Learning similarity metrics based on pairwise boosting

David Bertrán

Thesis submitted to the Louvain School of Engineering in partial fulfillment of the requirements for the degree of Electrical Engineering

Supervisor:
Prof. Christophe De Vleeschouwer

Jury Members:
Prof. Michel Verleysen
Dr. Pascaline Parisot

Louvain la Neuve, 2014
# Contents

1. Introduction ............................................................................................................. 1
   1.1. Motivation ........................................................................................................... 1
   1.2. Problem statement ............................................................................................. 2
   1.3. Solution overview ............................................................................................. 3
   1.4. Outline of the Thesis ......................................................................................... 5

2. Learning similarity metrics with Adaboost ........................................................... 7
   2.1. Introduction ........................................................................................................ 7
   2.2. Weak classifiers ................................................................................................. 10
   2.3. Threshold evaluation procedure ......................................................................... 13
   2.4. Weights updating .............................................................................................. 17
   2.5. Weighted correlation ......................................................................................... 18
   2.6. Factor of goodness ........................................................................................... 21
   2.7. Metric from the improved performance of the weak classifiers ...................... 24

3. Louvain Method ...................................................................................................... 27
   3.1. Introduction ....................................................................................................... 27
   3.2. Concepts ............................................................................................................ 27
   3.3. Method .............................................................................................................. 30
      3.3.1. 1st step ....................................................................................................... 30
      3.3.2. 2nd step ..................................................................................................... 32

4. Results validation ................................................................................................... 33
   4.1. Introduction ....................................................................................................... 33
   4.2. Input data .......................................................................................................... 33
   4.3. Methodology ..................................................................................................... 36
      4.3.1. Training and test set .................................................................................. 36
      4.3.2. Types of pairing ......................................................................................... 36
         4.3.2.1. Single pairing ..................................................................................... 37
         4.3.2.2. Complete Pairing .............................................................................. 38
         4.3.2.3. Multi-single Pairing .......................................................................... 38
      4.3.3. Testing the similarity and checking the clustering ...................................... 40
   4.4. Tests .................................................................................................................. 41
      4.4.1 Two independent Gaussian cultures ............................................................. 41
      4.4.2 Two intermixed Gaussian cultures ............................................................... 43
      4.4.3 Four Gaussian cultures with single pairing ................................................. 47
      4.4.4 Four Gaussian cultures with complete pairing ............................................ 50
      4.4.5 Four intermixed Gaussian cultures .............................................................. 51
      4.4.6 Counterexample of Complete Pairing ......................................................... 56
      4.4.7 An unsolvable case ....................................................................................... 57
      4.4.8 Improving with Multi-single Pairing ............................................................ 59
      4.4.9 Multi-single confirmed ............................................................................... 61
      4.4.10 Non-gaussian distribution ......................................................................... 64

5. Conclusions ............................................................................................................. 71

6. References .............................................................................................................. 73
1. Introduction

1.1. Motivation

The nucleolus is a specialized subcellular functional domain found in the nucleus of eukaryotic cells, involved in ribosome synthesis. From a functional point of view, after being produced in the nucleolus, ribosomes are exported to the cytoplasm where they translate messenger RNA (mRNA) into proteins. Because the nucleolus directly impacts the synthesis of proteins and thus the functions of the cell, it is an important organelle of the cell.

The nucleolus is not bounded by a membrane, which makes it extremely dynamic. Interestingly, earlier studies have shown that there is a strong correlation between the 3-D structure of the nucleolus and the potential diseases affecting the cell. Healthy/normal cells are characterized by spherical nucleoli, whilst diseased cells present conformation artifacts that can be visually observed through (epi-) fluorescence microscopy, as shown in Figure 1.1:

![Figure 1.1: Examples of normal (left) and pathological (right) cell nucleus. Brighter intensities correspond to the nucleolus.](image)

In this context, our project investigates how gene inhibition affects the nucleolar structure. Specifically, it aims at identifying which proteins—among the 700 proteins present in the nucleolus—are required to maintain a visually normal conformation of the nucleolus. It also aims at characterizing the different kinds of conformation artifacts resulting from the inhibition of genes encoding nucleolar components, using the so-called silencers. Based on the assumption that proteins that induce similar deformations of the nucleolus are involved in similar (dis)functions of the cell, the study will help the biologists to better understand the role of each protein, and to identify the proteins that are required for a particular cellular process, i.e. to elucidate the workings of normal and diseased cells.

From a technical point of view, our project assumes that a database of cell images is available, resulting from high-throughput screening experiments which systematically shut down each gene in the cell. The cells are grouped into cultures of cells, each culture corresponding to a specific silencer. In other words, all the cells from a given culture are subject to the same gene inhibition process, and should present similar artifacts on the nucleolus.

Given such a database, the objective of our project is to develop methods for automated analysis and quantitative characterization of the visual appearance of the nucleolus conformation observed in the database. In short, assuming that tools for automatic image analysis extract a feature vector that characterizes the visual appearance of the nucleolus, our project aims at clustering cell feature vectors in a way that is consistent with the observations made within and across cell cultures. Specifically, our first objective is to define a similarity metric between feature vectors, in such a way that two vectors extracted from cells from the same culture are considered as being similar, while two vectors corresponding to a pair of cultures that are known by the biologists to result in distinct pathologies are considered to be dissimilar. Given such a metric, our second objective is to derive clusters within the whole database, so as to identify the genes that result in similar/distinct
1.2. Problem statement

\( D^j = \{ d^j_i \}_{i \in [1,N_j]} \forall j = 0, ..., M \) denotes the set of \( N_j \) feature vectors that corresponds to the same \( j \)-th culture of samples (from a biological point of view). The index \( j=0 \) refers to the normal culture of data (samples corresponding to cells not affected by any silencer), while \( j=1, ..., M \) refers to \( M \) different types of abnormal cultures of data (from a biological point of view: samples corresponding to cells affected by different silencers).

We do not consider the case where a treatment produces healthy cells: \( D^j \) with \( j \neq 0 \) are always abnormal samples.

Let define a similarity function \( S \) for any pair of samples \( d^j_i \) and \( d^j_k \) belonging to the same culture \( D^j \) as similar with the value:

\[
S(d^j_i, d^j_k) = +1
\]

And for any pair of samples \( d^p_i \) and \( d^q_k \) (for \( p \neq q \)) belonging to culture \( D^p \) and culture \( D^q \) as dissimilar with the following value:

\[
S(d^p_i, d^q_k) = -1
\]

The objective is to cluster the samples from \( D^0 \) in a different cluster from the abnormal cultures \( D^j \) for \( j \neq 0 \) and also to cluster the abnormal samples.

Two samples \( d^j_i \) and \( d^j_k \) from \( D^j \) can be considered as belonging to the same cluster since their corresponding cells have been subject to the same treatment.

We can also affirm that the samples of any pair \( (d^a_i, d^a_k) \forall a \neq 0 \) are in different clusters because we know for sure that the cell corresponding to the second sample \( d^a_k \in D^a \) has a potential pathology whereas the first one \( d^a_i \in D^0 \) does not have it because \( D^0 \) is the normal culture.

There is a case where we have no information about how we should cluster two samples. We don't know if two samples \( d^a_i \) and \( d^b_i \) from different \( D^a \) and \( D^b \) cultures \((a \neq 0, b \neq 0 \) and \( a \neq b \)) are similar or not. We know the treatment process has been different but we cannot assure that the result of the process (our input data) is going to be different. Maybe two different treatments could obtain the same result and, indeed, in our topic, it is the case (different treatments of the cells produce the same nucleolus structure).

With all this information, our problem consists in learning a metric of similitude between pairs of samples (for all the possible needed combinations of pairs) that allows to cluster the samples with these conditions:

- The metric learned between a pair of samples corresponding to the normal culture \( D^p \) should allow to group the samples from that culture in the same cluster composed only of normal samples.
- The metric learned between a pair of samples from the same culture \( D^p \) should allow to group the abnormal samples from culture \( D^p \) in the same cluster (because if the treatment is the same, it is also the result).
- The metric learned between a sample from the normal culture \( D^0 \) and a sample from any
abnormal culture $D^a$ should allow to group the samples from each of these cultures in different clusters.

We can see that no condition is given for learning the metric of two samples corresponding to different abnormal cultures because sometimes they belong to the same cluster (pathology) and sometimes each abnormal culture belong to its own cluster.

Once a metric has been obtained between samples, a method to cluster all of them (using this metric) is needed following these objectives:

1. **Cluster the normal samples in a different cluster from the abnormal samples.**
2. **Cluster the abnormal samples depending on their features.**

### 1.3. Solution overview

Each sample is characterized by a vector of features extracted from an image. From now on we consider a sample as a point in a space of features. Each feature corresponds to a dimension of said space.

The next block diagram synthesizes the methodology we are going to use in this project in order to cluster a new set of samples (into the called clusters).

![Block diagram](image)

We obtain some training and test samples from cultures of samples. Of course, a sample in the training set cannot be in the test set. In figure 1.3.4 an example of training and test set is shown:
Fig 1.3.4: Training and test set; here each sample is represented by two features. Each color in the figure represents one culture. In other words, all the samples from the same color have been subject to the same treatment process (same silencers have been applied to cells). Clearly two cultures can be identical as the red and blue case. That means the treatments of the cultures have been different (marked with different colors) but the results showed in the space of features is the same.

In order to be able to cluster the four cultures from the test samples in three clusters a metric $m$ of similitude between the test samples must be computed and used as input of the Louvain Method. The figure 1.3.5 shows the result of the metric function able to compute a metric between two samples:

$$m(x^1, x^2)$$

Fig 1.3.5: Different possible results the $m$ function can obtain from a pair of new samples

The metric of similitude $m(x^1, x^2)$ for any pair of test samples $(x^1, x^2)$ (each one from any culture $D$) is used by the Louvain Method in order to cluster all the test samples (see fig. 1.3.6).

Fig 1.3.6: Clustering of samples (with a similarity metric between them) after applying the Luvain Method
To learn the metric function able to compute a metric of similitude between samples, a training set of samples is used. The training samples must belong to the same cultures used in the test set. Adaboost is the algorithm used to learn this metric function. The information the algorithm needs is the pairs from the training samples and the similitude $S$ between the samples of each pair. This similitude is defined as

$$S(x^1, x^2) = \begin{cases} +1 & \text{if } x^1, x^2 \in D^j \quad \forall j \in [0, M] \\ -1 & \text{if } x^1 \in D^j \text{ and } x^2 \in D^k \quad \forall j \neq k \in [0, M] \end{cases}$$

(same culture) (distinct cultures)

For Adaboost to know if the metric obtained from the training samples is good, it compares the sign of the metric with the sign of $S$.

Once the metric function is obtained, a metric between any sample can be computed. The goal is to be able to identify the cluster (output of Louvain Method) which a sample belongs to. In order to validate the results, we use the knowledge about which culture each test sample belongs to. From the cultures we can get the $S$ between test samples. That $S$ allows to validate if the similitude obtained by the metric is good or not.

Of course, future practical uses of our applications will obtain a metric from any set of samples without any knowledge about the culture it belongs to: just their features and the metric function will be needed to cluster the samples.

### 1.4. Outline of the Thesis

Once the solution of the problem has been established, we are going to explain in detail the Adaboost algorithm in Section 2. As it has been the algorithm implemented in the thesis, this section is the most detailed chapter.

In Section 3 the performance of the Louvain Method is explained in order to understand the bases of the algorithm used once the metric for clustering has been found.

Once the background is explained, in Section 4 the results are discussed as well as the methodology used to reach them.

Finally, in Section 5 some conclusions are written.
2. Learning similarity metrics with Adaboost

2.1. Introduction

This chapter faces the problem about how to learn a metric of similitude between samples. For that reason we start from a set of \(N\) pairs \(P = \{(x_i^1, x_i^2, l_i)\}_{i=1...N}\) labeled by \(l_i = S(x_i^1, x_i^2)\) for each sample \(x\) belonging to the space of features \(X\). The goal is to learn a metric function \(m\) which determines the grade of similitude between any pair of samples indicated by \(l_i = S(x_i^1, x_i^2)\). In Fig 2.1.1. a graphic example of samples is shown:

![Fig. 2.1.1. Each cross represents a sample in a 2D space of features. Pairs of samples belonging to the same color are similar \((l=+1)\) and samples from different colors are dissimilar \((l=-1)\).](image)

We must learn a metric function \(m\) which, for any pair \((x_i^1, x_i^2)\), the metric obtained is the one that ensures that this pair is similar or dissimilar as found by the result of \(S(x_i^1, x_i^2)\). Then, the objective is to obtain a metric function which:

- if the samples \(x_i^1\) and \(x_i^2\) of the pair \((x_i^1, x_i^2)\) are similar (i.e. \(S(x_i^1, x_i^2)=+1\)), then the metric \(m(x_i^1, x_i^2)\) should be positive (i.e. \(m(x_i^1, x_i^2)>0\)).

- if the samples \(x_i^1\) and \(x_i^2\) of the pair \((x_i^1, x_i^2)\) are dissimilar (i.e. \(S(x_i^1, x_i^2)=-1\)), then the metric \(m(x_i^1, x_i^2)\) should be negative (i.e. \(m(x_i^1, x_i^2)<0\)).

In other words, we are interested in a similarity metric function \(m(x_i^1, x_i^2)\) that measures the degree of similarity between both samples \(x_i^1\) and \(x_i^2\). Positive values of \(m(x_i^1, x_i^2)\) correspond to similarity; the highest is the value of \(m(x_i^1, x_i^2)\), the greatest is the similarity. Negative values of \(m(x_i^1, x_i^2)\) correspond to dissimilarity; the lowest the value of \(m(x_i^1, x_i^2)\) is, the greatest the dissimilarity is. Hence, the function should satisfy the following condition:

\[
\text{sign}(m(x^1, x^2)) = S(x^1, x^2)
\]

For the purpose of obtaining a metric as the one mentioned above, an algorithm called Adaboost\(^{[1]}\) has been used. Adaboost can learn the \(m\) function which allows to obtain the grade of similarity or dissimilarity as it is shown in Fig. 2.1.2.
Adaboost is an algorithm that improves the performance of a set of weak classifiers. The classical Adaboost algorithm obtains a classifier able to classify a sample in a class as built by Viola and Jones\cite{2}. Several improvements\cite{3} have been done. Some cases of similarity boosting have been also studied before and the results were successful\cite{4}. As we are interested in the similitude between pairs, the Adaboost algorithm we use is coined pairwise boosting\cite{5}.

Our Adaboost is pairwise boosting type. Our only use of adaboost is to obtain a metric function of similitude between samples. The algorithm usually use this metric as a combination of a set of weak classifiers to obtain a strong one, but we are not particularly interested in finding a strong classifier, our only interest is the similarity metric.

In our case, each weak classifier of this set tries to classify a pair of samples \((x^i_1, x^i_2)\) as similar ( +1 ) or dissimilar ( -1 ). It is enough that each weak classifier classifies better than 50% of the time (better than random). Adaboost use them in such a way that a set of these weak classifiers is iteratively selected.

At the end of this iterative process, a strong classifier is obtained by mixing all the weak ones. At each iteration the mix of the previous selected weak classifiers may still misclassify some samples; then a new weak classifier must be selected: the one which reduces, as far as possible, the total error. So, the new weak classifier is not intended for making a good classification: just to compensate the classification errors of all the previous weak classifiers.

The performance of the algorithm is presented in the next page.

A selected weak classifier is denoted as \(c_t\) for the iteration \(t\). Each \(c_t\) contributes to greater or lesser extent to classify a pair \((x^i_1, x^i_2)\) as similar ( \(c_t(x^i_1, x^i_2)=+1\) ) or dissimilar ( \(c_t(x^i_1, x^i_2)=-1\) ). The grade of contribution is provided by a parameter denoted as \(\alpha_t\), which also minimizes the global error minimizing the expression of \(Z_t\) (normalization of the weights).

