



# A domain adaptation technique through cluster boundary integration

Vishnu Manasa Devagiri<sup>1</sup> · Veselka Boeva<sup>1</sup> · Shahrooz Abghari<sup>1</sup>

Received: 19 May 2023 / Accepted: 19 September 2024  
© The Author(s) 2024

## Abstract

Many machine learning models deployed on smart or edge devices experience a phase where there is a drop in their performance due to the arrival of data from new domains. This paper proposes a novel unsupervised domain adaptation algorithm called DIBCA++ to deal with such situations. The algorithm uses only the clusters' mean, standard deviation, and size, which makes the proposed algorithm modest in terms of the required storage and computation. The study also presents the explainability aspect of the algorithm. DIBCA++ is compared with its predecessor, DIBCA, and its applicability and performance are studied and evaluated in two real-world scenarios. One is coping with the Global Navigation Satellite System activation problem from the smart logistics domain, while the other identifies different activities a person performs and deals with a human activity recognition task. Both scenarios involve time series data phenomena, i.e., DIBCA++ also contributes towards addressing the current gap regarding domain adaptation solutions for time series data. Based on the experimental results, DIBCA++ has improved performance compared to DIBCA. The DIBCA++ has performed better in all human activity recognition task experiments and 82.5% of experimental scenarios on the smart logistics use case. The results also showcase the need and benefit of personalizing the models using DIBCA++, along with the ability to transfer new knowledge between domains, leading to improved performance. The adapted source and target models have performed better in 70% and 80% of cases in an experimental scenario conducted on smart logistics.

**Keywords** Clustering techniques · Domain adaptation · Cluster integration

## Abbreviations

AI	Artificial intelligence	ISM	Inductive system health monitoring
ASM	Adapted source model	IM	Integrated model
ATM	Adapted target model	LTE	Long term evolution
AMI	Adjusted mutual information	ML	Machine learning
ARI	Adjusted Rand index	MI	Mutual information
DaLiAc	Daily life activities	ODIP	Online domain incremental pool
DIBCA	Domain integration bi-correlation clustering algorithm	OTDD	Optimal transportation dataset distance
DIBCA++	Optimized domain integration bi-correlation clustering algorithm	RI	Rand index
GNSS	Global navigation satellite system	SD	Source data
HAR	Human activity recognition	SM	Source model
		TD	Target data
		TM	Target model
		XAI	Explainable AI

✉ Vishnu Manasa Devagiri  
vishnu.manasa.devagiri@bth.se

Veselka Boeva  
veselka.boeva@bth.se

Shahrooz Abghari  
shahrooz.abghari@bth.se

<sup>1</sup> Department of Computer Science, Blekinge Institute of Technology, Valhallavägen 1, 37141 Karlskrona, Sweden

## 1 Introduction

Many smart applications are used in different types of environments, which causes a drop in models' performance due to changes in the data characteristics of these environments. In such circumstances, the model needs to be updated to accommodate the new data characteristics. Domain

adaptation, a subbranch of transfer learning, can be used to achieve this. In transfer learning (Zhuang et al. 2021), the source and target domains can be completely different, whereas, in domain adaptation (AlShehhi et al. 2021), the source and target domains are related, addressing the same problem but can have different data distributions.

This work is an extension of Devagiri et al. (2022), in which the Domain Integration Bi-correlation Clustering Algorithm (DIBCA) is introduced. The algorithm initially identifies the correlations between the source and target domain models and then integrates similar clusters from the two domains. DIBCA produces an integrated clustering model that can be used for both domains. The produced model is composable and can be split into two lighter adapted models, one per domain. This facilitates knowledge transfer across the domains and enriches the models with knowledge from other domains. DIBCA has used a learning algorithm proposed by Iverson (2004) that presents each cluster using the low and high attribute value vectors as representatives (also referred to as minimum bounding box); hence DIBCA is also designed to handle clusters using these low and high attribute value vectors. The algorithm's performance has been studied and evaluated in the areas of domain adaptation on a smart logistics use case using data obtained from our industrial partner and automatic data labeling on a publicly available Human Activity Recognition (HAR) data set (Reiss and Stricker 2012). The preliminary experimental results show the algorithm's potential in these two areas. On average, with respect to the domain adaptation task, the integrated and adapted models perform comparably or better than the source and target models, and for the labeling task, DIBCA labels up to 91.1% of classes correctly.

The cluster representatives (low and high attribute value vectors) used in the DIBCA and the learning algorithm are sensitive to outliers. For example, if one data point is far away from the other population of the cluster, it impacts how the cluster is represented. In order to mitigate this effect, the DIBCA and the learning algorithm are optimized in the current study to use the mean, standard deviation, and size of each cluster as representatives. This makes the optimized algorithm, DIBCA++, more robust to handling outliers than the previous version, as such information better reflects the clusters' actual data characteristics. In the current optimized version, instead of bounding boxes based on low and high attribute value vectors, confidence intervals are used to determine cluster boundaries and predict new data points of a cluster. This idea is inspired by the work done by Davidsen (1996), where the distribution of data points of each attribute or feature is assumed to be normally distributed. Since this is not generally the case in all situations, we use Chebyshev's inequality to determine the interval describing

cluster boundaries. Information about the improvements or modifications made to DIBCA++ and the learning algorithm when compared to its predecessors is elaborated further in Sects. 3.3 and 3.4, respectively.

In this work, DIBCA++ (optimized DIBCA) is introduced, and its modules are formally described. DIBCA++ is evaluated and compared with the DIBCA algorithm on richer data sets and in a variety of experimental scenarios to study its robustness and applicability. The summary of contributions of the work is elaborated below:

1. An improved learning algorithm is devised and used for the initial building of the domain cluster model.
2. An optimized version of the DIBCA algorithm (DIBCA++), robust to outliers, is proposed.
3. The potential of DIBCA++ in domain adaptation tasks is evaluated in new specially designed experimental scenarios in order to study and gain a complete and in-depth picture of the algorithm characteristics, e.g., we explore
  - a. how knowledge obtained from one device or user can be transferred to a different device or user. For example, this can also be used to help avoid a cold start.
  - b. the usefulness of the proposed algorithm in use-cases that require personalization, e.g., personalized predictions and recommendations.
  - c. how the similarity between the data of the source and target domains affects the algorithm performance.
  - d. the performance of the clustering models generated by the DIBCA++ compared to that of the clustering model built on the integrated data from both domains.
  - e. how the algorithm explainability features can be used to support the understanding and interpretation of the produced results.

This section is followed by related work in Sect. 2, where an overview of relevant existing studies is presented. Next, we provide a detailed description of the proposed algorithm along with a discussion about its explainability and applicability in Sect. 3. Details about the data sets and evaluation measures are presented in Sects. 4 and 5, respectively. The experiments conducted in our study are explained in Sect. 6. The results from the latter are analyzed and discussed in Sect. 7. Finally, the conclusions and future work of the study are presented in Sect. 8.

## 2 Related work

This section gives an overview of the domain adaptation state-of-the-art studies related to our work.

Transfer learning refers to transferring the knowledge acquired in one usually large (source) model, with rich knowledge, to another (target) model that is typically smaller. It aims to improve learners' performance on target domains by exploiting knowledge acquired from another task or domain (the source) (Zhuang et al. 2021). In comparison with continual learning (De Lange and Tuytelaars 2021), it does not involve continuous adaptation after learning the target task. Moreover, performance on the source task(s) is not considered during transfer learning. Pan and Yang (2010) and Zhuang et al. (2021) have published studies that overview recent works in the area of transfer learning. Pan and Yang (2010) in their work review and categorize transfer learning works in a wide range of fields including classification, regression, and clustering. They categorize transfer learning into three categories, inductive, transductive, and unsupervised, and highlight that more attention might be towards unsupervised transfer learning in the future. In the work done by Zhuang et al. (2021) the focus of review is on one type of transfer learning where the source and target have the same set of attributes, named homogeneous transfer learning. The Work done by Madadi et al. (2020) is also a survey, but it focuses on deep unsupervised domain adaptation dealing with classification problems.

Domain adaptation is a subbranch of transfer learning, where the knowledge is transferred between two domains addressing the same problem, having different data distributions (AlShehhi et al. 2021). In other words, it relaxes the classical machine learning assumption of having training and testing data drawn from the same distribution (Csurka 2017). Domain adaptation is widely used in various applications where training data distribution is different from the data distribution that is used in real-time. Some examples include personalized recommender systems and autonomous vehicles. Adversarial clustering is one of the popular methods used to reduce the shift between the source and target domains (Li et al. 2021a; Wang et al. 2022). Li et al. (2021a) propose *Cross-domain Adaptive Clustering*, a semi-supervised domain adaptation approach. Wang et al. (2022) proposes an unsupervised domain adaptation algorithm capable of preserving the inter-class compactness for image classification. Hundschell et al. (2023) in their work evaluate two state-of-the-art adversarial clustering techniques one based on Recurrent Neural Networks, called VRADA, and the other based on Convolutional Neural Networks, called CoDATS on four publicly available time-series data sets. Li et al.

(2021) study the universal domain adaptation problem, where the source and target domains have different label spaces. A novel domain consensus clustering algorithm is proposed, which separates the common and private clusters of both domains. For the common clusters, the class alignment technique is used to minimize the distribution shift. Xu et al. (2022) use domain adaptive meta-learning to avoid cold start in recommender systems.

One of the concerns in the field of domain adaptation is the privacy of source data. When the model from the source domain is being adapted to the target domain, the privacy of the source data is not protected in many of the studies. Tang et al. (2022) and Zhu (2021) highlight the importance and need for source data-free domain adaptation algorithms to preserve the privacy of source data. Tang et al. (2022) propose a source data-free self-supervised learning method entitled *semantic consistency learning on manifold*. In the work of Zhu (2021), which is also a source data-free domain adaptation algorithm, initially pseudo labels and prediction confidence for the target data are obtained using the source model. This is followed by using high-confidence prediction samples to cluster target data toward the centers of these samples.

Most of the existing domain adaptation approaches are based on deep learning (Madadi et al. 2020; Tang et al. 2022) or semi-supervised learning (Li et al. 2021a; Orbes-Arteainst et al. 2019) models that require significant computational resources, which make them inappropriate to be used on the edge devices. In the work of Tang et al. (2022), unsupervised domain adaptation is treated as the discriminative clustering task on target data based on information from closely related labeled source data. The authors propose a novel algorithm entitled DisClusterDA. Orbes-Arteainst et al. (2019) use knowledge distillation to generalize deep neural networks so that they can be used in semi-supervised domain adaptation problems. Some recent developments in the online domain of incremental continual learning (Gunasekara et al. 2022) are worth to be studied in this research context. Gunasekara et al. (2022) in their work propose a novel domain adaptation algorithm using an Online Domain Incremental Pool (ODIP) of learners and dynamic task predictor. The task predictor selects the appropriate neural network for the current task from the ODIP.

