Departamento de Matemáticas, Facultad de Ciencias
Universidad Autónoma de Madrid

Archetypal analysis and applications

Degree in Mathematics

Author: Guillermo García Cobo
Advisor: Javier Cárcamo Urtiaga

Course 2021-2022

Abstract

Archetypal analysis is an unsupervised learning technique that seeks to represent the observations of a sample through convex combinations of its extreme points. These extreme points, the so-called archetypes, can be considered as representatives of the sample, so they can be applied in dimensionality reduction and clustering. In addition, the use of convex combinations provides interpretability, a desirable property that other methods do not have.

This work consists of two parts. The first one seeks to theoretically define the archetypal analysis. To do this, we will first present the mathematical concepts necessary to understand the foundational basis of the archetypes. Among other things, we will remind notions of convexity and optimization. With these concepts clear, we will proceed to define the archetypes and the procedure originally proposed to calculate them.

In the second part, we will focus on the practical application of the described method. First, we will describe and compare more efficient variants of the archetype calculation procedure. On the other hand, after obtaining a substantial improvement in computing times, we will apply the archetypal analysis to two real examples: facial recognition and customer segmentation. Thanks to this, we will observe first-hand the advantages offered by archetypes over other unsupervised learning techniques.

This is a semi-automatic English translation of the original Spanish document.

Contents

CHAPTER 1
Introduction and preliminaries

1.1. Introduction

With the proliferation of areas related to machine learning, unsupervised learning is one of the areas receiving the most attention lately. The fact that it does not require any human intervention to learn, with the consequent savings in resources that this entails, has aroused the interest of many. On the other hand, one of the most desired properties of models is that they are interpretable. This is a great added value, since it allows us to fully understand the result they produce and what they are based on to produce it.

Archetype analysis is a technique that combines both characteristics, making use of convex combinations and extreme data points, as we will describe in the following pages. In order to understand this technique and its advantages over others, the document is organized as follows. In this Chapter 1, the mathematical concepts on which the archetypes are based will be described in detail. In Chapter 2, we will formally define the archetypes and how their computation was initially proposed. In Chapter 3, we will present more efficient alternatives to obtain archetypes, comparing them over different data sizes and obtaining significant improvements over the original procedure. In Chapter 4, we will apply archetypal analysis to two real examples, comparing its performance with respect to other unsupervised learning techniques. Finally, in Chapter 5, we will briefly describe possible future lines of work.

As for the bibliography, specific articles are cited throughout the paper. However, we review here all the papers consulted for each section. To establish the basics on convexity, we have consulted [6, 40, 19, 27, 34]. For convex functions, the articles are [7, 1]; while to deal with the concept of biconvexity the article we have consulted is [20]. To define the concepts related to convex optimization, [7, 17, 24, 15, 2, 21]. In order to define the central issue at hand (archetypal analysis), we consulted [11, 16, 28, 5, 29, 14, 13, 22, 3, 30, 4]. To compare the different proposed methods that compute the archetypes, the references are [9, 33, 37, 13, 30, 4]. Finally, to use the archetypes in real examples, [43, 42, 23, 36, 32] have been consulted.

1.2. Preliminary results

We proceed next to recall several notions and properties useful for the later sections.

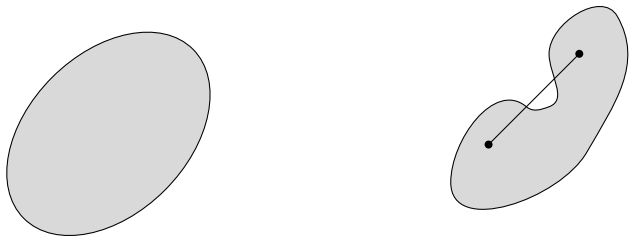
1.2.1. Convex sets

Definition 1.1. A subset $C \subset \mathbb{R}^n$ is said to be *convex* if given $x, y \in C$, it is held that for all $t \in [0, 1]$

$$tx + (1 - t)y \in C.$$

It is worth mentioning that, although for this work we will be satisfied with the previous definition, obviously this definition can be extended to more general spaces than the Euclidean ones.

Intuitively, a set is convex if we can join any pair of points with a segment contained in the set. An example of this idea is shown in Figure 1.1.



(a) Convex set (b) Non-convex set

Figure 1.1: Examples of convexity in $\mathbb{R}^2$
Adaptation from [41]

Before proceeding further, it is worth mentioning that, unless otherwise noted, v vectors are columnar. Also, v^T indicates the transposed vector.

Other examples of convex sets are as follows:

- The balls $B(x_0, \epsilon) = \{x \in \mathbb{R}^n : \|x - x_0\| < \epsilon\}$ (for any norm).
- Hyperplanes: $C = \{x : p^T x = \alpha\}$ with $p \in \mathbb{R}^n, \alpha \in \mathbb{R}$.
- Half-spaces: $C = \{x : p^T x \leq \alpha\}$ with $p \in \mathbb{R}^n, \alpha \in \mathbb{R}$.
- Arbitrary intersection of convexes: if C_i is convex for all $i \in I$, then $C = \bigcap_{i=1}^{I} C_i$ is convex.

Definition 1.2. The *convex combination* of $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ is given by

$$t_1 x_1 + t_2 x_2 + \cdots + t_k x_k,$$

with $\sum_{i=1}^{k} t_i = 1$ y $t_i \geq 0$, for all $i = 1, \ldots, k$.

We have the following proposition that will help us relate the convex combinations of a set to its convexity.

Proposition 1.3. *C is convex if and only if any convex combination of points of C is in C.*

Proof. Suppose first that C is convex. We proceed by induction on k, the number of points of the convex combination.
If $k = 1$, and $x \in C$, then $1\dot{x} \in C$. Note that the case $k = 2$ is by definition of convexity.
Let us assume it to be true for $k > 1$, and we will show them for $k + 1$. Let $x_1, \ldots, x_{k+1} \in C$. We want to see that

$$t_1x_1 + t_2x_2 + \cdots + t_{k+1}x_{k+1} \in C. \tag{1.1}$$

By induction hypothesis, we have that

$$y = \frac{t_1}{t_1 + \cdots + t_k}x_1 + \frac{t_2}{t_1 + \cdots + t_k}x_2 + \cdots + \frac{t_k}{t_1 + \cdots + t_k}x_k \in C.$$

Note that the terms are well defined, because if $t_1 + \cdots + t_k = 0$, the convex combination (1.1) equals $x_{k+1} \in C$, which is trivially true.
Now, we can express (1.1) as

$$t_1x_1 + t_2x_2 + \cdots + t_{k+1}x_{k+1} = (t_1 + \cdots + t_k)y + t_{k+1}x_{k+1},$$

which is in C by definition of convexity.
For the reciprocal, it suffices to note that since any convex combination of points of the set is in the set, it is also true for combinations of two elements and therefore the definition of convexity is satisfied. □

Definition 1.4. The *convex hull* of a subset C is defined as the smallest convex set containing C and is denoted by $\operatorname{conv}(C)$. Equivalently, by Proposition 1.3, we can see it as the set of all possible convex combinations of C:

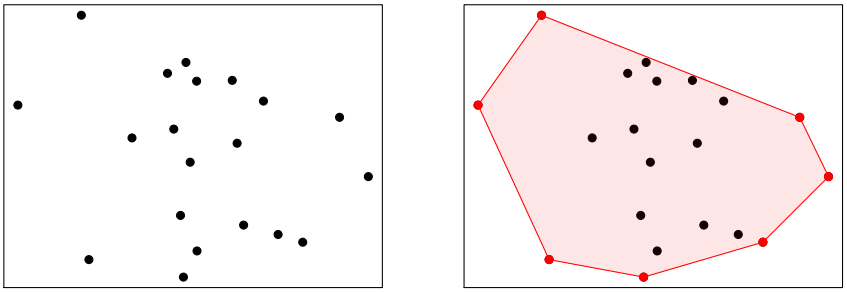
$$\operatorname{conv}(C) = \left\{ \sum_{i=1}^{k} t_i x_i : x_i \in C,\ t_i \geq 0,\ i = 1, \ldots, k, k \in \mathbb{N},\ \sum_{i=1}^{k} t_i = 1 \right\}.$$

We have the following properties of the convex hull:

- C is convex if and only if $C = \operatorname{conv}(C)$.
- $C \subset \operatorname{conv}(C)$.

In Figure 1.2 we can see an example of the convex hull of a point cloud in the plane.

Definition 1.5. Let C be a nonempty convex set. We say that $x \in C$ is an *extreme point* if x cannot be expressed as a proper convex combination in C. That is, if $x = tx_1 + (1 - t)x_2$, with $x_1, x_2 \in C$, then $t = 0$ or 1. We denote the set of extreme points of C as $\operatorname{Ext}(C)$.



(a) Nube de puntos (b) Cierre convexo

Figure 1.2: Example of convex hull of a point cloud in $\mathbb{R}^2$

Proposition 1.6. . *If $C \subset \mathbb{R}^n$ is convex, it holds that* $\text{Ext}(C) \subseteq \partial C$*, where ∂C denotes the boundary of the set C.*

Proof. Suppose $x \notin \partial C = \overline{C} - \operatorname{int} C$, where $\overline{C}$ denotes the closure of C and $\operatorname{int} C$ the interior of C. Then, $x \in \operatorname{int} C$, then there exists $r > 0$ such that the ball $B(x, r) \subset C$. Therefore, x would be in the middle of a segment included in C, then $x \notin \text{Ext}(C)$. □

Proposition 1.7. . *Let $A \subset \mathbb{R}^n$ be a set. Then, it is verified that* $\text{Ext}(\text{conv}(A)) \subset A$.

Proof. Let $x \in \text{conv}(A)$ be an extreme point. By definition of the convex hull, $x = \sum_i t_i a_i$ for $a_i \in A$, $\sum_i t_i = 1$ and $t_i \geq 0$. Now, since x is an extreme point, it must be given that only one $t_i > 0$. Then, such t_i is 1 and $x = a_i$. Therefore, $x \in A$. □

To help understand this concept, in Figure 1.2 the extreme points of the convex hull of the point cloud have been marked in red. That is, if A is the point cloud, the points marked in red are $\text{Ext}(\text{conv}(A))$.

