A Relational Gradient Descent Algorithm For Support Vector Machine Training

Mahmoud Abo-Khamis
Relational AI
mahmoud.abokhamis@relational.ai

Sungjin Im*
University of California, Merced
sim3@ucmerced.edu

Benjamin Moseley†
Carnegie Mellon University
mosleyb@andrew.cmu.edu

Kirk Pruhs‡
University of Pittsburgh
kirk@cs.pitt.edu

Alireza Samadian
University of Pittsburgh
samadian@cs.pitt.edu

Abstract

We consider gradient descent like algorithms for Support Vector Machine (SVM) training when the data is in relational form. For relational data the gradient of the SVM objective cannot be efficiently computed by known techniques as it suffers from the "subtraction problem". We first show that the subtraction problem cannot be surmounted by showing that computing any constant approximation of the gradient of the SVM objective function is $\#P$-hard, even for acyclic joins. However, we circumvent the subtraction problem by restricting our attention to stable instances, which intuitively are instances where a nearly optimal solution remains nearly optimal if the points are perturbed slightly. We give an efficient algorithm that computes a "pseudo-gradient" that guarantees convergence for stable instances at a rate comparable to that achieved by using the actual gradient. We believe that our results suggest that this sort of stability analysis would likely yield useful insight in the context of designing algorithms on relational data for other learning problems in which the subtraction problem arises.

1 Introduction

Many learning tasks faced by data scientists involve relational data that is stored in tables in a relational database. However, almost all standard algorithms for standard learning problems assume that the input consists of points in Euclidean space [20], and thus these algorithms are not designed to operate directly on relational data. The current standard practice for a data scientist, confronted with a learning task on relational data, is to first issue a feature extraction query to extract the data from the relational database by joining together the tables to materialize a design matrix $J = T_1 \bowtie \cdots \bowtie T_m$ where $T_1, \ldots, T_m$ are the input tables and $\bowtie$ is the natural inner join operator. Then (after converting any non-numerical data to numerical data) importing this design matrix J into a standard learning package to train the model. So conceptually in this standard practice, the learning package can be viewed as sitting on a layer above the database in the computing stack.

This standard practice is wasteful for several reasons, including: (1) computing relational joins are computationally expensive, both in terms of time and space, (2) the resulting design matrix will likely have significant redundancy and be much larger than the aggregate sizes of the tables, and (3) this redundancy in the design matrix will likely mean that the machine learning algorithm takes more time than should conceptually be necessary. More formally, if each table has n rows, the design matrix J can have as many as n^m entries. Thus, independent of the learning task, this standard practice necessarily has exponential worst-case time and space complexity as the design matrix can be exponentially larger than the underlying relational tables.

Thus there are several companies that are developing products, notably RelationalAI [2] and Google's BigQuery ML [1], that integrate machine learning di-

*Supported in part by NSF grants CCF-1409130, CCF-1617653, and CCF-1844939.

†Supported in part by NSF grants CCF-1725543, 1733873, 1845146, a Google Research Award, a Bosch junior faculty chair and an Infor faculty award.

‡Supported in part by NSF grants CCF-1421508, CCF-1535755, CCF-1907673, CCF-2036077 and an IBM Faculty Award.

rectly in the database. The ideal situation would be to design what we call relational algorithms for the standard learning tasks. Informally relational algorithms operate directly on the tables, without joining the tables, and are efficient (ideally nearly linear time, or at worst lower order polynomial time) in terms of the aggregate sizes of the tables (independent of the size of the design matrix). In this context, a natural research question is to determine which natural problems, and in particular which standard learning problems, admit relational algorithms.

This paper is a followup to the paper [27], which considered algorithmic problems that can be reduced to evaluating an FAQ-AI(1) query, which is Sum-Product formula over the rows in the design matrix that satisfy some additive inequality. Example problems that are related to the classic Support Vector Machines (SVM) learning problem and that can reduce to evaluating an FAQ-AI(1) query include:

1. Counting the number of points correctly (or incorrectly) classified by a linear classifier (hyperplane).
2. Finding the minimum distance of a correctly classified point to the boundary of a given linear classifier (hyperplane).
3. Computing the gradient of the SVM objective function at a particular point.

[27] showed how to develop pseudo-polynomial time algorithms for FAQ-AI(1) queries by reducing the evaluation of such queries to evaluating an unconstrained Sum-Product query over a dynamic programming semiring, which is a semiring design to implement a particular dynamic program. [27] then showed that if the original sum and product operators were approximation preserving then one could apply standard sketching techniques to obtain a Relational Approximation Scheme (RAS), which is a collection $\{A_\epsilon\}$ of relational algorithms where A_ϵ achieves $(1+\epsilon)$-approximation. Roughly speaking an operation $\oplus$ is approximation preserving if the following holds: if $\hat{x}$ is a good approximation of x and $\hat{y}$ is a good approximation of y then $\hat{x} \oplus \hat{y}$ is good approximation of $x \oplus y$. The common operators addition, multiplication, minimum and maximum are generally approximation preserving. Thus, the techniques from [27] yield a RAS for the first example problem above, namely determining the number of points classified correctly by a linear classifier. The most common example of an operator that is not approximation preserving is subtraction. If $\hat{x}$ is a good approximation of x and $\hat{y}$ is a good approximation of y then $\hat{x}-\hat{y}$ is not necessarily a good approximation of $x-y$ because if the values of x and y are close we are not even able to determine the sign of $x-y$ from $\hat{x}-\hat{y}$. Because of this, the techniques from [27] do not yield a RAS for the second and third problems above, namely finding the closest point to a hyperplane and computing the gradient of the SVM objective function. [27] called this the subtraction problem.

A natural research question is whether the subtraction problem can be surmounted or circumvented to obtain some sort of positive algorithmic result in problems where the subtraction problem seems inherent. Intuitively if $\hat{x}$ and $\hat{y}$ are approximately equal, then although one cannot learn a good relative approximation of $x-y$, one can know that x and y are approximately equal, which may have sufficient utility for some problems. As a test case, we consider the classic learning problem of (soft-margin linear) Support Vector Machine (SVM) training, and solving this problem using the classic gradient descent approach. SVM is identified as one of the five most important learning problems in [20], and is covered in almost all introductory machine learning textbooks. Gradient descent is probably the most commonly used computational technique for solving convex learning optimization problems [42].

1.1 Background

Before stating our results, we cover the minimal background that is required to understand our results, namely (1) gradient descent, (2) SVM training, (3) gradient descent for SVM, (4) relational data and relational algorithms, and (5) how the subtraction problem arises within the context of computing the gradient of the SVM objective function.

1.1.1 Gradient Descent

Gradient descent is a first-order iterative optimization method for finding an approximate minimum of a convex function $F : \mathbb{R}^d \to \mathbb{R}$, perhaps subject to a constraint that the solution lies in some convex body $\mathcal{K}$. In the gradient descent algorithm, at each descent step t the current candidate solution $\beta^{(t)}$ is updated according to the following rule:

$$\beta^{(t)} \leftarrow \beta^{(t-1)} - \eta_t G(\beta^{(t-1)}) \tag{1.1}$$

where η_t is the step size. In projected gradient descent, the current candidate solution $\beta^{(t)}$ is updated according to the following rule:

$$\beta^{(t)} \leftarrow \Pi_{\mathcal{K}}\left(\beta^{(t-1)} - \eta_t G(\beta^{(t-1)})\right) \tag{1.2}$$

where $\Pi_{\mathcal{K}}(\alpha) = \text{argmin}_{\beta \in \mathcal{K}} \|\alpha - \beta\|_2$ is the projection of the point α to the closest point to α in $\mathcal{K}$. In (projected) gradient descent, G is $\nabla F(\beta^{(t)})$, the gradient of F at $\beta^{(t)}$. There are lots of variations of gradient descent, including variations on the step size, and variations, like stochastic gradient descent[42], in which the gradient is only approximated for a uniform sample of the data.

