Distribution-Balanced Stratified Cross-Validation for Accuracy Estimation

Xinchuan Zeng and Tony R. Martinez
Computer Science Department, Brigham Young University, Provo, Utah 84602
Email: zengx@axon.cs.byu.edu, martinez@cs.byu.edu

Abstract

Cross-validation has often been applied in machine learning research for estimating the accuracies of classifiers. In this work, we propose an extension to this method, called *distribution-balanced stratified cross-validation* ($DBSCV$), which improves the estimation quality by providing balanced intraclass distributions when partitioning a data set into multiple folds. We have tested $DBSCV$ on nine real-world and three artificial domains using the $C4.5$ decision trees classifier. The results show that $DBSCV$ performs better (has smaller biases) than the regular stratified cross-validation in most cases, especially when the number of folds is small. The analysis and experiments based on three artificial data sets also reveal that $DBSCV$ is particularly effective when multiple intraclass clusters existi n a data set.

1 Introduction

In supervised learning, a classifier (e.g., decision tree, neural network) is constructed by using a learning algorithm based on a labeled training set. The performance of the constructed classifier is usually measured by its *generalization accuracy*, or *true accuracy* – the probability of correctly classifying unseen cases which are to be presented to the classifier in future.

The true accuracy, however, can not be evaluated directly, since we do not have available an independent and labeled data set of an arbitrarily large size (in practice) for testing the classifier. In general, there is a data set of a limited (often small) size available, and we have to estimate the true accuracy based on this data set. That is, both the construction of the classifier and the estimation of its true accuracy rely on the same data set.

A typical strategy for estimating the true accuracy is to divide the original data set into two subsets: training set and test set. The training set is applied to train a classifier and the test set is used to evaluate the accuracy of the classifier. This process is usually repeated multiple times (with different random partitions of the data set into training and test sets), and the average gives an estimation of the true accuracy. In machine learning research, several different estimation methods have been studied and applied (Breiman et al. 1984, Efron and Tibshirani 1993, Jain et al. 1987, Kohavi 1995, Weiss 1991, Weiss and Indurkhya 1994). These methods differ from each other in their strategies on how to partition the original training set and on how to combine the results of multiple partitions. Among them, the most common ones include standard *training-testing*, *bootstrap*, and *cross-validation*.

Applying a reliable method to estimating the true accuracy is important for building a classifier. For example, when training a neural network without *a priori* knowledge about the optimal architecture for the network, one way to choose the number of hidden nodes is to experiment with several different values, estimate the accuracy for each value, and then choose the one with the highest estimated accuracy. In this case, the architecture and performance of the network are directly related to the quality of the estimation method applied.

A reliable estimation method is also essential for selecting a classifier from several available choices. For example, if we have available a set of classifiers, (e.g., decision tree, neural network, nearest neighbor rule), which have been trained for a particular task, how should we choose the one with the best performance? We will rely on the estimated true accuracies of these classifiers to decide which one to choose. Obviously, unreliable estimations may lead to choosing a classifier that does not give the best performance.

Much work has been done to evaluate the performance of difference estimation methods, especially on small samples. Breiman et al. (1984) applied cross-validation for decision tree pruning in the CART program, and found that using stratified 10-fold cross-validation was satisfactory for choosing the correct tree. Jain et al. (1987) compared bootstrap and leave-one-out cross-validation on nearest neighbor classifiers and found that bootstrap has smaller confidence intervals than leave-one-out. Weiss (1991) studied stratified cross-validation and bootstrap method for nearest neighbor classifiers and the results showed that stratified 2-fold cross-validation has lower variance and is superior to leave-one-out. Weiss and Indurkhya (1994) evaluated the performance of cross-validation for decision tree pruning and showed that stratified 10-fold cross-validation can often yield unbiased trees. Dietterich (1998) studied five approximate statistical tests for comparing different learning algorithms, and proposed a test method called $5x2cv$, based on 5 iterations of 2-fold cross-validation. The author recommends applying $5x2cv$ when only small number of testing runs is used.

Kohavi (1995) compared several accuracy estimation methods empirically, based on a large number of simulations on some real and artificial data sets. The results showed that the cross-validation method performs better than other methods in terms of the bias and variance of the estimated accuracies. Kohavi also reported that *stratification* can further improve the performance of the cross-validation method and is able to achieve lower biases and smaller variances in estimated accuracies.

