Active Learning with Generalized Queries

(Spine title: Active Learning with Generalized Queries)

(Thesis format: Monograph)

by

Jun Du

Graduate Program in Computer Science

Submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
SCHOOL OF GRADUATE AND POSTDOCTORAL STUDIES

CERTIFICATE OF EXAMINATION

Supervisor	Examining Board
Dr. Charles Ling	Dr. Sylvia Osborn
	Dr. Nazim Madhavji
	Dr. Xiaoming Liu
	Dr. Huajie Zhang

The thesis by
Jun Du
entitled

Active Learning with Generalized Queries

is accepted in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy

Date	Chair of Examining Board

Abstract

We study active learning with generalized queries in the thesis.

In contrast to supervised learning, active learning can usually achieve the same predictive accuracy with much fewer labeled training examples, thus significantly reducing the labeling cost. However, previous studies of active learning mostly assume that the learner can only ask specific queries (i.e., require labels for specific examples by providing all feature values). For instance, if the task is to predict osteoarthritis based on a patient data set with 30 features, 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 knees, no history of diabetes, and so on (for all 30 features). However, amongst all these 30 features, many of them may be irrelevant to osteoarthritis diagnosis (such as, ID, name, history of diabetes, etc.). More importantly, for such specific queries, the answers provided by the oracle are also specific. That is, each responded label is only applicable to one specific query (i.e., one specific example).

In real-world situations, the oracles (usually human experts) are often more ready to answer generalized queries, such as "are people over age 50 with knee pain likely to have osteoarthritis?" Here only two relevant features (age and type of pain) are mentioned, and the other 28 are considered as don't-care. Real-world human oracles usually regard such queries as more intuitive and easy to comprehend. More importantly, as one such generalized query can represent a set of specific ones, the corresponding answer provided by the oracle is also applicable to this whole set of specific queries. For instance, in our previous example, the answer for the proposed query is applicable for all people over age 50 with knee pain. Therefore, the active learner can obtain more information from each generalized query (together with the corresponding answer), and furthermore improve the learning more effectively and efficiently.

In this thesis, we assume that the oracle is capable of answering such generalized queries, and develop different algorithms to implement such active learning with generalized queries, according to different real-world scenarios (i.e., under different assumptions). As far as we know, no previous work on active learning can deal with such generalized queries.

More specifically, we study active learning with generalized queries from the following four perspectives:

- We theoretically study *why* and *when* such generalized queries can help in active learning, and demonstrate the superiority of generalized queries over specific ones through toy examples and learning theories. (See Chapter 2 for details.)
- We assume that the oracle can answer generalized queries as easily as specific ones (i.e., with the same effort or cost). Thus we develop two novel active learning algorithms to ask as general as possible queries, and simultaneously keep the answers from the oracle as certain as possible. (See Chapter 3 for details.)
- We make a more realistic assumption that, the more general a query is, the higher cost (effort) it causes to request the label. We therefore study the generalized queries in a cost-sensitive framework, and discuss two scenarios to, either balance the trade-off of the predictive accuracy and the query cost, or minimize the total cost of misclassification and query. (See Chapter 4 for details.)
- We consider a more relaxed scenario that the oracle could only provide *ambiguous* answers to generalized queries. That is, the oracle would only respond with either "positive" ("yes") or "negative" ("no"), where "positive" indicates that at least one of the examples represented by the generalized query can be labeled positive, and "negative" indicates that all such examples would be labeled negative. We then develop another new algorithm to implement active learning with generalized queries under this condition. (See Chapter 5 for details.)

Our study in this thesis has thoroughly addressed the advantages and difficulties of active learning with generalized queries. The theoretical study has proved that the query complexity of active learning with generalized queries is significantly lower than active learning with specific ones. The empirical study for a variety scenarios

has also demonstrated that, to achieve certain predictive accuracy, active learning with generalized queries requires us to ask significantly fewer queries (or requires us to spend significantly lower labeling cost), compared with active learning with specific ones.

Keywords: active learning, generalized queries, oracle, query complexity, cost-sensitive learning, labeling cost, ambiguous answers

Acknowledgements

First of all, I would like to thank my advisor Dr. Charles Ling, who has provided great guidance and support in the past four years, not only for my research but also for many aspects of my life. I still remember when I started my Ph.D. study in 2007 as a freshman at Western, I had very limited knowledge about machine learning research, and had great difficulty in studying, working and living in an entirely different environment. He, as a mentor, started to provide great guidance for me, in research, in academic paper writing, in English practising, in adapting culture difference, etc. I benefit a lot from his constant instruction, and cannot imagine that this thesis could be finished without his support.

I thank the examiners of my thesis proposal, Dr. Sylvia Osborn and Dr. Nazim Madhavji. They carefully read my proposal and provided me great suggestions to improve my work.

Thanks are also given to other members of Data Mining and Business Intelligence Lab at Western: Da Kuang, Eileen Ni and Xiao Li. They are great co-workers and have provided valuable assistance and suggestions in this research. I appreciate the opportunity to work with them.

Professor Zhihua Cai was the advisor for my Master study at China University of Geosciences. He guided me to the research of data mining and recommended me to conduct further research at Western. I would always appreciate his guidance. My former co-workers Yuanyuan Guo and Wenyin Gong have shared a lot of discussion with me about doing research, which is a great help in finishing my thesis.

Finally, I could not have done this without the love and support of my family. My wife Xiaoqin gave me unconditional encouragement and support in continuing the research and finishing the thesis. My parents have always being there to support me in all of my endeavours. The smile of my eight-month-old daughter Lia is the most

beautiful thing I have ever seen in this world; it always inspires me to move forward through all the ups and downs.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction to Active Learning

1.1 What is Active Learning

Supervised learning is one of the most important tasks in the machine learning research area. In supervised learning, the learner is given a (usually large) set of training data with the corresponding labels (nominal or numeric), and is required to construct a learning model that achieves minimum generalization error (or equivalently, achieves minimum error on an unseen test data set drawn independently from the identical distribution). Figure 1.1 demonstrates the general framework of supervised learning.

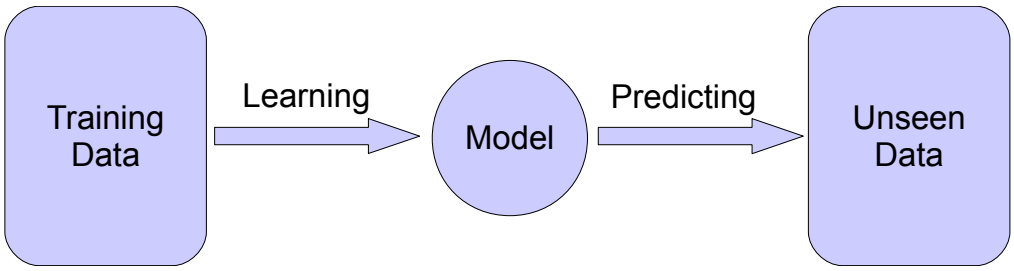


Figure 1.1: The framework of supervised learning.

In order to construct a highly accurate learning model in supervised learning, a considerable amount of labeled training data is usually required. Previous studies in Probably Approximately Correct learning (PAC learning) also have proved the lower bound of the number of training examples needed to achieve a certain generalization error (see [38] for details). However, in practice, such labeled training data are usually difficult (or highly costly) to acquire. For instance, in text categorization,

each document is expected to be tagged with some given class labels (such as, science, politics, sport, entertainment, etc.). This labeling process usually could only be done manually, thus it would cost a considerable amount of time, money or human resource. Under this circumstance, a new learning scheme, active learning, might be able to help.

Briefly speaking, instead of obtaining a whole batch of labeled training data as in the traditional supervised learning, active learning only *selectively* obtains the most useful labeled training data, such that the same generalization error could be achieved with much fewer training examples. The labeling cost, therefore, can be significantly reduced.

More specifically, in active learning, the learner is initially given a (usually small or even empty) set of training data (with the corresponding labels), and an initial learning model is constructed accordingly. Based on this model, the learner then *selectively* generates or chooses (from a given pool) the most useful unlabeled examples, and requests their labels from a given oracle. The process repeats iteratively, such that the training set gradually expands, and the generalization error of the updated learning model decreases consequently. Figure 1.2 demonstrates the general framework of active learning.

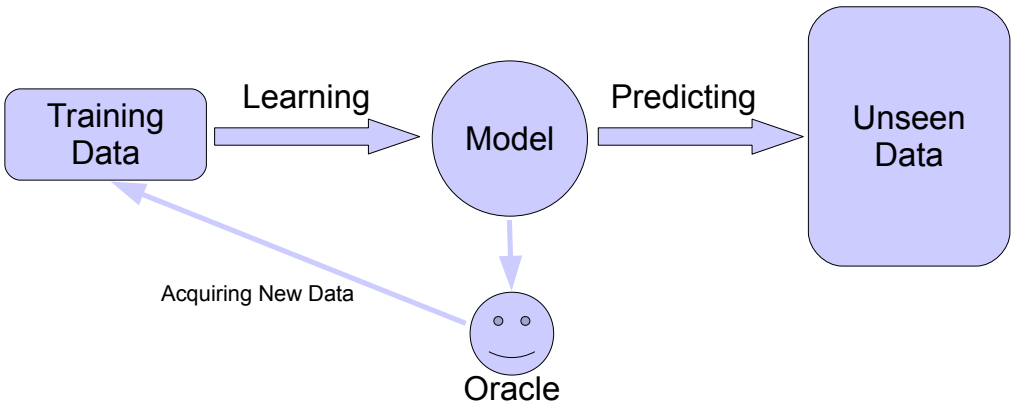


Figure 1.2: The framework of active learning.

The advantage of active learning is obvious: It usually requires much fewer labeled examples (thus significantly reducing the labeling cost) to construct a highly accurate learning model. Therefore, active learning has been widely applied to many real-world applications, such as text classification [37, 53, 27], information extraction [51], image

classification and retrieval [52, 59], video classification [58], speech recognition [54], cancer diagnosis [34], drug design [56], etc.

1.2 Active Learning Scenarios

Different scenarios can be encountered when applying active learning to real-world applications. The most typical ones include *membership query* and *pool-based active learning*. We will introduce these two scenarios in the following subsections.

1.2.1 Membership Query

Membership query is the first active learning scenario, proposed in [2]. Briefly speaking, in this setting, the learner is only given a set of labeled training data, but is allowed to *generate* any synthetic new examples and request the corresponding labels from the oracle.

The advantage of this type of active learning is that, as such generated examples can be arbitrary, they could always perfectly fit the query criteria (see Section 1.3 for details) and efficiently boost the predictive performance (in the ideal situation). Therefore, *membership querying* is commonly used in theoretical studies of active learning (see [21, 14, 12, 13]).

However, *membership query* also has a major flaw, especially when human experts act as oracles in many real-world situations. As there is usually no restriction in generating new synthetic examples in *membership query*, such examples could be *semantically meaningless*. For instance, it is likely for the learner to generate a new example as $[Name = John, Gender = male, Pregnant = yes, \cdots]$. When such meaningless examples are generated, the human oracles are not able to provide the corresponding labels, and the learning process therefore might be halted.

1.2.2 Pool-based Active Learning

Pool-based active learning is proposed in [30], and has been widely used in both academic studies and real-world applications. In this pool-based setting, in addition to the (usually small) set of labeled training data, a (usually large) set of unlabeled

data (called "pool") is also given. The learner, therefore, is expected to select the most valuable examples only from the pool, and request the corresponding labels from the oracle.

Compared with the previous *membership query*, *pool-based* active learning imposes a restriction on the new examples (i.e., the new examples could only be selected from the given pool). As a result, the selected examples, on one hand, might not be the optimal ones to improve the predictive performance (due to the limitation of the pool). On the other hand, however, it can be guaranteed that, these selected examples would always be semantically meaningful. For instance, as all examples existing in the pool should be meaningful, no example as $[Name = John, Gender = male, Pregnant = yes, \cdots]$ would ever be selected.

It is worth noting that, another analogous setting, called *stream-based* active learning, also catches attention in active learning studies. In this setting, the learner is also given two sets of data (labeled and unlabeled). However, instead of being able to access any unlabeled example at any time (as in the pool-based setting), the learner has access to only one example at a time in this case. More specifically, all the unlabeled examples are *sequentially* presented to the learner (i.e., one at a time), and the learner has to decide whether to request the label for the example or discard it. Such *stream-based* active learning is sometimes considered as an online version of the *pool-based* active learning [4].

1.3 Active Learning Query Strategies

When implementing active learning, the most important issue is how to measure the informativeness of the unlabeled examples. The most informative examples usually indicate the ones that can maximally improve the predictive performance. In previous active learning studies, a variety of criteria have been proposed to measure the informativeness, such that the optimal unlabeled examples could be generated or selected. In the following subsections, we will introduce in detail three simple and widely used query strategies (*uncertainty sampling*, *density-based sampling*, and *estimated error reduction*), and briefly review several others.

1.3.1 Uncertainty Sampling

Uncertainty sampling was first proposed in [30]. Since then, it has been most commonly used in both academic research and industry applications, due to simplicity and effectiveness. The basic idea of *uncertainty sampling* is quite intuitive: Given any unlabeled example, if the current learning model is already highly certain in predicting its label, this example might not provide much new information to improve the model; on the other hand, if the current model is quite uncertain in prediction, the example (with the corresponding label) is highly likely to help improve the model. Therefore, *uncertainty sampling* always tends to generate or select (from the pool) the most uncertain unlabeled examples, and request their labels from the oracle.

More specifically, different criteria have been proposed to measure the uncertainty of the unlabeled examples. Settles [46] summarizes and compares three popular uncertainty measurements: *least confident*, *margin sampling* and *entropy*.

Least confident considers the most uncertain example to be the one where the current learning model is least confident in prediction. It consequently can be interpreted as the example with the lowest posterior probability for the most probable class. We denote by U all the given unlabeled examples, and by Y all the possible label values, the most uncertain example, x^* in this case, can be formalized as:

$$x^* = \arg\min_{x \in U} \{P(y_1|x)\} \text{ where } y_1 = \arg\max_{y \in Y} P(y|x) \tag{1.1}$$

Margin sampling defines the certainty as the difference between the posterior probabilities for the first and second probable classes. Therefore, the most uncertain example can be formalized as:

$$x^* = \arg\min_{x \in U} \{P(y_1|x) - P(y_2|x)\} \tag{1.2}$$

$$\text{where } y_1 = \arg\max_{y \in Y} P(y|x), y_2 = \arg\max_{y \in Y \setminus y_1} P(y|x) \tag{1.3}$$

Entropy is usually regarded as the most common criterion to measure uncertainty. On the basis of information theory, *entropy* considers the example with the maximum entropy in posterior probability distribution as the most uncertain one. More formally,

$$x^* = \arg\max_{x \in U} \{ - \sum_{y \in Y} P(y|x) \log P(y|x) \} \tag{1.4}$$

Based on different intuition and rationality, these three criteria behave differently on multi-class problems. However, when binary-class problems are encountered, they are essentially the same. More specifically, for a binary-class problem, all of them always select the example with posterior probability closest to 0.5.

1.3.2 Density-based Sampling

Another simple strategy in active learning is *Density-based Sampling*. The rationale behind *density-based sampling* is also straightforward: To improve the predictive performance in the entire sample space, it is usually preferred that the learning model could make more accurate prediction in the highly dense sample space. Thus, the unlabeled examples with high density would, in general, help make more improvement for the current learning model. *Density-based sampling*, therefore, always tends to generate or select (from the pool) such examples with high density, and request their labels from the oracle.

However, *density-based sampling* usually cannot be applied alone in active learning; additional criteria are often required. More specifically, if the learner always selects the example with the highest sample density in each iteration, it is likely that most (or even all) selected examples are from the same (highly dense) area in the sample space. Consequently, the constructed learning model might behave well only in this area, but totally fail in all the others. To avoid this situation, one solution is to conduct clustering on the unlabeled data set beforehand, and then select the most *representative* examples in each cluster during the learning [39]. Another more typical strategy is to combine *density-based sampling* with other query strategies, such as *uncertainty sampling*. In this case, a trade-off has to be appropriately balanced between the two criteria, such that the selected examples would maximally improve the learning model [16].

1.3.3 Estimated Error Reduction

Both of the previous query strategies tend to improve the learning model by selecting the example base on *indirect* criteria (i.e., uncertainty or density). It is worth noting that, the ultimate goal of active learning is to improve the predictive accuracy of the learning model. Thus a more straightforward way is to select examples that directly maximally increase the predictive accuracy (or equivalently, maximally reduce the predictive error) in each learning iteration. This strategy is call *estimated error reduction*, and was proposed in [43].

Estimated error reduction roughly works as follows: For each example in the pool, a learning model is constructed based on the given training data plus this example (with the estimated label). Thus each example in the pool would correspond to one learning model. All these learning models are evaluated, and the one with the maximum predictive accuracy is chosen. The corresponding example in the pool, therefore, is considered as the optimal one that would maximally increase the predictive accuracy, and is selected for active learning.

This query strategy is simple, direct and rational, but with one crucial flaw. As the labels of all the examples in the pool are unknown, it would be difficult to construct a learning model for each of them. The solution is to *estimate* these labels based on the current learning model (i.e., the learning model constructed only on the training data). More specifically, we denote D the current training set, U the current unlabeled set, and Y the label set. Thus the selected example x^* would be:

$$x^* = \arg\max_{x \in U} \sum_{y \in Y} P(y|x) Acc(D \bigcup \{x, y\}) \tag{1.5}$$

where $P(y|x)$ denotes the estimated probability for x being labeled as y (estimated by the learning model constructed on the current training data alone), and $Acc(D \bigcup \{x, y\})$ denotes the predictive accuracy of the learning model constructed on D plus $\{x, y\}$.

1.3.4 Other Query Strategies

In additional to the previous query strategies, many more sophisticated algorithms (or criteria) are also developed. *Query-by-committee* (QBC) [48] is a more theory-based approach, and considers the example that minimizes the version space as optimal.

When the ensemble method is applied as the base learner in QBC, it could also be considered as a variant of *uncertainty sampling* [30]. *Variance reduction* [10] is developed based on the decomposition of the expected generalization error (i.e., the expected generalization error is decomposed into *noise*, *variance* and *bias*, see [5] for details). As the current learner cannot change anything about the noise and bias, the variance is expected to be maximally decreased to achieve low generalization error. Therefore, the example that maximally decreases the variance term would be selected in each learning iteration. In addition, *Fisher information ratio* [61], *expected model change* [47] are also proposed in previous active learning research, and have been applied to various scenarios. All of these strategies are elaborately designed and well accepted.

