Programming Assignment #1

Lecturer: Prof. Jaesik Park
Teaching Assistants: Cheolhong Min, Milo Deroussi, Youngjoong Kim

***** PLEASE READ THIS GRAY BOX CAREFULLY BEFORE STARTING THE ASSIGNMENT *****

Due date: 11:59 PM April 04, 2025

Evaluation policy:

- Late submission penalty
  - Late submission penalty of 30% per day will be applied to the total score.
    - 30% penalty until 11:59 PM April 05, 60% on April 06, 90% on April 07
  - After 11:59 PM April 08
    - 100% penalty is applied for the submission.
- Your code will be automatically tested using an evaluation program.
  - Each problem has the maximum score.
  - A score will be assigned based on the behavior of the program.
- We won't accept any submission via email - it will be ignored.
- Do not modify auxiliary files.
  - Such as: "test.py", and so on.
- Run your Python script and check your program before the submission.
- Please do not use any libraries other than primitive libraries and NumPy
  - Such as: "Scipy", "Scikit-learn"
  - Any submission using other libraries will be disregarded.
- Never try to copy your classmates' source code. Both of you and your classmates will get severe penalties, and the case will be reported to the University's office.

File(s) you need to submit:

- pa1.py (Do not change the filename!)
- Your Environment details (OS, Python version and Numpy version)

Any questions? Please use the **Classum** community (using a tag, "Assignment #1").

0. Prerequisite

Environment

- OS: Windows, Mac OS, or Linux
- Language: Python (> python3.8 is recommended)

Install Python and PIP. If you already have them, you can skip this step.

- For Windows and Mac Users
  : Download and install Python from https://www.python.org/downloads/.

  If installing on Windows, make sure to check **'Add python.exe to PATH'**.

```
$ python -c “import pip; print(‘installed’)”
installed
```

- For Linux Users (e.g., Ubuntu 22.04)

```
$ apt-get update
$ apt-get install -y python3 python3-pip
$ ln -s python3 /usr/bin/python
$ python -c “import pip; print(‘installed’)”
installed
```

Basic instruction

- Install the dependencies and run the script to test its functionality

```
$ pip install numpy
$ python test.py
[*] Nothing returned. Please check the return
value of `prepare_dataset`.
[*] Response;
TestResult(dataset=None, models=None, metric=None,
state=FAILED_STEP1, _traceback=)
```

Step-by-step guidelines

Our goal in this programming assignment is to implement a handwritten digit image classifier using a Gaussian kernel model. We will provide step-by-step guidelines. Follow them carefully and submit your assignment.

The assignment will be evaluated automatically using different datasets than the one provided. Do not hardcode any values, and make sure to correctly fill in the required sections of the skeleton code, `pa1.py`.

1. Prepare the digit dataset

a. Implement a function to load a CSV-formatted handwritten digit image dataset. The function should take the file path as the path argument and return a tuple containing the loaded images and their corresponding labels.

b. Input & Output
Input: a path to a CSV-formatted handwritten digit image dataset.

Output: The function should return a tuple of two arrays
- **Images**: A NumPy array of shape [#images, 16, 16] with np.float32 data type, where the values are scaled to the range in [0.0, 1.0].
- **Labels**: A NumPy array of shape [#images] with np.long data type, where values range from 0 to 9, indicating the digit represented by each image.

c. Example input & output

Input	Output
`Path("trainset.csv")`	`(array([[[0., 0., …, 0.], …], …], shape=(60, 16, 16), dtype=float32), array([0, 0, …, 9, 9]))`
`Path("evalset.csv")`	`(array([[[0., 0., …], …], …], shape=(60, 16, 16), dtype=float32), array([0, 0, …, 9, 9]))`

d. Example execution

```
>> from pathlib import Path
>> from pa1 import prepare_dataset
>> prepare_dataset(Path("trainset.csv"))
(array([[[0., 0., …], …], …], shape=(60, 16, 16),
dtype=float32), array([0, 0, …]))
```

After you correctly implementing the `prepare_dataset`;

```
$ python test.py
[*] Successfully load the dataset.
[*] Invalid value returned. Please check the return value of
`kernel_function`
[*] Response;
TestResult(dataset=..., models=None, metric=None,
state=FAILED_STEP2, _traceback=)
```

2. Write the kernel function

a. Implement a function to compute a Gaussian kernel. The function should take two image arrays as input and return the computed kernel value.

$$K(x, y) = \exp\left(-\frac{||x - y||_2^2}{2\sigma^2}\right)$$

b. Input & Output
Input: two image arrays and a bandwidth parameter, sigma.

