Interactive Data Exploration via Machine Learning Models

Olga Papaemmanouil*, Yanlei Diao±, Kyriaki Dimitriadou*, Liping Peng±
* Brandeis University, ±University of Massachusetts, Amherst
olga@cs.brandeis.edu, yanlei@cs.umass.edu, kiki@cs.brandeis.edu, lppeng@cs.umass.edu

Abstract

This article provides an overview of our research on data exploration. Our work aims to facilitate interactive exploration tasks in many big data applications in the scientific, biomedical and healthcare domains. We argue for a shift towards learning-based exploration techniques that automatically steer the user towards interesting data areas based on relevance feedback on database samples, aiming to achieve the goal of identifying all database objects that match the user interest with high efficiency. Our research realizes machine learning theory in the new setting of interactive data exploration and develops new optimizations to support "automated" data exploration with high performance over large databases. In this paper, we discuss a suite of techniques that draw insights from machine learning algorithms to guide the exploration of a big data space and leverage the knowledge of exploration patterns to optimize query processing inside the database.

1 Introduction

Today data is being generated at an unprecedented rate. Every day large data sets are collected from sensors and scientific instruments that monitor our environment. For instance, LSST [17], a leading effort in astronomical surveys, is expected to store 55 petabytes of raw imagery ultimately and the database catalog containing descriptions of these objects and their observations is expected to approach 50 petabytes in size. Although data volumes and the user community of big data sets continue to grow, the human ability to comprehend data remains as limited as before. Hence, in the "Big Data" era we are faced with an increasing gap between the growth of data and the limited human ability to comprehend the data. Our work on data exploration aims to deliver new software tools to bridge this increasing gap.

Database management systems (DBMSs) have been long used as standard tools to store data and query it to obtain valuable information. However, traditional DBMSs are suited for applications in which the structure and the content of the database, as well as the questions (queries) to be asked are already well understood by the user. Unfortunately, these fundamental assumptions made by the DBMSs are becoming less true as the volume and diversity of data grow. First, the structure and content of the database are hard to understand, even for database experts. Second, finding the right question to ask is a long running complex task by itself, often requiring a great deal of experimentation with queries, backtracking on the basis of query results and revision of results at various points in the process.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Therefore, we argue that fundamental new models and tools are needed to increase the usability of DBMSs. Towards this direction, our work proposes "interactive data exploration" as a new service of a database management system, and it offers a suite of new algorithms, methods and optimizations to support this service for a broad user community across science, healthcare, and business. Our approach leverages machine learning techniques and offers new data management optimization algorithms to provide effective data exploration results as well as high interactive performance over databases of big sizes.

2 Interactive Data Exploration: Overview & Challenges

To support the task of interactive data exploration, we introduce a new approach for system-aided exploration of big data spaces that relies on *automatically learning* user interests and infers "classification" models that retrieve data relevant to the user interests. To achieve this, we rely on an interactive learning approach that iteratively requests user feedback on strategically collected data samples. In a nutshell, the user engages in a "conversation" with the system by characterizing a set of data samples as relevant or irrelevant to his interest. The user feedback is incrementally incorporated into the system and used to gradually improve the effectiveness of the query steering process, that is, to lead the user towards interesting data areas and eventually generate a classification model that precisely characterizes the set of data matching the user interest.

This interactive query steering process is depicted in Figure 1. Initially, the user is presented with a sample set selected to capture the diversity of the overall data exploration space. The iterative query steering process starts when the user provides feedback on the relevance of these samples. Labeled samples are used as the training set of a classification model that characterizes the user interest, i.e., predicting the data objects relevant to the user based on the feedback collected so far (*Learning*). Subsequent iterations aim to refine the characterization of the user interest by exploring further the data space: it identifies promising data areas to be sampled further (*Space Exploration*) and it retrieves the next sample set to show to the user. To achieve that, we leverage current knowledge of the user interest as defined by the user model. New samples and the user feedback on them are incorporated with the already labeled samples and a new user model is built in the next iteration. The above steps are executed iteratively aiming to converge to a model that captures the user interest within acceptable accuracy. The steering process is terminated when the classifier's accuracy reaches a system-defined threshold (i.e., on the number of labeled objects or convergence rate) or the user terminates the process explicitly.

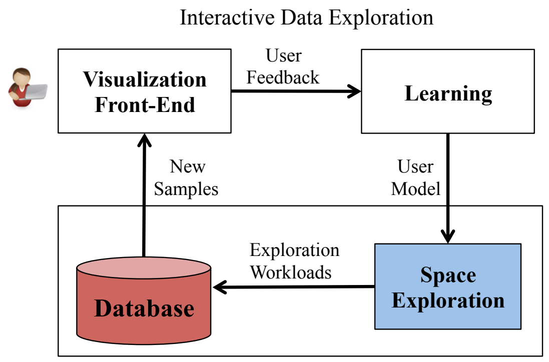


Figure 1: Interactive Data Exploration.