1.4 Overview of the Rest of the Thesis

The rest of the thesis is concerned with a new paradigm of active learning — active learning with generalized queries. Instead of asking specific queries in traditional active learning, we propose to ask *generalized queries* in each active learning iteration. As each generalized query can usually represent a set of specific ones, if it can be properly (and accurately) answered by the oracle, the performance of active learning can be significantly improved. More specifically,

- In Chapter 2, we first demonstrate the motivation of studying generalized queries in active learning, and discuss the consequent advantages and difficulties. Then, we theoretically study *why* and *when* such generalized queries can help, through learning theories and toy examples.
- In Chapter 3, we deal with the scenario that the oracle is capable of answering generalized queries as easily as specific ones (i.e., with the same effort or cost). Thus we develop two algorithms to ask as general as possible queries, and simultaneously attempt to keep the answers from the oracle as certain as possible.
- In Chapter 4, we consider a more realistic scenario that higher cost (effort) is required for the oracle to answer generalized queries. We accordingly study generalized queries in a cost-sensitive framework, and develop two methods to,

either balance the trade-off of the predictive accuracy and the query cost, or minimize the total cost of misclassification and query.

- In Chapter 5, we consider another realistic scenario that, instead of providing (accurate) probabilistic answers to the generalized queries, the oracle can only provide *ambiguous* "Yes-No" answers. More specifically, the oracle would only respond with either "positive" ("yes") or "negative" ("no"), where "positive" indicates that at least one of the examples represented by the generalized query can be labeled positive, and "negative" indicates that all such examples would be labeled negative. We then develop another new algorithm to implement active learning with generalized queries under this condition.

- Finally, in Chapter 6, we summarize our conclusions for active learning with generalized queries, and propose future works.

Chapter 2

Active Learning with Generalized Queries

2.1 What are Generalized Queries?

In all previous works on active learning, it is always assumed that the learner could only ask specific queries (i.e., require labels for specific examples with all feature values provided), and it is also assumed that the oracle could only answer such specific queries. For instance, if the task is to predict osteoarthritis based on a patient data set with 30 features, 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 knees, no history of diabetes, and so on (for all 30 features). However, the fact is that, many of these 30 features 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 responded label is only applicable to one specific query (i.e., one specific example).

In real-world situations, the oracles (usually human experts) are often more ready to answer generalized queries, such as "are people over age 50 with knee pain likely to have osteoarthritis?" Here only two relevant features (age and type of pain) are mentioned, and the other 28 are considered as don't-care. 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, we assume that the

oracle is more powerful: It can answer generalized queries by returning probabilistic labels.

The advantage of generalized queries is not only that they are more natural and relevant. More importantly, answers for such generalized queries can usually provide much more information, as one generalized query is often equivalent to many specific ones. In our previous example, the answer for the generalized query is applicable for all people over age 50 with knee pain. This allows active learner to improve learning more effectively and efficiently.

The difficulty of generalized queries is that answers from the oracle can often be uncertain.[1] For instance, the answer could be "Yes with a 85% probability" that people over age 50 with knee pain would have osteoarthritis. An overly general query, such as "are people over age 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 waste the effort of the oracle if these answers are directly discarded.

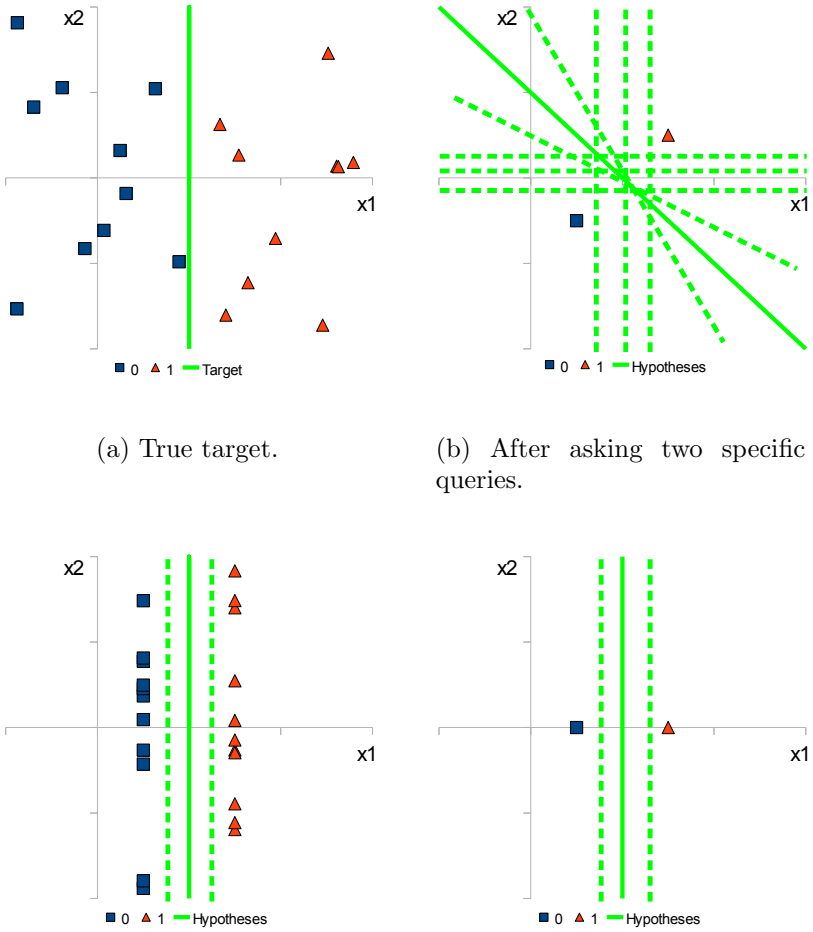
2.2 Why Do Generalized Queries Help?

In this section, we analyse in detail *why* (and *when*) generalized queries could help in active learning. Specifically, considering two scenarios (*data with irrelevant features* and *data without irrelevant features*), we analyse the effect of the generalized queries, intuitively through toy examples, and theoretically based on learning theory.

2.2.1 Data with Irrelevant Features

In this subsection, we study the effect of the generalized queries on data with irrelevant features. Specifically, we take a simple synthetic classification problem (with two features (x_1 and x_2) and binary class ($y = 0$ and $y = 1$)) as a toy example, to illustrate the advantage of generalized queries.

[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.



(a) True target.

(b) After asking two specific queries.

(c) After asking two generalized queries.

(d) After asking two specific queries with feature selection.

Figure 2.1: Illustration of data with irrelevant features.

We assume that x_2 is an irrelevant feature, and show the data distribution and the true boundary ($x_1 = 1$) in Figure 2.1(a). We can notice from Figure 2.1(a) that, the true boundary is perpendicular to the x_1 axis (i.e., only affected by x_1).

In traditional active learning with specific queries, the learning algorithm usually requests the label for one specific example in each iteration. For our toy example, we can show the training data after two learning iterations (i.e., after requesting the labels for two data points — one labeled "0" and the other labeled "1").[2] More specifically,

[2]We assume that the active learning initializes with an empty training set.

we obtain the labels ($y = 0$ and $y = 1$) for two data points ($x_1 = 0.5, x_2 = -0.5$) and ($x_1 = 1.5, x_2 = 0.5$) respectively, then we can see these two data points in Figure 2.1(b).

On the other hand, in active learning with generalized queries, the learner asks one generalized query in each iteration. As one generalized query often represents a set of specific examples, this whole set of examples (with their labels) can be included in the training set in each iteration. For our toy example, we can also show the training data after two learning iterations (i.e., after requesting the labels for two generalized queries). More specifically, we suppose the learner can successfully discover the irrelevant feature x_2, and construct two generalized queries [$x_1 = 0.5$, $x_2 = * \longrightarrow y =?$] and [$x_1 = 1.5$, $x_2 = * \longrightarrow y =?$] (with same x_1 values as in Figure 2.1(b)), where "$*$" represents the don't-care features in the queries. When these two generalized queries are answered by the oracle[3], two sets of the corresponding data points are included in the training set, as shown in Figure 2.1(c).

Theoretical studies consider active learning as a version space shrinking process [48]. Roughly speaking, after requesting the label(s) for new example(s) in each iteration, the version space will shrink to be consistent with all the given labeled examples. In the noise-free setting, when the version space shrinks to only one hypothesis, this remaining one is thus regarded as the *true hypothesis*. Armed with this active learning theory, we can clearly demonstrate the difference between active learning with specific and generalized queries.

In our toy example, the version space initially contains *all* the linear boundaries in the two-dimensional space (x_1-x_2 space) with an empty training set.[4] After two iterations, the active learning with specific queries obtains two data points in the training set (as in Figure 2.1(b)). Thus, the version space shrinks, and only the linear boundaries that *separate these two data points* remain. The dotted lines in Figure 2.1(b) show samples of all these hypotheses in the version space, and the solid line represents the current optimal hypothesis with the maximum margin [6]. On the other hand, the active learning with generalized queries obtains two sets of data points in the training set after two iterations (as in Figure 2.1(c)). The version space thus also shrinks, and only the linear boundaries that *separate these two sets of data points* remain. The

[3]We assume that these generalized queries can be responded with certain answers here, for better illustration.

[4]We only consider linear functions as our hypotheses for illustration.

dotted and solid lines in Figure 2.1(c) also show samples of all these hypotheses and the optimal one, respectively.

We can clearly see that, compared with the specific queries (Figure 2.1(b)), the generalized queries (Figure 2.1(c)) rule out significantly more hypotheses (such as, the linear boundaries that are not perpendicular to the x_1 axis) after two iterations. Based on active learning theory, this clearly indicates that generalized queries can significantly outperform specific ones, in terms of efficiently finding the *true hypothesis* (as well as speeding up the learning process). In addition, we can also notice that, after two iterations, the optimal hypothesis produced by the generalized queries (i.e., the solid line in Figure 2.1(c)) is already quite close to the true boundary (i.e., the solid line in Figure 2.1(a)); whereas, the optimal hypothesis produced by the specific queries (i.e., the solid line in Figure 2.1(b)) is still far away from the true one. This again demonstrates the advantage of generalized queries in this case.

One may notice that, the power of generalized queries comes from the ability of discovering irrelevant features. Thus, a question arises: Can active learning with specific queries have the same power when it is combined with feature selection [25]? Figure 2.1(d) illustrates this situation on our toy example.

Specifically, we suppose that the feature selection could also successfully discover and remove the irrelevant feature x_2, thus the entire active learning process is implemented only on a one-dimensional space (i.e., the x_1 axis). This indicates that feature selection changes the initial version space to be all the thresholds on the x_1 axis. As before, we suppose the labels of the same two data points ($x_1 = 0.5$ and $x_1 = 1.5$) are requested after two iterations, as shown in Figure 2.1(d). Then, the version space further shrinks to be all the thresholds on the x_1 axis that lie between these two data points. These thresholds can be recovered to the original x_1-x_2 space, and become all the linear boundaries perpendicular to the x_1 axis and between $x_1 = 0.5$ and $x_2 = 1.5$, as the dotted lines show in Figure 2.1(d). (Again, the solid line in Figure 2.1(d) represents the current optimal hypothesis.) We can easily tell from the comparison between Figures 2.1(c) and 2.1(d) that, combined with feature selection, the specific queries can yield almost the same version space and the optimal hypothesis as the generalized queries (after two learning iterations). This indicates that, the specific query indeed can have the same power as the generalized ones, when feature selection is applied. Note that, in the next subsection, we analyse the effect of generalized queries on *data without irrelevant features*, and we will show that feature selection

no longer helps in that situation, whereas generalized queries might still work (see Section 2.2.2 for details).

So far, we have demonstrated the superiority of generalized queries over specific ones, through a simple toy example. We now further theoretically analyse the sample complexity of active learning with generalized queries, under some certain conditions. Strictly speaking, the number of examples needed by generalized queries will always be more than that needed by specific ones, simply because one generalized query often represents a set of examples, and this whole set of examples is usually included in the training set in each iteration. However, in the current active learning setting, we care more about the number of queries, rather than the number of examples, as that is the true cause of the labeling cost.

Previous theoretical research [21, 14, 12, 13] has shown the sample (query) complexity of active learning under different assumptions. In particular, the analysis of the *Query-by-Committee* algorithm [21] shows that, the sample (query) complexity of the labeled examples is roughly $O(d \log 1/\epsilon)$ under some certain conditions, where ϵ is the desired generalization error and d is the *Vapnik-Chervonenkis dimension (VC dimension)* of the hypothesis space. This indicates an exponential improvement compared with the usual sample (query) complexity $\Omega(d/\epsilon)$ in a traditional supervised setting. Here we further analyse the query complexity of active learning with generalized queries.

Specifically, we consider linear separators in n-dimensional space (as well as other certain conditions, as in [14, 13]). The *VC dimension* of the hypothesis space thus is $d = n + 1$. We suppose there are r ($r < n$) irrelevant features; and in the ideal case, all of them can be successfully discovered in generalized queries. However, directly analysing generalized queries is quite difficult, as each query (representing a set of examples) often has complex impact on the version space. Instead, the previous illustration has shown that, in the best case, generalized queries have (roughly) the same effect in shrinking the version space as specific queries with feature selection. We thus can simplify the analysis of generalized queries by equivalently considering active learning with specific queries in a lower dimensional space (i.e., after the irrelevant features are eliminated). Therefore, after discovering all the r irrelevant features, generalized queries can reduce the dimension of the feature space from n to $n - r$, and the *VC dimension* from d to $d - r$ consequently. The query complexity thus would simply be $O((d - r) \log 1/\epsilon)$.

Some interesting conclusions can be further observed by comparing these query complexities. The query complexity for the specific queries ($O(d \log 1/\epsilon)$) grows linearly with the feature dimension n (as $(n = d-1)$ in this case); thus it suffers severely from large feature dimensions. In contrast, the query complexity for generalized queries ($O((d-r)\log 1/\epsilon)$) grows linearly only with the number of *relevant* features $n-r$ (as $(n-r = d-r-1)$ in this case). This indicates a significant improvement when data contains a large number of redundant features. In particular, when the number of relevant features $(n-r)$ is fixed (which commonly occurs in real-world applications), generalized queries only have constant query complexity ($O(log1/\epsilon)$) in terms of the feature dimension, thus achieving a linear improvement compared with specific ones (with complexity $O(d \log 1/\epsilon)$). This therefore theoretically demonstrates the superiority of generalized queries over specific ones.

2.2.2 Data without Irrelevant Features

In the previous subsection, we have shown clearly that generalized queries can shrink the version space significantly faster than specific ones, thus outperforming the latter in speeding up the learning process. However, so far, this conclusion is only valid for the data that contains irrelevant features. What if data contains *no* irrelevant features? Can generalized queries still help? We study this issue in this subsection, still through some toy examples.

We first consider the situation that the target is a decision tree (or a set of decision rules), as this type of target commonly occurs in real-world applications. For illustration, we suppose that the data contains five features $x_1 - x_5$ (all of which are relevant), and we also suppose that the target is a simple decision tree with six leaves $L_1 - L6$, as shown in Figure 2.2.

Such a target as a decision tree (or decision rules) has an interesting property: even in the case that all the features are relevant, given specific values for some features, the others might still be "conditionally irrelevant". For instance, we can see in Figure 2.2 that, given $x_1 = 0$, $x_3 = 0$ and $x_5 = 0$, the label will always be 0 (Leaf $L6$), regardless of the values for x_2 and x_4. Thus, in this case, we can still consider x_2 and x_4 as "conditionally irrelevant". We can observe many other similar cases in Figure 2.2. Indeed, given any target as a decision tree (or decision rules), we can always discover such "conditionally irrelevant" features. Thus, the generalized queries can always be

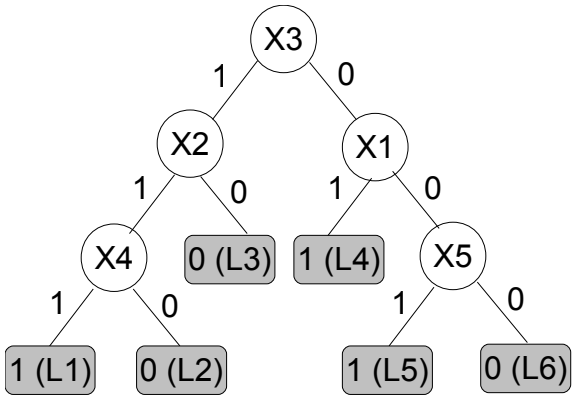


Figure 2.2: Synthetic decision tree.

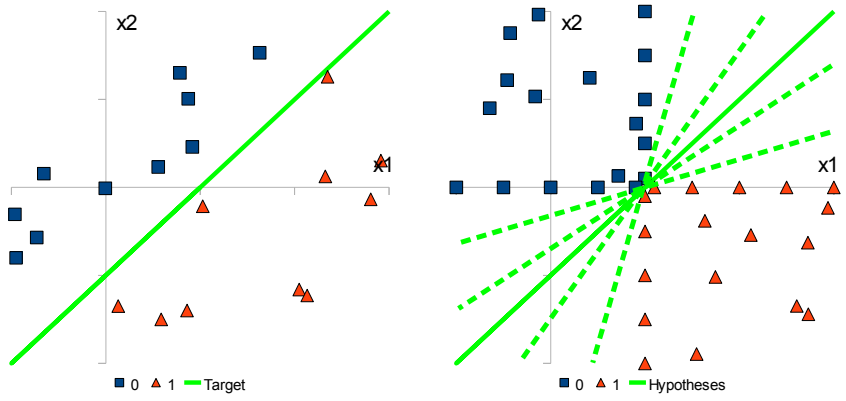
constructed from the specific examples, with those "conditionally irrelevant" features being don't-care. Again, as each generalized query represents a set of specific examples, the whole set of examples could be included in training set in each iteration, and the learning process could be speeded up consequently. Here we provide further theoretical analysis on active learning with generalized queries in this case.

Specifically, we consider decision trees with k leaves. Given n data features, Guestrin [23] has shown the hypothesis space for such a k-leaf decision tree is roughly $|H| = n^{k-1}(k+1)^{2k-1}$. Thus the sample complexity of such a decision tree in traditional supervised learning would be $O(k \ln k)$. (As the sample complexity for any consistent learner is $\frac{1}{2\epsilon}(\ln |H| + \ln \frac{1}{\delta})$ given ϵ and δ in PAC learning, the sample complexity for the k-leaf decision tree can be easily derived given the corresponding hypothesis space.) In active learning with generalized queries, we again suppose that the learner can always discover all the "conditional irrelevant" features; thus in such an ideal case, each generalized query can directly correspond to a leaf node in the decision tree (i.e., a decision rule in the tree). With the answers from the oracle, only k such generalized queries require to be asked, in order to discover all the k rules in the decision tree. Therefore, the query complexity of such active learning with generalized queries in this case would be $O(k)$. Compared with $O(k \ln k)$ we have derived previously, this clearly indicates a logarithmic improvement, and again demonstrates the superiority of generalized queries for decision trees.

