Attraction-Repulsion clustering with applications to fairness. *

Eustasio del Barrio¹, Hristo Inouzhe², and Jean-Michel Loubes³

¹²Departamento de Estadística e Investigación Operativa and IMUVA, Universidad de Valladolid, Spain.
³Université de Toulouse, Institut de Mathématiques de Toulouse, France.

Abstract

In the framework of fair learning, we consider clustering methods that avoid or limit the influence of a set of protected attributes, \( S \), (race, sex, etc) over the resulting clusters, with the goal of producing a fair clustering. For this, we introduce perturbations to the Euclidean distance that take into account \( S \) in a way that resembles attraction-repulsion in charged particles in Physics and results in dissimilarities with an easy interpretation. Cluster analysis based on these dissimilarities penalizes homogeneity of the clusters in the attributes \( S \), and leads to an improvement in fairness. We illustrate the use of our procedures with both synthetic and real data.

1 Introduction

Cluster analysis or clustering is the task of dividing a set of objects in such a way that elements in the same group or cluster are more similar, according to some dissimilarity measure, than elements in different groups. To achieve this task there are two main types of algorithms: partitioning algorithms, which try to split the data into \( k \) groups that usually minimize some optimality criteria, or agglomerative algorithms, which start with single observations and merge them into clusters according to some dissimilarity measure. Such methods have been investigated in a large amount of literature, hence we refer to [15] and references therein for an overview.

Supervised and unsupervised classification procedures are increasingly more influential in people’s life since they are used in credit scoring, article recommendation, risk assessment, spam filtering or sentencing recommendations in courts of law, among others. Hence controlling the outcome of such procedures, in particular ensuring that some variables which should not be taken into account due to moral or legal issues are not playing a role in the classification of the observations, has become an important field of research known as fair learning. We refer to [4, 6, 18] or [12] for an overview of such legal issues and mathematical solutions to address them. The main concern is to detect whether decision rules, learnt from variables \( X \), are biased with respect to a subcategory of the population driven by some variables called protected or sensitive variables. Such variables induce a bias in the observations, and are correlated to other observations. Hence avoiding this effect cannot be achieved by the naive solution of ignoring

---

*Research partially supported by FEDER, Spanish Ministerio de Economía y Competitividad, grant MTM2017-86061-C2-1-P and Junta de Castilla y León, grants VA005P17 and VA002G18. Research partially supported by ANITI program and DEEL IRT.
such protected attributes. Indeed, if the the data at hand reflects a real world bias, machine learning algorithms can pick on this behaviour and emulate it.

Recently, concerns about fairness have received an increasing attention, resulting into two main strategies to address it in the field of classification. One course of action is to transform the data in order to avoid correlation between the set of sensitive attributes and the rest of the data \[8,11\]. Another way is to modify the objective functions of the algorithms in a way that eliminates or reduces the unfairness \[16,20\].

In this work we consider the problem of fair clustering. Suppose we observe data that includes information about attributes that we know or suspect that are biased with respect to the protected class. If the biased variables are dominant enough, a standard clustering on the unprotected data will result in some biased clusters, therefore, if we classify new instances based on this partition, we will incur in biased decisions. Ideally, a fair clustering would be the situation in which, in the partition of the data, the proportions of the protected attributes are the same in each cluster (hence, the same as the proportions in the whole dataset). This notion of fairness is close to the disparate impact doctrine \([11]\) adapted to the clustering setting. The idea is to guarantee that a decision taken with respect to a particular cluster in the partition will not affect disproportionately some group of the population (codified by the protected class), since every group is represented in every cluster according to its proportion in the whole data. With our approach to fair clustering we try to avoid or mitigate these situations by reducing homogeneity with respect to the sensitive classes of the groups, but without imposing too hard fairness constraints such as group proportions that may result in an excessive reduction of the information conveyed by the data.

A possible approach for fair clustering was presented in \([5]\) based on the idea of constrained k-center and k-median clustering. The authors proposed a model where data are partitioned into two groups, codified by red and blue, where disparate impact is avoided by maintaining balance between the proportion of points in the two categories. Their goal is to achieve a partition of the data where the respective objective function is minimized while balance in each cluster is maintained over a given threshold. Their approach initially designed for data with two different protected classes has led to the following extensions. The k-center problem was tackled in \([8,20]\), the k-median in \([1]\) while k-means is further studied in \([2\,3\,22]\). These extensions impose constraints related to some minimum and/or maximum value for the proportions of the protected classes in each cluster. Yet, the constraints may result in an excessive modification of the clustering problem with a subsequent loss of geometrical information contained in the data.

In the following, we propose an alternative to the constrained clustering approach for obtaining fair partitions by incorporating some perturbations in the original dissimilarities of the data in order to favour heterogeneity with respect to the protected variable. For this, we present a new methodology inspired by particle physics and based on attraction-repulsion dissimilarities. These new dissimilarities aim at increasing the separation between points with the same values of the protected class and/or decreasing the separation between points with different values of the protected class. Hence they favour the formation of clusters that are more heterogeneous in the protected variable since they reduce the distance between points with different class values, and increase the distance between points with the same class values, leading to a gain in fairness.

Note that we do not need to find fairlets \([5]\), partitions of the data into small fair clusters, or core-sets \([22]\), small subsets of the data for which solving the fair clustering problem gives a reasonable approximation to the fair clustering problem in the whole dataset. Our method is more flexible than previous ones and enables to control better the trade-off between fairness and geometrical information of the data. The proposed dissimilarities depend on parameters that the practitioner can control and therefore he or she can impose bigger tendency to fairness. Guidance in the task of choosing the parameters is presented in Section \([5]\) and discussed through
some synthetic examples in Section 6.1. Moreover, attraction-repulsion dissimilarities can be combined with some common clustering techniques via an embedding, in particular, with multidimensional scaling (Section 2). Agglomerative hierarchical clustering is well suited for the use of dissimilarities, hence, in Section 3, we show how to adapt our proposals in a computationally efficient way when using this type of clustering. The proposed attraction-repulsion dissimilarities can also be adapted to the kernel-trick extension, applied to the unprotected variables \( X \), leading to non linear separation in \( X \) with a penalization for heterogeneity w.r.t \( S \) as shown in Section 4 and 6.1.2.

The paper falls into the following parts. Section 2 presents the attraction-repulsion dissimilarities. Clustering methods are developed in Section 3 while Section 4 is devoted to their extension to the kernel case. We study the importance of the choice of parameters in Section 5. Finally, applications for our technique are given in Section 6. We provide a thorough discussion on a synthetic dataset in 6.1.1, while in 6.3 we apply our methods to the Ricci dataset which describes the case of Ricci v. DeStefano of the Supreme Court of the United States, [24]. In Section 6.2, we provide comparison between our methods and the ones proposed in [5], that suggests that proportion constraints are too strong requirements if we want to retain structural information of the data.

2 Charged clustering via multidimensional scaling

Clustering relies on the choice of dissimilarities that control the part of information conveyed by the data that will be used to gather points into the same cluster, expressing how such points share some common characteristics. To obtain a fair clustering we aim at obtaining clusters which are not governed by the protected variables but are rather mixed with respect to these variables. For this, we introduce interpretable dissimilarities in the space \((X, S) \in \mathbb{R}^{d+p}\) aiming at separating points with the same value of the protected classes. Using an analogy with electromagnetism, the labels \( S \) play the role of an electric charge and similar charges tend to have a repulsive effect while dissimilar charges tend to attract themselves.

Our guidances for choosing these dissimilarities are that we would like the dissimilarities to

i) induce fairness into subsequent clustering techniques (eliminate or, at least, decrease dependence of the clusters on the protected attribute),

ii) keep the essential geometry of the data (with respect to non-protected attributes) and

iii) be easy to use and interpret.