In our framework, users are asked for feedback on data objects. In the back-end, each object is mapped to a set of features collected through domain-specific feature extraction tools. Data objects in our target applications include many layers of features. We make a concrete distinction between the feature space one uses for visualizing data objects for annotation and the feature space used for exploration. Specifically, while the front-end should visualize high-level features which humans can understand and interact with effectively (i.e., pictures, maps, summarized values), the back-end exploration is performed on an extended set of dimensions that include also lower level features. This distinction allows users to review data objects while concealing the detailed feature (attribute) set of the underlying database. At the same time, it allows for more effective exploration as the system learns user interests based on a set of features that is wider from the one humans can perceive. Hence, our learning-based approach can provide more insightful descriptions of the user interests than the one humans could generate manually.

Research Challenges Towards realizing our vision for a learning-based data exploration system, we have identified two main research challenges:

1. *Fast Convergence*: A core research issue is to decide our exploration areas and how to sample them for objects to present to the user for further feedback. The challenge is to make such *exploration decisions* in a way that the user model converges to the true model with the minimum number of labeled samples.

2. *Interactive Performance*: The learning and space exploration steps of each iteration must offer interactive performance, e.g., complete within seconds, as the user may be waiting online. This challenge becomes even more apparent in large databases.

Existing DBMSs are not designed to support multi-step exploration tasks with interactive performance over large databases. At the same time, existing machine learning algorithms cannot solve our data exploration problem either. Classification algorithms (e.g., [3]) can build the user model but do not deal with the question of *which* data samples to show to the user and cannot be used to tackle the challenge for fast convergence. Active learning solutions [24] aim to identify the most promising data samples to label in order to maximize the learning outcome while minimizing the user effort. However, our recent results [9] revealed that those algorithms are not designed for interactive data exploration over large databases as they examine and rank *all* database objects before they identify the best sample to show to the user. Therefore, they cannot offer fast convergence nor interactive performance on big data sets.

In a nutshell, existing machine learning algorithms and database systems do not address the two main challenges of interactive data exploration. Hence, our research advocates a close synergy between machine learning and database algorithms and offers methods and tools that address the above research challenges.

3 Learning-based Exploration Strategies

We next discuss two exploration strategies that target diverse types of user interests. One uses decision tree classifiers for predicting linear interest patterns and the second one uses SVM (Support Vector Machine) models for capturing non-linear interest patterns.

3.1 Decision trees: learning linear patterns

Decision-tree classifiers can be very effective in discovering linear patterns of user interests, i.e., interests captured by conjunctive and/or disjunctive of linear (range) predicates. We refer to such interests as *relevant areas*. We designed an exploration framework [7–9] that leverages decision tree learning to efficiently produce highly accurate user models. Decision trees are easy-to-interpret prediction models that describe the features characterizing our relevant objects. Hence, the classification rules can be easily translated to simple boolean expressions and therefore to query expressions that retrieve all objects predicted to be relevant to the user.

The exploration is performed in a d-dimensional space where each tuple represents a d-dimensional object and the relevant areas are hyper-rectangles with up to d dimensions. The exploration space may include attributes both relevant and irrelevant to the final expression that captures the true user interest. The steering process starts when the user provides feedback on the relevance of the first set of retrieved samples. We assume a binary feedback model where the user indicates whether a data object is relevant or not to her. The labeled samples are used to train a decision tree which may use any subset of the d attributes of the exploration space to characterize user interests. Each iteration refines the characterization of the user interest by *exploring* further the data space trying to collect more insight on what the user likes as well as *exploiting* the knowledge we have collected.

Exploration. To explore our data space, our framework defines sampling areas over multiple *exploration levels*. For a given level, we divide each normalized attribute domain into width ranges that cover a given percentage of the normalized domain, effectively creating a number of grid cells (Figure 2(a)). The width of each cell defines the granularity of the specific exploration level. A lower number leads to more grid cells of smaller width per dimension. Each cell in our grid covers a certain range of attribute values for each of the d exploration attributes. Therefore, each cell includes a set of unique attribute value combinations and it includes the data objects that match these attribute values. For the same exploration level we also construct k clusters, by clustering off line all data in the data space. By default our highest level creates a single cluster and each level doubles the number of clusters of its previous one.

Our exploration step starts by sampling at the highest exploration level (i.e., with one cluster and one grid cell) and moves on at each iteration to the next lower level until the user terminates the exploration. At each level it samples dense areas by collecting one random sample within each cluster of that level. Next, it samples sparse sub-areas by retrieving one random sample within specific grid cells of the same exploration level. These are the non-empty grid cells from within which no sample has been retrieved yet. The user is presented with the all collected samples and provides feedback. This approach adjusts the sample size to the skewness of our exploration space (i.e., we collect more samples from dense sub-areas) while it ensures that any sparse relevant areas will not be missed (i.e., sparse sub-areas are sufficiently explored).

