mint 接口测试任务 37 测试文档

1 测试环境

mindspore: 2.4.0
python: 3.9.18
pytest: 7.3.0

2 测试的 mint 接口

（1）mindspore.mint.nn.functional.pad
（2）mindspore.mint.optim.AdamW
（3）mindspore.mint.linalg.inv
（4）mindspore.mint.special.erfc
（5）mindspore.mint.special.expm1

3 测试问题分析

3.1 mindspore.mint.nn.functional.pad 测试问题

问题描述：mindspore.mint.nn.functional.pad 接口在输入浮点数类型张量，且选择 circular 模式的情况下执行时会出现启动内核失败的问题

对应测试代码：

```python
@pytest.mark.parametrize('mode', ['constant', 'reflect', 'replicate', 'circular'])
def test_modes(mode):
    """
    (1c) 测试不同的模式
    """
    data_lst = [[1.1, 1.2, 1.3, 1.4], [1.5, 1.6, 1.7, 1.8]]
    input_pt = torch.tensor(data_lst, dtype=torch.float32)
    input_ms = ms.tensor(data_lst, dtype=ms.float32)

    if mode == 'constant':
        output_pt = torch.nn.functional.pad(input_pt, (2, 2), mode, 1.2)
        output_ms = ms.mint.nn.functional.pad(input_ms, [2, 2], mode, 1.2)
    else:
        output_pt = torch.nn.functional.pad(input_pt, (2, 2), mode)
        output_ms = ms.mint.nn.functional.pad(input_ms, [2, 2], mode)

    # 测试问题： 浮点类型的张量 执行circular模式下的这一函数会出现启动内核失败的问题
    assert np.allclose(output_pt.numpy(), output_ms.numpy(), rtol=1e-3)
```

报错关键信息：

```
FAILED test_pad.py::test_modes[circular] - RuntimeError: Launch kernel failed, name:Default/PadV3-op0
```

