scPMP Clustering Tutorial

Andriana Manousidaki

2023-09-08

Introduction

Recent advances in single-cell technologies have enabled high-resolution characterization of tissue and cancer compositions. Although numerous tools for dimension reduction and clustering are available for single-cell data analyses, these methods often fail to simultaneously preserve local cluster structure and global data geometry.

To address those challenges, we developed a novel analyses framework, Single-Cell Path Metrics Profiling (scPMP), which leverages the usefulness of p-powered path metrics for dimension reduction and clustering. Distances between cells are measured in a data-driven way which is both density sensitive (decreasing distances across high density regions) and respects the underlying data geometry. By combining path metrics with multidimensional scaling, a low dimensional embedding of the data is obtained which respects both the global geometry of the data and preserves cluster structure.

Definition 1 *Path metric. Given a discrete data set X, the discrete p-power weighted path metric between $a, b \in X$ is defined as*

$$\ell_p(a,b) := \inf_{(x_0,\ldots,x_s)} \left(\sum_{i=0}^{s-1} \left\| x_{i+1} - x_i \right\|_2^p \right)^{\frac{1}{p}},$$

where the infimum is taken over all sequences of points $x_0, \ldots, x_s$ in X with $x_0 = a$ and $x_s = b$.

Below, we present a tutorial on the use of scPMP algorithm for the clustering of a data set that simulates single-cell gene expression of three clusters of cells.

Step 1: Load libraries, data set and clustering functions

First let's load required libraries:

```
library(RANN)
library(ClusterR)
library(plyr)
library(cluster)
library(mclust)
library(igraph)
library(RANN)
library(Matrix)
library(expm)
library(cccd)
library(pracma)
```

Load simulated single cell data set:

```
print_qc_plots = FALSE
source('../R_functions/simulation_of_beta_data_set.R')
```

```
## The folder already existsThe folder already exists
```

```
sim_data <- read.csv(file = '../Data_after_Imputation/beta_3_4_10_filtered_saver.csv')
true_labels <- read.csv(file = '../Data_after_Imputation/beta_cell_groups_3_4_10_after_filtering.csv')
true_labels <- true_labels[, -c(1, 2)]

rownames(sim_data) = sim_data[, 1]
sim_data <- sim_data[,-1]
true_labels <- true_labels[, -1]
```

Load the processing and scPMP clustering functions:

```
source('../R_functions/processing_prior_pm_clustering.R')
source('../R_functions/Path_Metrics_Clustering_R.R')
```

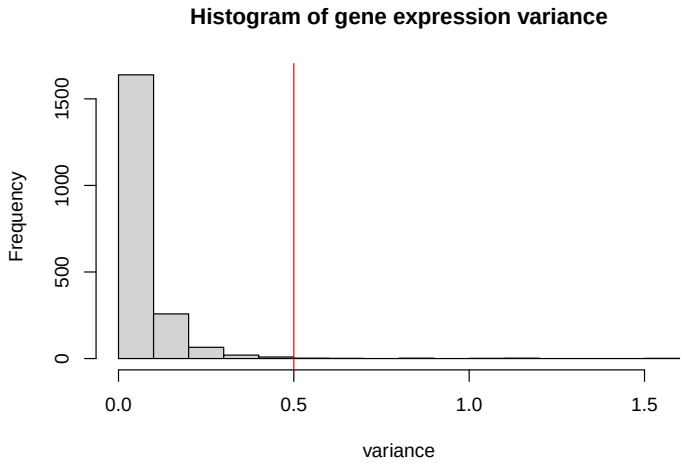
Step 2: Data set processing

For all the function we introduce, the input data set has d rows that correspond to features (genes) and n columns that correspond to different points (cells). The user is advised to first filter the data set to achieve best quality. Then we suggest using our processing function prior to clustering. This function performs the following:

1. log-normalization (parameter *LogNormalization*)
2. restriction to 2000 genes with the highest variance (parameters *EliminateGenes*, *NumberOfGenes*)
3. rescaling of genes with extremely high variance by using a cutoff value (parameter *var_cutoff*),
4. dimension reduction with PCA and
5. denoising by replacing each a point with the mean of its local neighborhood (parameters *LocalAvgAllPts*, *LocalAvgNbhdSize*).

Specifically, if the user prefers other type of normalization, they can set the parameter *LogNormalization* equal to FALSE and in the case that denoising isn't desired *LocalAvgAllPts* has to be set to FALSE.

The simulated data set after steps 1 and 2 has the following distribution of gene variances. Values more than 0.5 (red line) are extremely rare. For this reason the variance cutoff for the simulated data set is 0.5.