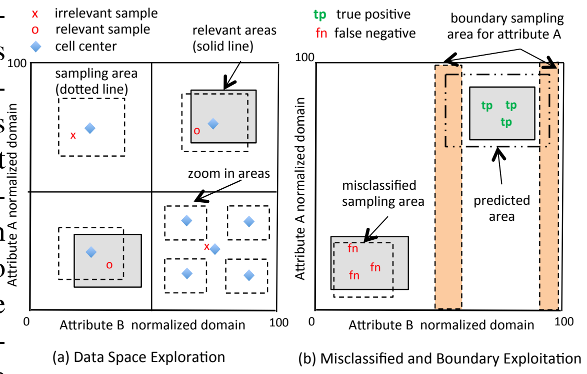

x irrelevant sample
o relevant sample
cell center
relevant areas (solid line)
sampling area (dotted line)
zoom in areas
Attribute A normalized domain
Attribute B normalized domain
0
100
100
(a) Data Space Exploration
tp true positive
fn false negative
boundary sampling area for attribute A
tp
tp
tp
misclassified sampling area
predicted area
fn
fn
fn
(b) Misclassified and Boundary Exploitation

Figure 2: Exploration and Exploitation

Exploitation. The exploitation step aims to leverage prior knowledge. At each iteration it uses the feedback collect so far as well as the latest user model to identify promising data areas to be sampled further. Specifically, we exploit the following information:

- *Misclassified objects*: We randomly sample around each false negative (i.e., objects labeled as relevant but classified as non-relevant by the latest user model) (Figure 2(b)). To further reduce the sampling areas (and hence the sampling overhead), clustering techniques are used to identify close-by false negatives (which most likely belong to the same relevant area) and create a single sample area around each of the generated clusters. This technique increases both the precision and the recall of the user model (since it increases the relevant samples) while reducing the false negatives.

- *Decision boundaries*: Given a set of relevant areas identified by the decision tree classifier, we randomly sample around each boundary (Figure 2(b)) aiming to refine them and improve the accuracy of our user model. The approach is applied in parallel to all the boundaries of the relevant areas, allowing us to shrink/expand each area as we get more feedback from the user.

In Figure 3 we demonstrate the impact our exploration and exploitation techniques. Our evaluation metric is the number of samples we need to reach different accuracy levels of our user model. We measure accuracy by the F-measure (i.e., the harmonic mean of precision and recall) and we assume the user's relevant objects lie within one single area that covers 7-9% of the normalized domain of the `rowc` and `colc` attributes of the `PhotoObjAll` table in the SDSS database [26]. In this figure, *Explore* uses only the exploration step, *Explore/ExploitFN* uses the exploration and the exploitation of the false negatives and *Explore/ExploitAll* uses the exploration and all exploitation steps (i.e., false negatives and boundaries). The results show that combining all three steps gives the best results, i.e., better accuracy

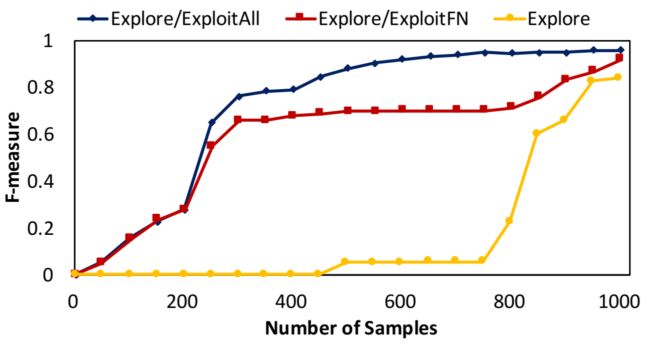


Figure 3: Exploration & Exploitation Impact

with fewer labeled samples. Specifically, using only the exploration step requires more than 800 labeled samples to reach an accuracy higher than 20%. Adding the false negative exploitation increases the accuracy by an average of 54%. Finally, adding the boundary exploitation further improves the accuracy by an average of 15%. Hence, all three phases are highly effective in predicting relevant areas while reducing the amount of user effort.

3.2 SVM: discovering non linear patterns

Non-linear patterns, that is, patterns that cannot be captured by range predicates, are prevalent in applications ranging from location-based searches to scientific exploration tasks using complex predicates. While our decision tree based approach can approximate non-linear patterns, it suffers from poor performance. For example, to predict a circle-shaped relevant area "$(rowc–742.76)^2+(colc–1022.18)^2 < 100^2$" on two location attributes $rowc$ and $colc$ in the SDSS data set [26], the decision tree model required over 1800 training samples and approximated the circle region with 95% accuracy using 71 range predicates combined through conjunction/disjunction, as illustrated in Figure 4. This motivated us to seek a more efficient approach to supporting non-linear patterns, reducing both the user labeling effort and the querying and sampling cost in the database.

Our exploration approach uses Support Vector Machines (SVMs) as the classification algorithm [7]. Here, the training set (i.e., labeled samples) in the *data space* is mapped, via a *kernel function*, to a higher-dimensional *feature space* where examples of different classes are linearly separable. Figure 5 shows a 3-dimensional feature space (manually selected by us) where the training points of the circle area in Figure 4 are linearly separated; in practice an SVM may need many more dimensions to see such linear separation. Then among the many hyperplanes that might classify the data in the feature space, SVM selects the one, called the *decision boundary* $\mathcal{L}$, with the largest distance to the nearest mapped samples of any class; this boundary $\mathcal{L}$ is used as the model of user interest as it separates relevant from irrelevant objects. The main challenge here is to identify at each iteration of the exploration process, the next to-be-labeled sample that can quickly improve the accuracy of the current user model $\mathcal{L}$.