Hence we propose the following dissimilarities.

**Definition 1 (Attraction-Repulsion Dissimilarities).**

\[
\delta_1 ((X_1, S_1), (X_2, S_2)) = 1'U1 + S_1'VS_2 + \|X_1 - X_2\|^2
\]

with \( U, V \) symmetric matrices in \( \mathbb{R}^{p \times p} \);

\[
\delta_2 ((X_1, S_1), (X_2, S_2)) = \left(1 + ue^{-v\|S_1 - S_2\|^2}\right)\|X_1 - X_2\|^2
\]

with \( u, v \geq 0 \);

\[
\delta_3 ((X_1, S_1), (X_2, S_2)) = \|X_1 - X_2\|^2 - u\|S_1 - S_2\|^2
\]

with \( u \geq 0 \).

Let \( 0 \leq u \leq 1 \) and \( v, w \geq 0 \),

\[
\delta_4 ((X_1, S_1), (X_2, S_2)) = \left(1 + \text{sign}(S_1'VS_2)u\left(1 - e^{-v\|S_1'VS_2\|^2}\right)e^{-w\|X_1 - X_2\|^2}\right)\|X_1 - X_2\|.
\]
Remark 1. $\delta_1((X,S),(X,S)) \neq 0$ and therefore it is not strictly a dissimilarity. Yet, for all practical purposes discussed in this work this does not affect the proposed procedures.

To the best of our knowledge this is the first time that such dissimilarities have been proposed and used in the context of clustering (in [13] repulsion was introduced modifying the objective function, only taking into account distances between points, to maintain centers of clusters separated). Dissimilarities (1) to (4) are natural in the context of fair clustering because they penalize the Euclidean distance taking into account the (protected) class of the points involved. Hence, some gains in fairness could be obtained.

The dissimilarities we consider are easily interpretable, providing the practitioner with the ability to understand and control the degree of perturbation introduced. Dissimilarity (1) is an additive perturbation of the squared Euclidean distance where the intensity of the penalization ability to understand and control the degree of perturbation introduced. Dissimilarity (1) is an additive perturbation of the squared Euclidean distance where the intensity of the penalization is controlled by matrices $U$ and $V$, with $V$ controlling the interactions between elements of the same and of different classes $S$. Dissimilarity (3) presents another additive perturbation but the penalization is proportional to the difference between the classes $S_1$ and $S_2$, and the intensity is controlled by the parameter $u$.

Dissimilarity (2) is a multiplicative perturbation of the squared Euclidean distance. With $u$ we control the amount of maximum perturbation achievable, while with $v$ we modulate how fast we diverge from this maximum perturbation when $S_1$ is different to $S_2$.

Dissimilarity (4) is also a multiplicative perturbation of the Euclidean distance. However, it has a very different behaviour with respect to (1)-(3). It is local, i.e., it affects less points that are further apart. Through $w$ we control locality. With bigger $w$ the perturbation is meaningful only for points that are closer together. With matrix $V$ we control interactions between classes as in (1), while with $u$ we control the amount of maximum perturbation as in (2). Again, $v$ is a parameter controlling how fast we diverge from the maximum perturbation.

We present in the following a simple example for the case of a single binary protected attribute, coded as $-1$ or $1$. This is an archetypical situation in which there is a population with an (often disadvantaged) minority, that we code as $S = -1$, and the new clustering has to be independent (or not too dependent) on $S$.

Example 1. Let us take $S_1, S_2 \in \{-1,1\}$. For dissimilarity (1) we fix $U = V = c \geq 0$, therefore

$$\delta_1((X_1,S_1),(X_2,S_2)) = c(1 + S_1S_2) + \|X_1 - X_2\|^2.$$ 

If $S_1 \neq S_2$, we have the usual squared distance $\|X_1 - X_2\|^2$, while when $S_1 = S_2$ we have $2c + \|X_1 - X_2\|^2$, effectively we have introduced a repulsion between elements with the same class. For dissimilarity (2) let us fix $u = 0.1$ and $v = 100$,

$$\delta_2((X_1,S_1),(X_2,S_2)) = \left(1 + 0.1e^{-100\|S_1 - S_2\|^2}\right)\|X_1 - X_2\|^2.$$ 

When $S_1 \neq S_2$ we have approximately $\|X_1 - X_2\|^2$, while when $S_1 = S_2$ we have $1.1\|X_1 - X_2\|^2$, again introducing a repulsion between elements of the same class. For dissimilarity (3), when $S_1 = S_2$ we have $\|X_1 - X_2\|^2$ and when $S_1 \neq S_2$ we get $\|X_1 - X_2\|^2 - 2u$, therefore we have introduced an attraction between different members of the sensitive class. When using dissimilarity (4), fixing $V = c > 0$, $u = 0.1$, $v = 100$, $w = 1$, we get

$$\delta_4((X_1,S_1),(X_2,S_2)) = \left(1 + 0.1\text{sign}(cS_1S_2)\left(1 - e^{-100(cS_1'\phi S_2'^2)}\right)e^{-\|X_1 - X_2\|}\right)\|X_1 - X_2\|.$$ 

If $S_1 = S_2$ we get approximately $(1 + 0.1e^{-\|X_1 - X_2\|})\|X_1 - X_2\|$, therefore we have a repulsion. If $S_1 \neq S_2$ we have approximately $(1 - 0.1e^{-\|X_1 - X_2\|})\|X_1 - X_2\|$, which can be seen as an attraction.
Our proposals are flexible thanks to the freedom in choosing the class labels. If we codify $S$ with $\{-1, 1\}$, as in the previous example, we can only produce attraction between different classes and repulsion between the same classes (or exactly the opposite if $V < 0$) in (1) and (4). On the other hand, if we codify $S$ as $\{(1, 0), (0, 1)\}$, we have a wider range of possible interactions induced by $V$. For example taking $V = ((1, -1)'|(-1, 0)'$) we produce attraction between different classes, no interaction between elements labelled as $(0, 1)$ and repulsion between elements labelled as $(1, 0)$. If we had three classes we could use $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ as labels and induce a personalized interaction between the different elements via a $3 \times 3$ matrix $V$. For example $V = ((0, -1, -1)'|(-1, 0, -1)'|(-1, -1, 0)'$ provides attraction between different classes and no interaction between elements of the same class. Extensions to more than three classes are straightforward. More details on parameter and class codification selection will be given in Section 4.

These dissimilarities can then be used directly in some agglomerative hierarchical clustering methods, as described in Section 3. Alternatively, we could use these dissimilarities to produce some embedding of the data into a suitable Euclidean space and use some optimization clustering technique (in the sense described in Chapter 5 in [10]) on the embedded data. Actually, the dissimilarities $\delta_i$ can be combined with common optimization clustering techniques, such as $k$-means, via some embedding of the data. We note that our dissimilarities aim at increasing the separation of points with equal values in the protected attributes while respecting otherwise the geometry of the data. Using multidimensional scaling (MDS) we can embed the original points in the space $\mathbb{R}^{d'}$ with $d' \leq d$ and use any clustering technique on the embedded data. Quoting [7], multidimensional scaling ‘is the search for a low dimensional space, usually Euclidean, in which points in the space represent the objects, one point representing one object, and such that the distances between the points in the space, match, as well as possible, the original dissimilarities’. Thus, applied to dissimilarities $\delta_i$, MDS will lead to a representation of the original data that conveys the original geometry of the data in the unprotected attributes and, at the same time, favours clusters with diverse values in the protected attributes.

Here is an outline of how to use the dissimilarities $\delta_i$ coupled with MDS for a sample $(X_1, S_1), \ldots, (X_n, S_n)$.

