Modeling high dimensional point clouds with the spherical cluster model

Anonymous authors
Paper under double-blind review

Abstract

A parametric cluster model is a statistical model providing geometric insights onto the points defining a cluster. The *spherical cluster model* (SC) approximates a finite point set $P \subset \mathbb{R}^d$ by a sphere $S(c, r)$ as follows. Taking r as a fraction $\eta \in (0, 1)$ (hyper-parameter) of the std deviation of distances between the center c and the data points, the cost of the SC model is the sum over all data points lying outside the sphere S of their power distance with respect to S. The center c of the SC model is the point minimizing this cost. Note that $\eta = 0$ yields the celebrated center of mass used in KMeans clustering. We make three contributions.

First, we show fitting a spherical cluster yields a strictly convex but not smooth combinatorial optimization problem. Second, we present an exact solver using the Clarke gradient on a suitable stratified cell complex defined from an arrangement of hyper-spheres. Finally, we present experiments on a variety of datasets ranging in dimension from $d = 9$ to $d = 10,000$, with two main observations. First, the exact algorithm is orders of magnitude faster than BFGS based heuristics for datasets of small/intermediate dimension and small values of η, and for high dimensional datasets (say $d > 100$) whatever the value of η. Second, the center of the SC model behave as a parameterized high-dimensional median.

The SC model is of direct interest for high dimensional multivariate data analysis, and the application to the design of mixtures of SC will be reported in a companion paper.

Keywords: subspace clustering, spherical clusters, centerpoints, medians, non smooth optimization.

1 Introduction

Clustering methods. Clustering, namely the task which consists in grouping data items into dissimilar groups of similar elements, is a fundamental problem in data analysis at large Xu & Wunsch (2005). Existing clustering methods may be ascribed to four main tiers, including hierarchical clustering Duda & Hart (1973), density based clustering Cheng (1995); Chazal et al. (2013), spectral clustering Von Luxburg (2007), as well as k-means Arthur & Vassilvitskii (2007) and variants related to Expectation-Maximization Dempster et al. (1977); Kasarapu & Allison (2015). The goal of clustering being to aggregate points into coherent groups, equally important are *cluster models* aiming at providing insights into the geometry of the data. This goal prompted the development of `k-subspaces` clustering techniques, which belong to two tiers. Sparse subspace clustering (SSC/ASSC) Elhamifar & Vidal (2013); Soltanolkotabi & Candes (2012); Li et al. (2018) reconstruct points as (sparse) linear combinations of neighbors, while parametric cluster model use explicit analytical form for clusters Parsons et al. (2004); Wang et al. (2009).

Geometric medians. Cluster models providing insights on the geometry of a point set also call for a discussion of high dimensional medians. The Fermat-Weber point is the point from $\mathbb{R}^d$ minimizing the sum of Euclidean distances to all data points. Unfortunately, this point is hard to compute and unstable Kupitz & Martini (1997); Weiszfeld (1937); Bajaj (1988); Cohen et al. (2016). Building on Helly's theorem, a median can be defined as any point whose Tukey depth is at least $\geq n/(d+1)$ Tukey (1975). Such a point is, however, hard to compute Clarkson et al. (1993); Miller & Sheehy (2009). The projection median is defined by projecting the dataset onto random lines, computing the univariate median for each projection, and computing a weighted average of the data points responsible for these univariate medians Durocher & Kirkpatrick (2009); Basu et al. (2012); Durocher et al. (2017). It is an elegant, stable and remarkably effective generalization of the univariate median.

Contributions. Two types of parametric cluster models have been proposed recently Wang et al. (2009): affine and spherical clusters (SC). The former accommodates potentially unbounded (large) clusters of arbitrary dimension. The latter defines compact (spherical) clusters based on the power distance of points with respect to a sphere whose radius is (a fraction of) the variance of distances to the cluster center–to be determined. However, the uniqueness of the SC center is not established, and no algorithm is presented to compute it. (The calculation presented assumes the center is known, and it solely observes that the result obtained is consistent with the usual center of mass when the fraction of the variance tends to zero Wang et al. (2009).)

We make two contributions. First, we establish the SC cluster model is well posed (the solution is unique) and also well poised (numerically tractable). Second, we present experiments showing that the center of the SC model behave as a parameterized high-dimensional median.

The application of SC embedded into affine subspaces of positive codimension will be presented in a companion paper.

All proofs and detailed algorithms are provided in the Supporting Information.

2 Parametric cluster models and Subspace Embedded Spherical Clusters

2.1 Notations

Let D be a set of n points in $\mathbb{R}^d$. We consider a partition of D into k clusters $C_1, \ldots, C_k$, with D_ℓ the set of points associated to cluster C_ℓ. The unbiased variance estimate for distances within cluster D_ℓ of center c_ℓ satisfies

$$\hat{\sigma}^2(c_\ell) = \frac{1}{n-1} \sum_{x_i \in D_\ell} \|x_i - c_\ell\|^2 \tag{1}$$

Let $A = c + V$ be an affine space, with c a point in $\mathbb{R}^d$ (think cluster center), and V a vector space. For any point $x \in \mathbb{R}^d$, we denote by $(x-c)_{\|V}$ the orthogonal projection of the vector $(x-c)$ onto V, and by $(x-c)_{\perp V}$ the orthogonal projection on $V^\perp$.

When fitting a model, the sum of squared distances from samples to the model is called the *residual sum of squares (RSS)*, or dispersion for short.

2.2 Parametric cluster models

Take C_ℓ for $\ell \in [\![1, k]\!]$ and suppose D_ℓ is known. Cluster C_ℓ is described by the parameter set $\theta_\ell = (\theta_{\ell,1}, ..., \theta_{\ell,r})$ and a function $d_\ell : (x, C_\ell(\theta_\ell)) \mapsto d_\ell(x, C_\ell(\theta_\ell))$, that is some distance from a point to the cluster. We call the description of C_ℓ by the function d_ℓ a *parametric cluster model*. We decompose the clustering problem into two sub-problems concerned with the minimization of a dispersion term based on squared distances:

Problem. 1 (Cluster optimization)**.** *Let C_ℓ be a parametric cluster.* Cluster optimization *is the optimization problem seeking the cluster parameters minimizing the dispersion*

$$\min_{\theta_\ell} \Phi_l, \text{ with } \Phi_l = \sum_{x \in D_\ell} d_\ell(x, C_\ell(\theta_\ell))^2. \tag{2}$$

Affine clusters. As a first generalization of `k-means`, one can consider the distance from a data point to an affine subspace, yielding `k-subspaces` clustering Wang et al. (2009):

Definition. 1 (Subspace cluster)**.** *Let $A = c + V$ be some affine subspace of $\mathbb{R}^d$ where $c \in \mathbb{R}^d$ is a point and V is an m-dimensional linear subspace. The* subspace cluster *$C_\ell(A)$ is a cluster, where the distance from a point x to the cluster is the distance to the subspace :*

$$d(x, C_\ell(A))^2 := d(x, A)^2 = \|(x-c)_{\perp V}\|^2 . \tag{3}$$

Spherical clusters. As noticed in Introduction, affine clusters may be confounded by noise, and suffer from their non compact nature. This latter aspect can be taken care of using *spherical clusters*. To see how, recall that the power of a point x with respect to a sphere $S(c, r)$ is defined by $\pi(x, S) = \|x - c\|^2 - r^2$. Following Wang et al. (2009), we define:

Definition. 2 (Spherical cluster)**.** *Let $\eta \in]0, 1[$ be is a hyperparameter, and let c_ℓ be a point called the* cluster center*. Given the set D_ℓ, the distance function associated to the* spherical cluster *$C_\ell(c_\ell)$ reads as*

$$d(x, C_\ell(c_\ell))^2 := \max\left(0, \|x - c_\ell\|^2 - \eta\hat{\sigma}^2(c_\ell)\right) = \max\left(0, \pi(x, S(c_\ell, \sqrt{\eta}\hat{\sigma}(c_\ell)))\right). \tag{4}$$

The rationale of this definition is that one wishes to find the center minimizing the cost of outliers–points outside the spherical cluster.

2.3 Subspace-embedded spherical cluster

We note that *subspace-embedded spherical clusters* can be defined for affine spaces of arbitrary co-dimension:

Definition. 3 (Subspace-embedded spherical cluster)**.** *Given a set of points D_ℓ, we define a* subspace-embedded spherical cluster *as a spherical cluster of center c embedded into an affine subspace V of $\mathbb{R}^d$. The corresponding distance reads as*

$$d_{SESC}(x, C_\ell)^2 = \|(x-c)_{\perp V}\|^2 + \mu \max\left(0, \left\|(x-c)_{\|V}\right\|^2 - \eta\frac{1}{n-1}\sum_{x_i \in D_\ell}\left\|(x_i - c)_{\|V}\right\|^2\right) \tag{5}$$

For a fixed data set D_ℓ, the sum of the previous quantities yields the dispersion to be minimized.

3 Spherical Cluster Optimization

This section studies problem 1 for the spherical cluster model (Def. 2). Since η is fixed, we simply denote F_η as F.

3.1 Functional decomposition and geometry of the sub-functions

For a fixed data set D_ℓ, we aim at minimizing over $\mathbb{R}^d$ the map

$$F_\eta(c) := \sum_{x_i \in D_\ell} \max\left(0, \|x_i - c\|^2 - \frac{\eta}{n-1} \sum_{x_j \in D_\ell} \|x_j - c\|^2\right). \tag{6}$$

To study the previous function, for each $x_i \in D_\ell$, let

$$\tilde{f}_{\eta,x_i}(c) := \|x_i - c\|^2 - \eta \frac{1}{n-1} \sum_{x_j \in D_\ell} \|x_j - c\|^2. \tag{7}$$

so that

$$F_\eta(c) = \sum_{x_i \in D_\ell} \max(0, f_{\eta,x_i}(c)). \tag{8}$$

We first analyze the sub-functions and f_{η,x_i} in order to analyze the main function F_η. In the sequel, we assume that (i) the set D_ℓ is fixed, (ii) $x_i \in D_\ell$, (iii) $0 < \eta < 1 - 1/n$ (iv) η is fixed, so that we drop η from the notations (e.g writing F, f_{x_i} instead of F_η, f_{η,x_i} to ease notations).

Studying the function f_{x_i} benefits from the geometry of the following *sink region* yielding a null cost:

Definition. 4. *The* sink region B_{x_i} *is the set over which* f_{x_i} *does not contribute to* F*, that is* $B_{x_i} := f_{x_i}^{-1}(-\infty, 0]$*. We denote* S_{x_i} *its topological boundary.*

Remark that since $\eta < 1 - \frac{1}{n}$ the intersection B of all B_{x_i} is necessarily empty, as any x belong to this set would verify $\|x - x_i\|^2$ strictly lower than the average $\frac{1}{n}\sum_{x_i \in D_\ell} \|x - x_i\|^2$.

The following results from an elementary calculation.

Lemma. 1 (Geometry of B_{x_i})**.** *Let* $\eta' = \frac{n-1}{n}\eta$*. Each map* f_{x_i} *is proportional to a spherical power, and takes the form*

$$f_{x_i}(c) = (1 - \eta')\left(\|c - c_i\|^2 - R_i^2\right) \tag{9}$$

Putting $\bar{x} := \frac{1}{n}\sum_{x_j \in D_\ell} x_j$*, the center* c_i *and the radius* R_i *of said sphere satisfy the following.*

$$\begin{cases} c_i &= \frac{x_i - \eta'\bar{x}}{1-\eta'}, \\ R_i^2 &= \left\|\frac{x_i - \eta'\bar{x}}{1-\eta'}\right\|^2 - \frac{\|x_i\|^2 - \frac{\eta'}{n}\sum_{x_j \in D_\ell}\|x_j\|^2}{1-\eta'} \end{cases} \tag{10}$$

As a consequence the sink region B_{x_i} *is a non-empty closed ball of* $\mathbb{R}^d$*, and* S_{x_i} *is its associated sphere.*

As an immediate corollary, $\max(0, f_{x_i})$ is a convex map. In $\mathbb{R}^d \setminus B_{x_i}$, it is quadratic with gradient $\nabla f_{x_i}(c) = 2(1 - \eta')(c - c_i)$, while being identically zero inside B_{x_i}.

3.2 Arrangement of hyper-spheres underlying the objective function

Our analysis of the sub-functions f_{x_i} shows that they are continuous and *piecewise quadratic*: zero on B_{x_i}, and a quadratic form on $\mathbb{R}^d \setminus B_{x_i}$. Thus, F is also continuous, convex and piecewise quadratic. Finding the optimal cluster center requires understanding the relationship between all sink regions.

Arrangement. An *arrangement* of hyper-surfaces is a decomposition of $\mathbb{R}^d$ into equivalence classes of points using their position with respect to these hyper-surfaces Halperin & Sharir (2017). We apply this concept to the spheres bounding the sink regions (Lemma 1).

For $x \in \mathbb{R}^d$ and $i \in [\![1, n]\!]$, consider the following signature which states whether point x lies outside/on/inside the spheres S_{x_i}. It is a length n vector with one entry in $\{-1, 0, 1\}$ for each sink-defining ball:

$$\sigma(x) := (sign(f_{x_1}(x)), \dots, sign(f_{x_n}(x))). \tag{11}$$

The signature defines an equivalence relation, where two points are equivalent if they have the same signature. We call *cells* the equivalence classes, and we use the notation $\mathcal{C}$ to denote them. By definition, cells are non-empty and characterized by the three-set partition $I^+(\mathcal{C}), I^0(\mathcal{C}), I^-(\mathcal{C})$ of $[\![1, n]\!]$, where the sets are defined respectively as the sets of indices i where f_{x_i} are positive, zero, and negative. Note that generically, $\tau+1 \leq d$ spheres in dimension d intersect along an $l = d - (\tau + 1)$ sphere; thus we let the *dimension* of a cell $\mathcal{C}$ be the number $d - \#I^0(\mathcal{C})$. Cells of dimension d are said to be *fully dimensional* and are open subsets of $\mathbb{R}^d$, while others are said to be of *positive codimension*.

Combinatorial decomposition of F. On a cell $\mathcal{C}$, F is determined by the value of f_{x_i} where i ranges among $I^+(\mathcal{C})$. More precisely, we have

$$F_{|C}(c) = \sum_{i \in I^+(\mathcal{C})} f_{x_i}(c) = f_{I^+(\mathcal{C})}(c), \tag{12}$$

where $f_J := \sum_{i \in J} f_{x_i}$ for any subset J of $[\![1, n]\!]$. In the same vein, we put $S_J := \bigcap_{i \in J} S_{x_i}$ so that in a generic configuration of spheres any cell with non empty I^0 is a relatively open subset of S_{I^0}. We define c_J to be the center of mass of all c_i where i ranges among J.

$$c_{I^+(J)} := \frac{1}{\#J} \sum_{i \in J} c_i. \tag{13}$$

Putting $R_J^2 := \|c_J\|^2 + (\#J)^{-1} \left(\sum_{i \in J} R_i^2 - \|c_i\|^2 \right)$, straightforward computations yield

$$f_J(c) = (1 - \eta')\#J \left[\|c - c_J\|^2 - R_J^2 \right]. \tag{14}$$

Since any full-dimensional cell $\mathcal{C}$ is open, F is twice differentiable in $\mathcal{C}$ with gradient and Hessian as follows:

$$\begin{cases} \nabla F_{|\mathcal{C}}(c) = 2(1 - \eta')\#I^+(\mathcal{C}) \left(c - c_{I^+(\mathcal{C})} \right) \\ HF_{|\mathcal{C}}(c) = 2(1 - \eta')\#I^+(\mathcal{C})\mathrm{Id}. \end{cases} \tag{15}$$

3.3 Strict convexity and optimization

The spherical cluster optimization is well-posed. Indeed we have seen that $B = \bigcap_{x_i \in D_\ell} B_{x_i}$ is empty, and from previous computations the Hessian of F is almost-everywhere positive definite outside of B.

Theorem. 1 (Strict convexity of F)**.** *Let D_ℓ be a set of n points of $\mathbb{R}^d$, with at least two distinct points, and assume that $0 < \eta < 1 - \frac{1}{n}$. Then the associated F map is $2(1 - \eta')$-strongly convex on $\mathbb{R}^d$, and is a fortiori strictly convex. Its minimization problem admits exactly one solution in $\mathbb{R}^d$.*

Minimum of F on a full-dimensional cell. From Eq. (15), it is clear that the minimum of F is attained on a full-dimensional cell $\mathcal{C}$ if and only if its gradient vanishes, translating to the following characterization.

$$\operatorname*{argmin}_{\mathbb{R}^d} F \in \mathcal{C} \iff c_{I^+(\mathcal{C})} \in \mathcal{C}. \tag{16}$$

To find the minimum of F, it is not enough to check all full-dimensional cells as it is absolutely possible that the minimum be attained on a cell with positive codimension (see e.g.) Fig. 1, left.), where no closed-form for a minimum can be obtained.

