Efficient and robust active learning methods for interactive database exploration

Enhui Huang, Yanlei Diao, Anna Liu, Liping Peng, Luciano Di Palma

Noname manuscript No.
(will be inserted by the editor)

Efficient and Robust Active Learning Methods for Interactive Database Exploration

Enhui Huang[†] · **Yanlei Diao**[*,†] · **Anna Liu**[*] · **Liping Peng**[*] · **Luciano Di Palma**[†]

Received: date / Accepted: date

Abstract There is an increasing gap between fast growth of data and the limited human ability to comprehend data. Consequently, there has been a growing demand of data management tools that can bridge this gap and help the user retrieve high-value content from data more effectively. In this work, we propose an interactive data exploration system as a new database service, using an approach called "explore-by-example." Our new system is designed to assist the user in performing highly effective data exploration while reducing the human effort in the process. We cast the explore-by-example problem in a principled "active learning" framework. However, traditional active learning suffers from two fundamental limitations: slow convergence and lack of robustness under label noise. To overcome the slow convergence and label noise problems, we bring the properties of important classes of database queries to bear on the design of new algorithms and optimizations for active learning-based database exploration. Evaluation results using real-world datasets and user interest patterns show that our new system, both in the noise-free case and in the label noise case, significantly outperforms state-of-the-art active learning techniques and data exploration systems in accuracy while achieving the desired efficiency for interactive data exploration.

Keywords Interactive Data Exploration · Active Learning · Label Noise

Enhui Huang
E-mail: enhui.huang@polytechnique.edu

Yanlei Diao
E-mail: yanlei.diao@polytechnique.edu

Anna Liu
E-mail: anna@math.umass.edu

Liping Peng
E-mail: lppeng@cs.umass.edu

Luciano Di Palma
E-mail: luciano.di-palma@polytechnique.edu

[†]École Polytechnique, France
[*]University of Massachusetts Amherst, USA

1 Introduction

Today data is being generated at an unprecedented rate. However, the human ability to comprehend data remains as limited as before. Consequently, there has been a growing demand of data management tools that can bridge the gap between the data growth and limited human ability, and help retrieve high-value content from data more effectively.

To respond to such needs, we build a new database service for interactive exploration in a framework called "*explore-by-example*" [23, 24]. In this framework, the database content is considered as table, and the user is interested in a subset of its tuples but not all. In the data exploration process, the system allows the user to interactively label tuples as "interesting" or "not interesting," so that it can construct an increasingly-more-accurate model of the user interest. Eventually, the model is turned into a *user interest query*[1] that will run the model as a user defined function (UDF) on the database to retrieve all relevant tuples.

In this work, we consider several target applications. First, when a scientist comes to explore a large sky survey database such as the Sloan Digital Sky Survey (SDSS) [69], he/she may not be able to express his/her data interest precisely. Instead, the scientist may prefer to navigate through a region of the sky, see a few examples of sky objects, provide yes or no feedback, and ask the system to find all other (potentially many more) relevant sky objects from the database.

[1] In our work we use the term, *user interest query*, to refer to the final query that represents the user interest, and *user interest model* to refer to an immediate model before it converges to the true user interest.

Second, consider many web applications backed by a large database, such as E-commerce websites and housing websites, which provide a simple search interface but leave the job of filtering through a long list of returned objects to the user. The new database service in the explore-by-example framework will provide these applications with a new way to interact with the user and, more importantly, help the user filter through numerous objects more efficiently.

To build an explore-by-example system, our approach is to cast it in an "*active learning*" framework: We treat the modeling of the user interest as a *classification* problem where all the user labeled examples thus far are used to train a classification model. Then active learning [13,66,73] decides how to choose a new example, from the unlabeled database, for the user to label next so that the system can learn the user interest efficiently. The active learning based explore-by-example approach offers potential benefits over alternative methods such as faceted search [45, 61, 62] and semantic windows [44] for the following reasons.

First, the user interest may include varying degrees of complexity, or the user does not have prior knowledge about the complexity and expects the system to learn a model as complex as necessary for his/her interest. For example, the user interest may involve more complex constraints, e.g., "$(\frac{rowc-a_1}{b_1})^2 + (\frac{colc-a_2}{b_2})^2 < c$" , from the SDSS example query set [68], where $rowc$ and $colc$ refer to the row and column center positions of each photometric object, or "$length * width > c$" from our Car User Study. If the user knows the function shape and constants in advance, some of the above examples (e.g., the ellipse pattern) can be supported by semantic windows [44] as pre-defined patterns. However, if the user does not have such knowledge, neither faceted search nor semantic windows can be applied, but explore-by-example still works regardless of how complex the predicates are, which functions and constants are used in the predicates, etc.

Second, increased dimensionality makes it harder for semantic windows to scale. For example, when the interest of the SDSS user involves both an ellipse pattern based on the location and a log pattern based on brightness, it will be more difficult for the system and the user to handle multiple semantic windows in data exploration (if such patterns can be predefined). In contrast, as the dimensionality increases, explore-by-example can keep the same user interface for data exploration and handles increased complexity via its learning algorithm "behind the scenes."

While prior work on explore-by-example has employed active learning [24], a main issue is the large number of labeled examples needed to achieve high accuracy. For instance, in the use case studied in [24], 300-500 labeled examples were needed to reach 80% accuracy, which is undesirable in many applications. This problem, referred to as *slow convergence*, is exacerbated when the user interest covers only a small fraction of the database (i.e., low selectivity) or the number of the attributes chosen for exploration is large (i.e., high dimensionality). Another major issue is that most existing active learning methods assume the labels provided by the user to be uncorrupted [14]. In practice, label noise often occurs in the user labeling process due to two main reasons. First, the user does not fully understand his/her interest and may have trouble classifying some examples correctly, especially those close to the decision boundary. Second, some examples are mislabeled accidentally. As stated in the survey [31], the occurrence of label noise may lead to (i) deterioration of prediction performance, (ii) requirement of more labeled examples and more complex learned models, and (iii) distortion of observed class distribution. Although a few pioneering efforts have considered the label noise problem in the context of active learning [3, 14, 41, 48, 71, 82, 83], they often focus on random classification noise. It is challenging to deal with more general and realistic label noise models.

In this work, we take a new approach to active learning-based database exploration. Instead of improving active learning in isolation from the database, we treat it as an internal module of the database system and ask the question: *what query and data properties from the database can we leverage to address the slow convergence problem and the label noise problem?* Based on the common properties that we observed in query traces from the SDSS [68] and a car database (described more in Section 6), we can indeed design new techniques that overcome or alleviate the slow convergence and label noise problems. Some of the key query and data properties we explore include:

Subspatial Convexity: Consider the database attributes as dimensions of a data space $\mathcal{D}$ and map the tuples to the space based on the values of their attributes. Then all the tuples that match the user interest form the positive region in $\mathcal{D}$; others form the negative region. We observe that in some lower-dimensional subspaces of $\mathcal{D}$, the projected positive or negative region is a convex object. For example, the SDSS query trace [68] includes 116 predicates, among which 107 predicates define a convex positive region in the subspace formed by the attributes used in the predicate, and 3 predicates define a convex negative region in their subspaces. For the car database, the 70 predicates defined on numerical attributes all form a convex positive region. Figure 2 shows a range of predicates from these two databases and their positive and negative regions.

Conjunctivity: Conjunctive queries are a major class of database queries that have been used in numerous applications. For data exploration, we are interested in the "conjunctive" property of the set of predicates that characterize the user interest. Among 45 queries provided by SDSS [68], 40 queries are conjunctive queries. For the car database, all user queries use the conjunctive property.

In this paper, we bring the subspatial convexity and conjunctivity of database queries, treated as true (yet unknown) user interest queries, to bear on the design of new algorithms and optimizations for active learning-based database exploration. These techniques allow the database system to overcome fundamental limitations of traditional active learning, i.e., the slow convergence problem when data exploration is performed with high dimensionality and low selectivity of the user interest query, and the label noise problem when the labels given by the user are corrupted. More specifically, our paper makes the following contributions.

1. Dual-Space Model (Section 3): By leveraging the subspatial convex property, we propose a new "dual-space model" (**DSM**) that builds not only a classification model, $F_{\mathcal{V}}$, from labeled examples, but also a polytope model of the data space, $F_{\mathcal{D}}$. On one hand, active learning theory improves $F_{\mathcal{V}}$ by choosing the next example that enables reduction of the version space $\mathcal{V}$ (the space of all classification models consistent with labeled data). On the other hand, our polytope model offers a more direct description of the data space $\mathcal{D}$ including the areas known to be positive, areas known to be negative, and areas with unknown labels. We use both models to predict unlabeled examples and choose the best example to label next. In addition, **DSM** allows us to prove exact and approximate lower bounds on the model accuracy in terms of the F1-score. To the best of our knowledge, this is the first provable result of an active learning method in the F1-score, while existing learning theory offers bounds only on classification errors [16, 27, 35–37], which treats positive and negative classes equally and hence does not suit queries of very low selectivity.

2. High-dimensional Exploration (Section 4): When the user interest involves a large number of attributes, by leveraging the conjunctive and subspatial convexity properties of user interest queries, we factorize a high-dimensional data space into low-dimensional subspaces, in some of which the projections of user positive or negative regions are convex. Our dual-space model with factorization, **DSM**$_F$, runs the polytope model in each subspace where the convex property holds. We formally define the class of queries that **DSM**$_F$ supports, the decision function it utilizes, and prove that it achieves a better lower bound of the F1-score than **DSM** without factorization.

3. Learning with Label Noise (Section 5): To handle noisy labels with minimum labeled examples, we develop a new active learner called Robust Dual-Space Model (**RDSM**), which is a seamless integration of a recently proposed automatic noise distillation method (*auto-cleansing*) [17] and our proposed **DSM**-based denoising methods. Regarding the user labeling process, we propose an adaptive method to strategically trade off the savings in user labeling effort against the risk of introducing more label noise. We also extend **RDSM** with factorization (**RDSM**$_F$) for data exploration in high dimensional space, and design optimization strategies, such as distributed computing and hyperparameter tuning, to improve our system in accuracy, interactive performance, and generality.

4. Evaluation (Section 6): We evaluated our system using two real datasets. The SDSS dataset [69] includes 190M tuples, for which the user interests are selective and their decision boundaries present varied complexity for detection. Without knowing these queries in advance, **DSM**$_F$ can achieve the F1-score of 95% within 10 labeled examples if the decision boundary B falls in a sparse region, and otherwise requires 40-160 labeled examples for up to 6D queries while maintaining per-iteration time within 1-2 seconds. For these queries, **DSM**$_F$ significantly outperforms learning methods including Active Learning (AL) [13,28] and Active Search [33], as well as recent explore-by-example systems, Aide [23, 24] and LifeJoin [19].

Our user study of a 5622-tuple car database validates the subspatial convexity and conjunctivity properties of user interest queries. It also shows the benefits of **DSM**$_F$ (a median of 10 labeled examples to reach high accuracy) over manual exploration (where the user wrote 12 queries and reviewed 98 tuples as the median), as well as over other AL methods.

In the face of label noise, evaluation using SDSS and car datasets shows the following results: (1) Our algorithm substantially outperforms alternative algorithms in accuracy for all label noise levels while maintaining per-iteration time within 1-3 seconds. In particular, compared to traditional active learning, our algorithm achieves 0.88x-22.14x higher accuracy for SDSS 4D-6D queries, and 0.14x-3.72x higher accuracy for the car queries obtained in our user study. (2) When the noise rate increases, while all algorithms decrease performance, our algorithm reduces at the slowest pace. (3) All of our techniques, *auto-cleansing*, **RDSM**, and factorization, contribute to alleviate the label noise problem. (4) Our distributed computing method improves the time per iteration from 20-30 seconds without parallelism to a few seconds, which improves the scalability of our system.

2 Background

In this section, we review our design of an explore-by-example system and present background on active learning.

2.1 System Overview

Our data exploration system is depicted in Figure 1. In the following discussion, we use the query Q_F, "$price \leq 30000$ and $length * width > 10.1$," extracted from our Car user study to exemplify the main concepts and core workflow. Suppose that the user has an implicit exploration goal Q_I: purchasing a spacious car at an affordable price. Since there

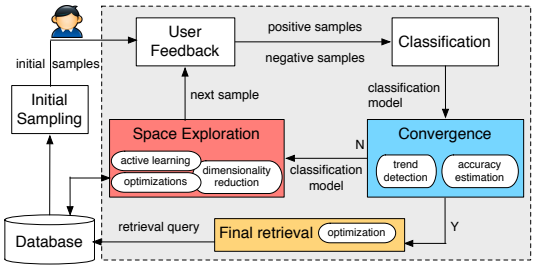


Fig. 1: System architecture for explore by example.

is a tradeoff between spaciousness and price, the user cannot come up with the query Q_F directly, but when presented with a concrete example, the user can determine if this car meets the expectations or not.

Initial Sampling. When exploring the database, the user is first presented with the database schema for browsing. Based on his/her best understanding of the implicit exploration goal, the user chooses an initial set of attributes, $\{A_i\}$, $i = 1, \ldots, d$, from the database. They form a superset of the relevant attributes that will eventually be discovered by the system. Let us consider the projection of the underlying database table[2] to $\{A_i\}$, and pivot the projected table such that each A_i becomes a dimension and the projected tuples are mapped to points in this d-dimensional space – the resulting space is called a *data space* where the user exploration will take place. For our example Q_I above, the data space involves attributes $price$, $length$, and $width$, as well as other irrelevant attributes that will be eventually filtered out in the data exploration process[3].

To bootstrap data exploration, the user is asked to give a positive example and a negative example. If the user does not have such examples, the system can run initial sampling over the data space to help find such examples. Since the initial sampling problem has been studied before [23, 49], our work in this paper focuses on data exploration after such initial examples are identified.

Iterative Learning and Exploration starts with a positive example and a negative example, which form the labeled dataset. In each iteration, four steps are executed in order, resulting in a new example labeled, as shown in Figure 1.

1. *Classification*. The labeled dataset is used to train a user interest model. Here, training is fast due to the small number of labeled examples.

2. *Convergence Test*. Our system assesses the current model to decide whether more iterations are needed. This unique feature, which was not available in earlier data exploration systems [23, 24], is enabled by our advanced data space model. We present the main idea below while deferring the details to Section 3. At the high level, our model builds a partitioning of the data space based on all available labeled examples. The data space is divided into three disjoint regions: the positive region, negative region, and unknown region. With the increase of labeled examples, both positive and negative regions expand while the uncertain region shrinks. With sufficient data, the uncertain region will eventually become empty and the positive region will converge to the user interest region. Based on the volumes of these regions, a lower bound of the F1-score can be obtained as the estimated accuracy for the classification model. If the estimated accuracy exceeds the user-defined threshold, the exploration will be terminated automatically by our system.

3. *Space Exploration*. The trained model is explored to retrieve a new example for user labeling which, among all unlabeled examples, best expedites the model convergence.

4. *Request of User Feedback*. The user labels the new example as positive or negative. Such feedback can be collected explicitly through a graphical interface as in our past work [22]. The newly labeled example is added to the labeled dataset. For our example Q_I, the user provides feedback based on the price, length, and width of a specific car in display. Note that although our final learned query is written as "Q_F: $price \leq 30000$ and $length * width > 10.1$," the user does not need to do such calculation. Instead, the user just looks at a particular car and decides whether it is a good tradeoff between space and price. It is our system that will learn the query Q_F to enclose all positive examples in a user interest region; that is, the boundary of the region is a learned concept and does not necessarily reflect how the user makes a decision. Our system also has an extension for the user to label the example for its subspaces, which we will defer to Section 4 when we introduce factorization.

The above process repeats until the model reaches a user-specified accuracy level or the maximum level of iterations (labeled examples) that the user can tolerate. At this point, the model for the positive class is translated to a user interest query, which encodes the model in a UDF and retrieves from the database all the tuples that the model classifies as relevant. In the car query example, the implicit goal Q_I is eventually characterized by the output of Q_F.

2.2 Active Learning for Data Exploration

The problem of seeking the next example for labeling from a database of unlabeled tuples is closely related to active learning (AL). Its recent results are surveyed in [66]. Below, we summarize the results most relevant to our work.

Pool-Based Sampling. Many real-world problems fit the following scenario: there is a small set of labeled data $\mathcal{L}$ and a large pool of unlabeled data $\mathcal{U}$ available. In active learning,

[2] This work considers a dataset that consists of a single table. It is left to our future work to study the extension for join queries.

[3] Feature selection to filter irrelevant attributes is addressed in our previous paper [40]. Due to space constraints, in this paper we assume that feature selection is performed the same as [40] and focus on other data exploration issues.

an example is chosen from the pool $\mathcal{U}$ in a greedy fashion, according to a utility measure used to evaluate all instances in the pool (or, if $\mathcal{U}$ is large, a subsample thereof). In our setting of database exploration, the labeled data $\mathcal{L}$ is what the user has provided thus far. The pool $\mathcal{U}$ is a subsample of size m of the unlabeled part of the database. The utility measure depends on the classifier in use, as discussed below.

