PDM Package Vignette

Rosemary Braun <braunr@mail.nih.gov>

March 7, 2012

Abstract

The purpose of this document is to walk the user through a demonstration of the PDM package. The commands are basically the same as those in the PDM package demo (`demo(PDM)`).

Contents

Overview

The PDM package carries out the Partition Decoupling Method (PDM) as described in [1]. The PDM [1, 2, 3] is an unsupervised machine-learning or clustering technique that consists of two iterated submethods: the first, spectral clustering [4, 5, 6], finds the dominant structures within the system, while the second "scrubbing" step removes this structure (by projecting the data onto the cluster centroids and taking the residuals) such that the next clustering iteration can articulate an independent set of clustering relationships. The two steps are repeated until the residuals are indistinguishable from noise. By performing successive clustering steps, factors contributing to the partitioning of the data at different scales may be revealed, as demonstrated in [1, 2].

Methodology

The following is a brief overview of the method; for a complete description of the PDM and spectral clustering, we refer the reader to the literature [1, 2, 4, 5, 6].

The first step, spectral clustering, serves to identify clusters of samples in high-dimensional gene-expression space. Spectral clustering offers several advantages over traditional clustering algorithms; most importantly, no constraint is placed on the geometry of the data, in contrast to the tree-like structure imposed by hierarchical clustering [7] or the necessity of convexity of the clusters for detection via distance-based k-means clustering [8, 9] and in Self Organizing Maps [10], meaning that clusters with nonlinear and

nonconvex boundaries may be articulated (cf. [1, 2, 4, 5, 6]). Spectral clustering also uses a low-dimensional embedding of the data, thus excluding the noisy, high-frequency components.

In spectral clustering, the data are represented as a complete graph in which nodes correspond to samples and edge weights $s_{i,j}$ correspond to some measure of similarity between a pair of nodes i and j. In the PDM package, the similarity is defined as a Gaussian function of the distance $t_{i,j}$ between samples i and j with a scaling parameter σ that may be tuned (by setting the parameter `sigma`) to set the scale of distances deemed significant (cf. [1, 2]):

$$s_{i,j} = \exp\left(-\frac{t_{i,j}^2}{2\sigma^2}\right). \tag{1}$$

The function that computes the distance $t_{i,j}$ may be supplied by the user as the `distanceFn` parameter, and should return distances scaled on $[0, 1]$; the PDM package also includes a correlation-based distance function `correlationDist` which gives $t_{i,j} = \sin(\arccos(\rho_{i,j})/2)$ (the default) and a Euclidean distance function `euclideanDist` for the user's convenience.

Spectral graph theory (see, e.g., [4]) is used to find groups of connected, high-weight (ie, high-similarity) edges that define clusters of samples. This problem may be reformulated as a form of the min-cut problem: cutting the graph across edges with low weights, so as to generate several subgraphs for which the similarity between nodes is high and the cluster sizes preserve some form of balance in the network. It has been demonstrated [4, 5, 6] that this problem may be solved through eigendecomposition of the graph Laplacian matrix $\mathbf{L}$, a matrix derived from the similarity matrix $\mathbf{S}$ (with entries s_{ij}) and the diagonal degree matrix $\mathbf{D}$ (where the ith element on the diagonal is the degree of entity i, $d_i = sum_j s_{ij}$) that encapsulates the geometry of the system. Several normalized forms of the Laplacian exist [6]; by default, PDM uses the symmetric normalized form

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\,\mathbf{S}\,\mathbf{D}^{-1/2}, \tag{2}$$

but the parameter `norm` may be set to `"symmetric"` (the default), `"rowsum"` (for the row-sum normalized Laplacian), or `"none"` (for the unnormalized Laplacian)—see [6] for descriptions.

