Sparsity-Inducing Fuzzy Subspace Clustering

Arthur Guillon, Marie-Jeanne Lesot and Christophe Marsala

Abstract This paper considers a fuzzy subspace clustering problem and proposes to introduce an original sparsity-inducing regularization term. The minimization of this term, which involves a ℓ_0 penalty, is considered from a geometric point of view and a novel proximal operator is derived. A subspace clustering algorithm, *Prosecco*, is proposed to optimize the cost function using both proximal and alternate gradient descent. Experiments comparing this algorithm to the state of the art in sparse fuzzy subspace clustering show the relevance of the proposed approach.

Arthur Guillon · Marie-Jeanne Lesot · Christophe Marsala
Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
✉ arthur.guillon@lip6.fr
✉ marie-jeanne.lesot@lip6.fr
✉ christophe.marsala@lip6.fr

Archives of Data Science, Series A
(Online First)
KIT Scientific Publishing
Vol. 5, No. 1, 2018

DOI: 10.5445/KSP/1000087327/08
ISSN 2363-9881

1 Introduction

Clustering analysis is a data mining technique which reveals the tendencies of a dataset by forming groups of data points that are similar to each other and dissimilar to points from other clusters. Subspace clustering (Agrawal et al., 1998; Parsons et al., 2004) generalizes clustering by allowing clusters to reside in different subspaces of the original space. These subspaces are not known beforehand: The *relevant* dimensions of the clusters are to be learned by the algorithm. As for standard clustering, the resulting clusters have to exhibit a strong internal similarity, however the latter is expressed in terms of dimensions local to the cluster.

Points may belong to one or more clusters. For instance, text documents may address one or more specific topics and belong to one or several corpora. This paper adopts a soft, or fuzzy, subspace clustering framework (FSC for short; see e.g. Deng et al., 2016), where both the memberships of points to clusters and the weights of the dimensions are degrees in $[0, 1]$. Indeed, first, as different clusters may reside in different subspaces, they may intersect and some data points belong to several clusters, with different degrees. Second, the features describing a subspace can be more or less relevant to a cluster, depending on how the data points are spread in the subspace.

It is customary that many original features are fully irrelevant to describe the clusters. Traditional FSC algorithm may identify them as inappropriate and assign them low, but non-zero, weights, polluting the description of the subspaces. A way to address this problem is to introduce sparsity in the solutions: By setting their weights to 0, a sparsity-inducing FSC algorithm effectively removes irrelevant features, identifying the data intrinsic dimensionality.

This paper proposes such a sparse weighting of the features. A new sparse FSC algorithm named *Prosecco* is introduced, which produces sparse description of the subspaces. To do so, it proposes an optimization problem involving an ℓ_0 penalty regularization term (Hastie et al., 2015). Although the optimization of such problems is computationally hard in the general case and dictates the use of surrogates such as ℓ_1 norm (Natarajan, 1995), the settings of this paper, which combine alternate optimization and proximal splitting (Guillon et al., 2016), allow the use of a polynomial algorithm which explores the space of potential solutions using elementary geometric tools.

The paper is structured as follows: Related works in subspace clustering with sparsity constraints are presented in Section 2. The new cost function is presented in Section 3, together with the original ℓ_0 problem studied in this paper. It is reformulated in a geometric framework, which leads to the proposition of an algorithmic expression of the proximal operator solving this problem, along with a theoretical proof of its correctness. The *Prosecco* algorithm is presented in Section 4. Two experimental studies are then considered in Section 5 to evaluate *Prosecco* and other FSC algorithms on artificial and real data.

2 Related Works

In this section, the general problem of sparse fuzzy subspace clustering is presented. Related works are then recalled. The following notations are used: Denoting the data matrix by $X = (x_i)_{i=1\dots n} \in \mathbb{R}^{n\times d}$, the subspace clustering task consists in identifying c clusters represented by their centres $C = (c_r)_{r=1\dots c} \in \mathbb{R}^{c\times d}$. Through the computation of membership degrees $U = (u_{ri}) \in [0,1]^{c\times n}$, $r = 1\dots c$ and $i = 1\dots n$, the data are fuzzily assigned to clusters. Specific to the subspace clustering task, the weights of dimensions $W = (w_{rp}) \in [0,1]^{c\times d}$, $r = 1\dots c$ and $p = 1\dots d$ are learned independently for each cluster, and assign the relevance of a given dimension p for a cluster C_r: The higher the weight, the closer the points assigned to C_r along this dimension.

2.1 Problem Statement

Subspace clustering (Agrawal et al., 1998) is a generalization of clustering that forms clusters of high internal similarity, but also identifies the representation most adapted to each of these clusters. As points assigned to different clusters may exhibit different common features, this representation is discovered simultaneously with the clusters and is necessary to form them. In practice, most subspace clustering algorithms identify subspaces by selecting the features most relevant to describe each cluster.

These subspaces can be of various sorts. Most often, (soft) projections on axes-parallel subspaces or linear combinations (including rotations) of the original axes are considered (Vidal, 2011). This point of view allows to understand

subspace clustering as a generalization of PCA which identifies several principal components in different subgroups of the original data (Vidal, 2011).

