2024 Spring Pattern Recognition Homework 1 Announcement

Presenter: TA Wei-Hsiang (Sean)
Release Date: 2024/03/20 12:00

Homework 1

- Deadline: 23:59, Apr. 3rd (Wed), 2024
- **Coding** (60%): Implement linear regression by **only** using ***numpy*.**
  - Submit your code in executable python files (.py).
  - Report the outcome and parameters by screenshots to the questions.
- **Handwritten Questions** (40%): Answer questions about linear regression.
  - Answer the questions in the report.
  - You **must use the template** and in **digital-typed** (**no handwritten scan**)
  - In English

Links

- Questions and Report template
- Sample code / Dataset

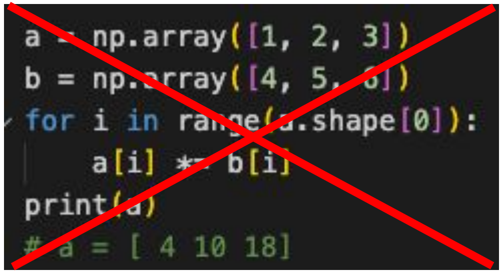
Coding Environment

- Recommnedation: Python 3.9 or higher
- Tips
  - We recommend you to use **virtual environments** when implementing your homework assignments.
  - Here are some popular virtual environment management tools
    - Poetry
    - Conda
    - Virtualenv

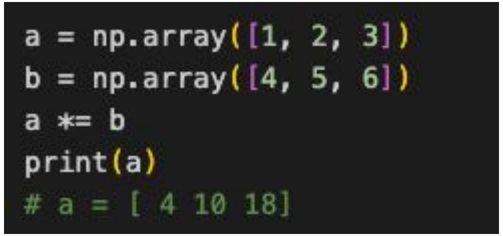
Numpy

- High efficient vector and matrix operations
- Numpy Tutorial: Link

element-wise multiply

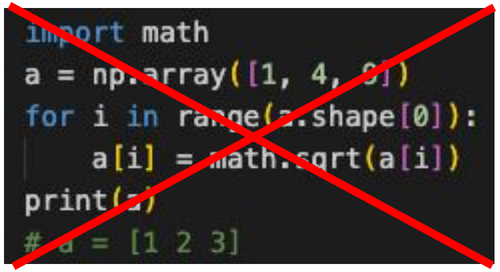


```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
for i in range(a.shape[0]):
    a[i] *= b[i]
print(a)
# a = [ 4 10 18]
```

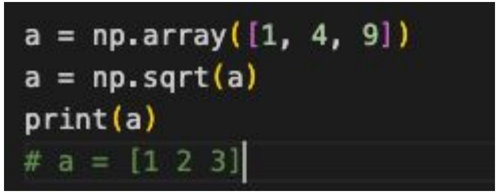


```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
a *= b
print(a)
# a = [ 4 10 18]
```

squre root



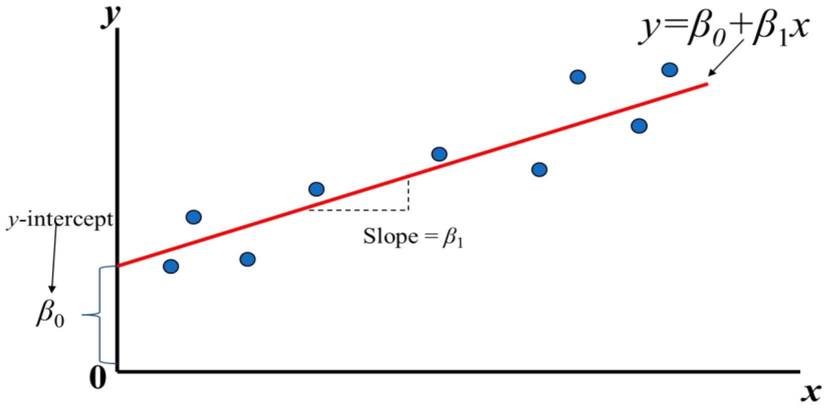
```
import math
a = np.array([1, 4, 9])
for i in range(a.shape[0]):
    a[i] = math.sqrt(a[i])
print(a)
# a = [1 2 3]
```