**Attraction-Repulsion MDS**

For any $l \in \{1, 2, 3, 4\}$

- **Compute the dissimilarity matrix** $[\Delta_{i,j}] = [\delta_i((X_i, S_i), (X_j, S_j))]$ with a particular choice of the free parameters.
- **If** $\min \Delta_{i,j} \leq 0$, **transform the original dissimilarity to have positive entries:**
  \[
  \Delta_{i,j} = \Delta_{i,j} + \min \Delta + \epsilon, \text{ where } \epsilon \text{ is small}.
  \]
- **For** $\delta_1, \delta_2, \delta_3$: $\Delta_{i,j} = \sqrt{\Delta_{i,j}}$.
- **Use MDS to transform** $(X_1, S_1), \ldots, (X_n, S_n)$ **into** $X'_1, \ldots, X'_n \in \mathbb{R}^{d'}$, **where** $D_{i,j} = ||X'_i - X'_j||$ **is similar to** $\Delta_{i,j}$.
- **Apply a clustering procedure on the transformed data** $X'_1, \ldots, X'_n$.

This procedure will be studied in Section 6 for some synthetic and real datasets.

### 3 Charged hierarchical clustering

Agglomerative hierarchical clustering methods (bottom-top clustering) encompass many of the most widely used methods in unsupervised learning. Rather than a fixed number of clusters, these methods produce a hierarchy of clusterings starting from the bottom level, at which each
sample point constitutes a group, to the top of the hierarchy, where all the sample points are grouped into a single unit. We refer to [19] for an overview. The main idea is simple. At each level, the two groups with the lowest dissimilarity are merged to form a single group. The starting point is typically a matrix of dissimilarities between pairs of data points. Hence, the core of a particular agglomerative hierarchical clustering lies at the way in which dissimilarities between groups are measured. Classical choices include single linkage, complete linkage, average linkage or McQuitt’s method. Additionally, some other methods are readily available for using dissimilarities, as, for example, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) introduced in [9].

When a full data matrix (rather than a dissimilarity matrix) is available it is possible to use a kind of agglomerative hierarchical clustering in which every cluster has an associated prototype (a center or centroid) and dissimilarity between clusters is measured through dissimilarity between mean vectors within each cluster. More precisely, if clusters \(i\) and \(j\) have \(n_i\) and \(n_j\) elements and mean vectors \(g_i\) and \(g_j\) then Ward’s dissimilarity between clusters \(i\) and \(j\) is

\[
\delta_W(i, j) = \frac{n_in_j}{n_i+n_j} \|g_i - g_j\|^2,
\]

where \(\| \cdot \|\) denotes the usual Euclidean norm. Other methods based on prototypes are the centroid method or Gower’s median method (see [19]). However, these last two methods may present some undesirable features (the related dendrograms may present reversals that make the interpretation harder, see, e.g., [10] and Ward’s method is the most frequently used within this prototype-based class of agglomerative hierarchical clustering methods.

Hence, in our approach to fair clustering we will focus on Ward’s method. Given two clusters \(i\) and \(j\) consisting of points \(\{(X_i, S_i)\}_{i=1}^{n_i}\) and \(\{(Y_j, T_j)\}_{j=1}^{n_j}\), respectively, we define the charged dissimilarity between them as

\[
\delta_{W,l}(i, j) = \frac{n_in_j}{n_i+n_j} \delta_l(\frac{1}{n_i} \sum_{i=1}^{n_i} X_i, \frac{1}{n_i} \sum_{i=1}^{n_i} S_i), (\frac{1}{n_j} \sum_{j=1}^{n_j} Y_j, \frac{1}{n_j} \sum_{j=1}^{n_j} T_j))
\]

where \(\delta_l, l = 1, \ldots, 4\) is any of the point dissimilarities defined by (1) to (4).

The practical implementation of agglomerative hierarchical methods depends on the availability of efficient methods for the computation of dissimilarities between merged clusters. This is the case of the family of Lance-Williams methods (see [17], [19] or [10]) for which a recursive formula allows to update the dissimilarities when clusters \(i\) and \(j\) are merged into cluster \(i \cup j\) in terms of the dissimilarities of the initial clusters. This allows to implement the related methods using computer time of order \(O(n^2 \log n)\). We show next that a recursive formula similar to the Lance-Williams form holds for the dissimilarities \(\delta_{W,l}\) and, consequently, the related agglomerative hierarchical method can be efficiently implemented. The fact that we are dealing differently with genuine and protected attributes results in the need for some additional notation (and storage). Given clusters \(i\) and \(j\) consisting of points \(\{(X_i, S_i)\}_{i=1}^{n_i}\) and \(\{(Y_j, T_j)\}_{j=1}^{n_j}\), respectively, we denote

\[
d_x(i, j) = \|\frac{1}{n_i} \sum_{i=1}^{n_i} X_i - \frac{1}{n_j} \sum_{j=1}^{n_j} Y_j\|.
\]

Note that \(d_x(i, j)\) is simply the Euclidean distance between the means of the \(X\)-attributes in clusters \(i\) and \(j\). Similarly, we set

\[
d_s(i, j) = \|\frac{1}{n_i} \sum_{i=1}^{n_i} S_i - \frac{1}{n_j} \sum_{j=1}^{n_j} T_j\|.
\]

**Proposition 1.** For \(\delta_{W,l}\) as in (5), \(d_x(i, j)\) as in (6) and \(d_s(i, j)\) as in (7) and assuming that clusters \(i, j\) and \(k\) have sizes \(n_i, n_j\) and \(n_k\), respectively, we have the following recursive formulas:
i) \( \delta_{W,1}(i \cup j, k) = \frac{n_i+n_k}{n_i+n_j+n_k} \delta_{W,1}(i, k) + \frac{n_j+n_k}{n_i+n_j+n_k} \delta_{W,1}(j, k) - \frac{n_k}{n_i+n_j+n_k} d_{W,x}^2(i, j); \)

ii) \( \delta_{W,2}(i \cup j, k) = \left( 1 + \frac{n_k}{n_i+n_j+n_k} \cdot \left( \frac{n_j}{n_i+n_j} \cdot d_{W,x}^2(i, k) + \frac{n_j+n_k}{n_i+n_j+n_k} \cdot d_{W,x}^2(j, k) - \frac{n_i+n_j+n_k}{n_i+n_j+n_k} \cdot d_{W,x}^2(i, j) \right) \right) \times \left( \frac{n_i+n_j+n_k}{n_i+n_j+n_k} \cdot d_{W,x}^2(i, k) + \frac{n_j+n_k}{n_i+n_j+n_k} \cdot d_{W,x}^2(j, k) - \frac{n_k}{n_i+n_j+n_k} \cdot d_{W,x}^2(i, j) \right); \)

iii) \( \delta_{W,3}(i \cup j, k) = \frac{n_i+n_k}{n_i+n_j+n_k} \delta_{W,3}(i, k) + \frac{n_j+n_k}{n_i+n_j+n_k} \delta_{W,3}(j, k) - \frac{n_k}{n_i+n_j+n_k} \delta_{W,3}(i, j), \)

where \( d_{W,x}^2(i, j) = \frac{n_i+n_j}{n_i+n_j+n_k} d_{W,x}^2(i, j). \)