A Fuzzy Subspace Clustering model

The fuzzy variant of Subspace Clustering has been introduced by Keller and Klawonn (2000) (K&K for short) who refine the Fuzzy c-Means cost function by introducing a weighted Euclidean distance. Using hyperparameters $m, v > 1$ which allow to introduce and control fuzziness into the solutions, they propose the following cost function

$$J_{\text{K\&K}}(C, U, W) = \sum_{r=1}^{c} \sum_{i=1}^{n} u_{ri}^{m} \sum_{p=1}^{d} w_{rp}^{v} (x_{ip} - c_{rp})^2 \,. \tag{1}$$

This cost function is an extension of the classic Fuzzy c-Means one, the latter corresponding to the case where all weights w_{rp} are equal to each other. It is minimized under the three constraint sets by the standard method of Lagrangian multipliers

$$\text{(C1) } \forall i \in \{1 \dots n\}: \quad \sum_{r=1}^{c} u_{ri} = 1 \,, \tag{2}$$

$$\text{(C2) } \forall r \in \{1 \dots c\}: \quad \sum_{i=1}^{n} u_{ri} > 0 \text{ and} \tag{3}$$

$$\text{(C3) } \forall r \in \{1 \dots c\}: \quad \sum_{p=1}^{d} w_{rp} = 1 \,. \tag{4}$$

This leads to an algorithm which alternates between the optimization of the three parameters and learns cluster centres C, membership degrees U and feature weights W.

Guillon et al. (2016) modify this cost function in order to express the third constraint as a penalty term and change the optimization scheme accordingly. An alternate optimization algorithm combining gradient and proximal descent is introduced. In the present paper, a similar optimization scheme is adapted to a new cost function, with a different penalty term.

2.2 Sparsity in Subspace Clustering

While standard FSC algorithms identify the features that are the most relevant to describe a cluster, they produce dense weight vectors $W_r \in \mathbb{R}^d$ such that $\forall p, w_{rp} > 0$ and thus do not necessarily find the dimensionality of each subspace: All dimensions receive non-zero weights, some of them being arbitrarily small. Although one may, of course, apply a threshold to select significant dimensions, this paper focuses on approaches resulting in sparsity by-design.

To address this problem, several authors have considered sparsity-inducing algorithms, which produce sparse description of the subspaces. Witten and Tibshirani (2010) integrate global feature selection to the standard k-means algorithm and add a ℓ_1 constraint to the optimization problem to produce sparse descriptions. This approach belongs to the family of LASSO-based feature selection algorithms (Tibshirani, 1996). Although these approaches work well in practice, they use the ℓ_1-norm as a surrogate to select features. Guillon et al. (2019) observe that the summation constraint on the vectors W_r forbids the use of the ℓ_1-norm; in the present paper, the proposed framework is simple enough so that no surrogate is necessary. Instead, a polynomial time stepwise feature selection is proposed, along with a proof of correctness.

Jing et al. (2007) study subspace clustering of sparse data such as text vectors. They introduce a subspace version of the k-means cost function with weights $w_{rp} \in [0,1]$. An entropic regularization term is added to the cost function, which is responsible for introducing both sparsity and fuzziness in the weight vectors

$$J_{\mathrm{EWKM}}(C,U,W) = \sum_{r=1}^{c}\sum_{i=1}^{n} u_{ri} \sum_{p=1}^{d} w_{rp}(x_{ip} - c_{rp})^2 + \gamma \cdot \sum_{p=1}^{d} w_{rp}\log(w_{rp})\,. \tag{5}$$

Their intent is to control the sparsity of the data (texts and occurrence of specific, domain-related terms) through the use of entropy. The use of the entropic regularization term allows them to derive a simple optimization algorithm named Entropy Weighting k-Means (*EWKM* for short), but complicates the modulation of the sparsity effect in the solutions, even by tuning the value of

regularization parameter: The computed solutions are often very sparse and *EWKM* fails to identify the right dimensionality of the clusters. Moreover, the update equations given by Jing et al. (2007) do not theoretically produce sparse vectors. In practice, however, the computed values are too small to be represented by floating point numbers, thus producing sparse vectors. Finally, the hyperparameter $\gamma > 0$ allows the user to balance the effects of the two terms. In the following, a fuzzy version of *EWKM* is considered, where $u_{ri} \in [0, 1]$, in order to allow the comparison with other FSC algorithms. This amounts to minimizing the following cost function

$$J'_{\text{EWKM}}(C, U, W) = \sum_{r=1}^{c} \sum_{i=1}^{n} u_{ri}^2 \sum_{p=1}^{d} w_{rp}(x_{ip} - c_{rp})^2 + \gamma \cdot \sum_{p=1}^{d} w_{rp} \log(w_{rp}). \tag{6}$$

Borgelt (2008) extends the K&K cost function by modifying the fuzzifier operating on the weights, so that sparse weight vectors are produced: Transposing the generalized fuzzifier proposed by Klawonn and Höppner (2003) for the membership degree, he proposes to replace the term w_{rp}^{ν} in Equation (1) by

$$\frac{1-\beta}{1+\beta} w_{rp}^2 + \frac{2\beta}{1+\beta} w_{rp}$$

where $\beta \in [0, 1)$ is a hyperparameter that allows to modulate the sparsity effect. This parameter is interpreted as a ratio between intra-cluster variances whose value setting appears critical yet hard. The *Prosecco* solution presented in this paper separates the optimization and sparsification of the weights and presents an algorithm based on the geometry of the unit simplex of $\mathbb{R}^d$.

Finally, in a different setting, Elhamifar and Vidal (2009) introduce Sparse Subspace Clustering (SSC), a family of algorithms which has become standard in the computer vision community. In the simplest setting, SSC recovers subspaces intersecting only at the origin by expressing each point as combination of its neighbours. As this self-expressive model includes a ℓ_1 sparsity-inducing constraint, only close neighbours of each point are selected. Subspaces can then be recovered and a standard spectral clustering algorithm is run to produce the clusters. Several improvements over this general idea have been made, including

the use of sparsity-inducing norms other than simple ℓ_1. However, to the best of our knowledge, this approach does not work well when clusters and subspaces lie in the same vector subspaces or when they intersect on more than one point, which often happens.

