Active Learning Methods for Interactive Exploration on Large Databases

Enhui Huang

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAX046

Thèse de doctorat

ÉCOLE POLYTECHNIQUE

IP PARIS

Active Learning Methods for Interactive Exploration on Large Databases

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École Polytechnique

École doctorale n°626 École Doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 06/07/2021, par

ENHUI HUANG

Composition du Jury :

Themis Palpanas Professor, University of Paris	Président & Rapporteur
Olga Papaemmanouil Associate Professor, Brandeis University	Rapporteur
Fabian Suchanek Professor, Télécom Paris University	Examinateur
Yannis Velegrakis Professor, Utrecht University	Examinateur
Yanlei Diao Professor, École Polytechnique	Directeur de thèse
Anna Liu Professor, University of Massachusetts	Co-Directeur de thèse

Acknowledgments

Throughout the journey of this Ph.D., I have received a great deal of support and assistance from many people.

First and foremost, I would like to express my sincere gratitude to my supervisors, Prof. Yanlei Diao and Prof. Anna Liu. Their excellent expertise and extensive experience have always encouraged me, not only in academic research but also in daily life. Without their invaluable guidance, continuous support, and patience, this thesis would not be possible.

It is my honor to have Prof. Themis Palpanas, Prof. Olga Papaemmanouil, Prof. Fabian Suchanek, and Prof. Yannis Velegrakis to be my committee members. I am deeply grateful for their insightful comments and suggestions.

Many thanks go to Prof. Quanhua Xu, who gave me the opportunity to study in France and helped me a lot. Special thanks go to Liping Peng and Kyriaki Dimitriadou, from whom I learned experiences that helped me start my Ph.D. journey more easily.

I would also like to thank my colleagues at École Polytechnique. I particularly thank Luciano Di Palma and Ahmed Abdelkafi for their close collaboration. I am grateful to all the CEDAR members for the enlightening conversations and the enjoyable moments we shared.

Furthermore, I would like to thank my friends, Min Wang, Yu Chen, Chao Yu, Xiaodan Lyu, Lyuke Zhu, Tien-Duc Cao, He Zhang, Chunyue Zheng, Haonan Zhang, Bingjie Li, Zehao Guan, Xueying Zheng, Lichen Li, and everyone else who stand by me through good and bad times, for their encouragement over the years.

Finally, I would like to express my heartfelt appreciation to my dear family, my father Mingquan Huang, my mother Yuefei Huang, and my brother Tianpeng Huang, for their unwavering love, support, and encouragement.

Abstract

Faced with an increasing gap between fast growth of data and limited human ability to comprehend data, data analytics tools are now in high demand in many applications across a broad set of domains. In particular, for interactive data exploration systems, an "explore-by-example" framework, which aims to assist the user in performing highly effective data exploration while minimizing the human effort, is becoming increasingly popular. However, the state-of-the-art explore-by-example systems still require a large number of labeled examples to achieve the desired accuracy and cannot handle noisy labels. To address both the slow convergence problem and the label noise problem, in this thesis, we cast the explore-by-example problem in a principled "active learning" framework, and bring the properties of important classes of the user interest to bear on the design of new algorithms and optimizations for active learning-based data exploration.

While recent work proposed a polytope-based data space model to filter examples returned by a traditional active learner and to compute an accuracy-based stopping criterion for active learning, our first contribution is to combine the polytope-based model and a traditional active learner into a new Dual-Space Model (**DSM**), jointly offering the prediction and sample acquisition functionalities and thereby expediting model convergence. We also provide a set of techniques to improve the interactive performance such that the per-iteration time is maintained within 1 to 2 seconds. Evaluation results using real-world datasets and user interest patterns show that the accuracy of our system is on average 26% higher than the state-of-the-art explore-by-example systems.

Our second contribution is to overcome the slow convergence problem when data exploration is performed in high dimensions. We factorize the high-dimensional space into a set of

low-dimensional spaces, build a **DSM** in each subspace and combine them to be a factorized **DSM** (**DSM**$_F$). We further prove that **DSM**$_F$ achieves a better accuracy-based stopping criterion than **DSM**. In case that the user may start exploration with more attributes than those needed in the final model, we propose an online feature selection method that adaptively selects the top-k relevant attributes. Experimental results using real-world workloads show that our system significantly outperforms the state-of-the-art explore-by-example systems in accuracy (19x$\sim$64x accuracy improvement) and convergence speed while maintaining the per-iteration time within 1 to 2 seconds and our online feature selection method improves accuracy from nearly 0 without feature selection, to above 80%.

Last but not least, we address the label noise problem when the labels provided by the user are potentially corrupted. We enhance the robustness of our system by integrating advanced data cleansing methods and a refinement of the polytope-based model into **DSM**$_F$. The resulting algorithm, referred to as the Robust Dual-Space Model (**RDSM**$_F$), is compared to traditional active learning for evaluation since the state-of-the-art explore-by-example systems fail to deal with noisy labels. Experimental results using real-world datasets and user interest patterns show that our proposed algorithm substantially outperforms traditional active learning in accuracy (up to 22x accuracy improvement) while achieving reasonable efficiency for interactive data exploration (1 to 3 seconds per iteration).

Resumé

Face à un écart grandissant entre la croissance rapide des données et la capacité humaine limitée à comprendre les données, les outils d'analyse de données sont en forte demande dans de nombreuses applications à travers un large éventail de domaines. En particulier, pour les systèmes interactifs d'exploration de données, un cadre « explorer par l'exemple », qui vise à aider un utilisateur humain à effectuer une exploration de données très efficace tout en minimisant son effort, devient de plus en plus populaire. Cependant, ces systèmes de pointe nécessitent toujours un grand nombre d'exemples étiquetés pour obtenir la précision souhaitée et ne peuvent pas gérer les étiquettes bruyantes. Pour résoudre à la fois le problème de la convergence lente et le problème du bruit des étiquettes, dans cette thèse, nous plaçons le problème d'« explorer par l'exemple » dans un cadre d'apprentissage actif fondé sur des principes et apportons les caractéristiques des classes importantes de l'intérêt de l'utilisateur, pour contribuer à la conception de nouveaux algorithmes et à l'optimisation pour l'exploration de données basée sur l'apprentissage actif.

Alors que des travaux récents ont proposé un modèle d'espace de données basé sur les polytopes pour filtrer les exemples renvoyés par un apprenant actif traditionnel et pour calculer un critère d'arrêt basé sur la précision pour l'apprentissage actif, notre première contribution vise à combiner le modèle basé sur les polytopes et l'apprenant actif traditionnel en un nouveau modèle à double espace (**DSM**), qui offre conjointement les fonctionnalités de prédiction et d'acquisition d'échantillons et ainsi l'accélération de la convergence des modèles. D'ailleurs, nous introduisons un ensemble de techniques pour améliorer les performances interactives de telle sorte que le temps par itération est maintenu entre 1 à 2 secondes. Les résultats de l'évaluation utilisant des jeux de données du monde réel et des

patterns d'intérêt des utilisateurs montrent que la précision de notre système est en moyenne 26% supérieure à celle des systèmes d'explorer par l'exemple de pointe.

Notre deuxième contribution consiste à surmonter le problème de la convergence lente lorsque l'exploration des données est effectuée en haute dimension. Nous factorisons l'espace de grande dimension en un ensemble d'espaces de faibles dimensions, construisons un **DSM** dans chaque sous-espace et les combinons pour former un **DSM** factorisé (**DSM**$_F$). Nous prouvons en outre que **DSM**$_F$ atteint un meilleur critère d'arrêt basé sur la précision que **DSM**. Dans le cas où l'utilisateur veut commencer l'exploration avec plus d'attributs que ceux requis dans le modèle final, nous proposons une méthode de sélection de caractéristique en ligne qui donne de manière adaptative les k attributs les plus pertinents. Les résultats expérimentaux montrent que notre système surpasse les systèmes de pointe en termes de précision (entre x19 et x64) et de vitesse de convergence tout en maintenant le temps par itération entre 1 à 2 secondes et notre méthode de sélection de caractéristique en ligne améliore la précision de près de 0 (sans sélection) jusqu'à plus de 80%.

Enfin, nous abordons le problème du bruit lorsque les étiquettes fournies par l'utilisateur sont potentiellement erronées. Nous renforçons la robustesse de notre système en intégrant des méthodes avancées de nettoyage de données et un raffinement du modèle basé sur les polytopes dans **DSM**$_F$. L'algorithme obtenu, appelé le modèle à double espace robuste (**RDSM**$_F$), est comparé à l'apprentissage actif traditionnel à des fins d'évaluation, car les systèmes d'explorer par l'exemple de pointe sont incapables de gérer les étiquettes bruyantes. Les résultats expérimentaux montrent que notre algorithme améliore considérablement l'apprentissage actif traditionnel en termes de précision (jusqu'à x22) tout en atteignant une efficacité raisonnable pour l'exploration interactive des données (1 à 3 secondes par itération).

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Today data is being generated at an unprecedented rate. For example, a recent IDC report predicts that the annual size of the global data will increase from 33 zettabytes in 2018 to 175 zettabytes in 2025 [1]. However, the human ability to comprehend data remains as limited as before. Consequently, there has been a growing demand for data management tools that can bridge the increasing gap between fast 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*" [31, 32]. In this framework, the database content is considered as a set of tuples, and the user is interested in some of them 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*[2] that will retrieve all relevant tuples from the database.

In our work, we consider several target applications. First, when a scientist comes to explore a large sky survey database such as SDSS [84], she may not be able to express her data interest precisely. Instead, she may prefer to navigate through a region of the sky, see a

[1] See the recent report at https://www.seagate.com/our-story/data-age-2025/

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

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. Second, we 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.

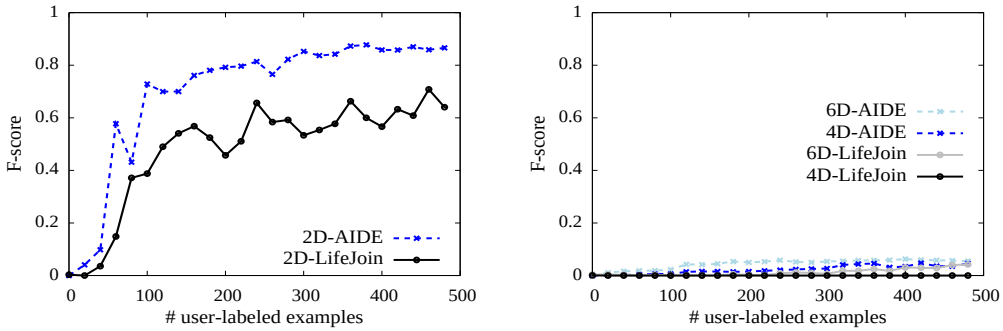
In this thesis, we propose an approach to building an explore-by-example system by casting it in an "*active learning*" framework. We treat the modeling of the user interest as a *classification* problem where all the user-labeled examples thus far are used to train a classification model. Then active learning [15, 78, 87] decides how to choose new examples, from the unlabeled database, for the user to label next so that the system can learn the user interest as fast as possible.

1.1 Technical Challenges

While prior work on explore-by-example has used active learning [32], we observe that direct application of active learning theory on large databases yields poor performance. The main technical challenges are as follows.

- *the slow convergence problem*. Existing active learning-based data exploration systems such as Aide [31, 32] and LifeJoin [25] need a large number of labeled examples to achieve high accuracy. For example, as shown in Figure 1.1(a) for a 2D query with 0.1% selectivity, LifeJoin cannot achieve 80% accuracy with 500 labeled examples, and Aide requires 300-500 labeled examples to reach 80% accuracy, which is undesirable in many applications.

  This slow convergence problem 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). As Figure 1.1(b) shows, for



(a) 2D query (0.1%) (b) 4D-6D queries (0.01%)

Figure 1.1: Slow convergence problem of Aide and LifeJoin systems.

4D-6D queries with 0.01% selectivity, Aide and LifeJoin fail to reach 10% accuracy even with 500 labeled examples.

- *the label noise problem.* Most existing active learning methods assume the labels provided by the user are uncorrupted [17]. However, in practice, label noise often occurs in the user labeling process due to two primary sources. First, the user does not fully understand her interest and may have trouble classifying some examples correctly, especially those close to the underlying true decision boundary. Second, some examples are mislabeled accidentally. As stated in the survey [37], the occurrence of label noise may lead to (i) deterioration of prediction performance, (ii) requirement of more labeled examples and more complex learned models, (iii) distortion of observed class distribution, and (iv) reduced quality of other related tasks, e.g., feature selection. In addition, although a few pioneer works have considered the label noise problem in the context of active learning, they often focus on random classification noise. It is challenging to deal with more general and realistic label noise models.

1.2 Main Ideas

In our 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 Sloan Digital Sky Survey [81] and a car database (described more in Section 4.3), we can indeed design new techniques that overcome or alleviate the slow convergence problem and the label noise problem. Some of the key query properties include:

Subspatial Convexity: Consider the database attributes as dimensions of a data space $\mathscr{D}$ 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 user interest region in $\mathscr{D}$. We observe that in some lower-dimensional subspaces of $\mathscr{D}$, the projected user interest region or its complement is a convex object. For example, the SDSS query trace [81] includes 116 predicates, among which 107 predicates define a convex user interest region in the subspace formed by the attributes used in the predicate, and 3 predicates define a convex region that is the complement of the user interest region in their subspaces. For the car database, the 70 predicates defined on numerical attributes all form a convex user interest region.

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 [81], 40 queries are conjunctive queries. For the car database, all user queries use the conjunctive property.

In this thesis, we bring the subspatial convexity and conjunctive properties 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 data exploration. More specifically, we propose a dual-space model based on the subspatial convexity property to overcome the slow convergence problem in low dimensions. Then by leveraging the conjunctive property, we design new algorithms and optimizations to expedite the convergence for high-dimensional data exploration. Furthermore, we address the label noise problem with the aid of both properties.

1.3 Contributions

We solved the challenges mentioned above:

- *the slow convergence problem* when data exploration is performed with low selectivity of the user interest query on large databases in low dimensions (Section 1.3.1) and in high dimensions (Section 1.3.2), respectively, and
- *the label noise problem* when the labels given by the user are corrupted (Section 1.3.3).

In the following, we elaborate on our main contributions.

1.3.1 Dual-Space Model

Based on the insight that existing active learning-based explore-by-example systems require a large number of labeled examples to achieve high accuracy, we propose a new active learner by leveraging the subspatial convex property to expedite the convergence for active learning-based data exploration in low dimensions. In particular, a common strategy to deal with categorical attributes in machine learning is to transform the categorical attributes into numerical attributes using one-hot encoding and then train machine learning models on the encoded attributes. However, many models are subject to the high dimensionality of the encoded attributes, leading to poor prediction performance. Therefore, we also provide an extension of the new active learner such that it could work more efficiently for categorical attributes. In addition, we devise some optimization methods to improve the time efficiency of our system. More specifically, we make the following contributions in Chapter 3.

Dual-Space Model (Section 3.1): By leveraging the subspatial convex property, we propose a new "dual-space model" (**DSM**) to characterize the user interest in the data exploration process. The key feature is that it builds not only a classification model, $F_{\mathscr{V}}$, from labeled examples, but also a polytope model of the data space, $F_{\mathscr{D}}$. On the one hand, active learning theory improves $F_{\mathscr{V}}$ by choosing the next example that enables reduction of the version space $\mathscr{V}$ (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 $\mathscr{D}$ 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. Besides, during the process of user labeling, we leverage **DSM** to label selected examples if those examples are known by **DSM** for saving user labeling effort and expediting the convergence. We call this functionality as *auto labeling* from **DSM**. The resulting algorithm can reduce the number of labeled examples required to reach high accuracy, hence achieving faster convergence. In addition, **DSM** allows us to prove exact and approximate lower bounds on the model accuracy in terms of F-score. While active learning theory offers provable bounds on classification errors [20, 34, 41–43], it treats positive and negative classes equally. Given the low selectivity of user interest queries, e.g., 1%, a classifier that assigns all tuples to the negative class has a low error rate of 1%, but fails to return any relevant tuples. Hence, we choose to bound F-score as it emphasizes accuracy for the positive class, i.e., the relevant tuples returned to the user. Our lower bounds also offer practical benefits by allowing the system to detect model convergence.

Extension for categorical attributes (Section 3.2): We propose a categorical polytope model to handle categorical attributes better. For a categorical attribute, after one-hot encoding, we split its encoded attributes into three sets: the positive set, the negative set, and the uncertain set. In particular, the positive (negative) set records the attributes which have appeared in positive (negative) examples so far, together with their appearance frequencies. The rest of the encoded attributes are stored in the uncertain set. As for the prediction, we use the majority voting of the frequencies from the positive set and the negative set. Our **DSM** combined with this extension boost the performance of data exploration for categorical attributes.

Optimizations (Section 3.3): We employ some theoretically guaranteed sampling methods combined with distributed computing to reduce the per-iteration time of our system for large-scale datasets.

1.3.2 High-dimensional Exploration

When the user interest involves a large number of attributes, the performance of both traditional active learning and **DSM** may suffer from the "curse of dimensionality". Therefore, we employ two approaches, *factorization* and *online feature selection* , to reducing dimensionality in data exploration. Our main contributions in Chapter 4 are:

Factorization (Section 4.1): As stated above, many of the available query traces reveal that user interest queries use conjunctive predicates. 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, $\textbf{DSM}_F$, runs a **DSM** in each subspace where the convex property holds. We formally define the class of queries that $\textbf{DSM}_F$ supports, the decision function it utilizes, and prove that it achieves a better lower bound of F-score than **DSM** without factorization.

Online feature selection (Section 4.2): The user may start exploration with more attributes than those needed in the final model. The redundant attributes not included in the final model are noise in data exploration and slow down model convergence. To remove irrelevant attributes, we propose an online feature selection method that adaptively selects the top-k relevant attributes.

Evaluation (Section 4.3): All of our techniques have been implemented in a Python-based prototype system. We evaluated our system using two real datasets, Sloan Digital Sky Survey (SDSS) dataset and Car database.

The SDSS dataset [84] includes 190M tuples, for which the user interests are selective and their decision boundaries present varied complexity for detection. Our results show that $\textbf{DSM}_F$ significantly outperforms learning methods including Active Learning (AL) [15, 35] and Active Search [39], as well as recent explore-by-example systems, Aide [31, 32] and LifeJoin [25]. More concretely, the main results are: (1) Without knowing these queries in advance, $\textbf{DSM}_F$ can achieve F-score of 95% within 10 labeled examples if the decision boundary B falls in a sparse region, and otherwise requires up to 40-140 labeled examples for 2D queries and 80-160 examples for 4D-6D queries while maintaining per-iteration time

within 1-2 seconds. (2) **DSM**$_F$ achieves an average F-score of 95% with 100 labeled examples and 98% with 200 labeled examples. **DSM**$_F$ significantly outperforms (3%-237% times higher F-score) learning methods including AL, which after 200 labeled examples achieves only an average of 81% and Active Search, which fails to achieve F-score of 20% for all queries. (3) For high-dimensional exploration with irrelevant attributes, our online feature selection technique improves F-score from nearly 0 without feature selection, to high F-score ($>$0.8) by adaptively selecting the relevant features. (4) Aide and LifeJoin fail to achieve 10% F-score when the user interest involves 4 attributes or more.

Our user study using a 5622-tuple car database validates the subspatial convex property and the conjunctive property 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 AL.

1.3.3 Learning with Label Noise

Traditional active learning implicitly assumes and expects uncorrupted labels provided by the user. However, in practice, label noise often occurs during the process of manual annotation. The user may not fully understand her interest and may have trouble labeling some examples correctly, especially for the ambiguous examples close to the underlying decision boundary. In some cases, the user may just mislabel some examples occasionally. In order to improve the robustness of our active learning-based data exploration system, we explore reasonable label noise models to characterize the human annotator noise and propose some denoising methods customized for our proposed **DSM**. We also analyze the effect of factorization on the label noise problem and provide several optimization methods to improve our system in terms of accuracy, efficiency, and generality. More concretely, in Chapter 5, we make the following contributions.

Label noise models (Section 5.1 & 5.5): To the best of our knowledge, a few pioneer works have studied the label noise problem in the active learning literature, but these methods are often limited to crowdsourcing (multiple users) systems with random classification noise [46, 56, 100]. As stated in Section 5.1, we assume that only one user is involved for labeling,

and we consider *bounded instance- and label-dependent label noise (BILN)* in which the label flip probability (or label noise rate) of an example is dependent on both the example and its true label, and the label noise rates of examples are upper bounded by some values less than 1. Although our proposed techniques are geared for all kinds of BILN, for interactive data exploration in the active learning framework, we focus on evaluating our techniques under a special case of BILN, *Boundary-Consistence Noise (BCN)*. The BCN model assumes that the examples closer to the decision boundary are more likely to be mislabeled, and it is considered as a reasonable model for human annotator noise in many real-world applications, such as speech recognition, spam filter, handwritten digit recognition, and annotation tasks on Amazon's Mechanical Turk [12, 33, 60]. In Section 5.5, we present some necessary procedures to simulate reasonable label noise models.

Robust Dual-Space Model (Section 5.2): Our proposed **DSM** performs excellently when the labeled examples are uncorrupted, but it cannot well handle noisy labels. Therefore, we propose a robust **DSM** (**RDSM**) through a seamless combination of a recently proposed automatic noise distillation method [23] and our proposed **DSM**-based denoising methods. We denote the automatic noise distillation method by ***auto-cleansing***. The **auto-cleansing** method is typically robust to BILN with theoretical guarantees when provided with enough labeled examples and relatively balanced class distribution. However, it fails to identify noisy labels perfectly in our cases due to the lack of labeled examples and highly biased class distribution. So we need the **DSM**-based denoising methods to further filter out mislabeled examples. We build **RDSM** in the following steps: (1) using the **auto-cleansing** method to collect confident examples from the available labeled examples which may contain noisy labels, (2) training a classification model $F_{\mathscr{Y}}$ with distilled examples, and (3) building a data space model $F_{\mathscr{D}}$ with an additional k nearest neighbors method-based refinement. Regarding the user labeling process, we propose an adaptive method to determine the source of labels. If the existing user-labeled examples are noisier than the existing **DSM**-labeled examples, we allow **RDSM** to provide labels for selected examples, otherwise not. This adaptive auto-labeling method strategically trades off between saving user labeling effort and preventing **RDSM** from introducing label noise.

High-dimensional exploration with label noise (Section 5.3): For data exploration performed with high dimensionality, the concurrence of highly imbalanced class distribution and noisy labels makes it extremely difficult to achieve high accuracy based on only a small amount of labeled data. To address this issue, we propose to apply factorization into **RDSM**. By leveraging the conjunctive property, our **RDSM** with factorization, $\textbf{RDSM}_F$, runs a **RDSM** in each subspace, in the same way as we factorize **DSM** to obtain $\textbf{DSM}_F$. If the conjunctive property holds for the user interest, on the one hand, we prove theoretically that the positive examples are less noisy in any low dimensional subspaces than in the original high dimensional space. Therefore, learning from the subspaces enables us to acquire more precise and valuable information about positive examples. On the other hand, we explain why factorization also brings more insights into the negative examples. Eventually, we illustrate, due to factorization, the performance improvement of traditional classifiers and the active learner in the presence of label noise.

Optimizations (Section 5.4): We devise several optimization strategies to improve our system in terms of accuracy, interactive performance, and generality. To achieve higher accuracy, we automatically fine-tune hyperparameters of the active learner (e.g., C in SVM) by building a regression model between the best value of the hyperparameter and the estimated noise rate obtained from the data cleansing methods. Take the parameter C in SVM for example, in general, the higher the estimated noise rate, the smaller C the regression model returns. Regarding interactive performance, we take full advantage of an open-source system Ray[3] to parallelize our code and eventually manage to reduce the time per iteration from 20-30 seconds to a few seconds.

Evaluation (Section 5.6): Evaluation using real-world datasets (SDSS and car database) and user interest queries (from the SDSS query release and the user study) shows the following results in the presence of label noise: (1) $\textbf{RDSM}_F$ substantially outperforms alternative algorithms in accuracy for all label noise levels while achieving desired interactive performance with per-iteration time within 1-3 seconds. In particular, compared to traditional active learning, $\textbf{RDSM}_F$ achieves 0.14x-3.72x higher accuracy for Car queries. Moreover,

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

for SDSS queries, the accuracy of **RDSM**$_F$ is 0.04x-0.3x higher for two-dimensional queries, 0.88x-22.14x higher for high-dimensional queries. (2) When the noise rate increases, all algorithms have decreased performance. However, **RDSM**$_F$ drops at the slowest pace. In other words, **RDSM**$_F$ is the most robust one. (3) To demonstrates the effectiveness of our robust data space model, we compare **RDSM**$_F$ with its version space part. It turns out that **RDSM**$_F$ offers significant performance gains over its version space part: 0.22x-0.38x for car queries, 0.02x-0.11x for two-dimensional SDSS queries, and 0.15x-1.5x for high-dimensional SDSS queries. (4) Active learning combined with **auto-cleansing** usually performs better than simply active learning. (5) For data exploration performed in high dimensions, active learning with factorized classifiers usually outperforms active learning with standard classifiers. The reason behind this is that factorization helps alleviate the label noise problem.

1.4 Thesis Outline

The thesis is organized as follows. In Chapter 2, we introduce the essential background on our active learning-based interactive data exploration system and our proposed techniques. We address the slow convergence problem for active learning-based data exploration performed with low dimensionality in Chapter 3 and with high dimensionality in Chapter 4. In Chapter 5, we overcome the label noise problem when the labels provided by the user are corrupted during the interactive data exploration. In Chapter 6, we summarize this thesis and discuss the future work.

Chapter 2

System Overview and Background

In this chapter, we review our design of an active learning-based explore-by-example system, present the essential background on data exploration, explore-by-example, active learning, and label-noise learning, and introduce a recently proposed polytope-based data space model which inspired our work.

2.1 System Overview

Our data exploration system is depicted in Figure 2.1. The main concepts and modules are described as follows.

Data space. When a user comes to explore a database, she is presented with the database schema for browsing. Based on her best understanding of the (implicit) exploration goal, she may choose a set of attributes, $\{A_i\}$, $i = 1, \ldots, d$, from a table for consideration. These attributes form a superset of the relevant attributes that will eventually be discovered by the system. Let us consider the projection of the underlying table 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.

Initial examples. To bootstrap data exploration, the user is asked to give a positive example and a negative example. If she does not have such examples, the system can run

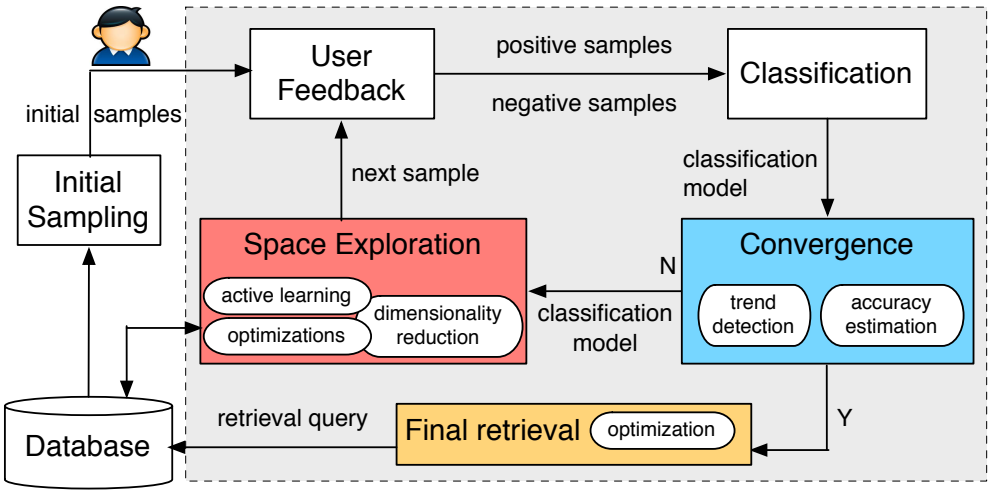


Figure 2.1: System architecture for explore by example.

initial sampling [31, 58] over the data space to help her find such examples. Since the initial sampling problem has been studied before, our work in this thesis focuses on data exploration after such initial examples are identified.

Iterative learning and exploration. The iterative exploration starts with a given positive example set and a negative example set, which form the initial labeled dataset. In each iteration, the labeled dataset is used to train a user interest model, which is fast due to the small size of training data. Before the model reaches convergence or a user-specified accuracy level, it is used next to explore the data space and retrieve a new example for display. In the next iteration, the user labels this example as positive or negative – such feedback can be collected explicitly through a graphical interface [30], or implicitly based on the user behavior.[1] The newly labeled example is added to the labeled dataset, and the above process repeats.

Convergence and final retrieval. At each iteration, our system assesses the current model to decide whether more iterations are needed. The process is terminated when the model accuracy has reached a user-defined threshold or the user stops data exploration. At this point, the model for the positive class is translated to a query which will retrieve from the database all the tuples classified as relevant. To provide better interpretability, our system

[1]Methods for collecting user feedback are in the purview of human-computer interaction and are beyond the scope of this thesis.

can visualize the final (nonparametric) model and fit a corresponding parametric model, the form of which can be suggested by the user after seeing the visualization.

Our system aims to (1) achieve accurate results with a minimum number of examples presented to the user (i.e., the amount of user labeling effort) and (2) restrict per-iteration time to within a few seconds. The per-iteration time refers to the time cost for one round of exploration, including classification, convergence detection, and space exploration.

2.2 Related Work

We survey some data exploration frameworks and techniques related to our work in Section 2.2.1, describe the active learning-based data exploration elaborately in Section 2.2.2, and eventually provide a survey of label-noise learning in Section 2.2.3.

2.2.1 Data Exploration Frameworks and Related Techniques

Our active learning-based *interactive data exploration* system aims to help the user in *query formulation* in an interactive and *example-based* way while minimizing user labeling effort. We next explore some related data exploration frameworks from different perspectives (interactive data exploration, query formulation, and example-based exploration) and recent techniques for the minimization of user labeling effort (active learning and active search).

Data exploration frameworks

Interactive data exploration has attracted a proliferation of recent interest. *Faceted search* [9, 52, 75] 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 [52, 74, 75] or offer an "interestingness" measure and its threshold [29]. *Semantic windows* [51] are predefined multidimensional shape-based and content-based predicates that a user can explore. These methods are different from our active learning approach. Recent work also supports time series data [65] or avoids false discoveries of statistical patterns [101] during interactive data exploration.

The active learning-based explore-by-example approach offers potential benefits over faceted search and semantic windows for several reasons.

First, the user interest may include varying degrees of complexity, or the user does not have prior knowledge about the complexity and hence expects the system to learn a model as complex as necessary for her interest. More specifically, if the user knows the relevant attributes and just wants to set the appropriate value for each attribute, an interactive GUI or faceted search [9, 52, 75] may be sufficient. However, the user interest may involve more complex constraints, e.g., "$(\frac{rowc-a_1}{b_1})^2+(\frac{rowc-a_2}{b_2})^2<c$", and "$x+2.5*\log_{10}(y)<23.3$" from the SDSS example query set [81], or "$length*width>c$" as seen in our car database. 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 [51] as predefined patterns. However, if the user does not have such prior knowledge, explore-by-example may work regardless of how complex the predicates are and which functions and constants are used in the predicates.

Second, increased dimensionality makes it harder for semantic windows to scale. For example, when the interest of the SDSS user involves both the ellipse and log patterns above, it will be more difficult for both the system and the user to handle multiple patterns for data exploration (even 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".

Query formulation has been surveyed in [24]. The closest to our work is LifeJoin [25], which we compared in Section 4.3.3. Query By Output (QBO) [88] 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, which are assumptions that we cannot make, and learns conjunctions of quantified Horn expressions (with if-then semantics) over nested relations [1].

Example-based exploration is a specific framework for data exploration which removes the need for the user to specify a query. Earlier work on example-based exploration focused on

a visualization front-end that aims to minimize the user effort to learn the SQL syntax [48, 68]. Recent work [63] 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 [79] 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 [50] helps users construct join queries for exploring relational databases, and [54] does so by asking the user to determine whether a given output table is the result of her intended query on a given database.

For a more comprehensive survey of example-based exploration, see [57].

Related techniques for minimizing user labeling effort

The following techniques - active learning and active search - have been widely used to minimize user labeling effort for data exploration.

Active Learning. Tong and Koller [87] 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 [20]. However, it relies on user labeling of an additional sample from the unlabeled pool, which is not required in our work. Recent papers [34, 41–43] offer probabilistic bounds for the classification error and sample complexity. Our work differs in that we focus on F-score, which suits selective user interest queries (imbalanced classes in classification).

Most learning theory makes no assumptions on convex data/class distributions [44]. Clustering techniques can assume that data is clustered in convex sets [44], but address a different problem from ours. In Active Learning, convexity assumptions occur in the Version Space [11, 87], which is the set of classifiers consistent with training data. Our algorithm can embrace any classifier developed through the version space, but also includes the new polytope model. In Section 2.2.2, we comprehensively show the essential parts of active learning-based data exploration.

Active Search. Active search [39, 59, 90] 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 [39] in the previous section. The works [59, 90] 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.

2.2.2 Active Learning for Data Exploration

The problem of dynamically seeking the next example for labeling from a large database of unlabeled tuples based on a few labeled examples is closely related to active learning. The recent results on active learning are surveyed in [78]. Below, we summarize those 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 $\mathscr{L}$ and a large pool of unlabeled data $\mathscr{U}$ available. In active learning, an example is chosen from the pool in a greedy fashion, according to a utility measure used to evaluate all instances in the pool (or, if $\mathscr{U}$ is large, a subsample thereof). In our setting of database exploration, the labeled data $\mathscr{L}$ is what the user has provided thus far. The pool $\mathscr{U}$ 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 [31, 32] 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 or Gradient Boosting models. 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 thesis we use Support Vector Machines as an example classifier.

Support Vector Machines. Support Vector Machines have been widely used for classification, regression, and other learning tasks such as outliers detection[2]. For a classification problem, a support vector machine (SVM) constructs a hyperplane in a high or infinite-dimensional space that has the largest distance to the nearest training examples of any class, intending to minimize the generalization error of the classifier.