Eigendecomposition of $\mathbf{L}$ contains information regarding the graph geometry. Specifically, smaller eigenvalues correspond to coarser geometry encapsulated by their corresponding eigenvector (akin to larger loadings corresponding to larger variance encapsulated by components in PCA). The 0-eigenvalues correspond to the number of connected components (here, one) and are trivial; the first non-zero eigenvector/eigenvalue pair ("Fiedler value" λ_1 and "Fiedler vector" v_1) describe the geometry having the coarsest structure. A one–dimensional summary via the Fiedler vector provides the greatest dimension reduction—but optimal with respect to the geometry—of the data. Finer resolution is provided by the dimension reductions obtained by increasing the dimensionality via the use of additional eigenvectors (in order according to increasing eigenvalue). Conceptually, this is similar to PCA, but with the benefit that the coordinates in this so-called "Laplacian eigenmap" may be nonlinear in the original coordinate space (additionally, it may be shown that the Laplacian eigenmap is identical, up to a multiplicative factor, to kernel PCA using a Gaussian kernel).

By using a low-dimensional nonlinear embedding the data as defined by the low-frequency eigenvectors and clustering the embedded data using k-means [8], the geometry of the data may be revealed. In order to use k-means on the embedded data, two parameters need to be set: the number of eigenvectors l to use (that is, the dimensionality of the embedded data) and the number of clusters k into which the data will be clustered.

Optimization of l

The optimal dimensionality of the embedded data is obtained by comparing the spectrum (eigenvalues of the Laplacian) to a reference distribution of spectra from resampled data. The motivation of this approach follows from the observation that the size of eigenvalues corresponds to the degree of structure (see [6]), with smaller eigenvalues corresponding to greater structure. Specifically, we wish to construct a distribution of null spectra select the eigenvalues from the true data that are significantly small with respect to this distribution (below the 0.05 quantile) or have significantly large eigengaps, defined as $\lambda_i - \lambda_{i-1}$ (above the

0.95 quantile). In doing so, we select the eigenvalues that indicate greater structure than would be expected by chance alone.

As described in [1], the resampling may be done on the distances $t_{i,j}$ (or similarities $s_{i,j}$), corresponding to "rewiring" the graph and thus preserving the marginal distributions of the data; alternatively, the data itself may be resampled row-wise, which destroys the marginal distributions but may in some cases be a more appropriate null model. This can be controlled by setting the `resample.by` parameter to `"distances"` or `"rows"`, respectively. The number of resamplings is controlled by the parameter `resample.runs` and defaults to 40.

Once the reference distribution of the spectrum is obtained, the true spectrum may be compared to it. The comparison criteria may be set by parameter `compare`: `"fiedgaps"` (the default) compares the spectral gaps to the 0.95 quantile of the resampled gap of the Fiedler value; `"gaps"` compares the spectral gaps to the 0.95 quantiles of the corresponding resampled gaps; `"fiedval"` compares the spectrum to the 0.05 quantile of the resampled Fiedler value; and `"vals"` compares the spectrum to the 0.05 quantiles of the corresponding resampled eigenvalues.

The dimensions l of the embedding may be set via the parameter `l` as `"sigEigvals"` (the default) to be the significant eigenvaluse determined as described above, `"k"` or `"double.k"` to equal the number or twice the number of clusters k (see below), or as a numeric value. Note that even if `l` is not set to `"sigEigvals"`, the resampling should still be adequately performed to assess the stopping criteria (see Stopping, below).

Optimization of k

The PDM package provides the facility for obtaining the number of clusters as described in [1, 3]. Specifically, one can use the number peaks in the density of the Fiedler vector—that is, the number of values about which the elements of v_1 are clustered—as the number of clusters, roughly analogous to finding regions of high density along the first principle component of the data. To obtain this value, we fit a Gaussian mixture model [11] with 2–30 components (assuming unequal variances), compute the Bayesian Information Criterion (BIC) for each mixture model, and choose the optimum number of components (for details of the BIC implementation, see [12, 13]).

The number of clusters k may be set via the parameter `k` as `"BIC"` (the default) to compute k as described above; `"NsigEigvals"` to set k equal to the number of significant eigenvalues (see the Optimization of l discussion, above); or a numeric value.

