# Concept Lattice based Composite Classifiers for High Predictability

Zhipeng XIE[1] Wynne HSU[1] Zongtian LIU[2] Mong Li LEE[1]

[1]School of Computing
National University of Singapore
Lower Kent Ridge Road, Singapore, 119260
{xiezp, whsu, leeml}@comp.nus.edu.sg

[2]School of Computing
Shanghai University of China
Shanghai, P.R.China, 200072
ztliu@mail.shu.edu.cn

**ABSTRACT**

Concept lattice model, the core structure in Formal Concept Analysis, has been successfully applied in software engineering and knowledge discovery. In this paper, we integrate the simple base classifier (Naïve Bayes or Nearest Neighbor) into each node of the concept lattice to form a new composite classifier. We develop two new classification systems, CLNB and CLNN, that employ efficient constraints to search for interesting patterns and voting strategy to classify a new object. CLNB integrates the Naïve Bayes base classifier into concept nodes while CLNN incorporates the Nearest Neighbor base classifier into concept nodes. Experimental results indicate that these two composite classifiers greatly improve the accuracy of their corresponding base classifier. In addition, CLNB even outperforms three other state-of-art classification methods, NBTree, CBA and C4.5 Rules.

## 1 Introduction

Classification is a kernel task in many data mining applications. To deal with this task, various methods have been developed, such as decision rule, Naïve-Bayes, decision

tree, nearest neighbor and neutral network. Different classification methods have different decision planes, and are appropriate for different situations. There is no one single method that is the best for all situations. As a result, in recent years, researchers begin to focus their efforts towards improving predictive accuracy through the integration of a number of different classifiers. Naïve Bayes tree learner NBTree [Kohavi96] and lazy Bayesian rule learning algorithm LBR [Zheng00] are examples of these recent efforts.

The main thrust of NBTree and LBR is in the use of a contextual rule for classification instead of the normal classification rule. In machine learning, a classification rule takes the following form:

**If** $P_1 \grave{U} P_2 \grave{U} \ldots \grave{U} P_r$ **then** $C_j$,

where each $P_i$ $(1 £ i £ r)$ is a descriptor (or attribute-value pair in relational table) of object, and $C_j$ is a class label. Such a rule means that an object will be classified as $C_j$ if it satisfies all the descriptors, $P_i (1 £ i £ r)$. NBTree and LBR generalize the above classification rule to define contextual rule:

**If** $P_1 \grave{U} P_2 \grave{U} \ldots \grave{U} P_r$ **then** use $CLS_i$,

where $CLS_i$ is a classifier called base classifier. Such a contextual rule means that $CLS_i$ can used to classify an object if the object satisfies all the descriptors. By thinking $C_j$ as a classifier that classifies any object as $C_j$, it is clear that normal classification rule is just a special case of a contextual rule.

Kohave et. al. presented Naïve Bayes tree learner, called NBTree [Kohavi96], that combines naïve bayesian classification and decision tree learning. It uses a tree structure to split the instance space into sub-spaces defined by the path of the tree. A naïve Bayesian classifier is generated in each sub-space. Each leaf of the naïve Bayesian tree contains a local naïve Bayesian classifier. As in many other learning algorithms that are based on tree structure, NBTree suffers from the small disjunct problem. To tackle this problem, Zheng Z., et. al. [Zheng00] applied lazy learning techniques to Bayesian tree induction and presented the resulting lazy Bayesian rule learning algorithm LBR. LBR constructs a bayesian rule specifically for an input test example and uses this rule to predict the class label of the example.

Due to the flexibility of allowing different classifiers for different sub-instances of the data space, both NBTree and LBR have achieved better accuracy than C4.5 and naïve Bayes classifiers. However, this improvement on accuracy is limited by their principle of local search. A local maxima of accuracy will stop further search for interesting and useful rules.

In this paper, we propose a framework that employs a more expressive structure, the concept lattice, to avoid local maxima. The concept lattice structure enables one to exhaustively extract all the bayesian rules. Here, strategies for pruning the concept lattice are very important for efficient learning. Three types of constraints are presented and integrated into the top-down construction procedure to prune the lattice structure. In addition, the proposed framework also works with any simple classification method so long as an efficient technique for accuracy estimation for that classification method is available.

We would like to highlight the following important fact: Given a test example, there will be multiple rules that are matched and the corresponding classifiers get activated. A majority voting strategy is then applied to classify the test example. Such voting strategy is similar to the multi-classifier techniques such as Bagging [Breiman96] and Boosting [Freund96] in that they all use multiple classifiers to vote on decision. However, they are also different in that, for our framework, only the activated classifiers (whether a classifier is activated or not is determined by the input test example) can take part in the voting, but for Bagging and Boosting, all the classifiers will be used to vote regardless of what the input test example is like. The reason for such difference is because each classifier (except the root classifier) in our algorithm framework is induced on a subset of training examples that share some common features, while each classifier in Bagging or Boosting is learnt on a sample of training set based on randomly sampling.