Classification Model. Previous explore-by-example systems [23, 24] used decision trees to build a classification model. It works well if the user interest pattern is a hyper-rectangle in the data space, whereas real-world applications may use more complex predicates. To support higher complexity, our system uses more powerful classifiers such as Support Vector Machines (SVM) or Gradient Boosting. The new techniques proposed in our work do **not** depend on the specific classifier; they can work with most existing classifiers. But the implementation of active learning does depend on the classifier in use. For ease of composition, in this paper we use SVM as an example classifier.

Active Learning Strategies have been studied to determine which examples should be labeled next to expedite model convergence, which are also called "*query strategies*" in the AL literature. Some of the most popular strategies [66] include: *Uncertainty sampling* queries the examples for which the current model is least certain of its prediction. For binary classification, the examples selected are those closest to the decision boundary. *Query by committee* maintains a committee of models trained on the current labeled dataset, allows all models to vote on the predictions for unlabeled data, and queries the examples on which the committee disagrees the most. *Expected error reduction* queries the examples that would minimize the model's generalization error. *Variance reduction* selects the examples that would minimize output variance. *Density-weighted methods* take into account not only the most uncertain examples but also the most representative ones of the underlying distribution (e.g. in dense regions) in the query strategy.

It is important to note that our work aims to develop a new approach that overcomes the slow convergence observed for many of the above AL methods [57,66] by adding new insights from the data and user interest patterns in database exploration. Our solution combines a new data space model and any AL method to work in parallel, where the data space model is orthogonal to the AL method used. As such, our approach is tailored for database exploration while being general with respect to the AL method in use.

More specifically, our implementation of the AL method is uncertainty sampling due to its wide use [66] and high efficiency in execution. A final note is that a formal description of uncertainty sampling is through the notion of *version space*, which is the space of all configurations of the classification model consistent with labeled data. For SVM classifiers, for example, the version space includes all possible decision boundaries of the SVM that lead to correct classification of the training examples. Uncertainty sampling is known to be an approximation of an optimal algorithm that bisects the version space with each selected example [73]. For the above reason, we call the learning algorithm in uncertainty sampling *version space*-based as they are designed to reduce the version space, and we will further augment it with a new *data space* model.

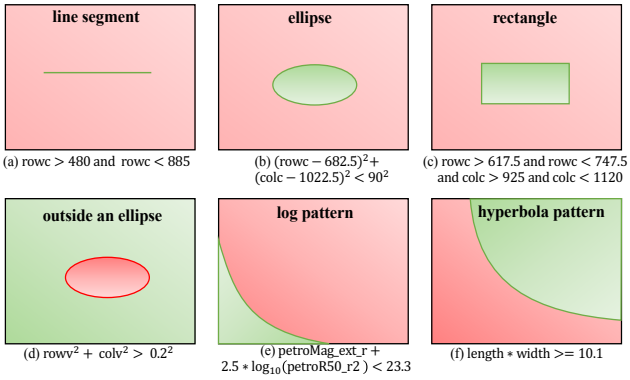


Fig. 2: Positive region (green) and negative region (red) of 6 example predicates from SDSS queries, where one of the two regions is convex

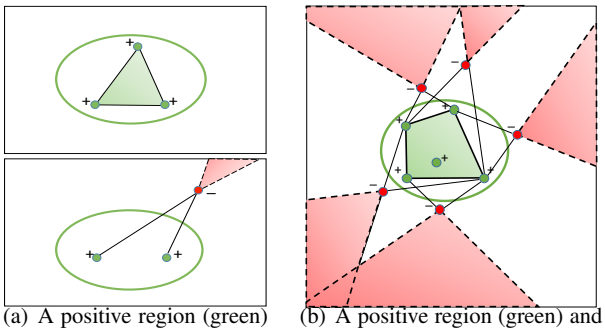
3 Dual-Space Model

With the goal to provide a service for database exploration, our work takes a new, a database-centric approach to tackle the slow convergence problem of existing active learning methods. We observe from existing query traces that in lower-dimensional subspaces, the projected positive or negative region of user interest query is often a convex object. Figure 2 shows 6 examples from the SDSS query trace [68], where 107 out of 116 predicates used define a convex positive region (colored in green) in the subspace formed by the attributes in the predicate, and 3 predicates define a convex negative region (colored in red) in their subspaces.

In this section, we utilize such subspatial convexity and introduce a dual-space (data and version space) model, which enables improved accuracy and provable lower bounds on the model accuracy. We begin with the simple case that the convex property holds for the user interest query over the entire data space $\mathcal{D}$, without factorization, and defer the extension to subspaces to Section 4. We refer to this class of queries as "*convex pattern queries*", $\mathbf{Q}_c \equiv \mathbf{Q}_c^+ \cup \mathbf{Q}_c^-$, where $\mathbf{Q}_c^+$ and $\mathbf{Q}_c^-$ denote those queries whose positive and negative regions are convex, respectively.

3.1 A Polytope Model in Data Space

The key idea behind our *data space* model is that at each iteration we use all available labeled examples to build a partitioning of the data space. It divides the data space into the



(a) A positive region (green) with 3 examples, and a negative region (red) with 1 negative and 2 positive examples

(b) A positive region (green) and 5 negative regions (red) built from 5 positive examples and 5 negative examples.

Fig. 3: Positive and negative regions in the polytope model.

positive region (any point inside which is known to be positive), the *negative region* (any point inside which is known to be negative) and the *uncertain region*. As more examples are labeled, we have more knowledge about the uncertain region, so part of it will be converted to either the positive or the negative region in later iterations. Eventually, with enough training data, the uncertain region becomes empty, and the positive region converges to the query region.

For simplicity, we begin with queries whose positive region is convex ($\mathbf{Q}_c^+$). When the query region Q is convex, any point on the line segment between two points $\boldsymbol{x}_1 \in Q$ and $\boldsymbol{x}_2 \in Q$ is also in Q. Then we have the following:

Definition 1 (Positive Region) Denote the examples that have been labeled as "positive" as $L^+ = \{e_i^+ \,|\, i = 1, \ldots, n^+\}$. The convex hull of L^+ is known to be the smallest convex set that contains L^+ [46] and is called the positive region, denoted as R^+.

It is known that the convex hull of a finite number of points is a *convex polytope* [34]. For example, the green triangle in Figure 3(a) and the green tetragon in Figure 3(b) are the positive regions formed by three and four positive examples, respectively, which are an approximation of the query region marked by the green ellipse. We can prove the following property of the positive region for convex queries, assuming that user labels are consistent with her interest:

Proposition 1 *All points in the positive region R^+ are positive.*

All the proofs in this section and the next are left to our tech report [39] due to space limitations.

Definition 2 (Negative Region) For a negative example $\boldsymbol{e}_i^-$, we can define a corresponding negative region R_i^- such that the line segment connecting any point $\boldsymbol{x} \in R_i^-$ and $\boldsymbol{e}_i^-$ does not overlap with the positive region R^+, but the ray that starts from $\boldsymbol{x} \in R_i^-$ and passes through $\boldsymbol{e}_i^-$ will overlap with R^+. More formally, $R_i^- = \{\boldsymbol{x} | \overline{\boldsymbol{x}\boldsymbol{e}_i^-} \cap R^+ = \emptyset \wedge \overrightarrow{\boldsymbol{x}\boldsymbol{e}_i^-} \cap R^+ \neq \emptyset\}$. Given n^- negative examples, the negative region R^- is the union of the negative region for each negative example, i.e., $R^- = \bigcup_{i=1}^{n^-} R_i^-$.

From the definition, we know that R_i^- is a convex cone generated by the conical combination of the vectors from the positive examples to the given negative example, i.e., $\overrightarrow{e_j^+ e_i^-}$ $(j = 1, \ldots, n^+)$. The red triangle in Figure 3(a) depicts such a convex cone. However, the union of $R_i^-, i = 1, 2, \ldots$ is non-convex. For example, the union of the five red polygons in Figure 3(b) is non-convex. Given more labeled examples, the result of the union will be more accurate for approximating the true negative region, which is outside the ellipse. We prove the following property of the negative region:

Proposition 2 *All points in the negative region R^- are negative.*

Definition 3 (Uncertain Region) Denote the data space as $\mathbb{R}^d$, the uncertain region $R^u = \mathbb{R}^d - R^+ - R^-$.

Formally, the polytope model makes a decision about an example $\boldsymbol{x}$ based on the following decision function, which takes values in $\{-1, 0, 1\}$ corresponding to R^-, R^u, and R^+ defined above:

$$F_{\mathcal{D}}(\boldsymbol{x}) = 1 \cdot \mathbb{1}(\boldsymbol{x} \in R^+) - 1 \cdot \mathbb{1}(\boldsymbol{x} \in R^-). \quad (1)$$

Our work also supports the case that the negative region of the query is convex ($\mathbf{Q}_c^-$). We can simply switch the above definitions such that we build a convex polytope for the negative region, and a union of convex cones for the positive region, one for each positive example. We also offer a test at the beginning of the data exploration process to choose between two polytope models, $Q \in \mathbf{Q}_c^+$ or $Q \in \mathbf{Q}_c^-$. The details are deferred to Section 4 where we offer a test procedure for all the assumptions made in the work.

Three-Set Metric. Our goal is not only to provide a new data space model, as described above, but also to design a new learning algorithm that enables a provable bound on the model accuracy. As stated before, our accuracy measure is the F1-score. Formally, the F1-score is evaluated on a *test set* $D_{test} = \{(\boldsymbol{x}_i, y_i)\}$, where $\boldsymbol{x}_i$ denotes a database object and y_i denotes its label according to the classification model. Then the F1-score is defined as, F1-score $= 2 \cdot \frac{precision \cdot recall}{precision + recall}$, where *precision* is the fraction of points returned by the model from D_{test} that are truly positive, and *recall* is the fraction of positive points in D_{test} that are actually returned by the model.

However, capturing the F1-score in our data exploration procedure is difficult because we do not have such a labeled test set, D_{test}, available. We cannot afford to ask the user to label more to produce one since the user labor

is an important concern. To bound the F1-score with limited labeled data, our idea is to run our polytope model, $F_{\mathcal{D}} : \mathbb{R}^d \rightarrow \{-1, 0, 1\}$, on an evaluation set. We define the *evaluation set* D_{eval} as the projection of D_{test} without labels y_i's. Then for each data point in D_{eval}, depending on which region it falls into, D_{eval} can be partitioned into three sets accordingly, denoted as D^+, D^- and D^u. We can compute a metric from the number of data points in the three sets, as follows.

Definition 4 (Three-Set Metric) Denote $D^+ = D_{eval} \cap R^+$, $D^- = D_{eval} \cap R^-$, $D^u = D_{eval} \cap R^u$, and $|S|$ means the size of set S. At a specific iteration of exploration, the three-set metric is defined to be $\frac{|D^+|}{|D^+|+|D^u|}$.

We will prove shortly that this metric is a lower bound of the F1-score evaluated on D_{test}. As more labeled examples are provided, data points will be moved from D^u to either D^+ or D^-. Eventually, with enough training data the uncertain set shrinks to an empty set and the Three-Set Metric rises to 100%, reaching convergence.

3.2 Dual-Space Model and Algorithm

We now propose a new algorithm for interactive data exploration by exploiting models from two spaces, including our data space model $F_{\mathcal{D}}$ with the Three-Set Metric, and a classification model $F_{\mathcal{V}}$ with uncertainty sampling derived from the version space.

We first define the dual-space model (**DSM**) by considering two functionalities of a model for data exploration.

(1) *Prediction*: Given an unlabeled example, **DSM** predicts the class label first based on the data space model $F_{\mathcal{D}}$. If the example falls in the positive region R^+, it is predicted to be positive; if it falls in the negative region R^-, it is predicted to be negative. If the example falls in the unknown region R^u, then the classification model $F_{\mathcal{V}}$ is used to predict the example to be positive or negative.

(2) *Sampling*: In the AL framework, the model is also used to guide the choice of the next example for labeling to expedite model convergence. As discussed in Section 2, active learning can use an uncertainty sampling method, $\mathcal{S}_{\mathcal{V}}$, for a given classification model to choose the next example. However, direct application of uncertainty sampling to our dual-space model raises a problem: the example chosen by $\mathcal{S}_{\mathcal{V}}$ may fall in the known positive or negative region of our data space model $F_{\mathcal{D}}$, hence wasting computing resources on such examples. In our work, we propose a new sampling method that is restricted to the unknown region of our data space model, denoted as $\mathcal{S}_{\mathcal{D}}$. However, if we sample only from our data space model, we may not get a representative sample to train the classifier. Therefore, we use a sampling ratio, γ, to alternate between the sampling methods, $\mathcal{S}_{\mathcal{D}}$ and $\mathcal{S}_{\mathcal{V}}$. For instance, when $\gamma = 1/3$, in each iteration we use $\mathcal{S}_{\mathcal{D}}$ with probability 1/3 and use $\mathcal{S}_{\mathcal{V}}$ otherwise.

Algorithm 1 Dual-Space Algorithm for Convex Queries

Input: database D, initial labeled data set D_0, accuracy threshold λ, sampling ratio γ, unlabeled pool size m

1: $D_{labeled} \leftarrow D_0$, $D_{unlabeled} \leftarrow D \setminus D_0$
2: $R^+ \leftarrow \emptyset$, $R^- \leftarrow \emptyset$
3: $D^+ \leftarrow \emptyset$, $D^- \leftarrow \emptyset$, $D^u \leftarrow D$
4: $\mathbf{D} \leftarrow (D^+, D^-, D^u)$
5: $D_{lu} \leftarrow D_0$, $D_{ld} \leftarrow \emptyset$
6: **repeat**
  // building the Dual-Space model:
7: **for** $\boldsymbol{x} \in D_{lu}$ **do**
8: $(R^+, R^-) \leftarrow$ `updateRegion`$(R^+, R^-, \boldsymbol{x})$
9: $\mathbf{D} \leftarrow$ `threeSets`$(R^+, R^-, \mathbf{D})$
10: $accu \leftarrow$ `threeSetMetric`$(\mathbf{D})$
11: $D_{labeled}$.`append`$(D_{lu} \cup D_{ld})$
12: $D_{unlabeled}$.`remove`$(D_{lu} \cup D_{ld})$
13: $classifier \leftarrow$ `trainClassifier`$(D_{labeled})$
  // uncertainty sampling:
14: $D_{lu} \leftarrow \emptyset$, $D_{ld} \leftarrow \emptyset$
15: **if** $rand() \leq \gamma$ **then**
16: $pool \leftarrow$ `subsample`(D^u, m)
17: $\boldsymbol{x} \leftarrow$ `getNextToLabel`$(pool, classifier)$
18: $D_{lu} \leftarrow$ `getUserLabel`$(\boldsymbol{x})$
19: **else**
20: $pool \leftarrow$ `subsample`$(D_{unlabeled}, m)$
21: $\boldsymbol{x} \leftarrow$ `getNextToLabel`$(pool, classifier)$
22: **if** $\boldsymbol{x} \in R^+$ **then**
23: $D_{ld} \leftarrow (\boldsymbol{x}, 1)$ // auto-labeling
24: **else if** $\boldsymbol{x} \in R^-$ **then**
25: $D_{ld} \leftarrow (\boldsymbol{x}, -1)$ // auto-labeling
26: **else**
27: $D_{lu} \leftarrow$ `getUserLabel`$(\boldsymbol{x})$
28: **until** $accu \geq \lambda$ **or** `reachedMaxNum()`
29: `finalRetrieval`$(D, (R^+, R^-), classifier)$

Now we present the full algorithm for interactive data exploration, as shown in Algorithm 1. The input is the database D, the initial labeled data set D_0 (a positive example $\boldsymbol{x}^+$, a negative example $\boldsymbol{x}^-$), a user-defined accuracy threshold λ, the sampling ratio γ, and the size of unlabeled pool m. First, we assign the initial labeled and unlabeled datasets (line 1). In particular, we assume the evaluation set D_{eval} to be the entire database D, which can be replaced with a sample of D if D is too large. More details can be found in Section 3.4. Then we initialize data structures, including setting the data space model (R^+, R^-) as empty sets, setting the partitions $\mathbf{D}$, and assigning the user-labeled set D_{lu} and the **DSM**-labeled set D_{ld} as empty sets. The algorithm next goes through iterative exploration.

Lines 7-13 update our **DSM** model. The data space model, R^+ and R^-, is updated with the newly labeled example(s) by the user (lines 7-8). This step incrementally updates our convex polytope for R^+ and the union of convex polytopes for R^- based on computational geometry [9]. Afterwards, the corresponding partitions $\mathbf{D}$ are incrementally updated (line 9). In this step, some examples are removed

from the unknown partition D^u and placed to the positive partition D^+ or the negative partition D^-. The accuracy is then estimated using the Three-Set Metric. We also keep track of the labeled and unlabeled examples using $D_{labeled}$ and $D_{unlabeled}$. We use the labeled examples to train a classifier, that is, the version space model in our **DSM**.

