Optimizing Feature Selection in High-Dimensional RNA-seq Data

Zoe I. Barinaga[1,2], Dakota Y. Hawkins[3], Cynthia A. Bradham[4]

[1]Boston University Bioinformatics BRITE REU Program, Summer 2021
[2]Department of Biomedical Engineering, University of Florida, Gainesville, FL
[3]Boston University Bioinformatics Graduate Program, Boston, MA
[4]Department of Biology, Boston University, Boston, MA

Introduction and Objectives

Machine learning methods have become increasingly popular in discovering trends in biological or computational data. One algorithm, Neighborhood Component Feature Selection (NCFS), takes a dataset matrix with class labels and outputs the significance, or 'weights,' of features in determining the rules for classification. However, despite Python being a leading language for data analysis, a user-friendly Python implementation of NCFS does not exist. In this project, the NCFS algorithm was implemented in the JAX Python library using an unpublished Numba implementation as a template.

This project fit into the lab's overarching goal of quantifying how chemical perturbances affect sea urchin skeletal development; NCFS would be used in a pipeline to identify the most important genes in classifying cells into their appropriate cell types.

Project Goal: Implement NCFS algorithm using JAX Python library to efficiently identify relevant feature weights on simulated datasets

Advantages of JAX:

-automatic differentiation
-function vectorization
-computational efficiency

Example Dataset

Class Labels
Cell Class Alpha
Cell Class Alpha
Cell Class Beta
Cell Class Beta

Sample	Gene A	Gene B	Gene C
Cell 1	1	1	0
Cell 2	1	1	0
Cell 3	0	1	0
Cell 4	0	1	0

FEATURES

Figure 1. Example dataset demonstrating terminology and logic used. Two cell classes are shown, alpha and beta, and gene expression of corresponding cells are tabulated.

Methods

Develop NCFS code implementation in Python JAX using Numba code as template for logic.

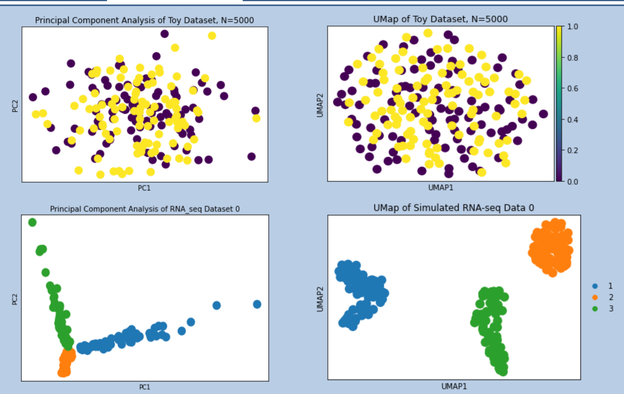
Numba
-translates Numba, NumPy to machine code
-specializes in 'for' loops

VS

JAX
-translates Numba, NumPy to machine code
-specializes in matrix and vector operations
-has auto-differentiation

-Toy datasets were made by creating arrays containing a user-specified number of samples and filling them with random numbers scaled between 0 and 1, and two features at specified indices were made to be significant with all others remaining irrelevant.

-RNA-seq datasets were made via simulation to consist of specified 'marker' genes whose expression determined cell classification; not as randomized



Toy dataset: 200 samples (2 populations), 5000 features

RNA-seq dataset: 200 samples (3 populations), 1000 features

Troubleshoot code with simulated datasets, calculate execution time, and test relative weaknesses/strengths of distance metrics.

