Cascading Customized Naïve Bayes Couple

Guichong Li[1], Nathalie Japkowicz[1], Trevor J. Stocki[2] and R. Kurt. Ungar[2],

[1] Computer Science, University of Ottawa,
800 King Edwards, Ottawa, Canada
{jli136, nat}@site.uottawa.ca
[2]Radiation Protection Bureau, Health Canada, Ottawa, ON, Canada
{trevor_stocki, kurt_ungar}@hc-sc.gc.ca

Abstract. Naïve Bayes (NB) is an efficient and effective classifier in many cases. However, NB might suffer from poor performance when its conditional independence assumption is violated. While most recent research focuses on improving NB by alleviating the conditional independence assumption, we propose a new Meta learning technique to scale up NB by assuming an altered strategy to the traditional Cascade Learning (CL). The new Meta learning technique is more effective than the traditional CL and other Meta learning techniques such as Bagging and Boosting techniques while maintaining the efficiency of Naïve Bayes learning.

Keywords: Naïve Bayes, Meta Learning, Cascade Learning.

1 Introduction

Naïve Bayes (NB) is a simple Bayesian classifier which assumes conditional independence of the domain's attributes. Research has shown that NB exhibits high accuracy over other classifiers in many cases [5]. Moreover, because it is efficient, due to its linear time complexity for training, NB has been widely applied to many practical applications such as text classification [20].

The main problem with NB is that it suffers from poor performance if the conditional independence assumption is violated [20]. Previous research emphasizes the improvement of NB by alleviating the assumption [5][19][26]. For example, NB can be improved by selecting an effective set of attributes which satisfy the conditional independence assumption [17] or by combining decision tree with Naïve Bayes [16]. More recent research focuses on One-Dependent-Estimation (ODE) [14][26][29][30] by building a simplified Bayesian network structure in that each node has a single parent node.

However, these improved approaches sometimes still suffer from either ineffectiveness or a high time and space complexity as compared to the original NB while they achieve quite a success in some circumstances. More simply, one can apply a Meta learning technique such as Adaptive Boosting (AdaBoost) [9] to improve individual classifiers. This seems more attractive because a Meta learner promises a high efficiency by using a linear combination of individual classifiers. Unfortunately, the previous Meta learning techniques attempted [2][25] failed to scale up NB due to the stability of NB [1][7][23][25].

In this paper, we propose a new Meta learning technique to scale up NB by altering the strategy of the traditional Cascade Learning (CL) [11][24][28]. When the traditional CL achieves its goal by constructing a new feature space for the Meta level, it is observed that it often failed to construct a proper feature space for discrimination in the Meta level. Instead, the proposed new Meta learning technique, called Cascading Customized Couple (CCC), is a domain-based CL built by the construction of sub-domains from the original domain.

The main advantages of this method consist of the following two aspects:

Firstly, it is more efficient than previously proposed approaches for scaling up NB because it is a linear combination of two NB classifiers without a significant increase of the time and space complexity of NB; Secondly, it is more effective than either the traditional CL in most cases by assuming an altered strategy to the CL, or other Meta learning techniques such as Bagging and Boosting techniques for scaling up NB.

2 Previous Research

2.1 Meta Learning and Cascade Learning

Previous research has proposed Meta learning techniques to scale up individual classifiers. Bootstrap aggregating (Bagging) [2] builds a Meta learner by building diverse models on subsamples (or bags) obtained by sampling uniformly with replacement on the original training set. Adaptive Boosting (AdaBoost) is a Meta Learning algorithm [9], which improves any learning algorithm by repeatedly calling a weak classifier. In each round, the weights of each incorrectly classified example are increased and the weights of each correctly classified example are decreased, so that the new classifier focuses on the incorrectly classified examples. MultiBoostAB [25] assumes the Adaboost technique and wagging (sampling with different weights) technique to enhance the original AdaBoost by wagging a set of sub-committee of classifiers so that each of them is formed by AdaBoost.

Cascade Learning (CL) [11][24][28] is quite different from the Bagging and Boosting techniques in that it emphasizes a straightforward relationship among individual components. For example, previously proposed cascade learning techniques such as cascade generalization [11] and stacked generalization [24][28] build a set of classifiers on the original domain. They also output class probability distributions for each instance in the original domain. A new domain can be constructed from the original domain by using these class probability distributions as its feature values in the Meta level.

Although Bagging, AdaBoost, and MultiBoostAB demonstrates success in many practical applications, they often suffer from failures to scale up NB due to the stability of NB [1][7][23][25]. CL, on the other hand, has difficulty creating a proper feature space for purpose of discrimination in the Meta level so that it is improper to be used for scaling up NB in many cases.