Then lines 14-27 implement uncertainty sampling using **DSM**. With probability γ, we perform uncertainty sampling from a pool restricted to the unknown partition of the evaluation set, D^u. Then the example chosen randomly is labeled by the user. With probability $1-\gamma$, we perform uncertainty sampling from a pool that is a subsample of all unlabeled examples in the database. Then the example chosen by uncertainty sampling is first run through our data space model, (R^+, R^-), to see if it falls in the positive or negative region and hence can be labeled directly by the model (we call this functionality *auto-labeling* from **DSM**). Otherwise, it will be labeled by the user. Note that auto-labeling can reduce the number of user-labeled examples required to reach high accuracy, achieving faster convergence. Due to auto-labeling, our labeled set includes both user-labeled examples and **DSM**-labeled examples.

Then the algorithm repeats until it has met the user accuracy requirement based on the lower bound offered by our Three-Set Metric, or reached the maximum of iterations allowed (line 28). Finally, we run the **DSM** model over the database to retrieve all tuples predicated to be positive.

3.3 Lower Bounds of the F1-score

Given how the **DSM** algorithm works, we now present formal results on the model accuracy achieved by **DSM**.

Exact Lower Bound. We begin with an exact lower bound of our accuracy measure, the F1-score.

Theorem 1 *The Three-Set Metric evaluated on D_{eval} is a lower bound of the F1-score if* **DSM** *is evaluated on D_{test}.*

The lower bound of **DSM** has several features. First, it is an *exact* lower bound throughout the exploration process for any evaluation set D_{eval}. Second, the metric is *monotonic* in the sense that points in the uncertain region D^u can be moved to the positive or negative region later, but not vice versa, and the metric goes to 1 when $D^u = \emptyset$. If the metric is above the desired accuracy threshold at some iteration, it is guaranteed to be greater than the threshold in later iterations, so we can safely stop the exploration.

Approximate Lower Bound. When D_{eval} is too large, we employ a sampling method to reduce the time to evaluate the Three-Set Metric. Let p and q be the true proportions of the positive and negative examples in D_{eval}, i.e., $p = \frac{|D^+|}{|D_{eval}|}$ and $q = \frac{|D^-|}{|D_{eval}|}$. Then the Three-Set Metric is $b = \frac{p}{1-q}$. Let $\hat{p}$ and $\hat{q}$ be the observed proportions of the positive and negative examples in a random draw of n examples from D_{eval}, and let $X_n = \frac{\hat{p}}{1-\hat{q}}$. Our goal is to find the smallest sample size n such that the error of the estimation X_n from the exact Three-Set Metric is less than δ with probability no less than λ. That is, $\Pr(|X_n - b| < \delta) \geq \lambda$.

The following theorem helps find the lower bound of n.

Theorem 2 $sup_\epsilon \| Pr(\sqrt{n}|X_n - b| < \epsilon) - \left(2\Phi(\frac{\epsilon(1-q)}{\sqrt{p(1-p-q)}}) - 1 \right) \| = O(1/\sqrt{n})$ *for any ϵ, where Φ is the cumulative distribution function of the standard Normal distribution.*

With the theorem, we can approximate the sample size by

$$2\Phi(\frac{\sqrt{n}\delta(1-q)}{\sqrt{p(1-p-q)}}) - 1 \geq \lambda.$$

Since $p(1-p-q)/(1-q)^2 \leq 1/4$, it is sufficient for n to satisfy $2\Phi(2\sqrt{n}\delta) - 1 \geq \lambda$ and therefore $n \geq \left(\Phi^{-1}\left(\frac{\lambda+1}{2}\right)\right)^2 / (4\delta^2)$.

3.4 Optimization

We improve time efficiency by designing new sampling methods and leveraging distributed computing. We begin by considering the sampling procedures in Algorithm 1. We present their implementation and optimization as follows.

Sampling for creating the evaluation set (OPT1). In line 1 of Algorithm 1, we assign the evaluation set D_{eval} to be the entire database D, to develop a lower bound of the **DSM** model. Ideally, we want D_{eval} to be as large as possible. However, for efficiency, we can only afford to have a memory-resident sample. The analysis in Theorem 2 indicates that we can choose a sample of size n from the database, and achieve an approximate lower bound of the F1-score of our **DSM** algorithm when evaluated on the database. The sample, denoted as $\tilde{D}_{eval}$, will expedite line 9 of Algorithm 1. For example, the SDSS database used in our experiments contains 190 million tuples. Denote the true lower bound as b. When $n = 50k$, our approximate lower bound $\hat{b}$ approximates b by $\epsilon \leq .005$ with probability 0.975 or higher. When $n = 1.9$ million, $\hat{b}$ approximates b by $\epsilon \leq .001$ with probability 0.994 or higher.

Sampling for pool-based active learning. Algorithm 1 contains two `subsample` procedures for pool-based *uncertainty sampling*, i.e., choosing the most uncertain example from the pool for labeling next. The `subsample` routine in line 16 creates a pool of unlabeled examples from the uncertain partition of the evaluation set, D^u, which is memory-resident. This can be implemented using reservoir sampling. The `subsample` routine in line 20, however, creates a pool,

from the unlabeled portion of the entire database, which is large and not materialized. We devise a sampling-based optimization to avoid scanning the unlabeled database in each iteration. Due to space constraints, the interested reader is referred to [39] for details.

Distributed computing (OPT2). Given a large dataset, it is time-consuming to update the data space model by building the three (positive, negative, and uncertain) partitions for the entire dataset every time a user-labeled example is received. We apply an open-source system Ray[4] to parallelize our code, by splitting the entire dataset into several subsets evenly and building the partitions within each subset. As such, we divide the partition-building task for the entire dataset into multiple independent sub-tasks, reducing the time per iteration from 20-30 seconds to a few seconds.

4 Factorized Dual-Space Model

Although **DSM** can reduce user labeling effort and offer better accuracy than traditional active learning, increased dimensionality will make the volume of its uncertain region grow fast and degrade its performance. To handle this issue, we leverage the property of *conjunctive queries* to factorize a high-dimensional data space into a set of low-dimensional spaces. By running the polytope model in each low-dimensional space, we can "simulate" **DSM** in the high-dimensional space with improved performance.

This extension, denoted as **DSM**$_F$, may require the user to label examples in some subspaces. Revisit our example query Q_I from Section 2. We observe that often when a user labels an example, the decision making process can be broken into a set of smaller questions, whose answers are then combined, through conjunctivity, to derive the final answer. In the example of Q_I, the user asks two smaller questions: one concerns $price$ (P) and the other concerns $length$ and $width$ (LW) for spaciousness. If a car is labeled positive, then its partial labels on P and LW are inferred to be positive. Otherwise, the user will be asked which values are not satisfactory. For example, if the user thinks the price is too high, through clicking a "dislike" button in the GUI, the price P will receive a negative partial label for the given car. As the user has gone through thinking to derive the overall negative label, specifying a subset of attributes that lead to the negative decision will not incur much more work.

4.1 Factorization

Formally, factorization concerns a user interest query Q that is defined on an attribute set $\boldsymbol{A}$ of size d and can be written in the conjunctive form, $Q_1 \wedge \ldots \wedge Q_m$. Each Q_i, $i \in [1 \ldots m]$, uses a subset of attributes $\boldsymbol{A}_i = \{A_{i1}, \cdots, A_{id_i}\}$. The family of attribute sets, $\boldsymbol{A} = (\boldsymbol{A}_1, \cdots, \boldsymbol{A}_m)$, is pairwise disjoint with $d = \sum_{i=1}^{m} d_i$, and called the *factorization structure*.

In practice, we can derive the factorization structure from available data in a database. If a query trace is available to show how attributes are used in predicates, e.g., individually or in a pair, we can run a simple algorithm over the query trace: Initially assign each attribute to its own partition. As the query trace is scanned, two partitions are merged if a predicate uses attributes from the two partitions. At the end, all the remaining partitions become the factorization structure $\boldsymbol{A}$. Even if a query trace is not available, it is still possible to work with domain experts and extract a factorization structure that reflects how attributes are used based on their semantics, e.g., $length$ and $width$ are often used in the same predicate to capture the spaciousness of the car.

DSC Queries. We next define a class of user interest queries that **DSM**$_F$ supports with benefits over active learning: that is, $\forall i = 1, \ldots, m$, either the positive region in the i^{th} subspace, defined as $\{\boldsymbol{x}_i \in \mathbb{R}^{|\boldsymbol{A}_i|} | Q_i(\boldsymbol{x}_i) > 0\}$, is convex (in the $\mathbf{Q}_c^+$ class), or the negative region, $\{\boldsymbol{x}_i \in \mathbb{R}^{|\boldsymbol{A}_i|} | Q_i(\boldsymbol{x}_i) < 0\}$ is convex (in $\mathbf{Q}_c^-$). We call such queries *Decomposable Subspatial Convex* (**DSC**) queries.

DSC allows us to combine a subspace whose positive region is convex with another subspace whose negative region is convex. However, the global query is **not** necessarily convex in either the positive or negative region. Figure 4(a) shows an example query, $Q = x^2 + y^2 > 0.2^2 \wedge 480 < z < 885$, which combines the ellipse pattern in Fig. 2(d), whose negative region is convex, with the line pattern in Fig. 2(a), whose positive region is convex. In the 3D space, the negative region (marked in red) is the union of the red cylinder in the middle of the figure and the red rectangles above and below the cylinder, while the positive region (in green) is the complement of it. Neither of the positive nor the negative region of Q is convex although it belongs to **DSC**.

p-**DSC** Queries. We also support a superset of **DSC** queries where the convex assumption holds only in some subspaces. We call this set *partial factorization* or p-**DSC**.

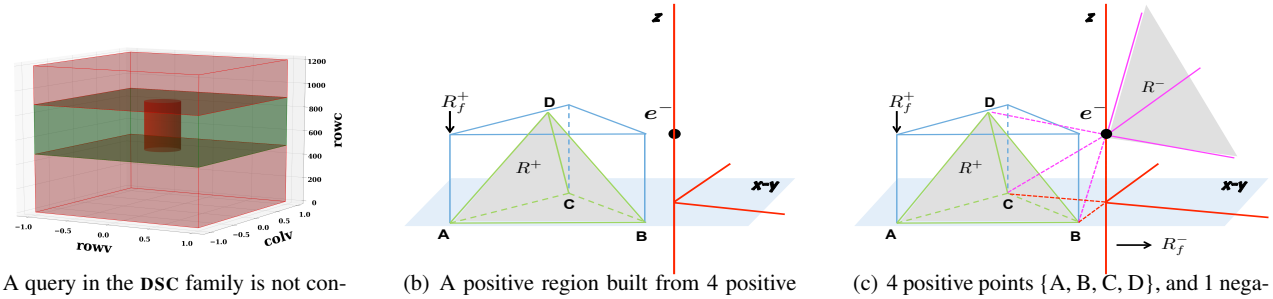
As a special case, if none of the subspaces permits the convexity property, we call such queries *Zero* **DSC** or 0-**DSC**.

4.2 Factorized **DSM** Model

We next extend our **DSM** model with factorization, denoted as **DSM**$_F$. The new model is composed of a factorized polytope model and a factorized classification model.

Factorized polytope model. Given the factorization structure, the polytope model runs in each subspace where the subspatial convexity holds. In the i^{th} subspace defined by $\boldsymbol{A}_i$, Q_i's positive and negative regions, denoted as R_i^+ and R_i^-, are built according to Definition 3.1 and 3.2, but

[4] Ray provides fast distributed computing at `https://ray.io/`



(a) A query in the **DSC** family is not convex in either the positive region (green) or the negative region (red).

(b) A positive region built from 4 positive points, {A, B, C, D}

(c) 4 positive points {A, B, C, D}, and 1 negative point e^- that is negative in the $(x\text{-}y)$ subspace and positive in the (z) subspace

Fig. 4: Illustration of factorization when 3D space $(x\text{-}y\text{-}z)$ is factorized into two subspaces $(x\text{-}y)$ and (z).

based on the "positive" and "negative" labels of projected examples for Q_i, as mentioned above. Then we build the positive and negative regions of Q from the positive and negative regions in the subspaces via the *conjunctive property*:

$$R_f^+ = \times_{i=1}^m R_i^+, \quad R_f^- = \left(\times_{i=1}^m (R_i^-)^c\right)^c \tag{2}$$

where $\times$ denotes the Cartesian product between two sets, and R^c denotes the complement of set R. Then the uncertain region of Q is, $R_f^u = \mathbb{R}^d - R_f^+ - R_f^-$.

Next we formally define the decision function for the polytope model with factorization. In each subspace defined on $\boldsymbol{A}_i$, let $F_{\boldsymbol{A}_i} : \mathbb{R}^{|\boldsymbol{A}_i|} \to \{-1, 0, 1\}$ be the decision function that divides the subspace into three disjoint regions corresponding to the negative, unknown, and positive regions, respectively. As in (Eq. 1),

$$F_{\boldsymbol{A}_i}(x) = 1 \cdot \mathbb{1}(\boldsymbol{x} \in R_i^+) - 1 \cdot \mathbb{1}(\boldsymbol{x} \in R_i^-). \tag{3}$$

For p-**DSC** queries, if the subspatial convexity does not hold in a subspace, the value of its decision function is set to zero.

The global decision function for the polytope model with factorization over the entire data space is then

$$F_{\mathcal{D}_f}(\boldsymbol{x}) = \min_{i=1}^m F_{\boldsymbol{A}_i}(\boldsymbol{x}_i) \tag{4}$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$, and $\boldsymbol{x}_i$ denotes the projection of $\boldsymbol{x}$ on the i^{th} subspace defined by $\boldsymbol{A}_i$.

Factorized classification model. To apply factorization to the classifier, we train a classifier in each subspace $h_i : \mathbb{R}^{|\boldsymbol{A}_i|} \to \{-1, 1\}, i = 1, \cdots, m$, and combine them to be a classifier $F_{\mathcal{V}_f} : \mathbb{R}^d \to \{-1, 1\}$ over the entire data space. According to the conjunctive property, we have

$$F_{\mathcal{V}_f} = \min_{i=1}^m h_i(\boldsymbol{x}_i) \tag{5}$$

We call $F_{\mathcal{V}_f}$ a factorized classifier.

We further propose a new sample acquisition method for $F_{\mathcal{V}_f}$. The sample acquisition criterion can be defined as:

$$\text{next\_sample} = \arg\min_x \left(\sum_{i=1}^m |d(h_i, x_i)| \right) \tag{6}$$

The example selected from the unlabeled data by the above criterion has, on average, the shortest distance to each decision boundary on subspaces. The reason behind using L_1 (Manhattan) distance metric to combine the distance information from all subspaces is that for the commonly used L_k norm, lower values of k are preferable in high dimensional space as shown in [4]. In addition, we observe from empirical results that when the sample acquisition methods are performed with high dimensionality, the examples selected by our criterion are usually closer to the true decision boundary, compared to the traditional uncertainty sampling.

Finally, consider the dual-space model. Let **DSM**$_F$ be the combination of **DSM** in §3.2 with the polytope data space model $F_{\mathcal{D}}$ replaced by the polytope model with factorization $F_{\mathcal{D}_f}$, and the classification model $F_{\mathcal{V}}$ replaced by the factorized classifier $F_{\mathcal{V}_f}$. Then the dual-space decision function of **DSM**$_F$, $H : \mathbb{R}^d \to \{-1, 1\}$, is:

$$H(\boldsymbol{x}) = \begin{cases} F_{\mathcal{D}_f}(\boldsymbol{x}), \; F_{\mathcal{D}_f}(\boldsymbol{x}) \neq 0 \\ F_{\mathcal{V}_f}(\boldsymbol{x}), \; \text{otherwise} \end{cases} \tag{7}$$

H is our final prediction model returned to approximate the user interest query. For 0-**DSC** queries, **DSM**$_F$ simply runs the classification model.

Illustration. Before presenting the formal results, we illustrate the intuition that factorization allows us to construct the positive and negative regions (R_f^+, R_f^-) as supersets of (R^+, R^-), hence reducing the unknown region and offering better accuracy. Figure 4(b) shows four positive points A, B, C, D when the 3D space $(x\text{-}y\text{-}z)$ is factorized into two subspaces $(x\text{-}y)$ and (z). It depicts the positive region R^+ as the pyramid shaded in grey and marked by the green lines. When we factorize R^+ into $(x\text{-}y)$ and (z) planes, we have R_{xy}^+ as a triangle marked ABC and R_z^+ as a line segment projected onto z. Then we construct R_f^+ from the triangle and the line segment based on Eq. 2, we obtain a prism marked by the blue lines, which is much bigger than R^+.

Figure 4(c) shows a negative point e^- and the negative region built for it. R^- is a convex cone shaded in grey and bounded by the solid purple rays emitting from e^-. When

e^- is projected onto (x-y), it is negative and defines a convex cone R^-_{xy} marked by the two solid red lines in the (x-y) plane. When e^- is projected onto z, it lies in the positive region. According to Eq. 2, the new negative region R^-_f extends the convex cone R^-_{xy} by all possible values of z, yielding a geometric shape enclosed by the three solid red lines in the figure. Again, the new R^-_f is much bigger than R^-.