In this work, we propose an approach to extend and improve the stratified cross-validation method. This approach, called distribution-balanced stratified cross-validation ($DBSCV$), partitions a data set into n folds in such a way that a balanced distribution in feature space is maintained for each class, in addition to stratification. For instance, if there are multiple clusters within a class, $DBSCV$ would uniformly partition each cluster into n folds. In contrast, a regular stratified cross-validation ($RSCV$) randomly partitions all cases of a class into n folds without considering intraclass distribution balance. A potential problem with $RSCV$ is that it may cause uneven partitions in terms of intraclass clusters, for example, one fold may contain more cases than other folds within a cluster. This undesirable distortion in distribution may affect the quality of estimated accuracies.

We have tested $DBSCV$ on several real-world and artificial data sets. The results show that, for most of the real-world data sets, the estimated accuracies using $DBSCV$ have lower biases compared to those using $RSCV$, especially when the number of folds is small. Further experiments on three artificial data sets demonstrate that the enhancements of $DBSCV$ are most obvious for data sets containing multiple intraclass clusters in the feature space.

The rest of the paper is organized as follows. Section 2 briefly outlines several commonly used accuracy estimation methods. Section 3 describes the new accuracy estimation method. Section 4 presents simulation results, and Section 5 summarizes the work.

2 Accuracy estimation methods

The most common parameter for measuring the performance of a classifier is its true accuracy, and it has to be estimated based on the original data set S. Often the size of S is quite small (in the order of several hundreds). This may be due to limited availability of the data source or high expense of collecting data. It is more challenging to make an accurate estimation to the true accuracy for a smaller data set. An estimated accuracy often deviates from the true accuracy due to a high variance and/or bias. The quality of an estimation also depends heavily on the methodology adopted. In the following we will briefly describe several commonly used methods: standard *training-testing*, *bootstrap*, and *cross-validation*.

Standard training-testing

In the standard *training-testing* method, the original data set S is randomly partitioned into two disjointed subsets, S_1 and S_2 (usually with 2/3 data in S_1 and the rest 1/3 in S_2). A classifier is then constructed using S_1 as the training set, and its accuracy is evaluated using S_2 as the test set. This process repeats N times (*e.g.*, 25 times), with a different random partition of the data set each time, and the true accuracy is estimated by averaging over these N iterations.

This method usually performs well for large data sets but poorly (has larger bias) for small data sets. One of the main reasons is that only a moderate portion (2/3) of the available data is used in training the classifier.

The consequence of excluding 1/3 of the available data S_2 is that S_1 becomes less representative of the original data set S, especially for small data sets. This usually leads to an inaccurate estimation which is usually lower than the true accuracy. Another source of inaccuracy by this method is that the test sets in different partitions are not disjoint, which in principle violates the requirement for the independence of testing cases.

Bootstrap

The *bootstrap* method (Efron and Tibshirani 1993), widely studied and applied in statistics, generates N (e.g. $N = 50$) subsets (S_1, S_2,... S_N) from the original data set S using a resampling technique.

To generate S_i from the data set S which has M patterns, select a pattern e randomly from S, add e to S_i, and then put a copy of e back into S (with *replacement*). This selecting process is repeated M time (then S_i would also include M patterns). Note that some patterns in S_i may have multiple copies while some may have none.

To build a classifier P_i, S_i is used as the training set. The test set T_i includes all the patterns that appear in S but not in S_i. The true accuracy is estimated by averaging over the accuracies of P_i ($i = 1, 2...N$). There have been several variants for the calculation of the true accuracy (Jain et al. 1987, Weiss 1991, Weiss and Indurkhya 1994).

Cross-validation

In a typical (n-fold) cross-validation method, a data set S is first randomly partitioned into n equally-sized, disjoint subsets (folds) S_1, S_2, ..., S_n. Each of the n folds is then in turn used as the test set, while the remaining (n-1) folds are used as the training set. A classifier is then constructed from the training set, and its accuracy is evaluated on the test set. This process repeats n times, with a different fold used as the test set each time. The estimated true accuracy by this method is the average over the n folds. One distinct feature of cross-validation is that all the n test sets are disjoint, and thus each case in the original training set is tested once and only once. In contrast, when using the standard training-testing or bootstrap methods, some cases might be tested multiple times while some others might not be tested at all.

An extension of regular cross-validation is *stratified* cross-validation (Breiman et al. 1984). In n-fold stratified cross-validation, a data set S is partitioned into n folds such that each class is uniformly distributed among the n folds. The result is that the class distribution in each fold is similar to that in the original data set S. In this sense, the partition is "balanced" in terms of class distributions. In contrast, regular cross-validation randomly partitions S into n folds without considering class distributions. A possible scenario with regular cross-validation is that a certain class could be distributed unevenly – some folds contain more cases of the class than other folds. This distortion in class distributions can cause a less reliable accuracy estimation. Kohavi (1995) reported that stratified cross-validation performs better (has smaller bias and variance) than regular cross-validation.