2.2 Naïve Bayes and Enhancement

Given a training set with a probability distribution P, in supervised learning, Bayesian learning defines a classifier with a minimized error, i.e.,

$$y_{\mathrm{i}} = c_{\mathrm{i}} = \arg\max_{c_i \in C} P(c_i \mid x) \equiv \arg\max_{c_i \in C} P(x|c_i)P(c_i)$$
$$= \arg\max_{c_i \in C} P(a_1, a_2, ..., a_{\mathrm{n}} \mid c_i)P(c_i) \qquad (2.1)$$

Naïve bayes (NB) [6][18] assumes the probabilities of attributes $a_1,a_2,\ldots,a_{\mathrm{n}}$ to be conditionally independent given the class c_{i}. Therefore, the right side of (2.1) becomes

$$P(x \mid c_i) = P(a_1, a_2, ..., a_n \mid c_i) = \prod_{j=1}^{n} P(a_j \mid c_i)$$

NB can be trained in the linear time complexity O(mn), where m is the number of attributes and n is the number of examples in the training set. Moreover, NB is a stable classifier, and has exhibited a high performance over other classifiers in many applications.

On the other hand, a number of studies promise the improvement of NB by overcoming the restrictions of the conditional independence assumption. We summarize these methods into the following three categories:

(I) select a subset of attributes such that they satisfy the conditional independence assumption. For example, Selective Bayesian Classifiers (SBC) [19] uses forward selection in a greedy search to find an effective subset of attributes, with which a better Naïve Bayes is built.

(II) Combine a decision tree with Naïve Bayes. For example, NBTree [16] builds a local NB on each leaf of a decision tree.

(III) Build a simplified Bayesian network structure by allowing a simple relationship between attributes given a class [14][26][29][30]. Tree Augmented Naïve Bayes (TAN) [10] extends tree-like Naïve Bayes, in which the class node directly points to all attribute nodes, and an attribute node has only one parent attribute. TAN has a time complexity of $O(m^2 logm)$ for structure learning, and then a time complexity $O(nm^2+km^2v^2+m^2 logm)$ for training, where m and n are defined as above; k is the number of classes, and v is the average number of values for each attribute.

TAN is also regarded as a One-Dependent Estimation (ODE) technique. Other ODE techniques [14][26][30][29] improve TAN without searching for the simplified Bayesain network structure. For example, Aggregating One-Dependence Estimators (AODE) [26] achieves higher accuracy than NB by averaging over a constrained group of 1-dependence NB models built on a small space. AODE has the time and space complexities of $O(nm^2)$ and $O(k(nm)^2)$, respectively. Although other ODE techniques including AODE is more efficient than TAN, they are still more complicated than NB with respect to the time and space complexities.

In this paper, we consider a novel Meta learning technique to scale up NB.

3 Cascading Customized Naïve Bayes Couple

3.1 Cascade Learning

As we know from Section 2.1, traditional CL defines the base level and constructs a new feature space for the Meta level [28]. Given a dataset $D = \{(y_i, x_i), i = 0,\ldots, n-1\}$, where y_i is a class value, x_i is a vector of attribute values, and n is the number of examples in D, for the purpose of scaling up NB, we build K NB models on D in the base level by using k-cross validation for diversity. The kth model outputs a probability prediction p_{ik} for an input (y_i, x_i), where $k = 0, \ldots, K-1$. Therefore, we obtain a new dataset $\{(y_i, p_{ik}), k = 0,\ldots, K-1$, and $i = 0,\ldots, n-1\}$ for the Meta level where another NB model is built on the new feature space.

However, this CL is not easily realized in practice because it is unclear how to construct a proper feature space for discrimination in the Meta level [24], and it is also difficult to make NB diverse because NB is a stable learner. To overcome this obstacle, we adopt an altered strategy for CL and propose a new definition of a domain-based CL as follows.

Definition 3.1. Domain-based Cascade Learning (DCL) is an ensemble learning technique that learns individual classifiers from the original domain by building each component on its own sub-domain which can be reconstructed in terms of the outputs of previously built component learners. All components together make a decision.

The above definition of DCL, or simply CL without any confusion, finds its roots in previous research related to cascade generalization [11] and stacked generalization [24][28]. Some ideas similar to CL in Definition 3.1 have been raised in previous research. For example, the Cascade-Correlation method learns the architecture of artificial neural networks using a strategy similar to CL [8]. A mixture of local experts [13] can be built on a partitioned domain by assuming the Expectation Maximization (EM) algorithm for the mixture [15]. Recursive Bayesian Classifiers (RBC) [17] is a suggestive schema that uses a hierarchy of probabilistic summaries instead of a single schema, i.e., RBC partitions instances and recursively builds simple NB on each partition. Our method proposed in this paper can be regarded as an implementation of this hierarchy.

