Study on the Impact of Partition-Induced Dataset Shift on *k*-fold Cross-Validation

Jose García Moreno-Torres, José A. Sáez, and Francisco Herrera, *Member, IEEE*

***Abstract*— Cross-validation is a very commonly employed technique used to evaluate classifier performance. However, it can potentially introduce dataset shift, a harmful factor that is often not taken into account and can result in inaccurate performance estimation. This paper analyzes the prevalence and impact of partition-induced covariate shift on different *k*-fold cross-validation schemes. From the experimental results obtained, we conclude that the degree of partition-induced covariate shift depends on the cross-validation scheme considered. In this way, worse schemes may harm the correctness of a single-classifier performance estimation and also increase the needed number of repetitions of cross-validation to reach a stable performance estimation.**

***Index Terms*— Covariate shift, cross-validation, dataset shift, partitioning.**

I. Introduction

In order to evaluate the expected performance of a classifier over a dataset, k-fold cross-validation schemes are commonly used in the classification literature [1]. Also, when comparing classifiers, it is common to compare them according to their performances averaged over a number of iterations of cross-validation. Even though it has been proved that these schemes asymptotically converge to a stable value, which allows realistic comparisons between classifiers [2], [3], in practice a very low number of iterations are often used. The most common variations are 2×5, 5×2, and 10×1, with this notation meaning 2-folds iterated five times, 5-folds iterated two times, and 10-folds iterated once, respectively. Note that when more than one iteration takes place, the partitions are assumed to be constructed independently.

The topic of data stability and classifier bias is very relevant to the field, as can be seen in the numerous attempts to design unbiased classifiers in the recent literature [4], [5], or in the recent research on streaming data [6] . While those designs are definitely worthwhile, we believe that a study of the intrinsic characteristics of the data is needed to have a full picture of the problem. Among the said data characteristics, the amount of partition-induced dataset shift is very relevant and, to the best of our knowledge, usually not taken into account. Here, we try to prove the relevance and need for accounting of this particular issue.

This paper studies the intrinsic variability present in k-fold cross-validation schemes from the point of view of dataset shift [7], [8], which is defined as the situation where the data the classifier is trained on and the data the classifier is going to be used on do not follow the same distribution. More specifically, we focus on covariate shift (a specific kind of dataset shift where the covariates follow a different distribution in the training and test datasets), and the situations where it may appear and cause inaccurate classifier performance estimations.

This paper analyzes how different partitioning methods can introduce dataset shift (or, more specifically, covariate shift) and the effect it has over both the reliability of the estimation of a classifier performance based on a low number of iterations of k-fold cross-validation, and the number of iterations needed to reach a stable classifier performance estimation.

To best analyze the impact of dataset shift, we use four different strategies to create the partitioning.

1) *Standard stratified cross-validation (SCV)*, which is the most commonly employed method in the literature. It places an equal number of samples of each class on each partition to maintain class distributions equal in all partitions. For an example of its use, see [9].
2) *Distribution-balanced SCV (DB-SCV)* [10], a method that attempts to minimize covariate shift by keeping data distribution as similar as possible between training and test folds by maximizing diversity on each fold and trying to keep all folds as similar as possible to each other.
3) *Distribution optimally balanced SCV (DOB-SCV)*, a slight modification of the above and an original contribution of the work presented here, tries to improve the performance of DB-SCV by taking into account more information when choosing in which fold to place each sample.
4) *Maximally shifted SCV (MS-SCV)*, a method designed for testing the maximal influence partition-based covariate shift can have on classifier performance by introducing the maximum possible amount of shift on each partition. To do so, it does the opposite as DB-SCV and creates folds that are as different as possible to each other.

While there is a published work that proposes a different cross-validation strategy [11] designed specifically to combat covariate shift, we chose not to include it in this paper

Manuscript received November 23, 2011; revised May 2, 2012; accepted May 3, 2012. Date of publication June 26, 2012; date of current version July 16, 2012. This work was supported in part by Project TIN2011-28488 and Project P10-TIC-6858. The work of J. G. Moreno-Torres and J. A. Sáez was supported by FPU grants from the Ministerio de Educación y Ciencia of the Spanish Government.

The authors are with the Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18001, Spain (e-mail: jose.garcia.mt@decsai.ugr.es; smja@decsai.ugr.es; herrera@decsai.ugr.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2199516

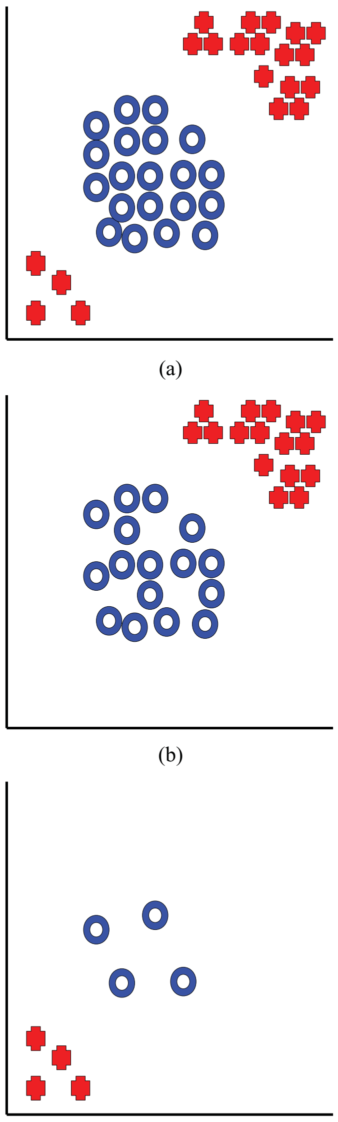

Fig. 1. Extreme example of partition-based covariate shift. Note that the examples on the bottom left of the "cross" class will be wrongly classified because of covariate shift. (a) Full dataset. (b) Training set. (c) Test set.

because it is designed to train classifiers in problems where covariate shift is already present, while the intent here is to analyze to what extent general-purpose cross-validation strategies generate extra covariate shift which is not intrinsic to the problem.

