AI VIET NAM – COURSE 2024

Python OOP – Exercise

Ngày 15 tháng 6 năm 2024

I. Câu hỏi tự luận

1. **Viết class và cài phương thức softmax.**

   Trong pytorch, torch.nn.Module là lớp cơ bản để từ đó xây dựng lên các mô hình hoặc các phương thức kích hoạt (activation funtion) như sigmoid, softmax,... Trong phần này, chúng ta xây dựng class Softmax và softmax_stable nhận đầu vào là mảng 1 chiều (tensor 1 chiều) dựa vào kế thừa từ lớp Module và ghi đè vào phương thức forward() theo công thức sau đây:

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

$$\text{softmax\_stable}(x_i) = \frac{\exp(x_i - c)}{\sum_{j=1}^{n} \exp(x_j - c)}$$
$$c = \max(x)$$

Hình 1: Softmax

```
# Examples 1
data = torch.Tensor([1, 2, 3])
softmax = Softmax()
output = softmax(data)
output
>> tensor([0.0900, 0.2447, 0.6652])

data = torch.Tensor([1, 2, 3])
softmax_stable = softmax_stable()
output = softmax_stable(data)
output
>> tensor([0.0900, 0.2447, 0.6652])
```

2. **Một Ward (phường) gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher. Một student gồm có name, yob (int) (năm sinh), và grade (string). Một teacher gồm có name, yob, và subject (string). Một doctor gồm có name, yob, và specialist (string). Lưu ý cần sử dụng a list để chứa danh sách của mọi người trong Ward.**

   (a) Cài đặt các class Student, Doctor, và Teacher theo mô tả trên. Thực hiện phương thức describe() method để in ra tất cả thông tin của các object.

   (b) Viết add_person(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor. Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward.

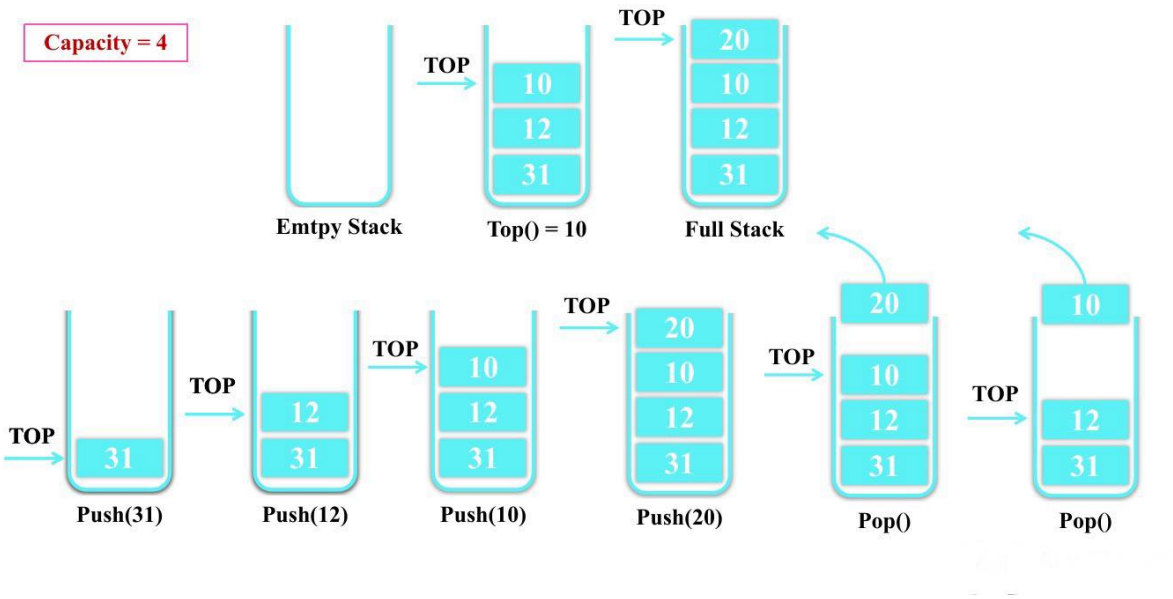
   (c) Viết count_doctor() method để đếm số lượng doctor trong ward.