This grade of contribution depends on the wrong classifications (of pairs) over the total number of pairs of samples. When a lot of classifications in iteration \(t\) are wrong we say we have a high global error in iteration \(t\). However, further in this chapter we explain the concept of the correlation \(r_t\).

Looking for the smallest global error in \(t\) is the same as looking for the maximum correlation.

Each pair of samples has a weight. These weights are useful to indicate that the pairs with biggest weights are the pairs with the biggest errors. The weight of a pair increases if the pair has been badly classified and otherwise decreases. With these weights we can consider more important to correctly classify the pairs with biggest weights (biggest errors) than others with small weights. The weights are normalized so they can be seen as a distribution of error for each pair. They can also be seen as a distribution of priority for each pair to be well classified by its similarity between samples.

For each pair \((x^i_1, x^i_2)\), the improvement of the classifiers that attempt to classify the pair as similar...
or dissimilar is given by
\[
C(x_1^i, x_2^i) = \text{sign}(\sum_{t=1}^{T} \alpha_t \cdot c_t(x_1^i, x_2^i))
\]

**Adaboost Algorithm**

**Given:** A set of pairs \( P = \{(x_1^i, x_2^i, l_i)\}_{i=1..N} \) labeled by \( l_i = S(x_1^i, x_2^i) \) for each sample \( x \) belonging to the space of features \( X \)

**Given:** A set of weak classifiers \( \{c_k\}_{k=1..K} \) where \( K \) is the number of weak classifiers in the set

**Output:** A set of functions \( c_t : X \times X \rightarrow \{-\alpha_t, \alpha_t\} \)

**Output:** A set of thresholds \( \tau_t \) for each function

1. Set initial set of weights \( W_1(i) = 1 / N \)
2. for all \( t = 1, \ldots, T \) do:
3. Let \( W^p = \sum_{i=1}^{N} W_t(i) \) and \( W^n = \sum_{i=1}^{N} W_t(i) \)
4. for all \( k = 1, \ldots, K \) do:
5. Compute for each sample \( x_i \) of each pairs \( P \) the projection function \( f(x_i) \)
6. Compute each feasible threshold \( \tau_k^i \) (see **Threshold rate Algorithm** in Section 2.3) of the \( k \)-th weak classifier between different projection functions for \( j=1,\ldots,J \).
7. Compute the correlation \( r(\tau_k^i) = \sum_{i=1}^{N} W_t(i) \cdot l_i \cdot c_t^i(x_1^i, x_2^i) \) between the labels \( l_i \) of each pair and the classification of that pairs by the \( k \)-th weak classifier with each feasible threshold \( \tau_k^i \).
8. Select \((k, j) = \arg\max_{k=1,\ldots,K, j=1,\ldots,J} [r(\tau_k^i)] \) and assign: \( \tau_j = \tau_k^i \) and \( r_i = r(\tau_k^i) \).
9. Select the weak classifier corresponding to the \( k \) (the one with the best \( r(\tau_k^i) \)):

\[
c_t = c_k
\]
10. Set \( \alpha_t = \frac{1}{2} \ln \frac{1+r_t}{1-r_t} \) that minimizes \( Z_t = \sum_{i=1}^{N} W_t(i) \cdot \exp(-\alpha_t \cdot l_i \cdot c_t(x_1^i, x_2^i)) \)
11. if \( \alpha_t \leq 0 \) stop; if \( r_t = 1 \) stop (the weak classifier and its selected threshold classifies perfectly all the pairs of samples)
12. Update the weights: \( W_{t+1}(i) = \frac{W_t(i) \cdot \exp(-\alpha_t \cdot l_i \cdot c_t(x_1^i, x_2^i))}{Z_t} \)

Let's remember that our objective is to find a metric function \( m \) that \( m > 0 \) if the samples are similar and \( m < 0 \) if they are different. We have already stated that \( \text{sign}(m(x_1^i, x_2^i)) = S(x_1^i, x_2^i) \) and, as we have defined the label \( l_i = S(x_1^i, x_2^i) \), it is clear that the function \( m \) must satisfy the condition:

\[
\text{sign}(m(x_1^i, x_2^i)) = l_i
\]

As \( C(x_1^i, x_2^i) = \text{sign}(\sum_{t=1}^{T} \alpha_t \cdot c_t(x_1^i, x_2^i)) \) tries to get the same sign as \( l_i \), we can compute the expression of \( m \) as:

\[
m(x_1^i, x_2^i) = \sum_{t=1}^{T} \alpha_t \cdot c_t(x_1^i, x_2^i)
\]

The outline of the rest of the chapter is as follows:
• In Section 2.2 we explain the conditions that weak classifiers must meet and which kind of weak classifiers we have used in our work. Each weak classifier needs a threshold in order to classify the pair of samples in a binary way.

• The procedure of the threshold evaluation used is explained in Section 2.3. The selection of each weak classifier depends on the errors accumulated by the previous weak classifiers and they are computed by assigning a weight for each pair of samples.

• The computation of these weights is explained in Section 2.4.

• A measure of goodness is needed to select each new weak classifier. This measure must use the weights explained in Section 2.4 because the new weak classifier must compensate the previous errors. That is why we compare the classification of the new classifier with the label of the most misclassified pairs with the previous weak classifiers. This parameter is called weighted correlation between the label and the classification and is explained in Section 2.5.

• In Section 2.6 the computation of the $\alpha_t$ parameter is explained. This parameter, also called in our work factor of goodness, provides the grade of contribution of each acquired weak classifier depending on the number of misclassifications and on the weights explained in Section 2.4.

• Finally, in Section 2.7 the computation of the metric with all the knowledge acquired is explained.

**2.2. Weak classifiers**

At each iteration, a new weak classifier is selected from the set of weak classifiers. Each of these weak classifiers must separate the space of features in two partitions. We can create as many types of weak classifiers as we can imagine. For example:

• A straight line separates a 2D space into two partitions
• A circle separates the 2D space in two parts: the region outside the circle and the region inside it.
• Any other bidimensional figure for a 2D space: a triangle, ellipse, rhombus, …
• Any n-dimensional figure for a n-D space as the sphere or cube in a 3D space.

The goal of the weak classifier is to classify each pair of samples as similar if both samples are in the same region or as dissimilar if they are in different regions. For example, if a circle is the weak classifier selected for the classification, it classifies a pair of samples as similar if both samples are inside or outside the circle (both in the same region) or as dissimilar if one is inside and the other is outside the circle (see figure 2.2.1).
Fig. 2.2.1 A weak classifier of circle type splitting the space and the result of its classification for a pair of samples if both are inside the circle (left), both are outside the circle (middle) or one inside and the other outside (right).

Each weak classifier is composed by some fixed parameters and a scalar called threshold. The threshold of a selected weak classifier depends on the samples from the training set while the other parameters do not. The way how this threshold is computed is explained in section 2.3. In order to understand this idea we expose some examples:

- **Circle**:
  a) The fixed parameters are the two coordinates of its center.
  b) The threshold (which must be a scalar) is its radius.

- **Square**:
  a) The fixed parameters are:
     1. Center of the square.
     2. Angle of orientation
  b) The threshold is the length from the center to one vertex.

- **Sphere** (in a 3D case):
  a) The fixed parameters are the three coordinates of its center.
  b) The threshold is any of the possible radius.

- **Straight line** (in 2D) which splits the space in two subspaces.
  a) The fixed parameter is its slope.
  b) The threshold is the minimum distance between the line and a reference point in the space of features.

- **Plane** (in 3D)
  a) The fixed parameters are the zenith and azimuth (its unitary vectors produce the perpendicular vector to the plane).
  b) The threshold is the minimum distance between the plane and a reference point in the 3D space of features.

Our decision has been to use a weak classifier similar to the circle. It has been adapted in order to become a sphere, hyper-sphere or superior in case the space of features have more than two or three kind of features. However, as the features have different order of magnitude the circle presents a
problem. Let suppose a space of features with its first dimension with an order of magnitude from 0 to 1 and the second dimension from 100 to 1000. The fixed parameter of a weak classifier can be the coordinate (0.4, 640).

Now, small circles (radius smaller than 1) are too small to be able to include the similar samples in a different region from its dissimilar samples. If the second feature from similar samples are between 600 and 700, the vertical distance between samples will be greater than 1 (than the radius). So a circle of radius from the magnitude order from the first feature is not able to include all the similar samples because the bigger magnitude order of the second feature. Big circles (radius between 0 and 1000 as the magnitude from the second feature) have a bigger radius than the vertical distance from the center to the similar samples included inside the circle, but they can not discriminate the similar from the dissimilar samples in the first feature (dimension) of the space.

In order to solve that problem, another figure based in the circle has been used: the ellipse (or ellipsoid, hyper-ellipsoid, etc).

The n-dimensional ellipsoid is oriented with each of its axis parallel to one coordinate axis. With that provision, the size of each axis of the ellipse can be adapted to the order of magnitude of the corresponding feature. In our previous 2D example, the minor axis of the ellipse will stay parallel to the first feature axis and measures about 1 unit long, whereas the major axis, about 1000 long, corresponds to the second feature.

For $\beta$ and $\gamma$ as the axis of a 2D space, we can define an ellipse by a center $(\beta_0, \gamma_0)$ and a semi-major and semi-minor axis lengths $a$ and $b$ (or $b$ and $a$) as:

$$\left(\frac{\beta - \beta_0}{a}\right)^2 + \left(\frac{\gamma - \gamma_0}{b}\right)^2 = 1$$

By multiplying the semi-major and semi-minor axis lengths by a scalar $R$ we can obtain as many ellipses with the same proportion $a/b$ as values we can give to $R$. So we can get:

$$\left(\frac{\beta - \beta_0}{a}\right)^2 + \left(\frac{\gamma - \gamma_0}{b}\right)^2 = R^2$$

and that way we can use $R$ as a threshold because $R$ reflects the global size of the ellipse, similar to the radius of the circle.

From this formula we can obtain the parameters from an ellipse:

- Fixed parameters: $\beta_0$, $\gamma_0$, $a/b$.
- Threshold: $R$.

To know if a sample is in one region of an ellipse or in the other, the distance from the center to the sample must be compared with the threshold $\tau$ (for the ellipse $\tau = R$). For that reason a projection function is needed to convert the sample into a single scalar.

The projection function, also called hash function in other works[6], for a single sample with $x_i^p = (\beta_i^p, \gamma_i^p)$ coordinates (where $p=1,2$ depending of which sample of the i-th pair it is) can be computed as:

$$f(x_i^p) = \sqrt{\left(\frac{\beta_i^p - \beta_0}{a}\right)^2 + \left(\frac{\gamma_i^p - \gamma_0}{b}\right)^2}$$

The function which determines if two samples are in the same region of the space, splitted by the $\tau$-
th selected weak classifier, is $c_t$. Thus, $c_t$ compares the threshold $\tau$ of any type of weak classifier with the scalar obtained from the projection function $f$.

$$
c_t(x^1, x^2) = \begin{cases} 
+1 & \text{if } (f(x^1) < \tau \land f(x^2) < \tau) \lor (f(x^1) \geq \tau \land f(x^2) \geq \tau) \\
-1 & \text{if } (f(x^1) < \tau \land f(x^2) \geq \tau) \lor (f(x^1) \geq \tau \land f(x^2) < \tau)
\end{cases}
$$

This function allows us to classify a pair of samples as similar or dissimilar. However, the classification of a weak classifier can be wrong. Indeed there is a lot of pairs badly classified. For example, consider the case of figure 2.2.2:

![Fig. 2.2.2: Samples (crosses) in a space of features splitted in such a way a lot of centers of weak classifiers of ellipse type form a grid. For example, one weak classifier selected is the one with the red center and a semi-major and semi-minor axis as it is shown.](image)

The selected weak classifier is an ellipse. The value of the semi-major axis $a$ is 500 and the value of the semi-minor axis $b$ is $10^{-10}$. The ellipse is centered in the red dot (from all the possible centers marked as blue dots). In this example we can see that all the green samples are inside the circle so they are classified as similar (because every one is in the same region of the space divided by the circle/ellipse). We can also observe the black samples are classified as similar between themselves because all the black samples are in the same region (outside the circle). It is the same case for the red ones and also for the blue ones. So, this classifier (this circle) is pretty good. However, the classification between a red and a black sample is wrong because they are dissimilar pairs (different colors) but they are both at the same region of the circle (outside it) and that means they are classified as similar. There is no way to separate those 4 groups with just one weak classifier. That is why we need to select more of them.

### 2.3. Threshold evaluation procedure

To obtain the threshold with the smallest error and the classification measures of true and false positive, an algorithm has been used. In this section we are going to describe the performance of this algorithm.
**Threshold rate Algorithm** (based in Greg Shakhnarovich algorithm\(^{[1]}\)):

**Given**: Set of labeled pairs of samples \( P = \{ (x_i^1, x_i^2, l_i) \}_{i=1}^N \) where \( l_i \) is the label which contains the information about if the pair is similar (\( l_i = +1 \)) or dissimilar (\( l_i = -1 \)).

**Given**: A projection function \( f \) in order to project a sample in the space of features as a scalar.

**Given**: Weights \( W = [w_1, w_2, \ldots, w_i, \ldots, w_N] \)

**Output**: Set of triples \( \{ (\tau_j, TP_j, FP_j) \}_{j=1}^J \) where the \( TP_j \) and \( FP_j \) are the estimated TP and FP rates for threshold \( \tau_j \).

13. Let \( v_{i,p} = f (x_i^p) \) for \( i=1, \ldots,N \) and \( p=1,2 \)

14. Let \( u_i < \ldots < u_{j-1} \) be the \( J-1 \) unique values of \( \{ v_{i,p} \} \)

15. Let \( \Delta_j = \frac{u_{j+1} - u_j}{2} \) for \( j=1, \ldots,J-2 \)

16. Let \( \tau_1 = u_1 - \Delta_1 \) and \( \tau_{j+1} = u_j + \Delta_j \) for \( j=1, \ldots,J-1 \)

17. For all \( i=1, \ldots,N \)
   (a) Let \( \delta_{i,1} = \begin{cases} +1 & \text{if } v_{i,1} \leq v_{i,2} \\ -1 & \text{if } v_{i,1} > v_{i,2} \end{cases} \)
   (b) Let \( \delta_{i,2} = \begin{cases} +1 & \text{if } v_{i,1} > v_{i,2} \\ -1 & \text{if } v_{i,1} \leq v_{i,2} \end{cases} \)

18. Sort records \( \{ (v_{i,p}, \delta_{i,p}, w_i, l_i) \}_{i=1}^N, p=1,2 \) by the values of \( v_{i,p} \)

19. for all \( j=1, \ldots,J \) do:
   (a) Let \( i_j = \max \{ i : v_i \leq T_j \} \)
   (b) \( TP_j = \sum_{l_i=1} w_i - \sum_{l_i, l_i+1} w_i \cdot \delta_i \)
   (c) \( FP_j = \sum_{l_i=1} w_i - \sum_{l_i, l_i-1} w_i \cdot \delta_i \)

The projection function \( f \) allows to obtain a scalar from each sample (some coordinates).

In the case of the circles, the scalar used is the distance from the sample to the center of the weak classifier. And, as we have seen, a similar concept works for an ellipse, where the threshold is:

\[
\tau = \left( \frac{\beta - \beta_o}{a} \right)^2 + \left( \frac{\gamma - \gamma_o}{b} \right)^2
\]

Thus, for a specific center \((\beta_o, \gamma_o)\) and proportion \(a/b\) the ellipse will be bigger or smaller selecting a bigger or smaller threshold.