We proceed to name the convex hull of a special set: that of the standard unit vector basis.

Definition 1.8. The *n-simplex*, denoted by Δ^n, is the subset formed by the convex hull of the $n+1$ standard unit vectors of $\mathbb{R}^{n+1}$. That is,

$$\Delta^n = \left\{ (t_1, \ldots, t_{n+1}) \in \mathbb{R}^{n+1} : \ t_i \geq 0, \ i = 1, \ldots, n+1, \ \sum_{i=1}^{n+1} t_i = 1 \right\}.$$

Note that the weights of any convex combination can be viewed as points in Δ^n. Therefore, whenever we want to obtain weights that form a convex combination, we must restrict the search to the appropriate simplex. This will be relevant later, when we search for valid convex combinations that minimize a certain criterion.

Figure 1.3 shows the standard 2-simplex contained in $\mathbb{R}^3$.

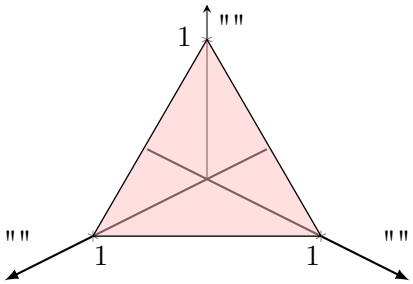


Figure 1.3: 2-simplex (Δ^2)
Adaptation from [35]

Finally, we state a very relevant theorem for archetype construction, as we will see later. This result relates convex combinations, convex hull and extreme points.

Theorem 1.9. *[Minkowski-Carathéodory] Let C be a compact convex subset contained in $\mathbb{R}^n$ of dimension n_C. Then, any point in C is a convex combination of at most $n_C + 1$ extreme points. In particular,*

$$C = \text{conv}(\text{Ext}(C)).$$

This result is sufficient for the data types we are going to deal with (real and finite), but there is an even more general theorem valid for infinite-dimensional sets.

Theorem 1.10. *[Krein-Milman] Let C be a compact convex subset contained in a locally convex vector space. Then,*

$$C = \overline{\text{conv}(\text{Ext}(C))}.$$

Both theorems, as well as their proof, can be found in Chapter 8 of [40] (Theorems 8.11 and 8.14, respectively).

1.2.2. Convex functions

Definition 1.11. A function $f : C \to \mathbb{R}$ is *convex* if its domain $C \subset \mathbb{R}^n$ is convex and for all $x_1, x_2 \in C$, and all $t \in [0, 1]$, it holds

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

We now proceed to state two useful results if in addition to convexity we assume differentiability conditions for the function.

Theorem 1.12. *Let $f : C \to \mathbb{R}$ be differentiable over a convex and open domain C. Then, f is convex if and only if for all $x_1, x_2 \in C$.*

$$f(x_1) \geq f(x_2) + \langle \nabla f(x_2), (x_1 - x_2) \rangle. \tag{1.2}$$

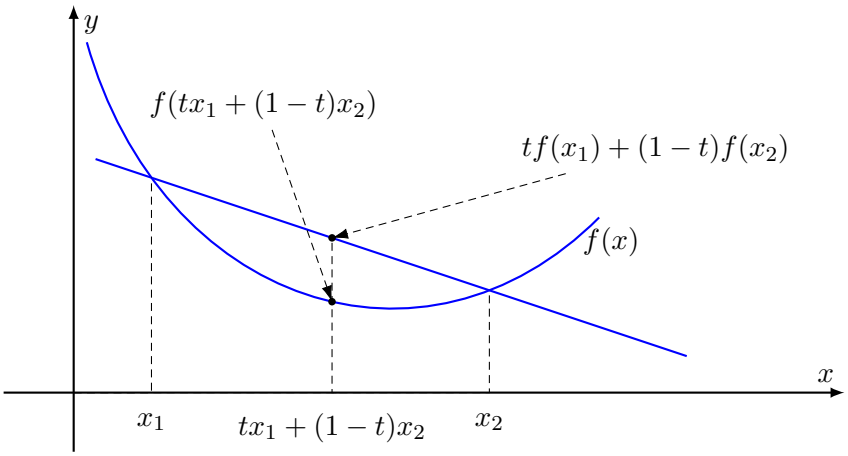


Figure 1.4: Example of convex function $f(x)$ en $\mathbb{R}^2$
Adaptation from [25]

Proof. Suppose first that f is differentiable and convex in C. Let $x_1, x_2 \in C$. Since f is convex in C, we have

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

By clearing we arrive at

$$\frac{f(x_2 + t(x_1 - x_2)) - f(x_2)}{t} \geq f(x_1) - f(x_2).$$

Finally, taking the limit when $t \to 0$ we arrive at (1.2).
Reciprocally, let $x, y \in C$, $t \in [0,1]$. Let $x^* = tx + (1-t)y$. Now, we apply (1.2) to the pairs (x, x^*), (y, x^*) to obtain:

$$f(x) \geq f(x^*) + \langle \nabla f(x^*), (x - x^*) \rangle. \tag{1.3}$$

That is,

$$f(y) \geq f(x^*) + \langle \nabla f(x^*), (y - x^*) \rangle. \tag{1.4}$$

Multiplying (1.3) by t and (1.4) by $1-t$ and adding them we obtain:

$$tf(x) + (1-t)f(y) \geq f(x^*) + t\langle \nabla f(x^*), (x - x^*) \rangle + (1-t)\langle \nabla f(x^*), (y - x^*) \rangle.$$

Recalling that $x^* = tx + (1-t)y$, we arrive at

$$tf(x) + (1-t)f(y) \geq f(tx + (1-t)y).$$

□

Theorem 1.13. *Let $f : C \to \mathbb{R}$ with two continuous derivatives over a convex and open domain C. Then, f is convex if and only if for all $x \in C$, $H_f(x)$ is positive semidefinite, where $H_f(x)$ is the Hessian matrix of f at point x.*

Proof. Suppose H_f is not positive semidefinite for $x \in C$. By continuity of the Hessian, there exists $y \in C$ such that, for all $t \in [0,1]$,

$$\langle (y-x), H_f(x+t(y-x))(y-x)\rangle < 0.$$

Now, with this and Taylor's second-order approximation, we have that for these x, y,

$$f(y) < f(x) + \langle \nabla f(x), (y-x)\rangle.$$

Using Theorem 1.12, we conclude that f is not convex.
For the reciprocal, let $x, y \in C$. Again, by Taylor, we know that for some $t \in [0,1]$, we have

$$f(y) = f(x) + \langle \nabla f(x), (y-x)\rangle + \frac{1}{2}\langle (y-x), H_f(x+t(y-x))(y-x)\rangle.$$

We know that H_f is positive semidefinite in C and $x + t(y-x) \in C$, then we have

$$f(y) \geq f(x) + \langle \nabla f(x), (y-x)\rangle.$$

By Theorem 1.12, we know that this implies convexity of f. □

1.2.3. Convex optimization

Definition 1.14. A *convex optimization problem* consists of solving the following:

$$\min_{x \in C} f(x)$$

where $f : D \to \mathbb{R}$ is convex and $C \subset D \subset \mathbb{R}^n$ is convex.

An important theorem that will allow us to eliminate the distinction between local and global minima for this type of problem is the following:

Theorem 1.15. *Let $f : D \to \mathbb{R}$ be convex. If x^* is a local minimum of f over a convex $C \subset D$, then x^* is also a global minimum of f over C.*

Proof. Since x^* is local minimum, for any $y \in C$, we can choose a sufficiently small $t > 0$ such that

$$f(x^*) \leq f(x^* + t(y - x^*)).$$

Since furthermore f is convex, we have

$$f(x^* + t(y-x^*)) = f(ty + (1-t)x^*) \leq tf(y) + (1-t)f(x^*).$$

Putting both expressions together, we arrive at

$$f(x^*) \leq tf(y) + (1-t)f(x^*),$$

which implies that $f(x^*) \leq f(y)$. Since y was an arbitrary point of C, we have proved that x^* is a global minimum of f over C. □

We will now give a characterization of these minima for the convex optimization problem.

Theorem 1.16. *Let $f : D \to \mathbb{R}$ be convex and differentiable. Then x^* is a (global) minimum of f over the convex set $C \subset D$ if and only if*

$$\langle \nabla f(x^*), (x - x^*) \rangle \geq 0, \text{ para todo } x \in C.$$

Proof. First, suppose that there exists $x \in C$ such that $\langle \nabla f(x^*), (x - x^*) \rangle < 0$. Then, $(x - x^*)$ is a local descent direction. Then, there exists $t \in (0, 1)$ such that

$$f(x^*) > f(x^* + t(x - x^*)) = f(tx + (1 - t)x^*).$$

Since $tx + (1 - t)x^* \in C$ by convexity, x^* would not be global minimum in C.
For the reciprocal, since f is convex, by Theorem 1.12 we have

$$f(x) \geq f(x^*) + \langle \nabla f(x^*), (x - x^*) \rangle \text{ for all } x \in C.$$

Then, if $\langle \nabla f(x^*), (x - x^*) \rangle \geq 0$, we have

$$f(x) \geq f(x^*) + \langle \nabla f(x^*), (x - x^*) \rangle \geq f(x^*).$$

Then x^* is a minimum. □

1.2.4. Biconvexity

Further on, the function we are going to try to minimize is not convex over the total set of variables, but it is convex in each variable fixing the rest. We define this concept below.

For what follows, we consider $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ two nonempty convex sets, and $B \subseteq X \times Y$ another set.

Definition 1.17. We define the *x-sections* and *y-sections* of B as follows:

$$B_x := \{y \in Y : (x, y) \in B\},$$
$$B_y := \{x \in X : (x, y) \in B\}.$$

Definition 1.18. The set $B \subseteq X \times Y$ is said to be *biconvex* in $X \times Y$ if B_x is convex for every $x \in X$ and B_y is convex for every $y \in Y$.

Definition 1.19. A function $f : B \to \mathbb{R}$ is *biconvex* on B if for each $x \in X$

$$f_x(\bullet) := f(x, \bullet) : B_x \to \mathbb{R}$$

is a convex function, and for each $y \in Y$

$$f_y(\bullet) := f(\bullet, y) : B_y \to \mathbb{R}$$

is a convex function.