3 Proposed Cost Function and Optimization

This section presents the cost function we propose, from which a FSC algorithm is derived. It features a penalty term which constrains the solution in two ways: First by ensuring that it is valid with respect to constraint (C3), secondly by introducing sparsity in the weighting of the dimensions.

3.1 Proposed Cost Function

The proposed cost function J is the sum of a subspace clustering model F and a penalty term G that induces sparsity:

$$J(C, U, W) = F(C, U, W) + \gamma \cdot G(W), \tag{7}$$

with

$$F(C, U, W) = \sum_{r=1}^{c} \sum_{i=1}^{n} u_{ri}^{m} \sum_{p=1}^{d} w_{rp}^{2} (x_{ip} - c_{rp})^2, \tag{8}$$

and

$$G(W) = \gamma \cdot \sum_{r=1}^{c} G_{\ell_0}(W_r), \tag{9}$$

where G_{ℓ_0} is defined by

$$G_{\ell_0}(W_r) = \begin{cases} \|W_r\|_0 & \text{if } \sum_{p=1}^{d} w_{rp} = 1 \\ +\infty & \text{otherwise} \end{cases}. \tag{10}$$

The F function is the same as the cost function from Equation (1), setting v to 2 in order to ease the mathematical analysis, and the cost function is optimized under the same constraint set (C1) and (C2). The second term, G, is the proposed penalty term: by giving an infinite weight to vectors W_r which do not sum up to 1, it prevents the trivial minimizer $W = 0$ and forbids invalid solutions by enforcing the summation constraint set (C3). Additionally, the proposed penalty makes a difference between the valid weight vectors, expressing the desired sparsity preference: The sparser the considered W_r, the smaller the penalty value.

While the differentiable function F can be minimized through gradient descent, the penalty term G, that can take infinite values and is non-differentiable, cannot. However, as G only depends on the weights of the subspaces, the optimization can be split in two parts: The optimization regarding C and U is standard and is briefly recalled in Section 4. Regarding W, advanced optimization techniques such as proximal gradient descent, presented in Section 3.2, allow us to derive an algorithm finding solutions to the proposed subspace clustering problem: By reformulating the minimization problem into a geometric one in Section 3.3, we propose in Section 3.4 a proximal operator for G, i.e., an algorithm which finds sparse weight matrices W, which is then proven correct in Section 3.5.

In the following, the parameter of interest is W, and the minimization of J is considered from the point of view of proximal operators, which allow to split the minimization of the F and G (Guillon et al., 2016).

3.2 Chosen Optimization Framework: Proximal Splitting

Proximal gradient descent is an iterative optimization scheme enriching standard gradient descent, which is usually employed to minimize a sum of convex functions, some of which are non-differentiable. All definitions and results in this section can be found e.g. in Parikh and Boyd (2013). Given $J(W) = F(W) + \gamma \cdot G(W)$, where F is differentiable, proximal gradient descent splits the descent as such

$$W^{t+1} = \mathrm{prox}_{\gamma G}\left(W^t - \eta \nabla F(W)\right) , \tag{11}$$

where $\nabla F(W)$ is the gradient of F seen as a function of W and $\mathrm{prox}_{\gamma G}$ is the proximal operator of G, defined by

$$\operatorname{prox}_{\gamma G}(W^0) = \underset{W \in \mathbb{R}^{c \times d}}{\operatorname{argmin}} \frac{1}{2}\|W - W^0\| + \gamma \cdot G(W)\,. \tag{12}$$

The proximal operator of G maps a given point W^0 to a point W which is the minimum of G in a certain neighbourhood of W^0 (because of the first term), this effect being implicitly modulated by the constant γ. Based on this operator, the proximal gradient scheme in Equation (11) splits the descent in two parts, first looking for a minimizer of F using the gradient, then projecting the current approximation back using the proximal operator of G.

The penalty function from Equation (10) is not convex and therefore its proximal operator is ill-defined. However, by looking at its definition, it can be seen that G only discriminates between valid solutions according to their sparsity, i.e., several solutions equally sparse receive the same cost. Therefore, G not being convex and having several minimizers for G_{ℓ_0} and J is not a problem. As G is separable, its proximal operator is expressed as the Cartesian product of the ones of G_{ℓ_0}:

$$\operatorname{prox}_{\gamma G}(W^0) = (\operatorname{prox}_{\gamma G}(W_1^0), \cdots, \operatorname{prox}_{\gamma G}(W_c^0))\,. \tag{13}$$

The penalty function G_{ℓ_0} can be understood geometrically as projections onto particular sets, and the corresponding proximal operator is derived as such: The minimization problem is interpreted as a projection one. First, recall that the characteristic function of a convex closed set C is given, as well as its proximal operator, by the following equations

$$\iota(V^0) = \begin{cases} 0 & \text{if} V^0 \in C \\ +\infty & \text{otherwise} \end{cases}, \tag{14}$$

$$\operatorname{prox}_{\gamma \iota}(V^0) = \underset{V \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \frac{1}{2}\|V^0 - V\|^2 + \gamma \cdot \iota(V) \right\} \tag{15}$$

$$= \underset{V \in C}{\operatorname{argmin}} \left\{ \frac{1}{2}\|V^0 - V\|^2 \right\} \tag{16}$$

$$= \pi_C(V^0)\,, \tag{17}$$

that is, the Euclidean projection $\pi_C(V^0)$ of V^0 on C. The same approach can be applied to the minimization of G_{ℓ_0}, which is seen as a sequence of projections on a family of sets, presented in the next section.

