Asking Generalized Queries to Ambiguous Oracle

Jun Du and Charles X. Ling

Department of Computer Science
The University of Western Ontario, London, Ontario, N6A 5B7, Canada
{jdu42,cling}@csd.uwo.ca

Abstract. Asking generalized queries (by regarding some features as don't-care) in active learning has been proposed and studied recently. As each generalized query is equivalent to a set of specific ones, the answers from the oracle can usually provide more information thus speeding up the learning effectively. However, as the answers to the generalized queries might be uncertain, previous studies often assume that the oracle is capable of providing (accurate) probabilistic answers. This assumption, however, is often too stringent in real-world situations. In this paper, we make a more realistic assumption that the oracle can only provide (non-probabilistic) ambiguous answers, similar to the setting in *multiple-instance learning*. That is, the generalized query is labeled positive if at least one of the corresponding specific queries is positive, and is labeled negative otherwise. We therefore propose an algorithm to construct the generalized queries and improve the learning model with such ambiguous answers in active learning. Empirical study shows that, the proposed algorithm can significantly speed up the learning process, and outperform active learning with either specific queries or inaccurately answered generalized queries.

1 Introduction

Active learning, as an effective paradigm to speed up the learning process, has been intensively studied in recent years. In most traditional active learning studies, the learner usually regards the specific examples directly as queries, and requests the corresponding labels from the oracle. For instance, given a prostatitis patient data set, the learner usually presents the entire patient example, such as $\{\text{ID} = 7354288, \text{name} = \text{John}, \text{age} = 50, \text{gender} = \text{male}, \text{weight} = 230, \text{blood-type} = \text{AB}, \text{fever} = \text{yes}, \text{painful-urination} = \text{yes}, \dots\}$ (with all the features), to the oracle, and requests the corresponding label whether this patient has prostatitis or not. However, in this case, many features (such as ID, name, blood-type, and so on) might be *irrelevant* to prostatitis diagnosis. Not only could queries like this confuse the oracle, but each answer responded from the oracle is also applicable for only one specific example.

In many real-world active learning applications, the oracles are often human experts, thus they are usually capable of answering more general queries. For

instance, given the same patient data set, the learner could ask a generalized query, such as “are men between 45 and 55 with fever and painful-urination likely to have prostatitis?”, where only four features (gender, age, fever and painful-urination) are provided, and all the rest are considered as don’t-care. Not only are such generalized queries more natural and relevant, each generalized query can often represent a set of specific examples, thus the answer for the query is also applicable to all these examples. This allows the active learner to improve learning more effectively.

However, when asking such generalized queries in active learning, the answers are often uncertain. For instance, the answer could be “yes, with 80% probability” that men between 45 and 55 with fever and painful-urination are likely to have prostatitis. In the ideal situation, the oracle is expected to provide such accurate probabilistic answers for the generalized queries, in order to improve the learning model accordingly. This requirement, however, is usually too stringent in reality. Instead, generalized queries are often responded with ambiguous answers in real-world situations. For instance, if the query is “are women (or boys under 10) likely to have prostatitis”, the specialist (oracle) would always respond “No”, indicating none of such people would have this disease. However, if the query is “are men between 45 and 55 likely to have prostatitis”, the answer would often be “Yes”, indicating some of such people indeed have this disease, but the accurate proportion (probability) is unknown. Clearly, such generalized queries and ambiguous answers commonly occur in our daily life, thus it is reasonable and desired to study them together.

In this paper, therefore, we make a more realistic assumption that the oracle can only provide ambiguous (non-probabilistic) answers to the generalized queries. That is: the oracle labels the query as negative *if (and only if)* all the examples represented by this query are negative; otherwise the query is always labeled as positive. The similar setting of ambiguous answers has been extensively studied in *multiple-instance learning* [1], and applied to many real-world applications, such as drug activity prediction [1], content-based image retrieval [2] and text categorization [3]; see Section 2 for more details.

In active learning scenario, such setting of ambiguous answers is reasonable yet flexible. On one hand, even though the oracle is still required to answer generalized queries, the answer only needs to be “Yes (positive)” or “No (negative)”, which is more natural and applicable in real-world situations. On the other hand, such ambiguous answers can also be regarded as a more general form of the specific (accurate) answers. Specifically, when some features are discovered as don’t-care and the query is generalized, the answer is indeed ambiguous (“Yes” indicates at least one specific example is positive, whereas “No” indicates all corresponding examples are negative). However, when no don’t-care feature is discovered and the query turns to be a specific one, such “Yes-No” response naturally becomes the *accurate* answer to the specific query. Therefore, such setting of ambiguous answers is more flexible than the regular setting in active learning, yet still applicable in many real-world situations.

