JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 29, Number 11, 2022

Pp. 1229–1232
DOI: 10.1089/cmb.2022.0251

UNIFAN: A Tool for Unsupervised Single-Cell Clustering and Annotation

DONGSHUNYI LI,[1] JUN DING,[2] and ZIV BAR-JOSEPH[1,3]

ABSTRACT

UNIFAN is an unsupervised cell type annotation tool for single-cell RNA sequencing data (scRNA-seq). Given single-cell expression data as input, UNIFAN outputs cell clusters as well as annotations for each cluster. The clustering process utilizes information on pathways and biological processes and these are also used to annotate the resulting clusters. In this software article, we focus on how to install UNIFAN and on the main steps involved in using UNIFAN for cell type annotations.

Keywords: cell annotation, cell type identification, clustering, gene expression.

1. INTRODUCTION

One of the first steps when analyzing single-cell RNA sequencing (scRNA-seq) data is cell type identification. In many cases wherein labeled data are not available, cell types are assigned by first clustering the cells and then assigning cell types to clusters based on the differentially expressed genes. Current methods cluster cells using only the provided expression data. Although this works well in some cases, noise and dropouts can significantly impact the outcome, leading to results that do not completely align with the correct cell type groupings.

To improve cell type identification, we developed an unsupervised cell type annotation method UNIFAN (Unsupervised Single-cell Functional Annotation) (Li et al, 2002) to simultaneously cluster and annotate cells using known gene sets. We showed that UNIFAN outperforms other methods for cell clustering and that the annotations make it much easier to assign correct cell types to the resulting clusters.

Here we will briefly introduce our software package for UNIFAN. We also provide detailed instructions and tutorials on our GitHub site (`https://github.com/doraadong/UNIFAN`).

[1]Computational Biology Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.
[2]Meakins-Christie Laboratories, Department of Medicine, McGill University Health Centre, Montreal, Canada.
[3]Machine Learning Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

2. INSTALLATION

UNIFAN is written in Python using PyTorch (Paszke et al, 2017) and supporting both CPU or GPU. First, install appropriate Pytorch version (≥1.9.0) at (https://pytorch.org/). Then, users can install UNIFAN by using:

(pip install git+https://github.com/doraadong/UNIFAN.git).

3. UNIFAN FUNCTIONS

Users may import UNIFAN as a package and use it in their code. Hereunder we briefly describe the main steps needed for training UNIFAN. Alternatively, users can use the command-line tools we provide to train the model. See (https://github.com/doraadong/UNIFAN#Command-line).

3.1. Input and preprocessing

UNIFAN takes AnnData (Wolf et al, 2018) files containing scRNA-seq gene expression data as input. Users can download the example *Tabula Muris* data https://figshare.com/ndownloader/files/24351086 (The Tabula Muris Consortium, 2018) and run the provided getExample.py script to preprocess the data using SCANPY (Wolf et al, 2018). In brief, given the count matrix, the script first filters out low-quality cells and genes and then normalizes each cell to 1e4 total read counts.

It then log-transforms the values, scales them to 0 mean and unit variance and clips them such that the maximum is 10. The script will also find the most variable genes selected using Seuratv3 (Stuart et al, 2019) implemented in SCANPY. The output from this script is processed gene expression data saved in AnnData containing only cells from the specified tissue.

UNIFAN uses gene sets for cell type annotations. We provide a complied set of gene sets from MSigDB (Subramanian et al, 2005) and Ernst et al (2007). Users can download these sets from our GitHub site.

3.2. Inferring gene set activity scores

Users need to set up a few variables before training. These include paths to the input/output and whether GPU is being used. All hyperparameters have default values set up. We briefly discuss how these values are selected in Section 4.

UNIFAN first infers the gene set activity scores for each single cell, using the gene expression values as input. In this step, it trains the following autoencoder-based model using the given gene sets:

model_gene_set=autoencoder(decoding_network='geneSet',...)

UNIFAN will then infer the gene set activity scores for each cell using this trained model.

3.3. Models pretraining and initialization for clustering

In this step, UNIFAN first pretrains the following autoencoder model using the gene expression values as input:

model_autoencoder=autoencoder(decoding_network='gaussian',...)

After pretraining, UNIFAN then infers the initial low-dimensional representation z_e for each cell. UNIFAN uses these low-dimensional representations to find the initial number of clusters and cell groupings using Leiden clustering (Traag et al, 2019).