3.2 Customized Classifier

The main idea behind this altered CL is related to a new classifier, called *Customized Classifier* (CC). For example, NB is a linear classifier [6], and it can become a CC. Suppose that a training set is divided into many small subsets. Therefore, an individual NB classifier can be built on the partitioned training set for classifying a target subset. We describe several related concepts as follows.

Definition 3.2. A labeled training set can be regarded as a *domain*, which describes some domain knowledge. A subset of the training set can be regarded as a *sub-domain*.

A sub-domain does not have to contain examples belonging to the same category although examples in each original class naturally constitute a sub-domain. Not any sub-domain but those sub-domains that cross the class boundary are more likely to be informative.

The original domain can be divided into many sub-domains if necessary, and sub-domains can be labeled by *artificial class labels* as additional classes. Therefore, a classifier can be customized on the partitioned domain in terms of the related sub-domain.

Definition 3.3. A *Customized Classifier* (CC) is a classifier, which can classify an input in the related sub-domain, and can reject the classification on an input outside the sub-domain.

Definition 3.3 can be further explained as follows. A CC can output the class distribution of an input with respect to both of the original classes and the additional classes. For an input outside the related sub-domain, the CC intends to classify it by outputting a class membership probability of 0 or an equal class membership probability for the original classes. This leads to the *rejection* of classification on the input by eliminating its effect on the final classification if an averaging combination rule is used.

Although the number of CC is not limited, we emphasize the use of a couple of individual CCs in this altered CL. In particular, we suggest Cascading Customized Couple (CCC) for scaling up NB. The principles of CC and CCC can be simply described in Example 3.1.

Example 3.1. Given a domain D with two original classes, c_1, c_2, without any loss of generality, two CC classifiers, denoted as H_1 and H_2, are built from D, where H_1 has its sub-domain S_1 and the outside of the sub-domain is labeled by c_3; H_2 has its sub-domain S_2 and the outside of the sub-domain is labeled by c_4.

Given an input $x \in S_1$ with a true label c_1, because x is in the sub-domain S_1 of H_1, H_1 can correctly classify x. Suppose we have $P_1(c_1|x) = 0.6$, $P_1(c_2|x) = 0.3$, and $P_1(c_3|x) = 0.1$

Because x is not in the sub-domain S_2 of H_2, H_2 cannot classify x into either c_1 or c_2. Instead, H_2 classifies x into its additional class c_4, Suppose we have

$P_2(c_1|x) = P_2(c_2|x) = 0.2$, and $P_2(c_4|x) = 0.6$, i.e., H_2 rejects classifying x into either c_1 or c_2 because $P_2(c_1|x) = P_2(c_2|x)$. Therefore,

$p_1 = (P_1(c_1|x) + P_2(c_1|x) / 2 = 0.4$

$p_2 = (P_1(c_2|x) + P_2(c_2|x) / 2 = 0.25$

As a result, $c_1 = \arg\max_i (P_1(c_i \mid x) + P_2(c_i \mid x)/2)$, where $i = 1, 2$. □

3.3 Learning Algorithm

The main issue is that it is difficult to exactly build CCs. Instead, we intend to build approximate CCs for diversity as follows.

Given a training set D, the initial domain is the whole training set with original class labels. The first CC_0 is built on D using a base learner, i.e., NB. CC_0 actually is a traditional classifier built on the original training set. The learning algorithm is completed if CC_0 totally fits the training set D without misclassifications.

Otherwise, the misclassifications need to be further classified. To this end, we can add additional classes to label the corresponding correct classifications. As a result, all misclassifications become a sub-domain, and the outside of the sub-domain is labeled by artificial labels as the additional classes. The original training set D becomes a new training set D_1 with the sub-domain containing the misclassifications and the additional classes corresponding to the classified examples. The second CC_1 is built on D_1 and the learning algorithm ends up with a couple of CC classifiers.

Because CC_1 is built in terms of the outputs of CC_0, we say that they are cascaded with each other on D, and they are combined with each other to become a CCC classifier. However, we intend to only build approximate CCs, i.e., CC_0 and CC_1, for diverse models.

CCC algorithm
input D: original domain;
       L: a specified base learner, e.g., NB
output H:*CCC*, the resulting CCC classifier
1 saveLabels(D)
2 $B = \varnothing$
   // first CC
3 h_1 = L(D), $B = B \cup \{h_1\}$
4 $E = h_1(D)$, CT = D – E
   // second CC
5 if(|CT| < |D|)
6   addClasses(CT, D, 0)
7   h_2 = L(D), $B = B \cup \{h_2\}$
8 $H(x) = \tilde{c} = \arg\max_{c \in C} \left(P''(H(x) = c)\right)$,

8.1 $P''(H(x) = c) = \dfrac{P'(H(x) = c)}{\sum_{c' \in C} P'(H(x) = c')}$

8.2 $P'(H(x) = c) = \dfrac{1}{|B|} \sum_{h \in B} P(h(x) = c)$, where $c \in C$

8.3 $P(h(x) = c) = \dfrac{P(h(x) = c)}{\sum_{c' \in C'} P(h(x) = c')}$

9 restoreLabels(D)
10 return H:CCC(B)

Figure 1. Cascading Customized Couple induction algorithm.