In this paper, by assuming that the oracle is capable of providing such ambiguous answers to the generalized queries, we propose a novel method to, first construct the generalized queries, and then update the learning model according to the ambiguous answers. Empirical study on UCI ([4]) data sets shows that, the proposed method can significantly speed up the learning process, and outperform active learning with either specific queries or inaccurately answered generalized queries.

The rest of the paper is organized as follows. Section 2 reviews previous works on active learning and multiple-instance learning. Section 3 describes our algorithm to ask generalized queries and improve the learning model with ambiguous answers. In Section 4, empirical study is conducted on real-world UCI data sets to verify the superiority of the proposed method. Section 5 presents conclusions.

2 Related Work

Most previous studies of active learning can be categorized into two types: the pool-based active learning and the membership query.¹ In the pool-based active learning, a pool of unlabeled examples is given, among which the learner can choose the examples and request the corresponding labels [7]. Briefly speaking, the pool-based active learner 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 the oracle, and update the learning model accordingly. The process repeats, until some stop criterion is met. On the other hand, active learning with membership queries (or direct query construction) can directly construct examples (without the need of the pool) and request the corresponding labels from the oracle [8,9]. Previous research shows that, in most situations, both of these two types of active learning can significantly reduce the number of labeled examples needed, compared with labeling examples randomly.

Without the restriction of the pool, membership query can always construct the *optimal* queries to improve the performance of the current model. However, in reality, these *optimal* queries might be unrealistic thus hard to be answered by the oracle (such as, handwritten character recognition, text classification, and so on). Therefore, the pool-based active learning is extensively studied in recently years. In this paper, the proposed learning algorithm can be adapt to both of these two types, and we will use pool-based setting for illustration.

The essence of active learning lies in measuring the “goodness” of the unlabeled examples. Many criteria have been proposed in the literature. *Uncertainty sampling* [7] considers the most uncertain example as most valuable, and has been extensively studied and widely used in many previous studies [10,11,6,12,13]. *Query-by-committee* (QBC) [14] is a more theory-based approach, and considers the example that minimizes the version space as optimal. In addition, other criteria, such as *variance reduction* [15], *Fisher information ratio* [16], and *estimated error reduction* [17], are also elaborately designed and well accepted.

¹ Stream-based active learning [5] is considered as another type in some literatures. In essence, it could be viewed as an online version of the pool-based active learning [6].

Most previous works of active learning assume that the oracle could only answer specific queries, with all features provided. [18,19] consider a more natural situation that the oracle is capable of answering generalized queries, and propose a novel algorithm to ask such queries and improve the learning model. However, as the answers for such generalized queries are often uncertain, it is also assumed in [18,19] that the oracle could provide (accurate) probabilistic answers to those queries. This assumption, however, is too stringent in many real-world situations.

On the other hand, a more relaxed assumption has been studied in *multiple-instance learning* ([1]), where examples are given in bags and oracle is only required to provide one ambiguous answer for each bag. More specifically, given a bag of unlabeled examples, the oracle will respond negative *if (and only if)* these examples are all negative, and respond positive otherwise. In this setting, it is more likely for the learner to be responded with a positive answer; and more importantly, such positive answer only indicates that at least one example in the given bag is positive, but the true label of each specific example is still unknown. Many algorithms, such as *diverse density* [20], *citation kNN* [21], *multiple-decision tree* [22] and *multiple-instance logistic regression* [23], have been developed to tackle such ambiguous answers, so as to predict labels of the future unseen bags. Despite of the ambiguity of the answers, multiple-instance learning has been applied to many real-world applications, such as drug activity prediction [1], content-based image retrieval [2], and text categorization [3].

In this paper, we apply such ambiguous oracle to active learning with generalized queries. More specifically, in active learning, the learner always tends to construct generalized queries and request the corresponding labels from the oracle. However, the oracle is only capable of respond with ambiguous answers. That is, given a generalized query, the oracle will respond negative *if (and only if)* the examples represented by the query are all negative, and respond positive otherwise. Such setting of ambiguous oracle relaxes the stringent assumptions in the previous studies of active learning with generalized queries, and is applicable to more real-world situations.

It is also worth noting that, our study in this paper is not a simple combination of active learning and multiple instances learning.² Instead of aiming to improve the predictive performance on the *unseen bags of examples* in multiple-instance learning, in this paper, we still attempt to improve the predictive performance on the *unseen specific examples* (as in traditional supervised setting). In addition, we also consider generalized queries in active learning scenario, thus the problem we are attempting to solve is more complex and difficult.

3 Algorithm

In this section, we design active learning algorithm to ask generalized queries and further improve the predictive performance based on the responded ambiguous answers. Specifically, we use *logistic regression* as the base active learner, due to

² [24] categorizes multiple-instance active learning into four scenarios, and develops a novel algorithm to deal with one of those formulations.

its good performance in probability estimation and the convenience of designing objective function (see Section 3.1 for details).