In addition to decision trees (or decision rules), targets as linear (or non-linear) boundaries in n-dimensional space also commonly occur in real-world applications. Apparently, if all of these n features directly affect the labels of the examples in any case,

no irrelevant (or even "conditionally irrelevant") features could be discovered. Thus, it seems that, generalized queries cannot be constructed, and furthermore, cannot contribute to speeding up the learning process.

However, another type of more flexible generalized queries, where the features could be described as *partially irrelevant*, can be applied. Specifically, we suppose the target is a linear boundary ($x_1 - x_2 - 1 = 0$) in a 2-dimensional space (x_1-x_2 space), as shown in figure 2.3(a). Although both of these two features (x_1 and x_2) can always affect the label (y) for any data point, a generalized query with *partially irrelevant* features, such as $[x_1 < 1, x_2 > 0 \longrightarrow y = ?]$, can still be constructed. Instead of regarding some features as entirely irrelevant, this type of generalized query imposes some certain constraints on the features; thus each query can still represent a set of specific examples. If such a generalized query can obtain a certain answer from the oracle, the whole set of corresponding examples could be included in the training set and the learning process could be speeded up accordingly. Figure 2.3(b) illustrates the situation after two learning iterations in this toy example. More specifically, with two generalized queries and the corresponding certain answers, $[x_1 < 1, x_2 > 0 \longrightarrow y = 0]$ and $[x_1 > 1, x_2 < 0 \longrightarrow y = 1]$, two sets of examples are included in the training data, and the version space shrinks efficiently. Such generalized queries consequently can significantly speed up the learning process.



(a) True target.

(b) After asking two generalized queries.

Figure 2.3: Illustration of data without irrelevant features.

Note that, for both of the above two types of targets, feature selection is no longer

able to help, as no irrelevant features could be discovered and removed. In contrast, the generalized queries can still work to efficiently shrink the version space and speed up the learning process. This thus again demonstrates the advantages of generalized queries.

To summarize, in this section, we analyse two scenarios of data with or without irrelevant features. Through toy examples and learning theory, we conclude that, active learning with generalized queries can indeed significantly improve the learning process, compared with the traditional supervised learning and active learning with specific queries.

2.3 Assumptions for Generalized Queries

We need to make some assumptions to implement active learning with generalized queries. The first fundamental assumption we make is that, the oracle is capable of answering such generalized queries. This assumption can usually be satisfied, especially when human experts act as oracles in most real-world applications. All the studies in the rest of the thesis are based on this assumption.

In addition, other assumptions are also made in the following chapters, to simulate various real-world scenarios. More specifically, we assume that the oracle can answer generalized queries as easily as specific ones (i.e., with the same effort or cost) in Chapter 3, and propose two algorithms to ask as general as possible queries in each learning iteration. We then make a more realistic assumption in Chapter 4 that, the more general the query is, the higher cost (effort) it causes to request the label, and accordingly study generalized queries in a cost-sensitive framework. However, in both of these two parts of research, we still implicitly assume that the oracle is capable of providing probabilistic answers for the generalized queries. Such an assumption might be difficult to satisfy in some real-world situations. Therefore, in Chapter 5, we make a more relaxed assumption that the oracle could only provide *ambiguous* "Yes-No" answers to the generalized queries, and develop another new algorithm to ask generalized queries with such ambiguous answers.

Chapter 3

Asking Generalized Queries to Improve Learning

3.1 Introduction

In this chapter, we assume that the oracle can answer the generalized queries as easily as the specific ones. The theoretical study in the previous chapter has demonstrated that, the more general a query is, the more it would contribute to improve the learning model. The learner thus is expected to ask as general as possible queries during the learning process. However, on the other hand, if the queries are too general, the answers from the oracle might be uncertain (as we have discussed in Chapter 2). Consequently, these uncertain answers might even degrade the learning by introducing noise.

Therefore, in this chapter, our task is to design an active learner that attempts to ask generalized queries and simultaneously obtain highly certain answers from the oracle. More specifically, we design a new algorithm called AGQ, for Active learner with Generalized Queries. AGQ can construct generalized queries with don't-care features, for either the pool-based or the membership-query active learner. AGQ is in essence different from some previous similar studies in active learning, such as, batch mode active learning, active learning with feature labeling, and active learning with feature selection; see Section 3.2 for details. In each active learning iteration, we propose a four-step procedure for AGQ to construct a generalized query and update the learning model; see Section 3.3 for details. However, AGQ can only generalize

specific feature values to don't-care. We then extend AGQ to AGQ$^+$, which can generalize specific feature values to meaningful new features for both nominal and numeric features. For example, AGQ$^+$ can ask such queries as "are people aged between 50 and 65, with moderate or severe knee pain, likely to have osteoarthritis?" Here, age (a numeric feature) is generalized to a range, and knee pain (a nominal feature) is generalized to a subset of values. These newly constructed features can form hierarchical structures, and are often meaningful in real-world applications. See Section 3.4 for the detailed description of AGQ$^+$. Experiments on synthetic and real-world data sets show that AGQ and AGQ$^+$ ask significantly fewer queries compared with the traditional active leaner. See Sections 3.5 for details. In addition, AGQ$^+$ can also automatically produce subsets for nominal features and ranges for numeric features, which can be used in further learning. To the best of our knowledge, this is the first work proposing active learning with generalized queries, and showing that it is highly effective.

One might argue that it may be difficult for the oracle or human expert to provide accurate probabilities of the labels for the generalized queries. As we will show in Section 3.6.1, when the probabilities of labels are contaminated with low noise, AGQ still learns quite well. That is, AGQ is robust with estimated probabilities of the labels. In addition, Section 3.6.2 studies the difference between AGQ (AGQ$^+$) and active learning with feature selection. We will show that active learning with feature selection performs significantly worse than the proposed AGQ methods, due to the overly general queries it produces. In Section 3.6.3, we will discuss the behaviour of AGQ (AGQ$^+$) with only few initial labeled examples, and propose an additional heuristic to handle this issue.

3.2 Related Work

All previous works on 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 chapter is the first work proposing active learning with generalized queries. Again, the main advantage of AGQ is that one generalized query is usually equivalent to many specific ones. Thus, the answer from the oracle is also for all of the specific queries.

Even though one generalized query is equivalent to multiple specific queries, our AGQ

method is still quite different from batch-mode active learning [28, 24]. In batch-mode active learning, the learning model requests labels for a batch of examples (i.e., multiple specific queries) in each iteration, thus the oracle is required to provide multiple answers for all these queries (i.e., with multiple costs). On the other hand, in AGQ, the oracle answers only one generalized query in each iteration (i.e., with one cost). Thus, AGQ costs much less than the batch-mode active learning, for answering queries in the learning process.

Druck et al. [17] 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 uncertainty sampling) and generalizes it to a query. Labeling such uncertain examples has been proved to be very effective in improving predictive accuracy (see Section 3.5 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 a label for only one feature, the answer from the oracle could be very uncertain. To deal with this problem, it is assumed in [17] 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 3.5 for details). In any case, every query of AGQ is counted, regardless of the certainty of the reply.

One major step of AGQ is to find irrelevant features and substitute them with the don't-care (i.e. "$*$") (see Section 3.3.2). However, our algorithm is very different from, and much better than, combining feature selection and the traditional active learning. We will discuss this in detail in Section 3.6.2.

3.3 AGQ Algorithm

In this section, we will describe a novel active learning algorithm called AGQ (Active learning with Generalized Queries). AGQ can generalize features (nominal or numeric) with specific values to don't-care features. In Section 3.4, we extend AGQ to AGQ$^+$, which generalizes from specific feature values to subsets of values for nominal features, or to ranges for numeric features. These newly constructed features can form meaningful hierarchies for further learning.

As most previous works on active learning are pool-based, and use uncertainty sampling to choose the most valuable unlabeled examples, in this section, we will also describe AGQ using uncertainty sampling in a pool-based setting. 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 features $X_1, X_2, ..., X_n$ 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 [53, 13, 35]. An initial learner L is built using the current labeled training data set 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 features in the most uncertain example above, and substitutes them with "$*$" (representing don't-care features).[1]

[1] Although feature selection [25, 32] can also discover irrelevant features, as we will show in Section 3.6.2, the AGQ method is significantly better than feature selection.

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 the 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 generalized query can effectively represent a set of specific ones. 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 worsen. (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.

3.3.1 Finding the Most Uncertain Example

Similar to the previous works on the pool-based active learning, AGQ first builds a predictive model based on the current set of labeled examples, and uses it to make prediction on 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.[2]

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 [7] of 100 j48

[2]For highly imbalanced and cost-sensitive data, an optimal threshold for classification can be calculated [20], or found via cross-validation [49], 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.

decision trees (implemented in Weka [57]) 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 [40]. The standard decision tree algorithm is chosen because it tends to build small trees; this facilitates us finding irrelevant features in the next step.

3.3.2 Constructing the Generalized Query

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

If a set of m features 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 features. However, there are $\binom{n}{m}$ subsets of m features (given a total of n features), and for each subset, 2^m value combinations (for binary features) must be tested. The task is clearly computationally expensive.

A heuristic, similar to the process of finding the largest itemsets in mining association rules [8, 31], is designed. More specifically, let D be the current don't-care feature list, and let x_u be the current most uncertain example. We gradually expand D by adding more irrelevant features via greedy search, as follows. For each feature X_i not currently in D, we generate a fixed number (100 in our experiments) of examples with randomly assigned values for features in D and X_i, all based on x_u. The number of examples is fixed to prevent combinatorial explosion of feature values when D grows. The feature value is randomly chosen according to the distribution of that feature value in the original data set. This most accurately reflects the distribution of examples in the domain.[3] The feature 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 features in D. This process is depicted with the pseudo code in Algorithm 1.

Clearly, this can generate the most general queries (i.e., queries with the most don't-care features) based on the current learning model. However, queries with too many

[3]The same random sampling method is used in Sections 3.3.3 and 3.3.4.

Algorithm 1: find_don't-care_features

Input: M, the current learning model; x_u, the most uncertain example; θ, the predefined threshold.
Output: D, the don't-care feature list.

```
p_u = probability of majority class for x_u (estimated by M);
D = ∅;
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 (estimated by M);
        end
        S_i = Σ_{n=1}^{100} (p_n − p_u)^2 / 100;
    end
    Choose X_i with the smallest S_i;
    if S_i < θ then
        Add X_i to D;
    else
        noChange = false;
    end
until noChange is false ;
```

don't-care features 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 features. 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 features (see Table 3.1 in Section 3.5.1). This will be especially true when the initial labeled training set is very small. In Section 3.6.3 we study AGQ when there are only two initial labeled examples.

3.3.3 Asking Generalized Queries to the Oracle

In our work, we assume that the oracle can answer generalized queries with don't-care features just as easily as specific queries (without don't-care features). We believe that in most real-world situations, human experts can easily answer such generalized queries with an estimated probability. As we will also show in Section 3.6.1, our AGQ performs well with a small error in probability estimation. Thus it is quite robust.

In Section 3.5.2 we will test AGQ on the UCI data sets [3], comparing it with the traditional pool-based active learner. An interesting question arises: as we do not know the target functions of the UCI data sets, 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 data set to represent the target function. This is the best model we can get as it is built from the whole data set. Specifically, we use the bagging of 100 j48 decision trees on the whole data set 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 features are replaced with specific values sampled randomly) are generated. To avoid combinatorial explosion when the generalized query has too many don't-care features, 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 an 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 $[name = Jane, gender = female, pregnant = yes, age = 30, \ldots]$, then the generalized query could be $[name = *, gender = *, pregnant = yes, age = *, \ldots]$. Unrealistic generalized queries (such as $[name = *, gender = male, pregnant = yes, age = *, \ldots]$) 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.

3.3.4 Updating the Training Data Set

Given the probability distribution to the generalized query from the oracle, we need to utilize it to expand the training data set and to build a better classifier. Again, because each generalized query effectively represents a set of specific ones, 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 feature is replaced randomly by a specific value of that feature. However, experiments (Section 3.5.2) 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, 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 features.[4]

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.

[4]Note that, as the generalized queries are constructed from the most uncertain examples, when updating the training set, these most uncertain examples are always added into the training set. Thus, AGQ always outperforms the traditional uncertainty sampling method.

3.4 AGQ$^+$ Algorithm

In the previous section, we proposed a novel algorithm AGQ that is composed of four steps to implement active learning with generalized queries. However, the AGQ algorithm in Section 3.3.2 can produce generalized queries with features that are either entirely irrelevant (i.e., generalized as "$*$"), or entirely specific (i.e., keeping the original specific value). That is, as long as the feature is relevant, it could be represented by only one specific value in the generalized queries. This is clearly very restrictive. In most real-world applications, however, nominal features can form subsets (of values), and numeric features can form ranges. What we hope is that the active learner can automatically form such new, high-level features when it asks generalized queries.

For example, to predict osteoarthritis, "knee pain" could be a relevant nominal feature with values "none", "moderate" and "severe", and "age" could be another relevant feature with numeric values. Then, in addition to generalizing the irrelevant features as "$*$", we may also generalize the relevant features to several nominal values (such as, "knee pain" being "moderate" or "severe") or a numeric interval (such as, "age" being $[50, 65]$). We can then construct generalized queries, such as "are people aged between 50 and 65, with moderate or severe knee pain, likely to have osteoarthritis?".

Not only are these generalized queries more natural and flexible to represent queries with different degrees of generalization, the new features constructed can also form hierarchical structures, and can be meaningful for further learning. For example, if a subset of nominal feature values or a range of a numeric feature is repeatedly generated by the active learner, then they can be meaningful high-level concepts, to be used in future learning, or transfer learning [11, 41]. See Section 3.5.3 for more discussions. We call this extension "AGQ$^+$", due to its powerful generalization ability.

AGQ$^+$ has different strategies in the second step of AGQ in constructing the generalized queries; the other three steps (i.e., finding the most uncertain example, asking generalized queries to the oracle, and updating the training data set) are the same as AGQ (see Section 3.3). An additional fifth step is added. In this step, subsets of nominal features and ranges of numeric features generated by AGQ$^+$ are consolidated, and hierarchies may be formed. In this section, we will mainly describe how AGQ$^+$ generates subsets of nominal features and ranges of numeric features.

3.4.1 Nominal Features

For nominal features, we first still find all the *strong-irrelevant* features (i.e., the don't-care features) as in Section 3.3.2. Then, we check all the remaining features by greedy search to identify the *weak-irrelevant* features (i.e., the features that can be generalized with several, but not all, nominal values). The main idea is that, the class probability of the examples with combinations of weak-irrelevant feature values should be the very similar. Our heuristic strategy is to use this property to discover which features are weak-irrelevant with corresponding values. More specifically, we still denote by x_u and D the current most uncertain example and the strong-irrelevant feature list (don't-care feature list) respectively. We also denote by W the weak-irrelevant feature list (with the corresponding feature values). Given x_u found by the current learning model, D constructed by Algorithm 1, and an initially empty W, we gradually expand W with the weak-irrelevant features (and the corresponding feature values), as follows. For each feature X_i not currently in D and W, and for each of its feature value $X_i = a_{ij}$, we generate a fixed number (100 in our experiments) of examples with randomly assigned values for the features in D and W, all based on x_u.[5] The current learning model then makes predictions on the class probabilities of these examples. If the model produces exactly the same class probabilities for all these examples and x_u, we add the current feature X_i (and the corresponding feature value a_{ij}) into W. Therefore, after checking all the rest features (together with the corresponding feature values), we can identify all the weak-irrelevant features and include them into W. Furthermore, we can construct the generalized query, by substitute all features in D with $*$ and all features in W with their corresponding values in W. This process is depicted with the pseudo code in Algorithm 2.

3.4.2 Numeric Features

For numeric features, we apply a similar strategy to identify weak-irrelevant features after obtaining all the strong-irrelevant ones. Roughly speaking, given a strong-irrelevant feature list D, an initially empty weak-irrelevant feature list W, and the current most uncertain example x_u, we gradually expand W with weak-irrelevant features (and their corresponding values). However, unlike nominal features, there

[5]For the features in D, we randomly assign any feature values; whereas for the features in W, we only randomly assign the corresponding feature values previously identified.

Algorithm 2: find_weak-irrelevant_features (nominal)

Input: M, the current learning model; x_u, the most uncertain example; D, the strong-irrelevant feature list (don't-care feature list).

Output: W, the weak-irrelevant feature list (nominal).

```
p_u = probability of majority class for x_u (estimated by M);
W = ∅;
foreach X_i ∉ (D ∪ W) do
    foreach X_i = a_ij 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_k (with available nominal values) for all X_k ∈ W;
            end
            p_n = probability of majority class for x_n (estimated by M);
        end
        S_ij = Σ_{n=1}^{100} (p_n − p_u);
    end
    if S_ij = 0 then
        Add X_i (with a_ij) to W;
    end
end
```

are infinite valid values for numeric features, and we need find a numeric range (instead of several nominal values) for each weak-irrelevant feature (such as, $[50, 65]$ for age). Thus, Algorithm 2 cannot be applied here. Instead, for each feature X_i not currently in D and W, we construct a numeric range $[a_i - \delta, a_i + \delta]$ based on the current feature value a_i and a pre-defined small number δ.[6] Then, we generate a fixed number (100 in our experiments) of examples with randomly assigned values for the features in D, W and the current feature, all based on x_u.[7] Again, the current learning model makes predictions on the class probabilities of these examples. If the model produces exactly the same class probabilities for all of these examples and x_u, the numeric range will be again expanded by δ. Otherwise, it stops. Then, the current feature (and its final numeric range) is included in W as a weak-irrelevant feature. Therefore, after checking all the remaining features, we can identify all the weak-irrelevant features,

[6] In our experiments, we set δ as 1/20 of entire valid feature range.

[7] For the features in D, we randomly assign any feature values; for the features in W, we randomly assign the values within the previously identified numeric range; whereas for the current feature, we randomly assign value within $[a_i - \delta, a_i + \delta]$.

Algorithm 3: find_weak-irrelevant_features (numeric)

Input: M, the current learning model; x_u, the most uncertain example; D, the strong-irrelevant feature list (don't-care feature list); δ, the pre-defined small number.

Output: W, the weak-irrelevant feature list (numeric).

```
p_u = probability of majority class for x_u (estimated by M);
W = ∅;
foreach X_i ∉ (D ∪ W) do
    UB = LB = a_i (current feature value);
    repeat
        UB = a_i + δ;
        LB = a_i − δ;
        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_k for all X_k ∈ W (within available numeric range);
                Randomly assign X_i within [LB, UB];
            end
            p_n = probability of majority class for x_n (estimated by M);
        end
        S_ij = Σ_{n=1}^{100} (p_n − p_u);
    until S_ij > 0 ;
    Add X_i (with numeric range [LB, UB]) to W;
end
```

and construct the generalized query by substituting all features in D with $*$ and all features in W with their corresponding numeric ranges. This process is depicted with the pseudo code in Algorithm 3.