Scrubbing

After the clustering step has been performed and each data point assigned to a cluster, we wish to "scrub out" the portion of the data explained by those clusters and consider the remaining variation. This is done by computing first the cluster centroids (that is, the mean of all the datapoints assigned to a given cluster), and then subtracting the data's projection onto each of the centroids from the data itself, yielding the residuals. The clustering step may then be repeated on the residual ("scrubbed") data, revealing layers of clusters that articulate relationships at multiple scales in the data.

Stopping

The spectral clustering and scrubbing steps are iteratively carried out by `PDM()` with identical parameters in each iteration for the specified number of layers or until one of two "failures" occur: a) there are no significant eigenvalues (see Optimization of l, above), meaning that the structure in the scrubbed data is indistinguishable from noise, a so-called "partition failure"; or b) the cluster centroids are linearly dependent, a so-called "projection failure." (It should be noted here that the residuals may still be computed in the latter case, but it is unclear how to interpret linearly dependent centroids; to continue PDM iterations despite projection failures, the parameter `scrubOnProjectionFailure` may be set to `TRUE`.)

Using PDM

PDM will attempt to cluster the *columns* of its input data set (note, this is the opposite of `kmeans`), and can accept as input a numeric `matrix`, `data.frame`, or output of a previous PDM run provided there were no failures (ie, it must have produced scrubbed data). We'll begin by loading some data:

```
> library(PDM)

by using mclust, invoked on its own or through another package,
you accept the license agreement in the mclust LICENSE file
and at http://www.stat.washington.edu/mclust/license.txt

> data(golub1999)
> set.seed(100)
```

`golub1999` is a list with gene expressions and phenotypes. Let's take a peek:

```
> names(golub1999)

[1] "expr"  "pheno"

> dim(golub1999$expr)

[1] 999  38

> length(golub1999$pheno)

[1] 38

> golub1999$expr[1:5, 1:4]

            ALL_19769_B.cell ALL_23953_B.cell ALL_28373_B.cell ALL_9335_B.cell
AB002559_at              759             1062              822            1068
AF000231_at              169               88              196             146
AF002020_at               54              235              150             221
AF009426_at               36               38              120              16
AJ000480_at              895             1016              634             920

> golub1999$pheno

ALL_19769_B.cell ALL_23953_B.cell ALL_28373_B.cell  ALL_9335_B.cell
         "B-ALL"          "B-ALL"          "B-ALL"          "B-ALL"
 ALL_9692_B.cell ALL_14749_B.cell ALL_17281_B.cell ALL_19183_B.cell
         "B-ALL"          "B-ALL"          "B-ALL"          "B-ALL"
ALL_20414_B.cell ALL_21302_B.cell   ALL_549_B.cell ALL_17929_B.cell
         "B-ALL"          "B-ALL"          "B-ALL"          "B-ALL"
ALL_20185_B.cell ALL_11103_B.cell ALL_18239_B.cell  ALL_5982_B.cell
         "B-ALL"          "B-ALL"          "B-ALL"          "B-ALL"
 ALL_7092_B.cell   ALL_R11_B.cell   ALL_R23_B.cell ALL_16415_T.cell
         "B-ALL"          "B-ALL"          "B-ALL"          "T-ALL"
ALL_19881_T.cell  ALL_9186_T.cell  ALL_9723_T.cell ALL_17269_T.cell
         "T-ALL"          "T-ALL"          "T-ALL"          "T-ALL"
ALL_14402_T.cell ALL_17638_T.cell ALL_22474_T.cell           AML_12
         "T-ALL"          "T-ALL"          "T-ALL"            "AML"
```

```
AML_13          AML_14          AML_16          AML_20
 "AML"           "AML"           "AML"           "AML"
 AML_1           AML_2           AML_3           AML_5
 "AML"           "AML"           "AML"           "AML"
 AML_6           AML_7
 "AML"           "AML"
```