1.1.2 SVM Training Conceptually, the input to SVM training consists of a collection $X = \{x_1, x_2, \dots, x_N\}$ of points in $\mathbb{R}^d$, and a collection $Y = \{y_1, y_2, \dots, y_N\}$ of associated labels from $\{-1, 1\}$. For convenience let us rescale the points so that each point in X lies within the hypercube $[-1, 1]^d$. A feasible solution is a d-dimensional vector β, sometimes called a hypothesis. The objective is to minimize a linear combination $F(\beta, X, Y)$ of the average "hinge" loss function of the points $L(\beta, X, Y) = \frac{1}{N}\sum_{x_i \in X} \max(0, 1 - y_i\beta \cdot x_i)$ and a regularizer $R(\beta)$. We will take the regularizer to be the 2-norm squared of β, as that is a standard choice [20], although this choice is not so important for our purposes. Thus, the objective is to minimize:

(1.3)
$$F(\beta, X, Y) = \frac{1}{N}\sum_{x_i \in X} \max(0, 1 - y_i\beta \cdot x_i) + \lambda||\beta||_2^2$$

Here the loss function measures how well the hypothesis β explains the labels, and one of the regularizer's purposes is to prevent overfitting. The λ factor intuitively specifies the amount that the loss has to decrease to justify an increase in the norm of β. When either X and Y are understood, for notational convenience, we may drop them from the objective.

1.1.3 Gradient Descent for SVM In Section A, we show that by a straightforward specialization of a standard convergence analysis for projected gradient descent to SVM one obtains Theorem 1.1, which bounds the number of descent steps needed to reach a solution with a specified relative error.

THEOREM 1.1. *Let $F(\beta)$ be the SVM objective function. Let $\beta^* = \text{argmin}_\beta F(\beta)$ be the optimal solution. Let $\widehat{\beta}_s = \frac{1}{s}\sum_{t=0}^{s-1}\beta^{(t)}$. Let $\eta_t = \frac{1}{8\lambda\sqrt{dt}}$. Then if $T \geq \left(\frac{4d^{3/2}}{\epsilon\lambda F(\widehat{\beta}_T)}\right)^2$ then projected gradient descent guarantees that*

$$F(\widehat{\beta}_T) \leq (1+\epsilon)F(\beta^*)$$

Thus, if the algorithm returns $\widehat{\beta}$ at the first time t where $t \geq \left(\frac{4d^{3/2}}{\epsilon\lambda F(\widehat{\beta}_t)}\right)^2$, then it achieves relative error at most ϵ.

1.1.4 Relational Data and Relational Algorithms One immediate difficulty that we run into is that if the tables have a sufficiently complicated structure, almost all natural problems/questions about the design matrix are NP-hard. For example, it is NP-hard to even determine whether or not the design matrix is empty (see for example [24, 35]). Thus, as we want to focus on the complexity of the algorithm (learning) problems, we conceptually want to abstract out the complexity of the join structure. The simplest way to accomplish this is to primarily focus on instances where the structure of the tables is simple, with the most natural candidate for "simplicity" being that the join is acyclic. You may find the definition of joins and acyclicity in Appendix B. Acyclic joins are the norm in practice and are a commonly considered special case in the database literature. For example, there are efficient algorithms to compute the size of the design matrix for acyclic joins.

Formally defining what a "relational" algorithm is problematic, as for each natural candidate definition there are plausible scenarios in which that candidate definition is not the "right" definition. But in this paper it is sufficient to think of a "relational" algorithm as one whose runtime is polynomially bounded in the aggregate sizes of the tables if the join is acyclic.

1.1.5 SVM Gradient and the Subtraction Problem The gradient of the SVM objective function F is

$$\nabla F = 2\lambda\beta - \frac{1}{N}\sum_{i \in \mathcal{L}} y_i x_i \tag{1.4}$$

where $\mathcal{L}$ is the collection of indices i satisfying the additive inequality $y_i\beta \cdot x_i \leq 1$. Note the term $2\lambda\beta$ term is trivial to compute, so let us focus on the term $G = \frac{1}{N}\sum_{i \in \mathcal{L}} y_i x_i$. Now let us focus on a particular dimension, and use x_{ik} to refer to the value of point x_i in dimension k. Let $L_k^- = \{i \mid i \in \mathcal{L} \text{ and } y_i x_{ik} < 0\}$ denote those points that are in $\mathcal{L}$ and whose the gradient in the k^{th} coordinate has negative sign. Conceptually each point in L_k^- pushes the gradient in dimension k up with "force" proportional to its value in dimension k. Let $L_k^+ = \{i \mid x_i \in \mathcal{L} \text{ and } y_i x_{ik} > 0\}$ denote those points that are in $\mathcal{L}$ and whose the gradient in the k^{th} coordinate has positive sign. And conceptually each point in L_k^+ pushes the gradient in dimension k down with "force" proportional to its value in dimension k.

The results in [27] can be applied to obtain a RAS to compute a $(1+\epsilon)$ approximation $\widehat{G}_k^+$ to $G_k^+ = \frac{1}{N}\sum_{i \in L_k^+} y_i x_{ik}$, and a RAS to compute a $(1+\epsilon)$ approximation $\widehat{G}_k^-$ to $G_k^- = \frac{1}{N}\sum_{i \in L_k^-} y_i x_{ik}$. However, the results in [27] cannot be applied to get a RAS for computing a $(1+\epsilon)$-approximation to $G = G_k^- + G_k^+$, because it suffers from the *subtraction problem* as G_k^- and G_k^+ will have different signs.

1.2 Our Results

We start with a rather discouraging negative result that shows we cannot surmount the subtraction problem in the context of computing the gradient of the SVM objective problem. In particular, we show in Section 2 that computing an $O(1)$ approximation to the partial derivative in a specified dimension is $\#P$-hard, even for acyclic joins. Thus we need to try to circumvent (not surmount) the subtraction problem. After some reflection, one reasonable interpretation of our $\#P$-hardness proof is that it shows that computing the gradient is hard on unstable instances. In this context, intuitively an instance is stable if a nearly optimal solution remains nearly optimal if the points are perturbed slightly. Intuitively one would expect real-world instances, where there is a hypothesis β that explains the labels reasonably well, to be relatively stable (some discussion of the stability of SVM instances can be found in [17]). And for instances where there isn't a hypothesis that explains the labels reasonably well, it probably doesn't matter what hypothesis the algorithm returns, as it will likely be discarded by the data scientist anyways.

Long story short, the main result of this paper is a relational algorithm that computes a "pseudo-gradient" that guarantees convergence for stable instances at a rate comparable to the standard bound on the convergence rate when using the real gradient. The algorithm design can be found in Section 3, and the algorithm analysis can be found in Section 4. Postponing for the moment our formal definition of stability, we state our main result in Theorem 1.2. The reader should compare Theorem 1.2 to the analysis of gradient descent in Theorem 1.1.

Theorem 1.2. *Let X be an (α, δ, γ)-stable SVM instance, as defined in Definition 4.3, formed by an acyclic join. Let n be the maximum number of rows in any table, d be the aggregate number of features over all the tables, and m be the number of tables. Let $\beta^* = \operatorname{argmin}_\beta F(\beta)$ be the optimal solution. Then there is a relational algorithm that can compute a pseudo-gradient in time $O(\frac{m}{\epsilon^2}(m^3 \log^2(n))^2(d^2 mn \log(n)))$, where $\epsilon = \min(\frac{\delta}{8}, \alpha)$. After $T = \left(\frac{256 d^{3/2}}{\lambda\delta F(\beta_a, X_a)}\right)^2$ iterations of projected descent using this pseudo-gradient there is a relational algorithm that can compute in time $O(\frac{1}{\epsilon^2}(m^3 \log^2(n))^2(d^2 mn \log(n)))$ a hypothesis $\widehat{\beta}$ such that:*

$$F(\widehat{\beta}, X) \leq (1+\gamma)F(\beta^*, X)$$

Note that most commonly n is the dominant parameter as usually $n \gg d$ and $n \gg m$, and our algorithm referenced in Theorem 1.2 computes the pseudo-gradient in time nearly linear in the dominant parameter. One reasonable interpretation of Theorem 1.2 is that it suggests that this sort of stability analysis might be a useful tool for other algorithmic learning problems on relational data that suffer from the subtraction problem.

