Applying community detection techniques for clustering predictors in linear models

Social and Economic Networks – Course Project

Miquel Torrens Dinarès
Barcelona Graduate School of Economics

June 27, 2016

Abstract

Performing variable selection for linear models in high dimensional settings is still nowadays an open challenge. The recent hype of Bayesian approaches has shed some light into this issue, but these are still computationally very demanding. Recent research has shown how under a block-orthogonal design matrix this task can be performed efficiently. The problem is that block-orthogonality is a rare condition and is hardly satisfied in practice. In this project, we analyse how we can apply community detection and clustering algorithms to group sets of predictors to help block-diagonalise generic matrices. As a result, we show a few lab examples that suggest that modularity-based methods can recover the number of underlying communities, and that spectral methods perform excellently in posteriorly spotting the clusters. Results also point towards their fitness to be scaled to high dimensions.

Keywords: variable selection, community detection, spectral clustering

Framework

Variable selection in linear models is not a fully solved theoretical problem in statistics. A great amount of literature has been inked on this issue but there exists no universal solution. This is especially the case when the ratio of number of predictors over number of observations grows large, and mandatory when this ratio is greater than one. Bayesian methods for feature selection have surged in recent years thanks to increasing computational power but some settings still remain unfeasible. These methods require to compute unkind integrals and, most of all, explore a model space that grows exponentially in the number of predictors.

In the work of Papaspiliopoulos and Rossell [6] it is shown how, under a block-orthogonal design of the covariance matrix, effective and scalable variable selection can

be performed in reasonable time. However, a missing element is to transform a generic design matrix into this desired form. A general way to block-orthogonalise the covariance matrix as accurately as possible is still required.

In this project we will attempt to use several graph partitioning techniques in order to perform such clustering of predictors that leads to block-orthogonalisation, which is a necessary condition for the algorithm proposed by these authors to work properly. Predictor block-orthogonality allows for mild computations and a huge reduction of the size of the model space. We refer to their work to address the issue of actual model selection, which is well beyond the scope of this course and project.

All the code used for this project can be found in the following repository:

- `https://github.com/mtorrens/community`

Analysis of the performance of different algorithms

In this section we will present the approach taken to cluster predictors of a design matrix under a linear model. Afterwards, we will discuss the algorithms that have been employed on some working examples artificially designed to test them. We refer to the previous report elaborated for this course for discussion on some of the methods. Finally, we will discuss the results in each of the examples.

Methodology

Consider the classic linear model where we want to predict $\mathbf{y} \in \mathbb{R}^n$ by estimating the parameters $\boldsymbol{\beta} \in \mathbb{R}^p$ such that:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, q\boldsymbol{\Sigma}), \tag{1}$$

where we let $\mathbf{X}_{n\times p}$ be the design matrix containing predictor information for n individuals on p features. In (1) we also denoted q as the residual variance, and the variance-covariance matrix as $\boldsymbol{\Sigma}$.

For simplicity, we will assume throughout that we have a design matrix where all p predictors are continuous. Furthermore, we assume each of them to have zero mean and unit variance —i.e., they are standardised—. This that the predictors will have a multivariate normal distribution with variance-covariance matrix:

$$\boldsymbol{\Sigma} = \mathbf{X}^T\mathbf{X}. \tag{2}$$

In order to cluster predictors, it suffices to use the matrix in (2) to search for predictors with high similarity. We will use graph partitioning techniques in order to accomplish effective clustering of a generic set of features. Once this set of predictors has been clustered, the variance-covariance matrix can be block-orthogonalised. The most straight-forward solution is to simply use value thresholding, albeit we refer again to the work in [6] for discussion and posterior solution to the variable selection problem.

To cluster predictors, our approach in this project will be to simulate multivariate normal data under certain artificially set covariance matrices. With these different datasets we will stress out the different algorithms to see how they perform under distinct designs.

Set of algorithms

The set of algorithms employed in this project is the following:

- Spectral clustering: this method is a straight relaxation of the minimum cut problem. It has a few variations, according to what is the exact objective function. We will use three of them:

  1. Unnormalised spectral clustering: this method tries to optimise the Ratio Cut, which is a variation of the minimum cut problem that takes into account the number of nodes in each cluster. We refer to [1] for details on the algorithm.
  2. Normalised spectral clustering according to Ng, Jordan and Weiss: this algorithm optimises the Ncut problem, which takes into account the aggregate degree-wise size of the resulting components. We refer to their work in [5] for detail.
  3. Normalised spectral clustering according to Shi and Malik: this version is related to a random walk, which tries to find a partition of the network such that it optimises how long a Markov Chain is expected to stay within a cluster, seldom jumping between communities. The article by Shi and Malik [7] contains the detailed methodology.