报错原因分析：

该报错产生的直接原因是底层计算内核启动失败，这可能是因为该函数接口的 circular 模式不稳定，可能对 float32 类型的张量存在支持缺陷。

3.2 mindspore.mint.optim.AdamW 测试问题

问题描述：mindspore.mint.optim.AdamW 优化器在 params 参数下输入列表、字典、元祖对象时报错有误

对应测试代码：

```
@pytest.mark.parametrize('mode', ['Tensor', 'None', 'List', 'Dict', 'Tuple', 'int_lr', 'minus_lr', 'minus_eps', 'betas', 'weight_decay', 'messy_input'])
def test_chaotic_input(mode):
    """
    (1d) 测试随机混乱输入，报错信息的准确性
    """
    # 测试输入参数为张量的情况
    if mode == 'Tensor':
        reported_flag = False
        input_ms = ms.tensor([1.1, 1.2, 1.3, 1.4], dtype=ms.float32)
        try:
            optimizer = ms.mint.optim.AdamW(params=input_ms)
        except Exception as e:
            reported_flag = True
            e = str(e)
            assert 'Tensor' in e
            print(e)
        if reported_flag == False:
            assert "No error message was reported when the input is tensor." == 1

    # 测试输入为空的情况
    if mode == 'None':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params=None)
        except Exception as e:
            reported_flag = True
            e = str(e)
            assert 'None' in e
            print(e)
        if reported_flag == False:
            assert "No error message was reported when the input is None." == 1

    # 测试输入为列表的情况
    if mode == 'List':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params=[1, 2, 3, 4, 5])
        except Exception as e:
            reported_flag = True
            e = str(e)
            assert 'List' in e
            print(e)
        if reported_flag == False:
            assert "No error message was reported when the input is List." == 1
    # 测试问题：报错存在异常：can only concatenate str (not "type") to str

    # 测试输入为字典的情况
    if mode == 'Dict':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params={"a":1, "b":2})
        except Exception as e:
            reported_flag = True
            e = str(e)
            assert 'Dict' in e
            print(e)
        if reported_flag == False:
            assert "No error message was reported when the input is Dict." == 1
    # 测试问题：报错存在异常：can only concatenate str (not "type") to str

    # 测试输入为元祖的情况
    if mode == 'Tuple':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params=(1, 2, 3))
        except Exception as e:
            reported_flag = True
            e = str(e)
            assert 'Tuple' in e
            print(e)
        if reported_flag == False:
            assert "No error message was reported when the input is Tuple." == 1
    # 测试问题：报错存在异常：can only concatenate str (not "type") to str

    # 测试学习率不为浮点数的情况
    ms_net = LeNet5_ms()
    if mode == 'int_lr':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params=ms_net.trainable_params(), lr=1)
        except Exception as e:
            print(e)
            reported_flag = True
            e = str(e)
            assert 'float' in e and 'int' in e
        if reported_flag == False:
            assert "No error message was reported when the type of learning rate is int." == 1
```

```
    # 测试学习率为负数的情况
    if mode == 'minus_lr':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params=ms_net.trainable_params(), lr=-0.5)
        except Exception as e:
            print(e)
            reported_flag = True
            e = str(e)
            assert 'learning rate' in e
        if reported_flag == False:
            assert "No error message was reported when learning rate is negative." == 1

    # 测试eps为负数的情况
    if mode == 'minus_eps':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params=ms_net.trainable_params(), eps=-1e-5)
        except Exception as e:
            print(e)
            reported_flag = True
            e = str(e)
            assert 'epsilon' in e
        if reported_flag == False:
            assert "No error message was reported when epsilon is negative." == 1

    # 测试betas范围不在[0,1)区间的情况
    if mode == 'betas':
        betas_lst = [(-0.5, 1.5), (1.0, 1.5), (0.5, -0.5), (0.5, 1.0)]
        for betas in betas_lst:
            reported_flag = False
            try:
                optimizer = ms.mint.optim.AdamW(params=ms_net.trainable_params(), betas=betas)
            except Exception as e:
                print(e)
                reported_flag = True
                e = str(e)
                assert 'beta' in e
            if reported_flag == False:
                assert "No error message was reported when beta parameter is invalid." == 1

    # 测试权重衰减参数为负数的情况
    if mode == 'weight_decay':
        reported_flag = False
        try:
            optimizer = ms.mint.optim.AdamW(params=ms_net.trainable_params(), weight_decay=-0.5)
        except Exception as e:
            reported_flag = True
            print(e)
            e = str(e)
            assert 'weight_decay' in e
        if reported_flag == False:
            assert "No error message was reported when weight_decay is negative." == 1

# 测试乱序输入
if mode == 'messy_input':
    reported_flag = False
    try:
        optimizer = ms.mint.optim.AdamW(params=(0.5, 0.5), betas=ms_net.trainable_params(), weight_decay=False, maximize=True)
    except Exception as e:
        reported_flag = True
        e = str(e)
        print(e)
        assert 'tuple' in e and 'list' in e
    if reported_flag == False:
        assert "No error message was reported when the input is messy" == 1
```

报错关键信息：

```
FAILED test_AdamW.py::test_chaotic_input[List] - assert 'List' in 'can only concatenate str (not "type") to str'
FAILED test_AdamW.py::test_chaotic_input[Dict] - assert 'Dict' in 'can only concatenate str (not "type") to str'
FAILED test_AdamW.py::test_chaotic_input[Tuple] - assert 'Tuple' in 'can only concatenate str (not "type") to str'
```

报错原因分析：

```
self = AdamW (), param_group = {'params': [1, 2, 3]}

    def _preprocess_param_group(self, param_group):
        """Preprocess param groups."""
        if not isinstance(param_group, dict):
            raise TypeError('Param group must be a dict.')

        params = param_group['params']
        if isinstance(params, Parameter):
            param_group['params'] = [params]
        elif isinstance(params, set):
            raise TypeError('Optimizer parameters need to be organized in ordered collections, but '
                            'the ordering of tensors in sets will change between runs. '
                            'Please use a list instead.')
        else:
            param_group['params'] = list(params)

        for param in param_group['params']:
            if not isinstance(param, Parameter):
>               raise TypeError("Optimizer can only optimize Parameters, but one of the params is " + type(param))
E               TypeError: can only concatenate str (not "type") to str

../../../anaconda3/envs/MindSpore/lib/python3.9/site-packages/mindspore/experimental/optim/optimizer.py:190: TypeError

During handling of the above exception, another exception occurred:
```