The goal of this paper is to analyze the accuracy of classifier performance prediction both when a low number of iterations of *k*-fold cross-validation are used and when enough of them are used so that a stable value has been achieved. More specifically, we study the following.

1) The accuracy of a single cross-validation experiment in terms of 1 v 1 classifier comparison, and whether different partitioning methods can have an impact on it.
2) The number of independent cross-validation experiments necessary to converge to a stable result in terms of 1 v 1 classifier comparison, also analyzing whether different partitioning methods produce different results.

Algorithm 1 SCV Partitioning Method

```
for each class $c_j \in C$ do
  $n \leftarrow \text{count}(c_j)/k$
  for each fold $F_i (i = 0, \ldots, k-1)$ do
    $E \leftarrow$ randomly select $n$ examples of class $c_j$ from $D$
    $F_i \leftarrow F_i \cup E$
    $D \leftarrow D \setminus E$
  end for
end for
```

A supplementary material website has been created for this paper, which can be found at http://sci2s.ugr.es/covariate-shift-cross-validation.

The remainder of this paper is organized as follows. Section II provides a background on cross-validation and dataset shift. In Section III, the different partitioning methods used for the experimentation in this paper are detailed. Section IV shows the datasets and classification algorithms used in the experimental study. Section V shows the strategy employed to test the suitability of each partitioning method, while Section VI shows the results obtained. This paper is then closed with a few concluding remarks and recommendations in Section VII.

II. Background

This section presents a brief introduction to classifier evaluation through cross-validation in Section II-A and to dataset shift in Section II-B, introducing the concepts relevant to this paper.

A. Cross-Validation for Classifier Evaluation

Cross-validation is a technique used for assessing how a classifier will perform when classifying new instances of the task at hand. One iteration of cross-validation involves partitioning a sample of data into two complementary subsets: training the classifier on one subset (called the training set) and testing its performance on the other subset (test set).

In *k*-fold cross-validation, the original sample is randomly partitioned into *k* subsamples. Of the *k* subsamples, a single subsample is retained as the validation data for testing the classifier, and the remaining $k - 1$ subsamples are used as training data. The cross-validation process is then repeated *k* times, with each of the *k* subsamples used exactly once as the test data. The *k* results from the folds are then averaged to produce a single performance estimation.

Cross-validation has been the subject of profuse study in the literature, some of the most interesting and relevant results are listed here:

1) repeated iterations of cross-validation asymptotically converge to a correct estimation of classifier performance [2];
2) ten-fold cross-validation is better than leave-one-out validation for model selection, and also better than other *k*-fold options [1];
3) *k*-fold cross-validation tends to underestimate classifier performance [1].

Algorithm 2 DB-SCV Partitioning Method

```
for each class $c_j \in C$ do
  $e \leftarrow$ randomly select an example of class $c_j$ from $D$
  $i = 0$
  while count($c_j$) $> 0$ do
    $F_i \leftarrow F_i \cup \{e\}$
    $D \leftarrow D \setminus \{e\}$
    $i = (i + 1) \bmod k$
    $e \leftarrow$ closest example to $e$ of class $c_j$ from $D$
  end while
end for
```

Algorithm 3 DOB-SCV Partitioning Method

```
for each class $c_j \in C$ do
  while count($c_j$) $> 0$ do
    $e_0 \leftarrow$ randomly select an example of class $c_j$ from $D$
    $e_i \leftarrow i$th closest example to $e_0$ of class $c_j$ from $D$ ($i = 1, \ldots, k-1$)
    $F_i \leftarrow F_i \cup \{e_i\}$ ($i = 0, \ldots, k-1$)
    $D \leftarrow D \setminus \{e_i\}$ ($i = 0, \ldots, k-1$)
  end while
end for
```

B. Dataset Shift

The term "dataset shift" refers to the issue where training and test data follow different data distributions [7], [8]. It can happen because of the intrinsic nature of the problem (for example, a classifier trained over financial data from the past five years and used to predict future market changes), or it can be introduced in cross-validation schemes without noticing.

This paper focuses on the latter, studying k-fold cross-validation strategies and the types and impact of dataset shift in them. There are two potential types of dataset shift.

1) *Prior Probability Shift:* It happens when the class distribution is different between the training and test sets [12]. In the most extreme example, the training set would not have a single example of a class, leading to a degenerate classifier. The problems caused by this kind of shift have already been studied, and it is commonly prevented by applying a SCV scheme [13].
2) *Covariate Shift:* In this case, it is the inputs that have different distributions between the training and test sets [14]. Fig. 1 depicts an extreme example of this type of shift that can also lead to extremely poor classifier performance. This type of shift is often ignored in the literature, and the analysis of its prevalence and potential impact is the main contribution of this paper.