Proof. For i) we just denote by \( R_s, S_t \) and \( T_r \) the protected attributes in clusters \( i, j \) and \( k \), respectively and note that

\[
\delta_{W,1}(i \cup j, k) = \frac{(n_i+n_j)n_k}{n_i+n_j+n_k} \left( 1'U1 + \frac{1}{n_j} \left( \sum_{s=1}^{n_j} R_s + \sum_{t=1}^{n_j} S_t \right) \right) V \frac{1}{n_k} \sum_{r=1}^{n_k} T_r + d_{W,x}^2(i, j, k) \\
= \frac{(n_i+n_j)n_k}{n_i+n_j+n_k} \left( 1'U1 + \frac{1}{n_j} \left( \sum_{s=1}^{n_j} R_s \right) \right) V \frac{1}{n_k} \sum_{r=1}^{n_k} T_r + d_{W,x}^2(i, j, k) \\
+ \frac{(n_i+n_k)n_j}{n_i+n_j+n_k} \left( 1'U1 + \frac{1}{n_i} \left( \sum_{t=1}^{n_i} S_t \right) \right) \left( \frac{1}{n_k} \sum_{r=1}^{n_k} T_r + d_{W,x}^2(i, j, k) \right) \\
+ \frac{n_j+n_k}{n_i+n_j+n_k} \left( 1'U1 + \frac{1}{n_j} \left( \sum_{t=1}^{n_j} S_t \right) \right) V \frac{1}{n_k} \sum_{r=1}^{n_k} T_r + d_{W,x}^2(i, j, k) \\
- \frac{n_k}{n_i+n_j+n_k} d_{W,x}^2(i, j, k) \\
= \frac{n_i+n_k}{n_i+n_j+n_k} \delta_{W,1}(i, k) + \frac{n_j+n_k}{n_i+n_j+n_k} \delta_{W,1}(j, k) - \frac{n_k}{n_i+n_j+n_k} d_{W,x}^2(i, j). \]

Observe that we have used the well-known recursion for Ward’s dissimilarities, namely,

\( d_{W,x}^2(i, j, k) = \frac{n_i+n_j}{n_i+n_j+n_k} d_{W,x}^2(i, j, k) + \frac{n_j+n_k}{n_i+n_j+n_k} d_{W,x}^2(j, k) - \frac{n_k}{n_i+n_j+n_k} d_{W,x}^2(i, j, k) \)

(8)

(see, e.g., [10]). The update formulas ii) and iii) are obtained similarly. We omit details.

From Proposition 1 we see that a practical implementation of agglomerative hierarchical clustering based on \( \delta_{W,l}, l = 1, 2 \) would require the computation of \( d_{W,x}^2(i, j), \) which can be done using the Lance-Williams formula. In the case of \( \delta_{W,2} \) we also need \( d_{s}^2(i, j), \) which again can be obtained through a Lance-Williams recursion. This implies that agglomerative hierarchical clustering based on \( \delta_{W,l}, l = 1, 2 \) can be implemented using computer time of order \( O(n^2 \log n) \) (at most twice the required time for the implementation of an ‘unfair’ Lance-Williams method).

We end this section with an outline of the implementation details for our proposal for fair agglomerative hierarchical clustering based on dissimilarities \( \delta_{W,l}. \)

Iterative Attraction-Repulsion Clustering For \( l \in \{1, 2, 3\} \)

- Compute the dissimilarity matrix \( [\Delta_{i,j}] = [\delta_i((X_i, S_i), (X_j, S_j))] \) with a particular choice of the free parameters.
- If \( \min \Delta_{i,j} \leq 0 \), transform the original dissimilarity to have positive entries: \( \Delta_{i,j} = \Delta_{i,j} + \min \Delta + \epsilon, \) where \( \epsilon \) is arbitrarily small.
- Use the Lance-Williams type recursion to determine the clusters \( i \) and \( j \) to be merged; iterate until there is a single cluster.
4 Fair clustering with kernels

Clustering techniques based on the minimization of a criterion function typically result in clusters with a particular geometrical shape. For instance, given a collection of points \( x_1, \ldots, x_n \in \mathbb{R}^d \), the classical \( k \)-means algorithm looks for a grouping of the data into \( K \leq n \) clusters \( C = \{ c_1, \ldots, c_K \} \) with corresponding means \( \{ \mu_1, \ldots, \mu_K \} \) such that the objective function

\[
\sum_{k=1}^{K} \sum_{x \in c_k} \| x - \mu_i \|^2
\]

is minimized. The clusters are then defined by assigning each point to the closest center (one of the minimizing \( c_i \)'s). This results in convex clusters with linear boundaries. It is often the case that this kind of shape constraint does not adapt well to the geometry of the data. A non-linear transformation of the data could map some clustered structure to make it more adapted to convex linear boundaries (or some other pattern). In some cases this transformation can be implicitly handled via kernel methods. We explore in this section how the charged clustering similarities that we have introduced can be adapted to the kernel clustering setup, focusing on the particular choice of kernel \( k \)-means.

Kernel \( k \)-means is a non-linear extension of \( k \) means that allows to find arbitrary shaped clusters introducing a suitable kernel similarity function \( \kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \), where the role of the squared Euclidean distance between two points \( x, y \) in the classical \( k \)-means is taken by

\[
d_k^2(x, y) = \kappa(x, x) + \kappa(y, y) - 2\kappa(x, y). \tag{9}
\]

Details of this algorithm can be found in [23].

In a first approach, we could try to introduce a kernel function for vectors \((X_1, S_1), (X_2, S_2) \in \mathbb{R}^{d+p}\) such that \( d_k^2 \) takes into account the squared Euclidean distance between \( X_1 \) and \( X_2 \) but also tries to separate points of the same class and/or tries to bring closer points of different classes, i.e., makes use of \( S_1, S_2 \). Some simple calculations show that this is not an easy task, if possible at all in general. If we try, for instance, a joint kernel of type \( \kappa((X_1, S_1), (X_2, S_2)) = \tau(S_1, S_2) + k(X_1, X_2), S_1, S_2 \in \{-1, 1\} \) with \( \tau, k \) Mercer (positive semi-definite) kernels (this covers the case \( k(X_1, X_2) = X_1 \cdot X_2 \) the usual scalar product in \( \mathbb{R}^d \) ), our goal can be written as

\[
d_k^2((X_1, S_1), (X_2, S_1)) > d_k^2((X_1, S_1), (X_2, S_2)), \tag{10}
\]

for any \( X_1, X_2 \), with \( S_1 \neq S_2 \). However, the positivity contraints on \( \tau \), imply that

\[
2\tau(S_1, S_2) > \tau(S_1, S_1) + \tau(S_2, S_2), \quad \tau^2(S_1, S_2) \leq \tau(S_1, S_1)\tau(S_2, S_2).
\]

But the solutions of this inequalities violate that \( \tau \) is positive-semi-definite. Therefore, there is no kernel on the sensitive variables that we can add to the usual scalar product. Another possibility is to consider a multiplicative kernel, \( \kappa((X_1, S_1), (X_2, S_2)) = \tau(S_1, S_2)k(X_1, X_2), S_1, S_2 \in \{-1, 1\} \) with \( \tau, k \) Mercer kernels. From \([10]\) we get

\[
2(\tau(S_1, S_1) - \tau(S_1, S_2)) k(X_1, X_2) < (\tau(S_1, S_1) - \tau(S_2, S_2)) k(X_2, X_2)
\]

which depends on \( k(X_1, X_2) \) and makes it challenging to eliminate the dependence of the particular combinations \( X_1, X_2 \).

The previous observations show that it is difficult to think of a simple and interpretable kernel \( \kappa \) that can be a simple combination of a kernel in the space of unprotected attributes and a kernel in the space of sensitive attributes. This seems to be caused by our desire to separate
vectors that are similar in the sensitive space, which goes against our aim to use norms induced by scalar products. In other words a naive extension of the kernel trick to our approach to fair clustering seems to be inappropriate.