  All spectral clustering algorithms require some fine tuning, that is defining similarity matrices and Laplacian graphs, including their parameters. To keep it simple, we stay with the starting absolute correlation similarity[1] and we will use fully connected graphs weighted by these. We have attempted other similarities, such as the Gaussian similarity, or graph Laplacians that employ ε-neighbourhoods and k nearest neighbours, but parameterising them is a non-trivial problem by itself. This notably complicates the vanilla problem and does not have a clear solution. See [1] for detailed discussion.
- The Girvan-Newman algorithm as studied in the lectures, optimising edge betweenness. It chooses the partition that optimises the modularity score for different clusters attempted by the algorithm. We use the `R` package `igraph` to run this algorithm.
- Modularity maximisation as studied in the lectures, clustering the leading non-negative eigenvector of the modularity matrix of the graph. Again, for this algorithm `igraph` is used.
- The algorithm presented by Zhang and Newman in their article [8]. It works by optimising node distances to aggregates of the group nodes in their spectral space. We refer to the article or the previous research report for detail on the algorithm. In this document we will denote this as the Zhang-Newman algorithm. This method requires to be supplied with the number of clusters but then there is no guarantee that eventually all clusters will be populated. For this reason in some examples we will use two versions: the "free" one that allows the algorithm to finish with any

[1] Note that under standardised features, correlation similarity is equivalent to using the absolute variance-covariance matrix as a similarity matrix.

number of populated clusters, and a "forced" one that requires all clusters to end up with at least one node.

Working examples

A few artificial examples have been generated in order to test the algorithms. All setups for these examples can be found in the Appendix. We have simulated data under all these theoretical setups and applied the algorithms on the sample covariance matrices of the simulations, precisely to take into account the noise present in real-world data. However, in our case this noise is very small, since under each setup we sample 100,000 observations. For "small" data, it could have a lasting effect, a specific case which we do not consider here. Thus, we will consider sizeable realisations of our theoretical models and observe if our algorithms can spot the underlying structure that we have designed.

Results obtained

In this subsection we will break down the results per example, one by one.

Table 1: Clustering results for $\mathbf{\Sigma}_0$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.6319	1 2 3 4 5 6 7 8 9	-
Unnormalised SC	-	-	-	-
Shi-Malik SC	-	-	-	-
Ng-Jordan-Weiss SC	-	-	-	-
Girvan-Newman	-	0.6319	1 2 3 4 5 6 7 8 9	Yes
Modularity maximisation	-	0.7257	1 2 3 4 5 2 6 1 7	No
Free Zhang-Newman	-	-	-	-
Forced Zhang-Newman	-	-	-	-

Notes on the table: "Input k" indicates the number of clusters suggested to the algorithm —when needed—, "Resulting clusters" are the numbered cluster assignments determined by each algorithm. Column "Perfect" indicates whether the algorithm fully recovered the designed clusters.

The first two examples are the two opposite extremes. Results can be found in Table 1 and Table 2 respectively. Obviously, these are only tested for the algorithms that choose the number of clusters themselves. We cannot derive much from them but it gives two starting insights. First, that under orthogonal or quasi-orthogonal design matrices, where the number of nodes is almost the number of clusters, one may be better off by disregarding modularity as the nature of the graph is too specific. Second, that under ill-conditioned setups like $\mathbf{X}_1$ —with corresponding variance-covariance matrix $\mathbf{\Sigma}_1$— Girvan-Newman suffers considerably, as the true modularity is 0. One should take that into account as the lack of community structure in ill-conditioned designs may lead to spurious detection of non-existing clusters. They are spurious in the sense that their similarity may be led by

small perturbations on the data generating process. Luckily, any of these two cases can be spotted quickly by the practitioner examining the nature of the design matrix.

