FITTING A SIMPLICIAL COMPLEX USING A VARIATION OF $k$-MEANS

PIOTR BEBEN

Abstract. In this paper we give a simple two stage algorithm for approximating a point cloud $S \subset \mathbb{R}^m$ by a simplicial complex $K$. The first stage is an iterative fitting procedure that generalizes $k$-means clustering, while the second stage involves deleting redundant simplices. A form of dimension reduction of $S$ is obtained as a consequence.

1. Introduction

The use of simplicial complexes as a means for estimating topology has found many applications to data analysis in recent times. For example, unsupervised learning techniques such as persistent homology \cite{cech,vietoris} often use what are known as Cech or Vietoris-Rips filtrations to capture multiscale topological properties of a point cloud $S \subset \mathbb{R}^m$. The simplicial complexes in these filtrations individually are not always a good representation of the actual physical shape of $S$ since, for example, they often have a higher dimension than $\mathbb{R}^m$. Our aim is to give an algorithm that can approximate $S$ to the greatest extent possible when fed any simplicial complex $K$ mapped linearly into $\mathbb{R}^m$. This algorithm has several nice properties, including a tendency towards preserving embeddings, as well as reducing to $k$-means clustering when $K$ is 0-dimensional. The resulting fitting is further refined by deleting simplices that have been poorly matched with $S$; the end result being a locally linear approximation of $S$. A lower dimensional representation of $S$ in terms of barycentric coordinates follows by projecting onto this approximation.

2. Algorithm Description

Fix $K$ be a (geometric) simplicial complex, and let $\{v_1, \ldots, v_n\} \in K$ be the set of vertices of $K$. The facets of $K$ are the simplices of $K$ that have the highest dimension in the sense that they are not contained in the boundary of any other simplex. $K$ may then be represented as a collection of facets, each of which is represented by the set of vertices that it contains. The dimension of a facet is equal to the number of vertices it contains minus 1. When we refer to the boundary $\partial \sigma$ of a simplex $\sigma$, we mean the union of its smaller dimensional boundary simplices, even when the simplex is embedded as a subset of $\mathbb{R}^m$. Its interior is the simplex minus this boundary.

Any point $x \in K$ is contained in a unique smallest dimensional simplex $\sigma_x \subseteq K$. We may represent $x$ uniquely as a convex combination

$$x = \sum_{v_j \in \sigma_x} \lambda_{j,x} v_j$$

over the vertices $v_j$ that are contained in $\sigma_x$, where $\sum_{v_j \in \sigma_x} \lambda_{j,x} = 1$ for some barycentric coordinates $\lambda_{j,x} \geq 0$. A map $g: K \to \mathbb{R}$ is said to be linear if it is linear on each simplex of $K$. Namely,

$$g(x) = \sum_{v_j \in \sigma_x} \lambda_{j,x} g(v_j).$$

So $g$ restricts to a linear embedding on each simplex of $K$, and is uniquely determined by the values $g(v_j)$ that it takes on its vertices $v_j$. Thus, we have a convenient representation of $g$ in terms of the $g(v_j)$’s.

2.1. Fitting. Fix $S \subset \mathbb{R}^m$ our finite set of data points. Suppose $f: K \to \mathbb{R}^m$ is any choice of linear map, meant to represent our initial fitting of $K$ to $S$. Starting with $f$, our aim is to obtain successively better fittings $f^\ell: K \to \mathbb{R}^m$ of $K$ to $S$, at each iteration giving a better reflection of the shape and structure of $S$. We do this as follows.
Algorithm 1 (Simplicial Means) First stage: fitting $K$ to $S$

