Cell Detection From X-Ray Microtomography Data

Spot those cells!

1 Overview

Each individual in the class will implement some cell detection code for finding cell bodies in X-ray microtomography data. This builds on ideas discussed in lecture as well as this paper `https://www.ncbi.nlm.nih.gov/pubmed/29085899`.

This data forms a cube with cells indicated by darker patches of the images. Other objects in the image include background tissue, blood vessels, and white matter.

Raw data and labeled training data are provided. The training data is processed so that a value of 1 indicates a cell is present at that voxel, and 0 indicates that voxel is not a cell.

You will write code to automate detecting cells from these data. The raw images are 3-D matrices (called tensors) consisting of integer image data from 0 to 255. The training data, referred to as X is broken into a training dataset X_{train} and testing dataset X_{test}. The training dataset is to be used for developing your algorithms.

Once you have developed your aglorithms, you apply the algorithm to the X_{test} to determine how you perform.

Your classifier will be a function $\hat{y} = f(X)$. This maps the data X to a tensor $\hat{y}$ of the same size, with values of 0 or 1. This is then compared to the true y_{test} to assess quality.

For assessing quality, the f1 metric is used, $f1 = 2pr/(p + r)$ where p is precision and r is recall. Code is already provided to compute this metric. You must do better than the provided threshold detector ($f1 = 0.25$).

2 Details

Starter code is available here: `https://github.com/circuitinstitute/intersession2020`. The program sourcetree can be used to access this git repository (you may use any git client you like though) `https://www.sourcetreeapp.com/`. Instructions to clone a repository can be found here `https://confluence.atlassian.com/get-started-with-sourcetree/clone-a-remote-repository-847359098.html`. You will write python code in a jupyter notebook (`https://jupyter.org/`). To make this easier, you can optionally use Google Colab, which does not require you to install software on your computer (`https://colab.research.google.com/`)

Annotated data (with labels) can be found here: `https://drive.google.com/open?id=1ONxQE82UxUOkQsxGKibVyATFY51ipU9h`. Raw data (the images) can be found here: `https://drive.google.com/open?id=1anSzjKnjIr9d3EcM5lj5BFH3asrvkjDY`.

If you are using colab, the notebook is already structured to allow you to upload files (`https://towardsdatascience.com/3-ways-to-load-csv-files-into-colab-7c14fcbdcb92` method number 2).

If you are familiar with python, data science, and/or machine learning, please feel free to experiment and attempt to produce a robust, high performing algorithm. This a great and practical challenge!

For those with limited python programming experience, I suggest you try to implement a simple histogram based segmentation algorithm: `https://scipy-lectures.org/advanced/image_processing/#segmentation` using the common scipy package.

3 Evaluation

- **code quality** (20 points) code is clean and well documented.
- **compute data statistics, display images** (30 points) You must complete code to display statistics about the data such as shape, min, and max values. You must also make example plots.
- **implement classifier** (35 points) You must implement a new classifier function $\hat{y} = f(X)$.
- **outperform threshold classifier** (15 points) You must outperform the provided classifier by getting a higher f1 score.
- **total** (100 points)

4 Submission Requirements

- Jupyter notebook with code and results

5 Submission Deadline

The proposals must be submitted to via Github no later than 15h20 on Friday, January 24th, 2020.