SamSPECTRAL: A Modified Spectral Clustering Method for Clustering Flow Cytometry Data

Habil Zare and Parisa Shooshtari

April 14, 2011

Contents

1 Introduction

Data analysis is a crucial step in most of recent biological research areas such as microarray techniques, gene expression and protein classification. A classical approach for analysing biological data is to first group individual data points based on some similarity criterion, a process known as clustering, and then compare the outcome of clustering with the desired biological hypotheses. Spectral clustering is a non-parametric clustering method which has proved useful in many pattern recognition areas. Not only it does not require a priori assumptions on the size, shape and distributions of clusters, but it has several features that make it an appropriate candidate for clustering biological data:

- It is not sensitive to outliers, noise or shape of clusters.
- It is adjustable so we can make use of biological knowledge to adapt it for a specific problem or dataset.
- There is mathematical evidence to guarantee its proper performance.

However, because of the machine limitations, one faces serious empirical barriers in applying this method for large data sets. SamSPECTRAL is a modification to spectral clustering such that it will be applicable on large size datasets.

2 How to run SamSPECTRAL?

SamSPECTRAL is an R package source that can be downloaded from BioCunductor. In Linux, it can be installed by the following command:

```
R CMD INSTALL SamSPECTRAL_x.y.z.tar.gz
```

where x.y.z. determines the version.

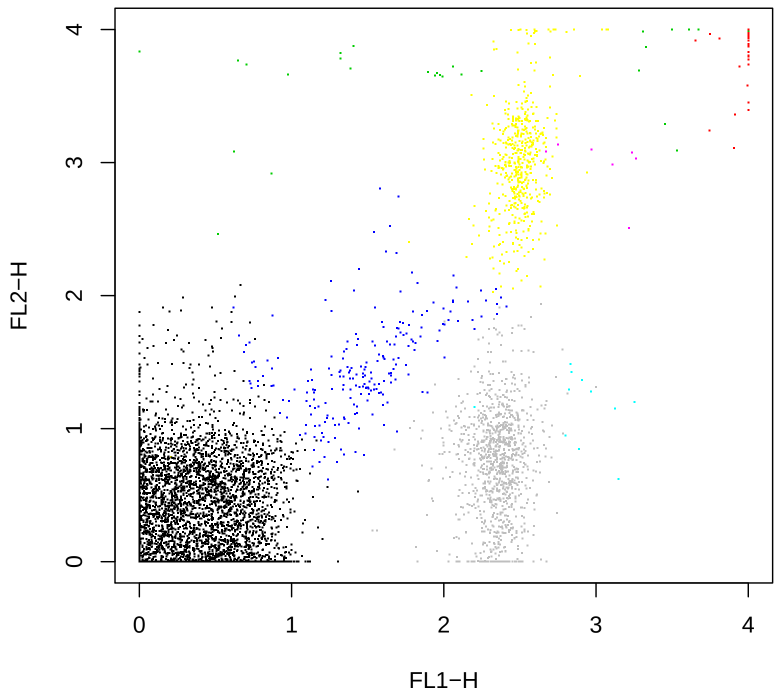
The main function of this package is SamSPECTRAL() which is loaded by using the command library(SamSPECTRAL) in R. Before running this function on a data set, some parameters are required to be set including: normal.sigma and separation.factor. This can be best done by running the algorithm on some number of samples (Normally, 2 or 3 samples are sufficient). Then the function SamSPECTRAL() can be applied to all samples in that data set to identify cell populations in each sample data.

The main function of this package is SamSPECTRAL() which is loaded by using the command library(SamSPECTRAL) in R. Before running this function on a data set, some parameters are required to be set including: normal.sigma and separation.factor. This can be best done by running the algorithm on some number of samples (Normally, 2 or 3 samples are sufficient). Then the function SamSPECTRAL() can be applied to all samples in that data set to identify cell populations in each sample data.