The learning process can be roughly broken into the following two steps in each iteration:

- **Step 1:** Based on the current training and unlabeled data sets, the learner constructs a generalized query (according to certain objective function).
- **Step 2:** After obtaining the ambiguous answer of the generalized query, the learner updates the learning model (according to certain objective function).

In the above two steps, objective functions are required for both constructing the generalized queries and updating the learning model. Therefore, in the rest of this section, we first design a *universal* objective function for both of the above two steps; and then present the implementation details for each of them.

3.1 Objective Function

In each learning iteration, when constructing the generalized query, the optimal query is expected to be the one that yields the best performance of the learning model; likewise, when updating the learning model, the optimal model parameters are also the ones that yield the best predictive performance. Therefore, we can design one universal objective function to evaluate and optimize both the generalized queries and the model parameters.

In the current setting, the desired objective function needs to suit all of the following requirements: 1) logistic regression; 2) active learning; 3) generalized queries; and 4) ambiguous answers.

In the traditional supervised learning, *maximum likelihood* is commonly used to train a logistic regression classifier. Thus, it could also be considered as the most primitive objective function to find the optimal queries and model parameters, as follows:

$$\langle q, \mathbf{w} \rangle_{opt} = \arg \max_{q, \mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q, \mathbf{w}) \quad (1)$$

where, $\langle q, \mathbf{w} \rangle_{opt}$ denotes the tuple of optimal query q and model parameter $\mathbf{w}$, $\mathbf{x}_i$ and y_i denote the i th example in the given training data set D . This primitive objective function satisfies the requirement of logistic regression.

In active learning, however, the labeled training data set is usually small, thus the classifier trained via maximum likelihood alone (Equation 1) might be unreliable. In addition to the training data, we are often given a large amount of unlabeled data in active learning (for the pool-based setting), and this set of data can also help to evaluate the generalized queries and the model parameters. Intuitively, if the query (q) can indeed improve the performance of the learning model, the updated model would also be more confident in predicting all the unlabeled data; likewise, if the parameter ($\mathbf{w}$) can indeed yield a high-performance model, the model would also predict unlabeled data more confidently. Therefore, in addition to the maximum likelihood on the labeled training data, the predictive certainty (calculated by *entropy*) on the unlabeled data can also be

considered as an additional measurement. This yields a more sophisticated objective function, as follows:

$$\begin{aligned}
\langle q, \mathbf{w} \rangle_{opt} &= \arg \max_{q, \mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q, \mathbf{w}) - \alpha \sum_{\mathbf{x}_j \in U, y \in \{0,1\}} H(p(y | \mathbf{x}_j; q, \mathbf{w})) \\
&= \arg \max_{q, \mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q, \mathbf{w}) \\
&\quad + \alpha \sum_{\mathbf{x}_j \in U} \sum_{y \in \{0,1\}} p(y | \mathbf{x}_j; q, \mathbf{w}) \log p(y | \mathbf{x}_j; q, \mathbf{w})
\end{aligned} \tag{2}$$

where $\mathbf{x}_j$ denotes the j th unlabeled example in the given unlabeled data set U , and α represents a trade-off parameter to balance the influence of the labeled and unlabeled data. This objective function satisfies the requirement of active learning.³

In our current setting, however, instead of requesting the labels for specific examples, the active learner always attempts to ask generalized queries. Moreover, these generalized queries are often responded with ambiguous answers by the oracle. Equation 2 therefore cannot suit this requirement. Instead, under current conditions, in each learning iteration, there always exist three data (query) sets: the initial training data set D , the query set Q which contains all the previous queries asked by the learner (one in each iteration), and the current unlabeled data set U . Therefore, in the t th learning iteration, the query q_t and the model parameter $\mathbf{w}_t$ can be optimized by: maximizing the likelihood with respect to both the initial training data D and the query set Q , and minimizing the predictive uncertainty with respect to the unlabeled data U , as follows:

$$\begin{aligned}
\langle q_t, \mathbf{w}_t \rangle_{opt} &= \arg \max_{q, \mathbf{w}} \alpha_1 \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q_t, \mathbf{w}_t) \\
&\quad + \alpha_2 \sum_{(q_k, y_k) \in Q} \log p(y_k | q_k; q_t, \mathbf{w}_t) \\
&\quad + \alpha_3 \sum_{\mathbf{x}_j \in U} \sum_{y \in \{0,1\}} p(y | \mathbf{x}_j; q_t, \mathbf{w}_t) \log p(y | \mathbf{x}_j; q_t, \mathbf{w}_t)
\end{aligned} \tag{3}$$

where all the notations are the same as in the previous objective functions, and α_1 , α_2 and α_3 represent the trade-off parameters to balance the influence of the three data (query) sets.