According to the above discussion, we propose the Cascading Customized Couple (CCC) induction algorithm to learn a CCC classifier, as shown in Figure 1.

The CCC algorithm builds a CCC classifier with its two inputs: the original training set *D* and a base learner L(), i.e., NB. Because the algorithm performs labeling on *D*, the algorithm initially saves all original labels of examples in the training set *D* by saveLabels() at Step 1 while it restores all original labels of examples at Step 9 by restoreLabels() after it builds the couple of approximate CC classifiers or simply CC classifiers without any confusion.

At Step 2, *B* is initialized as an empty set, which is used for collecting the resulting CC classifiers. The first CC learner h_1 is built on *D* at Step 3 using the base learner L() = NB() to build a traditional NB classifier. At Step 4, the misclassifications *E* of h_1 on *D* are computed, and the correct classifications CT on D are obtained by removing the misclassifications *E* from *D*.

At Step 5, if |CT| = |*D*|, then the first CC fits in *D*, CCC does not build the second CC for classifying training errors. Otherwise, the algorithm classifies those misclassifications contained in *E* from Steps 6 to 7.

At Step 6, addClasses() is used for adding artificial class labels to the original domain *D*, and re-label those correct classifications obtained at Step 4 to the corresponding additional classes in terms of their original class labels. In the last phase, at Step 8, the learning algorithm defines a CCC classifier, which is an ensemble learner containing the resulting CC classifiers in *B*, with a modified averaged combination schema for decisions, where *C* is a set of original class labels while *C*′ is a set of original class labels and artificial class labels.

```
addClasses algorithm
input   S: subdomain;
        i: beginning index
        D: original domain
output  D':  a new domain with additional classes
begin
1  L = ∅, L'= ∅, j = 0, k = i
2  foreach p ∈ S
3     c = p.classLabel
4     if(c ∉ L)
5          L[j] = c      // current class label
6          L'[j] = c.k   // c.k is a new class label
7          p.classLabel = L'[j]
8          k++; j++
9     else
10         p.classLabel = L'[j'], where L[j'] = c
11 addClassLabels(L', D)
end.
Proc addClassLabels(L', D)
begin
12   i = 0; k = |D.classes|
13   foreach c = L'[i]
14      D.classes[k + i] = c
15      i++
end
```

Figure 2. addClasses Algorithm in CCC Algorithm.

The CCC induction algorithm defines additional classes on a training set for building cascaded CC classifiers. These additional classes help define the hyperplanes of a CC classifier to classify its sub-domain. Additional classes are defined by invoking the procedure addClasses(), as shown in Figure 2, where input *i* is used as an indicator variable for defining new additional classes.

The addClasses procedure from Step 2 to Step 10 re-labels each instance *p* in *S* with a new additional class label *c.k*. But *p* will be re-labelled with the same label if the current label of *p* is the same as the current label of another instance. L records all the current labels and L′ records all the new additional class labels. Finally, at Step 11, the procedure extends the list of original class labels of D with new additional class labels.

saveLabels() at Step 1 and restoreLabels() at Step 9, as shown in Figure 1, can be simply implemented in linear time complexity. Also, addClasses() adds additional class labels into D for those correctly classified examples in CT in a linear time complexity. A base learner NB is a traditional induction algorithm, which has a linear training time O(*mn*)[6]. CCC has the time complexity O(*kmn*) for misclassifications, where *k* is the number of the original classes. Therefore, the CCC has a linear time complexity O(*kmn*) for training.

Because CCC changes the class labels of examples for training cascaded CC classifiers, it only requires an extra space O(*n*) for saving the original class labels of examples. Therefore, a customized NB classifier requires the space complexity of O((*k*+*k*′)*mv*+*n*) for its parameters and the original labels of examples during the training time, where *k*′ is the number of additional classes. The space complexity of the resulting CCC with a base learner NB for a nominal case is O((*k*+*k*′)*mv*) while the space complexity for a continuous case is O((*k*+*k*′)*m*).

3.4 An Example

Given a dataset V with two classes, i.e., the square class (minority), denoted as '□', and the diamond class (majority), denoted as '◊', as shown in Figure 3(a), we show how CCC works on this synthetic dataset.

