Lab 1 – The Basics of Python and Pytorch

Xingyu, Yunuo, Yachen, Yanwei

March 6, 2025

This lab aims to help the students refresh the basics of python, particularly, NumPy. You may use python functions to answer these questions, e.g., np.sum, torch.sigmoid, etc. But all the algorithm should be implemented by yourself unless otherwise specified, e.g., you should write a dataset class instead of using pytorch Dataset class in Problem 17.

Please write the report, including the codes, and screen-shot the results, and upload to **eLearning**, **by 17:00 April 3th, 2025.**

Late submissions may affect the final score, so please plan accordingly.

Submit **a Jupyter Notebook** (.ipynb) file with executable code to ensure full reproducibility.

Submit **a PDF report** that includes the code, screenshots of results, and necessary analysis and conclusions.

1. Write a Python function to sum all the numbers in a list.

2. Write a Python function that takes a list and returns a new list with unique elements of the first list.

   e.g.,

   Input:[1, 3, 3, 2, 3, 4, 3, 4, 5].

   Output: [1, 2, 3, 4, 5].

3. Write a Python function that checks whether a passed string is palindrome or not. A palindrome is a word, phrase, or sequence that reads the same backward as forward. For example, both “madam” and “nurses run” are palindromes.

4. Write a NumPy program to find the real and imaginary parts of an array of complex numbers.

   e.g.,

   Input: array [1.00000000+0.j 0.70710678+0.70710678j]

   Output: array [[1, 0], [0.70710678, 0.70710678]]

5. Write a Python program to add two ternary(base-3) numbers.

   e.g.,

   Input : (’12’, ’21’)

   Output : ’110’

6. You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list. You may assume the two numbers do not contain any leading zero, except the number 0 itself.

e.g.,

Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)

Output: 7 -> 0 -> 8

Explanation: 342 + 465 = 807.

Linked list is defined as follow

```
# Definition for singly-linked list.
# class ListNode:
    # def __init__(self, x):
        # self.val = x
        # self.next = None
```

7. Implement bubble sort
8. Implement merge sort
9. Implement quick sort
10. Implement heap sort
11. Implement linear regression model and use autograd to optimize it by Pytorch.
12. Implement logistic regression model and use autograd to optimize it by Pytorch.
13. Implement linear SVM model for binary classification task and use autograd to optimize it by Pytorch.

    Hint: you may use the loss of $\sum \max\left[0, 1 - y\left(wx + b\right)\right]$
14. Add a Frobenius norm penalty for the weight w in your SVM model by two different ways: (1) use a pytorch function to calculate the norm; (2) implement the code by yourself.

    Hint: Frobenius norm of a matrix A is $\|A\|_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2\right)^{\frac{1}{2}}$.
15. Learn how to use linear regression[1], logistic regression[2], and SVM[3] by scikit-learn.
16. Download CIFAR-10 dataset[4] and visualize some of its images.
17. Write a dataset class for loading CIFAR-10. Make sure it could be transferred to Pytorch *Dataloader*. The class should meet the following requirements: (1) Inherit pytorch's *DataSet* class; (2) Load the image file and save in proper way; (3) Override *__getitem__* and *__len__* methods.

    Hint: If you find this part a little hard, check the official code[5] and make sure you understand each part.
18. Read[6] and learn how to use *torchvision.transforms* to transform images.

[1]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression

[2]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

[3]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[4]https://www.cs.toronto.edu/~kriz/cifar.html

[5]https://pytorch.org/docs/stable/_modules/torchvision/datasets/cifar.html#CIFAR10

[6]https://pytorch.org/docs/stable/torchvision/transforms.html

19. Run one epoch for loading CIFAR-10 with Pytorch *Dataloader* and test the loading time of different *batch_ size* (1, 4, 64, 1024), different *num_ workers* (0,1,4,16), and whether use *pin_ memory* or not.

20. Calculate the mean and std of CIFAR-10' training set within each RGB channel.

21. Image to character painting

    (a) Target

    Converting the RGB color image to character painting with Python code

    - Character painting is a combination of a series of characters. We can think of characters as relatively large pixels. A character can represent a color. The more types of characters, the more colors can be represented, and the picture will be more hierarchical sense

    (b) Requirements

    - Python 3.5
    - pillow 5.1.0

    (c) Method

    i. Use PIL (pillow) to get the input picture

    ii. Use the following formula to map RGB values to gray values (note that this formula is not a real algorithm, but a simplified sRGB IEC61966-2.1 formula)

