Scaling up the Naive Bayesian Classifier: Using Decision Trees for Feature Selection

Chotirat "Ann" Ratanamahatana Dimitrios Gunopulos
Computer Science Department
University of California
Riverside, CA 92521
1-909-787-5190
{ratana, dg}@cs.ucr.edu

ABSTRACT
It is known that Naïve Bayesian classifier (NB) works very well on some domains, and poorly on some. The performance of NB suffers in domains that involve correlated features. C4.5 decision trees, on the other hand, typically perform better than the Naïve Bayesian algorithm on such domains. This paper describes a Selective Bayesian classifier (SBC) that simply uses only those features that C4.5 would use in its decision tree when learning a small example of a training set, a combination of the two different natures of classifiers. Experiments conducted on ten datasets indicate that SBC performs reliably better than NB on all domains, and SBC outperforms C4.5 on many datasets of which C4.5 outperform NB. Augmented Bayesian classifier (ABC) are also tested on the same data, and SBC appears to perform as well as ABC. SBC also can eliminate, on most cases, more than half of the original attributes, which can greatly reduce the size of the training and test data, as well as the running time. Further, the SBC algorithm typically learns faster than both C4.5 and NB, needing fewer training examples to reach high accuracy of classification.

Keywords
C4.5, Decision Trees, Feature Selection, Naïve Bayesian Classifier, Selective Bayesian Classifier.

1. INTRODUCTION
Two of the most widely used and successful methods of classification are C4.5 decision trees [25] and Naïve Bayesian learning (NB) [10]. While C4.5 constructs decision trees by using features to try and split the training set into positive and negative examples until it achieves high accuracy on the training set, NB represents each class with a probabilistic summary, and finds the most likely class for each example it is asked to classify.

Several researchers have emphasized on the issue of redundant attributes, as well as advantages of feature selection for the Naïve Bayesian Classifier, not only for induction learning. Pazzani [22, 23] explores the methods of joining two (or more) related attributes into a new compound attribute where the attribute dependencies are present. Another method, Boosting on Naïve Bayesian classifier [10] has been experimented by applying series of classifiers to the problem and paying more attention to the examples misclassified by its predecessor. However, it was shown that it fails on average in a set of natural domain [19]. Langley and Sage [18] use a wrapper approach for the subset selection to only select relevant features for NB. Cardie [5] uses the attributes from decision trees in combination with nearest neighbor methods. And in a domain for discovering patterns in EEG-signals, Kubat, Flotzinger, and Pfurtscheller [7] tried the use of Decision tree in feature selection for Naïve Bayesian classifier. And recently, Augmented Bayesian Classifiers [14] was introduced as another approach where Naïve Bayes is augmented by the addition of correlation arcs between attributes.

It has been shown that Naïve Bayesian classifier is extremely effective in practice and difficult to improve upon [8]. In this paper, we show that it is possible to reliably improve this classifier by using a feature selection method. Naïve Bayes can suffer from oversensitivity to redundant and/or irrelevant attributes. If two or more attributes are highly correlated, they receive too much weight in the final decision as to which class an example belongs to. This leads to a decline in accuracy of prediction in domains with correlated features. C4.5 does not suffer from this problem because if two attributes are correlated, it will not be possible to use *both* of them to split the training set, since this would lead to exactly the same split, which makes no difference to the existing tree. This is one of the main reasons C4.5 performs better than NB on domains with correlated attributes.

We conjecture that the performance of NB improves if it uses only those features that C4.5 used in constructing its decision tree. This method of feature selection would also perform well and learn quickly, that is, it would need fewer training examples to reach high classification accuracy.

We present experimental evidence that this method of feature selection leads to improved performance of the Naïve Bayesian Classifier, especially in the domains where Naïve Bayes performs not as well as C4.5. We analyze the behavior on ten domains from the UCI repository, 5 of which C4.5 achieves asymptotically

higher accuracy than NB (which seems to imply the presence of correlated features.), and 5 on which NB outperforms C4.5. We then compared our algorithm with the Augmented Bayesian classifier (ABC), and the experimental results justify our expectation. We also tested SBC on another sufficiently large synthetic dataset and our algorithm appeared to scale nicely. Our Selective Bayesian Classifier always outperforms NB and performs as well as, or better than both C4.5 and ABC on almost all the domains.

2. NAÏVE BAYESIAN CLASSIFIER

2.1 Description

The Naïve Bayesian classifier is a straightforward and frequently used method for supervised learning. It provides a flexible way for dealing with any number of attributes or classes, and is based on probability theory. It is the asymptotically fastest learning algorithm that examines all its training input. It has been demonstrated to perform surprisingly well in a very wide variety of problems in spite of the simplistic nature of the model. Furthermore, small amounts of bad data, or "noise," do not perturb the results by much.

