Active Learning with Generalized Queries

Jun Du
Department of Computer Science
The University of Western Ontario
London, ON, Canada
jdu42@csd.uwo.ca

Charles X. Ling
Department of Computer Science
The University of Western Ontario
London, ON, Canada
cling@csd.uwo.ca

***Abstract*—Active learning can actively select or construct examples to label to reduce the number of labeled examples needed for building accurate classifiers. However, previous works of active learning can only ask specific queries. For example, to predict osteoarthritis from a patient dataset with 30 attributes, specific queries always contain values of all these 30 attributes, many of which may be irrelevant. A more natural way is to ask "generalized queries" with don't-care attributes, such as "are people over 50 with knee pain likely to have osteoarthritis?" (with only two attributes: age and type of pain). We assume that the oracle (and human experts) can readily answer those generalized queries by returning probabilistic labels. The power of such generalized queries is that one generalized query may be equivalent to many specific ones. However, overly general queries may receive highly uncertain labels from the oracle, and this makes learning difficult. In this paper, we propose a novel active learning algorithm that asks generalized queries. We demonstrate experimentally that our new method asks significantly fewer queries compared with the previous works of active learning. Our method can be readily deployed in real-world tasks where obtaining labeled examples is costly.**

***Keywords*-active learning, generalized queries, supervised learning.**

I. Introduction

Active learning may hold the key for solving the data scarcity problem, i.e., the lack of labeled data, in supervised learning. Indeed, labeling a large set of data is often a costly and time-consuming process, but necessary to build an accurate classification model. For example, in webpage (image, news article, or face) classification, labels are often given by human experts, and thus, it is costly and time-consuming. Active learning, on the other hand, is able to actively request labels of a small number of selected or constructed examples, and thus, reducing the labeling cost significantly.

However, all previous works of active learning (see Section II for a detailed review) can only ask specific queries with all attribute values provided, and assume that the oracle could only answer such specific queries. For example, if the task is to predict osteoarthritis based on a patient dataset with 30 attributes, the previous active learners could only ask the specific queries as: does this patient have osteoarthritis, if ID is 32765, name is Jane, age is 35, gender is female, weight is 85 kg, blood pressure is 160/90, temperature is 98F, no pain in the knees, no history of diabetes, and so on (for all 30 attributes). Many of these 30 attributes may not be relevant to osteoarthritis in this case. Not only could specific queries like this confuse the oracles, but the answers returned are also specific: each label given is only for one specific query.

In real-world situations, the oracles (usually human experts) are often more readily to answer generalized queries, such as "are people over 50 with knee pain likely to have osteoarthritis?" Here only two relevant attributes (age and type of pain) are mentioned, and the other 28 are don't-care's. We have discussed with some experts in heart-disease diagnosis and used-car sale, and they regard this type of generalized queries intuitive and easy to comprehend. Thus, in this paper, we assume that the oracle is more powerful; it can answer generalized queries by returning probabilistic labels. Not only are such generalized queries more natural and relevant, answers from the oracle also provide much more information, as one generalized query is often equivalent to many specific queries. In this example, the answer for this query is for all people over 50 with knee pain. This allows the active learner to improve learning effectively and quickly.

The difficulty of the generalized queries is that the answers from the oracle can often be uncertain.[1] For example, the answer to the above generalized query can be "Yes with a 90% probability". An overly general query, such as "are people over 50 likely to have osteoarthritis?" (age only), might receive yes with only a 60% probability. Indeed, the experts in the heart-disease diagnosis and used-car sale also sometimes have to reply with low certainties in their answers. Highly uncertain answers can make learning difficult as they may introduce noise into the training data; or, they can cause a waste of queries if these answers are directly discarded.

In general, the more general a query is (with more don't-care attributes), the more powerful it is (representing more specific instances), but usually the more uncertain the answer

[1]This is true even if we assume that answers for specific queries are always 100% certain. However, in some real-world applications, answers for specific queries may also be uncertain. We will study this issue in our future work.

is from the oracle. Our task is to design an active learner that attempts to ask generalized queries with highly certain answers from the oracle. The task is not trivial. As far as we know, no previous work of active learning can deal with such generalized queries. See Section II for details.

In this paper we assume that the oracle is capable of answering generalized queries, and we propose a novel active learning paradigm in which such generalized queries can be asked and answered. We design a new algorithm called AGQ, for Active learner with Generalized Queries. AGQ can construct generalized queries with don't-care attributes, for either the pool-based or the membership-query active learner. See Section III for details. Experiments on synthetic and real-world datasets show that AGQ asks significantly fewer queries compared with the traditional active leaner. See Sections IV for details. To the best of our knowledge, this is the first work proposing active learning with generalized queries, and showing that it is highly effective.