Basic PDM usage

First, a demonstration of running PDM with default parameters:

```
> pdm.golub <- PDM(golub1999$expr)

Computing layer 1 ...
Computing layer 2 ...
Projection Failure: cluster centroids are not independent. Stopping with current layer.

> pdm.golub

PDMlayers object with:
Layers:  2
Samples:  38
Layer cluster assignments (first 10 samples):
                 layer.1 layer.2
ALL_19769_B.cell       2       3
ALL_23953_B.cell       2       3
ALL_28373_B.cell       2       3
ALL_9335_B.cell        2       3
ALL_9692_B.cell        2       3
ALL_14749_B.cell       2       1
ALL_17281_B.cell       2       3
ALL_19183_B.cell       2       3
ALL_20414_B.cell       2       3
ALL_21302_B.cell       2       1

 ... use "clusters()" to get complete clusters.

Available slots:
 clusters params scrubbed PDMspectra
```

How to the resulting clusters correspond to the true classifications?

```
> table(clusters(pdm.golub)$layer.1, golub1999$pheno)

    AML B-ALL T-ALL
  1   0     0     8
  2  11    19     0

> table(clusters(pdm.golub)$layer.2, golub1999$pheno)

    AML B-ALL T-ALL
  1   1     3     8
  2  10     0     0
  3   0    16     0
```

```
> combinedLayers <- paste("l1-", clusters(pdm.golub)$layer.1, ".l2-",
+     clusters(pdm.golub)$layer.2, sep = "")
> table(combinedLayers, golub1999$pheno)

combinedLayers AML B-ALL T-ALL
     l1-1.l2-1   0     0     8
     l1-2.l2-1   1     3     0
     l1-2.l2-2  10     0     0
     l1-2.l2-3   0    16     0
```

Example of accessing the spectrum and embedded data:

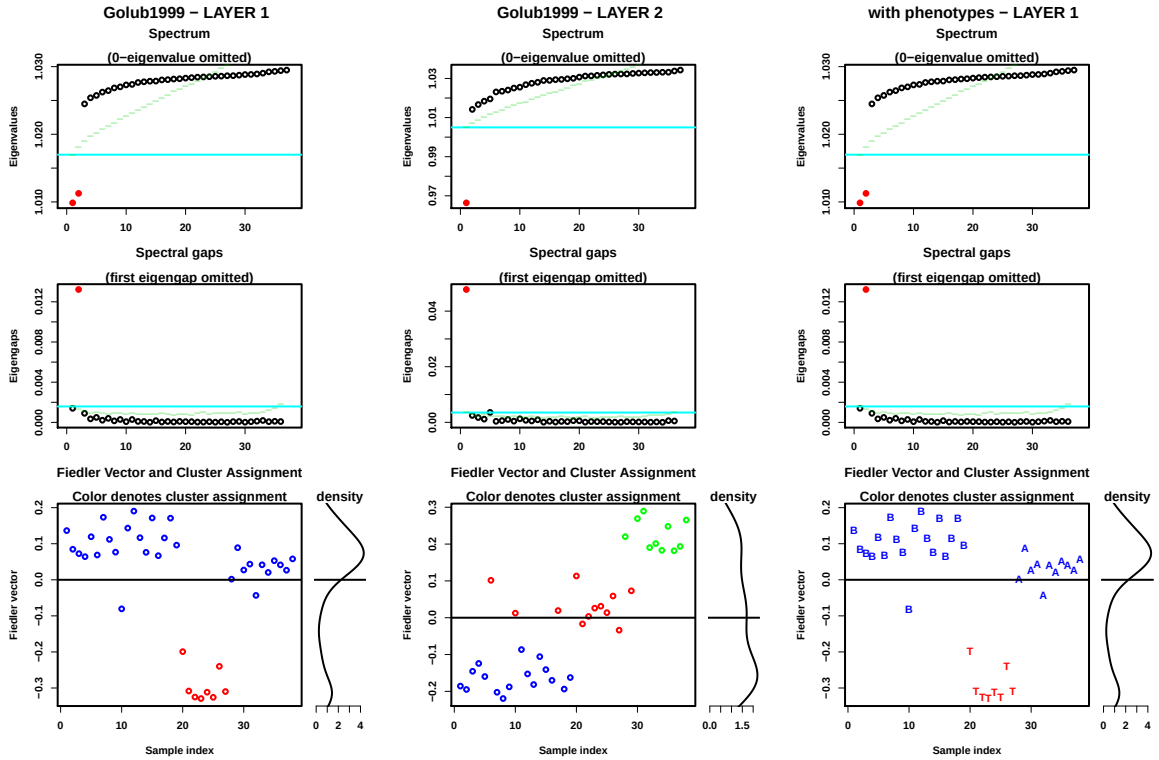
```
> spectrum(pdm.golub, layer = 1)

 [1] 1.009865 1.011265 1.024490 1.025392 1.025743 1.026238 1.026455 1.026856
 [9] 1.027009 1.027300 1.027373 1.027660 1.027758 1.027846 1.027865 1.028037
[17] 1.028075 1.028166 1.028210 1.028305 1.028375 1.028445 1.028477 1.028507
[25] 1.028573 1.028595 1.028653 1.028664 1.028739 1.028829 1.028848 1.028916
[33] 1.029039 1.029230 1.029300 1.029411 1.029492

> dim(embedding(pdm.golub, layer = 2))

NULL
```