Formal results. For both **DSC** and p-**DSC** queries, we offer formal results of the polytope model with factorization.

Proposition 3 *All points in the positive region R^+_f are positive, and $R^+ \subseteq R^+_f$ in each iteration of data exploration.*

Proposition 4 *All points in the negative region R^-_f are negative, and $R^- \subseteq R^-_f$ in each iteration of data exploration.*

Proposition 5 **DSM**$_F$ *improves the Three-Set Metric, the lower bound of the model accuracy in F1-score, over* **DSM**.

Proposition 3 and Proposition 4 state that the positive and negative regions constructed via factorization, (R^+_f, R^-_f), are a superset of (R^+, R^-) that the original **DSM** offers. Hence, the lower bound of the F1-score built from (R^+_f, R^-_f) is higher than that from (R^+, R^-) based on Def. 4.

Testing assumptions of DSM$_F$. Factorization relies on two assumptions: The first assumption is that we have a correct factorization structure, $\boldsymbol{A} = (\boldsymbol{A}_1, \cdots, \boldsymbol{A}_m)$, derived from a query trace or the database schema. The second assumption is that user interests take the form a conjunctive query (CQ). When either assumption is wrong, we may see conflicting examples in a polytope model for a specific subspace, i.e., a positive example labeled by the user appears in the negative region or vice versa. Our system uses a procedure to test online both assumptions behind **DSM**$_F$: At the beginning of data exploration, we build two polytope models for each subspace, one under the assumption that the positive region is convex, $\mathbf{Q}^+_c$, the other under the assumption that the negative region is convex, $\mathbf{Q}^-_c$. Over iterations, we count the number of conflicting examples of these two polytope models, and use a threshold, T, to turn off a polytope model if its count exceeds T. If both polytope models remain active in a given iteration, the one with the smaller count will be used; in the case of a tie, the model for $\mathbf{Q}^+_c$ will be used as more query patterns belong to this class. When both polytope models for a subspace are turned off, we use partial factorization until they are turned off for all subspaces, when we resort to the classifier.

5 Learning with Label Noise

Traditional active learning implicitly assumes that uncorrupted labels are provided by the user. However, in practice, label noise often occurs during the process of manual annotation. The user may not fully understand his/her interest and may have trouble labeling some examples correctly, especially for the ambiguous examples close to the underlying decision boundary. In other cases, the user may just mislabel some examples occasionally. Based on the observations of labeling behaviors in our user study, we assume in this work that the user exploration is overall moving toward a target interest region (e.g., not switching mind between two different regions), but the user may mislabel some examples. To improve the robustness of our AL-based data exploration system, we explore suitable label noise models to characterize the human annotator noise and propose denoising methods that are customized for our **DSM** model. We also analyze the effect of factorization on the label noise problem and provide several optimization methods to improve our system.

5.1 Label Noise Models and Noise Distillation

In this section, we present background on label noise models and some analysis about the data cleansing method used in our algorithm - *automatic noise distillation*.

5.1.1 Label Noise Models

There are mainly three types of label noise models [17, 31]: (i) In the random classification noise (RCN) model, the label flip probability (noise rate) is independent of the example and the corresponding true label (i.e., all examples have the same label flip probability). (ii) In the class-conditional random label noise (CCN) model, the label flip probability is independent of the example but dependent on the true label. The examples from the same class have the same label flip probability. (iii) The instance- and label-dependent label noise (ILN) model, where the label flip probability depends on both the example and true label, is the most general case of label noise, hence most complex yet broadly applicable. Compared to the RCN and CCN models, the ILN model has not been widely studied in the literature. In this work, we consider ILN for AL-based data exploration.

In particular, we use the assumption on label noise proposed by [17], shown as Assumption 1 below, to deal with a specific type of ILN. Let $\mathbf{X}$ denote the observation, Y the uncorrupted but unobserved label, and $\tilde{Y}$ the observed but noisy label. We assume that $(\mathbf{X}, Y, \tilde{Y}) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Y}$ are jointly distributed according to an unknown distribution, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the data space and $\mathcal{Y} = \{-1, 1\}$ is the label space. Given an example $\mathbf{x}$ and its corresponding true label y, the noise rate model in [17] is defined as $\rho_y(\mathbf{x}) = P(\tilde{Y} = -y \mid \mathbf{X} = \mathbf{x}, Y = y)$. For RCN, $\rho_{+1}(\mathbf{x}) = \rho_{-1}(\mathbf{x}) = \rho$ (ρ is a constant). For CCN, $\rho_y(\mathbf{x})$ is independent of $\mathbf{X}$ but dependent on Y. For ILN, $\rho_y(\mathbf{x})$ is dependent on both $\mathbf{X}$ and Y. A particular case of ILN is called *bounded instance- and label- dependent noise (BILN)* in [17], if the following assumption holds.

Assumption 1 (Key assumption in [17]) $\forall \mathbf{x} \in \mathcal{X}$, *we have the following bounds on the noise rates:*

$$\begin{cases} 0 \leq \rho_{+1}(\mathbf{x}) \leq \rho_{+1\max} < 1, \\ 0 \leq \rho_{-1}(\mathbf{x}) \leq \rho_{-1\max} < 1, \\ 0 \leq \rho_{+1}(\mathbf{x}) + \rho_{-1}(\mathbf{x}) < 1. \end{cases}$$

where $\rho_{+1\max}$ and $\rho_{-1\max}$ are upper bounds of noise rates for positive and negative examples respectively.

The first two restrictions on label noise rates indicate that the label noise rates (label flipping probabilities) of labeled examples must have upper bounds less than 1. The last restriction $0 \leq \rho_{+1}(\mathbf{x}) + \rho_{-1}(\mathbf{x}) < 1$ encodes the assumption made in [51] that a majority of the labels in the ILN model are correct on average, and it is derived from [12, 64] where $\rho_{\pm 1}$ are constant. In the rest of this paper, we always assume that Assumption 1 holds.

Boundary-Consistence Noise (BCN). Although our proposed algorithm is designed for any kind of label noise that satisfies Assumption 1, to demonstrate the effectiveness of our algorithm in AL-based data exploration, we focus on one special case of ILN, *Boundary-Consistence Noise (BCN)*, in evaluation. In the BCN model, the examples closer to the decision boundary are more likely to be mislabeled. As stated in [11, 25, 51], the BCN model is a reasonable model for human annotator noise, and it can be applied to many real-world applications. More details of the BCN model can be found in Section 5.4.

5.1.2 Automatic Noise Distillation and Limitations

We now introduce a data cleansing method with theoretical guarantees (first proposed in [17]), which automatically collects confident examples out of a potentially corrupted dataset for the label noise problem under Assumption 1. We refer to this method as *auto-cleansing*.

Lemma 1 *Denote by $\eta(\mathbf{x})$ the conditional probability $P(Y = +1 \mid \mathbf{X} = \mathbf{x})$. The Bayes optimal classifier is given by $g^*(\mathbf{x}) = sgn(\eta(\mathbf{x}) - \frac{1}{2})$.*

Corollary 1 (Corollary in [17]) *Denote by $\tilde{\eta}(\mathbf{x})$ the conditional probability $P(\tilde{Y} = +1 \mid \mathbf{X} = \mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$, and then we have*

$$\tilde{\eta}(\mathbf{x}) < \frac{1 - \rho_{+1\max}}{2} \Longrightarrow (\mathbf{x}, Y = -1) \textit{ is distilled;}$$
$$\tilde{\eta}(\mathbf{x}) > \frac{1 + \rho_{-1\max}}{2} \Longrightarrow (\mathbf{x}, Y = +1) \textit{ is distilled.}$$

When $\tilde{\eta}$, $\rho_{+1\max}$ and $\rho_{-1\max}$ are given, according to Corollary 1, a noisy example $(X_i, \tilde{Y}_i)$ is identified automatically as positive (negative) if $\tilde{\eta}(X_i) > \frac{1+\rho_{-1\max}}{2}$ ($\tilde{\eta}(X_i) < \frac{1-\rho_{+1\max}}{2}$). The labels of distilled examples are identical to those assigned by the Bayes optimal classifier under the clean distribution, i.e., g^*. In practice, $\tilde{\eta}$, $\rho_{+1\max}$ and $\rho_{-1\max}$ are usually unknown, but they can be estimated. More details can be found in [17].

This automatic noise distillation method is applicable to general label noise with robust performance guaranteed by the above theoretical results. Its effectiveness has been demonstrated empirically in the context of supervised learning. However, in the AL framework, under the combined effects of adverse factors (limited labeled data and severe class imbalance), the estimated $\tilde{\eta}$, $\rho_{+1\max}$ and $\rho_{-1\max}$ learned from the existing labeled examples are usually not accurate enough. Consequently, the confident examples collected by *auto-cleansing* may include a certain amount of label noise.

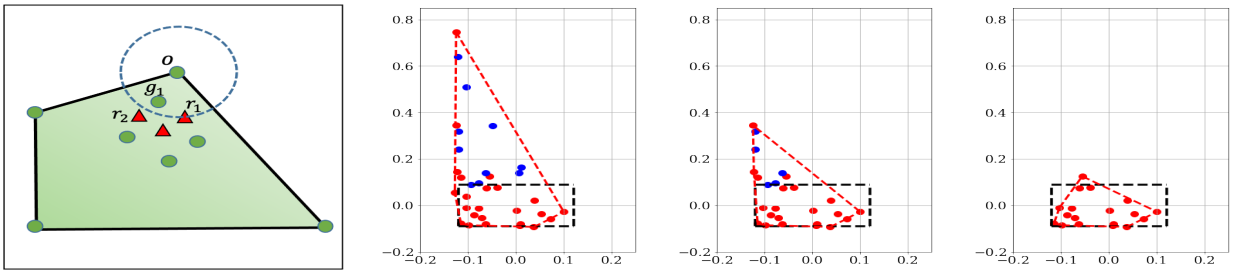
5.2 Robust Dual-Space Model

Given the limitations of *auto-cleansing* in the context of active learning, we propose a Robust Dual-Space Model (**DSM**) to further filter label noise and eventually build an accurate **DSM**. Here, we leverage the geometrical property of **DSM** and a k-nearest neighbors method to further remove noisy labels from the distilled examples collected by *auto-cleansing*. Since there is a tradeoff between introducing **DSM**-labeled examples to save user labeling effort and avoiding additional noisy labels from **DSM**-labeled examples, we also design an adaptive method based on estimated noise rates to bring in **DSM**-labeled examples strategically.

5.2.1 Data Space Model Refinement

To further filter noisy labels from the distilled examples collected by *auto-cleansing*, we propose a **DSM**-based data cleansing method called *Data Space Model Refinement*. In the following, we suppose that the subspatial convexity assumption holds for the data space under consideration. We use D_{ch} and D_{cc} to denote the labeled examples for building the convex hull and the convex cones, respectively.

Illustration. Before presenting the complete algorithm, we illustrate the intuition that the subset of examples in D_{ch} that are very close to the boundary of the convex hull are likely to be noisy. Our approach to cleansing the convex hull is that for each vertex of the current convex hull, if its k-nearest neighbors contain a conflicting example from D_{cc}, then we remove from the convex hull this vertex and all its neighbors that lie within a l-distance radius of this vertex, where l is defined to be the distance between the vertex and the discovered conflicting example. As shown in Figure 5(a), the green tetragon represents the convex hull formed by the examples in D_{ch} (green dots), and red triangles represent some examples from D_{cc} but inside the convex hull. Take the vertex o for example: its nearest neighbor is g_1 and its second nearest neighbor is r_1, regardless of the choice of k. If the assumption of subspatial convexity holds, both o



(a) Noise filtering for the convex hull (b) Initial convex hull (c) Intermediate convex hull (d) Final convex hull

Fig. 5: Illustration of data space model refinement

and g_1 are likely to be noisy examples, so we remove both o and g_1 to avoid the over-expansion of the convex hull. We refer to the process of constructing the convex hull based on current D_{ch} and examine every vertex of the convex hull to remove noise as *one round of noise filtering*.

Multiple-round noise filtering for the convex hull. One round of noise filtering, however, may not be adequate. We next demonstrate the effectiveness of the iterative refinement of the convex hull by displaying the gradually shrinking convex hulls over rounds of noise filtering. For ease of composition, we assume that the user interest query is in Q_c^+, and the convex hull of **DSM** is created from the positive examples. In addition, we use $k = 3$ for finding the k-nearest neighbors in the example in Figure 5.

Figure 5(b) shows the initial convex hull (enclosed by red dashed lines) built on the confident examples distilled by *auto-cleansing*. Here, the true query region is marked by the dashed black rectangle, the red points denote distilled positive examples, and the blue points denote distilled negative examples. Figure 5(c) shows the intermediate result of the convex hull after one round of noise filtering, and Figure 5(d) shows that after two-rounds of noise filtering. It is evident that in Figure 5(b), six negative examples (six red points outside the dashed black rectangle) are erroneously labeled, which leads to the over-expansion of the convex hull. Three out of them are the vertices of the initial convex hull, and have negative examples among their 3-nearest neighbors. By removing these mislabeled positive examples and some other positive examples from their neighbors, an intermediate convex hull is derived as shown in Figure 5(c). Since there still exist negative examples inside the convex hull, we perform another round of noise filtering and eventually obtain a conservative and much more accurate convex hull, as shown in Figure 5(d).

Refinement of the convex cones. To refine the convex cones, we perform simple filtering by checking whether an example x in D_{cc} is inside the refined hull. Only when $x \notin hull$, will x be used to build a convex cone.

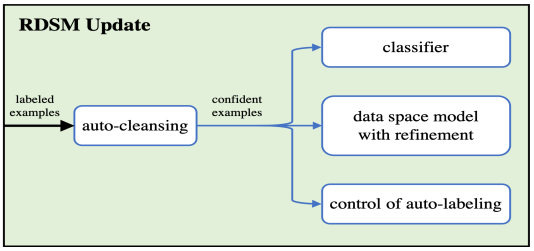


Fig. 6: **RDSM** update

5.2.2 Robust Dual-Space Model for Data Exploration

We now propose a Robust Dual-Space Model (**RDSM**) by integrating into **DSM** (i) the existing data cleansing method, *auto-cleansing* [17], (ii) our own data cleansing method, *data space model refinement*, and (iii) a new *adaptive auto-labeling* method in the sample acquisition procedure.

Like **DSM**, **RDSM** fulfills two functionalities, prediction and sampling, as an active learner for data exploration. But there are two major differences, in both model update and sample acquisition methods: At each iteration, we update **DSM** based on the labeled examples directly, and always allow **DSM** for auto-labeling in the sample acquisition process (lines 22-25 in Algorithm 1). In contrast, to update **RDSM**, as shown in Figure 6, we first use *auto-cleansing* to collect confident examples from labeled examples and then based on the confident examples, we 1) update the classifier, 2) build the data space model with refinement, and 3) compute estimated noise rates to control *auto-labeling*, which takes effect later in the sampling acquisition process.

We now present the full algorithm (Algorithm 2) of applying **RDSM** into active learning-based interactive data exploration and highlight the lines (marked in blue) that differ from Algorithm 1. Since the framework remains similar, we only discuss the two major differences.