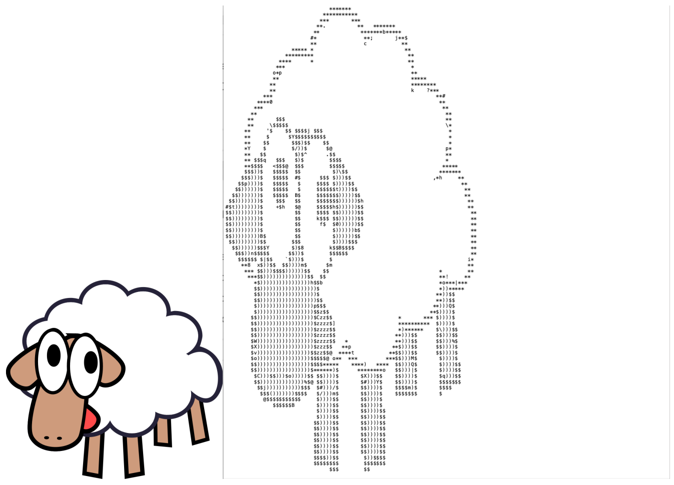
    - $gray = 0.2126 * r + 0.7152 * g + 0.0722 * b$

    iii. Create a character list (length and content are customized)

    iv. Map the gray value to characters and save the result with a string (note the corresponding picture size, add line breaks)

    v. Export character painting to a *.txt* file

    (d) Result



22. Numpy exercises

    - Consider a random 10x2 matrix representing cartesian coordinates, convert them to polar coordinates.
    - Create a 2D array subclass such that Z[i, j] == Z[j, i].

- Consider 2 sets of points P0, P1 describing lines (2d) and a set of points P, how to compute distance from each point j (P[j]) to each line i (P0[i],P1[i])?

23. Bilinear Interpolation

    Please implement the bilinear interpolation algorithm using python. Check this for an introduction to bilinear interpolation.

    Test samples:

    A =

    ((110, 120, 130),

    (210, 220, 230),

    (310, 320, 330))

    BilinearInterpolation(A, (1, 1)) == 110

    BilinearInterpolation(A, (2.5, 2.5)) == 275

24. Cartesian product

    Given an arbitrary number of vectors, build the cartesian product (every combinations of every item).

    e.g. [1, 2, 3], [4, 5], [6, 7] ==> [[1 4 6] [1 4 7] [1 5 6] [1 5 7] [2 4 6] [2 4 7] [2 5 6] [2 5 7] [3 4 6] [3 4 7] [3 5 6] [3 5 7]]

25. Extracting a subpart of an array

    Consider an arbitrary array, write a function that extract a subpart with a fixed shape and centered on a given element (pad with a *fill* value when necessary)

    e.g.

    In:

    >> Z = np.random.randint(0, 10, (5, 5))

    >> shape = (4, 4)

    >> fill = 0

    >> position = (1,1)

    >> Z

    [[3 6 8 5 9]

    [4 9 0 0 9]

    [6 1 4 0 8]

    [9 1 2 0 9]

    [4 1 7 5 0]]

    Out:

    [[0 0 0 0]

    [0 3 6 8]

    [0 4 9 0]

    [0 6 1 4]]

26. Matrix operations

    Please implement following matrix (just 2D) operations without numpy:

    - add
    - subtract
    - scalar multiply
    - multiply
    - identity
    - transpose
    - inverse

    Test samples:

    In:

    ```
    >> matrix_a = [[12, 10], [3, 9]]
    >> matrix_b = [[3, 4], [7, 4]]
    >> matrix_c = [[11, 12, 13, 14], [21, 22, 23, 24], [31, 32, 33, 34], [41, 42, 43, 44]]
    >> matrix_d = [[3, 0, 2], [2, 0, -2], [0, 1, 1]]
    ```

    Out:

    ```
    add(matrix_a, matrix_b) == [[15, 14], [10, 13]]
    subtract(matrix_a, matrix_b) == [[9, 6], [-4, 5]]
    scalar_multiply(matrix_b, 3) == [[9, 12], [21, 12]]
    multiply(matrix_a, matrix_b) == [[106, 88], [72, 48]]
    identity(3) == [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
    transpose(matrix_c) == [[11, 21, 31, 41], [12, 22, 32, 42], [13, 23, 33, 43], [14, 24, 34, 44]]
    inverse(matrix_d) == [[0.2, 0.2, 0.0], [-0.2, 0.30000000000000004, 1.0], [0.2, -0.30000000000000004, 0.0]]
    ```

27. Greatest common divisor

    Find the greatest common divisor(gcd) of two integers.

    Test samples:

    - GCD(3, 5) = 1
    - GCD(6, 3) = 3
    - GCD(-2, 6) = 2
    - GCD(0, 3) = 3

28. Find all consecutive positive number sequences whose sum is N

    e.g. 18+19...+22 = 9+10+...+16 = 100

    Find all consecutive positive number sequences whose sum is 1000, and report your results.