \otimes \mathbf{B})\mathbf{C}$ is to view this as an FAQ-expression [6] (a sum-product form): we think of $\mathbf{A}$ as a function $\psi_A(x, y)$, $\mathbf{B}$ as a function $\psi_B(z, t)$, and $\mathbf{C}$ as a function $\psi_C(y, t)$. What we want to compute is the function

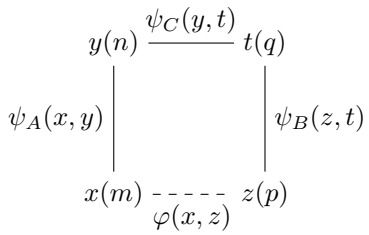
$$\varphi(x, z) = \sum_y \sum_t \psi_A(x, y) \psi_B(z, t) \psi_C(y, t). \tag{56}$$

This is nothing but a 4-cycle FAQ-query, and we should certainly pick between the following two strategies:

- eliminate t first (i.e. compute $\varphi_1(y, z) := \sum_t \psi_B(z, t)\psi_C(y, t)$ with a runtime of $O(npq)$), and then eliminated y (i.e. compute $\varphi(x, y) = \sum_y \varphi_1(y, z)\psi_A(x, y)$ with $O(mnp)$-time. The overall runtime is thus $O(np(m + q))$.

- or the symmetric strategy of eliminating y first, and then t for an overall runtime of $O(mq(n+p))$.

This is not surprising, since the problem is just matrix chain multiplication; or, in the FAQ-language of the InsideOut algorithm we want to pick the best tree decomposition and then compute a variable elimination order out of it [6]. We shall see later that a special case of the above that occurs often is when $\mathbf{B} = \mathbf{I}$, the identity matrix. In that case,



$\psi_B(z,t)$ is the same as the atom $z = t$, and thus it serves as a change of variables:

$$\varphi(x,z) = \sum_y \sum_t \psi_A(x,y)\psi_B(z,t)\psi_C(y,t) = \sum_y \psi_A(x,y)\psi_C(y,z).$$

In other words, we only have to marginalize out one variable instead of two. This situation arises, for example, in Eq. (32) and Eq. (33).

A.4 FAQ-width

A.4.1 Background: Fractional edge cover number and output size bounds

In what follows, we consider a conjunctive query Q over a relational database instance I. We use N to denote the size of the largest input relation in Q. We also use $Q(I)$ to denote the output and $|Q(I)|$ to denote its size. We use the query Q and its hypergraph $\mathcal{H}$ interchangeably.

Definition 6 (Fractional edge cover number ρ^*)**.** Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph (of some query Q). Let $B \subseteq \mathcal{V}$ be any subset of vertices. A *fractional edge cover* of B using edges in $\mathcal{H}$ is a feasible solution $\boldsymbol{\lambda} = (\lambda_S)_{S\in\mathcal{E}}$ to the following linear program:

$$\begin{aligned} \min \quad & \sum_{S\in\mathcal{E}} \lambda_S \\ \text{s.t.} \quad & \sum_{S: v\in S} \lambda_S \geq 1, \quad \forall v \in B \\ & \lambda_S \geq 0, \quad \forall S \in \mathcal{E}. \end{aligned}$$

The optimal objective value of the above linear program is called the *fractional edge cover number* of B in $\mathcal{H}$ and is denoted by $\rho^*_{\mathcal{H}}(B)$. When $\mathcal{H}$ is clear from the context, we drop the subscript $\mathcal{H}$ and use $\rho^*(B)$.

Given a conjunctive query Q, the fractional edge cover number of Q is $\rho^*_{\mathcal{H}}(\mathcal{V})$ where $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is the hypergraph of Q.

Theorem A.3 (AGM-bound [10, 24])**.** *Given a full conjunctive query Q over a relational database instance I, the output size is bounded by*

$$|Q(I)| \leq N^{\rho^*},$$

where ρ^* is the fractional edge cover number of Q.

Theorem A.4 (AGM-bound is tight [10, 24])**.** *Given a full conjunctive query Q and a non-negative number N, there exists a database instance I whose relation sizes are upper-bounded by N and satisfies*

$$|Q(I)| = \Theta(N^{\rho^*}).$$

Worst-case optimal join algorithms [4, 38, 39, 48] can be used to answer any full conjunctive query Q in time

$$O(|\mathcal{V}| \cdot |\mathcal{E}| \cdot N^{\rho^*} \cdot \log N). \tag{57}$$

A.4.2 Background: Tree decompositions, acyclicity, and width parameters

Definition 7 (Tree decomposition)**.** Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph. A *tree decomposition* of $\mathcal{H}$ is a pair (T, χ) where $T = (V(T), E(T))$ is a tree and $\chi : V(T) \to 2^{\mathcal{V}}$ assigns to each node of the tree T a subset of vertices of $\mathcal{H}$. The sets $\chi(t)$, $t \in V(T)$, are called the *bags* of the tree decomposition. There are two properties the bags must satisfy

(a) For any hyperedge $F \in \mathcal{E}$, there is a bag $\chi(t)$, $t \in V(T)$, such that $F \subseteq \chi(t)$.

(b) For any vertex $v \in \mathcal{V}$, the set $\{t \mid t \in V(T), v \in \chi(t)\}$ is not empty and forms a connected subtree of T.

Definition 8 (acyclicity)**.** A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is *acyclic* iff there exists a tree decomposition (T, χ) in which every bag $\chi(t)$ is a hyperedge of $\mathcal{H}$.

When $\mathcal{H}$ represents a join query, the tree T in the above definition is also called the *join tree* of the query. A query is acyclic if and only if its hypergraph is acyclic.

For non-acyclic queries, we often need a measure of how "close" a query is to being acyclic. To that end, we use *width* notions of a query.

Definition 9 (g-width of a hypergraph: a generic width notion [7])**.** Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph, and $g : 2^{\mathcal{V}} \to \mathbb{R}^+$ be a function that assigns a non-negative real number to each subset of $\mathcal{V}$. The *g-width* of a tree decomposition (T, χ) of $\mathcal{H}$ is $\max_{t \in V(T)} g(\chi(t))$. The *$g$-width of $\mathcal{H}$* is the *minimum* g-width over all tree decompositions of $\mathcal{H}$. (Note that the g-width of a hypergraph is a *Minimax* function.)

Definition 10 (*Treewidth* and *fractional hypertree width* are special cases of *g-width*)**.** Let s be the following function: $s(B) = |B| - 1$, $\forall V \subseteq \mathcal{V}$. Then the *treewidth* of a hypergraph $\mathcal{H}$, denoted by $\mathsf{tw}(\mathcal{H})$, is exactly its s-width, and the *fractional hypertree width* of a hypergraph $\mathcal{H}$, denoted by $\mathsf{fhtw}(\mathcal{H})$, is the ρ^*-width of $\mathcal{H}$.

From the above definitions, $\mathsf{fhtw}(\mathcal{H}) \geq 1$ for any hypergraph $\mathcal{H}$. Moreover, $\mathsf{fhtw}(\mathcal{H}) = 1$ if and only if $\mathcal{H}$ is acyclic.

A.4.3 Background: Vertex/variable orderings and their equivalence to tree decompositions

Besides tree decompositions, there is another way to define acyclicity and width notions of a hypergraph, which is *orderings* of the hypergraph vertices. And just like we refer to queries and hypergraphs interchangeably, we also refer to query variables and hypergraph vertices interchangeably.

In what follows, we use n to denote the number of vertices of the given hypergraph $\mathcal{H}$.

Definition 11 (Vertex ordering of a hypergraph)**.** A *vertex ordering* of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is simply a listing $\sigma = (v_1, \ldots, v_n)$ of all vertices in $\mathcal{V}$.

Definition 12 (Elimination sets U_j^σ of a vertex ordering σ)**.** Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ and a vertex ordering $\sigma = (v_1, \ldots, v_n)$, we define sets $U_1^\sigma, \ldots, U_n^\sigma \subseteq \mathcal{V}$, called the *elimination sets of* σ, as follows: Let $\partial(v_n)$ be the set of hyperedges of $\mathcal{H}$ that contain v_n. We define U_n^σ to be the union of all hyperedges in $\partial(v_n)$:

$$U_n^\sigma := \bigcup_{S \in \partial(v_n)} S.$$

If $n = 1$, then we are done. Otherwise, we remove vertex v_n and all hyperedges in $\partial(v_n)$ from $\mathcal{H}$ and add back to $\mathcal{H}$ a new hyperedge $U_n^\sigma - \{v_n\}$, thus turning $\mathcal{H}$ into a hypergraph with $n - 1$ vertices:

$$\begin{aligned} \mathcal{V} &\leftarrow \mathcal{V} - \{v_n\}, \\ \mathcal{E} &\leftarrow (\mathcal{E} - \partial(v_n)) \cup \big\{U_n^\sigma - \{v_n\}\big\}. \end{aligned}$$

The remaining elimination sets $U_1^\sigma, \ldots, U_{n-1}^\sigma$ are defined inductively to be the elimination sets of the resulting hypergraph (whose vertices are now $\{v_1, \ldots, v_{n-1}\}$).

When σ is clear from the context, we drop the superscript σ and use $U_1, \ldots, U_n$.

Proposition A.5 (Every vertex ordering has an "equivalent" tree decomposition [5])**.** *Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, for every vertex ordering σ, there is a tree decomposition (T, χ) whose bags $\chi(t)$ are the elimination sets U_j^σ of σ.*

By applying the GYO elimination procedure [3] on the bags of any given tree decomposition, we can obtain an "equivalent" vertex ordering:

Proposition A.6 (Every tree decomposition has an "equivalent" vertex ordering [5])**.** *Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, for every tree decomposition (T, χ), there is a vertex ordering σ such that every elimination set U_j^σ of σ is contained in some bag $\chi(t)$ of the tree decomposition (T, χ).*

A.4.4 FAQ-*width of an* FAQ *query*

Just like a conjunctive query, an FAQ query has a query hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. But unlike conjunctive queries, an FAQ query also specifies an order of its variables, which is the order in which we aggregate over those variables in the given FAQ-expression. (For example, in expression (56), we sum over t first, then over y, and we keep z and x as free variables. Hence, the FAQ query in (56) specifies the variable order $\sigma = (x, z, y, t)$.) Such a variable order for the query can also be interpreted as a vertex order σ for the query's hypergraph.

As we have seen earlier in Section A.3, the InsideOut algorithm for answering FAQ queries is based on *variable elimination*. To eliminate variable/vertex v_n, we have to solve a sub-problem consisting of a smaller FAQ query over the variables in the elimination set U_n^σ. This smaller query can be solved by an algorithm that is based on worst-case optimal join algorithms [4, 38, 39, 48]. From (57), this takes time [5]

$$O(|\mathcal{V}| \cdot |\mathcal{E}| \cdot N^{\rho^*_{\mathcal{H}}(U_n^\sigma)} \cdot \log N). \tag{58}$$

After eliminating v_n, the remaining variables $v_{n-1}, v_{n-2}, \ldots, v_1$ can be eliminated similarly. This variable elimination algorithm motivates the following width notion.

Definition 13 (FAQ-width of a given variable ordering σ)**.** Given an FAQ query φ with a variable ordering σ, we define the FAQ-width of σ, denoted by $\mathsf{faqw}(\sigma)$, to be

$$\mathsf{faqw}(\sigma) := \max_{j \in [n]} \left\{ \rho^*_{\mathcal{H}}(U_j^\sigma) \right\}. \tag{59}$$

By the above definition, the FAQ-width of a variable ordering σ is the same as the fractional hypertree width of the "equivalent" tree decomposition that is referred to in Proposition A.5.

Theorem A.7 (Runtime of *InsideOut* [6])**.** *Given an* FAQ*-query φ with a variable ordering σ, the InsideOut algorithm answers φ in time*

$$O\left(|\mathcal{V}|^2 \cdot |\mathcal{E}| \cdot \left(N^{\mathsf{faqw}(\sigma)} + |\varphi|\right) \cdot \log N\right), \tag{60}$$

where $|\varphi|$ is the output size in the listing representation.

Let φ be an FAQ query with variable ordering σ. In many cases, there might be a different variable ordering σ' such that if we were to permute the aggregates of φ in the order of σ' instead of σ, we would obtain an FAQ-query φ' that is "semantically-equivalent" to φ (i.e. that always returns the same answer as φ no matter what the input is). If this is the case, then we can run InsideOut on φ using the ordering σ' instead of σ, which can lead to a better runtime if $\mathsf{faqw}(\sigma')$ happens to be smaller than $\mathsf{faqw}(\sigma)$. We use $\mathsf{EVO}(\varphi)$ to denote the set of all such "equivalent" orderings σ'. (For a formal definition, see [6].) Therefore, it is best to consider all orderings σ' in $\mathsf{EVO}(\varphi)$, pick the one with the smallest $\mathsf{faqw}(\sigma')$, and use it in InsideOut algorithm. This motivates the following definition.

Definition 14 (FAQ-width of an FAQ query)**.** The FAQ-width of an FAQ query φ, denoted by $\mathsf{faqw}(\varphi)$, is the minimum one over all orderings σ' in $\mathsf{EVO}(\varphi)$, i.e.

$$\mathsf{faqw}(\varphi) := \min \left\{ \mathsf{faqw}(\sigma') \mid \sigma' \in \mathsf{EVO}(\varphi) \right\}. \tag{61}$$

Characterizing $\mathsf{EVO}(\varphi)$ for an arbitrary given FAQ-query φ is a technically involved problem (see [6] for hardness background and a general solution). However, the FAQ queries that we need for our machine learning tasks are of a special form that makes the problem easier. In particular, as we saw earlier in Section A.3, there is only type of aggregate operators that we use in such queries (which is the summation operator $\sum$). We refer to those special FAQ queries as FAQ-SS queries (see [6]). Our FAQ-SS queries in this work have only two types of variables:

- Variables that we are summing over, e.g. variables y and t in (56).
- Free variables (i.e. Group-by variables), e.g. variables x and z.