To quote from the stability analysis section of Tim Roughgarden's CACM survey article [39] on "beyond worst-case analysis":

> The unreasonable effectiveness of modern machine learning algorithms has thrown down the gauntlet to algorithms researchers, and there is perhaps no other problem domain with a more urgent need for the beyond worst-case approach.

To date, the sort of stability analysis we use in this paper has primarily been applied to clustering problems [39]. So to mix metaphors, we believe that at least to some extent, we have picked up Roughgarden's gauntlet and moved the ball forward a bit with respect to beyond worst-case analysis of learning algorithms.

1.3 Related Results Relational algorithms are known for certain types of Sum of Sums (SumSum) and Sum of Products (SumProd) queries. In particular, the Inside-Out algorithm [7] can evaluate a SumProd query in time $O(md^2n^h \log n)$, where m is the number of tables, d is the number of columns, and h is the fractional hypertree width [25] of the query. Note that $h = 1$ for the acyclic joins, and thus Inside-Out is a polynomial-time algorithm for acyclic joins. One can reduce SumSum queries to m SumProd queries [3], and thus they be solved in time $O(m^2d^2n^h \log n)$. The Inside-Out algorithm builds on several earlier papers, including [10, 23, 29, 25].

SumSum and SumProd queries with additive inequalities were first studied in [3]. [3] gave an algorithm with worst-case time complexity $O(md^2n^{m/2} \log n)$. So this is better than the standard practice of forming the design matrix, which has worst-case time complexity $\Omega(dn^m)$. Different flavors of queries with inequalities were also studied [28, 30, 6]. [27] showed that computing even very simple types of SumSum and SumProd queries with a single inequality is NP-hard. But an RAS for special types queries is introduced in [27]. The algorithm in [27] can obtain $(1+\epsilon)$ approximation for problems such as counting the number of rows on one side of a hyperplane in time $O(\frac{1}{\epsilon^2}(m^3 \log^2(n))^2(d^2mn^h \log(n)))$.

Algorithms for linear/polynomial regression on relational data are given in [40, 4, 5, 32, 33] and an algorithm for k-means clustering on relational data is given in [21].

Stability analysis, similar in spirit to our results, has been consider before in clustering problems [39, 18, 34, 8, 14, 22, 31, 38, 11, 13, 15]. For example, the NP-hard k-means, k-medians and k-centers clustering problems are polynomially solvable for instances in which changing the distances of the points by a multiplicative factor of at most 2 does not change the optimal solution [11, 13, 15]

SVM is covered in almost every introductory machine learning textbook, for example [20]. Optimization methods for learning problems, including variations of gradient descent, are discussed in [42]. One common variation of gradient descent is stochastic gradient descent, in which the gradient is computed on a random sample of the points. [19, 41] consider stochastic gradient descent in the context of linear SVM training, and empirically showed that stochastic gradient descent can improve the optimization speed compared to batch gradient descent in many situations. Still stochastic gradient descent needs multiple iterations over the whole data in order to achieve a meaningful guarantee, and thus necessarily would have worst-case time complexity exponential in the size of the relational data. A overview of the type of online convex optimization that we apply can be found in [16, 26].

2 Hardness of Gradient Approximation

LEMMA 2.1. *It is $\#P$ hard to $O(1)$-approximate the partial derivative of the SVM objective function in a specified dimension.*

Proof. We reduce the decision version of the counting knapsack problem to the problem of approximating the gradient of SVM. The input to the decision counting knapsack problem is a set of weights $W = \{w_1, w_2, \dots, w_m\}$, a knapsack size L, and an integer k. The output of the problem is whether there are k different combinations of the items that fit into the knapsack.

We create $m+1$ tables, each with two columns. The columns of the first m tables are (Key, E_i) for T_i and the rows are

$$T_i = \{(1,0),(1,w_i/L),(0,0)\}.$$

The last table has two columns $(\text{Key}, \text{Value})$, and it has two rows $(1,1),(0,-k)$. Note that if we take the join of these tables, there will be $m+2$ columns: $(\text{Key}, \text{Value}, E_1, E_2, \dots, E_m)$.

Let $\beta = (0,0,1,1,\dots,1)$ and $\lambda = 0$, so β is 0 on the columns Key and Value and 1 everywhere else. Then we claim, if the gradient of F on the second dimension (Value) is non-negative, then the answer to the original counting knapsack is true, otherwise, it is false.

To see the reason, consider the rows in J: there are 2^m rows in the design matrix that have $(1,1)$ in the first two dimensions and all possible combinations of the knapsack items in the other dimensions. More precisely, the concatenation of $(1,1)$ and w_S for every $S \in [m]$ where w_S is the vector that has w_i/L in the i-th entry if item i is in S or 0 otherwise. Further, J has a single special row with values $(0,-k,0,0,\dots,0)$. Letting G_2 be the second dimension of the gradient of SVM (the column Value), we have,

$$G_2 = \sum_{x \in J : 1-\beta\cdot x \geq 0} x_2$$

For the row with Key $= 1$ for each $S \in [m]$, we have $1-\beta\cdot x = 1-\sum_{i\in S} w_i/L \geq 0$ if and only if the items in S fits into the knapsack and $x_2 = 1$. For the single row with Key $= 0$, we have $1-\beta\cdot x = 1$, and its value on the second dimension is $x_2 = k$. Therefore,

$$G_2 = C_L(w_1,\dots,w_m) - k$$

where C_L is the number of subsets of items fitting into the knapsack of size L. This means if we could approximate the gradient up to any constant factor, we would be able to determine if G_2 is positive or negative, and as a result we would be able to answer the (decision version of) counting knapsack problem, which is $\#P$-hard. $\square$

3 Algorithm Design

3.1 Review of Row Counting with a Single Additive Constraint

We now summarize algorithmic results from [27] for two different problems, that we will use as black boxes.

In the first problem the input is a collection $T_1,\dots,T_m$ of tables, a label $\ell \in \{-1,+1\}$, and an additive inequality $\mathcal{L}$ of the form $\sum_{j\in[d]} g_j(x_j) \geq R$, where each function g_j can be computed in constant time. The output consists of, for each $j \in [d]$ and $v \in D(j)$, where $D(j)$ is the domain of column/feature j, the number $C^\ell_{j,v}$ of rows in the design matrix $J = T_1 \bowtie \dots \bowtie T_m$ that satisfy constraint $\mathcal{L}$, that have label ℓ, and that have value v in column j. [27] gives a relational algorithm, which we will call the Row Counting Algorithm, that computes a $(1+\epsilon)$-approximation for each such $\widehat{C}^\ell_{j,v}$ to each $C^\ell_{j,v}$, and that runs in time $O(\frac{m}{\epsilon^2}(m^3\log^2(n))^2(d^2mn^h\log(n)))$

In the second problem the input is a collection $T_1,\dots,T_m$ of tables, a label $\ell \in \{-1,+1\}$, and an expression of the form of $\sum_{j\in[d]} g_j(x_j)$, where the g_j functions can be computed in constant time. The output consists of, for each $k \in [0, \log_{1+\epsilon} N]$, the maximum value of H_k such that the number of points in the design matrix $J = T_1 \bowtie \dots \bowtie T_m$ with label $\ell \in \{-1,1\}$ satisfying the additive inequality $\sum_{j\in[d]} g_j(x_j) \geq H_k$ is at least $\lfloor(1+\epsilon)^k\rfloor$. [27] gives an algorithm for this problem, which we will call the Generalized Row Counting Algorithm, that runs in time $O(\frac{1}{\epsilon^2}(m^3 \log^2(n))^2(d^2mn^h \log(n)))$. Using the result of the algorithm, for any scalar distance H, it is possible to obtain a row count $\hat{N}(H)$ such that $N(H)/(1+\epsilon) \leq \hat{N}(H) \leq N(H)$, where $N(H)$ is the number of points in the design matrix with label ℓ satisfying the inequality $\sum_{j\in[d]} g_j(x_j) \geq H_k$.