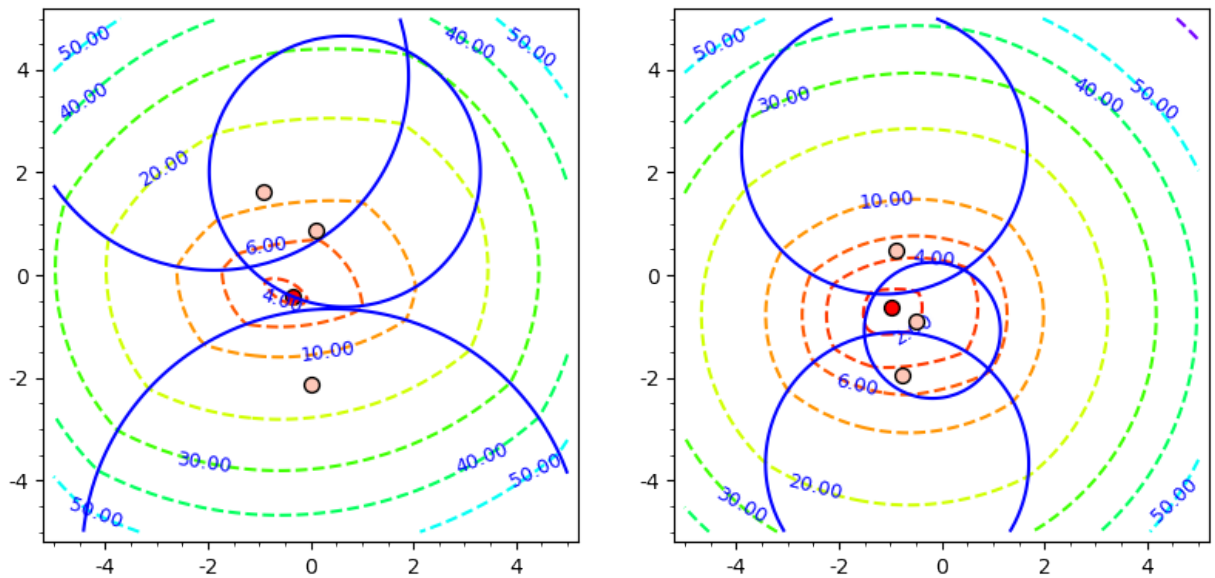


Figure 1: **Minima of F on cells of various dimensions.** Data point in orange, minima in red. Selected level sets (in dotted-lines) are also reported.

4 Optimization: computing the unique minimizer of F_η

Having established the strict convexity of F_η, we compute its unique minimizer. (As in the previous section, we simply denote F_η as F.) Our algorithm actually minimizes a function of the form $\sum_i \max(0, \|x - c_i\|^2 - R_i^2)$, and constructs a finite sequence of points (x_n) by induction. The last point is the optimum of F. It assumes infinite precision – the so called *real RAM model*, and also assumes that all points c_i are in a generic position and that when spheres S_i intersect. See Sections 4.4 and 4.5 for comments on these assumptions.

4.1 Subdifferential and generalized gradient

We face a non-smooth convex optimization problem without constraint. Guarantees on the speed of convergence of algorithms with such assumptions as a number of iterations are rather weak. While gradients exists almost-everywhere, the classical gradient descent method may get trapped to bottleneck situations Bubeck (2015) leading to a precision rate of $O\left(\frac{1}{\sqrt{T}}\right)$, where T is the number of steps. Further non-smooth investigation revolves around the use of the *subdifferential* or equivalently the *Clarke gradient* Clarke (1997) of the function. For a convex function, the subdifferential/Clarke gradient of F at x, denoted by $\partial_* F(x)$ is a convex set defined below, while the generalized gradient $\nabla_* F(x)$ is its element of least norm.

$$\begin{cases} \partial_* F(x) & := \{s, f(y) - f(x) \geq s \cdot (y - x), \forall y\} \\ \nabla_* F(x) & := \underset{u \in \partial_* F(x)}{\operatorname{argmin}} \|u\| . \end{cases} \tag{17}$$

Gradient samplings methods Burke et al. (2020) avoid the earlier described bottleneck configurations with a good descent direction obtained by approximating the generalized gradient. Given an arrangement of the space, the recent so-called stratified gradient sampling Leygonie et al. (2023) proposes to use the arrangement to efficiently determine a good descent direction. To tailor an exact algorithm, we use the structure at our disposal. Indeed, for any fixed x in $\mathbb{R}^d$, we let I^+, I^0 and I^- be the three-set partition associated to the cell of x. Then the subdifferential of F at x can be expressed as follows.

$$\partial_* F(x) = \{\nabla f_{I^+}(x) + \sum_{i \in I^0} \lambda_i \nabla f_i(x), 0 \leq \lambda_i \leq 1\}. \tag{18}$$

The generalized gradient can thus be expressed as a solution to the following quadratic programming (QP) problem, which admits a unique solution in λ when all c_i are in generic position – Sec. A:

$$\nabla_* F(x) = \underset{0 \leq \lambda_i \leq 1}{\operatorname{argmin}} \left\{ ||\nabla f_{I^+}(x) + \sum_{i \in I^0} \lambda_i \nabla f_i(x)||^2 \right\}. \tag{19}$$

Letting $(\alpha_i)_{i \in I^0}$ be the unique solution to this problem, we let [1]

$$\begin{cases} I^0_*(x) & := \{i \in I^0(x), 0 < \alpha_i < 1\} \\ I^+_*(x) & := I^+(x) \cup \{i \in I^0(x), \alpha_i = 1\} \\ I^-_*(x) & := I^-(x) \cup \{i \in I^0(x), \alpha_i = 0\}. \end{cases} \tag{20}$$

Moreover, we let $\mathcal{C}^*(x)$ be the cell with three-set partition $I^+_*(x), I^0_*(x), I^-_*(x)$.

Describing the semiflow of F The generalized gradient shines in the following proposition. Even though $\nabla_* F$ might not be continuous, by convexity of F from any starting point x there exists (see for instance Bolte et al. (2010); Marcellin & Thibault (2006)) a trajectory $t \mapsto x(t)$ (with $x(0) = x$) called a *semiflow* verifying almost-everywhere (for $t \in \mathbb{R}^+$):

$$x'(t) = -\nabla_* F(x(t)). \tag{21}$$

In particular $F(x(t))$ decreases at rate $\|\nabla F_*(x(t))\|^2$, and by strong convexity $x(t)$ reaches the argmin of F over $\mathbb{R}^d$ in a finite time, where it is stationary Marcellin & Thibault (2006); Josz (2023). Given the structure of our F, there are three possible behaviors for the semiflow with starting point x:

- If x is in a full dimensional cell, the semiflow starting from x begins by a segment heading towards $c_{I^+(x)}$, until it reaches a new cell or $c_{I^+(x)}$ which is the minimum.
- If x lies in a cell of positive codimension and $I^0_*(x)$ is empty, the semiflow enters the non-empty, full dimensional cell $\mathcal{C}^*(x)$ and follows a straight line in this cell until it meets a new cell, as described as above;
- Else, $I^0_*(x)$ is not empty. One can show that if the Clarke QP (Eq. 19) lies in what we call a non-degenerate position[2] if and only if for small t the semiflow enters the non-empty cell of positive codimension $\mathcal{C}^*(x)$, which is a subset of $S_{I^0_*(x)}$. Points x with a degenerate QP problem are of measure zero, however points x such that the trajectory $x(t)$ reaches a degenerate QP position, making the semiflow intractable, are not.

4.2 Exact algorithm

We develop algorithm `SC-Exact-Solver` mimicking the semiflow trajectory except for the third type of trajectory described above to seek for the minimum of F. See Algo. 5 for the pseudo-code – Sec. A.

It can be decomposed into three so-called main *procedures*, which are `Teleportation`, `LineDescent`, `SphereDescent`. The latter is further described using two procedures `SphereIntersection` and `MinSphereIntersection`. A sixth procedure used in `LineDescent` and `SphereDescent` is `ClarkeQP`. Except for the latter which consists in solving a classical QP programming problem, their pseudo-code can be found in – Sec A. Procedures `LineDescent`, `SphereDescent` are illustrated in Fig. 2.

(i) If x_n lies in a full dimensional cell $\mathcal{C}$ (usually at the start of the algorithm), we check if $I^+(\mathcal{C})$ contains $c_{I^+(\mathcal{C})}$. If so, we let x_{n+1} be $c_{I^+(\mathcal{C})}$ and we stop the algorithm. Since this step follows property specific to

[1] Not to be confused with the sets $I^+(\mathcal{C}), I^0(\mathcal{C}), I^-(\mathcal{C})$, defined from the sign of the power distance.

[2] We say that the QP problem of minimizing $\left\|u + \sum_i \lambda_i v_i\right\|, 0 \leq \lambda_i \leq 1$ lies in a non-degenerate position when the argmin w is such that the set of i such that $w \cdot v_i = 0$ is exactly the set of i such that the coefficient of v_i in the decomposition of w is neither 0 or 1. This condition is standard in the sense that for a given box, for almost all isometries acting on that box, the image box lies in a non-degenerate position.

F and not the insight of the semiflow, we call it the `Teleportation` procedure. If there is no teleportation, we obtain x_{n+1} from the `LineDescent` procedure within $\mathcal{C}$, which is described as follows. We seek the first point on the half-line starting from x_n heading towards $c_{I^+(\mathcal{C})}$ meeting another cell, and we let x_{n+1} be this point. This is done by solving for quadratic equations (in t) of the form $\|x_n + tu - c_i\|^2 = R_i^2$.

(ii) Else x_n starts in cell of positive codimension. Compute the generalized gradient of f at x as well as the associated $I_*^+(x_n), I_*^0(x_n), I_*^-(x_n)$ with the `ClarkeQP` procedure (*i.e.* solving Eq. 19).

- If the generalized gradient is zero, the minimum has been reached and we can stop the algorithm.
- If $I_*^0(x_n)$ is empty, follow the `LineDescent` procedure described earlier within the full dimensional cell $\mathcal{C} * (x_n)$. Take x_{n+1} to be the point given by this procedure.
- Else, the semiflow starting from x_n stays in $S_{I_*^0(x_n)}$ and we follow the `SphereDescent` procedure, which consists in the following. Compute the point y where $f_{I_*^+(x_n)}$ restricted to $S_{I_*^0(x_n)}$ reaches its minimum via a procedure called `MinSphereIntersection` described in more details in the appendix. If y is in the cell $\mathcal{C}^*$, let x_{n+1} be y. Else, compute the center c_S and radius R_S of $S_{I_*^0(x_n)}$ via the `SphereIntersection` procedure. Via the parameterization $[0,1] \mapsto c_S + R_S \frac{(1-\lambda)x_n + \lambda y - c_S}{\|(1-\lambda)x_n + \lambda y - c_S\|}$ of the geodesic on $S_{I_*^0(x_n)}$, check the first point on the geodesic leaving the cell C^*. Let x_{n+1} be this point.

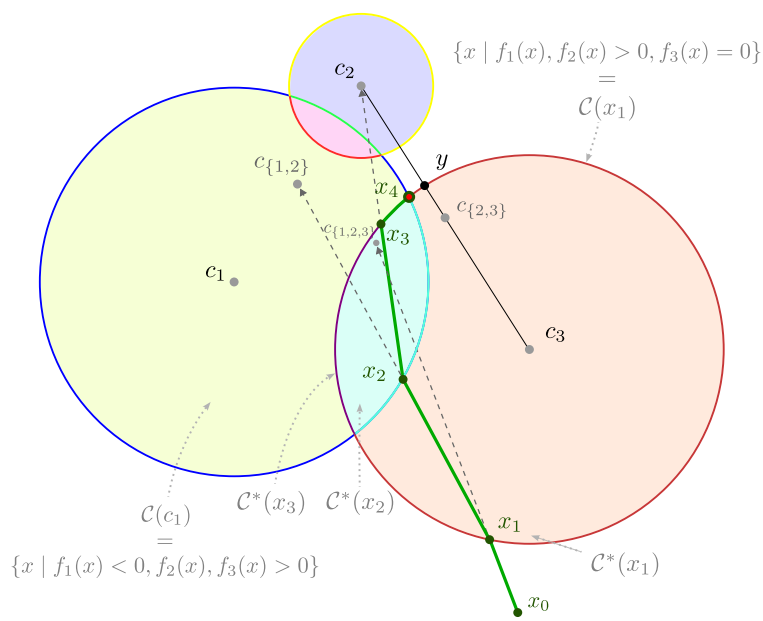


Figure 2: `LineDescent` **(from x_0, x_1, x_2) and** `SphereDescent` **(from x_3) steps. Underlying trajectories are depicted in dark green. Point y is obtained by** `MinSphereIntersection` **point x_3.**

Following the semiflow ensures that our algorithm converges in a known number of steps in a certain neighborhood of the point x^* where F reaches its minimum. The number of steps is related to the number of facets of the Clarke gradient $\partial_* F(x^*)$, which is 3^c where c is the number of spheres on which x^* lies.

Theorem. 2 (Algorithm convergence)**.** *There exists a neighborhood of the point x^* where F reaches its minimum, such that for any starting point in this neighborhood the algorithm converges in at most 3^c steps, where c is the number of spheres on which x^* lies.*

4.3 Combinatorial complexity

The complexity (number of cells of all dimensions) of the arrangement of n spheres in $\mathbb{R}^d$ is $O(n^d)$ Toth et al. (2017) and the bound is tight in the worst case. Despite this, we may expect a number of steps polynomial

in n if calculations remain *local* in the arrangement, a fact substantiated by our experiments. Note that for small values of η, the center of mass provides a warm start to the algorithm. Indeed the solution of the minimization problem of F_η varies continuously in η, and the barycenter is solution to the problem with $\eta = 0$.

Computations at each step are linear in the total number of points since `LineDescent` (resp. `SphereDescent`) computes the first sphere crossed by a line (resp. a sphere geodesics). Other computations involved at each step are at most cubic in the number of spheres on which the current point lies, be it by solving the `ClarkeQP` problem, inverting a matrix in `SphereIntersection` or computing an affine projection in `MinSphereIntersection`. Practically, our algorithm's complexity is nicely behaved–see Sec. 5.5.

4.4 Numerics

Arithmetics and number types. The algorithm from Sec. 4 is described assuming the real RAM model computing exactly with real numbers. On the other hand, geometric calculations (predicates, constructions) are known to be plagued with rounding errors Kettner et al. (2008). Serious difficulties may be faced for cascaded constructions, which iteratively embed new geometric objects (the points of the pseudo-gradient trajectory in our case). Advanced number types combining multiprecision and interval arithmetics can be used to maintain accurate representations in such cases. See *e.g.* random walk inside polytopes Chevallier et al. (2022) or trajectories of the flow complex (the Morse-Smale diagram of the distance function to a finite point set) Cazals et al. (2021).

In the sequel, we review the numerically demanding operations required by our algorithm, and refer the reader to Sec. 5 for experiments with our python based implementation.

Exact solver. The solver uses the following predicates and constructions:
•**Solving the `ClarkeQP` procedure.** The library *cvxpy* uses non-exact solvers to find the minimum of a QP problem. While those solvers usually give a result up to machine precision, we found out that they were prone to instability in minimizing functionals of the type $\|g + A\lambda\|^2$ when g is vector of norm largely greater than both 1 and than that of the columns of A, with constraints $0 \preceq \lambda \preceq 1$, in the sense that those solvers would claim the problem to be unfeasible. The equivalent problem of minimizing $\left\| \frac{1}{\|g\|} (g + A\lambda) \right\|^2$ was sufficient in addressing those issues. The precise computation of the vector λ is not needed as we only need to check for the index i with respectively $\lambda_i \in \{0\}, (0, 1)$ and $\{1\}$. The entries with values 0 and 1 are usually reached with precision greater than machine precision.

•**Solving the `LineDescent` and `SphereDescent` procedures.** Starting from a point x, with a prescribed direction u, procedure `LineDescent` seeks the first t such $x + tu$ changes cell, that is, the smallest positive t verifying $\|x + tu - c_i\|^2 = R_i^2$. This is obtained as roots of a second-degree polynomial. To weaken imprecision we chose to solve this equation with a renormalized vector u of norm 1. Similarly, given two points x, y on a sphere of radius R_S and center c_S, procedure `SphereDescent` computes the first point on the geodesic between x and y changing cell by solving for a quadratic equation.

