Midterm Presentation of Collaborative Software Development

Team C members: Xianghang Zhang, Hakan Yeniavci and Tong Liu
Presenter: Tong Liu, Xianghang Zhang

20.12.2021

Outline

1. Git
2. Technical details
3. Test driven development
4. Continuous integration deplotment
5. An example

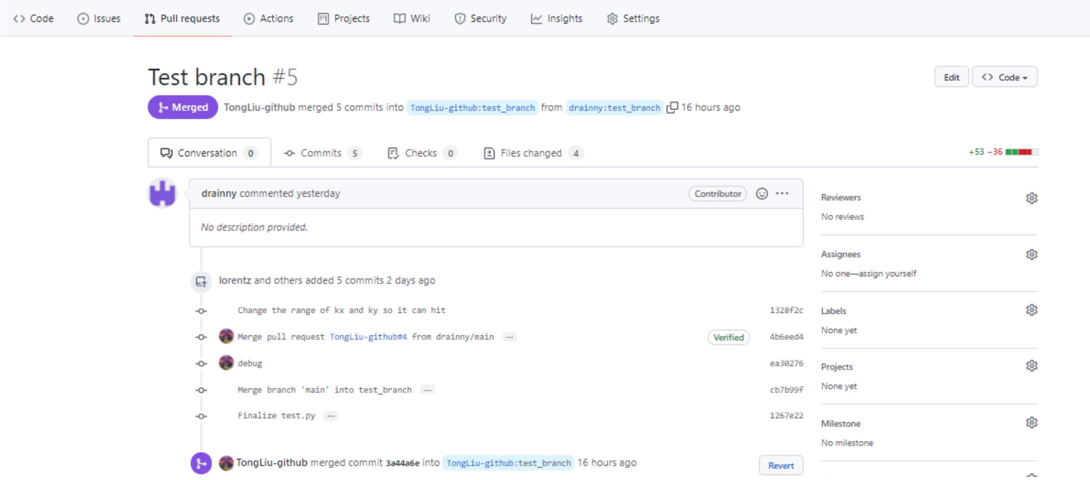
Basic use of git

- clone/push/pull/add/commit/…

```
(base) PS C:\Users\Administrator\Desktop\courses\csd> git clone https://github.com/TongLiu-github/csd_project.git
Cloning into 'csd_project'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 83 (delta 17), reused 14 (delta 5), pack-reused 45
Unpacking objects: 100% (83/83), 21.83 KiB | 29.00 KiB/s, done.
(base) PS C:\Users\Administrator\Desktop\courses\csd> cd .\csd_project\
(base) PS C:\Users\Administrator\Desktop\courses\csd\csd_project> git .\slide.pptx
git: '.\slide.pptx' is not a git command. See 'git --help'.
(base) PS C:\Users\Administrator\Desktop\courses\csd\csd_project> git add .\slide.pptx
```

```
(base) PS C:\Users\Administrator\Desktop\courses\csd\csd_project> git commit -m "add slide"
[main 3e9eada] add slide
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 slide.pptx
(base) PS C:\Users\Administrator\Desktop\courses\csd\csd_project> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 261.61 KiB | 9.69 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/TongLiu-github/csd_project.git
   0ef90e9..3e9eada  main -> main
```