3.2 Overview of Our Approach Recall from the introduction that the difficulty arises when a $\widehat{G}_k^+$ is approximately equal to $-\widehat{G}_k^-$. In this case, it would seem that by appropriately perturbing one of L_1^- or L_1^+ by a relatively small amount one could force $G = \widehat{G}^- + \widehat{G}^+$ for this perturbed instance. In which case, if we used $2\lambda\beta^{(t)} + (\widehat{G}^- + \widehat{G}^+)$ as the pseudo-gradient, then it would be the true gradient for a slightly perturbed instance. However, this isn't quite right, as there is an additional issue. If we perturb a point x_i, then the sign of $1 - y_i\beta \cdot x_i$ may change, which means this point's contribution to the gradient may discontinuously switch between 0 and $-y_ix_i$. To address this issue, when computing the pseudo-gradient, we use a new instance X' that excludes points that are "close" to the separating hyperplane $1 - y_i\beta \cdot x_i = 0$. That is, X' excludes every point that can change sides of the hyperplane in an ϵ-perturbation of each coordinate. This will allow us to formally conclude that if we used $2\lambda\beta^{(t)} + (\widehat{G}^- + \widehat{G}^+)$, where $\widehat{G}^-$ and $\widehat{G}^+$ are defined on X', as the pseudo-gradient, then it would be the true gradient for a slightly perturbed instance. After the last descent step, we choose the final hypothesis to be the ϵ-perturbation of any computed hypothesis $\beta^{(t)}$, $t \in [0, T]$ that minimizes the SVM objective.

In the analysis we interpret the sequence $\beta^{(0)}, \beta^{(1)}, \ldots, \beta^{(T)}$ as solving an online convex optimization problem, and apply known techniques from this area.

3.3 Pseudo-gradient Descent Algorithm Firstly, in linear time it is straight-forward to determine if the points in X lie in $[-1, 1]^d$, and if not, to rescale so that they do; this can be accomplished by, for each feature, dividing all the values of that feature in all of the input tables by the maximum absolute value of that feature. The initial hypothesis $\beta^{(0)}$ is the origin. For any vector v, let $u = |v|$ be a vector such that its entries are the absolute values of v, meaning for all j we have that $u_j = |v_j|$.

Algorithm to Compute the Pseudo-gradient:

1. Run the Row Counting Algorithm to compute, for each $j \in [d]$ and $v \in D(j)$, a $(1+\epsilon)$ approximation $\widehat{C}_{j,v}^-$ to $C_{j,v}^-$, which is the number of rows $x \in J$ with negative label and $x_j = v$, satisfying the additive inequality $1 + \beta^{(t)} \cdot x \geq \epsilon \left|\beta^{(t)}\right| \cdot |x|$.

2. Run the Row Counting Algorithm to compute, for each $j \in [d]$ and $v \in D(j)$, a $(1+\epsilon)$ approximation $\widehat{C}_{j,v}^+$ to $C_{j,v}^+$, which is the number of rows in $x \in J$ with positive label and $x_j = v$, satisfying the additive inequality $1 - \beta^{(t)} \cdot x \geq \epsilon \left|\beta^{(t)}\right| \cdot |x|$.

3. For all $k \in [d]$, compute $\widehat{G}_k^- = \sum_{v\in D(k):v<0} v\, \widehat{C}_{k,v}^- - \sum_{v\in D(k):v\geq 0} v\, \widehat{C}_{k,v}^+$.

4. For all $k \in [d]$, compute $\widehat{G}_k^+ = \sum_{v\in D(k):v\geq 0} v\, \widehat{C}_{k,v}^- - \sum_{v\in D(k):v<0} v\, \widehat{C}_{k,v}^+$.

5. The pseudo-gradient is then
$$\widehat{G} = \frac{\widehat{G}^- + \widehat{G}^+}{N} + \lambda\beta^{(t)}$$

Algorithm for a Single Descent Step: The next hypothesis $\beta^{(t+1)}$ is
$$\beta^{(t+1)} = \Pi_{\mathcal{K}}(\beta^{(t)} - \eta_{t+1}\widehat{G})$$

Here $\eta_t = \frac{1}{\lambda\sqrt{dt}}$ and $\Pi_{\mathcal{K}}(\beta)$ is the projection of β onto a hypersphere $\mathcal{K}$ centered at the origin with radius $\frac{\sqrt{d}}{2\lambda}$. Note that $\Pi_{\mathcal{K}}(\beta)$ is β if $\|\beta\|_2 \leq \frac{\sqrt{d}}{2\lambda}$ and $\frac{\sqrt{d}}{2\lambda\|\beta\|_2}\beta$ otherwise.

Algorithm to Compute the Final Hypothesis: After $T-1$ descent steps, the algorithm calls the Generalized Row Counting twice for each $t \in [0, T-1]$, with the following inputs:

- $\ell = 1$ and additive expression $1 - \beta^{(t)} \cdot x_i - \epsilon|\beta^{(t)}| \cdot |x_i|$
- $\ell = -1$ and additive expression $1 + \beta^{(t)} \cdot x_i - \epsilon|\beta^{(t)}| \cdot |x_i|$

Note that both of these expressions are equivalent to $1 - y_i\beta^{(t)} \cdot x_i - \epsilon|\beta^{(t)}| \cdot |x_i|$. Let the array H^+ be the output for the first call and H^- be the output for the second call. Note that H_k^+ and H_k^- are monotonically decreasing in k by the the definition of the Generalized Row Counting algorithm. Let L^+ be the largest k such that $H_k^+ \geq 0$ and L^- be the largest k such that $H_k^- \geq 0$. The algorithm then returns as its final hypothesis $\widehat{\beta}$, the hypothesis $\beta^{(\widehat{t})}$ where $\widehat{t}$ is defined by:

$$\widehat{t} = \operatorname*{argmin}_{t\in[T]} \widehat{F}(\beta^{(t)}, X) \tag{3.5}$$

where

$$\widehat{F}(\beta^{(t)}, X) = \frac{1}{N}\left(\sum_{k=0}^{L^+-1}(1+\epsilon)^k(H_k^+ - H_{k+1}^+) + (1+\epsilon)^{L^+}H_{L^+}^+\right) + \frac{1}{N}\left(\sum_{k=0}^{L^--1}(1+\epsilon)^k(H_k^- - H_{k+1}^-) + (1+\epsilon)^{L^-}H_{L^-}^-\right) + \lambda\left\|\beta^{(t)}\right\|_2^2 \tag{3.6}$$

Note that the values L^-, L^+, H^+ and H^- in the definition of $\widehat{F}$, in equation (3.6), all depend upon t, which we suppressed to make the notation somewhat less ugly.

4 Algorithm Analysis

In Section 4.1, we prove Theorem 4.1 which bounds the convergence of our projected pseudo-gradient descent algorithm in a rather nonstandard way by applying known results on online convex optimization [16, 26]. In Section 4.2, we introduce our definition of stability and then prove Theorem 1.2.

4.1 Perturbation Analysis Before stating Theorem 4.1 we need some definitions.

Definition 4.1.

- *A point p is an ϵ-perturbation of point q if every component of p is within $(1+\epsilon)$ factor of the corresponding component of q. Meaning in each dimension j we have $(1-\epsilon)q \leq p \leq (1+\epsilon)q$*
- *A point set X_a is an ϵ-perturbation of a point set X_b if there is a bijection between X_a and X_b such that every point in X_a is an ϵ-perturbation of its corresponding point in X_b.*
- *Let $\beta^* = \operatorname{argmin}_\beta F(\beta, X)$ to be the optimal solution at X.*
- *For any ϵ-perturbation X_a of X, define $\beta_a^* = \operatorname{argmin}_\beta F(\beta, X_a)$ to be the optimal solution at X_a.*
- *For a given hypothesis β, we call a point x with label y* close *if there is some ϵ-perturbation x' of x such that $1 - y\beta \cdot x' < 0$; otherwise it is called* far. *In other words, a point x with label y is close if $1 - y\beta \cdot x < \epsilon\,|\beta| \cdot |x|$*

Theorem 4.1. *Assume our projected pseudo-gradient descent algorithm ran for $T-1$ descent steps. Then for all hypotheses $\beta \in \mathbb{R}^d$ there exist ϵ-perturbations X_a and X_b of X such that*

$$F(\widehat{\beta}, X_a) \leq (1+\epsilon)F(\beta, X_b) + \frac{32d^{3/2}}{\lambda\sqrt{T}} \tag{4.7}$$

To prove Theorem 4.1, our main tool is a result from the online convex optimization literature [16, 26]. Intuitively, this theorem states that if we run gradient descent on a series of different convex cost functions (as opposed to a single cost function), the algorithm's average cost for the points it finds in each iteration of the gradient descent converges to the average cost of any optimal fixed point.