Given an FAQ-SS query φ, $\mathsf{EVO}(\varphi)$ contains every ordering σ' that lists all free variables *before* the non-free variables. For example, for the FAQ-SS query $\varphi(x, z)$ in (56), $\mathsf{EVO}(\varphi(x, z))$ contains all permutations of $\{x, y, z, t\}$ where $\{x, z\}$ come before $\{y, t\}$.

Proposition A.8. *For any* FAQ-SS *query φ without free variables, we have $\mathsf{faqw}(\varphi) = \mathsf{fhtw}(\mathcal{H})$, where $\mathcal{H}$ is the hypergraph of $\mathcal{H}$.*

Proof. In this case, $\mathsf{EVO}(\varphi)$ contains all $n!$ possible orderings. By Proposition A.6, for every tree decomposition (T, χ), there is an ordering σ' such that $\mathsf{faqw}(\sigma') \leq \mathsf{fhtw}((T, \chi))$. By Proposition A.5, for every ordering σ', there is a tree decomposition (T, χ) such that $\mathsf{fhtw}((T, \chi)) = \mathsf{faqw}(\sigma')$. Therefore, we have

$$\min_{\sigma' \in \mathsf{EVO}(\varphi)} \mathsf{faqw}(\sigma') \quad = \quad \min_{(T,\chi)} \mathsf{fhtw}((T, \chi)).$$

□

[5] To achieve this runtime, we need some additional ideas that are beyond the scope of this very brief introduction to FAQ. See [6] for more details.

Proposition A.9. *For any* FAQ-SS *query* φ *with* $f \geq 1$ *free variables, we have* $\mathsf{faqw}(\varphi) \leq \mathsf{fhtw}(\mathcal{H}) + f - 1$, *where* $\mathcal{H}$ *is the hypergraph of* $\mathcal{H}$.

Proof. Find a tree decomposition (T, χ) of $\mathcal{H}$ with minimal fhtw, i.e. where $\mathsf{fhtw}((T,\chi)) = \mathsf{fhtw}(\mathcal{H})$. WLOG let the f free variables be $v_1, \ldots, v_f$. Construct another tree decomposition $(T, \overline{\chi})$ by extending all bags $\chi(t)$ of (T,χ) with the variables $\{v_2, \ldots, v_f\}$, i.e. by defining $\overline{\chi}(t) := \chi(t) \cup \{v_2, \ldots, v_f\}$ for all t. By Definition 7, $(T, \overline{\chi})$ *is* indeed a tree decomposition. And because $\rho^*(\chi(t) \cup \{v_2, \ldots, v_f\}) \leq \rho^*(\chi(t)) + f - 1$, we have

$$\mathsf{fhtw}((T,\overline{\chi})) \leq \mathsf{fhtw}((T,\chi)) + f - 1.$$

Moreover, since (T,χ) must have a bag $\chi(t^*)$ that contains v_1, the corresponding bag $\overline{\chi}(t^*)$ of $(T,\overline{\chi})$ contains all the free variables $\{v_1, \ldots, v_f\}$. We designate t^* as the root of T, and then we run GYO elimination procedure [3] on the bags $\overline{\chi}(t)$ of $(T,\overline{\chi})$ to construct a vertex ordering σ' with $\mathsf{faqw}(\sigma') \leq \mathsf{fhtw}((T,\overline{\chi}))$. Moreover, if we choose to eliminate the vertices of the root t^* at the end of GYO elimination (after all other vertices have already been eliminated), we can make the free variables $\{v_1, \ldots, v_f\}$ appear before all other variables in σ', thus making sure that σ' is indeed in $\mathsf{EVO}(\varphi)$ and completing the proof. In particular, we apply GYO elimination as follows:

- If the tree T contains only one node t^*:
  - We eliminate vertices in $\overline{\chi}(t^*) - \{v_1, \ldots, v_f\}$ before eliminating $\{v_1, \ldots, v_f\}$.
  - We remove t^* from T, thus making T an empty tree.
- Otherwise, we pick a *leaf* node t of T (other than the root t^*). Let t' be the parent of t in T:
  - If $\overline{\chi}(t) \subseteq \overline{\chi}(t')$, then we remove node t from T along with the associated bag $\overline{\chi}(t)$.
  - Otherwise, $\overline{\chi}(t)$ must have a vertex u that is not in $\overline{\chi}(t')$. (Hence, by property (b) of Definition 7, u is not in $\overline{\chi}(t'')$ for all t'' in T other than t.)
    - If u is the only vertex in $\overline{\chi}(t)$, then we remove node t from T along with the associated bag $\overline{\chi}(t)$.
    - Otherwise, we remove u from $\overline{\chi}(t)$.
- We repeat the above steps until T becomes an empty tree.

□

B. MISSING DETAILS FROM SECTION 3

Proof of Theorem 3.1. We start with point evaluation:

$$\begin{aligned}
\frac{1}{2|D|}\sum_{(\mathbf{x},y)\in D}(\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle - y)^2 &= \frac{1}{2|D|}\sum_{(\mathbf{x},y)\in D}(\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle^2 - 2y\langle g(\boldsymbol{\theta}), h(\mathbf{x})\rangle + y^2) \\
&= \frac{1}{2|D|}\sum_{(\mathbf{x},y)\in D} g(\boldsymbol{\theta})^\top (h(\mathbf{x})h(\mathbf{x})^\top) g(\boldsymbol{\theta}) - \left\langle g(\boldsymbol{\theta}), \frac{1}{|D|}\sum_{(\mathbf{x},y)\in D} yh(\mathbf{x})\right\rangle + \frac{1}{2|D|}\sum_{(\mathbf{x},y)\in D} y^2 \\
&= \frac{1}{2} g(\boldsymbol{\theta})^\top \left(\frac{1}{|D|}\sum_{(\mathbf{x},y)\in D} h(\mathbf{x})h(\mathbf{x})^\top\right) g(\boldsymbol{\theta}) - \langle g(\boldsymbol{\theta}), \mathbf{c}\rangle + \frac{s_Y}{2} \\
&= \frac{1}{2} g(\boldsymbol{\theta})^\top \boldsymbol{\Sigma} g(\boldsymbol{\theta}) - \langle g(\boldsymbol{\theta}), \mathbf{c}\rangle + \frac{s_Y}{2}.
\end{aligned}$$

The gradient formula follows straightforwardly from (11) and the chain rule. □

Proof of Corollary 3.2. From (13) we have

$$\begin{aligned}
J(\boldsymbol{\theta}) - J(\boldsymbol{\theta} - \alpha\mathbf{d}) &= \frac{1}{2}\boldsymbol{\theta}^\top\boldsymbol{\Sigma}\boldsymbol{\theta} - \frac{1}{2}(\boldsymbol{\theta} - \alpha\mathbf{d})^\top\boldsymbol{\Sigma}(\boldsymbol{\theta} - \alpha\mathbf{d}) - \langle\boldsymbol{\theta}, \mathbf{c}\rangle + \langle\boldsymbol{\theta} - \alpha\mathbf{d}, \mathbf{c}\rangle + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{\theta} - \alpha\mathbf{d}\|_2^2 \\
&= \frac{1}{2}\boldsymbol{\theta}^\top\boldsymbol{\Sigma}\boldsymbol{\theta} - \frac{1}{2}\left(\boldsymbol{\theta}^\top\boldsymbol{\Sigma}\boldsymbol{\theta} - 2\alpha\boldsymbol{\theta}^\top\boldsymbol{\Sigma}\mathbf{d} + \alpha^2\mathbf{d}^\top\boldsymbol{\Sigma}\mathbf{d}\right) - \alpha\langle\mathbf{d}, \mathbf{c}\rangle + \lambda\alpha\langle\boldsymbol{\theta}, \mathbf{d}\rangle - \frac{\lambda\alpha^2}{2}\|\mathbf{d}\|_2^2 \\
&= \alpha\boldsymbol{\theta}^\top\boldsymbol{\Sigma}\mathbf{d} - \frac{\alpha^2}{2}\mathbf{d}^\top\boldsymbol{\Sigma}\mathbf{d} - \alpha\langle\mathbf{d}, \mathbf{c}\rangle + \lambda\alpha\langle\boldsymbol{\theta}, \mathbf{d}\rangle - \frac{\lambda\alpha^2}{2}\|\mathbf{d}\|_2^2.
\end{aligned}$$

□

Proof of Proposition 3.4. For any event E, let δ_E denote the Kronecker delta, i.e. $\delta_E = 1$ if E holds, and $\delta_E = 0$ otherwise. Recall that the input query Q has hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, and there is an input relation R_F for every

hyperedge $F \in \mathcal{E}$. Recall that we can write $\boldsymbol{\sigma}_{ij}$ in the tensor form as shown in Eq. (18). Plugging in the definition of h_i and h_j from (4); and, let $C_{ij} := C_i \cup C_j$ and $V_{ij} = V_i \cup V_j$, we have

$$\boldsymbol{\sigma}_{ij} = \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \prod_{f\in V_{ij}-C_{ij}} x_f^{a_i(f)+a_j(f)} \cdot \bigotimes_{f_i\in C_i} \mathbf{x}_{f_i} \otimes \bigotimes_{f_j\in C_j} \mathbf{x}_{f_j}.$$

As illustrated in Example 12, the tensor $\bigotimes_{f\in C_i} \mathbf{x}_f \otimes \bigotimes_{f\in C_j} \mathbf{x}_f$ is very sparse. For a fixed tuple $\mathbf{x}$, in fact, the tensor has only *one* 1 entry, corresponding to the combination of values of the attributes in C_{ij}. Hence, $\boldsymbol{\sigma}_{ij}$ is a function of the variables C_{ij}. In the FAQ-framework, the query representing $\boldsymbol{\sigma}_{ij}$ can be expressed as a Sum-Product queries with free (i.e., group-by) variables C_{ij}, defined by:

$$\varphi(C_{ij}) = \frac{1}{|D|} \sum_{x_{f'}: f'\in\mathcal{V}-C_{ij}} \prod_{f\in V_{ij}-C_{ij}} x_f^{a_i(f)+a_j(f)} \cdot \prod_{F\in\mathcal{E}} \delta_{\pi_F(\mathbf{x})\in R_F}. \tag{62}$$

Similarly, the tensor $\mathbf{c}_j$ can be sparsely represented by an aggregate query with group-by attributes C_j, which is expressed as the Sum-Product query

$$\varphi(C_j) = \frac{1}{|D|} \sum_{x_{f'}: f'\in\mathcal{V}-C_j} y \cdot \prod_{f\in V_j-C_j} x_f^{a_j(f)} \cdot \prod_{F\in\mathcal{E}} \delta_{\pi_F(\mathbf{x})\in R_F}. \tag{63}$$

The overall runtimes for computing the above FAQ-queries follow from applying the InsideOut algorithm and Theorem A.7 [6]. □

Proof of Proposition 3.5. The fact that $\mathsf{faqw}(i,j) \le \mathsf{fhtw}+c-1$ follows from Proposition A.9. Since $\boldsymbol{\sigma}_{ij}$ is a tensor of order at most c, and each attribute's active domain has size at most N, it follows that $|\boldsymbol{\sigma}_{ij}| \le N^c$. And, $|\boldsymbol{\sigma}_{ij}| \le |D|$ because the support of the tensor $\boldsymbol{\sigma}_{ij}$ cannot be more than the output size.

Fix a query Q with $\rho^* > \mathsf{fhtw} + c - 1 \ge c$. Consider a database instance I for which $|D|$ (the output size of Q) is $\Theta(N^{\rho^*})$. (The existence of such database instances is guaranteed by Theorem A.4.) From this (20) follows trivially. □

Proof of Proposition 3.6. We first analyze the time it takes to compute expression (11), which is dominated by the quadratic form $g(\boldsymbol{\theta})^\top \boldsymbol{\Sigma} g(\boldsymbol{\theta})$. To compute this quadratic form, for every pair $i,j \in [m]$ we need to compute $g_i(\boldsymbol{\theta})^\top \boldsymbol{\sigma}_{ij} g_j(\boldsymbol{\theta})$. This product is broken up into a sum of $t_i t_j$ terms when we expand g_i and g_j out. Each of those terms is computed in time $O(d_i d_j |\boldsymbol{\sigma}_{ij}|)$. The runtime for computing (12) is analyzed similarly. □

C. MISSING DETAILS FROM SECTION 4

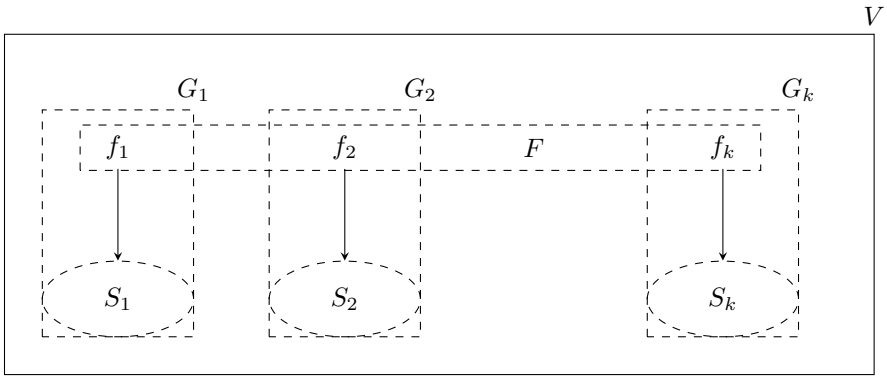


Figure 1: Groups of simple FDs. $G = G_1 \cup \cdots \cup G_k$.

