A Robust Active Learning Framework using Itemset based Dynamic Rule Sampling

Bhanukiran Vinzamuri
vinzamuri@research.iiit.ac.in

Vikram Pudi
vikram@iiit.ac.in

International Institute of Information Technology
Gachibowli, Hyderabad
Andhra Pradesh, India

Abstract

Active learning is a rapidly growing field of machine learning which aims at reducing the labeling effort of the oracle (human expert) in acquiring informative training samples in domains where the cost of labeling is high. Associative classification is a well established prediction method which possesses the advantages of high accuracy and faster learning rates in classification. In this paper, we propose a novel algorithm which unifies associative classification with active learning. The algorithm has two major procedures of *Rule generation* and *rule pruning*. The algorithm selects unlabeled instances from the pool of available samples and uses a unique *dynamic rule sampling* procedure for updating the model. The rules are dynamically sampled class association rules (CAR) which are generated using the mined Minimal infrequent itemsets. The results derived over 10 datasets from the UCI-ML repository for our approach have been compared with those from the *ACTIVE-DECORATE* algorithm. We also analyze our sampling method against the state of art sampling frameworks and show that our method performs better.

1 Introduction

Active learning is a well known problem in the field of Machine learning [18, 19]. It has been applied in applications where either the amount of unlabeled data is voluminous or the cost of labeling is high. Few examples of such applications include video annotation, speech recognition and information extraction. The algorithm requests the human expert (oracle) for labels of unlabeled instances and presents them as queries. The goal of this process is to achieve high accuracy using as few labeled instances as possible. An important feature of every active learning algorithm is the mechanism by which the queries are chosen to be presented to the oracle. This *sampling* strategy directly effects the performance of the algorithm.

Active learners differ from the traditional passive learners in the manner through which the classification model is built. Passive learners do not generate queries to seek training examples to learn from like active learners. We have depicted the basic difference between them in Figure 1.

In the existing literature, there are many settings for active learning [18] but mostly the pool based cycle is preferred. In the pool based cycle, instances are selected from the unlabeled data in a greedy fashion based on their value of metric such as entropy, information gain etc. Many sampling frameworks have been applied in this setting such as uncertainty, random and selective sampling [18]. Amongst all these settings here we focus on the *uncertainty sampling* framework [9]. The pseudo code of this uncertainty sampling framework is as shown in Algorithm 1.

In line 3 of Algorithm 1 a function has been used to find out the most informative examples. Few of the functions which have been used in this scenario are the *Least confidence* LC utility which is based on an

Algorithm 1 Active Learning with Uncertainty Sampling

Require: initial training set L and test set T. Use L to train the classifier C.

1: **repeat**
2: Use C to label the unlabeled examples in T
3: Use *uncertainty sampling* to find out k most informative examples and ask oracle H to label them
4: Augment L with these k examples and remove them from T
5: Use L to retrain the current classifier C
6: **until** predefined stopping threshold is met

International Conference on Management of Data
COMAD 2010, Nagpur, India, December 8–10, 2010

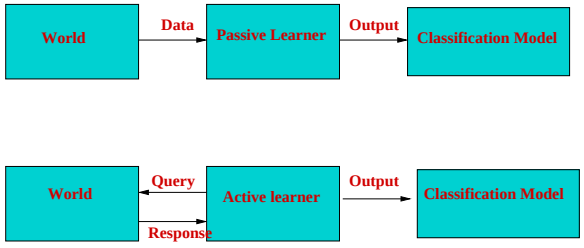

Figure 1: Active vs Passive learners

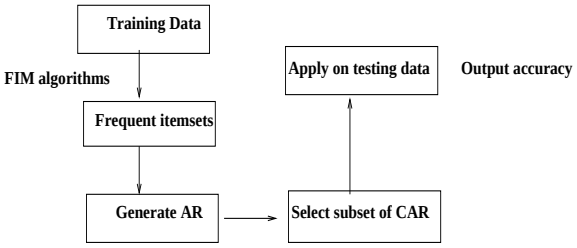

Figure 2: Associative classification

a-posteriori probability of the most likely label for an example. A simple approach however is to consider the margin between the first and second best label.

A small margin implies that the decision between the first and second best label is hard and hence such instances are said to be *ambiguous*. An important advantage of uncertainty sampling is it's low computational complexity compared to other statistical optimal approaches. Empirical studies also have found good performance for uncertainty sampling in practical use [19]. Based on this we define the Margin utility function ($MA Utility$) as follows.

Definition 1 MAUtility function*: Suppose that for an example* $p = (x)$ *the classification label space is* Y *then the MAutility function is defined as follows*

$$u_{MA}(p, \theta) = -(\max_{y' \in Y} P_\theta(y'|x) - \max_{y'' \in Y} P_\theta(y''|x))$$
$$where\ y'' \neq y'$$

1.1 Associative classification

Let $L = \{I_1, I_2 \ldots I_l\}$ be a set of items. An itemset I is a subset $I \subseteq L$. Suppose I is frequent and it consists of an item X then an association rule is defined as an implication of the form $X \rightarrow Y$ where $Y{=}I - X$. X is called the antecedent and Y is said to be the consequent. These rules are derived from the frequent itemsets which are mined from categorical datasets. Frequent itemset mining (FIM) algorithms [23, 24] such as Apriori, FP-Growth etc can be used to generate these rules. Associative classification is a method which uses association rules for classification [20]. However, in this method only those rules are selected whose *consequents* are class labels. Such rules are called Class Association Rules (CAR). The associative classification process has been displayed in Figure 2.

The *motivation* for us to use associative classification in pool based active learning is

1. Higher accuracy and scalability values in comparison to other contemporary classifiers.

2. Association rules can be easily interpreted by an oracle because of their intuitive structure.

3. **Incompleteness problem:** General associative classifiers use CAR mined from FIM algorithms for classification. The model built from such rules is generally pretty robust and accurate at classifying most of the records in the test set. However, it is noticed that these rules are not so successful in dealing with ambiguous records (explained in Introduction). So this effectively states that the knowledge acquired by the model is *incomplete* and we need additional knowledge for classifying ambiguous records correctly.