More formally, given n pairs of training examples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ where the training points $\mathbf{x}_i \in \mathbb{R}^d$ and the corresponding labels $y_i \in \{-1, +1\}$, the hyperplane (also called *decision boundary*) determined by SVM is defined as

$$\{x : y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = 0\}, \tag{2.1}$$

where $\phi(\mathbf{x})$ maps $\mathbf{x}$ into a higher-dimensional *feature space*. Correspondingly, the decision function of SVM is

$$h(x) = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}) + b) \tag{2.2}$$

There is an essential concept in SVM called *margin*, which is defined as the perpendicular distance between the decision boundary and the closest training point M in [13] and as $2M$ in [44]. Despite the difference in the definition, the ultimate goal remains the same - optimizing the parameters $\mathbf{w}$ and b to find the optimal decision boundary that creates the biggest margin. Therefore, SVM is also known as a *maximum margin classifier*.

C-Support Vector Classification. There are many SVM formulations for solving the optimization problem. In the following, we present the most common formulation *C-Support Vector Classification* (C-SVC). C-SVC [16, 26] solves the following primal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\phi\left(\mathbf{x}_i\right) + b\right) \geq 1 - \xi_i, i = 1, \ldots, n, \\ & \xi_i \geq 0, i = 1, \ldots, n, \end{aligned} \tag{2.3}$$

[2]https://scikit-learn.org/stable/modules/svm.html#

where $C > 0$ is the regularization parameter and $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$ are the slack variables which denote the smallest nonnegative numbers satisfying $y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right) \geq 1 - \xi_i$. In particular, using slack variables allows some training examples to be misclassified and thus enables SVM to deal with, in addition to the separable case, the nonseparable case where the classes overlap. The regularization parameter C trades off between maximizing the margin (by minimizing $\|\mathbf{w}\| = \mathbf{w}^T\mathbf{w}$) and penalizing misclassification of training examples. The higher C is, the fewer misclassified training examples are allowed. More details about C can be founded in Section 5.4.2.

Due to the possibly high dimensionality of the $\mathbf{w}$ and $\phi(\mathbf{x})$, instead of working out the primal problem directly, we usually solve its Lagrangian dual problem[3]. The dual problem is

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T \mathbf{Q}\alpha - \mathbf{e}^T\alpha \\ \text{subject to} \quad & \mathbf{y}^T\alpha = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \end{aligned} \tag{2.4}$$

where $\mathbf{e}$ is the vector of all ones, $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, $\mathbf{Q}$ is an n by n positive semidefinite matrix $\mathbf{Q}_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ and $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is the kernel function. The Lagrange multipliers α_i are upper-bounded by C. By solving the dual problem (2.4), according to the primal-dual relationship, the optimal $\mathbf{w}$ can be represented as

$$\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \phi\left(\mathbf{x}_i\right) \tag{2.5}$$

with nonzero coefficients α_i only for the the training points $\mathbf{x}_i$ which satisfy the constraint $y_i \left(\mathbf{w}^T \phi\left(\mathbf{x}_i\right) + b\right) = 1 - \xi_i$. These training examples are called *support vectors*. It can be derived that the maximum margin hyperplane is completely determined by support vectors. Eventually, the decision function can be written as

$$h(x) = \operatorname{sign}(\mathbf{w}^T \phi(\mathbf{x}) + b) = \operatorname{sign}\left(\sum_{i=1}^{n} y_i \alpha_i K\left(\mathbf{x}_i, \mathbf{x}\right) + b\right). \tag{2.6}$$

[3]https://en.wikipedia.org/wiki/Duality_(optimization)

Note that the decision boundary requires merely the inner product between $\phi(\mathbf{x})$ and each support vector $\phi(\mathbf{x}_i)$. Compared to the explicit computation of the coordinates of the data in a possibly high-dimensional, implicit feature space, the *kernel trick* which computes the inner products between the images of pairs of data in the feature space is often computationally cheaper. Commonly used kernel functions include the linear kernel, polynomial kernel, and radial basis function kernel (RBF kernel). Without prior knowledge of what the user interest may be, the RBF kernel is considered more flexible than the linear or polynomial kernels, hence used in this work. The RBF kernel is defined as: $K(\mathbf{x},\mathbf{x}') = \exp(-\gamma\|\mathbf{x}-\mathbf{x}'\|^2)$. The feature space of the RBF kernel is known to be of an infinite number of dimensions. Additional information on SVM can be founded in [13, 21, 44].

Uncertainty Sampling and Version Space Search. A common form of utility measure over the unlabeled pool $\mathscr{U}$ characterizes the degree of *uncertainty* that the current model experiences for classifying each data example in $\mathscr{U}$. Under uncertainty sampling, the example that leads to the highest degree of uncertainty in classification is chosen as the example for labeling. If the classifier is SVM-based, the uncertainty can be measured by the *distance* of each example from the decision boundary of the current SVM. To improve efficiency, prior work [15, 35] further proposed to restrict uncertainty sampling to a subsample of size $l < |\mathscr{U}|$, i.e., to choose the example closest to the current decision boundary among l examples with a probabilistic guarantee that this example is within top p% closest to the decision boundary among the pool $\mathscr{U}$. For classifiers that return a probability distribution over class labels such as Gradient Boosting Regression Trees and Random Forest, the uncertainty of each unlabeled example can be measured by the *entropy* of the estimated probability distribution. Then the example that yields the largest entropy in the pool is chosen for labeling.

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. From the perspective of *version space*, the ultimate goal of active learning is to choose the next unlabeled example such that the new label provided by the user will allow the version space to be bisected (or approximately) and hence accelerate the model convergence. Uncertainty sampling is a (rough) approximation of an optimal algorithm that bisects the

version space with each selected example [87]. For the above reason, we call active learning algorithms *version space-based* because they are designed to reduce the version space.

2.2.3 Label-Noise Learning

The occurrence of label noise in the process of manual annotation is ubiquitous. In recent years, we have witnessed the growing attention to label-noise learning. In this section, we first introduce a taxonomy of label noise. Then we review some representative methods to handle noisy labels for general classification and active learning. Finally, we briefly present some novel techniques with different formats of training data.

Label noise models

There are mainly three types of label noise models [23, 37]: (i) the random classification noise (RCN) model, (ii) the class-conditional random label noise (CCN) model and (iii) the instance- and label-dependent label noise (ILN) model. In the 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). In contrast, in the CCN model, the label flip probability is independent of the example but dependent on the true label. In other words, the examples from the same class have the same label flip probability. The ILN model, in which the label flip probability is dependent on both the example and the true label, is the most general case of label noise, and it is more complex and applicable. However, compared to the RCN model and the CCN model, the ILN model has not been widely studied in the literature. In our work, we consider one general case of the ILN model for active learning-based data exploration (Section 5.1.1).

Handling label noise for classification and active learning

For classification, the survey [37] thoroughly analyzed three types of methods to deal with label noise: label noise-robust methods, label noise-tolerant methods, and label noise cleansing methods. Since our system is model and dataset agnostic - allowing to plug in any

standard classifiers as part of the active learner and supporting various user interest queries, using *data cleansing* methods is more realistic and more applicable. The **auto-cleansing** method employed to detect label noise in our work (Section 5.1.2) falls into this category. In the following, we introduce some representative data cleansing methods briefly. Entropy-based filter (EBF) [82] uses information entropy computed from the estimated probability distribution (modeled by a Bayesian classifier) to detect mislabeled examples. However, EBF needs a manually defined threshold, and information entropy is merely a heuristic confidence score, while **auto-cleansing** can automatically collect confident examples out of noisy examples with theoretical guarantees. As k nearest neighbors classifiers (kNN) are sensitive to label noise, many kNN-based data cleansing methods have emerged [94, 95]. The "kta" method compared in Section 5.6.2 also leverages the sensitivity of kNN. Recent work [66, 67] proposed *Confident Learning*, which aims to improve label quality by using state-of-the-art algorithms to characterize and identify label noise in datasets. *Confident Learning* follows the principles of pruning noisy data [22, 70, 89], estimating noise by counting with probabilistic thresholds [64], and ranking examples to collect confidence examples [49, 69]. It is also model and dataset agnostic, and it is compared to **auto-cleansing** in Section 5.6.2.

In the context of active learning, the following approaches to dealing with noisy labels have received much attention.

- **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, 46, 56, 98]. 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 [17, 100] treat relabeling as an individual process triggered at each iteration or under some conditions. Other works [46, 56] 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 advantage that removed noisy examples have no effect on modeling [38], and it's observed that retaining mislabeled examples may be more harmful than removing some correctly labeled examples [19] and in general, removing noisy examples is more effective than relabeling them and a hybrid approach which combines removal and relabeling [27, 61]. On the other hand, relabeling may be too costly since humans may make similar mistakes repeatedly [85], and there is no guarantee that the relabeled examples are noise-free. Recent work [85] 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.

Novel techniques with different formats of training data

The following two techniques, which handle noisy labels inherently, build upon different formats of training data rather than example-label pairs.

Data Programming. When ground truth labels are not available, a series of recent studies [7, 73, 91] have employed the *data programming* paradigm to build training data for developing high-quality models. Furthermore, Snorkel [72], 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 [92] proposed an approach to generate LFs using an initially labeled dataset automatically, the size of the labeled datasets needed reaches a magnitude of hundreds.

Geometric Computing Under Uncertainty. For geometric computing over uncertain data, substantial probabilistic models have been explored recently. The details of these methods and their comparisons can be found in the survey [99]. In particular, in order to build a useful and robust convex hull for a set of uncertain points, prior work [4, 83] has studied on two uncertainty models: the *unipoint model* and the *multipoint model*. Under the *unipoint model* in which each input point probabilistically exists at a fixed location, the work [83] proposed an $O(n^3)$-time algorithm to build the most likely convex hull of uncertain points in $\mathscr{R}^2$, and the work [4] can compute the probability of a query point $q \in \mathscr{R}^d$ being inside the convex hull of uncertain points in $O(n \log n)$ time for $d = 2$ and in $O(n^d)$ time for $d \geq 3$. If we treat the probability of a point's existence in the *unipoint model* as the probability of a point x being noisy $\rho(x)$ and $\rho(x)$ is known, we can leverage the probabilistic model-based convex hull to construct our proposed data space model (Section 3). However, in our problem, the probability of data points being noisy is unknown and hard to predict due to limited labeled examples.

2.3 A Polytope Model in Data Space

For interactive data exploration, active learning algorithms often require a large number of user-labeled examples to reach high accuracy, known as the *slow convergence* problem. This problem becomes more severe when the database is large and the user interest query is selective. One reason for slow convergence is the limitation of uncertainty sampling itself: it may exert a lot of effort searching in sparse, noisy, or irrelevant regions of the input space [78].

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. We observe from existing

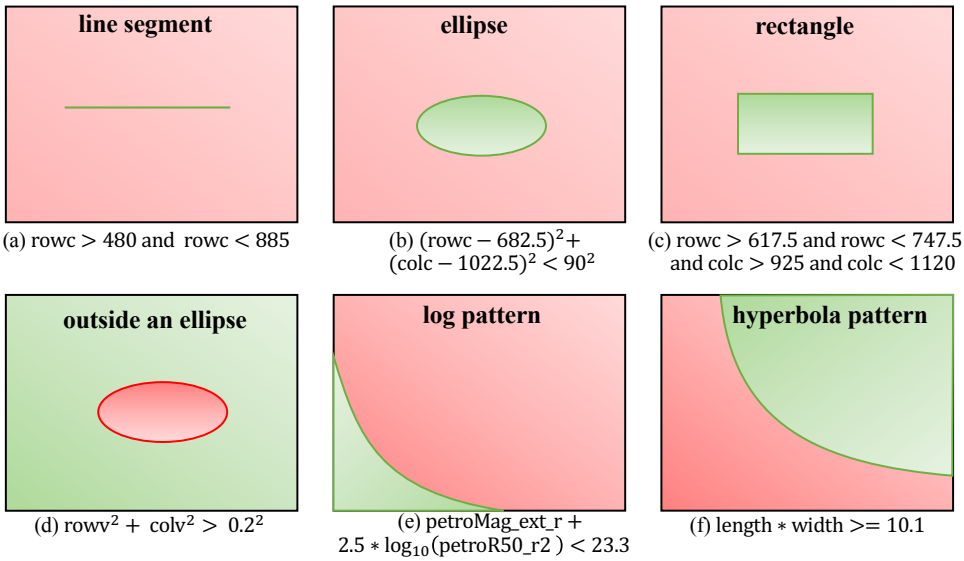


Figure 2.2: User interest regions (green) of 6 example predicates.

query traces that in some lower-dimensional subspaces, the projected user interest region or its complement is often a convex object. We call this class of queries "*convex pattern queries*", denoted as $\mathbf{Q}_c \equiv \mathbf{Q}_c^+ \cup \mathbf{Q}_c^-$, where $\mathbf{Q}_c^+$ refers to the queries whose positive regions are convex, and $\mathbf{Q}_c^-$ refers to the queries whose negative regions are convex. Figure 2.2 shows a range of queries in both classes, from real-world datasets and user interest regions.

In Chapter 3, 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. The model trained in the version space can be any traditional machine learning classifiers. As for the model trained in the data space, we call it the *data space model*. In the following, we present the data space model first proposed in [71] as a part of the TSM algorithm.

2.3.1 Three-Set Partitioning

The key idea behind the recently proposed data space model [71] 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 *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.

The *data space* model in [71] **only** supports convex queries whose user interest region is convex. When the user interest region Q is convex, any point on the line segment between two points $x_1 \in Q$ and $x_2 \in Q$ is also in Q. Under convex query and noise-free labeling assumptions, prior work [71] provided the following definitions and propositions.

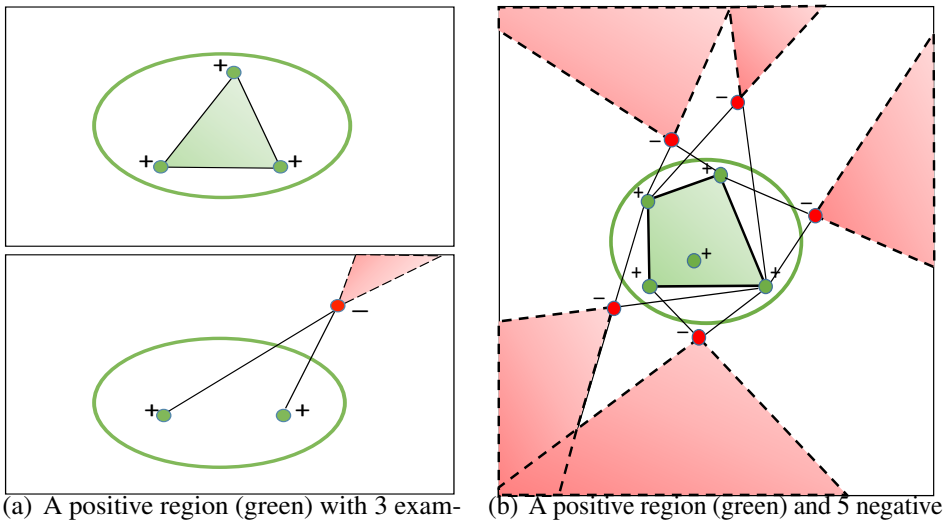
Definition 2.3.1 (Positive Region)**.** *Denote the examples that have been labeled as "positive" as $L^+ = \{e_i^+ \,|\, i = 1, \ldots, n^+\}$. The convex hull of L^+ is known to be the smallest convex set that contains L^+ [53] and is called the positive region, denoted as R^+.*

It is known that the convex hull of a finite number of points is a *convex polytope* [40]. For example, the green triangle in Figure 2.3(a) and the green tetragon in Figure 2.3(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. The following property of the positive region for convex queries can be deduced, assuming that user labels are consistent with her interest:

Proposition 2.3.1. *All points in the positive region R^+ are positive.*

Definition 2.3.2 (Negative Region)**.** *For a negative example e_i^-, we can define a corresponding negative region R_i^- such that the line segment connecting any point $x \in R_i^-$ and e_i^- does not overlap with the positive region R^+, but the ray that starts from $x \in R_i^-$ and passes through e_i^- will overlap with R^+. More formally, $R_i^- = \{x | \overline{xe_i^-} \cap R^+ = \emptyset \wedge \overrightarrow{xe_i^-} \cap R^+ \neq \emptyset\}$. Given n^- negative examples, the negative region R^- is the union of the negative region for each negative example, i.e., $R^- = \cup_{i=1}^{n^-} R_i^-$.*

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 2.3(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



(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.

Figure 2.3: Positive and negative regions in the polytope model.

Figure 2.3(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. The negative region is proved to have the following property:

Proposition 2.3.2. *All points in the negative region R^- are negative.*

Definition 2.3.3 (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 x based on the following decision function, which takes values in $\{-1, 0, 1\}$ corresponding to R^-, R^u, and R^+ defined above:

$$F_{\mathscr{D}}(x) = 1 \cdot \mathbb{1}(x \in R^+) - 1 \cdot \mathbb{1}(x \in R^-). \tag{2.7}$$

2.3.2 Lower Bound of F-score

As stated before, our accuracy measure is F-score. Formally, F-score is evaluated on a *test set* $D_{test} = \{(x_i, y_i)\}$, where x_i denotes an example and y_i denotes its label according to the

classification model. Then F-score is defined as:

$$\text{F-score} = 2 \cdot \frac{precision \cdot recall}{precision + recall},$$

where *precision* is the fraction of points returned from D_{test} by the model that are positive, and *recall* is the fraction of positive points in D_{test} that are returned by the model.

However, capturing F-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. The data space model can not only be considered as a classification model, but also enable a provable bound on the classification accuracy without knowing the ground truth. Prior work [71] proposed *Three-Set Metric* based on the data space model $F_{\mathscr{D}}$ to bound the F-score with limited labeled data and proved this metric is a lower bound of F-score.

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 2.3.4 (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* $\frac{|D^+|}{|D^+|+|D^u|}$.

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.

Concerning the lower bounds of the model accuracy, the following formal results are presented in [71].

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

Theorem 2.3.1. *The Three-Set Metric evaluated on* D_{eval} *is a lower bound of the F-score if* **DSM** *is evaluated on* D_{test}.

The above lower bound 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, prior work [71] employed 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., $p = |D^+|/|D_{eval}|$ and $q = |D^-|/|D_{eval}|$. 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 $X_n = \frac{\hat{p}}{1-\hat{q}}$. The 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 us find the lower bound of n.

Theorem 2.3.2. *$sup_\varepsilon \| Pr(\sqrt{n}|X_n - b| < \varepsilon) - \left(2\Phi(\frac{\varepsilon(1-q)}{\sqrt{p(1-p-q)}}) - 1 \right) \| = O(1/\sqrt{n})$ 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 $n \geq \left(\Phi^{-1}\left(\frac{\lambda+1}{2}\right) \right)^2 / (4\delta^2)$.

Chapter 3

Dual-Space Model

Although recent work [71] has proposed an algorithm TSM, based on the data space model, to deal with the slow convergence problem, TSM is built upon strong assumptions and has several limitations. TSM assumes convex query patterns formed by merely numerical attributes and no noisy labels provided by the user. In practice, the query patterns that TSM can handle are very limited. In addition, TSM, despite its effectiveness in low dimensions, has little or no improvement on both accuracy and convergence in high dimensions. This chapter focuses on making full use of the data space model to ameliorate the data exploration problem in low dimensions, generalizing the data space model to categorical variables, and leaving other issues to later chapters. More specifically, by leveraging the subspatial convex property, we propose the cornerstone of our work, Dual-Space Model (**DSM**), which is composed of a data space model and a traditional classification model. In the context of active learning-based data exploration, we use both models in **DSM** to jointly offer the prediction for unlabeled examples and sample acquisition functionalities. **DSM** can reduce the number of labeled examples required to reach high accuracy, thereby expediting the convergence. We also provide an extension of **DSM** to optimize the exploration for categorical attributes and a set of techniques to improve time efficiency for large-scale data exploration.

In this chapter, we introduce the **DSM**-based algorithm for data exploration in Section 3.1 and extend it for categorical attributes in Section 3.2. Then, we propose some optimization

methods to boost time efficiency in Section 3.3. In addition, we clarify the differences between our work and the TSM algorithm in prior work [71] in Section 3.4.

3.1 Dual-Space Model and Algorithm

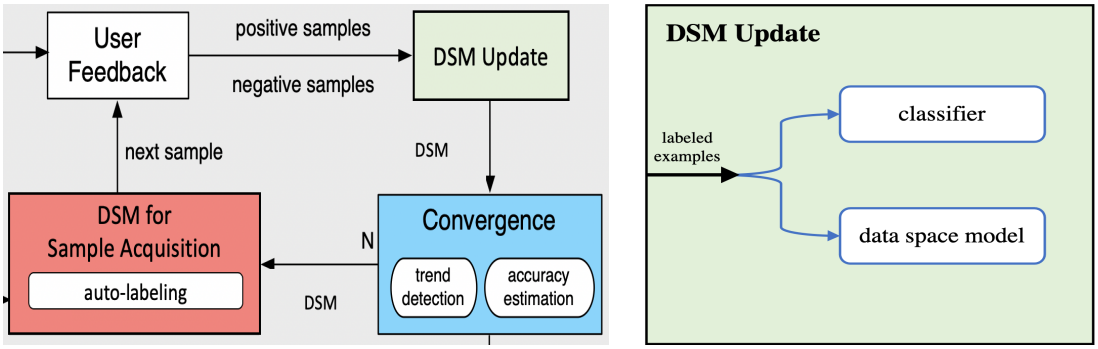
We now propose a new algorithm for interactive data exploration by exploiting models from two spaces, including our data space model $F_{\mathscr{D}}$ with the Three-Set Metric, and a classification model $F_{\mathscr{V}}$ with uncertainty sampling derived from the version space.

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 $F_{\mathscr{D}}$. 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 uncertain region R^u, then the classification model $F_{\mathscr{V}}$ is used to predict the example to be positive or negative.

At each iteration, **DSM** is updated for better prediction performance, which corresponds to "**DSM** Update" in Figure 3.1(a). More specifically, as shown in Figure 3.1(b), we build the classification model $F_{\mathscr{V}}$ and the data space model $F_{\mathscr{D}}$ simultaneously for the update of **DSM**.

(2) *Sampling*: In the active learning framework, the model is also used to guide the choice of the next example for labeling such that the new label can lead to significant improvement of the model accuracy. As discussed in Section 2.2.2, active learning uses an uncertainty sampling method, $\mathscr{S}_{\mathscr{V}}$, for a given classification model to choose the next example. However, directly application of uncertainty sampling to our dual-space model raises a problem: the example chosen by $\mathscr{S}_{\mathscr{V}}$ may fall in the known positive or negative region of our data space model $F_{\mathscr{D}}$, hence wasting computing resources on such examples. In our work, we propose an uncertainty sampling method that is restricted to the uncertain region of our data space model, denoted as $\mathscr{S}_{\mathscr{D}}$. However, if we sample only from the uncertain region of our data space model, we may not get a representative sample to train the classifier. Therefore, we use



(a) Iterative learning and exploration (b) **DSM** update

Figure 3.1: **DSM** for data exploration

a sampling ratio, γ, to alternate between the sampling methods, $\mathscr{S}_{\mathscr{D}}$ and $\mathscr{S}_{\mathscr{Y}}$. For instance, when $\gamma = 1/3$, in each iteration we use $\mathscr{S}_{\mathscr{D}}$ with probability 1/3 and use $\mathscr{S}_{\mathscr{Y}}$ otherwise.

The sampling process is performed in the step "**DSM** for Sample Acquisition" of Figure 3.1(a). If the example chosen by $\mathscr{S}_{\mathscr{Y}}$ falls in the known positive or negative region of the data space model $F_{\mathscr{D}}$, we label it by $F_{\mathscr{D}}$ to save user labeling effort. We call this **DSM**'s *auto-labeling*.

Now we present the full algorithm for interactive data exploration, as shown in Algorithm 1. The input is the database D, a positive example x^+, a negative example x^-, a user-defined accuracy threshold λ, and the sampling ratio γ. First, we extract an evaluation dataset D_{eval} from the database D (line 1). For now, let us assume D_{eval} to be D. Then we initialize data structures, setting the uncertain partition of the evaluation dataset to be D_{eval}. The algorithm next goes through iterative exploration.

Lines 8-14 update our **DSM** model. The data space model, R^+ and R^-, is updated with the newly labeled example(s) by the user (lines 8-9). This step incrementally updates our convex polytope for R^+ and the union of convex polytopes for R^- based on computational geometry [8]. Afterwards, as shown in Algorithm 2, the corresponding partitions of D_{eval} are incrementally updated (line 10). In this step, some examples are removed from the uncertain 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

Algorithm 1: Dual-Space Algorithm for Convex Queries

Input: database D, a positive example x^+, a negative example x^-, accuracy threshold λ, sampling ratio γ

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

Algorithm 2: `threeSets` Incrementally update partitions (Algorithm 3 in [71])

Input: data space model (R^+, R^-), three-set partitions (D^+, D^-, D^u)

1: **for** $x \in D^u$ **do**
2: **if** $x \in R^+$ **then**
3: $D^+ \leftarrow D^+ \cup \{x\}$, $D^u \leftarrow D^u \setminus \{x\}$
4: **else if** $x \in R^-$ **then**
5: $D^- \leftarrow D^- \cup \{x\}$, $D^u \leftarrow D^u \setminus \{x\}$
6: **return** (D^+, D^-, D^u)

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 15-28 implement uncertainty sampling using **DSM**. With probability γ, we perform uncertainty sampling from a pool that is restricted to the uncertain partition of the evaluation set, D^u. Then the example chosen by uncertainty sampling 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. Otherwise, it will be labeled by the user.

Then the algorithm proceeds to the next iteration. It 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 29). Finally, we run the **DSM** model over the database to retrieve all tuples predicated to be positive. For the `subsample` procedures used in the algorithm, we defer their details to Section 3.3.

3.2 Extension for Categorical Attributes

Our categorical **DSM** extension, inspired by the partitioning function for numerical attributes, which splits the data space into positive, negative, and uncertain regions, partitions the domain (all possible values) of a categorical attribute into the positive set (the values seen in positive examples), the negative set (the values seen in negative examples), and the uncertain

set. More specifically, we preprocess categorical attributes using one-hot encoding and then keep track of the appearance frequency of each encoded categorical attribute in the positive set and the negative set, respectively. For prediction over an unlabeled example using the categorical **DSM**, we compare the example's attribute value frequency in the positive set f_p to that of the negative set f_n . If $f_p > f_n$ ($f_p < f_n$), the example is labeled as positive (negative) in the space spanned by the encoded categorical attributes; otherwise, we use the classifier as a fallback to handle this categorical attribute.

3.3 Optimizations

Since our system is designed for interactive data exploration, the per-iteration time should be limited to within a few seconds. However, when the data exploration is conducted on a large-scale dataset, it becomes very time-consuming. To reduce the per-iteration time of our algorithm, on the one hand, we employ several `subsample` procedures in Algorithm 1 and present their implementation and optimization in Section 3.3.1 and Section 3.3.3. On the other hand, we leverage distributed computing to cut down the per-iteration time in Section 3.3.2.

3.3.1 Sampling for Creating the Evaluation Set (OPT1)

In line 1 of Algorithm 1, we construct an evaluation set D_{eval} to develop a lower bound of the **DSM** model. Ideally, we want D_{eval} to be as large as the entire database D. For efficiency, we can only afford to have a memory-resident sample. The analysis in Theorem 2.3.2 indicates that we can choose a sample of size n from the database, and achieve an approximate lower bound of the F-score of our **DSM** algorithm when evaluated on the database. The sample, denoted as $\tilde{D}_{eval}$, will expedite Algorithm 2. 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 $\varepsilon \leq .005$ with probability 0.975 or higher. When $n = 1.9$ million, $\hat{b}$ approximates b by $\varepsilon \leq .001$ with probability 0.994 or higher.

3.3.2 Distributed Computing (OPT2)

Another way to reduce the time cost of Algorithm 2 is to use distributed computing. We take full advantage of an open-source system Ray[1] to parallelize our code and eventually manage to reduce the time per iteration to within a few seconds even when the size of $\tilde{D}_{eval}$ is 1.9 million. To be more specific, we split the uncertain data set into several subsets and rebuild the partitions within each subset. In this case, the partitioning task for the uncertain data set is divided into multiple sub-tasks. Ray can allocate a CPU core for each sub-task and run the sub-tasks independently and simultaneously on all available CPU cores, and finally join the results in order.

3.3.3 Sampling for Pool-based Active Learning (OPT3)

Algorithm 1 contains two `subsample` procedures for pool-based uncertainty sampling, that is, choosing the most uncertain example from the pool for labeling next. The `subsample` routine in line 17 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 21, however, creates a pool, from the unlabeled portion of the entire database, which is large and not materialized.

Since sampling from the database at each iteration is costly, our work offers a sampling-based optimization by sampling the database once before interactive exploration starts to create a set called DBmirror, and then using the unlabeled instances in DBmirror, denoted as $\mathscr{U}$, as our pool in line 21. If we use SVM uncertainty sampling in line 22 with the random top l method [15, 35] (described in Section 2.2.2), we prove in Proposition 3.3.1 that we can use $\mathscr{U}$ to replace the pool and avoid subsampling the entire database at each iteration in line 21. Specifically, it guarantees that the probability of finding at least one user-to-label instance that is among the top p% of closest instances to the current boundary in the database, converge to $1-\eta$ as $m = |\text{DBmirror}|$ goes to infinity. For example, with $l = 5000, m = 1.9e+5$, the

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

probability is within $1e$-11 of $1-\eta$ when $p\% = .5\%$, and it is within 0.0023 of $1-\eta$ when $p\% = .1\%$.

Proposition 3.3.1 (Pool for random top sampling)**.** *Let $\mathscr{U}$ be the current unlabeled instances in DBmirror, and B be the current decision boundary. A random sample of size l is drawn from $\mathscr{U}$ and scanned to find the instance closest to B for the user to label, where l is chosen so that*

$$\Pr(\textit{at least one in the } l \textit{ draws is among the top } p\% \textit{ closest to } B \textit{ in } \mathscr{U}) = 1-(1-p)^l = 1-\eta.$$

Let A be the event that at least one in the l draws is among top $p\%$ closest to B in the unlabeled database. Since the probability of A depends on specific instances in $\mathscr{U}$, let $\Pr(A|\mathscr{U})$ denote the probability of A as a function of the random instances in $\mathscr{U}$. Then l chosen this way guarantees that

$$\Pr(A|\mathscr{U}) \xrightarrow{p} 1-\eta, \textit{as } m = |\textit{DBmirror}| \to \infty, \textit{further}$$

$$\sqrt{m}(\Pr(A|\mathscr{U}) - (1-\eta)) \xrightarrow{d} N(0, l^2 p(1-p)^{2l-1}).$$

Proof. Let X be the distance to B_k of a randomly chosen point from DB_k and x_p be the distance such that $p = P(X > x_p)$. Let $X_1, \cdots, X_{M-k}$ be the distances to B_k of the instances in DBmirror_k. X_i's are a random sample of X, that is, a point in DBmirror_k is equally likely to be any one of the instances in DB_k.

Let Y_i be the distances to B_k of the ith point in the sample of size L drawn from DBmirror_k, $i = 1, \cdots, L$. Then

$$P(A|\textit{DBmirror}_k) = P(A|X_1, \cdots, X_{M-k}) = 1-(1-\hat{p})^L$$

where $\hat{p} = P(Y_i > x_p|X_1, \cdots, X_{M-k}) = \frac{1}{M-k}\sum_{i=1}^{M-k} I\{X_i > x_p\}$. Note $E\hat{p} = EI\{X_i > x_p\} = P(X_i > x_p) = p$. Assume N is large and ignore the correlation among the X_i's due to sampling without replacement. According to the Law of Large Numbers, $\hat{p}$ converges in probability to

p and therefore

$$P(A|X_1 \cdots, X_{M-k}) = 1 - (1-\hat{p})^L \xrightarrow{p} 1 - (1-p)^L = 1 - \eta.$$

Further, According to the Central Limit Theorem, $\sqrt{M-k}(\hat{p}-p)$ converges in distribution to $N(0, p(1-p))$. In addition, with the Delta method,

$$\sqrt{M-k}(P(A|X_1 \cdots, X_{M-k}) - (1-\eta)) \xrightarrow{d} N(0, L^2 p(1-p)^{2L-1}).$$

□

3.4 Differences from Prior Work

Our proposed **DSM** offers distinct advantages over TSM in prior work [71] mainly from two perspectives:

- **The assumptions on user query patterns**
  As stated earlier, TSM has many assumptions and limitations: (1) it assumes convex query patterns; (2) it assumes that the training data contains no mislabeled examples; (3) it does not handle categorical attributes; (4) it fails to achieve desired performance with high dimensionality. In contrast, our **DSM** overcomes these limitations by offering a number of extensions: First, beyond convex queries, our **DSM** also supports the case that the negative region of the query is convex. We can simply switch the previous 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 $Q \in \mathbf{Q}_c^-$. Second, we further relax the convexity assumption for high-dimensional data exploration by factorizing the high-dimensional exploration problem into a set of low-dimensional exploration problems. The details are deferred to Section 4.1 where we offer a test procedure for all the assumptions made in our work. Third, TSM does not consider categorical attributes, while our **DSM** extension

for categorical attributes in Section 3.2 is quite effective in practice. Finally, to address the label noise problem, we propose a robust **DSM** in Chapter 5.

- **The usage of the data space model for data exploration**
  TSM is used merely to filter examples whose class labels are already known by the data space model during the user labeling process and to compute the lower bound of F-score to detect convergence. In contrast, our **DSM** jointly offers the prediction and sample acquisition functionalities and thereby expedites model convergence to a greater extent than TSM.

