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 In this project, the NCFS algorithm was implemented in the JAX Python library using an unpublished Numba implementation as a template.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 perturbations 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

<table>
<thead>
<tr>
<th>Class Labels</th>
<th>Sample</th>
<th>Gene A</th>
<th>Gene B</th>
<th>Gene C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell Class Alpha</td>
<td>Cell 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cell Class Alpha</td>
<td>Cell 2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cell Class Beta</td>
<td>Cell 3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cell Class Beta</td>
<td>Cell 4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Features

Methods

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

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.

Results

<table>
<thead>
<tr>
<th>Numba</th>
<th>JAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>translates Numba, NumPy to machine code</td>
<td>translates Numba, NumPy to machine code</td>
</tr>
<tr>
<td>specializes in ‘for’ loops</td>
<td>has auto-differentiation</td>
</tr>
</tbody>
</table>

- 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.

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

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


Acknowledgements

This work was funded, in part, by NSF grant DBI-1949908, awarded to the Boston University Bioinformatics BRITE REU program, as well as NSF grant IOS 1606752 (CAB) to the Bradham Lab.