   (d) Viết sort_age() method để sort mọi người trong ward theo tuổi của họ với thứ tự tăng dần. (hint: Có thể sử dụng sort của list hoặc viết thêm function đều được)

   (e) Viết compute_average() method để tính trung bình năm sinh của các teachers trong ward.

```
# Examples
# 2(a)
student1 = Student(name="studentA", yob=2010, grade="7")
student1.describe()
#output
>> Student - Name: studentA - YoB: 2010 - Grade: 7

teacher1 = Teacher(name="teacherA", yob=1969, subject="Math")
teacher1.describe()
#output
>> Teacher - Name: teacherA - YoB: 1969 - Subject: Math

doctor1 = Doctor(name="doctorA", yob=1945, specialist="Endocrinologists")
doctor1.describe()
#output
>> Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists


# 2(b)
print()
teacher2 = Teacher(name="teacherB", yob=1995, subject="History")
doctor2 = Doctor(name="doctorB", yob=1975, specialist="Cardiologists")
ward1 = Ward(name="Ward1")
ward1.add_person(student1)
ward1.add_person(teacher1)
ward1.add_person(teacher2)
ward1.add_person(doctor1)
ward1.add_person(doctor2)
ward1.describe()

#output
>> Ward Name: Ward1
Student - Name: studentA - YoB: 2010 - Grade: 7
Teacher - Name: teacherA - YoB: 1969 - Subject: Math
Teacher - Name: teacherB - YoB: 1995 - Subject: History
Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists

# 2(c)
```

```
print(f"\nNumber of doctors: {ward1.count_doctor()}")

#output
>> Number of doctors: 2

# 2(d)
print("\nAfter sorting Age of Ward1 people")
ward1.sort_age()
ward1.describe()

#output
>> After sorting Age of Ward1 people
Ward Name: Ward1
Student - Name: studentA - YoB: 2010 - Grade: 7
Teacher - Name: teacherB - YoB: 1995 - Subject: History
Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists
Teacher - Name: teacherA - YoB: 1969 - Subject: Math
Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists

# 2(e)
print(f"\nAverage year of birth (teachers): {ward1.compute_average()}")

#output
>> Average year of birth (teachers): 1982.0
```

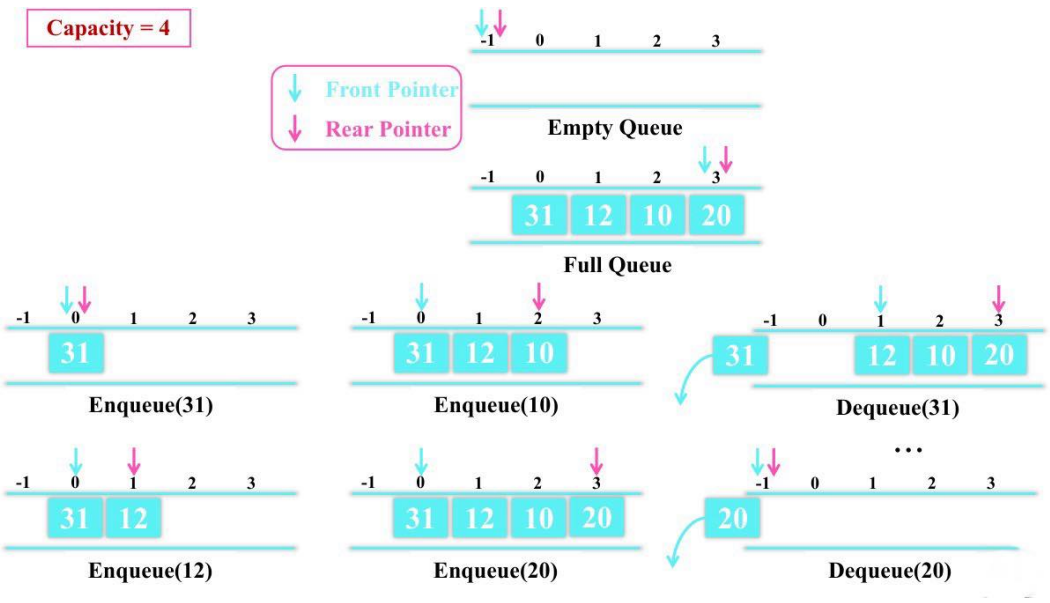


Hình 2: Stack

3. **Thực hiện xây dựng class Stack với các phương thức (method) sau đây**

   - **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa
   - **.is_empty()**: kiểm tra stack có đang rỗng
   - **.is_full()**: kiểm tra stack đã full chưa
   - **.pop()**: loại bỏ top element và trả về giá trị đó
   - **.push(value)** add thêm value vào trong stack
   - **.top()** lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó
   - **Không cần thiết phải thực hiện với pointer như trong hình minh họa**

```
stack1 = MyStack(capacity=5)

stack1.push(1)

stack1.push(2)

print(stack1.is_full())
>> False

print(stack1.top())
>>2

print(stack1.pop())
>> 2

print(stack1.top())
>> 1

print(stack1.pop())
>> 1

print(stack1.is_empty())
>> True
```