3.5 Summary

We presented the Dual-Space Model (**DSM**) composed of a data space model and a traditional classification model as the cornerstone of this thesis. We brought the subspatial convexity property of database queries to bear on the design of the data space model. For the active learning-based data exploration, the data space model serves as an active learner playing an important role in both prediction and sample acquisition. We also extended **DSM** for categorical attributes and proposed some optimization strategies to reduce the per-iteration time to within 1-2 seconds. Since our final algorithm for the interactive data exploration system consists of not only **DSM** but also some optimization methods for high-dimensional user interest patterns, we defer the experimental evaluation of our proposed techniques to Section 4.3.

Chapter 4

High-dimensional Exploration

When the user interest contains a large number of attributes, the performance of both traditional active learning and **DSM** may suffer from the "curse of dimensionality". The low selectivity of user interest queries makes it even harder to conduct the high-dimensional exploration fast and accurately. Therefore, we propose two approaches, *factorization* and *online feature selection*, to reducing dimensionality in data exploration. First, by leveraging the property of conjunctivity, we factorize the high-dimensional space into a set of low-dimensional spaces, build a **DSM** on each subspace and combine them to be a factorized **DSM** (**DSM**$_F$). 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 F-score than **DSM** without factorization. Due to factorization, **DSM**$_F$ is capable of handling more complex queries and improves performance in both accuracy and convergence speed over **DSM** without factorization. Second, in case that the user may start exploration with more attributes than those needed in the final model, to remove irrelevant attributes, we develop an online feature selection method that adaptively selects the top-k relevant attributes.

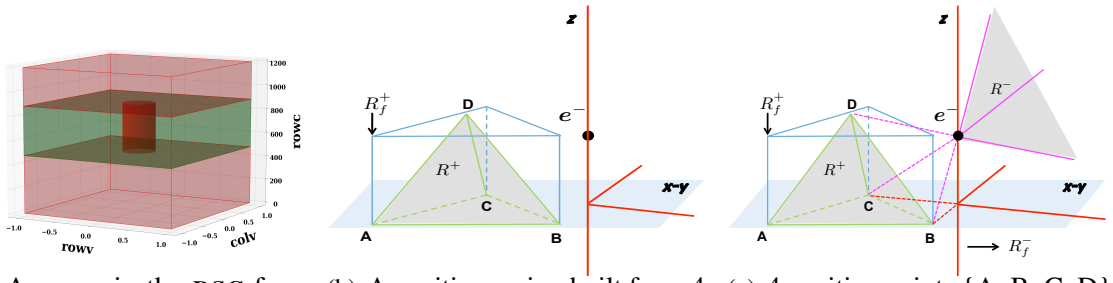
In this chapter, we introduce factorization to improve the performance of **DSM** built with high dimensionality in Section 4.1 and present online feature selection methods to remove irrelevant attributes in Section 4.2. The effectiveness of our proposed techniques are evaluated in Section 4.3.

4.1 Factorization

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 $\textbf{DSM}_F$, may require user labels of examples in some subspaces: If the user label is positive for an example, the label in each subspace is inferred to be positive. However, if the user label is negative for an example, she will be asked to specify the subset of attributes (and the subspaces thereof) that lead to the negative label. Since the user has gone through thinking when deciding the label, specifying a subset of attributes that lead to the negative label can be facilitated via a graphical interface such as in [30].

Factorization for DSC queries. Formally, factorization concerns a user interest query Q defined on an attribute set 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 $A_i = \{A_{i1}, \cdots, A_{id_i}\}$. The family of attribute sets, $A = (A_1, \cdots, A_m)$, is pairwise disjoint with $d = \sum_{i=1}^{m} d_i$, and is called the *factorization structure*.

In practice, we can derive the factorization structure from available data in a database. It is a partitioning of the database attributes with two properties: 1) Each attribute can appear in only one partition. 2) If several attributes may be used in the same predicate (e.g., the positional attributes *rowc-colc*, or the velocity attributes *rowv-colv* from SDSS), they must be assigned to the same partition. 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 A. Even if a query trace is not available, it is not unreasonable to assume that by working with domain experts, it is possible



(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^-), where e^- is negative in the (x-y) subspace and positive in the (z) subspace

Figure 4.1: Illustration of factorization when 3D space (x-y-z) is factorized into two subspaces (x-y) and (z).

to extract a reasonable factorization structure that reflects how attributes are used based on their semantics, e.g., the velocity attributes *rowv-colv* are often used in the same predicate.

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 $\{x_i \in \mathbb{R}^{|A_i|} | Q_i(x_i) > 0\}$, is convex (in $\mathbf{Q}_c^+$), or the negative region, $\{x_i \in \mathbb{R}^{|A_i|} | Q_i(x_i) < 0\}$ 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 4.1(a) shows an example query, $Q = x^2 + y^2 > 0.2^2 \wedge 480 < z < 885$, which combines the ellipse pattern in Figure 2.2(d), whose negative region is convex, with the line pattern in Figure 2.2(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 (marked 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 generalization *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**.

4.1.1 Factorized Dual-Space Model

Given the factorization structure, the polytope model runs in each subspace where the subspatial convexity is deemed true. First, in the i^{th} subspace defined by A_i, Q_i's positive and negative regions, denoted as R_i^+ and R_i^-, are built according to Definition 2.3.1 and 2.3.2, but based on the "positive" and "negative" labels of projected examples for Q_i, as described above. Then we build the positive and negative regions of Q from the positive and negative regions in the subspaces via the *conjunctive property*:

$$R_f^+ = \times_{i=1}^m R_i^+,\ R_f^- = \left(\times_{i=1}^m (R_i^-)^c\right)^c \tag{4.1}$$

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 A_i, let $F_{A_i} : \mathbb{R}^{|A_i|} \to \{-1, 0, 1\}$ be the decision function that divides the subspace into three disjoint regions corresponding to the negative, uncertain, and positive regions, respectively. As in (Eq. 2.7),

$$F_{A_i}(x) = 1 \cdot \mathbb{1}(x \in R_i^+) - 1 \cdot \mathbb{1}(x \in R_i^-). \tag{4.2}$$

For p-**DSC** queries, if the subspatial convexity is deemed not true in a subspace, the value of its decision function is a constant zero.

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

$$F_{\mathscr{D}_f}(x) = \min_{i=1}^m F_{A_i}(x_i) \tag{4.3}$$

where $x = (x_1, \ldots, x_d)$, and x_i denotes the projection of x on the i^{th} subspace defined by A_i.

Finally, consider the dual-space model. Let $\textbf{DSM}_F$ be **DSM** in Chapter 3 with the polytope data space model $F_{\mathscr{D}}$ replaced by the polytope model with factorization $F_{\mathscr{D}_f}$. Then the dual-space decision function of $\textbf{DSM}_F$, $H : \mathbb{R}^d \to \{-1, 1\}$, is:

$$H(x) = \begin{cases} F_{\mathscr{D}_f}(x), & F_{\mathscr{D}_f}(x) \neq 0 \\ F_{\mathscr{Y}}(x), & \text{otherwise} \end{cases} \tag{4.4}$$

where $F_{\mathscr{Y}}$ is the classification model. H is our final prediction model returned to approximate the user interest query. For 0-**DSC** queries, $\textbf{DSM}_F$ simply runs traditional active learning.

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 uncertain region and offering better accuracy. Figure 4.1(b) shows four positive points A, B, C, D when the 3D space (x-y-z) is factorized into two subspaces (x-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-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. 4.1, we obtain a prism marked by the blue lines, which is much bigger than R^+.

Figure 4.1(c) shows a negative point e^- and the negative region constructed 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. 4.1, the new negative region R_f^- extends the convex cone R_{xy}^- by all possible values of z, resulting in a geometric shape enclosed by the three solid red lines in the figure. Again, the new R_f^- is much larger than R^-.

4.1.2 Factorized Classifier and Sample Acquisition Strategy

The $\textbf{DSM}_F$ in the previous section only applies factorization to the data space model to improve performance, while the classifier is always trained directly on the original high-

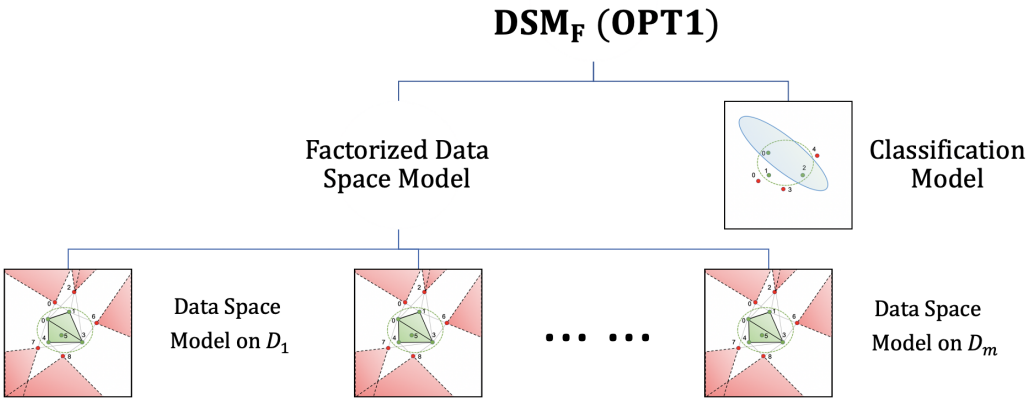


Figure 4.2: **DSM**$_F$ (OPT1): only factorize the data space model

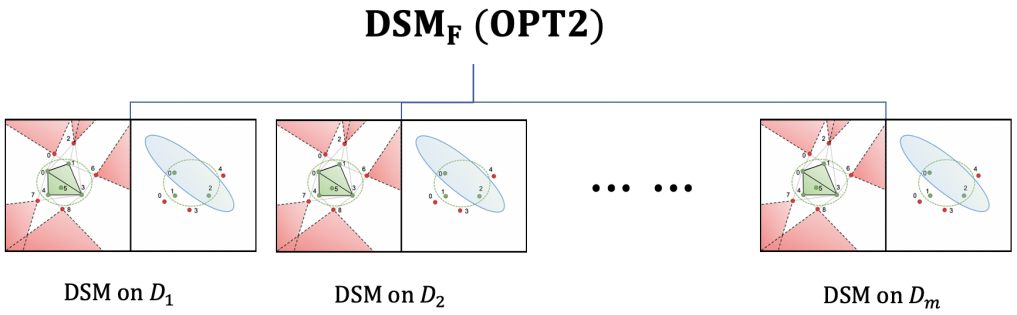


Figure 4.3: **DSM**$_F$ (OPT2): factorize both the data space model and the classifier

dimensional data space, as shown in Figure 4.2. However, general classifiers are also seriously affected by increased dimensionality. For example, if the user interest region is a right circular cylinder in a three-dimensional data space, it is challenging for a classifier to learn the exact boundary of this cylinder. Nevertheless, if we project this cylinder properly onto subspaces such that the corresponding projections are a filled circle in two-dimensional data space and an interval in one-dimensional data space, learning the projected regions on subspaces becomes much easier. Furthermore, by combining the partial classification models from subspaces according to some factorization rules, we can depict the boundary of the cylinder much faster and more accurately. We thus factorize the data space model and the classifier in our new version of **DSM**$_F$. Figure 4.3 shows the framework of our new **DSM**$_F$ in which we train a complete **DSM** in each subspace.

Regarding the factorization for the classifier, we train a classifier in each subspace $h_i : \mathbb{R}^{d_i} \to \{-1,1\}, i = 1,\cdots,m$ and combine them to be a classifier $F_{\mathscr{Y}_f} : \mathbb{R}^d \to \{-1,1\}$ over the entire data space, according to the factorization rules as follows:

$$F_{\mathscr{Y}_f} = \text{factorization\_rules}(h_1(x_1),\cdots,h_m(x_m))$$

For example, if the user interest query satisfies the conjunctive property defined in 4.1, we have

$$F_{\mathscr{Y}_f} = min(h_1(x_1),\cdots,h_m(x_m))$$

We call $F_{\mathscr{Y}_f}$ as a factorized classifier. Correspondingly, we propose a factorized sample acquisition method that selects the example closest to the decision boundary of the factorized classifier. The sample acquisition criterion can be formally defined as:

$$\text{next\_sample} = \arg\min_x \Big(\sum_{i=1}^{m} |d(h_i, x_i)| \Big) \tag{4.5}$$

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 distance metric (Manhattan Distance metric) to combine the distance information from all subspaces is that for the commonly used L_k norm, $L_k(x,y) = \sum_{i=1}^{d} \left(|x^i - y^i|^k\right)^{1/k}$, $x, y \in \mathbb{R}^d, k \in \mathscr{Z}$, as demonstrated in [5], lower values of k is preferable in high dimensional space. In other words, L_1 is the most preferable for high-dimensional applications. In addition, we observe from empirical results that when the sample acquisition methods are performed with high dimensionality, the examples selected by the criterion 4.5 are usually closer to the true decision boundary, compared to the traditional uncertainty sampling.

Eventually, the dual-space decision function of the new $\textbf{DSM}_F$, $H : \mathbb{R}^d \to \{-1,1\}$, can be formalized as:

$$H(x) = \begin{cases} F_{\mathscr{D}_f}(x), & F_{\mathscr{D}_f}(x) \neq 0 \\ F_{\mathscr{Y}_f}(x), & \text{otherwise} \end{cases} \tag{4.6}$$

4.1.3 Lower Bound of F-score with Factorization

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

Proposition 4.1.1. *All points in the positive region R_f^+ are positive and $R^+ \subseteq R_f^+$ at each iteration of the data exploration process.*

Proposition 4.1.2. *All points in the negative region R_f^- are negative and $R^- \subseteq R_f^-$ at each iteration of the data exploration process.*

Proof. If $R_f^+ \neq \emptyset$, for each point $x = (x_1, \cdots, x_m) \in R_f^+$ where x_I denotes the values of x on attributes A_I, we have $x_I \in R_I^+, \forall I \in \mathbb{Z} \cap [1,m]$. According to proposition 3.1, all points in R_I^+ are positive for Q_I. Therefore, x is positive for $Q = Q_1 \wedge \cdots \wedge Q_m$. We prove that all points in the positive region R_f^+ are positive.

If $R_f^- \neq \emptyset$ and given a point $x = (x_1, \cdots, x_m) \in R_f^-$, then there exists some subspace spanned by attributes A_I such that $x_I \in R_I^-$. Since all points in R_I^- are negative for Q_I based on proposition 3.2, and the target query Q is in a conjunctive form, x is negative for Q. Therefore, we conclude that all points in the negative region R_f^- are negative.

To prove $R^+ \subseteq R_f^+$ and $R^- \subseteq R_f^-$, we consider the following three cases:

1. If $Q \in \mathbf{Q}_c^+$, then Q's positive region's projection onto each subspace is convex and we can build a convex polytope for the projection of the positive region in each subspace. We see that the Cartesian product of convex sets is convex, so R_f^+ is convex by definition. At each iteration, given a finite set of positive points $\{x\}$, R^+ and R_f^+ are constructed on $\{x\}$. Since R^+ as a convex polytope is the smallest convex set that contains $\{x\}$, we conclude $R^+ \subseteq R_f^+$.

   To prove $R^- \subseteq R_f^-$, it is sufficient to prove that every arbitrary convex cone that R^- is built upon, is contained by R_f^-. Let e^- be the apex of a convex cone $R_{e^-}^-$ in Definition 3.2. Since e^- is negative for Q, there exists I such that its factorization vector e_I^- is negative for Q_I. Let $\pi_I(R_{e^-}^-)$ be the projection of the cone $R_{e^-}^-$ onto the subspace spanned by A_I. Then for a point $x \in R_{e^-}^-$, $x_I \in \pi_I(R_{e^-}^-)$.

Now define the convex cone construction in the subspace spanned by A_I with e_I^- as apex as

$$R^-_{e_I^-} = \{x_I \in \mathbb{R}^{d_I} | \overline{x_I e_I^-} \cap R_I^+ = \emptyset \wedge \overrightarrow{x_I e_I^-} \cap R_I^+ \neq \emptyset\}.$$

We will now show that $\pi_I(R^-_{e^-}) \subseteq R^-_{e_I^-}$. We have $\pi_I(R^-_{e^-}) = \{x_I \in \mathbb{R}^{d_I} | \overline{x_I e_I^-} \cap \pi_I(R^+) = \emptyset \wedge \overrightarrow{x_I e_I^-} \cap \pi_I(R^+) \neq \emptyset\}$. Since $x_I \in \pi_I(R^-_{e^-})$ and $\pi_I(R^-_{e^-})$ is a negative convex cone, $\overline{x_I e_I^-} \cap R_I^+ = \emptyset$ is true. On the other hand, since $\pi_I(R^+) \subseteq R_I^+$, $\overrightarrow{x_I e_I^-} \cap \pi_I(R^+) \neq \emptyset$ implies $\overrightarrow{x_I e_I^-} \cap R_I^+ \neq \emptyset$. This shows $\pi_I(R^-_{e^-}) \subseteq R^-_{e_I^-}$.

Therefore $x_I \in \pi_I(R^-_{e^-}) \subseteq R^-_{e_I^-} \subseteq R_I^-$, and $x \in R_f^-$. Finally, $R^-_{e^-} \subseteq R_f^-$ for any e^-, and hence $R^- \subseteq R_f^-$.

2. If $Q \in \mathbf{Q}_c^-$, the propositions can be proved by constructing **DSM** and **DSM**$_F$ in a reverse way 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.

3. If $Q \notin \mathbf{Q}_c$, then without factorization, **DSM** doesn't work and has $R^+ = \emptyset$, $R^- = \emptyset$. We therefore conclude $R^+ \subseteq R_f^+$ and $R^- \subseteq R_f^-$. In particular, for **DSC** queries, R_f^+ and R_f^- are nonempty regions. For p-**DSC** queries, although **DSM** can be built in each subspace, due to the conjunctive form of Q, $R_f^+ = \emptyset$ always holds true, but $R_f^- \neq \emptyset$ as long as there exists at least one subspace where $R_I^- \neq \emptyset$.

□

Proposition 4.1.3. **DSM**$_F$ *improves the Three-Set Metric, the lower bound of the model accuracy in F-score, over* **DSM**.

Proof. Three-set metric is defined to be $\frac{|D^+|}{|D^+|+|D^u|} = \frac{|D^+|}{|D|-|D^-|}$. According to Proposition 4.1.1 and Proposition 4.1.2, both positive region and negative regions are enlarged by the factorized **DSM**, which speeds up the convergence of three-set metric. □

Proposition 4.1.1 and Proposition 4.1.2 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 F-score built from (R_f^+, R_f^-) is higher than that from (R^+, R^-) based on Def. 2.3.4.

4.1.4 Testing Assumptions

Factorization replies on two assumptions, which our system will test when a user is performing data exploration online. The first assumption is that we have a correct factorization structure, $A = (A_1, \cdots, A_m)$, for a database derived from its query trace or schema. The second assumption is that user interests take the form a conjunctive query (CQ).

With the extension to handle categorical attributes in Section 3.2, the class of user interest queries is an *extended class of conjunctive queries*, in the form of $P_1 \wedge P_2 \wedge \ldots$, such that either P_i is an individual predicate, or P_i is defined on a categorical attribute A and can take the disjunctive form, $(A = 1) \vee (A = 2) \vee ...$

Test Procedure. When either the factorization structure or the CQ assumption is wrong, we start to 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. Based on this, our system uses the following procedure to test online both assumptions behind $\mathbf{DSM}_F$. 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. The test procedure also allows $\mathbf{DSM}_F$ to handle minor noisy user labeling by adjusting the threshold T, the number of conflicting examples that can be tolerated by each polytope model.

4.2 Online Feature Selection

The user exploration task may start with more attributes than those included in the final learned model (user interest query). The attributes that are not included in the final model

are noise in data exploration and cause slow convergence of the model. To remove such noisy attributes, we employ both offline dimensionality reduction on the database and online feature selection techniques.

We examined dimension reduction methods, including PCA for offline compression, and Random Forests (RF) and Gradient Boosting Regression Trees (GBRT) for online feature selection. These techniques are provided by the scikit-learn library[1].

Principled Component Analysis (PCA) defines a set of orthogonal directions that capture the maximum variance of a dataset, with an implicit hope that the variance along a small number of principal components provides a reasonable characterization of the entire dataset [80]. In our work, PCA is used as a query-agnostic dimensionality reduction technique. That is, we use it to compress a database table $D(A_1,\ldots,A_d)$ into a new table $D'(B_1,\ldots,B_k)$ with fewer columns, $k < d$. Each database object has two presentations using $(A_1,\ldots,A_d)$ and $(B_1,\ldots,B_k)$, respectively. In each iteration of data exploration, we display a new sample to the user using the $D(A_1,\ldots,A_d)$ representation, but train an SVM with all existing labeled samples on $D'(B_1,\ldots,B_k)$.

Random Forests (RF) are an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the mean prediction of the individual trees.

Gradient Boosting Regression Trees (GBRT) are a gradient boosting machine with decision trees as base-learners. The basic idea is to construct a series of decision trees in a forward stagewise manner, where each new decision tree is constructed to be maximally correlated with the negative gradient of the loss function – the error of the whole ensemble learned so far. The additive model of GBRT is a linear combination of the base learners.

We found that GBRT works best for reducing dimensions. Therefore, we outline how GBRT is used for online feature (attribute) selection in our work. In each iteration of data exploration, we feed all labeled examples to the GBRT learner, and ask the learner to return the top-k features that are deemed most important in learning. Then we build the classifier using these features and select the next example for labeling. We repeat the two steps in each

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

iteration until the choice of top-k features stabilizes. Then we build the full $\textbf{DSM}_F$ model until it converges. However, we observe two limitations of this approach:

Unbalanced training data. GBRT is very sensitive to unbalanced classes, that is, when the training data is dominated by the negative examples. This is common in data exploration because the true user interest is often a highly selective query. We draw upon two insights in this work to mitigate the imbalance problem. First, when applicable, our $\textbf{DSM}_F$ algorithm already maintains a list of positive examples from the evaluation set and we can add them to make the training data balanced. Second, before $\textbf{DSM}_F$ accumulates enough positive examples, we can also boost GBRT using synthetic positive data: if the user interest has a convex shape, as long as there are two positive examples, we can draw points from the line that connects these two examples, and treat these points as synthetic positive examples to balance the training data.

How many features to select? Another issue is that we do not know how many features to select, or the exact value of top-k. The user does not offer this information a priori. Choosing the right value of k is nontrivial because GBRT may not have high confidence for selecting the correct features in earlier iterations. To formalize the notion of confidence, we utilize the *feature importance scores* provided by GBRT. GBRT is a linear combination of decision tree base-learners, where each decision tree intrinsically selects features for splitting nodes. For each decision tree, the importance score of a feature is computed as weighted sum of impurity decreases for all the nodes where the feature is used [18]. Then this score is averaged over all trees. Given feature importance scores, we propose two strategies to characterize "sufficient confidence" of GBRT for feature selection.

Proportion of nonroot trees: The intuition is that all decision trees must be weak learners and hence find some features useful in distinguishing the positive class from the negative class. Based on this intuition, we wait until the iteration where all trees become nonroot trees, hence likely to be weak learners. Then we believe that the confidence of GBRT has reached a sufficient level.

Entropy of Importance Scores (EIS): The next intuition is that we prefer to have a lower entropy of the importance scores across all the features. That is, the distribution of importance

scores departs from a uniform distribution and becomes concentrated. Based on this, our second strategy is to wait until EIS has dropped significantly below the expected entropy of uniform importance scores, i.e., the importance scores of some features really stand out. Then we think that the confidence of GBRT has been sufficient.

Adaptive Feature Selection: We next devise adaptive strategies to decide top-k features, depending on whether the current iteration has reached the point of sufficient confidence for feature selection, based on any of the above strategies. Before reaching this point, we perform conservative feature filtering: we select top-k features that account for 50% of the total feature importance scores. After GBRT reaches the point of sufficient confidence, we perform aggressive feature selection by scanning the ranked list of feature importance scores and choosing the top-k features such that the k^{th} feature and the $k+1^{th}$ feature have the largest gap in the ranked list. When GBRT chooses the same top-k features for 10 iterations, we consider the choice of relevant features stable and use them to build $\mathbf{DSM}_F$.

4.3 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 [15, 35], active search [39], and explore-by-example systems, Aide [31, 32] and LifeJoin [25].

4.3.1 Experimental Setup

Datasets: Our evaluation used two datasets.

(1) Sloan Digital Sky Survey (SDSS, 190 million tuples): this dataset contains the "PhotoObjAll" table with 510 attributes[2]. 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 3.3 allowed us to use the 1% sample for data

[2]http://www.sdss3.org/dr8/

exploration, yet with bounded difference of $\varepsilon \leq .001$ 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.

To be more specific, to collect this Car dataset, we downloaded a dataset of cars from http://www.teoalida.com/. It contains 5622 vehicles from 407 models of 43 different makes between 2016 and 2018. Each vehicle is described by a unique identifier and attributes about the engine, the transmission, the warranty, etc. There were a lot of missing values. We manually checked authoritative websites such as Edmunds to fill in missing values. Moreover, we downloaded retail prices through the Edmunds API. After data cleaning, we had 27 attributes with sufficient data for data exploration. Among these attributes, 15 of them are numerical attributes and 12 are categorical attributes.

Note that in the study, we asked the user to manually try different SQL queries, for which we used the original 27 attributes. When we ran the simulation using our active learning system, we used the transformed dataset with *one-hot encoding* because most classifiers require the categorical attributes to be transformed using one-hot encoding. In such transformation, a categorical attribute with k values is transformed to k features, with one feature per distinct value. This can increase 27 attributes in our dataset to 547 features in total. We perform the above transformation on demand when the user chooses the attributes for exploration and the database sample is loaded into memory.

User Interest Queries: We extracted 8 query templates from the SDSS query release [81] to represent user interests. They allow us to run *simulations* of user exploration sessions, as in [31, 32], by using the true query answers as the proxy for user labeling. The 8 templates are shown in Table 4.1. Each template is instantiated with different constants to vary query selectivity in [0.01%, 10%]. Our system does **not** need to know these templates in advance but instead learns their decision boundaries on the fly. 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

Table 4.1: Query templates with different selectivity values

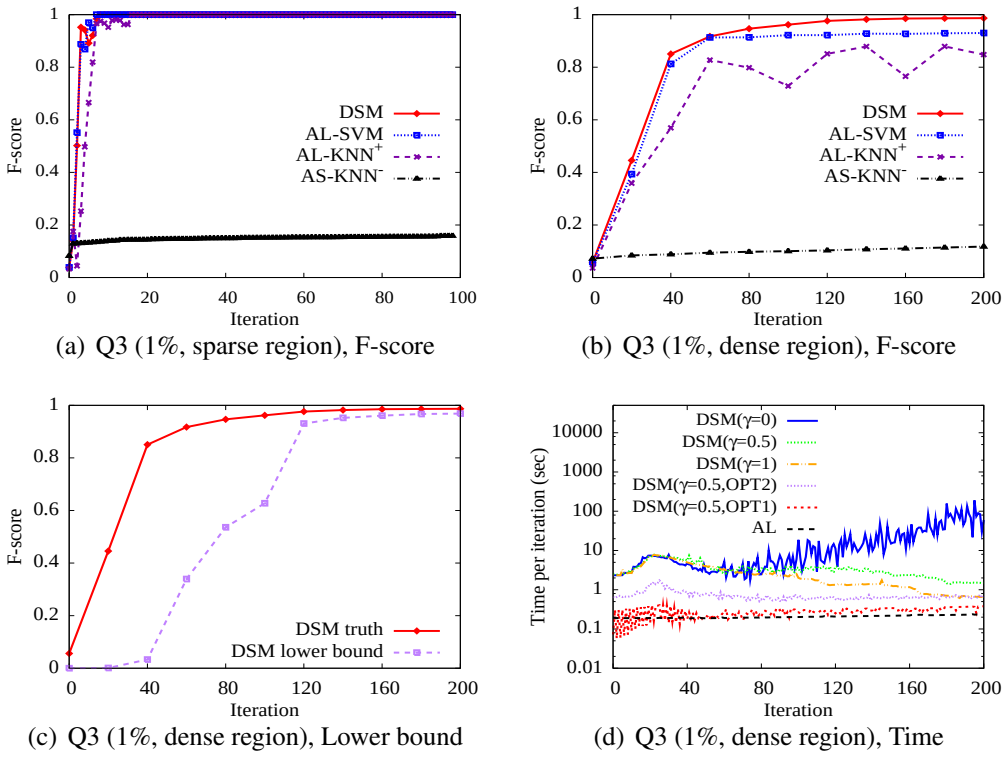
Query template
Q1 (rectangle): $rowc \in [a_1, a_2] \wedge colc \in [b_1, b_2]$
Q2 (ellipse): $((rowc - a_1)/b_1)^2 + ((colc - a_2)/b_2)^2 < c^2$
Q3 (rectangle): $ra \in [a_1, a_2] \wedge dec \in [b_1, b_2]$
Q4 (outside a circle): $rowv^2 + colv^2 > c^2$
Q5: 4D queries combining two queries from Q1-Q4
Q6: 6D queries combining three from Q1-Q4
Q7: 4D, $(x_1 > a + b \cdot x_2) \wedge (x_3 + c \cdot \log_{10} x_4^2 < d)$
Q8: 6D-11D queries with 4-9 irrelevant attributes

$\mathbf{Q}_c^-$ class. Q5, Q6 and Q8 combine these attributes, optionally with irrelevant 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 $x_4 \in \mathbb{R}$. For Car dataset, we obtained 18 queries from our user study. We defer a detailed discussion to §4.3.4 and present all these queries in Appendix A Table A.1.

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.

4.3.2 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 [15, 35] 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**) [39] 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$^-$. We also did hyper-parameter tuning to achieve best accuracy of **AS** under per-iteration time and memory constraints. All tests were run up to 200 labeled examples (if without specification) or when the lower bound of F-score reaches 99%. In all plots, the x-axis is the number of labeled examples.



(a) Q3 (1%, sparse region), F-score

(b) Q3 (1%, dense region), F-score

(c) Q3 (1%, dense region), Lower bound

(d) Q3 (1%, dense region), Time

Figure 4.4: **DSM** for 2D queries, compared to **AL** and **AS**

Expt 1 (2D queries): We run 2D queries from templates Q1 to Q4. Since the results show similar trends, we show the F-score, lower bound, and time measurement for Q3 (1%) in Figure 4.4(a)-4.4(d).

Regarding accuracy, **DSM** outperforms **AL**, and **AL** outperforms **AS**. (1) A factor that affects performance 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 4.4(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 F-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 most uncertain region of the model, e.g., close to the decision boundary, hence offering better F-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

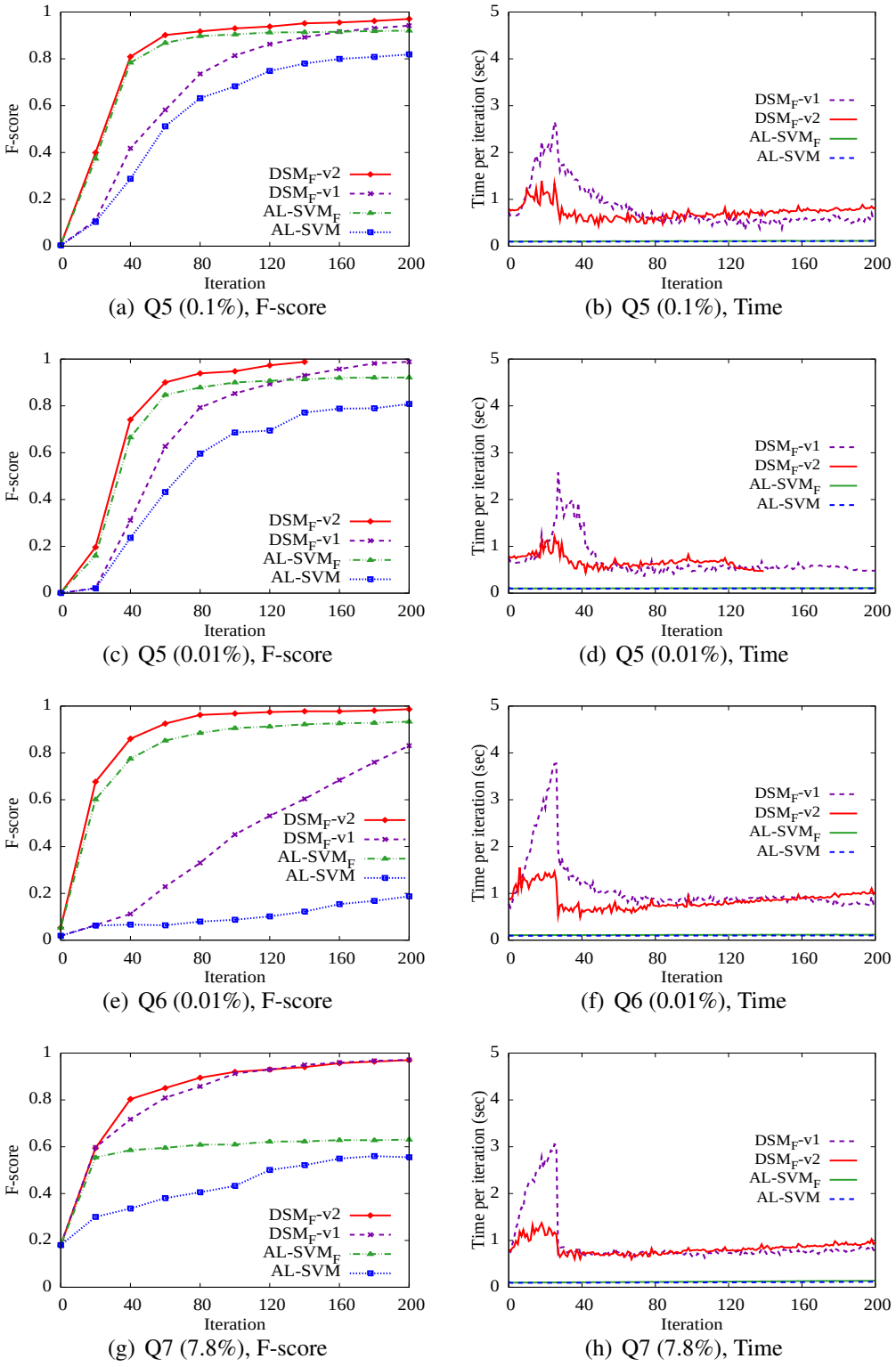
90 examples on average for Q1-Q4. **AL** works better with the SVM than the kNN classifier, but even **AL-SVM** cannot reach 95% for Q3 or most other workloads. Finally, **AS** performs much worse than **AL-KNN**$^+$ while both use a kNN classifier.

