

# Lazy Learning for Classification Based on Query Projections \*

Yiqiu Han<sup>†</sup>

Wai Lam<sup>‡</sup>

## Abstract

We propose a novel lazy learning method called QPAL. QPAL does not simply utilize a kind of distance measure between the query instance and training instances as many lazy learning methods do. It attempts to discover useful patterns known as *query projections*, which are customized to the query instance. The discovery for useful QPs is conducted in an innovative way. QPAL can guarantee to discover high-quality QPs in the learning process. We use some benchmark data sets and a spam email filtering problem to evaluate QPAL and demonstrate that QPAL achieves good performance and high reliability.

## 1 Introduction

The idea of lazy learning [1] has been proposed as the contrary of common eager learning algorithms. Common eager learning methods eagerly compile the training data into some concept descriptions (e.g., rule sets, decision trees, networks, graphical models). They attempt to seek a particular general hypothesis, which covers the entire instance space. In contrast to eager learning, lazy learning models do not involve any model construction before they encounter the unseen instance to be classified, implying that they do not conduct any processing until they are requested. The customized model and all the intermediate results are discarded when the learning process for this unseen instance completes. Therefore lazy learning algorithms need much less training costs but more storage and computational resources than eager algorithms during classification. Nevertheless lazy learning algorithms can make use of the characteristics of the unseen instance to explore a richer hypothesis space during classification.

---

\*The work described in this paper was substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Nos: CUHK 4187/01E, CUHK 4179/03E, and CUHK 4193/04E) and CUHK Strategic Grant (No: 4410001).

<sup>†</sup>Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong

<sup>‡</sup>Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong

In fact lazy learning methods sometimes significantly outperform some eager algorithms for particular learning tasks.

Lazy learning exhibits advantages in many learning scenarios. For example, some round-the-clock 24-hour online services such as spam email filtering may require to update the training data frequently, without interrupting the service. Common eager learning methods need to learn a new global classifier every time the training data is updated. When the training data is large and complex, it is not economical for the service provider to conduct eager learning frequently. Lazy learning methods have no such problems. Generally, the updating of training data is the only operation required by lazy learning methods.

Another learning scenario for which lazy learning is competitive is that the learning target class is not fixed and the attribute set is large. Under such a complex circumstance, the attribute set is usually not oriented to a specific learning task. There may be many irrelevant attributes as well as incomplete data. For example, suppose a global enterprise possesses a huge business data base with an extremely large number of attributes. The query instances to be classified are delivered from time to time, where the intended class attribute may vary from time to time too. One query might be “What will be the profit level given a specific district and a specific time period?”. Another query might be “Whether a new product of an existing category will be well accepted among a specific group of customers?”. Under these circumstances, there can be numerous tuples of a attribute set and a target class. It is not only expensive but sometimes also infeasible to conduct eager learning every time a query of different target class is given. In contrast, lazy learning has advantages in handling such problems in an efficient way. The reason is that lazy learning handles each classification as an independent learning process, and hence it can be customized to the unseen instance and focuses only on the local data patterns.

Many lazy learning algorithms are also described as “instance-based” or “memory-based”. Suppose we need to predict the class label of an unseen instance, called the *query*. Many lazy learning methods collect the training instances similar to the query instance for learning.

Generally a distance metric will be utilized to quantify the impact of each training instance. In this paper, we propose a novel lazy learning method, which takes a subset of the query attributes as a learning unit. Our learning method is called *Query Projection Analytical Learning (QPAL)* which explores the projections of the query instance for learning. A *query projection* (QP) is represented by a set of attribute values shared by the query and, potentially, some training instances [9]. The utilization of QPs for learning helps achieve a balance between precision and robustness with a richer hypothesis space. QPAL explores and analyzes QPs, attempting to generate an appropriate set of QPs. The final prediction is made by combining some statistics of the selected QPs.

The learning process of QPAL is customized to the query instance. There is no global model construction. Nothing is done until the query comes. Moreover, after answering a query, the customized model and all the intermediate results are discarded. Hence QPAL can be regarded as a lazy learning method. QPAL starts with the query, which can be regarded as the most specific QP. By gradually removing some attribute values, QPAL obtains more general QPs. Note that these QPs should be supported by training data. Otherwise they will not be considered. Then QPAL will iteratively combine these QPs to produce more general QPs until the stopping criteria is met. At each step, QPAL will investigate the empirical class distribution of the current QP, then decide whether it should be selected or discarded or reserved as seeds. In essence, QPAL has several distinct characteristics as follows.

First, QPAL focuses on the local QPs rather than learning a set of rules which form a general classifier. The learning process is tailored to the query rather than partitioning the whole attribute hyperspace to obtain a global classifier such as a decision tree. Particularly, for real-world problems where the training data needs to be frequently updated such as spam email filtering problem, QPAL has an advantage of reducing the cost of maintenance and operation.

Second, from the perspective of concept learning, the discovered QPs can be regarded as a kind of classification knowledge. A number of classical learning algorithms such as decision tree [15] or rule-based learning [12, 13] have a common characteristic. The data space is recursively partitioned in a greedy manner, which usually leads to the horizon effect [2]. The reason is that the heuristic used to guide the partition commonly looks no further than the next attribute to select. Several extensions [17, 18, 14] have been proposed to cope with the horizon effect, but the optimal result still cannot be guaranteed. However, QPAL does not

intend to construct a global model. Its learning process is customized to the query instance. Hence QPAL does not suffer from the horizon effect as many classical eager learning models do. We will demonstrate that QPAL can guarantee to discover high-quality QPs efficiently. It can guarantee a sufficiently low probability to reject any useful QPs with an expected accuracy higher than a specified value.

Third, unlike many existing lazy learning methods [5, 6, 10, 11], QPAL does not employ ordinary Euclidean distance as the weighting scheme of training instances. Instead, QPAL considers the weighting of a QP, or a group of training instances. It makes the best of the “more-general-than” relationship which is simple but reliable.

In order to evaluate the effectiveness of QPAL, we conducted extensive experiments with benchmark data sets and a spam email filtering problem. QPAL achieves good classification performance and it also exhibits higher reliability and scalability with attribute dimension and the number of training instances.