CCC with a base NB builds its first classifier. CCC runs the learned NB to correctly classify most examples, which are assigned to a minus class, denoted as a '−', and a solid dot class, denoted as '•', for the original diamond class and the original square class, respectively, as shown in Figure 3(b). The learner also misclassifies a few diamond examples and a few square examples. As we can see, the NB classifier as a linear classifier divides the original domain with a straight line.

CCC continues by building the second classifier and classifying the remaining misclassifications on the new sub-domain after all correct classifications are re-labeled with additional class labels, i.e., '−' and '•' for those classified data points belonging to the diamond class and the square class, respectively.

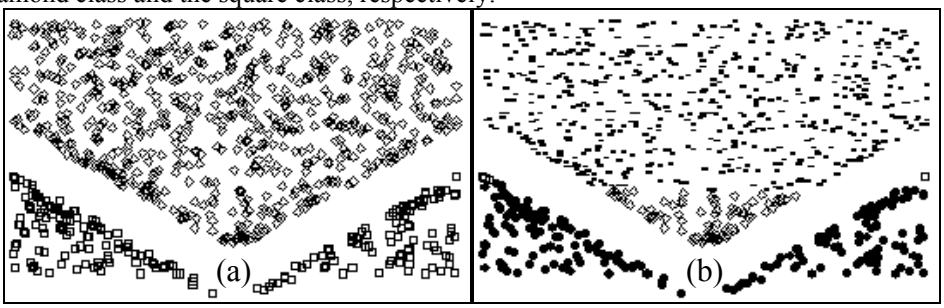

Figure 3. An Example of CCC with a base NB.

Finally, the resulting CCC is composed of two CC classifiers: the first one is can be regarded as the level-0 model outputting correct classifications and misclassifications instead of probability vectors in the traditional CL while the second one can be regarded as the level-1 model built on the customized domain. The combination rule defined at Steps 8, 8.1, 8.2, and 8.3 in CCC consists of two normalizations. Clearly, this new strategy is quite different from the traditional CL, Bagging, and AdaBoost techniques.

5 Experiments

We conducted experiments to evaluate the proposed new Meta learner CCC on 33 datasets. 32 benchmark datasets are chosen from the UCIKDD repository [12] and another one is a synthesized dataset, which is obtained from a scientific application [21]. The scientific application is described as follows: a possible method of explosion detection for the Comprehensive nuclear-Test-Ban-Treaty [21][22] consists of monitoring the amount of radioxenon in the atmosphere by measuring and sampling the activity concentration of Xe-131m, Xe-133, Xe-133m, and Xe-135 by radionuclide monitoring. Several samples are synthesized under different circumstances of nuclear explosions, and combined with various measured levels of normal concentration backgrounds to synthesize a training dataset, which is called Explosion, for use with machine learning methods. The characteristics of these datasets are described in Table 1, where the columns are the names of the datasets, the number of attributes (#attr), the number of instances (#ins), the number of classes (#c).

To justify the proposed Meta classifier CCC to scale up NB, we chose four Meta learning algorithms for scaling up Naïve Bayes (NB) [18], i.e., Stacking [28], AdaBoost [9], MultiBoostAB [25], and Bagging [2]. These induction algorithms are chosen from the Waikato Environment for Knowledge Analysis (Weka) tools [27].

NB is set with Gaussian Estimator for continuous values and Maximum Likelihood Estimator for nominal values; Stacking is set with 10 folds and NB as the base learner and Meta learner, thus denoted as SNB; AdaBoost runs with the base learner NB and 10 models, thus denoted as BNB; MutilBoostAB is set with 10 iterations and the base learner NB, thus denoted as MNB; Bagging is set with 10 iterations and the base learner NB, thus denoted as

Table 1. The characteristics of datasets.

Datasets	#attr	#ins	#c	Datasets	#attr	#ins	#c
Anneal	39	898	5	Lymph	19	148	4
Audiology	70	226	24	Mushroom	23	8124	2
Autos	26	205	6	P-tumor	18	339	21
Balance-s	5	625	3	Segment	20	2310	7
Breast-w	10	699	2	Sick	30	3772	2
Colic	23	368	2	Sonar	61	208	2
Credit-a	16	690	2	Soybean	36	683	18
Diabetes	9	768	2	Splice	62	3190	3
Glass	10	214	6	Vehicle	19	846	4
Heart-s	14	270	2	Vote	17	435	2
Hepatitis	20	155	2	Vowel	14	990	11
Hypothyroid	30	3772	4	Waveform	41	5000	3
Ionosphere	35	351	2	Zoo	18	101	7
Iris	5	150	3	Adult	15	48842	2
kr-vs-kp	37	3196	2	Shuttle	10	58000	7
Labor	17	57	2	Explosion	5	92630	2
Letter	17	20000	26				

BgNB. Other parameters of these learning algorithms are set with their default settings.