The procedure to obtain the threshold of a weak classifier consists of three steps:

1. All samples are sorted by their projection function

2. For each consecutive adjoined samples a local threshold is calculated (we can consider two samples are adjoined if the sorted values obtained from the project function are adjoined)

3. The final threshold is selected based on the local thresholds calculated before.

Then, if we sort all the samples by the value \( u \) of the project function, it is very easy to use a middle value between two sorted ones obtained with the project function.
Fig 2.3.1. Two adjoining samples and a weak classifier with a threshold between them. The threshold is able to disjoint them classifying the pair as dissimilar (they are separated in different regions of the space).

So, for each weak classifier all the samples sorted by the unique values $u_1 < u_2 < ... < u_i < u_{i+1} < ... < u_N$ from the projection function of each sample, we obtain all the possible thresholds as:

$$\tau_{i+1} = \frac{u_{i+1} + u_i}{2}$$

But once we have calculated all the possible thresholds we need to keep the one which classifies better all the pairs. Thus, for a given threshold $\tau$, we obtain a weak classifier $c$ with a specific threshold. We use the weak classifier $c$ in order to estimate the expected true positive rate defined as:

$$TP_{rate} = \frac{TP}{WP} = E_{x^1, x^2 \mid S(x^1, x^2) = +1} [Pr(c(x^1, x^2) = +1)]$$

and the false positive rate:

$$FP_{rate} = \frac{FP}{WN} = E_{x^1, x^2 \mid S(x^1, x^2) = -1} [Pr(c(x^1, x^2) = +1)]$$

for a given pair of samples $x^1$ and $x^2$ and for a label $S(x^1, x^2) = l$ between samples. If a pair is classified as similar, the result is called positive. If the pair is really similar the result is a true positive result. If the pair is indeed dissimilar, then the result is a false positive one. $WP$ denotes the sum of the weights between each similar pair of samples and $WN$ denotes the sum of the weights between each dissimilar pair of samples.

The only way to estimate these parameters is from the available examples of similar and dissimilar pairs of samples. Thus, the approach used in order to estimate TP is to calculate the similar pairs that are not separated by the threshold, i.e. similar pairs which are in the same region in the space of features. FP is estimated in a similar way, measuring the dissimilar pairs (because it's false positive) not separated by the threshold (so classified as positive), i.e. dissimilar pairs in the same region which means they are misclassified as positive (similar).

Moreover, in the Adaboost algorithm, as we can see below, each pair of samples has a weight $W$ which may be interpreted as the probability of selecting that pair. So instead of calculating the percentage of pairs separated by the threshold, we calculate the sum of the weights of these pairs. It can be thought as a measure of the importance of each pair.
The TP is calculated as the sum of weights of the correctly classified similar (also called positive). The correctly classified similar pairs are the ones with its samples not separated by the Threshold, as the central pair showed in figure 2.3.2.

Fig. 2.3.2: Illustration of the methodology used to compute the ratio of similar pairs which are misclassified. Only the left samples (crosses) of the threshold in the illustration are used for the computation. The weights $W$ of a pair of samples are assigned to each sample and multiplied by $\delta$. So only the left sample from the splitted pair will be the one used for computing its similar pair as misclassified.

With the data in the figure $TP=(0.1+0.4+0.3)$. So now, the problem is how to find the $W=0.2$ related to the pair through the threshold, the false negative pair (because is similar and separated as dissimilar).

We must remove the weights from the pairs with the samples separated by the threshold (those classified as dissimilar). It is the central pair in the figure: we must subtract 0.2 from the total sum.

For that purpose, the weights of each pair of samples are assigned to both samples $x^1$ and $x^2$ (so we duplicate the number of weights).

Fig 2.3.3: Weights assigned to each sample of the pair with positive and negative values. To obtain a positive and negative sign for the weights the binary indicator $\delta$ seen in the algorithm is used. If we sum the all the values on the left side of the threshold we obtain the weight of the pair separated by the threshold, which is the objective in order to compute the $FP=1-0.2=0.8$.

As we can see in Fig2.3.3, the new weights on the samples are multiplied by the binary indicator $\delta$, -1 for the sample of the pair with highest projection and +1 for the sample with the lowest one. That way every pair has a total weight of zero and, of course, all the weights add to 0 too.

For each pair, the sample with the highest projection value $f$ has a negative weight whereas the sample with lower $f$ has a positive weight.

Now, if we only focus on the weights at one side of the threshold (let say the left), the sum of its
weights is not zero (+0.2 in the figure). That sum is exactly the value we must subtract from the total sum of weights to get the sum of correctly classified similar pairs.

The procedure for the false positive rate is the same, using the dissimilar pairs instead of the similar ones.

2.4. Weights updating

For each pair of samples, a weight is assigned. Initially all the pairs of samples have the same normalized weight. At each iteration a new weak classifier is selected. For each weak classifier selected, an error is produced in some pairs. Thus, in those pairs the weight is increased while the weight of the well classified pairs is decreased. However, the weight must be increased or decreased in such a way that all the classification errors done in the previous weak classifiers must be considered in order to don't repeat them in the selection of the next ones. That means to increase or decrease the current weight instead of the initial weights. The rule for weight updating is as follows:

\[
W_{t+1}(i) = \frac{W_t(i) \cdot \exp(-\alpha_t \cdot l(x_i^1, x_i^2) \cdot c_t(x_i^1, x_i^2))}{Z_t}
\]

Where the components are:
- \(t\) is the current iteration
- \((x_i^1, x_i^2)\) is the \(i\)-th pair.
- the label \(l(x_i^1, x_i^2)\) of similarity from the data base which says whether the pair is similar (+1) or dissimilar (-1)
- the result of the weak classifier \(c_t\)
- A normalization constant \(Z_t\) (because the sum of the weights of all the pairs must be one) calculated as:

\[
Z_t = \sum_{i=1}^{N} W_t(i) \cdot \exp(-\alpha_t \cdot l(x_i^1, x_i^2) \cdot c_t(x_i^1, x_i^2))
\]

After applying the normalization factor \(Z_t\), the new sum of weights is equal to 1:

\[
\sum_{i=1}^{N} W_{t+1}(i) = 1
\]

The \(\alpha_t\) parameter is explained later in this chapter, but it is a measure of goodness for the weak classifier \(c_t\). If the weak classifier is good for most of the pairs then \(\alpha_t\) is big and the weights of the misclassified pairs increase a lot (the weights of the well classified pairs decrease also a lot towards 0). Otherwise, if the classifier is bad for most of the pairs, \(\alpha_t\) is small and the weights of the misclassified pairs will increase just a bit (because there are a lot of misclassified pairs and every one must increase its weight). It is possible to know if a pair has been well or badly classified just comparing the similarity label \(l_i\) and the similitude obtained by the weak classifier \(c_t\) between samples. Hence:
- If the weak classifier classifies correctly a pair \((x_i^1, x_i^2)\), the product of both parameters is +1 so the exponential will decrease the weight of that pair in the next iteration.
- If the weak classifier misclassifies a pair \((x_i^1, x_i^2)\), the product of both parameters is -1 so
the argument of the exponential is positive and the weight of the misclassified pair will increase.

In Fig 2.4.1, we can see the behavior of the update for the pair \( i \):

![Fig. 2.4.1: Weight updating for a pair \((x_{i1}^1, x_{i2}^1)\) decreasing the weight (green) in case the pair has been well classified with the current weak classifier or increasing it (red) in case the pair has been misclassified with the current weak classifier.](image)

For the same weak classifier \( c_t \) (with its own factor of goodness \( \alpha_t \)), the green cross is an example of weight which has been decreased because the pair \( i \) has been correctly classified as similar or dissimilar and the red cross is an example of weight which has been increased because the pair \( i \) has been misclassified.

### 2.5. Weighted correlation

So, now we know how to update the weights in order to increase the ones from the misclassified pairs. The misclassified pairs contribute to the error of the weak classifier. As we are interested on choosing a weak classifier which is able to solve the classification errors of the pairs with biggest weights, this error depends on the distribution of weights and can be computed as:

\[
\epsilon_t = \sum_{i=1}^{N} W_t(i) \cdot [\{l_i \neq c_t(x_{i1}^1, x_{i2}^1)\}] \quad \text{such that} \quad \{a\} = \text{indicator function} = \begin{cases} 1 & \text{if } a = \text{true} \\ 0 & \text{if } a = \text{false} \end{cases}
\]

One can see clearly in the formula that the \( i \)th pair \((x_{i1}^1, x_{i2}^1)\) is badly classified if the similarity classification given by \( c \) doesn't correspond with the similitude assigned to the label \( l_i = S(x_{i1}^1, x_{i2}^1) \). Hence, each \( i \)th badly classified pair adds its weight over the total sum of weights (which adds to 1).

In our work, instead of the weighted error, we have used the correlation between the classification given by \( c \) and the similitude assigned to \( l_i \):

\[
r_t = \sum_{i=1}^{N} W_t(i) \cdot l_i \cdot c_t(x_{i1}^1, x_{i2}^1)
\]
The pairs \((x^1_i, x^2_i)\) that are badly classified add a weight \(W_i(i)\) to the correlation while the pairs which are well classified remove its weight. So the correlation \(r_i\) is greater for the \(c_i\) which classifies the pairs \((x^1_i, x^2_i)\) with the biggest \(W_i(i)\) in the same set of similar or dissimilar as the label \(l_i\) does.

It can be demonstrated we can use any of this concepts, weighted error or weighted correlation, because they are related by:

\[
\epsilon_i = \frac{1 - r_i}{2}
\]

**Demonstration**

\[
\epsilon_i = \frac{1 - r_i}{2}
\]

**Given:**

\[
r_i = \sum_{i=1}^{N} W_i(i) \cdot l_i \cdot c_i(i)
\]

for the \(i\) corresponding to the \(i\)th pair \((x^1_i, x^2_i)\) of samples and \(t\) corresponding to the current iteration. For ease we use \(c_i(i)\) instead of \(c_i(x^1_i, x^2_i)\) for the weak classification of the \(i\)th pair.

\[
\epsilon_i = \sum_{i=1}^{N} W_i(i) \cdot [I_i \neq c_i(i)]
\]

such that the function \([a]=\text{indicator function}=\begin{cases} 1 & \text{if } a=\text{true} \\ 0 & \text{if } a=\text{false} \end{cases}\)

\[
\sum_{i=1}^{N} W_i(i) = 1 \quad \text{because the weight is normalized}
\]

We can decompose the sum of weights in the well and badly classified pairs:

\[
\sum_{i=1}^{N} W_i(i) = \sum_{i=1}^{N} W_i(i) \cdot [I_i = c_i(i)] + \sum_{i=1}^{N} W_i(i) \cdot [I_i \neq c_i(i)]
\]

(1)

We can also decompose the correlation in the same way:

\[
\sum_{i=1}^{N} W_i(i) \cdot (l_i \cdot c_i(i)) = \sum_{i=1}^{N} W_i(i) \cdot [I_i = c_i(i)] - \sum_{i=1}^{N} W_i(i) \cdot [I_i \neq c_i(i)]
\]

(2)

The second term from (2), \(\sum_{i=1}^{N} W_i(i) \cdot [I_i \neq c_i(i)]\) is the definition of the error \(\epsilon_i\), so the definition of the correlation in (2) can be written as:

\[
r_i = \sum_{i=1}^{N} W_i(i) \cdot [I_i = c_i(i)] - \epsilon_i
\]

(3)

Moreover, the sum in (3) can be written in another way using (1):

\[
\sum_{i=1}^{N} W_i(i) \cdot [I_i = c_i(i)] = \sum_{i=1}^{N} W_i(i) - \sum_{i=1}^{N} W_i(i) \cdot [I_i \neq c_i(i)]
\]

(4)

Which (given the weighted error definition and the normalization of the weights) can be written as:

\[
\sum_{i=1}^{N} W_i(i) \cdot [I_i = c_i(i)] = 1 - \epsilon_i
\]

(5)

Finally, if (5) is substituted in (3), the result is:

\[
r_i = (1 - \epsilon_i) - \epsilon_i = 1 - 2 \cdot \epsilon_i
\]

Thus:

\[
\epsilon_i = \frac{1 - r_i}{2}
\]
However, there is another way to calculate the correlation. We can think in the correlation as the difference between the well classified pairs and the badly classified. The first group is the one which makes positive the multiplication \( S(x_i^1, x_i^2) \cdot c_t(x_i^1, x_i^2) = +1 \) and the group of badly classified pairs is the ones with the result \( S(x_i^1, x_i^2) \cdot c_t(x_i^1, x_i^2) = -1 \).

All the pairs must be computed with its weight. Thus, we can compute the weighted number of pairs as the sum of the weights from the pairs which are similar (and we call these weights positive weights) plus the weights from the not similar pairs and we call these weights negative weights. In fact this means we have 2 sets, a set of weights with \( S(x_i^1, x_i^2) = +1 \) denoted as \( W^p \) and a set with \( S(x_i^1, x_i^2) = -1 \) denoted as \( W^n \). Both sets can be decomposed into 2 subsets, one with \( c_i(x_i^1, x_i^2) = +1 \) and another one with \( c_i(x_i^1, x_i^2) = -1 \).

So now indeed we have 4 disjoint sets: the True Positive set \( TP \) (similar pairs classified as similar), the False Negative set \( FN \) (similar pairs classified as dissimilar), the True Negative set \( TN \) (dissimilar pairs classified as dissimilar) and the False Positive set \( FP \) (dissimilar pairs classified as similar). Looking at the figure we can decompose the groups of well and badly classified weighted pairs in combinations of the subsets as \( r_t = (TP + TN) - (FP + FN) \).

\[ r_t = (TP + TN) - (FP + FN) \]

![Venn Diagram](image)

Fig 2.5.1: Venn diagram composed by the similar pairs (above) with its weights and dissimilar pairs (below) with its weights too. Each subset of the Venn diagram is used in order to compute the well and badly classified pairs.

However, as we can observe in the figure, we are interested in expressing the well and badly classified groups in terms of the known parameters. It is very easy to know the values of \( W^p \) and \( W^n \). And as we have explained in the Threshold evaluation procedure we also know the \( TP \) and \( FP \) parameters. The only unknown parameters are the \( TN \) and \( FN \) so we can express the correlation as follows:

\[ r_t = (TP + TN)_{\text{well classified pairs}} - (FP + FN)_{\text{badly classified pairs}} = (TP + W^n - FP) - (FP + W^p - TP) \]

And the final expression is:
\[ r_t = 2 \cdot (TP - FP) + W^n - W^p \]

The concept behind this formula is the same as in \( r_t = \sum_{i=1}^{N} W(t) \cdot l(i) \cdot c_t(x^i_1, x^i_2) \): Adaboost chooses the weak classifier with the biggest correlation between the similarity indicated on the label \( l \) and the result of the classification by \( c_t \). \( W^p \) and \( W^n \) are fixed (because they depend from the label which says if the pair is similar or not). Thus, Adaboost chooses the weak classifier with the biggest TP (if TP is bigger, then for a fixed \( W^p \) it means FN is smaller) and the smallest FP, so for a fixed \( W^n \) the TN (the dissimilar or also called Negative pairs classified as dissimilar/negative) is the biggest one.

### 2.6. Factor of goodness

In the previous sections the parameter \( \alpha_t \) appeared. This parameter measures the goodness of the selected weak classifier. If the weak classifier is a good one, \( \alpha_t \) is very high (it tends to infinity). In the opposite case, it tends to 0 (we suppose \( \alpha_t > 0 \)). Hence, we can see that the objective of \( \alpha_t \) is to minimize the error in the classification. Because of the limits where it tends, we use it in order to construct a classifier composed by all the selected weak classifiers:

\[
C(x^i_1, x^i_2) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t \cdot c_t(x^i_1, x^i_2) \right)
\]

For each iteration we have been interested in selecting a weak classifier which solves the accumulated errors (i.e. the weighted errors) obtained from the previous classifiers, but it must not be confused with the error obtained from the output function of Adaboost (function composed by the weak classifiers selected). We are interested in minimize the classification error obtained from the set of selected weak classifiers. This error is the percentage of wrong classifications. So it can be expressed as:

\[
E = \frac{1}{N} \sum_{i=1}^{N} \left[ [C(x^i_1, x^i_2) \neq l(x^i_1, x^i_2)] \right]
\]

for a given number \( N \) of pairs and for each pair \( i \). For ease of nomenclature, we are use \( C(i) \) and \( l(i) \) for a pair of samples \( (x^i_1, x^i_2) \) instead of \( C(x^i_1, x^i_2) \) and \( l(x^i_1, x^i_2) \).