In the proofs below, for each feature $w \in V$, $\mathbf{I}_w$ denote the identity matrix whose dimension is the size of the effective domain of w. This is not to be confused with the notation $\mathbf{I}_n$ which is an order-n identity matrix.

C.1 Missing rewriting steps in the example in Section 4.1

We first show that $\boldsymbol{\theta}_{\mathsf{country}} = (\mathbf{I}_{\mathsf{country}} + \mathbf{R}\mathbf{R}^\top)^{-1}\mathbf{R}\boldsymbol{\gamma}_{\mathsf{city}} = \mathbf{R}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}}$. By setting (27) to 0, we infer $\boldsymbol{\theta}_{\mathsf{country}} = (\mathbf{I}_{\mathsf{country}} + \mathbf{R}\mathbf{R}^\top)^{-1}\mathbf{R}\boldsymbol{\gamma}_{\mathsf{city}}$. To prove $\boldsymbol{\theta}_{\mathsf{country}} = \mathbf{R}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}}$, it is sufficient to show the identity

$$(\mathbf{I}_{\mathsf{country}} + \mathbf{R}\mathbf{R}^\top)^{-1}\mathbf{R} = \mathbf{R}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}.$$

To see this, multiply both sides on the left by $(\mathbf{I}_{\mathsf{country}} + \mathbf{R}\mathbf{R}^\top)$ and on the right by $(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})$), we obtain

$$(\mathbf{I}_{\mathsf{country}} + \mathbf{R}\mathbf{R}^\top)(\mathbf{I}_{\mathsf{country}} + \mathbf{R}\mathbf{R}^\top)^{-1}\mathbf{R}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R}) = (\mathbf{I}_{\mathsf{country}} + \mathbf{R}\mathbf{R}^\top)\mathbf{R}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})$$

which is equivalent to $\mathbf{R} + \mathbf{R}\mathbf{R}^\top\mathbf{R} = \mathbf{R} + \mathbf{R}\mathbf{R}^\top\mathbf{R}$.

We next show how to get the penalty form in the rewritten loss function:

$$J(\boldsymbol{\theta}) = \frac{1}{2|D|}\sum_{(\mathbf{x},y)\in D}(\langle\overline{g}(\boldsymbol{\gamma}),\overline{h}(\mathbf{x})\rangle - y)^2 + \frac{\lambda}{2}\left(\sum_{j\neq\mathsf{city}}\left\|\boldsymbol{\gamma}_j\right\|_2^2 + \left\langle(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle\right)$$

We start by replacing $\boldsymbol{\theta}_{\mathsf{country}} = \mathbf{R}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}}$ in the following sum in the original penalty term: $\left(\sum_{j\neq\mathsf{city}}\left\|\boldsymbol{\gamma}_j\right\|_2^2 + \left\|\boldsymbol{\gamma}_{\mathsf{city}} - \mathbf{R}^\top\boldsymbol{\theta}_{\mathsf{country}}\right\|_2^2 + \left\|\boldsymbol{\theta}_{\mathsf{country}}\right\|_2^2\right)$. We are only concerned with the last two terms in the summation, since they refer to $\boldsymbol{\theta}_{\mathsf{country}}$. We show that $\left\|\boldsymbol{\gamma}_{\mathsf{city}} - \mathbf{R}^\top\boldsymbol{\theta}_{\mathsf{country}}\right\|_2^2 + \left\|\boldsymbol{\theta}_{\mathsf{country}}\right\|_2^2 = \left\langle(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle$.

For the sake of brevity, define $\mathbf{B} = \mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R}$ so that $\boldsymbol{\theta}_{\mathsf{country}} = \mathbf{R}\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}$. Note that $\mathbf{B}$ is symmetric and thus its inverse is also symmetric: $\mathbf{B}^{-1} = \mathbf{B}^{-\mathbf{1}^\top}$. We rewrite the expression inside the ℓ_2-norm in the first term as follows:

$$\begin{aligned}\boldsymbol{\gamma}_{\mathsf{city}} - \mathbf{R}^\top\boldsymbol{\theta}_{\mathsf{country}} &= \boldsymbol{\gamma}_{\mathsf{city}} - \mathbf{R}^\top\mathbf{R}(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}} = \boldsymbol{\gamma}_{\mathsf{city}} - (\mathbf{I}_{\mathsf{city}} - \mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}}\\ &= \boldsymbol{\gamma}_{\mathsf{city}} - (\mathbf{B} - \mathbf{I}_{\mathsf{city}})\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}} = \boldsymbol{\gamma}_{\mathsf{city}} - \mathbf{B}\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}} + \mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}} = \mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}.\end{aligned}$$

We now rewrite the second term as follows:

$$\begin{aligned}\left\|\mathbf{R}\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}\right\|_2^2 &= \left\langle\mathbf{R}\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\mathbf{R}\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle = \left\langle(\mathbf{R}\mathbf{B}^{-1})^\top\mathbf{R}\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle = \left\langle\mathbf{B}^{-1}\mathbf{R}^\top\mathbf{R}\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle\\ &= \left\langle\mathbf{B}^{-1}(\mathbf{B} - \mathbf{I}_{\mathsf{city}})\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle = \left\langle(\mathbf{B}^{-1}\mathbf{B}\mathbf{B}^{-1} - \mathbf{B}^{-2})\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle = \left\langle(\mathbf{B}^{-1} - \mathbf{B}^{-2})\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle\\ &= \left\langle\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle - \left\langle\mathbf{B}^{-2}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle = \left\langle\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle - \left\langle\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle\\ &= \left\langle\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle - \left\|\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}\right\|_2^2.\end{aligned}$$

Putting together the two rewritings and we obtain the desired identity:

$$\begin{aligned}\left\|\boldsymbol{\gamma}_{\mathsf{city}} - \mathbf{R}^\top\boldsymbol{\theta}_{\mathsf{country}}\right\|_2^2 + \left\|\boldsymbol{\theta}_{\mathsf{country}}\right\|_2^2 &= \left\|\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}\right\|_2^2 + \left\langle\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle - \left\|\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}}\right\|_2^2 = \left\langle\mathbf{B}^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle\\ &= \left\langle(\mathbf{I}_{\mathsf{city}} + \mathbf{R}^\top\mathbf{R})^{-1}\boldsymbol{\gamma}_{\mathsf{city}},\boldsymbol{\gamma}_{\mathsf{city}}\right\rangle.\end{aligned}$$

C.2 Proof of Theorem 4.1

Proof. We start by breaking the loss term into two parts

$$\langle\boldsymbol{\theta},h(\mathbf{x})\rangle = \sum_{\|\mathbf{a}_V\|_1\leq d}\langle\boldsymbol{\theta}_{\mathbf{a}},h_{\mathbf{a}}(\mathbf{x})\rangle = \sum_{\substack{\|\mathbf{a}_V\|_1\leq d\\ \|\mathbf{a}_G\|_1=0}}\left\langle\boldsymbol{\theta}_{\mathbf{a}},\mathbf{x}^{\otimes\mathbf{a}}\right\rangle + \sum_{\substack{\|\mathbf{a}_V\|_1\leq d\\ \|\mathbf{a}_G\|_1>0}}\left\langle\boldsymbol{\theta}_{\mathbf{a}},\mathbf{x}^{\otimes\mathbf{a}}\right\rangle$$

and rewrite the second part:

$$\sum_{\substack{\|\mathbf{a}_V\|_1\leq d\\ \|\mathbf{a}_G\|_1>0}}\left\langle\boldsymbol{\theta}_{\mathbf{a}},\mathbf{x}^{\otimes\mathbf{a}}\right\rangle \tag{64}$$

$$= \sum_{\substack{\|\mathbf{a}_V\|_1\leq d\\ \|\mathbf{a}_G\|_1>0}}\left\langle\boldsymbol{\theta}_{\mathbf{a}},\mathbf{x}_{\overline{G}}^{\otimes\mathbf{a}_{\overline{G}}}\otimes\mathbf{x}_G^{\otimes\mathbf{a}_G}\right\rangle \tag{65}$$

$$= \sum_{\substack{\|\mathbf{a}_V\|_1\leq d\\ \|\mathbf{a}_G\|_1>0}}\left\langle\boldsymbol{\theta}_{\mathbf{a}},\mathbf{x}_{\overline{G}}^{\otimes\mathbf{a}_{\overline{G}}}\otimes\bigotimes_{\substack{i\in[k]\\ c\in G_i\\ a_c>0}}\mathbf{x}_c\right\rangle \tag{66}$$

$$= \sum_{\substack{\|\mathbf{a}_V\|_1\leq d\\ \|\mathbf{a}_G\|_1>0}}\left\langle\boldsymbol{\theta}_{\mathbf{a}},\mathbf{x}_{\overline{G}}^{\otimes\mathbf{a}_{\overline{G}}}\otimes\bigotimes_{\|\mathbf{a}_{G_i}\|_1>0}^{i\in[k]}\bigotimes_{\substack{c\in G_i\\ a_c>0}}\mathbf{R}_c\mathbf{x}_{f_i}\right\rangle \tag{67}$$

$$= \sum_{\substack{\|\mathbf{a}_V\|_1 \le d \\ \|\mathbf{a}_G\|_1 > 0}} \left\langle \boldsymbol{\theta}_{\mathbf{a}}, \mathbf{x}_{\overline{G}}^{\otimes \mathbf{a}_{\overline{G}}} \otimes \bigotimes_{\substack{i \in [k] \\ \|\mathbf{a}_{G_i}\|_1 > 0}} \left(\bigstar_{\substack{c \in G_i \\ a_c > 0}} \mathbf{R}_c \right) \mathbf{x}_{f_i} \right\rangle \tag{68}$$

$$= \sum_{\substack{\|\mathbf{a}_V\|_1 \le d \\ \|\mathbf{a}_G\|_1 > 0}} \left\langle \boldsymbol{\theta}_{\mathbf{a}}, \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{\substack{i \in [k] \\ \|\mathbf{a}_{G_i}\|_1 > 0}} \bigstar_{\substack{c \in G_i \\ a_c > 0}} \mathbf{R}_c \right) \left(\mathbf{x}_{\overline{G}}^{\otimes \mathbf{a}_{\overline{G}}} \otimes \bigotimes_{\substack{i \in [k] \\ \|\mathbf{a}_{G_i}\|_1 > 0}} \mathbf{x}_{f_i} \right) \right\rangle \tag{69}$$

$$= \sum_{\substack{\|\mathbf{a}_V\|_1 \le d \\ \|\mathbf{a}_G\|_1 > 0}} \left\langle \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{\substack{i \in [k] \\ \|\mathbf{a}_{G_i}\|_1 > 0}} \bigstar_{\substack{c \in G_i \\ a_c > 0}} \mathbf{R}_c \right)^{\top} \boldsymbol{\theta}_{\mathbf{a}}, \mathbf{x}_{\overline{G}}^{\otimes \mathbf{a}_{\overline{G}}} \otimes \bigotimes_{\substack{i \in [k] \\ \|\mathbf{a}_{G_i}\|_1 > 0}} \mathbf{x}_{f_i} \right\rangle \tag{70}$$

$$= \sum_{\substack{\|\mathbf{a}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{\substack{T \subseteq [k] \\ 0 < |T| \le d-q}} \sum_{U \in \mathcal{U}(T,q)} \left\langle \underbrace{\left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \bigstar_{c \in U \cap G_i} \mathbf{R}_c \right)^{\top}}_{\mathbf{R}_{\mathbf{a}_{\overline{G}}, U} \text{ defined in (30)}} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})}, \mathbf{x}_{\overline{G}}^{\otimes \mathbf{a}_{\overline{G}}} \otimes \bigotimes_{i \in T} \mathbf{x}_{f_i} \right\rangle \tag{71}$$

$$= \sum_{\substack{\|\mathbf{a}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{\substack{T \subseteq [k] \\ 0 < |T| \le d-q}} \left\langle \underbrace{\sum_{U \in \mathcal{U}(T,q)} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^{\top} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})}}_{\boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})}}, \mathbf{x}_{\overline{G}}^{\otimes \mathbf{a}_{\overline{G}}} \otimes \bigotimes_{i \in T} \mathbf{x}_{f_i} \right\rangle \tag{72}$$

$$= \sum_{\substack{\|\mathbf{a}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{\substack{T \subseteq [k] \\ 0 < |T| \le d-q}} \left\langle \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})}, \mathbf{x}_{\overline{G}}^{\otimes \mathbf{a}_{\overline{G}}} \otimes \bigotimes_{i \in T} \mathbf{x}_{f_i} \right\rangle \tag{73}$$

$$= \sum_{\|\mathbf{b}_{\overline{S}}\|_1 \le d} \left\langle \boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}}, \mathbf{x}_{\overline{S}}^{\otimes \mathbf{b}_{\overline{S}}} \right\rangle. \tag{74}$$

The equality at (71) is a bit loaded. What goes on there is that we broke the sum over $\mathbf{a}_V$ for which $\|\mathbf{a}_V\|_1 \le d$ and $\|\mathbf{a}_G\|_1 > 0$ into a nested triple sums. First of all, in order for $\|\mathbf{a}_G\|_1 > 0$, obviously $\|\mathbf{a}_{\overline{G}}\|_1 < d$ must hold, so we group by those tuples first. The remaining mass $\|\mathbf{a}_G\|_1$ can only be at most $d - \|\mathbf{a}_{\overline{G}}\|_1 = d - q$. Since all features in G are categorical, from the above analysis we have $\mathbf{a}_G = (a_g)_{g \in G} \in \{0,1\}^G$, i.e., $\mathbf{a}_G$ is a characteristic vector of a subset $U \subseteq G$. Let $T = \{i \mid U_i \neq \emptyset\}$. Then, in the second summation we group U by T. The third summation ranges over all choices of $U \cap G_i$, $i \in T$, for which the total mass is at most $d - q$. (Recall the definition of $\mathcal{U}(T,q)$ in (28).)