III. Partitioning Methods

This section presents a detailed explanation of the different partitioning methods used for testing in this paper, including the pseudo-code to make the replication of our experiments easier. Some assumptions made throughout the pseudo-codes are as follows.

1) The number of folds in a given cross-validation implementation is denoted as k.
2) Folds are named $F_i (i = 0, \ldots, k-1)$. They are treated as a set of examples, and are initially empty.
3) D is another set of examples, initially containing all the examples in the dataset.
4) There is a set of classes $C = \{c_1, \ldots, c_m\}$, where m is the number of classes.
5) There is a function count(c_i) that returns the number of examples of class c_i in D.
6) These methods detail the way to construct the test sets; the training sets are simply the remainder of the dataset.

This section also includes, in Section III-E, an analysis of the differences between DB-SCV and DOB-SCV.

A. SCV

This is the standard method most authors in the field of classification apply. A pseudo-code explaining how it works can be seen in Algorithm 1.

SCV is a simple method: it counts how many samples of each class there are on the dataset, and distributes them evenly on the folds, so that each fold contains the same number of examples of each class. This avoids prior probability shift, since if there is an equal distribution class-wise on each fold, training and test set will have the same class distribution. However, this method does not take into account the covariates of the samples, so it can potentially generate covariate shift.

B. DB-SCV

This method, proposed in [10], adds an extra consideration to the partitioning strategy as an attempt to reduce covariate shift on top of preventing prior probability shift. The method follows the steps detailed in Algorithm 2.

The idea is that by assigning close-by examples to different folds, each fold will end up with enough representatives of every region, thus avoiding covariate shift. To achieve this goal, DB-SCV starts on a random unassigned example and assigns it to the first fold. It then hops to the nearest unassigned neighbor of the same class, and assigns it to the second fold, repeating the process until there are no more examples of that class (when it gets to the last fold, cycles and continues with the first one again). The whole process is repeated for each class.

C. DOB-SCV

This method includes a variation from the one above, and is an original contribution of the work described in this paper. Its basic difference with DB-SCV lies in the order in which the examples are picked to be assigned to each fold. The specifics about this method can be found in Algorithm 3.

Instead of choosing samples one by one like DB-SCV does, DOB-SCV picks a random unassigned example, and then finds its $k-1$ nearest unassigned neighbors of the same class. Once it has found them, it assigns each of those examples to a different fold. The process is repeated until all examples are assigned.

Algorithm 4 MS-SCV Partitioning Method

```
for each class $c_j \in C$ do
  $n \leftarrow \text{count}(c_j)/k$
  $e \leftarrow$ randomly select an example of class $c_j$ from $D$
  for each fold $F_i (i = 0, \ldots, k-1)$ do
    for $s = 1 \rightarrow n$ do
      $F_i \leftarrow F_i \cup \{e\}$
      $D \leftarrow D \setminus \{e\}$
      $e \leftarrow$ closest example to $e$ of class $c_j$ from $D$
    end for
  end for
end for
```

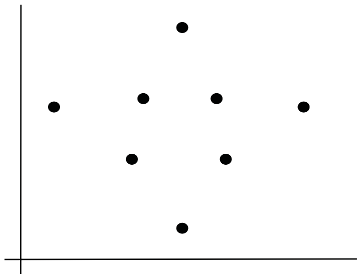
Fig. 2. Artificial dataset used to show differences between DB-SCV and DOB-SCV.

D. MS-SCV

This method is basically the opposite of the previous one in terms of covariate shift, trying to maximize it while keeping prior probability shift at a minimum. Its pseudo-code is shown in Algorithm 4.

MS-SCV is basically a mirrored version of DB-SCV: it picks an unassigned example at random, assigns it to a fold, and finds the nearest unassigned neighbor of the same class. However, instead of assigning it to the next fold, it assigns it to the same fold, and keeps assigning examples to the same fold until the maximum number of examples of that class have been assigned to the fold. Once that fold is "full," it goes to the next fold and repeats the process until all folds are filled. This procedure is again repeated for each class present in the dataset. In this case, the assignation of all close examples to the same fold creates incidences of severe covariate shift, since entire regions are kept without a single example representing them in some folds.

E. Difference Between DB-SCV and DOB-SCV

DB-SCV and DOB-SCV are similar methods with the same philosophy: they attempt to minimize covariate shift by distributing samples of the same class as evenly as possible in terms of their covariates. However, the way DB-SCV is designed makes it a little more sensitive to random choices, since the order in which it traverses the dataset depends only on the nearest neighbor each time, which, if the dataset is particularly poorly suited for this method, can lead to bad performance, while DOB-SCV is more resilient to this factor because of it restarting its exploration of the dataset more often and exploring several directions at once.

To illustrate the type of situation where DB-SCV could perform worse than DB-SCV, we have designed an artificial dataset which can be seen in Fig. 2. This dataset was built to clearly show the situation, since the visualization of high-dimensional datasets is not straightforward, and thus real-world datasets are less suitable for this task. To simplify, let all the samples of this dataset be of the same class and focus only on the avoidance of covariate shift. Also, assume for simplicity a fourfold partitioning scheme, so considering there are eight samples and two will be assigned to each fold.