- Pull/merge request



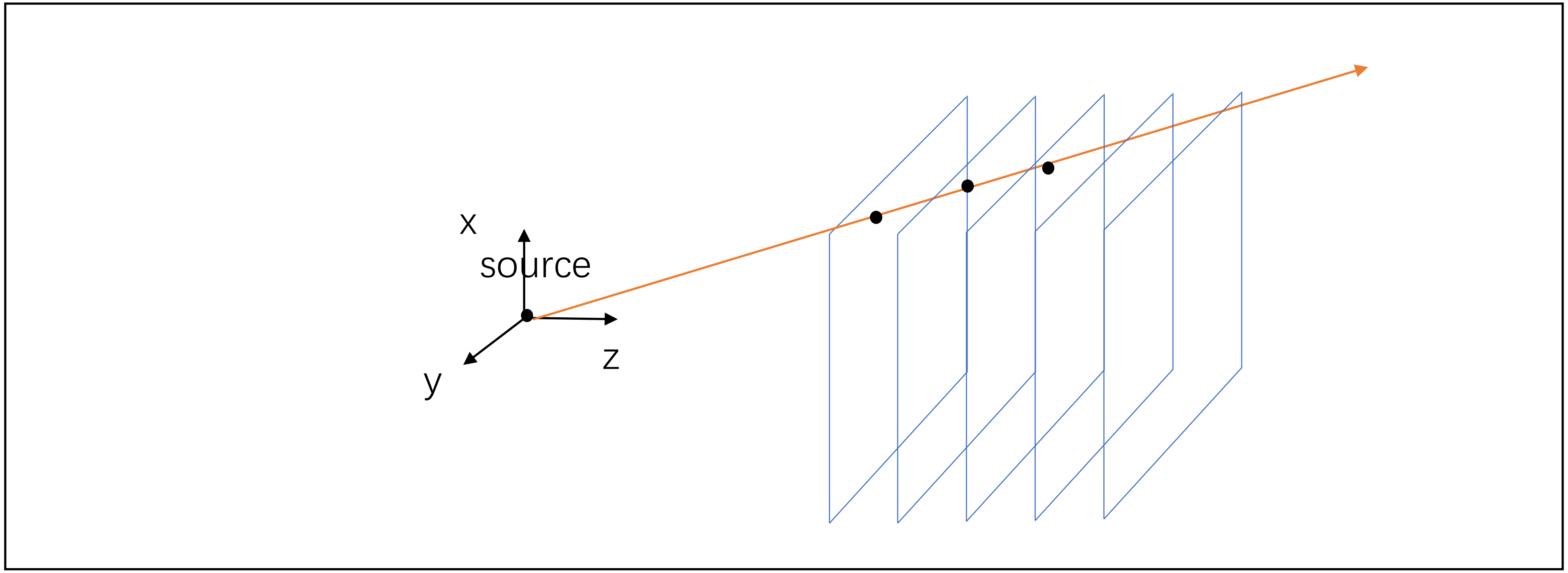
- Commit History example

```
* 2d5c0ff a first version of codes
*   1f59ae7 Merge pull request #2 from drainny/main
|\
| * ae023cb Defined a class of trajectory
| *   9fc8a41 Merge branch 'TongLiu-github:main' into main
| |\
| |/
|/
*   2ff13d2 add codes for xyz
*   ef43bd6 Merge pull request #1 from drainny/main
|\ \
| | * 1720716 Defined class of trajectory Wrote a generator for a trajectory Added a simple test
| |/
| * 37dda5c The first edit
|/
```

Outline

1. Git
2. Technical details
3. Test driven development
4. Continuous integration deplotment
5. An example

Problem review



- Parametric formulation:

A line in 3D traveling in the direction (a, b, c) can be described by:

$x = x_0 + ta$

$y = y_0 + tb$

$z = z_0 + tc.$

Suppose x_0 = y_0 = z_0 = 0, where it is the source place. Suppose $c = 1$. Then it becomes:

