

PROGRAMMING IN PYTHON II

Version Control and Collaboration – Git



Michael Widrich
Institute for Machine Learning

Copyright statement:

This material, no matter whether in printed or electronic form, may be used for personal and non-commercial educational use only. Any reproduction of this material, no matter whether as a whole or in parts, no matter whether in printed or in electronic form, requires explicit prior acceptance of the authors.

Outline

1. Motivation

2. Git

3. Github and Gitlab

4. Tasks

Motivation

- Typically your ML project will contain code, data, and reports
- We want to keep track of our project history ([Version Control](#))
 1. Reproducibility
 2. Reusability
 3. Bug-fixing
 4. Version conflicts, different published versions
- We want to be able to collaborate (or let others use our code)
 1. Common interface and tools for communication
 2. Modifications
 3. Different versions

■ Git

- ☐ Efficient tracking of directory contents
- ☐ Independent of types of files/data
- ☐ Suitable for version control and collaboration
- ☐ Supports distributed, non-linear workflows
- ☐ Commonly used in ML
- ☐ Large datasets usually not in Git or version control

■ We will not go into details but look at the basics of how to use Git

■ Tech Talk: Linus Torvalds on git

<https://www.youtube.com/watch?v=4XpnKHJAok8>

How to use Git? – Theory (1)

- Create a Git **repository** on your machine
 - Create a new repository
 - Clone/Fork an existing repository
- Create a file locally and **add** it to git index
 - The files are now **staged** and can be committed
- **Commit** new/modified files to the Git repository
 - The committed file states are now saved in the Git repository
 - Each commit includes a commit message
- You can now access the history of the Git repository (=previous commits)
- You can create **branches** for different versions and collaboration

How to use Git? – Theory (2)

- You can interact with other repositories (e.g. remote repositories on a server)
 - **pull** get latest versions of files from other repository
 - **push** push version of your repository to another repository
 - **merge** get latest versions of files from other repository and combine with latest file versions in your repository

How to use Git? – Practice

■ Install Git

- <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

■ Setup Git using `git config`

- <https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup>

■ Git is command-line but many GUIs exist

- E.g. PyCharm git integration

■ Literature and command-line Git guide:

- <https://git-scm.com/book/en/v2>
- <https://rogerdudler.github.io/git-guide/>

How to use Git? – PyCharm

- We will see how to use the git integration in PyCharm at the end of this lesson

Github and Gitlab (1)

- In ML you will most likely have to use Github or Gitlab at some point
- `https://github.com`
 - ☐ Platform that hosts Git projects
 - ☐ Many features for project/task management
 - ☐ Free accounts (for public and private projects)
 - ☐ Many open-source projects
 - ☐ Over 100 million repositories¹
 - ☐ Example: `https://github.com/git/git`

¹<https://venturebeat.com/2018/11/08/github-passes-100-million-repositories/>

Github and Gitlab (2)

- On Github/Gitlab you may create a **fork** of an existing repository
 - ☐ Creates a copy of the repository on your account
 - ☐ Can be modified by you
 - ☐ Can be pushed to the parent repository using a **pull request**
- Github/Gitlab provide management for **issues**
 - ☐ Can be tagged and assigned to users
 - ☐ If you create an issue, always make sure to do it properly or you will get ignored/banned
 - Check if there are similar issues already!
 - Do not hijack, stay on-topic!
 - Provide input/output/exact descriptions of the issue!
 - Read the rules if specified!
 - You want something from someone, not the other way around!

Tasks

1. Look up Git readme to find what "git" stands for
2. Install and configure Git
3. Activate Git intergration for a PyCharm project
 - ☐ Perform a [commit](#)
 - ☐ Perform a second [commit](#)
 - ☐ Create a new [branch](#) and commit it
 - ☐ Take a look at the Git [Log](#)
 - ☐ Perform a [checkout](#) of an old version
4. [Clone](#) a github project using PyCharm
 - ☐ Take a look at the [Log](#)
 - ☐ Modify a file and commit it to your local clone of the github repository
5. If you have a github account:
 - ☐ [Fork](#) a github project to your account