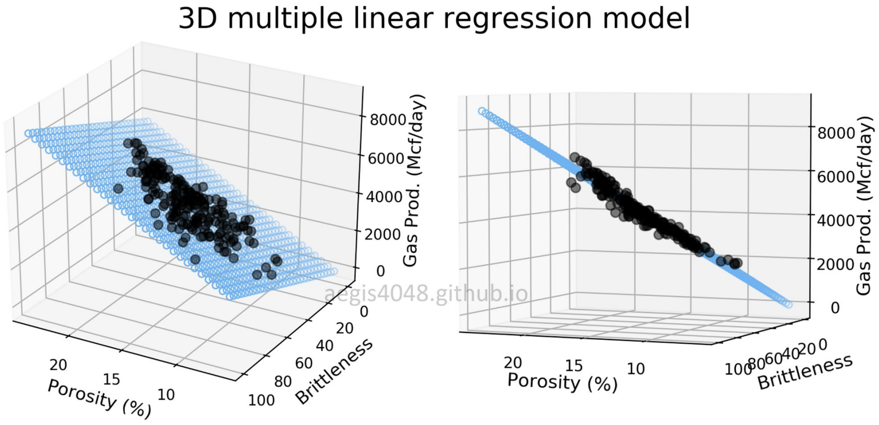


```
a = np.array([1, 4, 9])
a = np.sqrt(a)
print(a)
# a = [1 2 3]
```

Linear Regression

- Find the the slope (weights) and the intercept of given data





https://aegis4048.github.io/mutiple_linear_regression_and_visualization_in_python

How to find β0 and β1?

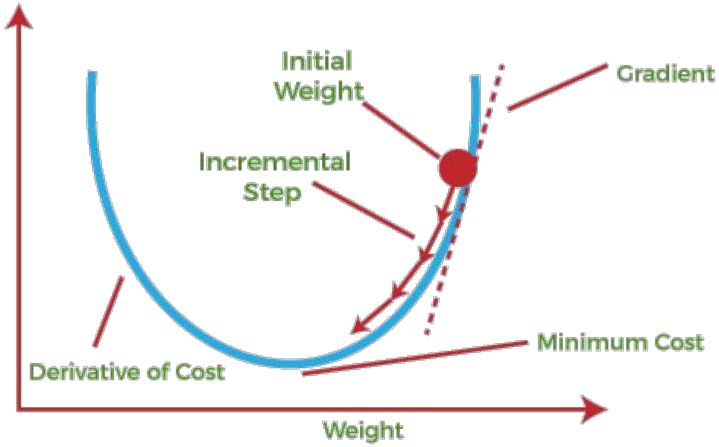
- Implement the closed-form solution (*Question 1-1*)

$$\hat{\beta} = (X^T.X)^{-1}X^T.Y$$

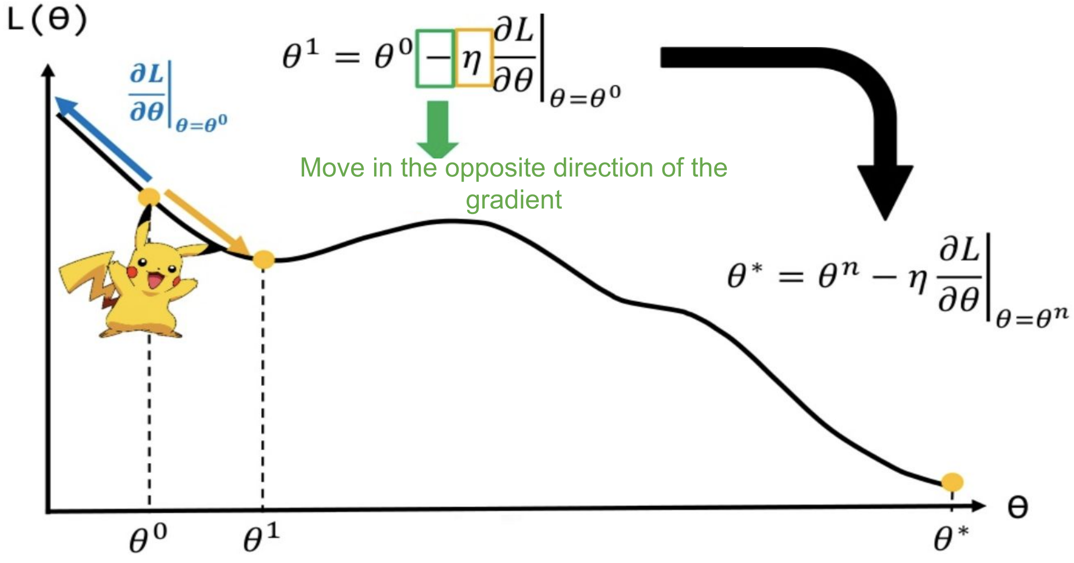
- How about a large dataset?
  - high dimensional data
  - huge amount of data

How to find β0 and β1?