DSM further offers a lower bound in F-score. Figure 4.4(c) shows that the lower bound is quite close to the true F-score.

The time cost of **DSM** depends on the sampling method. Recall from Algorithm 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 3.3, 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 3.3, which cuts down the per-iteration time to within 1-2 seconds when the size of the evaluation set is always 1.9 million. Although OPT1 is faster than OPT2, using a larger evaluation set is guaranteed to improve accuracy remarkably. 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 in the following experiments.

Expt 2 (Evaluation of our proposed techniques in high dimensions): We next show the results of our proposed techniques for 4D-6D queries in dense regions, as learning the queries in dense regions is more challenging than learning those in sparse regions. Without specification, the user queries presented in the rest of this thesis are in dense regions. As shown in Figure 4.5, we compare the following four algorithms:

- **AL-SVM**: **AL** with an SVM.



(a) Q5 (0.1%), F-score (b) Q5 (0.1%), Time

(c) Q5 (0.01%), F-score (d) Q5 (0.01%), Time

(e) Q6 (0.01%), F-score (f) Q6 (0.01%), Time

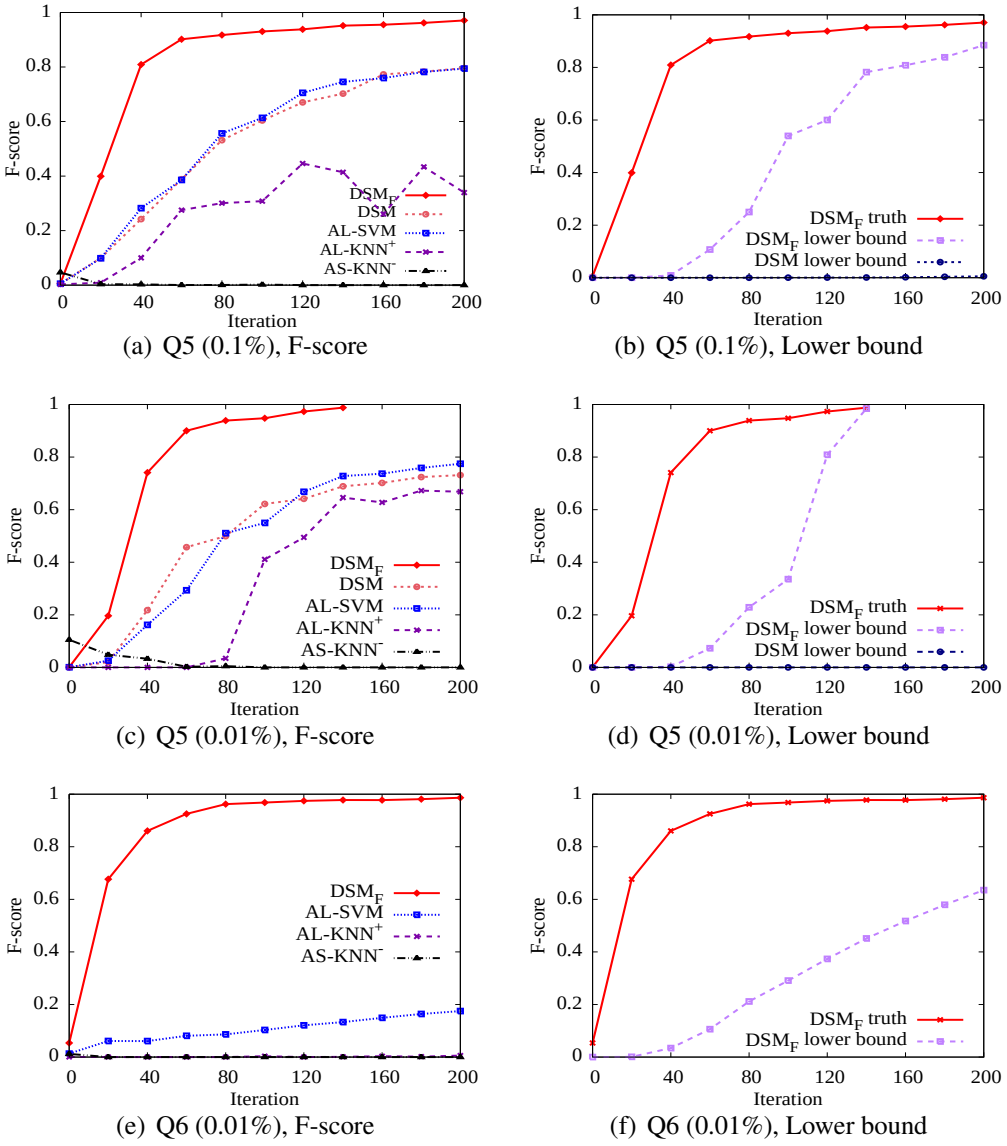
(g) Q7 (7.8%), F-score (h) Q7 (7.8%), Time

Figure 4.5: Comparison of two versions of **DSM**$_F$ in high dimensions

- **AL-SVM$_F$**: **AL** with a factorized SVM.
- **DSM$_F$**-v1: algorithm proposed in 4.1.1.
- **DSM$_F$**-v2: algorithm proposed in 4.1.2.

The main observations are: (1) **AL-SVM$_F$** always substantially outperforms **AL-SVM** in accuracy and **DSM$_F$**-v2 always converges considerably faster than **DSM$_F$**-v1, which demonstrates the effectiveness of the factorized classifier and sample acquisition strategy. The reason is that algorithms with the factorized classifier and sample acquisition strategy can recognize the true decision boundary more precisely and select more informative examples for user labeling. (2) The comparison within the pairs (**DSM$_F$**-v1, **AL-SVM**) and (**DSM$_F$**-v2, **AL-SVM$_F$**) show that our factorized data space model significantly improves the accuracy and accelerates the convergence. (3) As shown in the second column of Figure 4.5, the time per iteration is below 0.5 seconds for **AL-SVM** and **AL-SVM$_F$** while for **DSM$_F$**-v2, it keeps within 0.5-2 seconds and for **DSM$_F$**-v1, it's within 0.5-4 seconds (4) Since **DSM$_F$**-v2 achieves the best performance and desired efficiency, we use **DSM$_F$**-v2 as the default algorithm and denote it by **DSM$_F$** in the rest of this thesis.

Expt 3 (4D-6D queries): We now compare our best algorithm **DSM$_F$** with **AL** and **AS** and present the lower bound results provided by **DSM$_F$**. The main results of Figure 4.6 are: (1) Without factorization, **DSM** performs similarly to **AL** because the polytope model is dominated by the uncertain region. Then as shown in Figure 4.6(b), Figure 4.6(d) and Figure 4.6(f), **DSM$_F$** dramatically shrinks the uncertain region, and improves the lower bound and F-score. Interestingly, **DSM$_F$** performs even better for a lower selectivity (0.01%) of Q5 than the 1% version, as shown in Figure 4.6(a) and Figure 4.6(c). This can be explained by the diagrams in Figure 4.1. When the query region becomes smaller, more examples are negative, and for each such example x, we get more subspaces where x's projection falls in the negative region. Then the overall negative region built from these subspaces grows fast, hence reducing the size of the uncertain region. (2) Consistently, **DSM$_F$** outperforms **AL**, which further outperforms **AS**. Figure 4.6(e) shows that for the 6D query, **DSM$_F$** reaches 96% with 80 examples, **AL-SVM** cannot reach 20% with 200 examples, and **AS** has F-score

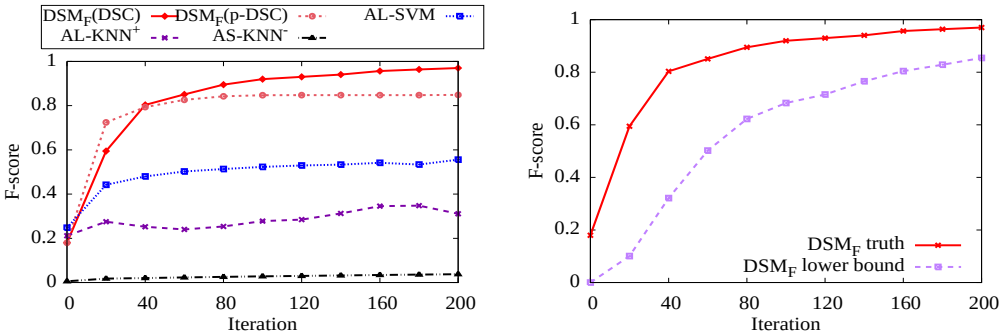


(a) Q5 (0.1%), F-score (b) Q5 (0.1%), Lower bound

(c) Q5 (0.01%), F-score (d) Q5 (0.01%), Lower bound

(e) Q6 (0.01%), F-score (f) Q6 (0.01%), Lower bound

Figure 4.6: **DSM$_F$** for 4D-6D queries, compared to **AL** and **AS**



(a) F-score for Q7 (b) Lower bound for Q7.v_1

Figure 4.7: Full vs. partial factorization with Q7

close to 0. (3) It is worth noting that the user interest region of the 6D query is not convex in the 6D data space while **DSM** only supports convex patterns. Therefore, unlike Figure 4.6(a) - 4.6(d), Figure 4.6(e) - 4.6(f) does not include **DSM**.

In summary, when the query decision boundary falls in a dense region, **DSM** (with factorization) achieves an average of 95% with 100 labeled examples and 98% with 200 labeled examples. It further maintains the per-iteration time within 1-2 seconds. **DSM** significantly outperforms Active Learning (**AL**) by 3%-237% times higher F-score. **AL** fails to reach 95% accuracy for most queries, and **AL-SVM**, the better one of the **AL**-based algorithms, achieves only an average of 75% with 100 labeled examples and 81% with 200 labeled examples. In particular, for 4D-6D queries, **AL-SVM** only reaches an average of 46% with 100 labeled examples and 58% with 200 labeled examples. Active Search (**AS**) cannot achieve F-score of more than 20% for most queries tested because it aims to maximize the number of positive examples returned within a budget, not classification accuracy.

Expt 4 (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 4.7(a) 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**,

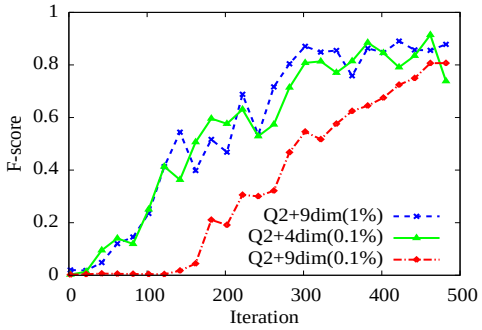


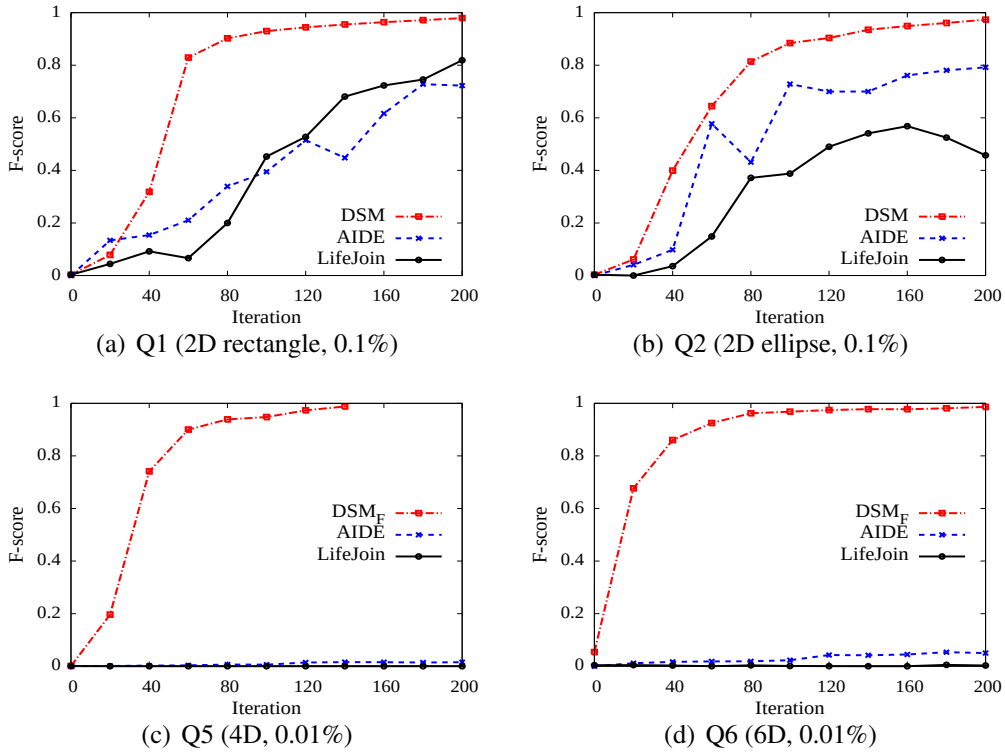
Figure 4.8: Feature selection for Q8 (6D-11D)

which cannot achieve more than 5% accuracy. Consistently, Figure 4.7(b) shows that $\mathbf{DSM}_F$ provides a tight lower bound of F-score for the non-convex pattern query Q7.v_1.

Expt 5 (Feature selection for 6D-11D queries): We now evaluate the optimizations proposed for GBRT as an online feature selection method in Section 4.2. To do so, we use the template Q8. Since the major results are similar, below we report results by extending Q2 with 4 irrelevant attributes (making it a 6D query) or with 9 irrelevant attributes (making the query 11D). Without feature selection, the F-score is 0 for these queries. As Figure 4.8 shows, after 200 iterations, feature selection improves the F-score by 20%-60%. However, if given more labeled examples, the F-score reaches above 80% for all of them. As stated earlier, GBRT is sensitive to selectivity and hence the combination of high dimensionality (11D) and low selectivity (0.1%) represents the hardest workload among the three.

4.3.3 Comparison to Alternative Systems

We next compare our system to two state-of-the-art explore-by-example systems: 1) **LifeJoin** [25] uses SVM for classification and active learning for seeking the next example for labeling. But it differs in the SVM implementation from our work. At each iteration, it uses all the labeled examples to train a collection of weak-learners (error-free regarding the training data), extracts basic predicates from these learners, and trains a **linear** SVM over these predicates. Then the SVM is used to seek the next example for labeling, which is the one closest to the SVM boundary. We implemented LifeJoin techniques as additional methods in our system. In particular, we used Random Forest (RF) with overfitted decision



(a) Q1 (2D rectangle, 0.1%) (b) Q2 (2D ellipse, 0.1%)

(c) Q5 (4D, 0.01%) (d) Q6 (6D, 0.01%)

Figure 4.9: Results for the comparison to Aide and LifeJoin systems.

trees to build the weak learns as RF is a better-known approach than a program synthesizer for this purpose. We made parameters consistent with those recommended in the paper, including the number of weak learners used (10) and the number of basic features (on the order of hundreds). 2) **Aide** [31, 32] 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.

In our experiments, **LifeJoin** and **DSM** use the same seeds as they are implemented in the same framework. **AIDE** has a standalone codebase. It uses a k-means based exploration method for skewed data. Hence, it uses its own seeds to discover cluster centroids and samples around the centroids. When comparing these systems, we exclude the overhead to find the first positive example as these systems use different methods / assumptions to find it.

Expt 5: F-score is reported in Figure 4.9(a) (rectangle pattern) and Figure 4.9(b) (ellipse pattern) for 2D query, in Figure 4.9(c) for 4D query, and Figure 4.9(d) for 6D query. The main observations are: (1) For all 2D queries, our system substantially outperforms **Aide** and

LifeJoin. For the rectangle pattern, as shown in Figure 4.9(a), **Aide** and **LifeJoin** provide comparable performances. However, for the ellipse pattern, **LifeJoin** is significantly worse in accuracy, as can be seen in Figure 4.9(b). (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.

4.3.4 User Study using a Car Database

The database used for our user study 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 his preference. We also ask the user to come up with the first positive example via (a) naming a car that he already has in mind, or (b) reviewing a sample set pre-computed 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: In the third phase, the user is asked to specify his interest precisely by (a) sending a SQL query to the database[3]; (b)

[3]For the 7 non-techncial people, we offered help to translate their requirements to SQL but otherwise did not influence the users in data exploration.

reviewing a subset of the returned tuples; (c) 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 used by $\mathbf{DSM}_F$. Table 4.2 shows in the "$\mathbf{DSM}_F$" column family that $\mathbf{DSM}_F$ 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 each other, with a small difference that non-technical people tend to specify simpler queries, hence easier for $\mathbf{DSM}_F$ 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% has a median of 29 labeled examples. Even if the decision boundary is often in a sparse region, high dimensionality makes **AL** lose performance.

Fourth, as can be seen in Table 4.2, our $\mathbf{DSM}_F$ algorithm outperforms active learning for all the queries, with drastic performance gains. Detailed profiling results reveal the following reasons: 1) *Class imbalance*: All queries have selectivity 0.9% or less. The classifier tends to ignore the minority class despite parameter tuning. A few misclassifications lead to a high loss in F-score. 2) *Uncertainty sampling* tends to select training examples close to the decision boundary. Sometimes, the distance between positive examples can be larger than the distance between positive and negative examples, which confuses maximum margin classifiers. 3) *Data distribution* can be so sparse that the positive examples for a convex

Table 4.2: Results of the car database using Manual Exploration, $\mathbf{DSM}_F$, Active Learning (AL). # A shows the number of attributes used in the user interest and # F shows the number of features (with one-hot encoding) after transforming all categorical attributes for use by the classifier. For manual exploration, T_2 shows the number of tuples reviewed in initial exploration (the 2nd phase) for the user to find the first positive example; and T_3 shows those reviewed in iterative exploration (3rd phase). Last three rows show the global Min, Max and Mdn respectively. For $\mathbf{DSM}_F$ and AL, the algorithm marked by '-' never reached the desired accuracy within 100 iterations.

Q	# A	# F	Manual Exploration			AL			$\mathbf{DSM}_F$		
			T_2	T_3	#SQL	0.8	0.95	0.99	0.8	0.95	0.99
1	6	58	0	122	20	23	-	-	10	12	13
2	8	37	13	104	49	29	39	-	4	5	6
3	8	35	8	193	29	26	27	30	14	16	19
4	6	14	17	52	17	-	-	-	7	10	13
5	6	418	18	80	10	63	-	-	9	15	17
6	4	49	12	67	8	13	21	23	5	6	7
7	6	59	2	187	16	32	42	-	6	8	9
8	8	26	11	202	17	8	9	9	4	4	5
9	4	12	11	35	8	15	21	-	5	6	7
10	8	31	3	47	11	87	-	-	9	14	16
11	6	26	16	412	11	-	-	-	11	23	26
Min	4	12	0	35	8	8	9	9	4	4	5
Max	8	418	18	412	49	-	-	-	14	23	26
Mdn	6	35	11	104	16	29	42	-	7	10	13
12	4	4	24	75	15	28	42	-	6	9	11
13	10	24	4	158	13	35	-	-	4	5	6
14	5	29	0	50	4	26	-	-	6	9	10
15	6	51	1	91	10	-	-	-	6	6	9
16	4	12	7	260	12	9	15	15	5	6	7
17	6	49	12	108	7	-	-	-	7	11	12
18	6	17	4	81	6	9	16	17	3	5	6
Min	4	4	0	50	4	9	15	15	3	5	6
Max	10	51	24	260	15	-	-	-	7	11	12
Mdn	6	24	4	91	10	28	-	-	6	6	9
Min_g	4	4	0	35	4	8	9	9	3	4	5
Max_g	10	418	24	412	49	-	-	-	14	23	26
Mdn_g	6	30	10	98	12	29	-	-	6	9	10

query appear to be several disjoint regions, which confuses the classifier, and uncertainty sampling cannot fix the issue. In contrast, **DSM**$_F$ combines samples from the uncertain region of the data space model with samples offered by uncertainty sampling, which leads to a more appropriate distribution of selected examples. 4) *Categorical* **DSM**: Except Q12, all of the other queries contain some categorical attributes. It's observed that categorical **DSM** considerably expedites the convergence and improves the accuracy.

4.4 Summary

We presented new algorithms, which leverage the subspatial convex and conjunctive properties of database queries, to overcome the slow convergence problem for active learning-based interactive data exploration. For data exploration performed with high dimensionality, our contributions are: (1) We design a factorized **DSM** (**DSM**$_F$) by breaking the high-dimensional problem into a set of low-dimensional sub-problems and build a **DSM** for each sub-problem. (2) We theoretically proved that **DSM**$_F$ provides a tighter lower bound of F-score than **DSM**. Thus, **DSM**$_F$ provides a more effective stopping criterion for accuracy-based systems. (3) We presented some procedures to test the convexity and conjunctivity assumptions. (4) We propose an online feature selection method to filter irrelevant attributes on the fly.

Our results show that our **DSM**$_F$ algorithm significantly outperforms active learning [15, 35], active search [39], and existing explore-by-example systems, Aide [31, 32] and LifeJoin [25], in accuracy and convergence speed, while maintaining the per-iteration time within 1-2 seconds. More specifically, the main results are: (1) For SDSS queries (including both low-dimensional and high-dimensional queries) , **DSM**$_F$ achieves F-scores in [92%, 96%] with 100 labeled examples and reaches 98% on average with 200 labeled examples. In contrast, active learning reaches only 81% on average with 200 labeled examples. **DSM**$_F$ achieves 3%-237% times higher F-score than active learning. (2) For Car queries collected from our user study, **DSM**$_F$ achieve 99% accuracy for all queries with a median of 10 labeled examples. By contrast, active learning cannot reach 95% for most queries even after 100 iterations and has a median of 29 labeled examples for 80%. (3) For high-dimensional

exploration with noisy (irrelevant) attributes, our online feature selection technique improves F-score from nearly 0 without feature selection, to high F-score (>0.8). (4) Active search cannot achieve F-score of more than 20% for most queries. (5) Recent systems in the explore-by-example framework, Aide and LifeJoin, fail to achieve F-score of 10% when the user interest involves 4 attributes or more and the query selectivity is below 1%.

Chapter 5

Learning with Label Noise

In previous chapters, we have presented some techniques for active learning-based interactive data exploration to overcome the slow convergence problem and improve the accuracy, under the assumption that the labels provided by the user are uncorrupted. However, in practice, label noise often occurs during the user labeling process due to two primary sources. First, the user does not fully understand her interest and may have trouble classifying some examples correctly, especially the ambiguous ones close to the underlying decision boundary. Second, some examples are mislabeled accidentally. The occurrence of noisy labels can degrade the performance of active learning, leading to decreased accuracy, slower convergence (i.e., more labeled examples required), and reduced quality of optimization strategies proposed in previous chapters. In this chapter, we focus on the label noise problem in the context of active learning. To handle noisy labels with minimal labeled examples, we develop a new active learner called Robust Dual-Space Model ($\mathbf{RDSM}_F$), similar to $\mathbf{DSM}_F$, composed of a traditional classifier and a data space model. In the active learning-based data exploration, at each iteration, we update $\mathbf{RDSM}_F$ in the following steps: (1) using a theoretically guaranteed noise distillation method to collect confident examples from noisy ones, (2) training the traditional classifier with distilled examples, and (3) building the data space model with k nearest neighbors method-based refinement. Regarding the labeling process, we propose an adaptive method to determine the source of labels. If the existing user-labeled examples

are noisier than the existing **DSM**-labeled examples, we use $\mathbf{RDSM}_F$ to provide labels for selected examples, otherwise not.

In this chapter, we propose a robust model based on **DSM** in Section 5.2 and then leverage the concept of factorization to improve its accuracy for data exploration performed with high dimensionality in Section 5.3. We further devise some optimization strategies to improve both accuracy and efficiency for our proposed model in Section 5.4. Lastly, we evaluate our proposed techniques in Section 5.6.

5.1 Label Noise Models and Analysis

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*. The symbols used in this chapter are summarized in Table 5.1.

5.1.1 Label Noise Models

We consider instance- and label-dependent label noise (ILN) for active learning-based data exploration. In our work, we make use of the following Assumption 5.1.1 (proposed by [23]) on label noise to deal with a specific type of ILN. Let $\mathbf{X}$ be the observation, Y be the uncorrupted but unobserved label and $\tilde{Y}$ be the observed but noisy label. We assume that $(\mathbf{X}, Y, \tilde{Y}) \in \mathscr{X} \times \mathscr{Y} \times \mathscr{Y}$ are jointly distributed according to a unknown distribution where $\mathscr{X} \subseteq \mathbb{R}^d$ is the data space and $\mathscr{Y} = \{-1, 1\}$ is the label space. Given an example $\mathbf{x}$ and its corresponding true label y, the noise rate model in [23]] is defined as $\rho_y(\mathbf{x}) = P(\tilde{Y} = -y \mid \mathbf{X} = \mathbf{x}, Y = y)$. For random classification noise (RCN), $\rho_{+1}(\mathbf{x}) = \rho_{-1}(\mathbf{x}) = \rho$ (ρ is a constant). For class-conditional random label noise (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 [23], if the following assumption holds.

Symbol	Description
$\mathbf{X}$	observation
Y	uncorrupted but unobserved label
$\tilde{Y}$	observed but noisy label
$\eta(\mathbf{x})$	conditional probability $P(Y=+1 \mid \mathbf{X}=\mathbf{x})$
$\tilde{\eta}(\mathbf{x})$	conditional probability $P(\tilde{Y}=+1 \mid \mathbf{X}=\mathbf{x})$
$\rho_y(\mathbf{x})$	label noise rate $\rho_y(\mathbf{x}) = P(\tilde{Y}=-y \mid \mathbf{X}=\mathbf{x}, Y=y)$
$\rho_{y\max}$	upper bound of $\rho_y(\mathbf{x})$ for the class y
$\mathbf{x}^j$	projection of $\mathbf{x}$ in the j-th subspace
y^j	label of $\mathbf{x}^j$ in the j-th subspace
$\rho^j_{y^j}(\mathbf{x})$	label noise rate in the j-th subsapce $P(\tilde{Y}^j=-y^j \mid \mathbf{X}^j=\mathbf{x}^j, Y^j=y^j)$
D	dataset
D_{ch}	labeled data used to build the convex hull
D_{cc}	labeled data used to build the convex cones
$\mathbf{Q}_c^+$	queries whose projected user interest regions in lower-dimensional subspaces are convex
$\mathbf{Q}_c^-$	queries whose complements of projected user interest regions in lower-dimensional subspaces are convex
$\mathbf{Q}_c$	convex pattern queries $\mathbf{Q}_c \equiv \mathbf{Q}_c^+ \cup \mathbf{Q}_c^-$
D_{ld}	**DSM**-labeled examples
ρ	noise rate of all examples
ρ_u	noise rate of the user-labeled examples
ρ_d	noise rate of the **DSM**-labeled examples
$\rho_{\max}$	upper bound of the noise rates for the BCN model
$d_{\mathbf{x}}$	distance from the example $\mathbf{x}$ to the true decision boundary

Table 5.1: Notation

Assumption 5.1.1 (Key assumption in [23])**.** $\forall \mathbf{x} \in \mathscr{X}$, *we have*

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

where $\rho_{+1\max}$ *and* $\rho_{-1\max}$ *are upper bounds of noise rates over 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 [60] that a majority of the labels in the ILN model are correct on average, and it is derived from [14, 77] where $\rho_{\pm 1}$ are constant. In the rest of this chapter, we always assume that Assumption 5.1.1 holds.

Boundary-Consistence Noise (BCN). Although our proposed algorithm is designed for any kinds of label noise that satisfies Assumption 5.1.1, to demonstrate the effectiveness of our algorithm in the active learning-based data exploration framework, we focus on one special case of ILN - *Boundary-Consistence Noise (BCN)* - for experimental evaluation.

In the BCN model, the examples closer to the decision boundary are more likely to be mislabeled. As stated in [12, 33, 60], the BCN model is considered as a reasonable model for human annotator noise, and it can be applied to many real-world applications, such as speech recognition, spam filter, handwritten digit recognition, annotation tasks on Amazon's Mechanical Turk, etc.

More details of the BCN model can be founded in Section 5.5. In addition, we also conduct a performance evaluation of our proposed algorithms for the most widely studied noise model - the RCN model - and put the experimental results of RCN to Appendix B as a supplementary to our experiments.

5.1.2 Analysis of Automatic Noise Distillation

We now introduce a theoretically guaranteed data cleansing method *auto* (first proposed in [23]) which automatically collects confident examples out of a potentially corrupted dataset for the label noise problem under Assumption 5.1.1. To avoid confusion, we rename *auto* as **auto-cleansing** in this thesis.

Theoretical guarantee. In the following, we present the core of **auto-cleansing** (Theorem 2 in [23]) and its immediate corollary.

Lemma 5.1.1. *Denote by* $\eta(\mathbf{x})$ *the conditional probability* $P(Y=+1 \mid \mathbf{X}=\mathbf{x})$. *The Bayes optimal classifier is given by* $g^*(\mathbf{x}) = sgn(\eta(\mathbf{x}) - \frac{1}{2})$.

Theorem 5.1.1 (Theorem in [23])**.** *Denote by* $\tilde{\eta}(\mathbf{x})$ *the conditional probability* $P(\tilde{Y}=+1 \mid \mathbf{X}=\mathbf{x})$.$\forall \mathbf{x} \in \mathscr{X}$, *given that* $UB(\rho_{\pm 1}(\mathbf{x}))$ *is an upper bound of* $\rho_{\pm 1}(\mathbf{x})$, *we have*

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

The following corollary can be derived immediately by replacing $UB(\rho_{+1}(\mathbf{x}))$ and $UB(\rho_{-1}(\mathbf{x}))$ with $\rho_{+1\max}$ and $\rho_{-1\max}$, respectively.

Corollary 5.1.1 (Corollary in [23])**.** $\forall \mathbf{x} \in \mathscr{X}$, *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 5.1.1, 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 with those assigned by the Bayes optimal classifier under the clean distribution, i.e., g^*.

Implementation. We next explain our implementation of **auto-cleansing** in detail. In practice, $\tilde{\eta}$, $\rho_{+1\max}$ and $\rho_{-1\max}$ are usually unknown, but they can be estimated as follows.

- **Estimation of $\tilde{\eta}$**

  As mentioned in [23], there are several methods to estimate $\tilde{\eta}$ such as probabilistic classification methods, kernel density estimation methods, and density ratio estimation methods. In our work, we train a random forest model with labeled examples, and then the predicted class probabilities given by the random forest model can be treated as an estimator $\hat{\tilde{\eta}}$ for $\tilde{\eta}$.

- **Estimation of $\rho_{+1\,\max}$ and $\rho_{-1\,\max}$**

  We employ the method proposed by [23] to estimate $\rho_{+1\,\max}$ and $\rho_{-1\,\max}$: (1) find the k-nearest neighborhood $\mathscr{N}_k(\mathbf{x})$ of a given example $\mathbf{x}$ in the data space, (2) utilize $\sum_{\mathbf{x}\in\mathscr{N}_k(\mathbf{x})} \frac{\tilde{\eta}(\mathbf{x})}{k}$ and $\sum_{\mathbf{x}\in\mathscr{N}_k(\mathbf{x})} \frac{1-\tilde{\eta}(\mathbf{x})}{k}$ as the approximate upper bounds, and (3) combined with the estimation of $\tilde{\eta}$, $\rho_{+1\,\max}$ and $\rho_{-1\,\max}$ can be estimated by $\sum_{\mathbf{x}\in\mathscr{N}_k(\mathbf{x})} \frac{\hat{\tilde{\eta}}(\mathbf{x})}{k}$ and $\sum_{\mathbf{x}\in\mathscr{N}_k(\mathbf{x})} \frac{1-\hat{\tilde{\eta}}(\mathbf{x})}{k}$ respectively.

Analysis. This automatic noise distillation method **auto-cleansing** proposed by [23] is applicable to general label noise with robust performance guaranteed by the above theoretical results, and its effectiveness has been demonstrated empirically in the context of supervised learning.

However, in the active learning framework, there are two factors that may deteriorate the performance of **auto-cleansing**:

- **Limited labeled examples.** The user has not yet provided enough labeled examples, especially at the beginning of the data exploration. Given very limited labeled examples, the probabilistic classification models used for the estimation of $\tilde{\eta}$ tend to be underfitting.

- **Imbalanced class distribution.** The user interest queries usually have extremely low selectivity, i.e., the user interest examples are scarce and thus hard to hit during the data exploration. Although active learning itself can, as claimed in [35], reduce the class imbalance of a labeled dataset, its performance still suffers for imbalanced datasets [97], and the labeled examples follow a skewed class distribution. Besides, it is demonstrated in [93] that class probability estimates for imbalanced data obtained

by supervised learning are unreliable. Thus, the estimated class probability learned from the existing labeled examples is usually not accurate enough.

In the active learning framework, under the combined effects of adverse factors (limited labeled data and severe imbalance), the estimations of $\tilde{\eta}$, $\rho_{+1\,\max}$ and $\rho_{-1\,\max}$ learned from the existing labeled examples are usually not accurate enough. Consequently, the confident examples collected by **auto-cleansing** may still include a certain amount of label noise.

5.2 Robust Dual-Space Model

Due to the limitation of **auto-cleansing** in the context of active learning, to further filter label noise and eventually build an accurate **DSM**, we propose a Robust Dual-Space Model (**DSM**) in which we leverage the geometrical property of **DSM** and k nearest neighbors method to remove potential 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 noisy labels from **DSM**-labeled examples, we also design an adaptive method based on estimated noise rates to bring in **DSM**-labeled examples strategically. Lastly, we present the details of applying **RDSM** into the active learning-based interactive data exploration.