II. Related Work

Most previous works of active learning can be divided into two paradigms: the pool-based active learning and the membership query.[2] In the pool-based active learning, a pool of unlabeled examples is given, and the learner can only choose examples to label from the pool [3]. Briefly speaking, the pool-based active leaner first evaluates each example in the pool, to decide which one can maximumly improve the performance of the current model. Then the learner acquires its label from oracle to update the labeled training set and the learning model, and the process repeats. On the other hand, active learning with membership queries (or direct query construction) can construct examples (without the need of the pool) and request labels [4], [5]. Previous experiments show that both of these active learning methods reduce the number of labeled examples needed, compared with labeling examples randomly.

The essence of active learning lies in the "goodness" measurement of the unlabeled examples with respect to the current model. Many criteria have been proposed in the literatures. *Uncertainty sampling* [3] considers the most uncertain example as the most valuable one, and has been thoroughly studied and widely used in many previous researches [6], [7], [2], [8], [9]. *Query-by-committee* (QBC) [10] is a more theory-based approach, and considers the example minimizing the version space as optimal. [11] implements QBC by constructing committees from ensemble methods (bagging, boosting, etc.), thus essentially transforms it to a variant of uncertain sampling. Besides, other criteria, such as *variance reduction* [12], *Fisher information ratio* [13], and *estimated error reduction* [14], are also elaborately designed and well accepted in active learning research area. In this paper, the proposed AGQ algorithm can be integrated with any of the above criteria, and the most widely used *uncertain sampling* is chosen for illustration and empirical study in the rest of the paper.

All previous works of active learning assume that the oracle could only answer specific queries, with all attribute values provided. To the best of our knowledge, our AGQ algorithm in this paper is the first work proposing active learning with generalized queries. Previously, [15] proposed active learning with feature labeling, which queries the label for one specific feature (for example, "$puck$" $\longrightarrow$ "$hockey$"), and is mainly used in natural language precessing. Although feature labeling is considered similar to the generalized query, our AGQ algorithm is significantly different in the following three aspects. First, instead of querying label for one specific feature, our AGQ could query the labels for multi-feature combinations (for example, "$puck$" + "ice" + "$player$" $\longrightarrow$ "$hockey$"). Thus, feature labeling is essentially a special case of our AGQ. In other words, our generalized query is a generic paradigm for both instance-based queries and feature-based queries. Second, AGQ always finds the most uncertain example (when integrated with uncertain sampling) and generalizes it to a query. Labeling such uncertain examples has been proved to be very effective in improving predictive accuracy (see Section IV-B for details). On the other hand, feature labeling generally finds the most predictive (or most frequent) feature for querying, thus the answer from the oracle may not provide much new information to improve the model. Third, and most importantly, as feature labeling always queries label for only one feature, the answer from the oracle could be very uncertain. To deal with this problem, it is assumed in [15] that the oracle could "skip" the uncertain queries. But in fact, the oracle has "worked" on those queries, and the oracle's effort is wasted. On the other hand, AGQ makes a minimal generalization of a specific query, thus the answers from the oracle tend to be certain. Our experiments show that the average certainty of the replies is 90% (see Section IV-B for details). In any case, every query of AGQ is counted, regardless of the certainty of the reply.

III. AGQ: Active Learning with Generalized Queries

In this paper we propose a new active learning paradigm in which the learner can ask generalized queries, and we assume that the oracle can answer such generalized queries. In this section, we will describe a novel active learning algorithm called AGQ (Active learning with Generalized Queries). AGQ can generalize attributes (nominal or numeric) with specific values to don't-care attributes.

As most previous works of active learning are pool-based, and use uncertain sampling to choose the most valuable unlabeled examples, in this paper, we will also describe

[2]Stream-based active learning [1] is considered as another paradigm in some literatures. In essence, it could be viewed as an online version of the pool-based active learning [2].

AGQ using uncertain sampling in pool-based paradigm. However, as our AGQ is a meta-learning method, it can be equally applied to the membership query active learning, or integrated with any other query strategy. We assume that examples are described by n nominal or numeric attributes $X_1, X_2, ..., X_n$ and the label Y of examples is binary, with values positive (1) and negative (0). The active learner is given an initial labeled training set R, and an unlabeled set U, from which the learner may choose examples to query for their labels from an oracle. A test set T is given but set aside to evaluate the accuracy of the learner during label acquisition.

The AGQ algorithm can be broken down into the following four major steps:

1) The first step is the same as in the previous pool-based active learning algorithms [6], [16], [9]. An initial learner L is built using the current labeled training dataset R. Then, L is used to predict each example in the pool U. The most uncertain example from the pool is chosen. (If the membership active learning is used, then the most uncertain example would be constructed in this step.)
   As an example, the specific example from the pool could be $[1, 0, 1, 1, 0, 1]$, with the predicted probability of 52% for the class 1 (and 48% for the class 0), according to the current model L. This is the most uncertain (the probability of the majority class is closest to 50%) among all examples in the pool.
2) AGQ then finds irrelevant attributes in the most uncertain example above, and substitute them with "$*$" (representing don't-care values).
   For example, the generalized query based on the example $[1, 0, 1, 1, 0, 1]$ could be $[1, *, 1, *, 0, 1]$.
3) AGQ submits this generalized query to the oracle, which will return a label with a probability distribution.
   For example, the oracle may return a probability of 0.9 for positive (and 0.1 for negative) for the generalized query $[1, *, 1, *, 0, 1]$.
4) AGQ will utilize the label and the probability distribution to update the training data, and iterate to Step 1 (to continue learning actively).
   For example, from the generalized query $[1, *, 1, *, 0, 1]$ and the probability distribution for the class (0.9 for class 1 and 0.1 for class 0), four specific examples, $[1, 0, 1, 0, 0, 1]$, $[1, 0, 1, 1, 0, 1]$, $[1, 1, 1, 0, 0, 1]$, and $[1, 1, 1, 1, 0, 1]$, each with a probability label (0.9 for 1 and 0.1 for 0), could be added into the training set. This represents the power of generalized queries: each can represent effectively a set of specific queries. This would be useful if the probability of the majority class is high (close to 1). Otherwise, noise is introduced into the training set, and as we will show later, accuracy can even be worse. (We will study other strategies of utilizing the probabilistic labels in our future work.)

We will discuss each step in detail in the following subsections.

A. Finding the Most Uncertain Example

Similar to the previous works of the pool-based active learning, AGQ first builds a predictive model based on the current set of labeled examples, and uses it to predict each example in the pool. The most uncertain example from the pool, the one with the probability of the majority class closest to 50%, is chosen as the result of this first step.[3]

As the probability of the prediction is crucial in choosing the most uncertain example, we use an ensemble of decision trees in AGQ. Specifically, the bagging [19] of 100 j48 decision trees (implemented in Weka [20]) is used. The probability distribution of the prediction is estimated by the prediction of the 100 trees in the ensemble. Such an ensemble of many trees improves the probability estimation, compared with a single tree [21]. The standard decision tree algorithm is chosen because it tends to build small trees; this facilitates us to find irrelevant attributes in the next step.

B. Constructing the Generalized Query

After finding the most uncertain (specific) example from the pool in the first step, AGQ needs to discover the irrelevant attributes (don't-care attributes).

If the set of m attributes are irrelevant, then the examples with any combination of their values would have the same prediction with similar probability estimation. The reverse may not be true, but it can be used as a heuristic to find the set of irrelevant attributes. However, there are $\binom{n}{m}$ subsets of m attributes (given a total of n attributes), and for each subset, 2^m value combinations (for binary attributes) must be tested. The task is clearly computationally expensive.

A heuristic, similar to the process of finding the largest itemsets in mining association rules [22], [23], is designed. More specifically, let D be the current don't-care attribute list, and let x_u be the current most uncertain example. We gradually expand D by adding more irrelevant attributes via greedy search, as follows. For each attribute X_i not currently in D, we generate a fixed number (100 in our experiments) of examples with randomly assigned values for attributes in D and X_i, all based on x_u. The number of examples is fixed to prevent combinatorial explosion of attribute values when D grows. The attribute value is randomly chosen according to the distribution of that attribute values in the original data set. This most accurately reflects the distribution of examples

[3]For highly imbalanced and cost-sensitive data, an optimal threshold for classification can be calculated [17], or found via cross-validation [18], and the example with the probability closest to the threshold is chosen as the most uncertain one. Thus, our algorithm can also deal with imbalanced and cost-sensitive data.

Algorithm 1: find_don't-care_attributes

Input: x_u, the most uncertain example; θ, predefined threshold.
Output: D, don't-care attribute list.

```
Initialize D = ∅;
Initialize p_u = probability of majority class for x_u;
Initialize noChange = true;
repeat
    foreach X_i ∉ D do
        for n = 1 to 100 do
            begin // Generate x_n
                x_n = x_u;
                Randomly assign X_j for all X_j ∈ D;
                Randomly assign X_i;
            end
            p_n = probability of majority class for x_n;
        S_i = Σ_{n=1}^{100} (p_n − p_u)^2 / 100;
    Choose X_i with the smallest S_i;
    if S_i < θ then
        Add X_i in D;
    else
        noChange = false;
until noChange is false ;
```

in the domain.[4] The attribute X_i with the smallest change in the probability distribution of all 100 examples is then regarded as irrelevant, and added into D if the smallest change is less than a pre-defined threshold. The process continues until D cannot be grown further. The generalized query is the one with don't-care (i.e., "$*$") for all attributes in D. This process is depicted with the pseudo code in Algorithm 1.

Clearly, this can generate most general queries (i.e., queries with most don't-care attributes) based on the current learning model. However, queries with too many don't-care attributes can be overly general, and labels from the oracle can be highly uncertain. Thus, we demand the threshold θ in Algorithm 1 to be a very small number (0.0001 in our case). This would allow AGQ to find the most general queries that, hopefully, also include all relevant attributes. Still, as the initial labeled training set can be very small, the current learning model can be inaccurate. Thus, AGQ may produce generalized queries with don't-care for relevant attributes (see Table I in Section IV-A). This will be especially true when the initial labeled training set is very small. This would increase the uncertainty of the labels given by oracle. It would be an interesting future research to see how this can be prevented.