In summary, the majority of the research work done in the field of domain adaptation is in the areas intersecting deep learning and computer vision. Domain adaptation in the temporal dimension is also less presented in the literature. Novel domain adaptation techniques that are resource-efficient and able to support robust model adaptation to new contexts are evidently needed. The proposed DIBCA++ is based on clustering techniques, which is its distinguishing characteristic and also makes the algorithm

applicable in resource-constrained computational setups. Our work contributes towards domain adaptation for time series data, which, as mentioned above, is a less explored area compared to computer vision and other deep learning algorithms applications. In addition, the algorithm uses a data-free approach, i.e., only the clustering models of source and target are required in the adaptation process. This preserves the privacy of both source and target domains.

### 3 Proposed algorithm

In this paper, we propose DIBCA++ (an optimized version of DIBCA), and an optimized version of the learning algorithm used by Abghari et al. (2022) to do initial clustering, such that both these algorithms are robust to outliers. The DIBCA algorithm was designed to be used in combination with clustering algorithms like Inductive System health Monitoring (ISM), proposed initially in Iverson (2004) and later adapted by Abghari et al. (2022), using low and high attribute value vectors of a cluster as cluster representatives (also called boundaries). The low and high attribute vectors are derived using the least and highest values for each attribute of a cluster, respectively. This makes them sensitive to the outliers, implying that even if one data point (or one of its attributes) is located away from the densely populated area of the cluster, it impacts the cluster representation.

In DIBCA++ and the learning algorithm proposed, this is eliminated by estimating the cluster boundaries using the cluster's mean and standard deviation. Mean, standard deviation, and additionally cluster size are considered representatives in the current version. Davidsson (1996) uses the probability density function of the normally distributed data (assuming data is normally distributed) to obtain the cluster boundaries. However, not all data is normally distributed, hence, in the current work, we use Chebyshev's inequality (Saw et al. 1984) to determine the cluster boundaries with the help of the clusters' mean, standard deviation, and a variable  $k$ . The value of  $k$  is determined based on the confidence level at which the cluster boundaries are defined. The advantage of using Chebyshev's inequality is that it holds true for a wide range of distributions. The proposed algorithm is intended to transfer existing historical knowledge to a model that will be used in a new domain. The algorithm can update or adapt the existing clustering solution, considering the data characteristics of newly added domains. This enriches the clustering solutions of both source (historical) and target (new) domains.

**Table 1** Summary of notations used

$C^1$	Source model
$C^2$	Target model
$C_i^1$	Cluster $i$ of Source domain
$C_i^2$	Cluster $i$ of Target domain
$d$	Data point
$d(., .)$	Euclidean distance between two data vectors
$D$	Labeled data set
$h_A$	Higher boundary value vector of Cluster $A$
$k$	Real number ( $> 1$ ), for Chebyshev's inequality
$l_A$	Lower boundary value vector of cluster $A$
$m_A$	Mean value vector of cluster $A$
$n$	Number of clusters in source domain
$n_{corr}$	Identified correlations between source and target domains
$n_c$	Number of clusters
$n_d$	Number of data points in the test set
$o$	Number of clusters in target domain
$R(., .)$	Range based distance measure between two clusters
$s_A$	Size of cluster $A$
$\mathcal{T}$	Threshold
$X_A$	Random variable of Cluster $A$
$\sigma_A$	Standard deviation value vector of cluster $A$

To facilitate the reader in the forthcoming explanation of different parts of our DIBCA++ algorithm, we summarize the used notations alphabetically in Table 1.

#### 3.1 Model generalization and cluster representation

The initial clustering model of the domain (historical/source and the new domain) can be built using any partitioning or a hierarchical clustering algorithm. The important part is that for each cluster,  $A$ , in the built model, three entities, mean value vector ( $m_A$ ), standard deviation value vector ( $\sigma_A$ ), and the size of the cluster ( $s_A$ ) are stored which are used as cluster representatives.  $m_A$  and  $\sigma_A$  are obtained by calculating the mean and standard deviation of each attribute of a cluster. The cluster size is needed to recalculate the mean and standard deviation of the integrated clusters.

Cluster representatives  $\sigma_A$ ,  $m_A$  together with a real variable  $k$  ( $k > 1$ ) are used to define a confidence area or boundaries for each cluster based on Chebyshev's inequality (Eq. 1) (Saw et al. 1984) as  $[m - k\sigma, m + k\sigma]$ . For a chosen  $k$ , value  $1/k^2$  gives the fraction of the values that can lie outside of the interval  $[m_A - k\sigma_A, m_A + k\sigma_A]$ . Hence,  $1 - (1/k^2)$  gives the fraction of values that lie within the stated interval, and  $(1 - (1/k^2)) * 100$  gives the percentage.

$$P(|X_A - m_A| > k\sigma_A) < \frac{1}{k^2} \tag{1}$$

Based on the evaluation of different  $k$  values, the experiments in the current study use  $k = 6$  or  $k = 10$ , which implies that a minimum of 97.2% or 99% of the data points lie within the interval  $[m_A - k\sigma_A, m_A + k\sigma_A]$ , respectively. The value of  $k$  is flexible and can be determined based on the requirements of the use case or application where the algorithm is used. As the value of  $k$  increases, the model becomes more generalized and categorizes more data points into the cluster. Therefore, for sensitive applications where it is desired to have fewer false positives, a lower  $k$  value is preferred. To assign a data point to a cluster, initially, the closest cluster to the data point is identified, then if the data point is within the cluster boundary, it is categorized into that cluster. If the aforementioned condition is not satisfied, the data point is categorized as being non-located (given label  $-1$ ) in the considered clustering solution.

### 3.2 Range-based distance measure

The range-based distance measure originally introduced by Devagiri et al. (2022) is inspired by the work that studies how the overlap of the interval alternatives' evaluations can be used to express valued preferences among the alternatives (Boeva and De Baets 2004). The latter work proposes a metric for the comparison of information available as a form of intervals similar to interval decision-making. The range-based distance measure is used to determine the closeness

between two clusters. It is defined by Eq. 2, where  $A$  and  $B$  are two clusters. In this equation, each cluster, e.g.,  $A$ , is represented by three entities, denoted by  $l_A$ ,  $h_A$ , and  $m_A$ , of its lower boundary, higher boundary, and mean value vectors, respectively. Notice that  $d(\cdot, \cdot)$  is the Euclidean distance between two data vectors.

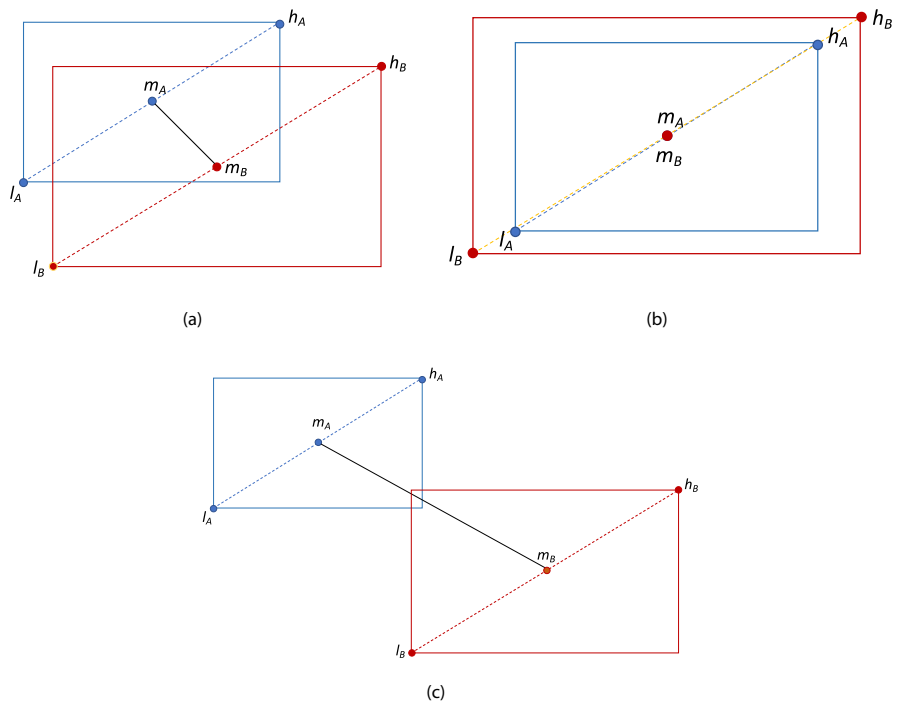
$$R(A, B) = \frac{d(m_A, m_B)}{d(l_A, m_A) + d(m_B, h_B)} \tag{2}$$

Figure 1 illustrates the degree of cluster overlap in a 2D space for different possible ranges of range-based distance measure. Values closer to zero imply that the clusters are similar to each other. The further away from zero, the more distinct they are. For example, one can notice in Fig. 1a that the distance between their mean vectors is smaller than the sum of the distance between the mean vector and the lower boundary vector of the first cluster and the distance between the mean vector and the higher boundary vector of the second cluster (i.e.,  $d(m_A, m_B) < d(l_A, m_A) + d(m_B, h_B)$ ). The latter implies to merge the two clusters since they are significantly overlapping, i.e.,  $R(A, B) < T$ , while in Fig. 1c  $d(m_A, m_B) \geq d(l_A, m_A) + d(m_B, h_B)$ , i.e., evidently the overlap between the two clusters is not significant to merge them.

### 3.3 DIBCA++

DIBCA algorithm is initially proposed by Devagiri et al. (2022). In this work, the algorithm is optimized to be robust to outliers and is evaluated in more experimental scenarios.

**Fig. 1** Visualisation of three different scenarios illustrating the calculation of a range-based distance metric in 2D space with respect to a specific threshold,  $T$ , expressing clusters closeness: (a)  $R(A, B) < T$ , i.e., the clusters significantly overlap with each other, and therefore, they are merged; (b)  $d(m_A, m_B) = 0$ , i.e., one cluster is a subset of the other cluster, and thus they are merged to form a new cluster; (c)  $R(A, B) \geq T$ , i.e., the clusters do not significantly overlap with each other, and therefore they are not merged



The core algorithm of DIBCA++ consists of three main modules: *Domain Correlation*, *Domain Integration*, and *Domain Adaptation*, which are elaborated on in this section. The module structure of DIBCA++ is a novelty that is not presented in the original work. The clear discrimination among the main phases of the algorithm and modularization of them can be considered an improvement in comparison with DIBCA. In the latter, those phases are not explicitly outlined and defined in separate algorithms, except for the final integration of the two domain models. In addition to this, the three main modules of DIBCA++ are designed to be robust to outliers.