1: set $f^0 \leftarrow f$ and $\ell \leftarrow -1$;
2: \textbf{repeat}
3: \hspace{1em} increment $\ell \leftarrow \ell + 1$;
4: \hspace{1em} \textbf{for} each $y \in S$ \textbf{do}
5: \hspace{2em} find a choice of $y' \in K$ such that $f^\ell(y')$ is nearest to $y$, and $\sigma_y \subseteq K$ the smallest dimensional simplex that contains $y'$;
6: \hspace{2em} compute $\lambda_{jy} \geq 0$ such that $\sum \lambda_{jy} = 1$ and $y'$ is the convex combination
7: \hspace{2em} $y' = \sum_{v_j \in \sigma_y} \lambda_{jy} v_j$.
8: \hspace{1em} \textbf{end for}
9: \hspace{1em} using $f^\ell$, construct a new linear map $f^{\ell+1}: K \rightarrow \mathbb{R}^m$, defined on each vertex $v_j$ by setting
10: \hspace{2em} $f^{\ell+1}(v_j) \leftarrow \frac{1}{|N_j^\ell|} \sum_{y \in N_j^\ell} \left( ((1 - \lambda_{jy}) f^\ell(v_j) + \lambda_{jy} y) \right)$
11: \hspace{1em} when $|N_j^\ell| > 0$, and $f^{\ell+1}(v_j) \leftarrow f^\ell(v_j)$ otherwise, where
12: \hspace{2em} $N_j^\ell = \{ y \in S \mid v_j \in \sigma_y \}$.
13: \hspace{1em} \textbf{until} a given stop condition has not been reached (e.g. $\max_j \|f^{\ell+1}(v_j) - f^\ell(v_j)\|$ is small)
14: \hspace{1em} set $g \leftarrow f^\ell$.
15: \hspace{1em} \textbf{return} $g$, each $\sigma_y$, $y'$, and the barycentric coordinates $\lambda_{jy}$.

The $|N_j^\ell| > 0$ assignment for $f^{\ell+1}(v_j)$ in Step 8 can be generalized to

$$f^{\ell+1}(v_j) \leftarrow \frac{1}{|N_j^\ell|} \sum_{y \in N_j^\ell} \left( \frac{1 - \lambda_{jy}}{1 + s} f^\ell(v_j) + \frac{\lambda_{jy} + s}{1 + s} y \right)$$

where $s \geq 0$ is the learning rate. Larger values for $s$ lead to faster convergence, but poorer fittings and reduced stability. None-the-less, taking $s$ to be small ($\leq 0.1$) but nonzero has every advantage in addition to preventing a simplex becoming stuck when the barycentric coordinates $\lambda_{jy}$ associated with its vertices are all near zero.

The intuition underlying Algorithm 1 is simple. Consider the case where at the $\ell^{th}$ iteration $f^\ell$ is a linear embedding. Then $v_j$ and $K$ can then be thought of as $f^\ell(v_j)$ and the subspace $f^\ell(K) \subseteq \mathbb{R}^m$, and Algorithm 1 in essence has $S$ attract $K$ towards it by having each point $y \in S$ exert a pull on a nearest point $y' \in K$. The caveat here is that in doing so the embedding $f^\ell$ must be kept linear, so the net effect of this pull must come down to its influence on the individual vertices $v_j$ of the simplex containing $y'$. The influence of each of these vertices $v_j$ should in turn decrease with some measure of distance of $y'$ to $v_j$. This is analogous to pulling on a string attached at some point along a perpendicular uniform rod floating in space, in which case the acceleration of an endpoint of the rod in the direction of the pull decreases with increasing distance from the string. Since the size or shape of the simplex in our context is irrelevant, the distance from $y'$ to $v_j$ is measured in terms of its barycentric coordinates $\lambda_{jy}$. In particular, if $y'$ lies near a boundary simplex opposite to $v_j$, then $\lambda_{jy}$ is near 0 and $y$ has little influence on $v_j$, while $y$ has full influence on $v_j$ when $\lambda_{jy} = 1$, or equivalently, when $y' = v_j$. The accumulation of these pulling forces on a vertex $v_j$ leads us to take the mean of $\lambda_{jy} v_j + \lambda_{jy} y$ over all $y \in N_j^\ell$; equivalently, over all $y \in S$ that are closest to a point $y'$ lying on the interior of a simplex that has $v_j$ as a vertex.