[4]The same random sampling method is used in Sections III-C and III-D.

C. Asking Generalized Queries to Oracle

In our work, we assume that the oracle can answer generalized queries with don't-care attributes just as easily as specific queries (without don't-care attributes). We believe that in most real-world situations, human experts can easily answer such generalized queries with an estimated probability.

In Section IV-B we will test AGQ on the UCI datasets [24], comparing it with the traditional pool-based active learner. An interesting question arises: as we do not know the target functions of the UCI datasets, nor do we have human oracles for them, how can such generalized queries be answered?

We design the following method to simulate human oracles to answer the generalized queries. We first train a model based on the original dataset to represent the target function. This is the best model we can get as it is built from the whole dataset. Specifically, we use the bagging of 100 j48 decision trees on the whole dataset to represent the target model. But still, this target model, as a black-box, cannot answer generalized queries directly. Since each generalized query effectively represents a set of specific queries, a set of such specific queries (in which the don't-care attributes are replaced with specific values sampled randomly) is generated. To avoid combinatorial explosion when the generalized query has too many don't-care attributes, the size of the set is fixed at 100. The target model then returns the predicted probability distribution of these 100 examples in the set.

One may argue that the generalized queries could be unrealistic thus hard to be answered by oracle (as in membership query). In the pool-based paradigm, AGQ chooses a specific example from the pool and generalizes it to a query. If the example is realistic, the generalized query is always realistic as well, so the oracle should be able to answer. For example, if the specific example is $[name = Jane, gentle = female, pregnant = yes, age = 30, \ldots]$, then the generalized query could be $[name = *, gentle = *, pregnant = yes, age = *, \ldots]$. Unrealistic generalized queries (such as $[name = *, gentle = male, pregnant = yes, age = *, \ldots]$) will never be constructed.

The next key step of AGQ is to utilize the generalized queries and their labels from the oracle to further improve learning.

D. Updating the Training Dataset

Given the probability distribution to the generalized query from the oracle, we need to utilize it to expand the training dataset and to build a better classifier. Again, because each generalized query effectively represents a set of specific queries, more than one specific example can be added into the original labeled training set. There are two issues to be resolved, however. One is how large the set of specific

queries should be; the second is how to label those examples in the set.

The first question is relatively easy to answer. Again to avoid combinatorial explosion, a set with a fixed size (100 in our experiments) of specific examples is generated first, in which each don't-care attribute is replaced randomly by a specific value of that attribute. However, experiments (Section IV-B) indicate that the number of new examples added may influence adversely the distribution of the initial training set. If the initial training set is too small, then the new examples added may be overwhelming, and thus changing the distribution of examples in the training set. Thus, the number of examples added into the training set is the minimum of 100, half of the size of the initial training set, and the number of value combinations of all don't-care attributes.

How should each specific query be labeled? As the oracle returns probability distribution of labels (such as 0.9 for positive, 0.1 for negative) for the generalized queries, specific examples can simply carry weighted labels if the learning model (bagging of 100 j48 trees here) can take weighted examples directly. Most learning algorithms (such as decision trees, naive Bayes, instance-based learning) can indeed take weighted examples naturally. Thus, in the above situation, every specific example carries a positive label with weight 0.9, and a negative label with weight 0.1.

Thus in AGQ, the labeled training set is usually increased by adding multiple labeled examples (with probability labels), rather than by adding just one labeled example in the traditional pool-based active learning. If examples added are mostly valid, and the probability of the majority class is near 1 (a highly certain label), the learning can be improved dramatically, as we will show in the experiments.

IV. Experiments with Generalized Queries

In this section, we conduct experiments on a synthetic dataset and 14 UCI [24] datasets to compare AGQ with the previous active learning algorithm that asks specific queries.

A. AGQ on Synthetic Dataset

In this subsection, we use synthetic data to empirically study the performance of AGQ, compared with the traditional pool-based active learning with uncertain sampling.[5]

In addition, we also present the performance of the *optimal* AGQ, which represents the best performance that AGQ could possibly achieve. Specifically, for each generalized query, the optimal AGQ gradually specifies the original attribute values for the don't-care attributes, till the oracle provides a certain answer ($P(Y = 1|X) \geq 0.95$ or $P(Y = 0|X) \geq 0.95$ in our experiments). The training set is thereafter expanded according to this query and the answer. That is, the training set is *only* updated when the oracle returns highly certain labels (≥ 0.95). However, some extra queries may still be asked to the oracle when the answer is not highly certain, which makes optimal AGQ not realistic. Here, we simply do not count those extra queries, and only count the "effective" ones — those with certainty great than (or equal to) 0.95. Thus, it could reflect the fewest number of queries that AGQ can ask, which indicates the best performance AGQ can ever achieve.