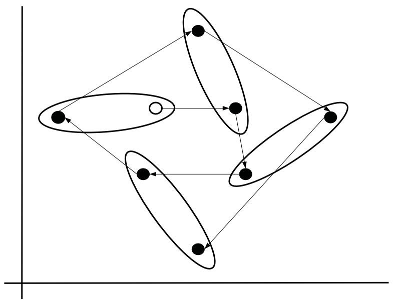
Fig. 3. Artificial dataset partitioned with DB-SCV. Arrows represent unassigned nearest neighbor exploration, white node is randomly chosen as starting point, and ellipses contain the partitions created.

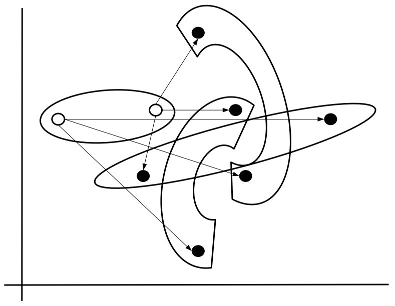
Fig. 4. Artificial dataset partitioned with DOB-SCV. Arrows represent unassigned nearest neighbor exploration, white nodes are randomly chosen as starting points, and shapes contain the partitions created.

In Fig. 3 the result of applying DB-SCV to the artificial dataset is shown. In it, arrows represent unassigned nearest neighbor exploration, white node is randomly chosen as starting point, and ellipses contain the partitions created. It can be seen how exploring only the nearest neighbor can lead the process to spiral around the center, resulting in a poor assignation of samples to folds and introduces a significant amount of covariate shift.

Fig. 4 shows the application of DOB-SCV to the same dataset, with the same starting point. It can be seen how DOB-SCV mostly avoids the pitfall because of its ability to see several neighbors at once, avoiding tunnel vision which can be costly in situations like this. The second white node in the figure corresponds to the second random choice, and we picked the one that results in the worst partitioning. The figure shows a better behavior for DOB-SCV than the one presented by DB-SCV, since the partitions created are better distributed in the domain space.

IV. Datasets and Classifiers

In this section, the experimental framework is presented showing the datasets used in Section IV-A and the classification algorithms studied in Section IV-B.

TABLE I
DATASETS USED FOR THE EXPERIMENTAL STUDY

Dataset	No. of attributes (R/I/N)	No. of examples
Appendicitis	7 (7/0/0)	106
Australian	14 (3/5/6)	690
Banana	2 (2/0/0)	5300
Bands	19 (13/6/0)	365
Breast	9 (0/0/9)	277
Bupa	6 (1/5/0)	345
Chess	36 (0/0/36)	3196
Crx	15 (3/3/9)	653
German	20 (0/7/13)	1000
Haberman	3 (0/3/0)	306
Heart	13 (1/12/0)	270
Hepatitis	19 (2/17/0)	80
Housevotes	16 (0/0/16)	232
Ionosphere	33 (32/1/0)	351
Mammographic	5 (0/5/0)	830
Monk-2	6 (0/6/0)	432
Mushroom	22 (0/0/22)	5644
Phoneme	5 (5/0/0)	5404
Pima	8 (8/0/0)	768
Saheart	9 (5/3/1)	462
Sonar	3 (60/0/0)	208
Spambase	57 (57/0/0)	4597
Spectfheart	44 (0/44/0)	267
Tic-tac-toe	9 (0/0/9)	958
Titanic	3 (3/0/0)	2201
Wdbc	30 (30/0/0)	569
Wisconsin	9 (0/9/0)	683

In order to achieve relevant results, 27 datasets and 9 classifiers were used. They can be seen in Sections IV-A and IV-B, respectively. All classifiers were compared against each other (resulting in 36 unique pairs) over their performance in the test set. The performance metric chosen was the area under the curve (AUC) [15], since it is less sensitive to imbalance than other commonly employed metrics, such as accuracy, and therefore allows us to obtain more solid conclusions.

A. Datasets

As has been mentioned before, we employed 27 datasets in our experimentation. They are all binary classification problems, and were obtained from the KEEL dataset repository [16]. When there were missing values, the whole example was eliminated. A list of the datasets used can be seen in Table I, where "(R/I/N)" refers to real, integer, and nominal attributes.

B. Classification Algorithms

Table II shows the nine classification algorithms employed in this paper, which were chosen to provide a wide range of classifiers. The parameters used were the default ones present in the KEEL tool [25], and are the ones suggested by the original authors of the methods.

TABLE II
CLASSIFICATION ALGORITHMS USED

Algorithm	Abbreviation	Type of classifier
Nearest neighbor $k = 1$ [17]	1 NN	Lazy learner
Nearest neighbor $k = 3$ [17]	3 NN	Lazy learner
C4.5 [18]	C4.5	Decision tree
Fuzzy unordered rule induction algorithm [19]	FURIA	Fuzzy rule-based (Mamdami)
Linear discriminant analysis [20]	LDA	Statistical
PART [21]	PART	Partial decision tree
Positive definite fuzzy classifier [22]	PDFC	Fuzzy rule-based (TSK)
Repeated incremental pruning to produce error reduction [23]	RIPPER	Rule-based
Support vector machine [24]	SVM	Support vector machine

V. ANALYZING PARTITIONING METHODS

We performed three independent experiments using the same procedure, where the only difference was the type of cross-validation scheme used. We tested the 2×5, 5×2, and 10×1 cross-validation schemes.