As described in [20], biconvex functions can have local minima, being a very general optimization problem. In this same article it is mentioned that without imposing additional conditions on the function to be optimized, the most that can be aspired to is to find partial optima, defined as follows.

Definition 1.20. Let $f : B \to \mathbb{R}$ be a biconvex function and let $(x^*, y^*) \in B$. It is said that (x^*, y^*) is a *partial optimum* of f on B if

$$f(x^*, y^*) \leq f(x, y^*) \ \forall x \in B_{y^*} \text{ y } f(x^*, y^*) \leq f(x^*, y) \ \forall y \in B_{x^*}.$$

In [20], it is shown that an algorithm that converges to these partial optima is as follows.

Definition 1.21. [Alternating convex optimization algorithm] Fixed an initial point (x^*, y^*), iterate as follows:

1. $x^* := \text{argmin}_{x \in B_{y^*}} f(x, y^*)$
2. $y^* := \text{argmin}_{y \in B_{x^*}} f(x^*, y)$

By biconvexity, both steps are a convex optimization problem. Moreover, the order of these steps is obviously interchangeable.

1.2.5. Gradient descent in convex optimization

When there is no closed solution to an optimization problem, a widely used technique is that of *gradient descent*. This relies on first-order local information to iteratively approximate a minimum. In each iteration, steps are taken in the direction of maximum decrement, the opposite of the gradient: $-\nabla f(x)$. The minimum we reach need not be global in general, although, as we have already seen, in convex optimization it must be.

We define below this method in general and the adaptations that are necessary to apply them to solve convex optimization problems.

Definition 1.22 (General gradient descent algorithm)**.** Starting from a differentiable f function and an arbitrary initial value x_0 in its domain, we iterate based on the following rule:

$$x_{k+1} := x_k - \alpha_k \nabla f(x_k), \ \alpha_k \in \mathbb{R}^+,$$

until a certain convergence criterion is satisfied. For example, if $\frac{\|x_{k+1} - x_k\|}{\|x_k\|} < \epsilon$ or $\nabla f(x_k) < \epsilon$.

The parameter α_k is the size of the step we take in each iteration and is often referred to as the *learning rate*. There are a multitude of ways to initialize this parameter, but a desirable one would be to advance to the minimum value of the function in the opposite direction of the gradient:

$$\alpha_k := \underset{\alpha \in \mathbb{R}^+}{\text{argmin}} \, f(x_k - \alpha \nabla f(x_k)).$$

Calculating such α_k constitutes a new (in fact, convex) optimization problem that would add an extra cost per iteration. To simplify the computation of α_k, a common strategy is to define it as a function inversely proportional to the number of iterations.

Once we have defined the general algorithm, it is time to adapt it to solve convex optimization problems. Recall that our objective is.

$$min_{x \in C} f(x),$$

then the solution we give must be in C.

To begin with, f must be differentiable in order to compute its gradient. But in addition, with the method we have described, it is possible that the algorithm leads us to solutions outside this set C, then we must adapt it to meet this constraint. We propose below two adaptations.

For the first one, we need the following mathematical tool:

Definition 1.23. The *projection* of a point y on C is defined as.

$$P_C(y) := \operatorname*{argmin}_{x \in C} d(y, x),$$

for a given distance d.

With this, we can define a first adaptation of the general algorithm that ensures that the convex optimization constraints are satisfied:

Definition 1.24. [Projected gradient descent algorithm] Starting from a differentiable f function and an arbitrary initial value $x_0 \in C$, we iterate based on the following rule:

$$x_{k+1} := P_C(x_k - \alpha_k \nabla f(x_k)),\ \alpha_k \in \mathbb{R}^+.$$

The idea behind this algorithm is to move in the direction of maximum decrement of f as far as the set C will allow us to go. While this version solves the convex optimization problem, it requires the sometimes costly operation of projection.

The second alternative is also based on restricting the moves of each iteration to the set C in question. To do so, we search for the direction of maximum gradient decrement by studying directions only between points of the set (which we will call *feasible directions*), and then move towards it without leaving C using it as a convex set. This algorithm was first described in 1956 by Princeton Ph.D. students Marguerite Frank and Philip Wolfe in [17], and was revisited by Jaggi in 2013 [24].

Definition 1.25. [Frank-Wolfe algorithm] Starting from a differentiable f function and an arbitrary initial value $x_0 \in C$, iterate based on the following rule:

$$s_k := \operatorname*{argmin}_{s \in C} \langle \nabla f(x_k), s \rangle$$

$$x_{k+1} := x_k + \alpha_k (s_k - x_k),\ \alpha_k \in [0, 1].$$

Some remarks may be mentioned:

- By calculating s_k, we are obtaining the feasible direction of maximum decreasing gradient from x_k. This is because

$$\operatorname*{argmin}_{s\in C}\langle\nabla f(x_k), s\rangle = \operatorname*{argmin}_{s\in C}\langle\nabla f(x_k), (s - x_k)\rangle.$$

  In the general algorithm, this direction is directly opposite the gradient, but we cannot use it since there may be no points in C to reach from x_k by following it.

- At each step, $x_{k+1} \in C$ by convexity, since

$$x_k + \alpha_k(s_k - x_k) = \alpha_k s_k + (1 - \alpha_k)x_k,\ x_k, s_k \in C,\ \alpha_k \in [0, 1].$$

- It is usual to take $\alpha_k = \frac{2}{k+2}$, with $k = 0, 1, \ldots$

- The extra cost of this algorithm lies in finding the feasible direction of maximum decreasing gradient in C. As we will see, for certain C this operation is less expensive than the P_C projection.

CHAPTER 2
Description of the original method

In this chapter we are going to present the method described by Cutler and Breiman in their seminal work on archetype analysis [11]. We will see what motivates the proposal of archetype analysis, where the so-called archetypes are found in the data set and the computational algorithm that was originally proposed.

2.1. Motivation and definition

In [11], the authors begin by exposing one of the main disadvantages that some dimensionality reduction methods such as *Principal Component Analysis* (PCA) have: the difficulty in interpreting the results obtained. To illustrate this idea, we describe a problem presented by Flury and Riedwyl in [16]: we have the measurements of the heads of 200 soldiers of the Swiss army, and we want to generate the measurements of a few "principal" heads to manufacture gas masks valid for the whole army. This dataset is available in R. See [22, 3].

With this objective, as these main patterns should represent the totality of the sample data, a first approach is that each data can be approximated as a combination of the main patterns. For this, one option is given by a variant of PCA. Let $x_1, \ldots, x_n$ be the sample data, we define the main patterns as:

$$z_1^*, \ldots, z_p^* = \underset{z_1,\ldots,z_p}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| x_i - \sum_{k=1}^{p} \alpha_{ik} z_k \right\|^2,$$

where $p < < n$.

It can be shown that these $z_1^*, \ldots, z_p^*$ correspond to the principal components of the data set. For this, the same arguments are used as in the definition of PCA. It is recommended to refer to [11, Section 1], together with [29, Section 4.1.1].

However, this procedure produces results that are difficult to interpret, as well as implausible in many cases. For example, as the authors comment, negative distances are obtained between points of the same principal pattern. This is because at no time have these patterns been forced to be plausible, i.e., to belong to the real population from which the data originate.

To solve this problem, it is imposed that the main patterns (from now on) are convex combinations of the sample data. That is,

$$z_k = \sum_{j=1}^{n} \beta_{kj} x_j,$$

with $\sum_{j=1}^{n} \beta_{kj} = 1$ y $\beta_{kj} \geq 0$.

Again, the goal is to study which combinations of these archetypes can represent the entire sample data. We then look for α_{ik} that minimize

$$\left\| x_i - \sum_{k=1}^{p} \alpha_{ik} z_k \right\|^2.$$

It again makes sense to impose that these combinations are convex ($\sum_{k=1}^{p} \alpha_{ik} = 1$ and $\alpha_{ik} \geq 0$), since we once again obtain interpretable results. For example, we can view the resulting coefficients as probabilities:

$$\alpha_{ik} = \mathbb{P}(x_i \in [z_k]),$$

where $[z_k]$ denotes a subpopulation or class represented by that archetype.

We can now formally define the archetypes of a data set.

Definition 2.1. Let $x_1, \ldots, x_n$ be a data set. Then, the corresponding *archetypes* $z_1^*, \ldots, z_p^*$ are those which minimize

$$\text{RSS} = \sum_{i=1}^{n} \left\| x_i - \sum_{k=1}^{p} \alpha_{ik} z_k \right\|^2 = \sum_{i=1}^{n} \left\| x_i - \sum_{k=1}^{p} \alpha_{ik} \sum_{j=1}^{n} \beta_{kj} x_j \right\|^2, \tag{2.1}$$

with $\sum_{k=1}^{p} \alpha_{ik} = 1, \sum_{j=1}^{n} \beta_{kj} = 1,\ \alpha_{ik}, \beta_{kj} \geq 0,\ p < \ < n.$

From now on, to express the condition that the weights form a convex combination, we will use the concept of n-simplex (Definition 1.8). That is, in the above definition we have $\alpha_i \in \Delta^{p-1}$, $\beta_k \in \Delta^{n-1}$, with $\alpha_i = (\alpha_{i1}, \ldots, \alpha_{in})$ and $\beta_k = (\beta_{k1}, \ldots, \beta_{kn})$, $i = 1, \ldots, n$, $k = 1, \ldots, p$.

Finally, it should be noted that the set of archetypes is not necessarily unique.

2.2. Location of archetypes

The Theorems 1.9 and 1.10 give us a first (accurate) intuition of what the archetypes of a data set are and where they are located. Both tell us that a convex set can be expressed as the convex combination of only its extreme points. Thus, if we transform our data set into a convex set C (via convex hull) and take $p = \text{card}(\text{Ext}(C))$, the archetypes will be the extreme points of it, since they are able to express through convex combinations any point of the data set with RSS = 0.

In the original paper, the authors state the following proposition, in line with what we have just reasoned.