The Naïve Bayesian classification system is based on Bayes' rule and works as follows. There are classes, say C_k for the data to be classified into. Each class has a probability $P(C_k)$ that represents the prior probability of classifying an attribute into C_k; the values of $P(C_k)$ can be estimated from the training dataset. For n attribute values, v_j, the goal of classification is clearly to find the conditional probability $P(C_k \mid v_1 \wedge v_2 \wedge \ldots \wedge v_n)$. By Bayes' rule, this probability is equivalent to

$$\frac{P(v_1 \wedge v_2 \wedge \ldots \wedge v_n \mid C_k) P(C_k)}{P(v_1 \wedge v_2 \wedge \ldots \wedge v_n)}$$

For classification, the denominator is irrelevant, since, for given values of the v_j, it is the same regardless of the value of C_k. The central assumption of Naïve Bayesian classification is that, within each class, the values v_j are all independent of each other. Then by the laws of independent probability,

$P(v_i \mid \{all\ the\ other\ values\ of\ v_j\},\ C_k) = P(v_i \mid C_k)$ and therefore

$$P(v_1 \wedge v_2 \wedge \ldots \wedge v_n \mid Ck) = P(v_1 \mid C_k) P(v_2 \mid C_k) \ldots P(v_n \mid C_k).$$

Each factor on the right-hand side of this equation can be determined from the training data, because (for an arbitrary v_i),

$$P(v_i \mid C_k) \approx [\#(v_i \wedge C_k)] / [\#(C_k)]$$

where "#" represents the number of such occurrences in the training set data. Therefore, the classification of the test set can now be estimated by

$P(C_k \mid v_1 \wedge v_2 \wedge \ldots \wedge v_n)$ which is proportional to

$$P(C_k)\, P(v_1 \mid C_k)\, P(v_2 \mid C_k)\, P(v_3 \mid C_k) \ldots P(v_n \mid C_k).$$

2.2 Problems

As mentioned above, the central assumption in Naïve Bayesian classification is that given a particular class membership, the probabilities of particular attributes having particular values are independent of each other. However, this assumption is often violated in reality. For example, in demographic data, many attributes have obvious dependencies, such as age and income.

A plausible assumption of independence is computationally problematic. This is best described by redundant attributes. If we posit two independent features, and a third (v_{r1}) which is redundant (i.e. perfectly correlated) with v_1, the classification expression is

$$P(v_1 \mid C_k)\, P(v_2 \mid C_k)\, P(v_{r1} \mid C_k)\, P(C_k),$$

which is effectively

$$P(v_1 \mid C_k)^2\, P(v_2 \mid C_k)\, P(C_k).$$

This means that the attribute v_1 has twice as much influence on the expression as v_2 has, which is a strength not reflected in reality. The increased strength of v_1 increases the possibility of unwanted bias in the classification. Even with this independence assumption, Hand and Yu illustrated that Naïve Bayesian classification still works well in practice [12]. However, some researchers have shown that although irrelevant features should theoretically not hurt the accuracy of Naïve Bayes, they do degrade performance in practice [13]. This paper illustrates that if those redundant and/or irrelevant attributes are eliminated, the performance of Naïve Bayesian classifier can significantly increase.

3. C4.5 DECISION TREES

Decision trees are one of the most popular methods used for inductive inference. They are robust for noisy data and capable of learning disjunctive expressions. A decision tree is a k-ary tree where each of the internal nodes specifies a test on some attributes from the input feature set used to represent the data. Each branch descending from a node corresponds to one of the possible values of the feature specified at that node. And each test results in branches, which represent different outcomes of the test. The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner.

The algorithm starts with the entire set of tuples in the training set, selects the *best* attribute that yields maximum information for classification, and generates a test node for this attribute. Then, top down induction of decision trees divides the current set of tuples according to their values of the current test attribute. Classifier generation stops, if all tuples in a subset belong to the same class, or

if it is not worth to proceed with an additional separation into further subsets, i.e. if further attribute tests yield only information for classification below a pre-specified threshold.

The decision tree algorithm usually uses an entropy-based measure known as "information gain" (although other measures are also possible) as a heuristic for selecting the attribute that will best split the training data into separate classes. Its algorithm computes the information gain of each attribute, and in each round, the one with the highest information gain will be chosen as the test attribute for the given set of training data. A well-chosen split point should help in splitting the data to the best possible extent. After all, a main criterion in the greedy decision tree approach is to build *shorter* trees. The best split point can be easily evaluated by considering each unique value for that feature in the given data as a possible split point and calculating the associated information gain.

3.1 Information Gain

The critical step in decision trees is the selection of the *best* test attribute. The *information gain* measure is used to select the test attribute at each node in the tree.

First, another related term called *entropy* needs to be introduced. In general, entropy is a measure of the purity in an arbitrary collection of examples. Let S be a set consisting of s data samples. Suppose the class label attribute has m distinct values defining m distinct classes, C_k. Let s_i be the number of samples of S in class C_k. The expected information needed to classify a given sample is given by

$$I(s_1, s_2, ..., s_m) = -\sum_{k=1}^{m} p_k \log_2(p_k),$$

where p_k is the probability that an arbitrary sample belongs to class C_k and is estimated by s_k / s.