3 Distribution-balanced stratified cross-validation

The distribution-balanced stratified cross-validation ($DBSCV$) algorithm first partitions the initial training set S of size N into k folds using the distribution-balanced stratified partition ($DBSP$) algorithm, and then applies regular cross-validation to build classifiers and estimate the accuracy.

The pseudocode of the algorithm $DBSP$ is given in Fig. 1, and further explanations are provided in the following.

The purpose of *Step (1)* is to initialize a reference feature vector $\mathbf{X_0}$ for building a sorted list L_i for each class.

Step (2) sorts the cases of each class C_i into a list L_i in such a way that each case in L_i is followed by its "nearest neighbor" in S_i. After sorting, those cases near to each other in the feature space (e.g., in the same cluster) will also be near to each other on the sorted list L_i. The distance $D(\mathbf{X_i^{(m-1)}}, \mathbf{X_{i,n}})$ in *Step (2)* is calculated using the following measure: (1) for continuous attributes, the Euclean metric is applied; (2) for nominal attributes, the distance is 0.0 if the two nominal values are same and 1.0 if different.

Based on the sorted list L_i, *Step (3)* partitions each L_i into k folds such that the cases near to each other on the list are partitioned into different folds. For instance, if $k = 3$, cases (1, 4, 7, ...), (2, 5, 8, ...), and (3, 6, 9, ...) are partitioned into *fold 1* , *fold 2*, and *fold 3* respectively [1].

Using the strategy of the algorithm $DBSP$, the cases distributed in a same cluster of a particular class will be (approximately) uniformly partitioned into the k folds, avoiding a potentially unbalanced partition when using the regular stratified partition.

After partitioning the initial training set S by the algorithm $DBSP$, the rest of the ($DBSCV$) procedure follows the standard steps of k-fold cross-validation.

4 Experiments

We have conducted experiments to evaluate three versions of k-fold cross-validation: (i) regular cross-validation (RCV); (ii) regular stratified cross-validation ($RSCV$), and (iii) distribution-balanced stratified cross-validation ($DBSCV$). We tested these algorithms on nine real-world data sets drawn from the UCI data repository (Merz and Murphy 1996), which properties are shown in Fig. 2. These sets reflect diverse types of domains, and most of them (except **tic-tac-toe**) have sizes equal to or larger than 1000 (many sets in the UCI data repository are relatively small). The reason for choosing a large set is to allow a more reliable estimation on the true accuracy. We also tested the algorithms on three artificial data sets, with 1, 2, and 8 intra-class clusters respectively, in order to study the cluster-dependence of the performance.

Experimental procedure

The classifiers used in the experiments are decision trees constructed by the $C4.5$ algorithm (Quinlan 1993). All the results reported in the following are based on the *pruned* version of the decision trees.

For a particular data set S_0 of size N_0, a sample S_1 of a fixed size N_1 is drawn randomly from S_0. S_1 is then used as the "original" training set (corresponding to S described in the last section) for evaluating the algorithms. The remaining part S_2 of size N_2 (with $S_1 \cup S_2 = S_0$, $N_1 + N_2 = N_0$) is applied as the "true accuracy" testing set.

The estimated accuracies of the three cross-validation algorithms are obtained in the following way. For data set S_0, a sample S_1 of size N_1 is first drawn randomly from S_0, and then each of the three algorithms is applied to the sample to give one estimated accuracy: $ACC^{(i)}_{RCV}$, $ACC^{(i)}_{RSCV}$, and $ACC^{(i)}_{DBSCV}$. This process repeats 100 times (i = 1, 2, ... 100), with a different random sample each time. The estimated accuracy for an algorithm is the average over the 100 iterations: $ACC^{(avg)}_{RCV}$, $ACC^{(avg)}_{RSCV}$, and $ACC^{(avg)}_{DBSCV}$.

We applied the following method to estimate the "true accuracy". A sample S_1 of size N_1 is drawn randomly from data set S_0. A classifier is built using S_1 and its accuracy $ACC_{TRUE}^{(i)}$ is tested on S_2 (= $S_0 - S_1$, as explained above). This process is repeated $1,000$ times (with a different random sample each time), and the average accuracy $ACC_{TRUE}^{(avg)}$ gives the estimation of the "true accuracy". A large number $1,000$ is chosen in order to get a more reliable estimation, i.e., a smaller confidence interval.

