《机器学习与模式识别》

如何在 Anaconda JupyterLab 中调试代码

2020 年 11 月 21 日（初稿）

2022 年 03 月 25 日（更新）

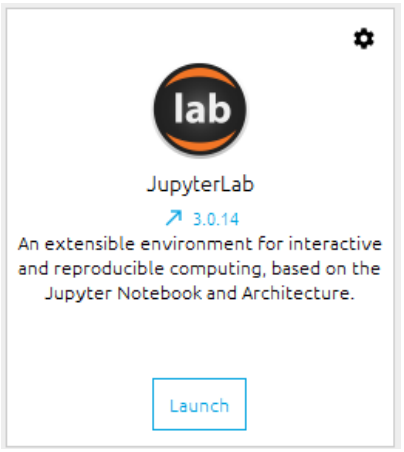
2022 年 09 月 01 日（更新）

前次建议用 Anaconda 的 Jupyter Notebook 跑代码，但是如果需要进行较多的代码调试与少量的代码修改（编写，复制-粘贴-改写），在 Anaconda JupyterLab 中就比较方便。下面简要说明。

建议使用 Anaconda 的的 2.2.0 版。更新的版本可能只能在 Windows 10 或 11 运行。

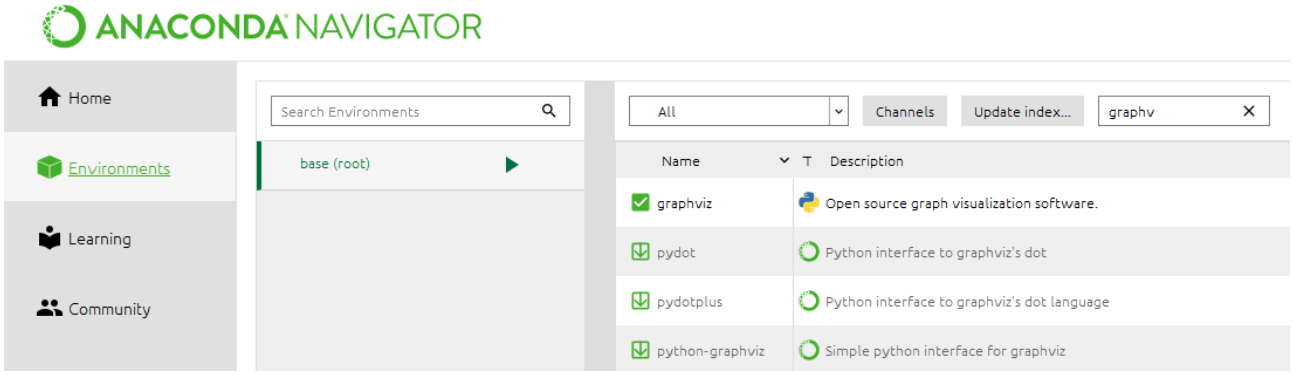
启动 JupyterLab

下载、安装 Anaconda。启动个人常用的浏览器。运行 Anaconda Navigator，点击 Home 页面中 JupyterLab 图标下的 Launch 按钮。正常情况下，你的浏览器中就会出现一个 JupyterLab 工作页。