Nonetheless, the difficulty comes from a naive desire to carry out the (implicit) transformation of the attributes and the penalization of homogeneity in the protected attributes in the clusters in a single step. We still may obtain gains in fairness, while improving the separation of the clusters in the unprotected attributes if we embed the X data into a more suitable space by virtue of some sensible kernel \( \kappa \) and consider the corresponding kernel version of \( \delta_l \), with \( \delta_l \) as in (1) to (4). Instead of using the Euclidean norm \( \|X_1 - X_2\| \) we should use \( d_\kappa(X_1, X_2) \). In the case of \( \delta_1 \), for instance, this would amount to consider the dissimilarity

\[
\delta_{\kappa,1} ((X_1, S_1), (X_2, S_2)) = 1'U1 + S_1'VS_2 + d_\kappa(X_1, X_2)^2, \tag{11}
\]

with similar changes for the other dissimilarities. Then we can use an embedding (MDS the simplest choice) as in Section 2 and apply a clustering procedure to the embedded data. This would keep the improvement in cluster separation induced (hopefully) by the kernel trick and apply, at the same time, a fairness correction. An example of this adaptation of the kernel trick to our setting is given in Section 6.1.2.

5 Parameter selection

The attraction-repulsion dissimilarities (1) - (4) introduced in Section 2 depend on two main sets of parameters. First, on several free parameters used to balance the influence of the variables \( X \) and the protected variable \( S \). Second, on the way the protected variables are labelled for the different classes, with the possibility to include different interactions between the groups. Here we propose some guidelines on the choice of these parameters.

The dissimilarities we consider can be divided into two groups: (2) and (3) do not depend on the codification of the class variable, while (1) and (4) do depend on such a choice. In our method, the level of fairness is imposed through the choice of the parameters in the dissimilarities. Contrary to other methods such as [5] where only completely fair solutions can be found (see for instance in Figure 1), choosing the parameters enables to balance fairness and the original structure of the data which may convey information that should not be erased by fairness constraints.

Consider first dissimilarities (2) and (3). They rely on two parameters \( u \) and \( v \). In the multiplicative dissimilarity (2), \( v \) is a parameter that measures how sudden is the change in the distance when switching from elements with different protected class to elements with same protected class. For \( v \) large enough, \( e^{-v\|S_1 - S_2\|^2} \) is small when \( S_1 \neq S_2 \), which implies that the fair dissimilarity only modifies the distance between points inside the same protected class, increasing heterogeneity of the clusters.

Once \( v \) has been fixed, the main parameter \( u \) controls the intensity of the perturbation in a similar way for both dissimilarities (2) and (3). To illustrate the effect of this parameter we focus on (2) and perform a fair clustering, with MDS or hierarchically, for different values of the intensity parameter \( u \) and measure the fairness of the clusters obtained. Such example is depicted in the left middle row of Figure 1. We can see that, as expected, increasing the values for \( u \) puts more weight on the part of the dissimilarity that enforces heterogeneity of the clusters. \( u = 0 \) leads to the usual clustering. Indeed, varying \( u \) from 0 to 4.5 in steps of 0.5 increases the fairness achieved for both clusters, with a saturation effect from 4.5 to 5 where we do not appreciate an improvement in fairness. Hence, maximum fairness is achieved for \( u = 4.5 \) and gives the lowest perturbation that achieves the highest level of fairness. If one aims at
preserving more of the structure of the original information at the expense of a lower level of fairness, some smaller value of $u$ can be selected. For example, in the right middle row of Figure 1 we provide the result of choosing $u = 1$. Hence $u$ balances both effects of closeness to the usual dissimilarities and the amount of heterogeneity reached in the clustering.

Next, dissimilarities 1 and 4, as described in Section 2, depend on the values chosen for the protected variable $S$, and a matrix $V$, which plays the role of the matrix of interactions for different classes. When dealing with a two-class discrimination problem where the protected class has only two values, labelling the classes as $\{-1, 1\}$ or $\{(1, 0), (0, 1)\}$ can lead to the same results for appropriate choices of $V$. However, for more than two protected classes we will use only the following vectorial codification: for $q$ different values of the protected class, we will codify the values as the $q$ unitary vectors $\{a_1, \ldots, a_q\}$ where $a_{i,j} = 1$ if $i = j$ and $a_{i,j} = 0$ if $i \neq j$ for $1 \leq i, j \leq q$.

To build the interaction matrix we proceed as follows. First, consider a matrix $\tilde{V}_{i,j}$ with $1 \leq i, j \leq q$. We fix $\tilde{V}_{i,j} = 0$ if we want no interaction between classes $i$ and $j$, in particular, if $i = j$ this means that there is no interaction between elements with the same class. We take $\tilde{V}_{i,j} = 1$ if we want repulsion (relative increase in the distance) between classes $i$ and $j$. We fix $\tilde{V}_{i,j} = -1$ if we want attraction (relative decrease in distance) between classes $i$ and $j$. Hence, if the practitioner believes that there is some discrimination, in the sense of disproportional impact, against a class represented by $a_\ast$, it is recommendable to set values of $\tilde{V}_{i,j} = -1$ for $j \neq i\ast$. As an example, in Section 6.1.2, we have chosen the interaction matrix $V = ((1,-1)'|(-1,0)')$, to model repulsion between elements of the same class $(1,0)$, attraction between elements of the classes $(1,0)$ and $(0,1)$, and no interaction between the elements of the same class $(0,1)$.

Then intensity of the interaction is modelled using a constant $v_0 > 0$, and we set $V = v_0 \tilde{V}$. In the previous example $v_0 = 1$. The parameter $v$ for dissimilarity 4 has the same meaning as the corresponding parameter for (2) and can be selected in the same way.

Finally, matrix $U$ for dissimilarity 1 represents and extra additive shift. In many cases it can be set to $U = 0$ (the zero matrix).

We provide an example to explain how to select the intensity $v_0$ for dissimilarity 1 in the top left image of Figure 1. Notice that using $V > 0$ and $S \in \{-1,1\}$ is the same as using $V = v_0 \tilde{V}$ with $\tilde{V} = ((1,-1)'|(-1,1)')$ and $S \in \{(1,0),(0,1)\}$. We plot the variation of the fairness in each cluster when we vary the intensity of the interaction between 0 and 4.4 with steps of size 0.44. There is a steady improvement in fairness in both clusters until the intensity reaches $v_0 = 3.52$, but from this level, as previously, there is no more improvement in fairness. Therefore, if a practitioner wants to achieve the highest level of fairness, $v_0 = 3.52$ should be the proper intensity, since it corresponds to the smallest perturbation to the geometry that achieves the best observed fairness. However, a smaller correction in fairness may be of interest, we have a representation of that top right in Figure 1 for $v_0 = 1.32$.

For dissimilarity 4, after choosing the interaction matrix $\tilde{V}$, we can try to find a maximum in fairness, fixing a grid formed by different combinations for the vector of parameters $(v_0, u, w)$. In the second and third column of Table 2 we see the fairness of the respective clusters when we look at the grid $(1, u, 0.05)$ with $u = 0, 0.098, \ldots, 0.98$. What we notice is an improvement in fairness for all values of $u$, therefore a practitioner would be advised to select $u = 0.98$ where we obtain the best fairness values.
6 Applications

6.1 Synthetic data

6.1.1 General example

We generate 50 points from four distributions,

\[ \mu_1 \sim N((-1, 0.5), \text{diag}(0.25, 0.25)), \mu_2 \sim N((-1, -0.5), \text{diag}(0.25, 0.25)); \]

\[ \mu_3 \sim N((1, 0.5), \text{diag}(0.25, 0.25)), \mu_4 \sim N((1, -0.5), \text{diag}(0.25, 0.25)), \]

and label the samples from \( \mu_1 \) and \( \mu_2 \) as \( S = 1 \) (squares) and the samples from \( \mu_3 \) and \( \mu_4 \) as \( S = -1 \) (circles). A representation of the data in the original space is given in the third column of Figure 1. We can think of the data as heavily biased in the \( x \) direction, therefore any sensible clustering procedure is going to have clusters that are highly homogeneous in the class \( S \) when the original coordinates are used. This is exemplified in Table 1, as we look for different number of clusters: with k-means we are detecting almost pure groups (1st row); the same happens with a complete linkage hierarchical clustering with the Euclidean distance (5th row) and with Ward’s method with the Euclidean distance (9th row).