Theorem 4.2. *[16, 26] Let $g_0, g_1, ..., g_{T-1} : \mathbb{R}^n \to \mathbb{R}$ be G-Lipschitz functions over a convex region $\mathcal{K}$, i.e., $||\nabla g_t(\beta)|| \leq G$ for all $\beta \in \mathcal{K}$ and all t. Then, starting at point $\beta^{(0)} \in \mathbb{R}^n$ and using the update rule of $\beta^{(t)} \leftarrow \Pi_{\mathcal{K}}\left(\beta^{(t-1)} - \eta_t \nabla g_{t-1}(\beta^{(t-1)})\right)$, with $\eta = \frac{D}{G\sqrt{t}}$ for $T-1$ steps, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1} g_t(\beta^{(t)}) \leq \frac{1}{T}\sum_{t=0}^{T-1} g_t(\beta^*) + \frac{2DG}{\sqrt{T}} \tag{4.8}$$

for all β^* with $||\beta^{(0)} - \beta^*|| \leq D$.

To apply Theorem 4.2, we set $g_t(\beta^{(t)}) = F(\beta^{(t)}, X^{(t)}, Y)$, where $X^{(t)}$ is an ϵ-perturbation of X, such that the pseudo-gradient at X is equal to the true gradient at $X^{(t)}$. We establish the existence of $X^{(t)}$ in Lemma 4.1. Thus, our projected pseudo-gradient descent algorithm updates the hypothesis in exactly the same way as stated in Theorem 4.2 (assuming that we use the same upper bounds on D and G). Then in definition 4.2 we identify the ϵ-perturbation Z that minimizes $F(\beta, Z)$, and then in Lemma 4.2 bound the relative error between $\widehat{F}(\beta, X)$ and $F(\beta, Z)$. Finally, in Lemma 4.3 and Lemma 4.4, we show the existence of X_b and X_a, respectively, and that allows us to conclude the proof of Theorem 4.1.

Lemma 4.1. *In every descent step t, the computed pseudo-gradient $\widehat{G}$ is the exact gradient of $F(\beta^{(t)}, X^{(t)})$ for some point set $X^{(t)}$ that is an ϵ-perturbation of X.*

Proof. To prove the claim, we show how to find a desired $X^{(t)}$ – this is only for the sake of the proof, and the algorithm doesn't need to know $X^{(t)}$. We call any point x with label y "far" if it satisfies the inequality

$$1 - y\beta^{(t)} \cdot x \geq \epsilon\left|\beta^{(t)}\right| \cdot |x|, \tag{4.9}$$

otherwise we call the point "close". Note that for a far point, there is no ϵ-perturbation to make the derivative of the loss function 0. That is, for any point x with label y, if $1 - y\beta \cdot x \geq \epsilon \sum_{j\in[d]} |\beta_j|\,|x_j|$, then we have $1 - y\beta \cdot x' \geq 0$ for any x' that is ϵ-perturbation of x. To see this, note that we have $1 - y\beta \cdot x' = 1 - \sum_{k=1}^{d} \beta_k x'_k \geq 1 - \sum_{k=1}^{d} (\beta_k x_k + \epsilon\,|\beta_k|\,|x_k|) \geq 0$ because of x' being ϵ-perturbation of x. On the other hand, for all the close points there exists a perturbation x' such that $1 - y\beta^{(t)} \cdot x' < 0$. We first perturb all of the close points such that they don't have any effect on the gradient.

Next, we need to show a perturbation of the far points for which the $\widehat{G}$ is the gradient of the loss function. Let X_f^+ and X_f^- be the set of far points with positive and negative labels. Let $X_f = X_f^+ \cup X_f^-$. We show the perturbation for each dimension k separately. Based on definition of $\widehat{G}_k^+$ and $\widehat{G}_k^-$ we have:

$$\begin{aligned}&\widehat{G}_k^+ + \widehat{G}_k^- \\ &= \sum_{v\in D(k)} v\,\widehat{C}_{k,v}^- - \sum_{v\in D(k)} v\,\widehat{C}_{k,v}^+ \\ &= \sum_{v\in D(k)} v\,(1\pm\epsilon)C_{k,v}^- - \sum_{v\in D(k)} v\,(1\pm\epsilon)C_{k,v}^+\end{aligned}$$

Note that in our notation $(1\pm\epsilon)$ can be any real number in $[1-\epsilon, 1+\epsilon]$ and not necessarily the values $1-\epsilon$ and $1+\epsilon$. Also, note that $C_{k,v}^+$ is the number of points in X_f^+ with value v in dimension k. Therefore,

$$\begin{aligned}\widehat{G}_k^+ + \widehat{G}_k^- &= \sum_{v\in D(k)} v\,(1\pm\epsilon)C_{k,v}^- - \sum_{v\in D(k)} v\,(1\pm\epsilon)C_{k,v}^+ \\ &= \sum_{x_i\in X_f^-} (1\pm\epsilon)x_{i,k} - \sum_{x_i\in X_f^+} (1\pm\epsilon)x_{i,k} \\ &= -\sum_{x_i\in X_f} (1\pm\epsilon)y_i x_{i,k}\end{aligned}$$

where the last term is $N\frac{\partial L(\beta^{(t)}, X^{(t)})}{\partial \beta_k^{(t)}}$ in which $X^{(t)}$ is an ϵ-perturbation of X. ☐

Definition 4.2. *Let $Z^{(t)}$ be an ϵ-perturbation of X such that for all $z_i \in Z^{(t)}$ and for all dimensions k*

$$z_{i,k} = \begin{cases}(1-\epsilon)x_{i,k} & y_i\beta_k^{(t)} \geq 0 \\ (1+\epsilon)x_{i,k} & y_i\beta_k^{(t)} < 0\end{cases}$$

Note that this ϵ-perturbation minimizes $F(\beta^{(t)}, Z^{(t)})$.

Lemma 4.2. *For all t, we have*

$$\frac{1}{1+\epsilon}F(\beta^{(t)}, Z^{(t)}) \leq \widehat{F}(\beta^{(t)}, X) \leq F(\beta^{(t)}, Z^{(t)}).$$

Proof. Consider a value t and let

$$N^+(\tau) = \\ \left|\{x_i \mid y_i = +1 \text{ and } 1 - \beta^{(t)} \cdot x_i - \epsilon\left|\beta^{(t)}\right| \cdot |x_i| \geq \tau\}\right|$$

and

$$N^-(\tau) = \\ \left|\{x_i \mid y_i = -1 \text{ and } 1 + \beta^{(t)} \cdot x_i - \epsilon\left|\beta^{(t)}\right| \cdot |x_i| \geq \tau\}\right|$$

Intuitively, $N^+(\tau)$ is the number of positively labeled points in X for which the hinge loss of any ϵ-perturbation of them is at least τ. Similarly, $N^-(\tau)$ is the number of negatively labeled points with the same property.