When $K$ is 0-dimensional, namely a collection of $n$ disjoint vertices $v_j$, then there is only a single barycentric coordinate $\lambda_{jy} = 1$ for each $y'$, so Algorithm 1 reduces to the classical $k$-means algorithm with clusters $v_j$. Algorithm 1 can therefore be seen as a high dimensional non-discrete generalization of $k$-means clustering. This is perhaps in the same spirit as persistent homology is a higher dimensional generalization of hierarchical clustering by way of simplicial complexes.
2.2. Preserving embeddings. To obtain a good fitting $g$, why not simply apply the $k$-means algorithm to the vertices $f(v_j)$? The answer is that it would not be a very good fitting since it forgets the arrangement of simplices spanning our vertices. One might still hope that forgetting simplices would not be a problem when $K$ has a very regular triangulation such as that of a mesh, since the triangulation itself carries little information. However, it is unlikely that $g$ would be an embedding (irrespective of $f$), and as such, $g$ would still be a poor reflection of the actual shape and structure of $S$. Take for example the case where $K$ is the following four vertex graph embedded in $\mathbb{R}^m = \mathbb{R}^2$

- $a$

  ![Graph Image]

and $S = \{a, b, c, y\} \subset \mathbb{R}^2$. The nearest vertex to $a$, $b$, $c$, and $y$ is $v_1$, $v_3$, $v_4$, and $v_2$ respectively, so one iteration of $k$-means on this data results in the edge $\{v_1, v_2\}$ intersecting the edge $\{v_3, v_4\}$. On the other hand, the nearest point to $y$ in $K$ lies on the edge $\{v_1, v_2\}$, so a single iteration of Algorithm 1 results in $K$ being stretched out towards $S$ without violating its embedding. In fact, for non-pathological $K$, Algorithm 1 has a strong tendency to retain $g: K \rightarrow \mathbb{R}^m$ (and each $f^j$) as embeddings, given that the initial fitting $f: K \rightarrow \mathbb{R}^m$ is an embedding.

2.3. Refinement. Once we have obtained a sufficiently good fitting $g: K \rightarrow \mathbb{R}^m$ from Algorithm 1, the next step is to refine it by getting rid of redundant simplices. These are the simplices $\sigma \subseteq K$ whose linearly embedded image $g(\sigma)$ has no points $g(y') \in g(\sigma)$ positioned near its interior, but instead, near its boundary $\partial g(\sigma) = g(\partial \sigma)$. We then project those $g(y')$ near the boundary into one of the simplices on the boundary in order to obtain a lower dimensional approximate representation of $y$. This process is repeated for the projections.