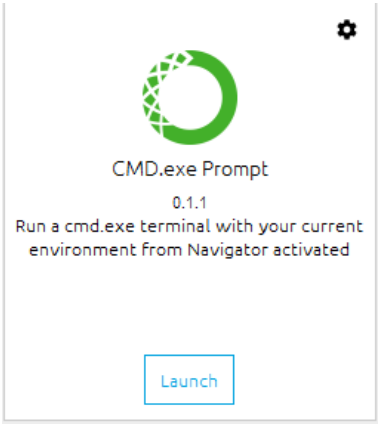


需要安装额外的工具包，有两种途径。

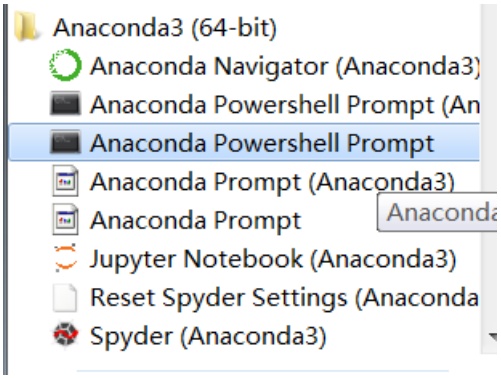
一种办法是，在 Anaconda 的 Environment 页面，选 All，输入希望安装或更新的工具包名称，Anaconda 会进行检索，并显示出来，勾选需要安装的包，并点击右下角的 Apply，Anaconda 即开始下载、安装。正常情况下，安装完成后，需要退出 Anaconda 以及在浏览器中打开的 Jupyter Notebook 或 JupyterLab，再全部依次重启。如果还是不能使用工具包，就重启计算机，这样安装好的包就能使用。



另一种办法是，在 Anaconda Navigator 的 Home 页面，点击 CMD.exe Prompt，

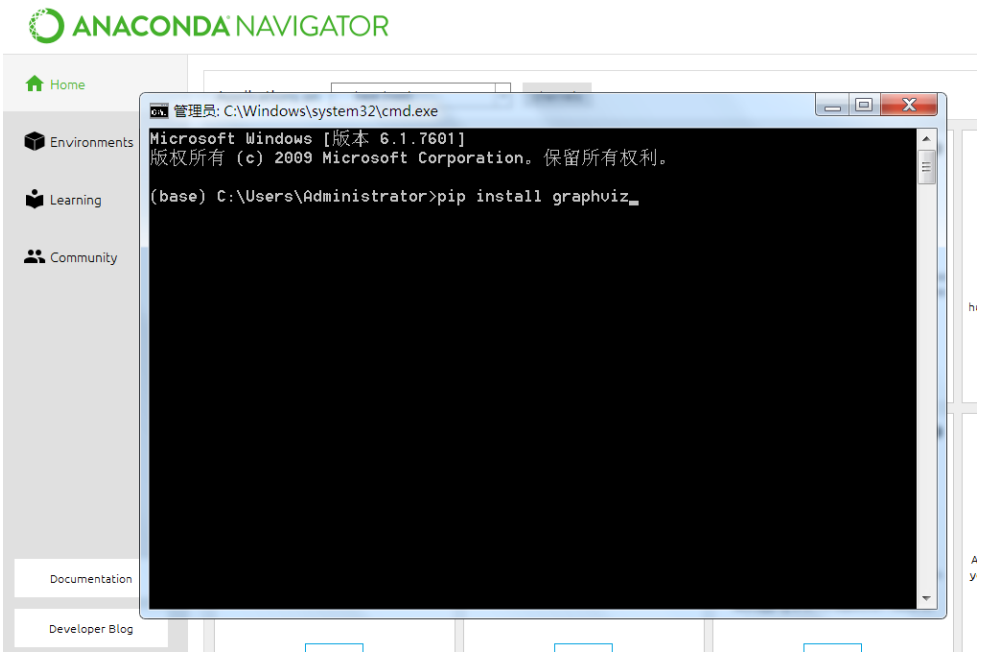


或者，为了能成功安装或更新代码库，不运行或关闭 Anaconda Navigator（运行后会占用某些库导致更新失败），而使用计算机左下角“开始”菜单中的 Anaconda3 目录下的 Anaconda Powershell Prompt：



出现一个 DOS 界面的命令行窗口。在其中执行命令：

```
pip  install  你需要安装的工具包名称
```



网络正常时，系统自动下载工具包（需要一些时间），安装完毕后有提示。如果工具包已经按照，系统会提示版本信息。

课程实验需要补充安装的一些包（其他缺失的工具包，类似安装）：

用前述命令行窗口，依次执行以下命令即可安装：

```
pip install graphviz

pip install scikit-plot

pip install cvxopt（凸规划库，部分代码中的 SVM 实现需要）

pip install autograd（计算多元函数梯度）
```