We have seen in the weighted correlation section how to update the weights using this methodology:

\[
W_0(i) = \frac{1}{N}
\]

\[
W_{t+1}(i) = W_t(i) \cdot \frac{\exp (-\alpha_t \cdot l(i) \cdot c_t(i))}{Z_t}
\]

For a given normalization factor \( Z_t \) and for the iterations from \( t=1 \) to the last iteration (and weak classifier) \( T \), the next inequation can be proved:

\[ E \leq \prod_{t=1}^{T} Z_t \]
Demonstration \[ E \leq \prod_{t=1}^{T} Z_t \]

Given:

\[ E = \frac{1}{N} \sum_{i=1}^{N} [[C(i) \neq l(i)]] \quad (1) \]

\[ W_{T+1}(i) = \frac{\exp\left(-\sum_{t=1}^{T} \alpha_t \cdot l(i) \cdot c_t(i)\right)}{N \cdot \prod_{t=1}^{T} Z_t} \quad (2) \]

\[ [[C(i) \neq l(i)]] \leq \exp\left(-l(i) \cdot \sum_{t=1}^{T} \alpha_t \cdot c_t(i)\right) \quad (3) \]

The inequation (3) is always true and it continues being true if we add a sum for all the pairs and if we divide both sides of the expression (3) by the numbers of pairs:

\[ \frac{1}{N} \sum_{i=1}^{N} [[C(i) \neq l(i)]] \leq \frac{1}{N} \sum_{i=1}^{N} \exp\left(-l(i) \cdot \sum_{t=1}^{T} \alpha_t \cdot c_t(i)\right) \quad (4) \]

It is possible to identify the left side of (4) as the definition of the error in (1), so:

\[ E \leq \frac{1}{N} \sum_{i=1}^{N} \exp\left(-l(i) \cdot \sum_{t=1}^{T} \alpha_t \cdot c_t(i)\right) \quad (5) \]

The right side of (4) and (5) is similar to the numerator from (2). Indeed, (2) can be rewritten as:

\[ \frac{1}{N} \cdot \exp\left(-\sum_{t=1}^{T} \alpha_t \cdot l(i) \cdot c_t(i)\right) = W_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t \quad (6) \]

If we add a sum to both sides of (6) the result is:

\[ \frac{1}{N} \cdot \sum_{i=1}^{N} \exp\left(-\sum_{t=1}^{T} \alpha_t \cdot l(i) \cdot c_t(i)\right) = \sum_{i=1}^{N} W_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t \quad (7) \]

Now the left side of (7) is identical to the right side of (5) so we can express the error as:

\[ E \leq \sum_{i=1}^{N} W_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t \quad (8) \]

\[ \prod_{t=1}^{T} Z_t \] is a constant for the i and for each iteration t the weights are all normalized. For that reason we can express (8) as:

\[ E \leq \sum_{i=1}^{N} W_{T+1}(i) \cdot \prod_{t=1}^{T} Z_t = \left(\prod_{i=1}^{N} Z_i\right) \cdot \left(\sum_{i=1}^{N} W_{T+1}(i)\right) = \prod_{t=1}^{T} Z_i \cdot \prod_{t=1}^{T} Z_t \]

We can observe in the inequation \[ E \leq \prod_{t=1}^{T} Z_t \] that it is possible to minimize the error if we minimize \[ Z_t \] at each iteration \[ t \]. We have seen in the previous section the expression of \[ Z_t \]:

\[ Z_t = \sum_{i=1}^{N} W_i(i) \cdot \exp\left(-\alpha_t \cdot l(i) \cdot c_t(i)\right) \]

In this expression there is only one parameter which we have not defined, the goodness factor. So to
minimize the error we must choose an $\alpha_i^{[7]}$ at each iteration which minimizes $Z$. If we derive $Z$, the result depends on the correlation $r_i$:

$$\alpha_i = \frac{1}{2} \cdot \ln \frac{1 + r_i}{1 - r_i}$$

\[ \text{Demonstration} \quad \alpha_i = \frac{1}{2} \cdot \ln \frac{1 + r_i}{1 - r_i} \]

We derivate $Z$:

$$\frac{d}{d(\alpha_i)} Z_i = 0 \quad (1)$$

The expression obtained is not simple, but for a variable $u \in [-1, 1]$ we can write an upper bound for the $Z_i$ expression as it follows:

$$\sum_{i=1}^{N} W_i(i) \cdot \exp\left(-\alpha_i \cdot u_i(i)\right) \leq \sum_{i=1}^{N} W_i(i) \cdot \left(\frac{1 + u_i(i)}{2}\cdot\exp(-\alpha_i) + \frac{1 - u_i(i)}{2}\cdot\exp(c_i)\right) \quad (2)$$

Our case is a particular one where $u_i(i) = l(i) \cdot c_i(i) \in [-1,1]$ and the inequation (2) is equal just when $u \in [-1,1]$ . Now we can rewrite the derive of $Z$ as:

$$\frac{d}{d(\alpha_i)} \left(\sum_{i=1}^{N} W_i(i) \cdot \left(\frac{1 + u_i(i)}{2}\cdot\exp(-\alpha_i) + \frac{1 - u_i(i)}{2}\cdot\exp(c_i)\right)\right)$$

And equating to 0 (in order to find the $\alpha_i$ for the minimum $Z$):

$$\frac{d}{d(\alpha_i)} \left(\sum_{i=1}^{N} \left(\frac{1}{2}\cdot\exp(-\alpha_i) + \frac{1 - u_i(i)}{2}\cdot\exp(c_i)\right)\right) = 0$$

After deriving, the result is:

$$- \sum_{i=1}^{N} \left(W_i(i) + W_i(i) \cdot u_i(i)\right) \cdot \exp(-\alpha_i) + \sum_{i=1}^{N} \left(W_i(i) - W_i(i) \cdot u_i(i)\right) \cdot \exp(c_i) = 0$$

Or what is the same:

$$\left(\sum_{i=1}^{N} W_i(i) + \sum_{i=1}^{N} W_i(i) \cdot u_i(i)\right) \cdot \exp(-\alpha_i) = \left(\sum_{i=1}^{N} W_i(i) - \sum_{i=1}^{N} W_i(i) \cdot u_i(i)\right) \cdot \exp(c_i)$$

The sum of all the weights is equal to 1 so the expression results as:

$$\left(1 + \sum_{i=1}^{N} W_i(i) \cdot u_i(i)\right) \cdot \exp(-\alpha_i) = \left(1 - \sum_{i=1}^{N} W_i(i) \cdot u_i(i)\right) \cdot \exp(c_i)$$

And if we substitute $u_i(i) = l(i) \cdot c_i(i)$ , we can identify the weighted correlation:

$$r_i = \sum_{i=1}^{N} W_i(i) \cdot u_i(i) = \sum_{i=1}^{N} W_i(i) \cdot l(i) \cdot c_i(i)$$

Then the equality results as:

$$\left(1 + r_i\right) \cdot \exp(-\alpha_i) = \left(1 - r_i\right) \cdot \exp(c_i)$$

And isolating $\alpha_i$ it results:

$$\exp(2 \cdot \alpha_i) = \frac{1 + r_i}{1 - r_i}$$

Thus,

$$\alpha_i = \frac{1}{2} \cdot \ln \frac{1 + r_i}{1 - r_i}$$
With the $\alpha_t$ expression which minimizes $Z_t$ and then the error, we can substitute $\alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t}$ at $Z_t$ expression and the new one obtained is:

$$Z_t = \sqrt{1 - r_t^2}$$

To demonstrate this expression it is needed to use the expression used in the previous demonstration

$$Z_t = \sum_{i=1}^{N} W_t(i) \cdot \left( \frac{1 + l(i) \cdot c_t(i)}{2} \cdot \exp(-\alpha_t) + \frac{1 - l(i) \cdot c_t(i)}{2} \cdot \exp(+\alpha_t) \right)$$

and substitute $\alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t}$ on it.

In order to minimize $Z_t$, we can see in the minimized expression of $Z_t$ that the solution is to maximize $r_t$. That makes sense. To reduce the total error of the classification we must maximize the weighted correlation for each iteration (so each time we select a new weak classifier). Because of the relation $\epsilon_t = \frac{(1 - r_t)}{2}$, it is logical to think that to minimize the error of the output function obtained by Adaboost we must select the weak classifiers with the minimum weighted error at each iteration.

This is the mathematical reason why, when we have explained the weighted correlation, we were looking for the weak classifier with the biggest correlation. It minimizes the total error, what is logic according to the concept of the weighted correlation we have seen before.

Now we can also represent the factor of goodness in function of $r_t$:

![Figure 2.6.1](image.png)

Fig 2.6.1: Plot of the function used in order to compute the goodness factor, $\alpha_t$, from the weighted correlation of the selected weak classifier.

As we can see, if the weighted correlation obtained by the weak classifier is very good (near to 1), the factor of goodness tends to infinity. Otherwise, if the weighted correlation is below 0.8 approximately, $\alpha_t$ is just proportional.

### 2.7. Metric from the improved performance of the weak classifiers

A graphic representation of the argument inside the sign expression
\[ C(x^1_i, x^2_i) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t \cdot c_t(x^1_i, x^2_i) \right) \]

is the next one:

\[ m(x^1_i, x^2_i) = \sum_{t=1}^{T} \alpha_t \cdot c_t(x^1_i, x^2_i) \]

for a pair \((x^1_i, x^2_i)\) drawn as green crosses. The pink circle are the different weak classifiers \(c_t\) and \(\alpha_t\), factor of goodness, is just a scalar (the better \(c_t\) the bigger \(\alpha_t\)).

Which can be interpreted as:

\[ m(x^1_i, x^2_i) = \sum_{t=1}^{T} \alpha_t \cdot c_t(x^1_i, x^2_i) \]

for a pair \((x^1_i, x^2_i)\) with the result of each weak classifier \(c_t\) shown for the plotted pair.

After learning the weak classifiers, using samples from a training set with the same distributions as in the test set, we can calculate if a new pair from the test set is similar or dissimilar by computing the sign function of the number obtained after from the figure. However, the aim of this project is to obtain a metric of similitude in order to cluster by this metric. That is the reason why the objective of Adaboost is to compute this operation

\[ m(x^1_i, x^2_i) = \sum_{t=1}^{T} \alpha_t \cdot c_t(x^1_i, x^2_i) \]

between test samples after learning the weak classifiers using training samples from the same kind of distributions as the test ones.
3. Louvain Method

3.1. Introduction

In the previous section a metric between samples has been found. A big value of the metric between two samples means that they are very similar and probably both samples belong to the same cluster. However, if two cultures have similar distributions of its samples, the similarity value between one sample of each culture will be greater than 0 (a positive value means the pair of samples are similar and a negative one means the samples are dissimilar).

For that reason, a clustering using this similarity relation is needed because maybe two different cultures must be clustered in just one cluster. From a biological point of view this means that 2 cultures of cells affected in different ways (different silencers applied) have produce the same result (same nucleolus structure so same pathology).

Several papers have been published about clustering by similarity, as seen in Newmann and Girvan[8]. Here the Louvain Method[9] is used:

![Fig. 3.1.1: Input and output of the Louvain Method](image)

As we can see, the Louvain Method would classify the samples (each node) looking at the links between all the nodes. The case above is an easy example but in a real case we have more kind of a bit similar links (or edges) and with different grades.

In this project, the Louvain Method has not been implemented but, as it is an important step in order to achieve our goal, we are going to have an overview of it.

3.2. Concepts

In the Louvain Method the samples are called nodes, and each pair of different nodes are linked through an edge. A weighted graph is so created, where the weight of an edge is the metric of similitude between the two nodes (samples) that the edge links.

From that graph, a matrix of weights A is defined. Each element of the matrix corresponds to an edge between one node (a row) and another (a column).

A binary matrix is also defined to indicate if there is an edge between samples or not. In our work, if we have not paired two test samples (nodes) no metric is computed and this must be indicated in the binary adjacency matrix B.

In figure 3.2.1, 4 nodes have been represented and also an edge (metric value) between them:
From this graph we can see there is no edges between some nodes. The next table indicates with a 1 the nodes connected by an edge and with a 0 the nodes that do not do that:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This kind of table allows to write the adjacency matrix as follows:

\[
G = \begin{pmatrix}
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{pmatrix}
\]

From the values in the graph a table of weights can be filled:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

And, given an order of nodes (in the example a, b, c, d), the table can be represented as the following matrix:

\[
A = \begin{pmatrix}
0 & 1 & 3 & 1 \\
1 & 0 & 2 & 0 \\
3 & 2 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{pmatrix}
\]

A new concept appears in this method. It is the concept of *community*. We could think in a community as a class, and in fact it is a class created from the similarity edges.

We could have as many communities as nodes we have (this is indeed the initial step of the method). These communities of nodes are continuously resized in the Louvain algorithm in order to
look for the best clustering. We denote as \( c_i \) the community in which the node \( i \) belongs. Let consider another node \( j \). It belongs to the \( c_j \) community. \( c_j \) can be the same community as \( c_i \) or not. The next function returns 1 if they are the same community and 0 if they are not:

\[
\delta(c_i, c_j)
\]

The concept that defines which is the best clustering is the concept of modularity. The modularity is defined as the difference between the real percentage of edges (sum of its values) inside the communities and the percentage inside communities if they are randomly distributed. The modularity \( Q \) is computed as:

\[
Q = \frac{1}{2 \cdot m} \cdot \sum_{i,j} (A_{i,j} - \frac{k_i \cdot k_j}{2 \cdot m}) \cdot \delta(c_i, c_j)
\]

Given the communities \( c_i \) and \( c_j \) from the node of the row \( i \) and the one in column \( j \) respectively from the matrix \( A \), the edge \( A_{i,j} \), the sum \( m \) of all the weights from the edges of the graph \( m = \frac{1}{2} \sum_{i,j} A_{i,j} \), the sum \( k_i \) of the weights from the edges of node \( i \) \( (k_i = \sum_j A_{i,j}) \) and the sum \( k_j \) of the weights from the edges of the node \( j \).

**Demonstration**

\[
Q = \frac{1}{2 \cdot m} \cdot \sum_{i,j} (A_{i,j} - \frac{k_i \cdot k_j}{2 \cdot m}) \cdot \delta(c_i, c_j)
\]

Given:

- the probability of having one edge between the nodes \( i \) and \( j \) \[^{[10]}\]:
  \[
P_{i,j} = \frac{k_i \cdot k_j}{2 \cdot m}
\]

- the real percentage of edges (sum of its values) inside the communities
  \[
  \sum_{i,j} A_{i,j} \cdot \delta(c_i, c_j)
  \]
  \[
  2 \cdot m
  \]
  \[^{(2)}\]

- the percentage of the sums of the values of edges from the nodes inside communities if they are distributed randomly (calculated as all the possible edges inside the same communities and normalized over all \( A_{ij} \))
  \[
  \sum_{i,j} P_{i,j} \cdot \delta(c_i, c_j)
  \]
  \[
  2 \cdot m
  \]
  \[^{(3)}\]

Modularity is defined as the difference between (2) and (3):

\[
Q = \sum_{i,j} A_{i,j} \cdot \delta(c_i, c_j) - \sum_{i,j} P_{i,j} \cdot \delta(c_i, c_j)
\]

Substituting (1) in (4):

\[
Q = \sum_{i,j} \frac{A_{i,j} \cdot \delta(c_i, c_j)}{2 \cdot m} - \sum_{i,j} \frac{P_{i,j} \cdot \delta(c_i, c_j)}{2 \cdot m} = \sum_{i,j} \frac{A_{i,j} \cdot \delta(c_i, c_j) - \frac{k_i \cdot k_j}{2 \cdot m} \cdot \delta(c_i, c_j)}{2 \cdot m}
\]

And applying common factor to the \( \delta(c_i, c_j) \)

\[
Q = \frac{1}{2 \cdot m} \cdot \sum_{i,j} (A_{i,j} - \frac{k_i \cdot k_j}{2 \cdot m}) \cdot \delta(c_i, c_j)
\]
3.3. Method

The Louvain Method uses an iterative process decomposed in two steps. Each iteration is called pass and all the passes use the same procedure in the two steps explained below.