This objective function suits all the requirements in our current setting, and is applied to both query searching and model updating in each learning iteration, as follows.

- In the query searching step, the objective function is applied to find the optimal query (q_{opt}). Specifically, given one candidate query (and the estimated label, see Section 3.2 for details), Equation 3 can be regarded as a univariate

³ The similar objective function has been applied in semi-supervised learning [25] and batch mode active learning [26]; and these previous studies have shown its applicability in real-world applications.

Fig. 1. The framework of the proposed algorithm

function of the model parameter $\mathbf{w}$ (denoted by $f(\mathbf{w})$). Thus, gradient descent can be directly applied to find the optimized $f(\mathbf{w})$. Thereafter, among all the candidate queries, the one that yields the maximum $f(\mathbf{w})$ is chosen and regarded as the optimal query (q_{opt}).

- In the model updating step, the objective function is applied to find the optimal model parameter ($\mathbf{w}_{opt}$). Specifically, given the optimal query (q_{opt}) and the true label provided by the oracle, Equation 3 is optimized in a similar way, and the optimal model parameter ($\mathbf{w}_{opt}$) can be determined.

This entire process is also illustrated in Figure 1. However, there are still some issues to be solved during this learning process, such as, how to search the candidate queries, how to estimate the posterior probability for each specific example ($p(y|\mathbf{x})$) and each generalized query ($p(y|q)$), how to set the trade-off parameters (α_1 , α_2 and α_3) and so on. We will describe the details and answer these questions in the following sections.

3.2 Constructing Generalized Queries

In each active learning iteration, constructing the optimal generalized query can be implemented by searching the best one in the query space (according to Equation 3). However, two issues need to be solved at this stage: how to search in the query space, and how to estimate the labels for the candidate queries. We will provide solutions for these two problems in this subsection.

In most traditional active learning studies, each unlabeled example is directly regarded as a candidate query. Thus, in each iteration, the query space simply contains all the current unlabeled examples, and exhaustive search is usually applied directly. However, when asking generalized queries, each unlabeled example can generate a set of candidate generalized queries, due to the existence of the don't-care features. For instance, given a specific example with d features, there exist $\binom{d}{1}$ generalized queries with one don't-care feature, $\binom{d}{2}$ generalized queries

with two don't-care features, and so on. Thus, altogether 2^d corresponding generalized queries could be constructed from each specific example. Therefore, given an unlabeled data set with l examples, the entire query space would be 2^{dl} . This query space is quite large (grows exponentially to the feature dimension), thus exhaustively evaluating every candidate is no longer realistic. Instead, we apply greedy search to find the optimal query in each iteration.

Specifically, for each unlabeled example (with d features), we first construct all the generalized queries with only one don't-care feature (i.e., $\binom{d}{1} = d$ queries), and choose the best as the current candidate. Then, based only on this candidate, we continue to construct all the generalized queries with two don't-care features (i.e., $\binom{d-1}{1} = d-1$ queries), and again only keep the best. The process repeats to greedily increase the number of don't-care features in the query, until no better query can be generated. The last generalized query thus is regarded as the best for the current unlabeled example. We conduct the same procedure on all the unlabeled examples, thus we can find the optimal generalized query based on the whole unlabeled set.

With such greedy search strategy, the computation complexity of searching is thus $O(d^2)$ with respect to the feature dimension d . This indicates an exponential improvement over the complexity of the original exhaustive search ($\Theta(2^d)$). Note that, it is true that such local greedy search cannot guarantee finding the *true* optimal generalized query in the entire query space, but the empirical study (see Section 4) will show it still works effectively in most cases.

With greedy search, all these candidate queries are expected to be evaluated by Equation 3, such that the optimal one could be determined. Note that, if the true labels for these candidate queries are known, the evaluation process can be implemented in exactly the same way as we will describe in Section 3.3. However, all the candidate queries are not yet labeled at the current stage, thus the objective function cannot be directly applied. Here, we use a simple strategy to estimate the label probabilities of these queries, and then evaluate them accordingly.

Specifically, given a specific query (with no don't-care feature), we simply assume that it is equally likely to be labeled positive or negative, thus we evaluate the query by regarding its label as 0.5 positive and 0.5 negative. However, in terms of the generalized queries with don't-care features, as they are expected to be responded with ambiguous answers (negative if all the examples represented are negative, and positive otherwise), we also attempt to estimate the probability of such ambiguous answer. More specifically, we suppose that each generalized query with n don't-care features can be represented by 2^n specific examples,⁴ and each of these specific example is still equally likely to be positive or negative. Thus the probability of such generalize query being negative would be calculate as 0.5^{2^n} (i.e., the probability of all the examples represented being negative), and the probability of being positive would consequently be $(1 - 0.5^{2^n})$. Therefore,

⁴ See Section 3.3 for details why and how each generalized query can be represented by this many examples.

for each candidate query, we apply this simple strategy to estimate its label probability and make the evaluation accordingly.

