Directional Clustering through Matrix Factorisation

Thomas Blumensath
ISVR Signal Processing and Control Group
University of Southampton
SO17 1BJ, UK

Abstract
This paper deals with directional clustering. Feature vectors are clustered conditional on between feature angles, required to be small within a cluster. This constraint arises in several applications, including document classification and human brain imaging. Using ideas from the field of constrained low-rank matrix factorisation and sparse approximation, a novel approach is presented that differs from classical clustering methods, such as semi-Nonnegative Matrix Factorisation (semi-NMF) or k-means clustering, yet combines some aspects of both. As in NMF, the matrix decomposition is iteratively refined to optimise a data fidelity term, however, no positivity constraint is enforced directly. Instead, as in k-means, each optimisation step is followed by a hard cluster assignment. This leads to an efficient algorithm that is here shown to outperform common competitors. In addition to a detailed theoretical analysis of some of the algorithm’s main properties, the approach is evaluated empirically on a range of toy problems, several standard text clustering data-sets and a high dimensional problem in brain imaging, where functional MRI data is used to partition the human cerebral cortex into distinct functional regions.

Keywords: Clustering, Iterative Hard Thresholding, Inverse Problems

1. Introduction
Clustering (Bishop, 2006; MacQueen, 1967), the problem of grouping objects into distinct classes, is a major problem in signal processing and machine learning. Clustering has a long history in statistics and data analysis and a wide range of approaches has been proposed over the years, from generic algorithms to problem specific solutions. Our aim here is the development of a generic strategy for directional clustering that is computationally efficient yet outperforms other popular methods.

One way to formulate the problem is as follows. Each observed $M$-dimensional feature vector $x_i$ can be modelled as a perturbed instance of a (nominal) cluster centre

$$x_i = d_k + e_i,$$  

where $e_i$ is a ‘noise’ term. With this formulation, unsupervised clustering can be achieved by an estimation of the cluster centres $d_k$ together with the assignment of each feature $x_i$ to one of these centres.

This formulation can be seen as a matrix factorisation problem. Assume that all $N$ column vectors $x_i$ are stacked into a matrix $X$ and that the $K$ centres are stacked into a matrix $D$. Let the errors $e_i$ make up the columns of a matrix $E$. With this notation, we
can then write the model as

\[ X = DS + E, \]  

(2)

where \( S \) is a coefficient matrix. For this equation to be equivalent to the model in (1), \( S \) will have to be a matrix with zero-one entries where each column has a single non-zero entry.

1.1 Clustering as constrained matrix factorisation

From a matrix factorisation perspective, if we knew the centres \( D \), we could estimate \( S \) from \( X \) by inversion of \( D \). However, to agree with (1), each column of \( S \) must be constrained to be 1-sparse, that is, it must contain a single non-zero entry, which has to be 1. Thus, the columns of \( S \) have to be sparse and quantised. In this respect, part of the clustering problem is a sparse approximation problem. However, the above argument assumed knowledge of \( D \), which in unsupervised problems is also unknown.

To estimate both \( D \) and \( S \) several constraints can be brought to bear. We have already discussed the fact that \( S \) is required to be sparse. Sparsity constrained matrix factorisations are used, for example, in many Independent Component Analysis (ICA) methods (Hyvärinen et al., 2001), however, the sparsity constraint in ICA does not enforce the sparse decomposition to be one-sparse (i.e. columns in \( S \) found in an ICA decomposition have typically more than one non-zero entry) and so ICA is not normally used for clustering directly.

Another constraint that can help in clustering is the realisation that the number of cluster centres \( K \) is also typically much lower than the number \( N \) of feature vectors and so, the decomposition \( DS \) will have a matrix rank of at most \( K \) and is thus a low-rank approximation of \( X \). Low-rank approximations are classical, with Principal Component Analysis being a typical example. However, more recently, low-rank decompositions have received renewed interest due to their ability to recover matrices from few measurements as happens for example if matrix entries are missing (Candes and Recht, 2009). This has led to several advances in computational methods to estimate low-rank decompositions, however, many of these ideas are not yet used to solve clustering problems.

Finally, the fact that the entries in \( S \) are either zero or one means that \( S \) is a nonnegative matrix. The clustering problem is thus also an instance of the semi-Nonnegative Matrix Factorisation problem, where a factorisation \( X \sim DS \) is computed with the constraint that the matrix \( S \) has only positive entries (Ding et al., 2010). This is a relaxation of more classical Nonnegative Matrix Factorisation (NMF)(Shahmaz et al., 2006; Ding et al., 2005)), where both \( D \) and \( S \) are constraint to be positive (Paatero and Tapper, 1994). Interestingly, in contrast to the methods discussed above, non-negative matrix factorisation ideas have been used heavily in clustering (Li and Ding, 2006). A justification for the use of non-negative matrix factorisations for clustering comes from the assumption that non-negativity is a very strong constraint. The assumption is that once a decomposition is found in which \( S \) has non-negative entries and given that the data actually follows a model similar to that in (1), then it is likely that each column of \( S \) will have a single large entry, with the other entries being small. In this case, a simple post-processing step is able to decide on a unique cluster assignment.

However, as sparsity of the columns of \( S \) is by no means guaranteed more generally, it is no surprise that both sparsity and non-negativity have also been used jointly in matrix decompositions, leading to sparse nonnegative matrix factorisations (Hoyer, 2004) which
have been used for clustering in (Kim and Park, 2008). An interesting variant of the approach, which has some similarity to our method, is that reported in (Peharz et al., 2010), which uses a hard constraint in each iteration to ensure that elements in the decomposition (say elements in the columns of $S$) are either strictly positive or exactly zero, such that there are only few non-zero elements.

As alluded to above, when using NMF or semi-NMF for clustering, a two stage approach is typically used in which the non-negative decomposition is calculated first. Only then is a unique cluster assignment made, that is, in our notation, a decomposition $X \sim DS$ is calculated first using positivity (and possibly additional sparsity) constraints on $S$. But these constraints do not normally guarantee that columns of $S$ are 1-sparse, that is, columns of $S$ typically have more than a single non-zero entry. A hard cluster assignment is thus typically made only after the initial factorisation has been found, for example, using a thresholding approach to find the appropriate 1-sparse matrix $S$.

In contrast to this two stage approach, we propose an iterative algorithm to optimise both, the matrix decomposition itself and the hard cluster assignment. In this respect, our approach is similar to traditional k-means clustering, where cluster assignment and cluster centre estimation are iterated. Our motivation for this comes from the field of sparse approximation, where it has been shown that an iterative optimisation of sparsity and data fidelity generally leads to greatly superior results to those achievable with a simple thresholding step applied only after the fidelity term is fully optimised.

1.2 Directional Clustering

Having formulated the general clustering problem as a constrained matrix decomposition

$$X = DS + E,$$  \hspace{1cm} (3)

where columns of $S$ are one sparse and quantised, we now turn to our more specific clustering problem - directional clustering. In many applications, such as text analysis or functional MRI data analysis, feature vectors $x_i$ are only given up to a scaling factor. One way to deal with this would be through a normalisation step that scales features $x_i$ to equal length prior to clustering. If we were, for example, to use standard k-means with these normalised vectors, then the cluster centres (which are convex combinations of clustered features) will be shorter than all the feature vectors themselves. In the iterative k-means approach, this can lead to cluster centres of varying length which in turn will have an effect on cluster assignment in the next step. An alternative would be to also normalise the cluster centres after each update. This approach, known as directional (or spherical) k-means (Zhong, 2005; Hornik et al., 2012), can be shown to optimise a cost function that only depends on the angle between features and cluster centres.

To see this, let $\bar{x}_i$ and $\bar{d}_k$ be normalised feature vectors and cluster centres of unit length. The Euclidean cluster cost function $\|\bar{x}_i - \bar{d}_k\|$ (used not only in standard k-means but also in NMF) then only depends on the angle between features and cluster centres.

$$\langle x_i, d_k \rangle \left/ \|x_i\|\|d_k\| \right.$$  \hspace{1cm} (4)

Thus, in this setting, we cluster directions rather than treating features as single points in space. Apart from the directional (or spherical) k-means algorithm (Zhong, 2005; Hornik...
et al., 2012), other clustering approaches have been adapted to this setting, such as those based on mixtures of von Mises Fisher distributions (Banerjee et al., 2005). Directional clustering is also related to subspace clustering (Elhamifar and Vidal, 2013), whenever the subspaces are one dimensional. Note however that in directional clustering, we assume that features have an unknown scaling, but that this scaling is positive. That is, we know the direction in which the feature points. This is a different assumption to that made in subspace clustering, where the "sign" or direction of the feature is also unknown. In effect, for one-dimensional subspaces, subspace clustering bases decisions on the absolute angle, that is on \[ \frac{\langle x_i, d_k \rangle}{\| x_i \| \| d_k \|} \].

To summarise the above discussion, directional clustering tries to optimise the following optimisation problem:

\[
\max_{\{d_k\}|C_k|} \sum_{k=1}^{K} \sum_{i \in C_k} \frac{\langle x_i, d_k \rangle}{\| x_i \| \| d_k \|} : \| d_k \| > 0
\]  

(5)

where we introduce the sets \( C_k \) which partition the feature vectors into the individual clusters, that is the sets \( C_k \subset [1, 2, \ldots, N] \) are such that \( C_k \cap C_{\hat{k}} = \emptyset \) for all \( k \neq \hat{k} \) and \( \bigcup_{k} C_k = [1, 2, \ldots, N] \). In words, we have to search over all partitions of the input feature vectors and over all possible vectors \( d_k \) to optimise the sum over the angles between the cluster centres \( d_k \) and the feature vectors assigned to these centres.

1.3 Motivation

We are here interested in a setting where we have many clusters \( \sim 100 \) and a very large number of data points \( \sim 100,000 \). We also assume that each feature vector is high dimensional \( \sim 10,000 \). In this setting two issues have to be overcome, 1) the computational complexity of the method needs to be low and 2) the method needs to be able to deal with the combinatorial explosion in cluster assignments due to the large number of clusters. For this reason, we do not pursue the optimisation of a more sophisticated statistical model, such as, for example, an extension of the von Mises Fisher mixture model, as this is likely to lead to higher computational demands. Instead, we here propose an approach that is based on a more general optimisation problem, which has the advantage of leading to a very efficient algorithm for clustering.