The main improvements between DIBCA and DIBCA++ can be highlighted as follows:

1. DIBCA uses the low and high attribute value vectors as cluster representatives, i.e., the vectors composing the least and highest values of each attribute of the cluster. These value vectors are also sensitive to outliers. Whereas DIBCA++ uses the mean, standard deviation, and size of the clusters as representatives, thus also making it robust to outliers.
2. DIBCA initially calculates the mean of each of the clusters of source and target domain using its two representatives, the low and high attribute value vectors. DIBCA++ on the other hand, uses one of its cluster representatives (mean) directly. It can be noted that the mean calculated in DIBCA is just an average of the low and high attribute vectors, whereas DIBCA++ uses the mean obtained based on all the data points of the cluster.
3. As the cluster representatives of DIBCA++ are different from those of DIBCA, the integration process done to obtain them for the new integrated clusters varies in both algorithms. The Domain Integration module of DIBCA++ uses the mean, standard deviation, and size of clusters to be integrated to obtain the new mean, standard deviation, and size of the newly integrated cluster. Whereas in DIBCA, the min and max values for each attribute are obtained from the union of all the cluster representatives that need to be integrated to form the new low and high attribute value vectors.
4. DIBCA++ is clearly divided into three main modules, which help in the easy adaptation of the algorithm to new scenarios by replacing some modules with new updated/optimized implementations. In DIBCA, these are not clearly distinguished; only the cross-labeling (part of the Domain Correlation module used in DIBCA++) and the integration phases (Domain Integration module in DIBCA++) are discussed separately.

### Domain correlation

The domain correlation module is used to identify the correlations between the clusters of the source and target

models. This is done in two steps: 1) identifying similar clusters across the domains using cross-labeling, and 2) using the range-based distance measure, Eq. 2, to find the correlation between similar clusters. In the cross-labeling phase, the source cluster representatives (mean value vector of each cluster) are labeled using the target clustering model, and vice versa, i.e., the target cluster representatives are labeled by the source clustering model. As the general use of the labeling function of a clustering algorithm is to determine the appropriate cluster to which the considered data point belongs, this step helps identify similar clusters in the two domains.

After the cross-labeling step, the mapping (similarity) between the source and target clusters can be categorized into four types: (i) mapping is in both directions, (ii) mapping from source to target (source representatives are labeled using target model), (iii) mapping from target to source (target representatives are labeled using source model), and (iv) no mapping. Clusters in case (i) are identified to be strongly correlated to each other as the similarity is identified in both directions. The similarity obtained in case (ii) is ignored as the mapping is done based on the target model, which is assumed to have less knowledge than the source model; these clusters remain as they are if no other mapping exists. The range-based correlation metric (see Sect. 3.2) is used to determine the correlation for clusters falling into the case (iii). Clusters in case (iv) are left as they are as these are not identified as similar to any other cluster in the other domain.

### Domain integration

The domain integration module is applied to produce the overall integrated model capturing the specific characteristics of the two domains. In this module, the cluster pairs, among which the correlations are identified from the previous step (the correlation module), are integrated (new mean, standard deviation, and cluster size are obtained for the integrated cluster). These are correlations in case (i), that is, when similarity is identified in both directions and filtered out pairs from case (iii) where range-based distance is less than the chosen threshold ( $\mathcal{T} = 0.45$ ). The threshold of 0.45 is chosen to ensure that the integrated clusters are not too far away from each other. It can be noted that the lower the threshold, the more strict the cluster correlation requirements. If the threshold is set to 0, only clusters with overlapping mean vectors are merged in the integration phase. As a result of the integration of the clusters based on the identified correlations, a clustering model is obtained, referred to as the integrated model, capturing the knowledge from both domains. The integrated clusters presenting knowledge of both domains are referred to as *common* clusters, and the rest containing knowledge of only a single domain are referred to as *private* clusters of the respective domain. The clustering solution composed of common and private clusters

from both domains is referred to as an *integrated* clustering solution.

**Domain adaptation**

The domain adaptation module uses the composable nature of the overall integrated clustering model to generate two adapted private domain models, one per domain. Each model adapts to the characteristics of its own domain but, at the same time, reflects the identified commonality between the two domains. These models are referred to as adapted models (e.g., source-adapted model and target-adapted model). The adapted models contain the domain-specific private clusters and the common clusters.

A high-level overview of the complete process of integrating knowledge from two different domains and generating both integrated and adapted models is presented in Fig. 2. Pseudo-code presenting the overview of DIBCA++ is presented in Algorithm 3.

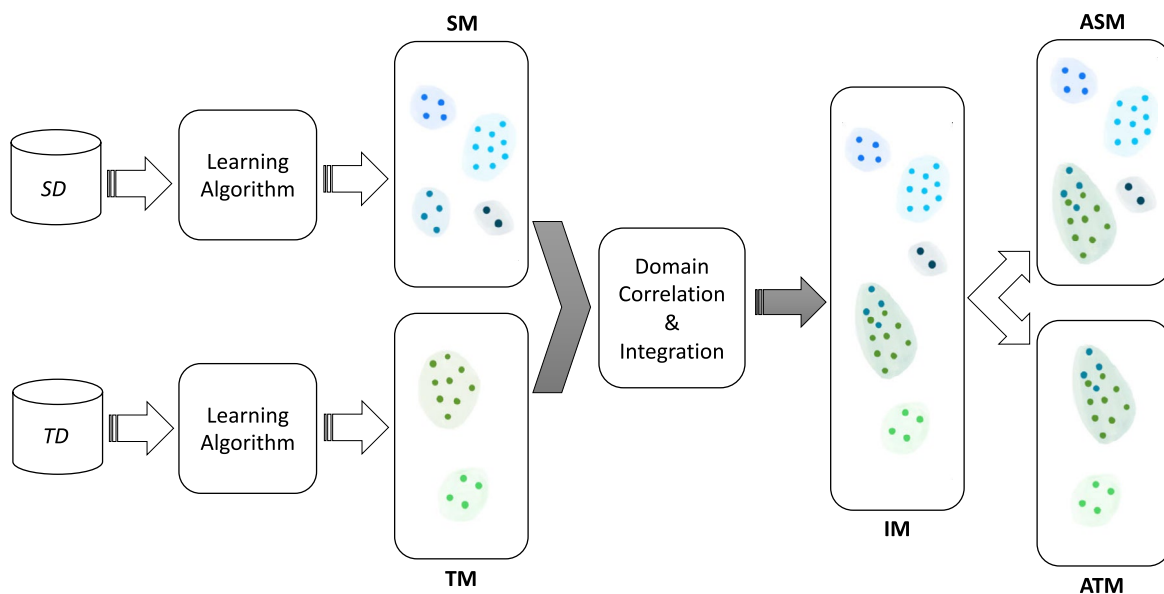
**3.4 Learning Algorithm**

As stated before, a clustering algorithm is required to be used in combination with DIBCA++. The algorithm introduced by Abghari et al. (2022) is improved to be robust to outliers and is proposed to be used in the current study.

The major improvements that are worth to be mentioned are as follows:

1. Instead of the low and high attribute value vectors (composed of the least and highest values of each attribute of the cluster), the modified version of the algorithm uses the mean, standard deviation, and size of the cluster as representatives.
2. Previously, the cluster boundaries were defined by the low and high value vectors, and the cluster’s center (mean or representative) is obtained by calculating the mean of these two vectors. This is modified such that the mean vector obtained using all the data points is used as the cluster representative, and the cluster boundaries are obtained by Chebyshev’s inequality using the mean and standard deviation of the clusters.
3. The mean vector obtained by using all the cluster data points is used to identify the closest cluster to the data point that needs to be assigned to a cluster. Previously, the mean of low and high value vectors is used for the same purpose.

The pseudo-code of the fitting and predict parts of the clustering algorithm (learning algorithm) used in DIBCA++ are presented in Algorithms 1 and 2. Note that this algorithm can be replaced with any other learning algorithm generating clusters represented by the mean, standard deviation, and size of the cluster.



**Fig. 2** High-level overview of building integrated and adapted models when data from two different domains are available. SD, TD, SM, TM, IM, ASM, and ATM stand for Source Data, Target Data, Source

Model, Target Model, Integrated Model, Adapted Source Model, and Adapted Target Model, respectively

**Algorithm 1** Learning Algorithm - Fitting

---

**Input:** Labeled data set  $D$

- 1: **for** each class (cluster) in  $D$  **do**
- 2:     Calculate  $m, \sigma, s$ .
- 3:     Obtain cluster boundaries ( $[m - k\sigma, m + k\sigma]$ ) using Chebyshev's inequality.
- 4: **end for**

**Output:** Representatives  $(m, \sigma, s)$  and Boundaries  $[m - k\sigma, m + k\sigma]$

---

**Algorithm 2** Learning Algorithm - Predict

---

**Input:** List of all cluster boundaries  $[m - k\sigma, m + k\sigma]$  and mean vectors  $(m)$ , data point  $(d)$

- 1: Identify the closest mean vector to  $d$ .
- 2: **if** all the feature values of  $d$  are within the cluster boundary of the selected cluster **then**
- 3:     Assign to the cluster.
- 4: **else**
- 5:     Assign -1.
- 6: **end if**

**Output:** Return the label for the data point.

---

**Algorithm 3** DIBCA++ Algorithm

---

**Input:** Clustering models  $C^1$  (source model) and  $C^2$  (target model) with cluster representatives obtained using Alg. 1.

- 1: Label  $m_{C_i^1}$ , ( $i = 1, \dots, n$ ) using  $C^2$  (Alg. 2)
- 2: Label  $m_{C_i^2}$ , ( $i = 1, \dots, o$ ) using  $C^1$  (Alg. 2)
- 3: **if** ( $m_{C_i^1} \in C_i^2$ )  $\wedge$  ( $m_{C_i^2} \in C_i^1$ ) **then**
- 4:     Correlation( $C_i^2, C_i^1$ ) = 0
- 5: **end if**
- 6: **for** each  $C_i^1 \in C^1$  **do**
- 7:     **for** each  $C_i^2 \in C^2$  **do**
- 8:         **if** ( $m_{C_i^2} \in C_i^1$ )  $\wedge$  (Correlation( $C_i^2, C_i^1$ ) <  $\mathcal{T} = 0.45$ ) (Using Eq. 2) **then**
- 9:             ClusterList.add( $C_i^2$ )
- 10:         **end if**
- 11:     **end for**
- 12:     **if** ClusterList  $\neq \emptyset$  **then**
- 13:         Integration( $C_i^1$ , ClusterList), calculate new  $m, \sigma, s$
- 14:         Common cluster of  $C^1$  and  $C^2$  is obtained.
- 15:     **end if**
- 16: **end for**
- 17: Clusters not integrated are private in their respective domains.

---

**3.5 Computational complexity**

The proposed algorithm is designed to be resource-efficient and operates using the cluster representatives of mean, standard deviation, and cluster size. In this section, the computational complexity of the main parts of the DIBCA++ algorithm is discussed. The complexity of the Fitting (Algorithm 1) and Predict (Algorithm 2) parts of the learning algorithm can be approximated to  $O(n_c)$  and  $O(n_d)$ , respectively, where  $n_c$  is the number of clusters or groups in the clustering solution and  $n_d$  is the number of data points in the test set. As stated before, it can be noted that the used learning algorithm, can be replaced with other suitable learning algorithms, and the computational complexity of this part varies based on the algorithm chosen.