Results on real-world data sets

Fig. 3 show the simulation results on the nine real-world data sets drawn from the UCI repository with sample size $N_1 = 300$. A small sample size $N_1 = 300$ is used for all the results shown in Fig. 3, because we are mainly interested on comparing the performance of different estimating methods for small samples. Discussion on the effects of different sample sizes is given later. Since we choose the nine data sets with large sizes, the size N_2 of the set S_2 for estimating the "true accuracy" is also large, allowing a more reliable estimation for the "true accuracy". As an example, for data set **page-blocks**, $N_0 = 5473$, $N_1 = 300$, and thus $N_2 = 5473 - 300 = 5173$.

Each graph in Fig. 3 compares the distribution balanced stratified cross-validation ($DBSCV$) to the regular stratified cross-validation ($RSCV$) method in terms of how close they are to the "true accuracy" ($TRUE$). We can see that for most data sets, $DBSCV$ is closer to $TRUE$ (i.e. has a smaller bias) than $RSCV$ [4].

The improvement of $DBSCV$ over $RSCV$ is most evident when the number of folds is small (< 6). As the number of folds increases, the performance difference of the two algorithms becomes smaller, but $DBSCV$ still performs better than $RSCV$ in most cases.

The above phenomenon can be explained as follows. When the number of folds k becomes smaller, the fraction of the data set used as the cross-validation training data $S_1^{(train)}$ also becomes smaller as compared to the whole sample S_1. For example, $S_1^{(train)}$ is $2/3$ of S_1 when $k = 3$, and is $1/2$ of S_1 when $k = 2$. As a result of the reduction in training data, a classifier constructed on $S_1^{(train)}$ is less accurate than the one constructed on the whole sample S_1. This explains the trend of larger bias in estimated accuracies as k becomes smaller, as shown in Fig. 3.

The following factors have important impacts on how much the reduction in training data can affect the bias of estimated accuracy. One factor is the size N_1 of the sample S_1. As can be expected, when sample size is small, the reduction in already scarce training data has a more serious impact on the estimated accuracy. This effect is obvious in our experiment because the sample size under study is relatively small (300).

Another factor is how to divide S_1 into k folds. If S_1 is divided in such a way that the content (e.g., attribute and class distributions of training cases) in $S_1^{(train)}$ is similar to that in S_1, the accuracy difference of the classifier trained on $S_1^{(train)}$ and that trained on S_1 will be smaller; otherwise, there will be a larger difference in accuracy, which causes a larger bias.

One way to keep a balanced partition of S_1 into k folds is to apply the $RSCV$ method, which uses stratifications to balance the partitions in terms of class distribution among the k folds. Several studies has shown that stratification can improve the performance (smaller bias) of the cross-validation method (Breiman et al. 1984; Weiss 1991; Kohavi 1995). Similar improvement was also observed in our experiment.

Although $RSCV$ can reduce the estimation bias compared to cross-validation without stratification, it still causes a relatively large bias, especially when the number of folds is small, as shown in Fig. 3. However, the estimated biases using $DBSCV$ are substantially smaller than those using

$RSCV$ for most data sets. $DBSCV$ balances not only class distribution but also feature-space distribution during the k-fold partition. Since each fold allocated by $DBSCV$ is more representative of the original data set than those with the other methods, the training set composed of one or several folds is also more representative and better reflects the original data set. As expected, a training set with this feature allows a more precise (with a smaller bias) estimation of the accuracy for the constructed classifier.

Another observation from Fig. 3 is that as the number of folds (k) increases, both $RSCV$ and $DBSCV$ move towards $TRUE$ (have smaller biases) and their difference becomes smaller. When k becomes larger, $S_1^{(train)}$ becomes more similar to S_1 (e. g., $S_1^{(train)}$ is 14/15 of S_1 when $k = 15$), and therefore it would lead to a smaller bias. However, as shown in Fig. 3, even when k is in the range of moderate or relatively larger values (6 to 15), $DBSCV$ still benefits from the balanced partition and performs better than $RSCV$ (although not as much as when k is smaller).

Note that if k reaches the allowed maximum value – the size (N_1) of the sample S_1 (i.e., N_1-*fold* or *leave-one-out* cross validation), all the three methods (RCV, $RSCV$ and $DBSCV$) become equivalent because there is only one way to partition the data set. Although *leave-one-out* is almost unbiased for accuracy estimation, it has the following two disadvantages. First, *leave-one-out* has a relatively larger variance compared to those using smaller folds, especially for small samples (Efron and Tibshirani 1993, Kohavi 1995). For an estimation method, the variance is another source of causing unreliable estimations, in addition to the bias. Second, *leave-one-out* is expensive in computation: a classifier needs to be constructed for each case in the data set, and thus the number of classifiers increases with the size of the data set. This high computation cost can limit the range of its applications.