•*Solving the `SphereIntersection` procedure.* The pair center/radius c_S, R_S used above is obtained as the center and radius of an intersection of spheres. As described in the appendix, the center is obtained through the computations of a projection onto an hyperplane defined by linear equations involving the centers of said spheres. Computing R_i is done by solving for a quadratic equation.

•*Solving for the `MinSphereIntersection` procedure.* Given an intersection of spheres S_I, with $I \subset \{1, \ldots, n\}, \#I \leq d$ the `MinSphereIntersection` procedure minimizes a function of the form $(f_J)_{|S_I}$ by computing a similar projection on a convex hull.

The genericity assumptions and the robustness of our routines are further discussed in the SI Section 4.5.

The BFGS solver. The Broyden–Fletcher–Goldfarb–Shanno quasi-Newton method (BFGS) is designed to be very efficient on twice differentiable function by approximating the Hessian matrix without any matrix inversion (in opposition to Newton's methods), using the gradient. When the gradient is not given, it is

estimated using finite differences. While the objective function F_η is not differentiable, it is also known that BFGS works well in practice for non-differentiable functions Lewis & Overton (2012). The next section challenges this observation for F_η.

4.5 Genericity assumptions

The genericity condition for our algorithm to work at a point x is that the intersection of spheres containing x is transverse. For k spheres, this existence of a non transverse intersection is checked as follows. The non transversality at point x reads as $\sum_i \lambda_i(x - c_i) = 0$ or equivalently $(\sum_i \lambda_i)x = \sum_i \lambda_i c_i$, which requires discussing two cases: (Case 1) $\sum_i \lambda_i \neq 0$, and (Case 2) $\sum_i \lambda_i = 0$. In each case, we need to check the points x satisfying these conditions satisfy the sphere equations.

To do so, Case 1 requires solving a QP problem. Case 2 requires computing the null space of the matrix $A = (\{[c_i\ 1]^T\}_{i=1,\ldots,k})$, and if $Ker(A) \neq 0$, one needs to further check the sphere feasibility conditions, which require solving another linear system.

Also note that at a given point x, the test simply boils down to checking that the vectors $x - c_i$ are linearly independent.

The exact computation of trajectories by our algorithm is more involved. It requires cascaded degree two and degree four algebraic numbers. (NB: degree two when intersecting a segment with a sphere; degree four when intersecting a geodesic along a sphere with another sphere.) A robust numerical solution could be obtained using say an interval number type with bounds of arbitrary precision, e.g. the iRRAM library Müller (2001).

In practice though:

- We do not check the transversality condition, as even in medium dimensional spaces, the points where the intersections are not transverse are scarce, and our trajectories do not cross them.

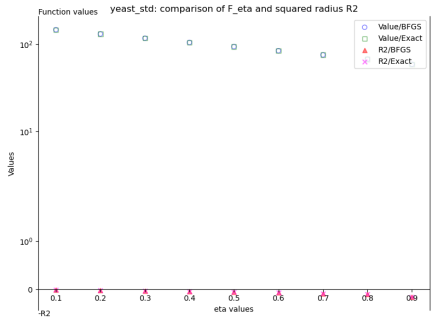
- We do not use elaborate number types, since the observed robustness of our floating point implementation did not require using them.

5 Spherical clusters: experiments

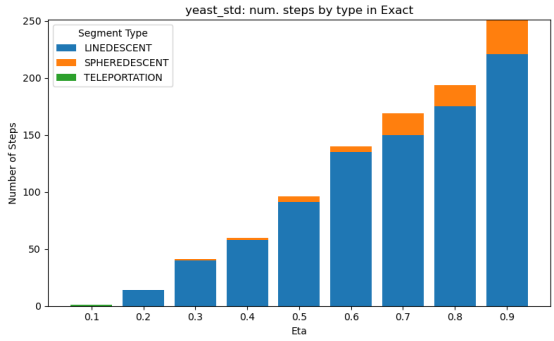
5.1 Implementation

Our implementation of the algorithm from Sec. 4 using python and numpy is denoted `SC-Exact-Solver` and is termed the *exact method*. It is available from the Core tier of the Structural Bioinformatics Library, in the Cluster spherical package. We compare the solution yielded by `SC-Exact-Solver` against that yielded by `SC-BFGS-Solver`– the optimization being done with BFGS. The cluster centers are denoted $\text{opt}_{\text{Exact}}$ and opt_{BFGS} respectively.

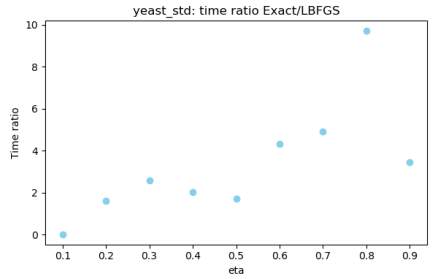
Calculations were run on a DELL precision 5480 equipped with 20 CPUs of type Intel(R) Core(TM) i9-13900H, 32Go or RAM, and running FedoraCore 42.



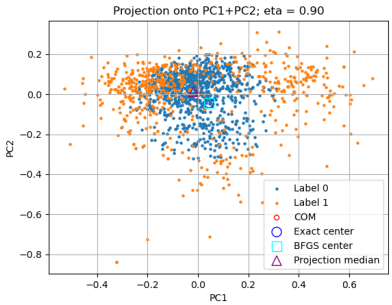
(A) F_η and $R^2 2$ function of η



(B) Step types in trajectory of SC-Exact-Solver



(C) Ratio $t_{\text{Exact}}/t_{\text{BFGS}}$



(D) Projection plot, $\eta = 0.9$

Figure 3: **Yeast landsat.** This dataset features 6435 points in dimension $d = 9$. **(A)** Dual plot **(B)** Steps types as a function of η **(C)** Running times t_{Exact} vs $t_{\text{L-BFGS-B}}$ **(D)** Projection plot with inliers and outliers

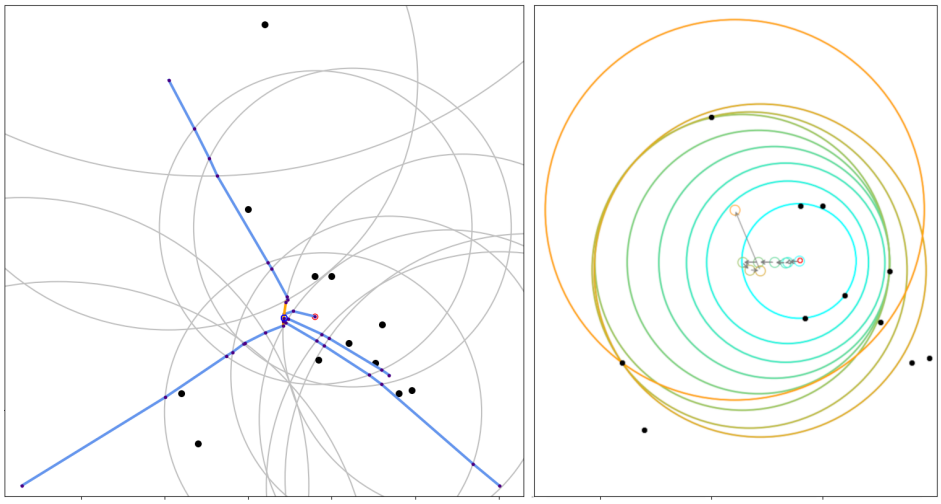


Figure 4: **Spherical cluster: illustrations on a toy 2D dataset. (Left)** Trajectories from five different starting points, with $\eta = 0.5$ (Line/Sphere descents in blue/orange). **(Right)** Evolution of the cluster center for η in $[0.1, 0.9]$ be step of 0.1.

5.2 Contenders, datasets and statistics

Contenders. We challenge SC-Exact-Solver with two contenders denoted SC-BFGS-Solver and SC-LBFGS-Solver respectively, using the BFGS and L-BFGS-B solvers provided by `scipy.optimize`. Note that the latter uses an approximation of the Hessian–as opposed to a $O(d^2)$ sized matrix.

Medium dimensional (MD) datasets. We ran experiments on ten standards datasets used in clustering experiments Celebi et al. (2013); Carrière & Cazals (2025), with size $n \in [1484, 200000]$ and dimension $d \in [9, 77]$ – Table 1. Following common practice, on a per dataset basis, we perform a min-max normalization on the coordinates to avoid overly large ranges.

High dimensional (HD) datasets. We use two datasets to explore the effect of high dimensionality. The `Proteins-HMM` dataset consists of $N = 1443$ protein sequences whose biological function is unknown Vicedomini et al. (2022). To identify putative functions, each sequence is scored by $d = 400$ Hidden Markov Models (HMM) corresponding to major known protein functions, yielding a d-dimensional point. Carbone et al. perform hierarchical clustering on these points (Ward's method), yielding 16 clusters (sizes in 11..176) of sequences expected to have identical functions. The Arcene dataset contains mass-spectrometric data meant to distinguish cancer versus normal patients, and has shape $(n, d) = (900, 10000)$. The $d = 10000$ features correspond to protein abundances in human sera, to which distractor features with no predictive power have been added.

Parameters. For each dataset, we explore values of η in $[0.1, 0.9]$ by steps of 0.1 – nine values in total.

Statistics and plots. We define (Fig. 3 and SI):
•*Spherical cluster square radius R^2.* The square radius with respect to which the power distance is computed, that is $\eta\hat{\sigma}^2$ – see Eq. 1.

•*Projection plot.* The plot of all points (data points, center of mass, SC centers) onto the first two principal directions. Inliers (resp. outliers) are displayed in blue (resp. orange). The number of outliers identified by our cluster model, is denoted #outliers(SC). Similarly #outliers(COM) stand for the number be outliers defined with respect to a sphere of the same radius centered at the center of mass.

•**Dual plot**. Reports F_η and R^2 as a function of η. (NB: R^2 values are represented negated on this plot.)

•**Stacked barplot.** The plot function of η counting the number of steps of each type (line descent, sphere descent, teleportation) in `SC-Exact-Solver`.

•**Time ratio plots.** The plots for $t_{\text{Exact}}/t_{\text{BFGS}}$ and $t_{\text{Exact}}/t_{\text{L-BFGS-B}}$, comparing the running times of `SC-Exact-Solver` against those of `SC-BFGS-Solver` and `SC-LBFGS-Solver` respectively.

•**Average outlier cost plots.** The plots $F_\eta(\text{opt}_{\text{Exact}})/\#\text{outliers(SC)}$ and $F_\eta(\text{opt}_{\text{Exact}})/\#\text{outliers(COM)}$.

•**Outlier ratio plot.** The plot #outliers(COM)/#outliers(SC).

•**Distance between points plot.** The plot comparing the distances between three special points: $\text{opt}_{\text{Exact}}$, opt_{BFGS}, and the projection median from Durocher et al. (2017).

5.3 Spherical cluster model

Trajectories and centers. We build up an intuition by observing the trajectories followed by our exact solver when varying the starting point, on a simple toy 2D example (Fig. 4(A)). We also note that even for such simple cases, the center moves in a complex way when varying η (Fig. 4(B)).

Running times and the burden of dimensionality. For the dataset `DS-Cluster`, we first inspect running times using the ratios $t_{\text{Exact}}/t_{\text{L-BFGS-B}}$ (Fig. 7, Tab. 3), and $t_{\text{Exact}}/t_{\text{L-BFGS-B}}$ (Fig. 8, Tab. 5). Using median values, the comparison shows that `SC-Exact-Solver` is faster than `SC-BFGS-Solver` up to up to $\eta = 0.7$ included, while `SC-Exact-Solver` is faster than `SC-LBFGS-Solver` up to $\eta = 0.3$ included. Increasing the value of η results in larger spheres and more complex arrangements, whence the burden observed.

We perform the same analysis for the dataset `Proteins-HMM`, for ratios $t_{\text{Exact}}/t_{\text{L-BFGS-B}}$ (Fig. 13, Tab. 7), and $t_{\text{Exact}}/t_{\text{L-BFGS-B}}$ (Fig. 14, Tab. 9). Using median values again, `SC-Exact-Solver` is two for four orders of magnitude faster than `SC-BFGS-Solver` and `SC-LBFGS-Solver`.

For the Arcene dataset, BFGS turned out to be unpractical. We observe that our exact algorithm is between two and five orders of magnitude faster than SC-LBFGS-Solver (Fig. 19).

Summarizing, SC-Exact-Solver is orders of magnitude faster than SC-LBFGS-Solver and SC-LBFGS-Solver for datasets of small/intermediate dimension and small values of η, and orders of magnitude faster than these two methods for high dimensional datasets.

Function values. Wen now compare the values yielded by the three contenders: SC-Exact-Solver vs SC-BFGS-Solver: Fig. 13 and Table 8; SC-Exact-Solver vs SC-LBFGS-Solver: Fig. 14 and Table 10. While these values are on par for all values of η, we note that the approximate solvers are more prone to numerical instabilities, in particular for DS-Cluster and for DS-HD.

Outliers and the selection of η. As noticed earlier, the SC center depends both on inliers and outliers. On all datasets processed, the outlier ratio #outliers(COM)/#outliers(SC) lies in the interval $\sim [1, 3]$, which illustrates the stringency of our criterion to identify such points.

The outlier cost plot $F_\eta(\text{opt}_{\text{Exact}})$/#outliers(SC) is of particular interest to capture the scale/cost of outliers. The general behavior of this plot is a monotonic decrease (*e.g.* Fig. 15, Fig. 17), indicating that *capturing* outliers is getting easier when increasing η. However several datasets exhibit a non monotonic behavior (Fig. 9, Fig. 11), showing that *gaps* must be crossed to capture certain outliers.

5.4 Projection median

We also compare $\text{opt}_{\text{Exact}}$ against the projection median computed as a weighted average Durocher et al. (2017). As expected, their distance increases as a function of η, showing that the cluster center behaves as a parameterized point set center. See Supporting Information, plots *Distance between points* plots.

5.5 Discussion: complexity in practice

Number of steps and multiplicity of cells. For all datasets and whatever the value of eta, we checked that the number of cells traversed is negligible with respect to the worst case complexity of the arrangement. We also checked that in practice the cells visited are so only once, contrary to what can be found in pathological sphere configurations with bad starting points. We draw a comparison with the celebrated simplex algorithm, which has exponential complexity in the worst case, and yet stays in use after almost 80 years. Our framework is similar in spirit, as the arrangement is exponential in dimension while the algorithm is effective in practice. Note that we were unable to build an example where the number of steps matches the total number of cells. The fact that our trajectory benefits from a *warm start* (warmer as eta decreases) at the center of mass certainly helps in reducing the number of cells to be crossed before reaching the minimum, explaining the efficiency of our algorithm against classical methods (whose underlying trajectory is not stopped when crossing a cell) when eta is not close to $1 - 1/n$.

Behavior in high-dimensions. In higher dimensions, our algorithm outperforms the BFGS and L-BFGS. The main factor for this behavior resides in computation times of the steps which are cubic in the number of spheres containing the current point of the trajectory–see Section 4.3). In practice, this number is small, on average between 2 and 3 as shown by our experiments: a vast majority of steps are either LineDescent procedures or SphereDescent on a small number of spheres–rarely more than five even in high dimensions.

6 Outlook

Spherical clusters embedded in affine spaces of fixed dimension provide useful insights into the geometry of high dimensional point clouds.

Our work shows spherical clusters are well defined by a non smooth strictly convex problem. We also show that this optimization problem is well poised and can be solved by an exact iterative procedure following a semiflow on a stratified complex defined by an arrangement of spheres. Quite remarkably, BFGS also solves

all the instances we processed to satisfaction. Yet, the exact solver is orders of magnitude faster than BFGS based heuristics for high dimensional datasets (say $d > 100$), and for dataset of medium dimensionality and small values of η. Our experiments also show that the center of spherical clusters behave as a high dimensional median parameterized by the fraction η of the variance of distances between the cluster center and all points.

Our work calls for future developments in two directions. From a theoretical standpoint, understanding the complexity of our exact method as a function of η appears as a challenging problem. Practically, spherical clusters were proposed to define mixtures of such clusters embedded into affine spaces of arbitrary codimension. We anticipate that such models will prove extremely useful in data analysis at large, providing at the same time compact clusters capturing the intrinsic dimension of the data.

Acknowledgements

References

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SODA*, pp. 1035. Society for Industrial and Applied Mathematics, 2007.

Chanderjit Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3:177–191, 1988.

Riddhipratim Basu, Bhaswar B Bhattacharya, and Tanmoy Talukdar. The projection median of a set of points in $\mathbb{R}^d$. *Discrete & Computational Geometry*, 47(2):329–346, 2012.

Jérôme Bolte, Aris Daniilidis, Olivier Ley, and Laurent Mazet. Characterizations of Łojasiewicz inequalities: subgradient flows, talweg, convexity. *Trans. Am. Math. Soc.*, 362(6):3319–3363, 2010. ISSN 0002-9947. doi: 10.1090/S0002-9947-09-05048-X.