2.1 An example

This example shows how SamSPECTRAL can be run on flow cytometry data. If f is a flow frame (which is normally read from an FCS file using flowCore), then the object "small" in the following example should be replaced by expr(f).

```
> library(SamSPECTRAL)
> data(small_data)
> full <- small
> L <- SamSPECTRAL(full, dimension = c(1, 2, 3), normal.sigma = 200,
+     separation.factor = 0.39)
> plot(full, pch = ".", col = L)
```



3 Adjusting parameters

For efficiency, one can set $m = 3000$ to keep the running time bellow 1 minute by a 2 GHz processor and normally the results remained satisfactory for flow cytometry data. The separation factor and scaling parameter (σ) are two main parameters that needed to be adjusted. The general way is to run SamSPECTRAL on one or two random data samples of a flow cytometry data set and try different values for σ and separation factor. Then, the selected parameters were fixed and used to apply SamSPECTRAL on the rest of data samples. An efficient strategy is explained by the following example.

3.1 Example

First we load data and store the transformed coordinates in a matrix called *full*:

```
> data(small_data)
> full <- small
```

The objects needed for creating this vignette can be directly computed or loaded from previously saved workspace to save time. The later increases the speed of building this vignette.

```
> run.live <- FALSE
```

The following parameters are rarely needed to be changed for flow cytometry data:

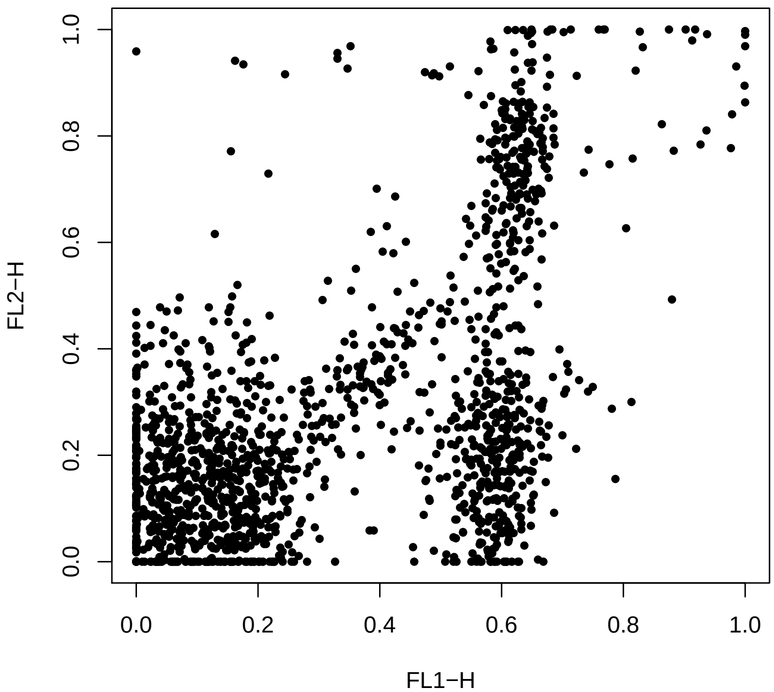
```
> m <- 3000
> community.weakness.threshold <- 1
> precision <- 6
> maximum.number.of.clusters <- 30
```

The following piece of code, scales the coordinates in range [0,1]:

```
> for (i.column in 1:dim(full)[2]) {
+     ith.column <- full[, i.column]
+     full[, i.column] <- (ith.column - min(ith.column))/(max(ith.column) -
+         min(ith.column))
+ }
> space.length <- 1
```