To summarize, in each learning iteration, we use greedy search to select candidate queries from the entire query space, and evaluate these candidates (with the estimated labels) according to Equation 3. The optimal query thus could be discovered.

3.3 Updating Learning Model

After discovering the optimal generalized query in each iteration, the active learner requests the corresponding label from the oracle, and then updates the learning model (i.e., optimize the model parameter $\mathbf{w}$ according to Equation 3). However, as the active learner always tends to ask generalized queries, and is always responded with ambiguous answers, the objective function is difficult to be directly specified.

Specifically, with logistic regression, the posterior probability for each example $\mathbf{x}$ can be specified as

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} \quad (4)$$

Therefore, in Equation 3, the part to maximize the log likelihood on the original training data D (i.e., the first term in Equation 3) is easy to calculate, so is the part to minimize the predictive uncertainty (entropy) on the unlabeled data U (i.e., the last term in Equation 3). However, it is difficult to specify the posterior probability for all the previous queries (i.e., $p(y|q)$, as in the middle term of Equation 3), due to the generalization of these queries and the ambiguity of the answers. We will solve this issue in this subsection.

The basic idea to estimate the posterior probability for each query ($p(y|q)$) works as follows:

- We first specify each generalized query with a set of representative examples, the posterior probability for each of these examples thus can be presented as in Equation 4.
- Then, we combine the probabilities of all these representative examples together, to form the probability for the corresponding generalized query.

More specifically, as generalized queries contain don't-care features, each generalized query can often represent a set of specific examples. For instance, if “temperature” is the don't-care feature in a generalized query, this query represents infinite examples with *any* temperature values (while keeping other features unchanged). Therefore, it seems difficult to find appropriate representative examples to specify the generalized queries.

However, in our current setting, the classifier is always a linear separator (due to logistic regression) and the labels of the generalized queries are negative if (and only if) *all* the corresponding examples are negative (due to ambiguous oracle). Under these conditions, we can have an intuition that, given any

generalized query with one don't-care feature, if (and only if) the corresponding example with the maximum value (for the don't-care feature) and the one with the minimum value are both negative, the generalized query will be labeled negative. For instances, we suppose "temperature" is the only don't-care feature, and its valid range is $[94F, 108F]$. Thus, a generalized query $\{\text{age} = 65, \text{gender} = \text{male}, \text{temperature} = *, \dots\}$ will definitely be labeled as negative, if (and only if) the two specific examples, $\{\text{age} = 65, \text{gender} = \text{male}, \text{temperature} = 94F, \dots\}$ and $\{\text{age} = 65, \text{gender} = \text{male}, \text{temperature} = 108F, \dots\}$, are both negative. This intuition can be illustrated in Figure 2.

This figure illustrates the label of a generalized query $\{x_1 = 2, x_2 = *\}$ (where the valid range of x_2 is $[-1, 2]$) with respect to three linear separators. Line "Query" denotes all the examples represented by the generalized query, where "A" $\{x_1 = 2, x_2 = 2\}$ and "B" $\{x_1 = 2, x_2 = -1\}$ represent two specific examples with the maximum and minimum values for the don't-care feature x_2 . Lines " $f_1(x_1, x_2)=0$ ", " $f_2(x_1, x_2)=0$ " and " $f_3(x_1, x_2)=0$ " represent three linear separators in the given 2-D space, and all of them can only provide ambiguous answers for the queries. We can clearly see that, " $f_3(x_1, x_2)=0$ " can simultaneously label both "A" and "B" as negative^a, it consequently labels the query as negative; whereas both " $f_1(x_1, x_2)=0$ " and " $f_2(x_1, x_2)=0$ " always label at least one of "A" and "B" as positive, they label the query as positive consequently.

^a We suppose all the examples *above* the linear separators are positive, and the ones *below* are negative.

Fig. 2. An illustration for representing generalized queries with specific examples

This illustration indicates that, given a generalized query with one don't-care feature, as long as the labels of the two specific examples (with maximum and minimum values for the don't-care feature) are known, the label for query can be easily determined.⁵ Therefore, we can simply represent such generalized query by these two specific examples in the learning process. Furthermore, we can extend the conclusion that, given a generalized query with two don't-care features, four examples with the combinations of the min and max values for the two features could be used to represent the query; and further, a generalized query with n don't-care features would be specified by 2^n examples.