The paper is organized as follows. Section 2 provides some background information on two simple classification methods, namely Naïve Bayes and Nearest Neighbor. A short discussion is made on the accuracy estimation techniques used in both methods. Section 3 presents our algorithm framework for embedding simple classifier into concept node of concept lattice. Three types of constraints for pruning the concept

lattice structure have been proposed. A post-processing pruning strategy is designed that is based on chi-square test. The majority voting strategy for classifying a new object is also presented. Section 4 shows the realization of our framework in the form of two new classification learning systems, called CLNB (Concept Lattice Naïve Bayes classification learning system) and CLNN (Concept Lattice Nearest Neighbour classification learning system). Experimental results on 26 datasets shows that both systems have shown great improvement in terms of the predictive accuracy over their corresponding base classifiers. In addition, *CLNB* even outperforms several state-of-art classifiers.

## 2 Simple Classifiers and Accuracy Estimation

For simplicity, we assume a dataset to be a relational data table with only nominal attributes, which consists of the descriptions of $n$ objects in the form of tuples. These $N$ objects have been classified into $q$ known classes, $C_1$, $C_2$, ..., $C_q$. Each object in the database is described by $m$ distinct attributes, $Attr_1$, ..., $Attr_i$, ..., $Attr_m$, so that in an instantiation of object description, an attribute $Attr_i$ takes on the value $v_{ij} \in domain(Attr_i)$. Let $U$ denote the set of objects and $A$ denote the set of attributes. Various kinds of classification method have been developed to induce classifiers on a dataset, and the classifier can be thought as a function assigning a class label to a newly-seen object.

Among the many existing classification methods, Naïve Bayes [Duda73] and Nearest Neighbor [Dasarathy91] are two simplest but efficient classification techniques and have been studied widely. They will be used to induce the base classifers to be incorporated into the concept nodes. Accuracy estimation is used to approximate classifier's performance. Accuracy estimation techniques will be used for the base classifiers, with efficiency taken into consideration.

### 2.1 Naïve Bayes Classifier

Naïve Bayes, as a typical eager learning algorithm, is simple and computational efficient. In spite of its simplicity, it has proved to be a surprisingly successful method,

and has outperformed much more complicated methods in many application domains. In addition, it is also robust to noise and irrelevant attributes and is easy to understand. Naïve Bayes is based on the assumption that attributes are conditionally mutually independent given the class label. Formally, the probability of a class label value $C_i$ for an unlabelled instance $V=(a_1, \ldots, a_m)$ consisting of $m$ attribute values is given by $P(C_i|V)=P(C_i)\cdot P(V|C_i)/P(V)$. According to the assumption, it holds that $P(V|C_i)=\prod_{k=1}^{m} P(a_k \mid C_i)$. The class label, with the highest probability given the instance $V$, is used as the predicted class. Note that we do not need to compute the value of $P(V)$. This is because $P(V)$ is a constant for a given $V$.

For Naïve Bayes, typically leave-one-out strategy is used to obtain the accuracy on a training set. This strategy can be implemented efficiently with a time complexity that is linear to the number of objects, number of attributes, and number of label values [kohavi96].

### 2.2 Nearest Neighbor Classifier

The $k$-Nearest Neighbor classification, also called memory-based or case-based learning, is lazy. It finds the $k$ nearest neighbors of a unlabelled instance $V$ in the training set according to some metric or "distance" function, and then predicts the class label of $V$ as the class that occurs the most frequently among all the $k$ neighbors. Various distance metrics have been developed for nearest neighbor algorithm, among which the probability-based metrics are the most promising [Blanzieri99]. In this paper, SF2 metric is used [Short81]. It relies on probabilistic consideration and was later generalized to multi-class by Myles and Hand [Myles90]. For any two instance $V_1=\{a_{11}, a_{12}, \ldots, a_{1m}\}$ and $V_2=\{a_{21}, a_{22}, \ldots, a_{2m}\}$, the SF2 distance between them is defined as follows:

$$sf2(V_1, V_2)=\sum_{i=1}^{|C|} \left| p(C_i \mid V_1) - p(C_i \mid V_2) \right|$$

Through applying Bayes theorem and using the same independent assumption as Naïve Bayes, we have

$$sf2(V_1, V_2) = \sum_{i=1}^{||C||} \left| \frac{\prod_{j=1}^{m} p(a_{1j} \mid C_i) p(C_i)}{\sum_{k=1}^{||C||} \prod_{j=1}^{m} p(a_{1j} \mid C_k) p(C_k)} - \frac{\prod_{j=1}^{m} p(a_{2j} \mid C_i) p(C_i)}{\sum_{k=1}^{||C||} \prod_{j=1}^{m} p(a_{2j} \mid C_k) p(C_k)} \right|,$$

where $||C||$ represents the number of classes.