The computational complexity of the correlation module of the algorithm can be approximated to  $O(no)$ , where  $n$  and  $o$  are the numbers of clusters in the source and target domains, respectively. The computational complexity of the integration phase can be approximated to  $O(n_{corr})$ , where  $n_{corr}$  is the number of the identified correlations between the source and target domain. In conclusion, the computational complexity of DIBCA++ is similar to that of the initial DIBCA (Devagiri et al. 2022).

**3.6 Algorithm explainability**

In this section, we discuss the DIBCA++ algorithm with respect to its explainability characteristics. Explainable Artificial Intelligence, also referred to as XAI in the literature, is a field stating the importance and necessity of Artificial Intelligence (AI) and Machine Learning (ML) models to be explainable and transparent. Explaining the results of AI algorithms is important for their trustworthiness and acceptance, especially if these algorithms are used to make essential decisions in fields like medicine, law, etc. Explainability increases AI acceptance even in areas such as recommender systems where incorrect predictions have less impact on users (Ribeiro et al. 2016).

On a broader level, XAI can be grouped into two categories, post-hoc and ante-hoc (Srihari 2020). In post-hoc methods, the algorithm itself is not explainable. The algorithm's results are analyzed using various techniques like plots, test cases, etc., to make them explainable. Many new algorithms are also being proposed to be used as an add-on to help explain the results of the main algorithm. On the contrary, ante-hoc algorithms are inherently explainable and inbuilt with the explainability factors from the design stage.

The proposed DIBCA++ algorithm is an ante-hoc algorithm, as different steps of the algorithm can be analyzed and interpreted. Following are a few examples. In the cross-labeling phase, we are able to initially identify the similarity of clusters from the two different domains. Let us assume that the clustering solutions in the source and target domains are called  $C_1$  and  $C_2$ , respectively. When a cluster representative in  $C_1$  is labeled as one of the clusters in  $C_2$ , it implies the clusters have a resemblance, as the representative of the cluster in  $C_1$  is within the range of the boundaries of the cluster in  $C_2$ , and vice-versa which is logical. The correlation between two clusters is further investigated by the various checks done in the algorithm. These checks are transparent, enabling an easy understanding of the final result, i.e., if the clusters should be merged (common clusters) or retained as they are (private clusters in respective domains).

In addition to the above, the output from different steps of the algorithm can be easily visualized to further support the interpretation and understanding of the obtained final results. Overall, the final results of the algorithm execution are interpretable and easier to understand. These results are unique and can also be traced back for reasoning if required.

**3.7 Algorithm applicability**

The DIBCA++ proposes a generic integration procedure of two clustering solutions based on cluster or class data models, in which each cluster or class is presented by its boundaries based on Chebyshev's inequality. Only the cluster's mean, standard deviation, and size are used as

representatives. The algorithm is applicable to domain adaptation problems in a variety of ML scenarios, namely supervised, unsupervised, and semi-supervised learning scenarios. For example, in an unsupervised scenario, both domain models are presented by clustering solutions in which each cluster models a specific behavioral scenario of the monitored phenomenon. In that way, the integrated model built by the DIBCA++ will be enriched with knowledge about the phenomenon behavior from both domains by identifying and refining scenarios common for both domains and preserving those unique for each separate domain. Such a domain adaptation model can be suitable for outlier detection tasks, where the monitored phenomenon (e.g., a process or a system) can be described by a set of clusters presenting its ordinary behavioral patterns. Everything that does not fit into any cluster will be interpreted as deviating behavior. Hence, the DIBCA++ can be applied to adapt the outlier detection model to new environmental or contextual conditions.

In the case of a supervised learning setup, both domain data models consist of labeled classes, i.e., each model presents the concepts specific to each domain, respectively. As a result of the integration procedure, the overall model will learn which concepts are common for both domains and also identify the private ones for each domain. This newly gained information will contribute to a better understanding of the monitored phenomenon and can eventually be used for further improvement of domain models. For example, domain private concepts can be extracted from the integrated data model. This property of the DIBCA++ makes it applicable, for example, for the customization of patient/user models in smart monitoring applications.

Finally, the DIBCA++ can be used in partially (semi-) supervised scenarios, where only the source model is based on labeled data, while the target model contains just non-labeled clusters. One use case of this scenario is the automatic data labeling task. The domain correlation procedure of the algorithm can be used to label those target concepts that have been identified to be common with the source model. The potential of the proposed algorithm in labeling scenarios has already been studied and demonstrated in the initial work on DIBCA (Devagiri et al. 2022).

In addition to the above-discussed ML scenarios where DIBCA++ can be used, there are some limitations that are worth mentioning. More specifically, the learning algorithm used in the current study applies Chebyshev's inequality to define the cluster boundaries. Even though this inequality holds true for most probability distributions, there might be cases when this is invalid. For example, if we consider normal distribution, which is one of the most common probability distributions, the majority (approx. 99.7%) of the data points lie within three times the standard deviation from the mean. Whereas for Chebyshev's inequality, six and ten times the standard deviation from the mean cover approximately 92.7%

and 99% of the data points, respectively. If the data follows a normal distribution, Chebyshev's inequality might produce clusters with higher boundaries than required; therefore, more attention should be paid to such cases. This can lead to mislabeling of data, especially when the clusters are closer to each other. Moreover, the learning algorithm used in conjunction with DIBCA++ should be able to represent the clusters by the mean, standard deviation, and size of the cluster.

## 4 Data sets

Two different data sets, one from the smart logistics use case and the other from the HAR data set, are used in the study.

Data from the smart logistics use case is provided by our industry partner. The Global Navigation Satellite System (GNSS) is used by the trackers to report their location. These tracking devices have limited memory and computational capacity, which needs to be considered during their operation. Trying to update the location continuously requires a significant amount of energy, which is wasted if the GNSS signal is not strong or unavailable when accessed. This motivates the need to detect and optimize when the tracker should try to access the GNSS location. The used data set contains information about five tracking devices. For each tracking device, signals from the Long Term Evolution (LTE) base stations, the number of available cells with radio signals, geographical location, etc., are collected.

DaLiAc (Daily Life Activities), a HAR data set generated during a study (Leutheuser et al. 2013), is used to evaluate the algorithm's potential in the domain adaptation task of recognizing and mapping similar activities from different users. HAR of different users can be categorized as a domain adaptation task as each user performs an activity in different ways, leading to having data with different distributions. Building new clustering solution covering all the activities considered for each user requires a large amount of data, which is difficult to obtain. Domain adaptation is useful in such situations where knowledge from one user model is used for building HAR models for others. The data set contains information about 19 participants (11 male, 8 female) aged between 18 and 55 performing 13 daily activities. Data consists of the accelerometer and gyroscope readings collected using four sensors placed on the right hip, chest, right wrist, and left ankle of each participant.

## 5 Evaluation measures

For the smart logistics use case, Adjusted Mutual Information is used, whereas, for the DaLiAc data set, two different extrinsic cluster evaluation metrics, namely Adjusted Rand Index and Adjusted Mutual Information are considered. The

implemented versions of the evaluation metrics available in scikit-learn by Pedregosa et al. (2011) are used.

### 5.1 Adjusted Rand index

Adjusted Rand Index (ARI) (Hubert and Arabie 1985) is the adjusted for chance version of the Rand Index (RI) (Rand 1971). RI calculates the similarity between two clustering solutions by counting the number of pairs of samples that are categorized as the same or different in the predicted clustering as opposed to the ground truth. ARI is corrected/adjusted for a chance so that random results produce a score close to 0. ARI is obtained using the formula stated in Eq. 3.

$$ARI = \frac{RI - Expected\ RI}{max(RI) - Expected\ RI}, \tag{3}$$

where RI for samples  $i$  and  $j$  is  $\sum_{ij} \binom{n_{ij}}{2}$ ,

$$max(RI) = \frac{1}{2} \left[ \sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right], \text{ and}$$

$$ExpectedRI = \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}, \text{ where } n \text{ is the number}$$

of elements,  $a_i$  and  $b_j$  are the sum of the pair of samples  $i$  and  $j$ , respectively.

### 5.2 Adjusted mutual information

Mutual Information (MI) is used to compare two clustering solutions. The obtained value is, however, higher if the number of clusters is high, whether more information is actually shared between the clusters or not. Adjusted Mutual Information (AMI) (Vinh et al. 2009, 2010) is a corrected for chance version of MI. Equation 4 is used to calculate AMI (Pedregosa et al. 2011).

$$AMI(U, V) = \frac{[MI - E(MI)]}{[avg(H(U), H(V)) - E(MI)]} \tag{4}$$

In Eq. 4,  $MI$  represents  $MI(U, V)$ , where  $U$  and  $V$  are the two clustering solutions,  $H(\cdot)$  is the entropy of the considered clustering solution (either  $U$  or  $V$  in this case),  $E(MI(U, V))$  is the expected mutual information between two random clustering solutions  $U$  and  $V$ .

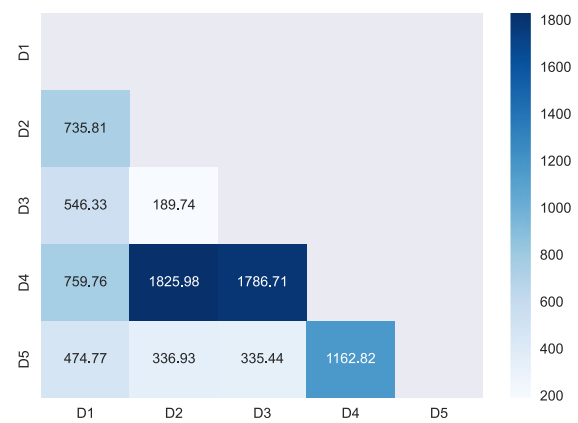


Fig. 3 Heat-map presenting the similarity between data obtained from different devices (D) based on the Optimal Transport Dataset Distance (OTDD). Low values represent that the data sets are similar to each other

## 6 Experiments

The section initially presents an overview of the preprocessing steps done on each of the considered data sets. This is followed by explaining the different experiments conducted in the study.<sup>1</sup>

### 6.1 Data pre-processing

Before conducting the experiments, a few steps of data processing are done on the data sets to make them ready to be used in the experiments.

For the first set of experiments, we used the data set from a smart logistics use case provided by one of our industrial partners. Data used in this study is collected from five different tracker devices. Statistical values of the cell signals, i.e., mean, standard deviation, min, and max, are obtained for each instance, which along with the signal strengths, are used as a feature set.

The second set of experiments is conducted on the DaLiAc data set initially containing 24 attributes, with six attributes (3D accelerometer and gyroscope readings) obtained from each sensor placed on different body locations. Each of the 3D values is aggregated using  $a = \sqrt{x^2 + y^2 + z^2}$  to obtain one value like in Lu et al. (2021) and Wang et al. (2018) studies, thus giving in a total of eight attributes.

To obtain stable results, each experiment is conducted three and five times on each data set for smart logistics and HAR use cases, respectively. Since the smart logistics and DaLiAc data sets contain time series data, TimeSeriesSplit from the scikit-learn library (Pedregosa et al. 2011) is used to split the data into five different folds of train and test data

<sup>1</sup> Access to the repository with experiments conducted on the public data set (DaLiAc, HAR) can be provided upon request.

