Published in final edited form as:

IEEE/ACM Trans Comput Biol Bioinform. 2013 ; 10(3): 632–644. doi:10.1109/TCBB.2013.38.

Generalized Query-Based Active Learning to Identify Differentially Methylated Regions in DNA

Md. Muksitul Haque,
School of Electrical Engineering and Computer Science and the School of Biological Sciences, Washington State University, PO Box 4236, Pullman, WA 99164-4236.

Lawrence B. Holder,
School of Electrical Engineering and Computer Science, Washington State University, PO Box 642752, Pullman, WA 99164-2752.

Michael K. Skinner,
School of Biological Sciences, Washington State University, Pullman, WA 99164-4236.

Diane J. Cook
School of Electrical Engineering and Computer Science, Washington State University, PO Box 642752, Pullman, WA 99164-2752.

Abstract

Active learning is a supervised learning technique that reduces the number of examples required for building a successful classifier, because it can choose the data it learns from. This technique holds promise for many biological domains in which classified examples are expensive and time-consuming to obtain. Most traditional active learning methods ask very specific queries to the Oracle (e.g., a human expert) to label an unlabeled example. The example may consist of numerous features, many of which are irrelevant. Removing such features will create a shorter query with only relevant features, and it will be easier for the Oracle to answer. We propose a generalized query-based active learning (GQAL) approach that constructs generalized queries based on multiple instances. By constructing appropriately generalized queries, we can achieve higher accuracy compared to traditional active learning methods. We apply our active learning method to find differentially DNA methylated regions (DMRs). DMRs are DNA locations in the genome that are known to be involved in tissue differentiation, epigenetic regulation, and disease. We also apply our method on 13 other data sets and show that our method is better than another popular active learning technique.

Index Terms—

Active learning; generalized query; DNA methylation; bioinformatics

1 Introduction

In many scientific domains, there is an abundance of data. Such domains or topics can range from networks to weather to biology. Much of these data are unlabeled and unknown; therefore, labeling these data is the first step in working with them. This requires the help of a domain expert (e.g., human experts). To make effective use of an expert's knowledge and time, a new machine learning approach has arisen, called active learning, that is designed to maximize the potential of the Oracle (the human expert) in labeling data. Active learning (ACL) has been widely studied but popular ACL methods show shortcomings. For example, in traditional active learning methods, it may not always be easy for the Oracle to label a very specific case. The query may contain many features, some with high precision values. Labeling instances also can have varied cost and quality issues. A better approach is to remove some of the irrelevant features for a certain query such that it results in a shorter and more readable query. This will lead to less confusion for the Oracle. Using such generalized queries will help achieve higher accuracy with fewer queries than traditional active learning methods.

Human experts or Oracles are more readily able to answer a generalized query. As an example, for a car purchasing data set, we may construct a generalized query such as "if the car has 4 doors and the price is $20,000 and the engine size is 3.0 liters, then is it a family car?" For a car expert Oracle, such a generalized query is easier to answer. There may be 20 other features of the car that can be used to construct specific queries, but for this case a generalized query suffices. If the answer to the question is "yes," the machine learning system will note that all cars with those three characteristics are family cars. But the problem with a generalized query is that sometimes the answer from the Oracle can be uncertain. For example, for the above query, the Oracle may answer yes with 85 percent probability, but an overly generic query such as the one above may give a yes answer with probability of 65 percent. Since highly uncertain answers can add noise to the learning process, it is known that the more generalized the query the more uncertain the answer can be.

The goal is to model an active learning system that can construct generalized queries with highly certain answers. For our approach, we use a pool-based uncertainty sampling method [1] where we pick the most uncertain query from the pool (according to uncertainty sampling the most uncertain example in the pool is the most valuable one) such that it adds more knowledge to the current model. For example, if the probability of the majority class is 50 percent, then an instance with an Oracle confidence closer to this majority class probability will be uncertain.

2 Contribution

The main contribution of this work is the use of active learning with a generalized query algorithm to obtain higher accuracy with fewer queries to the Oracle. This algorithm can construct generalized queries using a pool-based uncertainty method and will construct these queries with do not care features (irrelevant features in the most uncertain examples). Experiments performed on real-world data sets from the University of California, Irvine

(UCI) database [2], some with multiple classes, indicate that the active generalized query approach poses fewer queries than other active learners, allowing the learner to improve learning efficiency.

One of the main goals of active learning is to build a classifier in situations where few labeled instances are available. With most biological data sets, this seems to be a major concern. Epigenetic data can be of low volume and high dimensionality. We may have only a few labeled sites with several thousand features. In these cases, active learning-based approaches can be used to find the most relevant examples and features to build a reliable classifier. Especially with experiments where data comes in stages (e.g., first generation, second generation of a species), we can build a classifier early on in the experiment and use it to test on newer data from later experiments. We also use our approach to identify differentially DNA methylated regions. DMRs are important molecular modifications classified as epigenetic regulations. Two of the main biochemical mechanisms for epigenetic regulations are DNA methylations [3], [4] and Histone modifications [5]. They regulate chromatic structures, control gene expression, and regulate genome activity. Epigenetic effects include gene silencing, gene imprinting, X chromosome inactivation, and elements of carcinogenesis. Epigenetic regulation can have a major effect on phenotypic expression that is independent of the underlying DNA sequence. The position of a gene in a chromosome can influence its expression, because genes can relocate to other heterochromatic regions in the genome and can cause human diseases. Genes are epigenetically modified from our parents and later on in our life can impact disease and longevity. Further studies have shown that epigenetics has a part in transgenerational inheritance that impacts epigenetic markers that later on can influence health and risk for diseases [6]. DNA methylation-based biomarkers are very promising, and a large number of potential biomarkers have been identified for diseases such as cancer [7]. To classify DMR against non-DMRs in the genome using active learning would allow us to identify potential biomarkers.

3 Current State of ACL with Generalized Query Systems

A number of active learning approaches have been proposed in the last few years. The most common and widely used form of active learning is uncertainty sampling [1]. Uncertainty sampling considers the most uncertain example as the most important one and asks for the corresponding label from the Oracle. One problem with uncertainty sampling is that it may choose outliers, which are highly uncertain data points. Therefore, it does not always follow the underlying distribution of data points. Another popular active learning method is query by committee (QBC) [8]. QBC considers minimizing the version space, which is the subset of hypotheses that are consistent with the examples seen so far [9]. A popular technique is to use a QBC from an ensemble of methods and try to find the unlabeled example that leads to the maximum disagreement among the classifiers [10], [11]. Other techniques have been used such as variance reduction [12], Fisher information ratio [13], estimated error reduction [14], and density-weighted methods [15], [16], [17], [18]. Additional methods include batch mode active learning [19], [20]. This technique queries in groups of a batch, instead of a single instance at a time. Batch mode active learning needs the Oracle to label all of the instances in the batch, and it does not produce a generalized query. Another approach that groups multiple instances together is multiple instance learning [21]. This also does not lead

to a generalized query as the Oracle has to label all the instances. This technique differs from the batch mode learning in that here the entire group is labeled as positive if there is a single positive instance in the group, whereas the entire group is labeled as negative if all the instances are negative in the group [22].

A novel technique called rule-induced active learning query method (RIQY) [23] has been proposed based on rule induction. This technique is based on examining the underlying density distribution to find informative instances that are similar. The method avoids outliers by using a density-weighted method. A rule induction classifier is then applied to separate similar instances from the rest of the data and construct a generic query. The proposed method has been applied to two real-world data sets: 1) the human activity recognition data set from the Washington State University (WSU) repository [24] and 2) the UCI repository [2]. While selecting the most informative instance, this approach also looks into how similar or dissimilar this instance is from the previously chosen most informative instance that allows it to choose an instance that is similar to many other instances so that it teaches a concept well but at the same time makes sure it does not teach similar concepts repeatedly leading to effective learning.