Therefore, it may be useful to apply our procedures to the data to gain diversity in \( S \). In the first column of Figure 1 we study the relation between the gain in fairness from the increase in intensity of the corrections we apply and the disruption of the geometry of the original classes after MDS. In the first row we use dissimilarity (1), where we fix \( U = 0 \), and we vary \( V = 0, 0.44, 0.88, \ldots, 4.4 \). In the second row we work with dissimilarity (2), where we fix \( v = 20 \) and set \( u = 0, 0.5, 1, \ldots, 5 \). In the last row we work with dissimilarity (4) fixing \( V = 1 \), \( v = 20 \), \( w = 1 \) and we vary \( u = 0, 0.099, 0.198, \ldots, 0.99 \). We do not show results for dissimilarity (3), since in this example it gives results very similar to dissimilarity (1). Squares and circles represent the proportion of class \( S = 1 \) in the two clusters found by k-means after the MDS transformation. Crossed squares and circles represent the average silhouette index of class \( S = 1 \) and class \( S = -1 \). We recall that the silhouette index of an observation \( X_i \) is given by

\[ s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \]

where \( a(i) \) is the average distance to \( X_i \) of the observation in the same group as \( X_i \), and \( b(i) \) is the average distance to \( X_i \) of the observations in the closest group different than the one of \( X_i \) (see [21]). The average silhouette index of a group is the average of the silhouette indexes of the members of the group.

What we see top-left and middle-left in Figure 1 is that greater intensity relates to greater heterogeneity but also relates to lower silhouette index. This can be interpreted as the fact that greater intensity in dissimilarities (1) and (2) has a greater impact in the geometry of the original problem. In essence, the greater the intensity, the more indistinguishable \( S = 1 \) and \( S = -1 \) become after MDS, therefore, any partition with k-means will result in very diverse clusters in \( S \). By construction this is not what happens with dissimilarity (4). The strong locality penalty \( (w = 1) \) allows to conserve the geometry, shown by the little reduction in silhouette index (row 3 column 1), but results in smaller corrections in the proportions.

From the previous discussion, a practitioner interested in imposing fairness, with no interest in the original geometry should use high intensity corrections. However, a practitioner interested in gaining some fairness while still being able to keep most of the original geometry should go for low intensity or local corrections.
Figure 1: Top row: dissimilarity \( \text{(1)} \), Middle row: dissimilarity \( \text{(2)} \), Bottom row: dissimilarity \( \text{(4)} \). Left column: proportions of \( S = 1 \) in the clusters (squares and circles) and average silhouette indexes for \( S = 1 \) and \( S = -1 \) in the transformed space (crossed squares and circles), for varying input parameters. Middle column: two clusters in the transformed space for a particular choice of parameters. Right column: same two clusters in the original space.

Table 1: Proportion of class \( S = 1 \) in every group in different clustering procedures.

|                  | Proportion of squares in the group |
|------------------|-----------------------------------|
|                  | \( K = 2 \) | \( K = 3 \) | \( K = 4 \) |
| k-means Unperturbed | 0.03  0.99 | 0.02  0.88 | 0.98  1.00 | 0.06  0.02 |
| \( \delta_1 \)    | 0.15  0.90 | 0.11  0.49 | 0.95  0.75 | 0.94  0.16 | 0.14 |
| \( \delta_2 \)    | 0.09  0.96 | 0.43  0.04 | 0.97  0.08 | 0.97  0.85 | 0.06 |
| \( \delta_4 \)    | 0.13  0.96 | 0.96  0.05 | 0.42  0.96 | 0.11  0.13 | 0.95 |
| MDS               | \( \delta_1 \) | \( \delta_2 \) | \( \delta_4 \) |
| Complete Linkage  | Unperturbed | 0.99  0.07 | 0.99  0.08 | 0.99  1.00 | 0.08  0.05 |
| \( \delta_1 \)    | 1.00  0.32 | 1.00  0.40 | 0.25  1.00 | 0.56  0.25 | 0.00 |
| \( \delta_2 \)    | 0.72  0.22 | 0.72  0.30 | 0.00  1.00 | 0.30  0.24 | 0.00 |
| \( \delta_4 \)    | 0.78  0.11 | 1.00  0.43 | 0.11  1.00 | 0.43  0.47 | 0.00 |
| Ward's Method     | Unperturbed | \( \delta_1 \) | \( \delta_2 \) |
| \( \delta_1 \)    | 0.99  0.07 | 0.08  0.05 | 0.97  0.08 | 0.05  1.00 |
| \( \delta_2 \)    | 0.11  0.78 | 0.54  0.98 | 0.11  0.54 | 0.18  0.00 | 0.98 |

\[ \frac{9}{12} \]
In the rest of Figure 1 we show the actual clusters in the MDS embedding obtained with k-means (column 2) and the same clusters in the original space (column 3), for some moderate intensities. For dissimilarity (1) we take $V = 1.32$, for (2) $u = 1$ and for (4) we use $u = 0.99$. A short remark is that a rotation of a MDS is a MDS, and that is the cause of the rotations that we see in column 2. Indeed, after MDS the geometry of the groups is not heavily modified, but at the same time some corrections to the proportions are achieved when clustering. This corrections appear very natural once we return to the original space.

For the same values as the previous paragraph we present Table 1, where we look for 2, 3 and 4 clusters with MDS and k-means, but also using the approximation-free complete linkage hierarchical clustering and our Ward’s-like method. We see that there is an improvement in heterogeneity, regardless of the method used. However, the more clusters we want the smaller the improvement. This is associated with our small perturbation of the geometry, if we want more fairness for bigger number of clusters we have to use more intense perturbations.

6.1.2 Kernel trick example

Let us explore the adaptation of the kernel trick explained in Section 4. We consider the data in the top-left image of Figure 2. These data have a particular geometrical shape and are split into two groups. There is an inside ring of squares, a middle ring of circles, and then an outer ring of squares. There are 981 observations and the proportions of the classes are approximately 3 to 1 (circles are 0.246 of the total data).

It is natural to apply to the original data some clustering procedure as k-means or a robust extension as tclust (deals with groups with different proportions and shapes and with outliers). Looking for two clusters, we would be far from capturing the geometry of the groups, but the clusters would have proportions of the classes that are similar to the total proportion. Indeed, this is what we see in Figure 2 middle-left when we apply k-means to the original data. On the other hand, the kernel trick is convenient in this situation. We propose to use the kernel function $\kappa(x, y) = x_1^2y_1^2 + x_2^2y_2^2$, which corresponds to a transformation $\phi((x_1, x_2)) = (x_1^2, x_2^2)$. The data in the transformed space is depicted in the top-right of Figure 2. Our adaptation to the kernel trick uses $d_\kappa$ as defined in (9) and dissimilarity (4) in the form

$$\delta_{\kappa, 4}((X_1, S_1), (X_2, S_2)) = (1 + \text{sign}(S_1'V S_2)u(1 - e^{-v(S_1'V S_2)^2})e^{-w_d(X_1, X_2)})d_\kappa(X_1, X_2),$$

for $X_1, X_2$ in the original two dimensional space, as described in Section 4.