Before proving the lemma we prove the following claim:

$$F(\beta^{(t)}, Z^{(t)}) = \\ \frac{1}{N}\int_{\tau=0}^{\infty} N^+(\tau)\mathrm{d}\tau + \frac{1}{N}\int_{\tau=0}^{\infty} N^-(\tau)\mathrm{d}\tau + \lambda\left\|\beta^{(t)}\right\|^2.$$

Note that based on the definition of $Z^{(t)}$ it is the case that

$$1 - y_i\beta^{(t)} \cdot z_i = 1 - y_i\beta^{(t)} \cdot x_i - \epsilon\left|\beta^{(t)}\right| \cdot |x_i|.$$

Therefore, we have

$$N^+(\tau) = \left|\{y_i = +1 \in Z^{(t)} \text{ and } 1 - y_i\beta^{(t)} \cdot z_i \geq \tau\}\right|$$

and

$$N^-(\tau) = \left|\{y_i = -1 \in Z^{(t)} \text{ and } 1 - y_i\beta^{(t)} \cdot z_i \geq \tau\}\right|.$$

Hence,

$$\begin{aligned}L(\beta^{(t)}, Z^{(t)}) &= \frac{1}{N}\sum_i \max(0, 1 - y_i\beta \cdot z_i) \\ &= \frac{1}{N}\sum_{i:1-y_i\beta\cdot z_i\geq 0} 1 - y_i\beta \cdot z_i \\ &= \frac{1}{N}\sum_{i:1-y_i\beta\cdot z_i\geq 0}\int_{\tau=0}^{1-y_i\beta\cdot z_i}\mathrm{d}\tau \qquad (4.10) \\ &= \frac{1}{N}\int_{\tau=0}^{\infty}\sum_{i:1-y_i\beta\cdot z_i\geq\tau}\mathrm{d}\tau \\ &= \frac{1}{N}\int_{\tau=0}^{\infty}(N^+(\tau) + N^-(\tau))\mathrm{d}\tau\end{aligned}$$

Therefore,

(4.11)
$$F(\beta^{(t)}, Z^{(t)}) \\ = \frac{1}{N}\int_{\tau=0}^{\infty} N^+(\tau)\mathrm{d}\tau + \frac{1}{N}\int_{\tau=0}^{\infty} N^-(\tau)\mathrm{d}\tau + \lambda\left\|\beta^{(t)}\right\|^2.$$

The number of points with label ℓ satisfying $1 - \ell\beta^{(t)} \cdot x_i - \epsilon\left|\beta^{(t)}\right| \cdot |x_i| \geq \tau$ for any $\tau \in [H_k^\ell, H_{k+1}^\ell)$ is in

the range $[\lfloor(1+\epsilon)^k\rfloor, \lfloor(1+\epsilon)^{(k+1)}\rfloor)$. Therefore, the claim follows by replacing $N^+(\tau)$ in Equation (4.11) with $(1+\epsilon)^k$ for all the values of $\tau \in [H_k^+, H_{k+1}^+)$ and replacing $N^-(\tau)$ in (4.11) with $(1+\epsilon)^k$ for all the values of $\tau \in [H_k^-, H_{k+1}^-)$.
□

LEMMA 4.3. *For all hypothesis β, there exists an ϵ-perturbation X_b of X such that*

$$\min_s F(\beta^{(s)}, Z^{(s)}) \leq F(\beta, X_b) + \frac{2DG}{\sqrt{T}}$$

Proof. By Theorem 4.2

$$\frac{1}{T}\sum_{t=0}^{T-1} F(\beta^{(t)}, X^{(t)}) \leq \frac{1}{T}\sum_{t=0}^{T-1} F(\beta, X^{(t)}) + \frac{2DG}{\sqrt{T}} \tag{4.12}$$

Then

$$\begin{aligned} \min_s F(\beta^{(s)}, Z^{(s)}) &\leq \frac{1}{T}\sum_{t=0}^{T-1} F(\beta^{(t)}, Z^{(t)}) \\ &\leq \frac{1}{T}\sum_{t=0}^{T-1} F(\beta^{(t)}, X^{(t)}). \end{aligned} \tag{4.13}$$

The first inequality follows since the minimum is less than the average, and the second inequality follows from the definition of $Z^{(t)}$. Let $u = \text{argmax}_t F(\beta, X^{(t)})$, and $X_b = X^{(u)}$. Then

$$\frac{1}{T}\sum_{t=0}^{T-1} F(\beta, X^{(t)}) \leq \max_t F(\beta, X^{(t)}) = F(\beta, X_b) \tag{4.14}$$

Thus, combining lines (4.12), (4.13) and (4.14) we can conclude that:

$$\min_s F(\beta^{(s)}, Z^{(s)}) \leq F(\beta, X_b) + \frac{2DG}{\sqrt{T}} \tag{4.15}$$

□

LEMMA 4.4. *There exists an ϵ-perturbation X_a of X such that*

$$F(\widehat{\beta}, X_a) \leq (1+\epsilon)\min_s F(\beta^{(s)}, Z^{(s)})$$

Proof. Let $X_a = Z^{(\widehat{t})}$ where

$$\begin{aligned} &F(\widehat{\beta}, X_a) \\ &\leq (1+\epsilon)\widehat{F}(\widehat{\beta}, X) && \text{By Lemma 4.2} \\ &= (1+\epsilon)\min_s \widehat{F}(\beta^{(s)}, X) && \text{By definition of } \widehat{\beta} \\ &\leq (1+\epsilon)\min_s F(\beta^{(s)}, Z^{(s)}) && \text{By Lemma 4.2} \end{aligned}$$

□

4.2 Stability Analysis

Our formal definition of stability, which we give in Definition 4.3, is surely not the first natural formalization that one would think of. The most natural formulation would be something like that a good solution would stay a good solution if one slightly perturbed the points once. However, due to the nature of non-traditional approximation achieved in Theorem 4.1, we need (as best as we can tell) a formulation of stability that intuitively says that a good solution would stay a good solution if one slightly perturbed the points twice. From a practical point of view, there probably isn't any real difference between one and two slight perturbations, but we admit the mathematical elegance of this formulation is less than fully satisfying.

DEFINITION 4.3. *An SVM instance X is (α, δ, γ)-stable for $\delta \leq 1$ if for all X_a and X_b that are α-perturbations of X it is the case that:*

- *β_a^* is a $(1+\delta)$ approximation to the optimal objective value at X_b, that is, $F(\beta_a^*, X_b) \leq (1+\delta)\min_\beta F(\beta, X_b)$.*
- *If β_a is $(1+2\delta)$ approximation to the optimal SVM objective value at X_a then β_a is a $(1+\gamma)$ approximation to the optimal SVM objective value at X_b. That is if $F(\beta_a, X_a) \leq (1+2\delta)\min_\beta F(\beta, X_a)$ then $F(\beta_a, X_b) \leq (1+\gamma)\min_\beta F(\beta, X_b)$*

Proof. [Proof of Theorem 1.2] Let $\epsilon \leq \min(\delta/8, \alpha)$.

$$\begin{aligned} &F(\widehat{\beta}, X_a) \\ &\leq (1+\epsilon)F(\beta_a^*, X_b) + \frac{32d^{3/2}}{\lambda\sqrt{T}} \\ &\qquad [X_a \text{ and } X_b \text{ come from Theorem 4.1}] \\ &= (1+\epsilon)(1+\delta)F(\beta_a^*, X_a) + \frac{32d^{3/2}}{\lambda\sqrt{T}} \\ &\qquad [\text{By definition of stability}] \\ &= (1+\epsilon)(1+\delta)F(\beta_a^*, X_a) + \frac{\delta}{8}F(\widehat{\beta}, X_a) \\ &\qquad [\text{By definition of } T] \\ &\leq \frac{(1+\delta)(1+\epsilon)}{1-\delta/8}F(\beta_a^*, X_a) \\ &\qquad [\text{By algebra}] \\ &\leq (1+2\delta)F(\beta_a^*, X_a) \\ &\qquad [\text{ by definition of } \epsilon] \end{aligned}$$

Finally, since $\widehat{\beta}$ is $(1+2\delta)$ approximate solution at X_a, by the definition of stability, $\widehat{\beta}$ is a $(1+\gamma)$ approximate solution at X. □

References

[1] Google BigQuery ML website. https://cloud.google.com/bigquery-ml/docs/bigqueryml-intro.

[2] RelationalAI website. https://www.relational.ai/.

[3] Mahmoud Abo Khamis, Ryan R Curtin, Benjamin Moseley, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. On functional aggregate queries with additive inequalities. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 414–431. ACM, 2019.

[4] Mahmoud Abo Khamis, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Ac/dc: in-database learning thunderstruck. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, page 8. ACM, 2018.