3.3 Projections on Simplices

The projection principle of proximal optimization can be exploited for the proposed sparse fuzzy subspace clustering problem: The summation constraint of the W_r components is equivalent to requiring that $W_r \in \Delta$, where Δ is the unit simplex defined as $\Delta = \left\{ V \in \mathbb{R}^{+d} \;\middle|\; \sum_{p=1}^{d} V_p = 1 \right\}$. The faces of Δ are also of interest: if $s \subset \{1 \dots d\}$, let Δ_s denote the face of Δ with coordinates indexed by s equal to 0, i.e., $\Delta_s = \left\{ V \in \Delta \;\middle|\; \forall p \in s, v_p = 0 \right\}$. The Euclidean projection of $V \in \mathbb{R}^d$ on Δ (resp. Δ_s) is written $\pi(V)$ (resp. $\pi_s(V)$).

In order to obtain a proximal operator for G_{ℓ_0}, we replace ι by G_{ℓ_0} in Equation (15), which gives a new minimization problem involving the ℓ_0 norm

$$\text{prox}_{\gamma \cdot G_{\ell_0}}(V^0) = \text{argmin}_{V \in \Delta} \left\{ \tfrac{1}{2} \|V^0 - V\|^2 + \gamma \cdot \|V\|_0 \right\}, \tag{18}$$

that is, sparse solutions minimize the following cost function on Δ

$$cost_{\gamma, V^0}(V) = \tfrac{1}{2} \|V^0 - V\|^2 + \gamma \cdot \|V\|_0, \tag{19}$$

which expresses the cost of a potential solution on Δ. This proximal problem has two particularities: It involves the ℓ_0 norm and is constrained to the set Δ. In the following section, an algorithm is proposed to solve this problem, using successive projections onto the faces Δ_s.

Given an initial vector $V \in \mathbb{R}^d$ and γ, Eq. (18) amounts to finding the right Δ_s to project to. There exists no general formula for $\pi_s(V)$. However, under some hypothesis, the following formula can be given:

Proposition 1 Let $V \in \mathbb{R}^d$ and $s \subset \{1 \dots d\}$.
If $\forall p \in \{1 \dots d\}, V_p \in [0, 1]$ and $\sum_{p=1}^{d} V_p \leq 1$ then $\pi(V)$ and $\pi_s(V)$ are given by, $\forall p \in \{1 \dots d\}$

$$\pi(V)_p = v_p + \frac{1}{d}\left(1 - \sum_{q=1}^{d} v_q\right), \tag{20}$$

$$\pi_s(V)_p = \begin{cases} 0 & \text{if} p \in s \\ v_p + \frac{1}{d-\text{card}(s)}\left(1 - \sum\limits_{\substack{q=1 \\ q \notin s}}^{d} v_q\right) & \text{otherwise} \end{cases}. \tag{21}$$

The formulas are adapted from (Duchi et al., 2008), according to the hypothesis. They normalize a vector V by redistributing equally between each component, omitting the components p in s for the projection π_s.

Moreover, these projections have the following relationship, which makes them useful to the problem of interest in this paper:

Proposition 2 Let $s_1, s_2 \subset \{1 \dots d\}$ such that $s_1 \subset s_2$ and $V \in \mathbb{R}^d$, then

$$\pi_{s_2}(\pi_{s_1}(V)) = \pi_{s_2}(V)\,. \tag{22}$$

Moreover, for any $V \in \mathbb{R}^d$:

$$\|\pi_{s_2}(V) - V\|^2 = \|\pi_{s_2}(V) - \pi_{s_1}(V)\|^2 + \|\pi_{s_1}(V) - V\|^2\,. \tag{23}$$

Proof (sketch). Both equalities hold by orthogonality: If $s_1 \subset s_2$, then $\Delta_{s_2} \subset \Delta_{s_1}$ and $\pi_{s_1}(V)$ and $\pi_{s_2}(V)$ both belong to Δ_{s_1}. The second one is an application of the Pythagorean theorem. □

3.4 Proposed Algorithmic Proximal Operator

Equation (18) involves the minimization of an ℓ_0 norm on Δ. Although such minimization problems are NP-hard in general, the present settings allow to exactly solve this problem in polynomial time. After introducing a geometrically-inspired algorithm to do so, we present the *Prosecco* algorithm. All the fuzzy subspace clustering algorithms presented in this paper have been implemented in the Python programming language. The implementation can be found in the following GitHub repository: `https://github.com/aguillon/subspy`.

Algorithm 1: Proximal operator for G_{ℓ_0}.

procedure MinL0 (V^0, γ)
  $s \leftarrow \emptyset$
  $V^{\text{sol}} \leftarrow \pi(V^0)$ ▷ Best solution so far
  $V \leftarrow \pi(V^0)$
  while $s \neq \{1 \dots d\}$ **do**
    $p_0 \leftarrow \text{argmin}_{p \in \{1 \dots d\}} \{V_p^0 \mid p \notin s\}$
    $s \leftarrow s \bigcup \{p_0\}$
    $V \leftarrow \pi_s(V)$
    if $cost_{\gamma, V^0}(V) \leq cost_{\gamma, V^0}(V^{\text{sol}})$ **then**
      $V^{\text{sol}} \leftarrow V$
    end if
  end while
  return V^{sol}
end procedure

The algorithmic definition of the proximal operator given in Equation (18) is presented in Algorithm 1. It works as follows: Starting from $V^0 \in \mathbb{R}^d$ and a fixed γ, V^0 is projected to Δ. Then, by removing the smallest weights one after another and projecting back to the corresponding Δ_s, it decreases the cost of the potential solution and finds the best estimation of the subspace. The rest of this section provides justifications of this algorithm.