For each of the above validation schemes, we created 100 independent experiments using each of the methods described in Section III to test both single-experiment accuracy and number of iterations needed to converge to a stable result.

To evaluate single-iteration accuracy, we used the following procedure. Since the procedure is not trivial to understand, we include an example here with the case of 5×2 experiments; 10×1 and 2×5 are done analogously.

1) A reference is needed in order to know whether a classifier performance estimation is accurate. The said reference is based on the "true" performance of the classifiers. To obtain this "truth," we created an extra 200 independent partitions using SCV, and then averaged the performance of each classifier on each dataset over those 200 partitions. The performance is measured as AUC in the test set. The results can be seen in Table III.
2) Perform a Wilcoxon signed-ranks test with the averaged data for each classifier pair. The results can be seen in Table IV, where the numbers on each cell should be read as R+/R−/p-value (where R+ corresponds to row winning, and R− to column). Discard the classifier pairs where p-value > 0.1. In the table, the cases with p-value under 0.1 are marked in bold. We chose to focus only on the pairwise comparisons between classifiers where the true comparison showed a significant difference between classifiers, since it is harder to reach relevant conclusions from the cases where a significant difference could not be found.
3) For each of the 100 instances of 5×2 cross-validation created with each partitioning method, perform a Wilcoxon signed-ranks test for each classifier pair that

TABLE III
AVERAGED "TRUE" CLASSIFIER PERFORMANCE (AUC IN TEST SET)

Dataset	1 NN	3 NN	C45	FURIA	LDA	PART	PDFC	RIPPER	SVM
Appendicitis	0.7506	0.7450	0.7054	0.7265	0.7323	0.7028	0.7278	0.7305	0.6744
Australian	0.8228	0.8474	0.8449	0.8579	0.8649	0.6443	0.8262	0.8206	0.8045
Banana	0.8695	0.8822	0.8855	0.8780	0.5171	0.5617	0.8942	0.6637	0.9004
Bands	0.6904	0.6632	0.6211	0.6047	0.6021	0.5115	0.7068	0.6113	0.7060
Breast	0.5827	0.5842	0.5965	0.6255	0.6115	0.5049	0.6433	0.6151	0.5904
Bupa	0.6050	0.6266	0.6310	0.6554	0.6579	0.5206	0.6914	0.6349	0.6825
Chess	0.9692	0.9692	0.9930	0.9931	0.8510	0.8218	0.9955	0.9926	0.9839
Crx	0.8189	0.8542	0.8539	0.8653	0.5000	0.5561	0.8277	0.8272	0.8035
German	0.6275	0.6349	0.6303	0.6070	0.6438	0.5000	0.6490	0.6434	0.7056
Haberman	0.5462	0.5501	0.5745	0.5864	0.5637	0.5015	0.5575	0.5970	0.5564
Heart	0.7681	0.8038	0.7809	0.7970	0.8365	0.5254	0.8035	0.7539	0.7869
Hepatitis	0.7412	0.7085	0.6588	0.6810	0.7168	0.5857	0.7244	0.7217	0.7410
Housevotes	0.9505	0.9561	0.9646	0.9633	0.9712	0.9620	0.9506	0.9591	0.9483
Ionosphere	0.8750	0.8536	0.8736	0.8805	0.8205	0.8074	0.9352	0.8828	0.9308
Mammographic	0.7550	0.8107	0.8317	0.8342	0.8252	0.7722	0.8181	0.7453	0.8078
Monk-2	0.7419	0.9509	1.0000	1.0000	0.7756	0.5027	1.0000	0.9995	0.9611
Mushroom	1.0000	1.0000	1.0000	1.0000	0.5000	0.8905	1.0000	1.0000	1.0000
Phoneme	0.8690	0.8490	0.8331	0.8071	0.6837	0.5159	0.8474	0.8268	0.8377
Pima	0.6513	0.6713	0.7047	0.7005	0.7235	0.5044	0.6777	0.7029	0.6837
Saheart	0.5938	0.6067	0.6336	0.6375	0.6772	0.5050	0.6007	0.6250	0.6019
Sonar	0.8575	0.8344	0.7354	0.7796	0.7377	0.5416	0.8735	0.7297	0.8709
Spambase	0.8969	0.8981	0.9214	0.9278	0.8695	0.6334	0.9439	0.9230	0.9339
Spectfheart	0.6217	0.6369	0.6201	0.6008	0.5607	0.5000	0.6666	0.6505	0.7589
Tic-tac-toe	0.9104	0.8956	0.8152	0.9765	0.6510	0.5000	0.9884	0.9731	0.8856
Titanic	0.5227	0.5493	0.6911	0.6754	0.6996	0.5001	0.6826	0.6699	0.6824
Wdbc	0.9534	0.9642	0.9330	0.9452	0.9429	0.8149	0.9695	0.9299	0.9540
Wisconsin	0.9570	0.9640	0.9482	0.9568	0.9509	0.5473	0.9620	0.9606	0.9690

showed a significant difference in step 2 (those marked with bold font), using the p-values obtained in that step as the significance threshold. Count how many of the 100 instances achieve the same results. Table V shows an example table of results for DOB-SCV. A similar table is constructed for each of the other partitioning methods studied. The number on each cell is the number of DOB-SCV partitions where the Wilcoxon signed-ranks test declared a significant difference between the compared classifiers' performance. For example, the 71 in the PART versus 1 NN comparison means that 71 out of 100 independent Wilcoxon signed-ranks test (each one with a different partition) proved significant with a threshold of $p\text{-value} = 7.45E-08$. The p-value was obtained from that same cell in Table IV.