5.2.1 Data Space Model Refinement

In order 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 which we make full use of the geometrical property of **DSM** and k nearest neighbors method to cleanse **DSM** itself. In the following, we suppose that the subspatial convexity assumption holds for this method and elaborate on its application for general cases later. 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 examples from D_{ch}, which are near the boundary of the convex hull and very close to the examples from D_{cc} but inside the convex hull, are very likely to be noisy. More specifically,

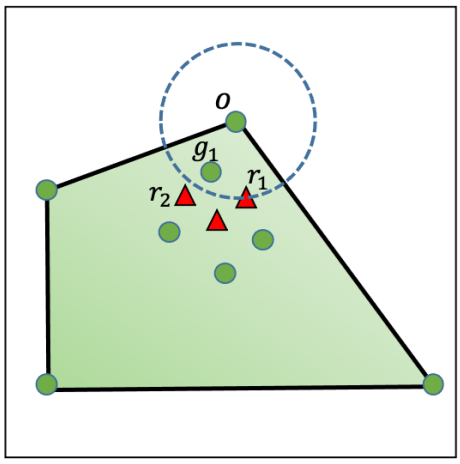
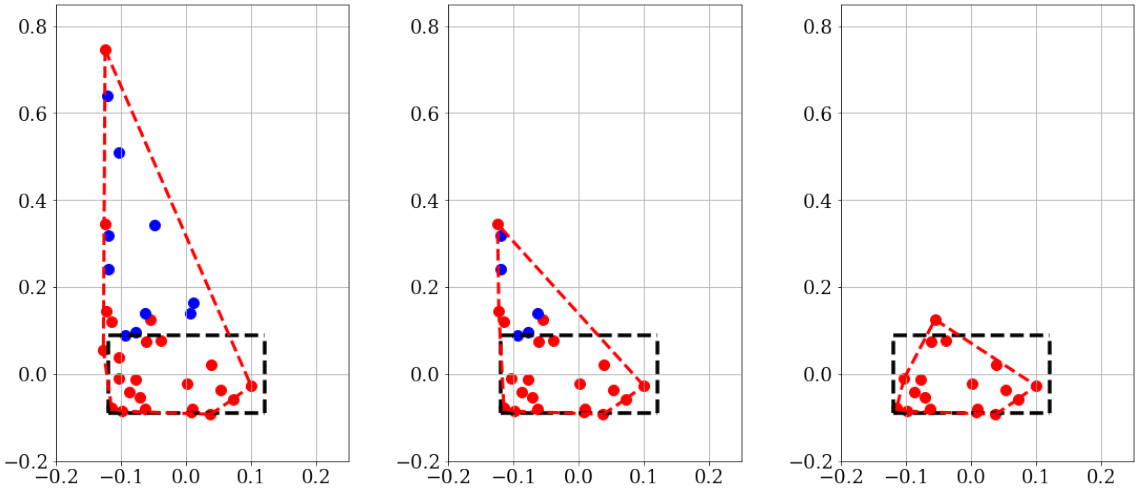


Figure 5.1: Noise filtering for the convex hull

our approach to cleansing the convex hull is that for each vertex of the current convex hull, if its k-nearest neighbors contain an abnormal example from D_{cc}, then we remove this vertex and all its neighbors that belong to D_{ch} but have a shorter distance to the vertex than that between the vertex and the discovered abnormal example. As shown in Figure 5.1, the green tetragon represents the convex hull formed by 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 and g_1 are noisy examples, so we remove both o and g_1 to avoid the over-expansion of the convex hull. We call 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.

An example of multiple-round noise filtering. However, one round of noise filtering may not be adequate. We next demonstrate the effectiveness of the iterative refinement of the convex hull by displaying the gradually shrunk convex hull over rounds of noise filtering. For ease of explanation, we assume that the user interest query is in Q_c^+, and the convex hull of the built **DSM** is created on top of positive examples. In addition, we use $k = 3$ for finding the k-nearest neighbors. In Figure 5.2, Figure 5.2(a) shows the initial convex hull built on the confident examples distilled by **auto-cleansing**, Figure 5.2(b) shows the intermediate result of the convex hull after only one round of noise filtering, and Figure 5.2(c) shows the final



(a) Initial convex hull (b) Intermediate convex hull (c) Final convex hull

Figure 5.2: Optimization on the construction of **DSM**

convex hull after a thoroughgoing two-round noise filtering. In each figure, the region inside the dashed black rectangle in the two-dimensional data space reflects the user interest region. The red points denote distilled positive examples, the blue points denote distilled negative examples, and the dashed red polygon depicts the correspondingly built convex hull. It is evident that in Figure 5.2(a), six negative examples (six red points outside the dashed black rectangle) are mistaken for positive examples confidently, which leads to the over-expansion of the convex hull. Three out of these misidentified positive examples are the vertices of the initial convex hull, and they have negative examples among their 3-nearest neighbors. By removing them and some positive examples from their neighbors, an intermediate convex hull is derived as shown in Figure 5.2(b). Since there still exist negative examples inside the intermediate convex hull, we perform another round of noise filtering and eventually obtain a conservative and much more accurate convex hull, as demonstrated in Figure 5.2(c).

Now we present the complete algorithm of Data Space Model Refinement in Algorithm 3. The input is the labeled data used to build the convex hull D_{ch}, the labeled data used to build the convex cones D_{cc} and the number of nearest neighbors k. The output is a refined **DSM** composed of a convex hull and corresponding convex cones. Algorithm 3 mainly consists of

Algorithm 3: Data Space Model Refinement

Input: labeled data used to build the convex hull D_{ch}, labeled data used to build the convex cones D_{cc}, number of nearest neighbors k

// building the refined convex hull:
1: $flag \leftarrow 1$
2: $hull \leftarrow$ `getConvexHull`(D_{ch})
3: $noise \leftarrow$ `findNoiseInCH`$(hull, D_{cc})$
4: **while** $|noise| > 0$ and $flag > 0$ **do**
5: $prev\_D_{ch} \leftarrow D_{ch}$, $prev\_noise \leftarrow noise$
6: $knbrs \leftarrow$ `kNearestNeighbors`$(D_{ch}, noise, k)$
7: **for** $x \in hull.vertices$ **do**
8: **if** $knbrs(x) \cap noise \neq \emptyset$ **then**
9: $fv \leftarrow$ `getNoisyExamples`$(knbrs(x), noise)$
10: $D_{ch}.remove(fv)$
11: $hull \leftarrow$ `convexHull`(D_{ch})
12: $noise \leftarrow$ `findNoiseInCH`$(hull, noise)$
13: **if** $D_{ch} = prev\_D_{ch}$ and $noise = prev\_noise$ **then**
14: $flag \leftarrow 0$
// building refined convex cones:
15: $cones \leftarrow \emptyset$
16: **for** $x \in D_{cc}$ **do**
17: **if** $x \notin hull$ **then**
18: $cone \leftarrow$ `getACone`$(hull, x)$
19: $cones.append(cone)$
20: **return** $(hull, cones)$

two parts: building a refined convex hull (line 1-13) and building refined convex cones (line 14-19).

Refinement of the convex hull. To refine the convex hull, we start by assigning a $flag$ as 1 (line 1), initializing the convex hull denoted by *hull* using the labeled set D_{ch} (line 2), and collecting the examples which are from D_{cc} but found inside the current *hull*, denoted as *noise* (line 3). In particular, $flag = 1$ if there exist potentially mislabeled examples among the vertices of *hull*, otherwise $flag = 0$. Lines 4-14 update *hull* and *noise* iteratively. For each update, we first find out the k nearest neighbors for each example located inside *hull*, i.e., $D_{ch} \cup noise$ (line 6), then identify potentially mislabeled examples out of the vertices of *hull* together with their neighbors and remove these noisy examples from *hull* (line 7-10),

and eventually rebuild *hull* based on the latest D_{ch} and update *noise* (line 11-12). If no noisy examples are discovered during an update, i.e., $flag = 0$ (line 14), or *noise* is empty, the loop for updating the convex hull stops.

Refinement of the convex cones. Regarding the refinement of convex cones, we perform simple filtering by checking whether an example x in D_{cc} is inside the refined hull or not. Only when $x \notin hull$, will it be used to build a convex cone.

5.2.2 Adaptive Auto-Labeling

Recall that in Section 3.1, under the assumption that the user-labeled examples are uncorrupted, we manage to save user-labeling effort and speed up model convergence by leveraging **DSM** to label selected examples that fall into **DSM**'s positive region or negative region. We call this labeling process **DSM**'s *auto-labeling*. However, even with advanced data cleansing methods composed by **auto-cleansing**, if the user-labeled examples are polluted by label noise, **DSM** built on confident examples may be in bad shape in some cases, especially when the noise rate is high or there are no enough labeled examples at the beginning. If a potentially imprecise **DSM** is allowed for *auto-labeling* all the time in any case, it will bring more noisy labels to the labeled dataset (user-labeled examples + **DSM**-labeled examples) and lead to more unsatisfactory performance. As a consequence, we need a more sophisticated method to guide *auto-labeling* such that *auto-labeling* can not only fulfill the expectations to minimize user-labeling effort and improve the model convergence but also avoid introducing many additional noisy examples into the labeled examples.

We propose an *adaptive auto-labeling* method to control the on-off switch of *auto-labeling*. The adaptive auto-labeling method is composed of the following two steps.

1. Estimation of the noise rate of user-labeled examples (ρ_u) and the noise rate of **DSM**-labeled examples (ρ_d)

   We consider the labeled examples filtered by **auto-cleansing** as mislabeled examples, and estimate ρ_d and ρ_u by computing the ratio of mislabeled examples in **DSM**-labeled examples and user-labeled examples, respectively. Formally, we have

Definition 5.2.1. *Assume that there are n labeled examples S, among which n_u examples S_u are labeled by the user and n_d examples S_d obtain labels from* **DSM***'s auto-labeling, $n_u + n_d = n$. We denote the labeled examples filtered by* ***auto-cleansing*** *as M and define the estimations of noise rates as follows:*

- *The estimated overall noise rate:*

$$\hat{\rho} = \frac{|M|}{n}$$

- *The estimated noise rate of user-labeled examples:*

$$\hat{\rho}_u = \frac{|M \cap S_u|}{n_u}$$

- *The estimated noise rate of* **DSM***-labeled examples:*

$$\hat{\rho}_d = \frac{|M \cap S_d|}{n_d}$$

In particular, if **DSM** is never used for *auto-labeling*, then $n_d = 0$ and $\hat{\rho} = \hat{\rho}_u$. For **DSM** with auto-labeling, $\hat{\rho}$ is a weighted average of $\hat{\rho}_u$ and $\hat{\rho}_d$.

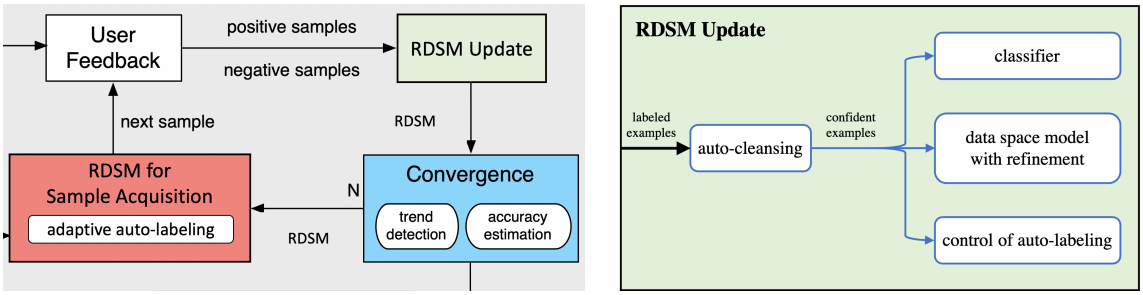
2. Criterion for an on-off switch of *auto-labeling*

   The criterion used in our current implementation is: if $\hat{\rho}_d < \hat{\rho}_u$, we turn on *auto-labeling*, otherwise we turn off *auto-labeling* and only require labels from the user.

The effectiveness of this adaptive auto-labeling method has been demonstrated empirically in our experimental evaluation in Section 5.6.

5.2.3 Robust Dual-Space Model for Data Exploration

We now propose a Robust Dual-Space Model (**RDSM**) by integrating the data cleansing methods (**auto-cleansing** and *data space model refinement*) and *adaptive auto-labeling* into **DSM**. Figure 5.3(a) shows that we utilize **RDSM**, which fulfills two functionalities - prediction and sampling, as the active learner for active learning-based interactive data exploration.



(a) Iterative learning and exploration (b) **RDSM** update

Figure 5.3: **RDSM** for data exploration

(1) *Prediction*: As stated in Section 3.1, we first use the refined data space model for prediction. Only when the example falls in the uncertain region, we resort to the classifier.

(2) *Sampling*: We leverage a hybrid sample acquisition method that alternates between uncertainty sampling from the unlabeled pool and uncertainty sampling from the uncertain region of the refined data space model, as discussed in Section 3.1.

There are two major differences between **RDSM** and **DSM**: the model update and the auto-labeling method (**RDSM Update** and **RDSM for Sample Acquisition** in Figure 5.3(a). More specifically, at each iteration, we update **DSM** based on the labeled examples directly and always allow **DSM** for auto-labeling. While for the update of **RDSM**, as depicted in Figure 5.3(b), 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 of applying **RDSM** into active learning-based interactive data exploration and highlight the lines (marked in blue) that differ from Algorithm 1, as shown in Algorithm 4. Since the framework remains similar, we only discuss the two major differences as follows:

1. Update of **RDSM**

   Lines 6-14 update the **RDSM**. More specifically, we collect clean labeled examples

Algorithm 4: Robust Dual-Space Model for Data Exploration

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

$D_{labeled} \leftarrow D_0$, $D_{unlabeled} \leftarrow D \setminus D_0$
$R^+ \leftarrow \emptyset$, $R^- \leftarrow \emptyset$
$D^+ \leftarrow \emptyset$, $D^- \leftarrow \emptyset$, $D^u \leftarrow D$
$D_{lu} \leftarrow D_0$, $D_{ld} \leftarrow \emptyset$
repeat
  // training the Robust Dual-Space Model:
  $D_{clean} \leftarrow$ `distill_by_auto` $(D_{labeled})$
  $classifier \leftarrow$ `train_classifier` (D_{clean})
  $auto\_labeling \leftarrow$ `compare_noise_rates` $(D_{labeled}, D_{clean})$
  $(R^+, R^-) \leftarrow$ `train_data_space_model` (D_{clean})
  if $D_{lu} = \emptyset$ and $D_{ld} \neq \emptyset$ **then**
    $(D^+, D^-, D^u) \leftarrow$ `threeSets` $(R^+, R^-, D^+, D^-, D^u)$
  else
    $(D^+, D^-, D^u) \leftarrow$ `rebuilding_threeSets` (R^+, R^-, D)
  $accu \leftarrow$ `threeSetMetric` (D^+, D^-, D^u)
  // applying the combined query strategy:
  $D_{lu} \leftarrow \emptyset$, $D_{ld} \leftarrow \emptyset$, $x^* \leftarrow \emptyset$
  if $rand() \leq \gamma$ **then**
    //sampling from the uncertain region in the data space model:
    $pool \leftarrow$ `subsample`(D^u, m)
    $x^* \leftarrow$ `sample_acquisition`$(pool, classifier)$
    $D_{lu} \leftarrow$ `get_labels_from_user`(x^*)
  else
    //uncertainty sampling around the classifier boundary:
    $pool \leftarrow$ `subsample`$(D_{unlabeled}, m)$
    $x^* \leftarrow$ `sample_acquisition`$(pool, classifier)$
    for $x \in x^*$ **do**
      if $x \in (R^+ \cup R^-)$ and $auto\_labeling$ **then**
        $D_{ld} \leftarrow D_{ld} \cup \{$`get_labels_from_rdsm`$(x)\}$
      else
        $D_{lu} \leftarrow D_{lu} \cup \{$`get_labels_from_user`$(x)\}$
  // updating the labeled and unlabeled data sets:
  $D_{labeled} \leftarrow D_{labeled} \cup D_{lu} \cup D_{ld}$
  $D_{unlabeled} \leftarrow D_{unlabeled} \setminus x^*$
until $accu \geq \lambda$ **or** `reachedMaxNum()`
`finalRetrieval`$(D, (R^+, R^-), classifier)$

Algorithm 5: `rebuilding_threeSets` Rebuild partitions

Input: data space model (R^+, R^-), database D

1: $D^+ \leftarrow \{x \in D | x \in R^+\}$
2: $D^- \leftarrow \{x \in D | x \in R^-\}$
3: $D^u \leftarrow D \setminus (D^+ \cup D^-)$
4: **return** (D^+, D^-, D^u)

D_{clean} distilled by **auto-cleansing** (line 6) and then use D_{clean} to train both a data space model with refinement (line 9) and a classifier (line 7).

Line 10-13 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 examples. Otherwise, they are updated incrementally. The former corresponds to Algorithm 5, and the latter corresponds to Algorithm 2. The implicit assumption of the incremental update method in Algorithm 2 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 less labeled examples, so we correct the existing partitions by rebuilding the partitions according to the latest data space model (Algorithm 5). Rebuilding the partitions enables **RDSM** to self-correct the partitions and thus alleviates the effect of label noise on its performance. The only downside of the rebuilding method is that it is very time-consuming compared to the incremental update method if D^u is small. While the design of incremental update can gradually reduce the time cost of three-set partitioning with the shrink of D^u over iterations, rebuilding the partitions over the entire dataset takes time. To improve the time efficiency for rebuilding the partitions, we employ some optimization methods, e.g., *distributed computing*. More details can be found in 5.4.

2. Adaptive Auto-labeling

   Line 8 determines whether to utilize **RDSM** later for auto-labeling by computing the estimated noise rates for both user-labeled examples $\hat{\rho}_u$ and **DSM**-labeled examples $\hat{\rho}_d$ and comparing them. The on-off switch *auto_labeling* is assigned to be *True* if $\hat{\rho}_d < \hat{\rho}_u$, otherwise *False*.

   Line 24 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 turning on auto-labeling all the time.

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.

5.3.1 Factorized Robust Dual-Space Model and Algorithm

For data exploration performed with high dimensionality, to further boost performance, we leverage the idea of factorization to devise the factorized **RDSM** (**RDSM**$_F$). Algorithm 6 presents how to use the **RDSM**$_F$ for data exploration. This algorithm is almost the same as Algorithm 4 except for the parts marked in blue. In particular, the `train_rdsm` method in line 9 updates **RDSM** in each subspace corresponding to lines 6-9 in Algorithm 4. In lines 18 and 22, we switch from uncertainty sampling to the factorized uncertainty sampling strategy according to Equation 4.5 in Section 4.1.2. In particular, **RDSM** can be regarded as a special case of **RDSM**$_F$ with $K = 1$.

5.3.2 Effect of Factorization

Now we provide some analyses 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

Algorithm 6: Factorized Robust Dual-Space Model for Data Exploration

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

$D_{labeled} \leftarrow D_0$, $D_{unlabeled} \leftarrow D \setminus D_0$
$classifier \leftarrow \{h_1, \cdots, h_K\}$
$R^+ \leftarrow \{R^{1+}, \cdots, R^{K+}\}$, $R^{j+} = \emptyset, \forall j = 1, \cdots, K$
$R^- \leftarrow \{R^{1-}, \cdots, R^{K-}\}$, $R^{j-} = \emptyset, \forall j = 1, \cdots, K$
$D^+ \leftarrow \emptyset$, $D^- \leftarrow \emptyset$, $D^u \leftarrow D$
$D_{lu} \leftarrow D_0$, $D_{ld} \leftarrow \emptyset$
repeat
  // training the Robust Dual-Space Model on each subspace:
  for k from 1 to K **do**
    $(R^{k+}, R^{k-}, h_k, auto\_labeling) \leftarrow$ `train_rdsm` $(D^k_{labeled})$
  if $D_{lu} = \emptyset$ and $D_{ld} \neq \emptyset$ **then**
    $(D^+, D^-, D^u) \leftarrow$ `threeSets` $(R^+, R^-, D^+, D^-, D^u)$
  else
    $(D^+, D^-, D^u) \leftarrow$ `rebuilding_threeSets` (R^+, R^-, D)
  $accu \leftarrow$ `threeSetMetric` (D^+, D^-, D^u)
  // applying the combined query strategy:
  $D_{lu} \leftarrow \emptyset$, $D_{ld} \leftarrow \emptyset$, $x^* \leftarrow \emptyset$
  if $rand() \leq \gamma$ **then**
    //sampling from the uncertain region in the data space model:
    $pool \leftarrow$ `subsample`(D^u, m)
    $x^* \leftarrow$ `sample_acquisition`$(pool, classifier)$
    $D_{lu} \leftarrow$ `get_labels_from_user`(x^*)
  else
    //uncertainty sampling around the classifier boundary:
    $pool \leftarrow$ `subsample`$(D_{unlabeled}, m)$
    $x^* \leftarrow$ `sample_acquisition`$(pool, classifier)$
    for $x \in x^*$ **do**
      if $x \in (R^+ \cup R^-)$ and $auto\_labeling$ **then**
        $D_{ld} \leftarrow D_{ld} \cup \{$`get_labels_from_rdsm`$(x)\}$
      else
        $D_{lu} \leftarrow D_{lu} \cup \{$`get_labels_from_user`$(x)\}$
  // updating the labeled and unlabeled data sets:
  $D_{labeled} \leftarrow D_{labeled} \cup D_{lu} \cup D_{ld}$
  $D_{unlabeled} \leftarrow D_{unlabeled} \setminus x^*$
until $accu \geq \lambda$ **or** `reachedMaxNum()`
`finalRetrieval`$(D, (R^+, R^-), classifier)$

examples in each subspace is lower than or equal to that in the original high-dimensional space. In addition, we explain how factorization affects the mislabeled negative examples. Last but not least, we further illustrate how factorization affects the prediction of classifiers and eventually improves performance.

Mislabeled positive examples. The user interest queries in our data exploration problems usually have extremely low selectivity, i.e., the user interest examples are scarce and hard to hit during the data exploration. It is well known that class imbalance pushes the decision boundary of classification models towards minority examples [6, 86, 93, 96]. In our problem setting, the minority examples are positive examples, which are often extremely rare compared to negative examples when the user comes to explore a very large dataset. 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 the context of data exploration, positive examples are always the minority. Therefore, we should pay 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 factorization rule - conjunctivity - defined in Section 4.1 , 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 for each class is defined below,

$$\rho_{+1}(x) = P(\tilde{Z} = -1 \mid X = x, Z = +1)$$

$$\rho_{-1}(x) = P(\tilde{Z} = +1 \mid X = x, Z = -1)$$

where $x = (x^1, x^2), x^j \in \mathbb{R}^{d_j}, j = 1,2$.

Lemma 5.3.1. *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

$$\begin{aligned}
\rho_{+1}(x) &= P(\tilde{Z} = -1 \mid X = x, Z = +1) \\
&= \frac{P(\tilde{Z} = -1, X = x, Z = +1)}{P(X = x, Z = +1)} \\
&= \frac{P(X = x, Z = +1) - P(\tilde{Z} = +1, X = x, Z = +1)}{P(X = x, Z = +1)} \\
&= 1 - \frac{P(\tilde{Y}^1 = +1, \tilde{Y}^2 = +1, X^1 = x^1, X^2 = x^2, Y^1 = +1, Y^2 = +1)}{P(X^1 = x^1, X^2 = x^2, Y^1 = +1, Y^2 = +1)} \\
&= 1 - \left(\frac{P(\tilde{Y}^1 = +1, X^1 = x^1, Y^1 = +1)}{P(X^1 = x^1, Y^1 = +1)}\right)\left(\frac{P(\tilde{Y}^2 = +1, X^2 = x^2, Y^2 = +1)}{P(X^2 = x^2, Y^2 = +1)}\right) \\
&= 1 - \big(1 - \rho^1_{+1}(x)\big)\big(1 - \rho^2_{+1}(x)\big) \\
&= \rho^1_{+1}(x) + \rho^2_{+1}(x) - \rho^1_{+1}(x)\rho^2_{+1}(x)
\end{aligned} \tag{5.1}$$

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

It is trivial to extend the Lemma 5.3.1 from two independent groups of attributes to a finite number of independent groups and obtain the following theorem.

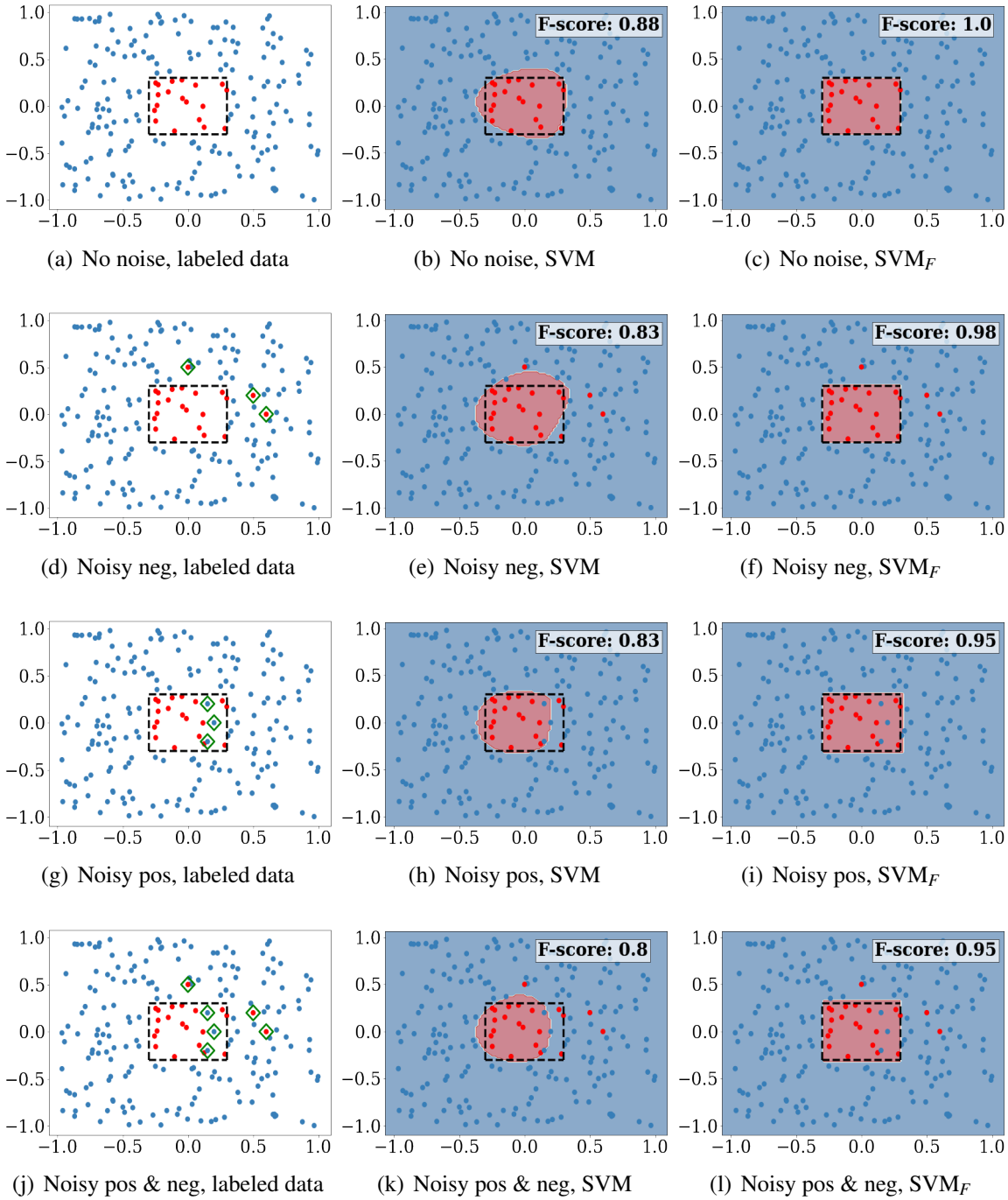
Theorem 5.3.1. *Denote by F the factorization structure composed of m groups of attributes $(X^1, X^2, \cdots, X^m)$, $X^j \in \mathbb{R}^{d_j}, j = 1,2,\cdots,m$ and $\{X^1, X^2, \cdots, X^m\}$ are independent, if the conjunctive property holds in F, we have $\rho^j_{+1} \leq \rho_{+1}, j = 1,2,\cdots,m$.*

Theorem 5.3.1 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 requested to specify which subspaces lead to the negative label, the

user may not label all the subspaces as negative. Thus, the classifiers can always gain some helpful information from the subspaces that remain the correct partial labels.

Mislabeled negative examples. We now focus on negative examples. The projection of negative examples in each subspace may fall into the projected positive region (user interest region). 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 and a positive partial label for the color. When a negative example is mislabeled as positive in the original high-dimensional space, according to the conjunctive property, its partial label in each subspace is inferred to be positive. Since the actual partial labels of the mislabeled negative example in some subspaces might be positive, these partial 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.

Analysis using a synthetic dataset. We next illustrate the influence of factorization on the prediction performance of general classifiers. For ease of explanation, we generate a 2D synthetic dataset. This dataset, which contains 200 labeled examples, is uniformly distributed in $[-1,1]^2$. We design a query pattern in which the positive examples $\mathbf{x} = (x_1, x_2)$ satisfy $|x_1| < 0.3$ *and* $|x_2| < 0.3$ and account for 9.5% of the entire dataset. From a geometrical perspective, the positive region of this query is the region within a rectangle in two-dimensional space. Learning such a pattern can be decomposed into two subtasks of learning an interval in the one-dimensional space. In Figure 5.4, red (blue) points denotes positive (negative) examples. The dashed rectangle represents the true decision boundary. We treat this dataset as training data and use it to train an SVM and a factorized SVM (SVM$_F$), respectively. In particular, SVM$_F$ has two SVMs, each trained in a one-dimensional subspace, and then combines the two SVMs according to the conjunctive property. For evaluation, we randomly sample 12k points from $[-1,1]^2$ as the test data to compute F-score. We first run SVM and SVM$_F$ with correct labels as a reference (see Figure 5.4(a)-5.4(c)), then insert



(a) No noise, labeled data (b) No noise, SVM (c) No noise, SVM$_F$

(d) Noisy neg, labeled data (e) Noisy neg, SVM (f) Noisy neg, SVM$_F$

(g) Noisy pos, labeled data (h) Noisy pos, SVM (i) Noisy pos, SVM$_F$

(j) Noisy pos & neg, labeled data (k) Noisy pos & neg, SVM (l) Noisy pos & neg, SVM$_F$

Figure 5.4: SVM vs. SVM$_F$

different types of noisy labels into the training data set and evaluate two algorithms on noisy examples (see Figure 5.4(d)-5.4(l)).

(1) *Without noisy labels.* As can be seen in Figure 5.4(b)-5.4(c), when there is no label noise, SVM without factorization achieves 88% accuracy, while SVM$_F$ obtains 100% accuracy. Consistent with our observations in Chapter 4, factorization improves the accuracy for learning without label noise.

(2) *With noisy labels.* We insert into the training data set mislabeled negative examples in Figure 5.4(d), mislabeled positive examples in Figure 5.4(g), and both types of noisy labels in Figure 5.4(g). The noisy examples are marked in green diamonds. In all cases, SVM$_F$ significantly outperforms merely SVM, and it captures the true decision boundary more accurately. When SVM only achieves 80% accuracy with both types of noisy labels, SVM$_F$ still reaches up to 95%.

In conclusion, factorization plays an important role in boosting the performance of classifiers not only in noise-free cases but also in the presence of label noise. The factorized classifiers achieve higher accuracy and capture a more precise decision boundary that can significantly improve sample acquisition.

5.4 Optimizations

We further propose a number of optimization strategies to reduce the time cost of updating the data space model and fine-tune the classifier for more robust performance in the presence of label noise.

5.4.1 Optimization on Interactive Performance

When data exploration is performed on a large dataset, it is time-consuming to update the data space model by rebuilding the three (positive, negative, and uncertain) partitions for the entire dataset every time a user-labeled example is received (Algorithm 5). To reduce

the time cost of updating the data space model while still ensuring the correctness of three partitions as the model evolves, we devise the optimization strategies from two aspects.

- **Distributed computing**

  Similar to Section 3.3.2, we apply an open-source system Ray[1] to parallelize our code, which allows us to reduce the time per iteration from 20-30 seconds to a few seconds. More specifically, we split the entire dataset (not only the uncertain dataset as described in Section 3.3.2) into several subsets and rebuild the partitions within each subset. As such, we divide the partition-building task for the entire dataset into multiple independent sub-tasks.

- **Reduce the frequency of updating the partitions**

  We observe that only a few mislabeled examples would not dramatically affect the result of the data cleansing method and the corresponding data space model built from the cleansed data, so there is no need to check the correctness of three sets very frequently. Instead of rebuilding the partitions every time we receive a user-labeled example, we only rebuild them for every k user-labeled example (k=20 in our experiments). Our experiments empirically show that reducing the frequency of rebuilding the partitions significantly reduces the time costs for most iterations while achieving decent performance in accuracy.

5.4.2 Fine-tuning the Classifier

To achieve higher accuracy and increase the generality, we automatically fine-tune hyper-parameters of the active learner. We run experiments under different levels of label noise and distinct values of the hyperparameter. Based on these experiments, we build a regression model between the best value of the hyperparameter and the estimated noise rate obtained from the **auto-cleansing** method.

We focus on tuning the regularization parameter C for SVM in this work. The parameter C, common to all SVM kernels, trades off perfect classification of training examples against

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

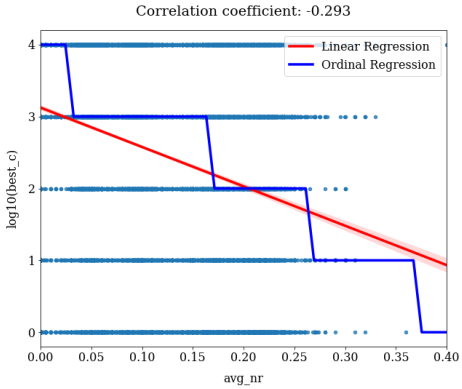


Figure 5.5: Parameter tuning of C for SVM

the maximization of the decision function's margin. A lower C encourages a larger margin, resulting in a simpler decision function, while a larger value of C aims at classifying all training examples correctly with a smaller margin. In the presence of label noise, the rule of thumb for tuning C is to decrease it[2].

