Anaconda 及 Jupyter notebook 入门

本文由伍宏忠整理于2018/11/28.

Anaconda 是什么?

Anaconda (https://anaconda.org/) 是一个包含数据科学常用包的 Python 发行版本。它基于 conda ——一个包和环境管理器——衍生而来。用户可以使用 conda 创建环境，以便分隔使用不同 Python 版本和不同程序包的项目。你还将使用它在环境中安装、卸载和更新包。通过使用 Anaconda，处理数据的过程将更加愉快。

Jupyter notebook 是什么?

Jupyter notebook (http://jupyter.org/) 是一种 Web 文档，能让你将文本、图像和代码全部组合到一个文档中。它事实上已经成为数据分析的标准环境。 Jupyter notebook 源自 2011 年的 IPython 项目，之后迅速流行起来。

为什么要安装 Anaconda ?

你可能已经安装了 Python，并且想知道为何还需要 Anaconda。首先， Anaconda 附带了一大批常用数据科学包，因此你可以立即开始处理数据。其次，使用 conda 来**管理包和环境**能减少将来在处理数据过程中使用到的各种库与版本时遇到的问题。

- 包管理器用于在计算机上安装库和其他软件。你可能已经熟悉 pip，它是 Python 库的默认包管理器。 conda 与 pip 相似，不同之处是可用的包以数据科学包为主，而 pip 适合一般用途。与此同时，conda 并非像 pip 那样专门适用于 Python，它也可以安装非 Python 的包。它是支持任何软件的包管理器。也就是说，虽然并非所有的 Python 库都能通过 Anaconda 发行版和 conda 获得，但同时它也支持非 Python 库的获得。在使用 conda 的同时，你仍可以使用 pip 来安装包。

- 除了管理包之外，conda 还是虚拟环境管理器。它类似于另外两个很流行的环境管理器，即 virtualenv (https://virtualenv.pypa.io/en/stable/) 和 pyenv (https://github.com/pyenv/pyenv)。用户可以使用 conda 环境管理器分隔不同项目的包，且常常要使用依赖于某个库的不同版本的代码。例如，你的代码可能使

用了 Numpy 中的新功能，或者使用了已删除的旧功能。实际上，不可能同时安装两个 Numpy 版本。你要做的应该是，为每个 Numpy 版本创建一个环境，然后在项目的对应环境中工作。在应对 Python 2 和 Python 3 时，此问题也会常常发生。你可能会使用在 Python 3 中不能运行的旧代码，以及在 Python 2 中不能运行的新代码。同时安装两个版本可能会造成许多混乱和错误，而创建独立的环境会好很多。用户还可以将环境中的包列表导出为文件，然后将该文件与代码打包在一起。这能让其他人轻松加载代码的所有依赖项。 pip 提供了类似的功能，即

```
    $: pip freeze > requirements.txt
```

安装 Anaconda

Anaconda 可用于 Windows、Mac OS X 和 Linux。可以在 https://www.anaconda.com/download/ (https://www.anaconda.com/download/) 上找到安装程序和安装说明。

如果计算机上已经安装了 Python，这不会有任何影响。实际上，脚本和程序使用的默认 Python 是 Anaconda 附带的 Python。

选择 Python 3.6 版本（你也可以根据具体的需要选择 Python 2 的版本）。此外，如果是 64 位操作系统，则选择 64 位安装程序，否则选择 32 位安装程序。选择下载合适的版本，并继续进行安装！

完成安装后，会自动进入默认的 conda 环境，而且所有包均已安装完毕。可以在终端或命令提示符中键入 `conda list` ，以查看你安装的内容。安装完成之后，推荐在默认环境下更新所有的包。打开 Windows 下的 Anaconda Prompt 或者 Mac 下的终端，键入：

```
    $: conda upgrade --all
```

并在提示是否更新的时候输入 y(Yes) 以便让更新继续。初次安装下的软件包版本一般都比较老旧，因此提前更新可以避免未来不必要的问题。

另外，强烈建议添加国内源，如中科大或者清华的 Anaconda 源。具体方法为：

```
    $: conda config --add channels
https://mirrors.ustc.edu.cn/anaconda/pkgs/free/
    $: conda config --set show_channel_urls yes
```

有时安装软件包的时候还会连接官方源，但是网络不好，可用下面命令将官方源移除：

```
$: conda config --remove channels defaults
```

如果想查看源项，可用命令：

```
$: conda config --show
```

管理包

安装了 Anaconda 之后，管理包是相当简单的。要安装包，请在终端中键入 `conda install package_name` 。例如，要安装 numpy，请键入 `conda install numpy` 。

你还可以同时安装多个包。类似 `conda install numpy scipy pandas` 的命令会同时安装所有这些包。还可以通过添加版本号（例如 `conda install numpy=1.10` ）来指定所需的包版本。

Conda 还会自动为你安装依赖项。例如，scipy 依赖于 numpy，因为它使用并需要 numpy。如果你只安装 scipy (`conda install scipy`)，则 conda 还会安装 numpy（如果尚未安装的话）。

大多数命令都是很直观的。要卸载包，请使用 `conda remove package_name` 。要更新包，请使用 `conda update package_name` 。如果想更新环境中的所有包（这样做常常很有用），请使用 `conda update --all` 。最后，要列出已安装的包，请使用前面提过的 `conda list` 。

如果不知道要找的包的确切名称，可以尝试使用 `conda search search_term` 进行搜索。例如，我知道我想安装 Beautiful Soup，但我不清楚确切的包名称。因此，尝试执行 `conda search beautifulsoup` 。