$x = ta$

$y = tb$

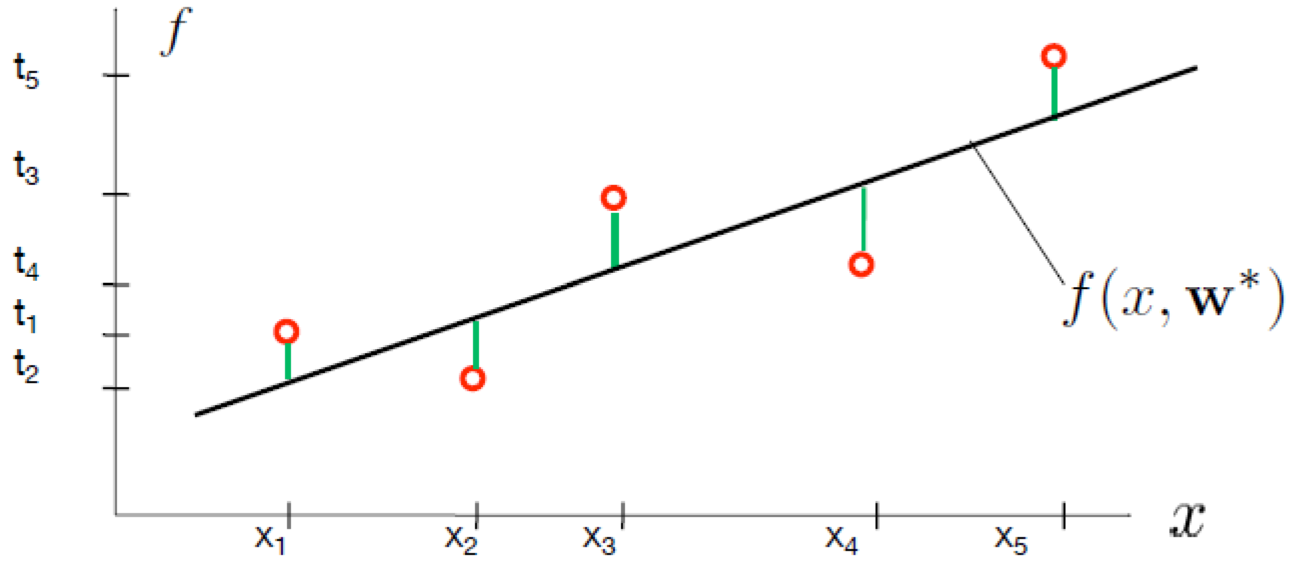
$z = t.$

- Source at least hits one sensor:

```
self.kx = r.uniform(-0.5, 0.5)
self.ky = r.uniform(-0.5, 0.5)
```

Basic: linear regression

- Parametric formulation:



- Assume: $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$
- Given: data points (x_1, t_1), (x_2, t_2), …
- Goal: predict the value *t* of a new example *x*
- Parametric formulation: $f(x, \mathbf{w}) = w_0 + w_1 x$

- To determine a function *f*, we need an error function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (f(x_i, \mathbf{w}) - t_i)^2$$

- We search for parameters w^*, s.t. $E(w^*)$ is minimal:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N} (f(x_i, \mathbf{w}) - t_i) \nabla f(x_i, \mathbf{w}) \doteq (0 \quad 0)$$

Basic: linear regression

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N} (f(x_i, \mathbf{w}) - t_i) \nabla f(x_i, \mathbf{w}) \doteq (0 \quad 0)$$

$$f(x, \mathbf{w}) = w_0 + w_1 x \quad \Rightarrow \quad \nabla f(x_i, \mathbf{w}) = (1 \quad x_i)$$

Using vector notation: $\mathbf{x}_i = (1 \quad x_i)^T \Rightarrow f(x_i, \mathbf{w}) = \mathbf{w}^T \mathbf{x}_i$

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N} \mathbf{w}^T \mathbf{x}_i \mathbf{x}_i^T - \sum_{i=1}^{N} t_i \mathbf{x}_i^T = (0 \quad 0) \Rightarrow \mathbf{w}^T \underbrace{\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T}_{A^T A} = \underbrace{\sum_{i=1}^{N} t_i \mathbf{x}_i^T}_{A^T T}$$

$$A = \begin{bmatrix} 1 & x_1 \\ \dots & \dots \\ 1 & x_N \end{bmatrix}$$

Programming:

```python
def linear_regression(self):
    """
    Linear regression.
    """
    A = np.vstack([np.ones(len(self.z_n)), self.z_n]).T
    y = np.array(self.y_n)[:, np.newaxis]
    x = np.array(self.x_n)[:, np.newaxis]
    self.ky = np.dot((np.dot(np.linalg.inv(np.dot(A.T, A)), A.T)), y)[0][0]
    self.kx = np.dot((np.dot(np.linalg.inv(np.dot(A.T, A)), A.T)), x)[0][0]
```

Total codes

- generator.py:

```python
class Generator:
    """A 3-dimensional line created by particle source

    Line is parametrized by  x = kx * z + bx; y = ky * z + by
    """
    def __init__(self, rsl):...

    def generate(self):...

    def hitPoints(self):...

    def observe(self):...
```