   In this perspective it is observed that even increasing the training data size is not a satisfactory solution. Hence, it boils down to effectively extracting knowledge from selected samples of data in the form of rules which can resolve the *ambiguities*. One must observe that the training data is always absolutely incomplete. However, one can definitely work on enhancing the performance of the classifier built on limited training samples with respect to the ambiguous records.

The paper is structured in the following manner. In Section **2** we discuss the basics of itemset theory and provide the theoretical background for our approach. In Section **3** we provide an overview of our approach and we also cover each of the preprocessing steps and settings associated with Step 1. In Section **4** we cover Steps 3-6 of our algorithm along with explaining their functionality and importance in the architecture we have provided. Section **5** consists of our experimental results which describe the data utilization and sampling experiments. Finally, we present our conclusions and assessments concerning the working and real time applicability of the framework.

2 Itemset theory and MIF theorems

In the context of association rule mining (ARM), we come across the idea of *minimal infrequent itemsets*. We begin by describing the definitions and settings in itemset theory.

Definition 2 Itemset*: Let* $D = \{t_1, t_2 \ldots t_r\}$ *represent a collection of* R *records. Each record consists of many items. Let* $L = \{I_1, I_2 \ldots I_l\}$ *be a set of items. An itemset* I *is a subset* $I \subseteq L$.

Definition 3 Support of itemset*: Given an itemset I a record t_i is said to contain I if $I \subseteq t_i$. The support set of an itemset I with respect to the dataset D is $D(I)$=$\{t_i \in D,\ I \subseteq t_i\}$. The support of an itemset I in dataset D is defined as the cardinality of the support set of I. So $Supp^D(I) = |D(I)|$.*

Definition 4 Frequent and Infrequent itemsets*: The relative support of an itemset which is the one used in FIM algorithms is $Supp^D(I)/|D|$. We then define a threshold which is called minimum support Γ. Based on the comparisons between the relative support and minimum support we can classify itemsets as the following. An itemset I is*

$$\Gamma\text{-}occurrent \text{ if } |D(I)| = \Gamma.$$
$$\Gamma\text{-}frequent \text{ if } |D(I)| \geq \Gamma.$$
$$\Gamma\text{-}infrequent \text{ if } |D(I)| < \Gamma.$$

This kind of categorization helps us identify different kinds of itemsets present in D. In addition to this now let us look at the notion of minimal infrequent itemsets.

Definition 5 Minimal infrequent itemset*:We say that an itemset I is minimal Γ-infrequent if I itself is Γ-infrequent and all of its proper subsets are Γ-frequent.*

These set of Minimal InFrequent itemsets (MIF) have also been referred to as the negative border in the literature. [11, 12]. Few algorithms have also been proposed to mine MIF [3] in the datasets and this is a *NP-Complete* problem. MIF have many utilities and have been used to identify rare patterns in domains such as fraud detection, informatics and risk assessment. In our current itemset based active learning scenario we try to exploit the information provided by MIF to identify rare and informative patterns which could help the active learner distinguish between classes effectively. We now provide few basic properties of MIF. These properties will help us understand the usability of MIF in our framework and we shall also use them to support our algorithmic procedures later

Lemma 2.1 *Rareness property: If an itemset I is an MIF then $Supp^D(I) < \Gamma$.*

If an itemset is minimal Γ-infrequent then it must be minimal Δ-occurrent for some $\Delta < \Gamma$. However it must be noted that minimal Δ-occurrent-c-itemset I is not a minimal Γ-infrequent for all $\Gamma > \Delta$. There may be some $(c-1)$ subset of I with support ϵ for $\Delta < \epsilon < \Gamma$.
This lemma proves that there must exist certain itemsets in D for an MIF to exist. The records holding these itemsets are called as *support rows*.

Theorem 2.2 *Given a minimal Γ-infrequent set $I = \{i_1, i_2 \ldots ic\}$, with $Supp^D(I) = \Delta$, $\Delta < \Gamma$, for each $1 \leq j \leq c$ there must exist $\Gamma - \Delta$ support rows in D containing itemset $I - \{i_j\}$ but not item i_j.*

Proof Suppose for some j there exists fewer than $\Gamma - \Delta$ support rows in D containing $I^{'} = I - \{i_j\}$ but not item i_j. Then $I^{'}$ which exists in $D(I)$ has $Supp^D(I^{'}) < \Gamma$ which implies $I^{'}$ is Γ-infrequent and hence I is not minimal Γ-infrequent.

There are many other properties of MIF [6] such as Recursive itemset property and others but these two properties and their related definition would be used in the next few sections for theoretical validation and justification.

3 Overview of our approach

3.1 Architecture

In this section, we present an overview of our algorithm and the major steps involved in it. The diagram given below in Figure 3 presents the architecture of our algorithm. The over all data available is initially partitioned into two groups of train and unlabeled pool data. The process flow of the algorithm is indicated in the diagram. We shall briefly explain each step's functionality in our algorithm here

1. **Step 1**: This deals with building the training model of CAR (Model) from the training data available.

2. **Step 2**: This deals with applying the current model on the unlabeled pool of samples and select instances to label based on our query framework. The output obtained here are a set of newly generated CAR which are generated based on the instances selected.

3. **Step 3**: This deals with effectively adding the CAR generated from Step 2 to the current model (Model) and updating it. This can also be called the actual active learning phase of our algorithm.

4. **Step 4**: This deals with applying the Model on the testing data. The testing data itself is extracted from the unlabeled pool. The outputs at this stage are the evaluation metrics of the learner on the testing data and the validation data which will be used later at periodic intervals. This validation data is generated over certain fixed number of active learning samplings over the unlabeled pool which is indicated by the multiple arrows.

5. **Step 5**: Once the validation data is generated we assess the rules added in the Model through the active learning process. This assessment is done based on the validation data to prune some redundant rules from the model. The output at this stage is the Updated Model.