Another technique is called active learning through querying informative and representative examples (QUIRE) [25]. In this pool-based active learning, two criteria are widely used for active query selection. They are informativeness and representativeness. While informativeness checks the ability of an instance to reduce the uncertainty of a statistical model, representativeness checks whether the chosen instance will represent the input pattern of unlabeled data. Comparing QUIRE to other baseline approaches (RANDOM/MARGIN/CLUSTER/IDE/DVAL) shows that QUIRE is able to outperform other baseline methods significantly.

One important active learning method is called active learner with generalized queries (AGQ+) [26]. It is known to produce meaningful new features that are automatically generated unlike previous approaches [27] where new features are manually adjusted. AGQ+ also constructs generalized queries with numeric attribute ranges that are automatically produced from raw numeric attribute data.

As the last three models of active learning show, there is a general need of using better techniques for grouping instances together.

4 Generalized Query-Based Active Learning

This paper proposes active learning methods that allow the learner to ask generalized queries to the Oracle, which is able to answer such generalized queries with high accuracy. Our generalized query-based active learning (GQAL) method closely follows the AGQ+ method described in [26].

Our GQAL method differs from the AGQ+ method in the following ways. First, we use a tree augmented naive Bayes (TAN) classifier instead of a decision tree-based classifier. Second, the GQAL technique supports more than two classes unlike the AGQ+ method that only supports binary classes. Third, the AGQ+ method has used an ensemble of 100 J48

decision trees [28] to improve the accuracy. In our approach, we have not used any ensemble-based method. We show that our TAN-only model outperforms uncertainty sampling. Finally, the AGQ+ method uses the whole data set to train the target function. Often, the whole data set may not be available with class labels to construct such a model.

One machine learning technique is called *supervised learning*, as the learning operates with the supervision by being given the class of each example. In supervised learning, the class of each example in the training set is known beforehand. Among supervised learning methods, naive Bayes classifier (NBC) is a simple Bayesian classifier with the assumption of independence among features, yet is competitive with other classifiers having more restrictive assumptions. A tree augmented naive Bayes classifier uses the simplistic approach of naive Bayes but augments the NBC by constructing correlation between features [29]. The advantage of TAN over NBC is that in the TAN model no exponential search is used, yet it outperforms NBC. Similar to NBC, in a TAN model, the feature node is pointed to by the parent node, but feature nodes in TAN can have additional parents. Because of such properties, the TAN model can avoid having “double counting” that often happens in NBC where some features are used but are highly correlated or are very similar. One of the advantages of naive Bayes over other classifiers is that naive Bayes has high bias and low variance when the training set is small compared to low bias and high variance classifiers such as decision trees. The disadvantage of decision tree-based classifiers is that they easily overfit, whereas NBC will converge quickly with less training data. The main disadvantage of NBC is that it does not take into account interaction between features that the TAN model handles.

Our experiments in Section 6 show that accuracies from the GQAL method are comparable to another well-known active learning method: uncertainty sampling. In addition, for our study, we have used data sets where there are more than two classes. In the following, we will discuss this active learning technique. The model uses generalized queries by masking some features as do not care features. For this algorithm, we used pool-based uncertainty sampling methods. Also, the proposed method can be easily extended to QBC and other methods. We define our data set as consisting of a number of numeric or discrete features $X_1, X_2, \ldots, X_n$ and label Y. We have a set of unlabeled examples U and start with a set of labeled examples R. The learner can query from the unlabeled examples and ask the Oracle to label these examples.

The following steps describe the GQAL algorithm:

1. We are given a set of labeled examples in a training set R. A learner L is trained on R. A set of unlabeled data U is given as an unlabeled set and for testing a separate test set T is used.
2. Learner L is used on the unlabeled training set U to find the most uncertain instance available. (In QBC, the chosen instance would be the one on which the committee disagrees the most). In our model, an uncertain instance is the one on which the learner is least certain of its label. For example, if the probability of the majority class is 50 percent, then an instance with Oracle confidence closer to this majority class will be uncertain. For example, in a binary class if the

classifier predicts both the class probabilities are close to 0.50 (e.g., 0.55 probability of the instance to belong to class “A” and 0.45 probability for the instance to belong to class “B”), then the classifier seems uncertain of which class the instance belongs to. This would make it an uncertain instance.

3. The algorithm then takes this uncertain instance, finds the do not care features in this instance, and replaces them with “*.” It also finds weak features (nominal or numeric depending on the type of features) and replaces them with a subset of values (in case the features is nominal) or range (in case the feature is numeric). By replacing do not care feature with “*” and replacing weak features with their range (or nominal values), it constructs the generalized query. Details are given in Section 4.2.

4. Then, the algorithm can pose this generalized query to the Oracle, which gives a label and a probability estimation that is the Oracle’s confidence about the query label. Hence, each instance can have a weighted label. For a generalized query such as [*, 1, *, 1, 1], it may return probabilities such as 0.85 for class “A” or 0.15 for class “B.”

5. GQAL will use this generalized query and match with existing instances. For example [*, 1, *, 1, 1] will match with [0, 1, 0, 1, 1], [0, 1, 1, 1, 1], [1, 1, 0, 1, 1], and [1, 1,1, 1, 1], where an “*” can match with a 0 or 1. Such unlabeled instances are labeled and moved from the unlabeled data set U to the labeled training set R.

6. The algorithm then learns on the updated training set R and tests on the remaining unlabeled examples in U.

7. The algorithm then returns to Step 2 and repeats this until it reaches a threshold of number of times querying the Oracle or a predefined accuracy. The predefined accuracy can be set by what fraction of the initial error rate is reduced by the learning process. With the initial training set R, the classifier starts with an initial error rate and that error rate reduces over iterations.

8. Once the learning is complete, GQAL will use the learner L to test on the testing set T.

Fig. 1 shows the important components of the GQAL method.

The four important components of this algorithm include:

1. selecting the most uncertain instance,

2. using do not care features to construct the generalized query,

3. querying the Oracle, and

4. updating the training and the testing set.

They are described as follows.

4.1 Select the Most Uncertain Instance

Our TAN classifier-based GQAL method is used to find the probability of decision for each instance. For finding the most uncertain instance, a pool-based active learning method is used. As described in GQAL algorithm Steps 1 and 2 after a learner L is trained on a training set R, GQAL uses the current labeled data to construct a predictive model, and then an unlabeled data set U is given as a pool of candidates, and GQAL uses the model to predict each example in the pool-set.

It then picks the most uncertain example from the pool. The most uncertain example can be one whose probability of the classification is the closest to that of the majority class. For example, for a two-class scenario assuming the probability of the majority class is 50 percent, an example can have an predicted probability of 48 percent to belong to the majority class but still may not be the most uncertain example in the pool (e.g., another examples in the pool have higher majority class probability of 49 percent).

4.2 Using Do Not Care Features to Construct the Generalized Query

Once the most uncertain instance is found, it is time to find which of the features are irrelevant. One way to find this is to make sure that any combination of the features' values will make the same prediction with the same probability estimation. If a set of m features are irrelevant, then for that instance all 2^m value combinations (for binary) must be tested. The task of finding the probability estimation of all 2^m combinations is computationally expensive.

A technique for finding the largest item set in mining association rules [30], [31] is used, where we divide the set of features into two subsets. We have D, the set of do not care features, X_u the uncertain instance, and A the list of features. We start D as an empty set and for each attribute in A that is not in D, we generate 100 randomly assigned values in D and A. The attribute in A with the least change in probability distribution over 100 examples is regarded as irrelevant. We check if the change is less than the predefined threshold and then add the attribute to D. We continue this process until D cannot grow any further. The details of this algorithm are presented in Algorithm 1. One thing to note is that the generalized query with too many do not care features may well result in an overly generic query and end up to be very uncertain. Hence, the threshold is taken to be very small (0.005), which allows the algorithm to find a query generalized enough to find the most relevant features. This method has been tested in a previous study [26].

Algorithm 1: Algorithm for Constructing Generalized Query.

