华为昇腾-昇思MindSpore《机器学习原理》精品课程

机器学习原理

精品课程

实验指导手册

版本:1.0



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2021。 保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  C:\Users\jwx341670\Desktop\华为标志 Huawei Logo 2018\竖版标志Vertical Version\PNG\HW_POS_RBG_Vertical-150ppi.png 和其他华为商标均为华为技术有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。  由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 华为技术有限公司 | |
| 地址： | 深圳市龙岗区坂田华为总部办公楼 邮编：518129 |
| 网址： | http://[e](http://e.huawei.com/).huawei.com |

目录

[22 基于MindSpore构建Min Pooling层 3](#_Toc129098819)

[22.1实验介绍 3](#_Toc129098820)

[22.1.1 数据集的介绍 4](#_Toc129098821)

[22.1.2 模型知识点的介绍 4](#_Toc129098822)

[22.2 实验环境要求 4](#_Toc129098823)

[22.3 实验详细设计与实现 5](#_Toc129098824)

[22.3.1 数据准备 5](#_Toc129098825)

[22.3.2 实验步骤 5](#_Toc129098826)

[22.4 实验总结 7](#_Toc129098827)

[22.5 实验任务与参考解答任务 7](#_Toc129098828)

[22.5.1 实验任务 7](#_Toc129098829)

[22.5.2 参考答案 8](#_Toc129098830)



# 基于MindSpore构建Min Pooling层

## 22.1实验介绍

在本实验中，我们将实现最小池化，包括最大池化API的调用，实现最小池化API。

* 掌握最大池化的原理与MindSpore中API的调用。
* 基于MindSpore实现二维最小池化。

### 数据集的介绍

简单起见，我们直接构建一个1x2x4x4的张量，调用np.random.randint函数生成随机数，值的范围在0到10。如下所示：

[[[[5. 8. 9. 5.]

[0. 0. 1. 7.]

[6. 9. 2. 4.]

[5. 2. 4. 2.]]

[[4. 7. 7. 9.]

[1. 7. 0. 6.]

[9. 9. 7. 6.]

[9. 1. 0. 1.]]]]

### 模型知识点的介绍

在池化过程中，MinPooling采样输出区域的最小值，MaxPooling输出区域的最大值，MaxPooling的原理如下：。

输入数据的维度为 ，MaxPooling在维度输出区域最大值。给定为为 和 ,运算如下所示：

其中和是池化核尺寸大小，和为池化操作的移动步长，和为最大池化输出的索引。

与MaxPooling相反，由于MinPooling使用场景极少，因此MindSpore中并未提供最小池化的API，但是提供了MaxPool，MindSpore框架中的MaxPool类对输入的多维数据进行二维的最大池化运算，因此可通过对输入数据的负值求最大池化，然后再求负值即可：，其中x为输入的数据。

## 实验环境要求

在动手进行实践之前，需要注意以下几点：

（1）确保实验环境正确安装，包括安装MindSpore。安装过程：首先登录[MindSpore官网安装页面](https://www.mindspore.cn/install)，根据安装指南下载安装包及查询相关文档。同时，官网环境安装也可以按下表说明找到对应环境搭建文档链接，根据环境搭建手册配置对应的实验环境。

（2）推荐使用交互式的计算环境Jupyter Notebook，其交互性强，易于可视化，适合频繁修改的数据分析实验环境。

（3）实验也可以在华为云一站式的AI开发平台ModelArts上完成。

（4）推荐实验环境：MindSpore版本=2.4；Python环境=3.11。

|  |  |  |  |
| --- | --- | --- | --- |
| **硬件平台** | **操作系统** | **软件环境** | **环境搭建链接** |
| CPU | Windows-x64 | MindSpore2.4 Python3.11 | [MindSpore环境搭建实验手册第二章2.1节和第三章3.1节](MindSpore环境搭建实验手册.docx) |
| GPU CUDA 10.1 | Linux-x86\_64 | MindSpore2.4 Python3.11 | [MindSpore环境搭建实验手册第二章2.2节和第三章3.1节](MindSpore环境搭建实验手册.docx) |
| Ascend 910 | Linux-x86\_64 | MindSpore2.4 Python3.11 | [MindSpore环境搭建实验手册第四章](MindSpore环境搭建实验手册.docx) |

## 实验详细设计与实现

### 数据准备

构建一个1x2x4x4的张量，调用np.random.randint函数生成随机数，值的范围在0到10。如下所示：

[[[[5. 8. 9. 5.]

[0. 0. 1. 7.]

[6. 9. 2. 4.]

[5. 2. 4. 2.]]

[[4. 7. 7. 9.]

[1. 7. 0. 6.]

[9. 9. 7. 6.]

[9. 1. 0. 1.]]]]