To compare the CCC with previous approaches to scale up NB, we also chose five algorithms to build the corresponding classifiers, i.e., SBC, TAN, NBTree, AODE, and HNB. Because the ODE classifiers such as AODE and HNB only can work on nominal cases, numeric attributes are discretized and missing values are replaced by using the unsupervised methods Discretize and ReplaceMissingValues in Weka, respectively, before experiments can be done.

Experiments were conducted by 2 runs of the 10-fold cross validation. For each round, classifiers were built on 9 folds, and were tested on the holdout fold. The process was repeated 10 times, and the results were averaged. We used the paired t-test and the Wilcoxon signed rank test for significance testing on the results (accuracies) from two classifiers. The Wilcoxon signed-rank test or Wilcoxon test is a non-parametric statistical hypothesis test for two repeated measurements under the assumption of independence of the differences. It is an alternative to the paired t-test when these measurements cannot be assumed to be normally distributed. Wilcoxon test is used for single training set rather than multiple datasets [4].

We first conducted experiments to compare CCC with the selected Meta learners to scale up NB. The results on first 30 benchmark datasets are shown in Table 2, where the columns of the corresponding approaches are followed by additional columns, which depict statistical test results ‘w ‘or ‘l’; ‘w’ represents a win of CCC against the corresponding approach while ‘l’ represents a loss of CCC against the corresponding approach; otherwise, both approaches are tied; the averaged accuracies are also reported in the bottom of the table.

As we can see, CCC can improve NB in many cases with respect to the paired t-test. CCC only degrades NB on P-tumor with respect to the Wilcoxon test. CCC also outperforms other Meta learners in most cases. In particular, CCC does not lose to MNB and Bagging in any case with respect to the paired t-test. We summarize all the results of the paired t-test, as shown in Table 3, where ##/##/## represents the numbers of win, tie, and loss of CCC over other Meta learning approaches, respectively. As we can see from the row ‘CCC’, it is clearly shown that CCC outperforms other Meta approaches to scale up NB.

Experimental results on three large datasets are shown in Table 4. The results show that CCC degrades NB and is inferior to other Meta learners only on the Adult case. In Explosion, CCC is tied with the other Meta learners due to a large variance in their results with respect to the paired t-test although the accuracy of CCC is higher than that of other Meta learners. We further analyze the Adult case, where the class distribution is 11687:37155. Our experiment is also to calculate the True Positive Rate (TPR) of the minority class. As a result, those TPR from CCC, NB, SNB, BNB, MNB, BgNB are 0.8117, 0.5116, 0.534, 0.5116, 0.5138, and 0.511, respectively. This shows that CCC can more precisely classify the minority class than other Meta learners on this case. We emphasize that this is rational in many practical applications [3].

We compared CCC with previously proposed approaches such as SBC, TAN, AODE, and HNB, which scale up NB by alleviating the conditional independence assumption. As mentioned before, experiments were done after numeric attributes were discretized with the Discretize method and missing values were replaced by using the ReplaceMissingValues method in Weka. Some experimental results are reported in Table 5.

As we can see, CCC is very competitive with these approaches to scale up NB in these cases although CCC might lose to these approaches (other approaches, e.g., AODEsr [30] and WAODE [14], are omitted without loss of generality) in other cases (omitted due to space limitation). In the Splice case, SBC, TAN, and HNB unexpectedly failed to scale up NB. CCC is more successful than SBC, TAN, and HNB in this case if the uncorrelated attribute ‘Instance Name’ remains in the training set [29]. The results in Table 5 are quite attractive in practical applications because CCC only builds two individual NB models.

CCC is much more efficient and has much less space demand than previously proposed approaches including recent ODE classifiers such as AODE and HNB.

Table 2. Performance (Accuracy) of CCC, NB and three Meta classifiers on 30 benchmark datasets. Those strings such as 'ww' represent the results of the corrected paired t-test (first) and Wilcoxon rank test (second), respectively. '-' is the case for tie.