Input: The most uncertain instance x_u
Probability of the majority class (x_u) is p_u
The predefined threshold $\Theta(\Theta = 0.005)$
Output: The don't care attribute list D
procedure:FindDontCareAttribute(x_u, p_u, Θ)
$D = \{\}$;
done = false;
while not *done* **do**
for all X_i *in for all* X_i *in (A-D)***do**
$n = 0$;
$\text{change}_i = 0$;
while $n < 100$ **do**
Generate instance x_n *and assign* x_u *to* x_n
Assign random values for all features in D;
Assign random X_i;
p_n = *majority class probability for* x_n
$change_i += (p_n - p_u)^2$
increment n;
end while
$\text{change}_i / = 100$;
end for
Select the X_i *with the smallest* $change_i$
if $change_i < \Theta$ **then**
Add X_i *to D;*
else
done = true
end if
end while
return D;
end procedure

4.3 Query the Oracle

We make the assumption that the Oracle can answer generalized queries correctly. Larger training sets can help build better classifiers, because we have labels for many examples with different attribute values. To simulate a human Oracle, we train the TAN classifier with the entire data set (apart from a small portion of the data set as a test set) to represent the target model. As the target model cannot answer generalized queries, once a generalized query is given to the Oracle we generate specific queries by setting do not care attributes in the generalized query with random values. To avoid exponential complexity, we generate 100 specific queries from the generalized query. The Oracle (the target model) then returns the predicted probability distribution of these 100 examples.

4.4 Update the Training Set and the Unlabeled Set

The Oracle gives the probability distribution of the generalized query. The generalized query not only helps the Oracle to easily label the data set but also allows us to add similar instances (to the training set) that match this generalized query. This technique helps us build a better classifier at each iteration. For our method, we allow a maximum of 100 examples to be added from the unlabeled data set to the training set over all the generalized queries asked to the Oracle. (For our experimentation section part (a), we used as many

instances as needed for the classifier to reduce its classification error by $\frac{3}{4}$.) If such examples are realistic, then that improves the learning substantially compared to traditional methods.

5 Handling Different Feature Types

Algorithm 1 is capable of constructing a generalized query dealing with discrete features, but allowing only specific types of features can add limitations to the learner. A single feature can have a subset of nominal values or a numeric range of values. For example, the feature "weather" can have nominal values such as cloudy, sunny, windy, or rainy, while the price of a product can have a numeric range from $50 to $100. So to incorporate these scenarios, we extend our current algorithm. However, Algorithm 1 sets the base of the main parts of the GQAL method. The following sections provide a brief overview of the numeric and nominal methods.

5.1 Nominal Features

For finding the set of nominal features, we still find the set D of strongly irrelevant features and then try to find the set W of weak irrelevant features from the most uncertain instance X_u. First, we start with an empty set W, then we gradually start filling up W with weak-irrelevant features using the following technique. After we have selected the most uncertain instance X_u, we find each feature X_i that is not in D and W. For each feature value $X_i = a_{ij}$, we randomly generated 100 feature values for features in D and W that are based on X_u. If all the examples produce the same class probability for that feature, then we add that feature and its value a_{ij} to the W list. Similarly, we go to the next features and repeat this until our weak features list W no longer expands. Finally, we replace all D with * and all the W with their respective values from the most uncertain example.

5.2 Numeric Features

Similar to the nominal features list for each most uncertain instance X_u, we have a strong irrelevant feature list D and weak irrelevant feature list W. Here, for each feature X_i that is not in the strongly irrelevant list D or the weakly irrelevant list W, we expand the current feature value by β. Unlike a nominal feature list, we need to find a numeric range, and thus, we have more values. Each time we expand the feature's value $X_i(x_i + \beta, x_i - \beta)$ and randomly assign values of features in D and W, we check with the current class probability for the most uncertain example X_u. If the class probability is the same, we expand the range. We stop once there is a difference in the class probability, and we add this X_i to the W list with the expanded range of values. Finally, we replace values of D with * and values of W with their numeric range. Details of the nominal and numeric feature algorithms are given in Algorithms 2 and 3 [26].

Algorithm 2: Algorithm for finding weak nominal features.

Input: The most uncertain instance x_u
The don't care feature list (D)
Probability of the majority class (x_u) is p_u
Output: The weak nominal feature list W
procedure: FindWeakNominalFeatures (x_u, p_u, D)
$W = \{\}$;
for all X_i *in* (A − D − W) **do**
for all $X_i = a_{ij}$ **do**
$n = 0$;
$\text{change}_{ij} = 0$;
while $n < 100$ **do**
Generate x_n *and assign* x_u *to* x_n
Assign random values for all features in D;
Assign possible nominal values for all features in W;
p_n = *majority class probability for* x_n
$change_{ij} += |(p_n - p_u)|$
increment n;
end while
if $\text{change}_{ij} = 0$ **then**
Add X_i *and* a_{ij} *to* W;
end if
end for
end for
return W;
end procedure

Algorithm 3: Algorithm for finding weak numeric features.

Input: The most uncertain instance x_u
The don't care feature list (D)
Predefined threshold β
Probability of the majority class (x_u) is p_u
Output: The weak numeric feature list W
procedure:FindWeakNumericFeature (x_u, p_u, D)
$W = \{\}$;
for all X_i *in* $(A - D - W)$ **do**
$high = low = a_i$
$\text{change}_{ij} = 0$;
do
$high = high + \beta$
$low = low - \beta$
$n = 0$;
while $n < 100$ **do**
Generate x_n *and assign* x_u *to* x_n
Assign random values for all features in D;
Assign random values for all features in W;
Assign random X_i *within* [low,high];
p_n = *majority class probability for* x_n
$change_{ij} += |(p_n - p_u)|$
increment n;
end while
while $\text{change}_{ij} = 100$;
high = high - β
low = low + β
if (*low < high*) **then**
Add X_i *with numeric range to* W;
end for
return W;
end procedure

5.3 Implementation

The code was implemented using the C language and run on the Linux platform.

6 Experimental Setup

We ran three different types of experiments on 14 data sets. All tests were performed using 10-fold cross validation. First, we ran GQAL on all of the data sets and calculated their accuracy and average queries using the Oracle query restriction as shown in Table 3. Next, we compared the GQAL method with the uncertainly sampling active learning method based on accuracy versus number of queries on the first 12 queries as shown in Fig. 3. Finally, we compared the GQAL method without query restriction with popular nonactive machine learners NB, SVM, and KNN, and the results are given in Table 4.

The 14 data sets used for the experiments are taken from diverse domains. Among the data sets, two data sets are nominal only, one is numeric/nominal and the remaining 11 are

numeric only. The number of instances for the data sets ranges from 150 to 8,124, and the number of features range from 4 to 38. Four of the data sets have more than two classes. Details of the data sets are given in Tables 1 and 2.

First the GQAL system was tested on the voting data set (438 instances). Fifty randomly chosen labeled instances initially were used to train the classifier. Then, generalized queries using do not care features were used to add 50 additional instances from the unlabeled pool to the training set. GQAL was used to find the most uncertain instances from the unlabeled pool. Finally, the learner was tested on the test set of 335 instances. Fig. 3 shows the last three consecutive generalized queries in a sample run. The last line “finally” shows the classification accuracy of the learner when applied on the test set.

The top line shows the iteration number between braces “(“and”)” followed by the most uncertain example in that iteration and its probability. The second line (with no number at beginning) is the generalized query whose features are masked by the do not care feature set. Positions marked by star can have any feature value. Each generalized query can match from 1 to 50 instances (50 max). A maximum of 50 queries can be asked and corresponding instances (instances matching the query) are asked for labeling. The maximum instances used for this training is 100 (50 initially + 50 in this stage). “Most uncertain” shows the instance the classifier is least sure of (of the rest of the set) belonging to any class. After the first line, the rest of the lines are instances having the same masked features set. All of them are moved from the unlabeled set to the training set letting the classifier learn how to classify those specific examples.

The output shows consecutive results after querying the Oracle, adding some additional data to the training set, and training the classifier on the updated training set. Once learning is done, we apply the learned classifier on the test set, and we achieve 304 correct and 31 incorrect classification results; accuracy achieved is 90.746 (Fig. 2).