Below we process the simulated data set:

```
set.seed(1)
s1 <- proc.time()

sim_data <- processing_prior_pm_clustering(dataset = sim_data,
                                           LogNormalization = TRUE,
                                           var_cutoff = 0.5,
                                           EliminateGenes = TRUE,
                                           NumberOfGenes = 2000,
                                           LocalAvgAllPts = TRUE,
                                           LocalAvgNbhdSize = 12)
```

Step 3: scPMP Clustering

At this step scPMP will construct the kNN graph of the data points. The default value of k is the minimum between 500 and the number of data points, n. Then, the p-powered path metric distance matrix of the points on the kNN graph is calculated.

When p is small the algorithm will cluster the data based on their geometry. On the other hand, for large values of p the algorithm will predict cluster that are separated by low density regions. Although the optimal balance depends on the data set, path metrics with a moderate p exhibit the best performance across a wide range of data sets. For this reason we suggest the use of $p = 2$.

Following, classic multidimesional scaling is used to extract an embedding based on the path metric distances of points (PM-MDS embedding). The embedding dimension can be controlled by the parameter *SpectralOpts.EmbeddingDimension*, but as a default setting we predict the embedding dimension using an eigenratio criterion. For data sets that have more than two thousands points we suggest the use of fast multidimensional scaling, which is available by setting the parameter *Fast_mds* equal to TRUE.

Finally, we apply k-means clustering on the PM-MDS embedding for *n_clust* number of clusters. By default, we merge tiny clusters (i.e. those with size less than $\sqrt{n}/2$) with their closest non-trivial cluster. For regular k-means clustering, users only need to set *run.adjusted.kmeans* equal to FALSE.

If *n_clust* is unknown, prediction of number of clusters can be performed using the silhouette criterion by setting the parameter *silhouette* equal to TRUE.

Below we cluster the simulated data set:

```
s2 <- proc.time()
pm_output <- Path_Metrics_Clustering_R(dataset = sim_data,
                                       n_clust = 3, # desired number of clusters
                                       p = 2, # metric parameter
                                       silhouette = FALSE, #prediction of clusters
                                       SpectralOpts.LearnMDSEmbeddingDimension = TRUE,
                                       Fast_mds = FALSE)
```

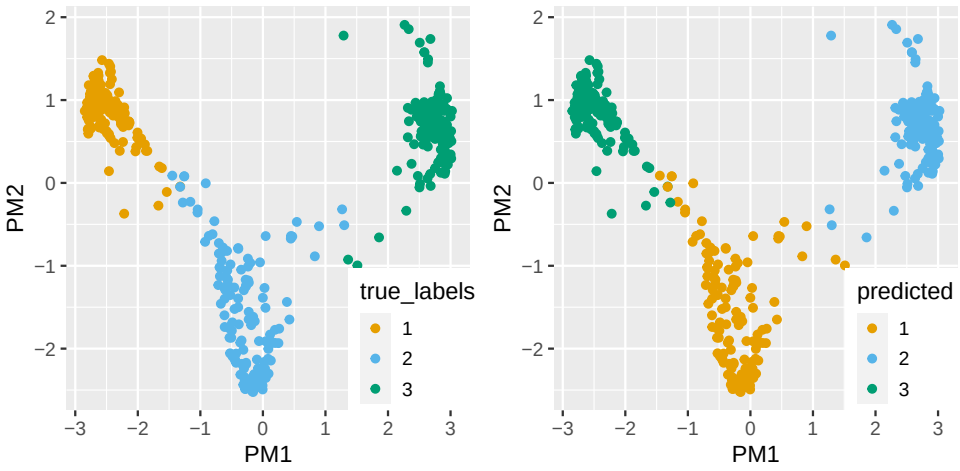
```
## [1] "Classic MDS starts"
```

```
end<-proc.time()
```

The output of scPMP algorithm is a list of four items: the predicted cluster labels, the PM-MDS embedding, the embedding dimension and lastly the eigenvalues of the embedding.

Using the first two dimensions of the embedding and coloring it based on true and predicted labels, we observe that there are only a few misclustered cells.

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```



Evaluation of clustering accuracy and runtime

Now we can numerically evaluate the accuracy of the clustering using Adjusted Rand Index (ARI), Entropy of Cluster Accuracy (ECA) and Entropy of Cluster Purity (ECP). ECA quantifies the variety of true labels within a predicted cluster and ECP quantifies the variety of predicted cluster labels within a true group. Furthermore we record the time needed for processing and clustering.

We observe that the scPMP algorithm achieved high clustering accuracy, with $ARI = 0.969$ and low values of ECP and ECA. The time needed for both clustering and processing was less than a minute.

```
source('../R_functions/clustering_evaluation.R')

accuracy<-clustering_evaluation(pm_output$pm_labels,t(true_labels))
accuracy
```

```
## $ARI
## [1] 0.9687011
##
## $ECP
## [1] 0.05803788
##
## $ECA
## [1] 0.05824176
##
## $number_of_clusters
## [1] 3
```

```
complete_rt=(end-s1)[3]/60
complete_rt
```

```
##   elapsed
## 0.1608333
```

```
clustering_rt=(end-s2)[3]/60
clustering_rt
```

```
##    elapsed
## 0.07616667
```