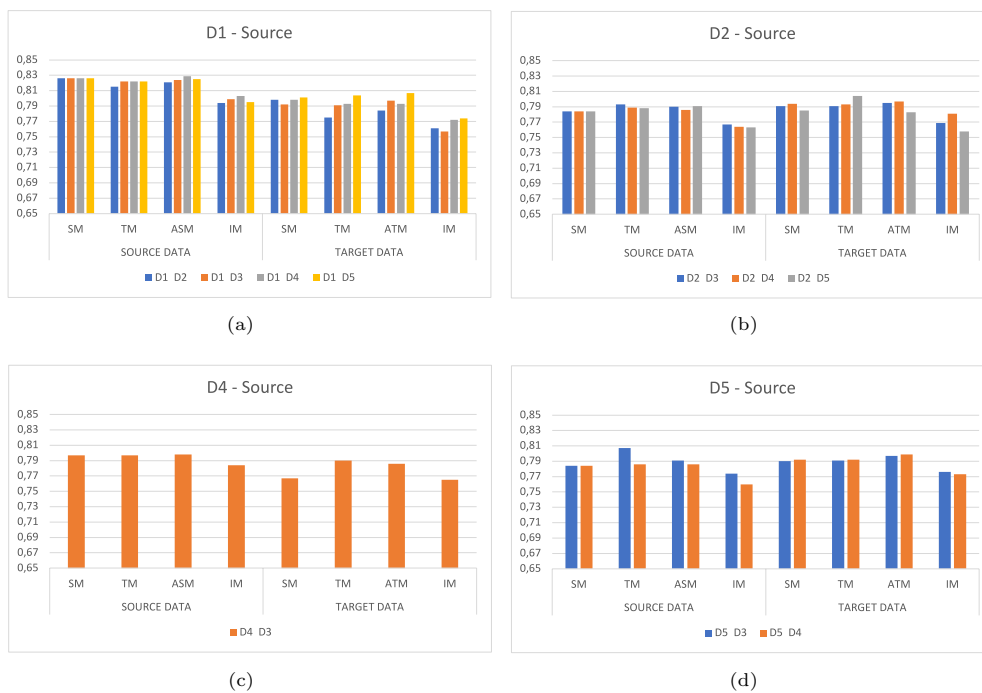
sets. Only three folds are used for the smart logistics use case, as low performance was observed in the initial folds where less data is used. Note that training data is incrementally added to each fold in TimeSeriesSplit. In the next step, data from each fold is standardized using  $z$ -score. This is done using the StandardScaler module from scikit-learn. Data standardization of both source and target data is done using the scale obtained from the data of the source domain. Mathematically, it can be represented as  $z = (x - \mu) / \sigma$ , where  $x$  is the data point that needs to be standardized,  $\mu$  is the mean of the source training data, and  $\sigma$  is the standard deviation of the source training data. Unless explicitly stated, all the results presented are aggregated evaluations obtained from experiments conducted on different folds (three for smart logistics and five for HAR data sets) of the time series cross-validation performed.

### 6.2 Experiments on smart logistics use case

We study and analyze the effect of similarity between the two domains to that of the performance of DIBCA++ with the experiments discussed in this section. The similarity between different domains (data from different devices) is evaluated using the Optimal Transportation Dataset Distance (OTDD) (Alvarez-Melis and Fusi 2020). The OTDD is based on optimal transport and also incorporates label information in the calculation. The OTDD is designed to evaluate

the transferability of ML models between data sets, which is useful information to have for tasks like transfer learning and domain adaptation. Figure 3 presents a heat map showcasing the data set distances between different pairs of devices. When calculating the OTDD, a sampled data set of 7, 500 instances with random seed '0' is used. Values closer to zero represent that the data sets are similar to each other. It is also interesting to observe that the results of similarity based on OTDD are comparable to the results obtained by Abghari et al. (2022), where the similarity between the same data sets is calculated based on the RI scores.

Two experiments, A1 and A2, are conducted to understand and analyze how the similarity of the data sets influences the performance of the different clustering models produced by DIBCA++. Evaluation measure AMI (see Sect. 5) is used to showcase the performance of the clustering solutions obtained for both A1 and A2. Note that we have also done an evaluation using ARI, but since there is no significant difference in the results generated by both evaluation metrics on the various models produced by DIBCA++, only AMI results are reported in the paper. In addition, the current work treats the GNSS activation problem differently from the setting applied by Devagiri et al. (2022). Namely, in the previous study, we have considered the prediction of the availability of GPS signal as a one-class classification problem, i.e., a version of the ISM algorithm published by Abghari et al. (2022) is applied to the GPS annotated data



**Fig. 4** Experiment A1-DIBCA++: Visualisation and comparison of the performance (all the values are average of 3 folds) of the DIBCA++ models for different combinations of source and target

data sets with respect to Adjusted Mutual Information. Each plot depicts the performance of the models when a different device is in the role of the source

to learn the normal system behavior and then recognizes everything different from it, such as when there is no GPS signal coverage. In the current study, we use the data from both classes and model each class by seven categories. Each category is presented by a unique cluster containing all the data points associated with the availability of the number of cells with signal strengths (ranging from 1 to 7 available

cells based on the LTE coverage). As a result, the data will be presented in 14 categories in total. In both experiments A1 and A2, these 14 categories are used as true labels, based on which the initial clustering solutions are obtained using the learning algorithm (Sect. 3.4, Algorithms 1 and 2). Only 70% of the training data of the target is used for training in



**Fig. 5** Experiment A2-DIBCA++: Performance comparison of the closet and distinct pairs of devices based on the similarity of the data sets. The notation Dx-Dy represents that Dx is the source and Dy is the target. Each plot presents the performance (using

Adjusted Mutual Information) of different models produced by DIBCA++ on **a** D2-D3 and D3-D2, closest pair based on similarity; **b** D2-D4 and D4-D2, distinct pair based on similarity

**Table 2** Information about users used in the experiments B1 and B2

User	Sex	age	Instances	Height	Weight	Handedness
U8	m	29	230,381	180	90	right
U10	f	27	240,494	169	59	right
U11	m	27	239,020	178	72	right
U12	m	18	253,317	175	70	right
U13	f	21	245,041	177	86	right
U14	f	22	243,036	180	55	left
U15	m	27	241,024	196	95	right
U18	f	24	228,697	158	54	right

**Table 3** Experiment B1-DIBCA++: Results obtained on DaLiAc data set with respect to Adjusted Rand Index using DIBCA++; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM-Adapted Source Model, ATM- Adapted Target Model,

IM- Integrated Model. Bold font in black represents the best model in the row, cyan color represents the better performance of IM in comparison to the corresponding SM or TM. Blue and red represent the better performance of ASM over SM and ATM over TM, respectively

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.248	0.222	<b>0.271</b>	<b>0.271</b>	0.201	0.261	0.248	0.233
U11	U8	0.280	0.230	<b>0.296</b>	<b>0.292</b>	0.230	0.272	<b>0.302</b>	0.256
U12	U8	<b>0.299</b>	0.236	0.288	0.288	0.236	0.260	<b>0.261</b>	0.257
U12	U11	<b>0.299</b>	0.210	0.284	0.267	0.220	0.276	<b>0.292</b>	<b>0.280</b>
U12	U15	0.299	0.220	<b>0.303</b>	<b>0.301</b>	0.250	0.264	<b>0.292</b>	<b>0.282</b>
U13	U10	<b>0.289</b>	0.196	0.274	0.274	0.235	0.246	0.237	<b>0.251</b>
U13	U14	0.289	0.255	<b>0.297</b>	<b>0.296</b>	0.250	0.283	0.262	0.246
U13	U18	<b>0.289</b>	0.232	0.278	0.278	0.266	0.265	<b>0.272</b>	0.264
U14	U10	<b>0.300</b>	0.186	0.287	0.279	0.195	0.262	<b>0.273</b>	<b>0.282</b>
U14	U18	<b>0.300</b>	0.227	0.278	0.271	0.252	0.277	<b>0.293</b>	0.268
U15	U8	0.274	0.227	<b>0.277</b>	<b>0.277</b>	0.215	<b>0.291</b>	0.290	0.256
U15	U11	0.274	0.164	0.270	0.267	0.245	0.275	<b>0.285</b>	0.259

**Table 4** Experiment B1-DIBCA++: Results obtained on DaLiAc data set with respect to Adjusted Mutual Information using DIBCA++; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM-Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model. Bold font represents the best

model in the row, cyan color represents the better performance of IM in comparison to the corresponding SM or TM. Blue and red represent the better performance of ASM over SM and ATM over TM, respectively

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.441	0.421	<b>0.451</b>	<b>0.451</b>	0.400	0.433	0.434	0.417
U11	U8	0.463	0.406	<b>0.474</b>	<b>0.472</b>	0.403	0.445	0.452	0.439
U12	U8	<b>0.459</b>	0.426	0.447	0.447	0.414	0.432	<b>0.433</b>	0.430
U12	U11	0.459	0.391	0.450	0.445	0.386	0.460	<b>0.471</b>	0.459
U12	U15	0.459	0.407	<b>0.461</b>	<b>0.463</b>	0.444	<b>0.475</b>	0.473	0.467
U13	U10	0.441	0.387	<b>0.443</b>	<b>0.443</b>	0.435	<b>0.444</b>	0.428	0.440
U13	U14	0.441	0.405	<b>0.441</b>	0.440	0.446	<b>0.457</b>	0.443	0.448
U13	U18	0.441	0.413	<b>0.441</b>	<b>0.441</b>	0.435	0.435	<b>0.438</b>	<b>0.436</b>
U14	U10	<b>0.475</b>	0.392	0.464	0.467	0.387	0.459	<b>0.459</b>	<b>0.459</b>
U14	U18	<b>0.475</b>	0.443	0.467	<b>0.475</b>	0.431	0.449	<b>0.458</b>	<b>0.452</b>
U15	U8	<b>0.480</b>	0.425	0.467	0.466	0.417	0.463	0.456	0.446
U15	U11	<b>0.480</b>	0.377	0.477	0.476	0.432	0.466	0.454	0.444

**Table 5** Experiment B2: Results obtained on DaLiAc data set when two data sets are combined. Red and blue represent the best and worst performance values, respectively

Users	ARI	AMI
U10 + U18	<b>0.223</b>	0.404
U11 + U8	0.246	0.410
U12 + U8	0.276	0.430
U12 + U11	0.278	0.436
U12 + U15	<b>0.286</b>	<b>0.440</b>
U13 + U10	0.237	<b>0.401</b>
U13 + U14	0.277	0.428
U13 + U18	0.250	0.405
U14 + U10	0.262	0.434
U14 + U18	0.255	0.417
U15 + U8	0.231	0.404
U15 + U11	0.262	0.434

both experiments (A1 and A2). This is done to generate a scenario where sufficient target data is unavailable.