3.5 Correctness of the Proposed Algorithm

As mentioned in the previous section, for specific values for γ and V^0 there can be several distinct minimizers for $cost$. However they are all equally suited to the needs of the present paper and thus it is sufficient that Algorithm 1 returns one of those minimizers.

Proposition 3 guarantees the existence of a minimum of the considered $cost$ function, Proposition 4 proves such a minimum can be obtained by iteratively setting components to 0. They lead to Theorem 1 stating the correctness of the proposed algorithm.

Proposition 3 Let $S = \left\{\pi_s(V^0) \;\middle|\; s \subset \{1 \dots d\}\right\}$.
There exists s^0 such that $\pi_{s^0}(V^0) \in S$ is a minimum of $cost$, that is:

$$\forall V \in \Delta,\ \ cost(\pi_{s^0}(V^0)) \leq cost(V)$$

Proof. Let $V \in \Delta$ and $s \subset \{1 \dots d\}$ the largest set such that $V \in \Delta_s$. First notice that

$$\begin{aligned} cost(\pi_s(V^0)) &= \tfrac{1}{2}\,\|\pi_s(V^0) - V^0\|^2 + \gamma \cdot \mathrm{card}(s) \\ &= \tfrac{1}{2}\,\|\pi_s(V^0) - \pi(V^0)\|^2 + \tfrac{1}{2}\|\pi(V^0) - V^0\|^2 + \gamma \cdot \mathrm{card}(s)\,, \end{aligned}$$

from Proposition 2, with $s_2 = \emptyset$. As $\pi(V^0)$ is the Euclidean projection of V^0 onto Δ,

$$\begin{aligned} cost(V) &= \tfrac{1}{2}\,\|V - V^0\|^2 + \gamma\ \cdot\ \|V\|_0 \\ &= \tfrac{1}{2}\,\|V - \pi(V^0)\|^2 + \tfrac{1}{2}\|\pi(V^0) - V^0\|^2 + \gamma\ \cdot\ \|V\|_0\,, \end{aligned}$$

by orthogonality. Moreover:

$$\begin{aligned} cost(V) &= \tfrac{1}{2}\Big(\|V - \pi_s(\pi(V^0))\|^2 + \|\pi_s(\pi(V^0)) - \pi(V^0)\| \\ &\qquad +\ \|\pi(V^0) - V^0\|\Big) + \gamma\ \cdot\ \|V\|_0\,, \end{aligned}$$

again by orthogonality. As $\|\pi_s(V^0)\|_0 \leq \|V\|_0$ and $\pi_s(\pi(V^0)) = \pi_s(V^0)$,

$$\begin{aligned} cost(V) &= \tfrac{1}{2}\Big(\|V - \pi_s(V^0)\|^2 + \|\pi_s(V^0) - \pi(V^0)\| \\ &\qquad +\ \|\pi(V^0) - V^0\|\Big) + \gamma\ \cdot\ \|V\|_0 \\ &= \tfrac{1}{2}\,\|V - \pi_s(V^0)\|^2 + cost(\pi_s(V^0))\,, \end{aligned} \tag{24}$$

therefore:

$$cost(V) \geq cost(\pi_s(V^0))\,. \tag{25}$$

Finally, for all $V \in \Delta$, there exists s such that $cost(\pi_s(V^0)) \leq cost(V)$ and $\pi_s(V^0) \in S$: As S is finite, there exists a global minimizer $\pi_{s^0}(V^0)$ (non-necessarily unique). □

As the space of solutions S is of size 2^d, this proposition implies that there exists an algorithm to minimise the $cost$ function with computational complexity $O(2^d)$. Algorithm 1 sorts the values of V_p for $p \in \{1 \dots d\}$ and progressively removes the smallest ones to decrease the cost, giving a polynomial algorithm. The following proposition justifies this principle:

Proposition 4 Let $\{V^0_{p_1}, \dots, V^0_{p_k}\}$ the k smallest values of V^0, $s = \{p_1, \dots, p_k\}$, and $V \in \Delta_s$ such that V contains exactly $k = \mathrm{card}(s)$ zeros. If V' is such that $V'_p = V_p$ for all p except for a couple (p_i, p_j) with $p_i \in s, p_j \notin s$, then $cost(V) \leq cost(V')$.

Proof. Given such V, V', first notice that, as $V \in \Delta$ and $V' \in \Delta$, $V'_{p_i} = V_{p_j}$ due to the summation constraint, $V_{p_i} = 0 = V'_{p_j}$ by hypothesis and $V^0_{p_i} < V^0_{p_j}$, also by hypothesis. Then:

$$\begin{aligned} cost(V) - cost(V') &= \textstyle\sum_{p=1}^{d}(V_p - V^0_p)^2 - \sum_{p=1}^{d}(V'_p - V^0_p)^2 \\ &= (V_{p_j} - V^0_{p_j})^2 + (V^0_{p_i})^2 \\ &\quad - (V'_{p_i} - V^0_{p_i})^2 - (V^0_{p_j})^2 \\ &= 2\,V_{p_i}(V^0_{p_i} - V^0_{p_j})\,, \end{aligned} \tag{26}$$

therefore:

$$cost(V) - cost(V') < 0\,. \tag{27}$$

□

This proposition proves that the minimizer can be chosen step by step, at each iteration. Indeed, at a given step of the algorithm, given two possible s, s' which differ only for two p, p', the one leading to a potential minimizer is the one corresponding to V^0 smallest entry (not yet in s). As Algorithm 1 keeps the minimum (see line 9), it is guaranteed to find the minimum:

Theorem 1 Given V^0 and γ, Algorithm 1 computes the solution to the minimalization problem defined in Equation (18).