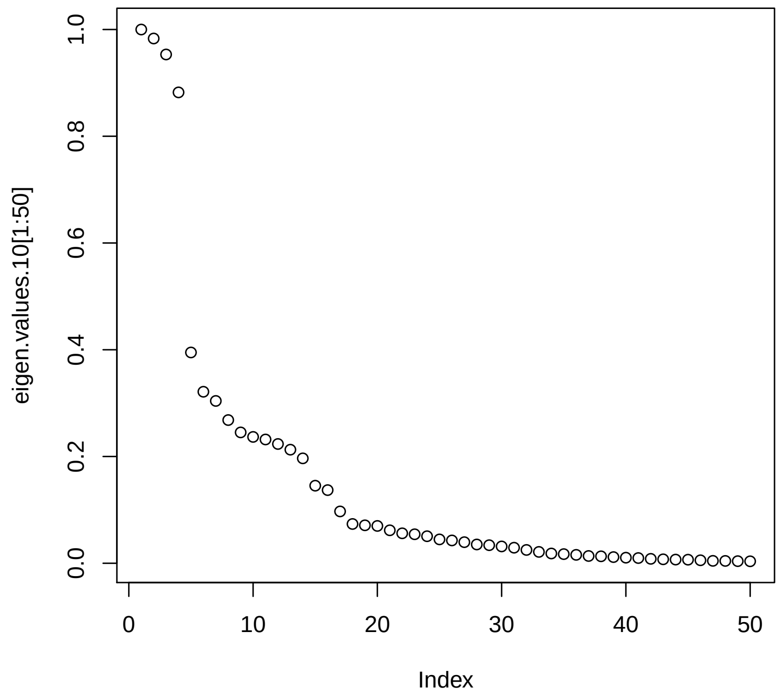
To perform faithful sampling, we run:

```
> society <- Building_Communities(full, m, space.length, community.weakness.thre
> plot(full[society$representatives, ], pch = 20)
```



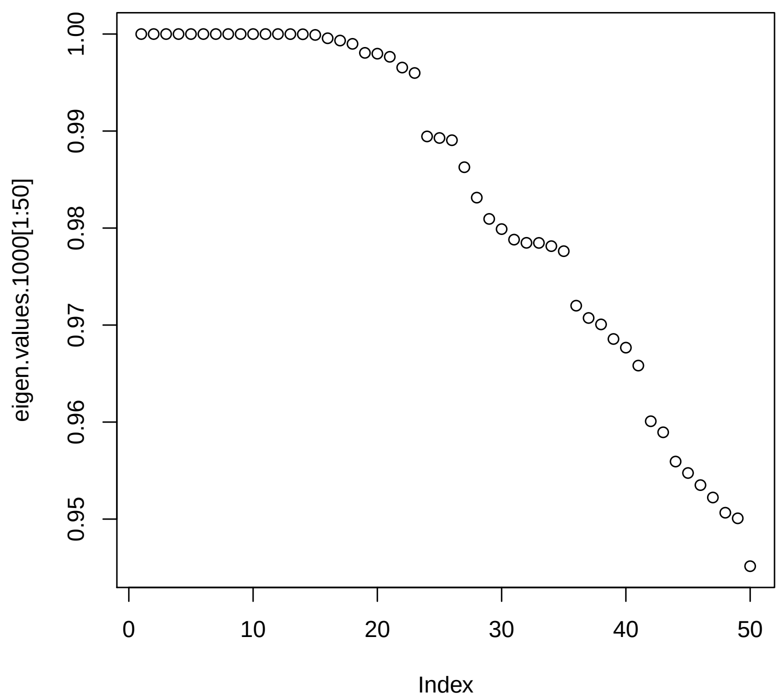
We intend to first find an appropriate value for σ and then set separation factor. Note that normal.sigma=$\frac{1}{\sigma^2}$, therefore, decreasing normal.sigma is equivalent to increasing σ and visa versa. We start with normal.sigma=10:

```
> normal.sigma <- 10
> conductance <- Conductance_Calculation(full, normal.sigma, space.length,
+     society, precision)
> if (run.live) {
+     clust_result.10 <- Civilized_Spectral_Clustering(full, maximum.number.of.c
+         society, conductance, stabilizer = 1)
+     eigen.values.10 <- clust_result.10@eigen.space$values
+ } else data("eigen.values.10")
> plot(eigen.values.10[1:50])
```