After the distance metric is defined, 1-Nearest neighbor classification procedure can be easily applied to predict the class of a given unknown instance through assigning it to the class of the nearest one with respect to the metric defined above.

Since the distance between two instances depends on the estimates of probabilities and the count information (which needs to be updated with the removal of instances), it is not efficient to implement the leave-one-out accuracy estimation strategy. To solve this problem, an approximate solution is adopted whereby we just compute all the pair-wise distances of the instances with the estimates of probabilities without updating the count information.

### 2.3 Probability Estimation

In implementing the above classifiers, techniques should be developed to estimate $p(a|C_i)$ and $p(C_i)$. The simplest probability estimates are the occurrence frequencies, which is used to estimate $p(C_i)$, that is, $p(C_i)=N(C_i)/N$, where $N$ is the number of the training examples, and $N(C_i)$ is the number of the training examples with class $C_i$. As for estimating the conditional probability $p(a_k|C_i)$, we adopt the Laplace-corrected estimate, which leads to $p(a_k|C_i)=(N(a_k, C_i)+f)/(N(C_i)+fn_j)$, where $n_j$ is the number of values of the $k$-th attribute, and $f$ is a multipicative factor (default value as $1/N$) [Domingos97].

## 3 Contextual and Composite Classifiers

Ever since R. Wille [Wille82] proposed the theory of formal concept analysis in the early 1980s, concept lattice has been widely and successfully used in many fields including data mining and machine learning. In knowledge discovery, concept lattice can be constructed from relational data set, from which various kinds of rules, such as implication rules [Godin94], association rules [Pasquier99] and classification rules

[Mephu94], can be extracted. Our paper focuses on classification through the incorporation of base classifiers into concept nodes. We present the details in the following subsections.

### 3.1 Contextual classifier: Formal concept meets base classifier

In formal concept analysis, formal context is a triple $K=(U, D, R)$, where $U$ is a set of objects, $D$ is a set of descriptors, and $R$ is a binary relation between $U$ and $D$. Two functions, $f$ and $g$, are defined in $K$ as: $\forall O_1 \subseteq U: f(O_1)=\{d \in D \mid \forall x \in O_1 (x\ R\ d)\}$ and $\forall D_1 \subseteq D: g(D_1)=\{x \in U \mid \forall d \in D_1 (x\ R\ d)\}$. Any pair $H_1=(O_1, D_1)$ is called a formal concept in $K$, if it satisfies $D_1=f(O_1)$ and $O_1=g(D_1)$, where $O_1$ is called as the extent of $H_1$, and $D_1$ the intent. We also use $Intent(H_1)$ and $Extent(H_1)$ to represent the intent and extent of concept $H_1$. Clearly, each concept represents a subspace of the training instance space. These subspaces overlap each other, which is different from the decision tree whose leaf nodes form a partition of instance space. On the set of concepts, hierarchical order relation $\leq$ is defined: If $H_1$ and $H_2$ are two formal concepts in $K$, $H_1$ is called a subconcept of $H_2$, denoted as $H_1 \leq H_2$, provided that $Intent(H_1) \supseteq Intent(H_2)$ (which is equivalent to $Extent(H_1) \subseteq Extent(H_2)$). In this case, $H_2$ is a supconcept of $H_1$. For any two different concepts $H_1$ and $H_2$, $H_1$ is called a child of $H_2$ ($H_2$ is called a parent of $H_1$), if $H_1 \leq H_2$ and there is no other concept $H_3$ satisfying $H_1 \leq H_3 \leq H_2$. The set of all the concepts in $K$ and the hierarchical order defined on it form the concept lattice in $K$. A dataset with object set $U$ and attribute set $A$ can be transformed into a formal context $K=(U, D, R)$ by setting $D=\{(Attr_i, v_{ij}) \mid Attr_i \in A, v_{ij} \in domain(Attr_i)\}$, and $xR(Attr_i, v_{ij})$ iff $Attr_i(x)=v_{ij}$ for any $x \in U$.

For simplicity of expression, a contextual rule $r$ will take the form of $r: H \Rightarrow CLS$, where $H$ is a formal concept, and $CLS$ is the base classifier induced on the extent of $H$. It is easy to covert it to the form introduced in Section 1:

**If** $\neg Intent(H)$ **then** $CLS$.

Clearly, the training set of this contextual classifier is $Extent(H)$, so the accuracy estimation method can be applied directly, and $acc(r)$ is used to denote the estimated accuracy of $r: H \Rightarrow CLS$.

For a contextual classifier $r_1$: $H_1$® $CLS_1$ and a object $x$, $r_1$ is said to be activated by $x$ if $Intent(H_1)$ Í $x$. When $r_1$ is activated by $x$, $CLS_1$ can be used to predict the class of object $x$, with the predicted class denoted by $r_1(x)$.