Let attribute A have v distinct values, $\{a_1, a_2, ..., a_v\}$. Attribute A can be used to partition S into v subsets, $\{S_1, S_2, ..., S_v\}$, where S_j contains those samples in S that have value a_j of A. Let s_{kj} be the number of samples of class C_k in a subset S_j. The *entropy,* or expected information based on the partitioning into subsets by A, is given by

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + ... + s_{mj}}{s} I(s_{1j}, ..., s_{mj})$$

The term $\frac{s_{1j} + \cdots + s_{mj}}{s}$ acts as the weight of the j^{th} subset and is the number of samples in the subset divided by the total number of samples in S. For a given subset S_j,

$$I(s_{1j}, s_{2j}, ..., s_{mj}) = -\sum_{k=1}^{m} p_{kj} \log_2(p_{kj})$$

where $p_{kj} = s_{kj} / |S_j|$ and is the probability that a sample in S_j belongs to class C_k. The entropy is zero when the sample is *pure*, i.e. when all the examples in the sample S belong to one class. Entropy has a maximum value of 1 when the sample is maximally impure, i.e. there are same proportions of positive and negative examples in the sample S.

The encoding information would be gained by branching on A is

$$InformationGain(A) = I(s_1, s_2, ..., s_m) - E(A)$$

The attribute with the highest information gain is chosen as the test attribute for the current node. Such approach minimizes the expected number of tests needed to classify an object and guarantees that a simple (but may not be the simplest) tree is found.

3.2 Information Gain Ratio

A simple decision tree algorithm only selects one decision tree given an example set, though there may be many different trees consistent with the data. The information gain measure mentioned in section 3.1 (implemented in ID3 decision trees) is biased in that it tends to prefer attributes with many values rather than those with few values. C4.5 suppresses this bias by using an alternative measure called *Information Gain Ratio*, which considers the probability of each attribute value. The *Split Information* takes into account the factor of an attribute having many values. It is defined as

$$SplitInformation(A) = -\sum_{j=1}^{v} \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|}$$

And the gain ratio is

$$GainRatio(A) = \frac{InformationGain(A)}{SplitInformation(A)}$$

By using *SplitInformation(A)*, which is proportional to the number of values an attribute A can take, *GainRatio(A)* effectively removes the bias of information gain towards features with many values. To resolve the issue when *SplitInformation(A)* becomes very small, C4.5 lists the set of attributes with the *InformationGain(A)* above the average information gain for that node and then it uses the *Gain Ratio* to select the best attribute from the list [24].

3.3 Tree Pruning

C4.5 builds a tree so that most of the training examples are classified correctly. Though this approach is correct when there is no noise, accuracy for unseen data might degrade in cases where there is a lot of noise associated with the training examples and/or the number of training examples is very small. To alleviate this so-called *overfitting* problem, C4.5 uses the post-pruning method. This approach allows C4.5 to grow a complete decision tree first, and then *post-prune* the tree. It tries to shorten the tree in order to overcome overfitting. This generally involves removal of some of the nodes or subtrees from

the original decision tree. Its goal is to improve (by pruning) the accuracy on the unseen set of examples.

As a result, C4.5 achieves further elimination of features through pruning. It uses rule-post pruning to remove some of the *insignificant* nodes (and hence, some *not so relevant* features) from the tree.

4. SELECTIVE BAYESIAN CLASSIFIER

Our purpose is to improve the performance of the Naïve Bayesian classifier by removing redundant and/or irrelevant attributes from the dataset, and only choosing those that are most informative in classification task. To achieve this, we use the trees that are constructed by C4.5.

4.1 Description

As described in section 3, the features that C4.5 selected in constructing its decision tree are likely to be the ones that are most descriptive in terms of the classifier, in spite of the fact that a tree structure inherently incorporates dependencies among attributes, while Naïve Bayes works on a conditional independence assumption. C4.5 will naturally construct a tree that does not have an overly complicated branching structure if it does not have too many examples that need to be learned. As the number of training examples increases, the attributes that are considered will usually be the ones that are not correlated. This is mainly because C4.5 will use only one of a set of correlated features for making good splits in training set. However, sometimes many of the branches may reflect noise or outliers (overfitting) in the training data. "Tree pruning" procedure in C4.5 attempts to identify and remove those least reliable branches, with the goal of improving classification accuracy on unseen data. Even after pruning, if the resulted decision tree is still too *deep* or grown into too many levels, our algorithm only picks attributes contained in the first few levels of the tree as the most representative attributes. This is supported by the fact that by the selection of attributes that split the data in the best possible way at every node, C4.5 will try to ensure that it encounters a leaf at the very earliest possible point, i.e. it prefers to construct shorter trees. And by its algorithm, C4.5 will find trees that have attributes with higher information gain nearer to the root. We conjecture that this simple method of feature selection would help improve Naïve Bayesian classifier's performance and learn quickly, that is, it would need fewer training examples to reach high classification accuracy.

4.2 Algorithm

Figure 1 shows the algorithm for the Selective Bayesian classifier. We first shuffle the training data and use 10% of that to run C4.5 on. This is to make sure that all the subsamples are not biased toward any particular classes. We find 10% of the training to be a good size for our feature selection process. If too small a portion is used, the decision tree may not be representative enough for the unseen data. And if too large a portion is used, it unnecessarily takes longer to construct a decision tree and the tree also are likely to be too complex. Once we run C4.5 and obtain the decision tree, we only pick attributes that only appear in the first 3 levels of the decision trees as the most relevant features. We hypothesize that if a feature in the deeper levels on any one execution of C4.5 is *relevant* enough, it will finally rises up and appear in one of the top levels of the tree in some other executions of C4.5. We form a union of all the attributes from each run, and finally, run the Naïve Bayesian classifier on the training and test data using only those features selected in the previous step.