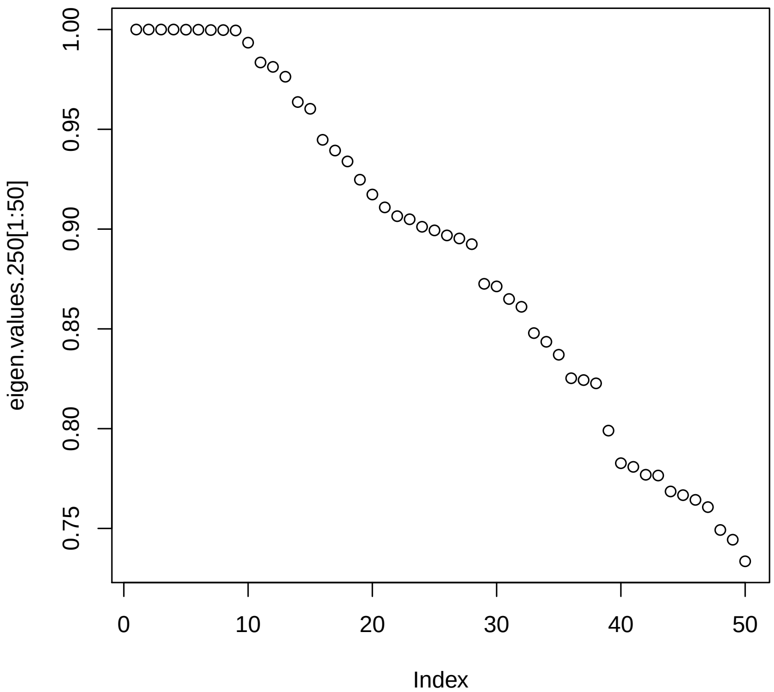
We observe that the eigen values curve does not have a "knee" shape. So we increase sigma:

```
> normal.sigma <- 1000
> conductance <- Conductance_Calculation(full, normal.sigma, space.length,
+     society, precision)
> if (run.live) {
+     clust_result.1000 <- Civilized_Spectral_Clustering(full,
+         maximum.number.of.clusters, society, conductance, stabilizer = 1)
+     eigen.values.1000 <- clust_result.1000@eigen.space$values
+ } else data("eigen.values.1000")
> plot(eigen.values.1000[1:50])
```



We observe that in the eigen values plot, ”too many” values are close to 1 but for this example we do not expect 20 populations. So we decrease sigma:

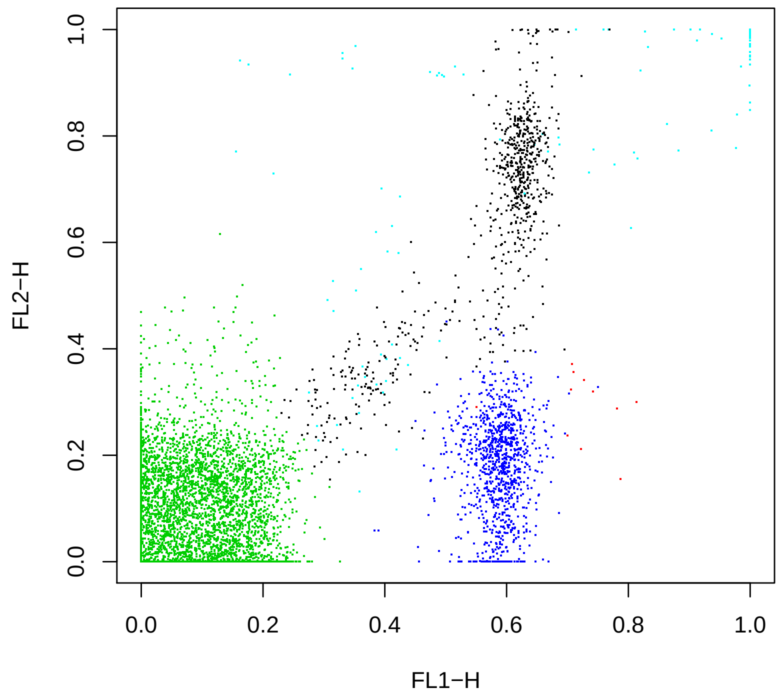
```
> normal.sigma <- 250
> conductance <- Conductance_Calculation(full, normal.sigma, space.length,
+     society, precision)
> clust_result.250 <- Civilized_Spectral_Clustering(full, maximum.number.of.clus
+     society, conductance, stabilizer = 1)
> eigen.values.250 <- clust_result.250@eigen.space$values
> plot(eigen.values.250[1:50])
```