- **a, b**: A NumPy array of shape [16, 16] with np.float32 data type, where the values are scaled to the range in [0.0, 1.0].
- **sigma**: a bandwidth parameter, denoted as sigma in the formula above

Output: the computed kernel value.

c. Example input & output

Input	Output
`kernel_function(np.ones((16, 16)), np.ones((16, 16)), 1.0)`	`1.0`
`kernel_function(np.ones((16, 16)), np.zeros((16, 16)), 10.0)`	`0.2780373`

For various reasons, the output value may differ from the example.
Do not worry about minor differences.

d. Example execution

```
>> from numpy as np
>> from pa1 import kernel_function
>> kernel_function(np.ones((16, 16)), np.zeros((16, 16)), 10.0)
np.float64(0.27803730045319414)
```

After you correctly implementing the `kernel_function`;

```
$ python test.py
[*] Successfully load the dataset.
[*] Kernel function passed the unit test.
[*] Invalid value returned. Please check the return value of
`train`
[*] Response;
TestResult(dataset=..., models=None, metric=None,
state=FAILED_STEP3, _traceback=)
```

TIP: Flatten an image array into a 1D array first.

3. Train the parameters

a. Implement a function to update the β parameters. The function should take the loaded dataset and other training hyperparameters as input and return the optimized β.

$$\hat{\beta}_i \rightarrow \beta_i + \alpha \left(y - \sum_j \beta_j K(X_i, X_j) \right)$$

Assume the initial value of β is an all-zero vector.

b. Input & Output

Input: an initial values of β and a dataset with training hyperparameters.

- **dataset**: a loaded dataset, the return values of `prepare_dataset`.
- **num_training_steps**: the number of updates.
- **learning_rate**: the step size, α.
- **sigma**: a bandwidth parameter of the Gaussian kernel.

Output: the optimized parameters, β

c. Example execution

```
>> from pathlib import Path
>> import numpy as np
>> from pa1 import prepare_dataset, train
>> images, labels = prepare_dataset(Path(“./trainset.csv”))
>> train((images, labels), 1000, 0.001, 1.0)
array([-0.33890304, -0.56380779, …, 0.6312599])
```

For various reasons, the output value may differ from the example.
Do not worry about minor differences.

After you correctly implementing the `train`;

```
$ python test.py
[*] Successfully load the dataset.
[*] Kernel function passed the unit test.
[*] Training done.
[*] Invalid return value. Please check the return value of
`classify`
[*] Response;
TestResult(dataset=..., models=None, metric=None,
state=FAILED_STEP4, _traceback=)
```

4. Classify the images

a. Implement a function to classify a given image using the trained binary classifier. The function should take trained parameters, their corresponding training images, and a single image array as input, then return the predicted label.

   TIP: Mapping the labels from 0 and 1 to -1 and 1 before training can improve performance. (`classify` function should be updated accordingly.)

b. Input & Output

   Input: a trained binary classifier and a target image array to classify.

   - **model**: a pair of trained parameters, β, and the corresponding training images.
   - **given**: A target image array to classify, with a size of [16, 16] and float-type data in the range [0, 1].

   Output: the estimated label, either 0 or 1.

c. Example execution

```
>> from pathlib import Path
>> import numpy as np
>> from pa1 import prepare_dataset, train, classify
>> images, labels = prepare_dataset(Path("./trainset.csv"))
>> params = train((images, labels), 1000, 0.001, 1.0)
>> model = (params, images)
>> classify(model, images[0])
0
>> classify(model, images[-1])
1
```

   For various reasons, the output value may differ from the example.
   Do not worry about minor differences.

   After you correctly implementing the `classify`;

```
$ python test.py
[*] Successfully load the dataset.
[*] Kernel function passed the unit test.
[*] Training done.
[*] Training accuracy: 1.00
[*] Evaluation accuracy: 0.88
[*] Response;
TestResult(dataset=..., models=None,
metric={'training-accuracy': 1.0, 'evaluation-accuracy':
0.8833333333333333}, state=SUCCESS, _traceback=)
```

5. Submit the assignment

After verifying that `state=SUCCESS`, submit your `pa1.py` to ETL.
We expect you to achieve more than 80% evaluation accuracy, which will allow you to obtain a high score on the test set without much difficulty.

When submitting, include a comment specifying your testing environment:

- OS
- Python version
- NumPy version

This information will be used for debugging if your submission does not work properly.