In the next section, we will perform extensive experiments with AGQ and AGQ$^+$.

3.5 Empirical Study

In this section, we conduct experiments on a synthetic data set and 14 UCI [3] data sets to compare AGQ with the previous active learning algorithm that asks specific queries. We then apply AGQ$^+$ on the same UCI data sets to see its advantages over AGQ.

3.5.1 AGQ on Synthetic Data Set

In this subsection, we use synthetic data to empirically study the performance of AGQ, compared with the traditional pool-based active learning with uncertainty sampling.[8]

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 feature values for the don't-care features, 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 greater 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 features, $X1$ - $X5$, and six leaves, $L1$ - $L6$, as in Figure 3.1. To simulate the real-world data set, we add another five *irrelevant* features, $X6$ - $X10$, to generate the synthetic data. We assume that all these features are binary, so is the class label. Therefore, with 10 binary features, 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 features are. We can also directly use the target function as the oracle to answer the generalized queries.

The experiment is repeated on the synthetic data set 20 times. Each time, the whole data set 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 data set respectively, and the rest is the unlabeled set.

Figure 3.2 plots the average error rates of the optimal AGQ (shown as "AGQ-Opt"), AGQ and the traditional pool-based active learning (shown as "Pool"). We can see

[8]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, uncertainty sampling in this case can also be regarded as an implementation of QBC.

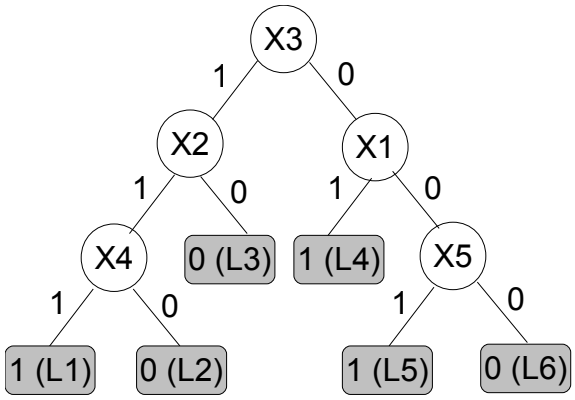


Figure 3.1: Target tree used to generate synthetic data.

clearly, from Figure 3.2, 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 3.3) is quite effective — AGQ asks generalized queries with certain labels; that is, they are not overly general. Overly general queries would receive uncertain labels, and would negatively affect learning. This can happen especially when the initial training set is very small. See Section 3.6.3.

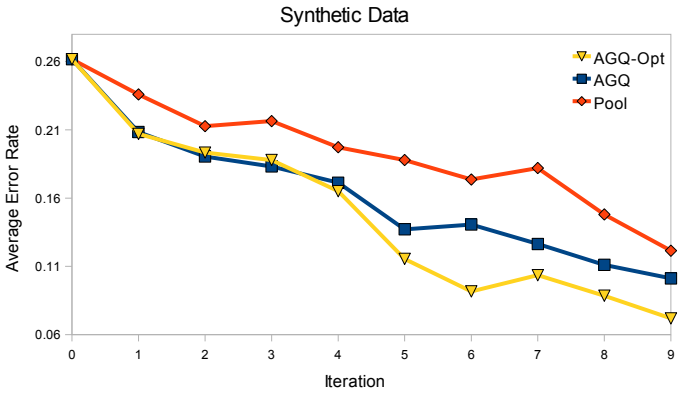


Figure 3.2: Comparison of the average error rate among "AGQ-Opt", AGQ and "Pool" on the synthetic data.

To further compare AGQ and "Pool", we extract a typical series of queries from them during the active learning process. Table 3.1 tabulates these queries (Query in the table), as 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 3.1 that AGQ always constructs generalized queries with don't-care features 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 3.1), but they still contain the most irrelevant features. 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").

	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 3.1: Comparison of five consecutive queries between AGQ and "Pool" on synthetic data.

To summarize from the experiment on the synthetic data, AGQ can often identify correctly the irrelevant features 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 the correct

labels) are included in the training set. This yields the outstanding performance of AGQ (similar to the optimal AGQ) on the synthetic data set, compared with the traditional pool-based active learning.

3.5.2 AGQ on UCI Data Sets

In this subsection, we use 14 real-world data sets from the UCI Machine Learning Repository [3] to compare AGQ with the optimal AGQ and the pool-based active learning algorithm. All of these data sets have binary class and no missing values. Information on these data sets is tabulated in Table 3.2.

Each whole data set (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 in the training set, we choose a small training set for each data set such that the "maximum reduction" of the error rate[9] is large enough (greater than 10%). The training sizes of the 14 UCI data sets range from 1/200 to 1/5 of the whole data sets, also listed in Table 3.2. The unlabeled set (U) is the whole data set (D) taking away the test set (T) and the training set (R).

Data Set	Type of Features	No. of Features	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 3.2: The 14 UCI data sets used in the experiments.

The experiment is repeated on each data set 20 times (i.e., each data set is randomly split 20 times), when comparing "AGQ-Opt", AGQ and "Pool". We stop training

[9]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) minus 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.

when the error rate of "Pool" is reduced by 3/4 of the "maximum reduction".

Figure 3.3 plots the average error rates of "AGQ-Opt", AGQ and "Pool" on a typical UCI data sets ("Hepatitis"), and the comparison on all the 14 data sets will be presented later. We can see from Figure 3.3 that, AGQ performs only slightly worse than "AGQ-Opt" but significantly better than "Pool", similar to the result on the synthetic data set. This again clearly demonstrates the advantage of AGQ: AGQ performs almost as well as "AGQ-Opt", and significantly outperforms "Pool".

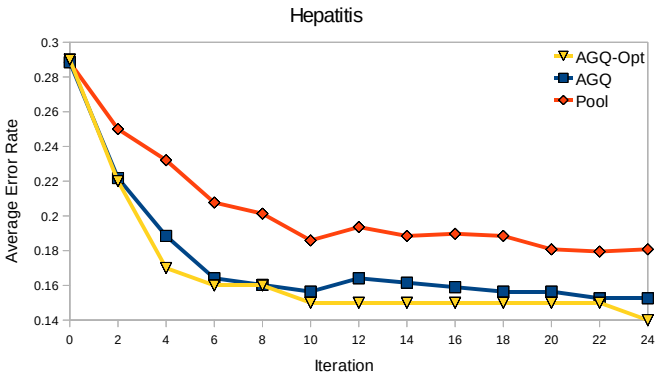


Figure 3.3: Comparison of average error rate among "AGQ-Opt", AGQ, and "Pool" on "Hepatitis".

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 data sets shows that, AGQ wins on 9, ties on 4, and loses on 1 data set, compared with "Pool". This clearly indicates that, with the same number of queries (same 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 features (and its percentage of the total features) in each query (Don't-care Features in the table), the average certainty of the oracle (Certainty of Oracle)[10], 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

[10]The certainty of oracle, calculated from the oracle described in Section 3.3, is always about the majority class (which can be either 1 or 0). Thus, the certainty value is between 0.5 and 1.

"Pool" (% of Iteration Reduction), and AGQ wins/ties/loses compared with "Pool" (AGQ w/t/l). Table 3.3 presents these statistics based on the 14 UCI data sets.

Data Set	Don't-care Features (% of Total Features)	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 3.3: Important statistics of AGQ and comparison with "Pool" on the 14 UCI data sets.

From Table 3.3 we can see that, on average, AGQ discovers 12.5 don't-care features, 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 data sets ("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 data set "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 feature (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 data set "kr-vs-kp" where AGQ loses, the certainty of the oracle is relatively high (94%), and 39% of the features 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 features are highly constrained. As there are a total of 36 features, the data set (containing about 3,000 examples) is very sparse; that is, only a small fraction of the feature value combinations is valid. Thus, the examples generated by AGQ from the generalized queries and included in training set (Section 3.3.4) are mostly invalid examples (i.e., meaningless board positions). These invalid

examples may severely change the distribution of the original data set thus degrading the performance of AGQ. We will study this issue further in our future work.

From Table 3.3 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 data sets where AGQ ties with "Pool", the two methods require almost the same number of iterations (queries). However, on the nine data sets where AGQ wins over "Pool", AGQ asks 61% fewer queries compared with "Pool". Over all 14 data sets, 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 data sets.

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

3.5.3 AGQ$^+$ on UCI Data Sets

In this subsection, we conduct the experiments on the same 14 UCI data sets, to compare the performance of AGQ$^+$ and AGQ. All the experimental configurations are the same as in the previous subsection. We perform the same t-test on the average error rates between AGQ$^+$ and AGQ. The results show that for the 14 UCI data sets, AGQ$^+$ wins on 3, ties on the other 11 data sets, compared with AGQ. This indicates that AGQ$^+$ can predict as well as AGQ in most data sets, and better than AGQ in some cases.

The advantage of AGQ$^+$ lies in not only its (slightly) better performance, but more importantly, its capability of producing natural and powerful generalized queries with meaningful new features during the active learning process. Consider the data set "Diabetes" as an example. This data set was originally used in [50] for predicting diabetes from eight numeric features of the patients. The meanings and valid range of these features can be found in Table 3.4. However, in [50], those numeric features were *manually* discretized into meaningful categories, in order to train a neural network model. With our AGQ$^+$ algorithm, in the learning process, the ranges of numeric feature are automatically produced, and the generalized queries are accordingly constructed, all based directly on the raw numeric features.

	Fea. Name	Fea. Range	Fea. Meaning
1	preg	$\{0-17\}$	Number of times pregnant
2	plas	$\{0-199\}$	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3	pres	$\{0-122\}$	Diastolic blood pressure (mm Hg)
4	skin	$\{0-99\}$	Triceps skin fold thickness (mm)
5	insu	$\{0-846\}$	2-Hour serum insulin (mu U/ml)
6	mass	$\{0-67.1\}$	Body mass index (weight in kg/(height in m)^2)
7	pedi	$\{0.078-2.42\}$	Diabetes pedigree function
8	age	$\{21-81\}$	Age (years)

Table 3.4: Feature ranges and meanings for "Diabetes".

To illustrate AGQ$^+$'s capability of producing such numeric ranges and generalized queries on "Diabetes", we list several typical feature ranges and queries constructed in the active learning process. Table 3.5 lists all the eight features (and class) of the "Diabetes". The upper part of Table 3.5 shows the manually discretized range for every feature used in [50]; the middle part shows several typical feature ranges formed by AGQ$^+$; and the lower part shows eight typical generalized queries and the corresponding probability answers from the oracle (each row represents one generalized query).

Form the middle part of Table 3.5, we can see that AGQ$^+$ automatically produces necessary ranges of the numeric features, some of which are roughly the same as the manual ones, while others are completely different. For example, feature "skin" is generalized with range $\{0-29\}$, which is very close to the manually discretized category $\{0-25\}$; feature "insu" is generalized with range $\{142-227\}$, also close to the manually discretized category $\{151-240\}$. In addition, some numeric ranges of the same feature can clearly form hierarchical structures. For example, for feature "insu", the second range $\{48-132\}$ is roughly a subset of the first range $\{0-133\}$; and the fourth range $\{143-227\}$ is also roughly a subset of the fifth range $\{145-399\}$. The similar phenomena can also be discovered from other features. Such (hierarchical) ranges can form new meaningful features without any human interference, and can be used in further learning.

From the lower part of Table 3.5, we can see that, AGQ$^+$ can generalize the features to numeric ranges in most queries, and also obtain relatively certain answers from the oracle. For example, in Query 1, feature "preg" is generalized with range $\{0-2\}$, feature "plas" is generalized with range $\{> 171\}$, and this query obtains a 100% certain answer from the oracle (see Column "class"). This clearly illustrates the behaviour of AGQ$^+$: it produces meaningful generalized queries (with automatically discretized feature categories), and obtains certain answers from the oracle. We can also notice

	preg	plas	pres	skin	insu	mass	pedi	age	class
	Feature Ranges Manually Discretized by [50]								
	$\{0-2\}$	$\{0-89.1\}$	$\{1-76.1\}$	$\{0-25\}$	$\{0-110\}$	$\{1-22.814\}$	$\{0-.244\}$	$\{21-24\}$	$\{0\}$
	$\{3-6\}$	$\{89.2-107.1\}$	$\{76.2.-98.1\}$	$\{26-32\}$	$\{111-150\}$	$\{22.815-26.84\}$	$\{.245-.525\}$	$\{25-30\}$	$\{1\}$
	$\{>7\}$	$\{107.2-123.1\}$	$\{>98.2\}$	$\{>33\}$	$\{151-240\}$	$\{26.841-33.55\}$	$\{.526-.805\}$	$\{31-40\}$	
		$\{123.2-143.1\}$			$\{>241\}$	$\{33.551-36.563\}$	$\{.806-1.11\}$	$\{41-55\}$	
		$\{143.2-165.1\}$				$\{>36.564\}$	$\{>1.11\}$	$\{>55\}$	
		$\{>165.2\}$							
	Typical Feature Ranges for Individual Features Formed by AGQ$^+$								
	$\{0-2\}$	$\{37-77\}$	$\{6-55\}$	$\{0-25\}$	$\{0-133\}$	$\{20-27\}$	$\{.078-.488\}$	$\{24-30\}$	
	$\{3-5\}$	$\{47-53\}$	$\{12-22\}$	$\{16-26\}$	$\{48-132\}$	$\{25-32\}$	$\{.190-2.064\}$	$\{27-33\}$	
	$\{4-9\}$	$\{82-122\}$	$\{34-58\}$	$\{28-38\}$	$\{78-162\}$	$\{35-56\}$	$\{.261-.495\}$	$\{39-51\}$	
	$\{7-11\}$	$\{144-184\}$	$\{48-60\}$	$\{34-44\}$	$\{143-227\}$	$\{>36\}$	$\{.579-.813\}$	$\{42-54\}$	
	$\{>7\}$	$\{151-190\}$	$\{74-87\}$	$\{40-80\}$	$\{145-399\}$	$\{>40\}$	$\{.736-1.906\}$	$\{48-54\}$	
	$\{9-13\}$	$\{160-199\}$	$\{87-100\}$	$\{44-63\}$	$\{190-360\}$	$\{45-65\}$	$\{>0.897\}$		
	$\{11-13\}$	$\{>171\}$	$\{>94\}$	$\{>44\}$	$\{257-427\}$		$\{>.997\}$		
			$\{104-116\}$						
	Eight Typical Queries Produced by AGQ$^+$ (each row represents a query)								
1	$\{0-2\}$	$\{>171\}$	64	30	180	34.1	0.33	38	1(100%)
2	0	147	85	$\{>44\}$	0	$\{>39\}$	0.38	24	0(100%)
3	7	133	84	0	0	$\{>37\}$	$\{.579-.813\}$	37	1(85%)
4	0	188	82	$\{0-29\}$	$\{143-227\}$	32	0.68	22	1(100%)
5	$\{>7\}$	122	56	0	0	33.3	$\{>.997\}$	33	1(100%)
6	$\{4-9\}$	81	78	40	$\{0-132\}$	$\{>40\}$	0.26	42	0(100%)
7	6	$\{>173\}$	$\{87-100\}$	0	0	40.8	1.46	$\{39-51\}$	0(100%)
8	$\{>7\}$	119	$\{74-87\}$	35	0	$\{25-32\}$	0.26	29	1(60%)

Table 3.5: Typical feature ranges and queries produced by AGQ$^+$ on "Diabetes".

from Table 3.5 that Query 8 obtains an uncertain answer from the oracle (with 60% probability estimation). However, this type of low certainty query rarely occurs in the whole learning process, thus would not significantly affect the performance of AGQ$^+$.

To summarize, AGQ$^+$ performs slightly better than AGQ. Most importantly, AGQ$^+$ is capable of producing meaningful intermediate features during the active learning process.

3.6 Discussion

In the previous section, we demonstrated the outstanding performance of AGQ and AGQ$^+$, compared with the traditional pool-based active learning that only asks specific queries. However, one may be still concerned about some other issues of the proposed methods, such as: What if the oracle cannot provide accurate probability estimation for the generalized queries? What is the difference between AGQ (or AGQ$^+$) and active learning with feature selection? How does AGQ (or AGQ$^+$) perform with very few initial labeled examples? We will answer these questions in this section. Note that, as the behavior of AGQ and AGQ$^+$ is mostly similar, we only consider AGQ in this section. All the conclusions are also applicable to AGQ$^+$.

3.6.1 Probability Estimation of the Oracle

In the previous sections, we assumed that for generalized queries, the oracle (or human expert) is capable of providing accurate probability distributions. However, in real-world applications, it is common that the oracle or human experts can only provide "approximate answers" (i.e., estimated probability distributions). We speculate that small perturbations in probability distribution will not dramatically affect the performance of AGQ. This is because small perturbations in label probabilities only represent light noise of examples added in the training set. These light noises could be cancelled out in the successive updates of the training set. With a robust base learning algorithm (such as the bagged decision trees), such small noises would be insensitive. In this subsection, we study this issue experimentally.

We conduct experiments to compare the original AGQ and AGQ with inaccurate probability answers on the 14 UCI data sets (used in Section 3.5.2). To simulate

inaccurate probability answers, we first calculate the exact probability answer as described in Section 3.3.3, and then randomly alter it with up to 10%, 20% and 50% noise (increase or decrease by up to 10%, 20% and 50% uniformly distributed random noise). All the other experimental configurations are the same as in Section 3.5.2.

Figure 3.4 plots the average error rates of AGQ, AGQ with 10% noise, AGQ with 20% noise, and AGQ with 50% noise, on a typical UCI data set ("Hepatitis"). We can see that the error rates of AGQ with a low level of noise (10% and 20%) are similar to AGQ without noise, but AGQ with a high level of noise (50%) is significantly worse.

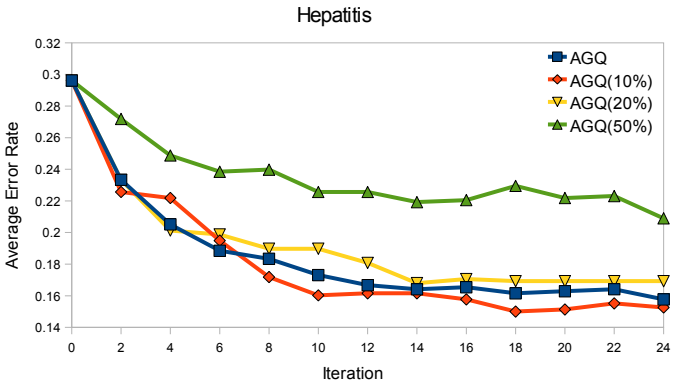


Figure 3.4: Comparison of the average error rate between AGQ and AGQ with inaccurate probability answers (with 10%, 20%, and 50% noise respectively) on "Hepatitis".