We choose the target function as a decision tree with five relevant attributes, $X1$ - $X5$, and six leaves, $L1$ - $L6$, as in Figure 1. To simulate the real-world dataset, we add another five *irrelevant* attributes, $X6$ - $X10$, to generate the synthetic data. We assume that all these attributes are binary, so is the class label. Therefore, with 10 binary attributes, we can generate $2^{10} = 1024$ different examples, and label them with the target function. With this synthetic data, we know what the target function is and what the irrelevant attributes are. We can also directly use the target function as the oracle to answer the generalized queries.

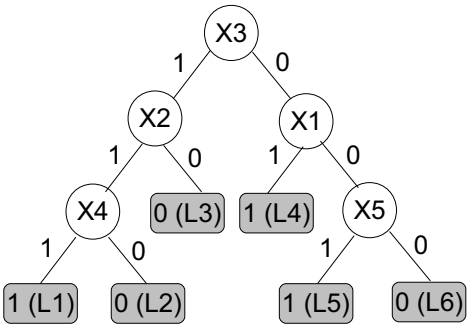

Figure 1. Target tree used to generate synthetic data.

The experiment is repeated on the synthetic dataset 20 times. Each time, the whole dataset is randomly split into three disjoint subsets: the training set, the unlabeled set, and the test set. The training set and the test set are always 2% and 25% of the whole dataset respectively, and the rest is the unlabeled set.

Figure 2 plots the average error rates of the optimal AGQ ("AGQ-Opt" in short), AGQ and the traditional pool-based active learning ("Pool" in short). We can see clearly from Figure 2 that, AGQ's performance is quite close to the (unrealistic) optimal AGQ, and is much better than "Pool". This indicates that the strategies we designed for AGQ (Section III) is quite effective — AGQ asks generalized queries with certain labels; that is, they are not overly general.

To further compare AGQ and "Pool", we extract a typical series of queries from them during the active learning process. Table I tabulates these queries (Query in the table), as

[5]Note that, as we also use a bagging of 100 decision trees for the traditional pool-based active learning (as same as for AGQ), the most uncertain example can also be considered as the example with the maximum disagreement for the current committee (constructed by the current 100 decision trees). Thus, uncertain sampling in this case can also be regarded as an implementation of QBC.

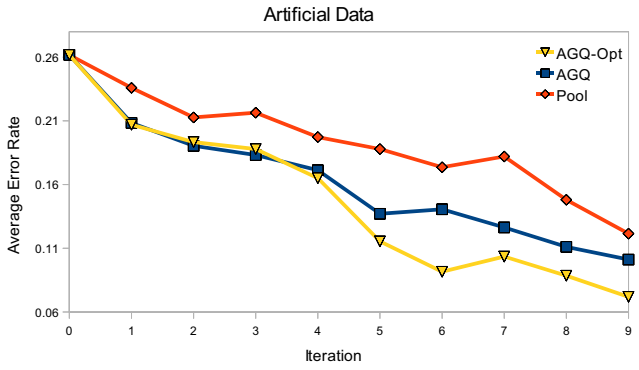

Figure 2. Comparison of the average error rate among "AGQ-Opt", AGQ and "Pool" on the synthetic data.

well as leaf(ves) in the target tree that these queries fall into (Classified by Leaf(ves)), ideal query according to the target tree (Ideal Query), answer from the oracle (Answer), number of specific examples generated to update the training set (No. of Examples), and error rate of the updated classifier (Error Rate). We can see from Table I that AGQ always constructs generalized queries with don't-care attributes while "Pool" can only choose the most specific queries. These generalized queries from AGQ may not be as general as the ideal queries (constructed directly from the target tree; see Figure 1), but they still contain most irrelevant attributes. Only one query (Query 2) is overly general (falling into two leaves), thus the answer to this query is highly uncertain (54%). However, such overly general queries rarely occur in AGQ learning. (Thus, the performance of AGQ is quite similar to the optimal AGQ, as we showed earlier.) In this case, answers for the other four queries from the oracle are highly certain (100%). Thus, AGQ can often include more examples with correct labels into the training set in each iteration, and obtain significantly lower error rates (compared with "Pool").

To summarize from the experiment on the synthetic data, AGQ can often identify correctly the irrelevant attributes and construct correctly the generalized queries with highly certain answers from the oracle. Thus the performance of the classifier is significantly improved when the corresponding multiple specific examples (with correct labels) are included into the training set. This yields the outstanding performance of AGQ (similar to the optimal AGQ) on the synthetic dataset, compared with the traditional pool-based active learning.

B. AGQ on UCI Datasets

In this subsection, we use 14 real-world datasets from the UCI Machine Learning Repository [24] to compare AGQ with the optimal AGQ and the pool-based active learning algorithm. All of these datasets have binary class and no missing values. Information on these datasets is tabulated in Table II.