The need for an efficient algorithm comes from our work in brain imaging, where the human brain is to be subdivided into hundreds of functional divisions based on features generated from Magnetic Resonance Imaging (MRI) data. Feature vectors are generated for hundreds of thousands of locations within the brain and each of these feature vectors can contain tens of thousands of elements. Furthermore, unknown feature scaling and spatially varying noise complicate the analysis. Extensive previous experiments with a wide range of methods have demonstrated the need for efficient methods that do not rely too heavily on different model assumptions (such as within cluster variance constraints), which in the application at hand, cannot normally be specified with certainty.

Another motivation for the development of the algorithm here comes also from our particular application in brain imaging, where, in addition to our requirement to cluster brain areas with similar brain activity, we also know that individual areas, which are typically represented by hundreds of features each, are spatially continuous. By designing a constrained
matrix factorisation based approach here, our long term goal is to explore the possibility to include additional constraints such as spatial contiguity of clusters into our formalism. This is conceptually easy within our proposed framework if we think of the hard thresholding step as a projection, computationally efficient approaches to this more complex projection are more difficult to resolve and are currently under separate investigation.

2. Directional Clustering via matrix factorisation

The directional clustering problem introduced above can also be formulated as a matrix decomposition problem

\[
\min_{D,S} \|X - DS\|_F. \tag{6}
\]

The difference now is that we assume that the columns of \(X\) are normalised to unit length and that the optimisation is over all \(D\) with unit norm columns. However, we still have the same constraint on \(S\), which is required to have 1-sparse columns where the non-zero entry is 1.

This is a minimisation with several non-convex constraints. For example, optimising over \(S\) with binary entries and over \(D\) with unit norm columns is not a trivial task, not to mention the combinatorial problem of optimising over all possible cluster assignments. To address these difficulties, we relax the two constraints on the scaling of \(D\) and \(S\) and use an iterative projection approach similar to the iterative hard thresholding algorithm used for sparse approximation. The relaxed optimisation will try and minimise the error \(\|x_i - s_{i,k}d_k\|^2\) over both \(s_{i,k}\) and \(d_k\), where now the magnitude of \(s_{i,k}\) and the length of \(d_k\) are unconstrained. As \(x_i\) has unit length, the cost is

\[
\|x_i - s_{i,k}d_k\|^2 = 1 + s_{i,k}^2\|d_k\|^2 - 2s_{i,k}\langle x_i, d_k \rangle, \tag{7}
\]

which, if minimised over \(s_{i,k}\), has a minimum at

\[
s_{i,k}^* = \frac{\langle x_i, d_k \rangle}{\|d_k\|^2}, \tag{8}\]

at which point the cost is

\[
\min_s \|x_i - s_{i,k}d_k\|^2 = 1 - \frac{\langle x_i, d_k \rangle^2}{\|d_k\|^2}, \tag{9}\]

that is, the cost at the minimal value of \(s\) is related to the angle between \(x_i\) and \(d_k\), as is the goal in directional clustering.

This suggests that we do not need to impose the non-convex constraints on \(D\) and \(S\). Thus, we solve the relaxed optimisation problem

\[
\min_{\{d_k\},\{s_i\},\{C_k\}} \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - s_i d_k\|^2, \tag{10}\]

where the \(s_i\) are the non-zero elements in \(S\). This remains a combinatorial problem and approximate optimisation has to be used. As in k-means, this is done iteratively. For given centres \(d_k\) and cluster assignment \(C_k\), the optimal choice for the \(s_i\) is that in (8).
On the other hand, if we assume knowledge of $C_k$ and $S$, then the optimal choice of $d_k$ is calculated minimising (6), which is minimised using

$$D = XS^T (SS^T)^{-1},$$

where for now, we assume that the inverse exists (See below for a more detailed discussion on this). If the columns of $S$ are one sparse, then this inverse is simply a diagonal matrix whose entries are the inverse of the sum of the squared entries of each row of $S$. Note that the non-zero entries in the $n^{th}$ row of $S$ indicate those feature vectors $x_i$ that are assigned to cluster $n$. Thus, the estimate of each cluster centre $d_k$ is a weighted sum of feature vectors

$$d_k = \sum_{j \in C_k} \frac{s_i}{\sum_{i \in C_i} s_i^2} x_j.$$

This is thus a more general formulation of the standard k-means algorithm, in which we would have $s_i = 1$ which would lead to $\sum_{i \in C_i} s_i^2 = \frac{1}{|C_i|}$.

3. The algorithm

To optimise (10) we take an approach that is inspired by methods recently developed in the area of compressed sensing and low-rank matrix completion (Blumensath and Davies, 2009; Blumensath, 2009). For a given matrix $X$, we need to find $D$, $S$ and the $C_k$. We do this iteratively, by alternatively optimising $D$, $S$ and the cluster assignment $C$. Summarising the above discussion, for given $S$, we can calculate the optimal $D_n$ as

$$D_n = XS_n^T (S_nS_n^T)^{-1}$$

where we assume for now that $S_nS_n^T$ is invertible (see discussion below).

Similarly, for a given $D_n$, $S_n = XS_n^T (S_nS_n^T)^{-1}$ if we ignore the hard cluster assignment requirement, then we need to solve the following minimisation

$$S_{n+1} = \text{arg min}_S \|X - D_nS\|_F^2$$

$$= \text{arg min}_S \|X - XS_n^T (S_nS_n^T)^{-1}S\|_F^2.$$  

(14)

This minimisation can be done explicitly if $(D^TD)$ is invertible, in which case we have $S_{n+1} = (D^TD)^{-1}D^TX$.

This optimal point $S_{n+1}$ does however not satisfy the sparsity requirement, that is, they do not provide a hard cluster assignment. To enforce this we could use a hard-thresholding step known from sparse optimisation (Blumensath and Davies, 2009), which finds the best approximation to the current estimate of $S$ in which each column is 1-sparse. It can easily been shown that, for a given $S$, this optimum is achieved by thresholding each column in $S$ in such a way that only the largest entry in each column is kept. For simplicity, we write this non-linear operation as $\hat{S}_n = P(S_n)$, where the notation $\hat{S}_n$ reminds us that this matrix has 1-sparse columns. It is important to note that we here base the thresholding rule on the values of the entries in $S$ and not on their magnitude as is done normally in sparse methods. This in effect ensures that after thresholding, $\hat{S}_n$ is a positive matrix (unless there
was a column in S that had only negative entries). Our method thus favours non-negative decompositions similar to semi-NMF.

An alternate to this approach is one that follows even closer the ideas from (Blumensath and Davies, 2009) and avoids the requirement for a matrix inverse. Inspired by an optimisation transfer approach (Blumensath and Davies, 2008), we can update the matrix $\bar{S}_n$ using the iteration

$$
S_{n+1} = \bar{S}_n + \mu_n D_n^T(X - D_n \bar{S}_n),
$$

$$
\bar{S}_{n+1} = P(S_{n+1}),
$$

(15)

where $\mu_n$ is a step size chosen appropriately (see below and the discussion in (Blumensath and Davies, 2010)).

Our clustering algorithm iterates through the steps derived above. There are several ways to initialise the algorithm. For example, we could initialise the method using the best low-rank approximation to $X$ of a given rank $K$. Alternatively, an initial decomposition based on the output of some alternative clustering method could be used.

The algorithm is summarised below.

1. **INPUT:** data matrix $X$, number of clusters $K$

2. initial decomposition of $X$ into low rank factorisation (e.g. using an SVD or some initial cluster assignment) $X = DS + E$.

3. iterate until some convergence criterion is met

   (a) Calculate cluster assignment: $\bar{S} = P(S)$

   (b) Check for empty clusters and randomly re-initialise (see subsection 4.2 below)

   (c) Update cluster centres: $D = X\bar{S}^T(\bar{S}\bar{S}^T)^{-1}$ (and optionally normalise columns of $D$ (see subsection 4.1))

   (d) Update cluster weights: $S = S + \mu D^T(X - DS)$ or $S = (D^TD)^{-1}D^TX$ (and optionally normalise rows of $S$ (see subsection 4.1))

Important, $D^TD$ is a $K$ by $K$ matrix and so, if $K$ is relatively small, then this matrix inverse can be computed relatively fast. For the applications for which we developed this approach, we typically have 100 clusters, but hundreds of thousands of feature vectors, each of length 1000 or so. In this setting, the computational bottle neck is the computation of $D^TX$.

4. Several important issues

4.1 A note on scale ambiguities, normalisation and hard thresholding

Before continuing in our development, it is important to note that the optimisation problem (10) has several indeterminacies, which are common to most matrix factorisation problems. A re-ordering of the columns in $D$ and rows in $S$ will not change the cost function, which thus has multiple minima (at least one for each permutation). The solution found with
our optimisation approach will thus depend on the initialisation. Another issue is that re-scaling of $d_k$ can always be counteracted by an appropriate inverse scaling of the associated $s_{i,k}$. This is a direct consequence of our desire that the cost is invariant to scaling. Thus, to prevent numerical stability issues when using an iterative algorithm that alternatively optimises $D$, $S$ and $C$, we use a re-scaling step that ensures that after each update of $D$, columns are re-normalised to unit length. As scaling of the columns of $D$ is arbitrary, cluster assignment (i.e. the thresholding step), which is based on a comparison between the entries in $S$, faces the same ambiguities. To overcome this ambiguity we use a re-scaling step that either normalises the columns in $D$ after each update, or, we normalise rows in $S$ before thresholding is applied. The first approach works well in most cases, though we found that for situations where clusters are roughly of equal size, the second approach is beneficial (see our simulations below).

We also mentioned normalisation of the columns in $X$ above. This is required as we are only interested in the direction and not the length of the features $x_i$. However, additional flexibility is provided if we allow the $x_i$ to be of different length. In this case, the contributed each feature makes to the error term in (10) (for the optimal choice of $s_i$) will vary accordingly to feature length. Assume that $\|x_i\| = c_i$ and that $\hat{x}_i = x_i/c_i$. Assume that for a given $d_k$ $s_i$ minimises $\|\hat{x}_i - \tilde{s}_i d_k\|^2$. In this case, $s_i c_i$ is the optimal scaling for $\|x_i - \tilde{s}_i d_k\|^2$ and the cost becomes:

$$\min_{\{d_k\},\{s_i\},\{C_k\}} \sum_{k=1}^{K} \sum_{i \in C_k} c_i^2 \|\hat{x}_i\|^2 + c_i s_i^2 \|d_k\|^2 - c_i s_i \langle x_i, d_k \rangle.$$  \hspace{1cm} (16)

Thus, if we scale any one $x_i$ the contribution of that $x_i$ to the overall clustering cost is also scaled. Thus, any large $x_i$ contributes more to the overall cost than small $x_i$. This property could be used to weight different $x_i$ in settings where some observations are known to contain more noise than others. However, for simplicity, we here assume all $x_i$ to contribute equally to the cost.