- Gradient Descent (*Question1-2 ~ Question1-6*)



Learning rate: How far for one step



$$\theta^1 = \theta^0 \boxed{-} \boxed{\eta} \frac{\partial L}{\partial \theta}\Big|_{\theta=\theta^0}$$

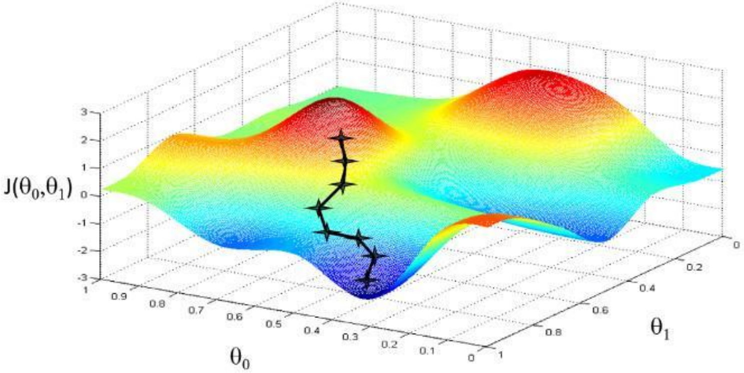
Move in the opposite direction of the gradient

$$\theta^* = \theta^n - \eta \frac{\partial L}{\partial \theta}\Big|_{\theta=\theta^n}$$

To better understand GD, see https://www.youtube.com/watch?v=yKKNr-QKz2Q

Gradient Descent

- x-axis and y-axis: the value of weights
- z-axis: the value of loss of the corresponding weights
- Goal: Find the weights that minimize the value of loss



https://www.linkedin.com/pulse/gradient-descent-visualization-python-ibrahim-sobh/

Dataset and Environment

- Student Performance Dataset
- Features
    - Hour Studied
    - Previous Score
    - Sleep Hours
    - Sample Question Papers Practiced
- Target
    - Performance Index (higher means better performance)

- Required packages: \`numpy\`, \`pandas\`, \`matplotlib\`, \`loguru\`, \`flake8\`, \`pytest\`

Linear Regression – Closed-form Solution

- Requirements
  - Implement Linear Regression by **closed-form** solution.
- Grading Criteria
  - (10%) Show the weights and intercepts of your linear model.
- Tips
  - There is only one answer.
  - You can check your answer by yourself using third-party libraries (such as scikit-learn).

Linear Regression – Gradient Descent

- Requirements
  - Update your weights and intercept by using **gradient descent**
    - you can implement mini-batch gradient descent or stochastic gradient descent if you want.
  - Use MSE (Mean Square Error) as your loss function.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

  - Tune the **learning rate** and **epoch** hyper-parameters (and **batch size** if you implement mini-batch gradient descent) to make your testing MSE loss as close as the closed-form solution.
  - Implement the L1 regularization into the regressor.

Linear Regression – Gradient Descent

- Grading Criteria
  - (10%) Show the weights and intercepts of your linear model.
  - (10%) Plot the learning curve. (x-axis=epoch, y-axis=training loss)
  - (20%) Show your error rate between your closed-form solution and the gradient descent solution.
    - error rate: (gradient_descent_loss - closed_form_loss) / closed_form_loss * 100
  - (Bonus 5%, cap: 60%) Implement L1 regularization, snapshot the weights and code implementation differences.