Datasets	CCC	NB		SNB		BNB		MNB		BgNB	
Anneal	95.43	86.41	ww	35.42	ww	93.65	-w	87.92	ww	86.75	ww
Audiology	72.13	72.12		60.86	ww	79.24	ll	72.12		72.12	
Autos	63.87	54.94	ww	38.79	ww	55.40	-w	59.55		56.64	-w
Balance-s	90.71	90.71		91.67		90.63		90.87		90.31	
Breast-w	96.07	96.14		96.14		95.35	-w	96.07		96.21	
Colic	83.00	78.61	-w	79.02	-w	77.04	ww	79.72	-w	78.48	-w
Credit-a	81.74	78.04	ww	78.41	ww	81.74		79.06	ww	78.12	ww
Diabetes	75.39	75.46		75.39		76.04		76.24		75.72	
Glass	48.85	47.42		28.94	ww	47.42		47.42		48.15	
Heart-s	84.81	84.26		84.44		83.52		84.63		84.07	
Hepatitis	84.60	83.00		83.31		83.90		84.63		83.65	
Hypothyroid	97.26	95.27	ww	94.07	ww	95.27	ww	95.39	ww	95.37	ww
Ionosphere	92.32	83.05	ww	83.34	ww	91.18		91.03		82.48	ww
Iris	94.67	95.00		95.00		95.67		95.67		95.67	
kr-vs-kp	94.79	87.81	ww	88.08	ww	94.82		89.57	ww	87.75	ww
Labor	93.67	94.67		93.83		93.00		95.50		95.50	
Letter	67.44	64.04	ww	56.16	ww	64.04	ww	64.55	ww	64.11	ww
Lymph	80.02	82.36		72.98	-w	79.40		82.36		82.71	
Mushroom	99.62	95.78	ww	96.98	ww	100.00	ll	99.59		95.78	ww
P-tumor	46.91	49.41	-l	26.69	ww	49.41	-l	49.41	-l	48.37	
Segment	85.19	80.30	ww	82.88	ww	80.30	ww	80.39	ww	80.41	ww
Sick	96.55	92.92	ww	92.70	ww	93.53	ww	93.23	ww	92.66	ww
Sonar	79.61	69.04	ww	69.27	ww	80.58		75.99	-w	69.74	ww
Soybean	92.38	92.90		92.46		91.88		93.04	-l	92.68	
Splice	95.91	95.49	-w	95.38	ww	93.97	ww	95.27	ww	95.44	-w
Vehicle	50.94	45.68	ww	45.74	ww	45.68	ww	45.68	ww	45.27	ww
Vote	93.56	90.22	ww	90.22	ww	95.97	ll	91.38	-w	90.11	ww
Vowel	73.23	63.43	ww	65.00	ww	79.75	ll	69.29	-w	63.84	ww
Waveform	80.64	79.96	ww	82.37	ll	79.96	ww	80.32	-w	79.99	ww
Zoo	97.05	95.09		95.55		97.05		97.55		95.09	
Average	**82.95**	79.98		75.70		82.18		81.45		80.11	

Table 3. Summary of paired t-test.

	NB	SNB	BNB	MNB	BgNB
SNB	6/21/3				
BNB	1/21/8	3/13/14			
MNB	0/23/7	2/17/11	6/22/2		
BgNB	0/30/0	3/21/6	8/21/1	6/24/0	
CCC	15/15/0	18/11/1	8/18/4	9/21/0	14/16/0

Table 4. Performance of CCC, NB, and other Meta classifiers on three Large datasets.

Dataset	CNB	NB		SNB		BNB		MNB		BgNB	
Adult	81.80	83.25	ll	83.34	ll	83.25	ll	83.29	ll	83.23	ll
Shuttle	96.03	93.01	ww	33.01	ww	92.82	ww	93.20	ww	92.93	ww
Explosion	99.15	91.02		99.68		91.02		91.02		91.12	
Average	89.98	86.82		72.93		87.32		87.24		86.85	

Table 5. Performance (Accuracy) of CCC, NB, and previous approaches for improving NB.

Datasets	CNB	NB		SBC		TAN		NBTree		AODE		HNB	
Balances	91.12	91.20		91.20		87.28	ww	91.20		89.84	-w	90.01	
Breastw	97.21	97.28		96.57	ww	94.64	ww	97.21		96.93		96.21	
Colic	81.91	79.17	-w	83.82		80.02	-w	80.02		80.95		81.08	
Credita	85.29	84.71		84.42		85.00		85.07		86.01		84.93	
Diabetes	75.66	76.04		76.76		75.07		75.33		76.95		76.69	
Hearts	84.81	84.63		80.00	ww	78.89	-w	82.59	ww	83.89	-w	82.41	
Hepatitis	84.25	83.60		82.27		84.54		80.73	ww	84.56		82.29	
Iris	95.33	94.67		96.67		91.00	ww	95.00		95.33		93.00	-w
krvskp	94.79	87.81	ww	94.34		92.10	ww	98.26	ll	91.27	ww	92.27	ww
Labor	98.17	98.17		82.67	ww	90.17	-w	97.33		94.67		92.83	-w
Lymph	84.38	84.02		77.62	ww	85.48		83.07		86.38		82.69	
Splice	95.91	95.49	-w	52.57	ww	52.57	ww	95.49		96.00		59.11	ww
Vote	93.78	90.22	ww	95.63	-l	94.71		94.60		94.48		94.25	
Zoo	95.05	94.05		91.59		94.09		94.59		94.55		98.05	
Average	89.83	88.65		84.72		84.68		89.32		89.42		86.13	