Plotting PDM output

Using the following example code:

```
> plot(pdm.golub, main = "Golub1999")
> plot(pdm.golub, 1, pch = golub1999$pheno, main = "with phenotypes")
> separateAt <- which(golub1999$pheno[-length(golub1999$pheno)] !=
+     golub1999$pheno[-1])
> abline(v = separateAt + 0.5, col = "grey")
```

we produce:



Running PDM with user-set parameters

A few examples: first, let's force 3 clusters and set sigma=0.5 through params, and run just one layer:

```
> pdm.golub.k3 <- PDM(golub1999$expr, max.Layers = 1, params = list(sigma = 0.5,
+     k = 3))

Computing layer 1 ...

> plot(pdm.golub.k3, pch = golub1999$pheno, main = "Golub PDM: k=3, sigma=0.5")
```

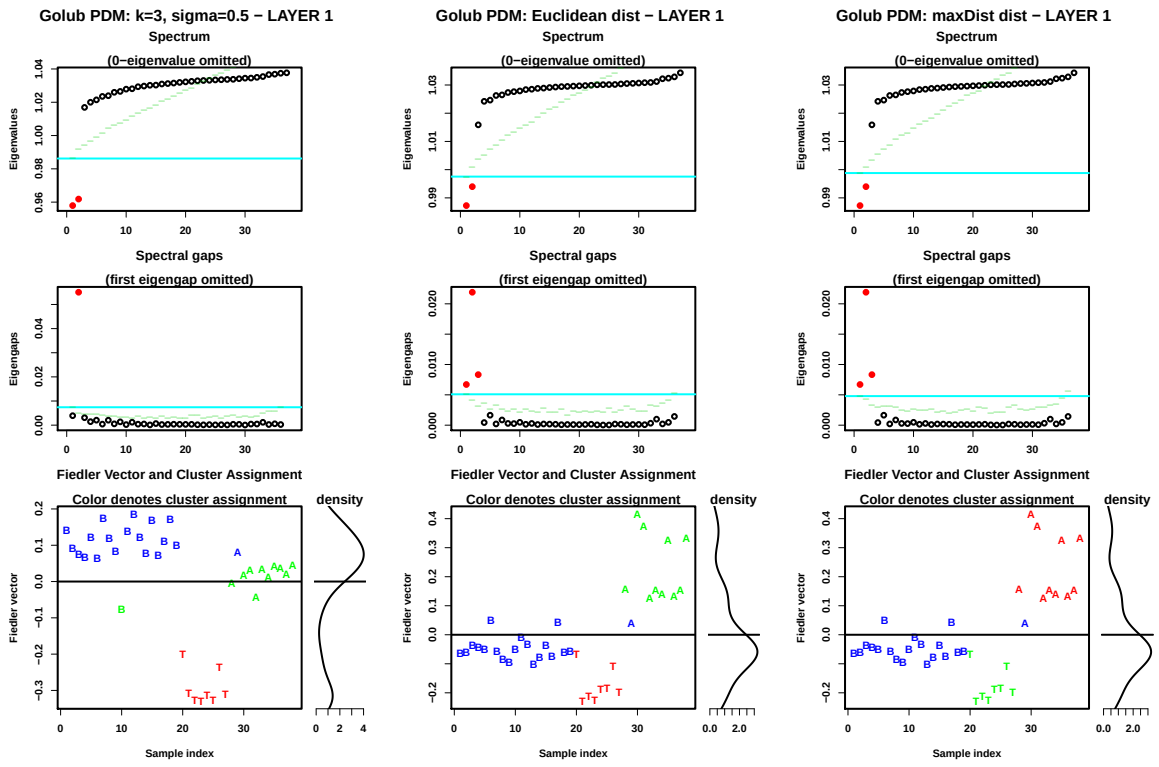
Let's attempt to run another layer from the previous one, now forcing 2 clusters:

```
> PDM(pdm.golub.k3, max.Layers = 1, params = list(sigma = 0.5,
+     k = 2))

Computing layer 1 ...
Projection Failure: cluster centroids are not independent. Stopping with current layer.

PDMlayers object with:
Layers:  1
Samples:  38
Layer cluster assignments (first 10 samples):
 [1] 1 2 1 2 2 2 1 1 1 1

 ... use "clusters()" to get complete clusters.

Available slots:
 clusters params scrubbed PDMspectra
```