Hình 3: Queue

4. **Thực hiện xây dựng class Queue với các chức năng (method) sau đây**

   - **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa
   - **.is_empty()**: kiểm tra queue có đang rỗng
   - **.is_full()**: kiểm tra queue đã full chưa
   - **.dequeue()**: loại bỏ first element và trả về giá trị đó
   - **.enqueue(value)** add thêm value vào trong queue
   - **.front()** lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó
   - **Không cần thiết phải thực hiện với các pointers như trong hình minh họa**

```
queue1 = MyQueue(capacity=5)

queue1.enqueue(1)

queue1.enqueue(2)

print(queue1.is_full())
>> False

print(queue1.front())
>> 1

print(queue1.dequeue())
>> 1

print(queue1.front())
>> 2

print(queue1.dequeue())
>> 2

print(queue1.is_empty())
>> True
```

II. Câu hỏi trắc nghiệm

Câu hỏi 1: Kết quả của đoạn code dưới đây là bao nhiêu.

```
import torch
import torch.nn as nn

data = torch.Tensor([1, 2, 3])
softmax_function = nn.Softmax(dim=0)
output = softmax_function(data)
assert round(output[0].item(), 2) == 0.09
output
```

a) [0.0900, 0.2747, 0.6652]

b) [0.0700, 0.2447, 0.6652]

c) [0.0900, 0.2447, 0.6652]

d) [0.1900, 0.2447, 0.6692]

Câu hỏi 2: Hoàn thành đoạn code sau đây theo công thức tính softmax.

```
import torch
import torch.nn as nn

class MySoftmax(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        ### Your Code Here

        ### End Code Here

data = torch.Tensor([5, 2, 4])
my_softmax = MySoftmax()
output = my_softmax(data)
assert round(output[-1].item(), 2) == 0.26
output
```

a) [0.0900, 0.2747, 0.6652]

b) [0.7054, 0.0351, 0.2595]

c) [0.0900, 0.2447, 0.6652]

d) [0.7054, 0.0351, 0.009]

Câu hỏi 3: Hoàn thành đoạn code sau đây theo công thức tính softmax.

```
import torch
import torch.nn as nn

class MySoftmax(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        ### Your Code Here

```

```
        ### End Code Here

data = torch.Tensor([1, 2, 300000000])
my_softmax = MySoftmax()
output = my_softmax(data)
assert round(output[0].item(), 2) == 0.0
output
```

a) [0.0900, 0.2747, 0.6652]

b) [0.7054, 0.0351, 0.2595]

c) [0., 0., nan]

d) [0.7054, 0.0351, 0.009]

Câu hỏi 4: Hoàn thành đoạn code sau đây theo công thức tính stable softmax.

```
import torch
import torch.nn as nn

class SoftmaxStable(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        x_max = torch.max(x, dim=0, keepdims=True)
        x_exp = torch.exp(x - x_max.values)
        partition = x_exp.sum(0, keepdims=True)
        return x_exp / partition

data = torch.Tensor([1, 2, 3])
softmax_stable = SoftmaxStable()
output = softmax_stable(data)
assert round(output[-1].item(), 2) == 0.67
output
```

a) [0.0900, 0.2747, 0.6652]

b) [0.0900, 0.2447, 0.6652]

c) [0., 0., nan]

d) [0.7054, 0.0351, 0.009]

Câu hỏi 5: Một người (person) có thể là student, doctor, hoặc teacher. Một student gồm có name (string), yob (int) (năm sinh), và grade (string). Các bạn thực hiện viết class Student theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def get_yob(self):
        return self._yob

    @abstractmethod
```

```
    def describe(self):
        pass


class Student(Person):
    def __init__(self, name:str, yob:int, grade:str):
        ### Your Code Here

        ### End Code Here

    def describe(self):
        ### Your Code Here

        ### End Code Here

student1 = Student(name="studentZ2023", yob=2011, grade="6")
assert student1._yob == 2011
student1.describe()
```

a) Student - Name: studentZ2023 - YoB: 2011 - Grade: 6

b) Student - Name: studentZ2023 - YoB: 6 - Grade: 2011

c) Student - Name: 6 - YoB: studentZ2023 - Grade: 2011

d) Tất cả đều sai

Câu hỏi 6: Một người (person) có thể là student, doctor, hoặc teacher. Một teacher gồm có name (string), yob (int), và subject (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def get_yob(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Teacher(Person):
    def __init__(self, name:str, yob:int, subject:str):
        ### Your Code Here

        ### End Code Here

    def describe(self):
        ### Your Code Here

        ### End Code Here

teacher1 = Teacher(name="teacherZ2023", yob=1991, subject="History")
assert teacher1._yob == 1991
teacher1.describe()
```

a) Teacher - Name: 1991 - YoB: teacherZ2023 - Subject: History

b) Teacher - Name: teacherZ2023 - YoB: 1991 - Subject: History

c) Teacher - Name: History - YoB: teacherZ2023 - Subject: 1991

d) Tất cả đều sai

Câu hỏi 7: Một người (person) có thể là student, doctor, hoặc teacher. Một doctor gồm có name (string), yob (string), và specialist (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def get_yob(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Doctor(Person):
    def __init__(self, name:str, yob:int, specialist:str):
        ### Your Code Here