Table 2: Clustering results for $\mathbf{\Sigma}_1$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0	1 1 1 1 1 1 1 1 1	-
Unnormalised SC	-	-	-	-
Shi-Malik SC	-	-	-	-
Ng-Jordan-Weiss SC	-	-	-	-
Girvan-Newman	-	0.0889	1 2 3 4 5 6 7 8 9	No
Modularity maximisation	-	0	1 1 1 1 1 1 1 1 1	Yes
Free Zhang-Newman	-	-	-	-
Forced Zhang-Newman	-	-	-	-

We move on to less trivial cases. Let us analyse the case for $\mathbf{X}_3$, where three natural clusters should appear. Results appear in Table 3. Naturally it should be an easier task for those algorithms to which we supply the number of clusters, which in general is unknown. At a first glance, it seems that modularity is not helping much in ranking the algorithms that recover the underlying structure —in fact the solution is not the one with highest modularity score—, although most algorithms can get very close to the correct solution. It is especially interesting that the two pure network algorithms behave decently, without guide of any kind on the number of clusters. However, and despite being a relatively simple example, only the two normalised spectral clustering methods as well as Zhang-Newman could fully recover the true structure.

Table 3: Clustering results for $\mathbf{\Sigma}_2$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.1562	1 1 1 2 2 2 3 3 3	-
Unnormalised SC	3	0.1431	1 1 1 2 2 2 2 3 2	No
Shi-Malik SC	3	0.1562	1 1 1 2 2 2 3 3 3	Yes
Ng-Jordan-Weiss SC	3	0.1562	1 1 1 2 2 2 3 3 3	Yes
Girvan-Newman	-	0.1802	1 1 1 2 2 1 3 3 3	No
Modularity maximisation	-	0.1779	1 1 1 2 3 3 4 4 4	No
Free Zhang-Newman	3	0.1562	1 1 1 2 2 2 3 3 3	Yes
Forced Zhang-Newman	-	-	-	-

A first complication of the previous example can be found with $\mathbf{\Sigma}_3$, whose results are shown in Table 4. In this case, we have to highly correlated clusters. Here we see that normalised spectral clustering techniques keep doing perfectly, and that Girvan-Newman, modularity maximisation and Zhang-Newman cannot distinguish the two highly correlated clusters. We see that their modularity is higher than the achieved by the rest of methods, including the theoretical design. This may be an insight on why they

fail to recover the structure, as their goal is to maximise modularity —and they do—, but in this context this seems not to fit well with the underlying structure. Nevertheless, we could argue that as both clusters are highly correlated, we may still be better off by joining them after all.

Table 4: Clustering results for $\mathbf{\Sigma}_3$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.3148	1 1 1 2 2 2 3 3 3	-
Unnormalised SC	3	0.2936	1 1 1 1 1 1 2 3 3	No
Shi-Malik SC	3	0.3148	1 1 1 2 2 2 3 3 3	Yes
Ng-Jordan-Weiss SC	3	0.3148	1 1 1 2 2 2 3 3 3	Yes
Girvan-Newman	-	0.3457	1 1 1 1 1 1 2 2 2	No
Modularity maximisation	-	0.3457	1 1 1 1 1 1 2 2 2	No
Free Zhang-Newman	3	0.3457	1 1 1 1 1 1 2 2 2	No
Forced Zhang-Newman	3	0.3285	1 2 1 1 1 1 3 3 3	No

A further complication is the inclusion of clusters with one single node. That is what we introduce in the following example corresponding to $\mathbf{\Sigma}_4$, whose results are in Table 5. When given the right amount of clusters, now all spectral algorithms recover the three communities, and even when given three clusters they can tell which are the main blocks. Girvan-Newman can tell that there are five blocks, although this may not be reliable as it fails to identify them. Modularity maximisation method does a much better job, getting very close to the true structure but missing one last single-node cluster. Zhang-Newman algorithm has the highest modularity score, although it can only spot two clusters, no matter what k is given. One possibility is that the small amount of nodes is damaging its performance in this context.