However, the number of concept nodes is very large even for a medium-size data set, and given the one-to-one correspondence relationship between concept nodes and contextual classifiers, it is not practical to calculate the entire set of all the contextual classifiers. So, effective constraints must be adopted to restrict the search space. This is discussed in the next subsection.

### 3.2 Using constraints to search contextual classifiers

Given a concept $H_1=(O_1, D_1)$ and any feature $d$ Î $D$-$D_1$, concept $H_2=(g(f(D_1 È \{d\})), f(D_1 È \{d\}))$ is called to be a direct subconcept of $H_1$. Any child $H_2$ of node $H_1$ is certainly a direct subconcept of $H_1$, while a direct subconcept of $H_1$ is not necessarily a child of it.

A set of contextual classifiers is called a composite classifier. A composite classifier, *RuleSet*, is reduced if it satisfies the following three types of constraints:

- Support Constraints: Two threshold values are used in this constraint. On the one hand, each contextual rule $r_1$: $H_1$® $CLS_1$ should satisfy $||Extent(H_1)|| ³ a ´ ||U||$, where $\alpha$ has a default value of 0.05. On the other hand, for any two contextual classifiers $r_1$: $H_1$® $CLS_1$ and $r_2$: $H_2$® $CLS_2$ in *RuleSet*, where $H_1$ is an ancestor of $H_2$, the size of $Extent(H_2)$ should be large than $\sigma/(1 - acc(H_1® CLS_1))$, the default value of $\sigma$ is 3. Both the support constraints are used to guarantee the generalization ability of the learnt model.

- Accuracy Constraint: For any two contextual classifiers $r_1$: $H_1$® $CLS_1$ and $r_2$: $H_2$® $CLS_2$ in *RuleSet*, where $H_1$ is an ancestor of $H_2$, the estimated accuracies of $r_1$ and $r_2$ should satisfy $acc(r_2) > acc(r_1) + d*log(||Extent(C_1)||/||Extent(C_2)||)$. Default value of $d$ is set as 0 in our experiment. The smaller the value of $d$ is, the larger is the search space to be explored.

- Reject Constraint: For a contextual classifier $r_1$: $H_1 \rightarrow CLS_1$ in *RuleSet*, if there is another contextual classifier $r_2$: $H_2 \rightarrow CLS_2$ in *RuleSet* that satisfies $Intent(H_2) \supset Intent(H_1)$, then the size of $Extent(H_1)$ should be not larger than $\gamma \times ||Extent(H_2)||$, where $\gamma$ has a default value of 0.9. This constraint is used to prevent the occurrence of contextual rules that are similar to each other.

Based on the definition of direct sub-concept and the three types of constraints, our algorithm, as the pseudo-code listed below, searches for the interesting contextual classifiers in a top-down manner. It begins with the most general node (root node). For each node, our algorithm will compute all its direct subconcepts, if it satisfies all the contraints; otherwise, it will be removed.

```
1  RuleSet_i = Æ, i=0, 1, …, ||D||;
2  Add r_Root: {(U, f(U))} → nil into RuleSet_||f(U)||;
3  FOR i=0 to ||D||−1 DO
4     FOR each contextual rule r: H → nil in RuleSet_||i|| DO
5        IF r satisfies the constraints of support and reject THEN
6           train a base classifier CLS on ||Extent(H)||;
7           update rule r from H → nil to be H → CLS;
8           IF r satisfies the constraint of accuracy THEN
9              SubCon={(g(f(Intent(H)∪{d})), f(Intent(H)∪{d})) | d ∈ D - Intent(H)};
10             FOR each concept H_s ∈ SubCon with ||Extent(H_s)|| ³ γ × ||U|| DO
11                Insert H_s → nil into Rule_||Intent(Hs)||;
12             ENDFOR
13          ELSE
14             Remove r from RuleSet_||i|| and delete it;
15          ENDIF
16       ELSE
17          Remove r from RuleSet_||i|| and delete it;
18       ENDIF
19    ENDFOR
20 ENDFOR
```

### 3.3 Pruning

When searching the contextual classifier space, the constraint used is relatively weak to ensure that most of the interesting patterns can be obtained. Once we have searched through the contextual classifier space, we adopt a stronger pruning strategy to ensure a reliable improvement over the root classifier (which is constructed on the whole training set). Chi-square test, which is based on the comparison of observed frequencies with the expected frequencies, is employed to define the statistical improvement on accuracy.