Proposition 2.2. *Let $x_1, \dots, x_n \in \mathbb{R}^d$ and $C = \text{conv}(\{x_1, \dots, x_n\})$ and $p \in \mathbb{N}$ be the chosen number of archetypes. We consider $N = \text{card}(\text{Ext}(C))$. One has:*

1. *If $p = 1$, $z_1 = \overline{x}$ (mean of $\{x_1, \dots, x_n\}$) is the only archetype.*
2. *If $1 < p < N$, then there exists a set of archetypes $\{z_1, \dots, z_p\} \subset \partial C$.*
3. *If $p = N$, then $\text{Ext}(C) = \{z_1, \dots, z_p\}$ is a set of archetypes.*

Proof. First, for all three cases it is trivial that the proposed archetypes are convex combinations of the data set.

1. It is well known that the mean, $\overline{x}$, is the unconstrained minimizer of RSS when $p = 1$.
2. Let $\{z_1, \dots, z_p\}$ be a set of archetypes (which need not be unique). Suppose $z_1 \in \text{int}\, C$. We take z_j with $j \neq 1$ another arbitrary archetype and define

$$z(t) = tz_1 + (1 - t)\, z_j, \text{ for } t > 1.$$

   We consider $t_0 > 1$ such that $z(t_0) \in \partial C$. We observe that

$$z_1 = \frac{1}{t_0} z(t_0) + \left(1 - \frac{1}{t_0}\right) z_j \in \text{conv}((z(t_0), z_j)\,.$$

   Therefore, $\text{conv}(\{z_1, \dots, z_p\}) \subseteq \text{conv}(\{z(t_0), \dots, z_p\})$, with $z(t_0) \in \partial C$. In addition, $\{z(t_0), \dots, z_p\}$ will also minimize RSS, so it is a set of archetypes. We can repeat this procedure with any archetype from the set $\{z_1, \dots, z_p\}$ that is in $\text{int}\, C$ and we obtain a new set of archetypes in ∂C.
3. By Theorem 1.9 we know that $C = \text{conv}(\text{Ext}(C))$, then RSS $= 0$ and $\text{Ext}(C)$ is a set of archetypes as we wanted.

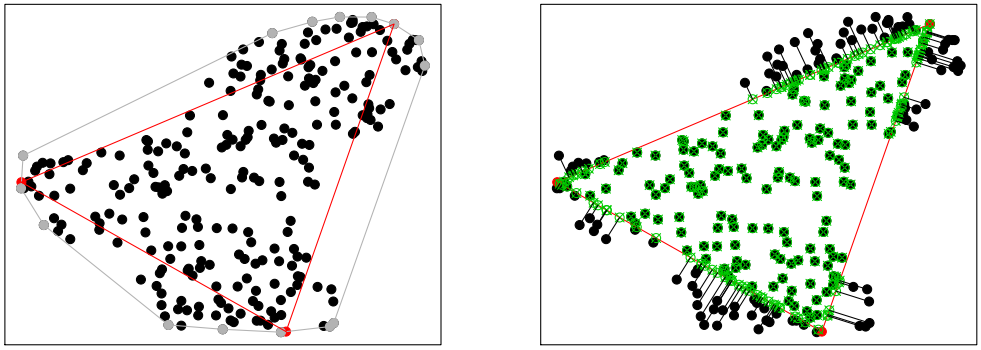
□

Note that at point 2 we cannot exchange ∂C for $\text{Ext}(C)$, since although we are guaranteed that with that straight line we will reach a point of ∂C, we cannot be sure that this point is in $\text{Ext}(C)$.

On the other hand, in the article [11] the proposition is written with $\{x_1, \dots, x_n\} \cap \partial(C)$, but $\text{Ext}(C)$ is sufficient to get RSS $= 0$. By Propositions 1.6 and 1.7, $\text{Ext}(C) \subseteq \{x_1, \dots, x_n\} \cap \partial(C)$, but this inclusion can be strict if, for example, there are three points in $\{x_1, \dots, x_n\} \cap \partial(C)$ contained in the same straight line. Therefore, our new condition requires fewer points.

In order to visualize the localization just described, three archetypes of a point cloud have been computed using the R [14] package. The results are collected in Figure 2.1. First, in Figure 2.1(a) we observe that the archetypes (red dots) are located on the boundary and that their convex hull attempts to approximate the convex hull of the total set. On the other hand, in Figure 2.1(b) we can see how the archetypes

approximate the data set. Recall that the convex hull of C is defined as the set of convex combinations of elements of C, so any point belonging to the convex hull of the archetypes will be expressed without error by a convex combination of the archetypes. However, points outside the convex hull should be approximated by the nearest point of the convex hull (projection). In fact, RSS is the sum of distances of these points to their respective projections. In Figure 2.1(b) we see in green the representation given by the archetypes. For points already belonging to the convex hull, the green point coincides with the original; while for points outside the convex hull we see the line joining the original point with the projected green point.



(a) Archetypes and Convex Hull (b) Set approximation

Figure 2.1: Example of archetypes and their approximations for a given point cloud. On the left, in gray, the convex hull of the points. In red, the archetypes and their convex hull. On the right, in green, the projected points.

To emphasize the idea that the more archetypes we consider, the better we approximate the *Convex Hull* and therefore the smaller RSS will be, the error on the data set of Figure 2.1 has been calculated when using between 1 and 5 archetypes. The decreasing curve of the error is observed in Figure 2.2.

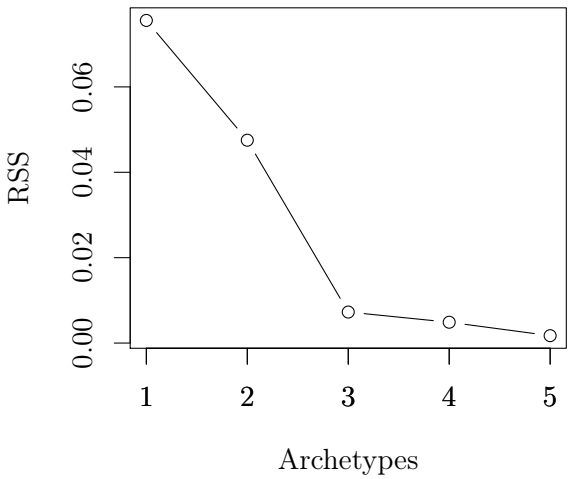


Figure 2.2: Evolution of RSS as a function of the number of archetypes.

2.3. Calculation of the archetypes

As we discussed above, to obtain the archetypes we must minimize the sum RSS in (2.1) by finding the optimal values of α and β. This problem is solvable by general least-squares optimizers, but in practice it is impractical. In the following, we describe the 1994 computational method, based on the ideas proposed in the original paper, as well as the details discussed in [13] by the developers of the R archetype library [14].

First, for ease of understanding and inspired by [13, 5, 30, 4] we will use matrix notation. Let $X \in M_{n\times m}$ be the data matrix (n data with m dimensions). Let $A \in M_{n\times p}$, $B \in M_{p\times n}$ be the matrices whose rows are α_i and β_k, respectively. With this, the archetypes are

$$BX = Z \in M_{p\times m}$$

and the objective to be minimized is

$$\text{RSS} = \|X - ABX\|^2 ,$$

where the matrix norm is the Frobenius norm.

It is noted that RSS is not convex in the product AB, since several examples are presented in the literature where there are distinct local minima (which contradicts Theorem 1.15). However, RSS is indeed convex in A and in B separately (fixing the other matrix). That is, as we state in Definition 1.19, RSS is *biconvex*. The proof is based on obtaining the Hessian matrices of RSS for A and for B, showing that they are positive semidefinite and using Theorem 1.13 to conclude that RSS is convex in both cases. It is suggested to consult [11, Section 5] and/or [30, Section 2] for the complete proof.

As we discussed in Section 1.2.4, an algorithm for optimizing a biconvex function is based on iterating over convex optimization problems in each variable (Definition 1.21). This alternate optimization methodology is the one they proposed in [11], which is as follows:

1. For a fixed set of archetypes Z, find α_i that minimize

   $$\|x_i - \alpha_i Z\|^2 ,$$

   with $\alpha_i \in \Delta^{p-1}$.

2. Recalculate the archetypes $\tilde{Z}$ for the α_i found, solving $X = A\tilde{Z}$.

3. For $\tilde{Z}$, find the β_k that minimize

   $$\|\tilde{z_k} - \beta_k X\|^2 ,$$

   with $\beta_k \in \Delta^{n-1}$.

4. Redefine the archetypes $Z = BX$ for the β_k found.

5. Recalculate RSS. If it is small enough or the iteration limit has been reached, terminate. If not, return to the 1 point.

To solve the 1 and 3 convex optimization problems, the authors propose to use a least-squares optimization algorithm [28], adding a penalty constant that forces $\alpha_i \in \Delta^{p-1},\ \beta_k \in \Delta^{n-1}$ to be satisfied. That is, if we want to solve

$$\|u - wT\|^2,$$

con $w \in \Delta$, we can solve, with non-negativity constraints over w,

$$\|u - wT\|^2 + M^2 \|1 - w\|^2.$$

The later literature also shares the alternating optimization technique, although, as we will see in the next chapter, more efficient methods are proposed to solve the two convex optimization subproblems.

CHAPTER 3
Improved methods for archetype computation

In this chapter we will study two methods that compute archetypes more efficiently. As discussed in the previous chapter, the authors proposed to solve the optimization problem with least squares methods by imposing the convex constraints by penalty. This idea works, but it is clearly not efficient to impose the constraints that way.

3.1. Gradient computation

Both methods are based on gradient descent, so the first step is to obtain the gradient of the function to be optimized, RSS.

Again using matrix notation, recall that

$$\mathrm{RSS} = \|X - ABX\|^2 .$$

In order to derive this expression, we will use properties of the trace derivative, so we rewrite it as follows:

$$\begin{aligned} \mathrm{RSS} &= \|X - ABX\|^2 \\ &= \mathrm{Tr}((X - ABX)^T (X - ABX)) \\ &= \mathrm{Tr}(X^T X - X^T B^T A^T X - X^T ABX + X^T B^T A^T ABX) \\ &= \mathrm{Tr}(X^T X - 2X^T ABX + X^T B^T A^T ABX) \end{aligned} \tag{3.1}$$

where in step (3.1) we have used that the trace is an invariant linear operator to transpose.