Update of RDSM. Lines 7-15 update the **RDSM**. More specifically, we collect clean labeled examples D_{clean} distilled by *auto-cleansing* (line 7) and then use D_{clean} to train both a data space model with refinement (line 10) and a classifier (line 8). Lines 11-14 update the three-set partitions (D^+, D^-, D^u): the partitions must be rebuilt if there exist user-labeled examples (D_{lu}) in the newly labeled ex-

Algorithm 2 Robust Dual-Space Model

Input: database D, initial labeled data set D_0, accuracy threshold λ, sampling ratio γ, unlabeled pool size m

1: $D_{labeled} \leftarrow D_0, D_{unlabeled} \leftarrow D \setminus D_0$
2: $R^+ \leftarrow \emptyset, R^- \leftarrow \emptyset$
3: $D^+ \leftarrow \emptyset, D^- \leftarrow \emptyset, D^u \leftarrow D$
4: $\mathbf{D} \leftarrow (D^+, D^-, D^u)$
5: $D_{lu} \leftarrow D_0, D_{ld} \leftarrow \emptyset$
6: **repeat**
    // training the Robust Dual-Space Model:
7: $D_{clean} \leftarrow$ `distill_by_auto`$(D_{labeled})$
8: $classifier \leftarrow$ `train_classifier`(D_{clean})
9: $auto\_labeling \leftarrow$ `compare_rates`$(D_{labeled}, D_{clean})$
10: $(R^+, R^-) \leftarrow$ `train_data_space_model`(D_{clean})
11: **if** $D_{lu} = \emptyset$ and $D_{ld} \neq \emptyset$ **then**
12: $\mathbf{D} \leftarrow$ `threeSets`$(R^+, R^-, \mathbf{D})$
13: **else**
14: $\mathbf{D} \leftarrow$ `rebuilding_threeSets`(R^+, R^-, D)
15: $accu \leftarrow$ `threeSetMetric`$(\mathbf{D})$
    // applying the combined query strategy:
16: $D_{lu} \leftarrow \emptyset, D_{ld} \leftarrow \emptyset, \boldsymbol{x}^* \leftarrow \emptyset$
17: **if** $rand() \leq \gamma$ **then**
        //sampling from the uncertain region :
18: $pool \leftarrow$ `subsample`(D^u, m)
19: $\boldsymbol{x}^* \leftarrow$ `sample_acquisition`$(pool, classifier)$
20: $D_{lu} \leftarrow$ `get_labels_from_user`$(\boldsymbol{x}^*)$
21: **else**
        //uncertainty sampling around the boundary:
22: $pool \leftarrow$ `subsample`$(D_{unlabeled}, m)$
23: $\boldsymbol{x}^* \leftarrow$ `sample_acquisition`$(pool, classifier)$
24: **for** $\boldsymbol{x} \in \boldsymbol{x}^*$ **do**
25: **if** $\boldsymbol{x} \in (R^+ \cup R^-)$ and $auto\_labeling$ **then**
26: $D_{ld} \leftarrow D_{ld} \cup \{$`get_labels_from_rdsm`$(\boldsymbol{x})\}$
27: **else**
28: $D_{lu} \leftarrow D_{lu} \cup \{$`get_labels_from_user`$(\boldsymbol{x})\}$
    // updating the labeled and unlabeled data sets:
29: $D_{labeled} \leftarrow D_{labeled} \cup D_{lu} \cup D_{ld}$
30: $D_{unlabeled} \leftarrow D_{unlabeled} \setminus \boldsymbol{x}^*$
31: **until** $accu \geq \lambda$ **or** `reachedMaxNum()`
32: `finalRetrieval`$(D, (R^+, R^-), classifier)$

amples. Otherwise, they are updated incrementally. The implicit assumption of the incremental update method is that $D^{\pm}$ (D^u) built at a certain iteration must be a superset (subset) of $D^{\pm}$ (D^u) built at any previous iteration. We use $D^{(\pm,t)}$ and $D^{(u,t)}$ to denote $D^{\pm}$ and D^u built at iteration t respectively and then the assumption can be formalized as: if $t_1 < t_2$, then $D^{(\pm,t_1)} \subseteq D^{(\pm,t_2)}$ and $D^{(u,t_2)} \subseteq D^{(u,t_1)}$. However, this assumption does not hold in the presence of label noise. As a consequence, we need to correct the existing partitions when receiving newly labeled examples from the user. We believe that the data space model created with more labeled examples is more accurate than that created with fewer labeled examples, so we correct the existing partitions by rebuilding the partitions according to the latest data space model. Rebuilding the partitions enables **RDSM** to self-correct the partitions and thus alleviates the effect of label noise on its performance.

Adaptive Auto-labeling. Line 9 determines whether to utilize **RDSM** later for auto-labeling. Recall that our labeled set includes both user-labeled and **DSM**-labeled examples. Hence, we compute the error rates for these two subsets, respectively, and compare them to decide whether auto-labeling is enabled. The estimated noise rate of user-labeled (**DSM**-labeled) examples $\hat{\rho}_u$ ($\hat{\rho}_d$) is defined to be the ratio of estimated mislabeled examples in user-labeled (**DSM**-labeled) examples. The on-off switch $auto\_labeling$ is assigned to be $True$ if $\hat{\rho}_d < \hat{\rho}_u$, otherwise $False$. Line 25 shows that only when $auto\_labeling$ is $True$, **RDSM** is qualified to provide labels. This adaptive strategy, which not only takes advantage of auto-labeling to save user labeling effort but also prevents introducing label noise from **RDSM**, is more realistic and applicable than using auto-labeling all the time. More details can be found in [39].

5.3 High-dimensional Exploration with Label Noise

In this section, we present how to optimize **RDSM** further through factorization for high-dimensional exploration and analyze how factorization affects learning with label noise.

For data exploration performed with high dimensionality, to further boost performance we leverage the idea of factorization to devise the factorized **RDSM**, denoted as **RDSM**$_F$. This method divides the high dimensional space into a set of lower dimensional subspaces and builds a **RDSM** in each subspace. **RDSM**$_F$ follows the same factorization rules as those of **DSM**$_F$, as described in Section 4.

We provide further analysis of the effect of factorization on learning with label noise. We prove that if the factorization structure has the conjunctive property, the noise rate of positive examples in each subspace is lower than or equal to that in the original data space. Besides, we explain how factorization affects the mislabeled negative examples.

Mislabeled positive examples. The user interest queries in our data exploration setting usually have very low selectivity, i.e., the user interest examples are scarce and hard to hit during the data exploration. It is well known that such class imbalance pushes the decision boundary of classification models towards minority examples [5, 72, 79, 80]. Further, if some minority examples are mislabeled while the majority examples are clean, the class imbalance problem becomes more severe. In general, mislabeled minority examples are more destructive than mislabeled majority examples. In our problem setting, the minority examples are positive examples, which are often rare when the user comes to explore a very large dataset. Hence, our design pays more attention to mislabeled positive examples.

Without loss of generality, we assume that the factorization structure consists of two groups of attributes (X^1, X^2), $X^j \in \mathbb{R}^{d_j}, j = 1, 2$ and $\{X^1, X^2\}$ are independent. In the jth subspace, we use Y^j and $\tilde{Y}^j$ to denote the true label and the observed label of the projected examples X^j,

and we define the noise rate of an example x in this subspace as, $\rho^j_{y^j}(x) = P(\tilde{Y}^j = -y^j \mid X^j = x^j, Y^j = y^j)$, $y^j \in \{-1, +1\}$. According to the conjunctivity in the original high-dimensional data space, we have the clean but unobserved label $Z = \min(Y^1, Y^2)$, the observed but corrupted label $\tilde{Z} = \min(\tilde{Y}^1, \tilde{Y}^2)$, and the overall noise rate is defined as $\rho_z(x) = P(\tilde{Z} = -z \mid X = x, Z = z)$, where $x = (x^1, x^2), x^j \in \mathbb{R}^{d_j}, j = 1, 2$.

Lemma 2 *Denote by F the factorization structure composed of two groups of attributes (X^1, X^2), $X^j \in \mathbb{R}^{d_j}, j = 1, 2$, and $\{X^1, X^2\}$ are independent, if the conjunctive property holds in F, we have $\rho^j_{+1} \leq \rho_{+1}, j = 1, 2$.*

Proof With the conjunctive property, we have

$$\begin{aligned}
\rho_{+1}(x) &= P(\tilde{Z} = -1 \mid X = x, Z = +1) \\
&= \frac{P(\tilde{Z} = -1, X = x, Z = +1)}{P(X = x, Z = +1)} \\
&= \frac{P(X = x, Z = +1) - P(\tilde{Z} = +1, X = x, Z = +1)}{P(X = x, Z = +1)} \\
&= 1 - \prod_{i=1,2} \left(\frac{P(\tilde{Y}^i = +1, X^i = x^i, Y^i = +1)}{P(X^i = x^i, Y^i = +1)} \right) \\
&= 1 - \big(1 - \rho^1_{+1}(x)\big)\big(1 - \rho^2_{+1}(x)\big) \\
&= \rho^1_{+1}(x) + \rho^2_{+1}(x) - \rho^1_{+1}(x)\rho^2_{+1}(x)
\end{aligned}$$

Then $\rho_{+1}(x) - \rho^1_{+1}(x) = \rho^2_{+1}(x)(1 - \rho^1_{+1}(x)) \geq 0 \Longrightarrow \rho^1_{+1}(x) \leq \rho_{+1}(x)$. Similarly, we can prove $\rho^2_{+1}(x) \leq \rho_{+1}(x)$.

It is trivial to extend the Lemma 2 from two independent groups of attributes to a finite number of independent groups and obtain the following theorem.

Theorem 3 *Denote by F the factorization structure composed of m groups of attributes $(X^1, X^2, \cdots, X^m)$, $X^j \in \mathbb{R}^{d_j}, j = 1, 2, \cdots, m$, and $\{X^1, X^2, \cdots, X^m\}$ are independent, if the conjunctive property holds in F, we have $\rho^j_{+1} \leq \rho_{+1}, j = 1, 2, \cdots, m$.*

Theorem 3 demonstrates that given a factorization structure with conjunctive property, the noise rate of positive examples in each subspace is lower than that in the original high-dimensional space. It coincides with the intuition that when a positive example is labeled as negative and the user is asked to specify which subspaces cause the negative label, the user may not label all the subspaces as negative. Thus, the classifiers can always gain useful information from the subspaces that obtain the correct subspatial labels.

Mislabeled negative examples. We now focus on negative examples. The projection of negative examples may fall into the projected positive region (user interest region) in some subspace. For example, a small black car does not suit the user who seeks a large black car, but it captures the user interest partially along the color dimension, leading to an overall negative label but a positive subspatial label for the color. When a negative example is mislabeled as positive in the original high-dimensional space, according to the conjunctive property, its subspatial label in each subspace is inferred to be positive. Since the actual subspatial labels of this example in some subspaces might be positive, these subspatial labels remain correct and can offer useful information in the corresponding subspaces. Compared to standard classifiers, which only learn from the mislabeled negative examples in high-dimensional space, the classifiers with factorization gain greater insight from the subspaces and become more robust.

5.4 Simulation of Label Noise Models

Boundary-Consistence Noise (BCN). As mentioned in Section 5.1.1, we focus on *Boundary-Consistence Noise (BCN)* for evaluation as the BCN model is considered a reasonable model for human annotator noise. Inspired by [25], we assume that the label noise is distributed as an *unnormalized Gaussian* centered at decision boundary with variance σ^2 and thus, the BCN model can be represented by

$$\begin{aligned}
P(\tilde{y} \neq y \mid \mathbf{x}) &= \rho_{\max} \exp\left(-\frac{(dist(\mathbf{x}, B))^2}{2\sigma^2}\right) \\
&\triangleq \rho_{\max} \exp\left(-\frac{d^2_{\mathbf{x}}}{\lambda}\right)
\end{aligned} \tag{8}$$

where $d_{\mathbf{x}} = dist(\mathbf{x}, B)$ represents the distance from the example $\mathbf{x}$ to the true decision boundary B and $\lambda = 2 * \sigma^2$ is a parameter to determine the label noise level. When $d_x = 0$, i.e. the example $\mathbf{x}$ lies on the decision boundary, $P(\tilde{y} \neq y \mid \mathbf{x})$ reaches its maximum $\rho_{\max}$. Given fixed $\rho_{\max}$ and an example $\mathbf{x}$, larger λ leads to higher noise rate. When λ is large enough, the BCN model approximates a RCN model with the label flip probability equal to $\rho_{\max}$.

To apply the BCN model for our simulation of human annotator noise, we need to address the following issues.

Estimation of $d_{\mathbf{x}}$. When the query pattern is not linear, it is nontrivial to attain $d_{\mathbf{x}}$ for any example $\mathbf{x}$. Although a perfect SVM classifier is trained based on the ground truth to compute $d_{\mathbf{x}}$ in [40], the estimated distance provided by SVM cannot perfectly reflect the true distance, which eventually leads to a distorted distribution of the BCN model. Therefore, in this work, we propose a new method to estimate $d_{\mathbf{x}}$ more precisely. The new distance estimation method mainly consists of three steps: 1) **Collect border points**: We run a k-nearest neighbors algorithm (provided by the scikit-learn library[5]) on the entire dataset and collect the examples whose neighbors contain at least t ($t \geq 1$) examples(s) from the other class. These collected examples must be very close to the boundary, and we call them *border points*. 2) **Compute distance**: Given an example $\mathbf{x}$, we

[5] https://scikit-learn.org/stable/

treat the distance from $\mathbf{x}$ to its *nearest* border points as the estimated $d_{\mathbf{x}}$. 3) **Scaling $d_{\mathbf{x}}$ to [0, 1] within each class**: According to the ground truth of the user interest, we separate the entire dataset into two sets, i.e., a positive set and a negative set. Then we scale $d_{\mathbf{x}}$ to [0, 1] by its maximum absolute value within each set.

Setup of $\rho_{\max}$. We treat $\rho_{\max}$ as a parameter to control different noise settings. We vary them in the range, [0.05, 0.1, 0.2, 0.3, 0.4, 0.5], in our experimental evaluation.

Tuning of λ. We assume that the examples **farthest** to the decision boundary have very low probability (at most τ) to be mislabeled under the BCN model. Given $\rho_{\max}$ and the upper bound τ ($\tau = 10^{-12}$ in our experiments), we have

$$\rho_{\max} \exp\left(-\frac{1}{\lambda}\right) \leqslant \tau \Longrightarrow \lambda \leqslant \frac{-1}{\log(\tau/\rho_{\max})}.$$

Noise injection during data exploration. We simulate noisy labels in our experimental study as follows: For a fixed user interest pattern and a fixed noise level $\rho_{\max}$, we set up λ as $\frac{-1}{\log(\tau/\rho_{\max})}$ and precompute the label flip probabilities in advance according to the BCN model defined in Equation 8. In the simulated user labeling process of data exploration, we flip a coin for the selected example $\mathbf{x}$ with the label flip probability, $P(\tilde{y} \neq y \mid \mathbf{x})$, to determine whether a noisy label is generated or not for this example.

6 Experimental Evaluation

We implemented all of our proposed techniques in a Python-based prototype for data exploration, which connects to a PostgreSQL database. In this section, we evaluate our techniques and compare to recent active learning algorithms [13, 28], active search [33], and explore-by-example systems including Aide [23, 24] and LifeJoin [19].

Datasets: Our evaluation used two datasets. (1) SDSS (190 million tuples) contains the "PhotoObjAll" table with 510 attributes. By default, we used 1% sample (1.9 million tuples, 4.9GB) to create an evaluation set for **DSM** and for pool-based uncertainty sampling – our formal results in Section 3.4 allowed us to use the 1% sample for data exploration, yet with bounded difference of $\epsilon \leq .001$ from the accuracy achieved over the full database with probability $\geq$0.994. (2) Car database (5622 tuples): this small dataset is used for our user study because it is more intuitive for users to perform explorative analytics. We defer a detailed discussion to §6.3.

User Interest Queries: We extracted 7 query templates from the SDSS query release [68] to represent user interests. They allow us to run *simulations* of user exploration sessions, as in [23, 24], by using the true query answers as the proxy for user labeling. The 7 templates are shown in Table 1.

Table 1: Query templates with different selectivity values

Query template
Q1 (rectangle): $rowc \in [a_1, a_2] \wedge colc \in [b_1, b_2]$
Q2 (ellipse): $((rowc - a_1)/b_1)^2 + ((colc - a_2)/b_2)^2 < c^2$
Q3 (rectangle): $ra \in [a_1, a_2] \wedge dec \in [b_1, b_2]$
Q4 (outside a circle): $rowv^2 + colv^2 > c^2$
Q5: 4D queries combining two queries from Q1-Q4
Q6: 6D queries combining three from Q1-Q4
Q7: 4D, $(x_1 > a + b \cdot x_2) \wedge (x_3 + c \cdot \log_{10} x_4^2 < d)$

Table 1. Each template is instantiated with different constants to vary query selectivity in [0.01%, 10%]. In particular, Q1-Q3 represent patterns that are convex in the positive region ($\mathbf{Q}_c^+$). Q4 retrieves tuples outside a circle, hence in the $\mathbf{Q}_c^-$ class. Q5-Q6 combine these attributes for scalability tests. Q7 includes a predicate on $x1$ and x_2 that belongs to $\mathbf{Q}_c^+$, and a log predicate on x_3 and x_4 that belongs to $\mathbf{Q}_c^-$ if $x_4 > 0$ or is non-convex if $x_4 \in \mathbb{R}$. We use Q7 to test partial factorization as discussed in §4.