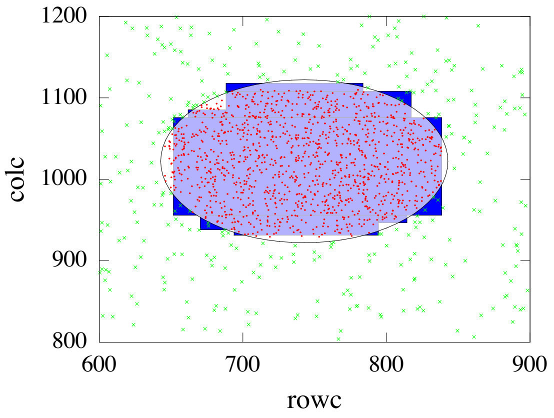


Figure 4: A circle area (green area) approximated by decision trees (blue area), with training points in 2-D data space (red: Y, green: N).

Exploration. Recent active learning theory [2] proposed to choose the example closest to the current decision boundary. However, they suggest a search through the entire data set in *each* iteration, which is prohibitively expensive. Precomputation to store the distance of each tuple to the current decision boundary is not possible either, because the decision boundary changes in each iteration. Our system puts active learning theory into practice: we find the unlabeled example closest to the current decision boundary $\mathcal{L}$ without retrieving all the tuples and evaluating their distances to $\mathcal{L}$. Our system uses two techniques for identifying samples to show to the user in each iteration.

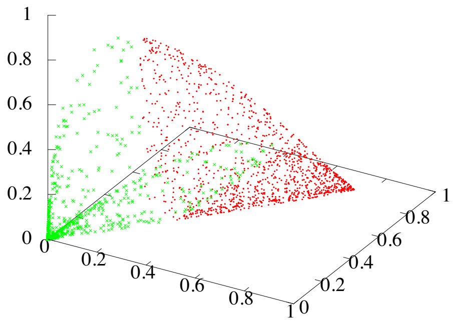


Figure 5: Linear separation of training points in a 3-D feature space using SVM (red: Y, green: N).

Bounding the search area using decision trees. We define a δ-region around the current SVM decision boundary $\mathcal{L}$ and form a two-class training data set such that points inside the δ-region are the relevant class and points outside the δ-region are not. Then a decision tree can be trained to approximate the δ-region and can be easily translated to an exploration query, Q, to send to the database $\mathcal{D}$.

Finally given the query result $Q(\mathcal{D}) \subseteq \mathcal{D}$, we iterate over this set and find the example closest to $\mathcal{L}$. Note that δ can be set to balance two trends: a too small δ can lead to too few training points in the relevant class while a too large δ may result in $Q(\mathcal{D}) = \mathcal{D}$.

Branch and bound search. Our system also builds indexes such as R-trees [1] and CF trees [29] over the database, and performs fast branch-and-bound search over these indexes. Take R-tree for example. Each R-tree node offers a hyper-rectangle, $[a_j, b_j],\ j = 1, \ldots, d$, as a minimum bounding box of all the data points reachable from this node. Given the current SVM decision boundary $\mathcal{L}$, we search the R-tree top-down in a depth-first fashion and always maintain the current closest tuple, $\boldsymbol{x}^*$, and its distance to $\mathcal{L}$, $f(\boldsymbol{x}^*, \mathcal{L}) \overset{\text{def}}{=} f^*$. Note that $f^* = +\infty$ before any leaf node is visited. For each intermediate node visited, we dynamically compute a lower bound of the distance from any point in its hyper-rectangle to $\mathcal{L}$ by calling a constrained optimization solver: $\min_{\boldsymbol{x}} f(\boldsymbol{x}, \mathcal{L})$ s.t. $a_j \leq \boldsymbol{x}^{(j)} \leq b_j,\ j = 1, \ldots, d$. If the lower bound is already higher than f^*, we can prune the entire subtree rooted at this node. Once we reach a leaf node, we can update $\boldsymbol{x}^*$ and f^* accordingly. Then the final $\boldsymbol{x}^*$ is the closest tuple to $\mathcal{L}$ in the entire database.

While the above optimizations may lead to a more efficient implementation of active learning theory, we also observe that existing learning theory falls short in two aspects.

Convergence rates. Although active learning theory aims to identify the best samples from input to expedite the convergence of the model, it offers only asymptotic results on the convergence rate [27]. To bring such theory to practice, it is crucial to know the accuracy of the user interest model at any point of the exploration process, so that the DBMS knows when the model is "good enough" and can return the rest of relevant objects from the database with high confidence. Our current research is performing an in-depth analysis of the convergence rate in order to provide useful runtime information on the effectiveness of a learned model.

Sparse or insufficient samples. Our initial results reveal that SVM-based exploration strategies suffer from slow convergence when the data space involves more than 4 or 5 dimensions (*high dimensionality*) and when the true user interest model amounts to a very small area in the total data space (*high selectivity*). In the case of high dimensionality, samples are sparsely distributed in the data space, which prevents SVM from learning the correct model efficiently even if the number of truly relevant dimensions is limited. In the case of high selectivity, the true user interest model may select less than 1% of the objects in the database. Existing active learning theory leads to a sample set that is strongly biased towards negative samples. Such imbalance between negative samples and positive samples also causes SVM to take long to learn the correct model. Our ongoing work is exploring new sampling techniques for the workloads where existing active theory does not work well.