Sébastien Bubeck. Convex optimization: algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3-4): 231–357, 2015. ISSN 1935-8237.

James V. Burke, Frank E. Curtis, Adrian S. Lewis, Michael L. Overton, and Lucas E. A. Simões. *Gradient Sampling Methods for Nonsmooth Optimization*, pp. 201–225. Springer International Publishing, Cham, 2020. ISBN 978-3-030-34910-3. doi: 10.1007/978-3-030-34910-3_6. URL `https://doi.org/10.1007/978-3-030-34910-3_6`.

G. Carrière and F. Cazals. Improved seeding strategies for k-means and k-GMM. *Transactions on Machine Learning Research*, In press(NA), 2025. URL `https://openreview.net/forum?id=4Ut2YnekhN`.

F. Cazals, B. Delmas, and T. O'Donnell. Fréchet mean and p-mean on the unit circle: decidability, algorithm, and applications to clustering on the flat torus. In D. Coudert and E. Natale (eds.), *Symposium on Experimental Algorithms*, Sophia Antipolis, 2021. Lipics. doi: 10.4230/LIPIcs.SEA.2021.15. URL `https://hal.inria.fr/hal-03183028`.

M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, 40(1):200–210, 2013.

F. Chazal, L. Guibas, S. Oudot, and P. Skraba. Persistence-based clustering in Riemannian manifolds. *J. ACM*, 60(6):1–38, 2013. doi: http://doi.acm.org/10.1145/2535927.

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE PAMI*, 17(8):790–799, 1995.

A. Chevallier, S. Pion, and F. Cazals. Improved polytope volume calculations based on Hamiltonian Monte Carlo with boundary reflections and sweet arithmetics. *J. of Computational Geometry*, 13(1):55–88, 2022. URL `https://jocg.org/index.php/jocg/article/view/3283`.

F.H. Clarke. *Nonsmooth Analysis and Control Theory*. Springer, 1997.

Kenneth L Clarkson, David Eppstein, Gary L Miller, Carl Sturtivant, and Shang-Hua Teng. Approximating center points with iterated radon points. In *Proceedings of the ninth annual symposium on Computational geometry*, pp. 91–98, 1993.

Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 9–21, 2016.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, 1973.

Stephane Durocher and David Kirkpatrick. The projection median of a set of points. *Computational Geometry*, 42(5):364–375, 2009.

Stephane Durocher, Alexandre Leblanc, and Matthew Skala. The projection median as a weighted average. *Journal of Computational Geometry*, 8(1):78–104, 2017.

Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

D. Halperin and M. Sharir. Arrangements. In *Handbook of discrete and computational geometry (3rd Edition)*. CRC Press, 2017.

Cédric Josz. Global convergence of the gradient method for functions definable in o-minimal structures. *Math. Program.*, 202(1-2 (A)):355–383, 2023. ISSN 0025-5610. doi: 10.1007/s10107-023-01937-5.

Parthan Kasarapu and Lloyd Allison. Minimum message length estimation of mixtures of multivariate gaussian and von mises-fisher distributions. *Machine Learning*, 100:333–378, 2015.

L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C. Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61–78, 2008.

Yaakov Kupitz and Horst Martini. Geometric aspects of the generalized fermat-torricelli problem. *Bolyai Society Mathematical Studies*, 6:55–129, 1997.

Adrian Lewis and Michael Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141, 01 2012. doi: 10.1007/s10107-012-0514-2.

Jacob Leygonie, Mathieu Carrière, Théo Lacombe, and Steve Oudot. A gradient sampling algorithm for stratified maps with applications to topological data analysis. *Math. Program.*, 202(1-2 (A)):199–239, 2023. ISSN 0025-5610. doi: 10.1007/s10107-023-01931-x.

Chun-Guang Li, Chong You, and René Vidal. On geometric analysis of affine sparse subspace clustering. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1520–1533, 2018.

Sylvie Marcellin and Lionel Thibault. Evolution problems associated with primal lower nice functions. *J. Convex Anal.*, 13(2):385–421, 2006. ISSN 0944-6532. URL `www.heldermann.de/JCA/JCA13/jca13.htm`.

Gary L Miller and Donald R Sheehy. Approximate center points with proofs. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pp. 153–158, 2009.

N. Müller. The iRRAM: Exact arithmetic in C++. In *Computability and Complexity in Analysis*, pp. 222–252. Springer, 2001.

Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105, 2004.

Mahdi Soltanolkotabi and Emmanuel J Candes. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, 40(4):2195–2238, 2012.

Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

John W Tukey. Mathematics and the picturing of data. In *Proceedings of the international congress of mathematicians*, volume 2, pp. 523–531. Vancouver, 1975.

R. Vicedomini, J. P. Bouly, E. Laine, A. Falciatore, and A. Carbone. Multiple profile models extract features from protein sequence data and resolve functional diversity of very different protein families. *Mol. Biol. Evol.*, 39(4):msac070, 2022.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Dingding Wang, Chris Ding, and Tao Li. K-subspace clustering. 2009.

Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3): 645–678, 2005.

The supporting information is structured as follows:

- Section A: Algorithms and proofs
- Section B: Experiments

A Algorithms and proofs

In this section of the Appendix, we develop the theoretical background and prove the results of the article. More precisely,

- We prove several intermediate results needed for some propositions used in the article. Intermediate results will be called *lemmas* while the results quoted in the article will be called *propositions*.
- We precisely describe procedures `LineDescent` and `SphereDescent`, as well as the procedures `SphereIntersection` and `MinSphereIntersection` used internally. We also explain the maths behind them.
- We describe the full `SC-Exact-Solver` algorithm and prove its convergence.

A.1 Lemmas and propositions

With the centers c_i and radii R_i as defined in Eqs. 10, we have:

Lemma. 2 (Freeness of the $\nabla f_i(x)$)**.** *For any x in R^d and for any $J \subset \{1, \dots, n\}$, the rank of the family $(\nabla f_i(x))_{i\in J}$ is the dimension of the affine hull of x and $(c_i)_{i\in J}$.*

As a consequence, if the centers c_i are in generic position, for any $x \in \mathbb{R}^d$ the family $(\nabla f_i(x))_{i\in I^0(x)}$ is free.

Proof. The rank of said family is the rank of the vectors $(x - c_i)_{i\in J}$, which is exactly the dimension of the affine hull of x and $(c_i)_{i\in J}$. For the second point, if $I^0(x)$ is empty (resp. a singleton) the result is trivial by convention (resp. because no radius is zero). Assuming $k = \#I^0(x) \geq 2$, x belongs to the intersection $S_{I^0_(x)} = \bigcap_{i\in I^0(x)} S_i$ which is a $d-k$ dimensional sphere centered around a point c_S of $\mathrm{Aff}((c_i)_{i\in I^0}$ orthogonal to the previous affine space, with radius R_S - see Lemma 4 for the exact computations. As a consequence the dimension of the affine hull of x, $(c_i)_{i\in I^0(x)}$ is 1 + the dimension of the affine hull of $(c_i)_{i\in I^0(x)}$ which is $k-1$ by genericity of the centers. □

The following lemma describes the full-dimensional cells in any neighborhood of a point. It is illustrated in Fig. 5 below.

Lemma. 3 (Full-dimensional cells around a point)**.** *Assume that the family $\{c_i\}$ lies in a generic position. Let $x \in \mathbb{R}^d$ be a point with $\#I^0(x) = k(x) > 0$. Then any neighborhood of x encounters exactly $2^{k(x)}$ cells of full-dimension, and any such cell is determined by $I^+ = I^+(x) \cup A$, $I^- = I^-(x) \cup B$, where $A \cup B$ is a 2-partition of $I^0(x)$.*

Proof. It is clear that the full dimensional cells possibly existing around x are included in the one described in the proposition.

Reciprocally, let $A \subset I^0(x)$ and $B = I^0(x) \setminus A$. Consider the set U_A the cone of vectors such that $\nabla f_i(x) \cdot u \geq 0, \nabla f_j(x) \cdot u \leq 0$ for all $i \in A$, $j \in B$. This is the polar (or dual) cone to the cone T_A generated by the vectors $(-\nabla f_i(x))_{i\in A}$ and $(\nabla f_j(x))_{j\in B}$; as such U_A has non-empty interior if and only if $T_A \cap (-T_A) = \{0\}$. This condition is satisfied (for every $A \subset I^0(x)$) as the family $(\nabla f_i(x))_{i\in I^0(x)}$ is free by the previous lemma. Any u in the interior of U_A satisfies $\nabla f_i(x) \cdot u > 0$, $\nabla f_j(x) \cdot u < 0$ for any $i \in A, j \in B$ and thus $I^+(x + tu) = I^+(x) \cup A$, $I^-(x + tu) = I^-(x) \cup B$ for $t > 0$ small enough. □

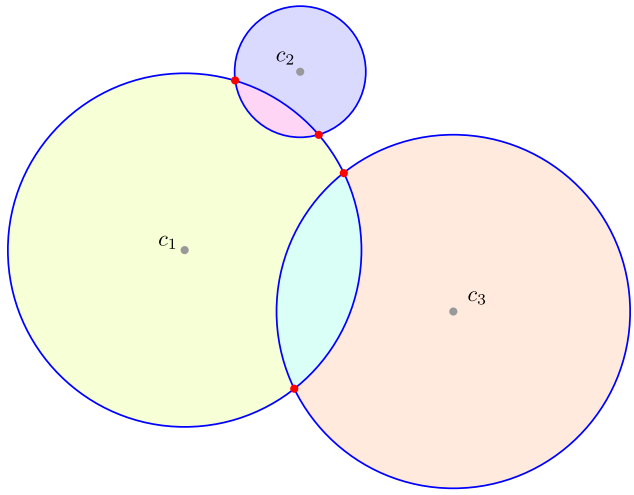


Figure 5: **Neighboring cells.** Each red dot has I^0 of cardinal two and 2^2 neighboring cells of full dimension. In blue, points with I^0 of cardinal 1 have 2^1 neighboring cells.

It is clear that F is smooth at point x if and only if x does not lie on a sphere. The Clarke differential of F at x will allow us to study F in a neighborhood of x. Thanks to Prop. 1 the Clarke gradient of F has an explicit representation (Fig. 6 for an illustration):

Proposition. 1 (Clarke gradient of F)**.** *For any $x \in \mathbb{R}^d$ we have :*

$$\partial_* F(x) = \{\nabla f_{I^+(x)} + \sum_{i \in I^0(x)} \lambda_i \nabla f_i, 0 \leq \lambda_i \leq 1\}. \tag{22}$$

When $0 \notin \partial_* F(x)$, the minimum of $\lambda \mapsto \left\| \nabla f_{I^+(x)} + \sum_{i \in I^0(x)} \lambda_i \nabla f_i \right\|$ is attained at a unique point.

Proof. Recall that the Clarke gradient of a Lipschitz function is obtained as the convex hull of limits of nearby gradients Clarke (1997). Denote by B the set on the right-handside of Eq. 22. B is a convex parallelepiped whose vertices are $\nabla f_{I^+(x) \cup A}(x) = \nabla f_{I^+(x)}(x) + \nabla f_A(x)$ where A ranges among the subsets of $I^0(x)$ (with the convention that $\nabla f_\emptyset = 0$); these are also exactly points determining the convex hull in the definition of $\partial_* F(x)$.

Now the gradients $(\nabla f_i)_{i \in I^0(x)}$ are free by Lemma 2. It follows that $\lambda \mapsto \left\| \nabla f_{I^+(x)} + \sum_{i \in I^0(x)} \lambda_i \nabla f_i \right\|^2$ is strictly convex and thus admit only one minimizer on the convex set $[0,1]^n$. □

Recall the definition of the sets $I^0_*(x), I^+_*(x), I^+_*(x)$ in Section 3, from which one defines the cell $\mathcal{C}^*(x)$ of the arrangement containing x. We have the following:

Proposition. 2. *When $I^0_*(x)$ coincides with $\{i, \nabla_* F(x) \cdot \nabla f_i(x) = 0\}$, we say that the Clarke QP problem is* non-degenerate *at x. In this case the exact gradient flow of F begins by a curve inside $\mathcal{C}^*(x)$.*

Proof. We will show that for t positive and small enough, the semiflow trajectory $t \mapsto x(t)$ starting at x lies in $\mathcal{C}^*(x)$ by showing separately that

- $x(t)$ stays in $S_{I^0_*(x)}$ for t small enough and positive;
- $I^+(x(t)) = I^+_*(x)$ and $I^-(x(t)) = I^-_*(x)$ for t small enough.

Remark that the coefficients α_i defining $\nabla F_*(x)$ are obtained through the affine projection of 0 onto the affine set generated by $\nabla f_{I^+_*(x)} + \nabla f_i(x)$ where i ranges among $I^0_*(x)$ by the non-degenerate condition. This

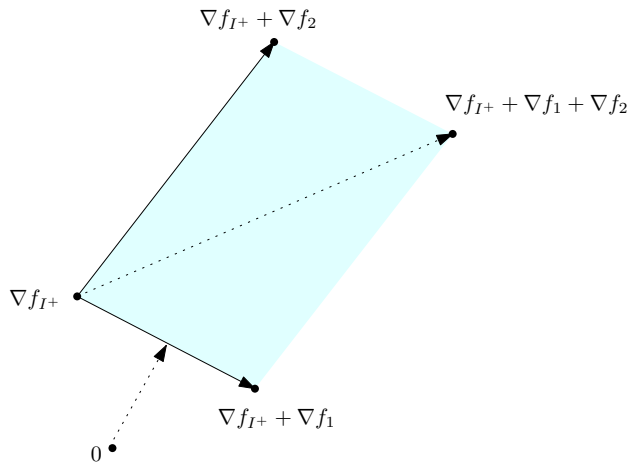


Figure 6: **Clarke gradient.** The Clarke gradient is the convex hull, and the generalized gradient if the projection of the origin onto that convex hull. On this example $I^0 = \{1, 2\}$, and $I^0_* = \{1\}$.

is equivalent to $\lambda_i(x)$ being the solution of the following non-singular linear equation:

$$\sum_{i \in I^0_*(x)} \lambda_i(x) \nabla f_i(x) \cdot \nabla f_j(x) = -\nabla f_{I^+_*(x)} \cdot \nabla f_j(x) \qquad \text{for every } j \in I^0_*(x). \tag{23}$$

Equivalently, putting A the full-rank matrix whose rows are $(\nabla f_i(x))_{i \in I^0_*}$, λ is a solution to

$$A(x)^\mathsf{T} A(x) \lambda(x) = -A(x)^\mathsf{T} \nabla f_{I^+_*}(x),$$

and the corresponding generalized Clarke gradient is equal to $\phi(x) = A(x)\lambda(x) + \nabla f_{I^+_*(x)}(x)$. Since $x \mapsto A(x)$ is a smooth map, the ODE $z'(t) = \phi(z(t))$ starting with $z(0) = x$ admits a solution for every $t \in \mathbb{R}^d$. This solution coincides with the exact gradient flow for small t, as by continuity we still have $0 < \lambda_i(z(t)) < 1$ for small t, so that the point of least norm in $\mathrm{Aff}(\nabla f_{I^+_*(z(t))} + \nabla f_i(z(t))_{i \in I^0_*(z(t)) \cup \emptyset}$ coincides with the point of least norm within the parallelogram with generated by the same vectors. This ensures that $\nabla f_i(t) \cdot z'(t) = 0$ for any $i \in I^0_*(z(t))$ so that the trajectory stays on $S_{I^0_*(x)}$. Moreover, for t small enough we have $I^+(z(t)) = I^+(x) \cup i \in I^0(x), \lambda_i(x) = 1$ and $I^-(z(t)) = I^-(x) \cup i \in I^0(x), \lambda_i(x) = 0$, so that the Clarke gradient $\partial_* F(z(t))$ is the parallelogram generated by the previous vectors $\nabla f_{I^+_*(x)}(x) + \nabla f_i(x)$, where i ranges among $I^0_*(x)$. $\square$

A.2 Procedures and their justification

Algorithm 1 – `LineDescent`
Compute the first point along the semi-line $x+tu$, $t>0$ changing cell. Compute the cell to which this point belongs.