The voting data set has some missing values, do not care features (*) will match any missing (empty) values. In those places, empty spaces will show up (e.g., instances 99, 132, and 134 have missing values).

After testing Algorithm 1 on the voting data set, we implemented Algorithms 2 and 3 and tested our method on 13 more data sets. They were taken from the Orange machine learning suite [32] and the UCI repository [2] along with our own epigenetic data set. The descriptions of the data sets are given in Table 1. All of the data sets are in tab file format that is also the input to our program.

7 Results

Experiment A.

For this experiment, we used 10-fold cross validation. At each fold, we first trained the classifier with a sizeable fraction of the total instances. The initial training size column in Table 3 specifies the fraction of the total instances that were used to train the initial classifier for each data set. During each iteration, we add more instances from the unlabeled set to the

training set and test against the remaining unlabeled set until the classifier error is reduced to $\frac{3}{4}$ of the maximum error. Then, finally, after we have built the learner with these instances, we again test it on the set aside test set. The results of GQAL on all 14 data sets are given in Table 3. We picked different types of databases and many of them with more than two classes, unlike some of the previous studies (e.g., [26]). Finally, we tested our approach on an Epigenetic data set from the Skinner Lab at Washington State University [33]. Details of the epigenetic data set are given in Section 7.1. One thing to note is that the performance of GQAL depends on how well the generalized queries are formed and how many instances match those generalized queries. Table 1 shows all the data sets used, type of features, total number of instances for each data set, number of features, and the class distribution. Table 3 shows the average number of do not care features, the average number of added instances, average Oracle confidence for each iteration, and the initial training set size. The initial training size ratio (column 5) is dependent on the size (total instances) of the data set. Initially, the learner L is trained with an average of 35 instances for each data set.

Experiment B.

We compared our GQAL-based method with another popular active learning method: uncertainty sampling. We again used both ACL based techniques on all 14 data sets. We show the results based on the first 12 iterations in Figs. 3a, 3b, 3c, 3d, 3e, 3f, 3g, 3h, 3i, 3j, 3k, 3l, 3m, and 3n. The results are based on 10-fold cross validation. The results are shown from iteration 1, and the accuracy axis has been started from different values for different data sets to show the performance comparison clearly. We can see that our GQAL method outperforms the uncertainty sampling method in most cases (except glass data set; with ionosphere, voting, and epigenetic data set showing close accuracy) with fewer queries. The reason is because of the GQAL method utilizing generalized queries. At each iteration, uncertainty sampling matches with a single instance while GQAL can match with multiple instances.

Although we get higher accuracy compared to uncertainty sampling, our result and accuracy can vary among data sets since the number of do not care features created for each query and the number of matching instances can vary. Having more instances and more features does not necessarily mean the average added instances per query will be higher or the percentage of do not care features will be more. As we can see, the reduction in the number of features can vary from 21 to 87 percent, while the number of average instances added can vary from 3 to 52. If we increase the number of queries to the Oracle, the accuracies will increase (as we have kept our observation limited to 3/4 of the maximum error rate) but due to the generalized query method increasing the number of queries to the Oracle will still keep the number of features in the query small.

Experiment C.

In addition to comparing our GQAL method with another active learning technique, we also compare our method with nonactive learners. We calculated the average maximum accuracy by our GQAL method on all data sets without query restrictions using 10-fold cross validation. Our results are compatible with several base learners (naive Bayes, KNN, SVM)

using 10-fold cross validation on the data sets. The results are given in Table 3. The ANOVA test between them did not show any statistically significant result (p-value = 0.3217). The t-test assuming unequal variances showed the GQAL result not to be statistically significant compared to KNN (p-value = 0.76), SVM (p-value = 0.6853), or NB (p-value = 0.1504). The results show a number of times when GQAL performs better than the nonactive learning methods on particular data sets, but overall it does not outperform the other classifiers on all occasions. One thing to note is that active learning uses fewer instances to train, while the nonactive learners use all of the instances available to train their classifiers.

7.1 Epigenetic Data Set Analysis

The new scientific paradigm in the biological sciences is that there exists an epigenetic genome (epigenome) in parallel to the genome that regulates genome activities. For finding such epigenetic sites, we looked for differentially DNA methylated regions. To determine whether an instance is DMR or non-DMR, first we tried to use a regular classifier such as naive Bayes, SVM, and KNN. Since most of the DMR and non-DMR sites are predicted, and only a few sites can actually be tested due to cost issues, it is not always possible to train a classifier using a large training set. While using only a few confirmed sites as the training set, the accuracy of the classifiers was low. Since there are only a few confirmed labeled sites and many unknown/unlabeled sites, we next tried a machine learning approach like active learning. Active learning can build a classifier with few confirmed sites and pick the next important unlabeled site to be labeled by the Oracle to build a better classifier. We show that using few instances, we can build a classifier using our GQAL method that can outperform nonactive learning methods such as NB, SVM, and KNN.

The sequence data set (epigenetics) in the Table 1 was used to identify DMRs. The data set is based on sets of DMR with vinclozolin-induced transgenerational changes in DNA methylation in Sertoli and Granulosa cells [34]. Epigenetics refers to the chemical modifications that happen in the genome that are independent of the underlying DNA sequence, but functionally relevant in terms of gene expression. Examples of such changes are DNA methylation and histone deacetylation [35]; both can regulate gene expression without changing DNA sequence in the regulated gene. The current study of the data set is focused on an investigation of how an environmental compound (endocrine disruptor) can promote an epigenetic transgenerational disease state. DNA methylation is investigated because it is the primary epigenetic mechanism that has been shown to mediate generational inheritance through the male germ line [36], [37]. Predicting regions to be DMR and correctly labeling them to be DMR or non-DMR is of crucial importance in epigenetics.

Mining of epigenetic profiles starts with extraction of interesting properties from the DNA sequence data. After sites of differentially methylated changes have been found between control and treatment (using statistical method and R [38]), the sites are annotated using Nimblegen GFF annotation files to find the gene associated (and their orientation) with each of the DMR regions. FASTA files are created from upstream and downstream of the target genes up to 100 Kb. After construction of FASTA files for extraction of genomic features, RepeatMasker was used to find SINE, LINE, ERVL, ERV, and other repeat elements from the upstream and downstream of the DMR locations. One of the common ways of extracting

genomic features from sequences is through repeat elements. Repeat elements and consensus sites detect interesting patterns from interesting sites. Other genomic features are GC content (percent of G (guanine) and C (cytosine) in the sequence) and CpG sites. Then, CpGislandSearcher [39] was used to find CpG islands in these regions. CpG islands denote high frequency of CpG sites. A CpG site is denoted by a C followed immediately by a G. Epigenetic sites have been found in low-CpG-density regions, and therefore, a lack of such feature in interested sites will be helpful. Another important genomic feature used is DNA Motifs [40], [41]. Common patterns between biologically relevant sites can be identified using Motif findings tools. DNA Motifs representing binding sites of transcription factors and can be represented by a probability matrix for each base position such that a certain combination of those sequences matches with every subsequence.

In the data set we have used, there are 130 negative and 425 positive sites for the DMR regions. Each of these regions corresponds to a gene promoter location. The database has 38 genomic features (26 repeat elements, 10 motifs, GC content, and CpG islands) for each of those regions. With 38 features, we get 79.78 percent accuracy using only 37 queries to the Oracle in our ACL-based generalized query model. This accuracy is closer to the majority class (76.57 percent), but this is still better than other learners such as KNN, SVM, and NB (Table 4). The results also show better accuracy with the first 12 queries over uncertainty sampling (Fig. 3n). The goal is to train the classifier with the help of the Oracle to predict new regions that can be tested for DMR properties. We have a number of newly predicted sites that need laboratory confirmation to verify them as positive DMR regions.