保持网络畅通，等待一段时间，系统自动下载安装，安装成功后的包名是 python-graphviz 和 scikit-plot。

为完整运行 Muller 书中的代码，需要安装 mglearn 包以及其他工具包：

```
pip install mglearn
```

注意，有时网络或服务器有问题导致安装失败，注意看窗口的提示信息。

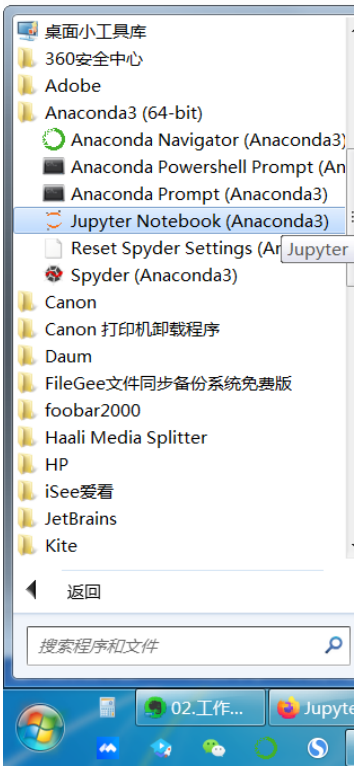
如何快速查看系统中到底安装了哪些 Python 工具包？

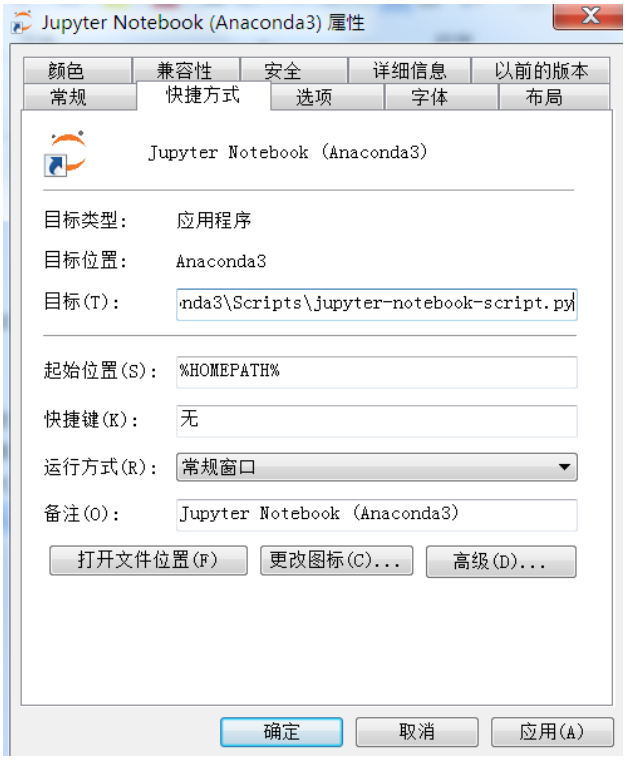
点击 Anaconda 的 CMD.exe Promt，在命令行对话框中输入下面的命令（中间的大于号两边有空格 > ），就可以在 D 盘根目录中的 list.txt 文件中，看到当前安装的各种 Python 包的版本信息。

```
conda list  >  D:\ list.txt
```

修改 Anaconda 中 JupyterLab 的缺省工作目录

在 桌面快捷方式，或找到 Windows 左下角的 开始按钮 - Anaconda-Jupyter Notbook，右键，选 属性，在弹出属性框的“快捷方式”中的目标栏目中，删掉最后面的 %USERPROFILE%。





在容量足够大的盘中创建 JupyterLab 或 JupyterNotebook 的工作文件夹（如：调试机器学习类代码工作文件夹），比如手工创建文件夹 F:\Jupyter，启动 Anaconda 的 CMD.exe Prompt 命令行窗口，输入：

```
jupyter notebook --generate-config
```

注意 generate 前面有两个短横杆，用一个短横杆与后面的 config 连接。Jupyter 与 notebook 之间，notebook 与 generate 之前的两个短横杠之间，都需要有空格。

然后回到 Windows 的资源管理器，到 C:\Users\Administrator\.jupyter 目录下，用记事本打开其中刚刚生成的 jupyter_notebook_config.py 文件，搜索找到其中的行：“The directory to use for notebooks and kernels”：

```
213  ## The directory to use for notebooks and kernels.
214  #c.NotebookApp.notebook_dir = ''
```