4) Average the results of each cell to obtain an aggregated estimation of how close a given partitioning method is to the "true" estimation. In the example of DOB-SCV for 5×2, that average turns out to be 55.684. This is how Table VI is constructed.

To sum up: For each dataset, we averaged the performance (AUC over the test set) of each classifier over the 200 cross-validation experiments, and then performed a Wilcoxon signed-ranks test [26] with a significance level of 0.1 to test whether there existed significant differences between their performances. We considered this Wilcoxon test to be the true comparison between each classifier pair.

To figure out the number of iterations needed for convergence to a stable result, we used the method recommended in [27], which determines the convergence based on reaching a Pearson correlation between accumulated average performances of consecutive instances of cross-validation of 0.9999. More specifically, the method follows these steps in Algorithm 5 (again, using the example of 5×2 with DOB-SCV, the others being analogous).

To achieve more significant results, we repeated this test 10 times for each dataset–classifier pair.

All the experiments were conducted using the KEEL software tool [25].

VI. RESULTS

This section presents a summary of the results obtained by the experiments run following the above framework. Because of space concerns, only a brief summary of the results are included; for a more detailed analysis check http://sci2s.ugr.es/covariate-shift-cross-validation. We first focus on

TABLE IV
WILCOXON SIGNED-RANKS TEST. R+ (R−) DENOTES THE R SCORE OF THE ROW (COLUMN) ALGORITHM. RESULTS ARE PRESENTED AS R+/R−/p-VALUE

	1 NN	3 NN	C45	FURIA	LDA	PART	PDFC	RIPPER	SVM
1 NN	-	119.5/258.5/0.1001	150/228/>0.2	120.5/257.5/0.1029	235/143/>0.2	**375/3/7.45E-08**	**38.5/339.5/9.90E-05**	143.5/234.5/>0.2	**85.0/293.0/1.12E-02**
3 NN	258.5/119.5/0.1001	-	185/193/>0.2	143.5/234.5/>0.2	**261.0/117.0/0.0859**	**377/1/2.98E-08**	**68.5/309.5/2.84E-03**	193.5/184.5/>0.2	120/258/0.1004
C45	228/150/>0.2	193/185/>0.2	-	120.5/257.5/0.1029	242/136/>0.2	**378/0/1.49E-08**	**90.5/287.5/1.68E-02**	180/198/>0.2	146/232/>0.2
FURIA	257.5/120.5/0.1029	234.5/143.5/>0.2	257.5/120.5/0.1029	-	**278.0/100.0/0.0319**	**378/0/1.49E-08**	**108.0/270.0/0.0521**	245.5/132.5/0.1814	185/193/>0.2
LDA	143/235/>0.2	**117.0/261.0/0.0859**	136/242/>0.2	**100.0/278.0/0.0319**	-	**340.0/38.0/**	**77.0/301.0/**	124/254/0.1225	**83.0/295.0/9.61E-03**
PART	**3/375/7.45E-08**	**1/377/2.98E-08**	**0/378/1.49E-08**	**0/378/1.49E-08**	**38.0/340.0/**	-	**1/377/2.98E-08**	**3/375/7.45E-08**	**3/375/7.45E-08**
PDFC	**339.5/38.5/9.90E-05**	**309.5/68.5/2.84E-03**	**287.5/90.5/1.68E-02**	**270.0/108.0/0.0521**	**301.0/77.0/**	**377/1/2.98E-08**	-	**310.5/67.5/**	**273.0/105.0/0.0435**
RIPPER	234.5/143.5/>0.2	184.5/193.5/>0.2	198/180/>0.2	132.5/245.5/0.1814	254/124/0.1225	**375/3/7.45E-08**	**67.5/310.5/**	-	139/239/>0.2
SVM	**293/85/1.12E-02**	258/120/0.1004	232/146/>0.2	193/185/>0.2	**295.0/83.0/9.61E-03**	**375/3/7.45E-08**	**105.0/273.0/0.0435**	239/139/>0.2	-

TABLE V
NUMBER OF TIMES WILCOXON SIGNED-RANKS OBTAINED THE SAME RESULT IN DOB-SCV AS IT DID IN THE "TRUE" RUN

	1 NN	3NN	C45	FURIA	LDA	PART	PDFC	RIPPER	SVM
1 NN	-	-	-	-	-	**71**	**42**	-	**23**
3 NN	-	-	-	-	**96**	**82**	**40**	-	-
C45	-	-	-	-	-	**30**	**60**	-	-
FURIA	-	-	-	-	**72**	**37**	**49**	-	-
LDA	-	**96**	-	**72**	-	**48**	**91**	-	**51**
PART	**71**	**82**	**30**	**37**	**48**	-	**66**	**59**	**56**
PDFC	**42**	**40**	**60**	**49**	**91**	**66**	-	**27**	**58**
RIPPER	-	-	-	-	-	**59**	**27**	-	-
SVM	**23**	-	-	-	**51**	**56**	**58**	-	-