Table 3.6 presents a summary of the t-test on the average error rates based on the 14 UCI data sets. Each entry in Table 3.6, $w/t/l$, means that the algorithm in the corresponding row wins on w, ties on t, and loses on l data sets, compared with the algorithm in the corresponding column. We can clearly see from Table 3.6 that AGQ with 10% noise is almost indistinguishable from the original AGQ (it ties with AGQ on 13 out of 14 data sets). AGQ with 20% noise is only slightly worse than AGQ without noise (it ties on 9 and loses on 5 data sets). However, AGQ with 50% noise is significantly worse (it loses on 12, and ties on 2 data sets). Clearly, high noise in the oracle answers will degrade the performance of AGQ, but low noise will not. Thus, AGQ is quite robust, and can tolerate a low level of noise in the probability distribution of oracle answers.

	AGQ (10%)	AGQ (20%)	AGQ (50%)
AGQ	1/13/0	5/9/0	12/2/0

Table 3.6: Summary of the t-test on the average error rates for comparing AGQ with AGQ (10% noise), AGQ (20% noise) and AGQ(50% noise).

3.6.2 AGQ vs. Feature Selection

One may notice that the essence of AGQ is to find irrelevant features, thus, it is closely related to feature selection. Feature selection (e.g., [25, 32, 33]) attempts to discover and discard irrelevant features to improve the predictive accuracy. Thus, would it work if we simply apply the pool-based active learning (which only asks specific queries) after irrelevant features are discovered and discarded by feature selection? How does it compare with our AGQ? We study this issue in this subsection.

Indeed, a straightforward way to make the traditional pool-based active learning to ask generalized queries is to simply apply feature selection as a pre-processing step in active learning. That is, feature selection is conducted on the initial labeled training examples, to identify and eliminate all irrelevant features. Then the traditional pool-based active learning is used to find the most uncertain specific query. Putting back the irrelevant features as the don't care, a generalized query is produced. However, as we are usually given only a small number of initial labeled examples in active learning, feature selection is most likely to be unreliable, thus eliminating too many relevant features in the pre-processing step.

A more sophisticated and improved method is to conduct feature selection in each active learning iteration. More specifically, in each iteration, active learning selects the most uncertain example based on the current learning model, and at the same time, feature selection identifies the irrelevant features based on the current labeled examples. Then, a generalized query can be constructed by substituting all irrelevant features as $*$ in the most uncertain example. Such generalized queries are asked to the oracle, and the labeled training set is updated, as in AGQ. In essence, irrelevant features are identified by feature selection, instead of using the method described in AGQ (Section 3.3.2).

We implement this feature selection active learning method, and compare its performance with the proposed AGQ on the same 14 UCI data sets. More specifically, we use the backward selection method to select irrelevant features, and use a bagging of 100 decision trees (the same base learning algorithm used in AGQ) as the classifier to

evaluate them. All the other experimental configurations are the same as in Section 3.5.

Figure 3.5 plots the average error rates of AGQ and the pool-based active learning with feature selection (called "Feature Selection") on a typical UCI data set ("Hepatitis"). We can see clearly that AGQ performs significantly better than "Feature Selection". We also perform the t-test on the average error rates on the 14 UCI data sets. AGQ wins on 12, and loses only on 2 data sets, compared with "Feature Selection". This clearly indicates that "Feature Selection" performs significantly worse than the proposed AGQ in most cases.

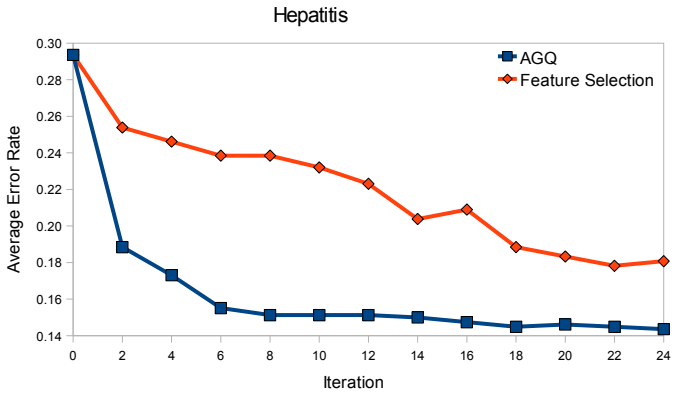


Figure 3.5: Comparison of the average error rate between AGQ and Feature Selection on "Hepatitis".

After looking into the queries produced by "Feature Selection" and the corresponding oracle answers, we find the reason. The generalized queries constructed by "Feature Selection" are often overly general. More specifically, even though "Feature Selection" identifies the irrelevant features (and generalizes them as $*$) in each iteration, the available labeled training examples in each iteration are still limited (especially in the first few iterations). Thus "Feature Selection" tends to identify more features as irrelevant, and constructs overly generalized queries. Consequently, the oracle could only provide uncertain answers to these overly general queries, thus degrading the active learning performance.

On the other hand, AGQ has a more strict criterion to identify irrelevant features, compared with "Feature Selection". More specifically, only when all generated examples with different feature values have very close class probability estimation (instead of the same class prediction in "Feature Selection"), the current feature could be re-

garded as irrelevant. (See Section 3.3.2 for details.) Thus, the overly general queries would not frequently occur in AGQ (see Table 3.1 in Section 3.5.1).

In addition, AGQ$^+$ can ask generalized queries with subsets of nominal feature values, or ranges of numeric features. "Feature Selection" will not be able to produce this type of queries. Thus, AGQ$^+$ is inherently more powerful than "Feature Selection".

3.6.3 AGQ with Very Few Initial Labeled Examples

In the previous sections, we mentioned that when the initial training set is very small, the constructed learning model could be unreliable, thus the discovered don't features could be unreliable as well. Indeed, with the limited information from few labeled examples, it is difficult (or even impossible) to correctly identify the don't-care features. Thus, in this subsection, we study the performance of AGQ with very few initial labeled examples.

Given very few initial labeled examples, the original AGQ is more likely to consider many features as don't-care, and construct overly general queries. With the uncertain answers from the oracle, these overly general queries can severely degrade the performance of AGQ. Here, we design an additional heuristic to deal with this issue. Roughly speaking, for each query, we bond the number of don't-care features to the size of the current training set. When the training set is small, only a small number of features could be considered as don't-care, due to the limited information provided by the labeled examples. On the other hand, when the training set is relatively large, more don't-care features are allowed to be discovered, as more reliable information is provided.

Specifically, when constructing the generalized query (as in Section 3.3.2), we add the current feature into the don't-care feature list (when all the other conditions are satisfied), only if the number of all don't-care feature value combinations is smaller than (or equal to) the current training set size. For example, given only two labeled training examples, at most one binary feature could be considered as don't-care; given four labeled training examples, at most two binary features (or one four-value feature) could be considered as don't-care; and so on.

We implement this heuristic on the base of the original AGQ algorithm, and compare its performance with "Pool" on the same 14 UCI data sets. All the experimental

configurations are the same as in Section 3.5, except we only include two labeled examples (one positive and one negative) in the initial training sets.

Figure 3.6 plots the average error rates of AGQ and "Pool" on a typical UCI data set ("Hepatitis"). We can see that AGQ still performs significantly better than "Pool" even with only two initial labeled examples.

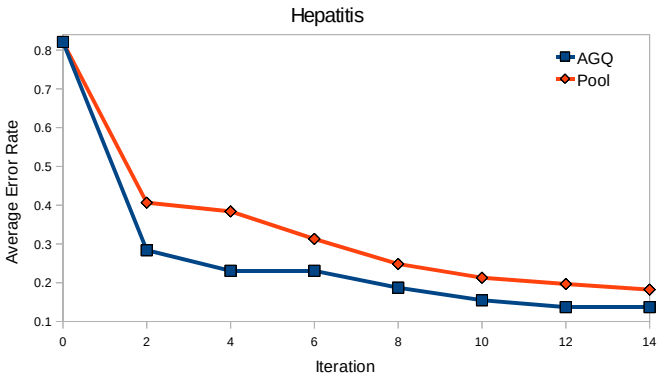


Figure 3.6: Comparison of the average error rate between AGQ and "Pool" on "Hepatitis", with two initial labeled examples.

In addition, the t-test of the average error rates on all the 14 UCI data sets shows that, AGQ wins on 8, ties on 6, and loses on 0 data set, compared with "Pool". This also clearly indicates the similar conclusion as in the previous subsection: the error rate of AGQ still decreases much faster than "Pool", even with only a few initial labeled examples.

3.7 Summary

We have made two assumptions in the chapter: (1) the oracle is capable of answering the generalized queries; (2) the cost (effort) of answering such generalized queries is the same as answering the specific ones. Under these assumptions, we have proposed a four-step strategy (AGQ) for the active learner to ask the generalized queries and efficiently improve the learning model. The AGQ algorithm is then extended to AGQ$^+$, which can produce subsets of nominal feature values or ranges of numeric features. (Therefore, AGQ could be considered as a special case of AGQ$^+$.)

Our experiments show that, compared with the traditional pool-based active learning, AGQ can achieve the same predictive accuracy with significantly fewer queries (36%

fewer on average). We also show that AGQ's performance is similar to the (unrealistic) optimal AGQ, and the performance of AGQ$^+$ is even more superior to AGQ in some cases. AGQ works well even with only two labeled examples in the initial training set. In addition, our experiments verify the robustness of the proposed algorithm: AGQ with inaccurate answers from the oracle (up to 20% perturbation) still performs comparably to the original AGQ on most tested UCI data sets.

Chapter 4

Asking Generalized Queries with Minimum Cost

4.1 Introduction

In Chapter 3, we have assumed that the oracle is capable of answering the generalized queries as easily as the specific ones, and proposed two algorithms to improve learning by asking such generalized queries. However, in many real-world situations, although the oracle is indeed capable of answering such generalized queries, the cost (effort) is often higher. For instance, it is relatively easy (i.e., with low cost) to diagnose whether one specific patient has diabetes or not, with all necessary information provided. However, it is often more difficult (i.e., with higher cost) to provide accurate diabetes diagnoses (accurate probability) for all men over age 60 and weighted between 220 and 240 pounds. In a real-world situation, more domain expertise is usually required for the oracles to answer such generalized queries well, thus the cost for asking generalized queries is often more expensive. Consequently, it yields a trade-off in active learning: on one hand, asking generalized queries can speed up the learning, but usually with high cost; on the other hand, asking specific queries is much cheaper (with low cost), but the learning process might be slowed down.

In this chapter, we study generalized queries in active learning, by developing delicate algorithms to handle uncertain answers and applying cost-sensitive framework to tackling querying cost. More specifically, we assume that the querying cost is known to be non-uniform, and ask generalized queries in the following two scenarios:

- **Scenario 1 (Balancing Acc./Cost Trade-off):** We consider only querying cost in this scenario. Thus, instead of tending to achieve high predictive accuracy by *asking as few as possible queries* (as in traditional active learning), the learning algorithm is required to achieve high predictive accuracy by *paying as low as possible querying cost*.
- **Scenario 2 (Minimizing Total Cost):** In addition to querying cost, we also consider misclassification cost produced by the learning model in this scenario.[1] Thus, the learning algorithm is required to achieve minimum total cost of querying and misclassification in the learning process.

In particular, we propose a novel method to, first construct generalized queries according to two objective functions in the above two scenarios, and then update the training data and the learning model accordingly. Empirical study in a variety of settings shows that, the proposed methods can indeed outperform the existing active learning algorithms in simultaneously maximizing the predictive performance and minimizing the querying cost.

Asking generalized queries in active learning is more natural and general; and assuming the non-uniform querying cost is more applicable in real-world situations. Our research of asking generalized queries with non-uniform cost can therefore be directly deployed in active learning applications.

The rest of the chapter is organized as follows. Section 4.2 reviews previous works on active learning, especially the assumptions made in the previous active learning studies. Section 4.3 describes our strategies to ask generalized queries with cost in two scenarios. In Section 4.4, empirical study is conducted on real-world data sets to verify the superiority of the proposed methods. Section 4.5 presents conclusions and future work.

4.2 Related Work

All of the active learning studies make assumptions. Specifically, most of the previous works assume that the oracles can only answer specific queries, and the costs for

[1] Here we only consider that both the querying cost and the misclassification cost are on the same scale. Extra normalization might be required otherwise.

asking these queries are uniform. Thus, most active learning algorithms (such as [30, 48, 53, 44, 4, 10, 61, 43]) are designed to achieve as high as possible predictive accuracy by asking a certain number of queries (or equivalently, asking as few as possible queries to achieve certain predictive accuracy).

In the previous chapter and [18], we relaxed the assumption of asking specific queries, and proposes active learning with generalized queries. However, it assumes that the oracles can answer these generalized queries as easily as the specific ones. That is, the costs for asking all the queries are still the same, regardless of the queries being specific or generalized. Druck et al. [17] also assumes that the oracle is capable of providing labels for features, and proposes active learning with feature labeling. However, the (non-uniform) cost of asking such queries is also not considered.

Margineantu [35], Kapoor et al. [29], Settles et al. [45] relax the assumption of uniform cost, and study active learning in a cost-sensitive framework. However, they limit their research to specific queries, and only consider that the costs for asking those specific ones are different.

In this chapter, we study generalized queries with cost in active learning. Specifically, we assume that the oracles can answer both specific and generalized queries, but with different cost. The more general a query is, the higher cost it causes. This assumption is more flexible, more general, and more applicable to the real-world applications. Under this assumption, considering uniform cost for generalized queries (such as [18]) and considering non-uniform costs for specific queries (such as [35, 29, 45]) can both be regarded as special cases. Table 4.1 illustrates the different assumptions in active learning studies. As far as we know, this is the first time to propose this more general assumption and design corresponding learning algorithms for active learning.

	Specific Queries	Generalized Queries
Uniform Cost	[30, 48, 53, 44, 4, 10, 61, 43]	[18, 17]
Non-uniform Cost	[35, 29, 45]	This Chapter

Table 4.1: Assumptions in active learning studies.

4.3 Algorithm

In this section, we design an active learning algorithm to ask generalized queries. Roughly speaking, the active learning process can be broken into the following two

steps in each learning iteration:

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

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

4.3.1 Constructing Generalized Queries

In each learning iteration, constructing the generalized queries can be regarded as searching for the optimal query in the query space, according to the given objective function. We propose two objective functions according to the previous two scenarios (in Sections 4.3.1.1 and 4.3.1.2), and design an efficient searching strategy to reduce the computational complexity (in Section 4.3.1.3).

4.3.1.1 Balancing Acc./Cost Trade-off

In scenario 1, we only consider querying cost, and still use accuracy to measure the predictive performance of the learning model, thus the learning algorithm is required to balance the trade-off between the predictive accuracy and the querying cost. We therefore design an objective function to choose query that yields maximum ratio of accuracy improvement to querying cost in each iteration.

More formally, Equation 4.1 shows the objective function for searching for a query in iteration t, where q^t denotes the optimal query, Q^t denotes the entire query space, $C_Q(q)$ denotes the querying cost for the current candidate query q, $\Delta Acc^t(q)$ denotes the accuracy improvement produced by q, which can also be presented by subtracting the accuracy in iteration $t-1$ (denoted by Acc^{t-1}) from the accuracy in iteration t (denoted by $Acc^t(q)$).[2]

[2]The accuracy improvement ($\Delta Acc^t(q)$) can be negative, when the accuracy after asking the query ($Acc^t(q)$) is even lower than the one before asking (Acc^{t-1}).

$$q^t = \arg\max_{q \in Q^t} \frac{\Delta Acc^t(q)}{C_Q(q)} = \arg\max_{q \in Q^t} \frac{Acc^t(q) - Acc^{t-1}}{C_Q(q)} \tag{4.1}$$

We can see from Equation 4.1 that, estimating $\Delta Acc^t(q)/C_Q(q)$ is required to evaluate the candidate query q. As we assume that the querying cost $C_Q(q)$ is known, we only need separately estimate the accuracies before and after asking q (i.e., Acc^{t-1} and $Acc^t(q)$).

Estimating Acc^{t-1} is rather easy. We simply apply cross-validation or leave-one-out to the current training data, and obtain the desired average accuracy. However, estimating $Acc^t(q)$ is a bit difficult. Note that, if we know the answer of q, the training data could be updated by using exactly the same strategy we will describe in Section 4.3.2 (*Updataing Learning Model*), and $Acc^t(q)$ thus could be easily estimated on the updated training data. However, the answer of q is still unknown in the current stage, thus here, we apply a simple strategy to optimistically estimate this answer, and then evaluate q accordingly.

Specifically, we first assume that the label of q is certainly 1.[3] Thus, we update the training data (using the same method as in Section 4.3.2), and estimate $Acc^t(q)$ accordingly. Then, we assume that the label of q is certainly 0, and again update the training data and estimate $Acc^t(q)$ in the same way. We compare these two estimates of $Acc^t(q)$, and optimistically choose the better (higher) one as the final estimate.

One might argue that it is inappropriate to assume the answer of q to be certainly 0 or 1, as the true answer is also likely to be uncertain. We clarify this doubt here. Specifically, in the case that the true answer of q is uncertain (for instance, "1 with 60% probability"), our strategy will introduce 40% noise when regarding it as certainly 1, and 60% noise when regarding it as certainly 0. In both cases, $Acc^t(q)$ will be underestimated due to the introduced noise, and q thus will be under-evaluated according to Equation 4.1. Consequently, q will be less likely to be chosen as the optimal query in the current iteration. On the other hand, in the case that the true answer of q is indeed certain (for instance, "1 with 100% probability"), our strategy will introduce zero noise when regarding it as certainly 1, and 100% noise when regarding it as certainly 0. The estimate of $Acc^t(q)$ with label 1 is thus likely to be relatively high due to the noise-free situation, whereas the estimate with label 0 is

[3] We only consider binary classification with labels 0 and 1 here, for better illustration.

likely to be low due to the severe noise. The higher estimate, i.e., the estimate with the true label 1, is then optimistically chosen in our strategy, thus q can be evaluated according to Equation 4.1 with the true label. Consequently, q will be more likely to be chosen as the optimal query in the current iteration. This way, we use a simple strategy to bias the chosen generalized queries towards the ones with certain answers, thus avoid introducing noise in the learning process.

4.3.1.2 Minimizing Total Cost

In Scenario 2, we consider both querying and misclassification costs, and require the learning algorithm to achieve minimum total cost in the learning process. We therefore design an objective function to choose the query that yields minimum total cost in each learning iteration.