6. **Step 6**: At the beginning of the next cycle the updated model is now used for sampling over

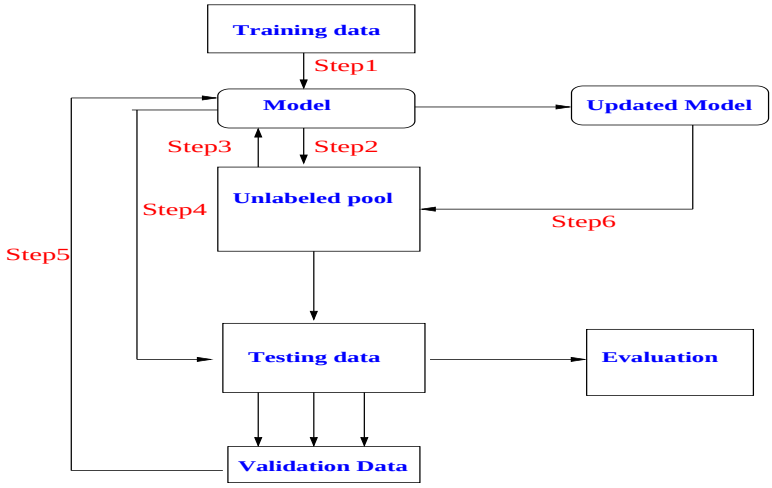

Figure 3: Architecture of the AAL algorithm

the unlabeled pool and the cycle repeats from Steps 3-5 until required number of active learning cycles are exhausted over the pool of data.

3.2 Pre-Processing

In this subsection, we describe the preprocessing stages associated with Step 1 of our architectural model. The two basic stages of pre-processing here are

1. Generation of Association Rules (AR) from training data and choosing a subset of Class Association Rules (CAR).
2. Building a preliminary associative classification model.

3.3 Generation of CAR

First we discretize the numerical data into categorical data using procedures which are listed in our experimental setup (Section 5.1). Once the data is prepared then we apply the Apriori algorithm [1] on the training data to derive the association rules. Later we separate the rules and retain those which satisfy minimum conviction [7] threshold value. We apply our pruning strategy here to eliminate the redundancies in the rules.

Superset pruning: It is known that rules with long antecedent lengths generally contribute less to associative classification as many such specific rules tend to make the model bulky. So such rules form a good search space for pruning. Our pruning criterion works as follows:
A rule is pruned if the conviction of the rule is lesser than any of its subsets and both of them share the same consequent.

Associative classification Model Once we have extracted the association rules from the dataset we isolate the set of CAR. From these set of CAR we identify a subset of CAR to use for classification. The criterion for selection of the subset of rules is based on the *coverage*.

The coverage for a rule is defined as the number of tuples covered by the rule. A rule covers a data tuple if the tuple satisfies the conditions on the left hand side/antecedent of the rule. The rules finally selected form our classification model.

4 Associative Active Learning

In this section, we look at the major procedures associated with our algorithm. This is followed by discussing the Steps 3-6 of our architectural design in detail.

4.1 Sampling function

As with any pool based active learning scenario we need to define our sampling function. The importance of the sampling function chosen impacts the active learning process tremendously. We looked at the *uncertainty sampling* strategy in the introduction which uses a function to identify the most informative examples.

Here we propose a sampling function for our current itemset based active learning framework. Suppose $D = \{t_1, t_2 \ldots t_r\}$ is a collection of R records and the label space consists of the classes $C = \{C_1, C_2 \ldots C_n\}$. Suppose that after building an associative classification model on D we obtain the model $R = \{r_1, r_2 \ldots r_p\}$.

While applying R on t_q where $t_q \in D$, let $R^{'} = \{r_1, r_2 \ldots r_k\}$ be the set of CAR which cover record t_q. This implies that all rules in $R^{'}$ can be applied on t_q to predict the consequent.

Then based on the *coverage* of the rules in $R^{'}$ on D we can assign scores to the contending classes in the predicted label space. Let $S = \{S_1, S_2 \ldots S_n\}$ be the set of scores for the predicted label space. Based on S we can define individual probabilities for the record t_q

to be classified in the n classes.

$$P_\theta(C_i|t_q) = S_i/(\sum_{i=1}^{n} S_i). \quad (1)$$

Based on eqn 1 we can derive the set of probabilities for all n classes. Once this is done we apply the Margin utility (MAUtility) function (1) to decide the ambiguity of the record.

Whenever a record's *MA Utility* does not satisfy the minimum threshold, it is called an *ambiguous* record. Such kinds of records are sampled and are queried for their labels. Step 2 of our model uses the MAUtility function to sample instances from the unlabeled pool. In every iteration a fixed number of instances are sampled and are passed on to Step 3.

4.2 Rule Generation

The instances sampled from Step 2 are now used in Step 3 which is the Rule generation phase of our algorithm. Once a record has been sampled the oracle provides the label for this record (the label from the original data provided is used). In this case we assume the original labelled data to act as an *oracle* H to label the sampled/queried instances (line 1 of Procedure 1) After this we generate rules from the instance instead of adding the instance to the set of existing training samples as in regular active learning algorithms.

These rules are generated using the Minimal Infrequent Itemsets (MIF). The MIF itself are generated during the ARM process. As a pre-processing step we remove all itemsets in the MIF which consist of the class labels. The pseudo code for the Rule generation phase is provided below.

Procedure1:*Rule_generation*

Require: MIFItemSet NB, Ambiguous instance I , TrainRuleSet $Train$, Oracle H

1: Acquire the label for I from H as l
2: RB=MineMIF(NB,I)
3: **if** len(RB) > 0 **then**
4: **repeat**
5: choose itemset in RB
6: Append the itemset with l and generate CAR $\{itemset \rightarrow l\}$
7: Add this CAR to $Train$
8: **until** All itemsets have been used
9: **else**
10: Append I to l to generate CAR $\{I \rightarrow l\ \}$
11: Add this CAR to $Train$
12: **end if**
13: **return** $Train$

In this *Rule_generation* phase we have used a method called *MineMIF*. This procedure simply mines the MIF NB to find all the itemsets which are subsets of the feature set of I. The condensed set consisting of all such full or partial matches is called the Retrieved Border RB (line 2 of Procedure1).