Proof. Algorithm 1 keeps the best minimum of $cost$ at each iteration. Although it explores the space of solutions one local choice after the other, according to Proposition 4 these local choices are optimal, and are given by repeated projections on faces of Δ according to Proposition 3. □

4 Proposed FSC Algorithm: *Prosecco*

The previous section gives an algorithm to produce sparse approximations of the weights vector $W_r \in \mathbb{R}^d$ of a given cluster C_r. This procedure can be used in an optimization scheme in order to minimize the function J. *Prosecco* relies on an alternate optimization algorithm which separates the update of C and U from the update of W, the former being derived with the method of Lagrange multipliers (as for the standard FCM algorithm), whereas the latter is approximated through proximal gradient descent, presented in Section 3.2 and based on the proximal operator given by Algorithm 2 (see page 17).

From Equation (7) and constraint sets (C1) and (C2), the classic following terms are derived for U and C

$$u_{ri} = \frac{d_{ri}^2 \cdot (1-m)^{-1}}{\sum_{s=1}^{c} d_{si}^2 \cdot (1-m)^{-1}}, \tag{28}$$

where

$$d_{ri}^2 = \sum_{p=1}^{d} w_{rp}^2 (x_{ip} - c_{rp})^2, \tag{29}$$

$$c_{rp} = \frac{\sum_{i=1}^{n} u_{ri}^m \cdot x_{ip}}{\sum_{i=1}^{n} u_{ri}^m}. \tag{30}$$

In an alternate optimization scheme, F is seen as a sole function of W. The following proposition states that F can be optimized by gradient descent, and that its gradient descent, given the right η, stays inside the bounds required by the previous algorithm in Proposition 1.

Proposition 5 Given U, C, let $W \in \Delta$. There exists $\eta > 0$ such that $\forall r, W_r - \eta \nabla F(W_r)$ satisfies the condition from Proposition 1.

Proof. Given U, C, let $W \in \Delta$, for any r, $W_r - \eta \nabla F(W_r)$ must be such that

$$W_r - \eta \nabla F(W_r) \in [0,1] \quad \wedge \quad \sum_{p=1}^{d} w_{rp} \leq 1\,. \tag{31}$$

The summation constraint is satisfied as $W \in \Delta$. As the gradient of F is $[\nabla F(W)]_{rp} = 2\sum_{i=1}^{n} u_{ri}^m w_{rp}(x_{ip} - c_{rp})^2$, the condition thus reduces to the constraint that, for all r and all p

$$w_{rp}\left(1 - 2\eta \sum_{i=1}^{n} u_{ri}^m (x_{ip} - c_{rp})^2\right) \in [0,1]\,. \tag{32}$$

Thus the following value for η is suitable:

$$\eta = \left(\max_{r,p}\left\{2\sum_{i=1}^{n} u_{ri}^m (x_{ip} - c_{rp})^2\right\}\right)^{-1}. \tag{33}$$

□

Algorithm 2: The *Prosecco* algorithm.

procedure PROSECCO (X, c, ϵ, γ)
  repeat
    repeat
      Update U according to Equation (28)
      Update C according to Equation (30)
    until $convergence(C, U, \epsilon)$
    repeat
      Compute η according to Equation (33)
      $W_{\text{temp}} \leftarrow W - \eta \nabla F(W)$
      $W \leftarrow$ MINL0 (W_{temp})
    until $convergence(W, \epsilon)$
  until $convergence(C, U, W, \epsilon)$
  Update U and C one last time.
end procedure

Finally, the *Prosecco* algorithm is given by Algorithm 2. It alternates between the optimization of U and C using their respective update equations, and the optimization of W using proximal gradient descent. In addition to the data

matrix X and the number of clusters c, *Prosecco* takes two parameters: A convergence parameter ϵ as threshold on matrix distances between updated and previous values and the hyperparameter γ which selects the level of sparsity of the solution. As this depends on the application as well as the data, it is recommended to run the algorithm several times using various values for γ. This is a general weakness of such algorithms (such as *EWKM* and *Borgelt* described in Section 2), and future research will focus on proposing an automated tuning of this parameter.

5 Experimental Study

In order to provide an experimental validation of *Prosecco*, two experiments are considered: The first one, consisting of artificial data, is described in Section 5.1. The goal of this experiment is to assess the ability of sparsity-inducing FSC algorithms to recover the dimensionality of hyperplanes. In Section 5.2, real data from the Newsgroups dataset is considered (Lichman, 2013).

5.1 Dimensionality Estimation

The first experiment assesses the ability of *Prosecco* and other FSC algorithms to estimate the dimensionality of simple subspaces: the considered artificial data sets are generated as axes-parallel hyperplanes of low dimensionality following a uniform distribution in a space of higher dimensionality $\mathbb{R}^d$. More precisely, d being fixed, k clusters of random dimensionality $d_r \in \{1 \dots d-4\}$ are generated (with $n_r = 600$ points each) with a low variance in d_r (randomly chosen) dimensions, and high variance in the other ones.