TABLE VI
SINGLE CROSS-VALIDATION EXPERIMENT AND AVERAGE ACCURACY OF THE WILCOXON SIGNED-RANKS TEST

CV scheme	DOB-SCV	DB-SCV	SCV	MS-SCV
2×5	52.687	51.250	31.500	1.250
5×2	55.684	52.737	39.789	0.000
10×1	49.857	51.095	45.333	1.762
Average	52.743	51.694	38.874	1.004

TABLE VII
NUMBER OF CROSS-VALIDATION EXPERIMENTS NEEDED TO CONVERGE TO A STABLE PERFORMANCE ESTIMATION

CV scheme	DOB-SCV	DB-SCV	SCV	MS-SCV
2×5	16.71±12.01	16.82±11.67	18.39±11.88	31.42±13.23
5×2	33.46±24.05	34.86±23.85	37.16±24.41	62.58±26.17
10×1	53.98±32.08	54.70±32.17	59.73±32.14	85.77±27.14

the single-experiment case, in Section VI-A and then present the results corresponding to the number of iterations needed to stabilize in Section VI-B.

A. Single Cross-Validation Example

Table VI shows a summary of the results regarding single-experiment reliability. The data in said table represents the percentage of the time taken by a single cross-validation experiment in comparing two datasets using a Wilcoxon signed-ranks test obtained the right result, which is understood to be the "true" one as defined in Section V. Remember that the significance level is set to the same as the "true" data achieved, and that only comparisons between classifiers that showed a significant difference in performance are considered.

Some interesting conclusions can be extracted by looking at these results.

1) Partition-induced covariate shift can significantly hamper the reliability of running a single experiment. MS-SCV produces a much worse accuracy than all other partitioning strategies.
2) Randomly distributing the examples of a dataset can sometimes induce covariate shift, as can be deduced from SCV having a lower accuracy than both DB-SCV and DOB-SCV.
3) DOB-SCV and DB-SCV obtain similar results, with a slight advantage in favor of DOB-SCV.

B. Number of Iterations Needed to Stabilize

Table VII shows the average number of cross-validation experiments needed to converge to a stable performance estimation, along with the standard deviation.

A number of interesting conclusions can be drawn from this table.

1) It can be seen that covariate shift can potentially have a serious impact in the stability of the results obtained by classifiers, as proven by the difference in iterations needed between MS-SCV and the other methods.
2) The sporadic appearance of covariate shift has an impact in convergence terms, as shown by the fact that both DB-SCV and DOB-SCV converge faster than SCV.
3) 2×5 experiments converge significantly faster than 5×2 and 10×1, and 5×2 also converges significantly faster than 10×1. This is because each instance of the 2×5 method is already comprised of five runs of twofold

Algorithm 5 Convergence Estimation Algorithm

```
for each method studied $m_i$ in Methods do
  for each dataset studied $d_j$ in Datasets do
    Estimate the performance of $m_i$ over $d_j$ using DOB-SCV $5 \times 2$ cross-validation, saving it in $\text{Estim}_{ij0}$
    Estimate the performance of $m_i$ over $d_j$ using a different instance of DOB-SCV $5 \times 2$ cross-validation, saving it in $\text{Estim}_{ij1}$
    $\text{Estim}_{ij1} \leftarrow E_{ij0} \cup \text{Estim}_{ij1}$
    $k \leftarrow 2$
    while PearsonCorrelation($E_{ij0}$, $E_{ij1}$)<0.9999 do
      $\text{Estim}_{ij0} \leftarrow \text{Estim}_{ij1}$
      $\text{Estim}_{ij1} \leftarrow \text{Estim}_{ij1} \cup \text{Estim}_{ijk}$, where $\text{Estim}_{ijk}$ is a new performance estimation obtained with a different DOB-SCV $5 \times 2$ instance
      $k \leftarrow k + 1$
    end while
    The convergence for $m_i$ over $d_j$ is defined as $k$: $\text{Conv}_{ij} \leftarrow k$.
  end for
end for
```

cross-validation, effectively using more information per iteration than 5×2 and 10×1. An analogous reason explains the difference between 5×2 and 10×1.

VII. Conclusion

We presented an experimental analysis on the impact partition-based covariate shift can have on the reliability of classifier performance through cross-validation.

We studied four different partitioning methods and showed that, when a covariate shift is introduced, single-experiment reliability decreases and the number of iterations required to reach a stable state increases.

We found that cross-validation approaches that try and limit the impact of partition-induced covariate shift are more reliable when running a single experiment, and need a lower number of iterations to stabilize. Among them, we showed that DOB-SCV is slightly more effective than DB-SCV, presenting an example of the type of situation where DOB-SCV can perform better than DB-SCV. We thus recommend cross-validation users to use DOB-SCV as the partitioning method in order to avoid covariate-shift-related problems.

We studied the number of iterations needed to reach a stable performance estimation with the different partitioning strategies, and found that DOB-SCV and DB-SCV outperform the others, which supports the claim that partition-induced covariate shift can hinder the reliability of classifier evaluation and the need for a specifically designed partitioning method to avoid it.