Each whole dataset (D) is first split randomly into three disjoint subsets: the training set (R), the unlabeled set (U), and the test set (T). The test set T is always 25% of D. To make sure that active learning can possibly show improvement when the unlabeled data are labeled and included into the training set, we choose a small training set for each dataset such that the "maximum reduction" of the error rate[6] is large enough (greater than 10%). The training sizes of the 14 UCI datasets range from 1/200 to 1/5 of the whole datasets, also listed in Table II. The unlabeled set (U) is the whole dataset (D) taking away the test set (T) and the training set (R).

The experiment is repeated on each dataset 20 times (i.e., each dataset is randomly split 20 times), when comparing "AGQ-Opt", AGQ and "Pool". We stop training when the error rate of "Pool" is reduced by 3/4 of the "maximum reduction".

Figure 3 plots the average error rates of "AGQ-Opt", AGQ and "Pool" on a typical UCI datasets ("Hepatitis"), and the comparison on all the 14 datasets will be presented later. We can see from Figure 3 that, AGQ performs only slightly worse than "AGQ-Opt" but significantly better than "Pool", similar to the result on the synthetic dataset. This again clearly demonstrates the advantage of AGQ: AGQ performs almost as well as "AGQ-Opt", and significantly outperforms "Pool".

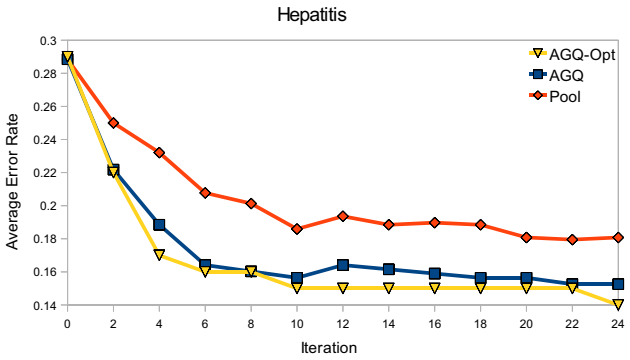

Figure 3. Comparison of average error rate among "AGQ-Opt", AGQ, and "Pool" on "Hepatitis".

In addition, the t-test (the paired two-tailed t-test with a 95% confidence level) on the average error rates based on the 14 UCI datasets shows that, AGQ wins on 9, ties on 4, and loses on 1 dataset, compared with "Pool". This clearly indicates that, with the same number of queries (same

[6]The "maximum reduction" of the error rate is the error rate on the initial training set R alone (without any benefit of the unlabeled examples) subtracts the error rate on R plus all the unlabeled data in U with correct labels. Thus, the "maximum reduction" reflects the upper bound on error reduction that active learning can achieve.

	AGQ	Pool
Query 1	[1, 1, 1, 0, *, *, *, *, *, *]	[1, 1, 1, 0, 1, 1, 1, 1, 0, 0]
Classified by Leaf(ves)	L2	L2
Ideal Query	[*, 1, 1, 0, *, *, *, *, *, *]	[*, 1, 1, 0, *, *, *, *, *, *]
Answer	0, 100%	0
No. of Examples	10	1
Error Rate	0.18	0.27
Query 2	[0, *, 0, 1, *, *, *, *, *, *]	[1, 0, 1, 1, 0, 0, 1, 0, 0, 1]
Classified by Leaf(ves)	L5, L6	L3
Ideal Query	-	[*, 0, 1, *, *, *, *, *, *, *]
Answer	0, 54%	0
No. of Examples	10	1
Error Rate	0.21	0.22
Query 3	[0, 1, 0, 1, 1, 0, 0, *, 1, *]	[1, 1, 1, 1, 0, 1, 1, 1, 0, 1]
Classified by Leaf(ves)	L5	L1
Ideal Query	[0, *, 0, *, 1, *, *, *, *, *]	[*, 1, 1, 1, *, *, *, *, *, *]
Answer	1, 100%	1
No. of Examples	8	1
Error Rate	0.16	0.26
Query 4	[0, 1, 0, 1, 0, 1, *, *, 0, *]	[1, 0, 1, 1, 0, 1, 0, 0, 1, 1]
Classified by Leaf(ves)	L6	L3
Ideal Query	[0, *, 0, *, 0, *, *, *, *, *]	[*, 0, 1, *, *, *, *, *, *, *]
Answer	0, 100%	0
No. of Examples	8	1
Error Rate	0.17	0.26
Query 5	[1, *, 0, *, 0, *, 1, *, *, *]	[1, 1, 1, 0, 0, 1, 0, 0, 1, 1]
Classified by Leaf(ves)	L4	L2
Ideal Query	[1, *, 0, *, *, *, *, *, *, *]	[*, 1, 1, 0, *, *, *, *, *, *]
Answer	1, 100%	0
No. of Examples	10	1
Error Rate	0.13	0.2

Table I
COMPARISON OF FIVE CONSECUTIVE QUERIES BETWEEN AGQ AND "POOL" ON SYNTHETIC DATA.