8 Discussion

Overall, the results show that our model performs better than other approaches by using pool-based uncertainty sampling and generalized queries. There are concerns regarding several issues:

1. Since we assume that the Oracle always provides us with the correct answer, can we trust the Oracle to provide us with a reliable probability estimation on the generalized query?
2. How well does our method perform when we have a training set with a small number of labeled examples?
3. What is the number of queries that needs to be asked to the Oracle to achieve high accuracy?
4. Is a feature selection technique better than active learning with generalized query?

We address these concerns with the following recommendations.

Here, we have assumed that the answer from the Oracle is always reliable. There are a number of techniques [37] used to check if there is any noise or unreliable answers, but we have not used those techniques in our approach. This can be added as an extension of the current work.

We have already stated that having fewer initial labeled examples in the training set can lead to unreliable answers from the Oracle. Fewer examples lead to generalized queries based on too many do not care features. Such experiments can lead to uncertain answers from the Oracle as it is difficult for the Oracle to correctly find the do not care features. To overcome this problem, it is possible to take a proportion of the features to be do not care features, which depends on the number of examples present for the training. For example, if we have 10 training examples, a maximum of five features can be labeled as do not care features and used to construct a generalized query, for 20 training examples up to 10 do not care features can be used, and so on.

We ran tests on some of the data sets counting the number of queries that were asked to the Oracle. We found from the tests that the number of queries ranged from 14 to 50 for different accuracy rates. The number can vary depending on which most uncertain example the model chooses and how many unlabeled examples are added to the training set. It also depends on whether the test involves a restricted number of Oracle queries. In unrestricted form, we can keep adding instances to the training set from the unlabeled set until we reach the maximum accuracy or run out of instances in the unlabeled training set.

Active learning has sometimes been compared to feature selection techniques. In feature selection, important features can be removed globally from the entire data set, but for generalized queries, some features are essential for some queries while they are not essential for other queries. Hence, different generalized queries to the Oracle make use of the importance of different feature sets at a time, which cannot be done using a one-time global feature selection method.

In comparison to GQAL versus uncertainty sampling, the performance of our GQAL method is dependent on how well it can convert the most uncertain instances to generalized query. If it converts them to generalized queries that are too generalized (have too many do not care attribute) or not generalized enough (have too few do not care attribute), then the performance of GQAL will degrade. Again, if the generalized query matches with one instance only each time (for the entire training and testing), then there is no difference between GQAL and uncertainty sampling. So, it is apparent that if the target concept is hard to capture through generalized query for any data set large or small, GQAL performance will suffer significantly.

9 Conclusions

This work starts with the current scenario in active learning and describes different applied techniques. Having visited current query-based active learning techniques, we look into a do not care features-based ACL implementation. We show accuracy versus number of queries and show that even with few queries we achieve performance commensurate with nonactive learning approaches. We also show how generalized query can be performed on instances when we have numeric and nominal features present. One problem with generalized query is that the answer from the Oracle can be uncertain. Although including multiple instances into generalized query can reduce the effect of noise in our GQAL method, having more instances with noisy labeling will lead to performance degradation. This happens when the

initial labeled instances represent a small subset of the training set. We also elaborate on how our approach can be improved when the number of labeled examples initially is very low.

A number of additions can be proposed to the existing framework:

1. consider different base learners apart from the TAN model,
2. more data sets,
3. dealing with noisy data,
4. dealing with noisy answers from the Oracle,
5. comparing results with other existing methods.
6. Since generalized query adds similar instances, we can avoid asking similar generalized queries to avoid biasing the learner toward only one type of examples. We can also ensure the dissimilarity among generalized queries to make the learner learn from a variety of examples in few queries, and
7. the GQAL method can also be extended to stream-based online active learning.

Our approach is the first of its kind to use ACL on an epigenetic data set. The number of queries used by GQAL on the epigenetic data set is fewer than all available examples. If we do not restrict the number of queries that can be asked to the Oracle for training, then GQAL with no query restriction will perform better on the test set than GQAL with query restrictions. Similarly with uncertainty sampling, if it can make use of all the examples for training, it performs better than GQAL with query restriction on the test set. However, our goal is to show that with fewer queries GQAL performs better than uncertainty sampling as it makes use of generalized query. So, when the numbers of queries are low, GQAL performs better than uncertainty sampling. Overall, we can state that our framework will become very useful in several domains including biology where only a small portion of the data is labeled and the rest are unlabeled data, and where we can get accurate classification results using minimum Oracle intervention.

Biographies

Md. Muksitul Haque received the BS degree from the Department of Computer Science and Engineering, University of Rajshahi, Bangladesh, in 2000, the MSE degree in software engineering from the Department of Computer Science, University of Alaska Fairbanks, in 2008, and the MS degree in computer science from the School of Electrical Engineering and Computer Science, Washington State University, in 2010. He is currently a research associate/computer analyst in the Skinner Lab, School of Biological Science, Washington State University. His research interests include studying the effects of early developmental exposures to environmental compounds on epigenetic features and the potential for its transgenerational transmission and applying machine learning (active learning, imbalance class) approaches to identify potential differentially methylated regions in the epigenome.

Lawrence B. Holder received the BS degree with honors in computer engineering and the PhD degree in computer science from the University of Illinois at Urbana-Champaign, in 1986 and 1991, respectively. He is currently a professor at the School of Electrical Engineering and Computer Science, Washington State University. His research interests include artificial intelligence, machine learning, data mining, graph theory, algorithms, security and bioinformatics.

Michael K. Skinner received the BS degree in chemistry from Reed College in Portland Oregon, the PhD degree in biochemistry from Washington State University, and the postdoctoral fellowship from the C.H. Best Institute, University of Toronto. He is currently a professor at the School of Biological Sciences, Washington State University. His research interests include mammalian reproduction and environmental epigenetics.

Diane J. Cook received the BS degree from Wheaton College in 1985, and the MS and PhD degrees from the University of Illinois at Urbana-Champaign, in 1987 and 1990, respectively. She is a Huie-Rogers Chair Professor at the School of Electrical Engineering and Computer Science, Washington State University. Her research interests include artificial intelligence, machine learning, data mining, robotics, smart environments, and parallel algorithms for artificial intelligence. She is a director in the AI Laboratory and a head of the CASAS Smart Home Project.

Referfences

[1]. Lewis DD and Catlett J, “Heterogeneous Uncertainty Sampling for Supervised Learning,” Proc. Int’l Conf. Machine Learning (ICML ‘94), pp. 148–156, 1994.

[2]. Frank A and Asuncion A, "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml/index.html, 2010.

[3]. Bock C and Lengauer T, "Computational Epigenetics," Bioinformatics, vol. 24, pp. 1–10, Jan. 2008. [PubMed: 18024971]

[4]. Weber M and Schubeler D, "Genomic Patterns of DNA Methylation: Targets and Function of an Epigenetic Mark," Current Opinion Cell Biology, vol. 19, pp. 273–280, 6 2007.

[5]. Bird A, "DNA Methylation Patterns and Epigenetic Memory," Genes Development, vol. 16, pp. 6–21, Jan. 2002. [PubMed: 11782440]

[6]. Manikkam M, Guerrero-Bosagna C, Tracey R, Haque MM, and Skinner MK, "Transgenerational Actions of Environmental Compounds on Reproductive Disease and Epigenetic Biomarkers of Ancestral Exposures," PLoS ONE, vol. 7, article e31901, 2012.

[7]. Anglim PP, Alonzo TA, and Laird-Offringa IA, "DNA Methylation-Based Biomarkers for Early Detection of Non-Small Cell Lung Cancer: An Update," Molecular Cancer, vol. 7, article 81, 2008.

[8]. Seung HS, Opper M, and Sompolinsky H, "Query by Committee," Proc. Fifth Ann. Workshop Computational Learning Theory, pp. 287–294, 1992.

[9]. Mitchell T, Machine Learning. McGraw-Hill, 1997.

[10]. Freund Y, Seung HS, Shamir E, and Tishby N, "Selective Sampling Using the Query by Committee Algorithm," Machine Learning, vol. 28, pp. 133–168, 1997.