Require: A set of centers c_i and radii R_i, an initial point x, a directing vector u, sets of indices I^+, I^0, I^-.
Ensure: $z = x + tu$ is the first point along the semi-line $x + tu$, $t > 0$ changing cell. I^+, I^0, I^- are the set of indices at z.

```
t ← +∞
ChangingIndices ← ∅                ▷ Set of indices in I⁺ ∪ I⁻ to which z belongs - usually a singleton.
for i ∈ I⁺ ∪ I⁻ do
    t_i ← Smallest positive solution to ‖x + tu − c_i‖² = R_i² (with t_i = ∞ if no solution).
    if t_i = t and t_i ≠ ∞ then
        Add i to ChangingIndices
    else if t_i < t then
        ChangingIndices ← {i}.
        t ← t_i.
Add elements of ChangingIndices to I⁰, Remove elements from ChangingIndices off of I⁺, I⁻.
return z = x + tu, I⁰, I⁺, I⁻
```

The `SphereDescent` procedure is less straightforward. It needs two subprocedures : `SphereIntersection` and `MinSphereIntersection`. The first one is done by solving for a linear system.

Lemma. 4 (Computations in the `SphereIntersection` procedure.)**.** *Let $2 \leq j \leq d$ and let $S_1, \dots S_j$ be spheres in $\mathbb{R}^d$ with centers $c_1, \dots, c_j$ (in generic position) and radii $R_1, \dots, R_j$ intersecting transversely. Then $S = \bigcap_{1 \leq i \leq j} S_i$ is manifold of dimension $d - j - 1$ described as the intersection of a sphere of radius R_S and center c_S intersected with an affine subspace given by the $j - 1$ equations*

$$x \cdot (c_1 - c_i) = K_i, \tag{24}$$

for all $2 \leq i \leq j$ with $K_i := \frac{1}{2}\left(R_1^2 - \|c_1\|^2 - R_i^2 + \|c_i\|^2\right)$. Moreover, the center c_S is the only convex combination of $c_1, \dots, c_j$ belonging to this affine subspace, and it is obtained as the solution of the linear system with j equations and j unknowns:

$$\begin{cases} \sum_i \lambda_i c_i \cdot (c_1 - c_k) & = K_k \quad \text{for every } 2 \leq k \leq j; \\ \sum_i \lambda_i & = 1. \end{cases} \tag{25}$$

The radius R_S^2 verifies for every i in $\{1, \dots, n\}$

$$\|c_i - c_S\|^2 + R_S^2 = R_i^2. \tag{26}$$

Proof. Let c_S be the solution of the Eq. 25. Denote by A the affine space defined by equations (on u) $u \cdot (c_1 - c_k) = K_k$ for every $2 \leq k \leq j$, to which belong any point of S by subtracting equations $\|x - c_i\|^2 = R_i^2$ to $\|x - c_1\|^2 = R_1^2$. Then c_S is the point at the intersection of A and the affine hull of every c_i. For any x in A by the Pythagorean theorem one has $\|x - c_i\|^2 = \|x - c_S + c_S - c_i\|^2 = \|x - c_S\|^2 + \|c_S - c_i\|^2$. One can check that $R_i^2 - \|c_i - c_S\|^2$ is a positive number independent of i, whose square root we define to be R_S. Thus x belongs to S if and only if $x \in A$ and $\|x - c_S\|^2 = R_S^2$. □

Algorithm 2 – `SphereIntersection`
Compute the center and radius of the $d - \#I$ dimensional sphere given by the intersection of transverse spheres S_I.

Require: A set of indices I with corresponding centers c_i and radii R_i
$\lambda \leftarrow$ solution of the system of Eqs. 25
$c_S \leftarrow \sum_{i \in I} \lambda_i c_i$
$R_S \leftarrow \sqrt{R_1^2 - \|c_1 - c_S\|^2}$
return c_S, R_S

The second one requires a projection onto an affine hull.

Lemma. 5 (Computations in the `MinSphereIntersection` procedure.)**.** *Let $S = \cap_{1 \leq i \leq j} S_i$ be an intersection of j spheres in $\mathbb{R}^d$ intersecting transversely with centers in a generic position. Let g be a map of the form $x \mapsto \alpha \left(\|x - c\|^2 - R^2 \right)$ with $c \notin S$. Then $g_{|S}$ is a smooth map whose differential vanishes on S exactly at point $c_S \pm R_S \frac{b}{\|b\|}$ where $b := c - \pi(c)$, π being the projection map onto the closest point in the affine subspace generated by the centers $c_1, \ldots, c_j$. These points are the argmin and the argmax of $g_{|S}$.*

Proof. Without loss of generality, take $\alpha = 1/2$ and $R = 0$. For any $x \in S$, $\nabla g_{|S}(x)$ is the orthogonal projection of $\nabla g(x) = x - c$ onto the tangent space $T_x S$. This space consists in a $d - j$ dimensional vector subspace of $\mathbb{R}^d$ defined by the equations (on v)

$$v \cdot (c_1 - c_i) = 0 \text{ for all } 2 \leq i \leq j \qquad v \cdot (x - c_S) = 0. \tag{27}$$

By the change of variable $x = c_S + R_S u$ and by the previous proposition, S can be parameterized by u ranging in the unit sphere orthogonal to all the $c_1 - c_j$. All in all $T_x S$ is the space orthogonal to $(c_1 - c_i)_{2 \leq i \leq j}, u$. The orthogonal projection onto such a space is zero if and only if it belongs to the space spanned by these vectors. Thus the differential vanishes in x if and only if

$$x - c = c_S + R_S u - c \in \mathrm{Vect}(c_1 - c_i, u).$$

which after some manipulation and seeing that c_S is a convex combination of every c_i, is equivalent to

$$c - c_1 \in \mathrm{Vect}(c_1 - c_i, u). \tag{28}$$

They are exactly two unit vectors orthogonal to every $c_1 - c_i$ such that the previous equation is verified. Indeed, write $c - c_1 = a + b$ where $a \in \mathrm{Vect}(c_1 - c_i)$ and b is orthogonal to them, which is equivalent to b being $c - \pi(c)$ as defined in the proposition. The previous equation 28 is verified if and only if $u = \pm \frac{b}{\|b\|}$. The argmin and the argmax being two distinct critical points of $g_{|S}$, they are necessarily these two points. □

Algorithm 3 – `MinSphereIntersection`
Compute the minimum y of a function of the type $\|x - c\|^2$ on a transverse intersection of spheres $S = \cap_{i \in I} S_i$.

Require: A set of indices I with corresponding centers c_i and radii R_i, a center c such that $\|x - c\|^2$ is to be minimized over the intersection S of the spheres indexed in I
Ensure: $y = \mathrm{argmin}_{x \in S} \|x - c\|^2$
$(c_S, R_S) \leftarrow$ SphereIntersection(I)
$\pi(c) \leftarrow$ the projection of c into the affine hull of $(c_i)_{i \in I}$.
$u \leftarrow \frac{c - \pi(c)}{\|c - \pi(c)\|}$
$(y_1, y_2) \leftarrow (c_S + R_S u, c_S - R_S u)$
if $\|y_1 - c\|^2 \geq \|y_2 - c\|^2$ **then return** y_2
else **return** y_1

The following lemma show how to compute the `SphereDescent` procedure, that is, to compute when a geodesic $[x, y]_S$ among a sphere S changes cell starting from x.

Lemma. 6 (Computations in the SphereDescent procedure.)**.** *Let x, y be two non-antipodal points on a sphere of radius R centered at C. Then the change of signs of f_i among the geodesic $[x, y]$ on the sphere are as follows.*

- *If $f_i(x)$ and $f_i(y)$ have different signs, and there is exactly one point in $[x, y]$ where f_i vanishes.*
- *If $f_i(x)$ and $f_i(y)$ have same signs, there is either zero or two points in $[x, y]$ where f_i vanishes.*

Moreover, such intersection points are computable thanks to the following process. Put $\nu : x \mapsto \frac{x}{\|x\|}$ and let $u = \nu(x - c)$ and $v = \nu(y - c - ((y - c) \cdot u)u)$, forming an orthonormal basis of $\mathrm{Vect}(x - c, y - c)$. Then S_i intersects with $S \cap \mathrm{Aff}(c, x, y)$ at two points if and only if

$$\|w\|^2 R^2 \geq D_i^2, \tag{29}$$

where $w_i^x = (c_i - c) \cdot u$ and $w_i^y = (c_i - c) \cdot v$, $w_i = w_i^x u + w_i^y v$ (and thus $\|w_i\|^2 = (w_i^x)^2 + (w_i^y)^2$), $D_i = \frac{1}{2}\left(R_i^2 - \|c_i - c\|^2 - R^2\right)$. If Eq. 29 is verified, then these two points are of the form $C + su + tv$ with

$$(s, t) = \frac{D_i}{\|w\|^2}(w_i^x, w_i^y) \pm \sqrt{R^2 - \frac{D_i^2}{\|w_i\|^2}}(-w_i^y, w_i^y). \tag{30}$$

Furthermore, $c + su + tv$ lies on the geodesic $[x, y]$ on S if and only if $(y - c) \cdot u \leq s$ and $t \geq 0$.

Proof. Since by the definition of v one has $(y - c) \cdot v > 0$ we have $y - c = (y - c) \cdot u + \sqrt{R^2 - (y - c) \cdot u^2}v$, and any point $c + su + tv$ lies on the spherical geodesic $[x, y]$ if and only if $s^2 + t^2 = R^2$ and $(y - c) \cdot u \leq s, t \geq 0$. Now the orthogonal projection of c_i onto the plane $P = \mathrm{Aff}(c, x, y)$ is obtained by $\Pi_P(c_i) = c + ((c_i - c) \cdot u)u + ((c_i - c) \cdot v)v = c + w_i$. The intersection $S_i^P = S_i \cap P$ is either empty or a circle of radius $R_i^P = \sqrt{R_i^2 - \|c_i - c\|^2 + \|w_i\|^2}$ within P, centered at $c + w_i$, depending on the sign of $R_i^2 - \|c_i - c\|^2 + \|w_i\|^2$. S_i^P and S have non-empty intersection if and only if $\|w_i\| \leq R + R_i^P$.

Decomposed as $c + su + tv$, the intersecting points are exactly the solutions of the quadratic system $s^2 + t^2 = R^2, \|su + tv - w_i\|^2 = (R_i^P)^2$ which is equivalent to:

$$\begin{cases} s^2 + t^2 &= R^2 \\ sw_i^x + tw_i^y &= D_i \end{cases}$$

where $D_i = \frac{1}{2}\left(R^2 + \|c - c_i\|^2 - R_i^2\right)$. It is simpler to see the solution as $c + \alpha w_i + \beta w_i^\perp$, with $w_i^\perp = -w_i^y u + w_i^x v$. Then from the second equation of the last system we have $\alpha = \frac{D_i}{\|w_i\|^2}$ and $\beta = \pm\frac{1}{\|w_i\|}\sqrt{R^2 - \frac{D_i^2}{\|w_i\|^2}}$. □

Lemma. 7 (SphereDescent decreases F)**.** *Let x be a point of $\mathbb{R}^d$ which does not minimizes F, with $\#I_*^0(x) \geq 2$, $g = f_{I_*^+(x)}$. As per the previous proposition let y be the point of $S = S_{I_*^+(x)}$ where $g_{|S}$ reaches its minimum. Let $[x, y]_{|S}$ be the geodesic between x and y on S (or any geodesic if x and y are antipodal). Let λ be the minimum of t such that the projection x_t of $tx + (1 - t)y$ on S (or equivalently, onto $[x, y]_S$) has one of $I^+(x_t)$, $I^-(x_t)$ different from respectively $I_*^+(x)$ and $I_*^-(x)$. Then λ is well-defined and we have*

$$F(x_\lambda) < F(x). \tag{31}$$

Proof. By Lemma 3, we know that x belongs to the closure of $\mathcal{C}^*(x)$. Moreover for t small enough, x_t belongs to this cell as by definition x_t stays on S. Now on the trajectory $t \mapsto x_t$, F coincides with $f_{I_*^+}(x)$ which is a strictly convex function. Its restriction to S has a unique minimum attained at y; thus F it decreases along the geodesic $[x, y]_S$. □

Algorithm 4 – `SphereDescent`

Compute the first point along the geodesic between x, y on a Sphere S_{I^0} of center C, radius R leaving the cell and the cell to which this point belong.

Require: A set of centers c_i and radii R_i, an initial point x, destination y, sets of indices I^+, I^0, I^-.

Ensure: $z = C + su + \sqrt{R^2 - s^2}v$ is the first point along the geodesic between x, y on a sphere leaving the cell.

$(s, t) \leftarrow (-R, 0)$
$u \leftarrow \frac{x-C}{\|x-C\|}$
$v \leftarrow y - C - (y - C) \cdot u$
$v \leftarrow \frac{v}{\|v\|}$
ChangingIndices $\leftarrow \emptyset$ ▷ *Set of indices in $I^+ \cup I^-$ to which the final z belongs - usually a singleton.*
for $i \in I^+ \cup I^-$ **do**
  $D_i \leftarrow \frac{1}{2}\left(R_i^2 - \|c_i - C\|^2 - R^2\right)$
  $w_i^x \leftarrow (c_i - C) \cdot u, w_i^y \leftarrow (c_i - C) \cdot v$
  $w_i \leftarrow (w_i^x)^2 + (w_i^y)^2$
  $\Delta_i \leftarrow R^2 - \frac{D_i^2}{w_i}$
  if $\Delta_i > 0$ **then** ▷ *If the circle on which the geodesic lies intersects with S_i*
    $s_+ \leftarrow \frac{D_i}{w_i} w_i^x + \sqrt{\frac{\Delta_i}{w_i}} w_i^y, s_- \leftarrow \frac{D_i}{w_i} w_i^x - \sqrt{\frac{\Delta_i}{w_i}} w_i^y$
    $t_+ \leftarrow \frac{D_i}{w_i} w_i^y - \sqrt{\frac{\Delta_i}{w_i}} w_i^x, t_- \leftarrow \frac{D_i}{w_i} w_i^x + \sqrt{\frac{\Delta_i}{w_i}} w_i^x$
    $L \leftarrow$ set of pairs among $(s_+, t_+), (s_-, t_-)$ verifying $s. \geq s, t. \geq 0$.
    $(s_{temp}, t_{temp}) \leftarrow$ the pair with maximal first element in L.
    if $s_{temp} = s$ **then** ▷ *Degenerate case where $z = c + su + tv$ (at that point in the algorithm) already lies on a sphere S_i with $i \notin I^0$.*
      Add i to ChangingIndices
    else
      ChangingIndices $= [i]$
      $(s, t) \leftarrow (s_{temp}, t_{temp})$
Add elements of ChangingIndices to I^0, Remove elements from ChangingIndices off of I^+, I^-.
return $z = C + su + tv$, I^0, I^+, I^-

A.3 Exact algorithm : pseudocode and convergence result

Algorithm 5 – `SC-Exact-Solver`
Compute the minimum of $F = \sum_i \max(0, \|x - c_i\|^2 - R_i^2)$.

Require: A set of centers c_i and radii R_i, an initial point x_0
Ensure: x is the point where the minimum of F is attained

```
x ← x_0, MinimumAttained ← False.
I^+(x), I^0(x), I^-(x) ← Signs(x)
while MinimumAttained is False do
    if #I^0(x) then
        if c_{I^+(x)} ∈ C(x) then                          ▷ Check if the minimum lies in C(x).
            Teleportation procedure : MinimumAttained ← True, x ← c_{I^+(x)}.
        else
            ▷ Compute the lowest t such that x − t∇f_{I^+(x)} is in another cell using the LineDescent
              algorithm and update the signs. ◁
            x, I^+, I^0, I^- ← LineDescent (x, −∇f_{I^+(x)}).
    else                                                   ▷ x lies on at least one S_i
        ▷ Compute ∇_*F(x) and the sets of indices I_*^+(x), I_*^0(x), I_*^-(x) via ClarkeQP ◁
        I_*^+(x), I_*^0(x), I_*^-(x) ← ClarkeQP(x, I^+(x), I^0(x), I^-(x)).
        if 0 ∈ ∂_*F(x) then                                ▷ Check if the minimum is attained at x.
            MinimumAttained ← True
        else if #I_*^0(x) = 0 then                         ▷ Traj. out of x is in a full dimensional cell.
            if c_{I_*^+(x)} ∈ C^*(x) then                  ▷ Check if the minimum lies in C^*(x).
                Teleportation procedure : x ← c_{I_*^+(x)}, MinimumAttained ← True
            else
                ▷ Compute the lowest t such that x − t∇f_{I_*^+(x)} is in another cell using the LineDescent
                  algorithm and update the signs. ◁
                x, I^+, I^0, I^- ← LineDescent (x, −∇f_{I_*^+(x)})
        else                                               ▷ Trajectory out of x is not in a full dimensional cell.
            ▷ Compute c_S, R_S respectively the center/radius of S_{I_*^0(x)} via SphereIntersection. ◁
            c_S, R_S ← SphereIntersection (I_*^0(x)).
            ▷ Compute the argmin of f_{|I_*^+(x)} on S_{I_*^0(x)} using the MinSphereIntersection algorithm. ◁
            y ← MinSphereIntersection (I_*^+(x), c_S, R_S)
            if y ∈ C^*(x) then
                x ← y
            else
                ▷ Compute the first intersection of the geodesic from x to y on S with another cell using
                  the SphereDescent algorithm. Update the signs. ◁
                x, I^+, I^0, I^- ← SphereDescent (x, y, S)
return x
```

Red = Computations taking the minimum of roots of second degree polynomials.
Violet = Computing the solution of a quadratic programming problem.
Olive = Computations solving a full-rank linear system and roots of degree two polynomials.
Teal = Exact computations of predicates.