Now, we are able to specify any generalized query with a set of representative examples, and the posterior probability for these specific examples can

⁵ Note that, in active learning, the minimum and maximum values of any feature can be reliably estimated from both the labeled and unlabeled data.

also be presented through Equation 4. Next, we will combine all these probabilities together, to form the posterior probability for the corresponding generalized query.

As we have introduced in Section 1, in the current setting, the ambiguous oracle provides negative answer only when all the corresponding specific examples are negative, and provides positive answer otherwise. This mechanism is similar to the labeling process in multi-instance learning (MIL). Therefore, we can simply apply the existing combining functions in MIL, to form the probabilities of the generalized queries. More specifically, we will apply the noise-or function [20]

$$p(y = 1|q) = 1 - \prod_{k=1}^n (1 - p(y = 1|\mathbf{x}_k)) \quad (5)$$

to form the probability of the generalized query, where q denotes the generalized query and $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ denotes the corresponding n representative examples. The probability for any generalized query therefore can be specified.⁶

To formalize, given the optimal generalized query (and the true ambiguous label) in each iteration, by combining Equations 3, 4 and 5, the optimal model parameter $\mathbf{w}$ can be determined by minimizing the following error function:

$$\begin{aligned} E(\mathbf{w}) = & -\alpha_1 \sum_{(\mathbf{x}_i, y_i) \in D} \{y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))\} \\ & -\alpha_2 \sum_{(q_k, y_k) \in Q} \{y_k \log(1 - \prod_{\mathbf{x}_{ki} \in q_k} (1 - \sigma(\mathbf{w}^T \mathbf{x}_{ki}))) + (1 - y_k) \sum_{\mathbf{x}_{ki} \in q_k} \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_{ki}))\} \\ & -\alpha_3 \sum_{\mathbf{x}_j \in U} \{\sigma(\mathbf{w}^T \mathbf{x}_j) \log \sigma(\mathbf{w}^T \mathbf{x}_j) + (1 - \sigma(\mathbf{w}^T \mathbf{x}_j)) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_j))\} \end{aligned} \quad (6)$$

where $\sigma(\mathbf{w}^T \mathbf{x}_i)$ can be further specified by Equation 4, and $\mathbf{x}_{ik}$ denotes the representative example for query q_k . In addition, gradient decent is applied to implement optimization, and the gradient of the error function with respect to $\mathbf{w}$ can be calculated:

$$\begin{aligned} \nabla E(\mathbf{w}) = & -\alpha_1 \sum_{(\mathbf{x}_i, y_i) \in D} \{(y_i - \sigma(\mathbf{w}^T \mathbf{x}_i)) \mathbf{x}_i\} \\ & -\alpha_2 \sum_{(q_k, y_k) \in Q} \left\{ \frac{(1 - \prod_{\mathbf{x}_{ki} \in q_k} (1 - \sigma(\mathbf{w}^T \mathbf{x}_{ki}))) - y_k}{1 - \prod_{\mathbf{x}_{ki} \in q_k} (1 - \sigma(\mathbf{w}^T \mathbf{x}_{ki}))} \sum_{\mathbf{x}_{ki} \in q_k} \sigma(\mathbf{w}^T \mathbf{x}_{ki}) \mathbf{x}_{ki} \right\} \\ & -\alpha_3 \sum_{\mathbf{x}_j \in U} \left\{ \left(\log \frac{\sigma(\mathbf{w}^T \mathbf{x}_j)}{1 - \sigma(\mathbf{w}^T \mathbf{x}_j)} \right) \sigma(\mathbf{w}^T \mathbf{x}_j) (1 - \sigma(\mathbf{w}^T \mathbf{x}_j)) \mathbf{x}_j \right\} \end{aligned} \quad (7)$$

Consequently, in each learning iteration, given the optimal generalized query and the ambiguous answer, the optimal model parameter can be obtained, and the learning model can be updated.

⁶ Note that, other combining functions, such as the softmax function [23] $p(y = 1|q) = \frac{\sum_{k=1}^n p(y=1|\mathbf{x}_k) e^{p(y=1|\mathbf{x}_k)}}{\sum_{k=1}^n e^{p(y=1|\mathbf{x}_k)}}$, can also be applied to form the probability of the generalized queries.

In the next section, we will conduct empirical study to discuss the setting for the trade-off parameters α_1 , α_2 and α_3 , and compare the proposed algorithm with the traditional ones.

4 Empirical Study

In this section, we empirically study the performance of the proposed algorithm with generalized queries and ambiguous answers, and compare it with the existing active learning algorithms on seven real-world data sets from UCI Machine Learning Repository [4].