UNIFAN then pretrains the annotator (classifier) using the inferred gene set activity scores as input:

model_classifier=classifier(...).

If users prefer to also use the most variable genes selected during preprocessing (the default option), these genes will also be used as input for the annotator. During pretraining, the annotator will treat the initialized clusters as true labels.

3.4. Clustering

Finally, UNIFAN trains the pretrained annotator together with the pretrained autoencoder (''model_autoencoder''). In each epoch, the annotator is trained by using the clustering results as the true labels for

cells. The output from the annotator $p(r)$ is in turn used to evaluate the annotator loss for the autoencoder. The annotator is optimized using its own loss function, separately from the autoencoder.

`model_annocluster=AnnoCluster(decoding_network='gaussian',...).`

If ground truth cell labels are available, UNIFAN will evaluate the clustering performance using adjusted Rand index and Normalized Mutual Information as implemented in scikit-learn (Pedregosa et al, 2011).

3.5. Find the annotations for each cluster

The trained UNIFAN model is saved in `annocluster_folder`. Annotations for each cluster can be retrieved from the annotator coefficients that are saved in `classifier_state_dict`. Since UNIFAN uses a logistic regression classifier for the annotator, the coefficients are saved in the field "`decoder.predictor.0.weight`." For each cluster, we have a vector of coefficients with the length equal to the feature size (for example, if using both gene set activity scores and the variable genes, then it equals to the number of gene sets plus the number of variable genes).

Given the names of the gene sets and the variable genes, which are saved as a separate file in previous steps (at `input_r_names_path`), users can look at the coefficients and find the highly weighted gene sets or genes, which are selected by the annotator as the most predictive features for a particular cluster.

4. HYPERPARAMETER TUNING

We employ a heuristic approach to automatically select the values for the hyperparameter γ during the training process, without using the ground truth cell labels. Except for γ, users may just use the default values for the other hyperparameters. As we stated in Li et al (2022), we conducted a grid search to select for the neural network configuration and some weighting hyperparameters for the loss functions using the *Tabula Muris* data set.

AUTHORS CONTRIBUTIONS

D.L., J.D., and Z.B.-J. developed the method; D.L. implemented the software and performed the analysis; and all authors analyzed the results and wrote the article.

ACKNOWLEDGMENT

The authors are very grateful to the RECOMB anonymous reviewers for their constructive comments.

SOFTWARE AVAILABILITY

UNIFAN is available under an MIT license at GitHub (`https://github.com/doraadong/UNIFAN`).

AUTHOR DISCLOSURE STATEMENT

The authors declare they have no conflicting financial interests.

FUNDING INFORMATION

This study was partially supported by NIH grants OT2OD026682, 1U54AG075931, and 1U24CA268108 to Z.B.-J.

REFERENCES

Ernst J, Vainas O, Harbison CT, et al. Reconstructing dynamic regulatory maps. Mol Syst Biol 2007;3(1):74; doi: 10.1038/msb4100115.

Li D, Ding J, Bar-Joseph Z. Unsupervised cell functional annotation for single-cell rna-seq. Genome Res 2022; doi: 10.1101/gr.276609.122.

Paszke A, Gross S, Chintala S, et al. Automatic differentiation in pytorch. 2017.

Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011; 12:2825–2830.

Stuart T, Butler A, Hoffman P, et al. Comprehensive integration of single-cell data. Cell 2019;177(7):1888–1902; doi: 10.1016/j.cell.2019.05.031.

Subramanian A, Tamayo P, Mootha VK, et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci U S A 2005;102(43):15545–15550; doi: 10.1073/pnas.0506580102.

The Tabula Muris Consortium. Single-cell transcriptomics of 20 mouse organs creates a tabula muris. Nature 2018; 562(7727):367–372; doi: 10.1038/s41586-018-0590–4.

Traag VA, Waltman L, Van Eck NJ. From louvain to leiden: Guaranteeing well-connected communities. Sci Rep 2019;9(1):1–12; doi: 10.1038/s41598-019-41695-z.

Wolf FA, Angerer P, Theis FJ. Scanpy: Large-scale single-cell gene expression data analysis. Genome Biol 2018;19(1): 1–5; doi: 10.1186/s13059-017-1382-0.

Address correspondence to:
Dr. Ziv Bar-Joseph
Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
USA

E-mail: zivbj@cs.cmu.edu