Next, in (72) we perform the reparameterization. Recall that $\mathbf{1}_{F_T|F}$ is the characteristic vector of the set $F_T := \{f_i\}_{i \in T}$ in the collection $F = \{f_1, \ldots, f_k\}$. The new parameter $\boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})}$ is indexed by the tuple $(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})$ whose support is $\overline{G} \cup F = \overline{S}$, i.e., the set of all features except for the ones functionally determined by features in F. After the reparameterization, the loss term is identical to the loss term of a PR^d model whose features are $\overline{S}$. This explains the collapsed pair $(\bar{g}, \bar{h})$ used in the theorem.

Next, we explore the new parameter and how it affects the penalty term. Consider a fixed pair $\mathbf{a}_{\overline{G}}$ and $T \subseteq [k]$ such that $T \neq \emptyset$ and $\|\mathbf{a}_{\overline{G}}\|_1 + |T| \le d$. The last condition is implicit for the set U to exist for which $U \cap G_i \neq \emptyset$ and $\|\mathbf{a}_{\overline{G}}\|_1 + |U| \le d$. Among all choices of U, we single out $U = F_T$ and write

$$\boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} = \sum_{\substack{U \subseteq G \\ U \cap G_i \neq \emptyset, \forall i \in T \\ \|\mathbf{a}_{\overline{G}}\|_1 + |U| \le d}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^{\top} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} = \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} + \sum_{\substack{F_T \neq U \subseteq G \\ U \cap G_i \neq \emptyset, \forall i \in T \\ \|\mathbf{a}_{\overline{G}}\|_1 + |U| \le d}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^{\top} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})}.$$

Now we are ready to write the penalty term $\|\boldsymbol{\theta}\|_2^2$ in terms of the new parameter $\boldsymbol{\gamma}$ and some "left-over" components of $\boldsymbol{\theta}$.

$$\|\boldsymbol{\theta}\|_2^2 = \sum_{\|\mathbf{a}_V\|_1 \le d} \|\boldsymbol{\theta}_{\mathbf{a}}\|_2^2$$

$$
\begin{aligned}
&= \sum_{\substack{\|\mathbf{a}_V\|_1 \le d \\ \|\mathbf{a}_G\|_1 = 0}} \|\boldsymbol{\theta}_{\mathbf{a}_V}\|_2^2 + \sum_{\substack{\|\mathbf{a}_V\|_1 \le d \\ \|\mathbf{a}_G\|_1 > 0}} \|\boldsymbol{\theta}_{\mathbf{a}_V}\|_2^2 \\
&= \sum_{\substack{\|\mathbf{a}_V\|_1 \le d \\ \|\mathbf{a}_G\|_1 = 0}} \|\boldsymbol{\theta}_{\mathbf{a}_V}\|_2^2 + \sum_{\substack{\|\mathbf{a}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{\substack{T \subseteq [k] \\ 0 < |T| \le d-q}} \sum_{U \in \mathcal{U}(T,q)} \left\| \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} \right\|_2^2 \\
&= \sum_{\substack{\|\mathbf{b}_{\overline{S}}\|_1 \le d \\ \|\mathbf{b}_F\|_1 = 0}} \left\| \boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}} \right\|_2^2 + \sum_{\substack{\|\mathbf{a}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{\substack{T \subseteq [k] \\ 0 < |T| \le d-q}} \left(\left\| \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} \right\|_2^2 + \sum_{\substack{W \in \mathcal{U}(T,q) \\ W \neq F_T}} \left\| \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} \right\|_2^2 \right) \\
&= \sum_{\substack{\|\mathbf{b}_{\overline{S}}\|_1 \le d \\ \|\mathbf{b}_F\|_1 = 0}} \left\| \boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}} \right\|_2^2 + \sum_{\substack{\|\mathbf{a}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{\substack{T \subseteq [k] \\ 0 < |T| \le d-q}} \left\| \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} - \sum_{\substack{U \in \mathcal{U}(T,q) \\ U \neq F_T}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^\top \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} \right\|_2^2 \\
&+ \sum_{\substack{\|\mathbf{a}_{\overline{G}}\|_1 = q \\ q < d}} \sum_{\substack{T \subseteq [k] \\ 0 < |T| \le d-q}} \sum_{\substack{W \in \mathcal{U}(T,q) \\ W \neq F_T}} \left\| \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{W|G})} \right\|_2^2 .
\end{aligned}
$$

Next, for every $W \in \mathcal{U}(T,q) - \{F_T\}$, we optimize out the parameter $\boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{W|G})}$ by noting that the new loss term does not depend on these parameters. Thus:

$$
\begin{aligned}
\frac{1}{2} \frac{\partial J}{\partial \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{W|G})}} &= \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{W|G})} - \mathbf{R}_{\mathbf{a}_{\overline{G}}, W} \left(\boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} - \sum_{\substack{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1) \\ U \neq F_T}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^\top \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} \right) \\
&= \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{W|G})} - \mathbf{R}_{\mathbf{a}_{\overline{G}}, W} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} .
\end{aligned}
$$

Setting this partial derivative to 0, we obtain $\boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{W|G})} = \mathbf{R}_{\mathbf{a}_{\overline{G}}, W} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})}$, which leads to

$$
\begin{aligned}
\boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} &= \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} - \sum_{\substack{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1) \\ U \neq F_T}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^\top \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} \\
&= \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} - \sum_{\substack{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1) \\ U \neq F_T}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^\top \mathbf{R}_{\mathbf{a}_{\overline{G}}, U} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} .
\end{aligned}
$$

Moving and grouping, we obtain

$$
\left(\bigotimes_{\substack{g \in \overline{G} \\ a_g > 0}} \mathbf{I}_g \otimes \bigotimes_{i \in T} \mathbf{I}_{f_i} + \sum_{\substack{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1) \\ U \neq F_T}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^\top \mathbf{R}_{\mathbf{a}_{\overline{G}}, U} \right) \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} = \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} .
$$

The matrix on the left can be completely factorized, as follows:

$$
\begin{aligned}
&\bigotimes_{\substack{g \in \overline{G} \\ a_g > 0}} \mathbf{I}_g \otimes \bigotimes_{i \in T} \mathbf{I}_{f_i} + \sum_{\substack{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1) \\ U \neq F_T}} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^\top \mathbf{R}_{\mathbf{a}_{\overline{G}}, U} \\
&= \sum_{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1)} \mathbf{R}_{\mathbf{a}_{\overline{G}}, U}^\top \mathbf{R}_{\mathbf{a}_{\overline{G}}, U} \\
&= \sum_{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1)} \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \bigstar_{c \in U \cap G_i} \mathbf{R}_c \right)^\top \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \bigstar_{c \in U \cap G_i} \mathbf{R}_c \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1)} \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right]^\top \right) \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \bigstar_{c \in U \cap G_i} \mathbf{R}_c \right) \\
&= \bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \sum_{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1)} \left(\bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right]^\top \right) \left(\bigotimes_{i \in T} \bigstar_{c \in U \cap G_i} \mathbf{R}_c \right) \\
&= \bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \sum_{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1)} \left(\bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right]^\top \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \right) \\
&= \bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \underbrace{\sum_{U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1)} \left(\left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right]^\top \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \right)}_{\mathbf{B}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \text{ defined in (29)}} \\
&= \bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \mathbf{B}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i}.
\end{aligned}
$$

Consequently, we can completely optimize out the remaining $\boldsymbol{\theta}$-components, solving for them in terms of the components of $\boldsymbol{\gamma}$:

$$
\begin{aligned}
\boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} &= \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \mathbf{B}^{-1}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} \\
\boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} &= \mathbf{R}_{\mathbf{a}_{\overline{G}}, U} \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|G})} \\
&= \mathbf{R}_{\mathbf{a}_{\overline{G}}, U} \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \mathbf{B}^{-1}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} \\
&= \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \bigstar_{c \in U \cap G_i} \mathbf{R}_c \right) \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \mathbf{B}^{-1}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} \\
&= \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})}.
\end{aligned}
$$

Since $\mathbf{B}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i}$ is a symmetric matrix, so is its inverse. For every $U \in \mathcal{U}(T, \left\|\mathbf{a}_{\overline{G}}\right\|_1)$, we simplify the norm:

$$
\begin{aligned}
&\left\| \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{U|G})} \right\|_2^2 \\
&= \left\| \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} \right\|_2^2 \\
&= \left\langle \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})}, \left(\bigotimes_{\substack{w \in \overline{G} \\ a_w > 0}} \mathbf{I}_w \otimes \bigotimes_{i \in T} \left[\bigstar_{c \in U \cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T, \left\|\mathbf{a}_{\overline{G}}\right\|_1, i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}}, \mathbf{1}_{F_T|F})} \right\rangle
\end{aligned}
$$

$$= \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T,\left\|\mathbf{a}_{\overline{G}}\right\|_1,i} \right)^\top \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T,\left\|\mathbf{a}_{\overline{G}}\right\|_1,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle$$

$$= \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \mathbf{B}^{-1}_{T,\left\|\mathbf{a}_{\overline{G}}\right\|_1,i} \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right]^\top \right) \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T,\left\|\mathbf{a}_{\overline{G}}\right\|_1,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle$$

$$= \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \mathbf{B}^{-1}_{T,\left\|\mathbf{a}_{\overline{G}}\right\|_1,i} \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right]^\top \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T,\left\|\mathbf{a}_{\overline{G}}\right\|_1,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle.$$

Thus, for a fixed T and $\mathbf{a}_{\overline{G}}$ with $\left\|\mathbf{a}_{\overline{G}}\right\|_1 = q$, we have

$$\sum_{U\in\mathcal{U}(T,\left\|\mathbf{a}_{\overline{G}}\right\|_1)} \left\| \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{U|G})} \right\|_2^2$$

$$= \sum_{U\in\mathcal{U}(T,q)} \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \mathbf{B}^{-1}_{T,q,i} \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right]^\top \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right] \mathbf{B}^{-1}_{T,q,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle$$

$$= \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \mathbf{B}^{-1}_{T,q,i} \left\{ \sum_{U\in\mathcal{U}(T,q)} \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right]^\top \left[\bigstar_{c\in U\cap G_i} \mathbf{R}_c \right] \right\} \mathbf{B}^{-1}_{T,q,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle$$

$$= \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \mathbf{B}^{-1}_{T,q,i} \mathbf{B}_{T,q,i} \mathbf{B}^{-1}_{T,q,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle$$

$$= \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \mathbf{B}^{-1}_{T,q,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle.$$

We next write $\|\boldsymbol{\theta}\|_2^2$ in terms of the new parameter $\boldsymbol{\gamma}$:

$$\begin{aligned}
\|\boldsymbol{\theta}\|_2^2 &= \sum_{\|\mathbf{a}_V\|_1\le d} \|\boldsymbol{\theta}_{\mathbf{a}}\|_2^2 \\
&= \sum_{\substack{\|\mathbf{a}_V\|_1\le d\\ \|\mathbf{a}_G\|_1=0}} \|\boldsymbol{\theta}_{\mathbf{a}_V}\|_2^2 + \sum_{\substack{\|\mathbf{a}_V\|_1\le d\\ \|\mathbf{a}_G\|_1>0}} \|\boldsymbol{\theta}_{\mathbf{a}_V}\|_2^2 \\
&= \sum_{\substack{\|\mathbf{a}_V\|_1\le d\\ \|\mathbf{a}_G\|_1=0}} \|\boldsymbol{\theta}_{\mathbf{a}_V}\|_2^2 + \sum_{\substack{\left\|\mathbf{a}_{\overline{G}}\right\|_1=q\\ q<d}} \sum_{\substack{T\subseteq[k]\\ 0<|T|\le d-q}} \sum_{U\in\mathcal{U}(T,q)} \left\| \boldsymbol{\theta}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{U|G})} \right\|_2^2 \\
&= \sum_{\substack{\left\|\mathbf{b}_{\overline{S}}\right\|_1\le d\\ \|\mathbf{b}_F\|_1=0}} \left\|\boldsymbol{\gamma}_{\mathbf{b}_{\overline{S}}}\right\|_2^2 + \sum_{\substack{\left\|\mathbf{a}_{\overline{G}}\right\|_1=q\\ q<d}} \sum_{\substack{T\subseteq[k]\\ 0<|T|\le d-q}} \left\langle \left(\bigotimes_{\substack{w\in\overline{G}\\ a_w>0}} \mathbf{I}_w \otimes \bigotimes_{i\in T} \mathbf{B}^{-1}_{T,q,i} \right) \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})}, \boldsymbol{\gamma}_{(\mathbf{a}_{\overline{G}},\mathbf{1}_{F_T|F})} \right\rangle.
\end{aligned}$$

□

C.3 Alternative to Corollary 3.2

One big advantage of a linear model in terms of BGD is Corollary 3.2, where we do not have to redo point-evaluation for every backtracking step. After the reparameterization exploiting FD-based dimensionality reduction, Corollary 3.2 does not work as is, because we have changed the penalty terms. However, it is easy to work out a similar result in terms of the new parameter space; see The point of the following proposition is that we only need to compute intermediate results involving the covariance matrix $\overline{\boldsymbol{\Sigma}}$ once while backtracking. For each new value of α, we will need to recompute the penalty's objective $\overline{\Omega}(\boldsymbol{\gamma} - \alpha\mathbf{d})$, which is an inexpensive operation. If $\lambda = 0$, we can even solve for α directly.