Taking into account the discussion at the end of Section 2 and Section 5, we use dissimilarity (12) with $S_1, S_2 \in \{(1, 0), (0, 1)\}$. In our setting circles are labelled as (1, 0) and squares as (0, 1). Now if we fix $u = 0$, use (12) to calculate the dissimilarity matrix $\Delta$ and use MDS, essentially, we will be in the space depicted top-right on Figure 2. Looking for two clusters with tclust, allowing groups with different sizes, we get the result depicted middle-right in Figure 2. We have captured the geometry of the clusters but the proportions of the class $S$ are not the best, as seen in row 1 columns 2 and 3 of Table 2 (ideally they should be close to 0.754). In order to gain diversity in what is referred as cluster 2 in Table 2 (in red in the plots), we vary the intensity $u$ of our local dissimilarity, with the other parameters set as indicated in Table 2. We see that as we increase the intensity of the interactions we gain in heterogeneity in cluster 2, and both proportions come closer to the total proportion 0.754 (columns 2-3). Again, this is achieved without destroying the geometry of the original classes after the MDS, as seen in the small variation of the average silhouette index in columns 4-5.

We plot the best performance, given by $u = 0.98$, after MDS in bottom-left and in the original space in bottom-right of Figure 2. It is clear that we have been able to capture the geometry of the groups and to produce relatively fair clusters.
Figure 2: Top row: data in the original space (left) and after transformation $\phi$ (right). Middle row: k-means in the original space (left) and tclust applied in the transformed space and plotted in the original one (right). Bottom row: tclust after fairness corrections applied in the transformed space (left) and represented in the original space (right).
Table 2: Effect of varying the intensity $u$ of the local dissimilarity \( ((1, -1)^t|(-1, 0)^t) \), for fixed $V = (v, w) = (20, 0.05)$. First two columns contain the proportion of points with $S = (0, 1)$ in the clusters found with tclust in the transformed space. Last two columns show the silhouette of the original classes in the MDS.

| $u$   | Prop. in cluster 1 | Prop. in cluster 2 | Silhouette for $(0, 1)$ | Silhouette for $(1,0)$ |
|-------|-------------------|-------------------|-------------------------|------------------------|
| 0.000 | 0.629             | 0.950             | -0.247                  | 0.502                  |
| 0.098 | 0.629             | 0.950             | -0.245                  | 0.502                  |
| 0.196 | 0.629             | 0.950             | -0.243                  | 0.499                  |
| 0.294 | 0.630             | 0.948             | -0.241                  | 0.495                  |
| 0.392 | 0.631             | 0.945             | -0.239                  | 0.491                  |
| 0.490 | 0.631             | 0.943             | -0.237                  | 0.486                  |
| 0.588 | 0.631             | 0.943             | -0.235                  | 0.481                  |
| 0.686 | 0.631             | 0.943             | -0.234                  | 0.476                  |
| 0.784 | 0.630             | 0.946             | -0.232                  | 0.471                  |
| 0.882 | 0.672             | 0.863             | -0.231                  | 0.467                  |
| 0.980 | 0.681             | 0.849             | -0.229                  | 0.465                  |

6.2 Comparison with fair clustering through fairlets

In this section we present a comparison of the results of our methods with results obtained by implementing, in Python and R, the fair clustering procedure introduced in [5] based on fairlets decomposition. Since our examples are concerned with two values for the protected class it is justified to use [5] for comparison since it is well suited for this situation. Our implementation of the case when the size of both protected classes is the same, which reduces to an assignment problem, is implemented using the function `max_bipartite_matching` of the package `igraph` in R.

In the case of different sizes, we have to solve a min cost flow problem as stated in [5], which can be done in Python with the function `min_cost_flow` of the package `networkx` (also it can be solved with a `min_cost_flow` solver in `ortools`).

We start with the data used for the example studied in Figure 1. We will address $k$-median clustering for which [5] has a fair implementation. Since the data has two groups of the same size we have to solve an assignment problem. The result is shown in the upper row of Figure 3. The method achieves a $k$-median objective function of 466.283, while the proportion of squares in each group is 0.5. In the bottom-left we present the result of $k$-median clustering when we use dissimilarity [1] with $V = 4.4$, which gives us a median objective function of 429.469 and proportion of squares of 0.47 and 0.54. Bottom right we see the result of a $k$-means clustering when we use dissimilarity [1] with $V = 3.2$ (which we saw is the best possible value when looking for maximum fairness), it achieves a $k$-median objective function of 444.105 and proportion of squares of 0.47 and 0.53.

We see that attraction-repulsion corrections achieve better clusters, since they have smaller $k$-median objective function, with a small decrease in fairness. This is true even when we use $k$-means to produce the clusters, which does not look to minimize the $k$-median objective function. Indeed, total fairness, with regard to disparate impact, is only achieved by the method introduced in [5], but this is on the cost of loosing almost any cluster structure.

To better see that we refer to Figure 4. We have implemented the min cost flow problem for the data used in Section 6.1.2, where we impose balance of 1/4 to the clusters, and plotted the result on the left. The $k$-median objective function is 8523.74 and group proportion of squares are 0.73 and 0.79. This, again, is very close to the total fairness with regard to disparate impact.
Figure 3: Top row: clusters obtained by fair $k$-median (5). Bottom row: clusters obtained by attraction-repulsion clustering with dissimilarity (1) and MDS, using $k$-median (left) and $k$-means (right).

Figure 4: Left: clustering obtained using fair $k$-median (5). Right: clustering where the cluster membership is assigned by a fair coin toss.
which would require proportions of 0.754. Our result using dissimilarity (4), which is shown in the bottom right of Figure 2, achieves a $k$-median objective function of 6751.01 and proportions of 0.68 and 0.85. Again, our method achieves better clusters for the $k$-median objective function with some decrease in fairness. However, our method captures properly the geometry of the data, while conserving a clear cluster structure, something that does not happen with the approach proposed in [5]. Indeed, we have plotted on the right of Figure 4 a situation when we have selected the labels of the points tossing a fair coin, i.e., disregarding any distance information. Both plots in Figure 4 look very similar, suggesting that imposing proportion constraints to the cluster groups is a very strong requirement. Hence, our approach seems to be justified, since it both preserves some cluster structure and produces an improvement in fairness.

6.3 A real data example

We analyse the Ricci dataset consisting of scores in an oral and a written exam of 118 firefighters, where the sensitive attribute is the race of the individual. This dataset was a part of the case Ricci v. DeStefano presented before the Supreme Court of the United States, where the combined score of the exam, which was a threshold for a promotion, was the question at issue.

We codify white individuals as $S = 1$ and black and hispanic individuals as $S = -1$. A representation of the data can be seen in Figure 5. From it, it is clear that the region delimited by oral scores higher than 72 is heavily dominated by white individuals. We may see this as a bias in this variable. In any case, a standard clustering method on this data may result in groups that are more homogeneous with respect to race than the data as a whole. We stress that the proportion of black and hispanic people in the data set is approximately 0.42.