### 实验步骤

步骤1 导入Python库&模块并配置运行信息

# mindspore.nn用于构建神经网络中的预定义构建块或计算单元

import mindspore.nn as nn

# Numpy是Python中科学计算的核心库,它提供了一个高性能的多维数组对象，以及用于处理这些数组的工具

import numpy as np

# dtype用于创建MindSpore中的数据类型对象

from mindspore import dtype as mstype

# ops提供计算函数和构建神经网络的一些方法

from mindspore import ops

# Tensor是可放在gpu上加速的张量

from mindspore import Tensor

import mindspore

# Cell是MindSpore中神经网络的基本构成单元，模型或神经网络层应当继承该基类

from mindspore.nn import Cell

步骤2 定义参数变量

np.random.seed(1)

# 1x2x4x4的输入数据

x = Tensor(np.random.randint(0, 10, [1, 2, 4, 4]), mstype.float32)

步骤3 数据的读取和处理

无需读取处理，直接输入x即可。

步骤4 模型构建训练

步骤 4.1 定义类\_PoolND检查输入是否规范

class \_PoolNd(Cell):

def \_\_init\_\_(self, kernel\_size, stride, pad\_mode, data\_format="NCHW"):

super(\_PoolNd, self).\_\_init\_\_()

# 检查pad\_mode是否为VALID或SAME

if pad\_mode != 'VALID' and pad\_mode != 'SAME':

raise ValueError('The pad\_mode must be VALID or SAME')

self.pad\_mode = pad\_mode

# 检查data\_format是否为NCHW或NHWC

if data\_format !='NCHW' and data\_format != 'NHWC':

raise ValueError('The format must be NCHW or NHWC')

self.format = data\_format

# NHWC数据格式仅支持GPU

if mindspore.get\_context("device\_target") != "GPU" and self.format == "NHWC":

raise ValueError(f"For '{self.cls\_name}, the 'NHWC' format only support in GPU target, but got device "

f"target {mindspore.get\_context('device\_target')}.")

# 检查是否为int或tuple，且必须为正数

def \_check\_int\_or\_tuple(arg\_name, arg\_value):

error\_msg = f"For '{self.cls\_name}', the '{arg\_name}' must be an positive int number or " \

f"a tuple of two positive int numbers, but got {arg\_value}"

if isinstance(arg\_value, int):

if arg\_value <= 0:

raise ValueError(error\_msg)

elif isinstance(arg\_value, tuple):

if len(arg\_value) == 2:

for item in arg\_value:

if isinstance(item, int) and item > 0:

continue

raise ValueError(error\_msg)

else:

raise ValueError(error\_msg)

else:

raise ValueError(error\_msg)

return arg\_value

# kernel\_size是一个正数或两个正数的元组

self.kernel\_size = \_check\_int\_or\_tuple('kernel\_size', kernel\_size)

# stride是一个正数或两个正数的元组

self.stride = \_check\_int\_or\_tuple('stride', stride)

def construct(self, \*inputs):

pass

def extend\_repr(self):

return 'kernel\_size={kernel\_size}, stride={stride}, pad\_mode={pad\_mode}'.format(\*\*self.\_\_dict\_\_)

请理解类\_PoolNd做了什么。

步骤 4.2 调用MaxPool类实现二维最小池化

请定义class MinPool2d()，继承\_PoolNd类后直接对输入数据的负值进行最大池化，然后再求负值返回即可。

# 请补充代码

步骤5 模型预测

使用数据x进行最小池化和最大池化，观察输出结果是否符合预期

# 直接调用定义好的最小池化类

pool = MinPool2d(kernel\_size=3, stride=1)

pool\_max=nn.MaxPool2d(kernel\_size=3, stride=1)

print(x)

# 最小池化

output = pool(x)

print(output)

# 最大池化

print(pool\_max(x))

预期结果为：

[[[[5. 8. 9. 5.]

[0. 0. 1. 7.]

[6. 9. 2. 4.]

[5. 2. 4. 2.]]

[[4. 7. 7. 9.]

[1. 7. 0. 6.]

[9. 9. 7. 6.]

[9. 1. 0. 1.]]]]

[[[[0. 0.]

[0. 0.]]

[[0. 0.]

[0. 0.]]]]

[[[[9. 9.]

[9. 9.]]

[[9. 9.]

[9. 9.]]]]

## 实验总结

由于最小池化使用场景极少，因此MindSpore中并未提供API，所以通过最大池化转换对输入数据的负值进行处理，再求负值即可。

## 实验任务与参考解答任务

### 实验任务

1. 理解class \_PoolNd代码。

2. 定义类class MinPool2d(\_PoolNd)实现二维最小池化。

### 参考答案

1.class \_PoolNd运行流程

class \_PoolNd()会检查输入pad\_mode是否为VALID或SAME，数据格式是否为NCHW或NHWC，kernel\_size和stride是否为正值。其中数据格式NHWC仅支持GPU，kernel\_size和stride可以是一个值也可以是两个值。

2．定义类class MinPool2d(\_PoolNd)实现二维最小池化。

class MinPool2d(\_PoolNd):

def \_\_init\_\_(self, kernel\_size=1, stride=1, pad\_mode="valid", data\_format="NCHW"):

# 检查输入是否规范

super(MinPool2d, self).\_\_init\_\_(kernel\_size, stride, pad\_mode, data\_format)

# 最大池化

self.max\_pool = ops.MaxPool(kernel\_size=self.kernel\_size,

strides=self.stride,

pad\_mode=self.pad\_mode,

data\_format=self.format)

def construct(self, x):

# minpool=- maxpool(-data)

out = -self.max\_pool(-x)

return out