Due to the limitations of *leave-one-out*, cross-validations with smaller folds (e.g., 2, 10) are often applied in machine learning research. Kohavi (1995) suggested that 10-fold cross-validation be applied even though resources might allow for computing with more folds, due to the consideration of the variance. Weiss (1991) showed that stratified 2-fold cross-validation has lower variance and is superior to leave-one-out for nearest neighbor classifiers. Dietterich (1998) showed the advantages of using the $5x2cv$ method, which is based on 5 iterations of 2-fold cross-validation, for statistical tests when only a small number of testing runs is used. In some applications, cross-validations with smaller folds are preferred due to various practical reasons such as real-time constraints and expensive training of classifiers (e.g., neural networks). For all those situations using a small number of folds, applying $DBSCV$ is particularly beneficial since the improvement by $DBSCV$ over other methods becomes larger when the number of folds becomes smaller, as shown in Fig. 3.

We also conducted experiments to compare the performance of different algorithms for different sample sizes (N_1 smaller and larger than 300). When sample size N_1 becomes smaller, there is a slightly larger amount of improvement by $DBSCV$, further demonstrating the advantage of $DBSCV$ for small samples. However, as the sample size N_1 becomes larger, the amount of improvement by $DBSCV$ becomes smaller. The reason is that when the sample size becomes larger, there are more cases in each fold, and thus $RSCV$ has a better chance to achieve balanced partitions among different folds.

Results on artificial data sets

As discussed earlier, the main reason to propose the $DBSCV$ algorithm is based on the hypothesis that $DBSCV$ can better (with a high degree of balance) partition the data set into k folds in the presence of multiple intraclass clusters. One way to test (at least partially) this hypothesis is as follows. If the hypothesis is valid, then we should observe the following phenomenon: $DBSCV$ is able to improve performance when multiple intraclass clusters exist, while performing the same as $RSCV$ otherwise.

For real-world domains, however, it is difficult to know if there are multiple intraclass clusters, partly due to the high dimensionality of feature spaces. For instance, for the nine real-world domains tested in this research, we do not have any knowledge about their clustering properties. For the data set *pendigits*, we conjecture that there is a certain degree of clustering. The reason is that digits of the same class (e.g., digit "7") were written by different writers multiple times; thus the potential difference in writing style by different writers should form multiple intraclass clusters. The improvement of $DBSCV$ over $RSCV$ on this data set is obvious, as shown in Fig. 3, providing partial support for the hypothesis.

A more direct way to test the hypothesis is to use low-dimensional artificial data sets where the intraclass clusterscan be explicitly enforced. We have constructed three 2-dimensional artificial data sets ($2D1C$, $2D2C$, $2D8C$) with 1, 2, and 8 intraclass clusters respectively. The class distributions of the three data sets are shown in the first row of Fig. 4. The data sets used in the simulation are sampled based on these distributions.

The simulation results of these data sets are shown in the second row of Fig. 4. The sample size N_1 is 300 for those results. The results reveal that when there is only one cluster per class ($2D1C$), both $RSCV$ and $DBSCV$ give similar accuracy estimations, but when there are multiple clusters per class ($2D2C$, $2D8C$), $DBSCV$ performs much better than $RSCV$, providing support for the hypothesis proposed above.

Another observation is that the performance improvement is more obvious on $2D8C$ (8 clusters) than on $2D2C$ (2 clusters), showing that the degree of the improvement by $DBSCV$ increases with the number of clusters.

5 Summary

In summary, we have presented a distribution-balanced stratified cross-validation ($DBSCV$) method for accuracy estimations. $DBSCV$ keeps a better distribution balance in the feature space when partitioning a sample into folds. This property enables the cross-validation training set to better capture the characteristics of the distribution in the original data set. We have tested $DBSCV$ on nine real-word and three artificial data sets. The results show that $DBSCV$ gives better accuracy estimations overall than $RSCV$. The improvement is more obvious when the number of folds becomes smaller.

We have also studied the correlation between the performance of $DBSCV$ and the amount of clustering in data sets. The experiment on three artificial data sets reveals that $DBSCV$ is particularly effective and superior to $RSCV$ when there exist multiple intraclass clusters in a data set.

Based on the results in this work, we recommend $DBSCV$ as a preferred cross-validation method compared to $RSCV$, since $DBSCV$ performs better than $RSCV$ almost uniformly for diverse types of domains and for a wide range of fold k. Applying $DBSCV$ is particularly beneficial in the following cases: (i) there is *a priori* knowledge that multiple intraclass clusters exist in a data set; (ii) a small fold k is required due to resource limitations or due to other considerations (for example, applying $DBSCV$ to the 2-fold cross-validation applied in the work by Weiss (1991) and in the

5x2cv test method proposed by Dietterich (1998)).