4 Supporting Interactive Performance

Our research aims to advance the state-of-the-art of database system design by developing new query processing and optimization techniques for new data exploration workloads. Next we describe two optimizations that are part of our future work.

4.1 Sample Acquisition Optimizations

Our data exploration approach adds a big processing overhead on the underlying data processing engine due to large numbers of sample acquisition queries issued to retrieve additional samples. These queries represent new interesting workloads in the data processing back-end. Next we discuss the unique characteristics of these exploration queries and propose optimizations for these workloads.

In the decision tree-based exploration, our results indicated that our data exploration workload consists mostly of numerous range queries on individual attributes. Specifically, for k d-dimensional data areas characterized as relevant by the classifier and for m misclassified objects we execute $(k + m) \times d$ range queries, or hyper-rectangle queries in the d-dimensional space, in each iteration[1]. To illustrate the workload better, consider two attributes, A and B, used in a decision tree. The exploration queries may include Q1: $A \in [a_1, a_2]$ and $B \in (-\infty, \infty)$, where dense sampling is performed for $A \in [a_1, a_2]$ and $B \in [b_1, b_2]$, and sparse random sampling is performed for $B \in (-\infty, b_1) \cup (b_2, \infty)$ to check if the attribute B is irrelevant to the user interest. Similarly, we may also have a simultaneous exploration query Q2: $A \in (-\infty, \infty)$ and $B \in [b_3, b_4]$, where areas of $A \in [a_3, a_4]$ and $B \in [b_3, b_4]$ are densely sampled while $A \in (-\infty, a_3) \cup (a_4, \infty)$ is randomly sampled. If we have a covering index on (A, B, C), the access patterns of the two queries in the index are illustrated in Figure 6. As the dimensionality of the exploration queries increases, e.g., to 5 or 10 attributes, the number of simultaneous queries and the overlap among them increase significantly. Similarly, in the SVM-based exploration, the exploration queries may be k-nearest neighbor queries from a set of data points (support vectors), which can be viewed as a set of ball-shaped queries in a d-dimensional space where significant overlap among these queries exists.

Since each iteration of query steering may issue many simultaneous queries with possible overlap in the data space, separate evaluation of these queries wastes processing resources and leads to poor performance. Traditional multiple-query optimization [20, 21, 23] and continuous query processing (e.g., [5, 6]) focus on analyzing query expressions and finding the common sub-expressions for shared processing. Our exploration queries, however, have more clearly-defined access patterns, e.g., a set of hyper-rectangles or ball-shaped areas in a d-dimensional space. Hence more effective optimizations may be possible. For instance, given a covering index that includes all attributes in the exploration queries, a single scan of all the leaf nodes of the covering index offers an improvement because it avoids repeated scans of the overlapped regions among queries. When the index is large itself, better prediction of necessary regions to visit allow us to skip a fraction of leaf nodes of the index. Furthermore, some queries could mix dense sampling in focused regions and sparse random sampling in wide regions as shown in Figure 6. Hence, the multi-query optimization will also consider quick random sampling using indexes [19].

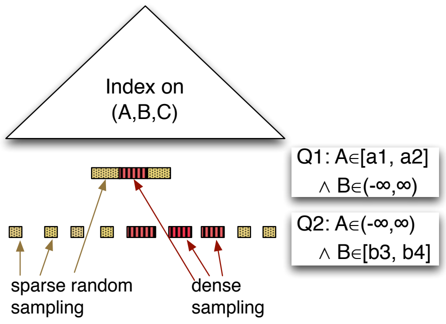


Figure 6: Access patterns of two exploration queries in a covering index.

4.2 Model-based Collaborative Filtering: Predicting exploration trajectories

Data exploration tasks can greatly benefit from information about past user profiles. To this end, we recently focused on how to leverage detailed information of user profiles with the end goal of improving the efficiency of interactive data exploration tasks. User profiles include feedback on data objects, the final classification model as well as its lineage, i.e., sequence of steering queries and samples that resulted from the exploration process.

To leverage past user exploration profiles, we employ a *model-based collaborative filtering* approach. Collaborative filtering (CF) uses the known preferences of a group of users to predict the unknown preferences of other users. Model-based CF systems involve learning a model based on data objects (aka items) ratings. This allows the system to learn to recognize complex patterns based on the training data, and then make intelligent *predictions* for collaborative filtering tasks based on the learned models without having to use the complete data set every time. This offers the benefits of speed and scalability making model-based CF techniques a good fit for real-time predictions on the basis of very large data sets.

In the context of our interactive data exploration framework clustering models can be used to predict relevant areas and recommend them to the back-end as promising sampling areas. Specifically, clustering algorithms can identify groups of users who appear to have similar relevance feedback. Once the clustering model is created, we can leverage it during an online exploration task as follows. Given the relevance feedback of the current

[1]In some cases, the data space can be reduced to include only relevant attributes however, in the worse case scenario, sampling queries are executed on all d dimensions of the exploration space.

user, the system predicts the cluster it belongs to, i.e., its neighbors with respect to their relevant objects. Then a traditional CF algorithm is applied to rate candidate areas to sample based on the interests of *only* these neighbors. At each iteration, more feedback is collected and our prediction of the "neighbors" improves helping the system to converge faster to the current user's classification model. Our clustering-based approach has better scalability than typical CF methods because it makes predictions within much smaller clusters rather than the entire user and object base. The complex and expensive clustering computation is run once offline and the resulting clusters can be used by any future user. Next we sketch the technique in more detail.