Table 5: Clustering results for $\mathbf{\Sigma}_4$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.2366	1 1 1 2 2 2 3 4 5	-
Unnormalised SC	5	0.2366	1 1 1 2 2 2 3 4 5	Yes
Unnormalised SC	3	0.2936	1 1 1 1 1 1 2 3 2	-
Shi-Malik SC	5	0.2366	1 1 1 2 2 2 3 4 5	Yes
Shi-Malik SC	3	0.2936	1 1 1 1 1 1 2 3 2	-
Ng-Jordan-Weiss SC	5	0.2366	1 1 1 2 2 2 3 4 5	Yes
Ng-Jordan-Weiss SC	3	0.2936	1 1 1 1 1 1 2 3 2	-
Girvan-Newman	-	0.3045	1 2 3 4 4 4 5 5 5	No
Modularity maximisation	-	0.2627	1 1 1 2 2 2 3 3 4	No
Free Zhang-Newman	5	0.3457	1 1 1 1 1 1 2 2 2	No
Free Zhang-Newman	3	0.3457	1 1 1 1 1 1 2 2 2	No

We can tighten this example even further and introduce some asymmetric correlation

in the second cluster, as seen in $\mathbf{\Sigma}_5$. Results can be found in Table 6. Again, spectral clustering methods perform excellently. They can spot the seven communities when suggested $k = 7$, and they can tell the most distinct clusters if given $k = 5$ —i.e. if we want to consider the three central predictors to be one cluster—. Given that we have few nodes per cluster, Girvan-Newman suffers in effective performance despite finding the highest modularity score, and Zhang-Newman and modularity maximisation struggle as well, although in this example they look qualitatively better than Girvan-Newman.

Table 6: Clustering results for $\mathbf{\Sigma}_5$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.2092	1 1 1 2 3 4 5 6 7	-
Unnormalised SC	7	0.2092	1 1 1 2 3 4 5 6 7	Yes
Unnormalised SC	5	0.2226	1 1 1 2 2 2 3 4 5	-
Shi-Malik SC	7	0.2092	1 1 1 2 3 4 5 6 7	Yes
Shi-Malik SC	5	0.2226	1 1 1 2 2 2 3 4 5	-
Ng-Jordan-Weiss SC	7	0.2092	1 1 1 2 3 4 5 6 7	Yes
Ng-Jordan-Weiss SC	5	0.2226	1 1 1 2 2 2 3 4 5	-
Girvan-Newman	-	0.2895	1 2 3 4 4 4 5 5 5	No
Modularity maximisation	-	0.2455	1 1 1 2 2 2 3 4 4	No
Free Zhang-Newman	7	0.2481	1 1 1 2 2 2 3 3 4	No
Free Zhang-Newman	5	0.2851	1 1 1 2 3 2 4 4 4	No

Seeing that with a small set of predictors some methods can suffer, in the remaining examples we will analyse bigger matrices. First, we consider the example corresponding to covariance matrix $\mathbf{\Sigma}_6$ in Table 7. This is a replica of $\mathbf{\Sigma}_2$, doubling symmetrically the amount of predictors. In higher dimensions results improve, as all spectral methods work well, as well as Zhang-Newman in this case, which already did well with $\mathbf{\Sigma}_2$. Girvan-Newman detects the number of clusters but cannot recover the structure. Modularity maximisation achieves a better result but splits the middle cluster in two. The combination of these results seems to confirm that modularity may not be the most appropriate measure in this context, albeit being able to spot approximately the correct amount of communities.

Table 7: Clustering results for $\mathbf{\Sigma}_6$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.1189	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	-
Unnormalised SC	3	0.1189	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes
Shi-Malik SC	3	0.1189	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes
Ng-Jordan-Weiss SC	3	0.1189	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes
Girvan-Newman	-	0.1486	1 1 1 1 1 1 2 2 2 2 3 2 2 2 2 2 2 2	No
Modularity maxim.	-	0.1405	1 1 1 1 1 1 2 2 2 3 3 3 4 4 4 4 4 4	No
Free Z-N	3	0.1189	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes

Example corresponding to matrix $\mathbf{\Sigma}_7$ has the same underlying structure as the previous example but increasing the collinearity of predictors. With this we want to stress out the effect of ill-behaved situations, following the observations made with $\mathbf{\Sigma}_1$. Results can be seen in Table 8. As we can see, normalised spectral methods do well as previously, but now both modularity maximisation and Zhang-Newman methods can compare to them. Increasing the number of nodes has been positive to their performance. Girvan-Newman spots the first community but cannot identify the third cluster. In general, however, we note that increasing predictor correlation has not damaged performance.