In future, we plan to extend this work in the following directions: (i) evaluate the performance of *DBSCV* on different types of classifiers (e.g., neural networks, etc.), in order to gain a better understanding of its behavior under a broader range of environments and its general utility; (ii) apply different distance metrics for characterizing clustering features when partitioning data sets using *DBSCV*.

References

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984). *Classification and Regression Trees*, Wadsworth International Group.

Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7), 1895-1923.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman & Hall.

Jain, A. K., Dubes, R. C. and Chen, C. (1987). Bootstrap techniques for error estimation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI**-9(5), 628-633.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1137-1143.

Merz, C. J., and Murphy, P. M. (1996). UCI repository of machine learning databases. **http://www.ics.uci.edu/~mlearn/MLRepository.html**

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufman, Los Altos, California.

Weiss, S. M. (1991). Small sample error rate estimation for k-nearest neighbor classifiers, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3), 285-289.

Weiss, S. M. and Indurkhya, N. (1994). Decision tree pruning: biased or optimal ?, in *Proceedings of the twelfth national conference on artificial intelligence*, 626-632.

Footnotes:

[1] If the number of cases in L_i is not dividable by k (*e.g.*, 8, 10, 11 in the above example), then it combines the remainings of each L_i into a new list $L^{(r)}$, partitions it into k parts (some parts may be empty), and adds each part to the corresponding fold.

[2] Assume that A_j has been normalized to the range of [0.0, 1.0].

[3] The choice of $X_{0,j}$ for a nominal attribute is arbitrary, because we use the metric in which the distance is 0.0 if two nominal values are the same and 1.0 if different. It is chosen in such a way just for simplicity.

[4] The results for the regular cross-validation (RCV) method (without stratification) are not shown in Fig. 3. These results are worse (have larger biases) than $RSCV$, and their curves are located below the curves of $RSCV$ in most cases.

Algorithm $DBSP$

Input: Training set S of size N with Q classes (C_1, C_2, ... C_Q).

Output: k disjoint subsets (folds) T_1, T_2, ...T_k from the partition of S ($= T_1 \cup T_2 \cup ... \cup T_k$).

Step (1):

Choose an *initial* reference feature vector $\mathbf{X_0}$ (=($X_{0,1}$, $X_{0,2}$, ... $X_{0,M}$), where M is the number of features) in such a way that the $j\underline{th}$ component $X_{0,j}$ is the minimum value of the attribute A_j. For a continuous attribute A_j, the value of $X_{0,j}$ is 0.0 [2]. For a nominal (unordered) attribute A_j, $X_{0,j}$ is any one of the possible values of A_j [3].

Step (2):

Let $\{\mathbf{X_{i,1}}, \mathbf{X_{i,2}}, ...\ \mathbf{X_{i,N_{C_i}}}\}$ be the N_{C_i} cases with the class label C_i (i = 1, 2, ... Q).
For i = 1, 2, ... Q
    Let list $L_i = \{\mathbf{X_i^{(0)}}\}$ where $\mathbf{X_i^{(0)}} = \mathbf{X_0}$, and let set S_i=$\{\mathbf{X_{i,1}}, \mathbf{X_{i,2}}, ...\ \mathbf{X_{i,N_{C_i}}}\}$ initially.
    For m = 1, 2, ... N_{C_i}
        Select vector $\mathbf{X_{i,s}}$ from S_i such that $s = arg\ min_n\{D(\mathbf{X_i^{(m-1)}}, \mathbf{X_{i,n}}) | \forall \mathbf{X_{i,n}} \in S_i\}$
        (i.e., select the case from S_i that is closest to the last case on the list L_i)
        Let $\mathbf{X_i^{(m)}} = \mathbf{X_{i,s}}$ and add $\mathbf{X_i^{(m)}}$ to the end of the list L_i.
        Remove $\mathbf{X_{i,s}}$ from the set S_i.
    End
End

Step (3):

Let all the k fold T_j be *empty* initially (j = 1, 2, ... k).
For i = 1, 2, ... Q
    $s := 0$;
    While $(Size(L_i) \geq k)$
        For j = 1, 2, ... k
            Add $\mathbf{X_i^{(s*k+j)}}$ to the set T_j;
            Remove $\mathbf{X_i^{(s*k+j)}}$ from L_i, and let $Size(L_i) := Size(L_i) - 1$;
        End
        $s := s + 1$;
    End
End
Combine the remaining cases in all lists L_i (i = 1, 2, ... Q) into a new list $L^{(r)}$.
If ($L^{(r)}$ is not *Empty*)
    Partition $L^{(r)}$ (one by one from the begining to the end of $L^{(r)}$) into k parts ($P_1 \cup P_2 \cup ... \cup P_k = L^{(r)}$).
    For i = 1, 2, ... k
        Add P_i into the fold T_i .
    End
End

End

Figure 1. The algorithm $DBSP$.