For any two contextual classifier $r_1$: $H_1$® $cls_1$ and $r_2$: $H_2$® $cls_2$, we can get a 2×2 contingency table:

| | $r_1$: $H_1$® $cls_1$ | $r_2$: $H_2$® $cls_2$ | Row Total |
|---|---|---|---|
| Correctly Classified | $n_{11}$=\|\|*Extent*($H_1$)\|\|´*acc*($r_1$) | $n_{21}$=\|\|*Extent*($H_2$)\|\|´*acc*($r_2$) | $n_{*1}=n_{11}+n_{21}$ |
| Wrongly Classified | $n_{12}$=\|\|*Extent*($H_1$)\|\|´(1 - *acc*($r_1$)) | $n_{22}$=\|\|*Extent*($H_2$)\|\|´(1 - *acc*($r_2$)) | $n_{*2}=n_{12}+n_{22}$ |
| Column Total | $n_{1*}$=\|\|*Extent*($H_1$)\|\| | $n_{2*}$=\|\|*Extent*($H_2$)\|\| | $n_{**}=n_{*1}+n_{*2}$ |

In the table above, $n_{11}$, $n_{12}$, $n_{21}$, and $n_{22}$ are the observed frequencies of the four celles. Let $m_{ij}$ represent the expected frequency corresponding to $n_{ij}$, for $i, j \hat{I} \{1, 2\}$. We have $m_{ij}=n_{i*}´n_{*j}/n_{**}$. For the table above, the Chi-square value is defined as:

$$c^2 = \sum \frac{(n_{ij} - m_{ij})^2}{m_{ij}}.$$

The threshold value 3.84 at the 95% significance level is adopted as the default value for determining the statistical difference between the accuracies of two contextual rules. If there is a statistical difference between the accuracies of $r_1$ and $r_2$ and the estimated accuracy of $r_1$ is higher than that of $r_2$, then we say that $r_1$ is statistically more accurate than $r_2$.

In our current implementation, we use a simple strategy to prune the set of contextual rules:

A contextual rule except the root contextual rule will be pruned, if it is not statistically accurate than the root contextual rule.

### 3.4 Using voting to classify new objects

Given an unseen object *x*, and a composite classifier *R* which is a set of contextual classifier, voting strategy is applied to predict the class. This is accomplished in four steps:

Step 1. Mark all the contextual classifiers activated by the unseen object. Usually, many classifiers will be activated for a given input object. (line1)

Step 2. For any two activated contextual rule $r_1: H_1 \to cls_1$ and $r_2: H_2 \to cls_2$, clear the "activated" status of $r_1$ if $Intent(H_1) \subset Intent(H_2)$. (line 3-5)

Step 3. For any activated contextual rule $r_1$, if there exists another activated contextual rule $r_2$, which is statistically more accurate than $r_1$, then clear the "activated" status of $r_1$. (line 6-8)

Step 4. Perform majority-voting strategy on the input object using the set of activated rules. When tie occurs, the vote of the contextual classifier with the highest accuracy is used as the tie-breaker. (line 10-18)

```
1   R_active={ r: H → CLS | r ∈ R and Intent(H) ⊃ x}
2   FOR each contextual rule r1: H1 → CLS1 ∈ R_active DO
3     IF ∃r2: H2 → CLS2 ∈ R_active (Intent(H1) ⊂ Intent(H2)) THEN
4         remove r1 from R_active;
5     ENDIF
6     IF ∃r2: H2 → CLS2 ∈ R_active (r2 is statistically more accurate than r1) THEN
7         remove r1 from R_active;
8     ENDIF
9   ENDFOR
10  count[i]=0 for each class i;
11  count[r1(x)]++ for each contextual rule r1 ∈ R_active;
12  set major=max_i{count[i]} and decisions={i|count[i]=major}
13  IF ||decisions||=1 THEN
```

14 return *decisions*[0] as the predicted class of *x*;
15 **ELSE**
16 Let $r_1$ be the contextual rule with the maximal accuracy in $\{r \in R_{active} | r(x) \in decisions\}$;
17 return $r_1(x)$ as the predicted class of *x*;
18 **ENDIF**

## 4 Experimental Results

The concept lattice framework has been implemented as a template class using Visual C++ in Win98 system. The concept lattice template takes a base classifier class as its parameter. For our experiments, we generate two new instantiations of the concept lattice framework: one with Naïve Bayes as the base classifier (CLNB), the other with Nearest Neighbor as base classifier (CLNN).

In our experiments, we use the same 26 datasets from UCI Machine Learning Repository [Merz96] as in [Liu98]. The detailed information about these datasets is listed in Table 1. Since the current version of our algorithm can only deal with nominal attribute, the entropy-based discretization algorithm [Fayyad93] is used for preprocessing.