Proposition C.1. *With respect to the new parameters (and new objective $\overline{J}$ defined in (31)), the Armijo condition $\overline{J}(\boldsymbol{\gamma}) - \overline{J}(\boldsymbol{\gamma} - \alpha\mathbf{d}) \leq \frac{\alpha}{2}\|\mathbf{d}\|_2^2$ is equivalent to*

$$\alpha\left(2\boldsymbol{\gamma}^\top\overline{\boldsymbol{\Sigma}}\mathbf{d} - \alpha\mathbf{d}\overline{\boldsymbol{\Sigma}}\mathbf{d} - 2\langle\mathbf{d}, \overline{\mathbf{c}}\rangle - \|\mathbf{d}\|_2^2\right) + \lambda\overline{\Omega}(\boldsymbol{\gamma})) \leq \lambda\overline{\Omega}(\boldsymbol{\gamma} - \alpha\mathbf{d}),$$

where $\mathbf{d} = \boldsymbol{\nabla}\overline{J}(\boldsymbol{\gamma})$. Furthermore, the next gradient of $\overline{J}$ is also readily available:

$$\frac{\partial\overline{J}(\boldsymbol{\gamma} - \alpha\mathbf{d})}{\partial\boldsymbol{\gamma}} = \mathbf{d} - \alpha\overline{\boldsymbol{\Sigma}}\mathbf{d} + \frac{\lambda}{2}\left(\frac{\partial\Omega(\boldsymbol{\gamma} - \alpha\mathbf{d})}{\partial\boldsymbol{\gamma}} - \frac{\partial\Omega(\boldsymbol{\gamma})}{\partial\boldsymbol{\gamma}}\right).$$

Proof. Let $\mathbf{d} = \boldsymbol{\nabla}\overline{J}(\boldsymbol{\gamma})$. Then,

$$\begin{aligned}\overline{J}(\boldsymbol{\gamma}) - \overline{J}(\boldsymbol{\gamma} - \alpha\mathbf{d}) &= \frac{1}{2}\boldsymbol{\gamma}^\top\overline{\boldsymbol{\Sigma}}\boldsymbol{\gamma} - \frac{1}{2}(\boldsymbol{\gamma} - \alpha\mathbf{d})\overline{\boldsymbol{\Sigma}}(\boldsymbol{\gamma} - \alpha\mathbf{d}) + \langle\boldsymbol{\gamma} - \alpha\mathbf{d}, \overline{\mathbf{c}}\rangle + \frac{\lambda}{2}(\overline{\Omega}(\boldsymbol{\gamma}) - \overline{\Omega}(\boldsymbol{\gamma} - \alpha\mathbf{d})) \\ &= \alpha\boldsymbol{\gamma}^\top\overline{\boldsymbol{\Sigma}}\mathbf{d} - \frac{\alpha^2}{2}\mathbf{d}\overline{\boldsymbol{\Sigma}}\mathbf{d} - \alpha\langle\mathbf{d}, \overline{\mathbf{c}}\rangle + \frac{\lambda}{2}(\overline{\Omega}(\boldsymbol{\gamma}) - \overline{\Omega}(\boldsymbol{\gamma} - \alpha\mathbf{d})).\end{aligned}$$

□

C.4 Specializing Theorem 4.1 to the LR model

This section specializes Theorem 4.1 to the LR-model. Let us first specialize expressions (28), (29), and (30). We start with (28). Since $d = 1$, the only valid choice of q is 0, and $|T| = 1$. If $T = \{j\}$, then $U \in \mathcal{U}(T, q)$ iff $U = \{c\}$ for some $c \in G_j$. In other words, we can replace $\mathcal{U}(T, q)$ by G_j itself. Next, consider (30): there is only one valid choice of $a_{\overline{G}}$ – the all 0 vector – and $U = \{c\}$ for some $c \in G_j$, the matrix $\mathbf{R}_{\mathbf{a}_{\overline{G}}, U}$ is *exactly* $\mathbf{R}_c$. Lastly, when $T = \{j\}$ the sum (29) becomes $\sum_{c\in G_j}\mathbf{R}_c^\top\mathbf{R}_c$. We have the following corollary:

Corollary C.2. *Consider a* LR *model with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_w)_{w\in V}$ and k groups of simple FDs $G_i = \{f_i\} \cup S_i$, $i \in [k]$. Define the following reparameterization:*

$$\boldsymbol{\gamma}_w = \begin{cases}\boldsymbol{\theta}_w & w \in V - G, \\ \sum_{c\in G_i}\mathbf{R}_c^\top\boldsymbol{\theta}_c & w \in F.\end{cases}$$

Then, minimizing $J(\boldsymbol{\theta})$ is equivalent to minimizing the function $\overline{J}(\boldsymbol{\gamma}) := \frac{1}{2}\boldsymbol{\gamma}^\top\overline{\boldsymbol{\Sigma}}\boldsymbol{\gamma} - \langle\boldsymbol{\gamma}, \overline{\mathbf{c}}\rangle + \frac{\lambda}{2}\Omega(\boldsymbol{\gamma})$, where $\Omega(\boldsymbol{\gamma}) := \sum_{w\in V\setminus G}\|\boldsymbol{\gamma}_w\|_2^2 + \sum_{i=1}^k\langle\mathbf{B}_i^{-1}\boldsymbol{\gamma}_{f_i}, \boldsymbol{\gamma}_{f_i}\rangle$, and matrix B_i for each $i \in [k]$ is given by

$$\mathbf{B}_i := \sum_{c\in G_i}\mathbf{R}_c^\top\mathbf{R}_c. \tag{75}$$

$\overline{J}$ is defined with respect to the FD-reduced pair of functions $\overline{g}, \overline{h}$ and a reduced parameter space of $\boldsymbol{\gamma}$. Its gradient is very simple to compute, where we specialize (32):

$$\frac{1}{2}\frac{\partial\Omega(\boldsymbol{\gamma})}{\partial\boldsymbol{\gamma}_w} = \begin{cases}\boldsymbol{\gamma}_w & w \in V - G, \\ \mathbf{B}_i^{-1}\boldsymbol{\gamma}_{f_i} & w \in F.\end{cases} \tag{76}$$

Moreover, once a minimizer $\boldsymbol{\gamma}$ of $\overline{J}$ is obtained, following (33), we can compute a minimizer $\boldsymbol{\theta}$ of J by setting

$$\boldsymbol{\theta}_w = \begin{cases}\boldsymbol{\gamma}_w & w \in V \setminus G, \\ \mathbf{R}_w\mathbf{B}_i^{-1}\boldsymbol{\gamma}_{f_i} & w \in G_i, i \in [k].\end{cases}$$

C.5 Specializing Theorem 4.1 to the PR^2 model

In this section we explores Theorem 4.1 for the special case of degree-2 polynomial regression. This case is significant for three reasons. First, due to the explosion in the number of parameters, in practice one rarely runs polynomial regression of degree higher than 2. In fact, PR^2 may be a sufficiently rich nonlinear regression model for many real-world applications. Second, this is technically already a highly non-trivial application of our general theorem. Third, this case shares some commonality with FaMa_r^2 model to be described in the next section.

As before, we first specialize expressions (28), (29), and (30). To do so, we change the indexing scheme of the model a bit. In the general model, we use $\mathbf{a}$ with $\|\mathbf{a}\|_1 \le d$ to index parameters. When the model is of degree 2, we explicitly write down the two types of indices: we use $\boldsymbol{\theta}_w$, $w \in V$ instead of $\boldsymbol{\theta}_{\mathbf{a}}$ with $\|\mathbf{a}\|_1 = 1$, and we use $\boldsymbol{\theta}_{cw}$ with $c, w \in V$ instead of $\boldsymbol{\theta}_{\mathbf{a}}$ when $\|\mathbf{a}\|_1 = 2$.

We start with (28). Since $d = 2$, two valid choices of q are 0 and 1.

- when $q = 1$, $|T| = \{i\}$ for some $i \in [k]$. The set $\mathcal{U}(\{i\}, 1)$ is the collection of singleton subsets of G_i. Hence, this is similar to the linear regression situation.
- when $q = 0$, $|T|$ is either $\{i\}$ or $\{i, j\}$. The set $\mathcal{U}(\{i, j\}, 0)$ consists of all 2-subsets U of G for which U contains one element from G_i and one from G_j. The set $\mathcal{U}(\{i\}, 0)$ contains all singletons and 2-subsets of G_i.

From this analysis, we can write down (29) explicitly (also recall the definition of $\mathbf{B}_i$ in (75)):

$$\begin{aligned}
\mathbf{B}_{\{i\},1,i} &= \sum_{c \in G_i} \mathbf{R}_c^\top \mathbf{R}_c = \mathbf{B}_i \\
\mathbf{B}_{\{i,j\},0,i} &= \mathbf{B}_i \\
\mathbf{B}_{\{i,j\},0,j} &= \mathbf{B}_j \\
\mathbf{B}_{\{i\},0} &= \sum_{c \in G_i} \mathbf{R}_c^\top \mathbf{R}_c + \sum_{\{c,t\} \in \binom{G_i}{2}} (\mathbf{R}_c \star \mathbf{R}_t)^\top (\mathbf{R}_c \star \mathbf{R}_t).
\end{aligned}$$

Next, consider (30): there are two valid choices for the pair $(\mathbf{a}_{\overline{G}}, U)$:

- when $\|\mathbf{a}_{\overline{G}}\|_1 = 0$, $U \in \mathcal{U}(\{i, j\}, 0)$ or $U \in \mathcal{U}(\{i\}, 0)$. In that case, we have

$$\begin{aligned}
\mathbf{R}_{\emptyset,\{c,t\}} &= \mathbf{R}_c \otimes \mathbf{R}_t && (c,t) \in G_i \times G_j \\
\mathbf{R}_{\emptyset,\{c\}} &= \mathbf{R}_c && c \in G_i \\
\mathbf{R}_{\emptyset,\{c,t\}} &= \mathbf{R}_c \star \mathbf{R}_t && \{c,t\} \in \binom{G_i}{2}.
\end{aligned}$$

- when $\|\mathbf{a}_{\overline{G}}\|_1 = 1$, $U \in \mathcal{U}(\{i\}, 0)$ for some $i \in [k]$; and in this case we use $w \in \overline{G}$ to represent $\mathbf{a}_{\overline{G}}$ ($a_w > 0$):

$$\mathbf{R}_{w,\{c\}} = \mathbf{I}_w \otimes \mathbf{R}_c.$$

Corollary C.3. *Consider the PR^2 model with k groups of simple FDs $G_i = \{f_i\} \cup S_i$, $i \in [k]$. Let*

$$\boldsymbol{\theta} = ((\boldsymbol{\theta}_w)_{w \in V}, (\boldsymbol{\theta}_{cw})_{c,w \in v})$$

be the original parameters, and $G = \cup_{i \in [k]} G_i$. Define the following reparameterization:

$$\boldsymbol{\gamma}_w = \begin{cases} \boldsymbol{\theta}_w & w \in V \setminus G \\ \displaystyle\sum_{c \in G_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c + \sum_{\{c,t\} \in \binom{G_i}{2}} (\mathbf{R}_c \star \mathbf{R}_t)^\top \boldsymbol{\theta}_{ct} & \substack{w = f_i \\ i \in [k].} \end{cases} \tag{77}$$

$$\boldsymbol{\gamma}_{tw} = \begin{cases} \boldsymbol{\theta}_{tw} & \{t,w\} \subseteq V \setminus G \\ \displaystyle\sum_{c \in G_i} (\mathbf{I}_w \otimes \mathbf{R}_c^\top) \boldsymbol{\theta}_{wc} & t = f_i, w \notin G \\ \displaystyle\sum_{(c,c') \in G_i \times G_j} (\mathbf{R}_c \star \mathbf{R}_{c'})^\top \boldsymbol{\theta}_{cc'} & \substack{\{t,w\} = \{f_i, f_j\}, \\ \{i,j\} \in \binom{[k]}{2}.} \end{cases} \tag{78}$$