6 Conclusion and future work

It has been observed that previously proposed Meta learning techniques, Bagging, AdaBoost, and MultiBoostAB, failed to scale up Naïve Bayes (NB) due to the stability of NB. In this paper, we propose a new Meta learning technique to improve NB. This technique is different from recent research which focuses on One-Dependent Estimation (ODE) techniques such as AODE and HNB. The new Meta learner adopts the Domain-based Cascade Learning (DCL), which is regarded as an altered strategy for traditional Cascade Learning for building diverse NB models. We propose the Cascading Customized Couple (CCC) algorithm, which only builds a couple of NB. Our analysis and experimental results show that CCC is more successful than the previously proposed Meta learning techniques used to scale up NB, and more efficient than those ODE techniques. CCC is also very competitive with the ODE techniques in some cases. This is very attractive in practical applications such as text classification in that one is confronted with large datasets.

Because it is observed in Table 2 that traditional Meta learning techniques such as AdaBoost can be more successful than CCC in some cases, it is suggested that CCC might be further enhanced by incorporating with the Boosting techniques for Bayesian learning.

References

[1] E. Baur and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105-139, 1999.

[2] L. Breiman. Bagging Predictors. *Machine Learning*, 24(3):123-140, 1996.

[3] N. V. Chawla, N. Japkowicz, and A. Kolcz. Editorial to the special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations*. 6(1):1-6, 2004.

[4] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1-30, 2006.

[5] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the sample Bayesian classifier. *Machine Learning*, 29:103-130, 1997.

[6] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. A Wiley Intersience Publication (2000).

[7] C. Elkan. *Boosting and Naïve Bayesian Learning*. Technical Report CS97-557, University of California, Davis, 1997.

[8] S. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems* 2, D. Touretzky, Ed. San Mateo, CA: Morgan Kaufman, 2:524–532, 1990.

[9] Y. Freund and R. E. Schapire. A short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, 1999.

[10] N. Friedman, D. Geiger, and M. Goldszmith. Bayesian network classifiers. *Machine Learning* 29: 131-163, 1997.

[11] J. Gama, P. Brazdil. Cascade generalization. *Machine Learning* 41:315-343, 2000.

[12] S. Hettich and S. D. Bay. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science. 1999.

[13] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive Mixtures of Local Experts. In *Neural Computation*, 3: 79-97, 1988.

[14] L. Jiang and H. Zhang. Weightily Averaged One-Dependence Estimators. In: *Proceedings of the 9th Biennial Pacific Rim International Conference on Artificial Intelligence*, 970-974, 2006.

[15] M. Jordan and R. Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation* 6:181-214, 1994.

[16] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International conference on Knowledge Discovery and Data Mining*, 202-207, 1996.

[17] P. Langley. Induction of recursive Bayesian classifiers. In *Proceedings of the 8th European Conference on Machine Learning*, 153-164, 1993.

[18] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence*, 223-228. AAAI Press and MIT Press, 1992.

[19] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 399-406. Morgan Kaufmann, 1994.

[20] J. Rennie, L. Shih, J. Teevan, D. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In *Proceedings of International Conference on Machine Learning*, 616-623, 2003.

[21] T. J. Stocki, X. Blanchard, R. D'Amours, R. K. Ungar, J. P. Fontaine, M. Sohier, M. Bean, T. Taffary, J. Racine, B. L. Tracy, G. Brachet, M. Jean, and D. Meyerhof. Automated radioxenon monitoring for the comprehensive nuclear-test-ban treaty in two distinctive locations: Ottawa and Tahiti. J. *Environ.Radioactivity* 80:305-326, 2005.

[22] J. D. Sullivan. The comprehensive test ban treaty. *Physics Today* 151, 1998.

[23] K. Ting and Z. Zheng. A study of Adaboost with naive Bayesian classifiers: weakness and improvement. *Computational Intelligence*, 19(2):186-200, 2003.

[24] K. Ting and I. Witten. Issues in Stacked Generalization. *Journal of Artificial Intelligence Research 10*, 271-289, 1999.

[25] G. I. Webb. MultiBoosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159-196, 2000.

[26] G. I. Webb, J. Boughton, and Z. Wang. Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, 58(1):5-24, 2005.

[27] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[28] D. Wolpert. Stacked generalization. In *Neural Networks*, 5:241-260, 1992.

[29] H. Zhang, L. Jiang, and J. Su. Hidden Naive Bayes. In: Twentieth National Conference on Artificial Intelligence, 919-924, 2005.

[30] F. Zheng, G. I. Webb. Efficient lazy elimination for averaged-one dependence estimators. In *Proceedings of the 23th International Conference on Machine Learning*, 1113-1120, 2006.