把上面第 2 行前面的注释符 # 去掉，并在 ’’ 中 填入刚刚创建的目录：F:\Jupyter：

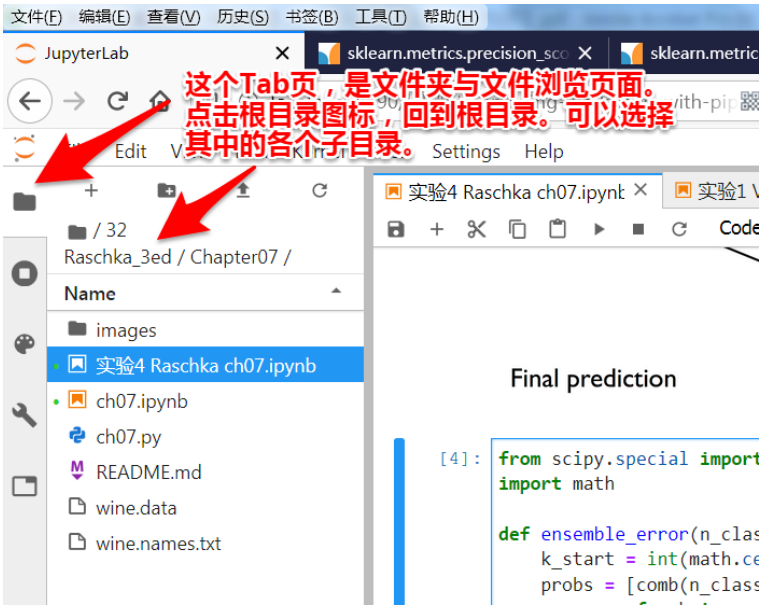
```
213  ## The directory to use for notebooks and kernels.
214  c.NotebookApp.notebook_dir = 'F:\Jupyter'
```

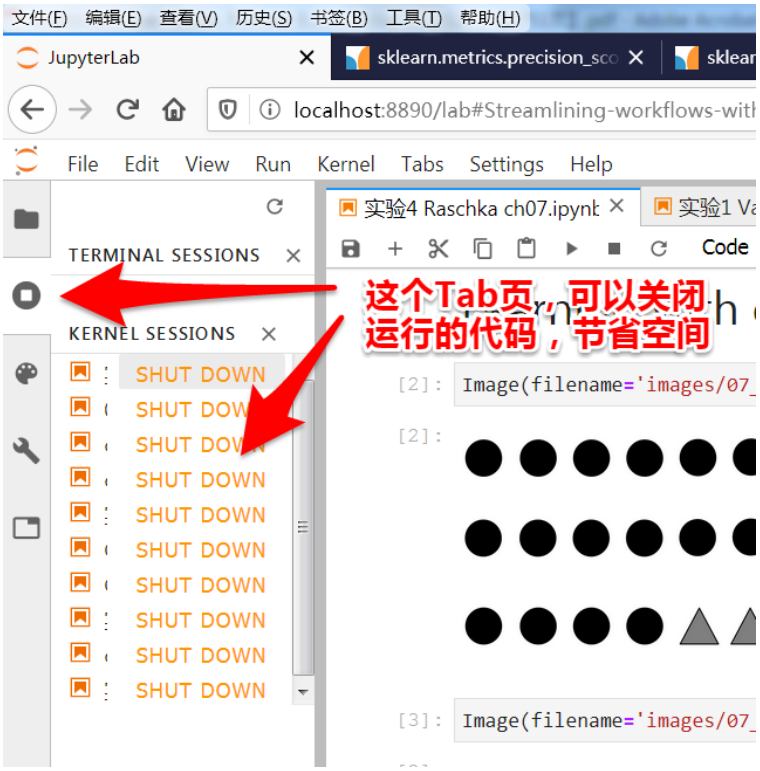
保存，退出。重启计算机。正常情况下，再通过 Anaconda 打开 JupyterLab，则 JupyterLab

的工作目录的根目录就是 F:\Jupyter。将需要调试的代码文件夹拷贝到根目录下即可。在 JupyterLab 左侧的文件浏览 Tab 栏即可打开看到其中的文件，双击即可调入 ipynb 代码运行。