In Table 3 we test our methods which look to decrease homogeneity, and possibly increase fairness. In this sense, we want to have groups that have a proportion of black and hispanic members close to 0.42. We will use both transforming the data via MDS and applying $k$-means, the alternative straightforward use of complete linkage hierarchical clustering and also
Table 3: Proportion of class $S = -1$ in every group in different clustering procedures. For $\delta_1$ we fix $U = 0, V = 500$; for $\delta_2$ we fix $u = 3.125, v = 10$; for $\delta_3$, $u = 333$ and for $\delta_4$, $V = 1, u = 0.99, v = 10, w = 1$.

| Method          | Unperturbed | $\delta_1$ | $\delta_2$ | $\delta_3$ | $\delta_4$ |
|-----------------|-------------|------------|------------|------------|------------|
| **k-means**     |             | $\delta_1$ | $\delta_2$ | $\delta_3$ | $\delta_4$ |
| Unperturbed     | 0.59        | 0.34       | 0.54       | 0.35       | 0.25       |
| MDS             |             | 0.52       | 0.39       | 0.51       | 0.59       |
| Complete Linkage|             | 0.27       | 0.30       | 0.23       | 0.30       |
| Ward’s Method   |             | 0.55       | 0.51       | 0.52       | 0.52       |

Ward’s method. The appropriate parameters for the dissimilarities are chosen to give a good performance (after a grid search as suggested in Section 5) and are specified in Table 3.

As we expected, applying clustering procedures to the original (unperturbed) data gives clusters that are significantly more homogeneous than the whole data set with respect to the protected class. On the other hand, using the suggested dissimilarities, we see improvements in the heterogeneity of the groups regarding the sensitive variable. Both additive dissimilarities (1) and (3) perform well in conjunction with MDS and $k$-means, giving proportions closer to 0.42 for 2,3 and 4 clusters (rows 2 and 3 in Table 3). In the hierarchical setting, it seems that dissimilarity (2) gives a nice performance. The local dissimilarity (4) seems to have little effect, which can be interpreted as a need of non-local effects to be able to affect the group formation.

In Figure 6 we compare the solutions of $k$-means for 2 and 4 clusters in the original data and in the data transformed via MDS and dissimilarity (1), which we saw gives a good performance. We also represent the solution given by Ward’s method and dissimilarity (2). For comparison we plot the solution obtained using fair $k$-median as in [5]. We see that the clusterings are similar, but our perturbation of the Euclidean distance that takes into account the composition of the groups is able to induce change in the borders between groups, making the clusters more heterogeneous in the protected class. As in the previous section the procedure imposing proportion constraints gives the best solution with regard to fairness, but it looses almost all cluster structure. On the other hand, our perturbations induce gains in fairness while producing clusters that reflect the structure of the data.

7 Conclusion

We have considered the problem of clustering data taking into account an extra variable $S$ which models a belonging to a group. A so-called fair clustering means that the cluster labels should not enable any inference on the values of $S$, hence promoting clusters which are not homogeneous with respect to the values of $S$. We have considered an algorithm based on attraction-repulsion
Figure 6: First row: $k$-means for 2 and 4 clusters in the unperturbed (original) data. Second row: $k$-means for 2 and 4 clusters in the MDS setting with $\delta_1$. Third row: Ward’s method for 2 and 4 clusters with $\delta_2$. Forth row: fair $k$-median as in [5] for 2 and 4 clusters. Circles represent not white individuals; squares represent white individuals.
dissimilarities, which offers some advantages over previous methods dealing with proportion constrained clustering approaches.

First, the method is more flexible in the sense that it is possible to control the amount of fairness we impose on the clusters and to consider the trade-off between fairness and the geometrical structure of the original data.

Furthermore, the method is extremely simple to implement, something shared with the methods proposed in [5], but not being the case for the other algorithms implementing fair clustering procedures that have been mentioned previously. For instance, we may use any clustering algorithm, since we are not restricted to clustering algorithms that minimize an objective function of some distance. In particular, this allows the use of hierarchical clustering. As a consequence, this method of fair clustering can be extended to the case where the data is not Euclidean, opening the field of applications for this method. For instance, fairness corrections can be included when clustering probability distributions or some other objects that present geometrical shapes. This can be achieved by replacing the squared euclidean distance $\|X_1 - X_2\|$ in dissimilarities (1) - (4) with an appropriate distance for the objects involved.

Finally, from a practitioners point of view it should be straightforward to use and easy to program with standard tools as R or Python as provided in \url{https://github.com/JMLToulouse/FairLearning}.

References

[1] Backurs, A. Indyk, P. Onak, K. Schieber, B. Vakilian, A. and Wagner, T. (2019) Scalable Fair Clustering.

[2] Bera, S. Chakrabarty, D. Negahbani, M. and College, D. (2019) Fair Algorithms for Clustering

[3] Bercea, I. Gross, M. Khuller, S. Kumar, A. Rösner, C. Schmidt, D. and Schmidt, M. (2018) On the cost of essentially fair clusterings.

[4] Besse, Philippe and Castets-Renard, Celine and Garivier, Aurelien and Loubes, Jean-Michel, (2018). Can everyday AI be ethical. Fairness of Machine Learning Algorithms, \texttt{arXiv:1810.01729}.

[5] Chierichetti, F. Kumar, R. Lattanzi, S. and Vassilvitskii, S. (2017) Fair clustering through fairlets. \textit{NIPS}.

[6] Chouldechova, A. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. \textit{Big data}, 5:153–163, 2017.

[7] Cox, T. F. and Cox, M. A. A. (2000) \textit{Multidimensional Scaling}.

[8] del Barrio, E. Gamboa, F. Gordaliza, P. and Loubes, J-M.(2018) Obtaining fairness using optimal transport theory. \texttt{arXiv:1806.03195}.

[9] Ester, M. Kriegel, H-P. Sander, J. and Xu, X. (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. \textit{AAAI}.

[10] Everitt B.S., Landau, S., Leese, M. and Stahl, D. (2011). \textit{Cluster Analysis, 5th Edition}. Wiley.
11. Feldman, M. Friedler, S. A. Moeller, J. Scheidegger, C. and Venkatasubramanian, S. (2015) Certifying and removing disparate impact. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

12. Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., and Roth, D. (2018) A comparative study of fairness-enhancing interventions in machine learning. arXiv:1802.04422F.

13. Ferraro, M. B. and Giordani, P. (2013) On possibilistic clustering with repulsion constraints for imprecise data. Information Sciences.

14. Fritz, H. García-Escudero, LA. and Mayo-Íscar, A. (2012) tclust: An R Package for a Trimming Approach to Cluster Analysis. Journal of statistical software.

15. Hennig, C., Meila, M., Murtagh, F. and Rocci, R. (2015). Handbook of Cluster Analysis. CRC Press.

16. Kehrenberg, T. Chen, Z. and Quadrianto, N. (2018) Interpretable Fairness via Target Labels in Gaussian Process Models. arXiv:1810.05598v2.

17. Lance, G. and Williams, W. (1967) A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. The Computer Journal.

18. Lum, K. and Johndrow, J. (2016) A statistical framework for fair predictive algorithms. arXiv:1610.08077L.

19. Murtagh, F. and Contreras, P. (2011) Algorithms for hierarchical clustering: an overview. WIREs Data Mining and Knowledge Discovery.

20. Rösner, C and Schmidt, M. (2018) Privacy Preserving Clustering with Constraints. 45th International Colloquium on Automata, Languages and Programming.

21. Rousseeuw, P. (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics.

22. Schmidt, M. Schwiegelshohn, C. and Sohler, C. (2018) Fair core-sets and streaming algorithms for fair k-means clustering.

23. Schölkopf, B. Smola, A. and Müller, K-R. (1998) Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation.

24. Supreme Court of the United States. (2009) Ricci v. DeStefano. 557 U.S. 557, 174.

25. Woodworth, B. Gunasekar, S. Ohannessian, M. I. and Srebro, N. (2017) Learning Non-Discriminatory Predictors. arXiv:1702.06081v3.

26. Zafar, M. B. Valera, I. Rodriguez, M. G. and Gummadi, K. P. (2017) Fairness Constraints: Mechanisms for Fair Classification. AISTATS.