In Section 3, we will introduce the concept of query projections and their utility in learning. Then we discuss the learning framework based on QPs. A QP selection metric and a set of rules are proposed to facilitate the discovery of useful QPs. In Section 4, we will show how undecided QPs are analyzed and present the learning algorithm. Section 5 empirically investigates the learning process of QPAL. In Section 6, we will discuss how to extend our QPAL to handle continuous attribute values. The experimental results and discussions are given in Section 7. Section 8 gives the conclusion and future work.

## 2 Related Work

A simple but effective way to conduct lazy learning is the classical  $k$  nearest neighbor (kNN) model and its variants [8, 5, 6, 10]. Intuitively, kNN model can be viewed as locating a fixed number of training instances, namely nearest neighbors, for the unseen instance. It uses their class labels to predict the unseen instance. The main difference between QPAL and kNN is that QPAL considers QPs closest to the unseen instance rather than the nearest training instances. Therefore, QPAL is more like a model-selection method rather than a case-based learning method. Second, QPAL does not require ordinary distance metric between instances as kNN does. kNN either assigns the same weight to every nearest neighbor or defines a weighting scheme based on the distance metric. The complicated weighting scheme might incur more computational cost and lose the advantage of pure lazy learning. QPAL uses a different scheme to perform both feature selection and

feature weighting. Third, kNN is lack of interpretability and might fail to uncover some useful but lower-order attribute patterns. QPAL can output QPs as discovered knowledge or explanations of the prediction.

Recently, Li et al. [11] proposed a learning framework, called DeEPs, using emerging patterns. It makes use of the frequency of an instance’s subsets of attribute values and the frequency-change rate of the subsets among training classes to perform learning. Both DeEPs and QPAL have the idea of using subsets of attribute values rather than using distance. QPAL is different from DeEPs with respect to the characteristics of QP-based patterns and the mechanism for obtaining good QPs. Moreover, DeEPs tends to favor those frequent patterns having an infinite rate, i.e., all associated instances having the same label. QPAL considers a tradeoff between the frequency and the pattern of class distribution. QPAL also utilizes an exploration algorithm rather than the set operations in DeEPs. These characteristics enable QPAL to select subsets of attribute values in a more principled manner.

Chang and Li has recently proposed a maximizing expected generalization algorithm for learning complex query concepts (MEGA) [4]. It learns an online query concept with the minimum number of labeled instances through active learning, and it can sustain learning under working conditions when no relevant examples are provided at the beginning of the active learning process. A divide-and-conquer method is used to divide high-dimensional features into a number of groups to speed up the learning. Our method resembles MEGA in the idea of decomposing query into groups of attributes, but the objective of learning is different. QPAL focuses on handling common learning problems such as classification. MEGA mainly addresses active online query answering problems.

### 3 Learning with Query Projections

**3.1 The Concept of Query Projection** Suppose the learning problem is defined on a class variable  $C$  and a finite set  $\mathbf{F} = (F_1, \dots, F_n)$  of discrete random variables, i.e., attributes. Each attribute  $F_i$  can take on values from respective domains, denoted by  $V(F_i)$ . To simplify the discussion without loss of generality, we assume that the class variable  $C$  is a Boolean variable since a multi-class variable can be broken into a set of binary variables. A query instance  $\mathbf{t}$  is denoted by a full set of attribute values  $\{t_1, t_2, \dots, t_n\}$  where  $t_i \in V(F_i)$ . Its QPs can be viewed as subsets of the set  $\{t_1, t_2, \dots, t_n\}$ . We denote the cardinality of a QP  $\mathbf{A}$  as  $|\mathbf{A}|$  where  $\mathbf{A} \subseteq \mathbf{t}$ .

For example, suppose the query is to classify whether the sale is good, or poor, or aver-

age. The attributes are  $\{Season = Fall, Region = Asia, Product = Laptop, Clients = Students\}$ . The manager cannot achieve the goal with incomplete or insufficient data. However, the data in his hand might deliver a convincing summary of the sale at a more general level, i.e.,  $\{Fall, Asia, ?, Students\}$  or  $\{Fall, ?, laptop, ?\}$ . These subsets of the query are QPs. This paper will show that some of them can contribute to learning, or at least bring users closer to the answer for the query. The QPs of a particular query constitute a lattice. The links between QPs reflect the “more-general-than” relationship [16], which can be utilized to explore QPs for learning. A QP is valid only when it is associated with some instances belonging to its concept.

Each valid QP can be regarded as a sub-classifier whose prediction is the majority class among its associated training instances. Suppose a QP has  $x$  associated training instances, which is called *frequency*. Among the  $x$  associated training instances,  $y$  instances sharing the same majority class label is called *majority count*. Then  $y/x$  is called the *majority rate* of that QP. It can also be viewed as the empirical accuracy of applying that particular QP for classification.

The ideal case is to have a QP whose associated instances 100% belonging to the same class. Since the query is a specialization of the given QP, the query should also belong to that class. However, in practice, there are several issues to be considered. First, sometimes the highest majority rate among all valid QPs is far less than the ideal case. Second, the empirical majority rate may be unreliable when there are insufficient associated instances. Third, there might be more than one QPs that can serve as good sub-classifiers.

**3.2 Exploring Valid QPs** Since some QPs can be regarded as sub-classifiers, we use a set of approximately optimal QPs to help learn the query. Suppose a set of qualified QPs are discovered by the method which we are going to discuss below. These discovered QPs are then combined to classify the given query. The final prediction is made by summarizing the frequency of each class on all selected QPs. The class with the maximal frequency serves as the predictor for the class label of the given query. In this step, since a particular single training instance may appear in different QPs, we can observe that it may be sampled multiple times proportional to its contribution to the learning. This usually happens when a training instance exhibits relatively high similarities to the query.

As there are  $2^n$  subsets of a query  $\{t_1, t_2, \dots, t_n\}$ , It is a computational challenge to search for a set of QPs with unknown size. In fact there are  $2^{2^n}$  combinations.

We develop techniques to cope with this problem. First, the number of valid QPs is usually much smaller than  $2^n$ , because a valid QP must have some associated training instances. The QPs without sufficient data support cannot provide any help in learning, and hence can be ignored. Second, we design a method to explore and discover useful QPs. It can systematically examine valid QPs.