Theorem. 3 (Algorithm convergence)**.** *There exists a neighborhood of the point x^* where F reaches its minimum, such that for any starting point in this neighborhood the algorithm converges in at most 3^c operations, where c is the number of spheres on which x^* lies.*

Proof sketch. At point x^*, the minimum of F is reached meaning that its Clarke gradient $\partial_* F(x^*)$ contains 0. Recall that this Clarke gradient $\partial_* F(x)$ of F at any point x is a box of the form

$$\{\nabla f_{I^+(x)}(x) + \sum_{i \in I^0(x)} \alpha_i \nabla f_i(x), 0 \leq \alpha_i \leq 1\}$$

From classical geometric combinatorics, such a box has 3^c facets, where c is the cardinal of $I^0(x)$. For any point x with $I^0(x) \subset I^0(x^*)$, $I^+(x^*) \subset I^+(x), I^+(x^*) \subset I^+(x)$, $\partial_* F(x)$ is a box of smaller dimension than $\partial_* F(x^*)$, and it is similar to a facet of the box $\partial_* F(x^*)$.

The underlying trajectory followed by our algorithm takes direction in the Clarke gradient of F, so that it actually gets closer to x^*. Now take a (small enough) neighborhood U of x^* so that:

- the cells intersecting with U are exactly the cells whose adherence contains x^*;
- Any semiflow trajectory starting from a point in U stays in U (typically think of U as a ball centered in x^*, thanks to the previous remark).

By the convex nature of the functional our underlying semiflow trajectory has necessarily decreasing norm of generalized gradient - it is indeed following either the flow of a convex function on a submanifold, or branching when computing the Clarke gradient to a new generalized gradient with even smaller norm. Considering x_n the sequence of elements obtained by our algorithm starting from any point in U, this means that the sequence $\|\nabla_* F(x_n)\|$ is strictly decreasing.

Since the Clarke gradients $\partial_* F(x_n)$ are close[3] to facets of the box $\partial_* F(x^*)$, this suffices to obtain convergence. Ordering the facets by the distance of 0 to said facets yields shows that the sequence of facets of $\partial_* F(x)$ associated to $\partial_* F(x_n)$ is strictly decreasing, and must thus converge to the full box.

[3]In the sense that the vectors defining the boxes at points x_n, x^* are at distance smaller than $2\|x^* - x_n\|$

B Experiments

B.1 Datasets clustering

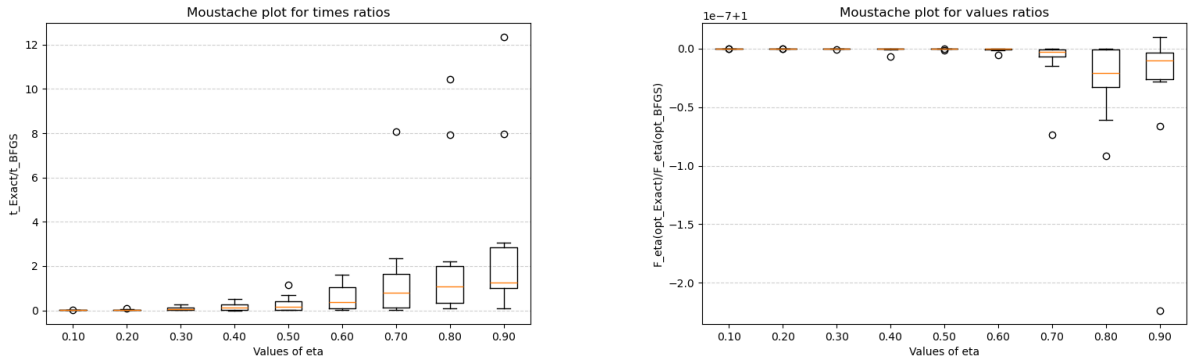
Name	Shape (n, d)
yeast	$(1484, 9)$
faults	$(1941, 28)$
mfeat	$(2000, 77)$
cloud	$(2048, 11)$
segmentation	$(2310, 20)$
spam	$(4601, 58)$
optdigits	$(5620, 65)$
landsat	$(6435, 37)$
pendigits	$(10992, 17)$
letter	$(20000, 17)$

Table 1: **The dataset `DS − Cluster`, consisting of 10 individual datasets.** The shape of a dataset: n: number of observations/points; d: dimension. Datasets selected in clustering experiments, see Celebi et al. (2013).

id	n	id	n	id	n	id	n
cluster_0	20	cluster_1	19	cluster_2	11	cluster_3	49
cluster_4	45	cluster_5	77	cluster_6	69	cluster_7	101
cluster_8	90	cluster_9	161	cluster_10	149	cluster_11	176
cluster_12	100	cluster_13	92	cluster_14	165	cluster_15	119

Table 2: **The `Proteins-HMM` dataset: 16 clusters in dimension $d = 400$ − from Vicedomini et al. (2022).** We use the clusters as individual datasets.

B.1.1 Dataset `DS-Cluster`: comparison `SC-Exact-Solver` vs `SC-BFGS-Solver`



Ratios $t_{\text{Exact}}/t_{\text{BFGS}}$ Ratios $F_\eta(\text{opt}_{\text{Exact}})/F_\eta(\text{opt}_{\text{BFGS}})$

Figure 7: `DS-Cluster`: `SC-Exact-Solver` vs `SC-BFGS-Solver`.

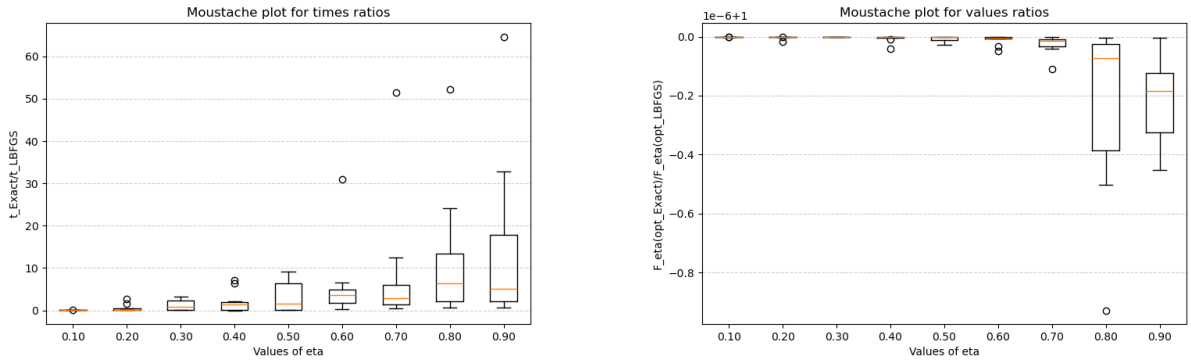
	min	median	max
$\eta = 0.10$	2.694e-04	1.001e-03	2.875e-03
$\eta = 0.20$	3.795e-04	6.018e-03	8.253e-02
$\eta = 0.30$	2.101e-04	4.999e-02	2.583e-01
$\eta = 0.40$	1.554e-04	1.112e-01	5.133e-01
$\eta = 0.50$	5.921e-04	1.635e-01	1.160e+00
$\eta = 0.60$	3.834e-03	3.567e-01	1.608e+00
$\eta = 0.70$	2.143e-02	7.780e-01	8.065e+00
$\eta = 0.80$	7.910e-02	1.076e+00	1.043e+01
$\eta = 0.90$	9.749e-02	1.252e+00	1.236e+01

Table 3: **DatasetCLU:** $t_{\textbf{Exact}}/t_{\textbf{BFGS}}$.

	min	median	max
$\eta = 0.10$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.20$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.30$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.40$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.50$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.60$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.70$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.80$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.90$	1.000e+00	1.000e+00	1.000e+00

Table 4: **DatasetCLU:** $F_\eta(\textbf{opt}_{\textbf{Exact}})/F_\eta(\textbf{opt}_{\textbf{BFGS}})$.

B.1.2 Dataset DS-Cluster: comparison SC-Exact-Solver vs SC-LBFGS-Solver



$t_{\text{Exact}}/t_{\text{L-BFGS-B}}$ $F_\eta(\text{opt}_{\text{Exact}})/F_\eta(\text{opt}_{\text{L-BFGS-B}})$

Figure 8: DS-Cluster: SC-Exact-Solver vs SC-LBFGS-Solver.

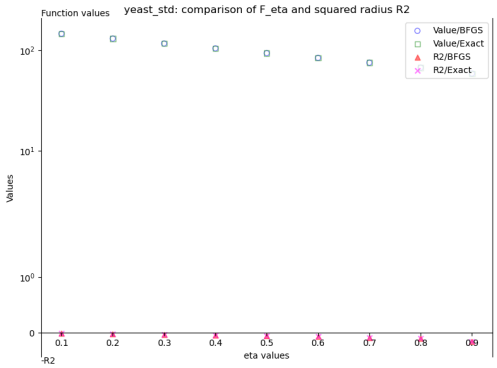
	min	median	max
$\eta = 0.10$	1.660e-03	8.070e-03	5.783e-02
$\eta = 0.20$	1.350e-03	1.403e-01	2.605e+00
$\eta = 0.30$	2.060e-03	7.913e-01	3.208e+00
$\eta = 0.40$	7.021e-04	1.304e+00	7.195e+00
$\eta = 0.50$	1.394e-02	1.521e+00	9.129e+00
$\eta = 0.60$	2.027e-01	3.627e+00	3.103e+01
$\eta = 0.70$	5.270e-01	2.777e+00	5.143e+01
$\eta = 0.80$	6.523e-01	6.386e+00	5.227e+01
$\eta = 0.90$	5.829e-01	5.068e+00	6.467e+01

Table 5: **DatasetCLU:** $t_{\textbf{Exact}}/t_{\textbf{L-BFGS-B}}$.

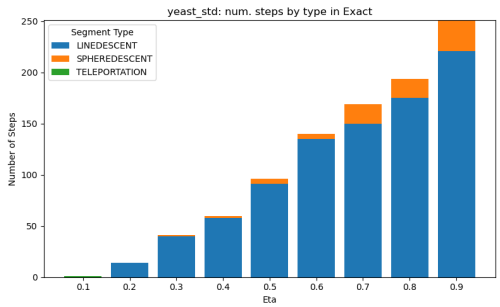
	min	median	max
$\eta = 0.10$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.20$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.30$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.40$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.50$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.60$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.70$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.80$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.90$	1.000e+00	1.000e+00	1.000e+00

Table 6: **DatasetCLU:** $F_\eta(\textbf{opt}_{\textbf{Exact}})/F_\eta(\textbf{opt}_{\textbf{L-BFGS-B}})$.

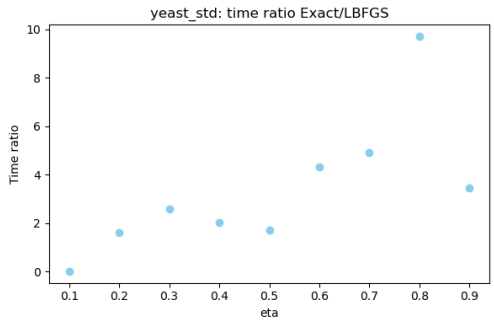
B.1.3 Illustration: DS-Cluster, dataset yeast_std



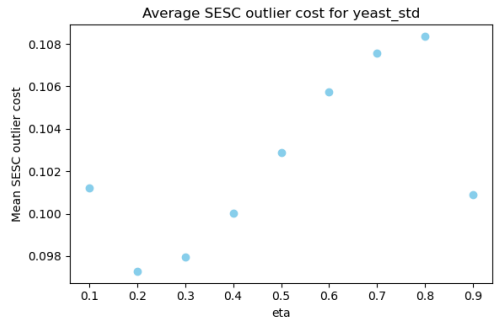
Dataset yeast_std: dual plot F_η and R^2



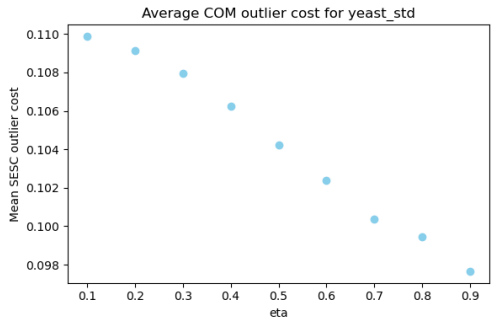
Dataset yeast_std: step types in trajectory



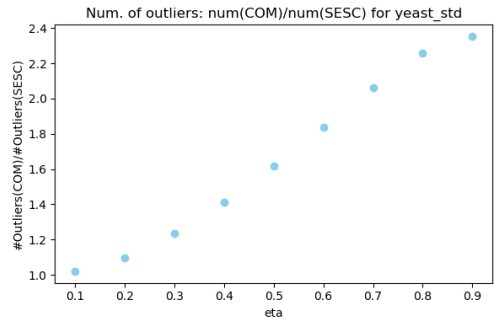
Ratio $t_{\text{Exact}}/t_{\text{BFGS}}$



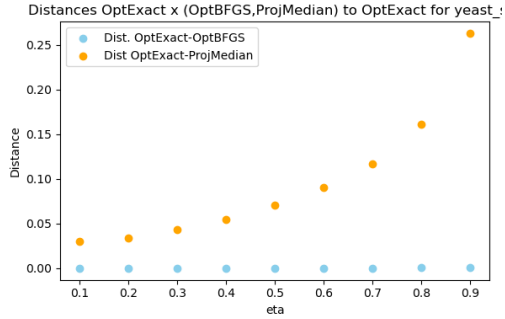
Mean SESC outlier cost



Mean COM outlier cost

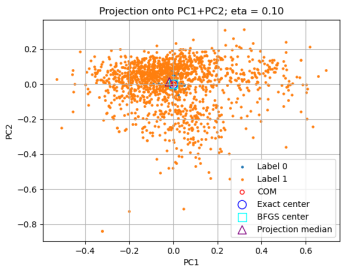


Outliers: num(COM)/num(SESC)

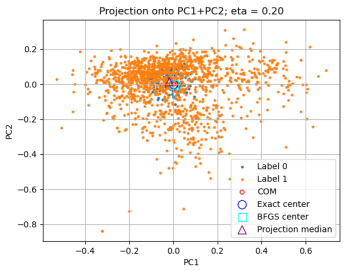


Distance between points

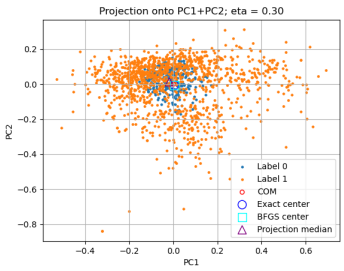
Figure 9: `DS-Cluster`, **dataset yeast_std – shape (1484, 9)**



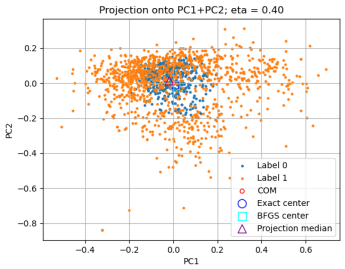
Projection plot yeast_std-, $\eta = 0.1$



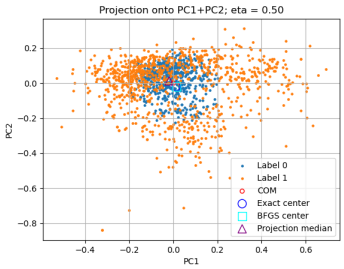
Projection plot yeast_std-, $\eta = 0.2$



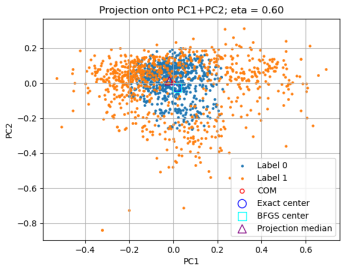
Projection plot yeast_std-, $\eta = 0.3$



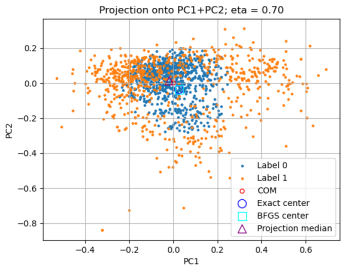
Projection plot yeast_std-, $\eta = 0.4$