Algorithm 2 Second stage: deleting redundant simplices, reducing dimension

```plaintext
1: input: a threshold value $\alpha \geq 0$, and each $y'$, $\sigma_y$, and $g: K \rightarrow \mathbb{R}^m$ from Algorithm 1
2: for each $y \in S$ do
3:    set $z \leftarrow y'$ and $\sigma \leftarrow \sigma_y$;
4:    while $\sigma$ is not a vertex do
5:        find a choice of $\tilde{z}$ in $\partial \sigma$ such that $g(z)$ is nearest to $g(\tilde{z})$, and $\tilde{\sigma} \subseteq \partial \sigma$ the smallest dimensional boundary simplex that contains $\tilde{z}$;
6:        if $\|[g(\tilde{z}) - g(z)]\| \leq \alpha$ then
7:            set $z \leftarrow \tilde{z}$ and $\sigma \leftarrow \tilde{\sigma}$;
8:        else
9:            exit the while loop;
10:   end if
11: end while
12: set $\tilde{y} \leftarrow z$ and $\tilde{\sigma}_y \leftarrow \sigma$, and $\tilde{\lambda}_{jy}$ (for each $v_j \in \tilde{\sigma}_y$) the barycentric coordinates of $\tilde{y}$ in $\tilde{\sigma}_y$;
13: end for
14: set $\tilde{K} \leftarrow \bigcup_{y \in S} \tilde{\sigma}_y$;
15: return $\tilde{K}$, and each $\tilde{y}$, $\tilde{\sigma}_y$, and $\tilde{\lambda}_{jy}$.
```
The output \( \tilde{K} \) is a subcomplex of \( K \) with \( g: K \rightarrow \mathbb{R}^m \) restricting to a linear map
\[ g: \tilde{K} \rightarrow \mathbb{R}^m, \]
and if \( g \) is a good fitting, \( g(\tilde{K}) \) is a locally linear approximation of \( \mathcal{S} \), and each \( g(\hat{y}) \) will often be a close approximation of the corresponding \( y \in \mathcal{S} \). Depending on how many simplices were deleted from \( K \) to get to \( \tilde{K} \), and how far down we went in projecting each \( g(y') \) into boundary simplices, \( \tilde{K} \) will typically have lower dimensional facets than \( K \). At the same time, its facets and simplices will often have much lower dimension than \( \mathbb{R}^m \), so each \( y \) will have a lower dimensional approximate representation in terms of the barycentric coordinates \( \hat{\lambda}_{j,y} \) of \( \hat{y} \) in \( \hat{\sigma}_y \). For example, if \( \hat{\sigma}_y \) happens to be a 1-dimensional simplex (edge) with vertices \( v_{j_1} \) and \( v_{j_2} \), then \( y \approx g(\hat{y}) = t g(v_{j_1}) + (1-t) g(v_{j_2}) \) where \((t,1-t) \in \mathbb{R}^2 \) are the barycentric coordinates of \( \hat{y} \) in \( \hat{\sigma}_y \). So \( y \) can be approximately represented by the barycentric coordinates \((t,1-t) \) together with an assignment of simplex \( y \rightarrow \hat{\sigma}_y \) (i.e. an assignment of those vertices in \( \hat{\sigma}_y \)). This generally gives a much more compact representation of \( y \), especially when \( m \) is large. In summary, points \( y \in \mathcal{S} \) are assigned to linear combinations of vertices \( g(v_j) \) of a nearest simplex to give a lower dimensional representation.

2.4. Some comparisons. This approach to dimension reduction is similar to that taken by \( k \)-means and self-organizing maps (SOM) [9, 8]. The assignment of vertices (nodes, neurons, classes, or clusters, in different language) however does not have to be discrete, so the reduction is not necessarily one to dimension 0 for every \( y \in \mathcal{S} \). Instead, it can be seen as a form of Fuzzy \( k \)-means [11,18], where multiple vertices in \( g(\tilde{K}) \) assigned to \( y \) to varying degrees in terms of barycentric coordinates, this being dependent on which simplex is nearest to \( y \) and where the nearest point \( y' \) lies inside this simplex. Like SOM (and unlike variants of \( k \)-means) the construction of \( g \) is topology preserving, with nearby or adjacent vertices of \( K \) and \( \tilde{K} \) tending to have nearby points in \( \mathcal{S} \) assigned to them. SOM however acts on individual vertices, using an explicit neighbourhood function on nearby vertices to effect this property. Since simplicial complexes are able to more efficiently reflect the shape of an object such as our point cloud \( \mathcal{S} \), one would expect that Algorithms 1 and 2 give a better approximation of \( \mathcal{S} \) than SOM, at the same time using fewer vertices. This is all at the expense of our reduction generally being above dimension 0 dependent on \( \tilde{K} \).