The method starts with each node being its own community:

![Fig 3.3.1: Start point from the Louvain Method](image)

3.3.1. 1st step

The first step consist in trying to include each node in each community (including itself). The figure 3.3.1.1 show all the possibilities for node $a$.

![Fig 3.3.1.1: first possible substep](image)

If we suppose the third one is the possibility with the biggest modularity $Q$, the next node will do the same starting with the communities as:

![Fig 3.3.1.2: Communities of nodes which compute the biggest Q](image)

So the possibilities of $b$ belonging to one of this four communities (including itself) are:
Fig 3.3.1.3: second possible substep
We could suppose now that the case with the biggest modularity is the second one, when $b$ stays in its own community.

![Image showing second possible substep]

Fig 3.3.1.4: Communities of nodes which compute the biggest $Q$

So $c$ will do the same with that configuration:

![Image showing third possible substep]

Fig 3.3.1.5: third possible substep

Now we could suppose the second option as the one with the best modularity:

![Image showing second option for modularity]

Fig 3.3.1.6: Communities of nodes which compute the biggest $Q$

After applying the same procedure to the node $d$ and $e$ the result could be:

![Image showing result after applying procedure to $d$ and $e$]

Fig 3.3.1.7: Communities of nodes which compute the biggest $Q$ at the end of the first step
3.3.2. 2nd step

The second step supposes each community becomes each own node. So using the same example we would have a new set of nodes:

These 2 steps form a pass because both steps are repeated consecutively until there is no change between one pass and the previous one. So after applying the next passes maybe the communities are like the next figure:

In the figure above we have two communities and we cannot reduce more the number of them. If the Louvain Method stops here it means that it is not possible to have a bigger modularity, i.e, the dissimilarity between the nodes of different communities is so strong that it's not possible to make it higher.
4. Results validation

4.1. Introduction

In this chapter we discuss the results obtained from our implemented Adaboost and its combination with the Louvain Method in order to classify a sample as normal or abnormal and also in order to cluster the abnormal samples.

The metric function learned from Adaboost follows the formula seen in Chapter 3:

\[ m(x^1, x^2) = \sum_{t=1}^{T} \alpha_t \cdot c_t(x^1, x^2) \]

for a pair \((x^1, x^2)\) of samples as input.

We evaluate the metrics obtained for each metric function \(m\) learned, which is composed by a set of \(T\) weak classifiers \(c_t\) and their factor \(\alpha_t\) of goodness. The objective is to obtain a metric between samples as input for the Louvain Method able to cluster the samples in groups where:

- All pairs \((x^1, x^2)\) with \(x^1, x^2 \in D^0\) of normal samples must have a high metric of similitude to group them in the same cluster without abnormal samples.
- All pairs of the samples \((x^1, x^2)\) with \(x^1, x^2 \in D^a\) belonging to an specific culture must have a high metric of similitude to group them into the same cluster.
- Pairs of abnormal samples \((x^1, x^2)\) with \(x^1 \in D^{a1}\) and \(x^2 \in D^{a2}\) from different cultures with similar distributions (same mean and variance) must have a high metric of similitude to group them in the same cluster.
- Pairs of samples \((x^1, x^2)\) with \(x^1 \in D^{a1}\) and \(x^2 \in D^{a2}\) from different cultures with dissimilar distributions must have a high metric of dissimilitude (negative metric) to group them in different clusters.

As no real data was ready when this project was started, we are working on artificial data instead of data from real cultures. In Section 4.2. is presented how this artificial data (feature vectors) has been generated. In Section 4.3. the methodology used, in order to obtain some results, is explained. Finally in Section 4.4. the validations are done for different configurations and different kind of pairing.

4.2. Input data

In our work, no real data has been used because the feature vectors have not been furnished to us. In order to plot the results only 2 features have been generated. To ensure that the algorithm works, different random configurations (of samples from vectors of 2 features) have been generated.

Before proceeding we must establish some concepts:

- **set**: or set of samples, a concrete set of points in the space of features
- **type of distribution**: generic type of statistical distribution (Gaussian, uniform, ...)
- **distribution**: a concrete statistical distribution (example: Gaussian with mean 6.2 and variance 2.3), used to generate the samples of a culture. It is possible to generate multiple cultures with the same distribution in a given set.
- **configuration**: collection of distributions, one distribution for each culture, which allows us to generate a set of samples. One configuration generates several realizations. A realization
of a configuration consists in a set of samples. Each realization (set of samples from the same configuration) is different since the samples of each set can vary at random according to the distribution that generates each one.

In Fig. 4.2.1, some of these configurations with fixed mean and variance chosen by us are shown.

![Configurations](image1)

Fig. 4.2.1. Configurations of Gaussian samples from 2 cultures (a and b), 3 cultures (c and d) and 4 cultures (e and f) using different means and variance in order to compare the difference between the results obtained from that changes. Each cross is a sample (coordinate in the 2D-space of features) and each color corresponds to a culture. The green ones are the crosses from the normal culture.

In a real case we do not know the features. Thus, we have also generated more complex configurations with random means and variances for each culture. For these configurations we have selected different kind of features and as we does not know which ones will be those extracted from real cells, we have assigned a random magnitude order to each axis (each feature dimension). This is because a feature could be the size of the nucleolus so about micrometers and another feature could be the grey intensity of the nucleolus image so between 0 and 1.

Hence, the cases of configurations in Fig 4.2.2 have random type of features (different magnitude orders between axis) and each culture has a random mean and a random variance.

![Configurations](image2)

Fig 4.2.2. Configurations of Gaussian samples with random cultures (random means and random variances) inside a random space of features.
From these cases we have used the case (b) because it is workable and not like c, with features badly extracted because they are not useful for visually identifying the difference between normal and abnormal (black) samples. And we have used also (b) because we can visually distinguish the clusters (not like a because the superposition of blue and black makes not clear if they should be in different clusters).

Of these cases we use the case (b). Case (c) is inappropriate because the green and black samples are too close to be clustered separately so new features must be extracted to differentiate visually the normal from all the abnormal cultures In case (a) the superposition of blue and black makes not clear if they should be in different clusters.

Moreover all these cultures belongs to Gaussian distributions, but in real case the distributions can be non-Gaussian. For that reason, also different types of distributions have been used in our work. More specifically, an \textit{arc of a circle} with

- a random center chosen with an uniform variable between the minimum and maximum value of each feature.
- a random radius (using a Gaussian variable) with mean chosen with an uniform variable.
- An angle randomly chosen for each sample between a random initial and final angle of the culture:
  - Initial angle: chosen with an uniform variable from 0 to $2\pi$.
  - Final angle: chosen with an uniform variable between 0 and $2\pi$.

The Fig. 4.2.3. shows the 2 cases with this type of distribution, analysed in \textit{Section 4.4}.

![Fig 4.2.3: Configurations of Mixed random cultures](image)

The pink arc on the right is well isolated. With only just one weak classifier, the pink culture can be well classified so it is no interesting because it does not provide any added difficulty to the case of a Gaussian distribution that occupy the same area. For that reason we use the configuration from the left, which has a Gaussian distribution isolated but the two arcs are mixed with the other cultures (and poses a new challenge).
4.3. **Methodology**

4.3.1. **Training and test set**

Two sets of samples being realizations from the same configuration have been used for each validation:

- **Training set**: Adaboost algorithm uses this set of samples to learn the best metric function composed of weak classifiers.

- **Test set**: This set of samples is used to check whether the learned metric is able to correctly cluster the new (test) samples. These samples are clustered with the metric through the Louvain Method in order to check if the clusters are correct or not.

Both realizations (training and test sets) must have identical distributions of samples. Only the samples must be random. For instance, in Fig. 4.3.1 an example of training and test set is presented. They clearly have identical distributions but the samples of each set (realization) are different from the samples from the same culture (distribution) belonging to the other set (realization).

![Fig 4.3.1](image_url)

Fig 4.3.1: Two realizations from the same configurarions. Any of them could be used as the Training set of samples and the other as the Test set.

Thus, for each sample from the test set, we know the ground truth, thus the culture each sample belongs to. This ground truth will allow us to check whether the implemented algorithm works well or not. If it does, for any new sample (without knowledge of the ground truth) the system is able to achieve the goals stated at the end of Section 1.2, that is:

1. To separate the abnormal samples from the normal (green crosses in all the figures) ones.
2. To cluster the abnormal samples by the values of its features.

4.3.2. **Types of pairing**

Adaboost uses pairs as input in order to learn the similitude between the training samples and which are the weak classifiers that provide the most accurate similarity metric function. All the pairs formed for the learning step are evaluated by the Adaboost.

The correlation between pairs similitude from the ground truth and the similitude obtained by the metric function applied to the same pairs must be maximum. So, the metric function depends on the
chosen pairs, it will be different depending on the pairing performed.

The conditions given in the problem statement are:

- A metric between any normal sample (green crosses in all the figures) and any abnormal sample must be learned because they are clearly dissimilar and the metric we want between them must be very negative:

\[ m(x_1, x_2) \ll 0 \text{ if } x_1 \in D^0 \text{ and } x_2 \in D^a \forall a \neq 0 \]

- A metric between two samples from the same culture must be learned too because two samples from the same culture must be in the same cluster for sure. Maybe two abnormal samples from different cultures are in the same cluster too but we certainly don't know anything between different abnormal cultures a priori.

\[ m(x_1, x_2) \gg 0 \text{ if } x_1, x_2 \in D^i \forall i \]

Considering these conditions we have used three kind of pairing as input for the Adaboost algorithm: Single pairing, Complete pairing and Multi-single pairing. The metrics obtained by Adaboost depends on which kind of pairing we have used as input. These types are presented next.

### 4.3.2.1. Single pairing

Is totally based in the conditions given by the problem statement. Because of them, we are not interested, a priori, in pairing two samples from different abnormal cultures. So, the pairs should be formed as shown in Fig. 4.3.2.1.

![Single Pairing Diagram](image_url)

**Fig 4.3.2.1.1:** The metric learned (using Adaboost algorithm) must be a high positive for all the pairs marked as cian because they are for sure in the same cluster and a high negative for all the pairs marked as red because it is crucial to separate the normal samples (green crosses) from the abnormal samples (blue and black crosses) in absolutely different clusters.

As only one culture (the normal one) is the single culture paired with other cultures, we call *Single Pairing* to this kind of pairing. As the normal culture plays a special role in this topology, we call it *base-culture*.

From the pairing in Fig. 4.3.2.1.1, the algorithm learns a metric function from training samples using single pairing. The metric between samples paired as in figure 4.3.2.1.1 will not do any effort to cluster the abnormal samples from two different abnormal cultures: all the samples from different abnormal cultures are susceptible to be considered similar after applying them the metric function learned, because Adaboost cannot find any pair that states any difference between them (blues and black pairs in the figure).
4.3.2.2. Complete Pairing

In the test step all the samples are going to be paired. A priori we shouldn't know ground truth, so we should compute the metric between all the pairs of samples. We have the information to validate the metric function over the test samples but we need to use them as we would in a real case.

Indeed, if we use the metrics to cluster the samples and the Louvain Method group the ones with biggest similitude, different cultures with same distributions will be in different clusters because no metric able to group them will be computed (so is not possible that two test samples from different cultures converge in one group if there is no metric between them). So we use complete pairing in order to be able to merge in a group similar samples from different cultures.

Sometimes, to learn the metric function, we are also going to pair every training sample with every other. This is called Complete Pairing. This kind of pairing will be useful to obtain a good clustering between abnormal samples.

In Complete Pairing, two abnormal cultures would converge in the same cluster only if the distributions of both cultures are similar (similar type of distribution and similar mean and variance).

4.3.2.3. Multi-single Pairing

In the most complex cases, the need of using Single Pairing from an abnormal culture as base-culture arises. In other words, we can make Single Pairing using an abnormal culture as culture base: each sample from that abnormal culture will be paired with all the samples from other cultures.

This idea is really useful because sometimes we will use diverse Single Pairings, with a different base-culture each. We call them Multi-single Pairing.

The diagram in figure 4.3.2.3.1 shows Single Pairing.

![Diagram](image)

Fig 4.3.2.3.1: Single Pairing. Each double arrow represents pairs of samples. So, a double arrow inside a culture representation represents all the samples in that culture are paired between them. A double arrow between two different cultures \(D^p\) and \(D^q\) (\(p \neq q\)) represent pairs composed with one sample from each culture.

This way of pairing samples presented in Fig 4.3.2.3.1 implies to obtain a metric of dissimilitude between base-culture and the other cultures. We can learn a metric function from each of the next combinations and then use all the weak classifiers learned to obtain a more accurate metric to
cluster the abnormal samples.

Each of the four diagrams in figure 4.3.2.3.1.2 is a Single Pairing, but they have distinct base-cultures. To discern each case we are going to use Single(color) to denote we have done a Single pairing in such a way that the color culture is the base-culture. In the case of green samples we can use normal instead of green because it is equivalent. We can also refer to $D^i$ instead of using the color of the culture $D$ because it is also equivalent.

If a set of weak classifiers are learned for each single pairing, we can use all of them to obtain the metric of (dis)similitude.

So, for each set of weak classifiers obtained for each diagram we obtain a metric function.

$$m^s(x^1, x^2) = \sum_{t=1}^{T} \alpha_t^s \cdot c_t^s(x^1, x^2)$$

In other words, we have the next sets for each metric function $m^s$ obtained for the $s$-th single pairing done with the $s$-th base-culture used:

- $\alpha^s = \{\alpha_t^s\}_{t=1,...,T}$ set of factor of goodness for the $s$-th single pairing done with the $s$-th base-culture.
- $c^s = \{c_t^s\}_{t=1,...,T}$ Set of weak classifiers (functions) for the $s$-th single pairing done with the $s$-th base-culture.

Hence, we have as many sets as possible single pairings. As the metric is a lineal operator, we can obtain a set of metrics where each metric $m^s$ is composed by all the elements from $\alpha^s$ and $c^s$:

$$\{m^s(x^1, x^2)\}_{s=1,...,S} = \{\sum_{t=1}^{T} \alpha_t^s \cdot c_t^s(x^1, x^2)\}_{s=1,...,S}$$

With each learned metric function we can obtain a combination able to approximate the kind of metric described in Section 4.1. The linear combination we are talking about is:

$$M(x^1, x^2) = \sum_{s=1}^{S} m^s(x^1, x^2) = \sum_{s=1}^{S} \sum_{t=1}^{T} \alpha_t^s \cdot c_t^s(x^1, x^2)$$

Moreover, we can use twice the weak classifiers from Single(normal) to create a stronger dissimilitude between the normal and the abnormal samples. If we do this, we increase the number of metric functions in the set $\{m^s\}$ because we have added another element identical to the one corresponding to the metric obtained from single pairing with the normal culture as the base-culture, so the number of elements now would be $S+1$. In a case where we have one normal and three abnormal cultures, without using twice the metric obtained from the normal culture as the
base-culture is the next one:
\[
M = \sum_{s=1}^{S} m^s(x^1, x^2) = \sum_{s=1}^{S} \sum_{t=1}^{T} \alpha^s_t \cdot c^s_t(x^1, x^2)
\]

If we use twice the metric function learned from single pairing with the normal culture as the base-culture, the sum of metric functions is:
\[
M = m^1(x^1, x^2) + m^2(x^1, x^2) + m^3(x^1, x^2) + m^4(x^1, x^2)
\]
In other words:
\[
M = 2 \cdot m^1(x^1, x^2) + m^2(x^1, x^2) + m^3(x^1, x^2) + m^4(x^1, x^2)
\]

For that reason, we can generalize the expression of the combination of metric functions multiplying them by a scalar \(\beta^s\):
\[
M = \sum_{s=1}^{S} \beta^s \sum_{t=1}^{T} \alpha^s_t \cdot c^s_t(x^1, x^2)
\]

As we can see, we only need to find each \(\beta^s\) able to obtain the most accurate metric. This means we should change all the factor of goodness from the set as by the factors of goodness of this new set:
\[
\alpha^s = (\beta^s \alpha^s)_{t=1, \ldots, T}
\]

In the last sections we explore some Single(abnormal) used in multi-single pairing and some multi-single pairing itself.