Cluster-based Exploration CF techniques use user ratings on data objects to calculate the similarity between users (or objects) and make predictions according to those calculated similarity values. However, similarity values are based on common data objects and therefore are unreliable when the common data objects rated by the users are therefore few. This is indeed the case of our interactive exploration framework: users provide feedback on few sample objects of the entire database and the intersection of labeled data sets of past users is often quite small, even when user interests highly overlap. To address this challenge we apply our clustering-based exploration on the level of the predicted relevant areas as opposed to labeled items by using past user models.

Each past user of our interactive exploration framework, $u_i \in \{u_1, u_2, ..., u_m\}$, is represented by its final classification model that characterizes the relevant areas for that user. Given a collection of user models one can identify a partitioning schema of the data space such that each partition $p_i \in \{p_1, p_2, ..., p_n\}$ involves items that are *all* relevant to the *same* set of users. Each user is represent by (partition, relevance) pairs which can be summarized in a user-partition table R. This table R contains the relevance score R_{ij} that is provided by the user u_i on the partition p_j. For a binary relevance feedback model R_{ij} is 1 if the partition p_j contains objects characterized as relevant by the model of user u_i and is 0 otherwise. Alternatively one can use a fixed partitioning schema of the exploration space in which case the relevance score R_{ij} is the degree of overlap between the user's u_i predicted relevant areas and the partition p_j.

We use a clustering algorithm on the user partition table R to form groups of users that appear to provide similar relevance feedback (identified the same partitions as relevant). We can then assign the current user to one of these clusters as follows. Given the active user's u_a latest decision tree, we calculate the overlap of the user's relevant areas with each data partition p_i. We create a vector with the overlap per partition R_i vector to characterize the current user. Full overlap indicates a relevance of 1 for that partition and no overlap a relevance of zero. Partial overlap can be calculated based on the size of the overlapping area. Using this vector the current user will be assigned to the cluster with the most relevant past users (i.e., users that explored similar partitions).

Once the neighborhood is obtained, a CF algorithm is used to predict the relevance score for all partitions. In particular, the relevance (prediction) score $R_{a,j}$ for the active user u_a on a partition p_j can be calculated by aggregating the neighbors relevance feedback on the partition p_j using a number of aggregation functions. Examples include averaging the relevance score of that partition over all the neighbors or calculating the weighted average of the neighbors relevance feedback where the weight is based on the Pearson similarity between the neighbor and the current user (i.e., the more similar two users are wrt to their feedback, the higher their weight on identifying promising sampling areas).

Our data exploration framework can leverage the output of the above process in multiple ways. For example, we can use the relevance score of each partition for the current user to apply a weighted sampling on them, i.e., more relevant partitions are sampled with higher ratio than less relevant ones. Diversification techniques [16] can also be combined with the relevance score in order to identify the most diversified set of samples with high relevance score to show to the current user. Collecting feedback on these samples will allow our system to collect more insight on the user's interests. All the above can be executed iteratively - each round improves the user model which improves also the relevance predictions on each partition increasing eventually the convergence rate of our exploration.

4.3 Dimensionality Reduction

Our initial results revealed that the exploration space is often characterized by high redundancy. As data sets become highly multi-dimensional data exploration suffers from slow convergence. This is due to the fact that many dimensions are correlated with others while some are irrelevant to the user's interest.

Offline Reduction Eliminating redundant feature combinations can be done offline by removing dimensions with low variation, since these dimensions will not be "useful" to our classifier. To demonstrate this with an extreme example, if all sky objects have the same brightness level, then the brightness attribute will not be a good separator of irrelevant/relevant objects. Variation along each dimension can be captured by the variance of its values and dimensions with relatively low variance can be removed. Principal component analysis (PCA) can also be used offline to identify only the uncorrelated dimensions our models should be using.

Online Reduction Online reduction techniques aim to identify dimensions irrelevant to the current user and hence eliminate them as early as possible from the exploration space. One approach we explore is based on the expectation that the distribution of relevant labels collected from the current user will be more scattered when projected on irrelevant dimensions and more clustered on specific areas of the relevant domains. Based on this, projecting the samples characterized as relevant on each dimension of our exploration area and comparing their distribution across each dimension independently could indicate the most relevant dimensions and allow us to adapt the number of samples to be higher for the relevant domains.

We also leverage past user models to identify dimensions irrelevant to the current user. Specifically, past user models (i.e., decision trees) reveal the relevant dimensions for past users. Hence, once the neighbors of the current user are identified we apply dense sampling on the relevant dimensions for these neighbors and sparse sampling on the rest of the exploration domains. The collected feedback is further used to identify the relevant dimensions for the current user as described above.