1. Shuffle the training data and take a 10% sample.
2. Run C4.5 on data from step 1.
3. Select a set of attributes that appear only in the first 3 levels of the simplified decision tree as *relevant* features.
4. Repeat 5 times (step 1-3)
5. Form a union of all the attributes from the 5 rounds.
6. Run Naïve Bayesian classifier on the training and test data using *only* the final features selected in step 5.

Figure 1. Selective Bayesian Classifier Algorithm: Feature Selection Using C4.5

One of the problems with using C4.5 to generate decision trees when there are too few training examples available is that it might give a constant decision (for example, classify all examples as Democrat in the voting domain) without generating the decision tree. In this case, the training set is re-sampled until a non-constant decision tree is produced.

The main difference between our algorithm and the one proposed by Kubat, Flotzinger, and Pfurtscheller[17] is that they build just one tree on the *entire* training set, and use all the attributes that occur in it. Instead, we only take 10% as a training set to build a tree and then keep the attributes only from the first 3 levels of the tree. This will make the tree-building process for feature selection much faster, especially on larger datasets.

5. EXPERIMENTAL EVALUATION

5.1 The Datasets

We used 10 datasets from the UCI repository [21] and one synthetic dataset, shown in Table 1. The Synthetic dataset, created with Gaussian distribution, contains 1,200,000 instances with 20 attributes and 2 classes. We chose 10 datasets from the UCI databases, 5 of which Naïve Bayesian classifier outperforms C4.5 and the other 5 of which C4.5 outperforms Naïve Bayesian classifier.

Table 1. Descriptions of domains used

Dataset	#Attributes	#Classes	#Instances
Ecoli	8	8	336
GermanCredit	20	2	1,000
KrVsKp	37	2	3,198
Monk	6	2	554
Mushroom	22	2	8,124
Pima	8	2	768
Promoter	57	2	106
Soybean	35	19	307
Wisconsin	9	2	699
Vote	16	2	435
SyntheticData	20	2	1,200,000

5.2 Experimental Design

- Each dataset is shuffled randomly.
- Produce *disjoint* training and test sets as follows.

  10% training and 90% test data

  20% training and 80% test data

  30% training and 70% test data

  … … … …

  80% training and 20% test data

  90% training and 10% test data

  99% training and 1% test data
- For each set of training and test data, run
  - Naïve Bayesian Classifier (NBC)
  - C4.5, and
  - Selective Bayesian Classifier (SBC)
- Repeat 15 times

The classifier accuracy is determined by *Random Subsampling* method, i.e. the holdout method that is repeated k times. The overall accuracy estimate is the mean of the accuracies obtained from all iterations. This will give us information about both the learning rates, as well as the accuracy of the learning algorithms used.

To see how well our algorithm matches up with others, we did run the Augmented Bayesian classifier (ABC) on all 10 datasets as well, using *SuperParent* proposed by Keogh and Pazzani [14] as a method of finding the set of augmented arcs.

And lastly, to help us clearly see the speedup and scalability of the Selective Bayesian classifier, we also ran an experiment on the synthetic data, which is much larger than what we have in the UCI repository.

5.3 Experimental Results

The results confirm the initial hypotheses. The performance of the Selective Bayesian classifier is quite impressive. Its asymptotic accuracy is as good as (or slightly better than) the better of C4.5 and NB on each of the domains.

Figure 2 – 11 depict the learning curves for the 10 UCI datasets. It is clear that SBC learns faster than both C4.5 and NBC on all the dataset, i.e. with small number of training data (e.g. 10%), the prediction accuracy for SBC is higher.

Note that all the C4.5 accuracies considered in this experiment are based on the simplified decision tree (with pruning). This accuracy is usually higher on the test (unseen) data, in comparison to the accuracy based on unpruned decision trees.

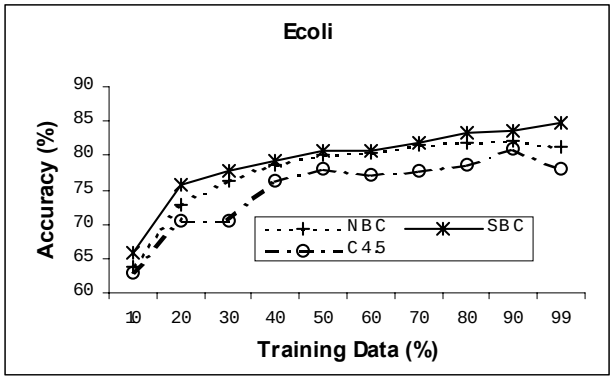

Figure 2. Ecoli dataset. 336 instances, 8 attributes, 8 classes. Attributes selected by SBC = 4.