Data Set	size	#attr	#num	#symb	#class	miss
pendigits	10992	16	16	0	10	no
tic-tac-toe	958	9	0	9	2	no
page-blocks	5473	10	10	0	5	no
chess	3196	36	0	36	2	no
car	1728	6	0	6	4	no
german-numeric	1000	24	24	0	2	no
mushroom	8124	22	0	22	2	yes
hypothyroid	3163	25	7	18	2	yes
wave21	2000	21	21	0	3	no

Figure 2: Description of nine *UCI* data sets. The meaning of symbol in each column: **size**: number of cases; **#attr**: number of attributes (not including class); **#num**: number of continuous attributes; **#symb**: number of nominal attributes; **#class**: number of classes; **miss**: missing attributes (Boolean).

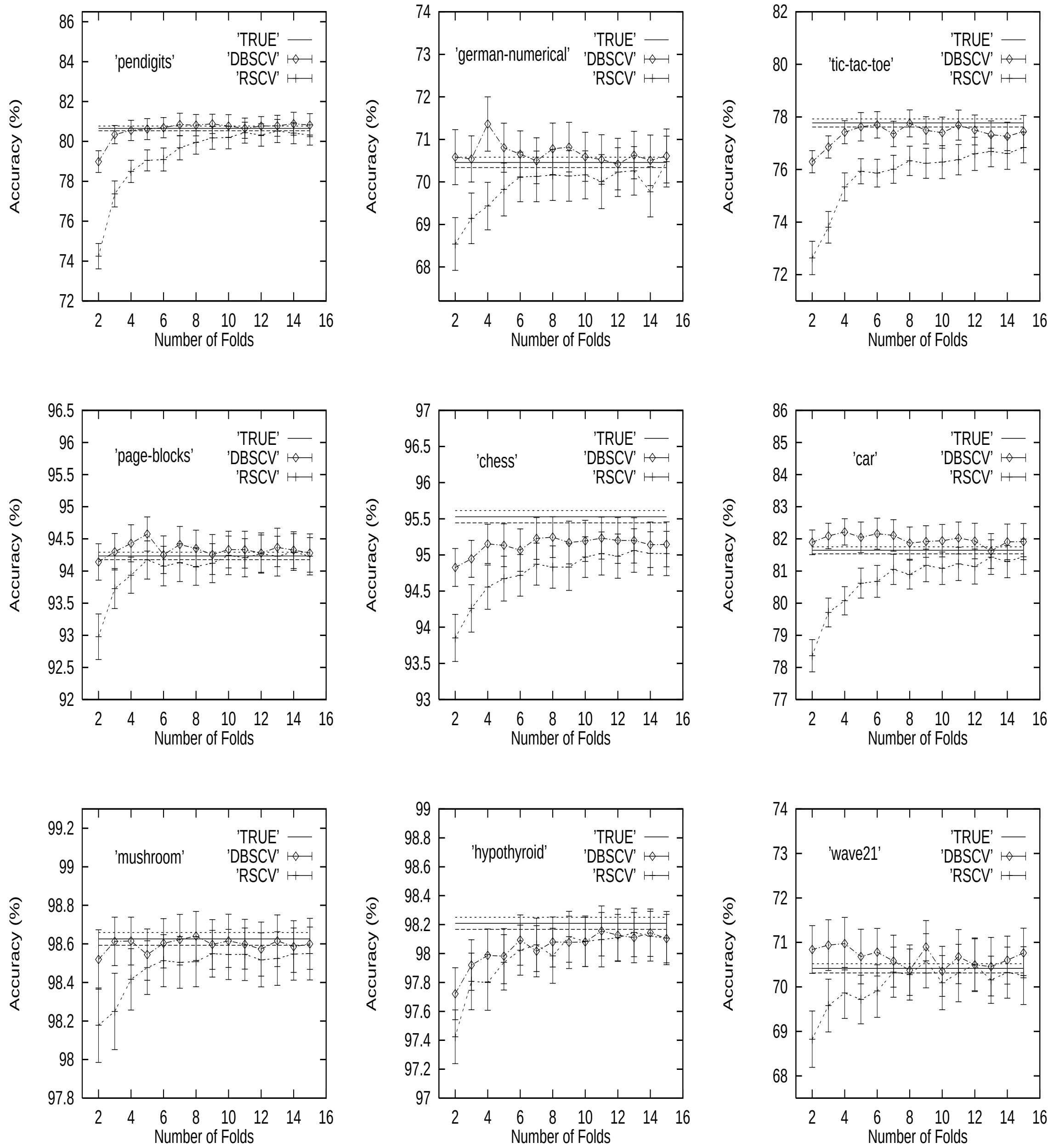

Figure 3: Simulation results on nine real-world domains using $C4.5$ decision trees as classifiers. The results compare the distribution balanced stratified cross-validation ($DBSCV$) to the standard stratified cross-validation ($RSCV$) for the qualities of accuracy estimations (how close