The end result of Algorithms 1 and 2 is that local patches in \( \mathcal{S} \) are approximated by the affine subspaces of \( \mathbb{R}^m \) determined by the simplices of \( g(\tilde{K}) \). This is similar to Cluster-PCA [6], as well as the first stage of Locally Linear Embedding (LLE) [12]. In Cluster-PCA a collection of affine \( d \)-dimensional subspaces of \( \mathbb{R}^m \) are iteratively fitted to their nearest points in \( \mathcal{S} \) via PCA, while LLE uses convex combinations of \( k \) nearest neighbours to each point in \( \mathcal{S} \) (sampled from a smoothly embedded \( d \)-manifold) in order to obtain an approximation of a local coordinate patch. In our case \( \mathcal{S} \) need not be a manifold, nor does the dimension \( d \) of the affine subspaces have to be known beforehand. Instead, one has to select a sufficiently high dimensional \( K \) with sufficiently many vertices (but not so many so as to lead to over-fitting) in order to obtain a good approximation of \( \mathcal{S} \). None-the-less, we are left with a convenient approximation of the shape of \( \mathcal{S} \) in terms of \( K \) and \( g \). These unsupervised learning techniques aside, there are also various shape approximation techniques used amongst computer graphics community, where, for example, a mesh is iteratively fitted to data by solving certain least-squares optimization problems [7].

2.5. Further dimension reduction. If \( g: \hat{K} \rightarrow \mathbb{R}^m \) happens to be an embedding and \( \hat{K} \) has dimension \( d \) with \( d \) much smaller than \( m \), then one would like to go a step further and obtain a lower dimensional embedding \( h: \hat{K} \rightarrow \mathbb{R}^k \) for some \( d \leq k < m \) that preserves as much of the properties of \( g \) as possible. Assuming our fitting \( g \) is good, a dimension reduction of \( \mathcal{S} \) into \( \mathbb{R}^k \) follows since each \( \hat{y} = y \in \mathcal{S} \) and \( \hat{y} \) is in \( g(\hat{K}) \).

Ignoring \( g \) for the time-being, there is a history of theoretical work dealing with the question whether any embedding into \( \mathbb{R}^k \) of a \( d \)-dimensional simplicial complex \( \hat{K} \) can exist in the first place. One of the most well-known results is Van Kampen’s generalization of Kuratowski’s graph planarity condition [10, 16, 13, 17], which gives an obstruction to embedding \( \hat{K} \) into \( \mathbb{R}^d \) lying in the degree 2\( d \) cohomology of a certain deleted product subcomplex of \( \hat{K} \times \hat{K} \). In fact, this is the largest dimension of interest since any simplicial complex of dimension \( d \) can be embedded linearly into \( \mathbb{R}^{2d+1} \) (for instance, by mapping each vertex to a unique point on the parametric curve \((t, t^2, \ldots, t^{2d+1}) \)). Some
more recent work has focused on PL-embedding into dimension \( \mathbb{R}^{d+1} \) \([2]\), or the tractability of embedding as a decision problem \([1]\).

A direct computational approach is possible when \( g(\hat{K}) \) is a triangulation of an embedded manifold; for example, by applying Isomap \([15]\) to its vertices \( g(v_j) \) and some choice of samples in \( g(\hat{K}) \).

Since the entire manifold \( g(\hat{K}) \) is already given, the geodesic distances between the samples can be given exactly, and what remains is the MDS stage of Isomap. In any case, \( g(\hat{K}) \) is not a sampled space, and moreover, it is often a much simpler object than \( S \) consisting of a much smaller collection of vertices and facets. Any dimension reduction of \( S \) by first reducing the dimension of the embedding of \( \hat{K} \) should be easier compared to a more head-on approach.

2.6. **Complexity.** The computationally intensive step that dominates Algorithm \([1]\) involves finding the points \( y' \in K \) for which \( f^\ell(y') \) is nearest to \( y \in S \). These points can be found as follows.