Table 8: Clustering results for $\mathbf{\Sigma}_7$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.0351	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	-
Unnormalised SC	3	0.0270	1 1 1 2 1 1 1 1 1 1 1 1 3 3 3 3 3 3	No
Shi-Malik SC	3	0.0351	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes
Ng-Jordan-Weiss SC	3	0.0351	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes
Girvan-Newman	-	0.0234	1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2	No
Modularity maxim.	-	0.0351	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes
Free Z-N	3	0.0351	1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3	Yes

We will analyse one final example corresponding to $\mathbf{\Sigma}_8$, whose results can be observed in Table 9. The number of predictors has been expanded to 21 and now the symmetric matrix includes negative correlations. Naturally, with absolute correlation distance one would expect four clusters in this setting. No algorithm has been able to fully recover the underlying structure, although some of them fall just short of perfection. Shi-Malik normalised spectral clustering, that scored perfectly on all previous tests, misplaces one single set, joining two separate clusters. The rest perform similarly, and most can recover the structure in sets of three easily. Surprisingly, modularity is higher for all algorithms compared to our design. Increasing dimension has qualitatively benefited most methods, thus scalability does not seem to be a concern.

Table 9: Clustering results for $\mathbf{\Sigma}_8$

Algorithm	Input k	Modularity	Resulting clusters	Perfect
Design	-	0.0763	1 1 1 2 2 2 3 3 3 4 4 4 3 3 3 2 2 2 1 1 1	-
Unnormalised SC	4	0.0832	1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 1 1 1	No
Shi-Malik SC	4	0.0851	1 1 1 2 2 2 3 3 3 4 4 4 3 3 3 4 4 4 1 1 1	No
Ng-Jordan-Weiss SC	4	0.0938	1 1 1 2 2 2 3 3 3 4 4 4 4 4 4 4 4 4 3 3 3	No
Girvan-Newman	-	0.0819	1 1 1 2 3 4 5 5 5 6 5 7 7 7 7 1 1 1 1 1 1	No
Modularity maxim.	-	0.1018	1 1 1 2 2 2 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1	No
Free Z-N	4	0.0840	1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 3 3 3 3 3 3	No

All in all, with these examples we can summarise a few main ideas derived from these results in the following section.

Conclusions

As a result of the outcomes of our artificial examples, spectral clustering algorithms are the ones that look most promising in the context of recovering the block-diagonal structure of predictors in a design matrix. It is especially worth mentioning the performance by normalised spectral clustering methods, which have recovered the underlying structures consistently throughout different settings.

Another insight we have observed is that in this context modularity may not be the best measure to find our communities. Algorithms based on modularity have lacked precision in general to recover such structures. Nonetheless, they behave quite well taking into account that they have no input information on the number of clusters, unlike the spectral clustering counterparts. They have been quite effective in finding a range of underlying communities. Therefore, a good alternative would be to run these methods as a starting point to find the right range of clusters, and posteriorly resort to spectral clustering algorithms with the resulting number of communities.

Additionally, we noticed that the limitation of number of nodes per cluster has had a negative effect on their performance. When increasing the number of features, results have appeared to be qualitatively better, which is promising in terms of scalability, thinking of settings with high predictor dimensionality —the most involved and sensitive case—.

Leaving the theoretical setups would be a natural step following this results, trying these methods on real and sizeable datasets where the community structure could be known ex-ante, and see if these insights also hold in reality. Testing these methods in truly high dimensions is an open challenge to come.

Appendix

- Covariance matrix Σ_0:

$$\Sigma_0 := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The identity matrix is an extreme, as it has nine linearly independent predictors. As a starting point, any algorithm that chooses the number of clusters should separate it in nine clusters of one node each.

- Covariance matrix $\mathbf{\Sigma}_1$:

$$\boldsymbol{\Sigma}_1 := \begin{pmatrix} 1 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 \\ 0.99 & 1 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 \\ 0.99 & 0.99 & 1 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 \\ 0.99 & 0.99 & 0.99 & 1 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 \\ 0.99 & 0.99 & 0.99 & 0.99 & 1 & 0.99 & 0.99 & 0.99 & 0.99 \\ 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 1 & 0.99 & 0.99 & 0.99 \\ 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 1 & 0.99 & 0.99 \\ 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 1 & 0.99 \\ 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 0.99 & 1 \end{pmatrix}$$

This covariance matrix is the opposite extreme. These predictors are extremely correlated and thus should exhibit no community structure.