The properties of the trace derivative that we are going to use have been obtained from [33, Section 2.5] and are as follows:

$$\frac{\partial}{\partial X} \mathrm{Tr}(AXB) = A^T B^T, \tag{3.2}$$

$$\frac{\partial}{\partial X} \mathrm{Tr}(A^T X^T BXA) = B^T XAA^T + BXAA^T. \tag{3.3}$$

Applying these properties on the derivative of (3.1) we obtain the sought gradients:

$$\nabla_A \mathrm{RSS} = \nabla_A = 2(ABXX^TB^T - XX^TB^T) = 2(AZZ^T - XZ^T) \tag{3.4}$$

$$\nabla_B \mathrm{RSS} = \nabla_B = 2(A^TABXX^T - A^TXX^T). \tag{3.5}$$

With this information, we would be able to attempt to solve the unconstrained optimization problem with gradient descent. However, as we explain in Section 1.2.5, certain adaptations are necessary to maintain convex constraints.

3.1.1. Description of improved methods

3.1.2. Projected gradient

The first method is the projected gradient method. This idea, which we already gave in Definition 1.24, is what the proposed [30] is based on.

First, it is necessary to define the projection to the corresponding simplex Δ, such that the parameters satisfy the constraints of forming a convex combination. Such a projection, for $x \in \mathbb{R}^n$, is defined as:

$$y = P_{\Delta^{n-1}}(x) = \frac{\tilde{x}}{\sum_{i=1}^{n} \tilde{x_i}},$$

with $\tilde{x_i} = \max(x_i, 0)$. Thus, we ensure that $y_i \geq 0$, $\sum_{i=1}^{n} y_i = 1$ (i.e., $y \in \Delta^{n-1}$).

We further define the projection onto a matrix as the projection applied to each of its columns.

Following what was described in Definition 1.24 and the gradient we have obtained in (3.4) and (3.5), we can now define the projected gradient descent algorithm. Starting from an initialization A_0 and B_0, we operate as follows:

1. $A_{t+1} = A_t - \mu_A \nabla_{A_t}$,
2. $A_{t+1} = P_{\Delta^{p-1}}(A_{t+1})$,
3. $B_{t+1} = B_t - \mu_B \nabla_{B_t}$,
4. $B_{t+1} = P_{\Delta^{n-1}}(B_{t+1})$,
5. $Z_{t+1} = B_{t+1}X$,
6. Recalculate RSS. If it is sufficiently small or the iteration limit has been reached, terminate. If not, return to point 1.

In [30] we also discuss different ways to initialize A_0 and B_0. In our case, since we are interested in studying the efficiency of the computational methods, we leave the initialization aside and keep it as simple as possible with diagonal matrices with 1s.

On the other hand, it should be noted that in [30] the implementation they use of the projected gradient is slightly different. This is because they adapt the gradient of RSS using the chain rule to take into account the operations performed in the projection. Details of this slight modification can be found in [30, Section 2.2].

3.1.3. Adaptation of the Frank-Wolfe Algorithm

The other method that improves on the initial proposal is to make use of the Frank-Wolfe algorithm, described in Definition 1.25. The authors of [4] detail the adaptation of this algorithm to the archetype calculation problem.

First, recall that it was necessary to compute the best feasible *feasible direction* at each step. It turns out that in the simplex case this computation is very straightforward. As shown in [9], the direction of maximum decrement is always towards the vertex of the simplex that minimizes the gradient.

That is, taking the canonical basis $\{e(i) : e(i)_i = 1 \wedge e(i)_j = 0 \ i \neq j\}$, the feasible direction at step k is $e(i')$, with $i' = \text{argmin}_i(\nabla f(x_k))_i$.

With this, we can define the proposed algorithm. First, as described in [9], we initialize the vector to be optimized to one of the vertices of the simplex. In our case, we will initialize all rows of the matrix we are optimizing to $e(1)$. Then, if we want to optimize the matrix A with α_i rows, the procedure is as follows:

1. $\alpha_i = e(1)$, for all i.
2. Recompute ∇_A.
3. For each $i \in \{1, \ldots, n\}$:
   (a) $j = \text{argmin}_j(\nabla_A)_{ij}$.
   (b) $\alpha_i = \alpha_i + \frac{2}{t+2}(e(j) - \alpha_i)$.
4. If the updates are sufficiently small or the iteration limit has been reached, terminate. If not, go back to step 2.

The procedure is the same for the optimization of the B matrix. Following the alternating optimization method, we get to minimize RSS.

3.2. Comparison of the methods

To get an idea of the higher efficiency of the methods described in the previous section with respect to the original method, computational tests have been performed for various input data sizes. For this purpose, the three algorithms have been implemented in both R and Python. Such implementation can be found in the public repository [10]. The timing tests presented below have been performed with the Python implementation, since the execution times of the algorithms are between two and three times faster than in R. Several tests have been run to determine the cause, reaching the conclusion that it is the multiplication of matrix strings where the biggest differences (up to x2) are found. Apparently this is due to the fact that R uses by default a version of BLAS (Basic Linear Algebra Subprograms), the library in charge of matrix multiplication, not optimized, as discussed in [37].

The tests consisted of running the three algorithms for random matrices with all combinations of the following parameter values:

- $n_samples$ (n, number of samples) $\in \{100, 1000, 10000\}$.
- $nfeatures$ (m, number of features) $\in \{5, 10, 25, 50, 50\}$.
- $archetypes$ (p, number of archetypes) $\in [1, 10]$

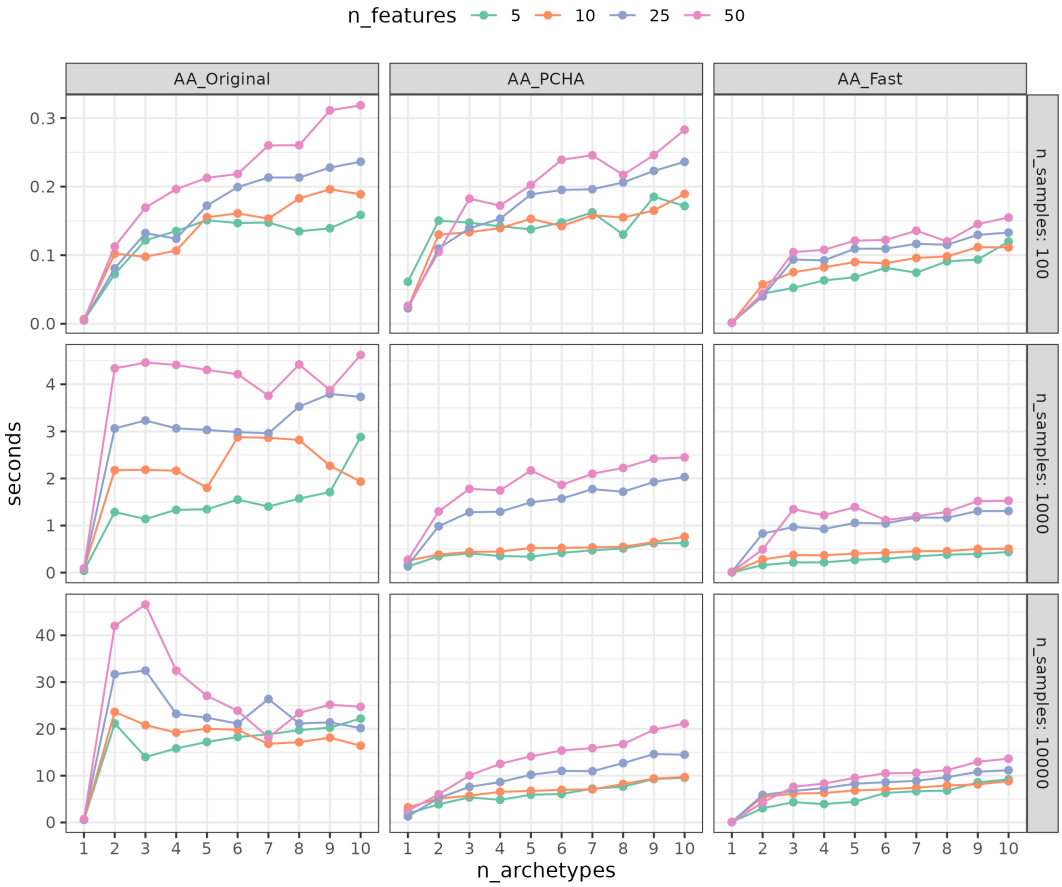


Figure 3.1: Execution time comparison.

In Figure 3.1 we observe that the original algorithm is very inefficient compared to the improved algorithms, showing then that imposing the convex constraints through penalty is not a good choice. As for the two improved algorithms, it is evident that the projected gradient version AA_PCHA is slower in all cases than the Frank-Wolfe based version AA_Fast. As we hinted in the definition of both methods, for certain sets the projection could be more costly than finding feasible directions. It seems clear that this is the case for the simplex Δ.

While runtime is something to consider, it is also necessary that these methods converge to a good solution. Therefore, we next evaluate the evolution of RSS (normalized by the number of data) for the same experiments as we have done for time.