Then, minimizing $J(\boldsymbol{\theta})$ is equivalent to minimizing the function $\overline{J}(\boldsymbol{\gamma}) := \frac{1}{2}\boldsymbol{\gamma}^\top \overline{\boldsymbol{\Sigma}}\boldsymbol{\gamma} - \langle \boldsymbol{\gamma}, \overline{\mathbf{c}} \rangle + \frac{\lambda}{2}\Omega(\boldsymbol{\gamma})$, where

$$\begin{aligned}\Omega(\boldsymbol{\gamma}) := &\sum_{w \notin G} \|\boldsymbol{\gamma}_w\|_2^2 + \sum_{\substack{c \notin G \\ t \notin G}} \|\boldsymbol{\gamma}_{ct}\|_2^2 + \sum_{i=1}^{k} \left\langle \mathbf{B}_{\{i\},0}^{-1} \boldsymbol{\gamma}_{f_i}, \boldsymbol{\gamma}_{f_i} \right\rangle \\ &+ \sum_{\substack{i \in [k] \\ w \notin G}} \left\langle (\mathbf{I}_w \otimes \mathbf{B}_i^{-1} \boldsymbol{\gamma}_{wf_i}, \boldsymbol{\gamma}_{wf_i} \right\rangle \\ &+ \sum_{ij \in \binom{[k]}{2}} \left\langle \mathbf{B}_i^{-1} \otimes \mathbf{B}_j^{-1} \boldsymbol{\gamma}_{f_i f_j}, \gamma_{f_i f_j} \right\rangle.\end{aligned}$$

The gradient of $\overline{J}$ is very simple to compute, by noticing that $\overline{J}$ is defined with respect to the FD-reduced pair of functions $\overline{g}, \overline{h}$ and a reduced parameter space of $\boldsymbol{\gamma}$. Its gradient can be computed by specializing (32):

$$\frac{1}{2}\frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_w} = \begin{cases} \boldsymbol{\gamma}_w & w \notin G \\ \mathbf{B}_{\{i\},0}^{-1} \boldsymbol{\gamma}_{f_i} & w = f_i \end{cases} \tag{79}$$

$$\frac{1}{2}\frac{\partial \Omega(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}_{tw}} = \begin{cases} \boldsymbol{\gamma}_{tw} & \{t, w\} \cap \{f_i\}_{i=1}^k = \emptyset \\ (\mathbf{I}_w \otimes \mathbf{B}_i^{-1}) \boldsymbol{\gamma}_{wf_i} & t = f_i, w \notin G \\ (\mathbf{B}_i^{-1} \otimes \mathbf{B}_j^{-1}) \boldsymbol{\gamma}_{f_i f_j} & \{t, w\} = \{f_i, f_j\}. \end{cases} \tag{80}$$

Moreover, once a minimizer $\boldsymbol{\gamma}$ of $\overline{J}$ is obtained, following (33), we can compute a minimizer $\boldsymbol{\theta}$ of J by setting

$$\begin{aligned}\boldsymbol{\theta}_w &= \begin{cases} \boldsymbol{\gamma}_w & w \in V \setminus G \\ \mathbf{R}_w \mathbf{B}_{\{i\},0}^{-1} \boldsymbol{\gamma}_{f_i} & w \in G_i, i \in [k] \end{cases} \\ \boldsymbol{\theta}_{ct} &= (\mathbf{R}_c \star \mathbf{R}_t) \mathbf{B}_{\{i\},0}^{-1} \boldsymbol{\gamma}_{f_i}, \forall \{c, t\} \in \binom{G_i}{2} \\ \boldsymbol{\theta}_{cw} &= \begin{cases} \boldsymbol{\gamma}_{cw} & w \in V \setminus G \\ (\mathbf{I}_w \otimes \mathbf{R}_c \mathbf{B}_i^{-1}) \boldsymbol{\gamma}_{wf_i}, & c \in G_i, w \notin G, i \in [k] \end{cases} \\ \boldsymbol{\theta}_{ct} &= (\mathbf{R}_c \mathbf{B}_i^{-1} \otimes \mathbf{R}_t \mathbf{B}_j^{-1}) \boldsymbol{\gamma}_{f_i f_j}, (c, t) \in G_i \times G_j.\end{aligned}$$

C.6 Proofs of results in Section 4.4

Proof of Theorem 4.2. We begin with a similar derivation, whereby "relevant terms" of $\langle g(\boldsymbol{\theta}), h(\mathbf{x}) \rangle$ are the terms where h contains a feature $c \in F_i$ for some $i \in [k]$:

$$\begin{aligned}&\text{relevant terms of } \langle g(\boldsymbol{\theta}), h(\mathbf{x}) \rangle \\ = &\sum_{\substack{c \in F_i \\ i \in [k]}} \langle \boldsymbol{\theta}_c, \mathbf{x}_c \rangle + \sum_{\substack{\{c,t\} \in \binom{F_i}{2} \\ i \in [k] \\ \ell \in [r]}} \left\langle \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_t^{(\ell)}, \mathbf{x}_c \otimes \mathbf{x}_t \right\rangle + \sum_{ij \in \binom{[k]}{2}} \sum_{\substack{c \in F_i \\ t \in F_j \\ \ell \in [r]}} \left\langle \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_t^{(\ell)}, \mathbf{x}_c \otimes \mathbf{x}_t \right\rangle + \sum_{\substack{c \in F \\ w \notin F \\ \ell \in [r]}} \left\langle \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_w^{(\ell)}, \mathbf{x}_c \otimes \mathbf{x}_w \right\rangle \\ = &\sum_{\substack{c \in F_i \\ i \in [k]}} \langle \boldsymbol{\theta}_c, \mathbf{R}_c \mathbf{x}_{f_i} \rangle + \sum_{\substack{\{c,t\} \in \binom{F_i}{2} \\ i \in [k] \\ \ell \in [r]}} \left\langle \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_t^{(\ell)}, \mathbf{R}_c \mathbf{x}_{f_i} \otimes \mathbf{R}_t \mathbf{x}_{f_i} \right\rangle \\ &+ \sum_{ij \in \binom{[k]}{2}} \sum_{\substack{c \in F_i \\ t \in F_j \\ \ell \in [r]}} \left\langle \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_t^{(\ell)}, \mathbf{R}_c \mathbf{x}_{f_i} \otimes \mathbf{R}_t \mathbf{x}_{f_j} \right\rangle + \sum_{\substack{i \in [k] \\ c \in F_i \\ w \notin F \\ \ell \in [r]}} \left\langle \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_w^{(\ell)}, \mathbf{R}_c \mathbf{x}_{f_i} \otimes \mathbf{x}_w \right\rangle \\ = &\sum_{\substack{c \in F_i \\ i \in [k]}} \left\langle \mathbf{R}_c^\top \boldsymbol{\theta}_c, \mathbf{x}_{f_i} \right\rangle + \sum_{\substack{\{c,t\} \in \binom{F_i}{2} \\ i \in [k] \\ \ell \in [r]}} \left\langle \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \otimes \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_{f_i} \right\rangle\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{ij \in \binom{[k]}{2}} \sum_{\substack{c \in F_i \\ t \in F_j \\ \ell \in [r]}} \left\langle \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \otimes \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_{f_j} \right\rangle + \sum_{\substack{i \in [k] \\ c \in F_i \\ w \notin F \\ \ell \in [r]}} \left\langle \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_w^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_w \right\rangle \\
= &\sum_{\substack{c \in F_i \\ i \in [k]}} \left\langle \mathbf{R}_c^\top \boldsymbol{\theta}_c, \mathbf{x}_{f_i} \right\rangle + \sum_{\substack{\{c,t\} \in \binom{F_i}{2} \\ i \in [k] \\ \ell \in [r]}} \left\langle \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \circ \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)}, \mathbf{x}_{f_i} \right\rangle \\
&+ \sum_{\substack{ij \in \binom{[k]}{2} \\ \ell \in [r]}} \left\langle \sum_{c \in F_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \otimes \sum_{t \in F_j} \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_{f_j} \right\rangle + \sum_{\substack{i \in [k] \\ w \notin F \\ \ell \in [r]}} \left\langle \sum_{c \in F_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \otimes \boldsymbol{\theta}_w^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_w \right\rangle \\
= &\sum_{i=1}^k \Bigg\langle \underbrace{\sum_{c \in F_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c + \sum_{\ell=1}^r \sum_{\{c,t\} \in \binom{F_i}{2}} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \circ \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)}}_{\boldsymbol{\gamma}_{f_i}}, \mathbf{x}_{f_i} \Bigg\rangle \\
&+ \sum_{\substack{ij \in \binom{[k]}{2} \\ \ell \in [r]}} \left\langle \underbrace{\sum_{c \in F_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)}}_{\boldsymbol{\gamma}_{f_i}^{(\ell)}} \otimes \underbrace{\sum_{t \in F_j} \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)}}_{\boldsymbol{\gamma}_{f_j}^{(\ell)}}, \mathbf{x}_{f_i} \otimes \mathbf{x}_{f_j} \right\rangle + \sum_{\substack{i \in [k] \\ w \notin F \\ \ell \in [r]}} \left\langle \underbrace{\sum_{c \in F_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)}}_{\boldsymbol{\gamma}_{f_i}^{(\ell)}} \otimes \boldsymbol{\theta}_w^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_w \right\rangle \\
= &\sum_{i=1}^k \left\langle \boldsymbol{\gamma}_{f_i}, \mathbf{x}_{f_i} \right\rangle + \sum_{\substack{ij \in \binom{[k]}{2} \\ \ell \in [r]}} \left\langle \boldsymbol{\gamma}_{f_i}^{(\ell)} \otimes \boldsymbol{\gamma}_{f_j}^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_{f_j} \right\rangle + \sum_{\substack{i \in [k] \\ w \notin F \\ \ell \in [r]}} \left\langle \boldsymbol{\gamma}_{f_i}^{(\ell)} \otimes \boldsymbol{\theta}_w^{(\ell)}, \mathbf{x}_{f_i} \otimes \mathbf{x}_w \right\rangle.
\end{aligned}
$$

The above derivation immediately yields the reparameterization given in the statement of the theorem, which we reproduce here for the sake of clarity:

$$
\begin{aligned}
\boldsymbol{\gamma}_w &= \begin{cases} \boldsymbol{\theta}_w & w \notin F \\ \boldsymbol{\theta}_{f_i} + \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c + \boldsymbol{\beta}_{f_i} & w = f_i, i \in [k]. \end{cases} \\
\boldsymbol{\gamma}_w^{(\ell)} &= \begin{cases} \boldsymbol{\theta}_w^{(\ell)} & w \notin \{f_1, \dots, f_k\} \\ \boldsymbol{\theta}_{f_i}^{(\ell)} + \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} & w = f_i, i \in [k]. \end{cases}
\end{aligned}
$$

Note that we did not define $\boldsymbol{\gamma}_w$ for $w \in S_i$, $i \in [k]$. The reason we can do so, is because we can optimize out $\boldsymbol{\theta}_c$ due to the following trick we have been using (as in the proof of Theorem 4.1). First, we rewrite all the terms in $\|\boldsymbol{\theta}\|_2^2$ in terms of $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}_c$, $c \in S_i$, $i \in [k]$:

$$
\begin{aligned}
\|\boldsymbol{\theta}\|_2^2 &= \sum_{w \notin F} \|\boldsymbol{\theta}_w\|_2^2 + \sum_{i=1}^k \sum_{t \in F_i} \|\boldsymbol{\theta}_t\|_2^2 + \sum_{\ell=1}^r \sum_{w \notin \{f_1, \dots, f_k\}} \left\|\boldsymbol{\theta}_w^{(\ell)}\right\|_2^2 + \sum_{\ell=1}^r \sum_{i=1}^k \left\|\boldsymbol{\theta}_{f_i}^{(\ell)}\right\|_2^2 \\
&= \sum_{w \notin F} \|\boldsymbol{\gamma}_w\|_2^2 + \sum_{i=1}^k \sum_{t \in F_i} \|\boldsymbol{\theta}_t\|_2^2 + \sum_{\ell=1}^r \sum_{w \notin \{f_1, \dots, f_k\}} \left\|\boldsymbol{\gamma}_w^{(\ell)}\right\|_2^2 + \sum_{\ell=1}^r \sum_{i=1}^k \left\|\boldsymbol{\theta}_{f_i}^{(\ell)}\right\|_2^2 \\
&= \sum_{w \notin F} \|\boldsymbol{\gamma}_w\|_2^2 + \sum_{i=1}^k \left\|\boldsymbol{\gamma}_{f_i} - \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c - \boldsymbol{\beta}_{f_i}\right\|_2^2 + \sum_{i=1}^k \sum_{t \in S_i} \|\boldsymbol{\theta}_t\|_2^2 + \sum_{\ell=1}^r \sum_{w \notin \{f_1, \dots, f_k\}} \left\|\boldsymbol{\gamma}_w^{(\ell)}\right\|_2^2 + \sum_{\ell=1}^r \sum_{i=1}^k \left\|\boldsymbol{\gamma}_{f_i}^{(\ell)} - \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\gamma}_c^{(\ell)}\right\|_2^2
\end{aligned}
$$

Since $\boldsymbol{\theta}_t$,$t \in S_i$, does not depend on the loss term, we have

$$\frac{1}{2}\frac{\partial J}{\partial \boldsymbol{\theta}_t} = \boldsymbol{\theta}_t - \mathbf{R}_t \left(\underbrace{\boldsymbol{\gamma}_{f_i} - \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c - \boldsymbol{\beta}_{f_i}}_{\boldsymbol{\theta}_{f_i}} \right) \qquad w \in S_i, i \in [k]. \tag{81}$$

By setting (81) to 0, we have $\boldsymbol{\theta}_t = \mathbf{R}_t \boldsymbol{\theta}_{f_i}$ for all $t \in F_i$, and thus

$$\boldsymbol{\theta}_{f_i} = \boldsymbol{\gamma}_{f_i} - \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c - \boldsymbol{\beta}_{f_i} = \boldsymbol{\gamma}_{f_i} - \sum_{c \in S_i} \mathbf{R}_c^\top \mathbf{R}_c \boldsymbol{\theta}_{f_i} - \boldsymbol{\beta}_{f_i},$$

which implies $\boldsymbol{\theta}_{f_i} = \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i})$. Hence, the following always holds:

$$\boldsymbol{\theta}_t = \mathbf{R}_t \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), \qquad \forall t \in F_i, i \in [k].$$