- Covariance matrix $\boldsymbol{\Sigma}_2$:

$$\boldsymbol{\Sigma}_2 := \begin{pmatrix} 1 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 2/3 & 1 & 2/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 1 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 2/3 & 1 & 2/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 1 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1 & 2/3 & 2/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 2/3 & 1 & 2/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 1 \end{pmatrix}$$

Here, there are three sets of predictors, equally spaced within the $[0, 1] \times \mathbb{R}^1$ space. It is a naive case with three clusters and three predictors each, which are not that highly correlated but still exhibit symmetric and sufficiently large distance.

- Covariance matrix $\boldsymbol{\Sigma}_3$:

$$\boldsymbol{\Sigma}_3 := \begin{pmatrix} 1 & 0.9 & 0.9 & 0.8 & 0.8 & 0.8 & 0 & 0 & 0 \\ 0.9 & 1 & 0.9 & 0.8 & 0.8 & 0.8 & 0 & 0 & 0 \\ 0.9 & 0.9 & 1 & 0.8 & 0.8 & 0.8 & 0 & 0 & 0 \\ 0.8 & 0.8 & 0.8 & 1 & 0.9 & 0.9 & 0 & 0 & 0 \\ 0.8 & 0.8 & 0.8 & 0.9 & 1 & 0.9 & 0 & 0 & 0 \\ 0.8 & 0.8 & 0.8 & 0.9 & 0.9 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.9 & 0.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 1 & 0.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0.9 & 1 \end{pmatrix}$$

This is a similar example, with the complication that now two of the clusters are highly correlated. It is more challenging to spot not two but the three clusters in the space.

- Covariance matrix $\boldsymbol{\Sigma}_4$:

$$\boldsymbol{\Sigma}_4 := \begin{pmatrix} 1 & 0.9 & 0.9 & 0.8 & 0.8 & 0.8 & 0 & 0 & 0 \\ 0.9 & 1 & 0.9 & 0.8 & 0.8 & 0.8 & 0 & 0 & 0 \\ 0.9 & 0.9 & 1 & 0.8 & 0.8 & 0.8 & 0 & 0 & 0 \\ 0.8 & 0.8 & 0.8 & 1 & 0.9 & 0.9 & 0 & 0 & 0 \\ 0.8 & 0.8 & 0.8 & 0.9 & 1 & 0.9 & 0 & 0 & 0 \\ 0.8 & 0.8 & 0.8 & 0.9 & 0.9 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.1 & 1 \end{pmatrix}$$

This case is identical to the previous example with one variation: the isolated cluster now breaks down into three clusters with a single predictor. An algorithm able to find the three clusters in the previous example should still find them here, but ideally it should spot two more.

- Covariance matrix $\boldsymbol{\Sigma}_5$:

$$\boldsymbol{\Sigma}_5 := \begin{pmatrix} 1 & 0.9 & 0.9 & 0.6 & 0.5 & 0.4 & 0 & 0 & 0 \\ 0.9 & 1 & 0.9 & 0.6 & 0.5 & 0.4 & 0 & 0 & 0 \\ 0.9 & 0.9 & 1 & 0.6 & 0.5 & 0.4 & 0 & 0 & 0 \\ 0.6 & 0.6 & 0.6 & 1 & 0.8 & 0.6 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.8 & 1 & 0.8 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0.4 & 0.6 & 0.8 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.1 & 1 \end{pmatrix}$$

We replicate the case adding an extra difficulty: the second community is also divided into three new clusters. These predictors, however, are correlated with the first cluster, which will difficult its detection. An algorithm should at least detect the previous five clusters, albeit ideally it should spot seven clusters.

- Covariance matrix $\boldsymbol{\Sigma}_6$:

$$\boldsymbol{\Sigma}_7 := \begin{pmatrix} 1 & 2/3 & 2/3 & 2/3 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2/3 & 1 & 2/3 & 2/3 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1 & 2/3 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2/3 & 2/3 & 2/3 & 1 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2/3 & 2/3 & 2/3 & 2/3 & 1 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2/3 & 2/3 & 2/3 & 2/3 & 2/3 & 1 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1 & 2/3 & 2/3 & 2/3 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 1 & 2/3 & 2/3 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 1 & 2/3 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 2/3 & 1 & 2/3 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 2/3 & 2/3 & 1 & 2/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 2/3 & 2/3 & 2/3 & 1 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 1 & 2/3 & 2/3 & 2/3 & 2/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 1 & 2/3 & 2/3 & 2/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 2/3 & 1 & 2/3 & 2/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 2/3 & 2/3 & 1 & 2/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 2/3 & 2/3 & 2/3 & 2/3 & 2/3 & 1 \end{pmatrix}$$

In this example we scale to higher dimensions. We have 18 predictors that cluster symmetrically in the fashion of $\mathbf{\Sigma}_2$, in the same three clusters but with more predictors per community. This is the naive example in a higher dimension to see without interference what methods can scalably keep performing well.