In experiment A1, all ten possible combinations of two-pair devices are considered to perform the experiments. For each pair of devices, the device with a higher number of instances is considered as the source (as more data implies more knowledge) domain, and the one with less data is the target domain. Figure 4 presents the results obtained.

In experiment A2, the closest and the distant pair of devices based on the similarity calculated are considered for comparison. Unlike A1, in this experiment, for each pair of devices, both are alternatively considered as the source and the other as the target, thus giving 4 different combinations of source and target (2 for each closest and distant pair). The results of this experiment are presented in Fig. 5.

### 6.3 Experiments on HAR use case

The potential of the DIBCA++ algorithm in the adaptation and personalization of users’ human activity clustering models is showcased in this experiment. Since the number

**Table 6** Experiment A1-DIBCA: Results obtained on smart logistics data set with respect to Adjusted Mutual Information using DIBCA; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM-Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
D1	D2	0.792	0.810	0.787	0.787	0.755	0.764	0.760	0.757
D1	D3	0.792	0.806	0.791	0.791	0.757	0.760	0.762	0.760
D1	D4	0.792	0.789	0.788	0.788	0.778	0.766	0.811	0.787
D1	D5	0.792	0.791	0.787	0.787	0.764	0.772	0.764	0.763
D2	D3	0.768	0.756	0.768	0.768	0.781	0.763	0.780	0.780
D2	D4	0.768	0.746	0.771	0.771	0.786	0.766	0.791	0.789
D2	D5	0.768	0.742	0.763	0.763	0.785	0.772	0.779	0.779
D4	D3	0.784	0.784	0.792	0.792	0.738	0.762	0.775	0.775
D5	D3	0.763	0.766	0.775	0.775	0.760	0.759	0.766	0.766
D5	D4	0.763	0.780	0.753	0.753	0.787	0.767	0.794	0.780

**Table 7** Experiment A1-DIBCA++: Results obtained on smart logistics data set with respect to Adjusted Mutual Information using DIBCA++; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM- Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
D1	D2	0.826	0.815	0.821	0.794	0.798	0.775	0.784	0.761
D1	D3	0.826	0.822	0.824	0.799	0.792	0.791	0.797	0.757
D1	D4	0.826	0.822	0.829	0.803	0.798	0.793	0.793	0.772
D1	D5	0.826	0.822	0.825	0.795	0.801	0.804	0.807	0.774
D2	D3	0.784	0.793	0.790	0.767	0.791	0.791	0.795	0.769
D2	D4	0.784	0.789	0.786	0.764	0.794	0.793	0.797	0.781
D2	D5	0.784	0.788	0.791	0.763	0.785	0.804	0.783	0.758
D4	D3	0.797	0.797	0.798	0.784	0.767	0.790	0.786	0.765
D5	D3	0.784	0.807	0.791	0.774	0.790	0.791	0.797	0.776
D5	D4	0.784	0.786	0.786	0.760	0.792	0.792	0.799	0.773

**Table 8** Experiment B1-DIBCA: Results obtained on DaLiAc data set with respect to Adjusted Rand Index using DIBCA; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM- Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.107	0.064	0.109	0.090	-0.002	0.070	0.080	-0.002
U11	U8	0.030	0.103	0.027	0.026	-0.000	0.134	0.131	-0.000
U12	U8	0.047	0.079	0.046	0.048	0.021	0.135	0.129	0.028
U12	U11	0.047	0.001	0.050	0.027	-0.007	0.030	0.036	0.013
U12	U15	0.047	0.049	0.047	0.051	-0.006	0.101	0.124	0.046
U13	U10	0.103	0.003	0.083	0.001	0.072	0.118	0.101	0.097
U13	U14	0.103	0.039	0.104	0.033	0.037	0.089	0.091	0.091
U13	U18	0.103	0.070	0.118	0.105	0.099	0.072	0.109	0.111
U14	U10	0.091	0.004	0.087	0.093	0.041	0.115	0.178	0.072
U14	U18	0.091	0.079	0.091	0.092	0.038	0.074	0.072	0.041
U15	U8	0.097	0.130	0.097	0.099	0.074	0.136	0.145	0.080
U15	U11	0.097	-0.015	0.122	0.122	0.051	0.030	0.099	0.101

**Table 9** Experiment B1-DIBCA: Results obtained on DaLiAc data set with respect to Adjusted Mutual Information using DIBCA; SD- Source Data, TD- Target Data, SM- Source Model, TM- Target Model, ASM- Adapted Source Model, ATM- Adapted Target Model, IM- Integrated Model

SD	TD	SD-SM	SD-TM	SD-ASM	SD-IM	TD-SM	TD-TM	TD-ATM	TD-IM
U10	U18	0.353	0.263	0.351	0.330	0.111	0.292	0.286	0.122
U11	U8	0.250	0.224	0.245	0.242	0.119	0.282	0.290	0.121
U12	U8	0.262	0.208	0.251	0.258	0.175	0.282	0.285	0.195
U12	U11	0.262	0.107	0.260	0.214	0.116	0.249	0.251	0.183
U12	U15	0.262	0.233	0.256	0.273	0.110	0.322	0.333	0.221
U13	U10	0.270	0.075	0.250	0.123	0.235	0.352	0.318	0.326
U13	U14	0.270	0.168	0.270	0.206	0.156	0.304	0.295	0.310
U13	U18	0.270	0.233	0.277	0.271	0.272	0.297	0.297	0.289
U14	U10	0.321	0.065	0.300	0.292	0.201	0.351	0.395	0.244
U14	U18	0.321	0.254	0.315	0.320	0.227	0.304	0.296	0.252
U15	U8	0.317	0.282	0.318	0.319	0.291	0.285	0.300	0.300
U15	U11	0.317	0.113	0.314	0.320	0.246	0.254	0.315	0.324

of ways of selecting source and target from 19 participants would result in too many combinations, 4 male (users 8, 11, 12, and 15) and 4 female (users 10, 13, 14, 18), users are randomly chosen from the available participants on which the experiments are conducted. Since men and women have different physical capabilities which can impact the way they

perform the activities, we chose to perform the experiments on men and women separately. Further information about the users used in the experiments of this study is stated in Table 2.

The considered users can be grouped into 12 pairs such that one user’s activities are considered as a source domain and the other as a target domain (6 male pairs and 6 female

pairs). Two different types of experiments (denoted as B1 and B2) are conducted.

Experiment B1 is conducted to showcase the potential of the algorithm in domain adaptation tasks. In this experiment, for each pair of users, the user with a higher number of instances is considered as the source domain, and the other user is the target. Initial clustering in each of the domains is done using the true labels, and the learning algorithm stated in Sect. 3.4 (see Algorithm 1). This, along with Algorithm 2, is used to assign new data points to one of the clusters. The results of this experiment are presented in Tables 3 and 4 with regard to ARI and AMI, respectively.

In experiment B2, for each pair of users, data from both source and target domains are combined to obtain one dataset. A clustering model on the whole combined data is built and evaluated using the learning algorithm (Algorithms 1 and 2). Experimental results of B2 are presented in Table 5. Results of Experiment B2 when compared with B1 (SD-ASM, SD-IM, TD-ATM, and TD-IM columns of Tables 3 and 4), can be used to understand and demonstrate the importance of personalized user models.

#### 6.4 Experiments with DIBCA

The performance of DIBCA++ is also compared with its predecessor, i.e., DIBCA. For this purpose, two experimental scenarios, A1 and B1, are exactly replicated on the smart logistics and HAR use cases using DIBCA. Tables 6 and 7 present the experimental results of A1 on smart logistics data set with regard to AMI using both DIBCA and DIBCA++, respectively. It can be noted that the contents of Table 7 have already been presented as a figure (Fig. 4), but in order to facilitate easy comparison between DIBCA and DIBCA++, we have decided to present the results of both the algorithms using tables. Tables 8 and 9 present the experimental results of B1 using DIBCA on the HAR data set based on ARI and AMI, respectively.

### 7 Result analysis and discussion

Discussion and analysis of the experimental results are presented in this section. In addition, the algorithm's explainability aspect is also showcased on a few selected instances of different experiments in Sect. 7.4. The abbreviations Source Data (SD), Target Data (TD), Source Model (SM), Target Model (TM), Adapted Source Model (ASM), Adapted Target Model (ATM), and Integrated Model (IM) are used in this section.

#### 7.1 Smart logistics

A heat map with similarities calculated for each pair of different data sets is displayed in Fig. 3. In experiment A1, the performance of the models produced by the DIBCA++ on the different pairs of devices is analyzed in light of these calculated similarities. Figure 4 presents the built models' performance on different source and target device pairs. From all the sub-figures, it can be observed that the performance of ASM when D1 is considered as the source is higher compared to all the other scenarios. It can be backed by the known fact that D1 contains richer data than its counterparts. Based on the calculated similarities between pairs of devices, when D1 is selected as the source, pairs D1-D3 and D1-D5 can be considered similar, D1-D2 and D1-D4 to be distinct from each other. From Fig. 4a, it can be observed that the performance of the ASM and ATM is directly proportional to the similarity of the data sets (i.e., higher the similarity between the two data sets (lower OTDD), higher the performance), except for D1-D4 pair concerning ASM. Whereas for scenarios when devices D2 and D5 act as a source (D2-D3, D2-D5, and D5-D3 are considered similar; D2-D4 and D5-D4 are distinct), as shown in Figs. 4b and 4d, the performance of ASM is directly proportional to similarity, and the performance of ATM is inversely proportional. Since there is just one pair when D4 is the source (Fig. 4c), it is difficult to interpret how the similarity and the performance of ASM and ATM are correlated. Overall, it is observed that the performance of the ASM and ATM models produced by DIBCA++ performed better 70% and 80% of the time when compared to that of SM and TM, respectively (see Fig. 4 or Table 7). These results on the studied use case show the potential of DIBCA++ to transfer knowledge between domains for improving performance. Note that all the 3 cases (30%) when ASM performance did not exceed the performance of SM are when D1 is considered as the source. It could be because of the fact that the SM build using D1 is already rich and could not gain much from the other domain.

In experiment A2, we specially inspect two pairs of devices, the closest (similar) and the distant (dissimilar) ones; see Figs. 5a and 5b, respectively. D2 and D3 are the two closest devices, and the experiments on both D2-D3 and D3-D2 pairs show better performance of ATM than that of TM, on TD. In the case of D2-D3, the ATM even outperforms SM on TD. So, when two devices are very similar, it does not matter which device is in the role of the source. However, this is not observed in the case of the two most distant devices, which are D2 and D4. The data collected by these devices presumably cover very different GNSS coverage scenarios. The experiments demonstrate strange behavior for D4-D2 on TD; namely, although the performance of SM and TM, ASM on SD are comparable, this does not

repeat on TD, while in the case of D2-D4, the ATM outperforms all other models on both source and target data. The above might be due not only to the fact that D2 and D4 have very different data sets (the highest OTDD value) but also because D2 has much richer data than D4. It is interesting to note that D2 has more instances in the data set than D4.