First, we observe in Figure 3.2 how, in all algorithms and for all sizes, the RSS decreases as we increase the number of archetypes. This is logical, since the more

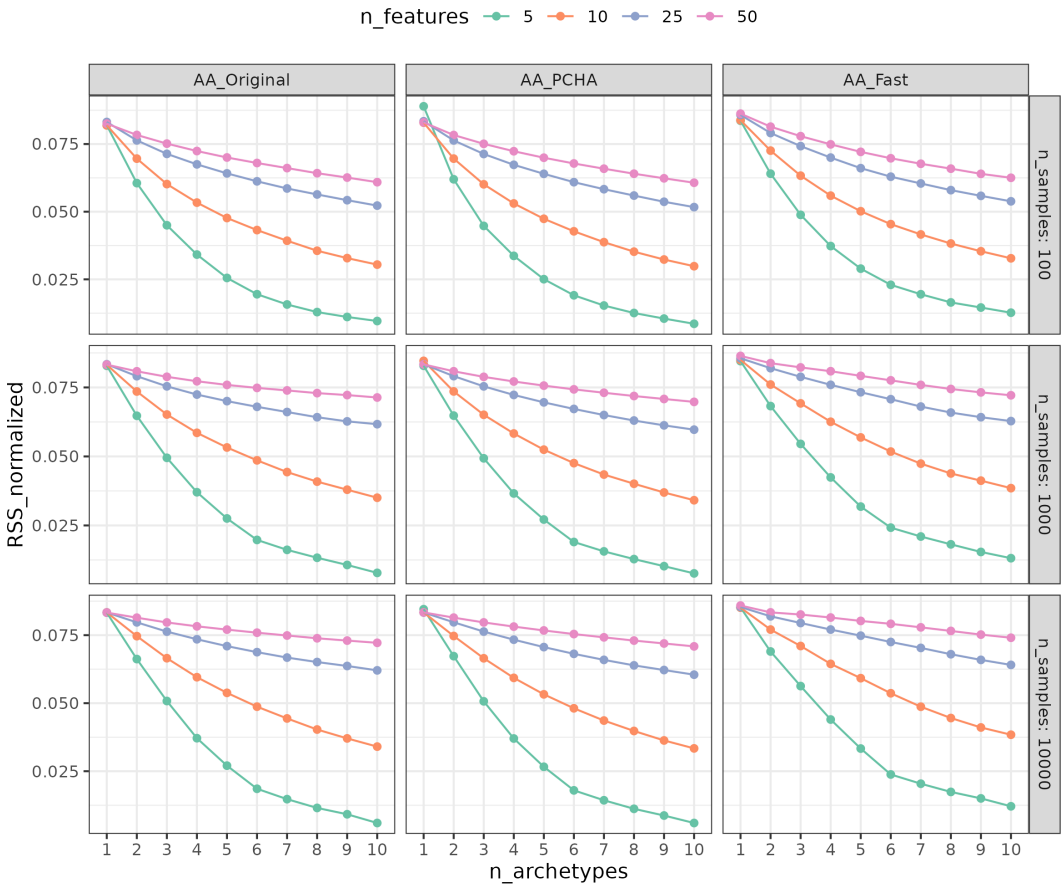


Figure 3.2: Comparison of RSS normalized.

archetypes we have, the better we will be able to approximate the convex hull of the set. On the other hand, we also see how, in general, the Frank-Wolfe based version *AA_Fast* achieves slightly worse results (higher RSS). We can attribute this to the fact that, while it is faster to save the projection, considering only *feasible directions* during optimization limits the search for minima. However, the observed difference is nowhere near as significant as the difference in execution times (see Figure 3.1).

In light of the presented results, we can consider *AA_Fast* as the most convenient computational method. In fact, the archetypes of the various examples presented in the next chapter have been obtained with this algorithm.

CHAPTER 4
Real examples of archetype analysis

Once we have described the basis that supports them and verified which is the most efficient way to obtain them, we are going to check the advantages that archetypes offer against other techniques in real examples.

4.1. Facial recognition

One problem we can try to solve with archetype analysis is that of facial recognition. This is based on, given two images, determining whether they are the same person. Unsupervised techniques such as PCA have been used for this purpose, for example in [43, 42]. The use of these methods seeks to define a space in which to represent the pair of images in order to compare their respective projections. In the case of the three techniques we are going to test, each one will construct its space based on a series of principal images, with a view that any image can be approximated as a combination of these.

We will now check the selection of 25 main images proposed by each of the three methods to be tested: PCA, k-means and archetype analysis. To do so, we will run the three algorithms on the data set [23, 36], which contains more than thirteen thousand face images as seen in Figure 4.1

Figure 4.1: Examples of images contained in the dataset.

First, in Figure 4.2 we see the results of PCA. The images shown are the top 25 components. The components collect information about the features that most

characterize the faces, such as the nose, mouth and eyes. However, as can be seen, the faces obtained are no longer intelligible and therefore interpretable. It is not denied that this representation is not appropriate to solve the problem, but it is clear that we lose control over the interpretability of the results.

Figure 4.2: Results of running PCA on the dataset.

Second, in Figure 4.3 we see the results of k-means. This time, the faces look more human-like than in the previous case. However, as is always the case when calculating the mean of data, this is a "smoothed" representation of the data. Thus, since we are obtaining the average faces of the population, we lose information about the details. In addition, k-means does not obtain the main images (centroids) with the objective that their combination approximates the rest of the images as well as possible, as PCA and archetype analysis do.

Finally, in Figure 4.4 we see the results of the archetype analysis. In contrast to the previous two methods, this time we do observe perfectly human faces, albeit somewhat disfigured. We can clearly observe different facial features among them: open and closed mouths, bushy and thin eyebrows, mustaches, glasses, etc. As with

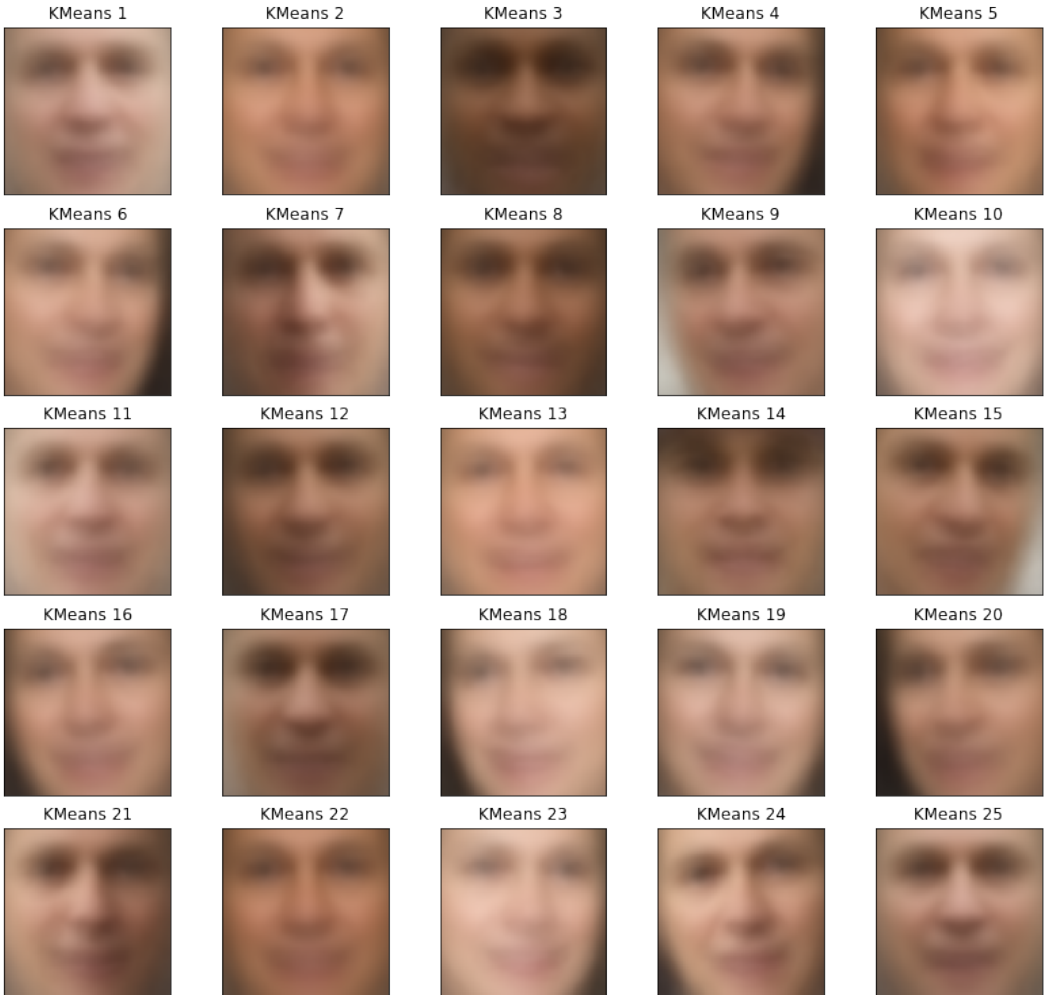


Figure 4.3: Results of running k-means on the dataset.

PCA, the main images have been obtained so that their combination approximates the rest of the images. But, in addition, as we commented in Section 2.1 of motivation, the fact that these combinations are convex makes them interpretable. To illustrate this fact, a function has been developed that plots the distribution of weights obtained over the archetypes for a given data. A couple of examples can be seen in Figure 4.5. These plots give us information on how much the input data resembles each of the archetypes, and we can then interpret the decisions made by our face detection model. This is because, after obtaining the spaces defined by the main images, the model will determine that they are similar faces if the set of weights expressing each face as a combination of the main images is similar. As we have seen, archetype analysis is the only method that allows us to interpret both the main images and the distribution of weights obtained on them.