### 4.1 Error-rate comparison

We first compare the accuracy results of the classifiers produced by CLNB and CLNN with those generated by two corresponding base classifiers (Naïve Bayes and Nearest Neighbor), and those generated by three other state-of-art classifiers: NBTree [kohavi96] (a state-of-art of hybrid classifier which improves the accuracy of naïve bayes classifier significantly), CBA [Liu98] (a classifier based on association rules), and C4.5Rules (Release 8). The error rates of the different algorithms on the experimental domains are listed in Table 2. All the error rates are obtained through 10-fold cross validation. We use the same training/test set split for all the classification methods in the experiments.

From the 26 data sets, it is clear that CLNB and CLNN produce more accurate classifiers than Naïve Bayes and Nearest Neighbor respectively. On average, the accuracy increases from 83.7% for Naïve Bayes to 86.4% for *CLNB*, and from 80.5% for Nearest Neighbor to 84.4% for CLNN. The average error rate of CLNB is also 2.4% lower than that of NBTree, 1.6% lower than CBA, and 3.1% lower than C4.5Rules.

| **Dataset** | **No. Attrs** | **No. Classes** | **Size** | **Dataset** | **No Atrs** | **No. Classes** | **Size** |
|---|---|---|---|---|---|---|---|
| anneal | 38 | 6 | 798 | australian | 14 | 2 | 690 |
| auto | 25 | 7 | 205 | breast-w | 10 | 2 | 699 |
| cleve | 13 | 2 | 303 | crx | 15 | 2 | 690 |
| diabetes | 8 | 2 | 768 | german | 20 | 2 | 999 |
| glass | 9 | 7 | 214 | heart | 13 | 2 | 270 |
| hepatitis | 19 | 2 | 155 | horse | 22 | 2 | 368 |
| hypo | 25 | 2 | 3163 | ionosphere | 34 | 2 | 351 |
| iris | 4 | 3 | 150 | labor | 16 | 2 | 57 |
| led7 | 7 | 10 | 3200 | lymph | 18 | 4 | 148 |
| pima | 8 | 2 | 768 | sick | 29 | 2 | 2800 |
| sonar | 60 | 2 | 229 | tic-tac-toe | 9 | 2 | 958 |
| vehicle | 18 | 4 | 846 | waveform | 21 | 3 | 5000 |
| wine | 13 | 3 | 178 | zoo | 16 | 7 | 101 |

Table 1: Datasets used.

| Dataset | NBTree | CBA | C4.5Rules | NB | CLNB | NN | CLNB |
|---|---|---|---|---|---|---|---|
| anneal | 1.0 | 2.1 | 5.2 | 1.6 | 1.4 | 1.3 | 1.1 |
| australian | 14.5 | 14.6 | 15.3 | 14.1 | 14.6 | 20.7 | 15.5 |
| auto | 22.8 | 19.9 | 19.9 | 27.7 | 20.5 | 26.3 | 19.9 |
| breast-w | 2.6 | 3.7 | 5.0 | 2.4 | 3.1 | 3.9 | 3.4 |
| cleve | 19.1 | 17.1 | 21.8 | 18.1 | 16.8 | 23.1 | 16.8 |
| crx | 14.2 | 14.6 | 15.1 | 14.5 | 13.5 | 21.4 | 15.2 |
| diabetes | 24.1 | 25.5 | 25.8 | 24.1 | 23.3 | 33.3 | 33.2 |
| german | 24.5 | 26.5 | 27.7 | 24.5 | 26.5 | 33.5 | 25.5 |
| glass | 28.0 | 26.1 | 31.3 | 28.5 | 26.6 | 32.7 | 31.3 |
| heart | 17.4 | 18.1 | 19.2 | 18.1 | 16.7 | 20.4 | 22.6 |
| hepatitis | 11.7 | 18.9 | 19.4 | 15.6 | 17.5 | 17.5 | 15.6 |
| horse | 18.7 | 17.6 | 17.4 | 21.7 | 18.2 | 26.9 | 19.6 |
| hypo | 1.0 | 1.0 | 0.8 | 1.8 | 1.4 | 1.5 | 1.5 |
| ionosphere | 12.0 | 7.7 | 10.0 | 10.5 | 8.8 | 13.1 | 8.0 |
| iris | 7.3 | 5.3 | 4.7 | 5.3 | 5.3 | 6.0 | 5.3 |
| labor | 12.3 | 13.7 | 20.7 | 5.0 | 5.0 | 5.0 | 5.0 |
| led7 | 26.7 | 28.1 | 26.5 | 26.7 | 26.8 | 53.8 | 53.8 |
| lymph | 17.6 | 22.1 | 16.5 | 19.0 | 21.0 | 25 | 16.9 |
| pima | 24.9 | 27.1 | 24.5 | 24.5 | 27.2 | 29.8 | 30.1 |
| sick | 22.1 | 2.8 | 1.5 | 4.2 | 2.9 | 3.4 | 3.4 |
| sonar | 22.6 | 22.5 | 29.8 | 21.6 | 4.8 | 1.4 | 1.4 |
| tic-tac-toe | 17.0 | 0.4 | 0.6 | 30.1 | 4.1 | 37.9 | 5.2 |
| vehicle | 29.5 | 31 | 27.4 | 40.0 | 27.3 | 43.0 | 30.1 |
| waveform | 16.1 | 20.3 | 21.9 | 19.3 | 15.8 | 20.6 | 17.1 |
| wine | 2.8 | 5.0 | 7.3 | 1.7 | 1.7 | 2.3 | 2.3 |
| zoo | 5.9 | 3.2 | 7.8 | 3.9 | 3.9 | 3.9 | 3.9 |
| **Average** | 16.0 | 15.2 | 16.7 | 16.3 | 13.6 | 19.5 | 15.5 |