Results

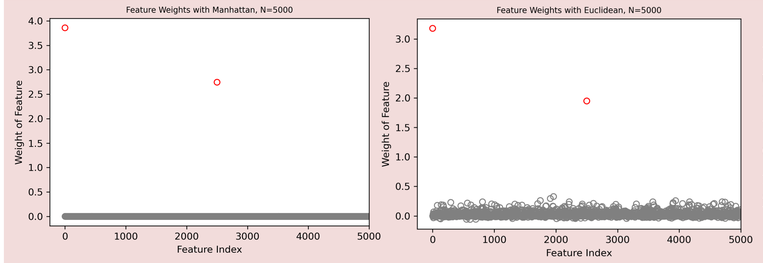
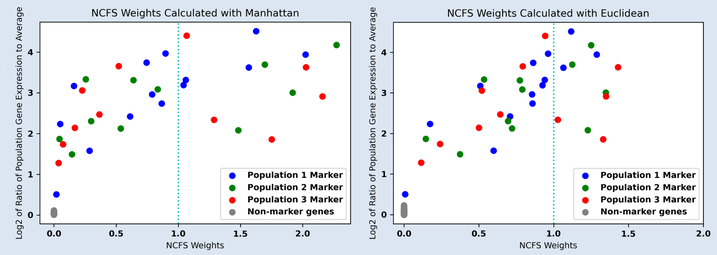


Figure 2. Plotted weights features/genes in simulated toy dataset of N=5000 features where 'Manhattan' and 'Euclidean' distance metrics were used. Only two features were specified as significant (w > 1): the first one and the 2500th one ($\frac{n}{2}$ th). The two relevant features are indicated by the red circles, while all other irrelevant features are indicated by the grey circles.

Figure 3. Plotted NCFS Feature Weights by Population Difference in Expression. This graph comes from feature weights calculated from simulated RNA-seq dataset 1, which contains three separate populations of cell types, when passed through the JAX implementation. The dataset was created with the intention that any given 'significant' (weight > 1) genes serve as a marker gene for only one population; therefore, the non-grey plotted significant features represent unique genes.



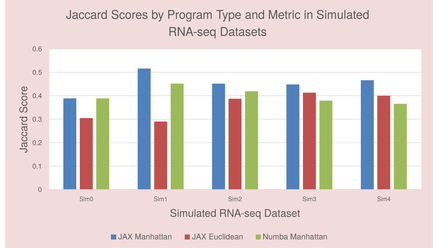


Figure 4. Jaccard Scores by Program Type and Distance Metric in Simulated RNA-seq Datasets. Jaccard scores of JAX Manhattan and JAX Euclidean were taken when loss iteration interval = 10.

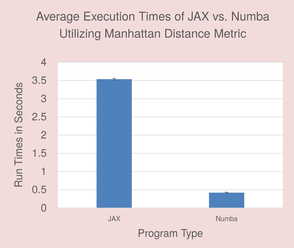


Figure 5. Bar graph of JAX and Numba Program Average Execution Times Using Manhattan Distance Metric. The run times are given in seconds and are averaged over fifty trials.

Conclusion

- NCFS algorithm can be implemented in JAX; JAX appears to have great potential for decreasing NCFS run time and auto-differentiating distance metrics
- Multiple distance metrics can be incorporated into JAX implementation. Autodifferentiation aspect of JAX was successfully utilized in code.
- Interval of iteration for calculating loss in JAX implementation can be adjusted; some intervals produce better results than others.
- Currently, JAX implementation takes approximately nine times the run time of the Numba implementation; however, JAX has the potential for GPU acceleration.

References

[4]Bradbury, J. et al. (2018). JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax.

[5]Harris, C. R. et al. (2020). "Array programming with NumPy." *Nature*, vol. 585, pp. 357-362.

[6]Lam, S. et al (2015). "Numba: A LLVM-Based Python JIT Compiler." *Association for Computing Machinery*, no. 7, pp. 1-6.

[7]VanderPlas, J. (2017). "Python Data Science Handbook by Jake VanderPlas." *O'Reilly*, 978-1-491-91205-8.

[8]Yang, W. et al. (2012). "Neighborhood Component Feature Selection for High-Dimensional Data." *Journal of Computers*, vol. 7, no. 1, pp. 161-168.

[9]Zayas, Carmen (2011). "Landscape structure in two reefs in La Parguera and the distribution of Lytechinus variegatus." *ResearchGate*, https://doi.org/ 10.13140/RG.2.1.2898.7283

Acknowledgements

This work was funded, in part, by NSF grand DBI-1949968, awarded to the Boston University Bioinformatics BRITE REU program, as well as NSF grant IOS 1656752 (CAB) to the Bradham Lab.