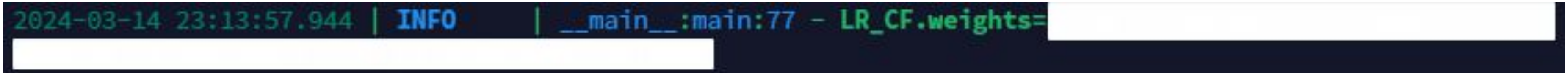
Points	**error rate**
20	**< 0.5%**
15	**< 1%**
10	**< 3%**
5	**< 5%**
0	**>= 5%**

Linear Regression – Gradient Descent

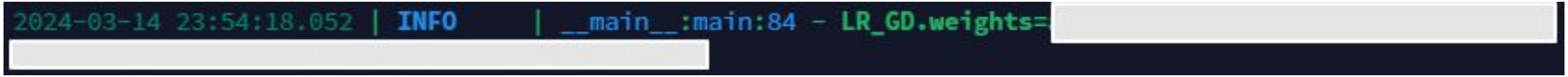
- Tips
  - Finding suitable hyper-parameters may cost you some time. Be patient!

Code Output

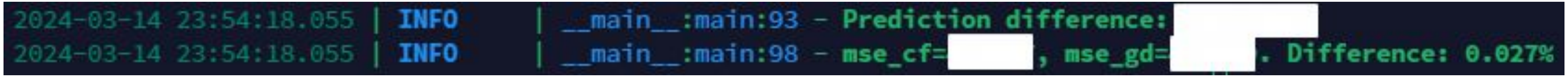
- Do not modify the main function architecture heavily.
- Your code output will look like this



```
2024-03-14 23:13:57.944 | INFO     | __main__:main:77 - LR_CF.weights=[redacted]
```



```
2024-03-14 23:54:18.052 | INFO     | __main__:main:84 - LR_GD.weights=[redacted]
```



```
2024-03-14 23:54:18.055 | INFO     | __main__:main:93 - Prediction difference: [redacted]
2024-03-14 23:54:18.055 | INFO     | __main__:main:98 - mse_cf=[redacted], mse_gd=[redacted]. Difference: 0.027%
```

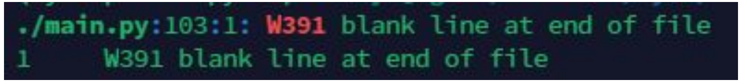
Python Coding Style Guide Reference

1. PEP8
2. Google Python Style

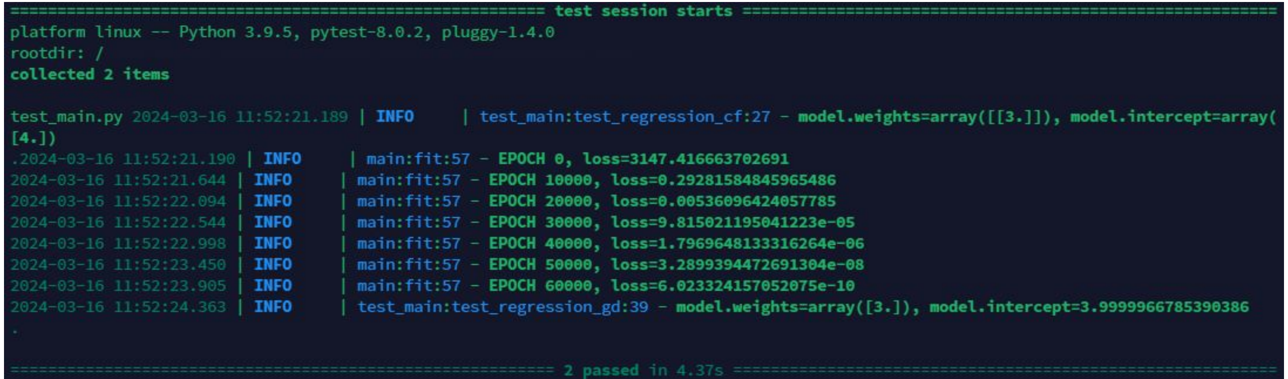
Additional Requirements

Code Check and Verification: **Lint** the code and show the **PyTest** results (10%)

- Code linting: `$ flake8 main.py`
    - -2pt per warning / error
- Run PyTest: `$ pytest ./test_main.py -s`
    - -5pt per failed case