We propose a quantitative approach to setting C, depending on the label noise rate. We extract the pairs of estimated label noise rate and the best C value from experimental traces and characterize the negative relationship by ordinal regression and linear regression, respectively. As depicted in Figure 5.5, in general, the higher the estimated noise rate, the smaller C the regression model returns. In our implementation, for new experiments, we first utilize **auto-cleansing** to obtain the estimated noise rate and then leverage the ordinal regression model to assign the C value for SVM. If our $\mathbf{RDSM}_F$ uses another classifier,, we can fine-tune the corresponding hyperparameters in the same way.

It is worth noting that the change of C value affects time efficiency. A larger C typically leads to increased training time [36], while a smaller C generally leads to more support vectors, which may cause a longer prediction time[3,4]. Our usage of SVM involves both training and prediction. However, our method of tuning C does not make a big difference to the time efficiency of our system. More details can be founded in Section 5.6.3.

[2]https://scikit-learn.org/stable/modules/svm.html#tips-on-practical-use
[3]https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#sphx-glr-auto-examples-svm-plot-rbf-parameters-py
[4]https://scikit-learn.org/stable/modules/svm.html#tips-on-practical-use

5.5 Simulation of Label Noise Models

In this section, we describe our simulation of two common noise models.

Boundary-Consistence Noise (BCN). As mentioned in Section 5.1.1, we focus on *Boundary-Consistence Noise (BCN)* for experimental evaluation as the BCN model is considered as a reasonable model for human annotator noise. Inspired by [33], 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

$$P(\tilde{y} \neq y \mid \mathbf{x}) = \rho_{\max} \exp\left(-\frac{(dist(\mathbf{x}, B))^2}{2\sigma^2}\right) \triangleq \rho_{\max} \exp\left(-\frac{d_{\mathbf{x}}^2}{\lambda}\right) \tag{5.2}$$

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 not trivial 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 [45], 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

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

collected examples must be very close to the boundary, and we call them *border points*.

2. **Compute distance**: Given an example $\mathbf{x}$, we 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 - 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 $\rho_{\max}$**

  The different values of $\rho_{\max}$ used for our experimental evaluation are [0.05, 0.1, 0.2, 0.3, 0.4, 0.5]

- **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

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

  In our simulation, we assign $\lambda = \frac{-1}{\log(\tau/\rho_{\max})}$ for given $\rho_{\max}$.

- **Noise injection during data exploration**

  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 5.2. During the 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 noise injection is needed.

Random Classification Noise (RCN). For RCN model, $P(\tilde{y} \neq y \mid \mathbf{x}) = \rho$, ρ is a constant. Therefore, the simulation of the RCN model is relatively simple by performing noise injection directly with label flip probability ρ during data exploration.

5.6 Experimental Evaluation

We have implemented our proposed techniques in Python. In this section, we evaluate our techniques and demonstrate the advantages of our algorithm over alternative algorithms in terms of accuracy (F-score) and interactive performance (execution time in each user labeling iteration). The experimental results in previous chapters have demonstrated that for the noise-free cases, (1) **AL-SVM** outperforms **AL-KNN$^+$**, **AL-KNN$^-$** and **Active Search**, (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 do not support learning with label noise. Thus, for the following experiments of learning with label noise, we use **AL-SVM** as the baseline. In addition, since it is much easier to recognize noisy labels when the decision boundary B falls into sparse regions, in the following, we only consider the harder problem where B falls into dense regions. All experiments start with two randomly chosen initial examples – one positive example and one negative example – and end with 200 user-labeled examples or when the lower bound of F-score reaches 99% (if **RDSM$_F$** is used). Each experiment is repeated 10 times with different initial examples, and the averaged scores are used for experimental evaluation. In the following plots, if not specified, the x-axis is the number of user-labeled examples.

5.6.1 Experimental Setup

Datasets: SDSS dataset and Car dataset introduced in Section 4.3

User Interest Queries:

- **For SDSS dataset:** As shown in Table 5.2, we extract a set of queries whose decision boundary lies in dense regions from the query templates of Table 4.1 in Section 4.3. These queries from the SDSS query release[6] reflect true user interests with varied dimensionalities (2D-6D), various patterns (**DSC** queries and p-**DSC** queries), and varied selectivities (0.01% - 7.8%). In our simulation of "human-in-the-loop" data exploration, we leverage these queries to obtain the ground truth of the user interest

[6]http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp

Dim	Query template	Selectivity
2D	**Q1 (rectangle):** $rowc \in [a_1, a_2] \wedge colc \in [b_1, b_2]$	(0.1%, 1%)
	Q2 (ellipse): $((rowc - a_1)/b_1)^2 + ((colc - a_2)/b_2)^2 < c^2$	(0.1%, 1%)
	Q3 (rectangle): $ra \in [a_1, a_2] \wedge dec \in [b_1, b_2]$	(0.1%, 1%)
	Q4 (outside a circle): $rowv^2 + colv^2 > c^2$	(0.1%, 1%)
4D-6D	**Q5**: 4D queries combining two queries from Q1-Q4	(0.01%, 0.1%)
	Q6: 6D queries combining three from Q1-Q4	0.01%
	Q7: 4D, $(x_1 > a + b \cdot x_2) \wedge (x_3 + c \cdot \log_{10} x_4^2 < d)$	(0.7%, 7.8%)

Table 5.2: SDSS query templates

and simulate the user labeling process by inquiring about the precomputed ground truth.

- **For Car dataset:** The same as that described in Section 4.3.

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.

5.6.2 Comparison of Data Cleansing Methods

We evaluate some state-of-the-art data cleansing methods by applying these methods to our system respectively to remove noisy examples before the labeled examples are fed to the classifier. Experimental results show that the **auto-cleansing** method best suits our problem. More specifically, we compare three algorithms denoted as **AL-SVM + auto**, **AL-SVM + cleanlab**, and **AL-SVM + kta** to **AL-SVM** without data cleansing beforehand. The corresponding data cleansing methods are listed as follows:

- **auto**: automatic noise distillation, i.e., the **auto-cleansing** method discussed in Section 5.1.2.

- **cleanlab**: a state-of-the-art algorithm powered by the theory of confident learning. The authors of [66, 67] proposed **cleanlab** for machine learning with noisy labels and provides an open-sourced Python package **cleanlab**[7].

[7]https://github.com/cgnorthcutt/cleanlab

$\rho_{\max}$	**AL-SVM + auto**	**AL-SVM + cleanlab**	**AL-SVM**	**AL-SVM + kta**
5%	**0.9539**	0.9519	0.9482	0.6986
10%	**0.9475**	0.9303	0.9299	0.6505
20%	**0.9009**	0.8502	0.8045	0.4181
30%	**0.7775**	0.7005	0.6313	0.3502
40%	**0.5568**	0.4854	0.4755	0.2972
50%	**0.3632**	0.3357	0.3143	0.2468

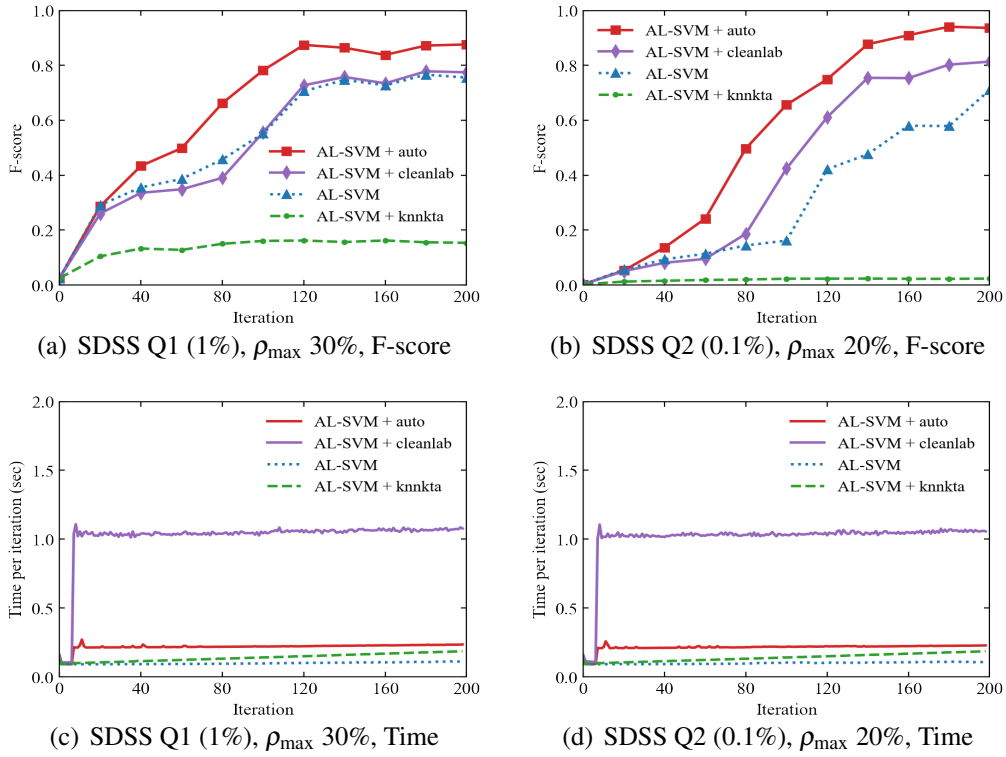
Table 5.3: Averaged F-score of different data cleansing methods at iteration 200

- **kta**: kernel target alignment (kta)-based outlier detection. Inspired by [10, 55], we use kta compute a confidence score for each labeled example and average confident scores over positive and negative sets, respectively. We consider the examples whose confidence score is larger than the average score of its class as confident examples.

Expt 1: Comparison of data cleansing methods combined with AL-SVM

We conduct the comparison experiments for the 2D query set, which consists of 8 queries with different patterns (Q1-Q4) and varied selectivity (0.1%, 1%). In particular, we use $k = 5$ for the methods which require the computation of k-nearest neighbors and start applying all the data cleansing methods into our system at iteration 10 to make a fair comparison. Table 5.3 shows, under the varied maximum of label noise rate $\rho_{\max}$ (5% - 50%), the averaged F-score of data cleansing methods over the query set. The main observations are: (1) When $\rho_{\max}$ ranges from 5% to 50%, **AL-SVM + auto** consistently achieves the highest accuracy. (2) Although for all algorithms, including **AL-SVM + auto**, a higher noise rate leads to lower accuracy, **AL-SVM + auto** reaches above 90% accuracy even when $\rho_{\max}$ is below or equal to 20%. (3) **AL-SVM + cleanlab** performs slightly better than **AL-SVM**, while **AL-SVM + kta** is even worse than **AL-SVM**, which does not use data cleansing methods at all.

Figures 5.6(a)-5.6(b) show the general trend of the data cleansing methods in terms of F-score. As shown in both figures, **AL-SVM + auto** outperforms the other algorithms at each



(a) SDSS Q1 (1%), $\rho_{\max}$ 30%, F-score

(b) SDSS Q2 (0.1%), $\rho_{\max}$ 20%, F-score

(c) SDSS Q1 (1%), $\rho_{\max}$ 30%, Time

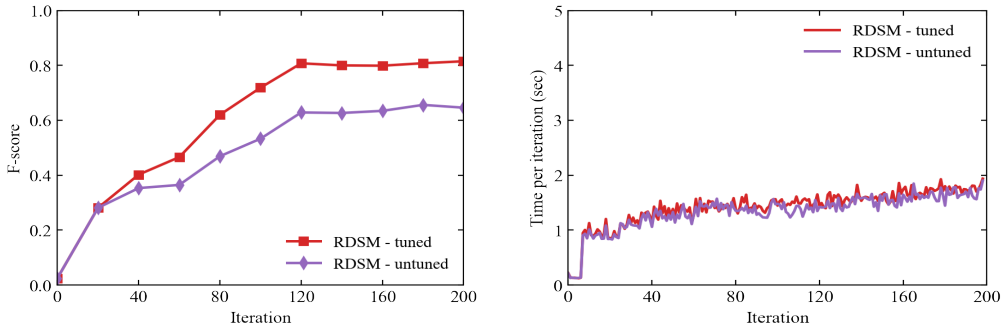
(d) SDSS Q2 (0.1%), $\rho_{\max}$ 20%, Time

Figure 5.6: Comparison of data cleansing methods

iteration. The performance of **AL-SVM + cleanlab** is better than **AL-SVM** in Figure 5.6(b), but very similar to **AL-SVM** in Figure 5.6(a). **AL-SVM + kta** performs the worst. In most cases, the observation on an individual experiment is consistent with the averaged results in Table 5.3.

In terms of interactive performance, as shown in Figure 5.6(c)-5.6(d), the per-iteration time of **AL-SVM + auto** is around 0.25 seconds. **AL-SVM** is the fastest with per-iteration time around 0.1 seconds. The time cost of **AL-SVM + kta** lies in the middle of **AL-SVM** and **AL-SVM + auto**. **AL-SVM + cleanlab** has the highest time cost - around 1 second per iteration, 4 times slower than **AL-SVM + auto**.

In conclusion, **AL-SVM + auto** usually outperforms the other algorithms in terms of accuracy and its time cost per iteration is far below a second. Therefore, we use **auto** for cleansing the labeled data set in our proposed algorithm.



(a) SDSS Q1 (1%), $\rho_{\max}$ 40%, F-score (b) SDSS Q1 (1%), $\rho_{\max}$ 40%, Time

Figure 5.7: Evaluation of parameter tuning

5.6.3 Effect of Fine-tuning the Classifier

Expt 2: Hyperparameter tuning

As stated in Section 5.4.2, we build an ordinal regression model to automatically determine the value of the regularization parameter C for SVM. The parameter C trades off the correct classification of training examples against the maximization of SVM's margin. Our proposed algorithm **RDSM** often performs similarly with or without parameter tuning. However, in some cases, parameter tuning improves accuracy substantially, for example, SDSS Q1 (1%) with $\rho_{\max} = 40\%$, as Figure 5.7 shows. We can see in Figure 5.7(a) that **RDSM** with parameter tuning (**RDSM**-tuned) outperforms **RDSM** without parameter tuning (**RDSM**-untuned) at all times and eventually offers around a 20% increase in accuracy over **RDSM**-untuned.

Regarding time efficiency, we treat distributed computing and rebuilding the partitions at each iteration as the default setting. Despite the varied C values for **RDSM**-tuned, its time cost is very similar to that of **RDSM**-untuned, as shown in Figure 5.7(b). This observation about time efficiency remains for all queries. In the following, our proposed algorithm is always run with parameter tuning.

5.6.4 Evaluation of Robust Dual-Space Algorithm

We evaluate our proposed algorithm in two cases: without factorization (**RDSM**) and with factorization (**RDSM**$_F$), and compare it to variants of active learning (**AL**). The evaluation

of **RDSM** without factorization for low-dimensional patterns demonstrates the effectiveness of our proposed data space model refinement and adaptive auto-labeling techniques beyond **auto-cleansing**. The evaluation of **RDSM**$_F$ using high-dimensional patterns shows how factorization helps improve the performance of both the data space model and the classifier.

For low-dimensional patterns, the algorithms considered are as follows:

- **AL-SVM**: **AL** with an SVM.
- **AL-SVM + auto**: **AL** with an SVM, combined with **auto-cleansing**.
- **RDSM**: Algorithm 6, without factorization.

For high-dimensional patterns, besides **AL-SVM**, we consider the following algorithms:

- **AL-SVM**$_F$: **AL** with a factorized SVM.
- **AL-SVM**$_F$ **+ auto**: **AL** with a factorized SVM, combined with **auto-cleansing**.
- **RDSM**$_F$: Algorithm 6, with factorization.

All the above algorithms use SVM as the classifier. In particular, factorized SVM refers to applying factorization into training an SVM and selecting the next samples, as discussed in Section 4.1.2.

Expt 3: Evaluation of RDSM without factorization

Figure 5.8 shows the averaged F-score comparison of **RDSM**, **AL-SVM + auto**, and **AL-SVM** over all 2D queries (8 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 cells in the upper-left corner show, when $\rho_{\max} = 5\%$, **RDSM** reaches up to 99% accuracy on average. The higher (lower) accuracy a cell is annotated with, the redder (bluer) the cell becomes. The main observations are:

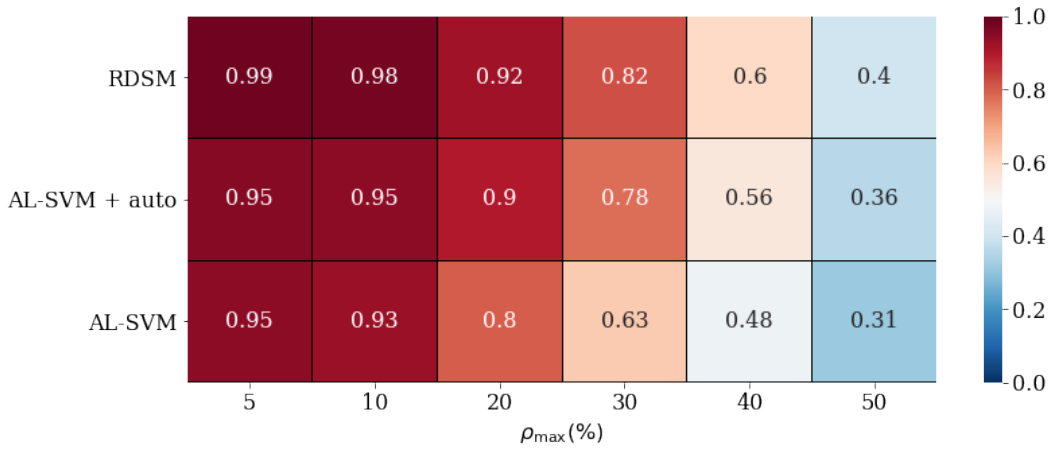
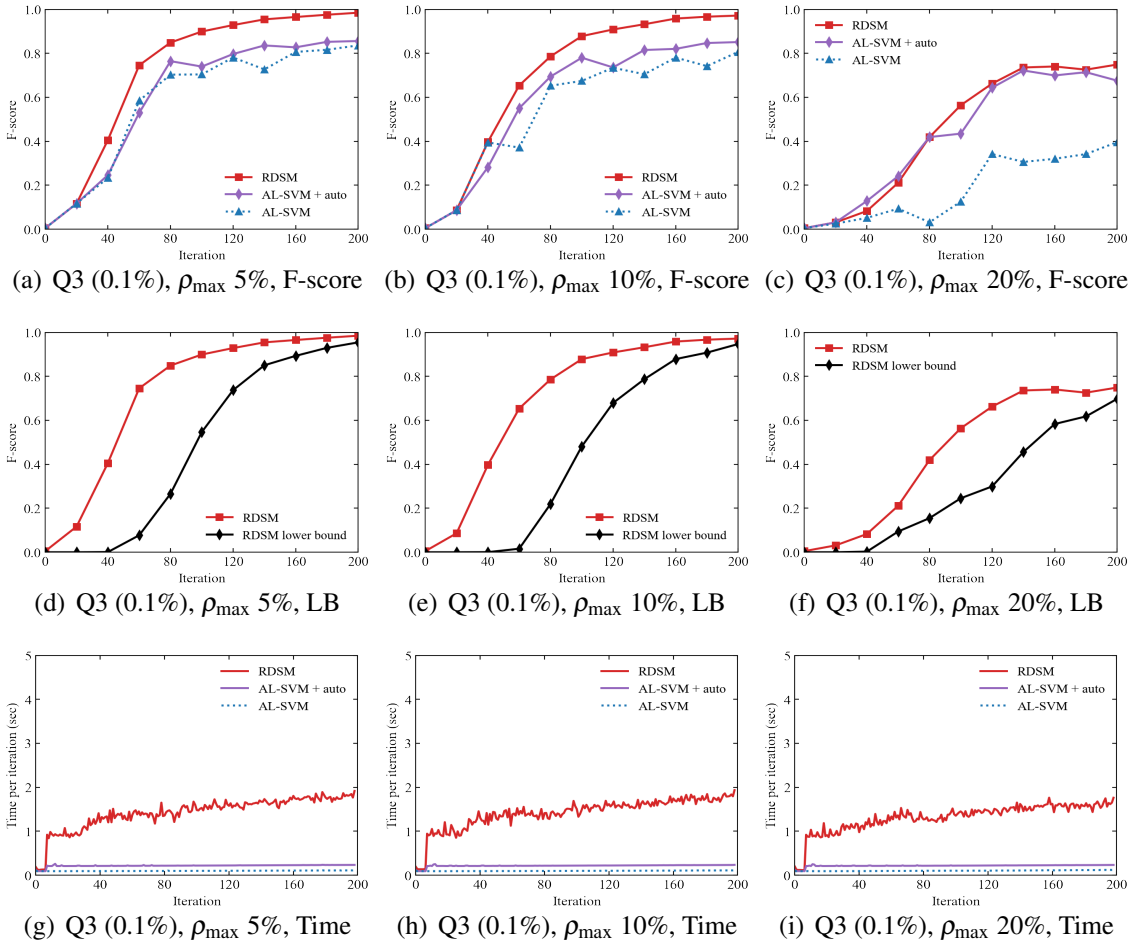


Figure 5.8: Averaged F-scores of **RDSM**, **AL-SVM + auto** and **AL-SVM** over SDSS 2D queries at iteration 200

(1) **RDSM** significantly outperforms the other two algorithms under every noise rate. More specifically, if we use $a \geq b$ to denote that the F-score of algorithm a is higher than or equal to that of algorithm b, according to the heat map, the inequality **RDSM**$\geq$ **AL-SVM + auto** $\geq$ **AL-SVM** always holds for each noise level. (2) Given a fixed ρ_{max}, there always exists a gap (2%-4%) between **RDSM** and **AL-SVM + auto** which generally 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 ρ_{max} becomes larger (i.e., the noise rate increases), all algorithms have increasingly worse accuracy. With ρ_{max} ranging from 5% to 50%, **RDSM** drops from 99% to 40%, **AL-SVM + auto** drops from 95% to 36% and **AL-SVM** drops from 95% to 31%. However, the accuracy of **AL-SVM** falls fastest when ρ_{max} rises from 5% to 30%. (4) **AL-SVM + auto** performs similarly to **AL-SVM** when ρ_{max} is relatively low (5%, 10%), but it achieves substantially higher accuracy than **AL-SVM** when ρ_{max} ranges from 20% to 50%.

We now present some experimental results of SDSS Q3 (0.1%) to show the general trends of different algorithms in terms of F-score, lower bound (LB), and time measurement. Since **RDSM** achieves decent performance for ρ_{max} up to 20%, we focus on these noise levels. The reasons behind choosing Q3 only are: (1) the results for different queries show similar trends,



(a) Q3 (0.1%), $\rho_{\max}$ 5%, F-score (b) Q3 (0.1%), $\rho_{\max}$ 10%, F-score (c) Q3 (0.1%), $\rho_{\max}$ 20%, F-score

(d) Q3 (0.1%), $\rho_{\max}$ 5%, LB (e) Q3 (0.1%), $\rho_{\max}$ 10%, LB (f) Q3 (0.1%), $\rho_{\max}$ 20%, LB

(g) Q3 (0.1%), $\rho_{\max}$ 5%, Time (h) Q3 (0.1%), $\rho_{\max}$ 10%, Time (i) Q3 (0.1%), $\rho_{\max}$ 20%, Time

Figure 5.9: **RDSM** for 2D queries, compared to **AL-SVM + auto** and **AL-SVM**

(2) Q3 (0.1%) is one of the most challenging 2D query patterns to learn, and it has the worst accuracy results for $\rho_{\max} = 20\%$ among all 2D queries.

Figures 5.9(a) - 5.9(c) show that **RDSM** surpasses the other algorithms in accuracy almost at every iteration, except at the beginning of Figure 5.9(c). **RDSM** loses to **AL-SVM + auto** at the beginning when $\rho_{\max} = 20\%$. It is due to the fact that under this noise level, given limited labeled examples, **auto-cleansing** cannot detect noisy labels precisely, and thus **RDSM** is subject to the noisy examples collected by **auto-cleansing**.

Regarding lower bound, although it has not been theoretically proven that the Three-Set Metric computed from **RDSM** must be a lower bound of the F-score. We observe from the

experimental results that **RDSM** usually captures a lower bound quite close to F-score when $\rho_{\max} \leq 20\%$, as shown in Figure 5.9(d) - 5.9(f).

With respect to time measurement, as shown in Figure 5.9(g) - 5.9(i), the time cost of each algorithm remains similar under different levels of label noise. Both **AL-SVM + auto** and **AL-SVM** have time cost per iteration below 0.5 seconds. The time cost per iteration of **RDSM** is within 2 seconds, which is reasonable for the requirement of interactive performance.

Expt 4: Evaluation of RDSM$_F$

The heat map in Figure 5.10 shows the averaged F-score comparison of **RDSM**$_F$, **AL-SVM**$_F$ **+ auto**, **AL-SVM**$_F$, and **AL-SVM** for 4D-6D queries (combined from SDSS Q1-Q4) at iteration 200. The heat map is drawn in the same format as that in Figure 5.8. The main results demonstrated in the heat map are: (1) Consistently, **RDSM**$_F$ substantially outperforms the other algorithms under every noise rate. For a fixed $\rho_{\max}$, the accuracy of algorithms is always in the order of **RDSM**$_F \geq$ **AL-SVM**$_F$ **+ auto** $\geq$ **AL-SVM**$_F \geq$ **AL-SVM**. (2) Thanks to the factorized data space model, **RDSM**$_F$ always outperforms **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 applying factorized classifier and sample acquisition strategy. (6) Similar to previous observations, due to the usage of **auto**, **AL-SVM**$_F$ **+ auto** outperforms **AL-SVM**$_F$ when $\rho_{\max}$ is higher than or equal to 20%, but its accuracy is close to that of **AL-SVM**$_F$ for relatively low $\rho_{\max}$ (5%, 10%).

Since **RDSM**$_F$ reaches above 90% accuracy for $\rho_{\max}$ up to 30%, we mainly focus on the cases where $\rho_{\max} \leq 30\%$. According to Figure 5.10, the distribution of scores under 5%

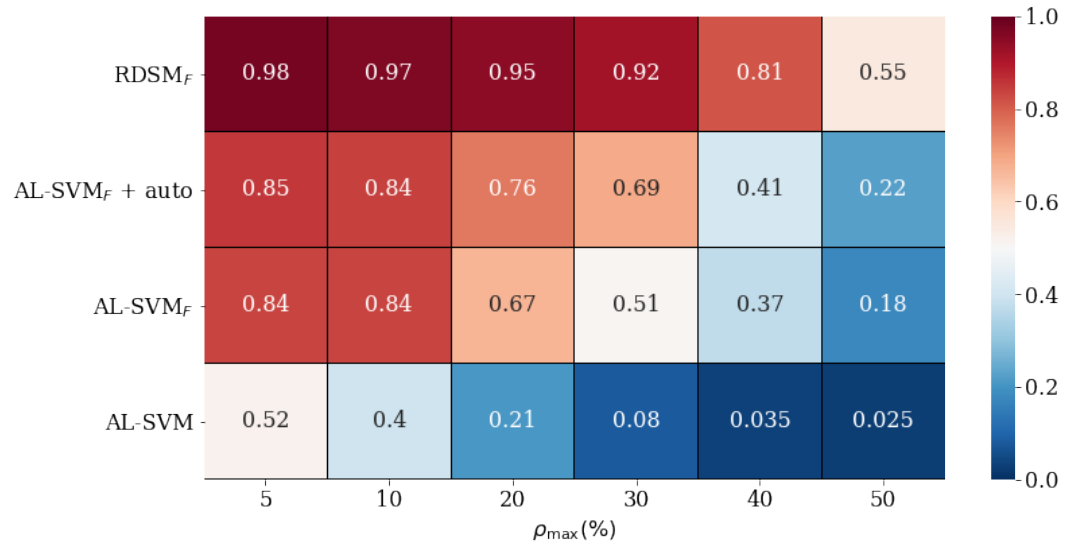
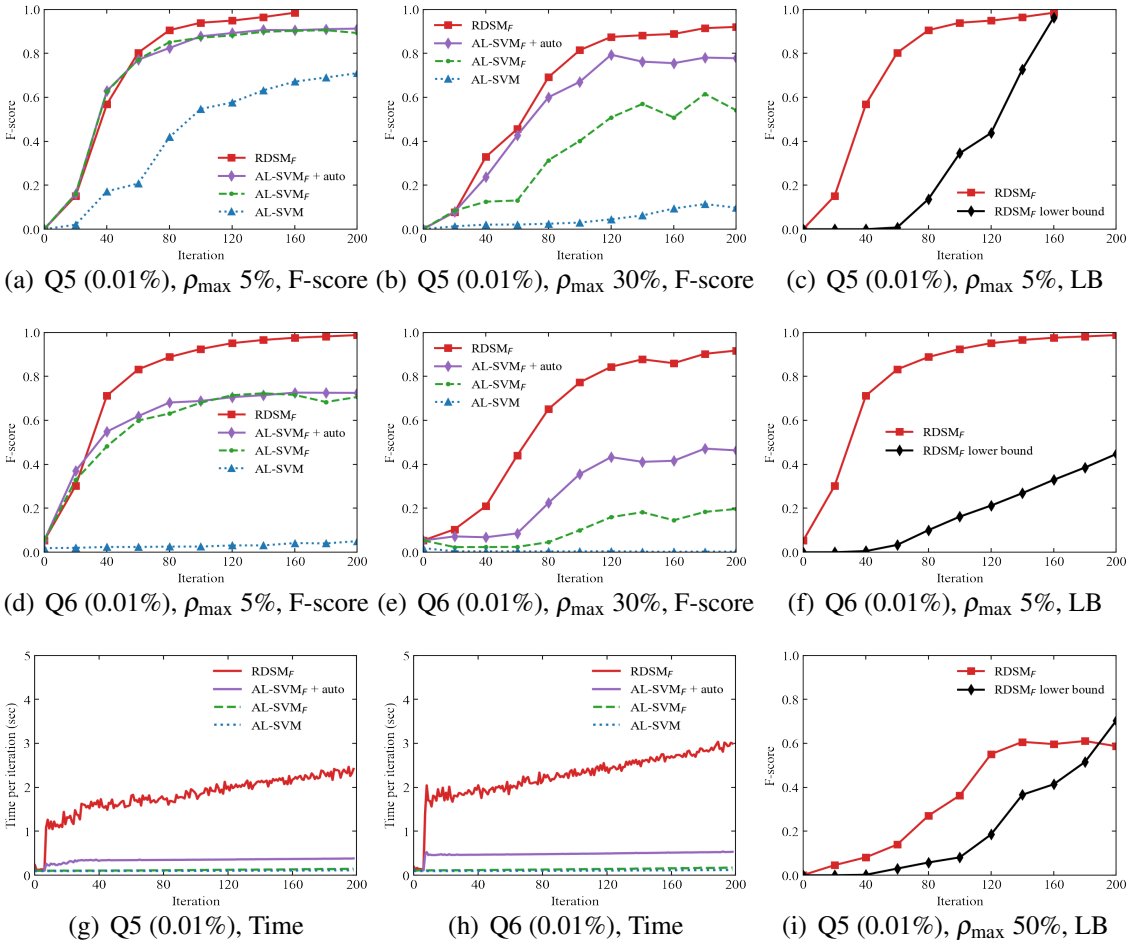


Figure 5.10: Averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM** over 4D-6D queries combining from SDSS Q1-Q4 at iteration 200

(30%) is similar to that under 10% (20%). Thus we present the results in terms of F-score, lower bound (LB), and time measurement for 5% and 30%. 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 lowest selectivity (lower selectivity typically leads to more difficulty in learning), we take the two queries as examples to show the general trends.

As shown in Figure 5.11(a) - 5.11(b) and Figure 5.11(d) - 5.11(e), **RDSM$_F$** outperforms the other algorithms significantly and consistently. 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.

Concerning lower bound, **RDSM$_F$** usually provides a lower bound quite close to F-score when $\rho_{\max}$ is low, as shown in Figure 5.11(f). However, when $\rho_{\max}$ is pretty high, e.g., 50%, as Figure 5.11(i) shows, the lower bound computed from **RDSM$_F$** may exceed the true F-score due to the misconstruction of the data space model. In addition, we observe from Figure 5.11(f) that the Three-Set Metric may not be a tight lower bound of F-score for some



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

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

(g) Q5 (0.01%), Time (h) Q6 (0.01%), Time (i) Q5 (0.01%), $\rho_{\max}$ 50%, LB

Figure 5.11: **RDSM$_F$** for queries from SDSS query templates Q5 and Q6, compared to **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM**

high dimensional queries. We should devise a tighter and label noise-robust lower bound based on the data space model in our future work.

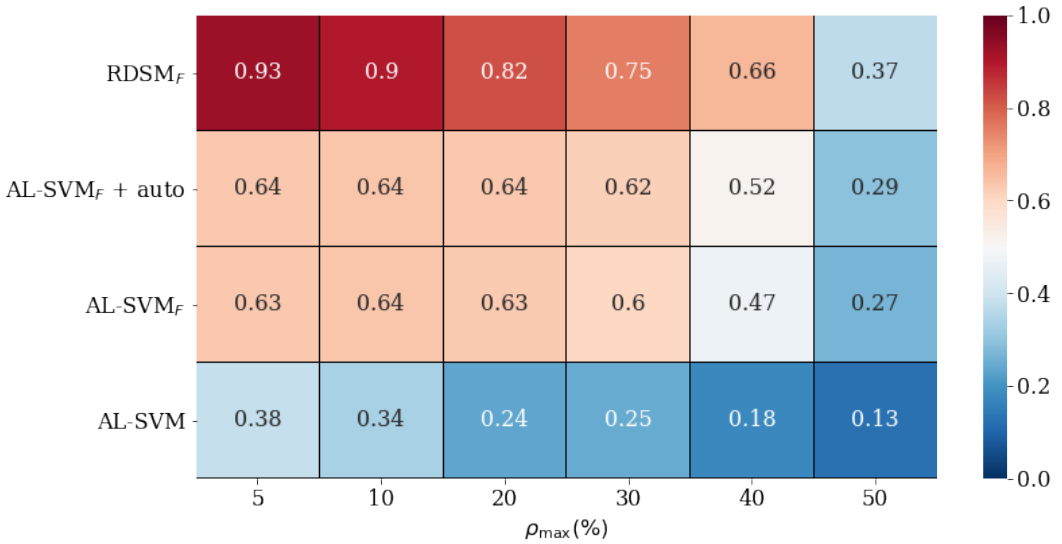
Regarding time measurement, like 2D queries, the time cost per iteration of each algorithm for high dimensional queries remains similar across different levels of label noise. Figure 5.11(g) - 5.11(h) shows that **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM** have time cost per iteration below 0.5 seconds. 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.