As we had earlier seen in Theorem 2.2 that for an MIF to exist the dataset must consist of certain itemsets. So considering Theorem 2.2 one can safely say that the *MineMIF* routine would retrieve such itemsets from I where $I \in UnlabeledPool$. However, it is not necessary that every ambiguous instance I we sample in this process has to be a support row. So this would lead to two cases of generating rules separately from support and non support rows. The condition in line 3 determines the kind of rules we generate here.

1. I is a Support row : This implies that RB would consist of itemsets obtained through NB from the *MineMIF* routine. In this case, we simply choose the itemsets which are stored in RB. While selecting the itemsets to generate the rules we only include those itemsets which satisfy *minimum antecedent length* threshold. The CAR are generated as explained in the pseudo code in Procedure 1.

2. I is not a Support row: This implies that RB would be an empty set. In this case as we find no full or partial matches in RB, we simply create one rule here. For this rule the antecedent is I and the consequent is l.

Lines 4-10 of Procedure 1 deal with the generation of dynamic rules. In lines 7 and 11 we update the current training model with the dynamic rule sampled in either case. We update the model with the rules to have an immediate impact. We also believe that this should definitely help in reduction of the number of labeling queries being posed.

However, since we are dealing with real life datasets it is quite possible that not all the dynamic rules being generated contribute dominantly towards the classifier performance. So, rather than simply adding rules we use another periodic pruning procedure to remove few of the generated dynamic rules.

4.3 Rule Pruning

In this subsection we look at Steps 4 and 5 in detail. As explained earlier our algorithm applies a periodic rule pruning strategy which aims at assessing the rules generated in Step 3 and controlling the rule size of the model. Simply speaking it tries to identify those dynamic rules among the ones added to the model which are being used *frequently* for classifying instances in the test data.

The data from the unlabeled pool which has been used by the active learner is separated and we create the testing data from the remaining instances. In Step 4 the model built so far until Step 3 is applied on the testing data. We then acquire the predicted labels of

instances in the testing data. Evaluation metrics such as accuracy etc are also obtained in this step.

Based on our interval size we then separate the set of records classified so far along with the labels predicted by the model for them. This data is called the validation data which is also simultaneously generated in Step 4. The applied supports of all the rules which were added in Step 3 of our algorithm are then calculated on the validation data generated during the interval. The formula for the applied support is given in eqn 2.

Let us suppose that the number of times the rule was used after being added to the model in this interval=n_s.
Let the total number of instances classified in this interval by the model=n_{total}.

$$Appliedsupport = n_s/n_{total} \quad (2)$$

Once we have calculated the applied support values in the interval we only retain the top k rules with the highest values of applied supports. The rule parameter k is chosen individually for each of the datasets we consider. The detailed information on setting this parameter is provided at 5.3. This process of updating the model based on the validation data is done in Step 5. So, this local assessment we do here in Steps 4 and 5 helps us bound the number of rules added to the model.

Procedure2: *Rule_Pruning*
Require: TrainRuleSet $Train$, rule parameter k, ValidationData $Validation$
1: **repeat**
2: Choose dynamic rule r in $Train$
3: Calculate the applied support for r in $Validation$ using eqn2
4: **until** All dynamic rules in $Train$ are evaluated
5: Retain in $Train$ only the top k rules
6: **return** $Train$

However, while pruning this method does not compromise on pruning the effective rules discovered in the process from Step 3. In Step 6 once we have the updated model after the end of the iteration we then apply it again on the unlabeled pool of instances. Steps 3-5 are then carried on until the required number of instances have been sampled.

4.4 Associative Active Learning (AAL) algorithm

In this section, we look at our Associative active learning (AAL) algorithm which unifies all the steps mentioned above. In line 2 of Algorithm 2 we build the associative classifier on the initial training set. Using the *MAUtility* function we identify the records from the pool (Step 2) to be passed on to Step 3 which is the rule generation phase. If the record satisfies the criterion then the *Rule_generation* procedure is called upon in line 7 to sample dynamic rules as explained earlier in Step 3. The labels are provided by the oracle H. These rules are then updated in the model.

As in all active learning frameworks we consider this instance as being utilized by the learner and we delete it from $Pool$. After the iteration is over the testing data is generated as $Test$ and the accuracy of the classifier is calculated based on the current model (Step 4) built so far (line 11). Here $Test$ consists all the instances which remain in the unlabeled pool after sampling is done (line 8).

Then in line 12 once the validation data $Validation$ is simultaneously generated in Step 4, we prune the rules through the *Rule_pruning* procedure as explained earlier in Step 4. The updated model at the end of the iteration is $UpdTrain$. Finally in line 13 we set the model $Train$ as the updated model $UpdTrain$ (Step 5) for further sampling iterations.

Algorithm 2 AAL Algorithm
Require: Oracle H, TrainRuleSet $Train$, Unlabeled Pool $Pool$, MIFItemSet NB, Threshold $\mu 1$, rule parameter k, Number of iterations max
iter=1
1: **while** iter $\leq max$ **do**
2: Use the associative classification model $Train$
3: **for all** record r in $Pool$ **do**
4: Calculate $MAUtility$ for r
5: **if** $MAUtility < \mu_1$ **then**
6: Acquire the label for r as l from H
7: UpdRules=$Rule\_generation(Train,r,NB,l)$
8: Remove r from $Pool$
9: **end if**
10: **end for**
11: $AssociativeClassifiy(UpdRules, Test)$
12: $UpdTrain=Rule\_pruning(UpdRules,k,Validation)$
13: $Train=UpdTrain$
14: **end while**

4.5 Illustrative example

Here we present the working of our algorithm on a small example for a 2 class classification problem (0,1). The total number of examples considered here are 12. Let the initial number of examples used for building the training data be 4 examples (Table 1). The model is now built using the training samples. We will use this model and apply it on the unlabeled pool (Table 2).