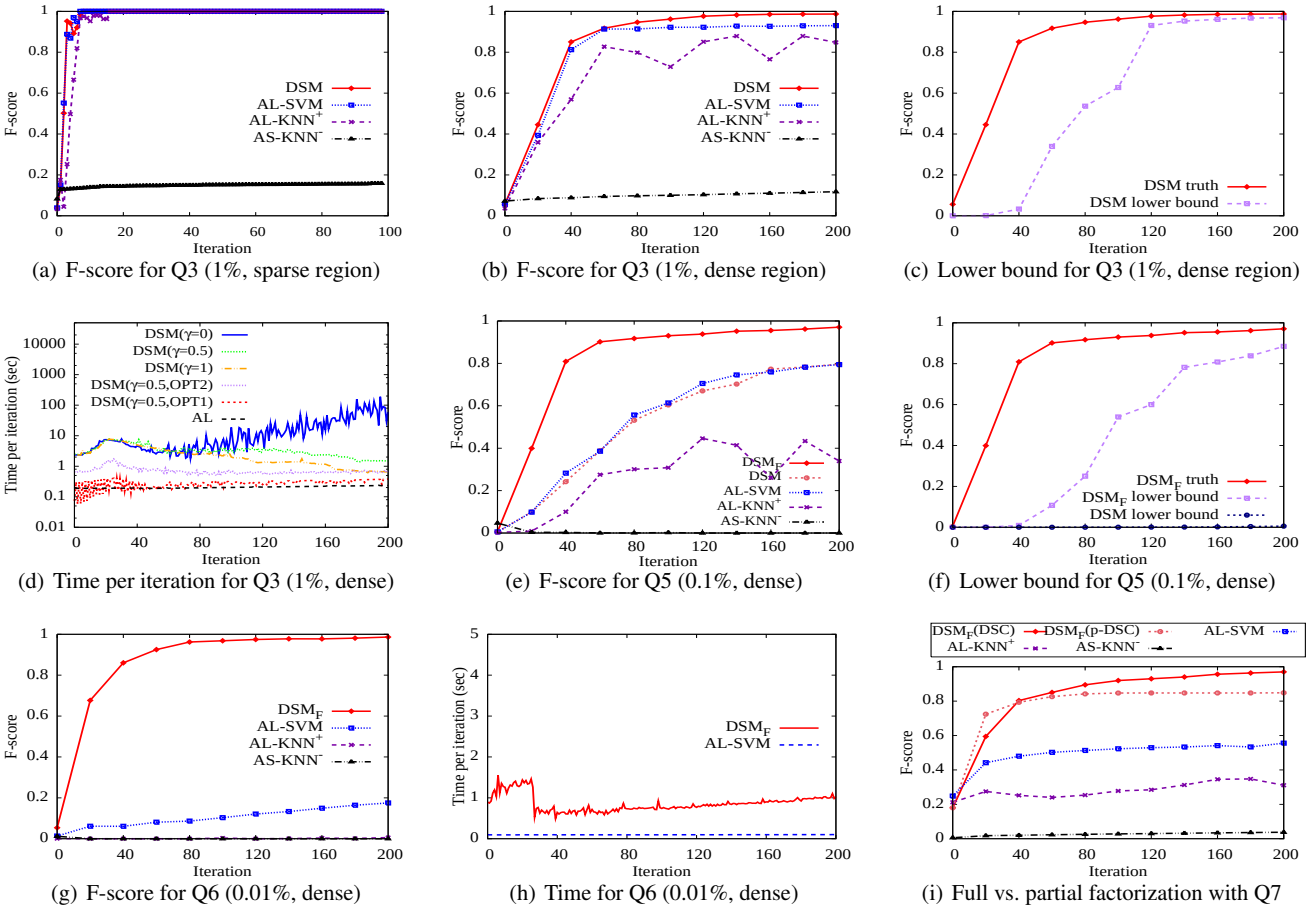
Servers: Our experiments were run on four servers, each with 40-core Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz, 128GB memory, Python 3.7 on CentOS 7.

6.1 Dual-Space Algorithm with Factorization

We evaluate our Dual-Space Model (**DSM**) and compare it to two ML techniques: (i) Active Learning (**AL**) runs uncertainty sampling [13,28] to obtain labeled examples for building a classifier, which is the version space part of **DSM**. We run **AL** with an SVM or a standard kNN classifier (kNN$^+$). (ii) Active Search (**AS**) [33] also follows an iterative procedure of asking the user to label an example and training a new model to guide the selection of the next example, but uses a strategy to maximize the number of positive examples in the set of selected examples, not classification accuracy. It uses a special variant of kNN classifier to enable optimization, denoted as kNN$^-$. In all plots, the x-axis is the number of labeled examples.

Expt 1 (2D queries): We run 2D queries from templates Q1 to Q4. Since the results show similar trends, we show the results for Q3 (1%) in Figure 7(a)-7(d).

Regarding F1-score, **DSM** outperforms **AL**, and **AL** outperforms **AS**. (1) An important factor is the data distribution around the decision boundary B of the user interest query. If B falls into a sparse region, separating the positive and negative classes is relatively easy for **DSM** and **AL**. Figure 7(a) shows that for Q3 whose decision boundary is in a sparse region, **DSM** and **AL-SVM** converge within 10 labeled examples. However, **AS** performs poorly, with the F1-score less than 20%. The reason is that **AS** compromises recall by searching close to existing positive examples. In contrast, **DSM** and **AL** aims to maximize classification accuracy by sampling the uncertain region of the model, e.g., close to

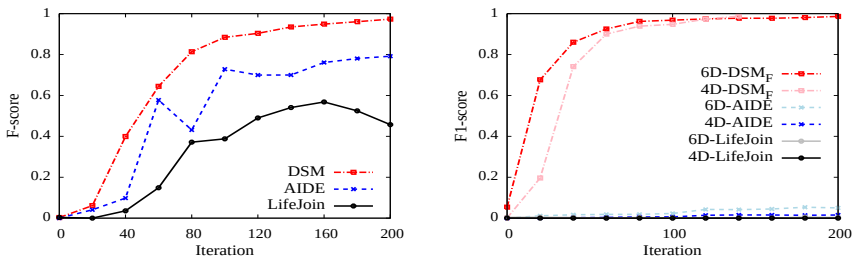


(a) F-score for Q3 (1%, sparse region) (b) F-score for Q3 (1%, dense region) (c) Lower bound for Q3 (1%, dense region)

(d) Time per iteration for Q3 (1%, dense) (e) F-score for Q5 (0.1%, dense) (f) Lower bound for Q5 (0.1%, dense)

(g) F-score for Q6 (0.01%, dense) (h) Time for Q6 (0.01%, dense) (i) Full vs. partial factorization with Q7

Fig. 7: **DSM** with factorization for SDSS 2D, 4D, and 6D queries, compared to **AL** and **AS** algorithms



(a) **DSM** vs Aide and LifeJoin (SDSS 2D) (b) **DSM** vs Aide and LifeJoin (SDSS 4D, 6D)

Fig. 8: Results of SDSS queries for partial factorization and comparison to Aide and LifeJoin systems

the decision boundary, hence offering better F1-score. (2) If B falls into a dense data region, learning an approximate $\tilde{B}$ that can accurately divide all unlabeled data points requires more labeled examples. To reach 95% accuracy, **DSM** requires 100 labeled examples for Q3 and 90 examples on average for Q1-Q4. **AL** works better with the SVM than the kNN classifier, but even **AL-SVM** fails to reach 95% for Q3 or most other workloads. Finally, **AS** performs much worse than **AL-KNN**$^+$ while both use kNN classifiers. **DSM** further offers a lower bound in F1-score. Figure 7(c) shows that the lower bound is quite close to the true F1-score.

The time cost of **DSM**, shown in Fig. 7(d), varies with the sampling method. Recall from Alg. 1 that with probability γ, it samples from the uncertain partition of the polytope model, D^u; otherwise, it does so from a subsample of the unlabeled examples in the database, $D_{unlabeled}$. γ=0 leads to high time costs because the examples retrieved from $D_{unlabeled}$ may repeatedly fall in the positive or negative region of the polytope model, wasting resources with no new information. γ=1 and γ=0.5 significantly reduce the time cost, with γ=0.5 offering slightly better accuracy due to balanced sampling. However, both settings exhibit a spike in

time cost in the early phase, which is the time to build the polytope model the first time by scanning the entire evaluation set. Finally, we improve γ=0.5 with the optimization (OPT1) from Section 3.4, where we start with a smaller evaluation set (n=50k) to avoid the initial spike, but later switch to a larger evaluation set (n=1.9 million) once its polytope model is built completely in the background. This optimization keeps the time cost per iteration within a second. Another way to reduce the time cost is to apply distributed computing (OPT2) from Section 3.4, which cuts down the per-iteration time to within 1-2 seconds when the size of the evaluation set is always 1.9 million. OPT1 and OPT2 can be combined to further improve the time performance; however, OPT1 may reduce accuracy due to use of a smaller sample. Since OPT2 keeps the time cost per iteration within a reasonable range, OPT2 with a 1.9-million evaluation set will be used as the default setting of our algorithm.

Expt 2 (4D-6D queries): We next show results of 4D-6D queries in dense regions as they present harder workloads. The main results of Figure 7(e)-7(h) are: (1) Without factorization, **DSM** performs similarly to AL as the polytope model is dominated by the uncertain region. **DSM**$_F$ dramatically shrinks the uncertain region, and improves the lower bound and F1-score. (2) Consistently, **DSM**$_F$ outperforms **AL**, which further outperforms **AS**. Figure 7(g) shows that for the 6D query, **DSM**$_F$ reaches 96% with 80 examples, **AL-SVM** cannot reach 20% with 200 examples, and **AS** is close to 0%. (3) **DSM**$_F$ has the time per iteration within 1-2 seconds, as shown in Figure 7(h) for the 6D query.

Expt 3 (Partial factorization): We next generate two queries from Q7 with 7.8% selectivity for Q7.v_1 and 0.7% selectivity for Q7.v_2 (based on the constants used in SDSS). Q7.v_1 is in the **DSC** family because its positive region is convex in the ($x1$, x_2) subspace, and its negative region is convex in (x_3, x_4). Q7.v_2 is in the p-**DSC** family because it is non-convex in (x_3, x_4). Hence, **DSM** supports Q7.v_2 with partial factorization. Figure 7(i) shows that **DSM**$_F$ works better for the **DSC** query than p-**DSC** query, as expected, but even partial factorization works much better than (i) **AL**, which does not reach 60% after 200 labeled examples, and (ii) **AS**, which cannot achieve more than 5% accuracy.

6.2 Comparison to Alternative Systems

We next compare our system to two state-of-the-art explore-by-example systems: 1) LifeJoin [19] uses SVM for classification and active learning for seeking the next example for labeling. But it differs in the SVM implementation from our work: LifeJoin proposed a hybrid approach that utilizes program synthesis to generate a number of interest functions (each function can be treated as a weak learner and the group forms an ensemble), and uses an SVM to find the weights for the generated functions. The example on which the weighted functions disagree the most is selected as the next example for labeling. We implemented LifeJoin as additional methods in our system. 2) Aide [23,24] uses decision trees as the classification model and customized methods for seeking the next example for labeling. We obtained the source code from the authors. When comparing these systems, we exclude the overhead to find the first positive example as these systems use different methods / assumptions to find them.

Expt 4: The F1-score is reported in Figure 8(a) for a 2D query, and in Figure 8(b) for 4D and 6D queries. (1) For the 2D query, our system outperforms Aide, which further outperforms LifeJoin. Overall, LifeJoin is significantly worse in accuracy. (2) For the 4D query, Aide and LifeJoin drop to below 10% in accuracy, while our system achieves 99% within 140 iterations. For the 6D query, again Aide and LifeJoin fail to work. This observation remains for any query that we tried beyond 2D. This is due to the combination of low selectivity and high dimensionality in data exploration. Additional analysis of these systems is available in [39].

6.3 User Study using a Car Database

We conducted a user study by building a car database[6]. The database includes 5622 vehicles with 27 attributes such as the model, year, length, height, engine power, and retail price. Our study has two objectives: (1) build a query trace to understand the characteristics of data exploration tasks in this domain; (2) use this trace as the ground truth of user interests to evaluate our system.

To develop the query trace, we designed task scenarios with different price constraints and usage patterns, e.g., buying a car for everyday city commute, outdoor sports, an elderly in the family, or a small business. The 18 users in our study belong to two groups: the first 11 users are CS professors and graduate students, while the rest of 7 are non-technical people. We asked each user to find all the cars that meet the requirements of the assigned task so that he can provide a recommendation service to customers who belong to the given scenario. Each user proceeds in three phases: 1) Review the schema: Since this is a familiar domain, most users can understand the schema quickly. 2) Initial exploration: We next show sample data to help the user understand the data and materialize the user preference. We also ask the user to come up with the first positive example via (a) naming a car that he/she already has in mind, or (b) reviewing a sample set precomputed for the given task based on the price constraints, body type, year, etc., and finding one that appears relevant. Two users chose option (a) while the others chose option (b). 3) Iterative manual exploration: Next, the user is asked to specify her interest precisely by (i) sending a SQL query

[6] The content is extracted from http://www.teoalida.com/

Table 2: Results of the car database using Manual Exploration, **DSM**$_F$, Active Learning (AL). "#attrs" shows the number of attributes used in the user interest and "# encoded features" shows the number of features (with one-hot encoding) after transforming all categorical attributes for use by the classifier. For manual exploration, "# tuples INIT" shows the number of tuples reviewed in initial exploration (the 2nd phase) for the user to find the first positive example; and "# tuples ITER" shows those reviewed in iterative exploration (3rd phase). Last three rows show the global Min, Max and Mdn respectively. For **DSM**$_F$ and AL, the algorithm marked by '-' never reached the desired accuracy within 100 iterations. The notation "SIM RES" over the last 6 columns indicates that these results are based on the simulation of user behaviors.

Q		# attrs	# encoded features	Manual Exploration # tuples INIT	# tuples ITER	#SQL	AL (SIM RES) 0.8	0.95	0.99	DSM$_F$ (SIM RES) 0.8	0.95	0.99
Q1	**Min**	4	12	0	35	8	8	9	9	4	4	5
∼	**Max**	8	418	18	412	49	-	-	-	14	23	26
Q11	**Mdn**	6	35	11	104	16	29	42	-	7	10	13
Q12	**Min**	4	4	0	50	4	9	15	15	3	5	6
∼	**Max**	10	51	24	260	15	-	-	-	7	11	12
Q18	**Mdn**	6	24	4	91	10	28	-	-	6	6	9
Global	**Min**	4	4	0	35	4	8	9	9	3	4	5
Global	**Max**	10	418	24	412	49	-	-	-	14	23	26
Global	**Mdn**	**6**	**30**	**10**	**98**	**12**	**29**	**-**	**-**	**6**	**9**	**10**

to the database (for the 7 non-technical people, we offered help to translate their requirements to SQL); (ii) reviewing a subset of the returned tuples; (iii) going back to revise the query. The steps are repeated until the user is satisfied with all returned tuples of the final query.

First, we verify that the selectivities of all 18 queries are in the range of [0.2%, 0.9%]. They contain 117 predicates in total: 70 of them are defined on numerical attributes and are all convex in the positive region. The rest 47 use categorical attributes, for which the notion of convexity does not apply. All the queries are conjunctive; the only disjunction is applied to the categorical attributes.

Second, we evaluate our system by running simulations using these queries as the true user interest. While the queries involve 4 to 10 attributes, the classifier requires categorical attributes to be transformed using one-hot encoding, resulting in 4 to 418 features. As for the initial examples for each experiment, the initial positive example is provided by the user at Step 2, and the initial negative example is randomly picked from the examples that are "far away" from the initial positive example. Table 2 shows in the "DSM" column family that **DSM**$_F$ achieve 99% for all queries, with a median of 10 labeled examples, despite the high-dimensionality. In contrast, the users manually wrote a series of queries, with a median of 12, and reviewed many tuples, with a median of 98. The results from two user groups are largely consistent, with a small difference that non-technical people tend to specify simpler queries, hence easier for **DSM** to learn. Note that the initial cost of finding the first positive example, with a median of 10 tuples, can be added fairly to both exploration modes.

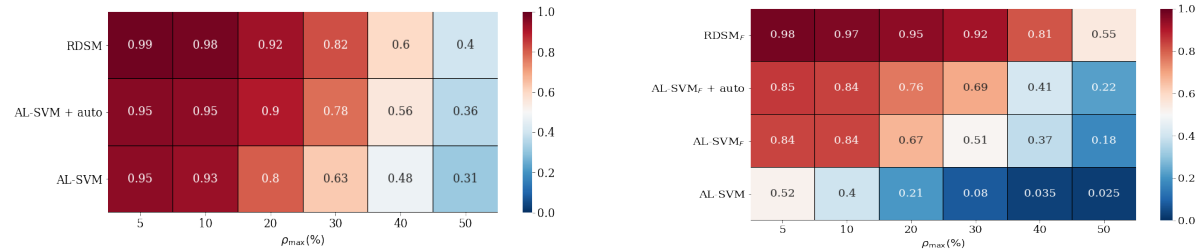
Third, the study verified our benefits over active learning (**AL**), which cannot reach 95% accuracy for most queries within 100 iterations, and for 80% accuracy requires a median of 29 labeled examples. Even though the decision boundary is often in a sparse region in this dataset, high dimensionality makes **AL** suffer in performance.

6.4 Learning with Label Noise

We now evaluate our algorithm for handling label noise and demonstrate its advantages over alternative methods. The previous results showed that in the noise-free cases, (1) **AL-SVM** outperforms **AL-KNN**$^+$, **AL-KNN**$^-$ and **AS**, (2) the alternative systems **Aide** and **LifeJoin** are greatly inferior to **DSM**$_F$ and they are often defeated by **AL-SVM**. Besides, both **Aide** and **LifeJoin** assume noise-free conditions and **cannot** handle label noise. Thus, the following experiments use **AL-SVM** as the baseline and focus on the harder problem in which the decision boundary falls into dense regions.

Expt 5 (Evaluation of **RDSM** without factorization): Figure 9(a) shows the averaged F-score comparison of **RDSM**, **AL-SVM + auto** (**AL-SVM** combined with *auto-cleansing*), and **AL-SVM** over all 2D queries (queries from SDSS query templates Q1-Q4) at iteration 200 in the form of a heat map. In the heat map, the x-axis labels correspond to varied $\rho_{\max}$ from 5% to 50%, and the y-axis labels correspond to various algorithms. The number inside the cell (i, j) corresponds to the averaged F-score over all 2D queries when algorithm j is run with $\rho_{\max} = i$. For example, as the red cell in the upper-left corner shows, when $\rho_{\max} = 5\%$, **RDSM** reaches up to 99% accuracy on average.