4.1 Experimental Configurations

We conduct experiments with two settings of the trade-off parameters (α_1 , α_2 and α_3) for Equation 3. Specifically, we first consider a uniform parameter setting, that is, $\alpha_1 = \alpha_2 = \alpha_3$ (the corresponding algorithm is denoted by “AL-GQA(u)”). We can notice from Equation 3 that, all the three terms (i.e., log likelihood on D , log likelihood on Q , and predictive entropy on U) are specified by the summation of the examples (queries) in each corresponding set. It therefore indicates that, with uniform trade-off parameters, those three terms are implicitly weighed by the number of examples (queries) in the corresponding set. For instance, in the initial learning iterations, D and Q usually contain fewer examples (queries), which consequently yields lower implicit weights on the first and second terms (i.e., log likelihood on D and Q); on the other hand, U usually contains a large amount of examples, thus the third term (i.e., predictive entropy on U) will be weighted higher. To compensate for this effect, we consider another non-uniform parameter setting: $\alpha_1 = 1/|D|$, $\alpha_2 = 1/|Q|^7$, and $\alpha_3 = 1/|U|$, where $|D|$, $|Q|$ and $|U|$ denote the size of the corresponding data (query) sets (the algorithm is denoted by “AL-GQA(n)”).

We also conduct experiments on active learning with specific queries (*uncertainty sampling* [7]; denoted by “AL-US”) and active learning with inaccurately answered generalized queries [18] (denoted by “AL-GQN”) for comparison. More specifically, “AL-US” always asks one specific example in each learning iteration, and the answer to the example is always accurate; whereas “AL-GQN” tends to ask generalized queries, and is always responded with inaccurate probabilistic answers (with up to 30% noise). In contrast, the proposed algorithms “AL-GQA(u)” and “AL-GQA(n)” also tend to ask generalized queries in each iteration, but are always responded with ambiguous (non-probabilistic) answers.

All of the seven UCI data sets have numeric features, binary class and no missing values. Information on these data sets is tabulated in Table 1. Each whole data set is first split randomly into three disjoint subsets: the training set, the unlabeled set, and the test set. The test set is always 25% of the whole data set. To make sure that active learning can possibly show improvement when the

⁷ When the query set Q is empty (i.e., $|Q| = 0$), we directly set $\alpha_2 = 0$, indicating that the empty query set plays no role in this situation.

Table 1. The seven UCI data sets used in the experiments

Dataset	No. of Attributes	No. of Examples	Class Distribution	Training Size
breast-w	9	699	458/241	1/20
diabetes	8	768	500/268	1/10
heart-statlog	13	270	150/120	1/10
hepatitis	19	155	32/123	1/5
ionosphere	33	351	126/225	1/20
sonar	60	208	97/111	1/5
spambase	57	4601	1813/2788	1/100

unlabeled data are labeled and included into the training set, we choose a small training set for each data set such that the “maximum reduction” of the error rate⁸ is large enough (greater than 10%). The training sizes of the seven UCI data sets range from 1/100 to 1/5 of the whole sets, also listed in Table 1. The unlabeled set is the whole set taking away the test set and the training set.

As for all the UCI data sets, we have neither true target functions nor human oracles to answer the generalized queries, we simulate the target functions by constructing learning models on the entire data sets in the experiments. The simulated target function provides ambiguous answer to each generalized query. The experiment is repeated 10 times on each data set (i.e., each data set is randomly split 10 times), and the experimental results are recorded for comparison.

4.2 Experimental Results

Based on the seven tested UCI data sets, Figure 3 plots the learning curves of the four algorithms, and presents the summary of t-test (the paired two-tailed t-test with a 95% confidence level) for comparison (where each entry, $w/t/l$, means that the algorithm in the corresponding row wins on w data sets, ties on t data sets, and loses on l data sets, compared with the algorithm in the corresponding column). We therefore can make some clear observations from these experimental results:

- On most tested data sets, both “AL-GQA(u)” and “AL-GQA(n)” perform significantly better than “AL-US”. This demonstrates the superiority of active learning with generalized queries and ambiguous answers to the traditional specific-query based learning.
- On most tested data sets, both “AL-GQA(u)” and “AL-GQA(n)” perform significantly better than “AL-GQN”. This indicates that, compared with inaccurate probabilistic answers, ambiguous answers are often more effective in speeding up active learning with generalized queries.

⁸ The “maximum reduction” of the error rate is the error rate on the initial training set D alone (without any benefit of the unlabeled examples) subtracting the error rate on D plus all the unlabeled data in U with correct labels. The “maximum reduction” roughly reflects the upper bound on error reduction that active learning can achieve.