Expt 5: RDSM$_F$ for p-DSC Queries

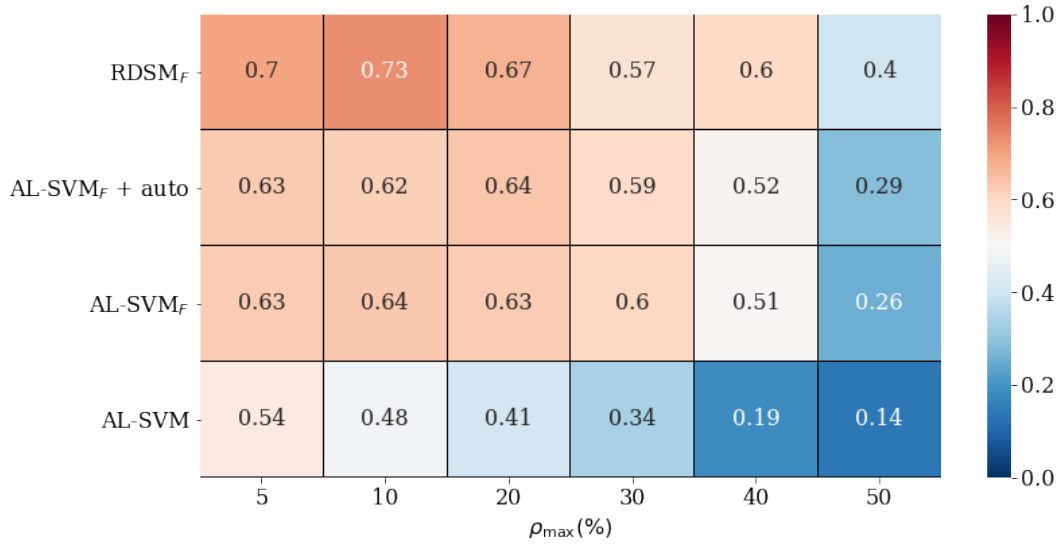
Our **RDSM**$_F$ always brings significant benefits to **DSC** queries, which can be broken down into a collection of sub-queries such that the positive or negative region of each sub-query is a convex object in the corresponding subspace. However, **RDSM**$_F$ is also applicable to some queries beyond **DSC** queries, such as p-**DSC** queries. p-**DSC** queries is a superset of **DSC** queries, where the convex assumption holds only in some subspaces. Figure 5.12(a) and Figure 5.12(b) shows the F-score heat maps for two 4D queries Q7.v1 and Q7.v2 respectively. As described in **Expt 4** in Section 4.3, 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). The F-score heat map of Q7.v1 is similar to Figure 5.10. For Q7.v1, **RDSM**$_F$ always substantially outperforms **AL-SVM**$_F$ **+ auto** and **AL-SVM**$_F$ which further outperform **AL-SVM**. **AL-SVM**$_F$ **+ auto** performs slightly better than **AL-SVM**$_F$. For the 4D p-**DSC** query Q7.v2, although the improvement of accuracy provided by **RDSM**$_F$ is not so significant as previous results of **DSC** queries, **RDSM**$_F$ has better performance than other algorithms under any noise rates except for 30%. Similar to the data space model, **auto** may also be subject to the non-convex pattern of Q7.v2. The accuracy of **AL-SVM**$_F$ **+ auto** is quite close to that of **AL-SVM**$_F$. For $\rho_{\max} = 30\%$ or $\rho_{\max} = 10\%$, **AL-SVM**$_F$ **+ auto** is even defeated by **AL-SVM**$_F$.

5.6.5 Evaluation of Optimization on Interactive Performance

In Section 5.4.1, we propose two approaches to reducing the per-iteration time cost. By utilizing distributed computing, we have managed to cut down the time cost from 20-30 seconds to within 3 seconds. If we further reduce the frequency of rebuilding partitions, the time cost may go down for the iterations that do not require rebuilding partitions. We use **RDSM**$_F$-F1 (**RDSM**-F1) and **RDSM**$_F$-F20 (**RDSM**-F20) to denote rebuilding partitions per iteration and rebuilding partitions every 20 iterations, respectively. As shown in Figure 5.13, the averaged F-scores over all SDSS queries of **RDSM**$_F$-F20 are very close to the corresponding ones of **RDSM**$_F$-F1.



(a) SDSS Q7.*v*1, **DSC** Query



(b) SDSS Q7.*v*2, *p*-**DSC** Query

Figure 5.12: F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM** for SDSS Q7 at iteration 200

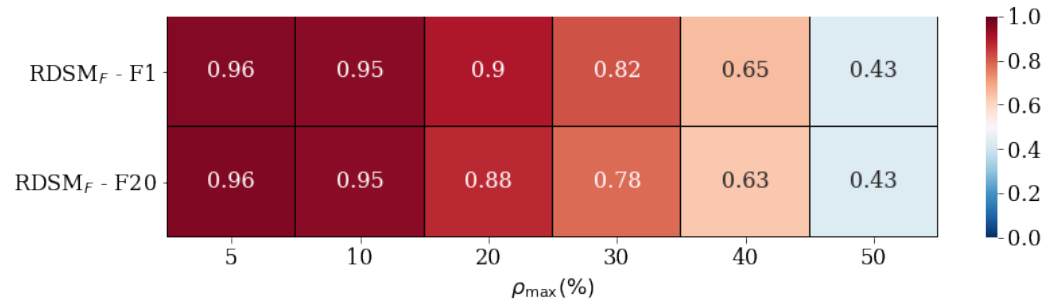
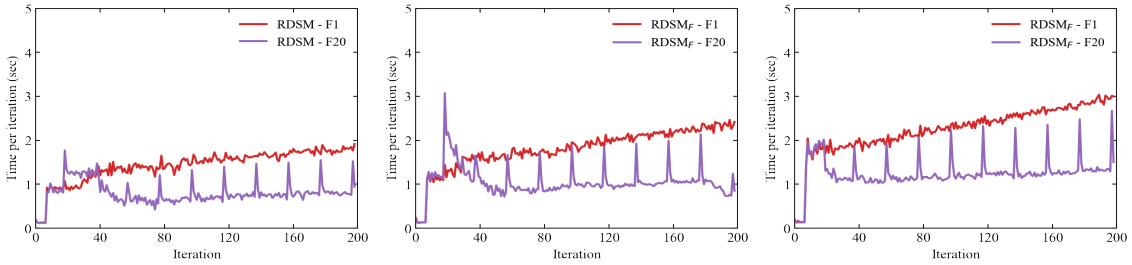


Figure 5.13: Averaged F-scores of **RDSM**$_F$-F1 and **RDSM**$_F$-F20 over all SDSS queries at iteration 200



(a) Q3, Time (b) Q5, Time (c) Q6, Time

Figure 5.14: Optimization on interactive performance

Regarding time measurement, the queries of the same dimensionality have similar time costs. Thus, in the following, we present the results of a 2D query Q3, a 4D query Q5, and a 6D query Q6. As Figure 5.14 shows, for the iterations where rebuilding partitions is not conducted, **RDSM**$_F$-F20 dramatically reduces the per-iteration time, with per-iteration time below 1 second for Q3, around 1 second for Q5, and within 1.5 seconds for Q6. The decrease in time cost is due to the shrinkage of uncertain regions after a certain number of iterations. When the uncertain region is small, Algorithm 2, which *incrementally* updates partitions, runs faster than Algorithm 5 with distributed computing. However, if the uncertain region is super large (e.g. at the beginning of data exploration), running Algorithm 2 is more expensive, as the initial spikes show in both Figure 5.14(a) and Figure 5.14(b).

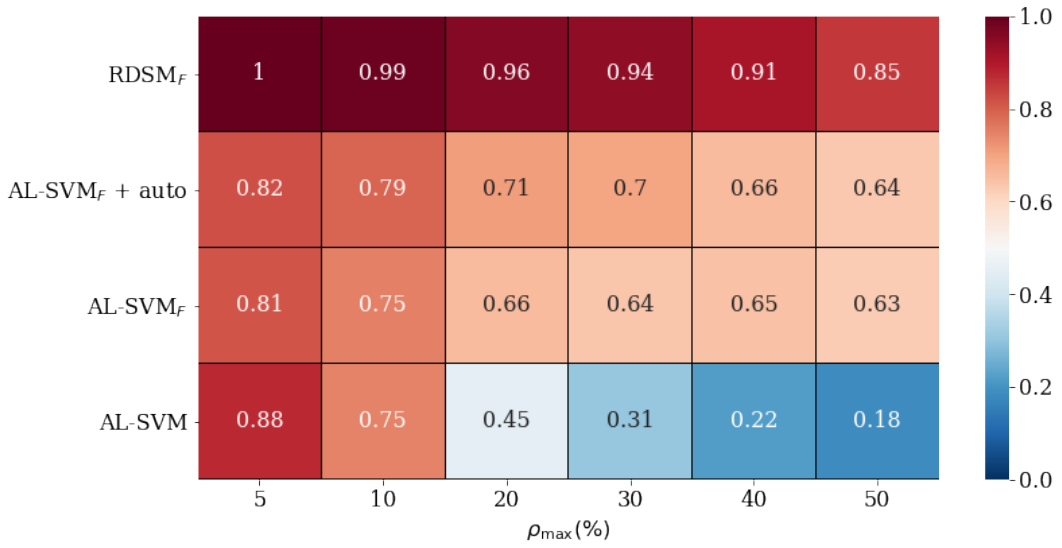


Figure 5.15: Averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over all Car queries at iteration 100

5.6.6 Evaluation on User Study Queries

We now conduct the evaluation for Car queries obtained from our user study. Figure 5.15 shows the averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over 18 Car 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). In other words, **RDSM$_F$**'s F-score is 14%-372% times higher than **AL-SVM**'s F-score. (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 (5%, 10%), **AL-SVM** achieves better performance than **AL-SVM$_F$** while for higher noise rates, **AL-SVM$_F$** significantly outperforms **AL-SVM**.

Since Car queries are more complex than SDSS queries and generally involve more attributes, the accuracy of all algorithms for Car queries is subject to variation. Therefore,

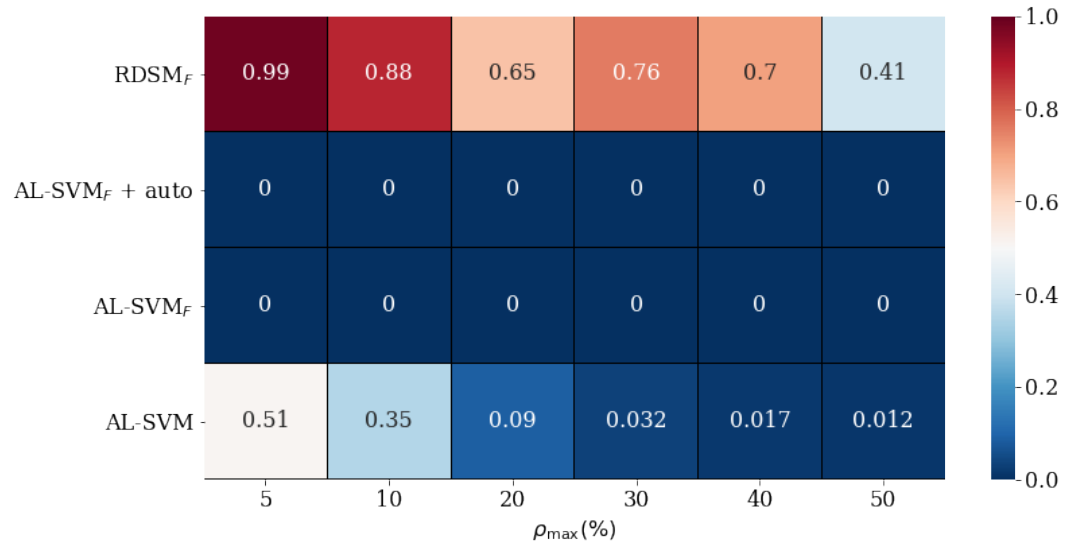


Figure 5.16: Minimum F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over all Car queries at iteration 100

we also provide the minimum and maximum F-scores of the algorithms over all Car queries in Figure 5.16 and Figure 5.17. In the worst cases, as shown in Figure 5.16, the F-scores of **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** are close to or equal to 0, while the worst case of **RDSM$_F$** still achieves high a F-score for lower noise rates. In the best cases, as shown in Figure 5.16, except for **AL-SVM**, all the other three algorithms can reach up to 100% accuracy.

Among all the queries we deal with in this work, Q5 (0.231% selectivity) has the highest dimensionality, including 418 attributes after one-hot encoding of the categorical attributes. In Figure 5.18, we show the trends of the algorithms for Car query Q5. Consistently, we observe that as the noise rate increases, the performance of all algorithms decreases, but **AL-SVM** drops dramatically faster. It is worth noting that when $\rho_{\max}$ rises from 5% to 50%, while **AL-SVM**'s accuracy drops from 95% to 8%, **RDSM$_F$** remains 100% accuracy for $\rho_{\max}$ from 5% to 40% and achieves 97% for $\rho_{\max} = 50\%$. **AL-SVM$_F$ + auto** and **AL-SVM$_F$** have similar performances, and their accuracies lie in between **RDSM$_F$** and **AL-SVM**.

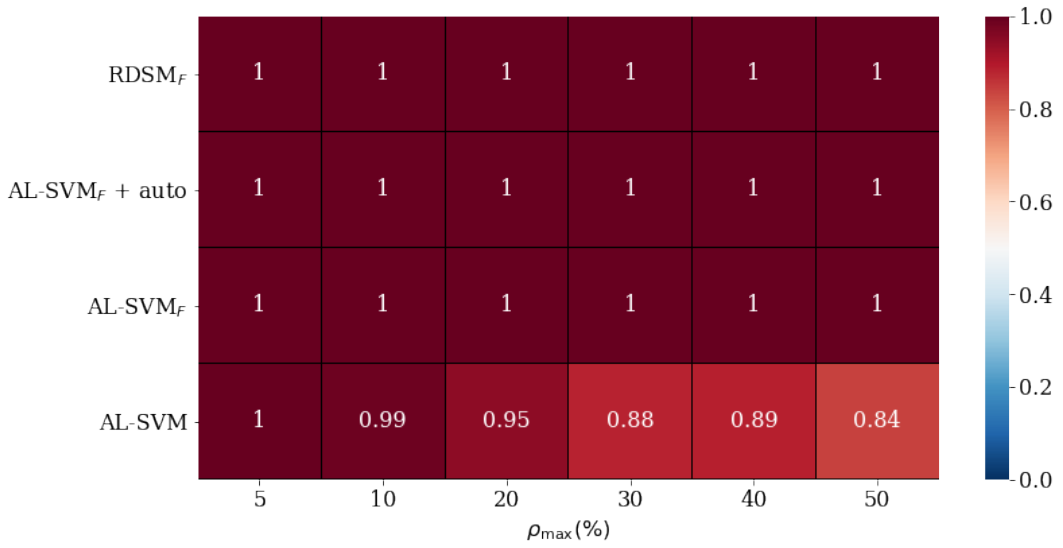
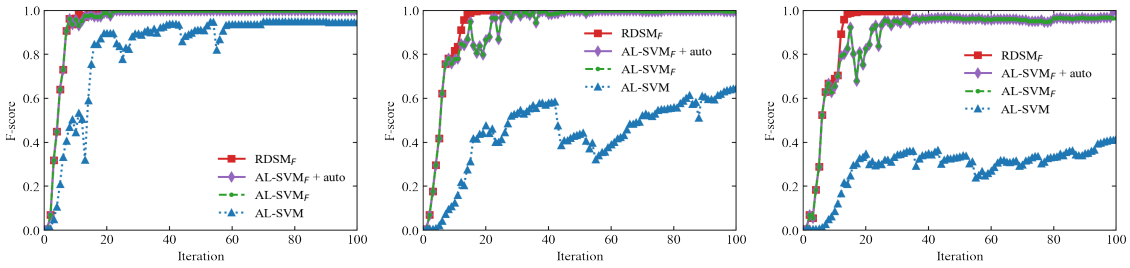
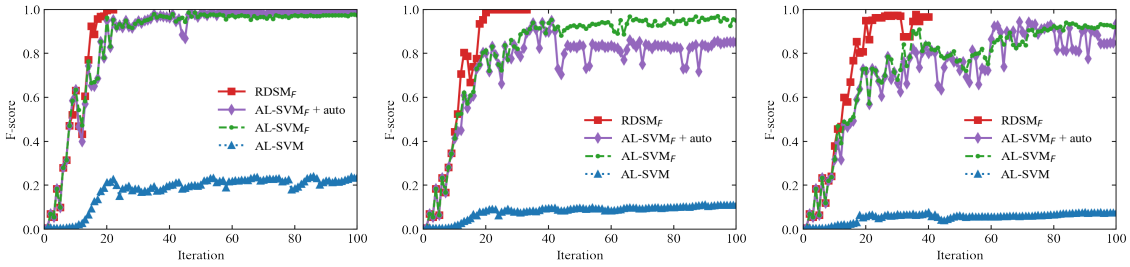


Figure 5.17: Maximum F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over all Car queries at iteration 100



(a) Car Q5, $\rho_{\max}$ 5%, F-score (b) Car Q5, $\rho_{\max}$ 10%, F-score (c) Car Q5, $\rho_{\max}$ 20%, F-score



(d) Car Q5, $\rho_{\max}$ 30%, F-score (e) Car Q5, $\rho_{\max}$ 40%, F-score (f) Car Q5, $\rho_{\max}$ 50%, F-score

Figure 5.18: Comparison of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** for Car Q5

5.7 Summary

In this chapter, we presented a robust algorithm **RDSM**$_F$ for active learning-based interactive exploration in the presence of label noise. By leveraging the subspatial convexity and conjunctive properties of user interest queries, **RDSM**$_F$ involves the following methods to handle the label noise problem: (1) We utilize a kNN-based method to refine the data space model and design an adaptive auto-labeling method to trades off between saving user labeling effort and preventing the data space model from introducing noisy labels. These two methods enable the data space model to be more accurate and more robust. (2) When the data exploration is performed in high dimensions, we apply factorization into modeling and demonstrate that factorization helps alleviate the label noise problem. (3) We propose a set of optimization strategies, including fine-tuning parameters (to improve accuracy and generality) and using distributed computing (to boost time efficiency).

Our proposed methods apply to *bounded instance- and label- dependent noise (BILN)* - a general case of label noise. Experimental results on real-world datasets and user interest queries, in the presence of label noise, show the following results: (1) **RDSM**$_F$ substantially outperforms alternative algorithms in accuracy under all label noise levels while achieving desired interactive performance with per-iteration time within 1-3 seconds. In particular, compared to traditional active learning, **RDSM**$_F$ achieves 0.14x-3.72x higher accuracy for Car queries. Moreover, for SDSS queries, the accuracy of **RDSM**$_F$ is 0.04x-0.3x higher for two-dimensional queries, 0.88x-22.14x higher for high-dimensional queries. (2) With the increase of the noise rate, all algorithms have decreased performance. However, **RDSM**$_F$ is the most robust one, dropping at the slowest pace. (3) **AL-SVM**$_F$ **+ auto** is the version space part of **RDSM**$_F$. For the SDSS dataset, **RDSM**$_F$ outperforms **AL-SVM**$_F$ **+ auto** in accuracy: 0.02x-0.11x higher for two-dimensional queries and 0.15x-1.5x higher for high-dimensional queries. For the car dataset, **RDSM**$_F$ offers 0.22x-0.38x performance gain over **AL-SVM**$_F$ **+ auto**. These results demonstrate the effectiveness of our robust data space model. (4) Thanks to the auto-cleansing method, **AL-SVM**$_F$ **+ auto** usually performs better than traditional active learning. (5) For data exploration performed in high dimensions, active learning with

factorized classifiers usually outperforms traditional active learning. The reason behind this is that factorization helps alleviate the label noise problem.

Chapter 6

Conclusions and Future Work

We present a summary of this thesis in Section 6.1 and discuss some interesting directions for future study in Section 6.2.

6.1 Thesis Summary

Data management tools, which can effectively retrieve high-quality content from massive data, are in high demand in the era of data explosion. With the goal to develop a high-performance interactive data exploration system, we concentrated primarily on the active learning-based explore-by-example framework and proposed novel and data-centric techniques to boost the accuracy of our system and reduce user labeling efforts while achieving desired interactive performance.

We first presented, in Chapter 3, the Dual-Space Model (**DSM**) composed of a data space model and a version space model (e.g., traditional classifiers) as the cornerstone of this thesis. We leveraged the subspatial convexity property of database queries to bear on the design of the data space model. For the active learning-based data exploration, the data space model serves as an active learner playing an important role in both prediction and sample acquisition. In particular, it supports both numerical attributes and categorical attributes. We also proposed some optimization strategies to reduce the per-iteration time to within 1-2

seconds. Experimental results using real-world data show that **DSM** significantly improves the accuracy and expedites the convergence of active learning.

In Chapter 4, for interactive data exploration in high dimensions, we proposed new algorithms to overcome the slow convergence of active learning by leveraging the subspatial convex and conjunctive properties of database queries. More specifically, we break the high-dimensional problem into a set of low-dimensional problems by applying factorization to **DSM**. We theoretically proved that factorized **DSM** (**DSM**$_F$) provides a tighter lower bound of F-score than **DSM**. Thus, **DSM**$_F$ provides a more effective stopping criterion for accuracy-based systems. In addition, we presented some procedures to test the convexity and conjunctivity assumptions and design an online feature selection method to filter irrelevant attributes on the fly. Our results show that our **DSM**$_F$ algorithm substantially outperforms active learning [15, 35], active search [39], and existing explore-by-example systems, Aide [31, 32] and LifeJoin [25], in accuracy and convergence speed, while maintaining the per-iteration time within 1-2 seconds.

Finally, in Chapter 5, we proposed a robust algorithm **RDSM**$_F$ for active learning-based interactive exploration in the presence of label noise. To handle the label noise problem, we leveraged subspatial convexity of database queries, factorization, and advanced data cleansing methods to develop the design of **RDSM**$_F$. Our main contributions are (i) refining the data space model by a kNN-based noise filtering method, (ii) devising an adaptive auto-labeling method to prevent more noisy labels from **DSM**-labeled examples, and (iii) showing the effect of factorization on learning with label noise. It is worth noting that our **RDSM**$_F$ is applicable to a general case of label noise, *bounded instance- and label- dependent noise*. Experimental results on real-world datasets and user interest queries show that our proposed algorithm **RDSM**$_F$ substantially outperforms alternative algorithms in accuracy while achieving desired interactive performance with per-iteration time within 1-3 seconds.

6.2 Future Work

For future study, we consider some interesting directions related to this thesis as follows.

- **Extension for More General User Interest Patterns**

  Our proposed algorithm significantly improves the accuracy and outperforms alternative systems for user interest queries that satisfy the subspatial convexity assumption. For user interest queries in which the subspatial convexity assumption does not hold, we have presented an example without noisy labels in Chapter 4 and with noisy labels in Chapter 5. Experimental results demonstrate that our algorithm still results in a remarkable improvement in performance. However, to further generalize our algorithm to get the maximum benefit from the Dual-Space Model, the extension for more general user interest patterns is needed. For example, if the user interest region is composed of several disjoint subregions, a possible direction of our extension is to (1) explore the entire dataset to identify possible subregions, (2) partition the data space based on the detected subregions such that each partition contains only one subregion, and (3) finally apply our algorithm in each partition respectively. It may also be feasible to utilize more complex geometric patterns rather than convex hulls to partition data space.

- **Probabilistic Models for Data Space Model**

  We have mentioned previously, in Chapter 5, the geometric computing under uncertainty. If the probabilistic existence of labeled examples is known, we can build probabilistic convex hulls and tackle noisy labels intrinsically. We have developed a probabilistic data space model for two-dimension data space and implemented it in Python[1], but how to deal with the degeneracy of the input in high dimensions and how to improve its efficiency remain open problems [4]. In addition, if the probabilistic existence is not provided, it is also challenging to compute it accurately.

- **Learning Factorization Structure from Data**

  We have proposed two approaches to derive the factorization structure: (1) extracting a reasonable factorization structure from existing query traces, (2) obtaining a proper factorization structure with the help of domain experts. When there are no available

[1] https://github.com/enhui-huang/probabilistic-dsm

query traces and the user is not an expert, it is vital to learn the factorization structure from data automatically.

- **Dealing with Missing Values**

  Missing values often occur in real-world datasets. For example, patient datasets are often subject to missing diagnostic tests, and consumer datasets often lack some important values related to buying preferences [76]. Nevertheless, the issue of missing values has not drawn enough attention in supervised learning [47]. Most traditional learning methods, without appropriate pre-processing, cannot deal with incomplete data directly. Their performance can significantly degrade in the presence of missing values, let alone the co-occurrence of data irregularities (e.g., imbalanced classification and missing values) [28]. To the best of our knowledge, there is very little literature on missing values in the context of active learning [62]. Handling missing values for the active learning-based data exploration is a very promising direction in future work.

- **Stopping criteria under Label Noise**

  Since the true user interest is unknown in our problem, we do not have a test set with true labels to estimate the actual accuracy. Our system stops data exploration under either of the following two conditions: (1) the user decides to terminate labeling and (2) the three-set metric is above the desired accuracy threshold. When there are no noisy labels, the three-set metric is proved to be a lower bound of F-score and monotonically increasing. Thus this metric can be treated as a stopping criterion. When it comes to the circumstances with label noise, the monotonicity and lower bound property do not always hold for the three-set metric, more sophisticated and robust stopping criteria are worth exploring in the future.

- **Dimensionality Reduction with Noisy Labels**

  In the face of limited labeled examples and highly imbalanced data distributions, it becomes extremely difficult to identify noisy labels and perform dimensionality reduction simultaneously. Integrating dimensionality reduction methods into learning with label noise in our work needs to be studied in depth.

Bibliography

[1] A. Abouzied, D. Angluin, C. H. Papadimitriou, J. M. Hellerstein, and A. Silberschatz. Learning and verifying quantified boolean queries by example. In *Symposium on Principles of Database Systems (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 *12th International Conference on Document Analysis and Recognition, ICDAR 2013, Washington, DC, USA, August 25-28, 2013*, pages 837–841. IEEE Computer Society, 2013.

[4] P. K. Agarwal, S. Har-Peled, S. Suri, H. Yildiz, and W. Zhang. Convex hulls under uncertainty. *Algorithmica*, 79(2):340–367, 2017.

[5] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In J. V. den Bussche and V. Vianu, editors, *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.

[6] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50. Springer, 2004.

[7] S. H. Bach, B. D. He, A. Ratner, and C. Ré. Learning the structure of generative models without labeled data. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 273–282. PMLR, 2017.

[8] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, Dec. 1996.

[9] S. Basu Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 13–22. ACM, 2008.

[10] R. Batuwita and V. Palade. FSVM-CIL: fuzzy support vector machines for class imbalance learning. *IEEE Trans. Fuzzy Syst.*, 18(3):558–571, 2010.

[11] 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.

[12] 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.

[13] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[14] A. Blum and T. M. Mitchell. Combining labeled and unlabeled data with co-training. In P. L. Bartlett and Y. Mansour, editors, *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT 1998, Madison, Wisconsin, USA, July 24-26, 1998*, pages 92–100. ACM, 1998.

[15] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619, 2005.

[16] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, pages 144–152. ACM, 1992.

[17] M. Bouguelia, S. Nowaczyk, K. C. Santosh, and A. Verikas. Agreeing to disagree: active learning with noisy labels without crowdsourcing. *Int. J. Mach. Learn. Cybern.*, 9(8):1307–1319, 2018.

[18] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[19] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *J. Artif. Intell. Res.*, 11:131–167, 1999.

[20] C. Campbell, N. Cristianini, and A. J. Smola. Query learning with large margin classifiers. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 111–118. Morgan Kaufmann, 2000.

[21] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, 2011.

[22] P. Chen, B. Liao, G. Chen, and S. Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1062–1070. PMLR, 2019.

[23] J. Cheng, T. Liu, K. Ramamohanarao, and D. Tao. Learning with bounded instance and label-dependent label noise. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1789–1799. PMLR, 2020.

[24] A. Cheung and A. Solar-Lezama. Computer-assisted query formulation. *Foundations and Trends® in Programming Languages*, 3(1):1–94, June 2016.

[25] A. Cheung, A. Solar-Lezama, and S. Madden. Using program synthesis for social recommendations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1732–1736, New York, NY, USA, 2012. ACM.

[26] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

[27] S. Cuendet, D. Hakkani-Tür, and E. Shriberg. Automatic labeling inconsistencies detection and correction for sentence unit segmentation in conversational speech. In A. Popescu-Belis, S. Renals, and H. Bourlard, editors, *Machine Learning for Multimodal Interaction , 4th International Workshop, MLMI 2007, Brno, Czech Republic, June 28-30, 2007, Revised Selected Papers*, volume 4892 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2007.

[28] S. Das, S. Datta, and B. B. Chaudhuri. Handling data irregularities in classification: Foundations, trends, and future challenges. *Pattern Recognit.*, 81:674–693, 2018.

[29] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM*, pages 3–12, 2008.

[30] Y. Diao, K. Dimitriadou, Z. Li, W. Liu, O. Papaemmanouil, K. Peng, and L. Peng. AIDE: an automatic user navigation system for interactive data exploration. *PVLDB*, 8(12):1964–1967, 2015.

[31] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 517–528. ACM, 2014.

[32] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2842–2856, 2016.

[33] J. Du and Z. Cai. Modelling class noise with symmetric and asymmetric distributions. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2589–2595. AAAI Press, 2015.

[34] R. El-Yaniv and Y. Wiener. Active learning via perfect selective classification. *Journal of Machine Learning Research*, 13(Feb):255–279, 2012.

[35] S. Ertekin, J. Huang, L. Bottou, and L. Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136. ACM, 2007.

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

[37] 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.

[38] D. Gamberger, N. Lavrac, and S. Dzeroski. Noise elimination in inductive concept learning: A case study in medical diagnosois. In S. Arikawa and A. Sharma, editors, *Algorithmic Learning Theory, 7th International Workshop, ALT '96, Sydney, Australia, October 23-25, 1996, Proceedings*, volume 1160 of *Lecture Notes in Computer Science*, pages 199–212. Springer, 1996.

[39] R. Garnett, Y. Krishnamurthy, X. Xiong, J. G. Schneider, and R. P. Mann. Bayesian optimal active search and surveying. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, pages 843–850, 2012.

[40] B. Grünbaum. Convex polytopes. In *Convex Polytopes*. Springer-Verlag New York, 2 edition, 2003.

[41] S. Hanneke. Refined error bounds for several learning algorithms. *The Journal of Machine Learning Research*, 17(1):4667–4721, 2016.

[42] S. Hanneke et al. Rates of convergence in active learning. *The Annals of Statistics*, 39(1):333–361, 2011.

[43] S. Hanneke et al. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3):131–309, 2014.

[44] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009.

[45] E. Huang, L. Peng, L. D. Palma, A. Abdelkafi, A. Liu, and Y. Diao. Optimization for active learning-based interactive database exploration. *Proc. VLDB Endow.*, 12(1):71–84, 2018.

[46] P. G. Ipeirotis, F. J. Provost, V. S. Sheng, and J. Wang. Repeated labeling using multiple noisy labelers. *Data Min. Knowl. Discov.*, 28(2):402–441, 2014.

[47] N. Ipsen, P.-A. Mattei, and J. Frellsen. How to deal with missing data in supervised deep learning? In *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*, 2020.

[48] 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.

[49] L. Jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2309–2318. PMLR, 2018.

[50] M. Kahng, S. B. Navathe, J. T. Stasko, and D. H. P. Chau. Interactive browsing and navigation in relational databases. *PVLDB*, 9(12):1017–1028, 2016.

[51] A. Kalinin, U. Cetintemel, and S. Zdonik. Interactive data exploration using semantic windows. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 505–516. ACM, 2014.

[52] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *2014 IEEE 30th International Conference on Data Engineering*, pages 472–483. IEEE, 2014.

[53] S. R. Lay. *Convex Sets and Their Applications*. Dover Publications, 2007.

[54] 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.

[55] C. Lin and S. Wang. Training algorithms for fuzzy support vector machines with noisy data. *Pattern Recognit. Lett.*, 25(14):1647–1656, 2004.

[56] C. H. Lin, Mausam, and D. S. Weld. Re-active learning: Active learning with relabeling. In D. Schuurmans and M. P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1845–1852. AAAI Press, 2016.

[57] M. Lissandrini, D. Mottin, T. Palpanas, and Y. Velegrakis. *Data Exploration Using Example-Based Methods*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.

[58] 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.

[59] Y. Ma, R. Garnett, and J. G. Schneider. Σ-optimality for active learning on gaussian random fields. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2751–2759, 2013.

[60] A. K. Menon, B. Van Rooyen, and N. Natarajan. Learning from binary labels with instance-dependent noise. *Machine Learning*, 107(8):1561–1595, 2018.