4.2 A note on rank, empty clusters and matrix inversion

The above algorithm requires the inversion of $D^T D$. This obviously requires that $D^T D$ is full rank and, as the columns of $D$ are weighted combinations of feature vectors, this implies that it is necessary for $X$ to have a rank that is at least $K$ so that our approach will only cluster features that span a space that is larger than the number of clusters we are trying to estimate.

A similar problem is that the estimate of $\bar{S}$ might have rows in which all entries are zero (remember, our thresholding operation works column-wise). If this happens, evasive action has to be taken\(^1\). The approach we suggest is to optimally estimate all those columns in $D$ for which there are non-zero rows in $\bar{S}$. The remaining rows are then re-instantiated. We here take an approach in which we set these columns randomly to elements from $X$.

---

\(^1\) This problem also arises in k-means clustering, where it can occur that there is a cluster centre with no feature vectors assigned to it.
4.3 A note on the estimation of the number of clusters

Our method requires the specification of the number of clusters. To do this automatically, it is possible to run the algorithm with different numbers of clusters and use an optimisation criterion, such as, for example, the Akaike information criterion to select the “optimal” size. Instead of running the algorithm for all cluster sizes within a range, this can be done efficiently using a line search (Kiefer, 1953).

5. Global minima, fixed points and convergence

We here concentrate on the analysis of one variant of the algorithm. Assume we use the steps

\[ S_{n+1} = S_n + \mu_n D_n^T (X - D_n S_n), \]  
\[ \bar{S}_{n+1} = P(S_{n+1}) \]  

and

\[ D_{n+1/2} = X \bar{S}_{n+1/2} (\bar{S}_{n+1/2} \bar{S}_{n+1/2})^{-1} \]

and normalise \( D_{n+1} \) and \( \bar{S}_{n+1} \) after each update of \( D_{n+1} \).

For notational convenience, we write

\[ \bar{S}_{n+1/2} = P(S_{n+1}) \]

and let \( \bar{S}_{n+1} \) and \( D_{n+1} \) be the rescaled versions, such that \( D_{n+1} \) has unit norm columns and such that \( D_{n+1} \bar{S}_{n+1} = D_{n+1/2} \bar{S}_{n+1/2} \).

5.1 Notation

In this section we will make use of the following notation.

- Let \( s_i \) be a row-vector containing the non-zero entries in \( \bar{S} \) for which the non-zero coefficient is in row \( i \) of \( \bar{S} \).
- Let \( X_k \) be the sub matrix of \( X \) containing those columns for which the columns in \( \bar{S} \) have a non-zero entry in row \( k \).
- Let \( \Phi \) be a positive, diagonal matrix.
- Let \( q_i \) be the \( i^{th} \) diagonal element of the matrix \( (\bar{S} \bar{S}^T)^{-1} \) and define \( p_i \) in the same way for matrix \( \Phi \). Note that \( p_i = 1/(\|q_i X_i s_i^T\|) \), so that \( q_i p_i = 1/\|X_i s_i^T\| \).
- Let \( X_i = U_i \Sigma_i V_i^T \) be the SVD of \( X_i \), which is a sub-matrix of \( X \) containing those columns in \( X \) clustered into cluster \( i \).
- Let \( s_i = \alpha_i^n V_i^T \) be the expansion of the cluster coefficients \( s_i \) in the svd basis \( V_i \).
- For two matrices \( A \) and \( B \), we will use the inner product notation \( \langle A, B \rangle = \sum_{i,j} a_{i,j} b_{i,j} \), where the \( a_{i,j} \) and \( b_{i,j} \) are the elements in the \( i^{th} \) row and \( j^{th} \) column of \( A \) and \( B \) respectively. Note that this is the inner product that induces the Frobenius norm, making the space of matrices a Hilbert space.
With this notation, assume we have clustered $X$ into some decomposition $DS$, where $S$ has one sparse columns. For the $i^{th}$ cluster, the feature in that cluster are modelled with a single cluster centre, the $i^{th}$ column in $D$. This column is multiplied by all those elements in $S$ that have a non-zero entry in row $i$. Thus, if $d_i$ is the $i^{th}$ column in $D$, then the features in cluster $i$ are approximated with scaled versions of $d_i$, i.e. $X_i \approx d_is_i$. Furthermore, $d_i$ itself is a function of $X_i$ and $s_i$, that is

$$d_i = \frac{X_is_i^T}{s_is_i^T},$$

(21)

or, if we normalise $d_i$, then

$$d_i = \frac{X_is_i^T}{\|X_is_i^T\|}.$$  

(22)

However, as in the normalisation step, both $D$ and $S$ are scaled, the normalisation constant cancels in the product $d_is_i$, which we thus write as

$$d_is_i = X_is_i^T,$$

(23)

### 5.2 Main results

In the remainder of this section we proof the following main results.

The first theorem we will derive will characterises the global minimum of the clustering cost function and states that the minimum over $D$ and $S$ is found for some partition of $X$ into sub matrices $X_i$ such that the non-zero elements in $S$, that is, the $s_i$ are right singular vectors of the sub matrices $X_i$ associated with the largest singular value.

The second set of results then shows that fixed points of the algorithm are also associated with $s_i$ that are right singular vectors of the feature matrix $X_i$.

We finally look at convergence and our third results shows that the algorithm converges to some cluster assignment, where cluster weights converge to the singular subspace of $X_i$ associated with the largest singular value. This convergence depends on the choice of the step size $\mu$.

### 5.3 The global minima

Our aim is the decomposition of the feature matrix $X$ into a low rank approximation $DS$, where $S$ has one sparse columns. With the columns in $X$ normalised to unit length, we thus aim at the minimisation of

$$\min_{D,S} \|X - DS\|_F^2,$$

(24)

under the constraints that $S$ has 1-sparse columns and $D$ has normalised columns.

The sparsity pattern of $S$ determines the cluster assignment. Thus, we start our analysis of the cost function by assuming that the cluster assignment, and thus the position of the non-zero elements in $S$ is fixed. Under this condition, we have the following result.

**Lemma 1** For a fixed cluster assignment the minimal cost is achieved for $s_i$ which are scaled versions of the right singular vector (or an element in the subspace spanned by the singular vectors) associated with the largest singular value(s) of $X_i$.
Proof Using the notation above we note that the cost function
\[ \|X - D_{n+1/2}S_{n+1/2}\|_F^2 = \|X - D_{n+1}S_{n+1}\|_F^2 \] (25)
can be written as (dropping the iteration subscript \(n\))
\[ \sum_i \|X_i - X_i s_i^T s_i \|_F^2. \] (26)

Importantly, we recognise that \(s_i^T s_i\) is an orthogonal projection of the rows of \(X\) onto the one dimensional subspace spanned by \(s_i\), the minimum over all \(s_i\) is thus found if \(s_i\) lies in the subspace spanned by the right singular vectors of \(X_i\) associated with the largest singular values.

As there are only finitely many ways to assign features to clusters, we have thus proven the following result.

**Theorem 2** The global minima of the clustering cost function is achieved for \(s_i\) that lie in the subspace spanned by the right singular vectors of \(X_i\) associated with the largest singular values, where the \(X_i\) are non-empty sub-matrices of \(X\), such that each column in \(X\) is in exactly one sub-matrix.

5.4 Stationary points

Let us next turn to the fixed points of the algorithm, that is, to an analysis of those \(S\) that satisfy the following condition
\[ \Phi \bar{S} = P(\bar{S} + \mu_n((\bar{S}\bar{S}^T)^{-1}\bar{S}X^T (X - X\bar{S}^T(\bar{S}\bar{S}^T)^{-1}\bar{S}), \] (27)
where \(\Phi\) is a diagonal matrix (a function of \(\bar{S}\)) that normalises the columns of the matrix \(X\bar{S}^T(\bar{S}\bar{S}^T)^{-1}\). Note that \(\bar{S}\bar{S}^T\) is diagonal and so is \(\Phi\). Because \(\bar{S}\) is one-column sparse, it is again instructive to re-write the above condition in terms of the row vectors \(s_i\).

With this notation, we have the following stationarity condition
\[ \mu_i \frac{s_i X_i^T}{\|X_i s_i^T\|} X_k (I - \frac{s_i^T s_k}{s_i s_k^T}) \begin{cases} c_i s_i, & \text{if } i = k \\ (1 + c_i) s_i, & \text{otherwise.} \end{cases} \] (28)

Here, the \(c_i\) are constants, \(P_{s_k} = \frac{s_k^T s_k}{s_i s_i^T}\) is a projection and \(P_{s_k}^\perp = (I - \frac{s_k^T s_k}{s_i s_i^T})\) its orthogonal complement.

Because \((I - \frac{s_k^T s_k}{s_i s_i^T})\) is a projection onto the row space orthogonal to \(s_i\), the row vectors of \(X_k (I - \frac{s_k^T s_k}{s_i s_i^T})\) are orthogonal to \(s_i\). Thus, \(s_i X_i^T \|X_i s_i^T\| X_k (I - \frac{s_k^T s_k}{s_i s_i^T})\) is a sum over vectors that are orthogonal to \(s_i\), which implies that the constant \(c_i\) above has to be zero.

We have thus shown the following.
Lemma 3 The stationary points $S$ satisfy the following condition

$$\mu \frac{s_i X_i^T}{\|X_i s_i^T\|} X_k (I - \frac{s_k^T s_k}{s_i s_i^T}) \begin{cases} = 0, & \text{if } i = k \\ < s_i, & \text{otherwise.} \end{cases}$$

(29)

To get an even better understanding of the fixed point condition above, let us write $X_i^{s_i} = X_i s_i^T s_i$ and let $E_i = X_i - X_i^{s_i}$. Note that $X_i^{s_i}$ and $E_i$ have orthogonal rows. Thus, the above lemma shows that $s_i$ is a fixed point if and only if

$$\langle X_i s_i^T, E_i \rangle = 0$$

(30)

and

$$\langle X_i s_i^T, E_k \rangle < s_i,$$

(31)

where the inequality must hold element wise and for all $k \neq i$. Importantly, the first equality above can also be stated as

$$s_i X_i^T X_i = c_i s_i,$$

(32)

for some $c_i$. As this is a typical eigenvalue problem we have proven the following lemma

Lemma 4 The stationary points of the algorithm provide a partition of the data set such that the non-zero elements in $S$ associated with cluster $i$ are eigenvectors of the matrix $X_i^T X_i$.