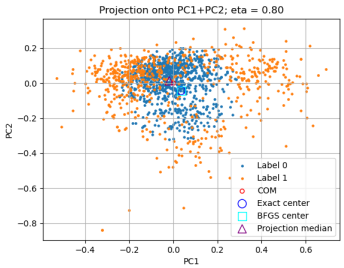
Projection plot yeast_std-, $\eta = 0.5$



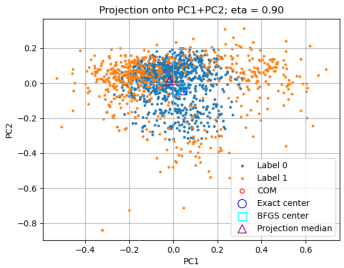
Projection plot yeast_std-, $\eta = 0.6$



Projection plot yeast_std-, $\eta = 0.7$



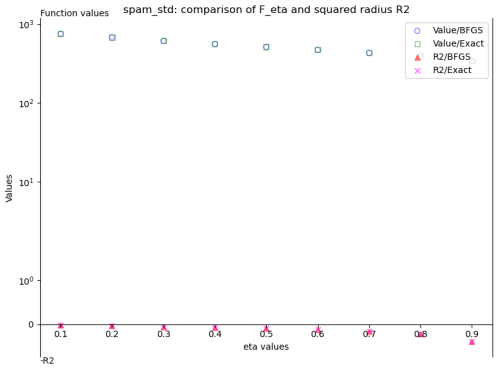
Projection plot yeast_std-, $\eta = 0.8$



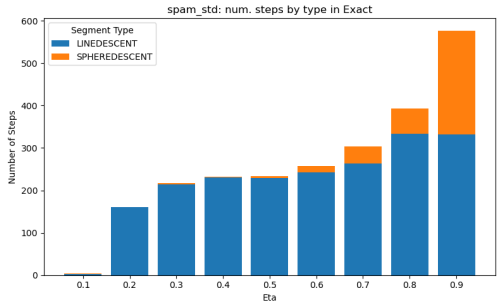
Projection plot yeast_std-, $\eta = 0.9$

Figure 10: `DS-Cluster`, **dataset yeast_std – shape (1484, 9)**

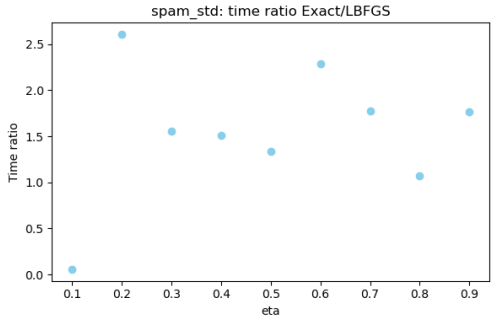
B.1.4 Illustration: `DS-Cluster`, dataset yeast_std



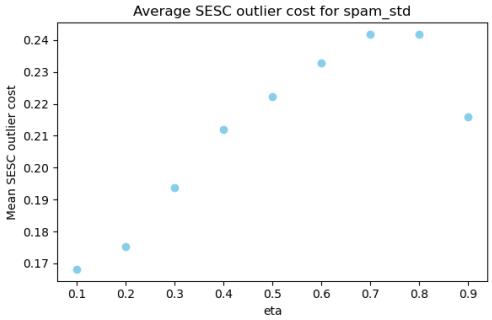
Dataset spam_std: dual plot F_η and R^2



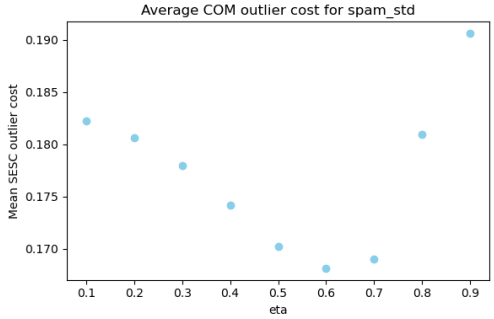
Dataset spam_std: step types in trajectory



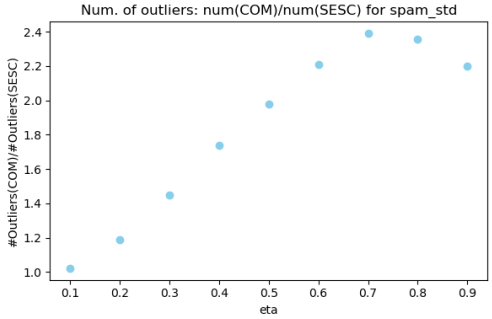
Ratio $t_{\text{Exact}}/t_{\text{BFGS}}$



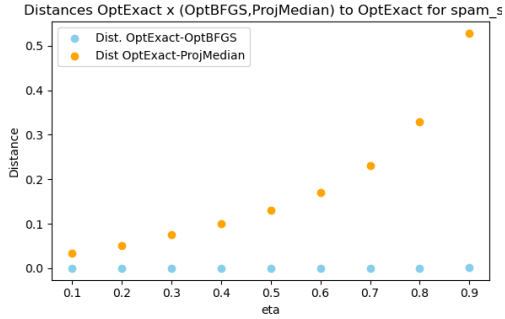
Mean SESC outlier cost



Mean COM outlier cost

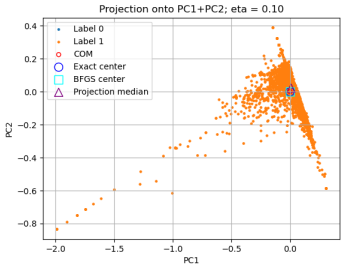


Outliers: num(COM)/num(SESC)

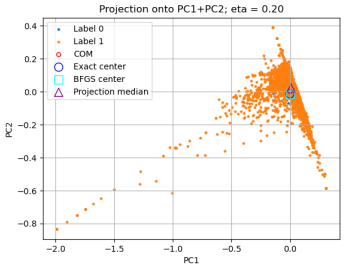


Distance between points

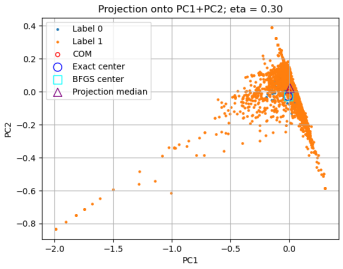
Figure 11: **Dataset spam_std – shape (4601, 58)**



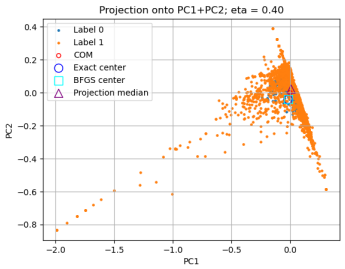
Projection plot spam_std-, $\eta = 0.1$



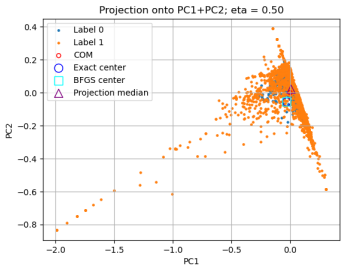
Projection plot spam_std-, $\eta = 0.2$



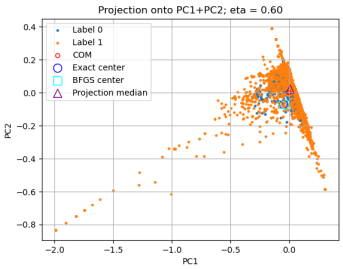
Projection plot spam_std-, $\eta = 0.3$



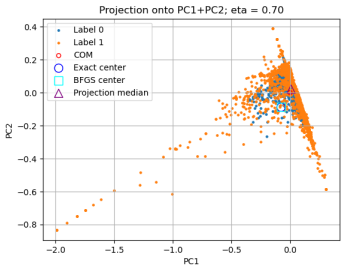
Projection plot spam_std-, $\eta = 0.4$



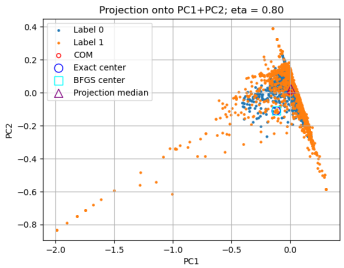
Projection plot spam_std-, $\eta = 0.5$



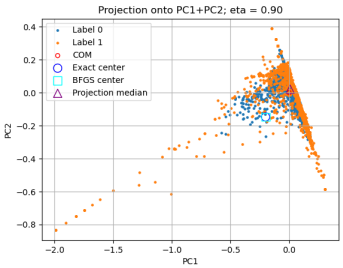
Projection plot spam_std-, $\eta = 0.6$



Projection plot spam_std-, $\eta = 0.7$



Projection plot spam_std-, $\eta = 0.8$

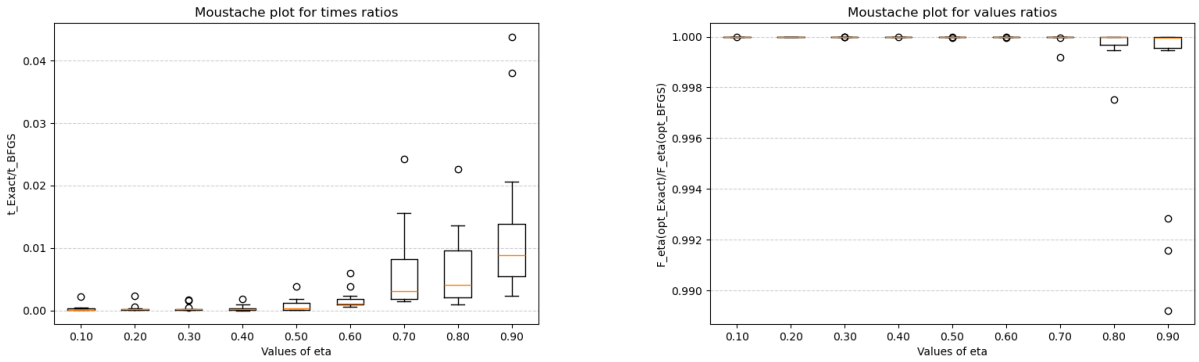


Projection plot spam_std-, $\eta = 0.9$

Figure 12: **Dataset spam_std – shape (4601, 58)**

B.2 Dataset `Proteins-HMM`

B.2.1 `Proteins-HMM`: comparison `SC-Exact-Solver` vs `SC-BFGS-Solver`



Ratios $t_{\text{Exact}}/t_{\text{BFGS}}$ Ratios $F_\eta(\text{opt}_{\text{Exact}})/F_\eta(\text{opt}_{\text{BFGS}})$

Figure 13: `Proteins-HMM`: `SC-Exact-Solver` vs `SC-BFGS-Solver`.

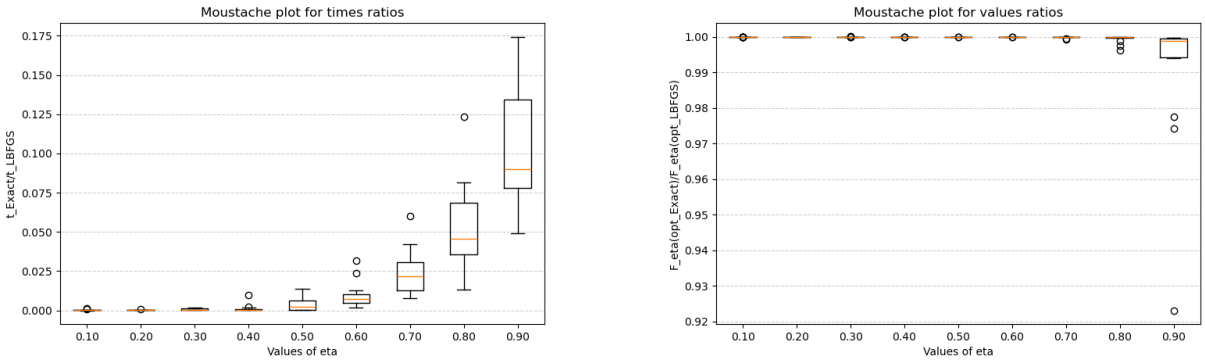
	min	median	max
$\eta = 0.10$	2.355e-05	1.253e-04	2.194e-03
$\eta = 0.20$	1.585e-05	1.434e-04	2.318e-03
$\eta = 0.30$	1.021e-06	1.457e-04	1.723e-03
$\eta = 0.40$	4.014e-07	1.507e-04	1.834e-03
$\eta = 0.50$	1.213e-06	2.618e-04	3.800e-03
$\eta = 0.60$	5.484e-04	1.088e-03	5.961e-03
$\eta = 0.70$	1.460e-03	3.091e-03	2.425e-02
$\eta = 0.80$	9.693e-04	4.121e-03	2.256e-02
$\eta = 0.90$	2.363e-03	8.852e-03	4.387e-02

Table 7: **ProtHMM:** $t_{\mathbf{Exact}}/t_{\mathbf{BFGS}}$**.**

	min	median	max
$\eta = 0.10$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.20$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.30$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.40$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.50$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.60$	9.999e-01	1.000e+00	9.999e-01
$\eta = 0.70$	9.992e-01	1.000e+00	9.992e-01
$\eta = 0.80$	9.975e-01	1.000e+00	9.975e-01
$\eta = 0.90$	9.892e-01	1.000e+00	9.892e-01

Table 8: **ProtHMM:** $F_\eta(\mathbf{opt}_{\mathbf{Exact}})/F_\eta(\mathbf{opt}_{\mathbf{BFGS}})$**.**

B.2.2 `Proteins-HMM`: comparison `SC-Exact-Solver` vs `SC-LBFGS-Solver`



Ratios $t_{\text{Exact}}/t_{\text{L-BFGS-B}}$ Ratios $F_\eta(\text{opt}_{\text{Exact}})/F_\eta(\text{opt}_{\text{L-BFGS-B}})$

Figure 14: `Proteins-HMM`: `SC-Exact-Solver` vs `SC-LBFGS-Solver`.

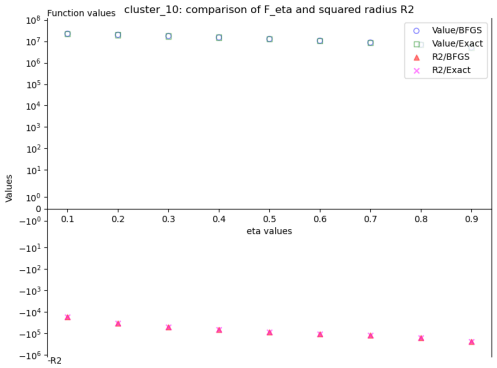
	min	median	max
$\eta = 0.10$	5.327e-05	1.287e-04	1.508e-03
$\eta = 0.20$	7.769e-05	1.436e-04	7.514e-04
$\eta = 0.30$	6.607e-05	2.455e-04	1.695e-03
$\eta = 0.40$	6.549e-05	2.572e-04	9.777e-03
$\eta = 0.50$	9.009e-05	2.133e-03	1.386e-02
$\eta = 0.60$	1.740e-03	7.108e-03	3.145e-02
$\eta = 0.70$	7.868e-03	2.164e-02	6.013e-02
$\eta = 0.80$	1.322e-02	4.556e-02	1.233e-01
$\eta = 0.90$	4.903e-02	8.999e-02	1.744e-01

Table 9: **ProtHMM:** $t_{\textbf{Exact}}/t_{\textbf{L-BFGS-B}}$**.**

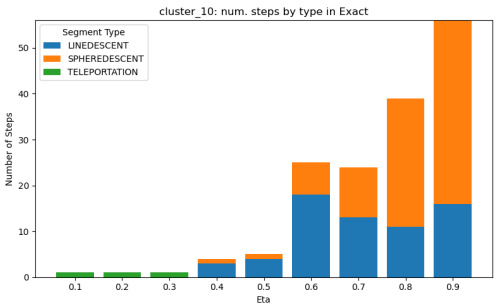
	min	median	max
$\eta = 0.10$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.20$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.30$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.40$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.50$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.60$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.70$	9.991e-01	1.000e+00	9.991e-01
$\eta = 0.80$	9.962e-01	9.999e-01	9.962e-01
$\eta = 0.90$	9.232e-01	9.987e-01	9.232e-01

Table 10: **ProtHMM:** $F_\eta(\mathbf{opt}_{\textbf{Exact}})/F_\eta(\mathbf{opt}_{\textbf{L-BFGS-B}})$**.**