        ### End Code Here

    def describe(self):
        ### Your Code Here

        ### End Code Here

doctor1 = Doctor(name="doctorZ2023", yob=1981, specialist="Endocrinologists")
assert doctor1._yob == 1981
doctor1.describe()
```

a) Doctor - Name: doctorZ2023 - YoB: 1981 - Specialist: Endocrinologists

b) Doctor - Name: 1981 - YoB: doctorZ2023 - Specialist: Endocrinologists

c) Teacher - Name: History - YoB: teacherZ2023 - Subject: 1991

d) Tất cả đều sai

Câu hỏi 8 Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher và cần sử dụng một list để chứa danh sách của mọi người trong Ward. Viết add_person(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor. Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward. Chọn đáp án đúng nhất bên dưới cho phương thức đếm số lượng doctor.

```python
class Ward:
    def __init__(self, name:str):
        self.__name = name
        self.__listPeople = list()

    def add_person(self, person:Person):
        self.__listPeople.append(person)

    def describe(self):
        print(f"Ward Name: {self.__name}")
        for p in self.__listPeople:
            p.describe()

    def count_doctor(self):
        ### Your Code Here

        ### End Code Here

student1 = Student(name="studentA", yob=2010, grade="7")
teacher1 = Teacher(name="teacherA", yob=1969, subject="Math")
teacher2 = Teacher(name="teacherB", yob=1995, subject="History")
doctor1 = Doctor(name="doctorA", yob=1945, specialist="Endocrinologists")
assert ward1.count_doctor() == 1
doctor2 = Doctor(name="doctorB", yob=1975, specialist="Cardiologists")
ward1 = Ward(name="Ward1")
ward1.add_person(student1)
ward1.add_person(teacher1)
ward1.add_person(teacher2)
ward1.add_person(doctor1)
ward1.add_person(doctor2)
ward1.count_doctor()
```

a) 4

b) 3

c) 2

d) 1

Câu hỏi 9 Thực hiện xây dựng class Stack với các chức năng (method) sau đây: initial- ization method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. .is_full(): kiểm tra stack đã full chưa. .push(value) add thêm value vào trong stack. Kết quả đầu ra là gì?

```python
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def is_full(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        ### Your Code Here

        ### End Code Here

stack1 = MyStack(capacity=5)
stack1.push(1)
```

```
assert stack1.is_full() == False
stack1.push(2)
print(stack1.is_full())
```

a) True

b) False

c) None

d) Raise an error

Câu hỏi 10: Thực hiện xây dựng class Stack với các chức năng (method) sau đây: initialization method nhận một input "capacity", dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. Phương thức is_empty(): kiểm tra stack có đang rỗng. Phương thức is_full(): kiểm tra stack đã full chưa. Phương thức push(value) add thêm value vào trong stack. Phương thức top() lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def is_full(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        ### Your Code Here

        ### End Code Here

    def top(self):
        ### Your Code Here

        # End Code Here

stack1 = MyStack(capacity=5)
stack1.push(1)
assert stack1.is_full() == False
stack1.push(2)
print(stack1.top())
```

a) 1

b) 2

c) None

d) Raise an error

Câu hỏi 11: Thực hiện xây dựng class Queue với các chức năng (method) sau đây: initialization method nhận một input "capacity", dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. Phương thức is_full(): kiểm tra queue đã full chưa. Phương thức enqueue(value) add thêm value vào trong queue. Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []
```

```

    def is_full(self):
        return len(self.__queue) == self.__capacity

    def enqueue(self, value):
        ### Your Code Here

        ### End Code Here

queue1 = MyQueue(capacity=5)
queue1.enqueue(1)
assert queue1.is_full() == False
queue1.enqueue(2)
print(queue1.is_full())
```

a) False

b) True

c) None

d) Raise an error

Câu hỏi 12: Thực hiện xây dựng class Queue với các chức năng (method) sau đây: initialization method nhận một input "capacity" dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. Phương thức is_full(): kiểm tra queue đã full chưa. Phương thức enqueue(value) add thêm value vào trong queue. Phương thức front() lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó. Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isEmpty(self):
        return len(self.__queue) == 0

    def is_full(self):
        return len(self.__queue) == self.__capacity

    def dequeue(self):

    def enqueue(self, value):

    def front(self):
        ### Your Code Here

        ### End Code Here

queue1 = MyQueue(capacity=5)
queue1.enqueue(1)
assert queue1.is_full() == False
queue1.enqueue(2)
print(queue1.front())
```

a) 4

b) 3

c) 2

d) 1