Evaluation of geometric fidelity of the PM-MDS embedding

To assess whether the embedding procedure preserves the global relative distances between cluster we calculate the geometric perturbation between cluster means in the PCA embedding (ground truth) and in the PM-MDS embedding. Lower values of geometric perturbation mean better preservation of the cluster structure in the PM-MDS embedding. We observe that the geometric perturbation is equal with 0.005, which is extremely low.

```
source('../R_functions/geometric_perturbation.R')
  gp <- geometric_perturbation(X=sim_data,U=pm_output$U1,Labels=true_labels)
  gp
```

```
## $Geometric_Perturbation
## [1] 0.002922139
##
## $Cluster_Centroid_Distances
##          [,1]     [,2]     [,3]
## [1,] 0.000000 3.388372 5.162430
## [2,] 3.388372 0.000000 3.752841
## [3,] 5.162430 3.752841 0.000000
```

Session Information

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Greek_Greece.1253  LC_CTYPE=Greek_Greece.1253
## [3] LC_MONETARY=Greek_Greece.1253 LC_NUMERIC=C
## [5] LC_TIME=Greek_Greece.1253
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] ggplot2_3.4.1      patchwork_1.1.2    SeuratObject_4.1.3 Seurat_4.3.0
##  [5] dplyr_1.1.2        SAVER_1.1.3        pracma_2.4.2       cccd_1.6
##  [9] expm_0.999-7       Matrix_1.5-3       igraph_1.3.5       mclust_6.0.0
## [13] cluster_2.1.2      plyr_1.8.8         ClusterR_1.3.0     RANN_2.6.1
##
## loaded via a namespace (and not attached):
##   [1] Rtsne_0.16             colorspace_2.0-3       deldir_1.0-6
##   [4] ellipsis_0.3.2         ggridges_0.5.4         rstudioapi_0.14
##   [7] proxy_0.4-27           spatstat.data_3.0-1    farver_2.1.1
##  [10] leiden_0.4.3           listenv_0.9.0          ggrepel_0.9.2
##  [13] fansi_1.0.3            codetools_0.2-18       splines_4.1.0
##  [16] doParallel_1.0.17      knitr_1.42             polyclip_1.10-4
##  [19] jsonlite_1.8.4         ica_1.0-3              png_0.1-8
##  [22] uwot_0.1.14            shiny_1.7.4            sctransform_0.3.5
##  [25] spatstat.sparse_3.0-0  compiler_4.1.0         httr_1.4.6
##  [28] fastmap_1.1.0          lazyeval_0.2.2         cli_3.5.0
##  [31] later_1.3.0            htmltools_0.5.4        tools_4.1.0
##  [34] gmp_0.6-9              gtable_0.3.3           glue_1.6.2
##  [37] reshape2_1.4.4         Rcpp_1.0.9             scattermore_0.8
##  [40] vctrs_0.6.2            nlme_3.1-152           spatstat.explore_3.0-5
##  [43] progressr_0.13.0       iterators_1.0.14       lmtest_0.9-40
##  [46] spatstat.random_3.0-1  xfun_0.36              stringr_1.5.0
##  [49] globals_0.16.2         mime_0.12              miniUI_0.1.1.1
##  [52] lifecycle_1.0.3        irlba_2.3.5.1          goftest_1.2-3
##  [55] future_1.32.0          MASS_7.3-54            zoo_1.8-11
##  [58] scales_1.2.1           promises_1.2.0.1       spatstat.utils_3.0-3
##  [61] parallel_4.1.0         RColorBrewer_1.1-3     yaml_2.3.6
##  [64] reticulate_1.26        pbapply_1.7-0          gridExtra_2.3
##  [67] stringi_1.7.8          foreach_1.5.2          rlang_1.1.0
##  [70] pkgconfig_2.0.3        matrixStats_0.63.0     evaluate_0.21
##  [73] lattice_0.20-44        tensor_1.5             ROCR_1.0-11
##  [76] purrr_1.0.1            labeling_0.4.2         htmlwidgets_1.6.2
##  [79] cowplot_1.1.1          tidyselect_1.2.0       parallelly_1.36.0
##  [82] RcppAnnoy_0.0.20       magrittr_2.0.3         R6_2.5.1
```

```
##  [85] generics_0.1.3         withr_2.5.0            pillar_1.9.0
##  [88] fitdistrplus_1.1-11    abind_1.4-5            survival_3.2-11
##  [91] sp_1.5-1               tibble_3.2.1           future.apply_1.10.0
##  [94] KernSmooth_2.23-20     utf8_1.2.2             spatstat.geom_3.0-3
##  [97] plotly_4.10.1          rmarkdown_2.22         grid_4.1.0
## [100] data.table_1.14.6      FNN_1.1.3.2            digest_0.6.31
## [103] xtable_1.8-4           tidyr_1.3.0            httpuv_1.6.7
## [106] munsell_0.5.0          viridisLite_0.4.1
```