The above fixed point characterisation highlights the importance of the singular value decomposition. It is thus instructive to write the algorithm in terms of singular values of $X_i$, assuming for now that cluster assignment does not change. In this case, we can re-write the update as

$$s_i^{n+1} = s_i^n + \mu \frac{s_i^n X_i^T}{\|s_i^n X_i^T\|} X_i \left( I - \frac{(s_i^n)^T s_i^n}{s_i^n (s_i^n)^T} \right)$$

(33)

as (dropping the subscript $i$ from $\alpha$ and $\Sigma$)

$$\alpha^{n+1} = \alpha^n + \mu \frac{\alpha^n}{\|\alpha^n \Sigma\|} \Sigma^2 \left( I - \frac{(\alpha^n)^T \alpha^n}{\|\alpha^n\|^2} \right),$$

(34)

where we have right multiplied the equation by $V$. Writing this update element wise, we see that the $k^{th}$ element in $\alpha$ (i.e. $\alpha_k$) is updated as

$$\alpha_k^{n+1} = \alpha_k^n + \mu \left( \frac{\sigma_k^2}{\|\alpha^n \Sigma\|} - \frac{\alpha^n \Sigma^2 (\alpha^n)^T}{\|\alpha^n \Sigma\| \|\alpha^n\|^2} \right) \alpha_k^n.$$

(35)

i.e.

$$\alpha_k^{n+1} = \left( 1 + \mu \left( \frac{\sigma_k^2}{\|\alpha^n \Sigma\|} - \frac{\|\alpha^n \Sigma\|}{\|\alpha^n\|^2} \right) \right) \alpha_k^n.$$

(36)
5.5 Convergence, preliminary results

To derive convergence results for the algorithm, we first derive a range of results that show
the convergence of several related quantities. We start the development by deriving the
algorithm through a majorized cost function, which is to be optimised under the constraint
that the columns of the solution have to be 1 column sparse.

\[
\min_A \|X - D_n A\|_F^2 + \frac{1}{\mu_n} \|\bar{S}_n - A\|_F^2 - \|D_n (\bar{S}_n - A)\|_F^2,
\]

(37)

where the minimisation is done over all matrices \(A\) that have 1-sparse columns. The argument for this basically follows that in (Blumensath and Davies, 2008). We can re-write this as

\[
\min_A \|X - D_n A\|_F^2 + \frac{1}{\mu_n} \|\bar{S}_n - A\|_F^2 - \|D_n (\bar{S}_n - A)\|_F^2
= \min_A \|X\|_F^2 + \|D_n A\|_F^2 - 2\langle D_n^T X, A \rangle + \frac{1}{\mu_n} \|\bar{S}_n\|_F^2 + \frac{1}{\mu_n} \|A\|_F^2 - \frac{1}{\mu_n} 2\langle \bar{S}_n, A \rangle
- \|D_n \bar{S}_n\|_F^2 - \|D_n A\|_F^2 + 2\langle D_n^T D_n \bar{S}_n, A \rangle
= \min_A -2\langle D_n^T X, A \rangle + \frac{1}{\mu_n} \langle A, A \rangle - \frac{1}{\mu_n} 2\langle \bar{S}_n, A \rangle + 2\langle D_n^T D_n \bar{S}_n, A \rangle
\]

Thus, we need to minimise

\[
\left\langle \frac{1}{\mu_n} A - \frac{2}{\mu_n} \bar{S}_n - \frac{2}{\mu_n} D_n^T (X - D_n \bar{S}_n), A \right\rangle.
\]

Taking derivatives w.r.t. the elements in \(A\) and setting to zero, we get

\[
A = \bar{S}_n + \mu D_n^T (X - D_n \bar{S}_n),
\]

(38)

that is

\[
A = (\bar{S}_n + \mu D_n^T (X - D_n \bar{S}_n)),
\]

(39)

which, to impose the sparsity constraint on the columns of \(A\) has to be thresholded appropri-
ately.

Importantly, we want the majorised cost function to bound the original clustering cost
function. To guarantee this, we need to choose \(\mu_n\) such that for all \(A\) with one sparse
columns, the majorisation term

\[
\frac{1}{\mu_n} \|\bar{S}_n - A\|_F^2 - \|D_n (\bar{S}_n - A)\|_F^2 > 1/c \|\bar{S}_n - A\|_F^2
\]

(40)

for some constant \(c > 0\) independent of \(n\). As the columns of \(D_n\) are normalised and as
columns in \((\bar{S}_n - A)\) are two sparse, \(\|D_n (\bar{S}_n - A)\|^2 \leq 4 \|(\bar{S}_n - A)\|^2\) so we can choose \(\mu < 1/4\). In fact, equality only holds if there are two columns in \(D_n\) that are equal in which case we can combine these two clusters and re-initialise the empty cluster. W.l.g we can thus assume that \(\mu = 0.25\). Also note that, if cluster assignment does not change between
iterations, then \(\bar{S}_n - A\) will have one sparse columns, in which case we can choose \(\mu < 1\).

This suggests a line search approach as suggested in (Blumensath and Davies, 2010). Where
we initially try \( \mu < 1 \), which is used as long as the cluster assignment does not change, but if it leads to a changing cluster assignment, we instead use \( \mu = 0.25 \).

Under this condition on \( \mu \), \( \bar{S}_{n+1/2} \) satisfies
\[
\| X - D_{n+1/2} \bar{S}_{n+1} \|_F^2 = \| X - D_{n+1/2} \bar{S}_{n+1/2} \|_F^2 \\
\leq \| X - D_n \bar{S}_{n+1/2} \|_F^2 \\
\leq \| X - D_n \bar{S}_{n+1/2} \|_F^2 + \frac{1}{\mu_n} \| \bar{S}_n - \bar{S}_{n+1/2} \|_F^2 - \| D_n (\bar{S}_n - \bar{S}_{n+1/2}) \|_F^2 \\
\leq \| X - D_n \bar{S}_n \|_F^2 \\
\] (41)
so that
\[
\frac{1}{\mu_n} \| \bar{S}_n - \bar{S}_{n+1/2} \|_F^2 - \| D_n (\bar{S}_n - \bar{S}_{n+1/2}) \|_F^2 \\
\leq \| X - D_n \bar{S}_n \|_F^2 - \| X - D_{n+1} \bar{S}_{n+1} \|_F^2. \\
\] (42)
where we used the minimality of \( D_{n+1/2} \) and the fact that \( D_{n+1} \bar{S}_{n+1} = D_{n+1/2} \bar{S}_{n+1/2} \).

This shows that
\[
\| X - D_{n+1} \bar{S}_{n+1} \|_F \leq \| X - D_n \bar{S}_n \|_F. \\
\] (43)
Thus, the sequence \( X - D_{n+1} \bar{S}_{n+1} \) is bounded and thus by the Bolzsan-Weierstrass theorem will have a convergent subsequence. Note that boundedness holds also if we re-initialise empty clusters in step b) of the algorithm, as long as we do this as discussed above. Because \( X \) is fixed, boundedness of \( X - D_{n+1} \bar{S}_{n+1} \) also implies boundedness of \( D_{n+1} \bar{S}_{n+1} \), i.e.
\[
\| X \|_F + M \geq \| X \|_F + \| X - D_{n+1} \bar{S}_{n+1} \|_F \\
\geq \| D_{n+1} \bar{S}_{n+1} \|_F \\
= \| D_{n+1/2} \bar{S}_{n+1/2} \|_F \\
= \| X \bar{S}_{n+1/2}^T (\bar{S}_{n+1/2} \bar{S}_{n+1/2}^T)^{-1} \bar{S}_{n+1/2} \|_F. \\
\]
Note that the last line also implies boundedness, as \( \bar{S}_{n+1/2}^T (\bar{S}_{n+1/2} \bar{S}_{n+1/2}^T)^{-1} \bar{S}_{n+1/2} \) is a projection operator projecting the rows of \( X \) (the inverse always exists by construction). Thus
\[
\| D_{n+1/2} \bar{S}_{n+1/2} \|_F \leq \| X \|_F \\
\] (44)
and
\[
\| X - D_{n+1} \bar{S}_{n+1} \|_F \\
= \| X (I - \bar{S}_{n+1/2}^T (\bar{S}_{n+1/2} \bar{S}_{n+1/2}^T)^{-1} \bar{S}_{n+1/2}) \|_F \\
\leq \| X \|_F. \\
\] (45)
Thus, using the Bolzsan-Weierstrass theorem, we have proven the following lemma.

**Lemma 5** There exist an \( X^* \) such that for all \( \epsilon \), we can choose an \( N_\epsilon < \infty \) such that
\[
\| X^* - D_n \bar{S}_n \|_F \leq \epsilon, \\
\] (46)
hold for infinitely many \( n_i > N_\epsilon \).
Assume the accumulation point $X^*$ in the above lemma is unique, that is, for all $\epsilon$ in the above lemma, let $n_j$ be the indices such that $\|X^* - D_{n_j} \tilde{S}_{n_j}\| \geq \epsilon$. If the set $n_j$ is finite, then there will be a maximal $n_j$ and we can choose $N_\epsilon > n_j$ and find that for all $n > N_\epsilon$ $\|X^* - D_n \tilde{S}_n\| \leq \epsilon$. This implies convergence of $D_n \tilde{S}_n$ to $X^*$. Thus, either $D_n \tilde{S}_n$ converges or there are at least two accumulation points.

We can also establish the following lemma.

**Lemma 6** Assume that $\mu_n$ is chosen such that

$$
\frac{1}{\mu_n} \|S_n - \tilde{S}_{n+1/2}\|^2 - \|D_n(S_n - \tilde{S}_{n+1/2})\|^2 > \frac{1}{c} \|S_n - \tilde{S}_{n+1/2}\|^2
$$

(47)

for some positive constant $c$. The matrix factorisation algorithm then produces a sequence of estimates $\tilde{S}_n$ that satisfy:

$$
\|\tilde{S}_{n+1/2} - \tilde{S}_n\|^2 \to 0. \quad (48)
$$

Furthermore, the sum

$$
\sum_{n=1}^{N} \|\tilde{S}_{n+1/2} - \tilde{S}_n\|^2
$$

(49)

converges and thus, by the Cauchy Convergence Criterion, so do the partial sums

$$
\sum_{n=N}^{N+p} \|\tilde{S}_{n+1/2} - \tilde{S}_n\|^2,
$$

(50)

where $p \geq 1$ is arbitrary.

