昇腾创新实践课

MindSpore基础操作

实验手册



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

[1 实验环境介绍 2](#_Toc119506210)

[1.1 实验介绍 2](#_Toc119506211)

[1.1.1 关于本实验 2](#_Toc119506212)

[1.1.2 实验环境介绍 2](#_Toc119506213)

[2 MindSpore基础操作 3](#_Toc119506214)

[2.1 实验介绍 3](#_Toc119506215)

[2.1.1 关于本实验 3](#_Toc119506216)

[2.1.2 实验目的 3](#_Toc119506217)

[2.1.3 背景知识 3](#_Toc119506218)

[2.1.4 实验设计 3](#_Toc119506219)

[2.2 实验过程 3](#_Toc119506220)

[2.2.1 MindSpore环境搭建 3](#_Toc119506221)

[2.2.2 张量和数据类型 4](#_Toc119506222)

[2.2.3 数据集加载 7](#_Toc119506223)

[2.2.4 全连接网络搭建 8](#_Toc119506224)

[2.2.5 模型训练与评估 10](#_Toc119506225)

[2.3 实验总结 11](#_Toc119506226)

# 实验环境介绍

## 实验介绍

### 关于本实验

MindSpore是一个全场景深度学习框架，旨在实现易开发、高效执行、全场景覆盖三大目标，其中易开发表现为API友好、调试难度低，高效执行包括计算效率、数据预处理效率和分布式训练效率，全场景则指框架同时支持云、边缘以及端侧场景。

本课程将介绍MindSpore的基础操作，通过学习本课程，学员将掌握MindSpore的基础模块功能，以及如何用MindSpore搭建简单神经网络的必备知识。

更多MindSpore基础操作练习，请查阅官网教程：

<https://www.mindspore.cn/tutorials/zh-CN/r1.7/index.html>

### 实验环境介绍

实验、介绍、难度、软件环境、硬件环境：

实验环境介绍

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验 | 实验介绍 | 难度 | 软件环境 | 开发环境 |
| MindSpore基础操作 | 介绍MindSpore的基础模块，搭建神经网络的必备知识 | 简单 | Python3.7 MindSpore1.7 | PC机 |

# MindSpore基础操作

## 实验介绍

### 关于本实验

本实验通过介绍MindSpore的数据结构与数据类型，MindSpore搭建神经网络用到的基础模块，比如数据集加载，神经网络搭建，模型训练与评估等，让学员熟悉MindSpore的基础用法，掌握MindSpore开发的简单流程。

### 实验目的

理解MindSpore开发基本流程。

理解MindSpore基础模块的功能。

掌握MindSpore的简单操作。

### 背景知识

神经网络知识，感知机，多层感知机，前向传播，反向传播，激活函数，损失函数，优化器，评估方法。

### 实验设计

1.张量和数据类型

2.数据集加载

3.全连接网络搭建

4.模型训练

5.模型评估

## 实验过程

### MindSpore环境搭建

请参考《MindSpore环境搭建实验手册》，完成**本地PC**的MindSpore r1.7安装。

### 张量和数据类型

张量（Tensor）是MindSpore网络运算中的基本数据结构。张量中的数据类型可参考dtype。

不同维度的张量分别表示不同的数据，0维张量表示标量，1维张量表示向量，2维张量表示矩阵，3维张量可以表示彩色图像的RGB三通道等等。

MindSpore张量支持不同的数据类型，包含int8、int16、int32、int64、uint8、uint16、uint32、uint64、float16、float32、float64、bool\_，与NumPy的数据类型一一对应。

在MindSpore的运算处理流程中，Python中的int数会被转换为定义的int64类型，float数会被转换为定义的float32类型。

指定MindSpore数据类型

导入MindSpore，设置Jupyter notebook的cell同时输出多行。

# 导入MindSpore

import mindspore

from mindspore import dtype

from mindspore import Tensor

# cell同时输出多行

from IPython.core.interactiveshell import InteractiveShell

InteractiveShell.ast\_node\_interactivity = "all"

指定数据类型。

# 指定数据类型

a = 1

type(a)

b = Tensor(a, dtype.float64)

b.dtype

输出：

int

mindspore.float64

张量构造

构造张量时，支持传入Tensor、float、int、bool、tuple、list和NumPy.array类型，其中tuple和list里只能存放float、int、bool类型数据。

Tensor初始化时，可指定dtype。如果没有指定dtype，初始值int、float、bool分别生成数据类型为mindspore.int32、mindspore.float32、mindspore.bool\_的0维Tensor， 初始值tuple和list生成的1维Tensor数据类型与tuple和list里存放的数据类型相对应，如果包含多种不同类型的数据，则按照优先级：bool < int < float，选择相对优先级最高类型所对应的mindspore数据类型。 如果初始值是Tensor，则生成的Tensor数据类型与其一致；如果初始值是NumPy.array，则生成的Tensor数据类型与之对应。

用数组创建张量

代码：

import numpy as np

from mindspore import Tensor

# 用数组创建张量

x = Tensor(np.array([[1, 2], [3, 4]]), dtype.int32)

x

输出：

Tensor(shape=[2, 2], dtype=Int32, value=

[[1, 2],

[3, 4]])

