start_guide

October 10, 2017

1 Quick Start Guide

1.1 python and conda enviornment setup

The first step is to setup python and other packages required by the CaImAn package. Although python can run perfectly fine by itself, and you can install the packages one by one manually, it is advised to use Conda, a package manager to make everything easier. Conda provides two major benefits: 1. conda manage different set of packages with virtual enviornment, so you can install different version of python and other packages in isolated and self-contained enviornments and they won't interfere with each other. It's like having Windows XP, Windows 7 and Windows 10 all on your machine at the same time, each with its own set of software (MS office, PS etc.) 2. conda makes compling and installing packages easier, since it handles components that are not written in python properly. This is especially valuable when comparing to pip - python's own package manager.

If you are interested in what's going on with all these python, conda and pip stuff and why the complication, **here** is a link that helped me a lot

So here we go, go to the **homepage of CaImAn** and follow the installation guide, which should guide you through installing python and conda as well as setting up the enviornment. All those commands refer to the lines that should be put into some sort of "terminal" or "command line prompt". Here are some guides about how to find them on **Windows**, **MacOS**, and **Linux**.

Although you can simply copy and paste the lines from the CaImAn page, it's highly advised to go through a **quick conda tutorial** so that you understand what those lines are actually doing, which is essential for starting up and managing the enviornments.

One gotcha on python version: if you are using linux, note that caiman suggest python 3.5 instead of the newest version, as the time of writing, opencv (one of the required packages) does not have a python 3.6 build, so you might want to specify python version explicitly.

1.2 testing the enviornment

The best thing to check whether the installation was successful is to run through the demo ipython notebook provided by the Caiman package. An ipython notebook is a nice format designed for sharing reusable codes, where texts are written in line with codes and the results. If you followed the Caiman installation process, you should already have necessary packages to run a notebook.

Open up the terminal of your system, activate your virtual enviornment with the `source activate` commands (**list the enviornments** if you forgot the name of your enviornments) and type in `jupyter notebook` then hit Enter.

This should open up a browser window that looks like the following. This is the interface of ipython notebook. See the **official documents**, **a shorter start guide** or **a nice cheatsheet** for how to play with the ipython notebook.

Navigate to CaImAn folder on your local machine then open up demo_caiman_pipeline.ipynb

The information in Ipython Notebook are organized by "cells". There are two major types of cells: "code" which contatins codes that can be executed and corresponding outputs, and "markdown" which contains pure text and images. The best way to tell them apart is to look for "In []" and "Out []" sign at the beginning of the line, which signify the input and output of the code cell. If there is nothing at the beginning, then it's a markdown cell.

The idea of testing the enviornment is run through (use the "Cell" dropdown in the menu, if you skipped all the tutorial above) each code cell in this notebook and see if there is any error occured. The output of those lines might look messy and puzzling, but you can recognize an error easily by looking for a long dashed line, the words `Traceback (most recent call last)` and a few block of codes each with an arrow pointing to the errored line. Here is how it looks:

If there is no error than you are good to go! If any **ImportError** occured, then check to see if you have that package installed properly

1.3 choosing an editor or IDE

Now, the notebook is best used for sharing and demonstration only. Do actually edit and run codes productively, you need some editor or **IDE**. In theory you can edit python codes with your system text editor, and run them with a direct `python path_to_the_script.py` commands. But it's always nice to have everything integrated together like matlab, hence the need for IDE. CaImAn recommended **spyder**, which should already be in your enviornment. You can start spyder with `spyder` after activating your enviornment. My current personal favorite is **PyCharm**. However, I encouraged you to **look around** and find out the best fit for yourself.

```
In [ ]: from IPython.display import display, Image
        display(Image())
```