Finally, two examples using a different distance function: the built-in Euclidean distance, and the user-supplied distance function:

```
> pdm.golub.euc <- PDM(golub1999$expr, params = list(distanceFn = euclideanDist))

Computing layer 1 ...
Computing layer 2 ...
Partition failure in layer  2 : no structure compared to null data; stopping with previous layer.

> plot(pdm.golub.euc, pch = golub1999$pheno, main = "Golub PDM: Euclidean dist")

> maxDist <- function(x) {
+     d <- as.matrix(dist(t(x), method = "euclidean"))
+     d <- d/max(d)
+     return(d)
+ }
> pdm.golub.max <- PDM(golub1999$expr, params = list(distanceFn = maxDist))

Computing layer 1 ...
Computing layer 2 ...
Partition failure in layer  2 : no structure compared to null data; stopping with previous layer.

> plot(pdm.golub.max, pch = golub1999$pheno, main = "Golub PDM: maxDist dist")
```

The associated plots are:



References

[1] Rosemary Braun, Greg Leibon, Scott Pauls, and Daniel Rockmore. Partition decoupling for multi-gene analysis of gene expression profili ng data. *Forthcoming;*, preprint:[arXiv:1002.3946], 2010.

[2] Gregory Leibon, Scott Pauls, Daniel Rockmore, and Ro bert Savell. Topological structures in the equities market network. *PNAS*, 105(52):20589–20594, December 2008.

[3] Gregory Leibon, Scott Pauls, Daniel Rockmore, and Ro bert Savell. Partition decomposition for roll call data. *In Preparation*, 2011.

[4] F.R.K. Chung. *Spectral graph theory*. Amer Mathematical Society, 1997.

[5] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2:849–856, 2002.

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

[7] M B Eisen, P T Spellman, P O Brown, and D Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95(25):14863–8, 1998.

[8] JA Hartigan and MA Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C, Applied Statistics*, 28:100–108, 1979.

[9] S Tavazoie, J D Hughes, M J Campbell, R J Cho, and G M Ch urch. Systematic determination of genetic network architecture. *Nat Genet*, 22(3):281–5, 1999.

[10] P Tamayo, D Slonim, J Mesirov, Q Zhu, S Kitareewan an d E Dmitrovsky, E S Lander, and T R Golub. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *PNAS*, 96(6):2907–12, 1999.

[11] G. McLachlan and D. Peel. *Finite mixture models*. Wiley-Interscience, 2004.

[12] C. Fraley and A.E. Raftery. MCLUST: Software for model-based cluster analysis. *Journal of Classification*, 16(2):297–306, 1999.

[13] C. Fraley and A.E. Raftery. MCLUST version 3 for R: Normal mixture modeling and model-based clu stering. *Technical Report, Department of Statistics, University of Washington*, 504, 2006.