to the "true accuracy" ($TRUE$)). The error bars show the 95% confidence intervals for $DBSCV$ and $RSCV$ (based on 100 runs). Each solid straight line shows the "true accuracy" $TRUE$ (which is estimated based on 1,000 runs), while the two dashed straight lines (above and below it) represent the corresponding 95% confidence intervals.

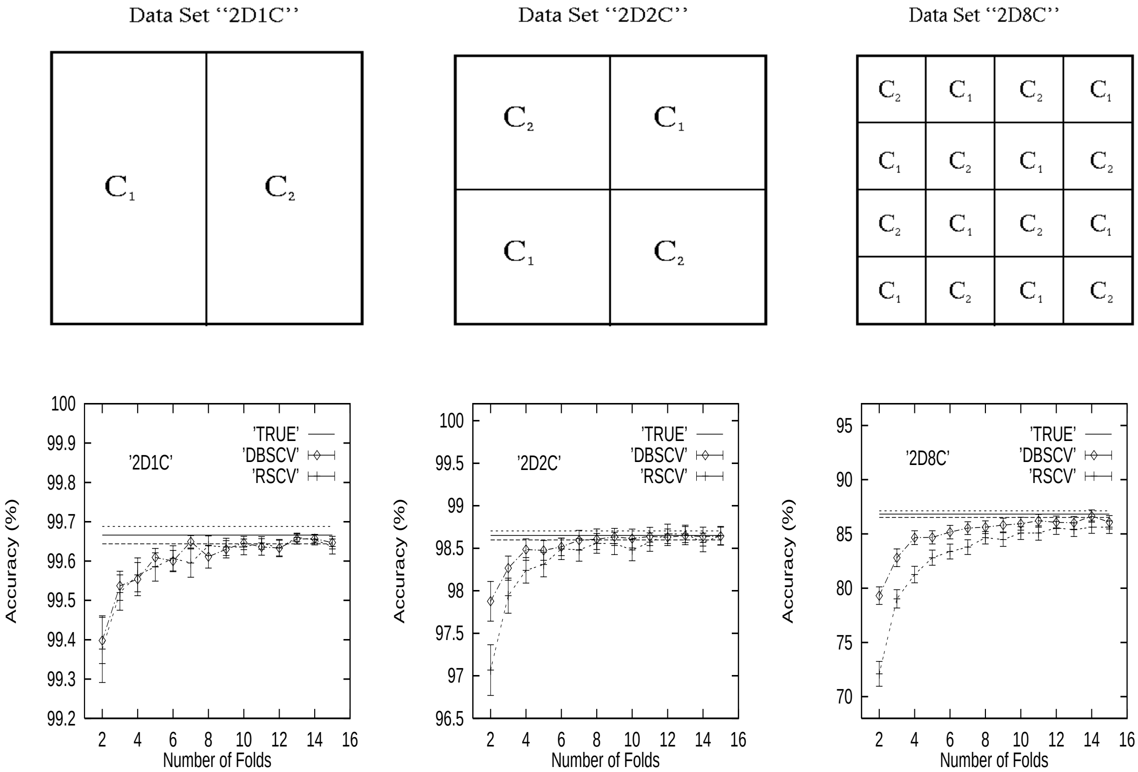

Figure 4: Three artificial data sets $2D1C$, $2D2C$, $2D8C$ with 1, 2, and 8 intraclass clusters are shown in the first row. Each of them has a 2-dimensional feature space and 2 classes. C_1 and C_2 denote the regions of *class 1* and *class 2* respectively. The probability density of sampling cases is uniform across all of the regions for all three data sets. The corresponding simulation results on each of the data sets using $C4.5$ decision trees as classifiers are shown in the second row (the meaning of each curve is the same as explained for Figure 2).