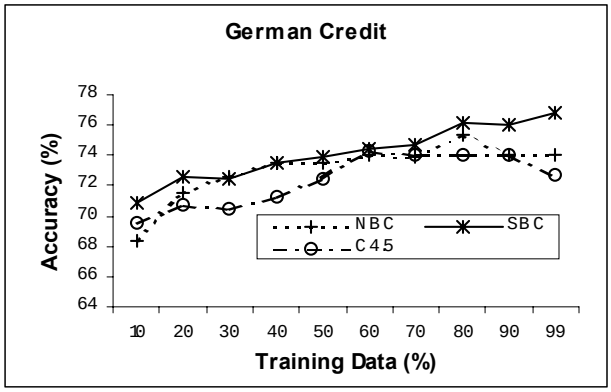

Figure 3. German Credit dataset. 1,000 instances, 20 attributes, 2 classes. Attributes selected by SBC = 6.

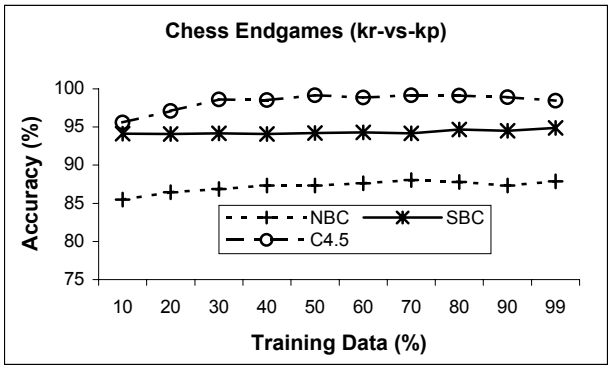

Figure 4. Kr-vs-Kp dataset. 3,198 instances, 37 attributes, 2classes. Attributes selected by SBC = 4.

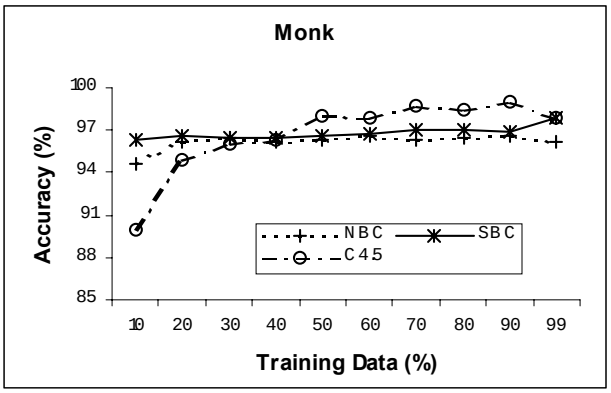

Figure 5. Monk dataset (prob.3). 554 instances, 6 attributes, 2 classes. Attributes selected by SBC = 4.

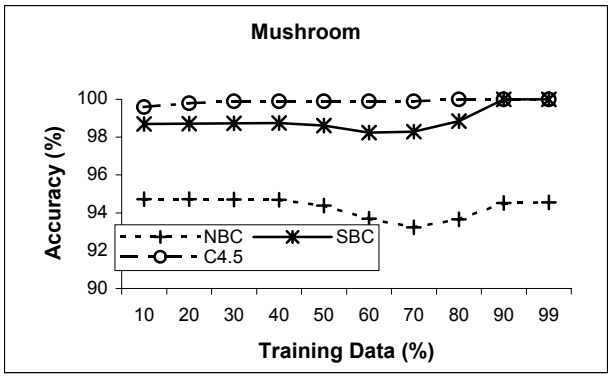

Figure 6. Mushroom dataset. 8,124 instances, 22 attributes, 2 classes. Attributes selected by SBC = 6.

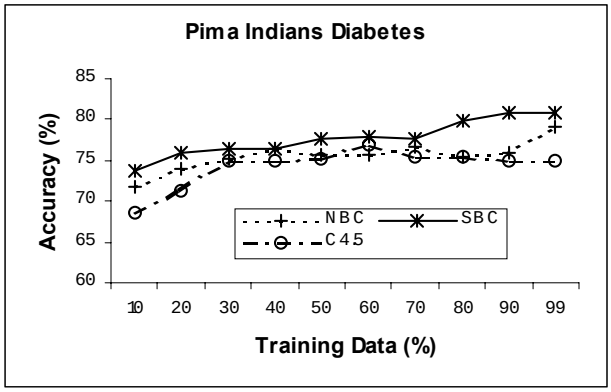

Figure 7. Pima-Indians dataset. 768 instances, 8 attributes, 2 classes. Attributes selected by SBC = 5.

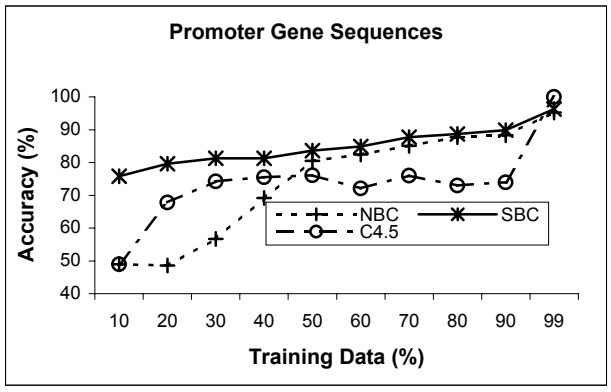

Figure 8. Gene Promoter dataset. 106 instances, 57 attributes, 2 classes. Attributes selected by SBC = 5.