- main.py:

```python
def isOk(traj, res, error = 0.1):...

for i in range(0, 1):
    generator = Generator(rsl = 2.5e-7)
    x_n_max, x_n_min, y_n_max, y_n_min = generator.observe()
    print(x_n_min, y_n_min)
    error = 1e-5
    calculate = fit(x_n_max, x_n_min, y_n_max, y_n_min)
    calculate.compute_observed()
    calculate.linear_regression()
    isOk(generator, calculate, error)
```

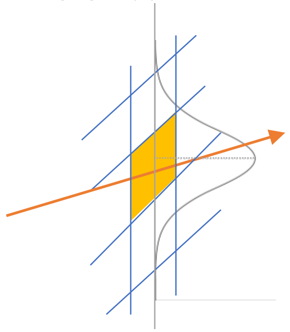
- fitting.py:

```python
class fit:
    """
    Linear regression with minimal sum of squared errors.
    """
    def __init__(self, x_n_max, x_n_min, y_n_max, y_n_min):...

    def compute_observed(self):...

    def linear_regression(self):...
```

- test part: …

The problem from a different view

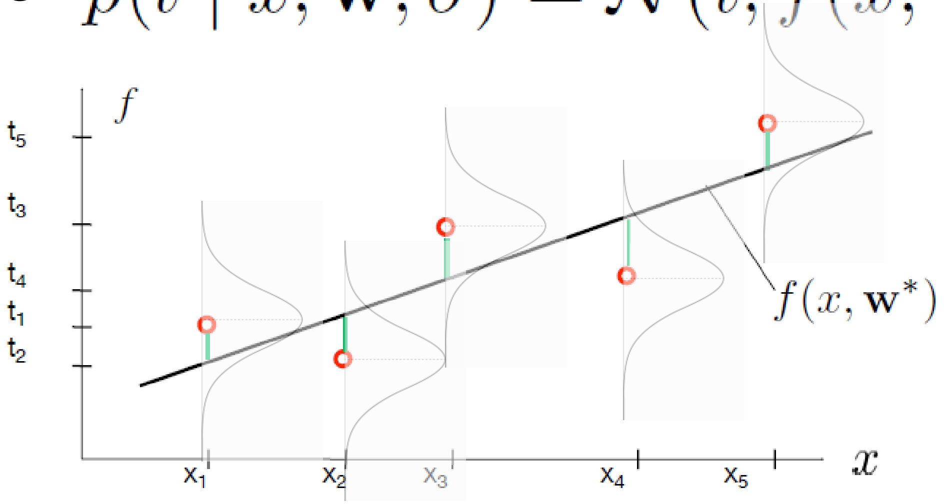
- We don't know the precise place within a resolution: assume a Gaussian distribution.



- Linear regression: assume that *y* is affected by Gaussian noise :

$$t = f(x, \mathbf{w}) + \epsilon \quad \text{where} \quad \epsilon \rightsquigarrow \mathcal{N}(.; 0, \sigma^2)$$

Thus, we have $p(t \mid x, \mathbf{w}, \sigma) = \mathcal{N}(t; f(x, \mathbf{w}), \sigma^2)$



The problem from a different view

- Aim: we want to find the **w** that maximizes p .

$p(t \mid x, \mathbf{w}, \sigma)$ is the likelihood of the measured data given a model.

Intuitively: find parameters **w** that maximize the probability of measuring the already measured data t.

"Maximum Likelihood Estimation"

σ is also part of the model and can be estimated.
For now, we assume σ is known.

Maximum Likelihood Estimation

- Given data points: $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$

  Assumption: points are drawn independently from *p*:

$$p(t|x, w, \sigma) = \prod_{i=1}^{N} p(t_i|x_i, w, \sigma^2)$$

$$= \prod_{i=1}^{N} N(t_i; w^T x_i, \sigma^2)$$

- Maximize its logarithm:

$$lnp(t|x, w, \sigma) = \sum_{i=1}^{N} lnp(t_i|x_i, w, \sigma)$$

$$\boxed{\mathcal{N} \to \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}}$$

$$= \frac{1}{2}\sum_{i=1}^{N}(-ln(\sigma^2) - ln(2\pi)) - \sum_{i=1}^{N} \frac{1}{2\sigma^2}(w^T x_i - t_i)^2$$

Constant

Is equal to *E(w)*

The parameters that maximize the likelihood are equal to the minimum of the sum of squared errors

Questions