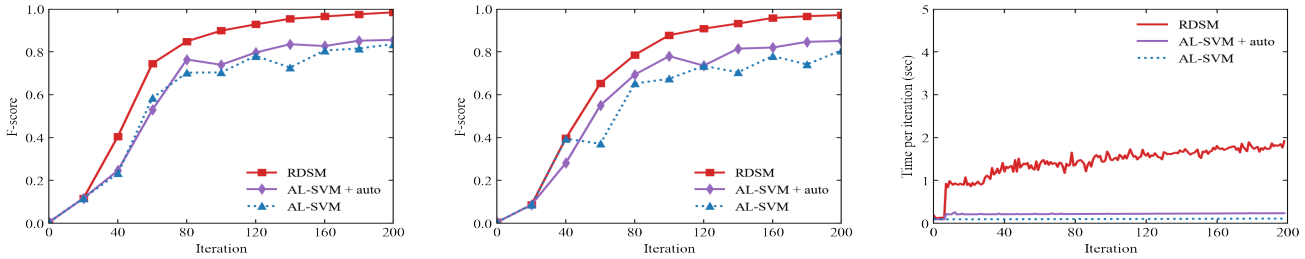
The main observations are: (1) **RDSM** outperforms the other two algorithms under every noise rate. (2) Given a fixed $\rho_{\max}$, there always exists a gap (2%-4%) between



(a) Averaged F-scores of **RDSM**, **AL-SVM + auto** and **AL-SVM** over SDSS 2D queries at iteration 200

(b) Averaged F-scores of **RDSM**$_F$, **AL-SVM**$_F$ **+ auto**, **AL-SVM**$_F$ and **AL-SVM** over 4D-6D queries at iteration 200

Fig. 9: Results for SDSS queries in the presence of label noise



(a) Q3 (0.1%), $\rho_{\max}$ 5%, F-score

(b) Q3 (0.1%), $\rho_{\max}$ 10%, F-score

(c) Q3 (0.1%), Time

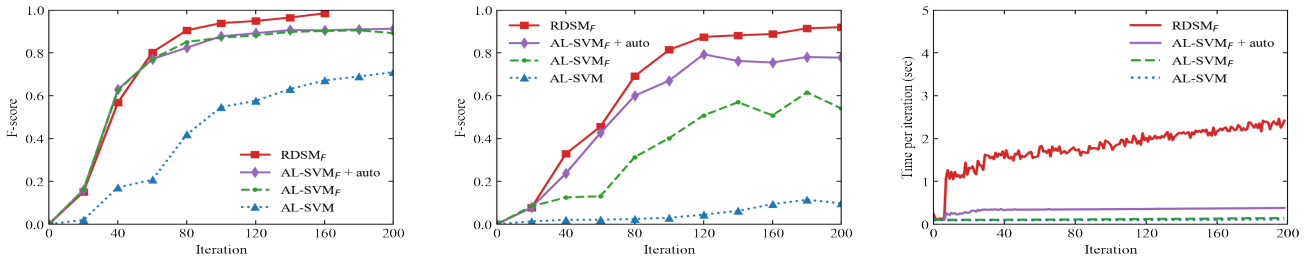
Fig. 10: **RDSM** for SDSS 2D queries, compared to **AL-SVM + auto** and **AL-SVM**

RDSM and **AL-SVM + auto**, while the latter outperforms **AL-SVM**. It empirically proves that the performance improvement of **RDSM** is attributed to not only the *auto-cleansing* method but also our proposed techniques — data space model refinement and adaptive auto-labeling. (3) As $\rho_{\max}$ becomes larger (i.e., the noise rate increases), all algorithms reduce accuracy while **RDSM** reduces at the slowest pace. For example, with $\rho_{\max}$ ranging from 5% to 30%, **RDSM** drops from 99% to 82%, **AL-SVM + auto** drops from 95% to 78% and **AL-SVM** drops from 95% to 63%. (4) **AL-SVM + auto** performs similarly to **AL-SVM** when $\rho_{\max}$ is relatively small, but it achieves substantially higher accuracy than **AL-SVM** when $\rho_{\max}$ becomes larger.

We now present some detailed results of different algorithms in terms of the F-score and time measurement. Since the results for different queries show similar trends, and Q3 (0.1%) is one of the most challenging 2D query patterns to learn, we present only Q3 (0.1%) in the interest of space. Figures 10(a) - 10(b) show that **RDSM** surpasses the other algorithms in accuracy almost at every iteration. With respect to time measurement, the time cost of each algorithm remains similar under different levels of label noise. As shown in Figure 10(c), both **AL-SVM + auto** and **AL-SVM** have time cost per iteration below 0.5 second. The time cost per iteration of **RDSM** is within 2 seconds, which is reasonable for the requirement of interactive performance.

Expt 6 (Evaluation of **RDSM** with factorization (**RDSM**$_F$)): The heat map in Figure 9(b) shows the averaged F-score comparison of **RDSM**$_F$, **AL-SVM**, **AL-SVM**$_F$ (**AL** with a factorized SVM), and **AL-SVM**$_F$ **+ auto** (**AL-SVM**$_F$ combined with *auto-cleansing*) for 4D-6D queries at iteration 200. The heat map is drawn in the same format as that in Figure 9(a). The main results are: (1) Consistently, **RDSM**$_F$ outperforms the other algorithms under every noise rate. (2) Thanks to the factorized data space model, **RDSM**$_F$ always outperforms the second-best algorithm **AL-SVM**$_F$ **+ auto** by a wide margin (13%-40%). (3) It is worth noting that compared to 2D queries, the gap between **RDSM**$_F$ and the second-best algorithm becomes larger. On the one hand, even with factorization, the other algorithms fail to achieve high accuracy for high-dimensional queries. On the other hand, **RDSM**$_F$ takes advantage of the factorized data space model to dramatically shrink the uncertain region and capture the true decision boundary in high dimensional space rapidly. (4) For each algorithm, its accuracy decreases as $\rho_{\max}$ increases. Particularly, **RDSM**$_F$ drops at the slowest rate, which empirically attests to the robustness of **RDSM**$_F$. (5) Compared to **AL-SVM**, **AL-SVM**$_F$ improves the accuracy dramatically. Except for 50% $\rho_{\max}$, **AL-SVM**$_F$ surpasses **AL-SVM** by around 40% accuracy, which shows the benefit of factorization. (6) Similar to previous observations, due to the usage of *auto-cleansing*, **AL-SVM**$_F$ **+ auto** outperforms **AL-SVM**$_F$ when $\rho_{\max}$ is high, but these two methods are similar for lower $\rho_{\max}$.

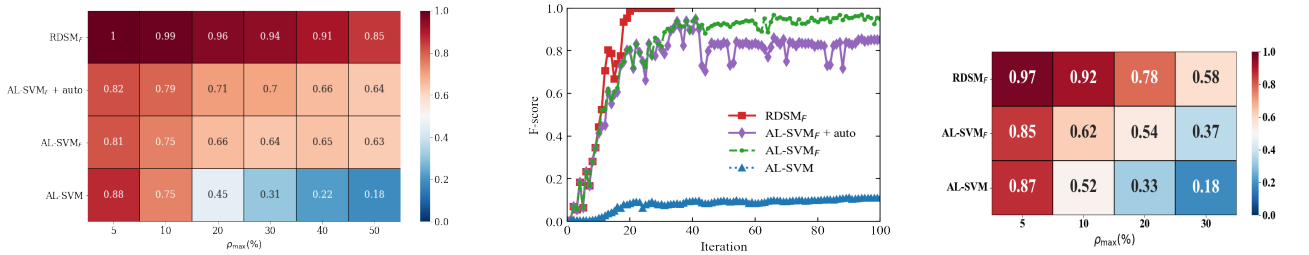
We next show some details in terms of the F1-score and time measurement. Since the experimental results have similar trends for all high-dimensional queries and the 4D query Q5 (0.01%) and the 6D query Q6 (0.01%) have the



(a) Q5 (0.01%), $\rho_{\max}$ 5%, F-score (b) Q5 (0.01%), $\rho_{\max}$ 30%, F-score (c) Q5 (0.01%), Time

(d) Q6 (0.01%), $\rho_{\max}$ 5%, F-score (e) Q6 (0.01%), $\rho_{\max}$ 30%, F-score (f) Q6 (0.01%), Time

Fig. 11: **RDSM$_F$** for SDSS 4D and 6D queries, compared to **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM**



(a) Averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over all 18 car queries at iteration 100

(b) Car query Q5, $\rho_{\max}$ 40%, F-score

(c) Averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$**, and **AL-SVM** over 5 car queries with false positive labels at iteration 100

Fig. 12: Results for the user study in the presence of label noise.

lowest selectivity (lower selectivity typically leads to more difficulties in learning), we take the two queries as examples to show the general trends. As shown in Fig. 11(a)-11(b) and 11(d)-11(e), **RDSM$_F$** outperforms the other algorithms after 20-40 iterations, and by a wide margin for Q6. It achieves above 90% accuracy even when $\rho_{\max}$ rises up to 30%, while the other algorithms have accuracy below 80% for $\rho_{\max} = 30\%$. **AL-SVM$_F$ + auto** and **AL-SVM$_F$** perform similarly for low $\rho_{\max}$, but for high $\rho_{\max}$, **AL-SVM$_F$** drops faster than **AL-SVM$_F$ + auto**. As for **AL-SVM** its highest accuracy is below 80% for Q5(0.01%) with $\rho_{\max} = 5\%$, while in other cases, its accuracy is close to 0.

Regarding time measurement, like 2D queries, the time cost per iteration of each algorithm for high dimensional queries remains similar across different levels of label noise. Figure 11(c) and 11(f) show that **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM** have time cost per iteration below 0.5 second. The highest time cost per iteration of **RDSM$_F$** is around 2.5 seconds for Q5 (0.01%) and around 3 seconds for Q6 (0.01%). In conclusion, for all the experiments, the time cost for **RDSM$_F$** is within 3 seconds per iteration, which is reasonable for the requirement of interactive performance. As our work aims to reduce user labeling efforts, 200 labeled examples and within 3 seconds per iteration seem to be acceptable. The most time-intensive procedure is updating the three-set based data space model at each iteration. If more iterations are needed, instead of building the data space model on the entire dataset, we can further keep the iteration time within 3 seconds via subsampling methods.

Expt 7 (Learning car queries with synthetic noisy labels): In this experiment, we simulate the Boundary-Consistence Noise using the approaches described in Section 5.4, and conduct the evaluation for the 18 car queries obtained from our user study. Figure 12(a) shows the averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over these queries. The main results include: (1) Consistent with the results of SDSS queries, **RDSM$_F$** substantially outperforms other algorithms under

every noise rate. In particular, **RDSM**$_F$ improves the performance from **AL-SVM** by 12%-69% (48% on average). (2) **RDSM**$_F$ is the most robust one whose accuracy drops at the slowest pace as the noise rate rises. It achieves 85% when $\rho_{\max}$ is up to 50%, while other algorithms, no matter the noise rate is low or high, cannot reach 85% in most cases. (3) The considerable improvement from **AL-SVM**$_F$ **+ auto** to **RDSM**$_F$ can be attributed to the factorized data space model. (4) For each noise rate, **AL-SVM**$_F$ **+ auto** performs slightly better than **AL-SVM**$_F$. (5) For lower noise rates, **AL-SVM** achieves better performance than **AL-SVM**$_F$ while for higher noise rates, **AL-SVM**$_F$ significantly outperforms **AL-SVM**.

Among all car queries, Q5 (0.231%) has the highest dimensionality, including 418 attributes after one-hot encoding of the categorical attributes. In Figure 12(b), we show the trends of the algorithms for the car query Q5 using an example $\rho_{\max} = 40\%$. Consistently, we observe that **RDSM**$_F$ significantly outperforms other algorithms. **AL-SVM**$_F$ **+ auto** and **AL-SVM**$_F$ have similar performances, and their accuracies lie in between **RDSM**$_F$ and **AL-SVM**.

Expt 8 (Learning car queries with real noisy labels): We next evaluate our algorithm using real noisy labels from our Car User Study. However, this study has some constraints: the user recorded only "positive examples" during initial exploration (Step 2) and at the end of manual exploration (Step 3) when the user verified the final results before terminating the exploration. So, we can treat the examples returned by the final query as the true positive examples. In contrast, during initial exploration, the user was not very clear about her true interest and might provide false positive examples. As such, this study provides only limited noisy labels in the initial exploration phase and in the form of false positives.

To generalize the noisy labels to all iterations in our experiment, we use label propagation to characterize the false positive region. More specifically, we obtain false positive examples from the initial exploration in the user trace, preprocess our dataset by propagating noisy labels to the k nearest neighbors of the false positives, and execute the data exploration on the polluted dataset to test the robustness of our model. In the process of label propagation, we flip the labels only for the negative examples in the neighborhood of the false positive examples under the assumption that the examples close to the false positives are more likely to be positive. We tune the value of k to control the label noise incurred in the data exploration to obtain a percentage error rate.

As shown in Figure 12(c), **RDSM**$_F$consistently outperforms the other algorithms by a large margin (14%-59% times higher), and **AL-SVM**$_F$ beats **AL-SVM** in most cases except in the 5% case. The scores in Expt 8 are not comparable to those in Expt 7 due to different distributions of label noise: In Expt 7, the label noise follows a Gaussian distribution. In Expt 8, to ensure that the k-nearest neighbors are actually hit as noisy labels in iterative data exploration, we impose a Bernoulli distribution on the label noise: if an example is in the k-nearest neighbor set, it is a noisy label; otherwise, it is not. The noise in Expt 8 often occurs close to the decision boundary, while the noise in Expt 7 may occur far from the boundary – hence the former has a more severe impact on performance than the latter. Overall, the results in Expt 8 demonstrate not only the robustness of our model to real noise, but also the capability of our model to address different types of noisy labels.

7 Related Work

Data Exploration. *Faceted search* iteratively recommends query attributes for drilling down into the database, but the user is often asked to provide attribute values until the desired tuple(s) are returned [45, 61, 62] or offer an "interestingness" measure and its threshold [21]. Semantic windows [44] are pre-defined multidimensional predicates that a user can explore. These methods are different from our active learning approach. Recent work also supports time series data [54] or avoids false discoveries of statistical patterns [84] during interactive data exploration. Finding best database objects based on user preferences [70] assumes a numeric weight per database attribute, which is different from the active learning approach to discover the user interest on the fly. The kBM-IGS framework proposed in [85] handles interactive search for multiple labels using a constrained budget. While our work only considers single label cases, the extension for multiple labels will be a promising direction. By interactive data exploration, the DExPLORER system in [58] aims to discover and rank examples and the EDA4Sum system in [81] detects highly uniform and diverse itemsets to provide data summaries, which are different from our scenario of finding desired examples. The ATENA system in [26] auto-generates EDA notebooks to offer the user more insights before he/she actively explores the dataset, which is complementary to our work. Recent work [55] proposes GUIDES, an innovative framework for guided Text-based Item Exploration (TIE), to explore structured and unstructured data in tandem and provide guidance in the dual space of attributes and text. However, GUIDES leverages an action-driven approach, example-based TIE operators are yet to be fully explored. A new query workload augmentation with labeling estimation framework DATAFARM [78] also adopts active learning for label forecasting, but it focuses on generating query workloads with labels, which is different from our goal. Beyond the requirements of regular data exploration, the authors of [7] address the joint exploration of items, people, and people's opinions on items in the Subjective Data Exploration (SDE) framework. Instead of finding interesting data sets, the approach in [65] aims to find the target users through guided exploration.

Query by Example is a specific framework for data exploration. Earlier work on QBE focused on a visualization front-end that aims to minimize the user effort to learn the SQL syntax [42, 56]. Recent work [53] proposes exemplar queries which treat a *query* as a sample from the desired result set and retrieve other tuples based on similarity metrics, but for graph data only. The work [67] considers data warehouses with complex schemas and learns the minimal project-join queries from a few example tuples efficiently. It does not consider selection with complex predicates, which is a focus of our work. The work [43] helps users construct join queries for exploring relational databases, and [47] does so by asking the user to determine whether a given output table is the result of her intended query on a given database. INODE [6] as an end-to-end data exploration system also provides explore-by-example approaches for finding interesting data patterns across multiple semantics, but it adopts existing approaches rather than developing new methods.

Query Formulation has been surveyed in [18]. The closest to our work is LifeJoin [19], which we compared in Section 6.2. Query By Output (QBO) [74] takes the output of one query on a database, and constructs another query such that running these two queries on the database are instance-equivalent. Dataplay [2] provides a GUI for users to directly construct and manipulate query trees. It assumes that the user can specify the value assignments used in his intended query, and learns conjunctions of quantified Horn expressions (with if-then semantics) over nested relations [1].