In each sampling iteration we choose 1 ambiguous sample and generate rules and update the model. This process is continued for 2 samplings. The way in which the new rules are generated and the validation and

Table 1: Training data

id	Features				Class
1	A	B	A	B	1
2	A	D	B	C	0
3	A	E	B	B	0
4	B	C	E	B	1

Table 2: Unlabeled pool

id	Features			
5	A	B	C	E*
6	A	C	B	E+
7	A	A	C	E+
8	A	C	A	E+
9	A	B	D	E*
10	C	B	E	D+
11	D	A	B	C+
12	C	C	A	B+

Table 3: Validation data

id	Features				PClass
6	A	C	B	E	0
7	A	A	C	E	1
8	A	C	A	E	1
10	C	B	E	D	0
11	D	A	B	C	0
12	C	C	A	B	0

testing data are used is explained in the working below. Also suppose that the minsupport value used here for ARM is 0.01.

Suppose that the records marked with * are ambiguous (id 5 and 9) according to the definition we have provided earlier. Then after applying the Apriori algorithm on the training data (Table 1) as described above the entries in $NB = (E, D), (A, C, E)$. Then based on the method explained above the Retrieved Border RB would be computed for the ambiguous records.

Suppose the labels assigned for these ambiguous instances are 1 and 0 respectively. Then the following set of rules would be added to the associative classification model. For id 5 the Retrieved Border would consist of (A, C, E) and hence we would add the rule $A \wedge C \wedge E \rightarrow 1$ to the model. For id 9 the Retrieved Border would consist of (D, E) and we would add the rule $E \wedge D \rightarrow 0$.

Once these rules are added to the model and at the end of 2 samplings the validation data (Table 3) is generated. The validation instances are marked in the unlabeled pool with a + sign. The labels (PClass) for instances in the validation data are assigned by the model as explained earlier during the rule pruning/Step 4. The process of applied support calculation over the validation data is displayed. Then we calculate the applied support values of the added rules over the validation data and retain the top k ones that satisfy the minimum support criterion alone. The applied support for $A \wedge C \wedge E \rightarrow 1$ over the validation data would be $2/6 = 0.333$ and that of $E \wedge D \rightarrow 0$ would be $1/6 = 0.166$. Assume that k here is set to 2. As the applied support values of both rules is greater than 0.01 (Apriori min support) both these rules are retained in the model.

The example we have provided here is very generic in nature and the same working is used for bigger real life datasets. More information about this will be provided in detail in the experimental section.

5 Experiments

In this section, we present the results derived for our AAL algorithm. We ran three major kinds of experiments which are

1. To measure the data utilization of AAL.
2. To analyze the error rate patterns at different learning stages of AAL.
3. To look at the labeling plots obtained for different datasets.

5.1 Experimental Setup

To evaluate the performance of the proposed AAL algorithm we ran experiments on 10 representative datasets from the UCI repository *. To facilitate the initial pre processing tasks such as normalization, replacing missing values and most importantly discretization we have used the Weka toolkit †. In this manner the nominal data is prepared to build an associative classification model.

We compare the performance of our algorithm against ACTIVE-DECORATE [14] from the data utilization perspective. ACTIVE-DECORATE is an algorithm which initially generates DECORATE [13] committees and later samples records from the testing data to update the committee members. DECORATE committees are generated by generating synthetic samples in the process generating many models. Further information about this can be derived from the related work section and references. All results reported below have been generated after a 5 fold cross validation. The set of available examples from real life datasets were separated as a pool of unlabeled instances and few training instances initially. The few training instances initially are needed to initiate the process of active learning. This has also been depicted in our architectural design in Figure 2.

In each iteration the AAL algorithm selects certain number of samples to be labeled and added to the training set. As the characteristics of each of the datasets are quite different from each another so we have used different sample sizes for each of them. The exact sample sizes will be mentioned explicitly when we explain the learning curves generated over these datasets. The ambiguity threshold μ_1 in AAL has been universally set to 0.3 for all the datasets. The configuration of the system used for running the experiments is a Pentium 4 Dual Core processor with 2 GB DDR2 RAM. The code was written in Python language.

*UC Irvine repository:http://archive.ics.uci.edu/ml/

†Weka: http://www.cs.waikato.ac.nz/ ml/weka/

5.2 Data utilization

As mentioned earlier active learning attempts to minimize the number of training samples and their corresponding labels required by the learner. In this experiment we will assess our AAL based on it's data utilization ratio. Before we define *data utilization* it is necessary to understand the definition of Target error. Target error here is defined as the error achieved by DECORATE determined by averaging the error values over the points corresponding to the last 50 training examples on the learning curve.

The data utilization factor is then defined as as the number of training examples an active learner requires to reach the target error rate divided by the number of samples required by the DECORATE algorithm. The formula is provided below.

Suppose for a given dataset D we use DECORATE and compute the Target error Err and the number of samples it needs to acquire Err on D as n_d. Suppose for D again we use any arbitrary active learner L and compute the target error rate $Err^{'}$ and the number of samples. Now the data utilization ratio is computed by finding the number of samples L needs so that $Err^{'}=Err$. Let us suppose that the number of samples needed by L in this case are n_l. Then the data utilization ratio for active learner L is defined as the ratio of.

$$\text{Data Utilization Ratio}=n_l/n_d$$

So this factor helps us understand how efficiently learner L uses instances from the unlabeled pool in comparison to DECORATE.

So essentially based on the definition of data utilization ratio we can compare ACTIVE-DECORATE, AAL against the DECORATE algorithm and also compare both the active learners individually. Table 4 presents the results derived on 10 UCI -ML datasets.

In Table 4, the columns indicate the number of data samples used by the each of the active learners with AAL being the rightmost. Each entry in the row also represents the data utilization percentage which is computed for ACTIVE-DECORATE and AAL using the number of samples used by DECORATE alone as a standard. For example if we consider the diabetes dataset the data utilization ratio of ACTIVE-DECORATE would be $201/234 = 0.86$ and that of AAL would be $83/234 = 0.35$.The learner with the least data utilization percentage (marked in bold) is considered ideal in each case and we compute the number of wins over 10 datasets for all the three learners.