However, calculating this total cost of querying and misclassification is a bit tricky. In real-world applications, the learning model constructed on the current training data is often used for the future prediction, thus the "true" misclassification cost should also be calculated according to the future predicted examples. We assume that the rough size of such "to-be-predicted" data is known in this chapter, due to the following reason. In reality, the quantity of such "to-be-predicted" data directly affects the quantity of resource (effort, cost) that should be invested to construct the learning model. For instance, if the model would be used for only few times and on only limited unimportant data, it might not be worth to invest much resource on model construction; on the other hand, if the model is expected to be extensively used on a large amount of important data, it would be even more beneficial to improve the model performance by investing more resource. In many such real-world situations, in order to determine how much resource should be invested to construct the model, it is indeed known (or could be estimated) that how extensively the model would be used in the future (i.e., the rough quantity of the to-be-predicted data).

It is exactly the same case in our current scenario of generalized queries. More specifically, if the current learning model will only "play a small role" (i.e., make predictions on only few examples) in the future, it may not worth paying high querying cost to construct a high-performance model. On the other hand, if a large number of examples need to be predicted, it would be indeed worthwhile to acquire more generalized queries (at the expense of high querying cost), such that an accurate model with low misclassification cost could be constructed.

This indicates that, the number of "to-be-predicted" examples is crucial in minimizing total cost. Therefore, we formalized the total cost after t iterations (denoted by C_T^t) in Equation 4.2, where C_Q^i denotes the querying cost in the ith iteration, C_M^t denotes the misclassification cost after t iterations, which further can be calculated as the product of the average misclassification cost[4] after t iterations (denoted by $AvgC_M^t$) and the number of future predicted examples (denoted by n).

$$C_T^t = \sum_{i=1}^{t} C_Q^i + C_M^t = \sum_{i=1}^{t} C_Q^i + AvgC_M^t \times n \tag{4.2}$$

To obtain the minimum total cost for the learning model, we greedily choose the query that maximally reduces the total cost in each learning iteration. More formally, Equation 4.3 shows the objective function for searching for a query in iteration t, where all notations keep the same meaning as above.

$$\begin{aligned} q^t &= \arg\max_{q \in Q^t} (C_T^{t-1} - C_T^t(q)) \\ &= \arg\max_{q \in Q^t} ((AvgC_M^{t-1} - AvgC_M^t(q)) \times n - C_Q(q)) \end{aligned} \tag{4.3}$$

In the current setting, we assume that $C_Q(q)$ and n are both known, thus we need estimate $AvgC_M^{t-1}$ and $AvgC_M^t(q)$ separately, according to Equation 4.3. We again adopt the similar strategy as in the previous subsection. Specifically, $AvgC_M^{t-1}$ could be directly estimated by cross-validation or leave-one-out on the original training set, and $AvgC_M^t(q)$ can be optimistically estimated by assuming the label of q is certainly 0 and 1 respectively (see Section 4.3.1.2 for details).

4.3.1.3 Searching Strategy

Given the above two objective functions for two scenarios, the learner is required to search the query space and find the optimal one in each iteration.

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

[4]Average misclassification cost represents the misclassification cost averaged on each tested examples.

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 $2^d l$. This query space is thus quite large (grows exponentially to the feature dimension), and it is unrealistic to exhaustively evaluate every candidate. 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 a greedy search strategy, the computational 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 a local greedy search cannot guarantee finding the *true* optimal generalized query in the entire query space, but the empirical study (see Section 4.4) will show it still works effectively in most cases.

4.3.2 Updating Learning Model

After finding the optimal query in each iteration, the learner will request the corresponding label from the oracle, and update the learning model accordingly. However, the generalized queries often contain *don't-care features*, and the labels for such generalized queries are also likely to be *uncertain*. In this subsection, we study how to update the learning model by appropriately handling such don't-care features and

uncertain answers in the queries.

Roughly speaking, we consider the don't-care features as missing values, and handle the uncertain labels by taking partial examples in the learning process. More specifically, we simply treat the generalized queries with don't-care features as specific ones with missing values. As many learning algorithms (such as decision tree based algorithms, most generative models, and so on) have their own mechanisms to naturally handle missing values, this simple strategy can be widely applied. In terms of the uncertain labels of the queries, we handle them by taking partial examples in the learning process. For instance, given a query with an uncertain label (such as, 90% probability as 1 and 10% probability as 0), the learning algorithm simply takes 0.9 part of the example as certainly 1 and 0.1 part as certainly 0. Taking partial examples into learning is often implemented by *re-weighting* examples, which is also applicable to many popular learning algorithms.

This simple strategy can elegantly update the learning model. However, a pitfall of the strategy also occurs. When updating the learning model, the current strategy always regards one generalized query as only one specific example (with missing values). This might significantly degrade the power of the generalized queries. On the other hand, if one generalized query is regarded as too many specific examples, it might also overwhelm the original training data. Therefore, here we regard each generalized query as n (same) examples (with missing values), where n is suggested to be half of the initial training set size by the empirical study.

So far, we have proposed a novel method to construct the generalized query and update the learning model in each active learning iteration. In particular, we have designed two objective functions to balance the accuracy/cost trade-off and minimize the total cost of misclassification and querying. Figure 4.1 summarizes the framework of asking generalized queries in the current setting, and demonstrates the entire active learning learning procedure. In the following section, we will conduct experiments on real-world data sets, to empirically study the performance of the proposed algorithms.

4.4 Empirical Study

In this section, we empirically study the performance of the proposed algorithms on 15 real-world data sets from the UCI Machine Learning Repository [3], and compare them with the existing active learning algorithms.

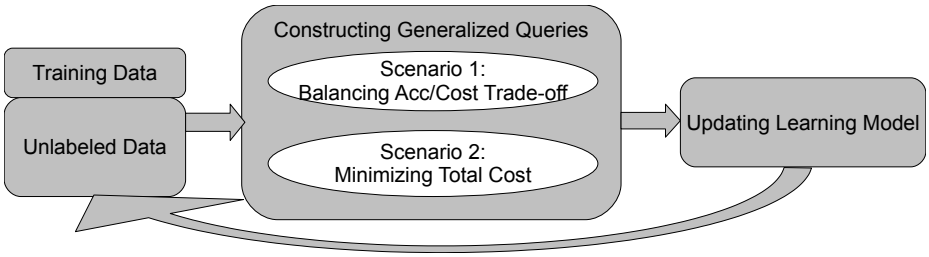


Figure 4.1: The framework of asking generalized queries in active learning.

4.4.1 Experimental Configurations

We compare the proposed algorithms with the traditional pool-based active learning (with uncertainty sampling[5]) [30] (denoted by "Pool") and the active learning with generalized queries [18] (denoted by "AGQ"). "Pool" and "AGQ" represent two special cases for querying cost: "Pool" only asks specific queries (with low querying cost), but cannot take advantage of the generalized queries to improve the predictive performance; on the other hand, "AGQ" tends to ask as general as possible queries to promptly improve the predictive performance, but with the expense of high querying cost. We expect that the proposed algorithms (for the two scenarios) can simultaneously maximize the predictive performance and minimize the querying cost, thus outperforming "Pool" and "AGQ".

All of the 15 UCI data sets have binary class and no missing values. Information on these data sets is tabulated in Table 4.2. 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 unlabeled data are labeled and included in the training set, we choose a small training set for each data set such that the "maximum reduction" of the error rate[6] is large enough (greater than 10%). The training sizes of the 15 UCI data sets range from 1/200 to 1/5 of the whole data sets, also listed in

[5]As we use an ensemble of bagged decision trees as the base learner in the experiments (see Sections 4.3.1.1 and 4.3.1.2), the most uncertain example can also be considered as the example with the maximum disagreement for the current committee (constructed by all the decision trees). Thus, uncertainty sampling in this case can also be regarded as an implementation of QBC [48, 21].

[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) subtracting the error rate on R 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.

Table 4.2. The unlabeled set is the whole data set taking away the test set and the training set.

Data Set	Type of Features	No. of Features	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
sick	nominal	27	3772	3541/231	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 4.2: The 15 UCI data sets used in the experiments.

In our experiments, we set the querying cost (C_Q) for any specific query as 1, and study the following three cost settings for generalized queries with r don't-care features, as follows:

- $C_Q = 1 + 0.5 \times r$: This setting represents a linear growth of C_Q with respect to r. For instance, the cost of asking a generalized query with two don't-care features is ($C_Q = 1+0.5\times 2 = 2$), which equals to the cost of asking two specific ones.

- $C_Q = 1 + 0.05 \times r$: This setting also represents a linear growth of C_Q with respect to r. However, the cost of asking generalized queries is rather low in this case. For instance, the cost of asking a generalized query with 20 don't-care features equals to the cost of asking two specific ones.

- $C_Q = 1+0.5\times r^2$: This setting represents a non-linear growth of C_Q with respect to r. The cost of asking generalized queries is extraordinary high in this case. For instance, the cost of asking a generalized query with only two don't-care features equals to the cost of asking three specific ones.

Note that, these settings of querying cost are only used here for empirically study, any other types of querying cost could be easily applied without changing the algorithms.

As for all the 15 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 regards each generalized query as a specific example with missing values, and provides the posterior class probability as the answer to the learner. 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.

4.4.2 Results for Balancing Acc./Cost Trade-off

In Scenario 1, we still use accuracy to measure the performance of the learning model. Thus, we use an ensemble of bagged decision trees [7] (implemented in Weka [26]) as the learning algorithm in the experiment, due to its generally good performance on classification [9]. Any other learning algorithms can also be implemented in real-world applications.

Figure 4.2 demonstrates the performance of the proposed algorithm considering only querying cost (denoted by "AGQ-QC"; see Section 4.3.1.1), compared with "Pool" and "AGQ" on a typical UCI data set "breast-cancer". We can see from the subfigures of Figure 4.2 that, with all the three querying cost settings, "AGQ-QC" can always effectively increase the predictive accuracy of the learning model with low querying cost, and outperform "Pool" and "AGQ". More specifically, in the case that ($C_Q = 1 + 0.5 \times r$), "AGQ-QC" significantly outperforms both "Pool" and "AGQ" during the entire learning process. In the case that ($C_Q = 1 + 0.05 \times r$), although "AGQ-QC" still outperforms the other two algorithm, it performs similarly to "AGQ". As the cost of asking generalized queries is rather low in this case, "AGQ-QC" tends to discover as more as possible don't-care features in the queries, thus producing similar predictive performance as "AGQ". In the case that ($C_Q = 1 + 0.5 \times r^2$), "AGQ-QC" still significantly outperforms the other algorithms. Note that, In this case, the cost of asking generalized queries is relatively high (i.e., grows quadratically with the number of don't-care feature), thus "AGQ" tends to discover as few as possible don't-care features, and consequently behaves similarly to "Pool".

To quantitatively compare the learning curves, we measure the actual values of the accuracies in 10 equal-distance points on the x-axis. The 10 accuracies of one curve are compared with the 10 accuracies of another using the two-tailed, paired t-test with

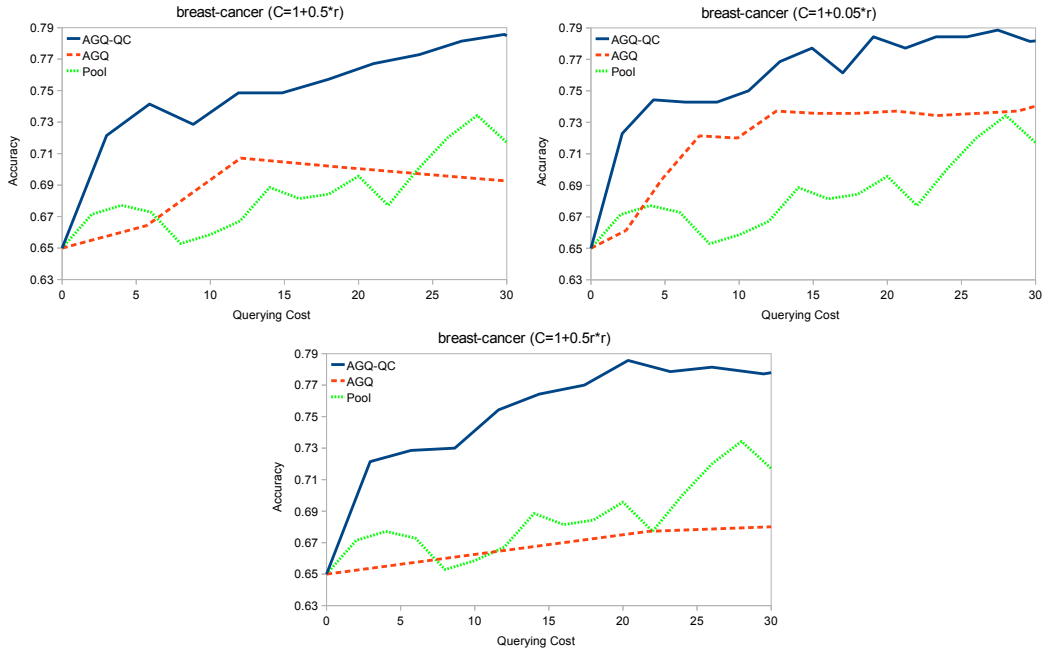


Figure 4.2: Comparison between "AGQ-QC", "AGQ" and "Pool" on a typical UCI data "breast-cancer", for balancing acc./cost trade-off.

95% confidence level. The t-test results on all the 15 UCI data sets with all the three querying cost settings are summarized in Table 4.3. Each entry in the table, $w/t/l$, means that the algorithm in the corresponding column wins on w, ties on t, and loses on l data sets, compared with the algorithm in the corresponding row. We can observe the similar phenomena from Table 4.3 that, "AGQ-QC" significantly outperforms "AGQ" when the querying cost is relatively high (such as, $C_Q = 1 + 0.5 \times r^2$ and $C_Q = 1 + 0.5 \times r$), and significantly outperforms "Pool" when the querying cost is relatively low (such as, $C_Q = 1 + 0.05 \times r$).

	AGQ-QC		
	$C = 1 + 0.5 \times r$	$C = 1 + 0.05 \times r$	$C = 1 + 0.5 \times r^2$
Pool	6/7/2	10/4/1	5/6/4
AGQ	14/0/1	6/7/2	15/0/0

Table 4.3: Summary of the t-test on the 15 UCI data sets and with three querying cost settings, for balancing acc./cost trade-off.

4.4.3 Results for Minimizing Total Cost

In Scenario 2, we use total cost (i.e., the sum of querying and misclassification costs) to measure the performance of the learning model. Thus, we use a cost-sensitive algorithm *CostSensitiveClassifier* based on an ensemble of bagged decision trees (also implemented in Weka [26]) as the learning algorithm in the experiments. Any other cost-sensitive learning algorithms can also be implemented in real-world applications. In addition, we set the false negative (FN) and false positive (FP) costs as 2 and 10 respectively[7], and we set the number of the future predicted examples as 1000. Still, any other settings can be easily applied without changing the algorithm.

Figure 4.3 demonstrates the performance of the proposed algorithm considering total cost (denoted by "AGQ-TC"; see Section 4.3.1.2), compared with "Pool" and "AGQ" on the same UCI data set "breast-cancer". We can see from Figure 4.3 that "AGQ-TC" effectively decreases the total cost of the learning model, and significantly outperforms "Pool" and "AGQ", with most querying cost settings. More specifically, we can discover the similar pattern between "AGQ-TC" and "AGQ" as in the previous subsection. When the querying cost is relatively low (such as $C_Q = 1 + 0.05 \times r$), "AGQ-TC" and "AGQ" tend to perform similarly; on the other hand, when the querying cost is relatively high (such as $C_Q = 1 + 0.5 \times r^2$), "AGQ-TC" often significantly outperforms "AGQ".

In addition, we can also notice from Figure 4.3 that, instead of keeping decreasing the total cost, the learning algorithm might also increase the total cost when obtaining more examples. This is reasonable, especially when the querying cost is relatively high. When the new examples are obtained in active learning, the querying cost is increased constantly; if these examples fail to reduce even more misclassification cost, the total cost is consequently increased.

The t-test results (similar to Table 4.3) on the 15 UCI data sets are summarized in Table 4.4. It clearly shows that, "AGQ-TC" performs significantly better than "AGQ" on most (or even all) tested data sets, when the querying cost is relatively high (such as, $C_Q = 1 + 0.5 \times r^2$ and $C_Q = 1 + 0.5 \times r$). When compared with "Pool", "AGQ-TC" still wins (or at least ties) on a majority of tested data sets,

[7] In general cost-sensitive problem, the absolute FN and FP costs would not affect the predictive results, as long as the cost ratio is fixed. However, here we also take querying cost into consideration, thus setting the absolute values for FN and FP cost is necessary.

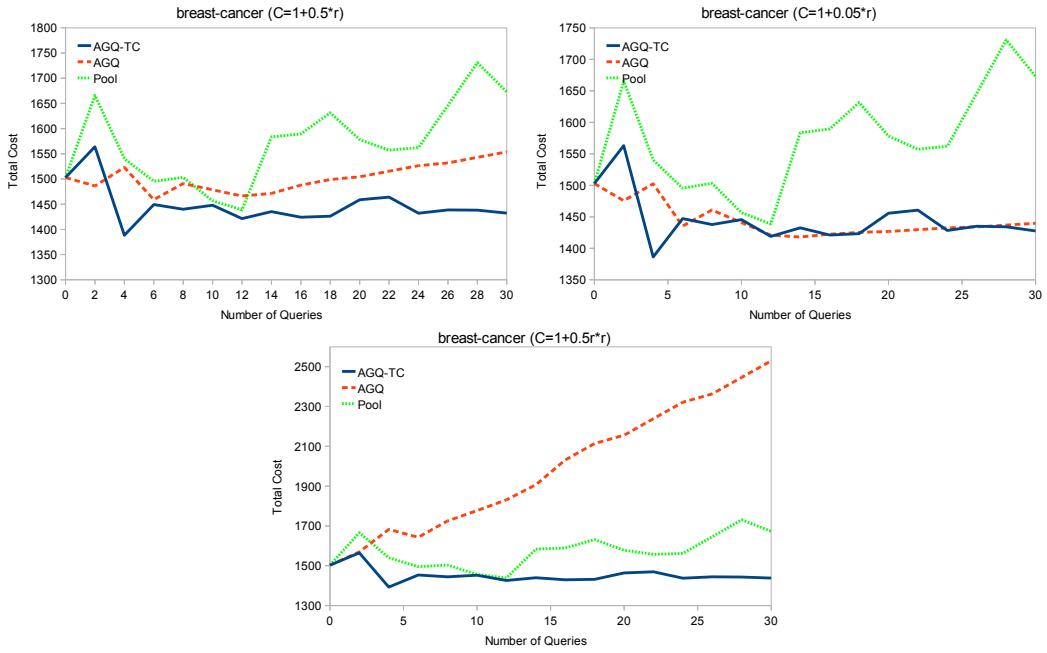


Figure 4.3: Comparison between "AGQ-TC", "AGQ" and "Pool" on a typical UCI data "breast-cancer", for minimizing total cost.

especially when the querying cost is relatively low (such as, $C_Q = 1 + 0.05 \times r$). These experimental results clearly indicate that "AGQ-TC" can indeed significantly decrease the total cost, and outperforms "AGQ" and "Pool" in the learning process.