Active Learning. Tong and Koller [73] provide a theoretical motivation on selecting new examples using the notion of a version space, but with unknown convergence speed. Related to our work is a lower bound on the probability of misclassification error on the unlabeled training set [16]. However, it relies on user labeling of an additional sample from the unlabeled pool, which is not required in our work. Recent papers [27, 35–37] offer probabilistic bounds for the classification error and sample complexity. Our work differs in that we focus on F1-score, which suits selective user interest queries (imbalanced classes in classification).

Most learning theory makes no assumptions on convex data/class distributions [38]. Clustering techniques can assume that data is clustered in convex sets [38], but address a different problem from ours. In Active Learning, convexity assumptions occur in the Version Space [10, 73], which is the set of classifiers consistent with training data. **DSM** can embrace any classifier developed through the version space, but also includes the new polytope model.

Active Search [33, 50, 75] aims to maximize the number of positive examples discovered, called the *Target Set Accuracy*, within a limited budget of user labeling effort. In comparison, our work aims to maximize the *F-score* of the model learned to approximate the true user interest. We reported the performance difference from [33] in the previous section. The works [50, 75] use a kernel function to measure similarity of items in order to maximize the utility of a set of selected items. Such kernel methods have the smoothness requirement, i.e., similar items have similar utility values, and require training data to tune the kernel for each use case (user interest), which do not suit our problem setting.

Crowdsourcing-based Active Learning with Label Noise. Many existing methods in the active learning literature address the label noise problem based on crowdsourcing techniques [3, 41, 48, 82]. However, these techniques depend on adding redundancy by collecting labels for each example from multiple users and aggregating the labels using some methods such as majority voting. In this context, these crowdsourcing techniques usually result in an additional labeling cost and can not support the scenarios with a single user. While our work assumes that only one user is involved, if combined with some crowdsourcing techniques to aggregate labels from the users, our proposed algorithms have the potential to support multiple users.

Active Learning with Relabeling Noisy Labels. In the active learning literature, a popular approach to deal with label noise is to relabel suspiciously mislabeled examples. Recent works [14, 83] treat relabeling as an individual process triggered at each iteration or under some conditions. Other works [41, 48] integrate relabeling into the sample acquisition process, which generalizes conventional sampling strategies with a tradeoff between assigning labels for more unlabeled examples and relabeling potentially noisy examples in the existing labeled data set to improve the label quality. Instead of relabeling, our work takes advantage of noise removal methods. On the one hand, label noise cleansing methods have the following advantages: (1) removed noisy examples have no effect on modeling [32], (2) it's observed that retaining mislabeled examples may be more harmful than removing some correctly labeled examples [15], and (3) in general, removing noisy examples is more effective than relabeling them and a hybrid approach which combines removal and relabeling [20, 52]. On the other hand, relabeling may be too costly since humans may make similar mistakes repeatedly [71], and there is no guarantee that the relabeled examples are noise-free. Recent work [71] proposed a *Bidirectional Active Learning with human Training* (BALT) model to improve the expertise of labelers and relabeling quality simultaneously. However, the BALT model relies on some *gold instances* (previously labeled and noise-free) which are not available in our setting.

Data Programming. When ground truth labels are not available, recent studies [8, 60, 76] have employed the *data programming* paradigm to build training data for developing high-quality models. Furthermore, Snorkel [59], a first end-to-end system based on *data programming*, is designed for training machine learning models with weak supervision. Snorkel asks the users to write *labeling functions* (LFs)

that express various weak supervision sources (e.g., domain expertise) and learns an accurate model through weak supervision on LFs. However, the typical LFs development cycles include multiple iterations of ideation, refining, evaluation, and debugging, which can be very time-consuming and expensive. In addition, it may be unrealistic to request a reasonable set of LFs from non-expert users. Even though recent work [77] proposed an approach to generate LFs using an initially labeled dataset automatically, the size of the labeled datasets needed reaches a magnitude of hundreds.

Data Discovery. Recent work considered correlated dataset search by capturing the relationships between datasets. Aurum [30] leverages an enterprise knowledge graph (EKG) to retrieve relevant datasets for a user-provided discovery query. The recent work [63] proposes a QCR index-based approach to efficiently support correlated table search for generalized join-correlation queries. COCOA [29] accelerates the calculation of correlation coefficients to select the most correlated features for user-defined machine learning tasks efficiently. The goal of our work is different: we aim to find examples of user interest in a dataset, rather than identify the relationships between datasets/tables and obtain the most correlated ones. However, these methods can be used to discover correlated features in our future work.

8 Conclusion and Future Work

We presented new algorithms for interactive data exploration in the active learning framework, which leverage the subspatial convex and conjunctive properties of database queries to overcome the slow convergence problem and the label noise problem. Our main results include the following: (1) In the absence of label noise, our algorithm significantly outperforms active learning [13, 28], active search [33], and existing explore-by-example systems including Aide [23, 24] and LifeJoin [19], in accuracy and convergence speed. (2) For bounded instance- and label-dependent label noise, our algorithm substantially improves accuracy and convergence speed, while other alternative algorithms fail to handle noisy labels. (3) Our algorithm maintains the per-iteration time within 1-3 seconds.

In future work, we plan to extend query patterns to multiple disjoint areas using exploration versus exploitation. Toward this goal, a possible approach is to explore the entire dataset to identify possible subregions, partition the data space based on the detected subregions such that each partition contains only one subregion, and then apply our algorithm in each partition respectively. It is also interesting to tackle noisy labels intrinsically by building probabilistic convex hulls. Another direction for future work is to generalize our techniques from database exploration, where subspatial convexity and conjunctivity properties are likely to hold, to more general active learning problems, e.g., for text/image classification, and understand their applicability and potential benefits in those settings.

References

1. A. Abouzied, D. Angluin, et al. Learning and verifying quantified boolean queries by example. In *PODS*, pages 49–60, 2013.
2. A. Abouzied, J. M. Hellerstein, and A. Silberschatz. Playful query specification with dataplay. *PVLDB*, 5(12):1938–1941, 2012.
3. A. Agarwal, R. Garg, and S. Chaudhury. Greedy search for active learning of OCR. In *ICDAR*, pages 837–841, 2013.
4. C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *ICDT*, page 420–434, Berlin, Heidelberg, 2001. Springer.
5. R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *ECML*, pages 39–50, 2004.
6. S. Amer-Yahia, et al. INODE: building an end-to-end data exploration system in practice. *SIGMOD Rec.*, 50(4):23–29, 2021.
7. S. Amer-Yahia, T. Milo, and B. Youngmann. Exploring ratings in subjective databases. In *SIGMOD*, pages 62–75. ACM, 2021.
8. S. H. Bach, B. He, A. Ratner, and C. Ré. Learning the structure of generative models without labeled data. In *ICML*, 273–282, 2017.
9. C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *TOMS*, 22(4):469–483, 1996.
10. K. Bellare, S. Iyengar, A. Parameswaran, and V. Rastogi. Active sampling for entity matching with guarantees. *ACM Transactions on Knowledge Discovery from Data*, 7(3):12:1–12:24, Sept. 2013.
11. A. Berthon, B. Han, G. Niu, T. Liu, and M. Sugiyama. Confidence scores make instance-dependent label-noise learning possible. *arXiv preprint arXiv:2001.03772*, 2020.
12. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.
13. A. Bordes, S. Ertekin, J. Weston, et al. Fast kernel classifiers with online and active learning. *JMLR*, 6:1579–1619, Dec. 2005.
14. M. Bouguelia, S. Nowaczyk, K. C. Santosh, and A. Verikas. Agreeing to disagree: active learning with noisy labels without crowdsourcing. *IJMLC*, 9(8):1307–1319, 2018.
15. C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *J. Artif. Intell. Res.*, 11:131–167, 1999.
16. C. Campbell, N. Cristianini, and A. J. Smola. Query learning with large margin classifiers. In *ICML*, pages 111–118, 2000.
17. J. Cheng, T. Liu, et al. Learning with bounded instance and label-dependent label noise. In *ICML*, pages 1789–1799. PMLR, 2020.
18. A. Cheung and A. Solar-Lezama. Computer-assisted query formulation. *Foundations and Trends® in Programming Languages*, 3(1):1–94, June 2016.
19. A. Cheung, A. Solar-Lezama, et al. Using program synthesis for social recommendations. In *CIKM*, 1732–1736. ACM, 2012.
20. S. Cuendet, D. Hakkani-Tür, et al. Automatic labeling inconsistencies detection and correction for sentence unit segmentation in conversational speech. *Int'l Workshop on MLMI*, 144–155, 2007.
21. D. Dash, J. Rao, N. Megiddo, et al. Dynamic faceted search for discovery-driven analysis. In *CIKM*, 3–12, 2008.
22. Y. Diao, et al. AIDE: an automatic user navigation system for interactive data exploration. *PVLDB*, 8(12):1964–1967, 2015.
23. K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: an automatic query steering framework for interactive data exploration. In *SIGMOD*, pages 517–528, 2014.
24. K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: an active learning-based approach for interactive data exploration. *TKDE*, 28(11):2842–2856, 2016.
25. J. Du and Z. Cai. Modelling class noise with symmetric and asymmetric distributions. In *AAAI*, pages 2589–2595, 2015.
26. O. B. El, T. Milo, and A. Somech. Automatically generating data exploration sessions using deep reinforcement learning. In *SIGMOD*, pages 1527–1537. ACM, 2020.

27. R. El-Yaniv and Y. Wiener. Active learning via perfect selective classification. *JMLR*, 13(1):255–279, Feb. 2012.
28. S. Ertekin, J. Huang, et al. Learning on the border: Active learning in imbalanced data classification. In *CIKM*, 127–136, 2007.
29. M. Esmailoghli, J. Quiané-Ruiz, et al. COCOA: correlation coefficient-aware data augmentation. In *EDBT*, 331–336, 2021.
30. R. C. Fernandez, Z. Abedjan, et al. Aurum: A data discovery system. In *ICDE*, 1001–1012, 2018.
31. B. Frénay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 25(5):845–869, 2014.
32. D. Gamberger, N. Lavrač, and S. Džeroski. Noise elimination in inductive concept learning: A case study in medical diagnosis. In *International Workshop on ALT*, pages 199–212. Springer, 1996.
33. R. Garnett, et al. Bayesian optimal active search and surveying. In *ICML*, pages 843–850, 2012.
34. B. Grünbaum. Convex polytopes. In *Convex Polytopes*. Springer-Verlag New York, 2 edition, 2003.
35. S. Hanneke. Rates of convergence in active learning. *The Annals of Statistics*, 39(1):333–361, 02 2011.
36. S. Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014.
37. S. Hanneke. Refined error bounds for several learning algorithms. *Journal of Machine Learning Research*, 17(1):4667–4721, 2016.
38. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
39. E. Huang. *Active Learning Methods for Interactive Exploration on Large Databases*. Theses, Institut Polytechnique de Paris, 2021.
40. E. Huang, L. Peng, et al. Optimization for active learning-based interactive database exploration. *PVLDB*, 12(1):71–84, 2018.
41. P. G. Ipeirotis, et al. Repeated labeling using multiple noisy labelers. *Data Min. Knowl. Discov.*, 28(2):402–441, 2014.
42. B. E. Jacobs and C. A. Walczak. A Generalized Query-by-Example Data Manipulation Language Based on Database Logic. *IEEE Transactions on Software Engineering*, 9(1):40–57, 1983.
43. M. Kahng, et al. Interactive browsing and navigation in relational databases. *PVLDB*, 9(12):1017–1028, 2016.
44. A. Kalinin, U. Cetintemel, and S. Zdonik. Interactive data exploration using semantic windows. In *SIGMOD*, 505–516, 2014.
45. N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and Interactive Cube Exploration. In *ICDE*, pages 472–483, 2014.
46. S. R. Lay. *Convex Sets and Their Applications*. Dover Publications, 2007.
47. H. Li, C.-Y. Chan, and D. Maier. Query from examples: An iterative, data-driven approach to query construction. *PVLDB*, 8(13):2158–2169, 2015.
48. C. H. Lin, Mausam, and D. S. Weld. Re-active learning: Active learning with relabeling. In *AAAI*, pages 1845–1852, 2016.
49. W. Liu, Y. Diao, and A. Liu. An analysis of query-agnostic sampling for interactive data exploration. *Communications in Statistics-Theory and Methods*, 47(16):3820–3837, 2018.
50. Y. Ma, R. Garnett, and J. G. Schneider. Σ-optimality for active learning on gaussian random fields. In *NIPS*, 2751–2759, 2013.
51. A. K. Menon, et al. Learning from binary labels with instance-dependent noise. *Machine Learning*, 107(8):1561–1595, 2018.
52. A. L. B. Miranda, et al. Use of classification algorithms in noise detection and elimination. In *HAIS*, 417–424, 2009.
53. D. Mottin, et al. Exemplar queries: Give me an example of what you need. *PVLDB*, 7(5):365–376, 2014.
54. R. Neamtu, et al. Interactive time series analytics powered by ONEX. In *SIGMOD*, pages 1595–1598, 2017.
55. B. Omidvar-Tehrani, A. Personnaz, et al. Guided text-based item exploration. In *CIKM*, 3410–3420. ACM, 2022.
56. G. Özsoyoglu and H. Wang. Example-Based Graphical Database Query Languages. *Computer*, 26(5):25–38, 1993.
57. L. D. Palma. *New Algorithms and Optimizations for Human-in-the-Loop Model Development*. PhD thesis, Polytechnic Institute of Paris, France, 2021.
58. X. Qin, et al. Interactively discovering and ranking desired tuples by data exploration. *VLDB J.*, 31(4):753–777, 2022.
59. A. Ratner, et al. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282, 2017.
60. A. J. Ratner, et al. Data programming: Creating large training sets, quickly. In *NeurIPS*, pages 3567–3575, 2016.
61. S. B. Roy, et al. Minimum-effort driven dynamic faceted search in structured databases. In *CIKM*, pages 13–22, 2008.
62. S. B. Roy, et al. Dynacet: Building dynamic faceted search systems over databases. In *ICDE*, pages 1463–1466, 2009.
63. A. S. R. Santos, et al. A sketch-based index for correlated dataset search. In *ICDE*, pages 2928–2941, 2022.
64. C. Scott, et al. Classification with asymmetric label noise: Consistency and maximal denoising. In *COLT*, 489–511, 2013.
65. M. Seleznova, et al. Guided exploration of user groups. *PVLDB*, 13(9):1469–1482, 2020.
66. B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence & Machine Learning. Morgan Claypool Publishers, 2012.
67. Y. Shen, et al. Discovering queries based on example tuples. In *SIGMOD*, pages 493–504, 2014.
68. Sloan digital sky survey: DR8 sample SQL queries. `http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp`.
69. A. Szalay, et al. Designing and mining multi-terabyte astronomy archives: The Sloan digital sky survey. *SIGMOD*, 451–462, 2000.
70. B. Tang, et al. Determining the impact regions of competing options in preference space. In *SIGMOD*, pages 805–820, 2017.
71. F. Tang. Bidirectional active learning with gold-instance-based human training. In *IJCAI*, pages 5989–5996, 2019.
72. Y. Tang, et al. Svms modeling for highly imbalanced classification. *IEEE Trans. Syst. Man Cybern. Part B*, 39(1):281–288, 2009.
73. S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:45–66, Mar. 2002.
74. Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *SIGMOD*, pages 535–548, 2009.
75. H. P. Vanchinathan, et al. Discovering valuable items from massive data. In *SIGKDD*, pages 1195–1204, 2015.
76. P. Varma, et al. Inferring generative model structure with static analysis. In *NeurIPS*, pages 240–250, 2017.
77. P. Varma and C. Ré. Snuba: Automating weak supervision to label training data. *Proc. VLDB Endow.*, 12(3):223–236, 2018.
78. F. Ventura, et al. Expand your training limits! generating training data for ml-based data management. *SIGMOD*, 1865–1878, 2021.
79. B. C. Wallace and I. J. Dahabreh. Class probability estimates are unreliable for imbalanced data (and how to fix them). In *ICDM*, pages 695–704. IEEE Computer Society, 2012.
80. G. Wu and E. Y. Chang. KBA: kernel boundary alignment considering imbalanced data distribution. *IEEE Trans. Knowl. Data Eng.*, 17(6):786–795, 2005.
81. B. Youngmann, S. Amer-Yahia, and A. Personnaz. Guided exploration of data summaries. *PVLDB*, 15(9):1798–1807, 2022.
82. J. Zhang, X. Wu, et al. Active learning with imbalanced multiple noisy labeling. *IEEE Trans. Cybern.*, 45(5):1081–1093, 2015.
83. X. Zhang, S. Wang, and X. Yun. Bidirectional active learning: A two-way exploration into unlabeled and labeled data set. *IEEE Trans. Neural Networks Learn. Syst.*, 26(12):3034–3044, 2015.
84. Z. Zhao, et al. Controlling false discoveries during interactive data exploration. In *SIGMOD*, pages 527–540, 2017.
85. X. Zhu, et al. Budget constrained interactive search for multiple targets. *PVLDB*, 14(6):890–902, 2021.