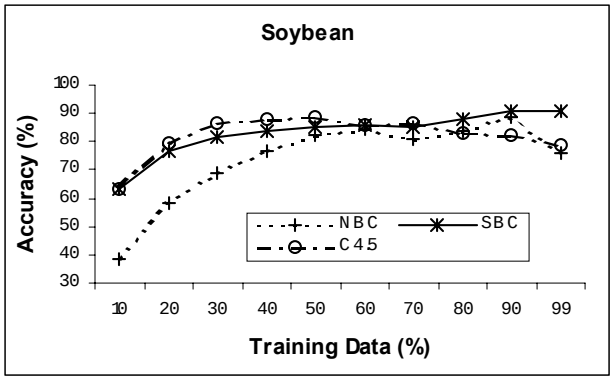

Figure 9. Soybean-large dataset. 307 instances, 35 attributes, 19 classes. Attributes selected by SBC = 12.

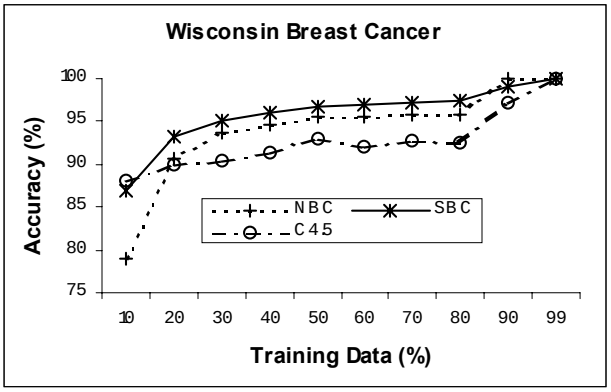

Figure 10. Wisconsin Breast Cancer dataset. 699 instances, 9 attributes, 2 classes. Attributes selected by SBC = 4.

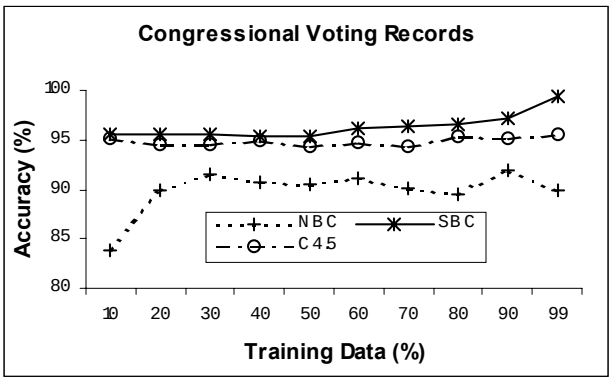

Figure 11. Congressional Voting dataset. 435 instances, 16 attributes, 2 classes. Attributes selected by SBC = 3.

To see a clearer picture on the SBC performance, we did an experiment on the same set of data using the Augmented Bayesian classifier (ABC) to see how the results would compare with our SBC. Table 2 shows the results for NBC, C4.5, ABC, and SBC learning algorithms using 80% of the data for training and 20% for testing (5-fold cross-validation). The figures reported in bold reflect the winning method on each dataset. The last three columns show the improvement of SBC over NBC, C4.5, and ABC, respectively. The last row of the table gives the mean accuracies and improvements for each learning algorithm.

From table 2, it is apparent that SBC outperforms the original NBC in every domain, giving the accuracy improvement up to 7.9%. SBC also outperforms both C4.5 and ABC in almost all the domains, giving the accuracy improvement up to 33.1% over C4.5. Even though, SBC cannot beat C4.5 in some datasets, it still gives quite big improvement over the Naïve Bayes (7.8%, 1.4%, and 6.0%) on such cases. It is also shown that SBC does perform as well as, or sometimes better than the Augmented Bayesian classifier learning algorithm.

Table 2. Accuracy of each learning method using 5-fold cross-validation

Dataset	NBC	C4.5	ABC	SBC	SBC vs NBC	SBC vs C4.5	SBC vs ABC
Ecoli	81.99	78.65	**84.35**	83.27	+1.6%	+5.9%	-1.3%
GerCredit	75.35	74.00	76.13	**76.21**	+1.1%	+3.0%	+0.1%
KrVsKp	87.81	**99.12**	94.87	94.69	+7.8%	-4.5%	-0.2%
Monk	96.16	**98.46**	98.02	97.47	+1.4%	-1.0%	-0.6%
Mushroom	93.23	**99.8**	97.98	98.85	+6.0%	-1.0%	+0.9%
Pima	75.03	75.35	78.13	**79.94**	+6.5%	+6.1%	+2.3%
Promoter	87.66	66.67	**88.66**	**88.72**	+1.2%	+33.1%	+0.1%
Soybean	84.02	83.20	**88.32**	**88.27**	+5.1%	+6.1%	-0.1%
Wisconsin	95.78	92.63	96.18	**97.38**	+1.7%	+5.1%	+1.2%
Vote	89.54	95.29	95.54	**96.61**	+7.9%	+1.4%	+1.1%
Mean	86.65	86.32	89.82	**90.14**	+4.0%	+5.4%	+0.4%

Table 3 shows the number of features selected for Selective Bayesian classifier. On almost all the datasets, surprisingly more than half of the original attributes were eliminated. 30% or less of all attributes *selected* were shown in bold. In other words, we can actually pay no attention to more than 70% of the original data and still achieve very high accuracy in classification.