上图所展示的 optimizer.py 文件中有一行代码为：

```
raise TypeError("Optimizer can only optimize Parameters, but one of the params is " + type(param))
```

这一行代码存在问题，因为 string 类型无法与 Type 类型直接进行拼接

建议改为：

```
raise TypeError(f"Optimizer can only optimize Parameters, but one of the params is {type(param)}")
```

3.3 mindspore.mint.linalg.inv 测试问题

问题描述：mindspore.mint.linalg.inv 接口在输入张量为奇异矩阵时不报错，而它所对应的 pytorch 框架下的 torch.linalg.inv 接口输入奇异矩阵会引发报错。

```
def test_random_input():
    """
    (1d) 测试混乱输入
    """
    # 测试张量最后两个维度不相等的情况
    input_ms = ms.ops.randn((2, 5), dtype=ms.float32)
    reported_flag = False
    try:
        output_ms = ms.mint.linalg.inv(input_ms)
        print(output_ms)
    except Exception as e:
        reported_flag = True
        e = str(e)
        assert ("dimension" in e) and ("same" in e)

    if reported_flag == False:
        assert "No error message was reported when the tensor's last two dimensions are not same." == 1

    # 测试其他类型的输入
    input_ms = "Tensor"
    reported_flag = False
    try:
        output_ms = ms.mint.linalg.inv(input_ms)
        print(output_ms)
    except Exception as e:
        reported_flag = True
        e = str(e)
        assert "string" in e and "Tensor" in e

    if reported_flag == False:
        assert "No error message was reported when the input is string." == 1
```

```python
# 测试1维张量
reported_flag = False
input_ms = ms.ops.randn((100), dtype=ms.float32)
try:
    output_ms = ms.mint.linalg.inv(input_ms)
    print(output_ms)
except Exception as e:
    reported_flag = True
    e = str(e)
    assert "at least" in e and "2" in e

if reported_flag == False:
    assert "No error message was reported when the tensor's dimension is less then 2." == 1


# 测试6维以上的张量
reported_flag = False
input_ms = ms.ops.randn((2, 2, 2, 2, 2, 2, 2), dtype=ms.float32)
try:
    output_ms = ms.mint.linalg.inv(input_ms)
    print(output_ms)
except Exception as e:
    reported_flag = True
    e = str(e)
    assert "larger than" in e and "6" in e

if reported_flag == False:
    assert "No error message was reported when the tensor's dimension is larger than 6." == 1

# 测试输入为空的情况
reported_flag = False
try:
    output_ms = ms.mint.linalg.inv(None)
    print(output_ms)
except Exception as e:
    reported_flag = True
    e = str(e)
    assert "None" in e and "Tensor" in e

if reported_flag == False:
    assert "No error message was reported when the input is None" == 1

# 测试输入矩阵奇异矩阵
input_ms = ms.tensor([[1, 2, 3], [4, 5, 6], [7, 8, 9]], dtype=ms.float32)
reported_flag = False
try:
    output_ms = ms.mint.linalg.inv(input_ms)
except Exception as e:
    reported_flag = True
    e = str(e)
    assert "singular" in e

if reported_flag == False:
    assert "No error message was reported when the input is singular matrix." == 1

# 测试问题：输入非奇异矩阵，函数不报错。
```

测试失败关键信息：

```
FAILED test_inv.py::test_random_input - AssertionError: assert 'No error message was reported when the input is singular matrix.' == 1
```

原因分析：mindspore.mint.linalg.inv 这一函数接口没有对输入矩阵的行列式进行判断，而是直接进行矩阵求逆，因此在输入张量为奇异矩阵时不存在报错信息。

3.4 mindspore.mint.special.erfc 测试问题

这一接口在测试时并未发现问题，对应测试代码可在 PR 链接中查看。

3.5 mindspore.mint.special.expm1 测试问题

这一接口在测试时并未发现问题，对应测试代码可在 PR 链接中查看。