5 Prototyping and Demonstration

We have implemented a prototype of our interactive data exploration framework that includes the techniques we described in Section 3.1 and 3.2. The system is designed on top of a relational database system and its architecture (Figure 7) includes three software layers: (1) an interactive visualization front-end, (2) the AIDE middleware that implements our exploration techniques and (3) a database backend supporting our data exploration. An initial prototype of our system was demonstrated at VLDB 2015 [7].

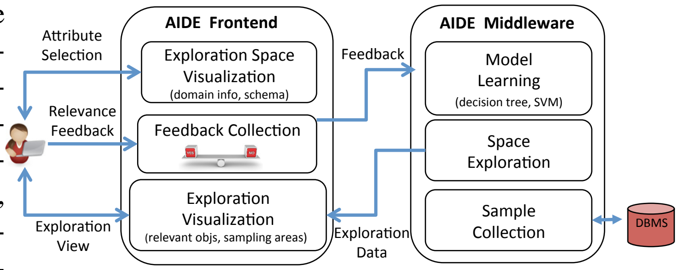
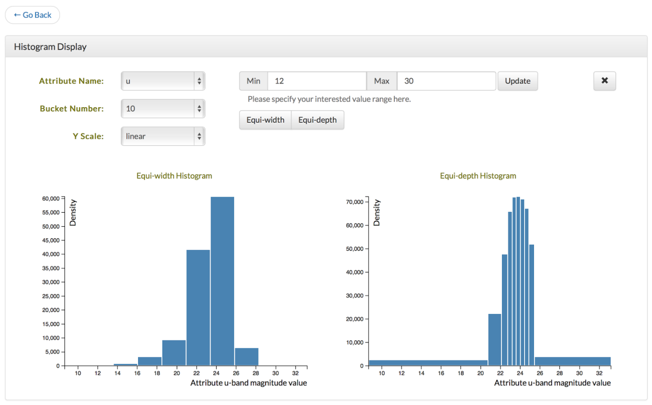


Figure 7: Architecture of our system

Our visualization front-end provides several functionalities. The user is initially presented with the database schema and he can select an initial subset of attributes of interest, which will be refined later by the data exploration. Our front-end can also visualize domain distributions of each attribute to further allow the user to filter attributes based on the domain characteristics and restrict the value ranges of the relevant attributes for consideration (e.g., focus on a dense region or a region close to a landmark). Users can select between different types of plots of the underlying data distributions, such as histograms and heat maps. Figure 8a shows a histogram example on an attribute in the SDSS [26] data set.

The system starts a series of iterations of sample labeling, model learning and space exploration. The front-end supports this process by visualizing various subspaces of the exploration attributes, presenting data samples to the user, collecting yes/no labels from the user regarding the relevance of the shown samples and showing the locations of labeled samples in the exploration space. Figure 8b shows an example of this interface.


Histogram Display

Attribute Name: u

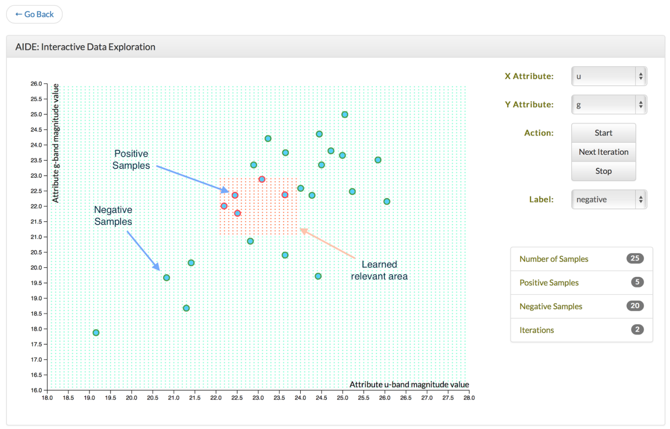
Bucket Number: 10

Y Scale: linear

Min 12 Max 30

Please specify your interested value range here.

(a) Histogram visualization for exploration attributes.


AIDE: Interactive Data Exploration

X Attribute: u

Y Attribute: g

Label: negative

Number of Samples	25
Positive Samples	5
Negative Samples	20
Iterations	2

(b) Exploration visualization (learned areas, labeled samples).

Figure 8: Front-end visualization interface.

Sitting below the visualization front-end is the "automatic user steering" layer (middleware in Figure 7), which is the heart of our system. This component is implemented in Java, with a few machine learning libraries integrated in the system. At each iteration it incorporates the newly collected labeled samples and generates a new classification model. At any point the user can request a visualization of the current user model (i.e., decision tree or SVM decision boundary) which entails highlighting the objects classified as relevant to the user. The user can then decide to stop the exploration (if he is satisfied with the current set of identified objects) or to proceed to the next round of exploration.

The database backend uses PostgreSQL. The database engine includes various sampling techniques implemented as stored procedures. These techniques are designed to support the exploration approaches we discussed above. For example one procedure supports the decision tree approach to learning linear patterns (§3.1) by selecting a predefined number of random samples within a given distance from the center of a d-dimensional area, while other procedures support random and weighted sampling.