**Proof** Convergence follows from the fact that the series $\sum_{n=1}^{N} \|\tilde{S}_{n+1/2} - \tilde{S}_n\|^2$ is monotonically increasing and bounded. Monotonicity is obvious, to show boundedness, write

$$
\sum_{n=1}^{N} \|\tilde{S}_{n+1/2} - \tilde{S}_n\|^2
\leq \ c \sum_{n=1}^{N} \frac{1}{\mu_n} \|\tilde{S}_{n+1/2} - \tilde{S}_n\|^2 - \|D_n(\tilde{S}_{n+1/2} - \tilde{S}_n)\|^2
\leq \ c \sum_{n=1}^{N} (\|X - D_n \tilde{S}_n\|^2 - \|X - D_n \tilde{S}_{n+1}\|^2)
\leq \ c \sum_{n=1}^{N} (\|X - D_n \tilde{S}_n\|^2 - \|X - D_{n+1} \tilde{S}_{n+1}\|^2)
\leq \ c (\|X - D_1 \tilde{S}_1\|^2 - \|X - D_{N+1} \tilde{S}_{N+1}\|^2)
\leq \ c \|X - D_1 \tilde{S}_1\|^2
$$

(51)

where the first inequality is due to the choice of $\mu_n$, the second inequality is (41) and where the third inequality is due to the optimality of $D_{n+1}$ (i.e. $\|X - D_{n+1} \tilde{S}_{n+1}\|^2 \leq \|X - D_n \tilde{S}_{n+1}\|^2$).

\[ \Box \]
Lemma 7 Assume that $\mu_n$ is chosen such that
\[ \frac{1}{\mu_n} \| S_n - \bar{S}_{n+1/2} \|^2 - \| D_n (S_n - \bar{S}_{n+1/2}) \|^2 > \frac{1}{c} \| S_n - \bar{S}_{n+1/2} \|^2 \]  
(52)
for some positive constant $c$. Assume there are no empty clusters in $\bar{S}_n$. The matrix factorisation algorithm then produces a sequence of estimates $D_n S_n$ that satisfy:
\[ \| D_{n+1} \bar{S}_{n+1} - D_n S_n \|^2 \to 0. \]  
(53)

Proof
Note that $\| \bar{S}_{n+1/2} - \bar{S}_n \|^2 \to 0$ implies that $\| \bar{S}_{n+1/2} S_{n+1/2}^{-1} \| - 1 \bar{S}_{n+1/2} - \bar{S}_n (\bar{S}_n S_n)^{-1} \| S_n \|^2 \to 0$, (Stewart, 1977), which in turn implies that $\| D_{n+1} \bar{S}_{n+1} - D_n S_n \|^2 \to 0$ (Remember, $D_{n+1} \bar{S}_{n+1} = D_{n+1/2} \bar{S}_{n+1/2} = X \bar{S}_n^{T} (S_n+1/2 \bar{S}_{n+1/2})^{-1} \bar{S}_n+1/2$).

5.6 Convergence
To proof convergence of the algorithm (that is, the existence of an $S^*$ such that $\| S^* - \bar{S}_n \| \to 0$, we distinguish three cases.

- After some $n$, cluster assignment does not change.
- Cluster assignment changes infinitely often due to changes in sparsity pattern in $\bar{S}_n$, that is, there is an infinite sequence of $\bar{S}_n$, such that $\bar{S}_n$ and $\bar{S}_{n+1/2}$ have different cluster assignments.
- Cluster assignment changes infinitely often due to empty clusters appearing after thresholding.

We now proof convergence for each of these cases independently.

5.7 Case 1: Convergence for fixed cluster assignment
Assume that cluster assignment does no longer change after some iteration. In this case, we can treat the algorithm for each cluster independently. For reference, the following theorem uses the following notation: Let $X_i$ be the sub matrix of $X$ with the features in cluster $i$ and let $X_i = U_i \Sigma_i V_i^T$ be the SVD of $X_i$. Let $\sigma_j$ be the diagonal elements of $\Sigma_i$ (that is, the singular values), which we will assumed are ordered $\sigma_j \geq \sigma_k$ whenever $j < k$. For each cluster, the algorithm produces a cluster centre (column) vector $d_i$ and cluster weight (row!) vectors $s_i$ such that $X_i \approx d_i s_i$, where $d_i \propto X_i \Sigma_i \bar{s}_i / (\Sigma_i \bar{s}_i^T)$. In the svd basis, this can be expressed as $s_i = \alpha V_i^T$ such that $d_i \approx U_i \Sigma_i \alpha^T / (\alpha^T \alpha)$ and $X_i \approx U_i \Sigma_i \alpha^T / (\alpha \Sigma_i \alpha^T) \alpha^T \alpha V_i^T$. That is, $\alpha$ is the representation of the cluster weights $s_i$ in the right singular vector basis $V_i$. Also, let $\alpha_j$ be the $j^{th}$ element of $\alpha$. We then have the following important result

Theorem 8 Assume there is an iteration $N$, such that the cluster assignment stays fixed for all iterations $n > N$. Assume that at iteration $N+1$ the vectors $\alpha^{N+1}$ are such that the element $\alpha_{i_{\text{max}}}^{N+1} \neq 0$. Let $I$ be the index set of the largest singular values, that is, $\sigma_{i_{\text{max}}} > \sigma_j$.
whenever $i_{\max} \in I$ and $j \notin I$. The algorithm then converges to a representation with $\sum_{i \in I} \alpha_i = 0$ and $\alpha_j = 0$ for all $j \notin I$. In other words, if the algorithm reaches an iteration after which cluster assignment no longer changes, and if at that iteration, the cluster weight vector $s_i$ is not orthogonal to the subspace spanned by the right singular vectors of the feature matrix $X_i$ associated with the largest singular values, then the weight vector $s_i$ will converge to a vector that lies in this subspace. In particular, if the largest singular value is unique, then the algorithm converges to a vector collinear to the associated singular vector with $\alpha_{i_{\max}} = \sigma_{i_{\max}}$.

**Proof** Let us first recall that, due to normalisation of $d_i$ and $s_i$, we have $\frac{\|\alpha \Sigma\|}{\|\alpha\|^2} = 1$. Thus, the update of the $k^{th}$ element in vector $\alpha$

$$
\alpha_k^{n+1} = \left(1 - \mu \frac{\|\alpha \Sigma\|}{\|\alpha\|^2} + \mu \frac{\sigma_k^2}{\|\alpha^n \Sigma\|}\right) \alpha_k^n
$$

simplifies to

$$
\alpha_k^{n+1} = \left(1 - \mu + \mu \frac{\sigma_k^2}{\|\alpha^n \Sigma\|}\right) \alpha_k^n
$$

Without loss of generality assume that $\alpha_k^n > 0$ (Note that the update does not change the sign of $\alpha_k$ so we can repeat the same argument for negative $\alpha_k$. Note however (see also below) that $\alpha_k^n = 0$ is not allowed as $\alpha_k$ will then remain constant.). Let us use the shorthand $c_k = \left(1 - \mu + \mu \frac{\sigma_k^2}{\|\alpha^n \Sigma\|}\right)$. Note that $0 < \mu \leq 1$ implies that $c_k$ is positive. Looking at the normalised update we then have

$$
\frac{(\alpha_k^{n+1})^2}{\|\alpha_k^{n+1}\|^2} = \frac{c_k^2}{\sum_i c_i^n (\alpha_i^n)^2} (\alpha_k^n)^2,
$$

which can be rewritten as

$$
\frac{(\alpha_k^{n+1})^2}{\|\alpha_k^{n+1}\|^2} = \frac{c_k^2}{\sum_i \lambda_i c_i^n (\alpha^n)^2} (\alpha_k^n)^2,
$$

where $\lambda_i = (\alpha_i^n)^2/\|\alpha^n\|^2$, so that $\sum_i \lambda_i c_i^2$ is a convex combination of the positive values $c_i^2$ (i.e. $\sum_i \lambda_i = 1$, $\lambda_i \geq 0$). We have thus shown that, for all $i$ for which $c_i^2 > \sum_i \lambda_i c_i^2$, the normalised $\alpha_i$ increase (i.e. $\frac{c_i^2}{\sum_i \lambda_i c_i^2} > 1$), whilst for those $c_i^2 < \sum_i \lambda_i c_i^2$, we have a relative decrease. Furthermore, if all $\alpha_i \neq 0$, then the largest relative increase is for the $\alpha_i$ associated with the largest singular values (as for those elements $c_i^2$ is maximal and as the maximum value of a set of positive numbers must be larger than any convex combination of the elements).

If we write $c_i^n = \frac{c_i^2}{\sum_i \lambda_i c_i^2}$, then we have the recursion

$$
\frac{(\alpha_k^{n+1})^2}{\|\alpha_k^{n+1}\|^2} = c_i^n \frac{(\alpha_k^n)^2}{\|\alpha^n\|^2} = \prod_{N=0}^n c_k^N \frac{(\alpha_0^N)^2}{\|\alpha^N\|^2},
$$

Assume the singular values $\sigma_i$ are ordered such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_M$. This implies the same ordering on the $c_i^n$, i.e. $c_1^n \geq c_2^n \geq \cdots \geq c_M^n$ for all $n$.  

---

**Directional Clustering via Matrix Factorisation**

[Note: The rest of the text is not necessary to understand the proof and is included for completeness.]
Note that \(\frac{(\alpha^k_n)^2}{|\alpha^k_n|^2} \leq 1\) and thus, the sequences \(\frac{(\alpha^k_n)^2}{|\alpha^k_n|^2}\) are bounded. Furthermore, for those \(k\) associated with the largest singular values, the sequence is increasing, as for those \(k\) \(\tilde{c}^k_n \geq 1\). This implies that for those \(k\), the sequence \(\frac{(\alpha^k_n)^2}{|\alpha^k_n|^2}\) converges.