References

[1] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. Artif. Intell.*, 1995, pp. 1137–1145.

[2] M. Stone, "Asymptotics for and against cross-validation," *Biometrika*, vol. 64, no. 1, pp. 29–35, 1977.

[3] B. Efron and G. Gong, "A leisurely look at the bootstrap, the jackknife, and cross-validation," *Amer. Stat.*, vol. 37, no. 1, pp. 36–48, 1983.

[4] J. Zhang, X. Wang, U. Krüger, and F.-Y. Wang, "Principal curve algorithms for partitioning high-dimensional data spaces," *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 367–380, Mar. 2011.

[5] H. Zhang, L. Cui, X. Zhang, and Y. Luo, "Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2226–2236, Dec. 2011.

[6] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–1914, Dec. 2011.

[7] J. Q. Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA: MIT Press, 2009.

[8] J. G. Moreno-Torres, T. Raeder, R. Aláiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognit.*, vol. 45, no. 1, pp. 521–530, Jan. 2012.

[9] J. R. Cano, F. Herrera, and M. Lozano, "Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability," *Data Knowl. Eng.*, vol. 60, pp. 90–108, Jan. 2007.

[10] X. Zeng and T. R. Martinez, "Distribution-balanced stratified cross-validation for accuracy estimation," *J. Experim. Theor. Artif. Intell.*, vol. 12, no. 1, pp. 1–12, 2000.

[11] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation," *J. Mach. Learn. Res.*, vol. 8, pp. 985–1005, Dec. 2007.

[12] A. Storkey, "When training and test sets are different: Characterizing learning transfer," in *Dataset Shift in Machine Learning*, J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, Eds. Cambridge, MA: MIT Press, 2009, pp. 3–28.

[13] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2005.

[14] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Stat. Plann. Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[15] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, Mar. 2005.

[16] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Logic Soft Comput.*, vol. 17, nos. 2–3, pp. 255–287, 2010.

[17] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[18] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1993.

[19] J. Hühn and E. Hüllermeier, "FURIA: An algorithm for unordered fuzzy rule induction," *Data Mining Knowl. Discovery*, vol. 19, no. 3, pp. 293–319, 2009.

[20] R. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, 1936.

[21] E. Frank and I. Witten, "Generating accurate rule sets without global optimization," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 144–151.

[22] Y. Chen and J. Wang, "Support vector learning for fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 6, pp. 716–728, Dec. 2003.

[23] W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 1–10.

[24] W. C. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 115–123.

[25] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. D. Jesus, S. Ventura, J. M. Garrell, J. Otero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "Keel: A software tool to assess evolutionary algorithms for data mining problems," *Soft Comput. Fusion Found., Methodol. Appl.*, vol. 13, no. 3, pp. 307–318, 2009.

[26] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, 2010.

[27] T. Raeder, T. R. Hoens, and N. V. Chawla, "Consequences of variability in classifier performance estimates," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 421–430.

Jose García Moreno-Torres received the M.Sc. degree in computer science from the University of Granada, Granada, Spain, in 2008. He is currently pursuing the Ph.D. degree with the Soft Computing and Intelligent Information Systems Group, Department of Computer Science and Artificial Intelligence, University of Granada, under the supervision of Prof. F. Herrera.

His current research interests include dataset shift, imbalanced classification, and bibliometrics.

Mr. Moreno-Torres was a recipient of an International "La Caixa" Scholarship, and conducted research as a Fellow of the IlliGAL Laboratory, University of Illinois at Urbana-Champaign, Urbana, under the supervision of Prof. D. E. Goldberg.

José A. Sáez received the M.Sc. degree in computer science from the University of Granada, Granada, Spain, in 2009. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Artificial Intelligence, University of Granada.

His current research interests include noisy data in classification, discretization methods, and imbalanced learning.

Francisco Herrera (M'10) received the M.Sc. degree in mathematics and the Ph.D. degree in mathematics from the University of Granada, Granada, Spain, in 1988 and 1991, respectively.

He is currently a Professor with the Department of Computer Science and Artificial Intelligence, University of Granada. He has published over 200 papers in international journals. He is the co-author of the book *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases* (World Scientific, 2001). His current research interests include computing with words and decision making, data mining, bibliometrics, data preparation, instance selection, fuzzy rule-based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, and memetic algorithms and genetic algorithms.

Prof. Herrera is currently an Editor-in-Chief of the international journal *Progress in Artificial Intelligence* (Springer) and serves as an Area Editor of the *Journal Soft Computing (Area of Evolutionary and Bioinspired Algorithms)* and the *International Journal of Computational Intelligence Systems (Area of Information Systems)*. He is an Associate Editor of several journals, including the IEEE TRANSACTIONS ON FUZZY SYSTEMS, *Information Sciences, Advances in Fuzzy Systems* and the *International Journal of Applied Metaheuristics Computing*. He serves as a member of the editorial boards of many journals, including *Fuzzy Sets and Systems, Applied Intelligence, Knowledge and Information Systems, Information Fusion, Evolutionary Intelligence,* the *International Journal of Hybrid Intelligent Systems,* and *Memetic Computation, Swarm and Evolutionary Computation*. He received the ECCAI Fellowship in 2009, the Spanish National Award on Computer Science ARITMEL to the Spanish Engineer on Computer Science in 2010, and the International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition) in 2010.