Table 3. Number of features selected

Dataset	#Attributes	# of Attributes selected
Ecoli	8	4
GermanCredit	20	**6**
KrVsKp	37	**4**
Monk	6	4
Mushroom	22	**6**
Pima	8	5
Promoter	57	**5**
Soybean	35	12
Wisconsin	9	4
Vote	16	**3**
SyntheticData	20	12

For speedup and scalability issues, we ran SBC on a large synthetic data just to see how fast it can learn. The running time for SBC on our synthetic data gave **1.14** and **4.24** speedup over the original NBC and C4.5, respectively. Note that we only used **2,000** instances out of the total of 1,200,000 instances for C4.5 feature selection process, which made it a very quick operation.

Hence, in practice, if the dataset is large enough, we can even sample much less than 10% of data for the feature selection process. The number of attributes selected by SBC was **12** out of the total of 20 attributes. Table 4 illustrates the mean elapsed time (user and system time) for each classifier on this synthetic data, using 1,000,000 instances for training and 200,000 instances for test data.

Table 4. Mean Elapsed time for Synthetic Dataset (sec)

NBC	C4.5	SBC
37.546	139.5	32.912

The running times of both SBC and NBC are much less than that of C4.5 because Bayesian classifier only needs to go through the whole training data once. They are also space efficient because they build up a frequency table in size of the product of the number of attributes, number of class values, and the number of values per attribute [26].

5.4 Discussion

From our experimental results, we have achieved the following:

1. C4.5 does pick good features for its decision tree (especially ones that are nearer to the root), which in turn asymptotically improves the accuracy of the Naïve Bayesian algorithm, when *only* those features are used in the learning process.
2. The running time of SBC is fastest among NBC, C4.5, and ABC. On average, SBC only selects less than 50% of the features from the original data to be used in the classification process (see Table 3). SBC, in comparison to NBC, learns faster because fewer attributes are involved in learning. However, it is obvious that most of the time spent in both algorithms was on I/O, reading the training data (both NBC and SBC read in the same training data, but SBC only uses the features selected by C4.5 in learning). That explains why SBC time did not reduce much from NBC time. If there exists a very fast preprocessing way of removing unwanted features from a very large dataset to reduce the size of the training data beforehand, SBC would only need 25.746 seconds (for synthetic dataset) and give 31.4% improvement in running time over NBC, and give 81.5% improvement in running time over C4.5. and
3. You can pick good features even if only a small sample of the original data is used. From our experiment, only 10% of the training data for the 10 UCI datasets and 0.2% of the training data for the Synthetic dataset seem to be very sufficient.

6. RELATED WORK

Much work has been done on feature subset selection. John, Kohavi, and Pfleger [13] define the problem of feature subset selection to be that of finding a subset of the original set of features of a dataset, such that the induction algorithm that is run on the data containing only the features from the subset generates a classifier with the highest possible accuracy. Other features may be discarded as their information increases the complexity of learning algorithm without increasing accuracy.

Blum and Langley [2] classify the feature subset selection techniques into three categories:

- **Embedded Selection Techniques.** These techniques include induction algorithms that implicitly carry out feature subset selection by inducing logical descriptions. Version Spaces [20] searches for a subset of features in the feature space by adding or removing a feature from the generated set, so that the prediction errors are lower for newer instances. Decision tree algorithms such as ID3 or its successor C4.5 and CART [4] are another form of embedded techniques.
- **Filter Techniques.** These approaches use a feature selection algorithm to select the subset, which is then passed onto the induction algorithm. A very simple filter approach would be to select *k* attributes that have the highest correlation with the output class. One metric that would measure such correlation would be the mutual information between the corresponding input feature and the output feature. The RELIEF algorithm [15] uses such a mechanism with added complexity to the feature evaluation function. FOCUS [1] is another approach that searches for the smallest possible subset of features that will completely split up the training set without any error. Cardie [5} uses a decision trees for nearest neighbor retrieval, whereas Kubat, Flotzinger, and Pfurtscheller [17] uses a decision trees to filter features for use with Naïve Bayesian classifier.
- **Wrapper Techniques.** The feature subset selection algorithm forms a wrapper on top of the induction algorithm. This is a search strategy to find an *optimal* subset of features by adding or deleting features from the input feature set, depending on the accuracy of the induction algorithm itself. This approach by Kohavi and John [16] also searches the feature space for the optimal subset by starting with an empty set of selected features. However, a major *disadvantage* associated with the wrapper mechanism is the computational cost involved. Faster evaluation techniques have been evolved to reduce the computation. Caruana and Freitag [5] evolved a technique where the decision trees are cached for later use in the process of search. Augmented Bayesian classifier (ABC) uses a different approach in constructing tree-augmented Bayesian networks by adding the correlation arcs between attributes and using a more efficient heuristic search to find the best arcs to add (SuperParent) [14].