打开浏览器（如火狐浏览器），再运行

如何在 JupyterLab 中调试 ipynb 代码？





Cell 有多种类型，一种是 Code，一种是 Markdown（文字）。

新建 Cell，即可输入代码或文字。

整体拷贝 Cell 也很简单：鼠标放在需要复制的 Cell 中任何位置，点击 JupyterLab 页面上面的复制按钮，然后，鼠标移到需要粘贴的位置的上一个 Cell 上，点击 JupyterLab 页面上面

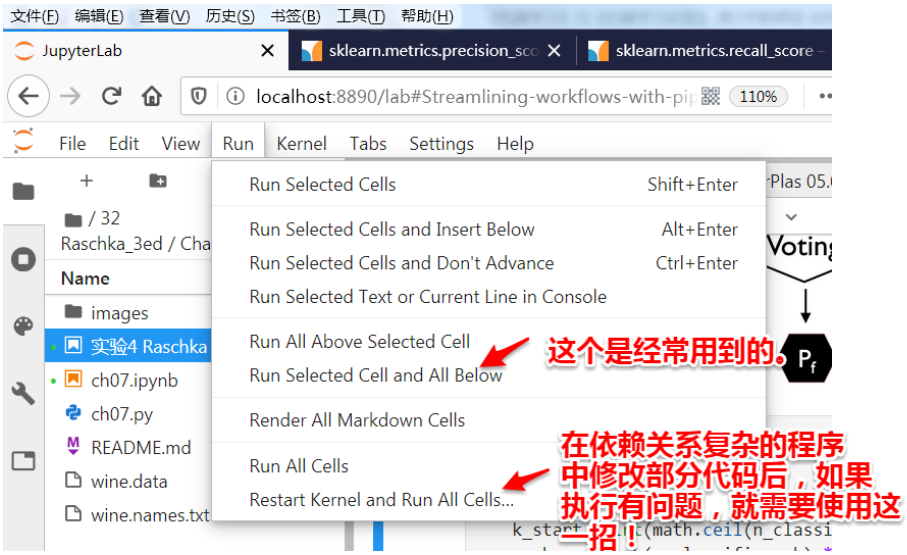
的粘贴按钮，整个 Cell 即被复制到当前 Cell 的下方。

删除一个 Cell 也很容易。点击 JupyterLab 页面上面的剪切按钮，但是不进行粘贴。这样就可以连续删除多个不需要的 Cell。

调试代码。很多时候，可能修改某个 Cell 中的数据或代码后，再次运行本 Cell 即可（点击 JupyterLab 页面上面的单步执行按钮，向右的三角形符号）。

如果某个 Cell 的每次执行，都影响了这个 Cell 的初始执行环境，这时即使不修改这个 Cell 中任何内容，Cell 的再次执行，其结果也会不同，这时就需要尽可能上溯到合适的上面的 Cell（这个需要根据代码具体情况而定），从这个 Cell 开始单步执行，或者从这个 Cell 开始，用 JupyterLab 的菜单栏中的 Run 中的 Run Selected Cell and All Below 命令执行（见截图），这时就既可以避免前面说的问题，又能不断修改某个 Cell。

JupyterLab 菜单栏中的 Run 菜单，里面有很多选项（见图），一般用到其中的两项即可：



如果程序调试多次后开始出现异常（一般是内部变量的值多次修改，且即位保存，多次运行就导致有副作用的交叉使用错误），就需要用 Run 菜单中的最后一项：Restart Kernel and Run All Cells（停止当前解释器核心，重新执行当前页面 ipynb 代码的全部 Cell）。

关于随机数的备注：对有些代码，如果希望在每次全面执行时，都能保持基本一致的结果（比如涉及重要算法改进的实验结果的论文，为了保证结果的可重现性，就需要这样的自我保护措施——很多优化算法的实际结果，往往与其中需要彩印的随机数有关，有时结果差异还较大，虽然对算法解决问题的有效性没有重大影响），就需要对函数中的 random_state 参数（或 RandomState 中的 seed 参数）进行设置。一般设置为 0 或 None 表示每次执行可以产生的任意随机数；如果设置为非零整数，则随机数发生器每次会按照相同的种子生产完全相同的随机数序列（随机数发生器生产的就是伪随机序列，不同的种子产生的下一个随机数其实一点都不随机，是完全确定的，只是总体看序列是随机的）。随机数在算法中的使用方式，则视具体算法而异，必要时需要翻阅手册说明。