Prosecco is compared to the two other FSC algorithms presented in Section 2, *EWKM* and *Borgelt*. For each experiment, the algorithms are run with the correct number of clusters c and the convergence parameter $\epsilon = 10^{-4}$. Besides, *Prosecco* is run with $\gamma = 1$, *Borgelt* with $\beta = 0.1$ and $\beta = 0.3$, *EWKM* with $\gamma \in \{0.5, 10, 100\}$. For *Borgelt* and *EWKM*, only the best result for each experiment is kept. In theory, *EWKM* does not induce sparsity, for the purpose of this experiment, the obtained weights w_{rp} below 10^{-10} are set to 0.

The algorithms' ability to estimate the correct dimensionality of the planes is evaluated as follows: Given a cluster C_r with known dimensionality d_r, the algorithms have to recover 95% of the cluster after defuzzification of the memberships u_{ri}, then to produce W_r such that $\|W_r\|_0 = d_r$. The metric used for comparison is the ratio of correct estimations.

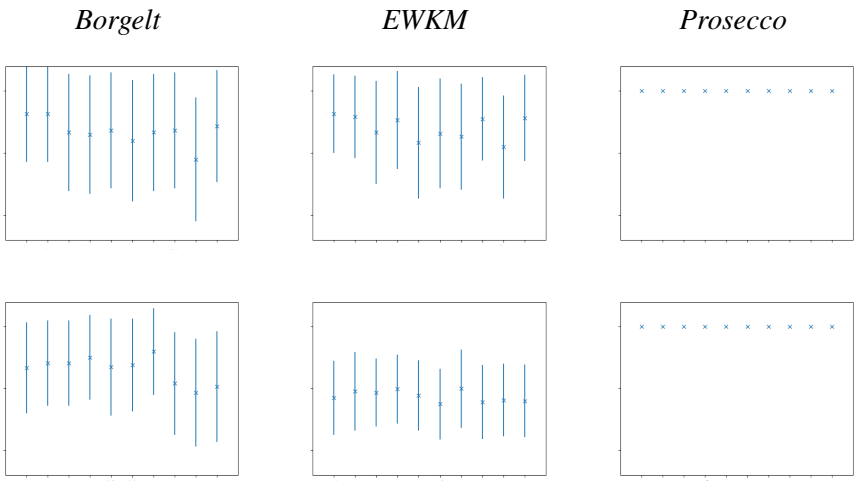

Figure 1: Results on artificial data, plotted with mean and standard deviation. From left to right: Algorithms *Borgelt*, *EWKM* and *Prosecco*. Top: 2 clusters. Bottom: 4 clusters. On the x-axis, number of dimensions (from 10 to 28); on the y-axis, ratio of correctly identified dimensions (from 0 to 1).

The obtained results are shown in Figure 1, as the average scores and their standard deviation over 30 executions, for parameters $k \in \{2, 4\}$ and $d \in \{10 \dots 28\}$. As shown by the first two columns, *Borgelt* and *EWKM* have trouble identifying the correct dimensionality of the clusters: As can be seen by the high standard deviation, they produce unstable results, which depend on their initialization. *EWKM* performs significantly worse for $k = 4$ clusters. *Prosecco* succeeds in evaluating the expected d_r everytime.

5.2 Real Data: The Newsgroups Dataset

Another preliminary experiment on high-dimensional and sparse real data is conducted on four classical datasets from the Newsgroups database (Jing et al., 2007) taken from the UCI Machine Learning Repository (Lichman, 2013). They consist of several collections of mails from various newsgroups, whose characteristics are given in Table 1: A2 and B2 are composed of mixed examples of two newsgroups, whereas A4 and B4 consist of examples from four newsgroups. A2 and A4 mix examples from newsgroups whose topics of discussion and vocabulary do not overlap. B2 and B4 are allegedly harder datasets, as they mix examples from newsgroups for which vocabulary might overlap. From each original newsgroup, $n_c = 400$ points are taken randomly for each experiment.

Standard preprocessing techniques are required in order to apply clustering algorithms. In these experiments, they consist in standard tokenization and stemming using the NLTK library (Bird et al., 2009). Data is then converted to the TF-IDF representation (Salton and Buckley, 1988). The resulting datasets typically present thousands of dimensions; for instance, $A2$ has 6331 dimensions.

Using the labels from the original dataset, the experiments measure the mean and standard deviation, over 100 runs, of the accuracy of each algorithm on the datasets. For *Prosecco* and *EWKM*, the tests are run with γ varying from 10^{-8} to 10^{9}, and only the best mean is kept. For the *Borgelt* algorithm, β takes values in $\{0.1, 0.2, \cdots, 0.9\}$.

Table 1: Original newsgroups used in the four considered datasets.

A2	B2	A4	B4
alt.atheism	talk.politics.mideast	comp.graphics	comp.graphics
comp.graphics	talk.politics.misc	rec.sport.baseball	comp.os.ms-windows
		sci.space	rec.autos
		talk.politics.mideast	sci.electronics

The results, presented in Table 2, show that *Prosecco* performs better than *EWKM* and *Borgelt* who are rather disappointing: *EWKM* performs significatively worse

than *Prosecco*. The *Borgelt* algorithm produces a trivial solution on $c = 4$ datasets, where each x_i is equally assigned to each cluster.

Table 2: Mean accuracy of the *Prosecco*, *EWKM* and *Borgelt* algorithms on the datasets from Table 1.