Evidently, it can be stated that the target model acquires knowledge from the source and improves its performance. When the knowledge in the source domain is very similar to that in the target, that is, there is no additional knowledge to be transferred, and the benefit of creating IM or ATM is not significant. In such a scenario, using the SM directly on the target domain data would be better. In addition to the above, we have noticed that the performance of the adapted (source or target) model is better than that of the integrated model in all the studied combinations, i.e., it is always recommendable to use the customized model.

## 7.2 Human activity recognition

Tables 3 and 4 summarized the results of DIBCA++ on the DaLiAc data set in experimental scenario B1. They showcase the results of the comparison of the performance of the three models, namely IM, ASM, and ATM, produced by the proposed algorithm to that of the initial SM and TM models used on the SD and TD in terms of ARI and AMI measures, respectively. In that way, for each of the 12 combinations of users studied, we compare eight different scenarios for each metric, i.e., 96 different scenarios per metric in total.

With respect to the ARI evaluation (see Table 3), one can notice that the IM produces better results in comparison to the two initial SM and TM in 5 and 4 (underlined numbers) of all the combinations, respectively. It is interesting, however, to observe that the same number is achieved by the ASM in comparison with the SM used on the source data (the scores in bold-italic) while the ATM is better than the TM applied on the target data (the scores in italic) in twice more combinations, i.e., 8. Evidently, the performance of the integrated model is further improved by building a personalized adapted model for each domain. In addition, we can see that using the SM on the target data (see column TD-SM) produces worse results than the initial TM as well as than the ATM and IM. This clearly shows that our domain integration procedure leads to richer adapted models, with benefits for both parties involved. It can also be observed that the SD-SM scenario performs better than the rest studied in 6 different combinations highlighted in bold font.

The same 96 scenarios discussed above are also evaluated in terms of AMI measure, and the obtained results are presented in Table 4. These confirm once more the findings extracted by analyzing Table 3. It can be observed that the IM has performed better in 6 and 3 cases when compared against SM and TM on source and target data, respectively.

Similarly to the results corresponding to the ARI, we can see that ASM performs better than SM the same number of times as IM, and ATM performs better than TM in 5 combinations, which is almost twice the times IM performs better than TM.

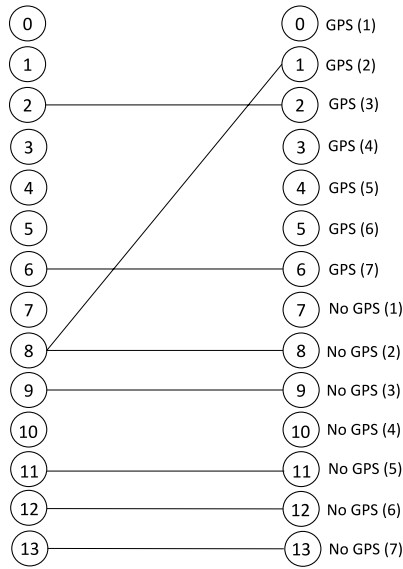
Table 5 reports the evaluation results of the clustering models built directly on the combined data of a pair of users (domains) with respect to the two evaluation measures, ARI and AMI (Experiment B2). This experiment showcases the need for personalized user models for higher performance. When the results of combined data sets of two users are compared to the scenarios where source and target domain models are built separately (Tables 3 and 4, columns SD-SM, SD-ASM, SD-IM, TD-TM, TD-ATM and TD-IM) drop in performance of the combined models can be observed in most cases. The evaluation results of the scenarios where TM is used on SD or SM on TD (columns SD-TM and TD-SM of Tables 3 and 4) are even worse, but it is expected to have the least performance in this case as the model's knowledge and the data used are from different domains.

The clustering model produced on the combined data of users U12 and U15 (see row U12+U15) outperforms all other pairs with respect to all the used measures, while the models of U10+U18 and U13+U10 are the worst-performing ones since they have generated the lowest results at least for a single evaluation measure. It is interesting to notice that the best-performing pair of models is obtained from males, as users in this pair may have performed most of the activities similarly, while the two low-performing models are based on data from female users, which might have occurred due to differences in how they perform the activities. It can also be observed that the U12-U15 pair is among one of the best-performing pairs in Tables 3 and 4.

## 7.3 Comparison with DIBCA

In this sub-section, the performance of DIBCA++ is compared to DIBCA with respect to the experimental scenarios A1 and B1. Tables 6 and 7 present the performance of DIBCA and DIBCA++ in experimental setup A1 on smart logistics use case based on AMI. It can be observed that DIBCA++ has a better performance in 82.5% of cases, and in 2.5% occasions, both algorithms performed similarly. In only 15% of the cases, DIBCA has shown better performance, and it can be noted that the margin of difference is also very small in some of these cases.

Tables 8 and 9 present the summarised results of DIBCA on the HAR data set with respect to ARI and AMI. On comparing these results with those obtained by DIBCA++ (Tables 8 and 3; Tables 9 and 4), it can be clearly seen that DIBCA++ has outperformed DIBCA in all the cases to a great extent. This might be because of more outliers



**Fig. 6** Visualisation of the correlations identified by the domain correlation module of the DIBCA++ on a selected pair of devices, D2-D5, one of the pairs on which ATM has the worst performance. In the cluster label 'A(x)', A indicates whether the cluster corresponds to one of the two main classes, i.e. available GPS signal or missing, while x ranges from 1 to 7 representing the available cells based on the LTE coverage, i.e., it denotes a specific category (scenario) in each one of the two main classes

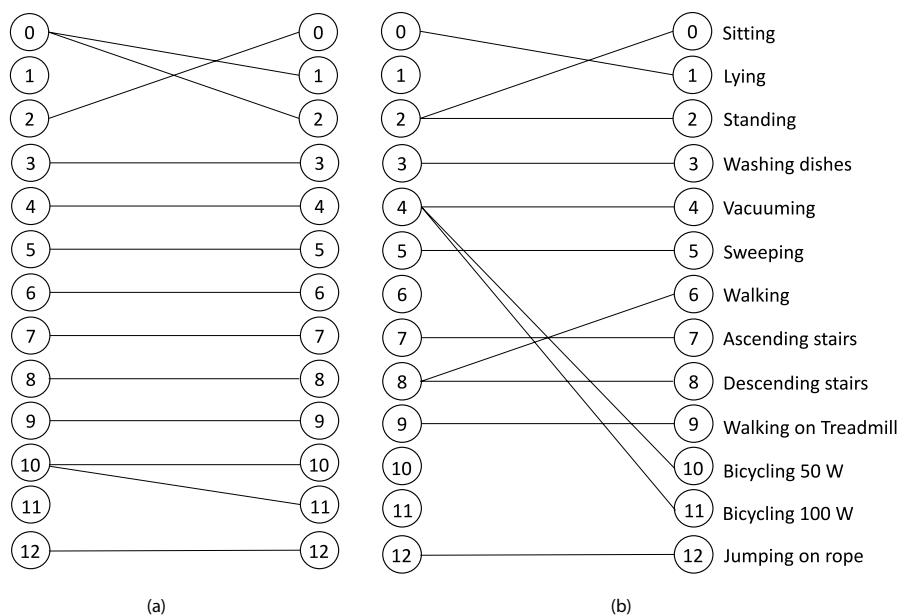
recorded when the sensors capture human activities, as different people perform the activities differently.

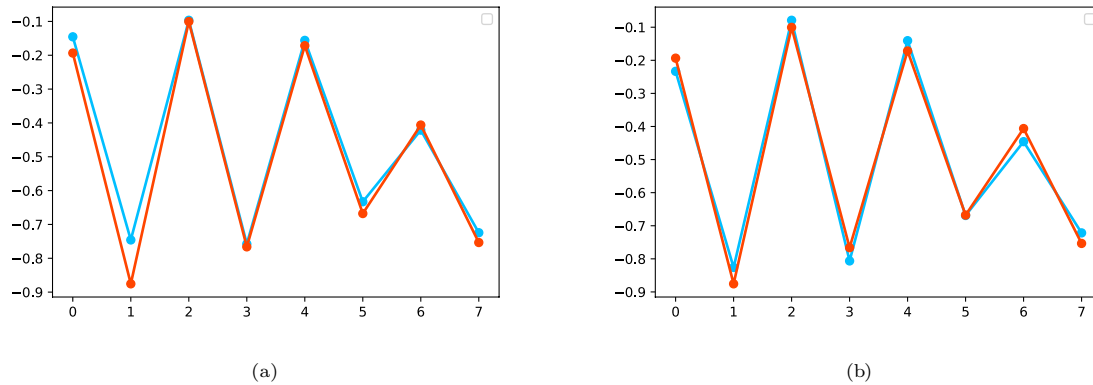
### 7.4 Explainability

In this section, we specifically study the explainability characteristics of DIBCA++. We initially select a pair of devices from the smart logistics use case to inspect how the correlations identified by the domain correlation module of DIBCA++ can be used to explain the algorithm performance on the selected pair. The performance of ATM on the D2-D5 pair is the worst one (see Fig. 4 or Table 7), which has provoked our curiosity to check out how the clusters of the D2 and D5 models are correlated (last fold of experiment using all the data is used for illustration), see Fig. 6. Interestingly, there is only one mismatch, namely the connection between cluster 8 from the D2 model and cluster 1 from the D5 model. The other connections are correct, but they correlate only with half of the clusters; the other half will exist as private clusters in the integrated model (IM) and their respective adapted models (ASM or ATM). In addition, a common cluster that unites cluster 8 from D2 and clusters 1 and 8 from D5 will be created in IM, ASM, and ATM. This cluster will contain data points from both classes, namely with and without GPS signals, which certainly will affect the ATM performance. However, the identified connection between cluster 8 from the D2 model and cluster 1 from the D5 model can also provide us with new information about the scenarios presented in the D2 and D5 models. This might be a hint that the scenarios modeled by these two clusters, in fact, present a transition between the two main classes, i.e., from GPS coverage to no GPS coverage and vice versa.

Next, two pairs of users from the HAR use case are chosen to further analyze and understand the results with respect to the explainability aspect of the DIBCA++. One best-performing and one least-performing pairs are picked up from

**Fig. 7** Visualisation of the correlations identified by the domain correlation module of the DIBCA++ on the selected two pairs of users. The correlated clusters are merged by the domain integration module of DIBCA++. **a** Correlations identified for user pair U12-U15, the best-performing pair; **b** Correlations identified for user pair U10-U18, one of the worst performing pairs





**Fig. 8** Visualization of the internal step, which can be used to explain the obtained correlation between cluster 0 of U12 and cluster 1 (instead of cluster 0) of U15. **a** Mean vectors of clusters 0 of U12 and U15; **b** Mean vectors of clusters 0 and 1 of U12 and U15 respectively

Table 5 (Experiment B2) to understand how the source and target models of these pairs are integrated (Experiment B1). User pairs U12-U15 and U10-U18 are chosen to be further analyzed. Note that U10-U18 is chosen over U13-U10 as for the first pair, the value of AMI is the second worst performing, whereas, for the second pair, the ARI value is the third worst performing. The performance of the selected pairs U12-U15 and U10-U18 is similar (they are one of the best and worst performing pairs) in experimental scenario B1 when the users' individual models are integrated using DIBCA++ (see Tables 3 and 4). Hence, they can be used to visualize the internal steps of the algorithm. For both the pairs, the fourth fold, i.e., the fold that uses all the data of the data set, is used to illustrate the results.