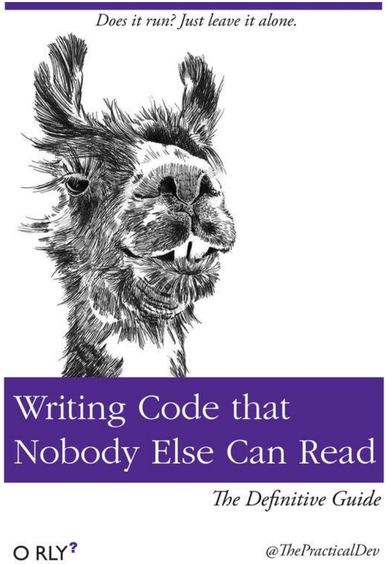
```
./main.py:103:1: W391 blank line at end of file
1       W391 blank line at end of file
```



```
=========================================================== test session starts ============================================================
platform linux -- Python 3.9.5, pytest-8.0.2, pluggy-1.4.0
rootdir: /
collected 2 items

test_main.py 2024-03-16 11:52:21.189 | INFO     | test_main:test_regression_cf:27 - model.weights=array([[3.]]), model.intercept=array(
[4.])
.2024-03-16 11:52:21.190 | INFO     | main:fit:57 - EPOCH 0, loss=3147.416663702691
2024-03-16 11:52:21.644 | INFO     | main:fit:57 - EPOCH 10000, loss=0.29281584845965486
2024-03-16 11:52:22.094 | INFO     | main:fit:57 - EPOCH 20000, loss=0.005360964240577 85
2024-03-16 11:52:22.544 | INFO     | main:fit:57 - EPOCH 30000, loss=9.815021195041223e-05
2024-03-16 11:52:22.998 | INFO     | main:fit:57 - EPOCH 40000, loss=1.7969648133316264e-06
2024-03-16 11:52:23.450 | INFO     | main:fit:57 - EPOCH 50000, loss=3.2899394472691304e-08
2024-03-16 11:52:23.905 | INFO     | main:fit:57 - EPOCH 60000, loss=6.023324157052075e-10
2024-03-16 11:52:24.363 | INFO     | test_main:test_regression_gd:39 - model.weights=array([3.]), model.intercept=3.9999966785390386
.

============================================================ 2 passed in 4.37s =============================================================
```


Does it run? Just leave it alone.
Writing Code that Nobody Else Can Read
The Definitive Guide
O RLY?
@ThePracticalDev

Handwritten Questions (40%)

2-1 (10%) Please describe the Vanishing Gradient Problem in detail, and provide at least two solutions to overcome this problem.

2-2 (15%) Gradient descent often suffers from the issue of getting stuck at local minima. Please provide at least two methods to overcome this problem and discuss how these methods work.

2-3 (15%) What are the basic assumptions of Linear regression between the features and the target? How can techniques help Linear Regression extend beyond these assumptions? Please at least answer one technique.

Report

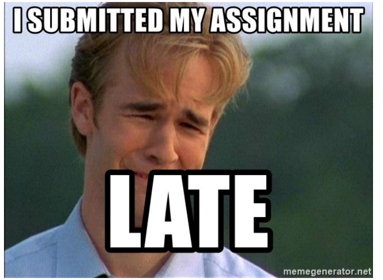
- Please follow the report template format. (-5pts if not use the template)
- Link

Submission

- Compress your **code** and **report** into a **.zip file** and submit it to E3.
- Report should be written in English. (-5 pts if not English)
- <STUDENT ID>_HW1.zip
  - main.py
  - setup.cfg
  - test_main.py
  - <STUDENT ID>_HW1.**pdf** (NO .doc, .docx or others format)
- Don't put the data (e.g. train.csv / test.csv) into submission file

Other rules

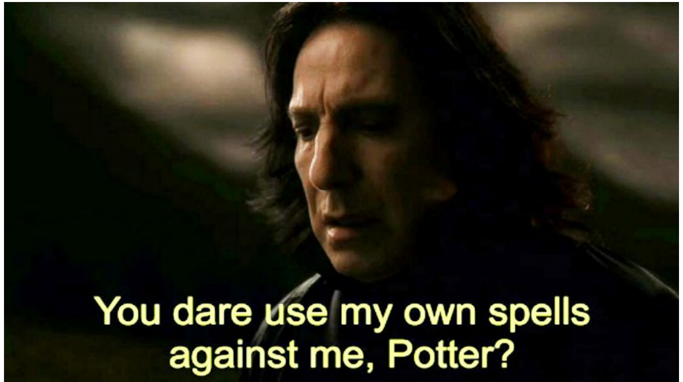
- **Late Policy**: A penalty of **20 points** per additional late day. (-20pt / delayed.day)
    - For example, If you get 90 points but delay for two days, your will get only 50 points!