7. CONCLUSION

A simple method that uses C4.5 decision trees to select features has been described. This is to be used to improve Naïve Bayesian learning. The empirical evidence shows that this method is very fast and surprisingly successful, given the very different natures of the two classification methods. This Selective Bayesian classifier is asymptotically at least as accurate as the best of C4.5, Naïve Bayes, and Augmented Bayes on each of the domains on which the experiments were performed. Further, it learns faster than both C4.5 and NB on each of these domains.

This work suggests that C4.5 decision trees systematically select good features for Naïve Bayesian classifier to use. We believe the reasons are that C4.5 does not use redundant attributes in constructing decision trees, since they cannot generate different splits of training data. When few training examples are available, C4.5 uses the most relevant features it can find. The high accuracy SBC achieves with few training examples is indicative of the fact that using these features for probabilistic induction leads to higher accuracy both in Bayesian classifier and C4.5 itself in each of the domains we have examined.

8. REFERENCES

[1] Almuallim, H. and Dietterich, T.G. (1991). Learning with many irrelevant features. In Proceedings AAAI-91, volume 2, pp. 547-552, Anaheim, CA.

[2] Blum, A.L., and Langley, P. (1997). Selection of Relevant Features and Examples in Machine learning. Artificial Intelligence, 97, pp. 245-271.

[3] Boz, O. Feature Subset Selection by Feature Relevance. Submitted for the ICML 2002.

[4] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, P.J. (1984). Classification and Regreession Trees. Wadsworth International Group. Belmont, CA.

[5] Cardie, C. Using Decision Trees to Improve Case-based Learning. ICML 1993, pp. 25-32.

[6] Caruana, R., and Freitag, D. (1994). Greedy Attribute Selection. In: Cohen, W.W., and Hirsh, H. (eds). Prodeedings of the 11th International Conference on Machine Learning. San Mateo, CA: Morgan Kaufmann, pp.28-36.

[7] Domingos, P. and Pazzani, M. (1997). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In Proceedings of the ICML 1996, pp.105-112.

[8] Domingos, P. and Pazzani, M. On the Optimality of the Simplie Bayesian Classifier under Zero-One Loss. Machine Larning, 29(2/3): 103-130, November/December 1997.

[9] Duda, R.O. and Hart, P.E. (1973). Pattern Classification and Scene Analysis. New York, NY: Wiley and Sons.

[10] Elkan, C. Boosting and Naïve Bayesian Learning. Technical Report No. CS97-557, Department of Computer Science and Engineering, University of California, San Diego, Spetember 1997.

[11] Han, J. and Kamber, M. (2001) Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, CA.

[12] Hand, D. and Yu, K. (2001) Idiot's Bayes—Not So Stupid After All? International Statistical Review (2001), 69, pp.385-398.

[13] John, G.H., Kohavi, R., and Pfleger, K. (1994). Irrelevant Features and the Subset Selection Problem. In: Cohen, W.W., & Hirsh, H. (eds). ICML 1994, pp. 121-129.

[14] Keogh, E. and Pazzani, M. Learning Augmented Bayesian Classifiers: A comparison of distribution-based and classification-based approaches. Uncertainty 99, 7th Int'l Workshop on AI and Statistics, pp. 225-230.

[15] Kira, K. and Rendell, L.A. (1992). A practical approach to feature selection. In Proceedings of the 9th International Conference on Machine Learning. pp. 249-256, Aberdeen, Scottland.

[16] Kohavi, R. and John, G.H. (1997). Wrappers for feature subset selection. Artificial Intelligence, 97(1-2): 273-323.

[17] Kubat, M., Flotzinger, D., and Pfurtscheller, G. (1993). Discovering patterns in EEG-signals: Comparative study of a few methods. European Conference on Machine Learning, Vienna, 1993. LNCS vol. 667, pp. 366-371.

[18] Langley, P. and Sage, S. Induction of Selective Bayesian Classifiers. Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (1994). Seattle, WA: Morgan Kaufmann

[19] Ming, K. and Zheng, Z. Improving the Performance of Boosting for Naïve Bayesian Classification. In Proceedings of the PAKDD-99, pp.296-305, Beijing, China.

[20] Mitchell, T.M. (1977). Version Spaces: A candidate elimination approach to rule learning. In Proceedings of the 5th International Joint Conference on Artificial Intelligence. pp.305-310, Cambridge, MA.

[21] Murphy, P.M. and Aha, D.W. (1994). UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Dept. of Information and Computer Science.

[22] Pazzani, M. (1995). Searching for dependencies in Bayesian classifiers. Preliminary Papers of the 5th

International Workshop on Artificial Intelligence and Statistics. Ft. Lauderdale, FL.

[23] Pazzani, M. (1996). Constructive Induction of Cartesian Product Attributes. Information, Statistics and Induction in Science. Melbourne, Australia.

[24] Quinlan, J.R.(1990). Induction of Decision Trees. In Reading in Machine Learning. Morgan Kaufmann, Dordrecht, The Netherlands. Originally published in Machine Learning 1:81-106, 1986.

[25] Quinlan, J.R. (1993). C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann.

[26] Zhang, H., Ling, C.X., and Zhao, Z. The Learnability of Naïve Bayes. Canadian Conference on AI 2000: 432-411.