	A2	**B2**	**A4**	**B4**
Prosecco	0.910 ± 0.089	0.737 ± 0.116	0.476 ± 0.008	0.443 ± 0.029
EWKM	0.613 ± 0.061	0.583 ± 0.050	0.351 ± 0.064	0.312 ± 0.033
Borgelt	0.549 ± 0.036	0.526 ± 0.31	– ± –	– ± –

During the experiments, we observed that *EWKM* was several times faster than the *Prosecco* and *Borgelt* algorithms. We attribute these differences in speed to the fact that *EWKM* is only based on numerical computations, implemented using the NumPy (Walt et al., 2011) library. *Borgelt* and *Prosecco*, on the other hand, rely on procedures which explicitely search and set to 0 the smallest values in the arrays. These procedures are written in Python code and are thus slower.

6 Conclusion

This paper proposes an original cost function for fuzzy subspace clustering with a regularization term promoting sparse description of the subspaces. This regularization term features an ℓ_0 penalty which is non-differentiable and prevents the use of the usual implementation techniques. However, the considered settings allow for an exact recovery of the dimension weights through the use of proximal gradient descent.

A fuzzy subspace clustering algorithm based on alternate optimization and proximal descent is proposed under the name *Prosecco*. It is compared to other FSC algorithms on benchmarks of artificial and real data. The results of these benchmarks show that *Prosecco* provides an efficient estimation of subspaces dimensionality as well as a practical fuzzy clustering algorithm. Finally, proofs of correctness of the proposed algorithm and proximal operator are given.

Future research will include the application of *Prosecco* to more challenging datasets, as well as the post-processing of the results: As *Prosecco* and other FSC algorithms produce sparse descriptions of the subspaces, their description becomes clearer. The goal is then to quantify this gain and to use this advantage to produce meaningful summaries to the user.

Moreover, the choice of the regularization parameter γ will be studied both from a theoretical and a practical point of view. As this problem is not limited to the *Prosecco* algorithm, a general study of balancing parameters for other sparsity-inducing FSC algorithms will be conducted as well.

References

Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. New York. DOI: 10.1145/276305.276314.

Bird S, Klein E, Loper E (2009) Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc.

Borgelt C (2008) Feature weighting and feature selection in fuzzy clustering. In: IEEE International Conference on Fuzzy Systems, Institute of Electrical and Electronics Engineers, Hong Kong, pp. 838–844. DOI: 10.1109/fuzzy.2008.4630468.

Deng Z, Choi KS, Jiang Y, Wang J, Wang S (2016) A survey on soft subspace clustering. Information Sciences 348:84–106. DOI: 10.1016/j.ins.2016.01.101.

Duchi J, Shalev-Shwartz S, Singer Y, Chandra T (2008) Efficient projections onto the L1-ball for learning in high dimensions. In: Proceedings of the 25th International Conference on Machine learning, Association for Computing Machinery, New York, pp. 272–279. DOI: 10.1145/1390156.1390191.

Elhamifar E, Vidal R (2009) Sparse subspace clustering. In: IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Institute of Electrical and Electronics Engineers, Miami, pp. 2790–2797. DOI: 10.1109/CVPR.2009.5206547.

Guillon A, Lesot MJ, Marsala C, Pal NR (2016) Proximal Optimization for Fuzzy Subspace Clustering. In: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, Cham, pp. 675–686. DOI: 10.1007/978-3-319-40596-4_56.

Guillon A, Lesot MJ, Marsala C (2019) A proximal framework for fuzzy subspace clustering. Fuzzy Sets and Systems 366:34–45. DOI: 10.1016/j.fss.2018.06.006.

Hastie T, Tibshirani R, Wainwright M (2015) Statistical learning with sparsity: The lasso and generalizations. Chapman and Hall/CRC, Boca Raton.

Jing L, Ng MK, Huang JZ (2007) An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. IEEE Transactions on Knowledge and Data Engineering 19(8):1026–1041. DOI: 10.1109/TKDE.2007.1048.

Keller A, Klawonn F (2000) Fuzzy clustering with weighting of data variables. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 8(06):735–746, Berthold RM, Lenz H, Bradley E, Kruse R, Borgelt C (eds.). DOI: 10.1142/S0218488500000538.

Klawonn F, Höppner F (2003) What is fuzzy about fuzzy clustering? Understanding and improving the concept of the fuzzifier. In: Advances in Intelligent Data Analysis V. Springer, Berlin, pp. 254–264. DOI: 10.1007/978-3-540-45231-7_24.

Lichman M (2013) Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. URL: `http://archive.ics.uci.edu/ml`.

Natarajan BK (1995) Sparse approximate solutions to linear systems. SIAM Journal on Computing 24(2):227–234. DOI: 10.1137/S0097539792240406.

Parikh N, Boyd S (2013) Proximal algorithms. Foundations and trends in optimization 1(3):123–231. URL: `https://web.stanford.edu/~boyd/papers/prox_algs.html`.

Parsons L, Haque E, Liu H (2004) Subspace Clustering for High Dimensional Data: A Review. ACM SIGKDD Explorations Newsletter 6(1):90–105. DOI: 10.1145/1007730.1007731.

Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Information processing & management 24(5):513–523. DOI: 10.1016/0306-4573(88)90021-0.

Tibshirani R (1996) Regression shrinkage and selection via the lasso. Royal Statistical Society. DOI: 10.1111/j.2517-6161.1996.tb02080.x

Vidal R (2011) Subspace Clustering. IEEE Signal Processing Magazine 28(2):52–68. DOI: 10.1109/MSP.2010.939739.

Walt Svd, Colbert SC, Varoquaux G (2011) The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science and Engineering 13(2):22–30. DOI: 10.1109/MCSE.2011.37.

Witten DM, Tibshirani R (2010) A framework for feature selection in clustering. Journal of the American Statistical Association 105(490):713–726. DOI: 10.1198/jasa.2010.tm09415.