[5] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Indatabase learning with sparse tensors. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, SIGMOD/PODS '18, pages 325–340, New York, NY, USA, 2018. ACM.

[6] Mahmoud Abo Khamis, Hung Q. Ngo, Dan Olteanu, and Dan Suciu. Boolean tensor decomposition for conjunctive queries with negation. In *ICDT*, pages 21:1–21:19, 2019.

[7] Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. Faq: Questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pages 13–28, New York, NY, USA, 2016. ACM.

[8] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. In *Artificial intelligence and statistics*, pages 1–8, 2009.

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

[10] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Trans. Inf. Theor.*, 46(2):325–343, 2006.

[11] Haris Angelidakis, Konstantin Makarychev, and Yury Makarychev. Algorithms for stable and perturbation-resilient problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 438–451, 2017.

[12] Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 739–748. IEEE, 2008.

[13] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1-2):49–54, 2012.

[14] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *Journal of the ACM (JACM)*, 60(2):1–34, 2013.

[15] Maria Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM Journal on Computing*, 45(1):102–155, 2016.

[16] Nikhil Bansal and Anupam Gupta. Potential-function proofs for first-order methods. *CoRR*, abs/1712.04581, 2017.

[17] Jinbo Bi and Tong Zhang. Support vector classification with input data uncertainty. In *Advances in neural information processing systems*, pages 161–168, 2005.

[18] Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(5):643–660, 2012.

[19] Léon Bottou and Yann L Cun. Large scale online learning. In *Advances in neural information processing systems*, pages 217–224, 2004.

[20] Andriy Burkov. *The Hundred Page Machine Learning Book*.

[21] Ryan R. Curtin, Benjamin Moseley, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Rk-means: Fast clustering for relational data. *CoRR*, abs/1910.04939, 2019.

[22] Amit Daniely, Nati Linial, and Michael Saks. Clustering is difficult only when it does not matter. *arXiv preprint arXiv:1205.4891*, 2012.

[23] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, 1996.

[24] Martin Grohe. The structure of tractable constraint satisfaction problems. In *International Symposium on Mathematical Foundations of Computer Science*, pages 58–72. Springer, 2006.

[25] Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. In *SODA*, pages 289–298, 2006.

[26] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[27] Mahmoud Abo Khamis, Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Alireza Samadian. Approximate aggregate queries under additive inequalities. *CoRR*, abs/2003.10588, 2020.

[28] Anthony Klug. On conjunctive queries containing inequalities. *J. ACM*, 35(1):146–160, January 1988.

[29] J. Kohlas and N. Wilson. Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artif. Intell.*, 172(11):1360–1399, 2008.

[30] Paraschos Koutris, Tova Milo, Sudeepa Roy, and Dan Suciu. Answering conjunctive queries with inequalities. *Theory of Computing Systems*, 61(1):2–30, Jul 2017.

[31] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 299–308. IEEE, 2010.

[32] Arun Kumar, Jeffrey Naughton, and Jignesh M. Patel. Learning generalized linear models over normalized data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1969–1984. ACM, 2015.

[33] Arun Kumar, Jeffrey Naughton, Jignesh M. Patel, and Xiaojin Zhu. To join or not to join?: Thinking twice about joins before feature selection. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 19–34, New York, NY, USA, 2016. ACM.

[34] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu–linial stable instances of max cut and minimum multiway cut. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 890–906. SIAM, 2014.

[35] Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *Journal of the ACM (JACM)*, 60(6):42, 2013.

[36] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms: [extended abstract]. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 37–48, New York, NY, USA, 2012. ACM.

[37] Hung Q. Ngo, Christopher Ré, and Atri Rudra. Skew strikes back: New developments in the theory of join algorithms. In *SIGMOD Rec.*, pages 5–16, 2013.

[38] Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59(6):1–22, 2013.

[39] Tim Roughgarden. Beyond worst-case analysis. *Communications of the ACM*, 62(3):88–96, 2019.

[40] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 3–18. ACM, 2016.

[41] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[42] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning.* Mit Press, 2012.

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

A Analysis of Gradient Descent for SVM

Theorem A.1 and Corollary A.1 give bounds on the number of iterations on projected gradient descent to reach solutions with bounded absolute error and bounded relative error, respectively.

THEOREM A.1. *[16, 26] Let $\mathcal{K}$ be a convex body and F be a function such that $\|\nabla F(\beta)\|_2 \le G$ for $\beta \in \mathcal{K}$. Let $\beta^* = \text{argmin}_{\beta\in\mathcal{K}} F(\beta)$ be the optimal solution. Let D be an upper bound on $\left\|\beta^{(0)} - \beta^*\right\|_2$, the 2-norm distance from the initial candidate solution to the optimal solution. Let $\widehat{\beta}_s = \frac{1}{s}\sum_{t=0}^{s-1}\beta^{(t)}$. Let $\eta_t = \frac{D}{G\sqrt{t}}$. Then after $T-1$ iterations of projected gradient descent, it must be the case that*

$$F(\widehat{\beta}_T) - F(\beta^*) \le \frac{2DG}{\sqrt{T}}$$

COROLLARY A.1. *Adopting the assumptions from Theorem A.1, if $T \ge \left(\frac{4DG}{\epsilon F(\widehat{\beta}_T)}\right)^2$ then*

$$F(\widehat{\beta}_T) \le (1+\epsilon)F(\beta^*)$$

That is, projected gradient descent achieves relative error ϵ.

The gradient of SVM objective F is

$$\nabla F = 2\lambda\beta - \frac{1}{N}y_i\sum_{i\in\mathcal{L}}x_i \tag{A.1}$$

where $\mathcal{L}$ is the collection $\{i \mid \beta\cdot x_i \le 1\}$ of indices i where x_i is currently contributing to the objective. Note that in this hinge loss function, the gradient for the points on the hyperplane $1-\beta\cdot x = 0$ does not exist, since the gradient is not continuous at this point. In our formulation we have used the sub-gradient for the points on $1-\beta\cdot x = 0$, meaning for a β on the hyperplane $1-\beta\cdot x = 0$, we have used the limit of the gradient of the points that $1-\beta' x > 0$ when β' goes to β. For all the points that $1-\beta' x > 0$, the gradient is x; therefore, the limit is also x.

Assume $\beta^{(0)}$ is the origin and adopt the assumptions of Theorem A.1. Then $\nabla F(\beta^*) = 0$ implies for any dimension j

$$\left|\beta_j^*\right| = \left|\frac{1}{2N\lambda}\sum_{i\in\mathcal{L}}x_{ij}\right| \le \frac{1}{2\lambda} \tag{A.2}$$

where the additional subscript of j refers to dimension j. And thus

$$\left\|\beta^{(0)} - \beta^*\right\|_2 \le \|\beta^*\|_2 \le \sqrt{d}\max_{j\in[d]}\left|\beta_j^*\right| \le \frac{\sqrt{d}}{2\lambda} \tag{A.3}$$

Thus let us define our convex body $\mathcal{K}$ to be the hypersphere with radius $\frac{\sqrt{d}}{2\lambda}$ centered at the origin. Thus for $\beta \in \mathcal{K}$,

$$\begin{aligned}\|\nabla F(\beta)\|_2 &= \sqrt{\sum_{j\in[d]}\left(2\lambda\beta_j - \frac{1}{N}\sum_{i\in\mathcal{L}} x_{ij}\right)^2} \\ &\le \sqrt{\sum_{j\in[d]} 4(\lambda\beta_j)^2 + 2\left(\frac{1}{N}\sum_{i\in\mathcal{L}} x_{ij}\right)^2} \\ &\le 2\lambda\sum_{j\in[d]}|\beta_j| + \sqrt{2}\frac{1}{N}\left|\sum_{i\in\mathcal{L}} x_{ij}\right| \\ &\le 2d + \sqrt{2}d \\ &\le 4d.\end{aligned}$$

The first inequality is because $(a-b)^2 \le 2a^2 + 2b^2$, and the second inequality is because $\sqrt{\sum_i a_i^2} \le \sum_i |a_i|$.