Dataset	Type of Attributes	No. of Attributes	No. of Examples	Class Distribution	Training Size
breast-cancer	nominal	9	277	196/81	1/5
breast-w	numeric	9	699	458/241	1/10
colic	nominal/numeric	22	368	232/136	1/5
credit-a	nominal/numeric	15	690	307/383	1/20
credit-g	nominal/numeric	20	1000	700/300	1/100
diabetes	numeric	8	768	500/268	1/10
heart-statlog	numeric	13	270	150/120	1/10
hepatitis	nominal/numeric	19	155	32/123	1/5
ionosphere	numeric	33	351	126/225	1/20
kr-vs-kp	nominal	36	3196	1669/1527	1/100
mushroom	nominal	22	8124	4208/3916	1/200
sonar	numeric	60	208	97/111	1/5
tic-tac-toe	nominal	9	958	332/626	1/10
vote	nominal	16	435	267/168	1/20

Table II
THE 14 UCI DATASETS USED IN THE EXPERIMENTS.

number of iterations), the error rate of AGQ decreases much faster than "Pool".

To further analyse the performance of AGQ and "Pool", we extract some important statistics during the active learning process. They include the average number of don't-care attributes (and its percentage of the total attributes) in each query (Don't-care Attributes in the table), the average certainty of the oracle (Certainty of Oracle)[7], average number of specific examples generated to update the training set in each iteration (Number of Examples), the average number of iterations of AGQ and "Pool" when their error rates are reduced by 3/4 of the "maximum reduction" (Iteration of AGQ and Iteration of "Pool"), percentage of iteration reduction between AGQ and "Pool" (% of Iteration

[7]The certainty of oracle, calculated from the oracle described in Section III, is always about the majority class (which can be either 1 or 0). Thus, the certainty value is between 0.5 and 1.

Reduction), and AGQ wins/ties/loses compared with "Pool" (AGQ w/t/l). Table III presents these statistics based on the 14 UCI datasets.

From Table III we can see that, on average, AGQ discovers 12.5 don't-care attributes, and includes 16.5 examples into the training sets in each iteration. Moreover, the certainty of the oracle for the constructed generalized queries is as high as 90.21% on average. This explains the good performance of AGQ: it can ask generalized queries, most with certain answers from the oracle. In the three datasets ("breast-w", "ionosphere" and "sonar") where AGQ ties with "Pool", we can notice that the certainties of the oracle are relatively low (87%, 86% and 73% respectively); this probably introduces more noise in the training sets, thus degrading the performance. In the dataset "tic-tac-toe" where AGQ also ties with "Pool", though the certainty of the oracle is high (100%), AGQ could only discover 0.07 don't-care attribute (on average), and include only 1.3 examples (on average) in each iteration. This is probably why AGQ is not much different from the traditional pool-based active learner. For the dataset "kr-vs-kp" where AGQ loses, the certainty of the oracle is relatively high (94%), and 39% of the attributes are discovered as don't-care in each query. So why does AGQ still lose to "Pool"? A detailed study shows that, "kr-vs-kp" is the Chess end-game board-positions, thus the attributes are highly constrained. As there are a total of 36 attributes, the dataset (containing about 3,000 examples) is very sparse; that is, only a small fraction of the attribute value combinations is valid. Thus, the examples generated by AGQ from the generalized queries and included into training set (Section III-D) are mostly invalid examples (i.e., meaningless board positions). These invalid examples may severely change the distribution of the original dataset thus degrading the performance of AGQ. We will study this issue further in our future work.

From Table III we can compare the number of iterations (queries) that AGQ and "Pool" have required to achieve 3/4 of the "maximum reduction" on the error rate. We notice that, on the four datasets where AGQ ties with "Pool", the two methods require almost the same number of iterations (queries). However, on the nine datasets where AGQ wins over "Pool", AGQ asks 61% fewer queries compared with "Pool". Over all 14 datasets, AGQ asks, on average, 36% fewer queries compared with "Pool". This clearly shows the advantage of AGQ: it requires much fewer queries than "Pool" on the tested UCI datasets.

To summarize, AGQ performs significantly better than "Pool" on most UCI datasets (9 out of 14). Moreover, on those datasets where AGQ wins, it requires 61% fewer queries needed for "Pool" to achieve the same error rate reduction. This clearly demonstrates the power of the generalized queries and the advantage of AGQ.

V. Conclusions and Future Works

Previous active learning algorithms assume that the oracle can only answer specific queries that represent single examples. However, in real-world applications, the oracles are often more readily to answer "generalized queries" with don't-care attributes. Answers to such generalized queries can provide more information to improve learning. The difficulty of generalized queries is that the answers from the oracle can be uncertain, thus noisy labels might be introduced and performance might be degraded. This easily happens especially when the initial labeled training set is small. In this paper, we propose a novel active learning algorithm (AGQ) to ask as general queries as possible with still highly certain labels. Our experiments show that, compared with the traditional pool-based active learning, AGQ can achieve the same error rates with significantly fewer queries (36% fewer on average). We also show that AGQ's performance is similar to the (unrealistic) optimal AGQ. AGQ can be readily deployed in real-world data mining tasks where obtaining labeled examples is costly.