Table 2: Error rates (%) of $CL_{NB}$, $CL_{NN}$, NB, NN, C4.5, CBA and NBTree

### 4.2 Computational Requirements

To give an idea of the computational requirements of CLNB and CLNN, four measurements are used:

(1) running time for training (in second);

(2) running time for testing (in second);

(3) number of contextual rules generated before pruning; and

(4) number of contextual rules generated after pruning;

The results of our experiments are listed in Table 3. All the values are averaged over ten folds.

We discovered an important fact from Table 2 and Table 3: for those datasets with more than 50 contextual rules after pruning, the average accuracy improvement of CLNB over NB is 6.3% (on 11 datasets), and the average accuracy improvement of CLNN over NN is 7.9% (on 11 datasets). On the contrary, for the datasets with less than 50 contextual rules after pruning, the average accuracy improvement of CLNB over NB is only 0.03% (on 15 datasets), and the average accuracy improvement of CLNN over NN is 1.12% (on 15 datasets).

Clearly, in our experiments, the accuracy improvement of composite classifier over corresponding base classifier is mainly caused by those data sets with more contextual classifiers generated after pruning.

| Dataset | CLNB | | | | CLNN | | | |
|---|---|---|---|---|---|---|---|---|
| | no. nodes | | running time | | no. nodes | | running time | |
| | bef. pru | aft. pru. | for bld. | for clss. | bef. pru | aft. pru | for bld. | for class. |
| anneal | 15.9 | 3.5 | 1.685 | 0.06 | 16.5 | 1.8 | 2.258 | 0.16 |
| australian | 243 | 75.9 | 1.336 | 0.17 | 867.6 | 557.2 | 3.866 | 0.93 |
| auto | 310.1 | 61.6 | 2.481 | 0.06 | 121.8 | 37.9 | 0.933 | 0.16 |
| breast-w | 14.8 | 6.5 | 0.126 | 0 | 17 | 9.9 | 0.23 | 0 |
| cleve | 202.2 | 44.1 | 0.48 | 0 | 254.5 | 109.2 | 0.532 | 0.22 |
| crx | 280.2 | 86.3 | 1.565 | 0.11 | 712.7 | 456.1 | 3.333 | 0.83 |
| diabetes | 35.7 | 25.6 | 0.159 | 0 | 38.5 | 26.7 | 0.303 | 0.18 |
| german | 591.4 | 392.8 | 4.107 | 0.42 | 1747 | 1482 | 8.825 | 2.3 |
| glass | 24.7 | 5.6 | 0.084 | 0 | 15.5 | 8.6 | 0.07 | 0.06 |
| heart | 107.2 | 25.2 | 0.263 | 0 | 114.2 | 49.7 | 0.224 | 0.05 |
| hepatitis | 178.4 | 43.8 | 0.718 | 0 | 88.1 | 21.7 | 0.355 | 0.06 |
| horse | 1033 | 150.9 | 4.641 | 0.28 | 1868 | 1293.4 | 7.531 | 2.19 |
| hypo | 29.5 | 15.1 | 3.558 | 0.51 | 22.8 | 10.1 | 11.99 | 5.22 |
| ionosphere | 309.7 | 19.6 | 3.631 | 0.05 | 375.7 | 201.6 | 3.082 | 1.09 |
| iris | 2.3 | 1 | 0.006 | 0 | 2.7 | 1.3 | 0.006 | 0 |
| labor | 1 | 1 | 0.01 | 0 | 1 | 1 | 0 | 0 |
| led7 | 142.2 | 119 | 3.114 | 0.28 | 92.9 | 89.4 | 22.98 | 1.56 |
| lymph | 95.4 | 4.1 | 0.378 | 0 | 69.6 | 14.4 | 0.229 | 0 |
| pima | 32.8 | 22.2 | 0.122 | 0.11 | 36.5 | 22.2 | 0.299 | 0.05 |
| sick | 94.3 | 78.3 | 6.484 | 0.58 | 55 | 47.8 | 11.06 | 3.23 |
| sonar | 721.2 | 99.8 | 7.878 | 0.36 | 989.3 | 448.7 | 7.332 | 1.22 |
| tic-tac-toe | 376.9 | 287.8 | 0.713 | 0.18 | 447.2 | 382.1 | 0.957 | 0.16 |
| vehicle | 1413 | 1282 | 12.92 | 1.95 | 1478 | 1369.5 | 12.32 | 2.47 |
| waveform | 154.5 | 134.3 | 10.98 | 1.05 | 459.9 | 332.9 | 24.72 | 5.27 |
| wine | 1 | 1 | 0.005 | 0 | 1 | 1 | 0 | 0 |
| zoo | 3 | 1 | 0.012 | 0 | 1 | 1 | 0 | 0 |
| **Average** | 246.6 | 114.9 | 2.594 | 0.237 | 380.5 | 268.4 | 4.742 | 1.051 |