- **No Plagiarism**: You should complete the assignment by yourself. Students engaged in plagiarism will be penalized heavily. Super serious penalty.
    - e.g. -100pt for the assignment or failed this course, etc
    - Report to academic integrity office

AI-Assistant

- Not recommended but no forbidden
- Copy-and-Paste answers from the AI-Assiant will be seen as Plagiarism
  - However, you can have your own answer first then rephrase it by AI-Assiant.
- Some questions might be parts of final exam, make sure you understand the concept

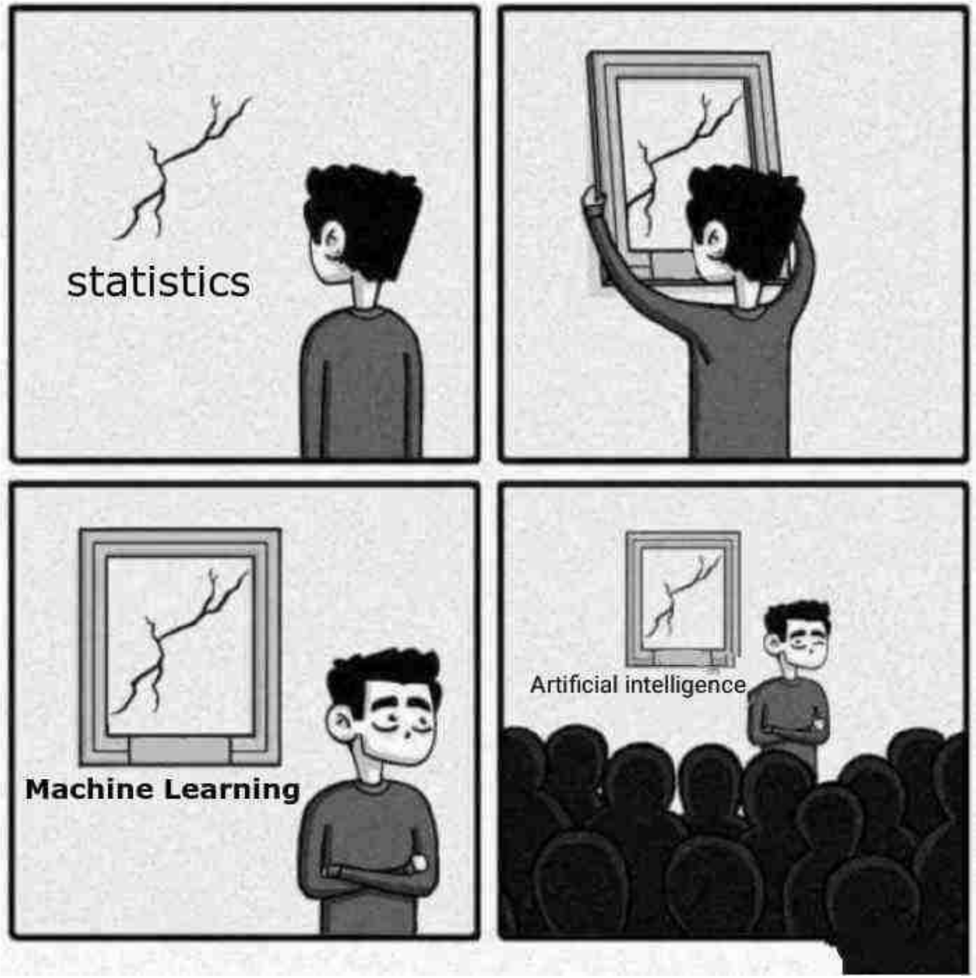


FAQs

- Why can’t my gradient descent model converge?
  - Make sure you calculate the gradients correctly.
  - Use smaller learning rate.
- Can I use deep learning frameworks such as TensorFlow, PyTorch or other library such as math?
  - **No!** In HW1, you are request using only Numpy to implement linear regression and gradien descent. You can use matplotlib to plot the results.

- If you have other questions, ask on **E3 forum** first! We will reply as soon as possible.

Have Fun!


statistics
Machine Learning
Artificial intelligence