### 4.3.3. Testing the similarity and checking the clustering

To test the Adaboost performance we make two different checks. Our Adaboost algorithm is used to obtain a metric function. We can know if a new pair is considered as similar or dissimilar using the \(\text{sign}\) function of that metric. In other words, if the similitude metric of the pair of samples is positive, the samples are considered as similar. If it is negative, the samples are considered as dissimilar.

From the testing set we know the ground truth, thus which culture each sample belongs to. For that reason we know the true similarity between samples:

- Two paired samples from the same culture are similar, so the metric should be positive.
- Two \(\text{paired}\) samples (if they are not paired they are not considered and no metric is computed) from different cultures are dissimilar so the metric should be negative.

If we call \(\text{label}\) to the real similitude (with the knowledge of which are the cultures the samples belong to) and \(\text{classification}\) to the result interpretation of the \(\text{sign}\) function of the metric, we can compare how many pairs of samples from the test set have the same value in both parameters (\(\text{label}\) and \(\text{classification}\)) and how many pairs of samples have different values. If the values of \(\text{label}\) and \(\text{classification}\) are the same for a pair of samples, it means that the metric is good for that pair and if the values are different, it means that the metric function of the Adaboost algorithm has failed. This kind of validation called \(\text{concordance}\) is useful for the first cases.

The other way to analyze the metrics consists on using a histogram of the metrics. We look for peaks of positive metrics between samples from the same culture (or different cultures but with similar distributions) and peaks of negative metrics (between samples from different cultures with different distributions). In fact, several histograms will be drawn, one for each pair of cultures. Each histogram shows how good the learned metric is at discriminating those two cultures. Those histograms have in the abscissa the bins of the metrics, and in the ordinate the number of samples from each bin.
Learning similarity metrics based on pairwise boosting

The Adaboost algorithm allows to obtain a metric function which computes a similarity metric between samples. That metric is used for the Louvain Method which clusters (based on that metric) the samples. We plot the result of the Louvain Method to visually conclude if the clustering goal has been achieved. We can visually know if two cultures are the same type of cultures or not, so we just need to check if the Louvain Method has visually produced a good clustering.

4.4. Tests

In this Section we show the results obtained from the different configurations presented in Section 4.2. At the beginning of each subsection the test parameters used are presented and also the mean and variance for each controlled Gaussian.

4.4.1 Two independent Gaussian cultures

<table>
<thead>
<tr>
<th>Test parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultures:</strong> 2</td>
</tr>
<tr>
<td><strong>Colors:</strong> green(normal), blue(abnormal)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature X</th>
<th>Feature Y</th>
<th>Feature X</th>
<th>Feature Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>variance</td>
<td>mean</td>
<td>variance</td>
</tr>
<tr>
<td>Green (normal)</td>
<td></td>
<td>1.2</td>
<td>5</td>
</tr>
<tr>
<td>Blue</td>
<td></td>
<td>-0.2</td>
<td>7</td>
</tr>
</tbody>
</table>

Fig. 4.4.1.1. shows the training (left) and testing sets (right).

![Training samples from 2 Gaussian cultures](image1)

![Test samples from 2 Gaussian cultures](image2)

Fig 4.4.1.1. Training (left) and testing (right) sets of samples with isolated cultures. The green crosses belong to the normal culture, and the blue crosses to an abnormal one.

We use the training set to learn the metric function using the Adaboost selection of weak classifiers. In the Figure 4.4.1.2 we can see the weak classifier selected by Adaboost over the training samples used for learning.
This example was run for ten iterations but the algorithm stopped after the first one because the result obtained with that weak classifier was perfect. If only one classifier is needed to obtain the perfect result, we call it strong classifier. All the red circles are the representation of centers from weak classifiers (from Ellipse type) and the one in blue is the selected one.

We can clearly see that the green samples are similar between them because they are in the same region (inside the circle). The blue samples are also similar between them because they are outside the circle (same region). And the pairs with one normal sample and one abnormal sample are dissimilar because each sample from the pair is in a different region.

We have applied the same strong classifier to the test set (see Fig. 4.4.1.3) and the classifier classifies correctly the 100% of the pairs.
If we compute the histogram of the metric between each pair, only two values appear (as we can observe in Fig. 4.4.1.4) because the metric is computed for only one iteration as:

\[
m(x_i^1, x_i^2) = \sum_{t=1}^{T} \alpha_t \cdot c_t(x_i^1, x_i^2)
\]

for a pair of samples \(x^1\) and \(x^2\) and for only one weak classifier \(c_1\) (in this case for \(T=1\) we only have \(c_1\)) and for a positive value of \(\alpha_1=1\).

Imagine a normal sample from the test set outside the ellipse: it would mean an error. Increasing the number of weak classifiers will not solve that problem because the metric from a point of view of the training samples is perfect, but there still can be random test set samples misclassified. This means we need to increase the number of training samples at the learning step: that way we minimize the possibility of random unexpected points in previously empty regions.

### 4.4.2 Two intermixed Gaussian cultures

<table>
<thead>
<tr>
<th>Test parameters</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultures:</strong></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td><strong>Pairing:</strong></td>
<td>single</td>
<td></td>
</tr>
<tr>
<td><strong>Base-culture:</strong></td>
<td>green(normal)</td>
<td></td>
</tr>
<tr>
<td><strong>Colors:</strong></td>
<td>green(normal), blue(abnormal)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mean</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature X</td>
<td>Feature Y</td>
</tr>
<tr>
<td>Green (normal)</td>
<td>1.2</td>
</tr>
<tr>
<td>Blue</td>
<td>-0.2</td>
</tr>
</tbody>
</table>

If we increase the variance of the mentioned cultures we observe that no strong classifier has been found, i.e. there is no weak classifier capable to correctly classify all the pairs as similar and dissimilar only by itself. The training and test sets of the new configuration are shown in Fig. 4.4.2.1.
Fig. 4.4.2.1: Training (left) and testing (right) sets of samples with overlapped cultures. The green crosses belong to the normal culture and the blue crosses to an abnormal one.

Now the normal and abnormal samples are a bit mixed. Thus, the algorithm will not stop at the first iteration. In each iteration, it selects another weak classifier that minimizes the error obtained by the previous ones. In Figure 4.4.2.2 we can see the weak classifiers learned from the training set and applied to the test set.

Fig. 4.4.2.2: Learned weak classifiers from the training set (left) which each one splits the features space in two parts. The set of weak classifiers are applied to the test set (right) in order to obtain a metric between the samples from the test set.

If we compute which pairs have a metric with a correct sign (correct similarity), the result is not as good as for isolated cultures:

right metrics = 92.23%
wrong metrics = 7.77%

However, it is very good as it is shown in the histogram of Fig. 4.4.2.3.
As we can see in the Fig. 4.4.2.2, a lot of normal samples (green) are in the same intersected region of the weak classifiers. Almost every abnormal sample is in another intersected region of weak classifiers.

That is why we see in 4.4.2.3:

- high similarity between greens
- high similarity between blues
- high dissimilarity between green and blue samples.

If we look carefully at figure 4.4.2.2, and compare the sign of the metric with the ground truth containing the similitude between pairs, we can conclude that some samples from the histogram are in the wrong side. But as far as we can observe two kind of mountains in the histogram, we can say the Adaboost algorithm is working pretty well.

The Louvain Method can cluster the samples by the learned metric. For the test set we know the culture each sample belongs to, but in a real case we are not going to be able to know it. That is why we need to use the Louvain Method and a metric of similitude between all the samples (obtained by the selected classifiers by the adaboost algorithm). In Fig. 4.4.2.4 the comparison between the original cultures from the test set and the clusters done by the Louvain Method is shown.

Fig 4.4.2.4: Clustering in normal and abnormal samples.
Thus, it is clear that some samples have been clustered in the wrong cluster. For example the two samples isolated in the upper right corner. This can be solved if we use a bigger training set with more samples covering all the possible space as we have proposed in the previous case (Fig 4.4.2.5).

Fig 4.4.2.5: New training set of samples with bigger amount of them. 500 samples each culture.

Now the weak classifiers obtained are the ones from Fig. 4.4.2.6.

Fig 4.4.2.6 Weak classifiers obtained for a bigger set of training samples.

When we compute which pairs have a metric with a correct sign (correct similarity) we can see that, although the result is not as good as for isolated cultures, we have improved the previous results:

right metrics = 94.18%
wrong metrics = 5.82%

We have only used 100 samples for each culture in the next training examples because the computational time with 500 samples per culture for 4 cultures is too expensive in terms of computation time and the difference of error is not so big.

We can conclude: the more training data we have, the better performance has the system.
4.4.3 Four Gaussian cultures with single pairing

| Test parameters |
|-----------------|-----------------|-----------------|
| **Cultures:**   | **Pairing:**    | **Base-culture:** |
| 4               | single          | green(normal)   |
| **Colors:**     |                 | green(normal), blue(abnormal), red(abnormal), black(abnormal) |

<table>
<thead>
<tr>
<th>mean</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature X</td>
<td>Feature Y</td>
</tr>
<tr>
<td>Green (normal)</td>
<td>1.2</td>
</tr>
<tr>
<td>Blue</td>
<td>-0.1</td>
</tr>
<tr>
<td>Black</td>
<td>2</td>
</tr>
<tr>
<td>Red</td>
<td>-0.1</td>
</tr>
</tbody>
</table>

The next example illustrates a case which should cluster into three clusters: one normal and two abnormal (red and blue have the same distribution and are therefore indistinguishable).

The next example illustrates a case which should cluster into three clusters: one normal and two abnormal (red and blue have the same distribution and are therefore indistinguishable).

First we single-pair all the training samples. When computing the metric function only one classifier (strong classifier) is selected, because normal samples are isolated (see Fig. 4.4.3.2).
Fig 4.4.3.2: Strong classifier learned by Adaboost using the training set (left) and applied to the test set (right).

We can see that this time, there is no error. All the normal samples from the test set are inside the ellipse and all the abnormal (not paired between different cultures) outside. So from the adaboost performance point of view this result is perfect:

right metrics = 100%
wrong metrics = 0.00%

Only one classifier has been used. So we have again a strong classifier. We can conclude with the example of two Gaussian that: a strong classifier will be obtained when the basic culture (in these cases the normal one) used in the single pairing is isolated.

And as long as we have only one classifier, the histogram will show only two values for a metric between samples paired as single(green) (see Fig. 4.4.3.3).

These results are proof that our implementation of Adaboost algorithm has been good and validates our weak classifiers. However, as the set is the test set we must do a complete pairing between test samples.

The learned metric function using single pairing and applied to the test samples computes also only two metric values as we can see in the histogram of Fig 4.4.3.4.
Hence, in the histogram there are clearly two different values of the metric and clearly only one dissimilarity measure so it is logical that the Louvain Method groups the samples in only two clusters as seen in Fig 4.4.3.5.

When single pairing, samples can be grouped into only two groups: one normal and one abnormal. It is necessary to emphasize this conclusion as this was the first goal we were pursuing.

The second goal was to be able to cluster each abnormal sample: it has not been reached.

We can not obtain different abnormal clusters because in the single pairing the weak classifiers only try to create the dissimilarity between the normal culture and the others and keep all the normal samples in the same region.

There is only one final abnormal cluster containing all the abnormal samples because, when single pairing, there are not two abnormal samples paired as dissimilar. No dissimilar pairing means there is no dissimilar metric. And no dissimilar metric means no effort to clusterize those samples separately.

For that reason, in the next section a Complete Pairing is performed on the same actual set.
4.4.4 Four Gaussian cultures with complete pairing

<table>
<thead>
<tr>
<th>Test parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultures:</strong> 4</td>
</tr>
<tr>
<td><strong>Base-culture:</strong> [none]</td>
</tr>
</tbody>
</table>

With complete pairing on the same sets from Section 4.4.3 (thus the ones of Fig 4.4.3.1), the learned weak classifiers try to separate all the different abnormal cultures. Dissimilarity between cultures forbids a unique strong classifier able to separate absolutely all the cultures. In Fig. 4.4.4.1 we can see the weak classifiers selected, which try to separate each culture of all other.

![Weak classifiers](image1)

![Weak classifiers](image2)

Fig. 4.4.4.1: Weak classifiers learned from the training samples (left) using Adaboost and applied to the test samples (right).

When having two identical cultures (blue-red), a lot of errors arise because Adaboost cannot learn a difference between them. We can observe the normal samples (green) are in the same region and the black ones too. Adaboost tries to separate the red from the blue samples, but it is useless because of their identical distribution. Maybe a test red sample is where a training blue sample was, and if in the exact same region there is a test blue sample the metric computed between these test samples (blue and red) will be positive and computed as similar.

Lets analyse the metric between the test samples using the learned weak classifiers. We analyse the histograms shown in Figure 4.4.3.7 of the metric between the different kind of pairs.

![Histograms](image3)

Fig. 4.4.4.2: Histograms of the metric between test samples. Histograms (1,1), (2,2), (3,3) and (4,4) correspond to samples from the same culture. The rest corresponds to the metric between dissimilar cultures. Cultures 1, 2, 3 and 4 correspond to green, blue, black and red samples respectively.
The behaviour of the weak classifiers applied to each kind of pair is explained in the next table with the correspondence between the number of culture and the colour of the samples of the pairs:

<table>
<thead>
<tr>
<th>histogram</th>
<th>colors</th>
<th>metric</th>
<th>commentaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>Green-green</td>
<td>good</td>
<td>Only one value: all normal samples almost in the same intersected region</td>
</tr>
<tr>
<td>3,3</td>
<td>Black-black</td>
<td>good</td>
<td>Only one value: all the black samples almost in the same intersected region</td>
</tr>
<tr>
<td>1,3</td>
<td>Green-black</td>
<td>good</td>
<td>Pairs by one sample from each culture, metric for each pair always the same.</td>
</tr>
<tr>
<td>1,2</td>
<td>Green-blue</td>
<td>good</td>
<td>Both cases are the same, blue and red distributions have same mean and variance. Metrics indicate a clear dissimilarity</td>
</tr>
<tr>
<td>1,4</td>
<td>Green-red</td>
<td>good</td>
<td>Both cases are the same, blue and red distributions have same mean and variance. Metrics are mostly dissimilar.</td>
</tr>
<tr>
<td>2,3</td>
<td>Blue-black</td>
<td>good</td>
<td>Similar: blue and red samples have identical distributions. A blue sample could be where a red one is or vice versa. No weak classifiers are able to create a dissimillitude between samples with similar distributions.</td>
</tr>
<tr>
<td>3,4</td>
<td>Black-red</td>
<td>good</td>
<td>Better cases are the same, blue and red distributions have same mean and variance. Metrics are mostly dissimilar.</td>
</tr>
<tr>
<td>2,2</td>
<td>Blue-blue</td>
<td>good</td>
<td>Better cases are the same, blue and red distributions have same mean and variance. Metrics are mostly dissimilar.</td>
</tr>
<tr>
<td>4,4</td>
<td>Red-red</td>
<td>good</td>
<td>Better cases are the same, blue and red distributions have same mean and variance. Metrics are mostly dissimilar.</td>
</tr>
<tr>
<td>2,4</td>
<td>Blue-red</td>
<td>good</td>
<td>Better cases are the same, blue and red distributions have same mean and variance. Metrics are mostly dissimilar.</td>
</tr>
</tbody>
</table>

If we compute the classification of similarity and dissimilarity of Adaboost, we can see that it does not classify correctly because almost a 20% consist on errors from the blue and red abnormal cultures:

- right metrics = 80.05%
- wrong metrics = 19.95%

1 over 5 pairs of samples are misclassified. However, we admit these errors as we want to include all the blue and red samples in the same cluster. We can observe the result of the clustering in Fig 4.4.4.3.