Note also that,

$$\begin{aligned}
\sum_{t \in F_i} \|\boldsymbol{\theta}_t\|_2^2 &= \sum_{t \in F_i} \left\| \mathbf{R}_t \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) \right\|_2^2 \\
&= \sum_{t \in F_i} \left\langle \mathbf{R}_t^\top \mathbf{R}_t \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) \right\rangle \\
&= \left\langle \left(\sum_{t \in F_i} \mathbf{R}_t^\top \mathbf{R}_t \right) \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) \right\rangle \\
&= \left\langle \mathbf{B}_i \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) \right\rangle \\
&= \left\langle (\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) \right\rangle .
\end{aligned}$$

Due to the fact that $\boldsymbol{\theta}_{f_i}^{(\ell)} = \boldsymbol{\gamma}_{f_i}^{(\ell)} - \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\gamma}_c^{(\ell)}$, we can now write the penalty term in terms of the new parameter $\boldsymbol{\gamma}$:

$$\begin{aligned}
\|\boldsymbol{\theta}\|_2^2 &= \sum_{w \notin F} \|\boldsymbol{\gamma}_w\|_2^2 + \sum_{i=1}^{k} \sum_{t \in F_i} \|\boldsymbol{\theta}_t\|_2^2 + \sum_{\ell=1}^{r} \sum_{w \notin \{f_1, \ldots, f_k\}} \left\| \boldsymbol{\gamma}_w^{(\ell)} \right\|_2^2 + \sum_{\ell=1}^{r} \sum_{i=1}^{k} \left\| \boldsymbol{\theta}_{f_i}^{(\ell)} \right\|_2^2 \\
&= \sum_{w \notin F} \|\boldsymbol{\gamma}_w\|_2^2 + \sum_{i=1}^{k} \left\langle (\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}), \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) \right\rangle + \sum_{\ell=1}^{r} \sum_{w \notin \{f_1, \ldots, f_k\}} \left\| \boldsymbol{\gamma}_w^{(\ell)} \right\|_2^2 + \sum_{\ell=1}^{r} \sum_{i=1}^{k} \left\| \boldsymbol{\gamma}_{f_i}^{(\ell)} - \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\gamma}_c^{(\ell)} \right\|_2^2 .
\end{aligned}$$

□

Proof of Proposition 4.3. The goal is to derive the gradient of $\Omega(\boldsymbol{\gamma})$ w.r.t the parameters $\boldsymbol{\gamma}$. Since $\boldsymbol{\beta}_{f_i}$ is a function of $\boldsymbol{\gamma}_c^{(\ell)}$, $\ell \in [r]$, $c \in F_i$, the following is immediate:

$$\begin{aligned}
\frac{1}{2}\frac{\partial \|\boldsymbol{\theta}\|_2^2}{\partial \boldsymbol{\gamma}_w} &= \begin{cases} \boldsymbol{\gamma}_w, & w \notin F \\ \mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i} - \boldsymbol{\beta}_{f_i}) & w = f_i, i \in [k]. \end{cases} \\
\frac{1}{2}\frac{\partial \|\boldsymbol{\theta}\|_2^2}{\partial \boldsymbol{\gamma}_w^{(\ell)}} &= \boldsymbol{\gamma}_w, \qquad w \notin F, \ell \in [r].
\end{aligned}$$

Next, we have to simplify $\boldsymbol{\beta}_{f_i}$ to facilitate fast computation:

$$\begin{aligned}
\boldsymbol{\beta}_{f_i} &:= \sum_{\ell=1}^{r} \sum_{\{c,t\} \in \binom{F_i}{2}} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \circ \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)} \\
&= \sum_{\ell=1}^{r} \left[\mathbf{R}_{f_i}^\top \boldsymbol{\theta}_{f_i}^{(\ell)} \circ \sum_{c \in S_i} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} + \sum_{\{c,t\} \in \binom{S_i}{2}} \mathbf{R}_c^\top \boldsymbol{\theta}_c^{(\ell)} \circ \mathbf{R}_t^\top \boldsymbol{\theta}_t^{(\ell)} \right]
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\theta}_{f_i}^{(\ell)} \circ \sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)} + \sum_{\{c,t\}\in\binom{S_i}{2}}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\left(\boldsymbol{\gamma}_{f_i}^{(\ell)} - \sum_{t\in S_i}\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\right) \circ \sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)} + \sum_{\{c,t\}\in\binom{S_i}{2}}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)} \circ \sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)} - \sum_{t\in S_i}\sum_{c\in S_i}\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\circ\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)} + \sum_{\{c,t\}\in\binom{S_i}{2}}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)} \circ \sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)} - \sum_{t\in S_i}\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)} - \sum_{\{c,t\}\in\binom{S_i}{2}}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)} \circ \sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)} - \sum_{t\in S_i}\mathbf{R}_t^\top(\boldsymbol{\theta}_t^{(\ell)}\circ\boldsymbol{\theta}_t^{(\ell)}) - \sum_{\{c,t\}\in\binom{S_i}{2}}\mathbf{R}_c^\top\boldsymbol{\theta}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\theta}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)} \circ \sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)} - \sum_{t\in S_i}\mathbf{R}_t^\top(\boldsymbol{\gamma}_t^{(\ell)}\circ\boldsymbol{\gamma}_t^{(\ell)}) - \sum_{\{c,t\}\in\binom{S_i}{2}}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\gamma}_t^{(\ell)}\right].
\end{aligned}
$$

Next, we derive the partial derivative w.r.t. $\boldsymbol{\gamma}_{f_i}^{(\ell)}$ for a fixed $i\in[k]$, $\ell\in[r]$; in this computation we make use of (43) above:

$$
\begin{aligned}
\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}^{(\ell)}} &= \frac{1}{2}\frac{\partial\left\langle(\boldsymbol{\gamma}_{f_i}-\boldsymbol{\beta}_{f_i}),\mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i}-\boldsymbol{\beta}_{f_i})\right\rangle}{\partial\boldsymbol{\gamma}_{f_i}^{(\ell)}} + \frac{1}{2}\frac{\partial\left\|\boldsymbol{\gamma}_{f_i}^{(\ell)}-\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}^{(\ell)}}\\
&= \left(\sum_{c\in S_i}\mathrm{DIAG}(\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)})\right)\mathbf{B}_i^{-1}(\boldsymbol{\beta}_{f_i}-\boldsymbol{\gamma}_{f_i}) + \boldsymbol{\gamma}_{f_i}^{(\ell)} - \sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\\
&= \boldsymbol{\gamma}_{f_i}^{(\ell)} - \underbrace{\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}}_{\boldsymbol{\delta}_i^{(\ell)}} - \underbrace{\left(\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\right)}_{\boldsymbol{\delta}_i^{(\ell)}}\circ\mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i}-\boldsymbol{\beta}_{f_i})\\
&= \boldsymbol{\gamma}_{f_i}^{(\ell)} - \boldsymbol{\delta}_i^{(\ell)} - \boldsymbol{\delta}_i^{(\ell)}\circ\left(\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}}\right)
\end{aligned}
$$

Lastly, we move on to the partial derivative w.r.t. $\boldsymbol{\gamma}_w^{(\ell)}$ for a fixed $i\in[k]$, $w\in S_i$, $\ell\in[r]$:

$$
\begin{aligned}
\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_w^{(\ell)}} &= \frac{1}{2}\frac{\partial\left\|\boldsymbol{\gamma}_w^{(\ell)}\right\|_2^2}{\partial\boldsymbol{\gamma}_w^{(\ell)}} + \frac{1}{2}\frac{\partial\left\langle(\boldsymbol{\gamma}_{f_i}-\boldsymbol{\beta}_{f_i}),\mathbf{B}_i^{-1}(\boldsymbol{\gamma}_{f_i}-\boldsymbol{\beta}_{f_i})\right\rangle}{\partial\boldsymbol{\gamma}_w^{(\ell)}} + \frac{1}{2}\frac{\partial\left\|\boldsymbol{\gamma}_{f_i}^{(\ell)}-\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\right\|_2^2}{\partial\boldsymbol{\gamma}_w^{(\ell)}}\\
&= \boldsymbol{\gamma}_w^{(\ell)} + \mathbf{R}_w\left(\sum_{c\in F_i}\mathrm{DIAG}(\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)})\right)\mathbf{B}_i^{-1}(\boldsymbol{\beta}_{f_i}-\boldsymbol{\gamma}_{f_i}) + \mathbf{R}_w\left(\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)} - \boldsymbol{\gamma}_{f_i}^{(\ell)}\right)\\
&= \boldsymbol{\gamma}_w^{(\ell)} + \mathbf{R}_w\left(\boldsymbol{\gamma}_{f_i}^{(\ell)} + \boldsymbol{\delta}_i^{(\ell)}\right)\circ\left(\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}}\right) + \mathbf{R}_w\left(\boldsymbol{\delta}_i^{(\ell)} - \boldsymbol{\gamma}_{f_i}^{(\ell)}\right)\\
&= \boldsymbol{\gamma}_w^{(\ell)} + \mathbf{R}_w\left[\boldsymbol{\gamma}_{f_i}^{(\ell)}\circ\left(\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}}\right) + \left(\boldsymbol{\delta}_i^{(\ell)}\circ\left(\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}}\right) + \boldsymbol{\delta}_i^{(\ell)} - \boldsymbol{\gamma}_{f_i}^{(\ell)}\right)\right]\\
&= \boldsymbol{\gamma}_w^{(\ell)} + \mathbf{R}_w\left[\boldsymbol{\gamma}_{f_i}^{(\ell)}\circ\left(\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}}\right) - \left(\frac{1}{2}\frac{\partial\left\|\boldsymbol{\theta}\right\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}^{(\ell)}}\right)\right].
\end{aligned}
$$

In particular, we were able to reuse the computation of $\frac{1}{2}\frac{\partial\|\boldsymbol{\theta}\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}^{(\ell)}}$ and $\frac{1}{2}\frac{\partial\|\boldsymbol{\theta}\|_2^2}{\partial\boldsymbol{\gamma}_{f_i}}$ to compute $\frac{1}{2}\frac{\partial\|\boldsymbol{\theta}\|_2^2}{\partial\boldsymbol{\gamma}_w^{(\ell)}}$. There is, however, still one complicated term $\boldsymbol{\beta}_{f_i}$ left to compute. We simplify $\boldsymbol{\beta}_{f_i}$ to make its evaluation faster as follows.

$$
\begin{aligned}
\boldsymbol{\beta}_{f_i} &= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)}\circ\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)} - \sum_{t\in S_i}\mathbf{R}_t^\top(\boldsymbol{\gamma}_t^{(\ell)}\circ\boldsymbol{\gamma}_t^{(\ell)}) - \sum_{\{c,t\}\in\binom{S_i}{2}}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\gamma}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)}\circ\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)} - \frac{1}{2}\sum_{t\in S_i}\mathbf{R}_t^\top(\boldsymbol{\gamma}_t^{(\ell)}\circ\boldsymbol{\gamma}_t^{(\ell)}) - \frac{1}{2}\sum_{c\in S_i}\sum_{t\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\circ\mathbf{R}_t^\top\boldsymbol{\gamma}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)}\circ\underbrace{\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}}_{\boldsymbol{\delta}_i^{(\ell)}} - \frac{1}{2}\sum_{t\in S_i}\mathbf{R}_t^\top(\boldsymbol{\gamma}_t^{(\ell)}\circ\boldsymbol{\gamma}_t^{(\ell)}) - \frac{1}{2}\sum_{c\in S_i}\mathbf{R}_c^\top\boldsymbol{\gamma}_c^{(\ell)}\circ\sum_{t\in S_i}\mathbf{R}_t^\top\boldsymbol{\gamma}_t^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\boldsymbol{\gamma}_{f_i}^{(\ell)}\circ\boldsymbol{\delta}_i^{(\ell)} - \frac{1}{2}\sum_{t\in S_i}\mathbf{R}_t^\top(\boldsymbol{\gamma}_t^{(\ell)}\circ\boldsymbol{\gamma}_t^{(\ell)}) - \frac{1}{2}\boldsymbol{\delta}_i^{(\ell)}\circ\boldsymbol{\delta}_i^{(\ell)}\right]\\
&= \sum_{\ell=1}^{r}\left[\left(\boldsymbol{\gamma}_{f_i}^{(\ell)} - \frac{1}{2}\boldsymbol{\delta}_i^{(\ell)}\right)\circ\boldsymbol{\delta}_i^{(\ell)} - \frac{1}{2}\sum_{t\in S_i}\mathbf{R}_t^\top(\boldsymbol{\gamma}_t^{(\ell)}\circ\boldsymbol{\gamma}_t^{(\ell)})\right].
\end{aligned}
$$

This completes the proof. □

D. DIAGRAMS

D.1 In-database vs. Out-of-database learning: High-level diagram

Figure 2 depicts the high-level differences between a typical out-of-database learning setting and our in-database learning approach.