[61] A. L. B. Miranda, L. P. F. Garcia, A. C. P. L. F. de Carvalho, and A. C. Lorena. Use of classification algorithms in noise detection and elimination. In E. Corchado, X. Wu, E. Oja, Á. Herrero, and B. Baruque, editors, *Hybrid Artificial Intelligence Systems, 4th International Conference, HAIS 2009, Salamanca, Spain, June 10-12, 2009. Proceedings*, volume 5572 of *Lecture Notes in Computer Science*, pages 417–424. Springer, 2009.

[62] S. Moon, C. McCarter, and Y. Kuo. Active learning with partially featured data. In C. Chung, A. Z. Broder, K. Shim, and T. Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*, pages 1143–1148. ACM, 2014.

[63] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: Give me an example of what you need. *Proceedings of the VLDB Endowment*, 7(5):365–376, 2014.

[64] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari. Cost-sensitive learning with noisy labels. *J. Mach. Learn. Res.*, 18:155:1–155:33, 2017.

[65] R. Neamtu, R. Ahsan, C. Lovering, C. Nguyen, E. A. Rundensteiner, and G. N. Sárközy. Interactive time series analytics powered by ONEX. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 1595–1598, 2017.

[66] C. G. Northcutt, L. Jiang, and I. L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research (JAIR)*, 70:1373–1411, 2021.

[67] C. G. Northcutt, T. Wu, and I. L. Chuang. Learning with confident examples: Rank pruning for robust classification with noisy labels. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'17. AUAI Press, 2017.

[68] G. Özsoyoglu and H. Wang. Example-Based Graphical Database Query Languages. *Computer*, 26(5):25–38, 1993.

[69] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[70] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: A loss correction approach. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2233–2241. IEEE Computer Society, 2017.

[71] L. Peng, E. Huang, Y. Xing, A. Liu, and Y. Diao. Uncertainty sampling and optimization for interactive database exploration. *UMass Technical Report*, 2017.

[72] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282, 2017.

[73] A. J. Ratner, C. D. Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3567–3575, 2016.

[74] S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM)*, pages 13–22, 2008.

[75] S. B. Roy, H. Wang, U. Nambiar, G. Das, and M. Mohania. Dynacet: Building dynamic faceted search systems over databases. In *International Conference on Data Engineering (ICDE)*, pages 1463–1466, 2009.

[76] M. Saar-Tsechansky and F. J. Provost. Handling missing values when applying classification models. *J. Mach. Learn. Res.*, 8:1623–1657, 2007.

[77] C. Scott, G. Blanchard, and G. Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In S. Shalev-Shwartz and I. Steinwart, editors, *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, volume 30 of *JMLR Workshop and Conference Proceedings*, pages 489–511. JMLR.org, 2013.

[78] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[79] Y. Shen, K. Chakrabarti, S. Chaudhuri, B. Ding, and L. Novik. Discovering queries based on example tuples. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 493–504, New York, NY, USA, 2014. ACM.

[80] J. Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014.

[81] Sloan digital sky survey: Dr8 sample sql queries. http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp.

[82] J. Sun, F. Zhao, C. Wang, and S. Chen. Identifying and correcting mislabeled training instances. In *Future Generation Communication and Networking, FGCN 2007, Ramada Plaza Jeju, Jeju-Island, Korea, December 6-8, 2007, Proceedings*, pages 244–250. IEEE Computer Society, 2007.

[83] S. Suri, K. Verbeek, and H. Yildiz. On the most likely convex hull of uncertain points. In H. L. Bodlaender and G. F. Italiano, editors, *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 791–802. Springer, 2013.

[84] A. S. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, D. Slutz, and R. J. Brunner. Designing and mining multi-terabyte astronomy archives: the sloan digital sky survey. *ACM SIGMOD Record*, 29(2):451–462, 2000.

[85] F. Tang. Bidirectional active learning with gold-instance-based human training. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5989–5996. ijcai.org, 2019.

[86] Y. Tang, Y. Zhang, N. V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. *IEEE Trans. Syst. Man Cybern. Part B*, 39(1):281–288, 2009.

[87] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[88] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 535–548, New York, NY, USA, 2009. ACM.

[89] B. van Rooyen, A. K. Menon, and R. C. Williamson. Learning with symmetric label noise: The importance of being unhinged. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 10–18, 2015.

[90] H. P. Vanchinathan, A. Marfurt, C. Robelin, D. Kossmann, and A. Krause. Discovering valuable items from massive data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1195–1204, 2015.

[91] P. Varma, B. D. He, P. Bajaj, N. Khandwala, I. Banerjee, D. L. Rubin, and C. Ré. Inferring generative model structure with static analysis. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 240–250, 2017.

[92] P. Varma and C. Ré. Snuba: Automating weak supervision to label training data. *Proc. VLDB Endow.*, 12(3):223–236, 2018.

[93] B. C. Wallace and I. J. Dahabreh. Class probability estimates are unreliable for imbalanced data (and how to fix them). In M. J. Zaki, A. Siebes, J. X. Yu, B. Goethals, G. I. Webb, and X. Wu, editors, *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pages 695–704. IEEE Computer Society, 2012.

[94] D. R. Wilson and T. R. Martinez. Instance pruning techniques. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*, pages 403–411. Morgan Kaufmann, 1997.

[95] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Mach. Learn.*, 38(3):257–286, 2000.

[96] G. Wu and E. Y. Chang. KBA: kernel boundary alignment considering imbalanced data distribution. *IEEE Trans. Knowl. Data Eng.*, 17(6):786–795, 2005.

[97] H. Yu, X. Yang, S. Zheng, and C. Sun. Active learning from imbalanced data: A solution of online weighted extreme learning machine. *IEEE Trans. Neural Networks Learn. Syst.*, 30(4):1088–1103, 2019.

[98] J. Zhang, X. Wu, and V. S. Sheng. Active learning with imbalanced multiple noisy labeling. *IEEE Trans. Cybern.*, 45(5):1081–1093, 2015.

[99] W. Zhang. *Geometric computing over uncertain data*. PhD thesis, Ph. D. Thesis, Department of Computer Science, Duke University, 2015.

[100] 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.

[101] Z. Zhao, L. De Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 527–540, New York, NY, USA, 2017. ACM.

Appendix A

User Interest Queries

Query	Predicate
1	class = 'minivan' **AND** price_msrp $\leq$ 30000 **AND** length $>$ 5 **AND** length * width $>$ 10.1 **AND** fuel_type = 'regular unleaded' **AND** transmission ! $=$ '8-speed shiftable automatic' **AND** transmission ! $=$ '9-speed shiftable automatic'
2	price_msrp $\leq$ 22132 **AND** basic_year $\geq$ 5 **AND** drivetrain_year $\geq$ 10 **AND** horsepower $>$ 156 **AND** body_type != 'suv' **AND** transmission = '6-speed shiftable automatic' **AND** year $\geq$ 2016 **AND** fuel_tank_capacity $\geq$ 65
3	year $\geq$ 2016 **AND** length * height * width $\geq$ 15.0 **AND** basic_year $\geq$ 4 **AND** class = 'full-size car' **AND** price_msrp $<$ 100000 **AND** engine_type = 'gas'
4	body_type = 'truck' **AND** height $\geq$ 1.9 **AND** torque $\geq$ 3800 **AND** price_msrp $\leq$ 30000 **AND** year = 2017 **AND** base_engine_size $\geq$ 5
5	class = 'full-size van' **AND** body_type = 'van' **AND** engine_type = 'gas' **AND** model = 'nv cargo' **AND** price_msrp $<$ 27000 **AND** horsepower $<$ 300
6	height $<$ 1.51 **AND** drivetrain_year $\geq$ 10 **AND** transmission ! $=$ '6-speed manual' **AND** transmission ! $=$ '7-speed manual' **AND** class = 'subcompact car'

7	class = 'mid-size car' **AND** transmission = '6-speed shiftable automatic' **AND** drivetrain_year $>$ 5 **AND** price_msrp $<$ 29000 **AND** basic_km $\geq$ 80467 **AND** body_type != 'suv'
8	length $\geq$ 6 **AND** body_type != 'sedan' **AND** fuel_type != 'premium unleaded (recommended)' **AND** drive_type != 'front wheel drive' **AND** fuel_type != 'premium unleaded (required)' **AND** basic_year $\geq$ 4 **AND** drivetrain_year $\geq$ 5 **AND** price_msrp $<$ 32000 **AND** height $>$ 2.5
9	(body_type = 'truck' **OR** body_type = 'van') **AND** price_msrp $<$ 30000 **AND** height $\geq$ 2.5 **AND** length $>$ 6
10	body_type = 'truck' **AND** horsepower $>$ 350 **AND** drive_type = 'four wheel drive' **AND** engine_type != 'diesel' **AND** fuel_tank_capacity $>$ 100 **AND** year $\geq$ 2017 **AND** suspension != 'stabilizer bar stabilizer bar' **AND** price_msrp $<$ 35000
11	(body_type = 'suv' **OR** body_type = 'truck') **AND** (drive_type = 'all wheel drive' **OR** drive_type = 'four wheel drive') **AND** height $>$ 1.8 **AND** price_msrp $<$ 33000 **AND** length $>$ 5.9 **AND** suspension like '%independent%'
12	price_msrp $<$ 25000 **AND** horsepower $>$ 200 **AND** year = 2017 **AND** length $>$ 5.5
13	price_msrp $<$ 23000 **AND** basic_year $\geq$ 5 **AND** drivetrain_year $\geq$ 10 **AND** basic_km $\geq$ 80000 **AND** drivetrain_km $\geq$ 100000 **AND** engine_type **IN** ('gas', 'hybrid') **AND** body_type **IN** ('sedan', 'hatchback') **AND** drive_type = 'front wheel drive' **AND** fuel_tank_capacity $\geq$ 55 **AND** year $\geq$ 2017
14	price_msrp $<$ 13000 **AND** drive_type = 'front wheel drive' **AND** transmission **LIKE** '%manual%' **AND** length $<$ 4.5 **AND** horsepower $<$ 110
15	transmission **LIKE** '%automatic%' **AND** price_msrp $<$ 25000 **AND** class **NOT LIKE** '%pickup%' **AND** class **NOT LIKE** '%suv%' **AND** basic_year $\geq$ 4 **AND** year $\geq$ 2017 **AND** height $\leq$ 1.62

16	price_msrp $<$ 26000 **AND** body_type **IN** (‘van’, ‘truck’) **AND** height $<$ 2.5 **AND** height $>$ 2 **AND** basic_year $>$ 3
17	horsepower $>$ 150 **AND** year = 2017 **AND** make **IN** (‘hyundai’, ‘honda’) **AND** length $<$ 4.5 **AND** engine_type **IN** (‘gas’, ‘diesel’) **AND** price_msrp $<$ 20000
18	price_msrp $<$ 30000 **AND** body_type **IN** (‘sedan’, ‘suv’) **AND** engine_type = ‘hybrid’ **AND** year = 2017 **AND** basic_year $\geq$ 5 **AND** drivetrain_year $\geq$ 10

Table A.1: True Queries obtained from the Cars User Study

Appendix B

Additional Plots for Label Noise Experiments

Evaluation results of our proposed algorithm under Random Classification Noise (RCN) are listed as follows:

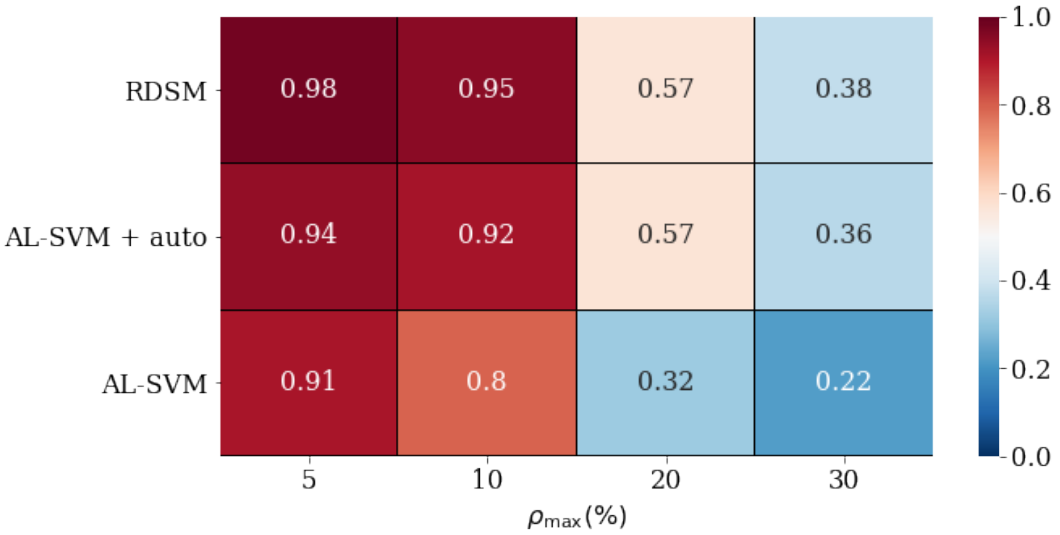


Figure B.1: RCN: Averaged F-scores of **RDSM**, **AL-SVM + auto** and **AL-SVM** over SDSS 2D queries at iteration 200

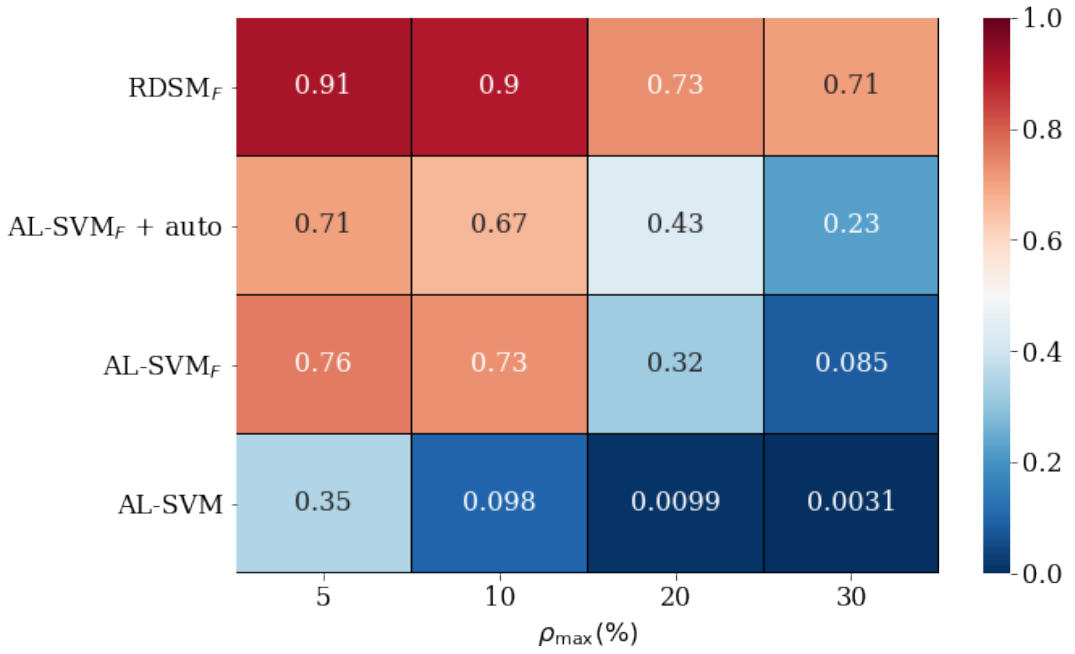
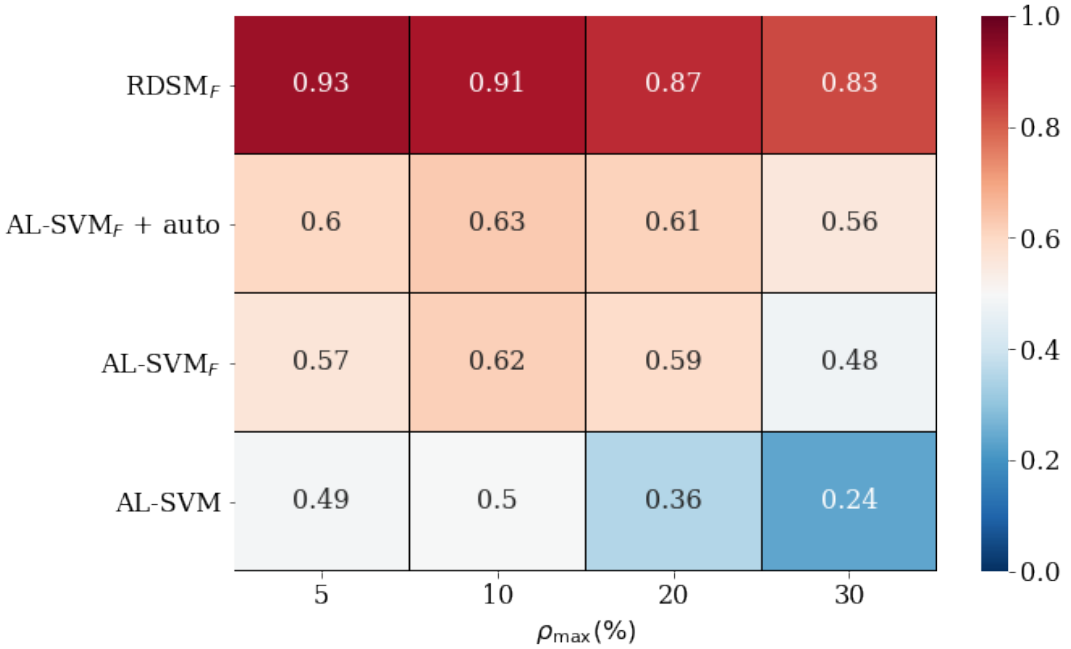
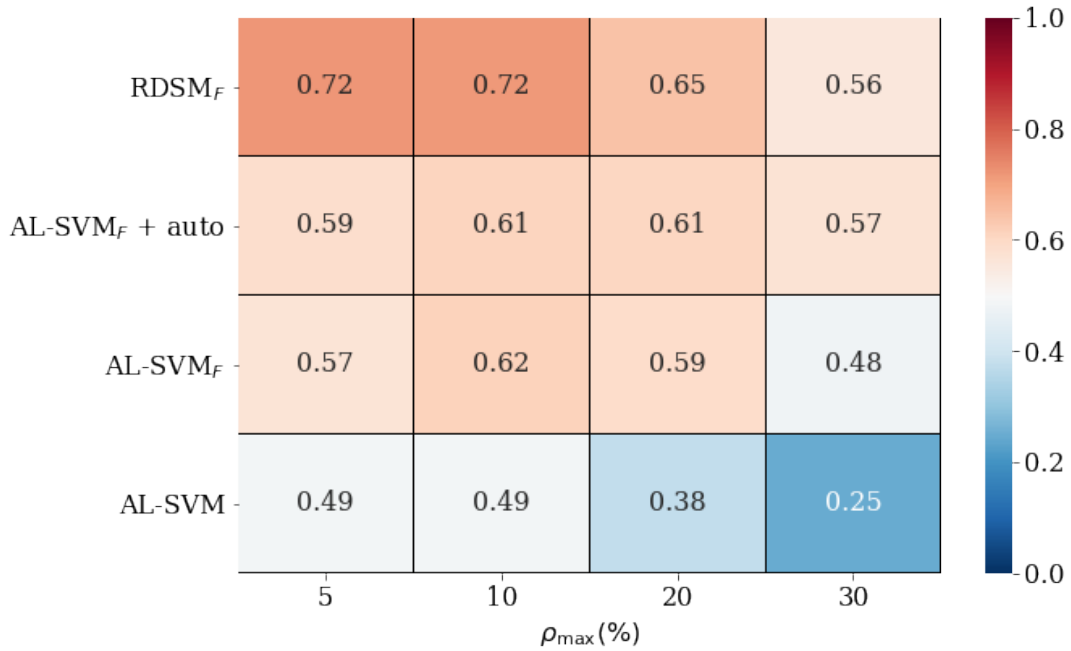


Figure B.2: RCN: Averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM** over 4D-6D queries combining from SDSS Q1-Q4 at iteration 200



(a) SDSS Q7.*v*1, **DSC** Query



(b) SDSS Q7.*v*2, *p*-**DSC** Query

Figure B.3: RCN: F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$** and **AL-SVM** for SDSS Q7 at iteration 200

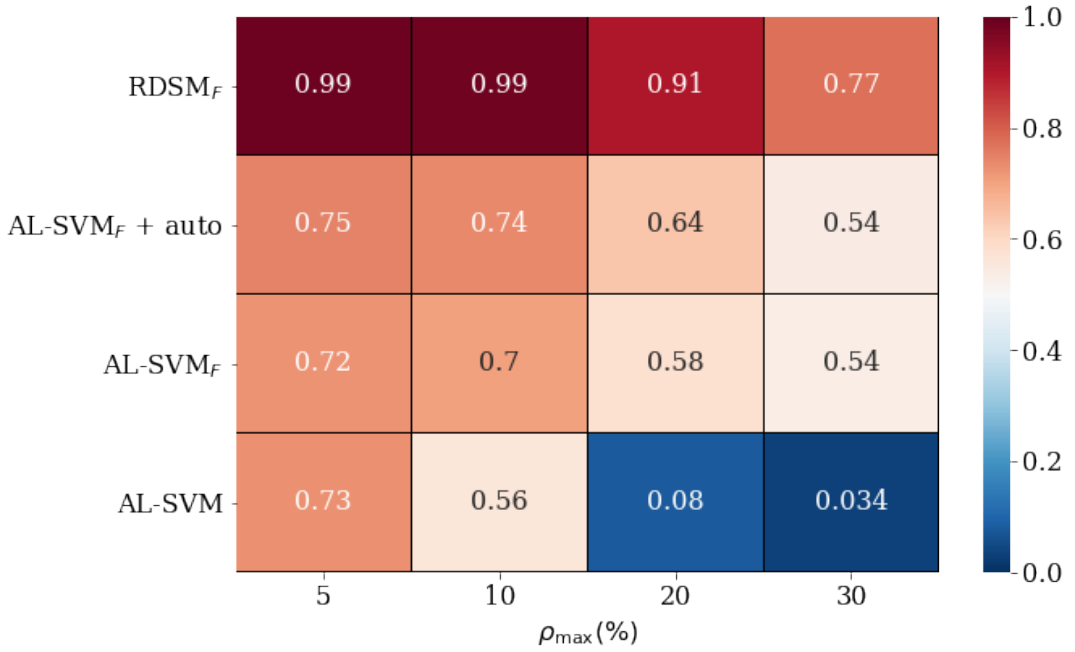


Figure B.4: RCN: Averaged F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over all Car queries at iteration 100

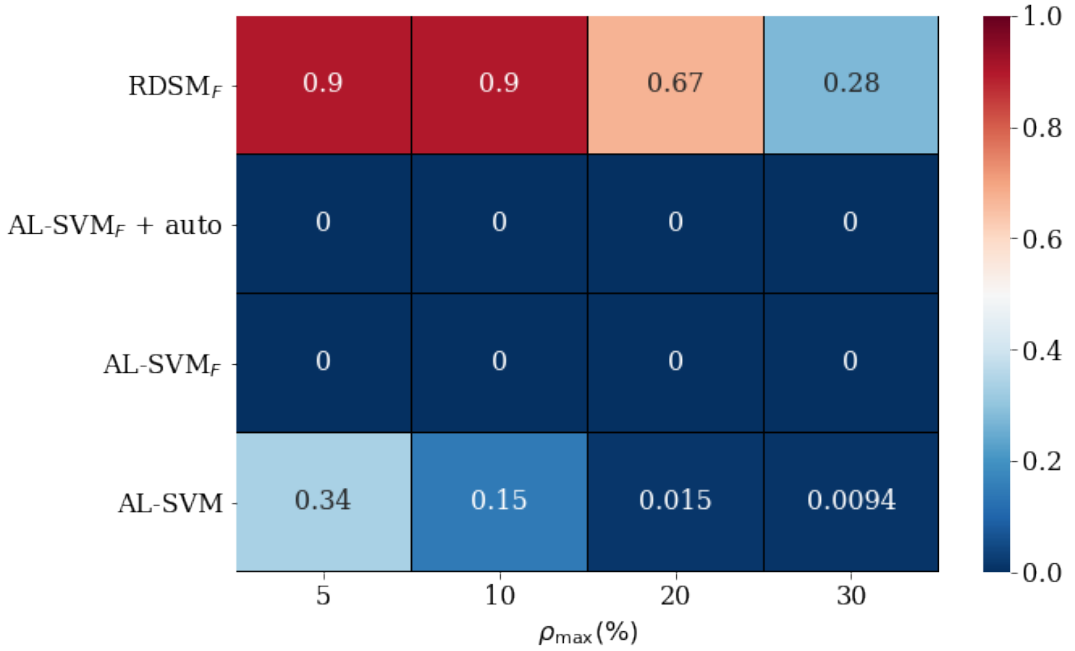


Figure B.5: RCN: Minimum F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over all Car queries at iteration 100

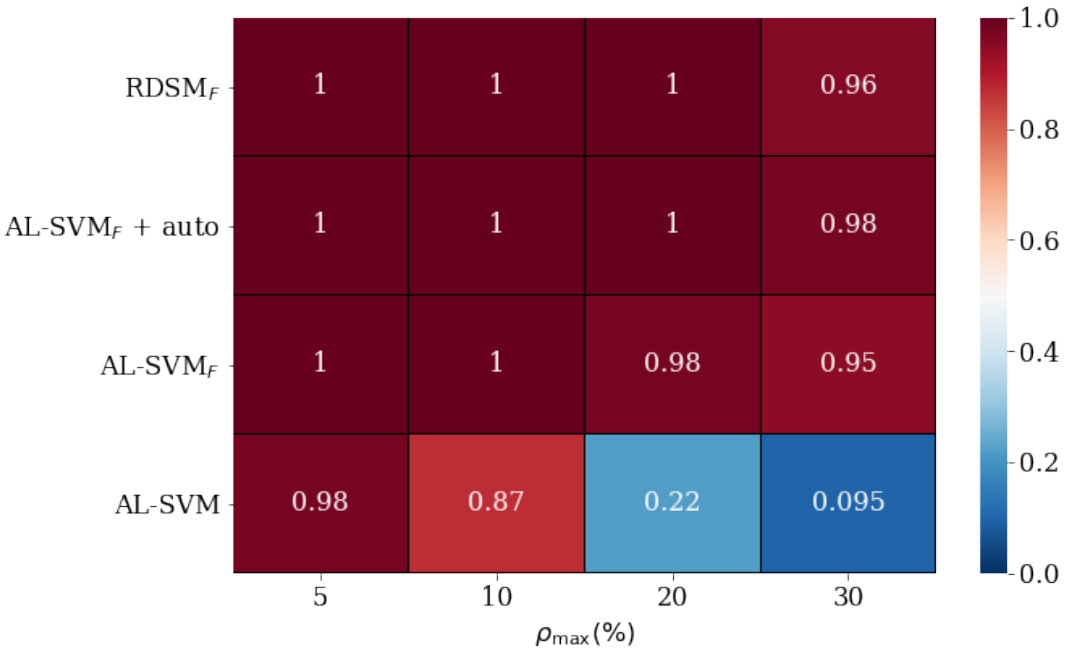


Figure B.6: RCN: Maximum F-scores of **RDSM$_F$**, **AL-SVM$_F$ + auto**, **AL-SVM$_F$**, and **AL-SVM** over all Car queries at iteration 100

ECOLE DOCTORALE

Titre : Méthodes d'apprentissage actif pour l'exploration interactive sur les grandes bases de données

Mots clés : apprentissage actif, exploration de données interactive, modèles de classification, exploration de données de grande dimension, bruit des étiquettes

Résumé : Face à un écart grandissant entre la croissance rapide des données et la capacité humaine limitée à comprendre les données, les outils d'analyse de données sont en forte demande dans de nombreuses applications à travers un large éventail de domaines. En particulier, pour les systèmes interactifs d'exploration de données, un cadre « explorer par l'exemple », qui vise à aider un utilisateur humain à effectuer une exploration de données très efficace tout en minimisant son effort, devient de plus en plus populaire. Cependant, ces systèmes de pointe nécessitent toujours un grand nombre d'exemples étiquetés pour obtenir la précision souhaitée et ne peuvent pas gérer les étiquettes bruyantes. Pour résoudre à la fois le problème de la convergence lente et le problème du bruit des étiquettes, dans cette thèse, nous plaçons le problème d'« explorer par l'exemple » dans un cadre d'apprentissage actif fondé sur des principes et apportons les caractéristiques des classes importantes de l'intérêt de l'utilisateur, pour contribuer à la conception de nouveaux algorithmes et à l'optimisation pour l'exploration de données basée sur l'apprentissage actif.

Alors que des travaux récents ont proposé un modèle d'espace de données basé sur les polytopes pour filtrer les exemples renvoyés par un apprenant actif traditionnel et pour calculer un critère d'arrêt basé sur la précision pour l'apprentissage actif, notre première contribution vise à combiner le modèle basé sur les polytopes et l'apprenant actif traditionnel en un nouveau modèle à double espace (**DSM**), qui offre conjointement les fonctionnalités de prédiction et d'acquisition d'échantillons et ainsi l'accélération de la convergence des modèles. D'ailleurs, nous introduisons un ensemble de techniques pour améliorer les performances interactives de telle sorte que le temps par itération est maintenu entre 1 à 2 secondes. Les résultats de l'évaluation utilisant des jeux de données du monde réel et des patterns d'intérêt des utilisateurs montrent que la précision de notre système est en moyenne 26% supérieure à celle des systèmes d'explorer par l'exemple de pointe.

Notre deuxième contribution consiste à surmonter le problème de la convergence lente lorsque l'exploration des données est effectuée en haute dimension. Nous factorisons l'espace de grande dimension en un ensemble d'espaces de faibles dimensions, construisons un **DSM** dans chaque sous-espace et les combinons pour former un **DSM** factorisé (**DSM**$_F$). Nous prouvons en outre que **DSM**$_F$ atteint un meilleur critère d'arrêt basé sur la précision que **DSM**. Dans le cas où l'utilisateur veut commencer l'exploration avec plus d'attributs que ceux requis dans le modèle final, nous proposons une méthode de sélection de caractéristique en ligne qui donne de manière adaptative les k attributs les plus pertinents. Les résultats expérimentaux montrent que notre système surpasse les systèmes de pointe en termes de précision (entre x19 et x64) et de vitesse de convergence tout en maintenant le temps par itération entre 1 à 2 secondes et notre méthode de sélection de caractéristique en ligne améliore la précision de près de 0 (sans sélection) jusqu'à plus de 80%.

Enfin, nous abordons le problème du bruit lorsque les étiquettes fournies par l'utilisateur sont potentiellement erronées. Nous renforçons la robustesse de notre système en intégrant des méthodes avancées de nettoyage de données et un raffinement du modèle basé sur les polytopes dans **DSM**$_F$. L'algorithme obtenu, appelé le modèle à double espace robuste (**RDSM**$_F$), est comparé à l'apprentissage actif traditionnel à des fins d'évaluation, car les systèmes d'explorer par l'exemple de pointe sont incapables de gérer les étiquettes bruyantes. Les résultats expérimentaux montrent que notre algorithme améliore considérablement l'apprentissage actif traditionnel en termes de précision (jusqu'à x22) tout en atteignant une efficacité raisonnable pour l'exploration interactive des données (1 à 3 secondes par itération).

Title : Active Learning Methods for Interactive Exploration on Large Databases

Keywords : active learning, interactive data exploration, classification models, high-dimensional exploration, label noise

Abstract : Faced with an increasing gap between fast growth of data and limited human ability to comprehend data, data analytics tools are now in high demand in many applications across a broad set of domains. In particular, for interactive data exploration systems, an "explore-by-example" framework, which aims to assist the user in performing highly effective data exploration while minimizing the human effort, is becoming increasingly popular. However, the state-of-the-art explore-by-example systems still require a large number of labeled examples to achieve the desired accuracy and cannot handle noisy labels. To address both the slow convergence problem and the label noise problem, in this thesis, we cast the explore-by-example problem in a principled "active learning" framework, and bring the properties of important classes of the user interest to bear on the design of new algorithms and optimizations for active learning-based data exploration.

While recent work proposed a polytope-based data space model to filter examples returned by a traditional active learner and to compute an accuracy-based stopping criterion for active learning, our first contribution is to combine the polytope-based model and a traditional active learner into a new Dual-Space Model (**DSM**), jointly offering the prediction and sample acquisition functionalities and thereby expediting model convergence. We also provide a set of techniques to improve the interactive performance such that the per-iteration time is maintained within 1 to 2 seconds. Evaluation results using real-world datasets and user interest patterns show that the accuracy of our system is on average 26% higher than the state-of-the-art explore-by-example systems.

Our second contribution is to overcome the slow convergence problem when data exploration is performed in high dimensions. We factorize the high-dimensional space into a set of low-dimensional spaces, build a **DSM** in each subspace and combine them to be a factorized **DSM** (**DSM**$_F$). We further prove that **DSM**$_F$ achieves a better accuracy-based stopping criterion than **DSM**. In case that the user may start exploration with more attributes than those needed in the final model, we propose an online feature selection method that adaptively selects the top-k relevant attributes. Experimental results using real-world workloads show that our system significantly outperforms the state-of-the-art explore-by-example systems in accuracy (19x$\sim$64x accuracy improvement) and convergence speed while maintaining the per-iteration time within 1 to 2 seconds and our online feature selection method improves accuracy from nearly 0 without feature selection, to above 80%.

Last but not least, we address the label noise problem when the labels provided by the user are potentially corrupted. We enhance the robustness of our system by integrating advanced data cleansing methods and a refinement of the polytope-based model into **DSM**$_F$. The resulting algorithm, referred to as the Robust Dual-Space Model (**RDSM**$_F$), is compared to traditional active learning for evaluation since the state-of-the-art explore-by-example systems fail to deal with noisy labels. Experimental results using real-world datasets and user interest patterns show that our proposed algorithm substantially outperforms traditional active learning in accuracy (up to 22x accuracy improvement) while achieving reasonable efficiency for interactive data exploration (1 to 3 seconds per iteration).

Institut Polytechnique de Paris
91120 Palaiseau, France