Fig. 3. Learning curves and summary of t-test of the four algorithms on seven UCI data sets

- “AL-GQA(u)” and “AL-GQA(n)” work comparably well on most tested data sets. Note that, “AL-GQA(u)” usually starts from lower error rates in the initial iteration (without asking any queries), and then keeps improving the predictive performance; whereas, “AL-GQA(n)” often has rather high error rates in the initial iteration, but the predictive performance can be promptly improved once it starts asking queries.

To conclude, the experimental results clearly demonstrates the advantage of generalized queries and ambiguous answers: such type of active learning is superior

in speeding up the learning process, compared with active learning with either specific queries or inaccurately answered generalized queries.

5 Conclusions

In this paper, we assume that the active learner can ask generalized queries, and the oracle is capable of respond with ambiguous answers (i.e., positive if at least one corresponding specific example is positive, and negative otherwise). After demonstrating the wide applicability of this setting, we develop a novel algorithm to ask generalized queries and update learning model with ambiguous answers in active learning. Empirical study on UCI data sets demonstrates the superiority of such type of active learning, and shows that the proposed algorithms can significantly speed up the learning process, compared with active learning with either specific queries or inaccurately answered generalized queries.

References

1. Dietterich, T.G., Lathrop, R.H., Perez, T.L.: Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence* 89(1-2), 31–71 (1997)
2. Zhang, Q., Goldman, S.A., Yu, W., Fritts, J.: Content-based image retrieval using multiple-instance learning. In: *ICML 2002: Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 682–689. Morgan Kaufmann Inc., San Francisco (2002)
3. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: *Advances in Neural Information Processing Systems* 15, vol. 15, pp. 561–568 (2003)
4. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
5. Cohn, D.A., Atlas, L., Ladner, R.E.: Improving generalization with active learning. *Machine Learning* 15(2), 201–221 (1994)
6. Baram, Y., El-Yaniv, R., Luz, K.: Online choice of active learning algorithms. *Journal of Machine Learning Research* 5, 255–291 (2004)
7. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: Cohen, W.W., Hirsh, H. (eds.) *Proceedings of ICML 1994*, 11th International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann Publishers, San Francisco (1994)
8. Angluin, D.: Queries and concept learning. *Machine Learning* 2(4), 319–342 (1988)
9. Ling, C.X., Du, J.: Active learning with direct query construction. In: *KDD 2008: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 480–487. ACM, New York (2008)
10. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2, 45–66 (2002)
11. Saar-Tsechansky, M., Provost, F.: Active sampling for class probability estimation and ranking. *Machine Learning* 54(2), 153–178 (2004)
12. Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective sampling for nearest neighbor classifiers. *Machine Learning* 54(2), 125–152 (2004)
13. Margineantu, D.D.: Active cost-sensitive learning. In: *The Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland (2005)

14. Seung, H.S., Oppor, M., Sompolinsky, H.: Query by committee. In: COLT 1992: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 287–294. ACM, New York (1992)
15. Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. *Journal of Artificial Intelligence Research* 4, 129–145 (1996)
16. Zhang, T., Oles, F.J.: A probability analysis on the value of unlabeled data for classification problems. In: Proc. 17th International Conf. on Machine Learning, pp. 1191–1198 (2000)
17. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Proc. 18th International Conf. on Machine Learning, pp. 441–448. Morgan Kaufmann, San Francisco (2001)
18. Du, J., Ling, C.X.: Active learning with generalized queries. In: Proceedings of the 9th IEEE International Conference on Data Mining, pp. 120–128 (2009)
19. Du, J., Ling, C.X.: Asking generalized queries to domain experts to improve learning. *IEEE Transactions on Knowledge and Data Engineering* 22(6), 812–825 (2010)
20. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: Advances in Neural Information Processing Systems, vol. 10, pp. 570–576 (1998)
21. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: A lazy learning approach. In: ICML 2000: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 1119–1126. Morgan Kaufmann Inc., San Francisco (2000)
22. Zucker, J.-D., Chevalere, Y.: Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. application to the mutagenesis problem. In: Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, pp. 204–214. Springer, Heidelberg (2001)
23. Ray, S., Craven, M.: Supervised versus multiple instance learning: an empirical comparison. In: De Raedt, L., Wrobel, S., De Raedt, L., Wrobel, S. (eds.) ICML. ACM International Conference Proceeding Series, vol. 119, pp. 697–704. ACM, New York (2005)
24. Settles, B., Craven, M., Ray, S.: Multiple-instance active learning. In: Advances in Neural Information Processing Systems, vol. 20, pp. 1289–1296 (2008)
25. Grandvalet, Y., Bengio, Y.: Semi-supervised learning by entropy minimization. In: Advances in Neural Information Processing Systems, vol. 17, pp. 529–536 (2005)
26. Guo, Y., Schuurmans, D.: Discriminative batch mode active learning. In: Advances in Neural Information Processing Systems, vol. 20, pp. 593–600 (2008)