In our method, QPs are explored in sequence based on their cardinalities. Suppose the largest cardinality among all QPs is  $k$ . The exploration starts with QPs with the largest cardinality  $k$ . They can be enumerated and explored directly by investigating all qualified training instances, i.e., sharing exactly the same  $k$  attribute values with the query.

For a valid QP  $\mathbf{S}$  whose  $|\mathbf{S}| < k$ , it must satisfy one of the following conditions.

*Cond1:*  $\mathbf{S}$  has associated training instances which share exactly  $l$  identical attribute values with the query.

*Cond2:*  $\mathbf{S}$  is a common subset of two or more explored QPs whose cardinalities are all larger than  $l$ .

*Cond3:*  $\mathbf{S}$  is a pure subset of an explored QP whose cardinality is larger than  $l$ . All  $\mathbf{S}$ 's associated training instances are also inherited from that explored QP. QPs belonging to this condition need not to be considered in the exploration discussed below.

Thus all valid QPs with cardinality  $l < k$  can be exhaustively enumerated in two ways. The first way is to explore all training instances which share exactly  $l$  identical attribute values with the query. The second way is to check all common subsets of explored valid QPs with cardinalities larger than  $l$ .

Consequently, we can explore all valid QPs systematically, from the largest cardinality to the smallest cardinality, via scanning the training data. The whole process is conducted by examining all available training instances. Then the number of valid QPs is constrained by both  $2^n$  and the number of training instances that share at least one attribute value with the query  $\mathbf{t}$ . They are ranked according to their number of identical attribute values with the query. Thus at each step only a small fraction of training data needs to be read into memory. With this connection between QPs and instances, operations on QPs are transformed into operations on training instances.

The learning process is usually completed after only examining those training data which is closely related to the query. For problems with large attribute dimensions

but relatively sparse training data, our method has obvious advantage in the capability of exploring all valid QPs. Even for problems with both large attribute dimensions and large number of data instances, our method can still efficiently reduce the search space and discover useful QPs with the aid of a set of rules, as discussed below.

**3.3 The Metric for Selecting Useful QPs** The purpose of exploring QPs is to help learn the query. Hence we propose a selection metric to find the most useful QPs. As we have stated, the learning for the query prefers QPs with a reliably high majority rate.

To select the most useful QPs, we set two thresholds which can be adjusted to meet the demand in practice. One is the minimal tolerance accuracy for a QP, denoted by  $q$ . The other is the minimal expected accuracy for a QP, denoted by  $p$ .  $p$  is always selected to be greater than  $q$ .

Suppose a QP has the expected accuracy equal to  $p$ . We denote the probability of observing its  $y$  majority class instances among its all  $x$  associated training instances by  $P(y/x)$ , expressed as follows:

$$(3.1) \quad P(y/x) = \frac{x!}{y!(x-y)!} p^y (1-p)^{x-y}$$

For an arbitrary QP whose  $y/x \leq q$ , its actual accuracy might be higher than the minimal tolerance accuracy  $q$ . However, through Equation 3.1, we can control the possibility of incorrectly rejecting such kind of QPs. If  $P(y/x)$  of this QP is less than a threshold  $\gamma$ , we will reject to consider it further. Equation 3.2 illustrates the condition of rejecting a QP whose majority rate is lower than  $q$ .

$$(3.2) \quad \left(\frac{y}{x} \leq q\right) \wedge \left(\frac{x!}{y!(x-y)!} q^y (1-q)^{x-y} \leq \gamma\right)$$

Consequently, the QPs with a low majority rate  $y/x$  and relatively large frequency  $x$  will be filtered out. We only have less than  $\gamma$  (e.g., 5%) chance to incorrectly reject an useful QP with an accuracy higher than  $q$ .

On the contrary, if  $y/x > p$ , we also consider the possibility to accept this QP as with an accuracy higher than  $p$ . Equation 3.3 depicts the fact that the probability of a QP having an expected classification accuracy less than  $p$  is no more than  $\gamma$ .

$$(3.3) \quad \left(\frac{y}{x} \geq p\right) \wedge \left(\frac{x!}{y!(x-y)!} p^y (1-p)^{x-y} \leq \gamma\right)$$

Since  $p$  is selected to be greater than  $q$ , QPs with large frequency  $x$  and high majority rate  $y/x$  are preferred and accepted as sub-classifiers. We only have less than

$\gamma$  (e.g., 5%) chance to select an unreliable QP with expected accuracy less than  $p$  (e.g., the majority rate of the whole training data set).

If a QP has neither been accepted nor been rejected, it is then stored for further analysis, which we will discuss in Section 4. In the analysis, the common subsets of stored QPs are used to generate new useful QPs as stated by the condition *Cond2* mentioned in Section 3.2.

**3.4 Discovering Useful QPs** Figure 1 depicts the complete process of exploring QPs. First all valid QPs are ranked and examined by scanning the training data. Then we use a selection metric to find useful valid QPs. If a QP is accepted as a good predictive sub-model, it will be selected as discovered knowledge. If a QP is rejected, it will be discarded. If a QP is neither selected nor discarded, it is then stored. The common subsets of stored QPs are used to generate new QPs which could not be explored directly from the training data. Finally the discovered QPs are combined to predict the query as stated in Section 3.5. In this exploration, a set of straightforward but effective rules are utilized to improve the efficiency. These rules are designed on a basis of the subset relationship between QPs.

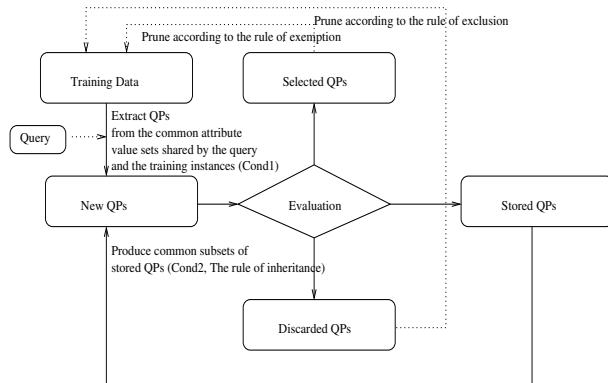


Figure 1: Exploring QPs for a given query.