[11]. McCallum A and Nigam K, "Employing EM and Pool-Based Active Learning for Text Classification," Proc. Int'l Conf. Machine Learning (ICML '98), pp. 350–358, 1998.

[12]. Cohn DA, Ghahramani Z, and Jordan MI, "Active Learning with Statistical Models," J. Artificial Intelligence Research, vol. 4, pp. 129–145, 1996.

[13]. Zhang T and Oles FJ, "A Probability Analysis on the Value of Unlabeled Data for Classification Problems," Proc. Int'l Conf. Machine Learning (ICML '00), pp. 1191–1198, 2000.

[14]. Roy N and Mccallum A, "Toward Optimal Active Learning through Sampling Estimation of Error Reduction," Proc. Int'l Conf. Machine Learning (ICML '01), pp. 441–448, 2001.

[15]. Xu Z, Yu K, Tresp V, Xu X, and Wang J, "Representative Sampling for Text Classification Using Support Vector Machines," Proc. European Conf. IR Research (ECIR '03), pp. 393–407, 2003.

[16]. Nguyen HT and Smeulders A, "Active Learning Using Preclustering," Proc. Int'l Conf. Machine Learning (ICML '04), pp. 79–89, 2004.

[17]. Xu Z, Akella R, and Zhang Y, "Incorporating Diversity and Density in Active Learning for Relevance Feedback," Proc. European Conf. IR Research (ECIR '07), pp. 246–257, 2007.

[18]. Settles B and Craven M, "An Analysis of Active Learning Strategies for Sequence Labeling Tasks," Proc. Empirical Methods in Natural Language Processing, pp. 1070–1079, 2008.

[19]. Brinker K, "Incorporating Diversity in Active Learning with Support Vector Machines," Proc. Int'l Conf. Machine Learning (ICML '03), pp. 59–66, 2003.

[20]. Guo Y and Schuurmans D, "Discriminative Batch Mode Active Learning," Proc. Advances in Neural Information Processing Systems, vol. 20, pp. 593–600, 2008.

[21]. Dietterich TG, Lathrop RH, and Lozano-Perez T, "Solving the Multiple Instance Problem with Axis-Parallel Rectangles," Artificial Intelligence, vol. 89, pp. 31–71, 1997.

[22]. Settles B, "Active Learning Literature Survey," technical report, http://www.cs.cmu.edu/~bsettles/pub/settles.activelearning.pdf, 2010.

[23]. Rashidi P and Cook D, "Ask Me Better Questions: Active Learning Queries Based on Rule Induction," Proc. Int'l Conf. Knowledge Discovery and Data Mining, pp. 904–912, 2011.

[24]. Cook D, Holder L, Shirazi B, and Schmitter-Edgecombe M, "WSU CASAS Smart Home Project," http://ailab.eecs.wsu.edu/casas/datasets.html, 2010.

[25]. Huang S, Jin R, and Zhou Z, "Active Learning by Querying Informative and Representative Examples," Proc. Neural Information Processing Systems, 2010.

[26]. Du J and Ling CX, "Asking Generalized Queries to Domain Experts to Improve Learning," IEEE Trans. Knowledge and Data Eng, vol. 22, no. 6, pp. 812–825, 6 2010.

[27]. Smith JW, Everhart JE, Dickson WC, Knowler WC, and Johannes RS, “Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus,” Proc. Symp. Computer Applications and Medical Care, pp. 261–265, 1988.

[28]. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, and Witten IH, “The WEKA Data Mining Software: An Update,” SIGKDD Explorations, vol. 11, pp. 10–18, 2009.

[29]. Friedman N, Geiger D, and Goldszmidt M, “Bayesian Network Classifiers,” Machine Learning, vol. 29, pp. 131–163, 1997.

[30]. Agrawal R, Imielinski T, and Swami AN, “Mining Association Rules between Sets of Items in Large Databases,” Proc. ACM Sigmod Int’l Conf. Management of Data, pp. 207–216, 1993.

[31]. Liu B, Hsu W, and Ma Y, “Integrating Classification and Association Rule Mining,” Proc. Int’l Conf. Knowledge Discovery and Data Mining, pp. 120–128, 1998.

[32]. Curk T, Demsar J, Xu Q, Leban G, Petrovic U, Bratko I, Shaulsky G, and Zupan B, “Microarray Data Mining with Visual Programming,” Bioinformatics, vol. 21, pp. 396–398, Feb. 2005. [PubMed: 15308546]

[33]. Skinner MK, “Skinner Laboratory, Center for Reproductive Biology,” http://skinner.wsu.edu/, 2010.

[34]. Guerrero-Bosagna C, Savenkova M, Haque MM, Sadler-Riggleman I, and Skinner MK, “Environmentally Induced Epigenetic Transgenerational Inheritance of Altered Sertoli Cell Transcriptome and Epigenome: Molecular Etiology of Male Infertility,” PLoS ONE, vol. 8, no. 3, article e59922, 2013.

[35]. Itoh M, Goto S, Akutsu T, and Kanehisa M, “Fast and Accurate Database Homology Search Using Upper Bounds of Local Alignment Scores,” Bioinformatics, vol. 21, pp. 912–921, Apr. 2005. [PubMed: 15509606]

[36]. Skinner MK, Manikkam M, and Guerrero-Bosagna C, “Epigenetic Transgenerational Actions of Environmental Factors in Disease Etiology,” Trends Endocrinology Metabolism, vol. 21, pp. 214–222, Apr. 2010.

[37]. Anway MD, Cupp AS, Uzumcu M, and Skinner MK, “Epigenetic Transgenerational Actions of Endocrine Disruptors and Male Fertility,” Science, vol. 308, pp. 1466–1469, 6 2005. [PubMed: 15933200]

[38]. Guerrero-Bosagna C, Settles M, Lucker B, and Skinner M, “Epigenetic Transgenerational Actions of Vinclozolin on Promoter Regions of the Sperm Epigenome,” PLoS ONE, vol. 5, article e13100, 2010.

[39]. Takai D and Jones PA, “Comprehensive Analysis of CpG Islands in Human Chromosomes 21 and 22,” Proc. Nat’l Academy of Sciences USA, vol. 99, pp. 3740–3745, Mar. 2002.

[40]. Das MK and Dai HK, “A Survey of DNA Motif Finding Algorithms,” BMC Bioinformatics, vol. 8, no. Suppl. 7, article S21, 2007.

[41]. Stormo GD, “DNA Binding Sites: Representation and Discovery,” Bioinformatics, vol. 16, pp. 16–23, Jan. 2000. [PubMed: 10812473]