If \(\alpha^k_n \neq 0\), then for \(\mu < 1\) \(\tilde{c}^k_n \neq 0\). Alternatively, for \(\mu = 1\), if there are singular values that are zero (i.e. \(\sigma_m = 0\)), then \(\alpha^m_n = 0\) for all \(n > 1\). In this case, we apply the following argument only to those \(\alpha_i\) for which \(\sigma_i \neq 0\). Thus, without loss of generality assume that \(\alpha^k_n \neq 0\) and that \(\tilde{c}^n_i \neq 0\). In this case, for all \(i\) for which \(\sigma_i\) is maximal, \(1 \geq \frac{(\alpha^n_i)^2}{|\alpha^n_i|^2} \geq \frac{(\alpha^n_k)^2}{|\alpha^n_k|^2}\) for all \(n\). Convergence of \(\frac{(\alpha^n_i)^2}{|\alpha^n_i|^2}\) then also implies that the sequences \(\tilde{c}^n_i\) converge to 1. (Because \(\lim_{n \to \infty} \frac{(\alpha^n_i)^2}{|\alpha^n_i|^2} = \lim_{n \to \infty} \tilde{c}^n_i (\frac{(\alpha^n_i)^2}{|\alpha^n_i|^2}) = (\lim_{n \to \infty} \tilde{c}^n_i) (\lim_{n \to \infty} \frac{(\alpha^n_i)^2}{|\alpha^n_i|^2})\).

Thus in the limit,

\[
\tilde{c}^n_i = \frac{(c^n_i)^2}{\sum_i \lambda^n_i (c^n_i)^2} \to 1
\]

However, as \(\sum_i \lambda^n_i = 1\), we also have the requirement that

\[
\tilde{c}^n_i = \frac{(c^n_i)^2}{\sum_i \lambda^n_i (c^n_i)^2} = \frac{1}{\sum_i \lambda^n_i (c^n_i)^2} > \frac{1}{\sum_i \lambda^n_i} = 1
\]

unless \(\lambda^n_i = 0\) for all \(c^n_i < c^n_1\). Thus convergence of \(\tilde{c}^n_i\) to zero implies convergence of \(\lambda^n_i\) to zero for all \(i\) other than those \(i\) associated with the largest singular values. But this implies that \(\lambda^n_i = \frac{(\alpha^n_i)^2}{|\alpha^n_i|^2} \to 0\) for those \(i\) which in turn implies that \(\sum_k (\alpha^k_n)^2/|\alpha^n|^2 \to 1\), where we sum over those \(k\) associated with the largest singular values.

\[\square\]

### 5.8 Case 2: infinite changes in sparsity pattern

By Lemma 6 there is an \(N\) such that \(\|\tilde{S}_{n_i} - \tilde{S}_{n_i+1/2}\| \leq \epsilon\) for all \(\epsilon > 0\). Let \(n_i > N\) be an infinite sequence of indices such that \(\tilde{S}_{n_i}\) has a different support to \(\tilde{S}_{n_i+1/2}\). Let \(I\) be the set of indices of columns in \(S\) that have elements that change infinitely often from zero to a non-zero value and vice versa. As the difference \(\|\tilde{S}_{n_i} - \tilde{S}_{n_i+1/2}\|_F \to 0\), this implies that \(\|S_I\|_F \to 0\), where \(S_I\) is the sub matrix made of columns of \(S\) indexed by \(I\). Thus, \(S_I \to 0\), whilst the columns in \(\tilde{S}\) not indexed by \(I\), say \(\tilde{S}_{I^c}\) will converge to right singular vectors of the feature vector matrix using arguments that mirror those described above.

### 5.9 Case 3: infinitely many empty clusters

Assume there is an infinite sequence of \(\tilde{S}_{n_i}\) for which \(\tilde{S}_{n_i}\) has empty clusters. We know that after empty cluster re-initialisation and normalisation,

\[
\|X - D_{n_i+1/2} \tilde{S}_{n_i+1}\|_F \\
\leq \|X(I - \tilde{S}_{n_i+1/2}^T \tilde{S}_{n_i+1/2} \tilde{S}_{n_i+1/2}^T)^{-1} \tilde{S}_{n_i+1/2}\|_F \\
\leq \|X\|_F
\]

(61)

(where we set the inverse of the zero element in \(\tilde{S}_{n_i+1/2}^T \tilde{S}_{n_i+1/2}\) to zero) so that there is an infinite sequence \(\|X - D_{n_i+1} \tilde{S}_{n_i+1}\|_F\) that is bounded. The Bolzano-Weierstrass theorem
then implies the existence of an infinite convergent subsequence.

\[ \|X^\star - D_n\bar{S}_n\|_F \rightarrow 0, \]  

(62)

where the \(\bar{S}_{n_i-1}\) have empty clusters.

As our arguments are independent of exactly which of the columns in \(X\) we use to re-initialise the empty cluster (as long as we don’t choose one from a cluster with a single element), we assume that w.l.g. we take that element for which \(\|x_i - d_is_j\|\) is maximal. But the fact that \(\|X - D_n\bar{S}_n\|_F\) converges for all \(n\) then implies that \(\|x_i - d_is_j\| \rightarrow 0\), that is, \(\|X - D_n\bar{S}_n\|_F \rightarrow 0\).

5.10 Convergence theorem

We have thus proven the following theorem.

**Theorem 9** The algorithm produces a sequence of \(D_n\bar{S}_n\), such that either \(\|X - D_n\bar{S}_n\|_F \rightarrow 0\) or such that \(\bar{S}_n \rightarrow \bar{S}^\star\), where the nonzero elements in row \(i\) converge to an element in the space spanned by the right singular vectors associated with the largest singular values of the feature vector matrix \(X_i\) containing those columns in \(X\) for which the \(i^{th}\) row in \(\bar{S}^\star\) has non-zero entries.

In other words, the algorithm either finds \(K\) vectors \(d_i\) such that each feature \(x_i\) is a multiple of one \(d_i\) or it partitions the features into distinct clusters such that the feature vectors of each cluster are modelled with left and right eigenvectors associated to the largest eigenvalue of the feature sub matrix.

6. Numerical Results

The performance of our new approach is here evaluated on several problem sets and contrasted to alternative approaches. This is done using artificial data as well as real data-sets. There is no universally agreed performance metric to evaluate clustering performance. We thus evaluated our results using a selection of popular metrics, maximum cluster overlap (as in (Ding et al., 2010)), Dice similarity (DICE) (Dice, 1945), Normalised Mutual Information (NMI) (Strehl and Ghosh, 2002) and Adjusted Rand Index (RI) (Hubert and Giraud, 1985).

Cluster overlap and dice similarity are measures that measure the similarity between two individual clusters. If \(A\) is the set of features assigned to one cluster and \(B\) the set of feature assigned to the other cluster, then the cluster overlap between \(A\) and \(B\) is

\[ CO(A, B) = \frac{|A \cap B|}{N}, \]  

(63)

where \(N\) is the number of features\(^2\). Dice’s similarity measure is a related normalised measure

\[ DICE(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \]  

(64)

Both of these measures only measure similarity between two clusters but not directly between two clusterings. To compute a measure that can compare entire clusterings, we

\(^2\) The notation \(|A|\) refers to the number of elements in set \(A\).
calculate the similarity between any pair of clusters taken from the two clusterings. This provides a similarity matrix. We then permute this matrix greedily, so that each entry along the diagonal is no smaller than any other entry in the sub-matrix formed from the elements that are below and to the right of that diagonal element. A measure for the similarity between the two clusters is then the average over the matrix diagonal.

Normalised Mutual Information (NMI) and the Adjusted Rand Index (RI) can be used to compare clusterings directly. NMI is computed as follows. Let $C_1$ be a partitioning of a set of $N$ features into $k_1$ distinct clusters and $C_2$ a partitioning of the same features into $k_2$ clusters. Let $n^1_i$ be the number of features in cluster $i$ in clustering 1 and $n^2_j$ the number of features in cluster $j$ in clustering 2. Similarly, let $n_{i,j}$ be the number of features that are both, in cluster $i$ in partition 1 and in cluster $j$ in partition 2. The NMI is then

$$NMI(C_1, C_2) = \frac{\sum_{i=1}^{k_1} \sum_{j=1}^{k_2} n_{i,j} \log \left( \frac{N n_{i,j}}{n^1_i n^2_j} \right)}{\sqrt{\left( \sum_{i=1}^{k_1} n^1_i \log \frac{n^1_i}{N} \right) \left( \sum_{j=1}^{k_2} n^2_j \log \frac{n^2_j}{N} \right)}}$$

RI is computed as follows. Using the same notation as before, let $t_1 = \sum_{i=1}^{k_1} 0.5n^1_i(n^1_i - 1)$ and $t_2 = \sum_{j=1}^{k_2} 0.5n^2_j(n^2_j - 1)$ and $t_3 = 2t_1t_2/(N(N-1))$.

$$RI(C_1, C_2) = \frac{\sum_{i=1}^{k_1} \sum_{j=1}^{k_2} 0.5n_{i,j}(n_{i,j} - 1) - t_3}{0.5 \ast (t_1 + t_2) - t_3}.$$  

We start with experiments on artificial data, evaluating the performance when fundamental properties of the data, (such as overall noise, relative size of different clusters and relative cluster noise) change. One of our main motivations is the analysis of functional brain imaging data and the next two sets of experiments are designed to shed light on the performance of the method specifically in this context. We start with an artificially generated set of data that has some properties of real functional brain imaging data. We then demonstrate how the method can be used on a real data set. A fundamental limitation of the evaluation of the technique on real brain imagine data is that there is no ground truth available and we are thus left to analyse the performance in terms of consistency of the method when different data-sets are clusters. To demonstrate the applicability of the method more widely, we conclude this section with several clustering experiments used previously, including two on text data. On all of these problems, we compare our proposed algorithm to several competitors, including spherical k-means and semi-NMF.

6.1 Comparison of Different Versions of our Approach using a Synthetic Data Set

The synthetic data sets in the first set of test problems were generated by randomly generating matrices $D^*$ and binary $S^*$ from which the observations were constructed as $X = D^*S^* + E$, where $E$ is an i.i.d. Gaussian noise term. $D$ was a 1000 by 10 matrix (i.e. we generated 10 cluster centres) and $S$ was of dimension 10 by 100 (that is, we generated 100 observations).
We varied the standard deviation of $E$ from 0.01, 0.1, 1 and 10 and contrasted two different regimes, one, in which the average number of features in each cluster were identical and one in which one cluster had 91 features and all other clusters had a single feature.

We compared our method with several variations and averaged the results over 1000 random problem instances. The results are shown in Tables 1 and 2, where, for each noise level, we have highlighted the best performing algorithm version in bold. Results are here reported in terms of NMI, as the other measures gave qualitatively similar results (see also the comparison of performance metric in the next subsection).