In QPAL, two QPs can only be compared with each other when they have the subset relationship. This can significantly improve the learning efficiency. This approach also avoids introducing any apriori assumptions such as dependence or independence of attributes, a global Euclidean distance metric, or a weighting scheme for different attributes. This makes the learning from QPs more flexible and more robust. We introduce a property called *Posterior Property*, which can be viewed as the inverse of the common *Apriori Property* used in classical association rule learning. Apriori property can be expressed as “nonempty subsets of a frequent item

set must also be frequent.” Posterior Property can be expressed as “A specialization (superset) of a good QP is also a good QP”. For instance, if  $S_1$  has a high majority rate and  $S_1 \subset S_2$ , then  $S_2$  has a greater chance to have a high majority rate. As an extreme case, if  $S_1$  has a 100% majority rate, so does  $S_2$ . If  $S_2$  is observed to have low majority rate, it implies that  $S_1$  also cannot have 100% majority rate.

Based on the above arguments and having  $S_1 \subset S_2$ , we design the following rules:

- *Rule of exemption* : If  $S_2$  has been selected, which means that it has sufficiently high majority rate and sufficient associated training instances, then  $S_1$  should not be considered any more because  $S_1$  is a generalization of  $S_2$  and contains less information from the query. Returning to the previous example,  $\{Fall, Asia, ?, Students\}$  can suppress  $\{Fall, Asia, ?, ?\}$  if the former has sufficient data support. Meanwhile,  $\{Fall, Asia, ?, Students\}$  cannot suppress  $\{Fall, ?, laptop, ?\}$  although the former has larger cardinality.
- *Rule of exclusion* : If  $S_2$  is discarded, then  $S_1$  should not be considered any more due to the same reason as the Rule of exemption. This rule can also be regarded as an extension of the Rule of exemption.
- *Rule of inheritance* : If  $S_2$  has the same associated training instance set as  $S_1$ ,  $S_2$  need not to be considered as a valid QP since  $S_1$  can replace it without any cost.

The above rules show that QPs can be exploited by analyzing their relationship and observing their class distributions. The decision of selecting or discarding a particular QP will trigger a family of QPs to be pruned. These operations can significantly accelerate the searching for an appropriate set of useful QPs. Figure 2 describes pseudo-code of the discovery process of useful QPs. Note that most operations have been transformed into operations on training instances, as shown in Steps 1-2. Steps 6 and 9 use Equations 3.2 and 3.3 to decide whether to accept or reject a QP. Steps 7 and 11 employ the rules discussed above to accelerate the discovering process. The processing of undecided QPs at Step 14 is discussed in Section 4. These components constitute our QPAL learning method.

### 3.5 Predict the Query with Discovered QPs

After QPAL discovers an appropriate set  $r$  of useful QPs, a majority voting is conducted among the training instances associated with selected QPs. Each discovered QP can be viewed as a sub-classifier and the final

---

```

1 Sort training instances according to the highest cardinality
  of associated QPs.
2 Put all training instances associated with no-empty QPs into  $\Delta$ .
3 Initialize the active set  $\mathbf{u}$  and the final set  $\mathbf{r}$  to be empty.
4 FOR  $j = n$  to 1
5   FOR every QP whose cardinality is  $j$  and owns
     associated instances in  $\Delta$ 
6     IF Equation 3.2 holds for this QP.
7     Remove all instances whose associated QPs are all
     subsets of this specific QP.
8   ELSE
9     IF Equation 3.3 holds for this QP.
10    Insert this QP into  $\mathbf{r}$ .
11    Remove all instances whose associated QPs are all
     subsets of this specific QP.
12  ELSE
13    Insert this QP into  $\mathbf{u}$ .
14  Analyzing the QPs in  $\mathbf{u}$  as stated in Section 4.
15  IF  $|\Delta| == 0$ 
16    BREAK
17  IF  $|\mathbf{r}| = 0$ 
18  Move all QPs with the largest cardinality into  $\mathbf{r}$ .
19 Return the discovered QP set  $\mathbf{r}$ .
```

---

Figure 2: The pseudo-code of the discovery process of useful QPs

prediction is a combination of those sub-classifiers. The class with the maximum sum of frequencies in all discovered QPs is the classification result of the query instance.

It should be noted that associated training instances are sampled for multiple times if they are associated with more than one QP. Consequently, the training instances closer to the query tend to have a larger impact on the final prediction, because they have larger chances to associate with more discovered QPs. Moreover, the sampling scheme leads to a weighting scheme of the discovered QPs. A QP with more support from training instances plays a more important role in the final decision. The weight for a discovered QP is proportional to its observed frequency, which can be viewed as the confidence on its associated sub-classifier.

#### 4 Generate Useful QPs from Undecided QPs

As we have discussed, the QPs are extracted from the training data and then considered in a systematic manner. In this process, there might be some QPs that can be neither rejected nor accepted, as shown in Step 14 in Figure 2. For example, a QP of the largest cardinality has the frequency  $x = 5$  and the majority count  $y = 4$ . With the high majority rate, i.e., 80%, it will not be rejected. But its low frequency makes it

also unacceptable. Generally these QPs are reserved in a temporary QP set  $\mathbf{u}$ . If the whole process does not produce any interesting QPs with Equation 3.3,  $\mathbf{u}$  is analyzed to generate some useful QPs for learning.

The generation of useful QPs is conducted in an iterative manner. First, QPs are divided into groups according to their majority class. In each group, the mating of any pair of QPs will reproduce a new QP, which is the common subset of its parents. These newly generated QPs will be considered to be accepted or rejected. If there are still no interesting QPs satisfying Equation 3.3, the remaining QPs, i.e., QPs not satisfying Equation 3.2, will become the updated QP set  $\mathbf{u}$ . This will iterate until some useful QPs are found. If there is no new generation of QPs, i.e.,  $\mathbf{u}$  becoming empty, this generation process will automatically terminate. The details of the generation is described in Figure 3.