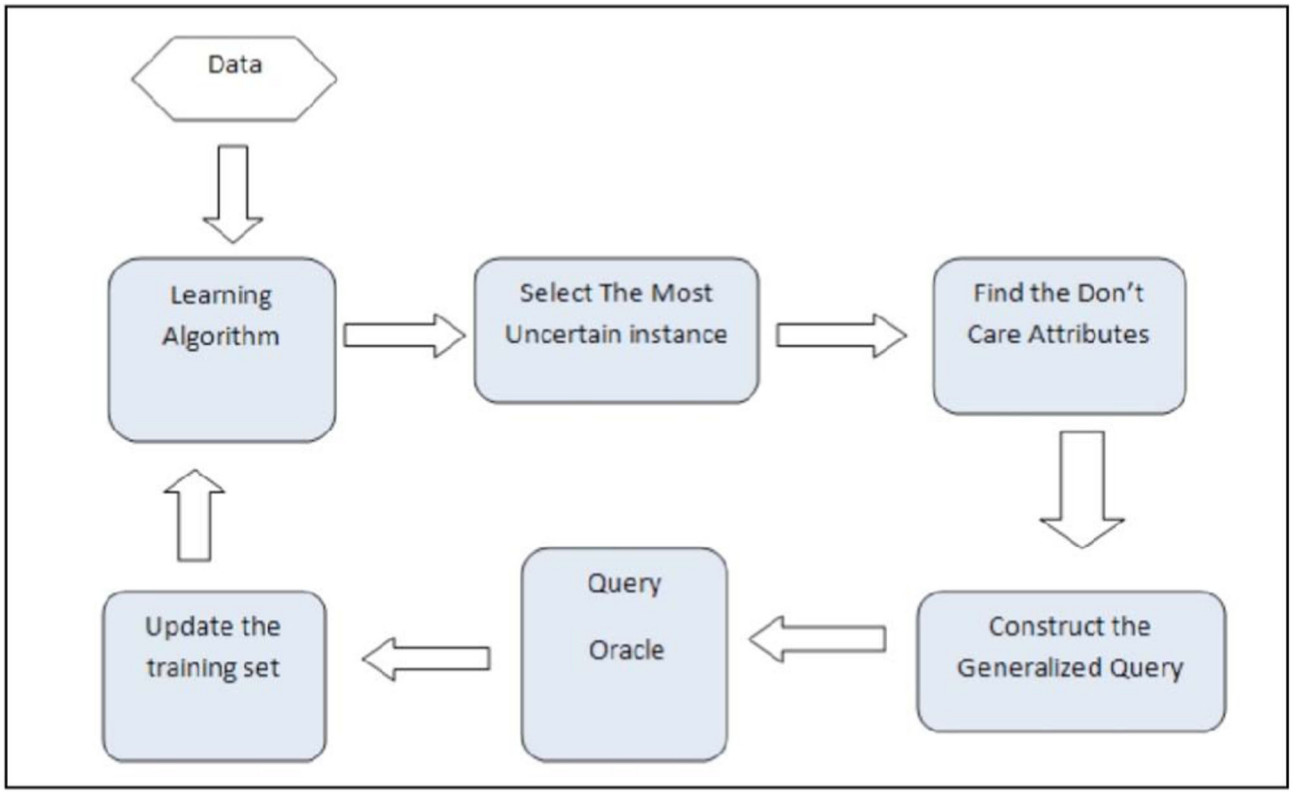

Fig. 1.
Important parts of the GQAL method.

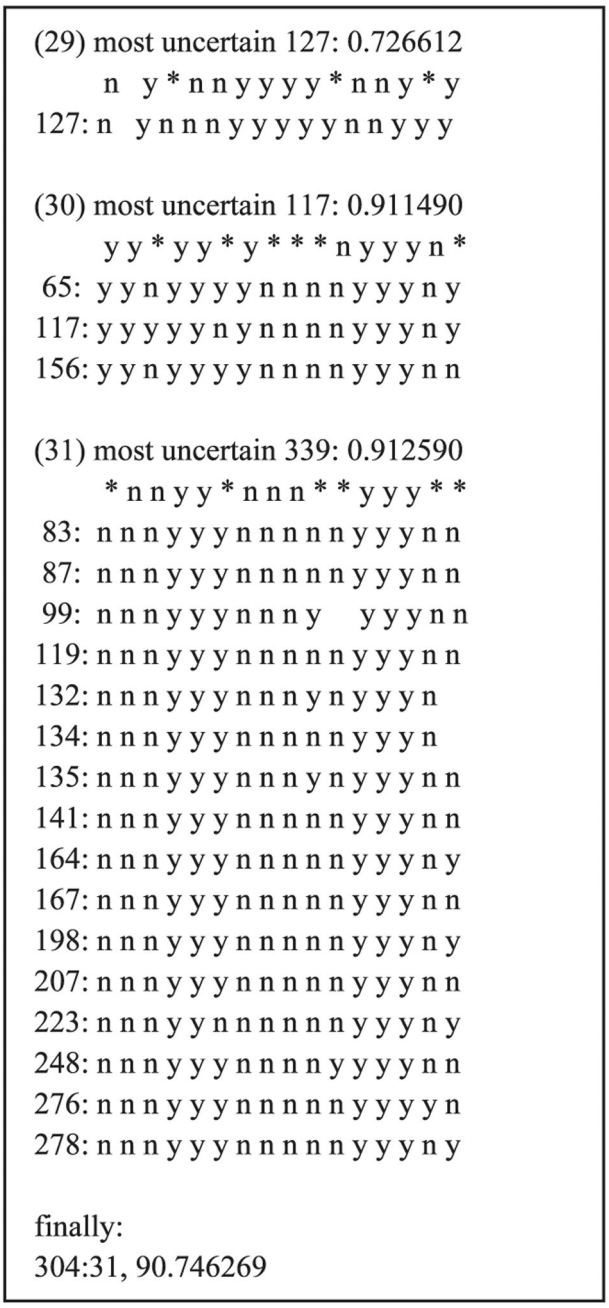

Fig. 2.
ACL method on the voting database.