This is ”a right” value for normal.sigma because the curve has now a knee shape. Even some variation to this parameter does not change the shape significantly (200 or 300 can be tried).

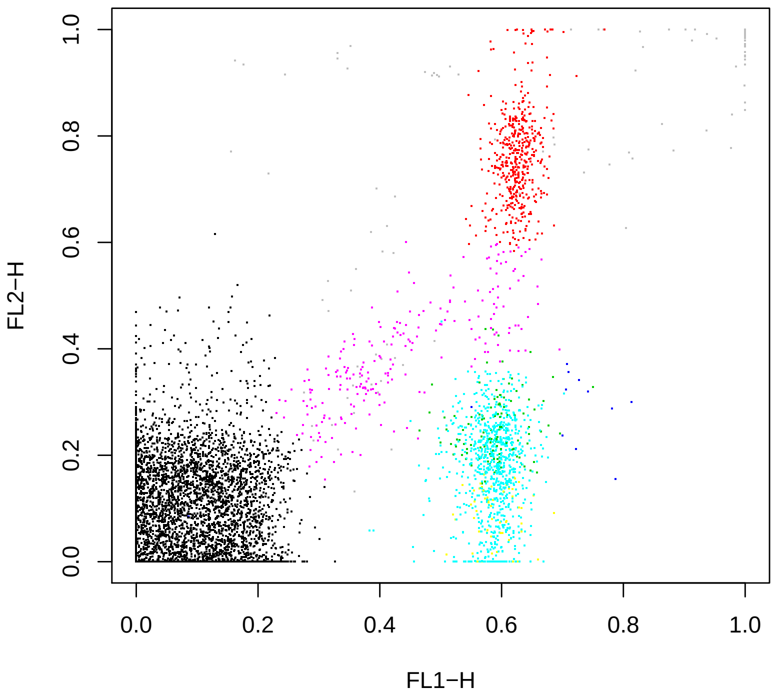
Now having sigma been adjusted, separation factor can be tuned:

```
> labels.for_num.of.clusters <- clust_result.250@labels.for_num.of.clusters
> number.of.clusters <- clust_result.250@number.of.clusters
> L33 <- labels.for_num.of.clusters[[number.of.clusters]]
> separation.factor <- 0.1
> component.of <- Connecting(full, society, conductance, number.of.clusters,
+     labels.for_num.of.clusters, separation.factor)$label
> plot(full, pch = ".", col = component.of)
```



This value is too small for the separation factor and a population is combined by mistake. Therefore, we increase septation factor to separate the components more:

```
> separation.factor <- 0.5
> component.of <- Connecting(full, society, conductance, number.of.clusters,
+     labels.for_num.of.clusters, separation.factor)$label
> plot(full, pch = ".", col = component.of)
```



This is the right value for separator factor as all population are now separated.

Now, we can fix these values for the parameters; normal.sigma=250 and separation.factor=0.5. One can run the SamSPECTRAL algorithm on the rest of the data set without changing them, hopefully, obtaining as appropriate results.

4 Reference

Zare, H. and Shooshtari, P. and Gupta, A. and Brinkman R.B: **Data Reduction for Spectral Clustering to Analyse High Throughput Flow Cytometry Data.** *BMC Bioinformatics*, 2010, **11**:403.