Table 1: Performance of variations of our method with equally sized clusters in terms of NMI.

<table>
<thead>
<tr>
<th>Update of $S$:</th>
<th>$S + \mu D^T(X - DS)$</th>
<th>$(D^T D)^{-1}D^TX$</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalisation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>std</td>
<td>0.01, 0.1, 1, 2</td>
<td>0.01, 0.1, 1, 2</td>
</tr>
<tr>
<td>$D$</td>
<td>0.951, 0.939, 0.841, 0.564</td>
<td>0.984, 0.965, 0.861, 0.577</td>
</tr>
<tr>
<td>$S$</td>
<td>0.948, 0.931, 0.841, <strong>0.587</strong></td>
<td><strong>0.989, 0.976, 0.872, 0.584</strong></td>
</tr>
<tr>
<td>NONE</td>
<td>0.951, 0.936, 0.842, 0.571</td>
<td>0.982, 0.967, 0.866, 0.584</td>
</tr>
</tbody>
</table>

Table 2: Performance of variations of our method with widely varying cluster sizes in terms of NMI.

<table>
<thead>
<tr>
<th>Update of $S$:</th>
<th>$S + \mu D^T(X - DS)$</th>
<th>$(D^T D)^{-1}D^TX$</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalisation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>std</td>
<td>0.01, 0.1, 1, 2</td>
<td>0.01, 0.1, 1, 2</td>
</tr>
<tr>
<td>$D$</td>
<td>0.445, 0.388, 0.329, 0.270</td>
<td><strong>0.876, 0.730, 0.408, 0.260</strong></td>
</tr>
<tr>
<td>$S$</td>
<td>0.491, 0.367, 0.325, 0.266</td>
<td>0.834, 0.665, 0.308, 0.252</td>
</tr>
<tr>
<td>NONE</td>
<td>0.616, 0.378, 0.334, <strong>0.273</strong></td>
<td>0.57, 0.679, 0.343, 0.256</td>
</tr>
</tbody>
</table>

From these results we see that, apart from the condition with very high noise, an update of $S$ based on the pseudo-inverse of $D$ is advantageous. If cluster size is roughly equal between clusters, then a pre-thresholding normalisation of the rows of $S$ seems to perform better, whilst for clusters of varying size, normalisation of columns of $D$ works best. Interestingly, if we use the gradient type update $S + \mu D^T(X - DS)$, then an algorithm without column normalisation seems to be the best choice in both conditions.

6.2 Comparison of Different Algorithms on Synthetic Data Sets

The synthetic data sets were generated again by randomly generating matrices $D^*$ and binary $S^*$ and i.i.d. Gaussian noise $E$.

Three different datasets were generated:

1. **Dataset 1**: $D \in \mathbb{R}^{M \times K}$ was generated with i.i.d Gaussian zero-mean unit-variance entries. $S \in \mathbb{R}^{K \times N}$ was generated with each column set to zero apart from one entry
whose location was chosen at random and whose value was set to 1. For this data-set, all clusters had thus roughly the same number of observations per cluster.

2. **Dataset 2:** This was generated in the same way as dataset 1, with the exception that \( S \) was generated deterministically so that each cluster had different numbers of observations \( x_i \). We here used an extreme example, where there were 3 clusters with only 1 observation, 2 clusters with 3 observations, and 1 cluster each with 6, 10, 14, 24 and 36 observations respectively.

3. **Dataset 3:** This was generated in the same way as dataset 2, with the exception that the cluster centres in \( D \) where each scaled by a zero-mean, unit-variance Gaussian. Thus each cluster did have a different level of noise compared to the size of the cluster centre (or, after normalisation of each \( x_i \) each cluster had a different amount of within cluster variance).

To each of these datasets, four different levels of noise were added with the entries in \( E \) having a variance of 0 (no noise, i.e. the \( x_i \) are cluster centres), a variance of 1, a variance of 4 and a variance of 9 (See figures (1) to (3) for average SNR values for each condition). Noise was added before normalisation of the observations and results are averaged over 100 different realisations of each datasets and noise condition.

The results for the three datasets are shown in Figures 1, 2 and 3, where we compare our method with semi-NMF and standard as well as spherical k-means.

The figures are partitioned into four rows, one for each noise level, and four columns, one for each algorithm. Figure 1 shows the performance in terms of all four different performance measures. As all of these provide qualitatively similar results and to increase clarity of the figures, the other figures only display Normalised Mutual Information. The SNR values next to each row are empirical estimates for the level of noise added.

It is clear that for the experiments reported here, our approach outperforms all other reproaches over all datasets and noise conditions. Other key observations are

1. The four performance measures we have used in Figure 1 are qualitatively similar.
2. The semi-NMF algorithm sometimes performs better than k-means and sometimes it performs worse.
3. Spherical k-means performs better than non-spherical k-means run on normalised vectors.
4. Unsurprisingly, increasing the noise clearly reduces performance.
5. There is a clear performance decrease when going from dataset 1 to dataset 2, though going from dataset 2 to dataset 3 only reduces performance slightly.

The difference between the standard k-means (Euclidean distance) and the spherical k-means is small. We also tried two EM algorithms, one based on a Gaussian mixture model and one based on a von Mises-Fisher Mixture model, but as these methods performed poorly\(^3\), we do not show the results here.

---

\(^3\) This was mainly due to the methods difficulty in estimating within-cluster variance, a problem that could potentially be overcome with a full Bayesian model, though this would further increase the computational burden for this rather slow approach.
Figure 1: Performance of our algorithm, semi-NMF clustering and two k-means variants (spherical and standard) for artificial dataset 1. Performance is measured with NMF, RI, DICE and cluster overlap and for 4 noise levels.

Figure 2: Performance of our algorithm, semi-NMF clustering and two k-means variants (spherical and standard) for artificial dataset 2. Performance is shown in terms of NMF only (RI, DICE and cluster overlap are qualitatively similar) and for 4 noise levels.
Figure 3: Performance of our algorithm, semi-NMF clustering and two k-means variants (spherical and standard) for artificial dataset 3. Performance is shown in terms of NMF only (RI, DICE and cluster overlap are qualitatively similar) and for 4 noise levels.

6.3 Synthetic functional Brain Data

We developed the approach for a specific problem in brain imaging and the next artificial data sets simulate this. We are interested in the clustering of a spatial data-set, where each spatial location has an associated time-series (the feature vector). The aim is then to cluster the time-series or features to recover the spatial clusters. To simulate such a data-set, we generated a spatial grid \((64 \times 64)\) and split this grid into 40 spatially connected regions. This was done by randomly selecting 40 cluster seed locations on the grid. The seeds are then grown by adding one randomly chosen spatial neighbourhood point to one randomly chosen cluster. This is repeated until the entire spatial grid is covered. An example can be seen in the top left of Figure 4. Whilst these clusters have clear spatial structure, this was not used in the clustering itself, where features were grouped based on the similarity of their time-series (or feature vector).

For each cluster, these feature vectors were drawn from different distributions. We thus generated three different datasets.

1. Features within each cluster were generated from an i.i.d. Gaussian distribution, with a mean that was itself drawn from an i.i.d. zero-mean, unit variance Gaussian. The within cluster variance was varied between 1 and 3, producing SNR values of 0dB, -3 dB and -9dB respectively. This is intended as a very rough simulation of a functional Magnetic Resonance Imaging dataset (see below).

2. The data was generated as in 1) above, but additional spatial smoothing was applied to simulate spatial correlation between features as observed in real brain imaging data. Smoothing was achieved by averaging spatially close feature vectors using a Gaussian smoothing kernel. The amount of smoothing varied within each data-set and the Gaussian kernel had a standard deviation that varied from 0.2 to 5 pixels.
3. Cluster centres were generated from a Beta distribution with both parameters set to 2. For each of the clusters, observations were then drawn from a Beta distribution whose parameters were calculated such that the distribution had a variance of 0.2 and a mean equal to the cluster’s mean. Each observation $x_i$ thus had entries between 0 and 1. This data is a rough approximation simulating brain connectivity data as estimated using diffusion Magnetic Resonance Imaging techniques (see below).

We again evaluate the cluster assignment using Normalised Mutual Information (the other measures again show similar differences between approaches). The results for the three different datasets are shown in figures 5, 6 and 7, with a visual representation of the spatial clusters and their estimates for the Beta distributed feature vectors shown in figure 4. The different methods are arranged horizontally with small random horizontal perturbations added for better visualisation. Figure 5 is additionally split into three columns, one for each level of within cluster variance.

We here compared our approach to semi-NMF, spherical k-means, standard k-means with normalised features and an EM algorithm based on a von Mises-Fisher (vMF) mixture model. In general, our approach outperforms the other approaches, especially for moderate to low noise. Only the -9dB SNR condition does not point to a clear winner, with the two k-means algorithms performing similar to our approach. The directional k-means again performed slightly better in general than its standard counterpart. The EM algorithm does again not perform well, the reason being again the indeterminacy in the variances. However, the von Miese Fisher algorithm seems to perform slightly better for the Beta distributed features.

We also run our method on the same datasets using a recursive scheme in which we changed the number of clusters to optimise the Akaike information Criterion (AIC). This method was able to correctly estimate the number of clusters ($\pm 2$) and AIC optimal clusters were found to have a NMI similarity to the original clusters comparable to those observed when specifying the correct number of clusters.

6.4 Application to brain parcellation

Our third experiment evaluates our new clustering method on actual brain imaging data. Neuroscientists are interested in a detailed understanding of connections in the human brain and modern Magnetic Resonance Imaging (MRI) techniques offer two complementary methods to study these brain connections (Biswal et al., 1995; Klein et al., 2007). Diffusion MRI (Klein et al., 2007) methods allow estimates of major fibre bundles to be computed and, by tracking individual fibres, the connection between distant brain parts can be studied (so called structural connectivity). An alternative view of brain connectivity is offered by functional MRI studies. For example, by measuring blood oxygenation changes in the brain during rest, statistical relationships between the activation of different brain regions can be estimated (Biswal et al., 1995). If brain activation in distinct regions shows statistical dependancy, then these regions must exchange information and must therefore be connected in some way (so called functional connectivity). MRI studies often measure brain properties on three dimensional spatial grids of 2 to 4 millimetres and, for the average human brain, this leads to very high dimensional problems, where the connection between hundreds of thousands of brain areas has to be estimated. This cannot be done reliably and a funda-
Our approach
semi−NMF
spherical k−means
k−means
vMF EM

Figure 4: Example of spatial distribution of feature vectors (top left) and estimates calculated with different methods. vMF: EM algorithm based on von Mises-Fisher Mixture Model.