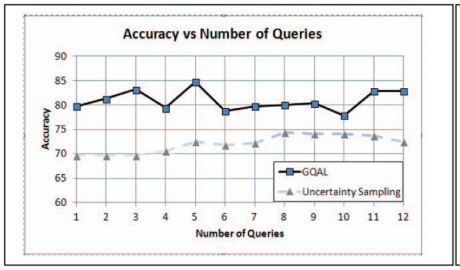

Fig 3(a) Chess Dataset

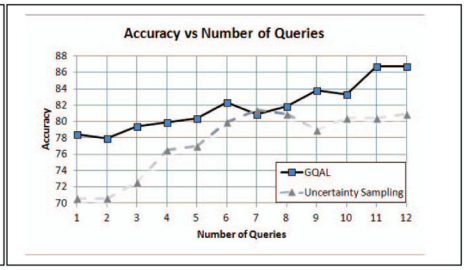

Fig 3(b) Breast-cancer dataset

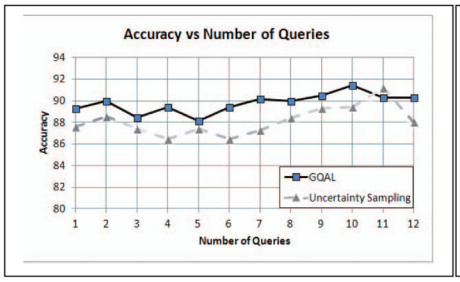

Figure 3(c) Ionosphere Dataset

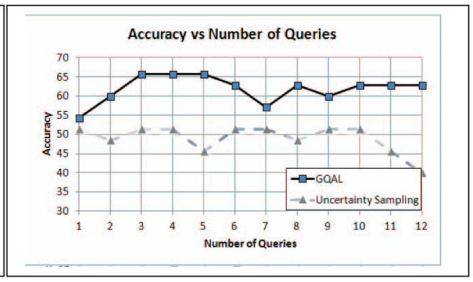

Figure 3(d) Bupa Dataset

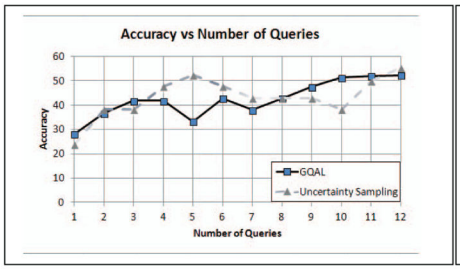

Figure 3(e) Glass Dataset

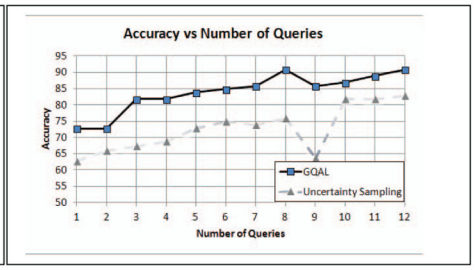

Figure 3(f) Hepatitis Dataset

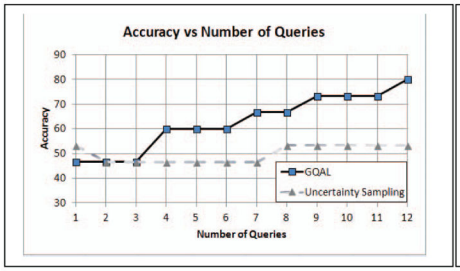

Figure 3(g) Iris Dataset

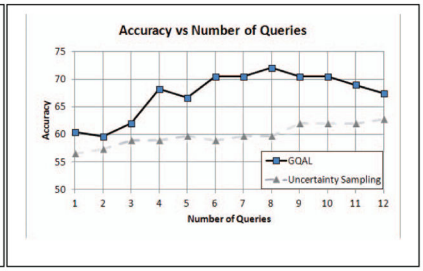

Figure 3(h) Monk Dataset

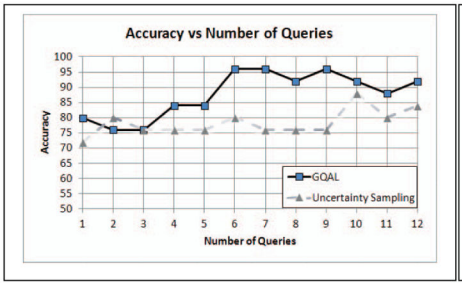

Figure 3(i) Shuttle-landing-control Dataset

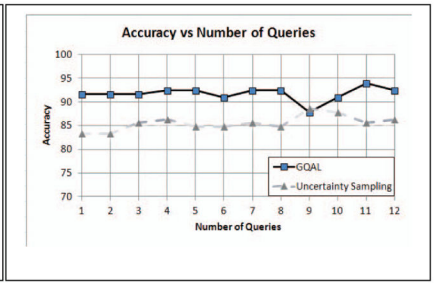

Figure 3(j) Voting Dataset

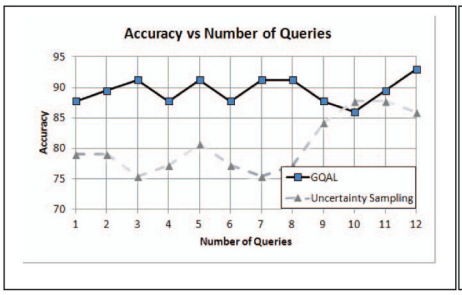

Figure 3(k) Wdbc Dataset

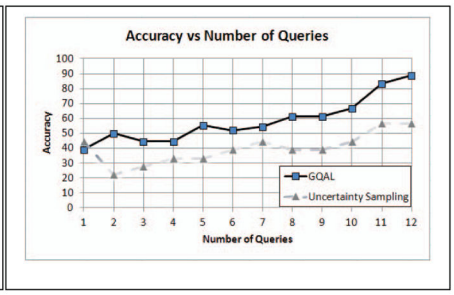

Figure 3(l) Wine Dataset

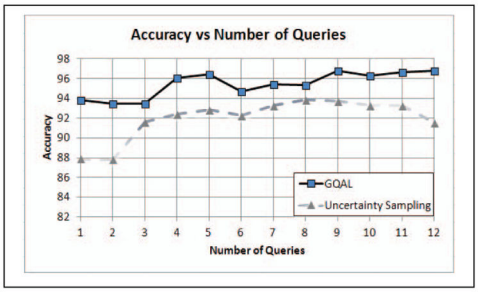

Figure 3(m) Mushroom Dataset

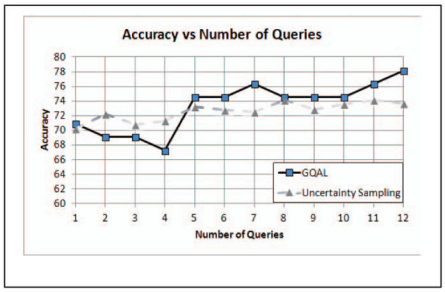

Figure 3(n) Epigenetic Dataset

Fig. 3.

(a) Chess data set. (b) Breast-cancer data set. (c) Ionosphere data set. (d) Bupa data set. (e) Glass data set. (f) Hepatitis data set. (g) Iris data set. (h) Monk data set. (i) Shuttle-landing-control data set. (j) Voting data set. (k) Wdbc data set. (l) Wine data set. (m) Mushroom data set. (n) Epigenetic data set.

TABLE 1

Description of Selected Data Sets

DataSet Name	Type of features	Total Instances	Number of Features	Class distribution
Chess	nominal	3198	36	1669:1527
breast-cancer-wisconsin	numeric	683	9	444:239
Ionosphere	numeric	351	34	126:225
Bupa	numeric	345	6	145:200
Glass	numeric	214	10	70:76:17:13:9:29
Hepatitis	numeric/nominal	155	19	32:123
Iris	numeric	150	4	50:50:50
Monk	numeric	432	16	290:142
shuttle-landing-control	numeric	253	6	108:145
Voting	nominal	435	16	267:168
Wdbc	numeric	569	30	212:357
Wine	numeric	178	13	59:71:48
Mushroom	nominal	8124	22	4208:3916
Epigenetic	numeric	555	38	130:425

TABLE 2

Description of UCI and Epigenetic Data Set

ataSet Name	Description
Chess	Predicting win on Chess game with King+Rook vs King+Pawn on a7.
breast-cancer-wisconsin	Classifying benign and malignant based on patient cancer information.
Ionosphere	Classifying of radar data taken from ionosphere.
Bupa	Predicting liver disorder from blood test result.
Glass	Predicting type of glasses, which are defined according to their oxide content.
Hepatitis	Classification of hepatitis patients based on their symptoms.
Iris	Predicting class of iris plant based on sepal, petal, length and width.
Monk	The three monk problem.
shuttle-landing-control	Evaluating whether to use automatic vs manual control for landing.
Voting	Classify votes as Republican or Democrat based on voter information.
Wdbc	Predict benign or malignant based on Wisconsin diagnostics breast cancer data.
Wine	Predicting the origin of wines using chemical analysis.
Mushroom	Classifying mushroom to whether they are edible or poisonous.
Epigenetic	Description provided in the text

TABLE 3

Data Sets Used and Their Results Based on Applying the GQAL Method

DataSet Name	Don't Care Features (% of total features)	Average Added Instances	Average Oracle Confidence	Initial Training Size
chess	15.1 (42%)	11.2	92.72%	1/100
breast-cancer-wisconsin	2.4 (26%)	14	94.44%	1/20
ionosphere	7.21 (21%)	14.07	94.93%	1/10
bupa	1.8 (30%)	2.2	91.02%	1/10
glass	4.6 (46%)	2.125	87.93%	1/5
hepatitis	16.6 (87%)	8.8	97.30%	1/5
iris	1.9 (45%)	4.7	94.54%	1/5
monk	3.4 (21%)	24.8	85.77%	1/20
shuttle-landing-control	3.3 (55%)	15.6	96.68%	1/5
voting	6 (37%)	5.3	97.51%	1/20
wdbc	12.9 (43%)	15.4	87.59%	1/20
wine	7.1 (55%)	7.8	97.51%	1/5
mushroom	7.4 (34%)	52.7	99.81%	1/200
epigenetic	17.3 (45%)	40.2	98.30%	1/20

TABLE 4

Comparison of Accuracy from Our GQAL Query-Based Approach with Other Learners

DataSetName	GQAL	KNN	SVM	NB
chess	84.89%	96.68%	93.93%	87.67%
breast-cancer-wisconsin	93.41%	96.04%	97.36%	96.77%
ionosphere	94.88%	87.46%	91.44%	87.75%
bupa	64.64%	60.29%	66.67%	65.51%
glass	85.08%	69.65%	50.97%	70.15%
hepatitis	83.63%	76.88%	77.38%	20.62%
iris	91.33%	96%	94.67%	92.00%
monk	64.81%	80.11%	66.90%	35.40%
shuttle-landing-control	98.02%	95.68%	93.28%	93.28%
voting	92.64%	93.09%	93.56%	90.34%
wdbc	92.27%	95.79%	95.79%	94.20%
wine	95.58%	84.30%	92.13%	67.52%
mushroom	100%	100%	99.99%	99.70%
epigenetic	79.78%	69.55%	78.56%	73.15%

IEEE/ACM Trans Comput Biol Bioinform. Author manuscript; available in PMC 2021 July 01.