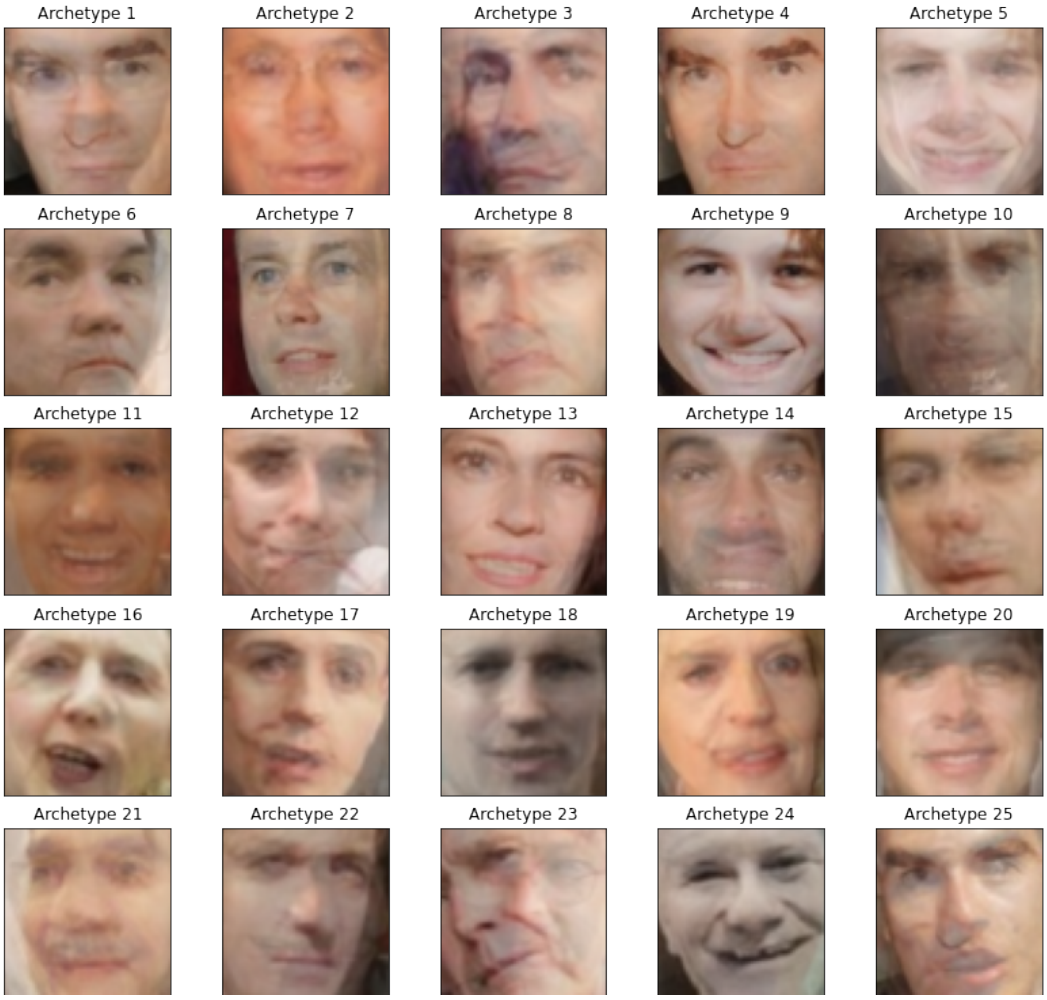
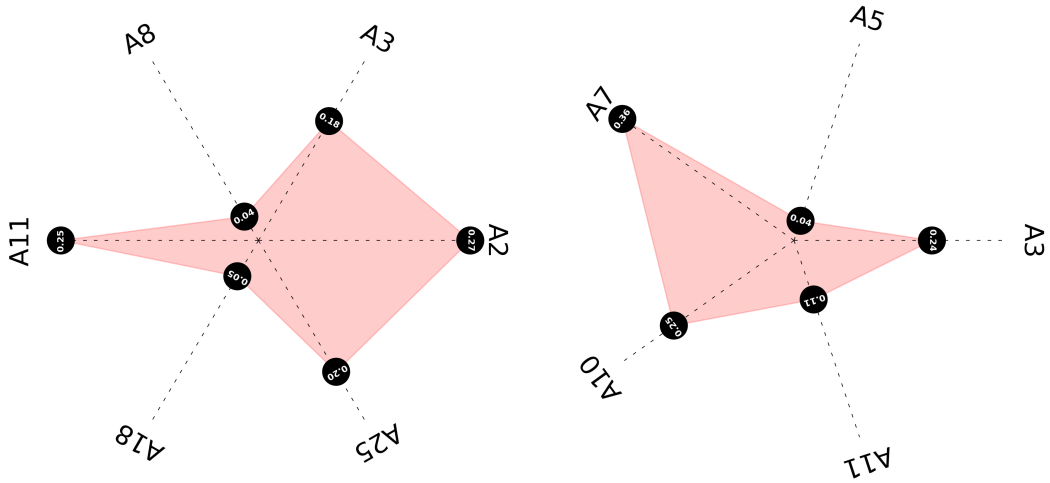


Figure 4.4: Results of running archetype analysis on the dataset.

4.2. Customer segmentation

Another example where we can observe the advantages of archetypes analysis is in the customer segmentation problem. This consists of grouping a company's customers who have similar behaviors in order to use this information to create targeted marketing campaigns, improve the design of specific products or identify consumer trends. In this case, we will apply k-means and archetype analysis on the [32] dataset. After cleaning the data, the following variables have been obtained:

- *Age*: age.
- *Education_Level*: integer between 0 and 4 representing the education level.
- *Has_Partner*: binary variable indicating whether the client has a partner.



(a) Weights for Face 2 of 4.1

(b) Weights for Face 4 of 4.1

Figure 4.5: Weights distributions obtained on the principal images 4.4

- *Has_Children*: binary variable indicating whether the client has children.
- *Income*: client's annual income in dollars.
- *Spending*: total amount spent in dollars.
- *Food*: total amount spent on food in dollars.
- *Wine*: total amount spent on wine in dollars.
- *Gold*: total amount spent on gold in dollars.
- *Discounts*: number of purchases made at a discount.
- *Seniority*: number of days of seniority as a customer.
- *Recency*: number of days since the customer's last purchase.

In Figure 4.6 we can see the three archetypes and the three centroids obtained after running their respective algorithms on the data. We will refer to each of them by its index preceded by A if it is archetype or K if it is centroid. Since the units of measurement are different in each case, the percentiles obtained for each variable are presented.

We note certain similarities between the results, although also some key differences. First, it appears that both algorithms have obtained very similar representatives for the demographic variables. Thus, $A1$ and $K1$ are practically equivalent in all variables, representing middle-aged and educated customers, with a partner but no children, with high income and expenses. On the other hand, in demographic terms, $A2$ seems to be related to $K3$, and $A3$ to $K2$. In the first case, we find a client who is younger than average but with a medium-high level of education, with a partner and children.

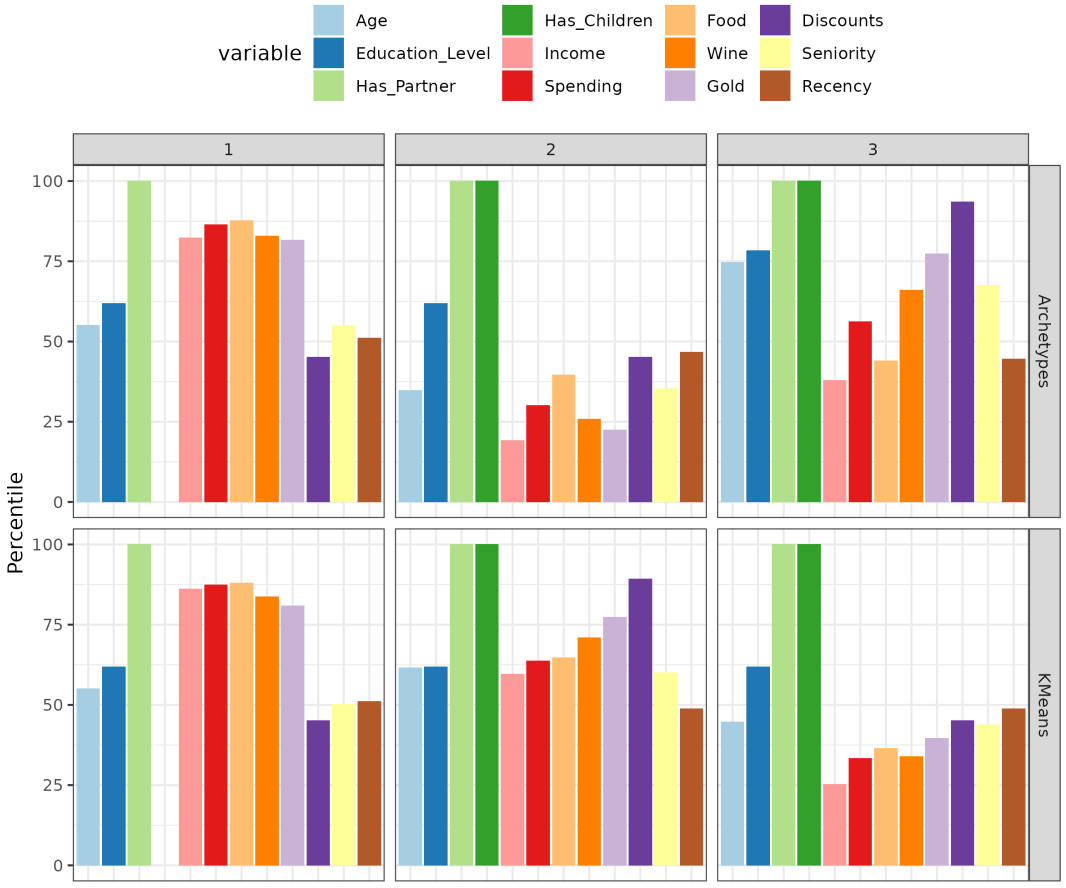
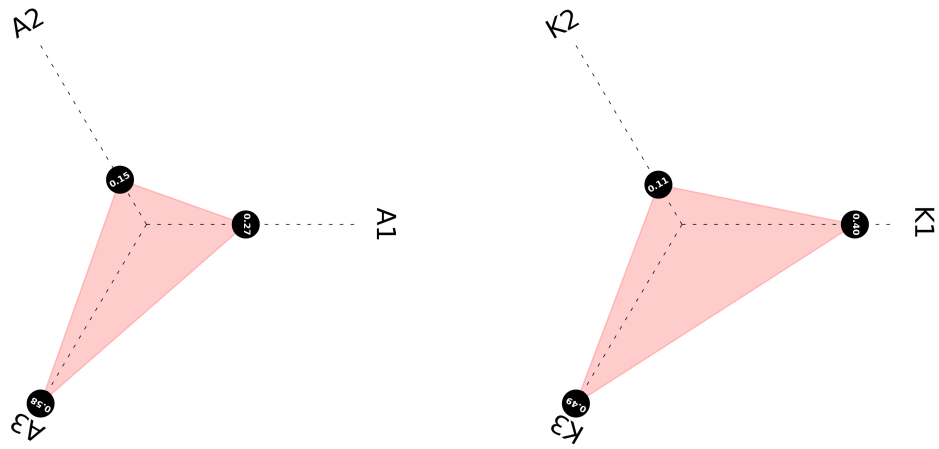


Figure 4.6: Results of running archetype analysis and k-means.

Due to their youth, income and expenses are low. In the second case, they are older people, also with a partner and children, and with greater economic power. We begin to observe in both cases how the archetypes contain more extreme values, which is undoubtedly an advantage to be able to really distinguish behaviors between groups. This is evident in the consumption variables. While all of them are very similar in each centroid, in the archetypes we observe important differences that provide value. For example, $A2$, due to its younger age and therefore lower budget, focuses its spending mostly on food. On the other hand, $A3$, of older age and economic power, has much higher expenditures on less indispensable items such as wine and gold. Such relevant conclusions cannot be obtained in the case of centroids, since, as in the example of the images, the mean smooths out the details.