**Algorithm 3** Find nearest point on a simplex along with its barycentric coordinates

1: **input:** \( S \subseteq \mathbb{R}^m \), a simplex \( \sigma \subseteq K \), and linear map \( f^\ell : K \rightarrow \mathbb{R}^m \);
2: if \( \sigma \) is a vertex \( v_j \) then
3: set \( \lambda^\sigma_{iy} \leftarrow 1 \) and \( y^\sigma \leftarrow f^\ell(v_j) \) for each \( y \in S \);
4: else
5: let \( r := |S| \) and pick some ordering of \( S = \{y_1, \ldots, y_r\} \);
6: Let \( M \) be the \( m \times d \) matrix whose \( i^{th} \) column is the vector \( w_{i+1} - w_1 \), where \( \{w_1, \ldots, w_{d+1}\} \subseteq \{f^\ell(v_1), \ldots, f^\ell(v_n)\} \) are the vertices of \( f^\ell(\sigma) \);
7: Let \( S \) be the \( m \times r \) matrix whose \( j^{th} \) column is the vector \( y_j - w_1 \);
8: compute the Moore-Penrose pseudoinverse \( \bar{M} \) of \( M \), the \( 1 \times r \) matrix \( B' \) whose \( j^{th} \) entry is \( 1 - \sum_i x_{ij} \), and take the \( (d+1) \times r \) matrix \( B := \begin{bmatrix} B' \\ B \end{bmatrix} \);
9: partition \( S \) into disjoint subsets \( \bar{S} \) and \( S_1, \ldots, S_{d+1} \) where
   \[ \bar{S} := S - \bigcup_i S_i \]
   and
\[ S_k := \{ y_j \in S \mid (B)_{kj} < 0 \} - \bigcup_{i < k} S_i \]
10: for each \( y_j \in \bar{S} \), set \( \lambda^\sigma_{ijy} \leftarrow (B)_{ij} \) for \( 1 \leq i \leq d + 1 \), and \( y^\sigma \leftarrow \sum_i \lambda^\sigma_{ijy} \omega_i \);
11: for each \( S_k \) that is non-empty do
12: let \( \sigma_k \) be the boundary simplex of \( \sigma \) containing every vertex \( w_i \) except \( w_k \), and set \( \lambda^\sigma_{wy} \leftarrow 0 \);
13: repeat this algorithm for \( \sigma_k \) in place of \( \sigma \) and \( S_k \) in place of \( S \), letting \( y^\sigma := y'^\sigma_k \) and \( \lambda^\sigma_{iy} := \lambda^{\sigma_k}_{iy} \) (for \( i \neq k \)) denote its output for each \( y \in S_k \);
14: end for
15: output: \( y^\sigma \) and \( \lambda^\sigma_{iy} \) for each \( y \in S \).

The output \( y^\sigma \) is the nearest point to \( y \) lying on \( f^\ell(\sigma) \), and the \( \lambda^\sigma_{iy} \)'s are its barycentric coordinates in \( f^\ell(\sigma) \). The smallest dimensional simplex \( \sigma_y \subseteq \sigma \) for which \( y^\sigma \) is in \( f^\ell(\sigma_y) \) can quickly be found by looking at which of the \( \lambda^\sigma_{iy} \)'s are zero.