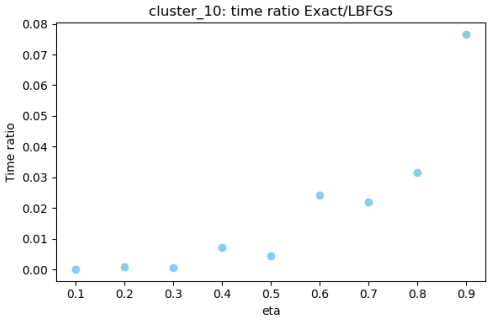
B.2.3 Illustration: `Proteins-HMM`, Cluster 10



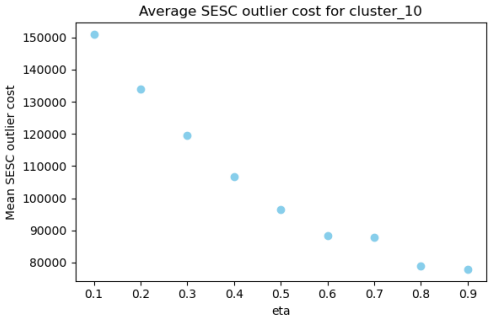
Dataset cluster_10: dual plot F_η and R^2



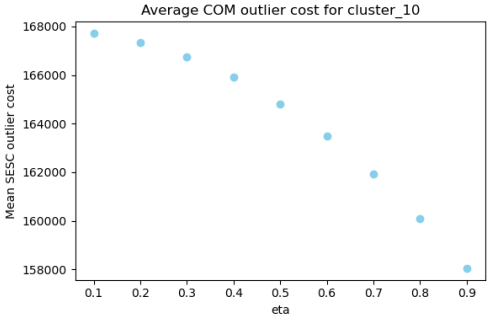
Dataset cluster_10: step types in trajectory



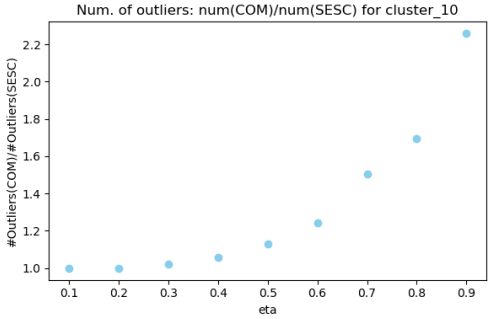
Ratio $t_{\text{Exact}}/t_{\text{BFGS}}$



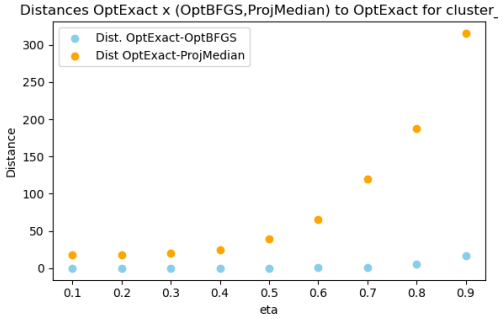
Mean SESC outlier cost



Mean COM outlier cost

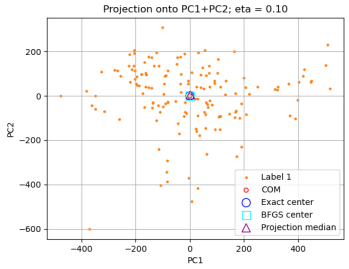


Outliers: num(COM)/num(SESC)

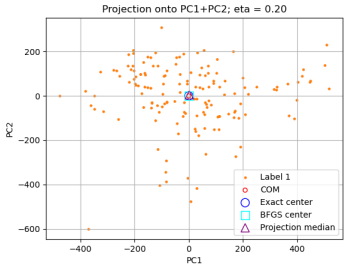


Distance between points

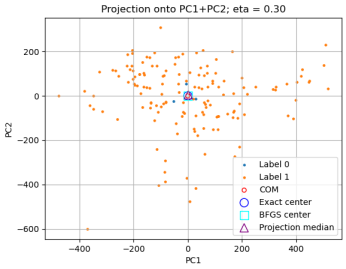
Figure 15: **Dataset cluster_10 – shape (149, 400)**



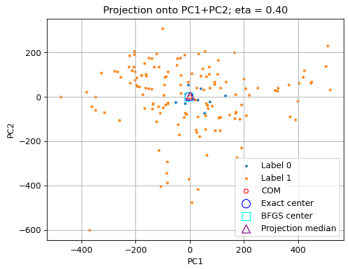
Projection plot cluster_10-, $\eta = 0.1$



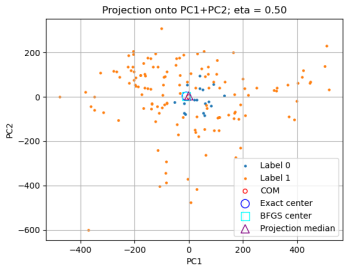
Projection plot cluster_10-, $\eta = 0.2$



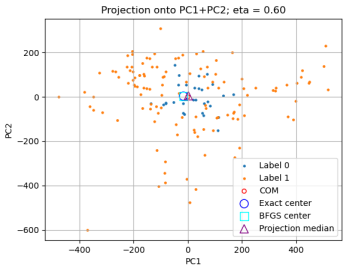
Projection plot cluster_10-, $\eta = 0.3$



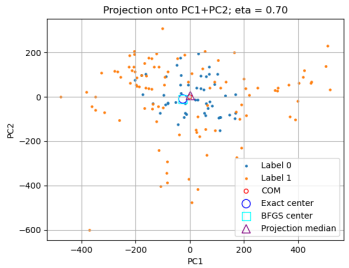
Projection plot cluster_10-, $\eta = 0.4$



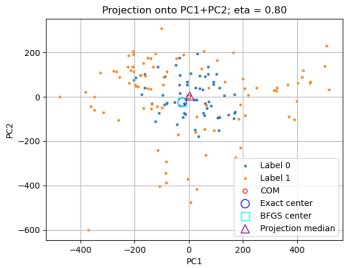
Projection plot cluster_10-, $\eta = 0.5$



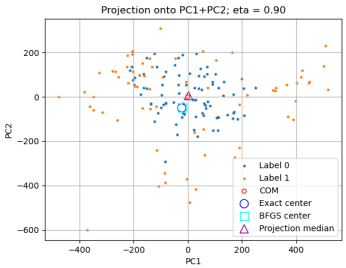
Projection plot cluster_10-, $\eta = 0.6$



Projection plot cluster_10-, $\eta = 0.7$



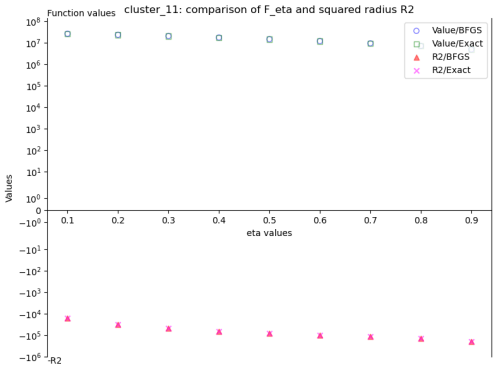
Projection plot cluster_10-, $\eta = 0.8$



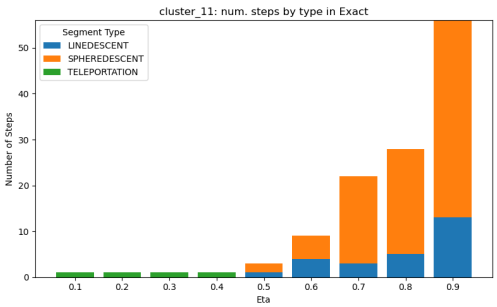
Projection plot cluster_10-, $\eta = 0.9$

Figure 16: **Dataset cluster_10 – shape (149, 400)**

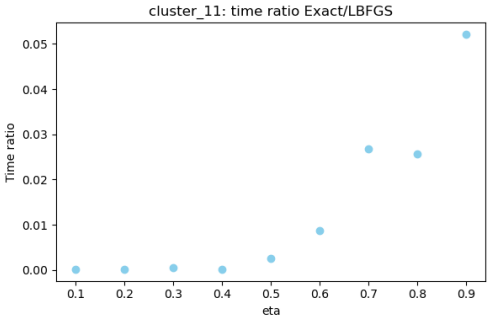
Illustration: `Proteins-HMM`**, Cluster 11**



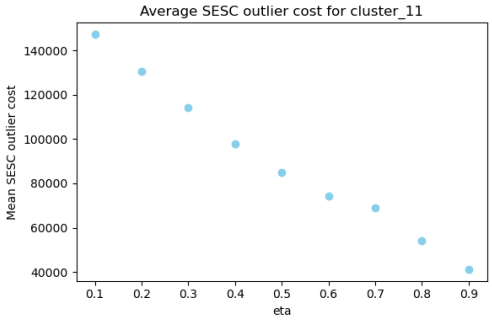
Dataset cluster_11: dual plot F_η and R^2



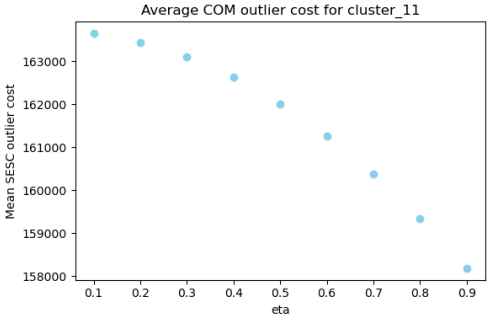
Dataset cluster_11: step types in trajectory



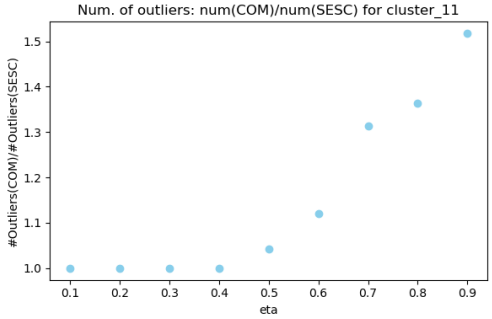
Ratio $t_{\text{Exact}}/t_{\text{BFGS}}$



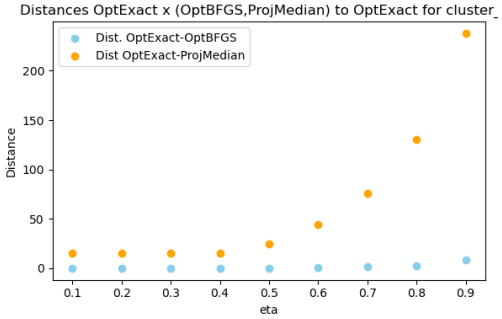
Mean SESC outlier cost



Mean COM outlier cost

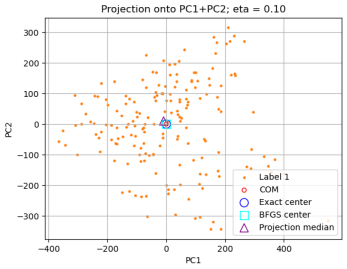


Outliers: num(COM)/num(SESC)

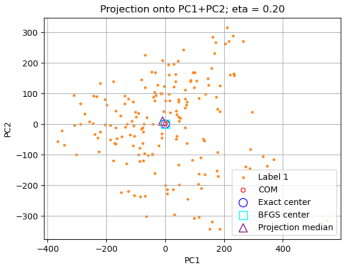


Distance between points

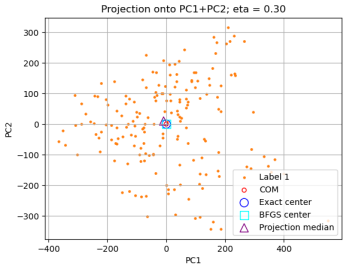
Figure 17: **Dataset cluster_11 – shape (176, 400)**



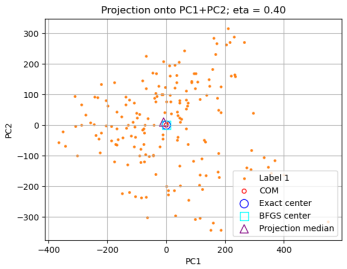
Projection plot cluster_11-, $\eta = 0.1$



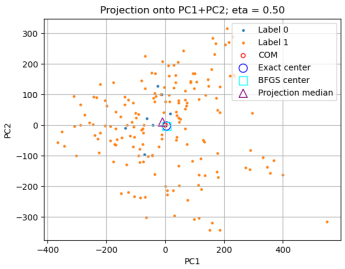
Projection plot cluster_11-, $\eta = 0.2$



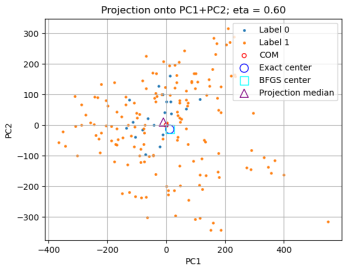
Projection plot cluster_11-, $\eta = 0.3$



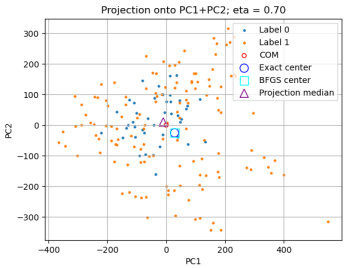
Projection plot cluster_11-, $\eta = 0.4$



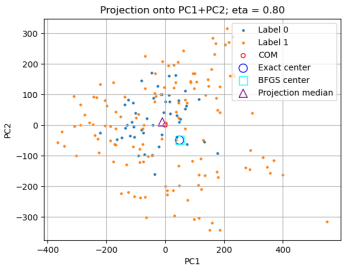
Projection plot cluster_11-, $\eta = 0.5$



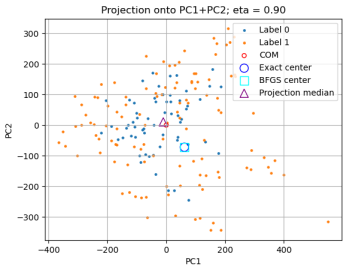
Projection plot cluster_11-, $\eta = 0.6$



Projection plot cluster_11-, $\eta = 0.7$



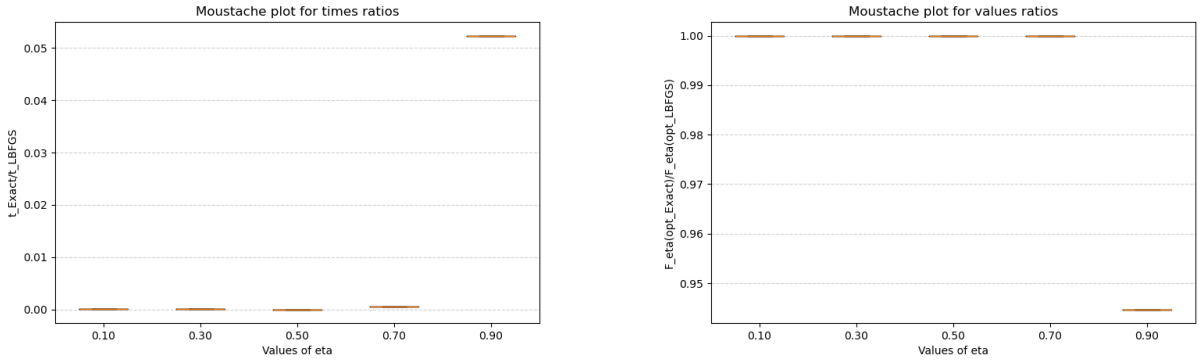
Projection plot cluster_11-, $\eta = 0.8$



Projection plot cluster_11-, $\eta = 0.9$

Figure 18: **Dataset cluster_11 – shape (176, 400)**

B.2.4 Dataset Arcene



Ratios $t_{\text{Exact}}/t_{\text{BFGS}}$ Ratios $F_\eta(\text{opt}_{\text{Exact}})/F_\eta(\text{opt}_{\text{L-BFGS-B}})$

Figure 19: **Arcene:** `SC-Exact-Solver` **vs** `SC-LBFGS-Solver`**.**

	min	median	max
$\eta = 0.10$	4.559e-05	4.559e-05	4.559e-05
$\eta = 0.30$	5.732e-05	5.732e-05	5.732e-05
$\eta = 0.50$	4.431e-05	4.431e-05	4.431e-05
$\eta = 0.70$	5.946e-04	5.946e-04	5.946e-04
$\eta = 0.90$	5.234e-02	5.234e-02	5.234e-02

Table 11: **DatasetHD:** $t_{\textbf{Exact}}/t_{\textbf{L-BFGS-B}}$**.**

	min	median	max
$\eta = 0.10$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.30$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.50$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.70$	1.000e+00	1.000e+00	1.000e+00
$\eta = 0.90$	9.447e-01	9.447e-01	9.447e-01

Table 12: **DatasetHD:** $F_\eta(\textbf{opt}_{\textbf{Exact}})/F_\eta(\textbf{opt}_{\textbf{L-BFGS-B}})$**.**