---

```

1 Divide all QPs in  $\mathbf{u}$  into groups of different majority class.
2 Initialize an empty set  $\mathbf{v}$ .
3 FOR each group of QPs
4   FOR any pair of QPs in the current group
5     Produce the children QP  $m$ .
6     IF Equation 3.2 holds for  $m$ .
7     Discard  $m$ .
8   ELSE
9     IF Equation 3.3 holds for  $m$ .
10    Insert  $m$  into  $\mathbf{r}$ .
11  ELSE
12    Insert  $m$  into  $\mathbf{v}$ .
13  IF  $|\mathbf{r}| = 0$ 
14     $\mathbf{u} = \mathbf{v}$ 
15  IF  $|\mathbf{u}| \neq 0$ 
16    GOTO 2
17  ELSE
18    END
```

---

Figure 3: Generate useful QPs from undecided QPs

From another perspective, the analyzing of undecided QPs in  $\mathbf{u}$  is to work through valid QPs belonging to the condition *Cond2* mentioned in Section 3.2. For those QPs cannot be explored by using the training data directly, we need to use existing QPs, i.e.,  $\mathbf{r}$ , to access them. The QPs rejected by Equation 3.2 and the QPs accepted by Equation 3.3 are excluded according to the rules in Section 3.4. Therefore, all valid QPs can be scanned by QPAL efficiently. This property helps QPAL to avoid the horizon effect. Particularly, suppose there is a valid QP with 100% accuracy. QPAL can guarantee the discovery of this QP, which cannot be done by common eager learning algorithms such as decision tree

or association rule-based learning. The proof is given below.

**Proof.** Suppose there exists a QP  $S_i$  with 100% expected accuracy. Then each QP  $S_j$  satisfying  $S_i \subseteq S_j \subseteq \mathbf{t}$  should also be with 100% expected accuracy. Hence they will not be discarded in the learning process and at least one of them will be employed according to Equation 3.3. Common eager classification rule learning methods cannot achieve this, because all QP  $S_k$  satisfying  $S_k \subseteq S_i \subseteq \mathbf{t}$  may not have distinguishable information gain or observed accuracy although  $S_i$  can. This observation can be found in the two synthetic problems in Section 5.

Although the generation process adds some computational cost. We can show that the time complexity of QPAL is still acceptable, and even lower than many common eager learning algorithms. First of all, The time complexity of QPAL is bounded by the number of valid QPs rather than theoretical  $2^{2^n}$ . The number of valid QPs are actually bounded by the size of training data. Since QPs must be associated with training instances sharing at least one identical attribute value with the query, the computational cost can be significantly reduced. Second, the rules we introduce in Section 3.4 can greatly accelerate the discovery of useful QPs. Once a QP is accepted or rejected, a number of QPs will be removed from further consideration.

Furthermore, the computational cost of the generation process can be greatly reduced by constraining the size of  $\mathbf{u}$ , or by tightening the thresholds in Equation 3.2 and 3.3, or by imposing a limit on the minimum cardinality of QPs to be analyzed. In addition, the generation process is only employed when no useful QPs are found via direct exploration.

## 5 Empirical Investigation of QPAL

To investigate the learning process of QPAL, we synthesized two learning problems. They have the same attribute set  $\mathbf{F} = \{F_1, F_2, F_3, F_4\}$  where each  $F_i$  is defined on the value domain  $\{1, 2, 3, 4\}$ . The class label is binary taking on values of 0 or 1. We defined the class concept to be learned for the first synthetic problem as  $((F_1 = F_2) \vee (F_3 = F_4))$ . The class concept to be learned for the second synthetic problem is  $((F_1 = F_2) \wedge (F_3 = F_4))$ . We generated two synthetic data sets, one for each synthetic problem. Each data set consists of 256 instances, which covers all possible instantiations of  $\mathbf{F}$ , from  $(1,1,1,1)$  to  $(4,4,4,4)$ . In the data set of the first synthetic problem, 112 instances, i.e., 7/16 in ratio, are labeled as “positive” and 144 “negative” instances, i.e., 9/16 in ratio. For example,  $(1,1,2,3)$  and  $(2,2,3,3)$  are labeled as positive while  $(4,3,1,2)$  and  $(1,2,2,3)$  are labeled as negative. In the data set of

the second synthetic problem, there are 16 positive instances, i.e., 1/16 in ratio, and 240 negative ones, i.e., 15/16 in ratio. For example,  $(1,1,4,4)$  and  $(2,2,3,3)$  are labeled as positive while  $(3,3,1,4)$  and  $(1,2,2,3)$  are labeled as negative.

One characteristic of the synthetic learning problems is that the class labels are evenly distributed over different values for different attributes, as depicted in Figure 4. For instance,  $P(F_i|C)$  remains constant for any value of any attribute. There is no difference between the information gain of any attribute values. Hence most classical learning models cannot handle this problem effectively. we have tested QPAL

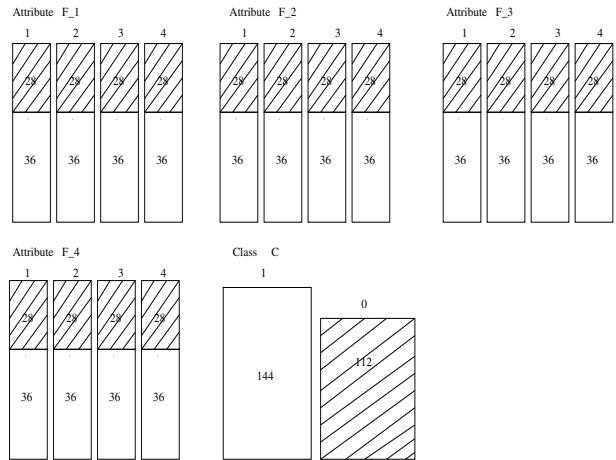


Figure 4: The class distribution of each attribute for the data set of the first synthetic problem

as well as different classical learning models, namely, Bayesian Network, Naive Bayes, kNN, Support Vector Machine (SVM), Lazy Bayesian Rule (LBR) [20], and Decision Tree (J48). For these classical learning models, we employed Weka-3-2-6 machine learning software package [19] freely available on the Web (<http://www.cs.waikato.ac.nz/ml/weka/>). Default parameter setting was used for each classical learning model. The leave-one-out method was used for measuring the classification performance.