So it works all right for this kind of configuration.

### 4.4.5 Four intermixed Gaussian cultures

<table>
<thead>
<tr>
<th>Test parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultures:</strong></td>
</tr>
<tr>
<td><strong>Pairing:</strong></td>
</tr>
<tr>
<td><strong>Base-culture:</strong></td>
</tr>
<tr>
<td><strong>Colors:</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td><strong>Green (normal)</strong></td>
</tr>
<tr>
<td><strong>Blue</strong></td>
</tr>
<tr>
<td><strong>Black</strong></td>
</tr>
<tr>
<td><strong>Red</strong></td>
</tr>
</tbody>
</table>

We must always assure that there will be a cluster composed of only the normal culture separated from the abnormal cultures (this is our first objective). We are going to show a case where we cannot assure this rule (see Fig 4.4.5.1).

Fig. 4.4.5.1: Training (left) and test (right) set of samples from different cultures overlapped. The blue and red cultures have the same mean and variance.

When having intermixed samples there is always some clustering fuzziness because the frontiers between cultures are not very clean. In the figures we can see an empty zone around (2,3.5) in the learning set that could easily produce a wrong metric in classifying the black samples from the test set occupying that same zone. Also, the two black crosses about (0.5, 2) and (1, 0.5) in the training set would probably produce a misclassification of the green samples in the test set in the same zone. In general, randomness of such samples involves an indeterminacy of metrics. As usual in statistics, the more samples, the more accurate is the metrics. More samples mean less points without data and more information for each zone. For example, more samples around (0.5, 2) would show the supremacy of one colour and, therefore, would improve the metrics and the classification of the test set samples. However, the computational time is too expensive for us and we have only used 100 samples in each culture in the training sets.

In Fig 4.4.5.2, the weak classifiers of the current case are shown.
Of course, the error obtained with this configuration will be bigger than the one obtained before (with the previous configuration of 4 cultures with low variances) because of the overlapping:

<table>
<thead>
<tr>
<th>Metrics from samples belonging to cultures with low variances</th>
<th>Metrics from samples belonging to cultures with high variances</th>
</tr>
</thead>
<tbody>
<tr>
<td>right metrics = 80.05%</td>
<td>right metrics = 70.11%</td>
</tr>
<tr>
<td>wrong metrics = 19.95%</td>
<td>wrong metrics = 29.89%</td>
</tr>
</tbody>
</table>

The histograms of metrics between samples of different cultures and also of the same culture are given in Fig 4.4.5.3.

Fig. 4.4.5.3: Histograms of the metric between test samples. Histograms (1,1), (2,2), (3,3) and (4,4) correspond to samples from the same culture. The rest corresponds to the metric between dissimilar cultures. Cultures 1, 2, 3 and 4 correspond to green, blue, black and red samples respectively.

The metrics of each histogram is discussed in the next table:
Finally, if we apply these metrics between all the test samples as input to the Louvain Method we can see the result presented in Fig 4.4.5.4.

<table>
<thead>
<tr>
<th>histogram</th>
<th>colors</th>
<th>metric</th>
<th>commentaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>Green-green</td>
<td>good</td>
<td>There is some error and some of the metrics are considered dissimilar but they are mostly similar. Errors are produced in overlapped areas.</td>
</tr>
<tr>
<td>3,3</td>
<td>Black-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,3</td>
<td>Green-black</td>
<td>wrong</td>
<td>They have positive metrics! They should be negative!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Adaboost has been focused in separating blue and red samples, the objective of discriminating normal samples from and any abnormal sample has been relegated.</td>
</tr>
<tr>
<td>1,2</td>
<td>Green-blue</td>
<td></td>
<td>Dissimilar (negative) metrics.</td>
</tr>
<tr>
<td>1,4</td>
<td>Green-red</td>
<td></td>
<td>As the weak classifiers are selected in order to solve all the time the errors produced between blue and red samples, they are not so focused in obtaining the dissimilitude between the blue (or red) samples and the green or the black ones. For that reason some of the metrics we can see in these histograms are positives.</td>
</tr>
<tr>
<td>2,3</td>
<td>Blue-black</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>3,4</td>
<td>Black-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,2</td>
<td>Blue-blue</td>
<td></td>
<td>These histograms look similar because blue and red samples have identical distributions. The metrics are mostly positive because the position of a blue sample from a red one is totally random. A blue sample could be where a red one is and vice versa.</td>
</tr>
<tr>
<td>4,4</td>
<td>Red-red</td>
<td>good</td>
<td>No weak classifiers are able to create a dissimilitude between samples with similar distributions. Some errors must be produced also because of the overlapped areas.</td>
</tr>
<tr>
<td>2,4</td>
<td>Blue-red</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4.4.5.4: Result of clustering 4 cultures

As we have said, some positive similitude has been found between the green and black samples (colors from the left of Fig. 4.4.5.4) of the test set. The Adaboost algorithm has been focused in solving the errors between the blue an red samples. That was a good idea, to cluster them in a single cluster. However our first priority is to obtain a cluster for only the normal (green) samples.

The solution to solve this problem consists in duplicating each normal sample from the test set. That means each normal sample will be paired with every other sample the twice. We have tripled the number of normal samples. Now Adaboost is also focused on the priority of having the normal samples in one cluster without any abnormal culture. Visually the plot of the samples will not change, but we can see how other weak classifiers have been selected to preserve the normal samples in one region all together (see Fig 4.4.5.5).
We can see here some green crosses from the test set in a region with black crosses from the training set. However it seems that the most part of normal samples are strongly separated from any other. The histograms are presented in Fig 4.4.5.6.

The next table describes the histograms:

<table>
<thead>
<tr>
<th>histogram</th>
<th>colors</th>
<th>metric</th>
<th>commentaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Green-green</td>
<td></td>
<td>These metrics indicate the green samples are only similar between samples from the same green (normal) culture.</td>
</tr>
<tr>
<td>1.2</td>
<td>Green-blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Green-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Green-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>Blue-black</td>
<td></td>
<td>Many pairs have negative metrics. As the focus now is to separate the green culture from the others, the dissimilitude between black and blue or red has decreased.</td>
</tr>
<tr>
<td>3.4</td>
<td>Black-red</td>
<td></td>
<td>Similar pairs in different peaks because the regions used to compute the metrics are not the same.</td>
</tr>
<tr>
<td>3.3</td>
<td>Black-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Blue-blue</td>
<td></td>
<td>Still similar because is not possible to obtain dissimilitude between samples from the same distribution</td>
</tr>
<tr>
<td>4.4</td>
<td>Red-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>Blue-red</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Here it makes no sense to compute the error because we know there are some errors from the metric
function and we want them in order to cluster the blue and red ones in the same cluster. Considering as errors all the similarities between blue and red samples, the total errors would be very high, but those errors are not real: they would be errors only if red and blue samples had to be in two disjoin clusters. As red and blue samples must be in the same cluster (for there is no real difference between them), the mentioned errors cannot be computed as such. Hence, if we apply the Louvain Method with these new metrics we obtain the clusters we can see in Fig 4.4.5.7.

![Fig. 4.4.5.7: Clustering of 4 cultures in 3 clusters](image)

Some error in the overlapped zones is produced. Therefore, from here, the question is how much we must duplicate the normal samples.

We can also observe the pair of normal samples in a cluster of abnormal samples (we have explained the only way to avoid these errors is by using more training samples).

So finally we have achieved our second goal: clustering the abnormal samples.

In the next sections some random configurations are used to test the Adaboost algorithm and it is presented that not all the abnormal cultures are correctly clustered because the Louvain Method merge the samples with biggest similitude instead of separating the ones with the biggest dissimilitude.

The next step should be to compare what is even a better solution: Pair all the samples giving more pairs to the normal samples or an other option of combining the sets of weak classifiers of different kind of single pairings.

### 4.4.6 Counterexample of Complete Pairing

<table>
<thead>
<tr>
<th>Test parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultures:</strong></td>
</tr>
<tr>
<td><strong>Pairing:</strong></td>
</tr>
<tr>
<td><strong>Base-culture:</strong></td>
</tr>
<tr>
<td><strong>Colors:</strong></td>
</tr>
</tbody>
</table>

The configuration of Fig 4.4.6.1 is proposed in order to discard the idea of the complete pairing. Even the replicas of the normal samples are not useful because the clustering of the abnormal samples becomes wrong. This example has been generated with random mean and variances.
Fig 4.4.6.1: Two different sets of samples from a configuration with four random Gaussian cultures.

If we pair all the samples, green and red cultures merge to the same cluster. In fact there is a high metric of similitude between them. We must triplicate each normal (green) sample to obtain a metric function able to cluster only the normal culture in its own cluster. But just duplicating the normal samples, the ones from the black and blue cultures are grouped into the same cluster when visually they are clearly different. We can see in Fig 4.4.6.2 the result of the Louvain Method when the normal samples are not duplicated, when they are duplicated, or when they are triplicated.

Fig 4.4.6.2: Results of the Louvain Method from metrics learned from all the samples paired with the same initial weight to the normal samples (a), with the normal samples duplicated (b) so more weight to each normal samples and triplicated (c).

Not only the two upper cultures are badly clustered, but also some samples from the lower culture are assigned to the cluster of normal samples.

We can discard Complete Pairing because if we try to solve the objective of obtaining the normal samples in a different cluster from the abnormal samples then we may cluster in a wrong way the abnormal samples.

A more complex method is needed to learn a fully useful metric.

For that reason in the next sections we test the results obtained from a multi-single pairing.

### 4.4.7 An unsolvable case

<table>
<thead>
<tr>
<th>Test parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultures:</strong> 4</td>
</tr>
<tr>
<td><strong>Pairing:</strong> single</td>
</tr>
<tr>
<td><strong>Base-culture:</strong> blue</td>
</tr>
<tr>
<td><strong>Colors:</strong> green(normal), blue(abnormal), red(abnormal), black(abnormal)</td>
</tr>
</tbody>
</table>

The importance of the Multi-single pairing consist in the impossibility of discriminate two cultures
with the same distribution. For that reason we analyse what happens when we try to create a
dissimilitude between one abnormal culture and another with the same distribution.

If a couple of cultures $D^j$ and $D^k$ have similar features, the weak classifiers learned from the pairing
type $Single(D^j)$ or $Single(D^k)$ can not separate both cultures as dissimilar. For example, we have
tried to obtain a dissimilitude from the blue culture and the others seen in Fig 4.4.7.1, but as we can
see below it is not possible to obtain a dissimilitude between blue and red cultures.

Fig 4.4.7.1: Four gaussian cultures with red and blue samples belonging to cultures with the same distribution.

When pairing $Single(blue)$, we obtain the next weak classifiers over the training samples and
applied to the test samples (see Fig 4.4.7.2).

Fig 4.4.7.2: Weak classifiers learned from the new pair of the training set and applied to the test set

From these weak classifiers we can see the effort to separate the blue samples from the green and
black. Those weak classifiers also try to separate in different regions the blue from the red samples,
but of course this is not possible. If we look at the histograms of the metrics in figure 4.4.7.3 we can
see that the metrics are as expected.
Fig 4.4.7.3: Histograms of the similarity metric between pairs. Cultures 1, 2, 3 and 4 correspond to green, blue, black and red samples respectively.

<table>
<thead>
<tr>
<th>histogram</th>
<th>colors</th>
<th>metric</th>
<th>commentaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Blue-blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>Red-red</td>
<td>good</td>
<td>The blue and red samples have the same distribution so the histograms are similar between them. The pairs are mostly dissimilar.</td>
</tr>
<tr>
<td>2.4</td>
<td>Blue-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Green-blue</td>
<td></td>
<td>The number of similar pairs is smaller or the number of dissimilar pairs is bigger, exactly as the weak classifiers try to be learned.</td>
</tr>
<tr>
<td>2.3</td>
<td>Blue-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Green-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Black-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Green-green</td>
<td></td>
<td>As we are focused in creating a dissimility between blue and red and trying to make similar the Green-green and black-black, grouping, all the green and black samples in the same region is enough, so the metric is similar between any of these samples.</td>
</tr>
<tr>
<td>1.3</td>
<td>Green-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Black-black</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From these results we can conclude it is not possible to obtain a dissimilitude between similar distributions. For that reason, the input data must contain features which discriminate enough the normal samples from the rest.

**4.4.8 Improving with Multi-single Pairing**

<table>
<thead>
<tr>
<th>Test parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cultures:</strong></td>
</tr>
<tr>
<td><strong>Pairing:</strong></td>
</tr>
<tr>
<td><strong>Base-culture:</strong></td>
</tr>
<tr>
<td><strong>Colors:</strong></td>
</tr>
</tbody>
</table>

We start from the same configuration of the previous section. If we use each sets of weak classifiers obtained from each possible single pairing [Single(green), Single(blue), Single(black), Single(red)] we have four times the initial number of classifiers. Each one keep its factor $\alpha$ of goodness. The weak classifiers obtained from all the combinations, which try to create a dissimilitude between one culture and the others, are the ones in figure 4.4.8.1.
In the figure 4.4.8.1 we can observe that most of the normal (green) samples are in the same region. Black ones are well separated too. If we look at the metrics in figure 4.4.8.2 we can see they are pretty good.

Each pair of samples from the same culture have a high metric of similitude. It is also the case of the 2-4 blue-red pairs because both samples belong to a distribution with the same mean and variance. The other histograms have a high number of metrics of dissimilitude between pairs and also a smaller number of pairs considered as dissimilar.

Hence it seems this is a good strategy to combine the weak classifiers selected by each Single(color) pairing. If we use these metrics as input for the Louvain Method we get the clustering shown in Fig 4.4.8.3.
We can see that with the metric function composed by all the weak classifiers of each combination of pairs we obtain a really good clustering.

### 4.4.9. Multi-single confirmed

| Test parameters |
|-----------------|-----------------|-----------------
| **Cultures:**   | 4               | **Base-culture:** [everyone] |
| **Pairing:**    | multi-single    | **Colors:**     |
| **Colors:**     | green(normal), blue(abnormal), red(abnormal), black(abnormal) |

Now we are going to try multi-single pairing against the previously seen set from Section 4.4.6 where the metric obtained with a complete pairing failed:

However, if we use the combination of weak classifiers obtained from different kind of *single pairings*, each one trying to create a dissimilitude between one culture and the others, then results are better. We could even think in an analog way and replicate the number of weak classifiers used in the single pairing with the normal as the culture-base. In next sections we consider a more subtle option in order to emphasize the dissimilitude between the normal samples and the abnormal ones. In Figure 4.4.9.2 the result of the Louvain Method from the metrics obtained by combining each set
of weak classifiers is shown (left). In the right diagram we count twice the weak classifiers obtained trying to discern between normal and abnormal samples.