It can be observed that out of 10 datasets AAL out performs ACTIVE-DECORATE in 7 of them. In particular for datasets such as *diabetes*, *sonar* and *lymph* the data utilization ratio is significantly lesser than the others. It is also noticed that over these ten datasets the average data utilization ratio of AAL is **77.2**% of that used by ACTIVE-DECORATE.

5.3 Learning curves/Error rate patterns

Apart from the data utilization perspective we can evaluate and visualize the learning process of an active learner and understand their error rates. The learning curves can be generated by recording the error values reported by our model at the end of each sampling iteration. Ideally it is expected that the error rate must decrease as more number of instances are utilized. We evaluate the occurrence of this phenomenon in the current experiment. In this experiment we have selected four datasets which are *diabetes*, *german*, *image segmentation* and *soybean* for plotting the learning curves. The learning curves are plotted by considering the number of iterations/samplings (x-axis) against the error rates (y-axis) observed.

After each sampling iteration we add a fixed number of rules specific to the dataset to the model before proceeding for the next iteration. For the *diabetes*, *german* datasets the number of rules generated and added after each iteration (k rule parameter) was set to 4 and for the *segment* dataset it was set to 7.

The k parameter is chosen from the range of values defined in an interval. The bounds for this interval are decided based on the *minimum* and *maximum* number of queries(instances) we can use for active learning here. The k values denoted here are the ones for which AAL performed the best (*lowest error rate*) with *minimum* data utilization among all the values considered in the interval. In this manner once the sampling iteration size has been decided on each dataset AAL was run over for 20 iterations.

While comparing the learning curves we used different kinds of sampling methods available in the literature. The sampling methods used for the experiment below are as follows

1. **Random Sampling:** In this we randomly choose a sample from the available distribution and add it to the model. Each sample has the same chance of being chosen to be added.

2. **Uncertainty Sampling:** As explained in the introduction we choose the samples which the model is most uncertain about. This has been explained in the Introduction section.

3. **Selective sampling using Label variation/Variation Sampling:** In this method we use a selective sampling procedure which chooses samples to label based on how much expected change it brings to the base model. The change is calculated through a defined function and those examples are chosen which are assigned the maximum expected change scores. [8]

4. **Dynamic Rule Sampling:** The sampling method used by our proposed AAL algorithm.

Table 4: Data utilization with respect to ACTIVE-DECORATE

Dataset	**DECORATE**	**ACTIVE-DECORATE**	**AAL**	**Target error**
Iris	32(1.0)	30(**0.94**)	30(0.94)	5.25%
Diabetes	234(1.0)	201(0.86)	83(**0.35**)	25.1%
Glass	118(1.0)	100(0.85)	81(**0.686**)	27%
Breast-w	30(1.0)	39(1.30)	20(**0.67**)	3.94%
Heart-h	49(1.0)	39(0.8)	30(**0.612**)	19.93%
Heart-c	50(1.0)	36(0.72)	30(**0.6**)	20.97%
Hepatitis	39(1.0)	23(**0.59**)	42(1.07)	16.96%
Lymph	27(1.0)	24(0.88)	13(**0.48**)	22.21%
Sonar	125(1.0)	99(0.79)	50(**0.4**)	18.39%
Soybean	492(1.0)	144(**0.29**)	190(0.386)	6.59%
Number of Wins	0	3	**7**	

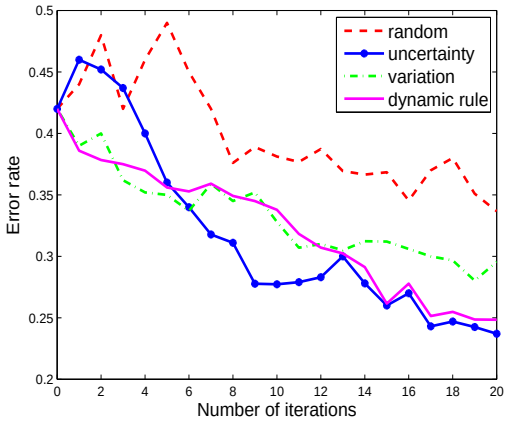

Figure 4: Diabetes dataset (k=**4**)

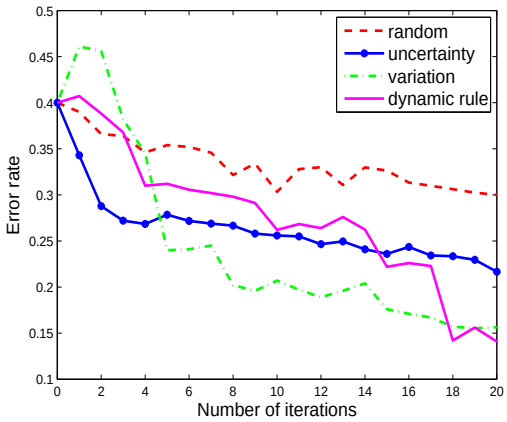

Figure 5: German dataset (k=**4**)

5.3.1 On Binary datasets

We obtain the learning curves for the *diabetes* and *german* datasets below with the settings described earlier. Here each iteration implies a sampling routine where records are sampled and used by the learner. The k parameter from the AAL algorithm chosen for each dataset is mentioned below the figures.

It is noticed that for the diabetes dataset (Figure 4) uncertainty sampling and dynamic rule based sampling(DR) used by AAL attain the lowest error rates. For the german dataset (Figure 5) we notice that DR closely ties with the variation based sampling with DR being better. However, both these datasets are simple binary datasets, so we further explore the learning curves obtained from a few multi class datasets.

5.3.2 On multiclass datasets

The datasets here for the next sampling experiment are are soybean(19 classes) and Image segmentation(7 classes). As variation sampling is known to be highly computationally intensive on multiclass datasets we avoid including it here in this experiment on multi class datasets. To effectively compare the performance of AAL here we compare it against a multi class svm and active learning framework (SVM+AL).