- In out-of-database learning (the red part of the diagram), a feature extraction query is issued to the database engine which computes, materializes and exports the output table. Depending on the chosen data model, the output table is then fed into an appropriate machine learning tool that learns and outputs the best model parameters $\boldsymbol{\theta}^*$.
- In our in-database learning framework (the **green** part), we have a unified model formulation that subsumes many commonly-used models: Choosing a specific model reduces mainly to choosing a function h of the features and a function g of the model parameters. The function h, the feature extraction query, and the input tables are all fed into a factorization-based query optimizer, which computes a relatively small "digest" in the form of a matrix $\boldsymbol{\Sigma}$ and a vector $\boldsymbol{c}$. Only this small digest along with the function g are needed by the gradient-descent trainer to carry out all iterations of gradient-descent and compute $\boldsymbol{\theta}^*$, without ever having to go back to the original data tables.

D.2 In-database vs. Out-of-database learning: Low-level diagram

Figure 3 reveals more details about some of the key ideas behind the performance improvements of our framework over out-of-database learning.

- In out-of-database learning, the database engine takes the input tables (of size $\leq N$) and joins them into a potentially much larger output table of size $|D|$, which might in turn get blown up even more inside the machine learning tool!
- In our framework, the input query, the data tables, and function h are first translated into a language that is suitable for aggregate query specification and optimization, which is the language of FAQ [6]. In particular, each entry $\boldsymbol{\sigma}_{i,j}$ (and $\boldsymbol{c}_j$) of our target matrix $\boldsymbol{\Sigma}$ (and vector $\boldsymbol{c}$) is expressed as the answer to an FAQ query $\varphi^{(i,j)}$ (or $\varphi^{(j)}$). All those queries are fed into an FAQ query optimizer. The optimizer factorizes each query $\varphi^{(i,j)}$ into small sub-queries $\varphi_a^{(i,j)}, \varphi_b^{(i,j)}, \varphi_c^{(i,j)}, \ldots$ and solves them individually. Each sub-query results in a table of size $\leq N^{\mathsf{faqw}}$, which can be much smaller than the size of the output table D. By solving the FAQ queries $\varphi^{(i,j)}$, we obtain $\boldsymbol{\Sigma}$ and $\boldsymbol{c}$, which are all that is needed for gradient-descent... For more details, see Section 3.

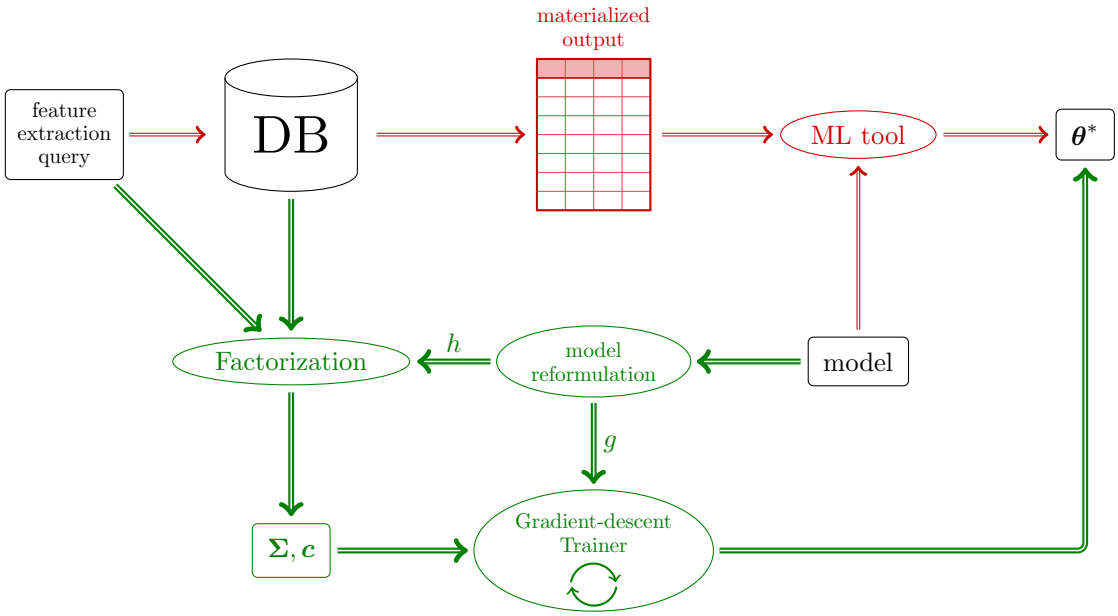


Figure 2: **In-database** vs. Out-of-database learning: High-level diagram. (See Section D.1.)

D.3 FD-aware in-database learning

Figure 4 depicts (in **blue**) the enhancements that we introduce to our framework in order to take advantage of FDs in the input database instance and reduce our previous runtime even further (but still compute *the same* $\boldsymbol{\theta}^*$ as before).

- As explained earlier, computing each entry of the matrix $\boldsymbol{\Sigma}$ and of the vector $\boldsymbol{c}$ requires solving an FAQ query. However, by utilizing functional dependencies we can filter out many of those entries as unneeded for later stages, thus significantly reducing the number of FAQ queries that we have to solve. After the filtering process, $\boldsymbol{\Sigma}$ and $\boldsymbol{c}$ shrink down to $\overline{\boldsymbol{\Sigma}}$ and $\overline{\boldsymbol{c}}$, which we compute and feed to gradient-descent (GD). We also filter the function g down to $\bar{g}$ and feed the latter to GD. Now, we run GD in the space of $\boldsymbol{\gamma}$ (instead of the original higher-dimensional space of $\boldsymbol{\theta}$). During each iteration of GD, in order to compute the objective function $\overline{J}(\boldsymbol{\gamma})$ and its gradient $\boldsymbol{\nabla}\overline{J}(\boldsymbol{\gamma})$, we need to use the matrices $\mathbf{R}$ that represent the functional dependencies. And after GD finishes, we have to convert the resulting optimal solution $\boldsymbol{\gamma}^*$ back into the original space to get $\boldsymbol{\theta}^*$. Such conversion also requires the FD-matrices $\mathbf{R}$. For more details, see Section 4.

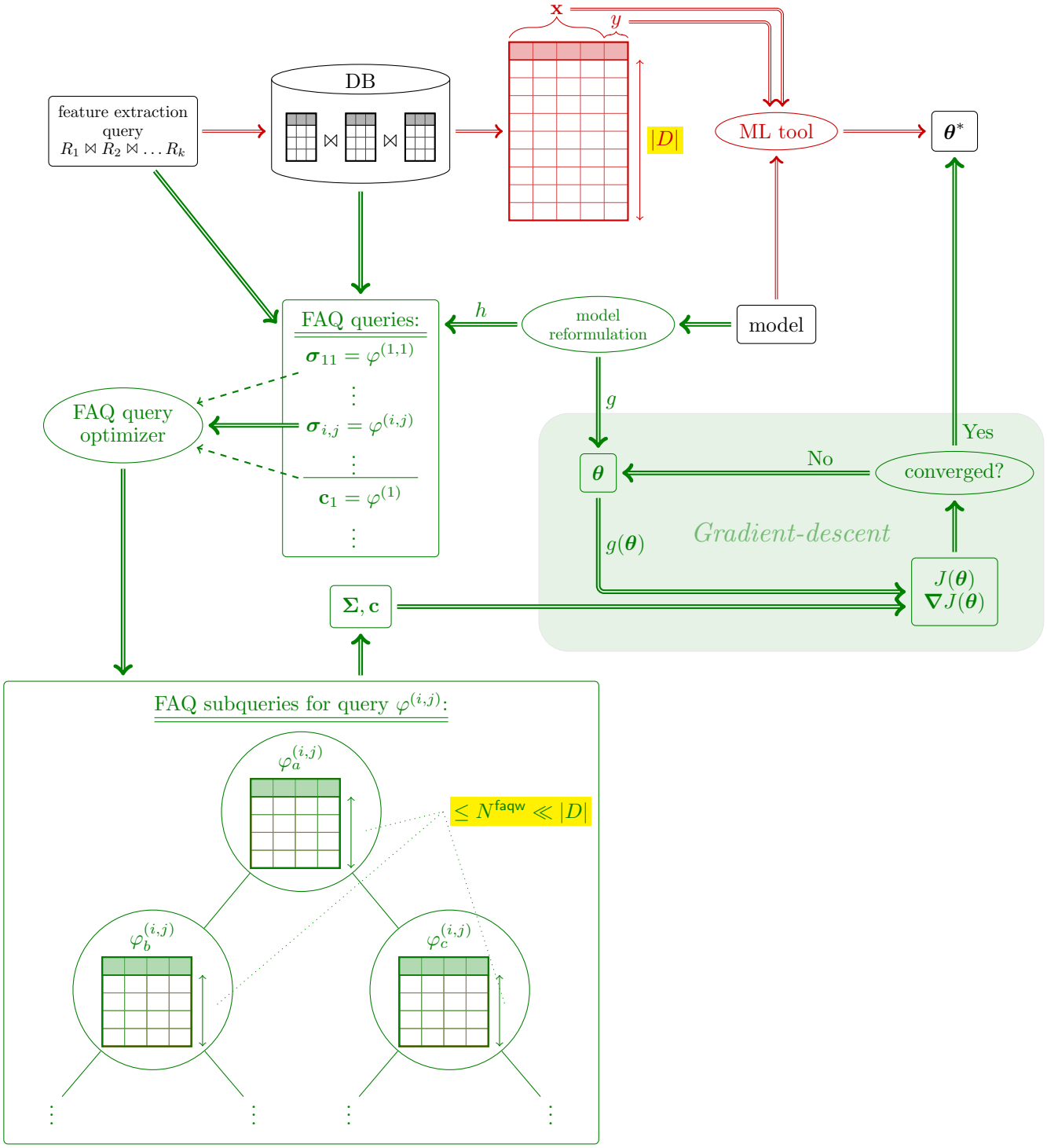


Figure 3: **In-database** vs. Out-of-database learning: Low-level diagram. (See Section D.2.)

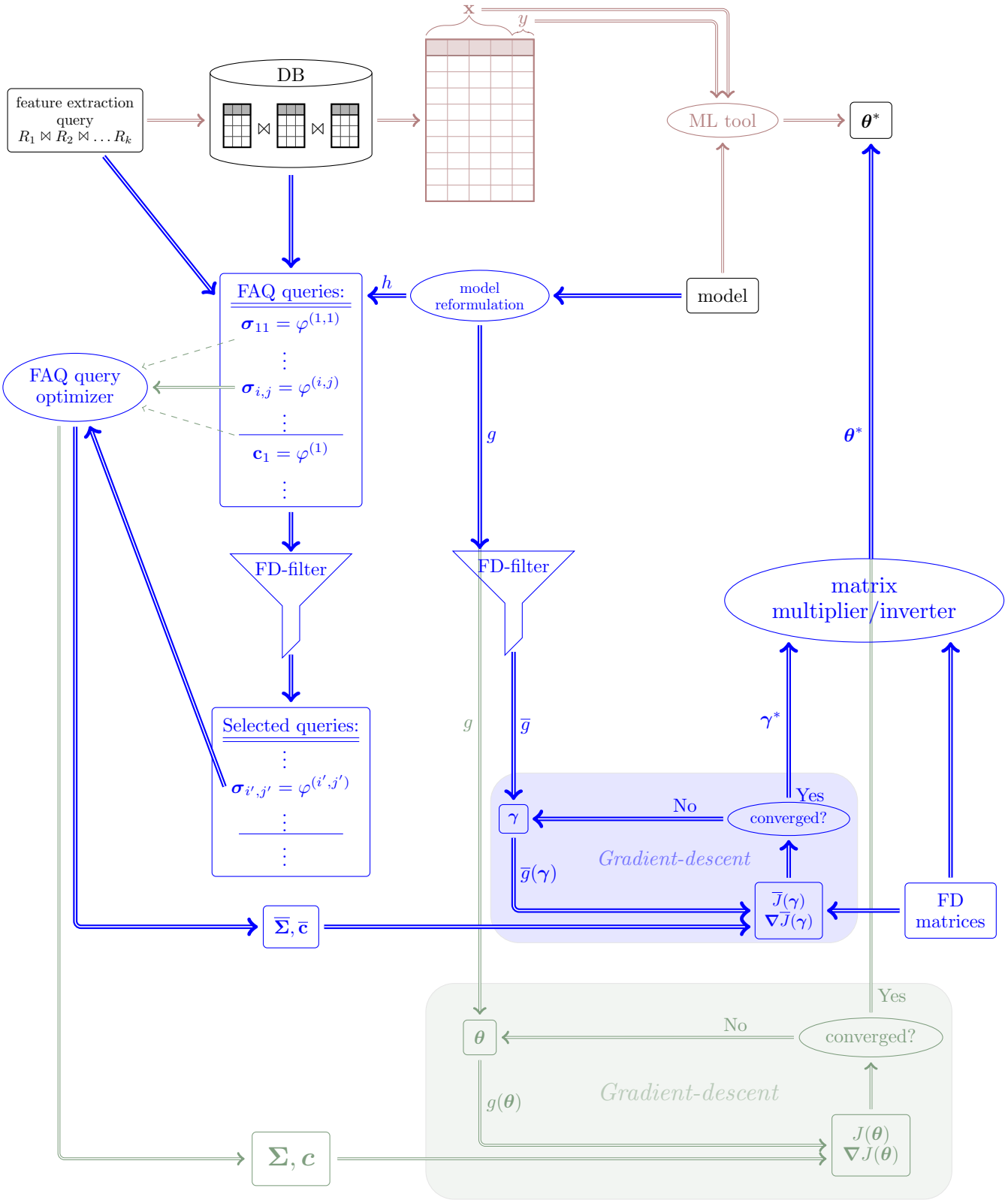


Figure 4: **FD-aware** in-database learning (vs. in-database learning without FD). (See Section D.3).