Archetypes not only allow us to better capture important details, but also, as we have already discussed, give us the option to obtain reliable and interpretable similarity percentages with other data through convex combinations. This would allow marketing teams to determine the profile, or combinations of profiles, to which a new company customer belongs; allowing them to customize the products offered or strategies applied to them. In Figure 4.7 we show the weights obtained by minimizing



(a) Weights on archetypes

(b) Weights on centroids

Figure 4.7: Weights distributions obtained for a random data from [32]

the convex combination that best approximates a random sample data with respect to archetypes and centroids. We observe that the weights of 4.7(b) are smoother than those of 4.7(a), possibly due, once again, to the smoothing of the mean implicit in the centroids. Last but not least, the error of the k-means approximation is 11% larger than that of the archetypal approximation.

CHAPTER 5
Conclusions and future work

Throughout this paper, we have gained a deep understanding of archetypal analysis, both its theoretical foundations and its advantages over other similar methods in real situations. Along the way, we have also explored various convex optimization algorithms that have led us to improve the original proposal by far.

In view of the potential offered by this technique, it is worth commenting on possible lines of future work. Firstly, there are already works that extend the analysis of archetypes from various perspectives. On the one hand, [26] combines archetypal analysis with deep neural networks. [39] proposes to use distributional assumptions on the observed vector to find the archetypes; while [38] adapts archetypal analysis to another type of data, in this case qualitative. On the other hand, [12] adds robustness to the calculation method by modifying the cost function, thus reducing the effect of *outliers*. Related to this, [8] uses the projections to the space generated by the convex hull of archetypes to detect *outliers*.

It is clear that multiple lines of research have emerged from the concept of archetypal analysis. From our side, the following is proposed. While convexity provides us with good properties for defining archetypes, taking for granted that the support of our data is a convex set may be a too demanding an assumption. Seeking to relax this assumption, we aim to explore more general ways of defining archetypes that allow us to approximate more general supports. To this end, we propose two possible starting points. First, one can explore the concept of α-convex sets, a generalization of convexity. In particular, we would try to define generalized archetypes in a way that approximates the α-convex hull, proposed in [31]. On the other hand, [18] describes a local approximation of the convex hull (*Local convex hull*) by nearest neighbors that would also be interesting to consider. In any case, the main pending task of the generalization we propose is to define an error function that measures the quality of the approximation generated by a given set of archetypes, now that we do not have convex combinations and closures at our disposal. By minimizing this error function, we would obtain a number of points that would allow us to understand more general supports of our data.

Bibliography

[1] Angel, T. Convex functions. http://www.math.udel.edu/~angell/Opt/conv_fcn.pdf, Accedido 2022-01-03.

[2] Arora, S. Cos 521 advanced algorithm design: Lecture19. https://www.cs.princeton.edu/courses/archive/fall13/cos521/lecnotes/lec19.pdf, Accedido 2022-01-03, 2013.

[3] Atkinson, A. C., Riani, M., and Cerioli, A. Swiss heads R dataset. https://www.rdocumentation.org/packages/fsdaR/versions/0.4-8/topics/swissheads, Accedido 2022-01-07.

[4] Bauckhage, C., Kersting, K., Hoppe, F., and Thurau, C. Archetypal analysis as an autoencoder. In *Workshop New Challenges in Neural Computation* (2015), pp. 8–16.

[5] Bauckhage, C., and Thurau, C. Making archetypal analysis practical. In *Pattern Recognition, 31st DAGM Symposium* (09 2009), Springer, pp. 272–281.

[6] Berrendero, J. R. Investigación operativa: Conjuntos convexos. https://verso.mat.uam.es/~joser.berrendero/cursos/Matematicas-IO/io-tema2-16.pdf, Accedido 2021-11-20.

[7] Berrendero, J. R. Investigación operativa: Funciones convexas y optimización convexa. https://verso.mat.uam.es/~joser.berrendero/cursos/Matematicas-IO/io-tema4-16.pdf, Accedido 2022-01-03.

[8] Cabero, I., Epifanio, I., Piérola, A., and Ballester, A. Archetype analysis: A new subspace outlier detection approach. *Knowledge-Based Systems 217* (2021), 106830.

[9] Clarkson, K. L. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Trans. Algorithms 6*, 4 (sep 2010).

[10] Cobo, G. G. Archetypal analysis implementation. https://github.com/atmguille/archetypal-analysis, Accedido 2022-05-17.

[11] Cutler, A., and Breiman, L. Archetypal analysis. *Technometrics 36*, 4 (1994), 338–347.

[12] Eugster, M. J., and Leisch, F. Weighted and robust archetypal analysis. *Computational Statistics & Data Analysis 55*, 3 (2011), 1215–1225.

[13] Eugster, M. J. A., and Leisch, F. From Spider-Man to Hero – archetypal analsis in R. *Journal of Statistical Software 30*, 8 (2009), 1–23.

[14] Eugster, M. J. A., Leisch, F., and Seth, S. archetypes: Archetypal Analysis R package. https://cran.r-project.org/package=archetypes, Accedido 2022-01-07, 2019.

[15] Fernandez-Granda, C. Optimization-based data analysis: Convex optimization. https://cims.nyu.edu/~cfgranda/pages/OBDA_fall17/notes/convex_optimization.pdf, Accedido 2022-01-03, 2017.

[16] Flury, B., and Riedwyl, H. *Multivariate Statistics: A Practical Approach*. Chapman & Hall, Ltd., London, 1988.

[17] Frank, M., and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly 3* (1956), 95–110.

[18] Getz, W., and Wilmers, C. A local nearest-neighbor convex-hull construction of home ranges and utilization distributions. *Ecography 27* (08 2004), 489–505.

[19] Gordon, G. Optimization: Convex sets. https://www.cs.cmu.edu/~ggordon/10725-F12/scribes/10725_Lecture3.pdf, Accedido 2021-11-20, 2012.

[20] Gorski, J., Pfeuffer, F., and Klamroth, K. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research 66*, 3 (Dec 2007), 373–407.

[21] Gu, Q. Optimization: Lecture 7. https://piazza.com/class_profile/get_resource/is58gs5cfya7ft/it3isd93pmp5ft#:~:text=For%20convex%20function%2C%20we%20can,is%20also%20a%20global%20minimum.&text=Theorem%201%20(Local%20Minimum%20is,over%20a%20convex%20set%20D, Accedido 2022-01-03, 2016.

[22] Hewson, P. Swiss heads R dataset. https://rdrr.io/cran/Flury/man/swiss.heads.html, Accedido 2022-01-07.

[23] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Tech. Rep. 07-49, University of Massachusetts, Amherst, October 2007.

[24] Jaggi, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning* (Atlanta, Georgia, USA, 17–19 Jun 2013), S. Dasgupta and D. McAllester, Eds., vol. 28 of *Proceedings of Machine Learning Research*, PMLR, pp. 427–435.

[25] jpayansomet. How one can draw a convex function? https://tex.stackexchange.com/questions/394923/how-one-can-draw-a-convex-function, Accedido 2022-01-03, 2017.

[26] Keller, S. M., Samarin, M., Wieser, M., and Roth, V. Deep archetypal analysis. In *Pattern Recognition* (Cham, 2019), G. A. Fink, S. Frintrop, and X. Jiang, Eds., Springer International Publishing, pp. 171–185.

[27] Lavrov, M. Math 484 nonlinear programming: Convexity. https://faculty.math.illinois.edu/~mlavrov/docs/484-spring-2019/ch2lec1.pdf, Accedido 2021-11-20, 2019.

[28] Lawson, C. L., and Hanson, R. J. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[29] Mathar, R., Alirezaei, G., Balda, E., and Behboodi, A. *Fundamentals of Data Analytics With a View to Machine Learning*. Springer, 2020.

[30] Mørup, M., and Hansen, L. K. Archetypal analysis for machine learning and data mining. *Neurocomputing 80* (2012), 54–63. Special Issue on Machine Learning for Signal Processing 2010.

[31] Pateiro-López, B., and Casal, A. Generalizing the convex hull of a sample: The r package alphahull. *Journal of Statistical Software 34* (04 2010), 1–28.

[32] Patel, A. Customer personality analysis. analysis of company's ideal customers. https://www.kaggle.com/imakash3011/customer-personality-analysis, Accedido 2022-03-03, 2021.

[33] Petersen, K. B., and Pedersen, M. S. The matrix cookbook, Oct. 2008. Version 20081110.

[34] Pérez, J. Geometría: Conjuntos convexos. https://www.ugr.es/~jperez/docencia/GeomConvexos/cap1.pdf, Accedido 2021-11-20.

[35] RockyRock. Using pfg plots to plot unit simplex in 3 dimensions. https://tex.stackexchange.com/questions/251264/using-pfg-plots-to-plot-unit-simplex-in-3-dimensions, Accedido 2021-11-20, 2018.

[36] Sanderson, C., and Lovell, B. C. Multi-region probabilistic histograms for robust and scalable identity inference. In *Advances in Biometrics* (Berlin, Heidelberg, 2009), M. Tistarelli and M. S. Nixon, Eds., Springer Berlin Heidelberg, pp. 199–208.

[37] Santillan, C. Improving R perfomance by installing optimized BLAS/LAPACK libraries. https://csantill.github.io/RPerformanceWBLAS/, Accedido 2022-02-25, 2018.

[38] Seth, S., and Eugster, M. J. Archetypal analysis for nominal observations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 5 (2016), 849–861.

[39] Seth, S., and Eugster, M. J. A. Probabilistic archetypal analysis. *Machine Learning 102*, 1 (Jan 2016), 85–113.

[40] Simon, B. *Convexity: An Analytic Viewpoint (Cambridge Tracts in Mathematics)*. Cambridge University Press, Cambridge, 2011.

[41] Torbjørn. Alignment of tikz pictures in subfigures. https://tex.stackexchange.com/questions/302589/alignment-of-tikz-pictures-in-subfigures, Accedido 2021-11-20, 2016.

[42] Turk, M., and Pentland, A. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience 3*, 1 (01 1991), 71–86.

[43] Çarikçi, M., and Özen, F. A face recognition system based on eigenfaces method. *Procedia Technology 1* (2012), 118–123. First World Conference on Innovation and Computer Sciences (INSODE 2011).