6 Related Work

Numerous recent research efforts focus on data exploration. The vision for automatic, interactive navigation in databases was first discussed in [4] and more recently in [28]. YMALDB [10] supports data exploration by recommending to the user data similar to his query results. DICE [15] supports exploration of data cubes using faceted search and in [12] they propose a new "drill-down" operator for exploring and summarizing groups of tuples. SciBORQ [25] relies on hierarchical database samples to support scientific exploration queries within strict query execution times. Idreos et al. [18] proposed a system for interactive data processing tasks aiming to reduce the time spent on data analysis. In [22] they interactively explore the space based on statistical properties of the data and provide query suggestions for further exploration. In [11] they propose a technique for providing feedback during the query specification and eventually guiding the user towards her intended query. In [13] users rely on prefetching and incremental online processing to offer interactive exploration times for window-based queries. SearchLight [14] offers fast searching, mining and exploration of multidimensional data based on constraint programming. All the above systems differ from our approach: we rely on the user's feedback on data samples to create progressively an effective training set for generating machine learning models that predict the user's data interests.

7 Conclusions

This article provided an overview of our on-going work on learning-based data exploration. We highlighted the research challenges and a set of solutions that attempt to address them. Our proposed techniques can be crucial for deriving insights from huge and complex data sets found in many discovery-oriented applications. Human exploration effort across large data sets will be significantly reduced, as users will be methodically steered through the data in a meaningful way. Such automated steering, fully exploiting user interests and application characteristics while grounded in rigorous learning theory, will assist users in discovering new interesting data patterns and eliminate expensive ad-hoc exploratory queries.

8 Acknowledgments

This work was funded by NSF grants IIS-1253196, IIS-1218524, and IIS-1218524 and a gift from HP Labs.

References

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. 1990.

[2] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *J. Mach. Learn. Res.*, 6:1579–1619, Dec. 2005.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[4] U. Çetintemel, M. Cherniack, J. DeBrabant, Y. Diao, K. Dimitriadou, A. Kalinin, O. Papaemmanouil, and S. Zdonik. Query steering for interactive data exploration. In *Proceedings of the 6th Biennial Conference in Innovative Data Systems Research (CIDR)*, 2013.

[5] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for internet databases. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD*, 2000.

[6] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer. Path sharing and predicate evaluation for high-performance XML filtering. *ACM Trans. Database Syst.*, 28(4):467–516, 2003.

[7] Y. Diao, K. Dimitriadou, Z. Li, W. Liu, O. Papaemmanouil, K. Peng, and L. Peng. AIDE: An Automatic User Navigation Service for Interactive Data Exploration (Demonstration). In *VLDB*, 2015.

[8] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-Example: An Automatic Query Steering Framework for Interactive Data Exploration. In *33rd ACM Special Interest Group in Data Management (SIGMOD)*, 2014.

[9] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. AIDE: An Active Learning-based Approach for Interactive Data Exploration. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28(11):2842 – 2856, 2016.

[10] M. Drosou and E. Pitoura. YMALDB: exploring relational databases via result-driven recommendations. *VLDB Journal*, 22:849–874, 2013.

[11] L. Jiang and A. Nandi. SnapToQuery: Providing Interactive Feedback During Exploratory Query Specification. *VLDB 2015*.

[12] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran. Smart drill-down: A new data exploration operator. *VLDB 2015*.

[13] A. Kalinin, U. Cetintemel, and S. Zdonic. Interactive data exploration using semantic windows. In *SIGMOD*, 2014.

[14] A. Kalinin, U. Çetintemel, and S. B. Zdonik. Searchlight: Enabling integrated search and exploration over large multidimensional data. *VLDB 2015*.

[15] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and Interactive Cube Exploration. In *ICDE*, 2014.

[16] H. Khan, M. Sharaf, and A. Albarrak. DivIDE: efficient diversification for interactive data exploration. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management (SSDBM' 14)*, 2014.

[17] Large synoptic survey telescope: the widest, fastest, deepest eye of the new digital age. `http://http:/www.lsst.org/`.

[18] Martin Kersten and Stratos Idreos and Stefan Manegold and Erietta Liarou. The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds. *International Conference of Very Large Databases (VLDB)*, 4(12), 2011.

[19] F. Olken. *Randomized sampling from databases*. PhD thesis, University of California, Berkeley, 1993.

[20] A. Rosenthal and U. S. Chakravarthy. Anatomy of a mudular multiple query optimizer. In *Proceedings of the 14th International Conference on Very Large Data Bases*, VLDB '88, pages 230–239, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[21] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. Efficient and extensible algorithms for multi query optimization. *SIGMOD Rec.*, 29(2):249–260, May 2000.

[22] T. Sellam and M. L. Kersten. Meet Charles, big data query advisor. In *biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[23] T. K. Sellis. Multiple-query optimization. *ACM Trans. Database Syst.*, 13(1):23–52, Mar. 1988.

[24] B. Settles. *Active Learning*. Morgan & Claypool, 2012.

[25] L. Sidirourgos, M. Kersten, and P. Boncz. SciBORQ: Scientific data management with Bounds On Runtime and Quality. In *CIDR*, 2011.

[26] Sloan digital sky survey. `http://www.sdss.org/`.

[27] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, Mar. 2002.

[28] A. Wasay, M. Athanassoulis, and S. Idreos. Queriosity: Automated Data Exploration. In *IEEE International Congress on Big Data*, 2015.

[29] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD*, 1996.