Theorem A.2. *Let the convex body $\mathcal{K}$ be the hypersphere with radius $\frac{\sqrt{d}}{2\lambda}$ centered at the origin. Let $F(\beta)$ be the SVM objective function. Let $\beta^* = \operatorname{argmin}_\beta F(\beta)$ be the optimal solution. Let $\widehat{\beta}_s = \frac{1}{s}\sum_{t=0}^{s-1}\beta^{(t)}$. Let $\eta_t = \frac{1}{8\lambda\sqrt{dt}}$. Then after $T-1$ iterations of projected gradient descent, it must be the case that*

$$F(\widehat{\beta}_T) - F(\beta^*) \le \frac{4d^{3/2}}{\lambda\sqrt{T}}$$

Theorem 1.1 then follows by a straightforward application of Theorem A.2.

B Background

We start by defining the joins.

Definition B.1. (Join) *Let $T_1, \dots, T_m$ be a set of tables and for each table T_i, let C_i denote the set of columns in T_i. The join of $T_1, \dots, T_m$ denoted by $T_1 \bowtie \cdots \bowtie T_m$ is a table J with set of column $C = \bigcup_j C_j$ such that a tuple/row x is in J if an only if its projection onto C_j is in T_j for all j.*

The structure of a join can be modeled as a hypergraph in which each vertex v_i of the hypergraph is associated with column f_i and each hyperedge S_j is associated with a table T_j such that $v_i \in S_j$ if and only if $f_i \in C_j$. In what the following, we use n to denote the size of the largest input table in the join query $Q = T_1 \bowtie \cdots \bowtie T_m$. We also use J to denote the output and $|J|$ to denote its size. We use the join query Q and its hypergraph $\mathcal{H}$ interchangeably.

B.1 Fractional edge cover number and output size bounds

Definition B.2. (Fractional edge cover number) *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 $\vec{\lambda} = (\lambda_S)_{S\in\mathcal{E}}$ to the following linear program:*

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

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 join 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 B.1. (AGM-bound [12, 25]) *Given a join query Q over a relational database instance I, the output size is bounded by*

$$|J| \le n^{\rho^*},$$

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

Theorem B.2. (AGM-bound is tight [12, 25]) *Given a join query Q and a non-negative number n, there exists a database instance I whose relation sizes are upper-bounded by n and satisfies*

$$|J| = \Theta(n^{\rho^*}).$$

Worst-case optimal join algorithms [43, 36, 37] can be used to answer any join query Q in time

$$O(|\mathcal{V}| \cdot |\mathcal{E}| \cdot n^{\rho^*} \cdot \log n). \tag{B.4}$$

B.2 Tree decompositions, acyclicity, and width parameters

Definition B.3. (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 B.4. (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 B.5. (g-width of a hypergraph [9]) *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 B.6. (*fractional hypertree 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.

B.3 Algebraic Structures In this section, we define some of the algebraic structures used in the paper. First, we discuss the definition of a monoid. A monoid is a semi-group with an identity element. Formally, it is the following.

Definition B.7. *Fix a set S and let $\oplus$ be a binary operator $S \times S \to S$. The set S with $\oplus$ is a monoid if (1) the operator satisfies associativity; that is, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ for all $a, b, c \in S$ and (2) there is identity element $e \in S$ such that for all $a \in S$, it is the case that $e \oplus a = a \oplus e = a$.*

A commutative monoid is a moniod where the operator $\oplus$ is commutative. That is $a \oplus b = b \oplus a$ for all $a, b \in S$.

Next, we define a semiring.

Definition B.8. *A semiring is a set R with two operators $\oplus$ and $\otimes$. The $\oplus$ operator is referred to as addition and the $\otimes$ is referred to as multiplication. This is a semiring if,*

1. *it is the case that R and $\oplus$ are a commutative monoid with 0 as the identity.*
2. *R and $\otimes$ is a monoid with identity 1.*
3. *the multiplication distributes over addition. That is for all $a, b, c \in R$ it is the case that $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ and $(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$.*
4. *the 0 element annihilates R. That is, $a \otimes 0 = 0$ and $0 \otimes a = 0$ for all $a \in R$.*

A commutative semiring is a semiring where the multiplication is commutative. That is, $a \otimes b = b \otimes a$ for all $a, b \in S$.

B.4 FAQ-AI Query The input to FAQ-AI problem consists of three components:

- A collection of relational tables $T_1, \ldots T_m$ with real-valued entries. Let $J = T_1 \bowtie T_2 \bowtie \cdots \bowtie T_m$ be the design matrix that arises from the inner join of the tables. Let n be an upper bound on the number of rows in any table T_i, let N be the number of rows in J, and let d be the number of columns in J.
- An FAQ $Q(J)$ that is either a SumProd query or a SumSum query. We define a SumSum query to be a query of the form:

$$Q(J) = \bigoplus_{x \in J} \bigoplus_{i=1}^{d} F_i(x_i)$$

where $(R, \oplus, I_0)$ is a commutative monoid over the arbitrary set R with identity I_0. We define a SumProd query to be a query of the form:

$$Q(J) = \bigoplus_{x \in J} \bigotimes_{i=1}^{d} F_i(x_i)$$

where $(R, \oplus, \otimes, I_0, I_1)$ is a commutative semiring over the arbitrary set R with additive identity I_0 and multiplicative identity I_1. In each case, x_i is the entry in column i of x, and F_i is an arbitrary function with range R.

- A collection $\mathcal{L} = \{(G_1, L_1), \ldots (G_b, L_b)\}$ where G_i is a collection $\{g_{i,1}, g_{i,2}, \ldots g_{i,d}\}$ of d functions that map the column domains to the reals, and each L_i is a scalar.

FAQ-AI(k) is a special case of FAQ-AI when the cardinality of $\mathcal{L}$ is at most k.

The output for the FAQ-AI problem is the result of the query on the subset of the design matrix that satisfies the additive inequalities. That is, the output for the FAQ-AI instance with a SumSum query is:

$$Q(\mathcal{L}(J)) = \bigoplus_{x \in \mathcal{L}(J)} \bigoplus_{i=1}^{d} F_i(x_i) \tag{B.5}$$

And the output for the FAQ-AI instance with a SumProd query is:

$$Q(\mathcal{L}(J)) = \bigoplus_{x \in \mathcal{L}(J)} \bigotimes_{i=1}^{d} F_i(x_i) \tag{B.6}$$

Here $\mathcal{L}(J)$ is the set of tuples $x \in J$ that satisfy all the additive inequalities in $\mathcal{L}$, that is for all $i \in [1, b]$, $\sum_{j=1}^{d} g_{i,j}(x_j) \leq L_i$, where x_j is the value of coordinate j of x.

We now illustrate how some of the SVM related problems can be reduced to FAQ-AI(1). First consider the problem of counting the number of negatively labeled points correctly classified by a linear separator. Here each row x of the design matrix J conceptually consists of a point in $\mathbb{R}^{d-1}$, whose coordinates are specified by the first $d-1$ columns in J, and a label in $\{1, -1\}$ in column d. Let the linear separator be defined by $\beta \in \mathbb{R}^{d-1}$. A negatively labeled point x is correctly classified if $\sum_{i=1}^{d-1} \beta_i x_i \leq 0$. The number of such points can be counted using SumProd query with one additive inequality as follows: $\oplus$ is addition, $\otimes$ is multiplication, $F_i(x_i) = 1$ for all $i \in [d-1]$, $F_d(x_d) = 1$ if $x_d = -1$, and $F_d(x_d) = 0$ otherwise, $g_{1,j}(x_j) = \beta_j x_j$ for $j \in [d-1]$, $g_{1,d}(x_d) = 0$, and $L_1 = 0$. Next, consider the problem of finding the minimum distance to the linear separator of a correctly classified negatively labeled point. This distance can be computed using a SumProd query with one additive inequality as follows: $\oplus$ is the binary minimum operator, $\otimes$ is addition, $F_i(x_i) = \beta_i x_i$ for all $i \in [d-1]$, $F_d(x_d) = 1$ if $x_d = -1$, and $F_d(x_d) = 0$ otherwise, $g_{1,j}(x_j) = \beta_j x_j$ for $j \in [d-1]$, $g_{1,d}(x_d) = 0$, and $L_1 = 0$.