管理环境

用户可以使用 conda 创建环境以隔离项目。要创建环境，请在终端中使用 `conda create -n env_name list of packages` 。在这里， `-n env_name` 设置环境的名称（-n 是指名称），而 `list of packages` 是要安装在环境中的包的列表。例如，要创建名为 my_env 的环境并在其中安装 numpy，请键入 `conda create -n my_env numpy` 。

创建环境时，可以指定要安装在环境中的 Python 版本。这在你同时使用 Python 2.x 和 Python 3.x 中的代码时很有用。要创建具有特定 Python 版本的环境，请键入类似于 `conda create -n py3 python=3` 或 `conda create -n py2 python=2` 的命令。

创建了环境后，在 OSX/Linux 上使用 `source activate my_env` 进入环境。在 Windows 上，请使用 `activate my_env` 。

进入环境后，你会在终端提示符中看到环境名称，它类似于 `(my_env) ~ $:` 。环境中只安装了几个默认的包，以及你在创建它时安装的包。你可以使用 conda list 检查这一点。在环境中安装包的命令与前面一样： `conda install package_name` 。不过，这次你安装的特定包仅在你进入环境后才可用。要离开环境，请键入 `source deactivate` （在 OSX/Linux 上）。在 Windows 上，请使用 `deactivate` 。

保存和加载环境

共享环境这项功能确实很有用，它能让其他人安装你的代码中使用的所有包，并确保这些包的版本正确。你可以使用 `conda env export > environment.yaml` 将包保存为 YAML。命令的第一部分 `conda env export` 用于输出环境中的所有包的名称（包括 Python 版本）。导出命令的第二部分 `> environment.yaml` 将导出的文本写入到 YAML 文件 environment.yaml 中。现在可以共享此文件，而且其他人能够用于创建和你项目相同的环境。

要通过环境文件创建环境，请使用 `conda env create -f environment.yaml` 。这会创建一个新环境，而且它具有同样的在 environment.yaml 中列出的库。

列出环境

如果忘记了环境的名称，可以使用 `conda env list` 列出你创建的所有环境。你会看到环境的列表，而且你当前所在环境的旁边会有一个星号。默认的环境（即当你不在选定环境中时使用的环境）名为 base(或root)。

删除环境

如果你不再使用某些环境，可以使用 `conda env remove -n env_name` 删除指定的环境（在这里环境名为 env_name）。

共享环境

在 GitHub 上共享代码时，最好同样创建环境文件并将其包括在代码库中。这能让其他人更轻松地安装你的代码的所有依赖项。对于不使用 conda 的用户，通常可以使用 `pip freeze` （在此处了解详情

(https://pip.pypa.io/en/stable/reference/pip_freeze/)）将一个 pip `requirements.txt` 文件导出并包括在其中。

了解更多信息

要详细了解 conda 以及它如何融入到 Python 生态系统中，请查看这篇由 Jake Vanderplas 撰写的文章：Conda myths and misconceptions (https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/)（有关 conda 的迷思和误解）。此外，如果你有多余精力，也可以参考这篇 conda (https://conda.io/docs/using/index.html) 文档。

在 Anaconda 环境下安装 Jupyter notebook

打开命令行，执行命令：

```
    (your_env)$: conda install jupyter notebook
```

然后在命令行输入 `jupyter notebook`，你的系统所对应的默认浏览器将会启动，此时 Jupyter notebook 便在运行状态。

设置 Jupyter notebook 可被远程访问

1. 生成一个 notebook 配置文件

默认情况下，配置文件 ~/.jupyter/jupyter_notebook_config.py 并不存在，需要自行创建。使用下列命令生成配置文件：

```
    $: jupyter notebook --generate-config
```

如果是 root 用户执行上面的命令，会发生一个问题：

```
    $: Running as root is not recommended. Use --allow-root to
bypass.
```

提示信息很明显，root 用户执行时需要加上 –allow-root 选项。

```
    $: jupyter notebook --generate-config --allow-config
```

执行成功后，会出现下面的信息：

```
    $: Writing default config to:
/root/.jupyter/jupyter_notebook_config.py
```

1. 生成密码

- 自动生成 从 jupyter notebook 5.0 版本开始，提供了一个命令来设置密码： `jupyter notebook password` ，生成的密码存储在 jupyter_notebook_config.json。

```
    $: jupyter notebook password
    Enter password:  ****
    Verify password: ****
    [NotebookPasswordApp] Wrote hashed password to
/Users/you/.jupyter/jupyter_notebook_config.json
```

- 手动生成 除了使用提供的命令，也可以通过手动安装，打开 `ipython` 执行下面内容：

```
    In [1]: from notebook.auth import passwd
    In [2]: passwd()
    Enter password:
    Verify password:
    Out[2]:
'sha1:67c9e60bb8b6:9ffede0825894254b2e042ea597d771089e11aed'
```

`sha1:67c9e60bb8b6:9ffede0825894254b2e042ea597d771089e11aed` 这一串就是要在 jupyter_notebook_config.py 添加的密码。

1. 修改配置文件 在 jupyter_notebook_config.py 中找到下面的行，取消注释并修改。

```
    c.NotebookApp.ip='*'
    c.NotebookApp.password = u'sha:ce...刚才复制的那个密文'
    c.NotebookApp.open_browser = False #表示不打开本地浏览器
    c.NotebookApp.port =8888 #可自行指定一个端口, 访问时使用该端口
```

以上设置完以后就可以在服务器上启动 jupyter notebook， `jupyter notebook` , root 用户使用 `jupyter notebook --allow-root` 。打开 `IP:指定的端口` , 输入密码就可以访问了。

需要注意的是不能在隐藏目录 (以 . 开头的目录)下启动 jupyter notebook, 否则无法正常访问文件。