The results in Table 1 show that most classical learning models we tested cannot handle the first problem effectively. Only QPAL and Bayesian Network can achieve perfect (100%) accuracy. For other models, Decision Tree (J48) performs better. However, it can be observed that the Decision Tree algorithm can only output an empty decision tree. The reason is that Decision Tree uses a greedy manner to chose an attribute as a node once at a time. When there are no difference between the information gain of different attributes, the partitioning process of the attribute space cannot con-

Problem	QPAL	Bayesian Network	J48	kNN	LBR	Naive Bayes	SVM
1st	100	100	81.22	62.5	62.5	56.25	44.75
2nd	100	93.75	93.75	93.75	93.75	93.75	93.75

Table 1: Classification performance, measured by accuracy percentage, of different learning algorithms for synthetic learning problems.

tinue or cannot produce a reliable model. After investigating the output of Naive Bayes, we found that Naive Bayes can only conduct majority voting for this problem. As for kNN and Lazy Bayesian Rule (LBR), only the instances satisfying  $((F_1 = F_2) \wedge (F_3 = F_4))$  or  $((F_1 \neq F_2) \wedge (F_3 \neq F_4))$  can be classified correctly. These two cases occupy 1/16 and 9/16 of the whole instances respectively and they precisely sum up to 62.5%. It can be observed that the Euclidean distance between two instances cannot be treated as a reliable measure of their similarity in terms of the target class concept.

Consider the scenario of determining the class label of the query instance (2,2,3,2) in the first learning problem. The QPs are organized in a lattice as shown in Figure 5. The associated training instances are also shown. The non-shaded dot denotes a negative instance whereas the shaded dot denotes a positive instance. It can be observed that the useful pattern (2,2,?,?) is in the middle of the lattice, whose associated 15 training instances have the same class label. QPAL first investigates valid QPs with the largest cardinality 3, i.e.,  $\{(2,2,3,?), (2,2,?,2), (2,?,3,2), (?,2,3,2)\}$ . They are divided into two groups with different majority classes. After the generation, the group of positive class produces a useful new QP (2, 2, ?, ?) whereas the group of negative class produces none. Therefore, QPAL outputs the QP (2, 2, ?, ?) as the explanation of predicting the (2,2,3,2) as “positive”. For kNN algorithm, the nearest neighbors on the top of the lattice cannot offer reliable prediction because of the existence of “irrelevant nearest neighbors”. Although they are among the nearest training instances to the query instance, their associated QPs may not be useful for the prediction. For Decision Tree, the bottom QPs in Figure 5 represent single attribute values. They have no difference in terms of information gain. So the construction of a decision tree is not feasible. Figure 5 illustrates that QPAL benefits from the effective exploration for QPs of different attribute dimensions, which may not be fully utilized by other common learning models.

The output QPs demonstrates its good interpretability of the classification decision. Returning to the example above, If the instance to be classified is (2,2,3,2), the discovered QPs  $\mathbf{r}$  becomes  $\{(2,2,?,?), (?,?,3,?), (?,?,?,2)\}$ . The final class prediction is “positive”. If the instance to be classified is (1,1,2,2),

then  $\mathbf{r}$  is  $\{(1,1,?,?), (?,?,2,2)\}$ . The final class prediction is positive. It can be observed that QPAL appropriately captures useful QPs rather than handling each attribute separately or considering the complete attribute set as a whole. These selected QPs can provide a more comprehensible explanation for the classification decision.

For the second synthetic problem, the difference is that the positive class in the second synthetic problem is only a tiny fraction of the whole data set, which introduces challenges for the learning task. The results in Table 1 reveal that all classical learning models cannot achieve perfect performance. The reason is that the number of training instances is too small for the positive class. Hence all instances are predicted to be the majority class which occupies 93.75% of the data set. However, QPAL can perform the learning task in a perfect manner. The generation process successfully filters out the irrelevant QPs and selects the useful ones. It shows that QPAL has a good tolerance for the data sparseness problem.

Since QPAL focuses on the local QPs of the query instance, it can effectively discover useful patterns for prediction, as shown in this investigation. This empirical investigation suggests that QPAL is good at discovering some useful knowledge.

## 6 Handling Continuous Attribute Values and Missing Values

QPAL can be extended to handle continuous attribute values easily. A simple technique is to employ discretization methods [7] before QPAL processes the data. However, this will make the whole learning framework not a pure lazy learning because the discretization needs to preprocess the data.

An alternative method is to handle the continuous attribute values in the QPAL model. In the implementation of QPAL, each training instance is transformed into a binary string where each bit is associated with an attribute value. If a training instance shares the same value as the query on a particular attribute, then the corresponding bit is set to “1”, otherwise it is set to “0”. Thus the query can be represented by a binary string of all “1”. Such a binary string is in fact another format of QP. Two or more training instances may have the same binary string. Hence the task of discovering

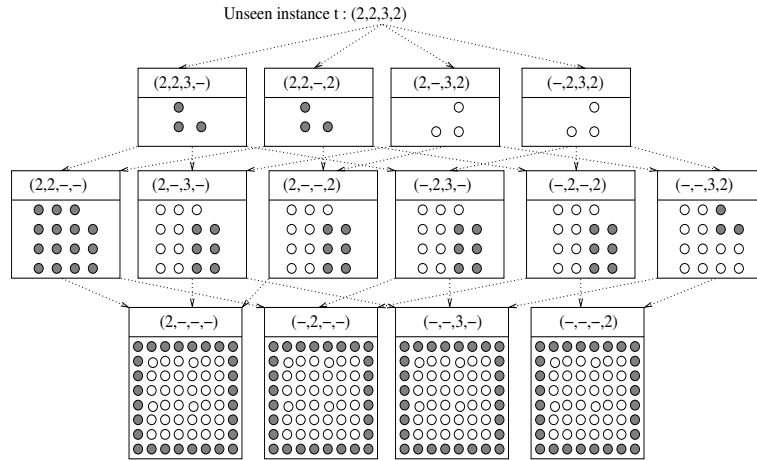


Figure 5: The lattice of QPs and associated training instances given a query instance (2,2,3,2)

QPs can be accomplished by a series of Boolean operations which can be computed efficiently.

We can integrate the handling of continuous attributes values into this transformation module. The neighborhood concept is used to replace the concept of equality. In other words, if a training instance falls into the neighborhood of the query at a given continuous attribute. Then the corresponding bit is set to 1, otherwise 0. The neighborhood is defined as  $x$  training instances with the closest values at the given attribute. The value  $x$  is usually specified as one half of the minimal class frequency among the training data. For computation, the training data only needs to be scanned only one time. A priority queue with the size  $x$  will remember the closest  $x$  training instances for each attribute. The transformation to binary strings is then guided by these priority queues.