The product \( \bar{M}S \) and pseudoinverse \( \bar{M} \) can be determined using singular value decomposition of \( M \) in \( \mathcal{O}(dmr) \) time and \( \mathcal{O}(\min\{m^2d, md^2\}) \) time respectively, so all steps before the final for-loop can be executed in \( \mathcal{O}(dmr + \min\{m^2d, md^2\}) \). Typically \( r \geq \max\{m, d\} \), in which case this simplifies to \( \mathcal{O}(dmr) \). Since the recursive call in each iteration of the for-loop is executed only for those dimension \( d - 1 \) boundary simplices \( \sigma_i \subseteq \sigma \) for which \( S_i \) is non-empty, and since the \( S_i \)'s are all disjoint subsets of the current input \( S \), then at most \( r_d = \min\{r, (d + 1)!\} \) recursive calls are made. So the total execution time of the above algorithm is \( \mathcal{O}(dmrr_d) \).
The nearest point $f^\ell(y')$ to $y$ is given as a choice of $y^\sigma$ over all facets (not simplices) of $\sigma \subseteq K$ that minimizes $\|y^\sigma - y\|$ ($y'$ itself is determined by the $\lambda^\sigma_i$'s). As a result, the time complexity of each iteration of Algorithm 1 is at worst $O(Ndmrr_d)$ when $r \geq \max\{m, d\}$, where $N$ is the number of facets of $K$ and $d$ is the dimension of the highest dimensional facet in $K$. This is largely due to the cost of finding each $y'$ and $y^\sigma$ precisely. Significant improvements can be made if an approximation scheme is used. For example, since $f^\ell$ does not change much in consecutive iterations in Algorithm 1 for large enough $\ell$, $y'$ can be approximated in the $(\ell + 1)^{th}$ iteration by searching only those facets adjacent to the simplex that contains $y'$ in the previous iteration. In effect, this means sending an appropriate subset $S_\sigma \subseteq S$ in place of $S$ as input for Algorithm 3. At the same time, Algorithm 1 is easy to parallelize, as Algorithm 3 can be applied simultaneously to each facet $\sigma$ of $K$.

3. SOME TESTS ON SYNTHETIC DATA

Algorithm 1 was tested without the pruning stage (Algorithm 2). This was done on 2 and 3 dimensional data $S$ using grids and triangulated meshes $K$ embedded into $\mathbb{R}^2$ and $\mathbb{R}^3$. Precise values for nearest points were found using Algorithm 3 without any form of approximation, while the learning rate $s$ was set at 0.1. Implementation is in MATLAB R2015b. Source code can be found at https://github.com/pbebenSoton/smeans.

![Figure 1. 200 points sampled from a swiss roll, $K$ a 1-dimensional 5 x 5 grid. The edges containing few points near their interior end up being deleted during the pruning stage.](image1)

![Figure 2. 200 points sampled from a swiss roll, $K$ a line subdivided into 60 segments. The performance of this fitting is similar to classical SOM. Taking a grid of ambient dimension as in Figure 1 gives better results, using fewer vertices.](image2)
FITTING A SIMPLICIAL COMPLEX USING A VARIATION OF $k$-MEANS

Figure 3. 300 points sampled noisily from a circle and a line, $K$ a triangulated $6 \times 6$ mesh.

Figure 4. A noisy sampling of 350 points containing a circle, and $K$ a disjoint union of two $3 \times 3$ mesh clusters. Such clusters yield better fittings for highly disjoint data (as opposed to a single large mesh).
Algorithm 1 quickly converges to a stable solution for non-pathological complexes $K$ such as above. In most cases it also preserves the embedding of $K$, and where it does not, then it is usually very close to an embedding, with only a few simplices intersecting over a relatively small area (intersections are more common for less noisy data). All of these properties are less likely hold for irregularly triangulated $K$; for example, when the degree of some vertices is much higher than others.