The sampling procedure we use for comparison is called representative sampling. [22, 16] It differs from the earlier methods in sampling points which are also effective representatives of the distribution in the dataset. We have used the multiclass svm suite here for inferring the margin and points lying within the margin. It is noticed from the learning curves that for the segment dataset (Figure 6) uncertainty, representative perform well with DR being marginally better. However for the soybean dataset (Figure 7) representative and DR outperform other routines with both of them ending up at the same error rate.

After observing the learning curves over these four datasets one can say that the dynamic rule (DR) sampling is more robust and consistent than the other sampling methods. The phenomenon of decrease in error

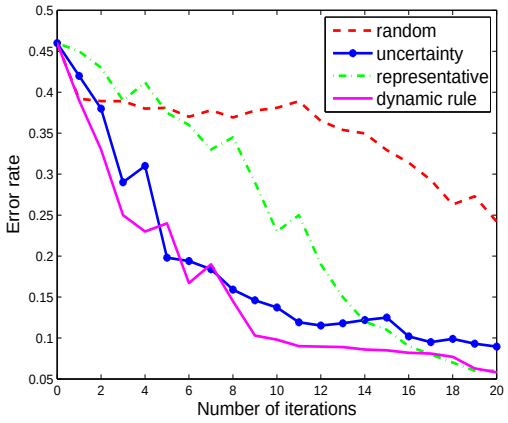

Figure 6: Segment dataset (k=7)

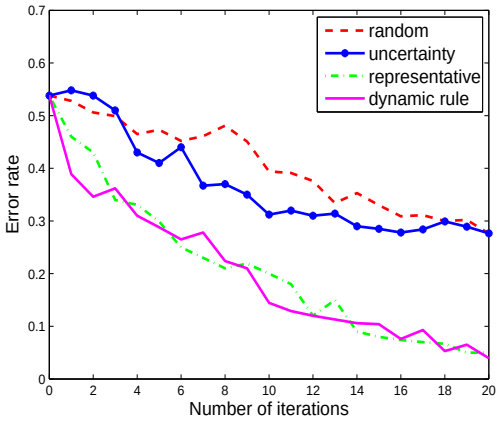

Figure 7: Soybean dataset(k=10)

rates with iterations is prominently visible with DR sampling in all the datasets especially with multi class datasets. This can be attributed to the following reason.

In multi class domains general classifiers have to face many *ambiguous* instances while classifying. This is because the predicted space is also highly populated. In such cases rules which can help us discriminate between classes can be very effective. As our AAL based DR sampling effectively generates such rare rules from the Minimal infrequent itemsets (MIF) present in the dataset, the results are more pronounced in multi class datasets.

5.4 Labeling rate plots

In this experiment we look at the labeling rate plots derived for the datasets we have considered. In our earlier experiments itself we have proved how our AAL was superior to ACTIVE-DECORATE from the the data utilization perspective. In continuation with our first experiment on data utilization here we look at the labeling plots obtained from AAL on few datasets. These plots will help us understand the number of learning queries being posed by the learner during each sampling iteration. The x-axis represents the number of records seen (labeled + unlabeled) and the y-axis represents the number of labels requested by the learner.

We produce the labeling plots for the diabetes (Figure 8) and waveform (Figure 9) datasets respectively. For the diabetes dataset one can observe that the initial number of queries posed are very high which is followed by a steep and pronounced decrease as the number of iterations increase. The plot obtained here is pretty much as expected. However, in the case of the waveform dataset the phenomenon of decreasing queries is visible after a few iterations. The behavior patterns cannot be standardized for all the real life datasets because of their differing complexity and structure. However, the consistent dip in the number of queries being posed should be visible after a few iterations.

So all these experiments prove that sampling rules dynamically is a superior procedure in comparison to the remaining sampling frameworks. Even though association rules summarize datasets pretty effectively they cannot encompass all the necessary information. Hence the knowledge from the Minimal infrequent itemsets (MIF) may be crucial to build better learning models. We believe that these results also support our initial hypothesis of using MIF in active learning.

6 Related Work

In this section, we look at the related work in the area of active learning. The earliest active learning procedure suggested was QBC(Query by committee) which builds a committee of learners and accordingly chooses samples to be labeled by the oracle. QBC was found to detect controversial examples and spend less time on sampling outliers. Query by bagging and boosting procedures were further developed which applied the bagging and adaboost machine learning techniques in a QBC framework [10] to create the committee members intelligently. However it was observed that Query by Boosting did not consistently outperform Query by bagging. Active learning with multiple views [15] was then proposed by Muslea where all the committee members represent redundant views on the learner. This method was called *Co-testing*. These disjoint feature vectors can be independently used for classification.

Melville and Mooney [13] further proposed the DECORATE algorithm which relies on generating diverse committees. The committee members are generated using synthetic samples always ensuring that the models added to the committee are superior in comparison to their predecessors. ACTIVE DECORATE [14] was also developed which used DECO-

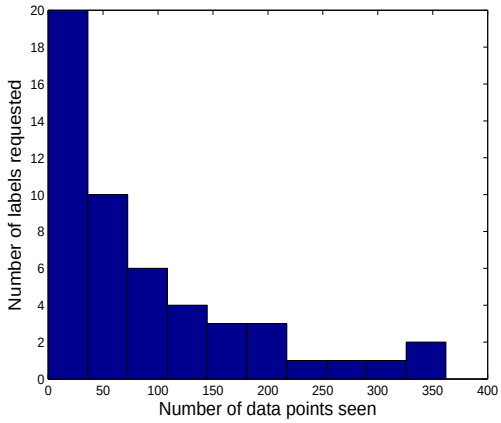

Figure 8: Diabetes dataset

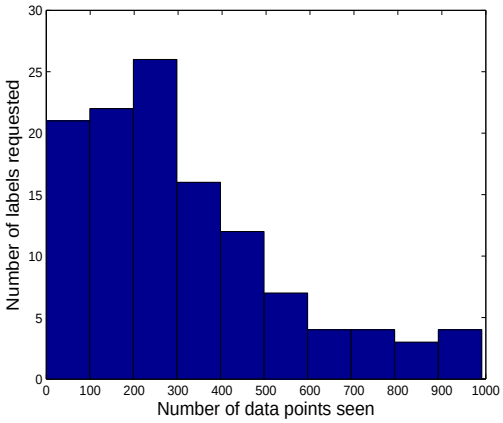

Figure 9: Waveform dataset

RATE committees in an active learning scenario and sampled records whose margin was pretty small. This work has been extended to further enhance the diversity of the committee members by using misclassification sampling procedures while generating the diversity data. Support vector machines have also been employed in active learning [21]. Cohn and David [17] proposed an svm based active learning approach where instances closer to the margin hyperplane were evaluated based on an predictive error function. The evaluation was based on computing the cumulative error for the instances by adding them with the labels (-1,+1) in the model.