In real-world problems, the instances usually contain missing attribute values. QPAL can handle these missing values very easily without any additional processing. The reason is that lazy learning makes each query instance has its independent learning process. Hence the attributes with missing values are just removed from the learning process for the query as if the problem is defined only on the attributes with known values.

## 7 Experiments and Discussions

**7.1 Benchmark Data Sets** We have conducted extensive experiments on 18 benchmark data sets from the UCI repository of machine learning database [3] to evaluate our QPAL framework. These data sets, collected from different real-world problems in various domains, are shown in Table 2.

We partitioned each data set into 10 even portions

Data Set	Number of Attributes	Number of Classes	Number of Instances
Annealing	38	5	898
Breast(W)	9	9	699
Contact(L)	4	3	24
Credit(A)	15	2	690
Glass	9	7	214
Heart(C)	13	5	303
Hepatitis	19	2	155
Ionosphere	34	2	351
Iris	4	3	150
Kr-vs-Kp	36	2	3,196
Labor	16	2	57
Letter	16	26	20,000
Lymph	18	4	148
Mushroom	22	2	8,124
Sonar	60	2	208
Soybean	35	19	683
Vowel	13	11	990
Zoo	17	7	101

Table 2: Description of 18 benchmark data sets

and then conducted 10-fold cross-validation. The performance is measured by the accuracy which is defined as the percentage of the number of correctly classified instances over the total number of testing instances. In these experiments, we have also investigated the performance of Naive Bayes, SVM, kNN, Lazy Bayesian Rule (LBR) and Decision Tree (J48) provided by Weka-3-2-6 machine learning software package. All these models except for kNN use default settings during the entire evaluation process. For kNN, we have conducted runs for  $k = 1, 5$  and 10. We reported the results for  $k = 5$  because it achieves the best average classification accuracy among different  $k$ .

The results of lazy learning methods on different data sets are depicted in Table 3. kNN achieves good performance on data sets such as Annealing, Credit, Letter, Vowel, Sonar, and Zoo. This observation suggests that kNN has some advantages for large data sets or data sets with large attribute dimensions. Our QPAL shares these advantages. Compared with existing lazy learning approaches, on most of the data sets, QPAL achieves good performance. For some data sets such as Contact, Glass, Hepatitis, Labor, and Sonar, QPAL outperforms all other classical lazy classifiers. The results also show that QPAL excels at handling data sets with incomplete data. The reason is due to the flexible framework of QPAL. It should be noted that despite its simplicity, lazy learning method kNN can achieve outstanding performances on some data sets when  $k = 1$ . For example, on data sets of Annealing, Letter, Sonar and Vowel, 1NN obtains 99.11%, 96.06%, 86.57% and 99.29% respectively. It significantly outperforms existing learning methods. However, 1NN suffers on other data sets due to its simplicity. QPAL can obtain comparable performance of 1NN on the above data sets, with less negative effect on other data sets. On average, the classification accuracy of QPAL on these 18 benchmark data sets is 90.6%, which shows improvement over other lazy learning models.

We also compared the performance of QPAL and some existing eager learning methods, i.e., SVM and J48. The average accuracy of each classifier is shown in Table 4. According to our experiment results, lazy learning methods have advantages over eager learning methods on some learning problems. On these 18 benchmark data sets, QPAL achieves the best average score 90.6%.

**7.2 Spam Email Filtering Problem** Spam emails have become an increasingly important concern. The demand for spam email filtering rises rapidly. Black list or white list are usually used to filter out spam emails. Various intelligent methods have also been proposed to

Data Set	QPAL	kNN	NaiveBayes	LBR
Annealing	97.1	97.1	86.5	<b>97.2</b>
Breast(W)	96.7	96.4	96.0	<b>97.4</b>
Contact(L)	<b>87.5</b>	66.7	75.0	70.83
Credit(A)	84.9	<b>85.8</b>	77.7	85.6
Glass	<b>75.0</b>	67.8	48.5	65.9
Heart(C)	82.6	81.8	<b>84.5</b>	82.4
Hepatitis	<b>90.0</b>	85.8	83.8	85.3
Ionosphere	90.3	84.9	82.4	<b>90.9</b>
Iris	<b>96.0</b>	95.3	96.0	94.7
Kr-vs-Kp	96.5	96.0	87.6	<b>97.3</b>
Labor	<b>96.5</b>	86.0	94.7	89.3
Letter	86.8	<b>95.5</b>	64.2	60.3
Lymph	<b>83.8</b>	83.8	83.8	81.5
Mushroom	99.8	<b>100.0</b>	95.8	99.9
Sonar	<b>87.0</b>	84.6	65.9	74.6
Soybean	92.1	90.2	92.8	<b>93.0</b>
Vowel	91.5	<b>93.7</b>	61.4	86.5
Zoo	<b>96.2</b>	95.1	95.2	95.8
Average	<b>90.6</b>	88.1	81.8	86.0

Table 3: Classification performance of QPAL and other lazy learning methods.

	QPAL	J48	SVM
Average	<b>90.6</b>	86.3	87.4

Table 4: Classification performance comparison of QPAL and eager classifiers.

tackle this problem. We investigate applying our QPAL learning framework to spam email filtering.

We used the data set from the Data Mining Cup 2003 (DMC2003) contest, whose task is to identify spam emails by means of data mining techniques. The training data set has 8,000 instances and the testing data set has 11,177 instances. Among those testing instances there are 4,374 spam emails and 6,803 non-spam emails. Each instance representing an email is recorded by 834 attributes. The attributes represent different tactics for identifying spam emails, for example, whether the email contains unsafe java script, or contains some keywords, or has a particular characteristic style. A complete description of all the attributes can be found in Open Source Project SpamAssassin (see <http://spamassassin.org>). The evaluation metric is measured by the filter-out rate which is defined as the number of emails being filtered out as spam emails divided by the number of emails being processed.