Figure 7 presents the correlations between these pairs identified by the domain correlation module of DIBCA++. It can be observed that the correlations extracted for U12-U15 are among similar clusters or activities. While in the case of the U10-U18, there are more mismatches when compared to U12-U15. This explains the good and bad performance of U12-U15 and U10-U18, respectively in experiment B1 (Tables 3 and 4). It can also be observed that in the case of U10-U18, the ASM produces higher improvement in comparison with SM than in the case of U12-U15 (see columns SD-SM and SD-ASM). In general, the identified correlations indirectly state that U10 and U18 have differences in the way they perform the same activities, and hence, this negatively affects the model performance when their data are combined together (Table 5) as personalization is lost. The case is the opposite when the data of U12 and U15 are integrated.

In order to further understand the reason behind the mismatches in the correlations, we visualize one more layer of how DIBCA++ works, and a random pair of mismatched correlations is selected, i.e., cluster 0 of U12 (left column), U15 (right column), and cluster 1 of U15 (see Fig. 7a).

Figure 8 presents the mean vector pairs of both clusters. Note that mean vectors are used to find the closest cluster (see Algorithm 2). Figure 8a depicts the mean vectors of cluster 0 from both U12 and U15, whereas Fig. 8b plots the mean vectors of cluster 0 of U12 and cluster 1 of U15. From both figures, it can be clearly observed that cluster 0 of U12 is closer to cluster 1 of U15, explaining the reasoning behind the decision of the algorithm. Through this, we would like to showcase the explainability aspect of the algorithm, which can be useful in building trust and acceptance to use the designed algorithm in a wider range of applications.

## 8 Conclusion and future work

In this paper, we have proposed a domain adaptation algorithm entitled DIBCA++, which is an optimized and robust version of its predecessor DIBCA. DIBCA++ has been designed to be resource-efficient and robust to outliers. In addition, we have developed an improved learning algorithm used for building the initial clustering models of the domains. This work also presents a detailed description of the main modules of DIBCA++, which has been extensively studied and evaluated with respect to well-designed experimental scenarios in two different real-world use cases, namely Human Activity Recognition (HAR) and smart logistics. The experimental results have shown that the DIBCA++ is capable of transferring knowledge between domains that, indeed, leads to improved performance results. In one of the experiments on the smart logistics use case, it has been observed that the adapted source and target models performed better than the original source and target models 70% and 80% of the time, respectively. The results on HAR also showcase improved performance when data from different domains are handled individually compared to when

the data is combined, thus highlighting the importance of building personalized models.

In addition to the above-mentioned experiments, DIBCA++'s performance has also been compared to that of DIBCA. The experimental results showcase the better performance of DIBCA++ compared to the original DIBCA. It is convincingly the best performer on the HAR use case and outperforms DIBCA in 82.5% of the conducted experimental scenarios for the smart logistics use case.

Another advantage of the proposed algorithm is its explainability feature. It facilitates the analysis and interpretation of the obtained results, which supports a better understanding and acceptance of the algorithm and eventually can lead to the extraction of new knowledge about the studied data phenomenon.

Our future plans aim to evaluate the usability of the DIBCA++ in new applied domain adaptation scenarios. A sequential version of the algorithm will also be tested and further evaluated. The sequential version of DIBCA++ would allow continuous updating of the current integrated model and generating of new adapted domain models when data from a new domain is available.

**Acknowledgements** This research was funded partly by the Knowledge Foundation, Sweden, through the Human-Centered Intelligent Realities (HINTS) Profile Project (contract 20220068). In addition, the data used in one of the experiments were provided due to the involvement in the Sony RAP 2020 Project "Distributed and Adaptive Edge-based AI Models for Sensor Networks".

**Author Contributions** *VMD*: conceptualization, methodology, formal analysis and investigation, implementation, validation, visualization, writing—original draft preparation, writing—review and editing; *VB*: conceptualization, methodology, formal analysis and investigation, writing—original draft preparation, writing—review and editing, supervision, funding acquisition; *SA*: methodology, visualization, writing—review and editing, supervision.

**Funding** Open access funding provided by Blekinge Institute of Technology.

**Data availability** *Data set 1*: Access to the data is restricted due to the company's privacy policy. *Data set 2*: Human Activity Recognition (DaLiAc) data set is publicly available at <https://www.mad.tf.fau.de/research/activitynet/daliac-daily-life-activities/>.

## Declarations

**Conflict of interest** The authors do not have any conflicting interests.

**Ethical and informed consent for data used** Not applicable (N/A).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in

the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abghari S, Boeva V, Casalicchio E, Exner P (2022) An inductive system monitoring approach for gnss activation. In: Maglogiannis I, Iliadis L, Macintyre J, Cortez P (eds) Artificial intelligence applications and innovations. Springer International Publishing, Cham, pp 437–449
- AlShehhi M, Damiani E, Wang D (2021) Toward domain adaptation for small data sets. *Internet Things* 16:100458. <https://doi.org/10.1016/j.iot.2021.100458>
- Alvarez-Melis D, Fusi N (2020) February. Geometric dataset distances via optimal transport. In *NeurIPS 2020*. ACM
- Boeva V, De Baets B (2004) A new approach to admissible alternatives in interval decision making. In: 2004 2nd international IEEE conference on 'intelligent systems'. Proceedings (IEEE Cat. No.04EX791), vol 1, pp 110–115
- Csurka G (2017) Domain adaptation in computer vision applications. Springer International Publishing, Cham
- Davidsson P (1996). Coin classification using a novel technique for learning characteristic decision trees by controlling the degree of generalization. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*
- De Lange M, Tuytelaars T (2021). Continual prototype evolution: Learning online from non-stationary data streams. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8230–8239
- Devagiri V M, Boeva V, Abghari S (2022) Domain adaptation through cluster integration and correlation. In: *2022 IEEE international conference on data mining workshops (ICDMW)*, pp 1–8
- Gunasekara N, Gomes H, Bifet A, Pfahringer B (2022) Adaptive online domain incremental continual learning. In: Pimenidis E, Angelov P, Jayne C, Papaleonidas A, Aydin M (eds) *Artificial Neural Networks and Machine Learning - ICANN 2022*, Cham. Springer International Publishing, pp 491–502
- Hubert L, Arabie P (1985) Comparing partitions. *J Classification* 2(1):193–218
- Hundscheil S, Weber M, Mandl P (2023) An empirical study of adversarial domain adaptation on time series data. In: Rutkowski L, Scherer R, Korytkowski M, Pedrycz W, Tadeusiewicz R, Zurada JM (eds) *Artificial Intelligence and Soft Computing*, Cham. Springer International Publishing, pp 39–50
- Iverson DL (2004) 01. Inductive system health monitoring. In *IC-AI*, pp. 605–611
- Leutheuser H, Schuldhaus D, Eskofier BM (2013) Hierarchical, multi-sensor based classification of daily life activities: comparison with state-of-the-art algorithms using a benchmark dataset. *PloS one* 8(10):e75196. <https://doi.org/10.1371/journal.pone.0075196>
- Li J, Li G, Shi Y, Yu Y (2021a). Cross-domain adaptive clustering for semi-supervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2505–2514
- Li G, Kang G, Zhu Y, Wei Y, Yang Y (2021) Domain consensus clustering for universal domain adaptation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp 9757–9766
- Lu W, Chen Y, Wang J, Qin X (2021) Cross-domain activity recognition via substructural optimal transport. *Neurocomputing* 454:65–75. <https://doi.org/10.1016/j.neucom.2021.04.124>

- Madadi Y, Seydi V, Nasrollahi K, Hossieni R, Moeslund T (2020) Deep visual unsupervised domain adaptation for classification tasks: a survey. *IET Image Process* 14(14):3283–3299. <https://doi.org/10.1049/iet-ipr.2020.0087>
- Orbes-Arteainst M, Cardoso J, Sørensen L, Igel C, Ourselin S, Modat M, Nielsen M, Pai A (2019). Knowledge distillation for semi-supervised domain adaptation. In L. Zhou, D. Sarikaya, S. M. Kia, S. Speidel, A. Malpani, D. Hashimoto, M. Habes, T. Löfstedt, K. Ritter, and H. Wang (Eds.), *OR 2.0 Context-Aware Operating Theaters and Machine Learning in Clinical Neuroimaging*, Cham, pp. 68–76. Springer International Publishing
- Pan SJ, Yang Q (2010) A survey on transfer learning. *IEEE Trans Knowl Data Eng* 22(10):1345–1359. <https://doi.org/10.1109/TKDE.2009.191>
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. *J Mach Learn Res* 12:2825–2830
- Rand WM (1971) Objective criteria for the evaluation of clustering methods. *J Am Stat Assoc* 66(336):846–850
- Reiss A, Stricker D (2012). Introducing a new benchmarked dataset for activity monitoring. In: *The 16th IEEE International Symposium on Wearable Computers (ISWC)*
- Ribeiro MT, Singh S, Guestrin C (2016). why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, New York, NY, USA, pp. 1135–1144. Association for Computing Machinery
- Saw JG, Yang MCK, Mo TC (1984) Chebyshev inequality with estimated mean and variance. *Am Stat* 38:130–132
- Srihari S (2020). Explainable artificial intelligence: An overview. *Journal of the Washington Academy of Sciences*
- Tang H, Wang Y, Jia K (2022) Unsupervised domain adaptation via distilled discriminative clustering. *Pattern Recognit* 127:108638. <https://doi.org/10.1016/j.patcog.2022.108638>
- Tang S, Zou Y, Song Z, Lyu J, Chen L, Ye M, Zhong S, Zhang J (2022) Semantic consistency learning on manifold for source data-free unsupervised domain adaptation. *Neural Netw* 152:467–478
- Vinh NX, Epps J, Bailey J (2009). Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09*, pp. 1073–1080
- Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J Mach Learn Res* 11(95):2837–2854
- Wang H, Tian J, Li S, Zhao H, Wu F, Li X (2022) Structure-conditioned adversarial learning for unsupervised domain adaptation. *Neurocomputing* 497:216–226. <https://doi.org/10.1016/j.neucom.2022.04.094>
- Wang J, Chen Y, Hu L, Peng X, Yu PS (2018). Stratified transfer learning for cross-domain activity recognition. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10
- Xu J, Song J, Sang Y, Yin L (2022) Cdaml: a cluster-based domain adaptive meta-learning model for cross domain recommendation. *World Wide Web* 1573–1413. <https://doi.org/10.1007/s11280-022-01068-5>
- Zhu M (2021). Source free domain adaptation by deep embedding clustering. In: *2021 18th Int. Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 309–312
- Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q (2021) A comprehensive survey on transfer learning. *Proc IEEE* 109(1):43–76. <https://doi.org/10.1109/JPROC.2020.3004555>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.