	AGQ-TC		
	$C = 1 + 0.5 \times r$	$C = 1 + 0.05 \times r$	$C = 1 + 0.5 \times r^2$
Pool	6/7/2	10/4/1	6/6/3
AGQ	15/0/0	6/6/3	15/0/0

Table 4.4: Summary of the t-test on the 15 UCI data sets and with three querying cost settings, for minimizing total cost.

4.4.4 Approximate Probabilistic Answers

In the previous experiments, we have assumed that the oracle is always capable of providing accurate probabilistic answers for the generalized queries. However, in real-world situations, it is more common that only "approximate probabilistic answers" are provided (especially when the oracles are human experts). We speculate that small perturbations in the probabilistic answers will not dramatically affect the performance

of the proposed algorithms. This is because small perturbations in label probabilities only represent light noises of examples included in the training set. These light noises could be cancelled out in the successive updates of the training set. With a robust base learning algorithm (such as the bagged decision trees), such small noises would be insensitive. In this subsection, we study this issue experimentally.

To simulate the approximate probabilistic answer, we first calculate the exact accurate probabilistic answer from the target model, and then randomly alter it with up to 20% noise (increase or decrease by up to 20% uniformly distributed random noise). Figures 4.4 and 4.5 demonstrate the performance the proposed algorithms with such approximate probabilistic labels (denoted by "AGQ-QC (appr)" and "AGQ-TC (appr)" receptively), compared with the original "AGQ-QC" and "AGQ-TC", with the setting ($C_Q = 1 + 0.5 \times r$) and on the typical data ("breast-cancer").

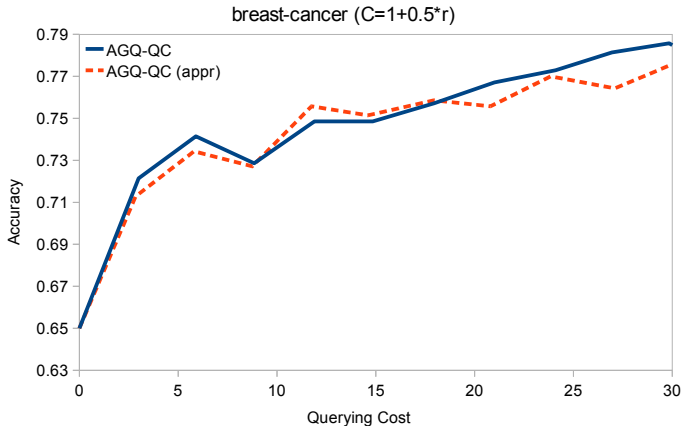


Figure 4.4: Comparison of the performance between "AGQ-QC" and "AGQ-QC (appr)" (with up to 20% noise) on "breast-cancer".

We can clearly see from these figures that, when only the approximate probabilistic answers are provided by the oracle, the performance of the proposed algorithms are not significantly affected. The similar experimental results can be shown with other settings and on other data sets. This indicates that, the proposed algorithms are rather robust with such more realistic approximate probabilistic answers, thus can be directly deployed in real-world applications.

To conclude, we summarize observations from the experimental results, as follows:

- In general, according to the two scenarios, the proposed algorithms ("AGQ-QC"

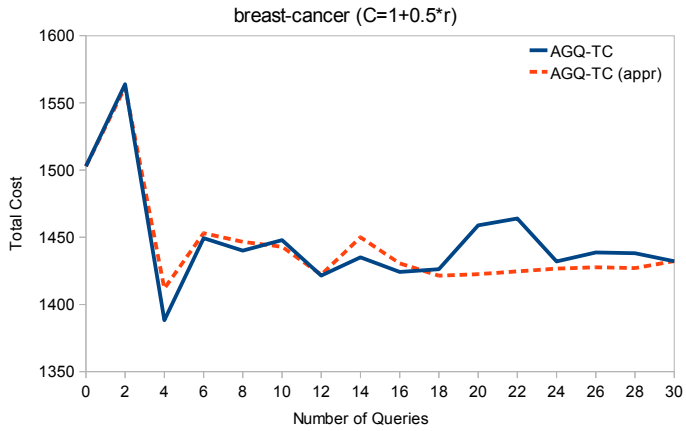


Figure 4.5: Comparison of the performance between "AGQ-TC" and "AGQ-TC (appr)" (with up to 20% noise) on "breast-cancer".

and "AGQ-TC") are indeed superior to the previous active learning algorithms, with all the querying cost settings, and on most tested UCI data sets.

- In terms of balancing accuracy/cost trade-off (in scenario 1), the proposed algorithm ("AGQ-QC") can comprehensively beat "AGQ" when the querying cost is relatively high, and beat "Pool" when the querying cost is relatively low.
- In terms of minimizing total cost (in scenario 2), the proposed algorithm ("AGQ-TC") can outperform "Pool" in most cases (especially with low querying cost), and significantly outperform "AGQ" with high querying cost.
- Both of the proposed algorithms can perform robustly, when only approximate (instead of accurate) probabilistic answers are provided for the generalized queries.

4.5 Summary

In this chapter, we assume that the oracles are capable of answering generalized queries with non-uniform costs, and study active learning with generalized queries in cost-sensitive framework. We propose a novel method to construct the generalized query and update the learning model in each active learning iteration. In particular, we design two objective functions to choose generalized queries in the learning process, so as to either balance the accuracy/cost trade-off or minimize the total cost of

misclassification and querying. The empirical study in a variety of settings verifies the superiority of the proposed methods over the existing active learning algorithms, in terms of simultaneously maximizing the predictive performance and minimizing the querying cost.

In our future work, we will design global search strategies (instead of the local search in this chapter) to find the optimal generalized queries in each learning iteration, thus further improving the performance of the proposed algorithms.

Chapter 5

Asking Generalized Queries to Ambiguous Oracle

5.1 Introduction

In both of the previous two chapters, we implicitly assume that the oracle is capable of providing probabilistic answers for the generalized queries. For instance, given a prostatitis patient data set, we suppose a specific query (i.e., a specific patient example) is {ID = 7354288, Name = John, Age = 50, Gender = male, Weight = 230, Blood-Type = AB, Fever = yes, Urination-Pain = yes, $\cdots$} (with all the features), and the oracle is required to diagnose whether this patient has prostatitis or not. A generalized query, therefore, might be "are men between age 45 and 55 with fever and painful urination likely to have prostatitis?", where only four features (gender, age, fever and urination-painful) are provided. It is assumed in previous chapters that the oracle is able to provide (accurate) probabilistic answers for such generalized queries, as "Yes, with 80% probability". This requirement, however, is usually too stringent in reality.

Instead, generalized queries are often answered with ambiguous answers in real-world situations. For instance, if the query is "are women (or boys under age 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 age 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 chapter, 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 such ambiguous answers has been extensively studied in *multiple-instance learning* [15], and applied to many real-world applications, such as drug activity prediction [15], content-based image retrieval [60] and text categorization [1]; see Section 5.2 for more details.

In an active learning scenario, such a 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 out to be a specific one, such a "Yes-No" response naturally becomes the *accurate* answer to the specific query. Therefore, such a setting of ambiguous answers is more flexible than the regular setting in active learning, yet still applicable in many real-world situations.

In this chapter, 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 ([3]) 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 chapter is organized as follows. Section 5.2 reviews previous works on active learning and multiple-instance learning. Section 5.3 describes our algorithm to ask generalized queries and improve the learning model with ambiguous answers. In Section 5.4, empirical study is conducted on real-world UCI data sets to verify the superiority of the proposed method. Section 5.5 presents conclusions.

5.2 Related Work

Most previous works on active learning assume that the oracle could only answer specific queries, with all features provided. We previously 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 [18, 19]. 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* ([15]), where examples are given in bags and the oracle is only required to provide one ambiguous answer for each bag. More specifically, given a bag of unlabeled examples, the oracle will respond negatively *if (and only if)* these examples are all negative, and respond positively otherwise. In this setting, it is more likely for the learner to be responded to with a positive answer; and more importantly, such a 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* [36], *citation kNN* [55], *multiple-decision tree* [62] and *multiple-instance logistic regression* [42], 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 [15], content-based image retrieval [60], and text categorization [1].

In this chapter, we apply such an 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 responding with ambiguous answers. That is, given a generalized query, the oracle will respond negatively *if (and only if)* the examples represented by the query are all negative, and respond positively otherwise. Such a 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 chapter is not a simple combination

of active learning and multiple instances learning.[1] Instead of aiming to improve the predictive performance on the *unseen bags of examples* in multiple-instance learning, in this chapter, we still attempt to improve the predictive performance on the *unseen specific examples* (as in a traditional supervised setting). In addition, we also consider generalized queries in an active learning scenario, thus the problem we are attempting to solve is more complex and difficult.

5.3 Algorithm

In this section, we design an 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 its good performance in probability estimation and the convenience of designing an objective function (see Section 5.3.1 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 a certain objective function).
- **Step 2:** After obtaining the ambiguous answer of the generalized query, the learner updates the learning model (according to a 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.

5.3.1 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;

[1] Settles et al. [47] categorizes multiple-instance active learning into four scenarios, and develops a novel algorithm to deal with one of those formulations.

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:

$$< q, \mathbf{w} >_{opt} = \arg\max_{q,\mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q, \mathbf{w}) \tag{5.1}$$

where, $< q, \mathbf{w} >_{opt}$ denotes the tuple of optimal query q and model parameter $\mathbf{w}$, $\mathbf{x}_i$ and y_i denote the ith 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 5.1) 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:

$$\begin{aligned} < q, \mathbf{w} >_{opt} &= \arg\max_{q,\mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q, \mathbf{w}) - \alpha \sum_{\mathbf{x}_j \in U, y \in \{0,1\}} H(p(y | \mathbf{x}_j; q, \mathbf{w})) \\ &= \arg\max_{q,\mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q, \mathbf{w}) \\ &\quad + \alpha \sum_{\mathbf{x}_j \in U} \sum_{y \in \{0,1\}} p(y | \mathbf{x}_j; q, \mathbf{w}) \log p(y | \mathbf{x}_j; q, \mathbf{w}) \end{aligned} \tag{5.2}$$

where $\mathbf{x}_j$ denotes the jth 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.[2]

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 to with ambiguous answers by the oracle. Equation 5.2 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 tth 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:

$$\begin{aligned} < q_t, \mathbf{w}_t >_{opt} = \arg\max_{q,\mathbf{w}} \quad & \alpha_1 \sum_{(\mathbf{x}_i, y_i) \in D} \log p(y_i | \mathbf{x}_i; q_t, \mathbf{w}_t) \\ + \quad & \alpha_2 \sum_{(q_k, y_k) \in Q} \log p(y_k | q_k; q_t, \mathbf{w}_t) \\ + \quad & \alpha_3 \sum_{\mathbf{x}_j \in U} \sum_{y \in \{0,1\}} p(y | \mathbf{x}_j; q_t, \mathbf{w}_t) \log p(y | \mathbf{x}_j; q_t, \mathbf{w}_t) \end{aligned} \tag{5.3}$$

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 5.3.2 for details), Equation 5.3 can be regarded as a univariate function of the model parameter $\mathbf{w}$ (denoted by $f(\mathbf{w})$). Thus, gradient decent 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}).

[2]The similar objective function has been applied in semi-supervised learning [22] and batch mode active learning [24]; and these previous studies have shown its applicability in real-world applications.

- 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 5.3 is optimized in a similar way, and the optimal model parameter ($\mathbf{w}_{opt}$) can be determined.

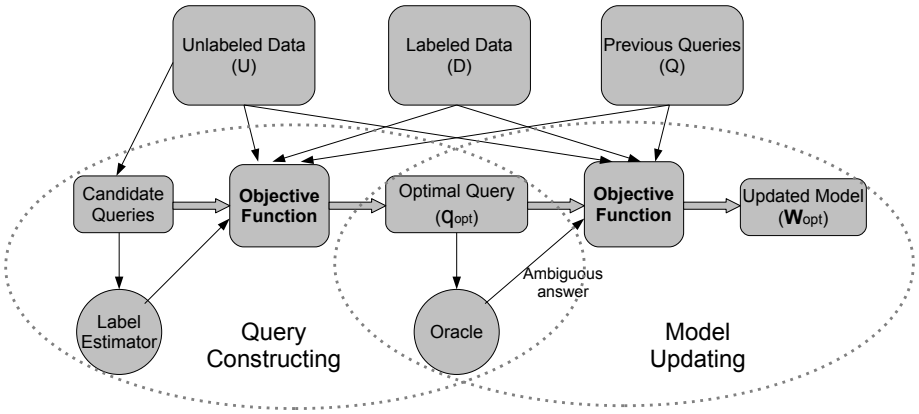


Figure 5.1: The framework of the proposed algorithm.

This entire process is also illustrated in Figure 5.1. However, there are still some issues to be solved during this learning process, such as, how to search for 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.

5.3.2 Constructing Generalized Queries

In each active learning iteration, constructing the optimal generalized query can be implemented by searching for the best one in the query space (according to Equation 5.3). 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 $2^d l$. 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 a greedy search strategy, the computational 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 a local greedy search cannot guarantee finding the *true* optimal generalized query in the entire query space, but the empirical study (see Section 5.4) will show it still works effectively in most cases.

With greedy search, all these candidate queries are expected to be evaluated by Equation 5.3, 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 5.3.3. 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 to with ambiguous answers (negative if all the examples represented are negative, and positive otherwise), we also attempt to estimate the probability of such an ambiguous answer. More specifically, we suppose that each generalized query with n don't-care features can be represented by 2^n specific examples,[3] and each of these specific examples is still equally likely to be positive or negative. Thus the probability of such a generalize query being negative would be calculated 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, 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 5.3. The optimal query thus could be discovered.

5.3.3 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 5.3). However, as the active learner always tends to ask generalized queries, and is always responded to 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

$$p(y=1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} \tag{5.4}$$

Therefore, in Equation 5.3, the part to maximize the log likelihood on the original training data D (i.e., the first term in Equation 5.3) 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 5.3). 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 5.3), due to the generalization of these queries and the ambiguity of the answers. We will solve this issue in this subsection.

[3]See Section 5.3.3 for details why and how each generalized query can be represented by this many examples.

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

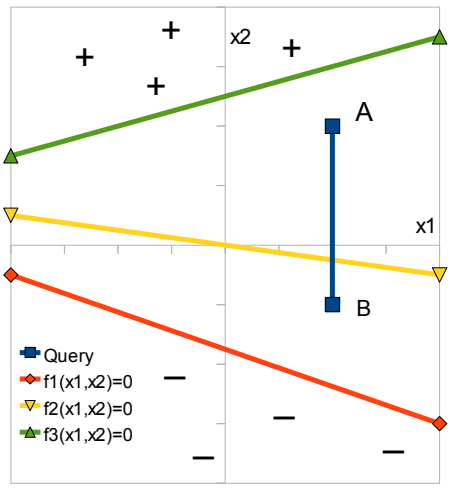
- We first specify each generalized query with a set of representative examples; the posterior probability for each of these examples thus can be presented as in Equation 5.4.
- Then, we combine the probabilities of all these representative examples together, to form the probability for the corresponding 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 seperator (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 instance, we suppose "temperature" is the only don't-care feature, and its valid range is $[94F, 108F]$. Thus, a generalized query {Age = 65, Gender = male, Temperature = $*$, $\cdots$} will definitely be labeled as negative, if (and only if) the two specific examples, {Age = 65, Gender = male, Temperature = 94F, $\cdots$} and {Age = 65, Gender = male, Temperature = 108F, $\cdots$}, are both negative. This intuition is illustrated in Figure 5.2.

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 the query can be easily determined.[4] Therefore, we can simply represent such a generalized query by these

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



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 "$f_1(x_1, y_1) = 0$", "$f_2(x_1, x_2) = 0$" 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, "$f_3(x_1, x_2)$" can simultaneously label both "A" and "B" as negative[a], it consequently labels the query as negative; whereas both "$f_1(x_1, x_2)$" and "$f_2(x_1, x_2)$" 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.

Figure 5.2: An illustration for representing generalized queries with specific examples.

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 also be presented through Equation 5.4. Next, we will combine all these probabilities together, to form the posterior probability for the corresponding generalized query.

As we have introduced in Section 5.1, in the current setting, the ambiguous oracle provides a negative answer only when all the corresponding specific examples are negative, and provides a 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 [36]

$$p(y=1|q) = 1 - \prod_{k=1}^{n}(1 - p(y=1|\mathbf{x}_k)) \tag{5.5}$$

to form the probability of the generalized query, where q denotes the generalized query and $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ denotes the corresponding n representative examples. The probability for any generalized query therefore can be specified.[5]

To formalize, given the optimal generalized query (and the true ambiguous label) in each iteration, by combining Equations 5.3, 5.4 and 5.5, the optimal model parameter $\mathbf{w}$ can be determined by minimizing the following error function:

$$\begin{aligned} E(\mathbf{w}) \quad = \quad & -\alpha_1 \sum_{(\mathbf{x},y)\in D} \{y_i \log \sigma(\mathbf{w}^T\mathbf{x}_i) + (1-y_i)\log(1-\sigma(\mathbf{w}^T\mathbf{x}_i))\} \\ & -\alpha_2 \sum_{(q_k,y_k)\in Q} \{y_k \log(1 - \prod_{\mathbf{x}_{ki}\in q_k}(1-\sigma(\mathbf{w}^T\mathbf{x}_{ki})) + (1-y_k)\sum_{\mathbf{x}_{ki}\in q_k}\log(1-\sigma(\mathbf{w}^T\mathbf{x}_{ki}))\} \\ & -\alpha_3 \sum_{\mathbf{x}_j\in U} \{\sigma(\mathbf{w}^T\mathbf{x}_j)\log\sigma(\mathbf{w}^T\mathbf{x}_j) + (1-\sigma(\mathbf{w}^T\mathbf{x}_j))\log(1-\sigma(\mathbf{w}^T\mathbf{x}_j))\} \end{aligned} \tag{5.6}$$

where $\sigma(\mathbf{w}^T\mathbf{x}_i)$ can be further specified by Equation 5.4, 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:

$$\begin{aligned} \nabla E(\mathbf{w}) \quad = \quad & -\alpha_1 \sum_{(\mathbf{x}_i,y_i)\in D} \{(y_i - \sigma(\mathbf{w}^T\mathbf{x}_i))\mathbf{x}_i\} \\ & -\alpha_2 \sum_{(q_k,y_k)\in Q} \{\frac{(1-\prod_{\mathbf{x}_{ki}\in q_k}(1-\sigma(\mathbf{w}^T\mathbf{x}_{ki})) - y_k)\sum_{\mathbf{x}_{ki}\in q_k}\sigma(\mathbf{w}^T\mathbf{x}_{ki})\mathbf{x}_{ki}}{1-\prod_{\mathbf{x}_{ki}\in q_k}(1-\sigma(\mathbf{w}^T\mathbf{x}_{ik}))}\} \\ & -\alpha_3 \sum_{\mathbf{x}_j\in U} \{(\log\frac{\sigma(\mathbf{w}^T\mathbf{x}_j)}{1-\sigma(\mathbf{w}^T\mathbf{x}_j)})\sigma(\mathbf{w}^T\mathbf{x}_j)(1-\sigma(\mathbf{w}^T\mathbf{x}_j))\mathbf{x}_j\} \end{aligned} \tag{5.7}$$

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.