Supposing there is convergence, what is the objective function that is being minimized? Since Algorithm 1 generalizes $k$-means clustering, an obvious first choice is the sum of squared distances $d_k = \sum_{y \in S} ||y - f^k(y')||^2$. Indeed, there are cases where this function happens to be minimized between every successive iteration; for instance, in the example in Figure 5. On the other hand,
if we start off as in Figure 8 with ℓ = 0 and f^0(K) containing every point in S, then f^ℓ(K) is unlikely to contain S for large enough ℓ, so d_ℓ will not always minimized. In this situation it is more natural to consider something along the lines of the Hausdorff distance d_H(S, f^ℓ(K)) = inf{ε ≥ 0 | f^ℓ(K) ∈ (S)_ε and S ⊆ (f^ℓ(K))_ε} where A_ε is the ε-neighbourhood of a subset A ⊆ R^m. In any case, it is not immediately clear what the objective function might be, or if there even is a single one. Another question is: under what modifications to Algorithm [1] is preservation of embedding guaranteed for some given (or all) K? One possibility is to assume all facets of K have the same ambient dimension m as R^m, then to update the vertices f^ℓ(v_j) in each iteration one at a time, at each update step considering only those y ∈ N^f_j that are inside a convex neighbourhood B of f^ℓ(v_j), where B is contained in the union of the (currently updated) facets which have f^ℓ(v_j) as a vertex. This can be shown to preserve embedding, keeps vertices on the boundary fixed, and therefore points y inside f^ℓ(K) remain as such in subsequent iterations. In order to get a good fitting, one selects a large enough triangulation of (say) an m-ball K together with a sufficiently expansive initial embedding f such that: there are enough interior facets in K to approximate S. The main downside is that K may have to be infeasibly large in order to achieve this, especially when our dimension m is large.

References

1. James C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Kluwer Academic Publishers, Norwell, MA, USA, 1981.
2. A. Björner and A. Goodarzi, On Codimension one Embedding of Simplicial Complexes, ArXiv e-prints (2016).
3. Gunnar Carlsson, Topology and data, Bull. Amer. Math. Soc. (N.S.) 46 (2009), no. 2, 255–308. MR 2476414
4. L. Cayton, Algorithms for manifold learning, Univ. of California at San Diego Tech. Rep (2005).
5. Herbert Edelsbrunner and John Harer, Persistent homology—a survey, Surveys on discrete and computational geometry, Contemp. Math., vol. 453, Amer. Math. Soc., Providence, RI, 2008, pp. 257–282. MR 2405684
6. K. Fukunaga and D. R. Olsen, An algorithm for finding intrinsic dimensionality of data, IEEE Trans. Comput. 20 (1971), no. 2, 176–183.
7. Hughes Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, Mesh optimization, Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), SIGGRAPH ’93, ACM, 1993, pp. 19–26.
8. T. Kohonen, M. R. Schroeder, and T. S. Huang (eds.), Self-organizing maps, 3rd ed., Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.
9. Teuvo Kohonen, Self-organized formation of topologically correct feature maps, Biological Cybernetics 43 (1982), no. 1, 59–69.
10. Casimir Kuratowski, Sur le problème des courbes gauches en topologie, Fundamenta Mathematicae 15 (1930), no. 1, 271–283 (fre).
11. Jiří Matoušek, Martin Tancer, and Uli Wagner, Hardness of embedding simplicial complexes in R^d, CoRR abs/0807.0336 (2008).
12. Lawrence K. Saul and Sam T. Roweis, Think globally, fit locally: Unsupervised learning of low dimensional manifolds, J. Mach. Learn. Res. 4 (2003), 119–155.
13. Arnold Shapiro, Obstructions to the imbedding of a complex in a euclidean space. I. The first obstruction, Ann. of Math. (2) 66 (1957), 256–269. MR 0089410
14. C. O. S. Sorzano, J. Vargas, and A. P. Montano, A survey of dimensionality reduction techniques, ArXiv e-prints (2014).
15. J. B. Tenenbaum, V. de Silva, and J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000), 2319–2323.
16. E. R. van Kampen, Komplexe in euklidischen Räumen, Abh. Math. Sem. Univ. Hamburg 9 (1933), no. 1, 72–78. MR 3069580
17. Wu Wen-tsün, A theory of imbedding, immersion, and isotopy of polytopes in a euclidean space, Science Press, Peking, 1965. MR 0215305
18. M.-S. Yang, A survey of fuzzy clustering, Mathematical and Computer Modelling 18 (1993), no. 11, 1 – 16.

School of Mathematics, University of Southampton, Southampton SO17 1BJ, United Kingdom
E-mail address: P.D.Beben@soton.ac.uk; pdbcas@gmail.com