QPAL is employed in a pure lazy learning manner without any customization to this data set. The empirical results show that QPAL can obtain prominent performance. In particular, QPAL performs very well in preserving non-spam emails while keeping a high filter-out rate for spam emails. This characteristic is very important in business applications because many users

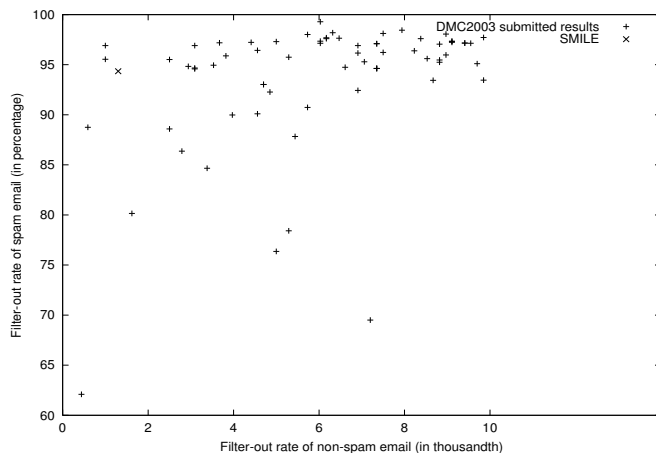


Figure 6: Performance of filtering spam emails

cannot bear a high risk of losing useful emails. The result is shown in Figure 6 where the submitted results of DMC2003 are also shown for comparison. The X-axis in the graph represents the ratio of non-spam emails being incorrectly filtered out. The Y-axis represents the ratio of spam emails being correctly filtered out. The performance of each submitted group is expressed by the symbol “+”. The performance of QPAL is indicated by the symbol “x”. The closer is the symbol to the top-left corner, the better is the performance.

It can be observed that QPAL is able to obtain a favorable tradeoff between the filtering of spam emails and the protection of normal emails. QPAL filters out more than 94% spam emails, which is far better than the filtering performance by many ISP which ranges from 60% to 80%. Even when compared with the submitted results of DMC2003, the performance of QPAL is comparable in terms of spam email filter-out rate. Note that most of the submitted results are based on customized non-lazy learning models while QPAL conducts filtering in a lazy learning manner. In a lazy learning manner, the user need not to train the learning model frequently as spam emails change their patterns from time to time to avoid being filtered out. Specifically, the required task for QPAL only includes adding some typical spam emails into the training set. This property can help to cope with those intelligent spam email creators and rapidly changing spam emails.

As for the filter-out rate of non-spam email, QPAL shows a superior performance where only 0.13% of the non-spam emails are incorrectly filtered out. Meanwhile, most of the submitted results for DMC2003 cannot reach that level. People typically favor the filters which can protect more useful emails rather than the filters which can delete more spam emails. It is also

common that there are a lot of spam emails disguised as a non-spam email, and they cannot be filtered out without raising the risk of losing non-spam emails. As a result, QPAL is indeed very competitive for spam email filtering.

## 8 Conclusions and Future Work

This paper proposes a novel query-driven lazy algorithm called QPAL, which attempts to discover local QPs for classifying the query instance. By customizing the learning process to the query and searching in an innovative way, it can avoid the horizon effect which commonly exists in many eager learning algorithms. We show that QPAL can guarantee the discovery of all QPs with perfect (100%) expected accuracy in polynomial time. To improve the efficiency of lazy learning on large data sets, QPAL can guarantee a low probability of rejecting any QPs with an expected accuracy higher than a specified value. The experimental results on a real world problem and benchmark data sets also demonstrate that our learning algorithm achieves prominent learning performance.

A promising future direction is to explore more sophisticated methods for assessing QPs. The class distributions on parent and children QPs can also be considered in making decisions on a particular QP as reinforcements. Moreover, some non-lazy learning techniques can be introduced into the learning framework, for example, the useful QPs selected for a given query can be stored for future usage if they meet some requirements. Another direction is to apply QPAL in online active learning. The rationale is that QPAL can organize the relevant QPs in a lattice structure and hence provide potentially useful instances in a hierarchical way to accelerate the learning with little user intervention.

## References

- [1] Aha, D.: Editorial. *Artificial Intelligence Review, Special Issue on Lazy Learning*, 11:7–10, February 1997.
- [2] Aha, D.: Relating relational learning algorithms. In S. Muggleton, editor, *Inductive Logic Programming*, pages 233–260. Academic Press, 1992.
- [3] Blake, C., Keogh, E., and Merz, C.: UCI repository of machine learning databases. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [4] Chang, E., and Li, B.: MEGA— the maximizing expected generalization algorithm for learning complex query concepts. *ACM Transactions on Information Systems (TOIS)*, 21(4):347–382, 2003.
- [5] Dasarathy, B.: *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, 1991.

- [6] Dasarathy, B.: Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517, March 1994.
- [7] Fayyad, U., and Irani, K.: Multi-interval discretization of continuous-valued attributes as preprocessing for machine learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.
- [8] Friedman, J.: Flexible metric nearest neighbor classification. Technical report, Stanford University, November 1994.
- [9] Han, Y., and Lam, W.: Lazy learning by scanning memory image lattice. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 447–451, 2004.
- [10] Lam, W., and Han, Y.: Automatic textual document categorization based on generalized instance sets and a metamodel. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):628–633, 2003.
- [11] Li, J., Dong, G., Ramamohanarao, K. and Wong, L.: DeEPs: A new instance-based discovery and classification system. *Machine Learning*, 54:99–124, 2004.
- [12] Li, W., Han, J., and Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 369–376, 2001.
- [13] Liu, B., Hsu, W., and Ma, Y.: Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 80–86, 1998.
- [14] Matheus, J.: Adding Domain Knowledge to SBL Through Feature Construction. In *Proceedings of the National Conference on Artificial Intelligence*, pages 803–808, 1990.
- [15] Mehta, M., Rissanen, J., and Agrawal, R.: MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 216–221, 1995.
- [16] Mitchell, T.: *Machine Learning* McGraw Hill, pages 24–25, 1997.
- [17] Seshu, R.: Solving the parity problem. In *Proceedings of the Fourth European Working Session on Learning*, pages 263–271, 1989.
- [18] Utgoff, P. and Brodley, C.: An incremental method for finding multivariate splits for decision trees. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 58–65. University of Texas, Austin, Texas, 1990.
- [19] Witten, I. and Frank, E.: *Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.
- [20] Zheng, Z., Geoffrey, I., and Kai M.: Lazy Bayesian rules: a lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In *Proceedings of 16th International Conf. on Machine Learning*, pages 493–502, Morgan Kaufmann, San Francisco, CA,