In the next section, we will conduct empirical study to discuss the setting for the

[5]Note that, other combining functions, such as the softmax function [42] $p(y=1|q) = \frac{\sum_{k=1}^{n} p(y=1|\mathbf{x}_k)e^{p(y=1|\mathbf{x}_k)}}{\sum_{k=1}^{n} e^{p(y=1|\mathbf{x}_k)}}$, can also be applied to form the probability of the generalized queries.

trade-off parameters α_1, α_2 and α_3, and compare the proposed algorithm with the traditional ones.

5.4 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 [3].

5.4.1 Experimental Configurations

We conduct experiments with two settings of the trade-off parameters (α_1, α_2 and α_3) for Equation 5.3. 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 5.3 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 example 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: $\alpha_1 = 1/|D|$, $\alpha_2 = 1/|Q|$[6], and $\alpha_3 = 1/|U|$, where $|D|$, $|Q|$ and $|U|$ denote the size of the corresponding data (query) sets (the corresponding algorithm is denoted by "AL-GQA(n)").

We also conduct experiments on active learning with specific queries (*uncertainty sampling* [30]; 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

[6]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.

queries, and always responds 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 always respond 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 5.1. 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 unlabeled data are labeled and included in the training set, we choose a small training set for each data set such that the "maximum reduction" of the error rate[7] 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 5.1. The unlabeled set is the whole set taking away the test set and the training set.

Data Set	No. of Features	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

Table 5.1: The seven UCI data sets used in the experiments.

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.

[7]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.

5.4.2 Experimental Results

Based on the seven tested UCI data sets, Figure 5.3 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:

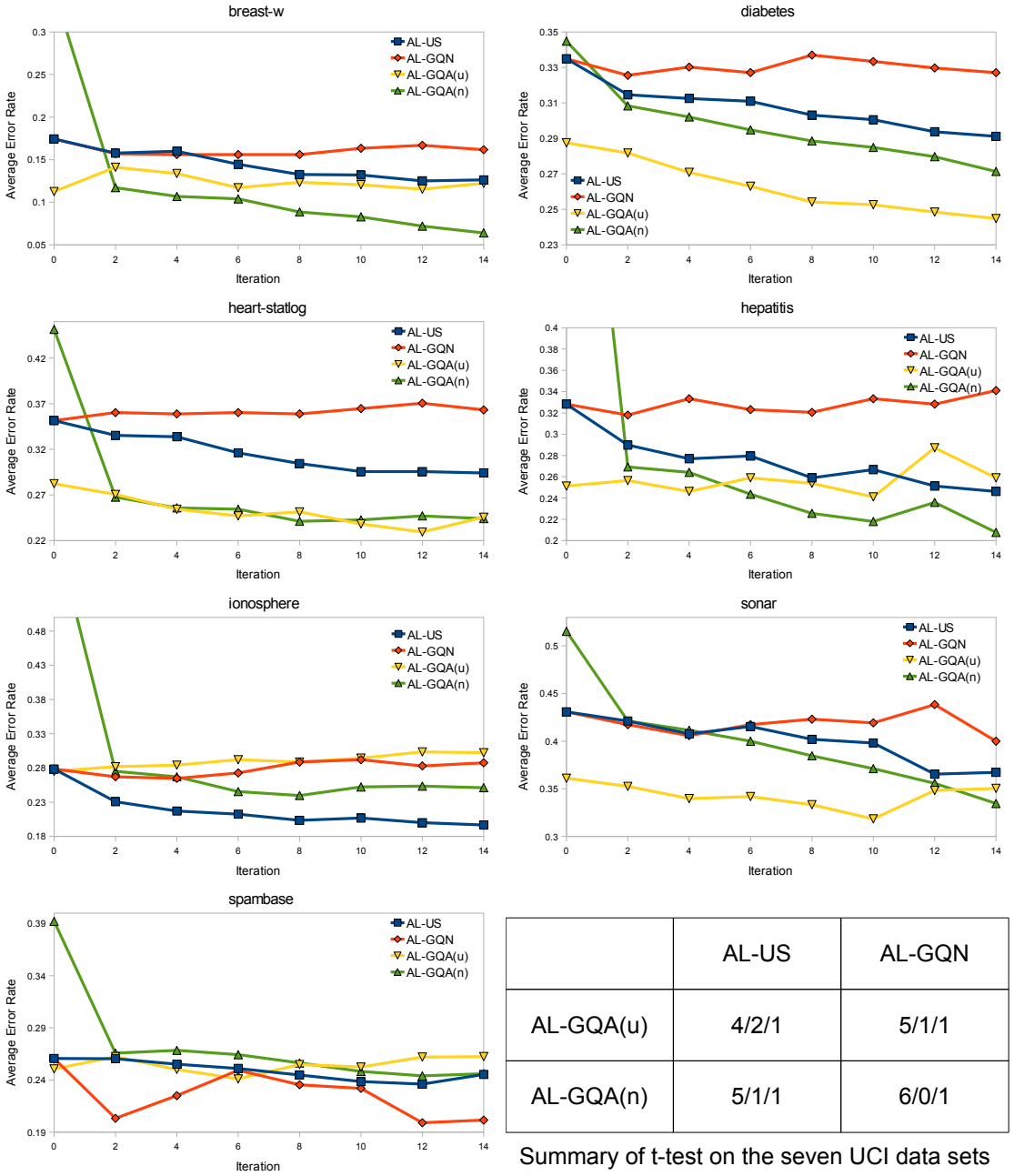
- On most tested data sets, both "AL-GQA(u)" and "AL-GQA(n)" perform significantly better than "AL-US". This demonstrates the superiority of active learning with generalized queries and ambiguous answers to the traditional specific-query based learning.
- On most tested data sets, both "AL-GQA(u)" and "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.
- "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 demonstrate 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.

5.5 Summary

In this chapter, 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 this 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.



	AL-US	AL-GQN
AL-GQA(u)	4/2/1	5/1/1
AL-GQA(n)	5/1/1	6/0/1

Summary of t-test on the seven UCI data sets

Figure 5.3: Learning curves and summary of t-test of the four algorithms on seven UCI data sets.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we propose a new active learning scheme — active learning with generalized queries. In contrast to the traditional active learning with specific queries, in this new learning scheme, the learner can ask generalized queries in each iteration, and the oracle is capable of answering such queries. As one generalized query can usually represent a set of specific ones, the corresponding answer provided by the oracle is also applicable to this whole set of specific queries. Therefore, the active learner can obtain more information from each generalized query (and the corresponding answer), and furthermore improve the learning more effectively and efficiently.

We thoroughly study active learning with generalized queries from four aspects and draw the corresponding conclusions, as follows.

In Chapter 2, we demonstrate the superiority of generalized queries over specific one in active learning, through toy examples and learning theory. More specifically,

- When data contain irrelevant features, it can be proved that the query complexity for active learning with generalized queries is significantly lower than active learning with specific ones.
- When data contain no irrelevant features, we consider two types of target models: decision trees (decision rules) and linear functions. It can be shown that, with both of these two target models, generalized queries can still significantly speed up the learning process, compared with specific ones.

In Chapter 3, we assume that the oracle can answer generalized queries as easily as specific ones (i.e., with the same effort or cost), and develop two novel active learning algorithms to improve learning by asking generalized queries. More specifically,

- We propose a four-step strategy (AGQ) for the active learner to ask the generalized queries and efficiently improve the learning model.
- We extend AGQ to a more powerful algorithm AGQ$^+$, such that more flexible generalized queries can be constructed and the learning performance can be further improved.
- Empirical study shows AGQ can achieve same accuracy with significant fewer queries (36% fewer on average) compared with traditional active learning with specific queries, and AGQ$^+$ can perform even better on some cases.
- The AGQ algorithm is quite robust, and can tolerate a low level of noise in the probability estimation of oracle answers.
- Despite the similarity of AGQ and active learning with feature selection, the former can perform significantly better than the latter on most cases.
- Combined with an extra heuristic, AGQ can perform equivalently well even when the initial training set contains only few labeled examples.

In Chapter 4, we assume that, the more general the query is, the higher cost (effort) it causes for the learner to request the label. We study the generalized queries in a cost-sensitive framework to balance the trade-off of the predictive accuracy and the query cost or minimize the total cost of misclassification and query. More specifically,

- We consider querying cost only, and accordingly propose an algorithm to achieve high predictive accuracy by paying as low as possible querying cost.
- We consider both querying and misclassification cost, and accordingly propose an algorithm to achieve minimum total cost of querying and misclassification.
- Empirical study shows that, the propose algorithms can significantly outperform the traditional active learning with specific or generalized queries.

- Both of the proposed algorithms can perform robustly when only approximate probabilistic answers are provided for the generalized queries.

In Chapter 5, we assume that the oracle can only provide *ambiguous* answers to generalized queries (i.e., positive if at least one corresponding specific example is positive, and negative otherwise), and aim to improve active learning with such ambiguous answers. More specifically,

- We develop a novel algorithm to ask generalized queries and update the learning model with the corresponding ambiguous answers in active learning.
- Empirical study 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.

6.2 Future Work

In out future work, we plan to further study active learning with generalized queries from the following two aspects.

First, we would further theoretically study the effect of generalized queries in active learning. More specifically, we plan to derive more detailed query complexity of active learning with generalized queries (given different hypothesis spaces), and compare it with traditional active learning with specific queries. The advantage (and possible disadvantage) of generalized queries are expected to be thoroughly analysed.

Second, we plan to collaborate with a human specialist and apply active learning with generalized queries to a real-world medical diagnosis and symptoms analysis application. The learning algorithm is required to propose reasonable generalized queries during the learning process, and the specialist is required to answer these queries as an oracle. Such active learning with generalized queries is expected to significantly speed up the entire learning process, and eventually build up a automatic diagnosis system.

References

[1] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 561–568, 2003.

[2] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[3] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007. URL `http://archive.ics.uci.edu/ml/`.

[4] Yoram Baram, Ran El-Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[6] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM.

[7] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[8] Peter Buneman and Sushil Jajodia, editors. *Mining Association Rules between Sets of Items in Large Databases*, Washington, D.C., FebruaryJune–FebruaryAugust˜ 1993.

[9] Rich Caruana and Alexandru N. Mizil. An empirical comparison of supervised learning algorithms. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 161–168, New York, NY, USA, 2006. ACM.

[10] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[11] Wenyuan Dai, Qiang Yang, Gui R. Xue, and Yong Yu. Boosting for transfer learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 193–200, New York, NY, USA, 2007. ACM.

[12] Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems 17*, pages 337–344, 2004.

[13] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems 18*, 2005.

[14] Sanjoy Dasgupta, Adam T. Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. *Journal of Machine Learning Research*, 10:281–299, 2009.

[15] Thomas G. Dietterich, Richard H. Lathrop, and Tomas L. Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, January 1997.

[16] Pinar Donmez, Jaime G. Carbonell, and Paul N. Bennett. Dual strategy active learning. In *ECML '07: Proceedings of the 18th European conference on Machine Learning*, pages 116–127, Berlin, Heidelberg, 2007. Springer-Verlag.

[17] Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602, New York, NY, USA, 2008. ACM.

[18] Jun Du and Charles X. Ling. Active learning with generalized queries. In *Proceedings of the 9th IEEE International Conference on Data Mining*, pages 120–128, 2009.

[19] Jun Du and Charles X. Ling. Asking generalized queries to domain experts to improve learning. *IEEE Transactions on Knowledge and Data Engineering*, 22 (6):812–825, 2010.

[20] Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[21] Yoav Freund, Sebastian H. Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3): 133–168, 1997.

[22] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, volume 17, pages 529–536, 2005.

[23] Carlos Guestrin. Pac-learning, vc dimension and margin-based bounds. *Lecture Notes, Carnegie Mellon University*, 2007.

[24] Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 593–600, 2008.

[25] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

[26] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[27] Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. Large-scale text categorization by batch mode active learning. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 633–642, New York, NY, USA, 2006. ACM.

[28] Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 417–424, New York, NY, USA, 2006. ACM.

[29] A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *Proceedings of International Joint Conference on Articial Intelligence (IJCAI)*, pages 877–882, 2007.

[30] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 148–156, New Brunswick, US, 1994. Morgan Kaufmann Publishers, San Francisco, US.

[31] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.

[32] Hang Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

[33] Huan Liu and Hiroshi Motoda, editors. *Computational Methods of Feature Selection*. Data Mining and Knowledge Discovery. Chapman & Hall/CRC, 1 edition, October 2007.

[34] Ying Liu. Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of Chemistry Information and Computer Science*, 44:1936–1941, 2004.

[35] Dragos D. Margineantu. Active cost-sensitive learning. In *the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[36] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, volume 10, pages 570–576, 1998.

[37] Andrew Mccallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 350–358, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[38] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1 edition, March 1997.

[39] Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, pages 79+, New York, NY, USA, 2004. ACM.

[40] Foster Provost and Pedro Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, September 2003.

[41] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 759–766, New York, NY, USA, 2007. ACM.

[42] Soumya Ray and Mark Craven. Supervised versus multiple instance learning: an empirical comparison. In Luc De Raedt, Stefan Wrobel, Luc De Raedt, and Stefan Wrobel, editors, *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 697–704. ACM, 2005.

[43] Nicholas Roy and Andrew Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA, 2001.

[44] M. Saar-Tsechansky and F. Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, February 2004.

[45] B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, 2008.

[46] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[47] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 1289–1296, 2008.

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

[49] 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.

[50] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care*, pages 261–265. IEEE Computer Society Press, 1988.

[51] Cynthia A. Thompson, Mary E. Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *Proc. 16th International Conf. on Machine Learning*, pages 406–414. Morgan Kaufmann, San Francisco, CA, 1999.

[52] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM.

[53] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2: 45–66, 2002.

[54] Gokhan Tur, Dilek Hakkani-Tür, and Robert E. Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186, February 2005.

[55] Jun Wang and Jean D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1119–1126, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[56] Manfred K. Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active learning with support vector machines in the drug discovery process. *Journal of Chemical Information and Computer Sciences*, 43, 2003.

[57] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.

[58] Rong Yan, Jie Yang, and Alexander Hauptmann. Automatically labeling video data using multi-class active learning. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 516+, Washington, DC, USA, 2003. IEEE Computer Society.

[59] Cha Zhang and Tsuhan Chen. An active learning framework for content-based information retrieval. *Multimedia, IEEE Transactions on*, 4(2):260–268, 2002.

[60] Qi Zhang, Sally A. Goldman, Wei Yu, and Jason Fritts. Content-based image retrieval using multiple-instance learning. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 682–689, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[61] Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proc. 17th International Conf. on Machine Learning*, pages 1191–1198, 2000.

[62] Jean-Daniel Zucker and Yann Chevaleyre. 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*, pages 204–214. Springer-Verlag, 2001.

Vita

Name	Jun Du
Post-secondary Education and Degrees	China University of Geosciences, China 1996–2000 B.Sc. China University of Geosciences, China 2002–2005 M.Sc. The University of Western Ontario, Canada 2007–2011 Ph.D.
Honours and Awards	Ontario Graduate Scholarship, 2009-2011 Western Graduate Research Scholarship, 2007-2011 First Place, AI Session at UWO Research in Computer Science, 2011 UWO Graduate Thesis Research Award, 2010 Ontario Graduate Scholarship in Science and Technology, 2008-2009 Canadian Conference on Artificial Intelligence Travel Award, 2008
Related Work Experience	Lecturer The University of Western Ontario, Canada 2011–2011 Research Assistant The University of Western Ontario, Canada 2007–2011 Teaching Assistant The University of Western Ontario, Canada 2007–2011

Publications

1. Jun Du, Charles X. Ling, and Z.-H. Zhou, "When Does Co-training Work in Real Data?" IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), vol. 23, no. 5, pp. 788-799, May 2011.

2. Jun Du and Charles X. Ling. "Asking Generalized Quereis to Minimize Cost". Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), May 2011.

3. Jun Du and Charles X. Ling. "Active Teaching for Inductive Learners". Proceedings of the 11th SIAM International Conference on Data Mining (SDM), April 2011.

4. Jun Du and Charles X. Ling, "Active Learning with Human-Like Noisy Oracle". Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), December 2010.

5. Jun Du, Eileen Ni, and Charles X. Ling, "Adapting Cost-sensitive Learning to Reject Option". Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM), October 2010.

6. Jun Du and Charles X. Ling, "Asking Generalized Queries to Ambiguous Oracle". Proceedings of the 2010 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), September 2010.

7. Jun Du and Charles X. Ling, "Asking Generalized Queries to Domain Expert to Improve Learning". IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), vol. 22, no. 6, pp. 812-825, June 2010.

8. Jun Du and Charles X. Ling, "Active Learning with Generalized Queries". Proceedings of the 9th IEEE International Conference on Data Mining (ICDM), December 2009.

9. Wenyin Gong, Zhihua Cai, Charles X. Ling, and Jun Du, "Hybrid Differential Evolution based on Fuzzy C-means Clustering". Proceedings of the 11th Genetic and Evolutionary Computation Conference (GECCO), July 2009.

10. Jun Du and Charles X. Ling, "Co-Training on Handwritten Digit Recognition". Proceedings of the 22nd Canadian Conference on Artificial Intelligence (CAI), May 2009.

11. Charles X. Ling, Jun Du, and Z.-H. Zhou, "When Does Co-Training Work in Real Data?" Proceedings of the 13th Pacific/Asia Conference on Knowledge Discovery and Data Mining (PAKDD), April 2009.

12. Charles X. Ling and Jun Du, "Active Learning with Direct Query Construction". Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), August 2008.

13. Jun Du, Zhihua Cai, and Charles X. Ling. "Cost-sensitive Decision Trees with Pre-pruning". Proceedings of the 20th Canadian Conference on Artificial Intelligence (CAI), Montreal, Canada, May 2007.