Table 3: Computational Requirements

## 5 Conclusion and Future Work

In this paper, we have presented an algorithm framework for integrating base classifier into concept node of concept lattice. The algorithm framework is realized in the form of two novel hybrid classification methods, CLNB and CLNN using two simple classification methods, Naïve Bayes and Nearest Neighbor, respectively. Experimental results on 26 datasets indicate that both the hybrid classification methods perform better than their corresponding base dassifiers and CLNB even outperforms state-of-the-art classifiers. Future research work includes looking into different approach for probability estimation, such as the smoothed estimation for parameter used in [Friedman97], to improve the Naïve Bayes probability estimation based on count information; and investigate on the use of detailed voting information to classify a test example. For example, if we consider all the evidences of the votes from contextual classifiers, we may be able to use some techniques (like evidence theory) to accumulate the collected evidence. This may result in an improvement of the performance of our algorithms.

## References


[Aha97] D. W. Aha. Lazy Learning, Kluwer Academic Publishers, 1997.

[Blanzieri99] E. Blanzieri, and F. Ricci. A minimum risk metric for nearest neighbor classification. in Proceedings of the Sixteenth International Conference on Machine Learning (ICML99), Bled, Slovenia, June 1999.

[Breiman96] L. Breiman. Bagging predictors. Machine Learning, Vol 24, pages 123-140, 1996.

[Dasarathy91] B. Dasarathy (Ed.). Nearest Neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press, 1991

[Domingos97] P. Domingos, and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning, 1997, 29: 103-130

[Duda73] R. Duda, P. Hart. Pattern Classification and Scene Analysis. New York: John Wiley &Sons, 1973.

[Fayyad93] U. M. Fayyad and K. B. Irani. Multi-inteval discretization of continuous-valued attributes for classification learning. IJCAI-93, pp. 1022-1027

[Freund96] Y. Freund. Boosting a weak learning algorithm by majority. Information and Computation, 1996, 121(2): 256-285

[Friedman97] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. Machine Learning, 1997, 29(2): 131-163

[Ganter99] B. Ganter and R. Wille. Formal Concept Analysis: Mathematical Foundations. Springer, 1999

[Godin94] R. Godin, and R. Missaoui. An incremental concept formation approach for learning from databases. Theoretical Computer Science, Special Issue on Formal Methods in Databases and Software Engineering, October 1994, 133: 387-419

[Kohavi96] R. Kohavi. Scaling up the accuracy of naïve-Bayes classifiers: A decision-tree hybrid. Proceedings of the second International Conference on Knowledge Discovery and Data Mining. Menlo Park, CA: The AAAI Press. 1996. pp. 202-207

[Liu98] B. Liu, W. Hsu, and Y. Ma, Integrating Classification and Association Rule Mining. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. New York, USA, 1998.

[Mephu94] E. Mephu-Nguifo. Galois Lattice: A Framework for Concept Learning. Design, Evaluation and Refinement. In: Proc. of 6th Intl. Conf. on Tools with Artificial Intelligence, New Orleans, USA, November 6-9, 1994, IEEE Press. pp. 461-467

[Merz96] C. J. Merz, and P. Murphy. UCI repository of machine learning database [http://www.cs.uci.edu/~mlearn/MLRepository.html], 1996

[Myles90] J. P. Myles and D. J. Hand. The multiclass metric problem in nearest neighbour discrimination rules. Pattern Recognition, 1990, 23(11): 1291-1297

[Pasquier99] N. Pasquier, J. Bastide, R. Taouil, L. Lakhal. Efficient mining of association rules using closed itemset lattices. Information Systems, 1999, 19(4): 33-54.

[Short81] R. D. Short and K. FuKunaga. The optimal distance measure for nearest neighbour classification. IEEE Transactions on Information Theory, 1981, 27: 622-627

[Wille82] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (eds.), Ordered Sets, Reidel, Dordrecht, 1982. pp. 445-470

[Zheng00] Z. Zheng, G. I. Webb. Lazy learning of Bayesian rules. Machine Learning, 2000, 41(1): 53-84.