- Covariance matrix $\mathbf{\Sigma}_7$:

$$\mathbf{\Sigma}_7 := \begin{pmatrix}
1.0 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
0.9 & 1.0 & 0.9 & 0.9 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
0.9 & 0.9 & 1.0 & 0.9 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
0.9 & 0.9 & 0.9 & 1.0 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
0.9 & 0.9 & 0.9 & 0.9 & 1.0 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 1.0 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 1.0 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 1.0 & 0.9 & 0.9 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1.0 & 0.9 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 0.9 & 1.0 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 0.9 & 0.9 & 1.0 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 1.0 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 1.0 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 \\
0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 1.0 & 0.9 & 0.9 & 0.9 & 0.9 \\
0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1.0 & 0.9 & 0.9 & 0.9 \\
0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 0.9 & 1.0 & 0.9 & 0.9 \\
0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 0.9 & 0.9 & 1.0 & 0.9 \\
0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 1.0
\end{pmatrix}$$

Here we increase the correlations between the predictors, and to some extent all of them will be correlated. With a larger number of predictors, high correlations may cause the algorithms to mix up membership. It will be relevant to observe which of them can still detect the three communities.

- Covariance matrix $\mathbf{\Sigma}_8$:

$$\mathbf{\Sigma}_8 := \begin{pmatrix}
1 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 & -0.6 & -0.6 & -0.6 & -0.9 & -0.9 & -0.9 \\
0.9 & 1 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 & -0.6 & -0.6 & -0.6 & -0.9 & -0.9 & -0.9 \\
0.9 & 0.9 & 1 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 & -0.6 & -0.6 & -0.6 & -0.9 & -0.9 & -0.9 \\
0.6 & 0.6 & 0.6 & 1 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 & -0.6 & -0.6 & -0.6 \\
0.6 & 0.6 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 & -0.6 & -0.6 & -0.6 \\
0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 & -0.6 & -0.6 & -0.6 \\
0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 1 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 \\
0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 \\
0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 & -0.3 & -0.3 & -0.3 \\
0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 1 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 & 0 & 0 & 0 \\
-0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 1 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 \\
-0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 \\
-0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1 & 0.6 & 0.6 & 0.6 & 0.3 & 0.3 & 0.3 \\
-0.6 & -0.6 & -0.6 & -0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 1 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 \\
-0.6 & -0.6 & -0.6 & -0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.6 & 0.6 \\
-0.6 & -0.6 & -0.6 & -0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1 & 0.6 & 0.6 & 0.6 \\
-0.9 & -0.9 & -0.9 & -0.6 & -0.6 & -0.6 & -0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 1 & 0.9 & 0.9 \\
-0.9 & -0.9 & -0.9 & -0.6 & -0.6 & -0.6 & -0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 1 & 0.9 \\
-0.9 & -0.9 & -0.9 & -0.6 & -0.6 & -0.6 & -0.3 & -0.3 & -0.3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.3 & 0.6 & 0.6 & 0.6 & 0.9 & 0.9 & 1
\end{pmatrix}$$

This final example explores the generalised performance in the whole $[-1, 1]$ space. An algorithm should find four clusters as negatively highly correlated predictors belong to the same cluster –they carry similar information—.

References

1. von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4), pp.395-416.

2. Newman, M. (2004). Analysis of weighted networks. *Physical Review E*, 70(5).

3. Newman, M. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23), pp.8577-8582.

4. Newman, M. (2013). Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4).

5. Ng, A., Jordan, M. and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14, pp.849-856.

6. Papaspiliopoulos, O. and Rossell, D. (2016). Scalable variable selection and model averaging under block-orthogonal design. *arXiv*, pp.1–19.

7. Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), pp.888-905.

8. Zhang, X. and Newman, M. (2015). Multiway spectral community detection in networks. *Physical Review E*, 92(5).