Fig 4.4.9.2: Results of the Louvain Method from metrics learned from different kind of pairings. The same number of weak classifiers for each kind of pairing is used in the left. The set of weak classifiers used for the pairs to discern between normal and abnormal samples is used twice in the right.

We can see the case of duplicating the weak classifiers used to create a dissimilarity between normal and abnormal samples produce a softer change than duplicating the normal samples for a complete pairing.

As it seems this strategy works, we are going to discuss the examples applying it. First we are going to use the same case. In figure 4.4.9.3 the learned weak classifiers from the different kind of pairing able to create a dissimilarity are shown.

Fig 4.4.9.3: Weak classifiers learned from different pairings of training samples
We can observe that each set of weak classifiers has its own goal:

- The first set tries to create a dissimilitude between the green (normal) samples and the others. We can see the black and blue samples in the same region; so the metric between a blue and a black sample is as similar as the one from two black samples (or blue ones).
- The second set tries to obtain a metric with high dissimilitude between the blue samples and the others.
- The third set is focused in obtaining a dissimilitude between black and any other culture.
- The fourth set obtains a total dissimilitude between the red culture and any other culture with just a single strong classifier.

After using all the weak classifiers ensembles from each set in a single set of classifiers we can apply them to the test set of samples as in Fig 4.4.9.4.

Fig 4.4.9.4: All the weak classifiers learned from the training set and applied to the test set.

With so many weak classifiers with different objectives each one, it is difficult to see if the different colors are in very different regions ones from others. For that reason, in Fig 4.4.9.5 the histograms of the metrics of similitude between all the samples are shown.

Fig 4.4.9.5: Histograms of similarity metrics between samples. Cultures 1, 2, 3 and 4 correspond to green, blue, black and red samples respectively.
From these histograms we can deduce, that this metric function seems perfect to cluster the samples with the Louvain Method as we have seen in figure 4.4.9.2 (at the left of it).

Using twice or three times more those weak classifiers from Single(normal) used to discriminate the normal from the abnormal samples, we can obtain a bigger metric of dissimilitude between the normal and abnormal cultures. However, as it is described in Section 4.3.2.3, this is the same as multiply each factor of goodness by a scalar depending the single-pair contribution used. So we can obtain metrics as the presented in Section 4.3.2.3:

\[
M(x^1, x^2) = \sum_{s=1}^{S} \beta_s^5 m^s(x^1, x^2) = \sum_{s=1}^{S} \sum_{t=1}^{T} (\beta_s \cdot \alpha_t) \cdot c_t(x^1, x^2)
\]

4.4.10. Non-gaussian distribution

<table>
<thead>
<tr>
<th>histogram</th>
<th>colors</th>
<th>metric</th>
<th>commentaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>Green-green</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,2</td>
<td>Blue-blue</td>
<td>good</td>
<td>They have a high number of pairs with a high metric of similitude. So the metrics between pairs from the same culture are very good.</td>
</tr>
<tr>
<td>3,3</td>
<td>Black-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4,4</td>
<td>Red-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,2</td>
<td>Green-blue</td>
<td>regular</td>
<td>Some pairs have a high similitude but it has increase the number of pairs with a metric of dissimilitude</td>
</tr>
<tr>
<td>1,3</td>
<td>Green-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,3</td>
<td>Blue-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3,4</td>
<td>Black-red</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These histograms contain the worst metrics. However, the number of pairs with high similitude is not so big. If we compare with any of the histograms from pairs belonging to the same cultures, the biggest peak in those histograms is near 40,000 pairs while in these histograms is near 10,000.

Moreover, a lot of pairs have a smaller metric of similitude than the pairs from identical cultures. The Louvain method uses the most similar metric to group a pair of samples in the same cluster so these metrics are going to work for us.

From these histograms we can deduce, that this metric function seems perfect to cluster the samples with the Louvain Method as we have seen in figure 4.4.9.2 (at the left of it).

Using twice or three times more those weak classifiers from Single(normal) used to discriminate the normal from the abnormal samples, we can obtain a bigger metric of dissimilitude between the normal and abnormal cultures. However, as it is described in Section 4.3.2.3, this is the same as multiply each factor of goodness by a scalar depending the single-pair contribution used. So we can obtain metrics as the presented in Section 4.3.2.3:

\[
M(x^1, x^2) = \sum_{s=1}^{S} \beta_s^5 m^s(x^1, x^2) = \sum_{s=1}^{S} \sum_{t=1}^{T} (\beta_s \cdot \alpha_t) \cdot c_t(x^1, x^2)
\]

4.4.10. Non-gaussian distribution

<table>
<thead>
<tr>
<th>Test parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cultures: 4</td>
</tr>
<tr>
<td>Colors: green(normal), blue(abnormal), red(abnormal), black(abnormal), purple(abnormal)</td>
</tr>
</tbody>
</table>

Fig 4.4.10.1: Two different sets of samples from a configuration with three random Gaussian cultures and two more distributions with Gaussian radius and uniform angle.

We have used another kind of distributions in order to generalize a bit more the examples.

For each single pairing we have obtained the best weak classifiers. In Figure 4.4.10.2 the weak
Learning similarity metrics based on pairwise boosting

classifiers obtained to discriminate the normal culture from the abnormal is shown.

Fig 4.4.10.2: Weak classifiers learned to discriminate the normal culture using Single(green) pairing

We can observe that we have used a bigger grid of weak classifiers because of the big distance between the black samples and the others. The weak classifiers try to get the green samples in just one region but that seems hard because of the red and purple samples. The other sets of weak classifiers are the four sets in figure 4.4.10.3.

Fig 4.4.10.3: Weak classifiers learned to discriminate each abnormal culture: The blue culture (upper left), the black culture (upper right), the red culture (down left) and the purple culture (down right).
So we have $S$ sets of weak classifiers. We have $S=5$ in the example.
Each of these sets of weak classifiers compose a metric $m^s$ function as the following for a pair of samples $(x^1, x^2)$:

$$m^s(x^1, x^2) = \sum_{t=1}^{T} \alpha_t \cdot c_t(x^1, x^2) \quad s = 1, \ldots, S$$

Thus, we can considerate, for each metric, a set of weak classifiers learned such as:

$$c^s = \{c_t(x, \cdot)\}_{t=1}^{T} \text{ for } T \text{ as the number of weak classifiers that compose the metric function.}$$

And we can also considerate for each set of weak classifiers, the corresponding set of goodness factors:

$$\alpha^s = \{\alpha_t\}_{t=1}^{T}$$

With this notation, we consider that if a set $\alpha^s$ is multiplied by a scalar $\beta^s$ all the elements inside the set are multiplied by the same scalar:

$$\alpha^{s'} = [\beta^s \alpha^s]_{t=1}^{T}$$

If we assign the same scalar $\beta^s = 1$ for each goodness factor set, the metrics obtained are like the ones in figure 4.4.10.4.

Fig 4.4.10.4: Histograms of the metrics learned by the weak classifiers (with the original contribution for each one of them).

First of all we can see the ranges of similitude are very high. We can see metrics of similitude over 50 and metrics of dissimilitude under -50. However, we only observe metrics of dissimilitude between the black culture (culture 3) and any other. Particularly that means there is no dissimilitude match between the normal/green culture (culture 1) with any other, and this is our prioritized objective.

If we try to assign different weights in an exhaustive way to the weak classifiers for the normal samples and to the ones for the abnormal samples we obtain different results. In the next case we multiply by 200 the factor of goodness from the classifier set of the normal culture and by 7 each goodness factor of each set of weak classifiers dedicated to discriminate each of the abnormal cultures. The resulting histograms are shown in Fig 4.4.10.5.
The next table summarize the histograms:

<table>
<thead>
<tr>
<th>histogram</th>
<th>colors</th>
<th>Metrics</th>
<th>commentaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Green-green</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Blue-blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Black-black</td>
<td>good</td>
<td>They are all similar.</td>
</tr>
<tr>
<td>4.4</td>
<td>Red-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Purple-purple</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Green-blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Green-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>Green-purple</td>
<td>wrong</td>
<td>All these histograms have a lot of pairs with a metric which classifies the pairs as similars when they are dissimilar pairs. We can say immediately the metric obtained is not good enough because of these cases.</td>
</tr>
<tr>
<td>2.4</td>
<td>Blue-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Blue-purple</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>Red-purple</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Green-black</td>
<td>regular</td>
<td>The metric is well distributed, they are as much similar as dissimilar metrics aprox so the dissimilar ones are the good ones.</td>
</tr>
<tr>
<td>2.3</td>
<td>Blue-black</td>
<td>regular</td>
<td>Similar metric but for the majority of pairs a very low metric.</td>
</tr>
<tr>
<td>3.4</td>
<td>Black-red</td>
<td>regular</td>
<td>The metrics are distributed but a lot of them are negative as we want.</td>
</tr>
<tr>
<td>3.5</td>
<td>Black-purple</td>
<td>good</td>
<td>There is a lot of negative metrics as we want.</td>
</tr>
</tbody>
</table>

After some tries, the best option that we have been able to find in this case, is to use only the weak classifiers used to discriminate the normal culture. That means to multiply by 1 the factor of goodness of that weak classifiers and by 0 to the factors of the ones used to discriminate each other culture. So, in practice it is as applying the weak classifiers in Fig 4.4.10.6 from the training samples (paired appropriately) to the test samples.
The resulting histograms are given in Fig 4.4.10.7.

Fig 4.4.10.7: Histograms of the metrics learned by the weak classifiers from the normal culture.

And they are described in the next table:
The Louvain Method uses the highest similitude but as some metrics are wrong, it is not able to cluster the samples in a good way. We would need a learning step in order to assign a weight (the scalar $\beta$) to each set of weak classifiers. The result of the clustering is the one shown in Fig 4.4.10.8.

The metrics mark these pairs as similar so they are wrong.

Looking at these histograms it seems the black culture is the more dissimilar culture. The metrics are pretty dissimilar but also positive and near to 0, so some of them are wrong but they are globally right.

<table>
<thead>
<tr>
<th>histogram</th>
<th>colors</th>
<th>metrics</th>
<th>commentaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>Green-green</td>
<td>good</td>
<td>They are all similar as we want.</td>
</tr>
<tr>
<td>2,2</td>
<td>Blue-blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3,3</td>
<td>Black-black</td>
<td>good</td>
<td>A lot of pairs have positive metrics, however the biggest peaks belongs to a negative metric. What should be enough for the posterior clustering</td>
</tr>
<tr>
<td>4,4</td>
<td>Red-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5,5</td>
<td>Purple-purple</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,2</td>
<td>Green-blue</td>
<td>good</td>
<td>These metrics have a big negative value as we want in order to separate the green (normal) from the black samples.</td>
</tr>
<tr>
<td>1,4</td>
<td>Green-red</td>
<td>good</td>
<td>A lot of pairs are considered similar with this metrics so they are wrong.</td>
</tr>
<tr>
<td>2,4</td>
<td>Blue-red</td>
<td>wrong</td>
<td>There is an important peak with low similitude. But a biggest peak has higher similitude.</td>
</tr>
<tr>
<td>1,3</td>
<td>Green-black</td>
<td>good</td>
<td>The metrics mark these pairs as similar so they are wrong too.</td>
</tr>
<tr>
<td>1,5</td>
<td>Green-purple</td>
<td>wrong</td>
<td>A lot of pairs are considered similar with this metrics so they are wrong.</td>
</tr>
<tr>
<td>2,5</td>
<td>Blue-purple</td>
<td>wrong</td>
<td></td>
</tr>
<tr>
<td>1,3</td>
<td>Green-black</td>
<td>good</td>
<td>Looking at these histograms it seems the black culture is the more dissimilar culture. The metrics are pretty dissimilar but also positive and near to 0, so some of them are wrong but they are globally right.</td>
</tr>
<tr>
<td>2,3</td>
<td>Blue-black</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3,4</td>
<td>Black-red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3,5</td>
<td>Black-purple</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fig 4.4.10.8:** Clusters obtained using the Louvain Method.

We can see that it has not been possible to cluster in different clusters the red, blue and part of the purple cultures. In future projects, we recommend to use another learning step to assign a good ponderation of weights $\beta$ (to be multiplied by the factor of goodness) the weak classifiers set belonging to each set to a different type of pairing $\text{Single}(D)$. 
5. Conclusions

The Adaboost algorithm is a very powerful tool to improve the performance of a set of weak classifiers creating a strong one composed by some of them iteratively selected. The Adaboost algorithm used here differs from the classical one because ours is based in classifying by similarity. It has been proved that we can use it to learn a strong metric of similitude between samples, beyond discriminating if a pair of samples is similar or no.

This degree or metric of similitude between all the samples can be used by the Louvain Method to group different groups of very similar samples.

Our objectives have been

- to cluster the normal samples (the ones not affected by any treatment) in a different cluster than the abnormal ones (affected by different treatments).
- to cluster the abnormal samples not by its treatments, but by the impact of them in the feature space.

For that reason we have needed to learn a metric function able to obtain a metric of:

- dissimilitude between normal samples and abnormal samples
- similitude between samples affected by the same treatment
- similitude between samples affected with different treatments but with the same impact for each treatment. In fact, we have empirically proved that it is not possible to obtain a dissimilarity from cultures with same distribution. This is the key of the clustering success.

However, in order to obtain a metric like the one we want, a combination of different metric functions is needed. Each metric function in the combination has its own purpose. Each metric function have a good behavior for the conditions given to it. Each condition consists in a different kind of pairing from the training samples used to learn the metric function. For a single condition or type of pairing we can conclude for the learning of the metric function that:

- the more training samples are used, the more accurate is the metric function. This is because the statistical model of the training samples is more accurate too. We have not used a lot of training samples because the learning step requires more computational time.
- The more weak classifiers we have (and, probably, more types of them), the more adaptable to the training samples the metric function is.

The results obtained after learning some metric functions from some specific artificial data allow us to conclude the next assertions about each type of pairing:

- single pairing only allows to discriminate the used culture-base. This only allows us to obtain usually two clusters.
- complete pairing is useful for some Gaussian cases but a counterexample has been presented.
- a combination (that we call multi-single pairing) of single pairings using each culture as culture-base is the best solution.

Multi-single pairing seems to be the best option. Our suggestion for future works is adding another step of learning. This second learning step must calculate the optimal meta-parameters for each set.
of weak classifiers learned for each single pairing. With this method the resulting metric is seen as a ponderation of metrics based in single pairing. We have assigned some manual weights (or meta-parameters) and we have obtained different results. Some of them better than others.

It can be another option just to remove the weak classifiers belonging to single pairings with a culture-base not able to be dissimilar from any other culture. This would be equivalent to use a meta-parameter equal to 0 so it does not exclude the previous proposal.

However, these methods have specially worked well for Gaussian distributions, so the next step is to try with more non-Gaussian distributions.

Another proposal for the future is to cluster, not by grouping the most similar samples as the Louvain Method does, but by disjointing the most dissimilar ones.

We have not considered the case where a treatment could be a placebo (produce healthy cells). Two normal cultures would complicate the algorithm and increase strongly the time of execution. It could be better to discard previously second normal cultures: a separate application could be made. That way the final method becomes more modularized.
6. References


   of Computer Vision 57(2), 137–154, 2004

   http://www.stanford.edu/~hastie/Papers/AdditiveLogisticRegression/alr.pdf

[4] Liu Ren, Gregory Shakhnarovich, Jessica K. Hodgins, Hanspeter Pfister and Paul Viola,
   Learning Silhouette Features for Control of Human Motion, 2004. ACM Transactions on Graphics,

   Audio Fingerprint, 2009. IEEE TRANSACTIONS ON INFORMATION FORENSICS AND
   SECURITY, VOL. 4, NO. 4, DECEMBER 2009


   http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.9277&rep=rep1&type=pdf

   http://arxiv.org/pdf/condmat/0308217

   Experiment (2008) P10008


    preserving modularity, 2007