In our future research, we will study the performance of AGQ with different base learning algorithms (we only use the bagging of decision trees in this paper). Strategies for dealing with highly uncertain answers from the oracle, and for preventing dramatic changes of data distribution when new examples are included in the training set are also interesting research issues to further improve the performance of AGQ.

Acknowledgment

The authors acknowledge the valuable assistance of other members of the Data Mining and E-Business Lab of The University of Western Ontario in this research.

References

[1] D. A. Cohn, L. Atlas, and R. E. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[2] Y. Baram, R. El-Yaniv, and K. Luz, "Online choice of active learning algorithms," *Journal of Machine Learning Research*, vol. 5, pp. 255–291, 2004.

[3] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Proceedings of ICML-94, 11th International Conference on Machine Learning*, W. W. Cohen and H. Hirsh, Eds. New Brunswick, US: Morgan Kaufmann Publishers, San Francisco, US, 1994, pp. 148–156.

[4] D. Angluin, "Queries and concept learning," *Machine Learning*, vol. 2, no. 4, pp. 319–342, April 1988.

[5] C. X. Ling and J. Du, "Active learning with direct query construction," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2008, pp. 480–487.

Dataset	Don't-care Attributes (% of Total Attributes)	Number of Examples	Certainty of Oracle	Iteration of "Pool"	Iteration of AGQ	% of Iteration Reduction	AGQ (w/t/l)
breast-cancer	2.7 (30%)	14.54	95%	35	18	49%	W
breast-w	5.35 (59%)	32.31	87%	18	18	0%	T
colic	13.15 (60%)	35.68	91%	15	8	47%	W
credit-a	6.38 (43%)	16.43	88%	12	5	58%	W
credit-g	8.54 (43%)	4.97	87%	50	12	76%	W
diabetes	3.02 (38%)	27.31	89%	50	16	68%	W
heart-statlog	5.92 (46%)	12.52	89%	50	25	50%	W
hepatitis	13.47 (71%)	14.96	96%	24	5	79%	W
ionosphere	27.15 (82%)	8	86%	29	29	0%	T
kr-vs-kp	14.89 (39%)	14.48	94%	38	50	-32%	L
mushroom	17.81 (81%)	20	94%	10	6	40%	W
sonar	48.27 (80%)	20	73%	41	34	17%	T
tic-tac-toe	0.07 (1%)	1.28	100%	108	108	0%	T
vote	7.28 (46%)	8.31	94%	12	5	58%	W
Average	12.53 (51.36%)	16.49	90.21%	35.14	24.21	36%	9/4/1

Table III
IMPORTANT STATISTICS OF AGQ AND COMPARISON WITH "POOL" ON THE 14 UCI DATASETS.

[6] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[7] M. Saar-Tsechansky and F. Provost, "Active sampling for class probability estimation and ranking," *Machine Learning*, vol. 54, no. 2, pp. 153–178, February 2004.

[8] M. Lindenbaum, S. Markovitch, and D. Rusakov, "Selective sampling for nearest neighbor classifiers," *Machine Learning*, vol. 54, no. 2, pp. 125–152, February 2004.

[9] D. D. Margineantu, "Active cost-sensitive learning," in *the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[10] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*. New York, NY, USA: ACM Press, 1992, pp. 287–294.

[11] N. Abe and H. Mamitsuka, "Query learning strategies using boosting and bagging," in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 1–9.

[12] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.

[13] T. Zhang and F. J. Oles, "A probability analysis on the value of unlabeled data for classification problems," in *Proc. 17th International Conf. on Machine Learning*, 2000, pp. 1191–1198.

[14] N. Roy and A. Mccallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2001, pp. 441–448.

[15] G. Druck, G. S. Mann, and A. McCallum, "Learning from labeled features using generalized expectation criteria," in *SIGIR*, S. H. Myaeng, D. W. Oard, F. Sebastiani, T. S. Chua, M. K. Leong, S. H. Myaeng, D. W. Oard, F. Sebastiani, T. S. Chua, and M. K. Leong, Eds. ACM, 2008, pp. 595–602.

[16] L. S. Dasgupta, "Coarse sample complexity bounds for active learning," in *Neural Information Processing Systems*, 2005.

[17] C. Elkan, "The foundations of cost-sensitive learning," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001, pp. 973–978.

[18] V. S. Sheng and C. X. Ling, "Thresholding for making classifiers cost-sensitive," in *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, 2006.

[19] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[20] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., ser. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, June 2005.

[21] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Machine Learning*, vol. 52, no. 3, pp. 199–215, September 2003.

[22] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, P. Buneman and S. Jajodia, Eds., Washington, D.C., FebruaryJune–FebruaryAugust˜ 1993, pp. 207–216.

[23] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *Knowledge Discovery and Data Mining*, 1998, pp. 80–86.

[24] A. Asuncion and D. J. Newman, "UCI machine learning repository [http://www.ics.uci.edu/˜mlearn/mlrepository.html]," 2007.