Figure 5: Normalised Mutual Information between original and estimated spatial cluster assignments (using Gaussian distributed feature vectors with 3 different levels of variance). vMF EM: von Mises-Fisher Mixture Model.

mental first step is the decomposition of the brain into a smaller set of brain areas. Whilst regions can be defined based on neural anatomy found in post-mortem studies, or through the agglomeration of large brain imaging studies that use specific cognitive tasks to study a specific brain region, there are many reasons (such as the large variability in functional brain anatomy between people) why these partitions are not ideal substrates on which to base connectivity analysis. There is thus now an extensive literature on the development
Figure 6: Normalised Mutual Information between original and estimated spatial cluster assignments (using Gaussian distributed feature vectors and spatial Gaussian smoothing with spatially varying variance and additional noise leading to an SNR of 0). vMF EM: von Mises-Fisher Mixture Model.

Figure 7: Normalised Mutual Information between original and estimated spatial cluster assignments (using Beta distributed feature vectors). GMM - Gaussian Mixture Model; vMF EM: von Mises-Fisher Mixture Model.

of algorithms to partition the human brain based on functional MRI data acquired during rest (Mumford et al., 2010; Craddock et al., 2012; Power et al., 2011; Zhang et al., 2011; Lashkari et al., 2010; Bellec et al., 2010; Shen et al., 2010; Yeo et al., 2011; Blumensath et al., 2013). We here test our algorithm on the same problem.

We used fMRI and structural MRI data from 66 subjects, collected during the initial stages of phase 2 of the human connectome project (http://humanconnectome.org/). The data had 2mm isotropic spatial resolution and a temporal resolution of 1.4 seconds. The data was processed using a preliminary version of the Human Connectome Project’s structural and functional minimal preprocessing pipelines, final versions to be published.
separately (Glasser et al. unpublished). Briefly, this involved brain extraction, registration of different MRI modalities, bias field correction, registration to a standard brain template and cortical surface modelling. Functional data were motion corrected, distortion corrected, mean normalized and resampled to the cortical surface. Standard surface smoothing and temporal filtering was applied and ICA based noise reduction used.

For each of the 66 subjects, the dataset consisted of a set of approximately 64000 functional MRI time series, each with approximately 1000 temporal samples each. We split the dataset into two, with 33 subjects each. For each of these splits, we combined the data across subjects by estimating the 1000 left singular vectors of the spatio-temporal data matrix (concatenated in the temporal direction over the 33 subjects). We thus produced two sets of feature vectors, where each vector had a length of 1000 and was associated with one of the vertex locations on the cortical grid representation.

As there is no ground truth available for this experiment, we estimate the performance based on the ability of an algorithm to reliably identify clusters in each of the two split datasets. The results are compared visually in figure 8, where we show an inflated representation of the left and right cortical surface and the estimated clusters from the two datasets (left vs. right). Grey levels for the clusters were matched to ease visual comparison.

A numerical evaluation in terms of the Dice\textsuperscript{4} similarity between the clusters derived form each of the two datasets is shown in Figure 9. The results obtained for different number of clusters and different methods is shown. Before calculating dice similarity, we split all clusters we estimated into spatially contiguous regions and then discarded very small clusters (we here removed clusters that had less than 20 features, though the flavour of the results does not vary much if we use another threshold). Also shown are results for the directional k-means algorithm and a recently developed region growing based method that explicitly enforces clusters to be spatially connected (Blumensath et al., 2013).

We also tried the normalised cuts spectral clustering method of (Shi and Malik, 2000) on this problem. As it is not feasible to calculate and save the entire similarity matrix for all features, we here generated sparse versions by thresholding the correlation at 0.5 and 0.4. However, the results did not compare well to the other methods tested and are thus omitted.

We can see that our method performs much better than the region growing approach and better than the semi-NMF algorithm. To interpret these results, it must be remembered that the region growing algorithm enforces clusters to be spatially connected. This is known to introduce additional biases into the estimated clusters, which in turn generally means that clusters are more repeatable. Our approach does not include such an additional spatial constraint and is thus not affected by the associated bias and is thus a more reliable indicator of intrinsic data-structure.

6.5 Performance on standard data sets

We conclude this section with an analysis of more general data-sets used elsewhere in the clustering literature. In particular, we used the following 3 datasets:

---

4. We used dice similarity here, as this is a common measure used in the field
Directional Clustering via Matrix Factorisation

Figure 8: Repeatability of clustering of the cortical surface based on resting-state fMRI data. Clusters derived from two different groups of 10 subjects each are shown on the left and right on an inflated rendering of the cortical surface. Right hemisphere (top) and left hemisphere (bottom).

• Data set 1 WAVE: This data-set, generated for (Breiman et al., 1984, p. 49-55, 169) can be retrieved from http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+(Version+1)). The data-set consisted of 5000 features each with 21 elements. Features were from 3 different classes and contained gaussian noise.

• Data set 2a,b NEWS: is a text analysis data-set consisting of bag of words feature vectors, generated originally for (Lang, 1995). We used the version of the database in which there are 20 Newsgroups sorted by date (retrieved from http://qwone.com/jason/20Newsgroups/). We used subsets of this data with 500 features of length 53975 and clustered these into the 20 classes. Different subsets were used with version (a) of the dataset generated by randomly taking 25 features from each newsgroup whilst dataset (b) was generated by randomly taking subsets of varying size from each newsgroup (the number of features varied exponentially between 1 and 102).

• Data set 3 MXM: was a subset of the bag of word features generated for (Bertin-Mahieux et al., 2011), (retrieved from http://labrosa.ee.columbia.edu/millionsong/musixmatch). This dataset contains bag of words representations for the lyrics from a music database. We extracted a subset of the BoW features, corresponding to music from 6 different musical genres (techno, rock, pop, punk, country and hip hop). There were 5000 BoW features in this data-set of length 12921. We used these features to see if we could use a blind clustering approach to distinguish the different musical genres based on the lyrics alone.
Our approach semi-NMF k-means region growing

Figure 9: Comparison of four different approaches for clustering of the cortical surface based on resting-state fMRI data. Repeatability measured in terms of average Dice similarity between cluster regions plotted for different numbers of clusters. For each approach, the clusters were derived from two different groups of 10 subjects. Before the calculation of Dice similarity, clusters were split into spatially homogeneous regions and small clusters were removed.

The result of the analysis of the three data-sets are shown in table 3, measured using Normalised Mutual Information and contrasting our approach to semi-NMF and spherical k-means. Whilst overall performance on these data-sets is low (they are difficult data-sets to cluster with simple blind clustering methods), it is evident that our approach outperforms the other approaches.

Table 3: Comparision of different methods on different data sets measured in NMI

<table>
<thead>
<tr>
<th></th>
<th>WAVE</th>
<th>NEWS(a)</th>
<th>NEWS(b)</th>
<th>MXM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our approach</td>
<td>0.3676</td>
<td>0.1864</td>
<td>0.1447</td>
<td>0.0545</td>
</tr>
<tr>
<td>semi-NMF</td>
<td>0.3466</td>
<td>0.1148</td>
<td>0.0947</td>
<td>0.0489</td>
</tr>
<tr>
<td>spherical k-means</td>
<td>0.2801</td>
<td>0.1636</td>
<td>0.1253</td>
<td>0.0507</td>
</tr>
</tbody>
</table>
7. Discussion

Our clustering approach has several advantages over other standard methods. For example, it overcomes limitations of the k-means algorithm, which is not able to account for varying variance within different clusters. Whilst this problem is also overcome in maximum likelihood based approaches, the indeterminacies that exist here are overcome by normalisation in our approach. Whilst this particular problem can also be overcome using Bayesian ideas, our method is computationally more efficient.

Clustering is a combinatorial problem. Especially for large clustering problems, many algorithms, including k-means, seem to struggle to deal efficiently with this challenge. The flexibility in our method offered by the variable scaling (size of entries in $S$) seems to allow the algorithm more flexibility during optimisation. Our method is also computationally relative efficient as it only requires matrix products and inversion of small ($K \times K$) matrices and so is comparable in complexity to k-means type methods but consistently seems to outperform these. Alternative implementations of our approach can even be used without explicit matrix inversion, making the method suitable for settings in which the number of clusters is relatively large.

In comparison to semi-NMF based clustering our matrix factorisation not only tries to minimise a data fidelity term (as is done in semi-NMF), but simultaneously tries to enforce hard cluster assignment. In contrast to k-means, instead of basing this hard assignment purely on the similarity of the current cluster centre estimate and the features $x_i$, we base our decision on the elements in the matrix $S$. These elements are chosen to minimise the error term and this is done using the pseudo inverse of the matrix $D$. This is in stark contrast to the approach taken in spherical k-means, where this selection is instead done using the transpose of $D$ (assuming the columns $d_k$ and features $x_i$ are normalised). The advantage of the use of the pseudo inverse is that now the elements in $S$ also encode information on the similarity between different cluster centres. If there are two cluster centre vectors that are similar, and if both were similar to a feature vector, then k-means would assign the feature to the centre that is nearest. However, if there were a third cluster centre that is orthogonal to the first two but which is highly correlated with the feature, but less so than the first two centres, then k-means would ignore the third centre, whilst in our method, as the first two centres are used jointly to approximate the feature, their individual magnitudes in $S$ can be lower than the magnitude associated with the third feature. Our method might thus choose the third feature. In a sense, our method takes uncertainty into account if different cluster centres are similar, a feature not found in k-means.

8. Conclusions

We have here proposed a simple algorithm that can efficiently cluster feature vectors based on their direction. The approach is based on matrix factorisation ideas and these allowed us to design an algorithm that is applicable to relatively large clustering problems where hundreds of thousands of feature vectors are clustered into hundreds of clusters. Our method was shown to outperform other standard approaches on several toy problems as well as on a clustering problem that arises in human brain imaging. Our method also outperformed competitors on clustering methods found in the literature. There remain several aspect of
the method that require further investigation. Of particular interest are conditions on the
original cluster features that would guarantee the algorithm to cluster the features correctly.
Current work is being undertaken to address this issue.

Acknowledgments

This research was supported through EPSRC grants EP/K037102/1 and EP/J005444/1 as well as a Faculty of Engineering and the Environment funded New Frontiers Fellowship.
The author would like to thank the HCP consortium for the provision of the pre-processed fMRI data.

References