Apart from different algorithms many sampling procedures have also been proposed for active learning. Variation in labels based [8], representative based [22] and density based sampling [5] are few procedures of choosing records to be labeled by the oracle. In representative sampling using SVM those points are identified which lie within the margin. These points are later clustered using a k means algorithm and the cluster centers are then sampled for labeling.

On the other hand density based sampling uses expectation maximization principles and a logistic regression classifier to estimate the posterior probabilities. It has generally proven to be effective because it considers the underlying distribution and chooses representatives of large clusters. One major advantage is that it tries to cover the input space quickly. Our method in comparison to this exhibits faster reduction in error. It along with density sampling is a dynamic strategy in comparison to the static strategies proposed above. Hybrid methods such as DUAL sampling [5] which combine both density and uncertainty sampling have also been proposed.

SIMPLE, SELF-CONF and KFF [2] are few famous online active learning algorithms. Each of them have different induction components. For each trial the querying function requests the expert to label the object and this is added to the training data. Every time a new classifier is induced for each of these trials. Apart from individually applying active learners even an ensemble of active learners like COMB can be applied for obtaining better performance. More recently Dasgupta and Kalai have proposed perceptron based active learning approaches [4]. The algorithm uses a modified perceptron based update where in it only stores the current hypothesis rather than storing previously seen data points. The goal of their work is to provide a simpler solution for the problem of developing a linear separator for data distributed uniformly over the unit sphere.

7 Conclusions and Future Work

In this paper, we have handled the problem of active learning from an itemset point of view. The problem of active learning is popular among the research community because it facilitates in building effective learners. Online learning and Meta learning approaches are being applied in different kinds of domains such as music retrieval, protein structure prediction etc.

To identify rare and informative occurrences in data we have used the idea of Minimal Infrequent Itemsets (MIF) here. Our motivation here relies on the fact that traditional learners obviously train well on the frequent patterns but do not train adequately on the rare and informative patterns. We believe such information is important as it can definitely help in building high precision discriminative learners.

Traditional active learning so far has been perceived in various forms such as ensemble based, unit sampling and hybrid sampling but this is the first of its kind approaches which unifies association rules with active learning. The problem of Associative active learning can be perceived and designed in further ways by modifying the sampling function being used etc. Further research could be focused on developing efficient performance metrics for active learners.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. pages 487–499. VLDB, 1994.

[2] Y. Baram, R. Yaniv, and K. Luz. Online choice of active learning algorithms. pages 255–291. Journal of Machine Learning Research, 2004.

[3] E. Boros, V. Gurvich, and L. Khachiyan. On the complexity of generating maximal frequent and minimal infrequent itemsets. pages 133–141. 19th Annual Symposium on Theoretical Aspects of Computer Science, 2002.

[4] S. Dasgupta, A. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. pages 281–299. Journal of Machine Learning Research, 2009.

[5] P. Donmez, J. Carbonell, and P. Bennet. Dual strategy active learning. pages 116–127. ECML PKDD, 2007.

[6] D. Haglin and A. Manning. On minimal infrequent itemset mining. pages 141–147. DMIN, 2007.

[7] A. Jorge and P. Azevedo. An experiment with association rules and classification: Post-bagging and conviction. pages 137–149. Discovery Science, 2005.

[8] P. Juszczak and R. Duin. Selective sampling based on the variation in label assignments. pages 375–378. ICPR, 2004.

[9] W. Lewis, D. Gale. A sequential algorithm for training text classifiers. pages 3–12. ACM SIGIR, 1994.

[10] H. Mamitsuka and N. Abe. Efficient data mining by active learning. pages 258–267. Progress in Discovery Science, 2002.

[11] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed reprepsentations. pages 189–194. KDD, 1996.

[12] H. Mannila and H. Toivonen. Level wise search and borders of theories in knowledge discovery. pages 241–258. Data Mining and Knowledge Discovery, 1997.

[13] P. Melville and R. Mooney. Constructing diverse classifier ensembles using artificial training examples. pages 505–512. International Joint Conference on Artificial Intelligence IJCAI, 2003.

[14] P. Melville and R. Mooney. Diverse ensembles for active learning. pages 584–591. ICML, 2004.

[15] I. Muslea, S. Minton, and C. Knoblock. Selective sampling with redundant views. pages 621–626. National conference on Artifical intelligence, AAAI, 2000.

[16] H. Nguyen and A. Smeulders. Active learning using pre-clustering. ICML, 2004.

[17] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. ICML, 2000.

[18] B. Settles. Active learning literature survey. Number 1648, 2009.

[19] B. Settles, M. Craven, and L. Friedland. An analysis of active learning strategies for sequence labeling tasks. pages 1069–1078. Empirical methods in Natural Language processing, 2008.

[20] F. Thabtah. A review of associative classification mining. pages 36–65. The Knowledge engineering Review, Volume 22, 2007.

[21] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. pages 45–66. Journal of Machine Learning Research, 2001.

[22] Z. Xu and K. Yu. Representative sampling for text classification using support vector machines. ECIR, 2003.

[23] M. Zaki and M. Ogihara. Theoretical foundations of association rules. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 1998.

[24] M. Zaki, S. Parthasarathy, and M. Ogihara. New algorithms for fast discovery of association rules. pages 283–286. KDD, 1997.