用数值创建张量

代码：

# 用数值创建张量

y = Tensor(1.0, dtype.int32)

z = Tensor(2, dtype.int32)

y

z

输出：

Tensor(shape=[], dtype=Int32, value= 1)

Tensor(shape=[], dtype=Int32, value= 2)

用Bool创建张量

代码：

# 用Bool创建张量

m = Tensor(True, dtype.bool\_)

m

输出：

Tensor(shape=[], dtype=Bool, value= True)

用tuple创建张量

代码：

# 用tuple创建张量

n = Tensor((1, 2, 3), dtype.int16)

n

输出：

Tensor(shape=[3], dtype=Int16, value= [1, 2, 3])

用list创建张量

代码：

# 用list创建张量

p = Tensor([4.0, 5.0, 6.0], dtype.float64)

p

输出：

Tensor(shape=[3], dtype=Float64, value= [4.00000000e+000, 5.00000000e+000, 6.00000000e+000]

用常量创建张量

代码：

# 用常量创建张量

q = Tensor(1, dtype.float64)

q

输出：

Tensor(shape=[], dtype=Float64, value= 1)

张量的属性

张量的属性包括形状（shape）和数据类型（dtype）。

形状：Tensor的shape，是一个tuple。

数据类型：Tensor的dtype，是MindSpore的一个数据类型。

代码：

x = Tensor(np.array([[1, 2], [3, 4]]), dtype.int32)

x.shape # 形状

x.dtype # 数据类型

x.ndim # 维度

x.size # 大小

输出：

(2, 2)

mindspore.int32

2

4

张量的方法

asnumpy()：将Tensor转换为NumPy的array。

代码：

y = Tensor(np.array([[True, True], [False, False]]), dtype.bool\_)

# 将Tensor数据类型转换成NumPy

y\_array = y.asnumpy()

y

y\_array

输出：

Tensor(shape=[2, 2], dtype=Bool, value=

[[ True, True],

[False, False]])

array([[ True, True],

[False, False]])

### 数据集加载

MindSpore.dataset提供API来加载和处理各种常见的数据集，如MNIST, CIFAR-10, CIFAR-100, VOC, ImageNet, CelebA等。

加载MNIST数据集

下载MNIST数据集并在开发环境解压：<https://zhuanyejianshe.obs.cn-north-4.myhuaweicloud.com/chuangxinshijianke/cv-nlp/MNIST.zip>

mindspore.dataset.MnistDataset

代码：

import os

import mindspore.dataset as ds

import matplotlib.pyplot as plt

dataset\_dir = "./MNIST/train" # 数据集路径

# 从mnist dataset读取3张图片

mnist\_dataset = ds.MnistDataset(dataset\_dir=dataset\_dir, num\_samples=3)

# 设置图像大小

plt.figure(figsize=(8,8))

i = 1

# 打印3张子图

for dic in mnist\_dataset.create\_dict\_iterator(output\_numpy=True):

plt.subplot(3,3,i)

plt.imshow(dic['image'][:,:,0])

plt.axis('off')

i +=1

plt.show()

输出：



MindSpore还支持加载多种数据存储格式下的数据集，用户可以直接使用mindspore.dataset中对应的类加载磁盘中的数据文件。

加载NumPy数据集

mindspore.dataset.NumpySlicesDataset

代码：

import mindspore.dataset as ds

data = ds.NumpySlicesDataset([1, 2, 3], column\_names=["col\_1"])

for x in data.create\_dict\_iterator():

print(x)

输出：

{'col\_1': Tensor(shape=[], dtype=Int32, value= 2)}

{'col\_1': Tensor(shape=[], dtype=Int32, value= 3)}

{'col\_1': Tensor(shape=[], dtype=Int32, value= 1)}

### 全连接网络搭建

全连接神经网络

全连接层

mindspore.nn.Dense

in\_channels：输入通道

out\_channels：输出通道

weight\_init：权重初始化，Default 'normal'.

代码：

import mindspore.nn as nn

from mindspore import Tensor

# 构造输入张量

input = Tensor(np.array([[1, 1, 1], [2, 2, 2]]), mindspore.float32)

print(input)

# 构造全连接网络，输入通道为3，输出通道为3

net = nn.Dense(in\_channels=3, out\_channels=3, weight\_init=1)

output = net(input)

print(output)

输出：

[[1. 1. 1.]

[2. 2. 2.]]

[[3. 3. 3.]

[6. 6. 6.]]

激活函数

矫正线性单元激活函数

mindspore.nn.ReLU

代码：

input\_x = Tensor(np.array([-1, 2, -3, 2, -1]), mindspore.float16)

relu = nn.ReLU()

output = relu(input\_x)

print(output)

输出：

[0. 2. 0. 2. 0.]

搭建模型

所有神经网络的基类

mindspore.nn.Cell

代码：

import mindspore.nn as nn

class MyCell(nn.Cell):

# 定义算子

def \_\_init\_\_(self, ):

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

# 全连接层

self.fc = nn.Dense()

# 激活函数

self.relu = nn.ReLU()

# 建构网络

def construct(self, x):

x = self.fc(x)

x = self.relu(x)

return x

### 模型训练与评估

损失函数

交叉熵损失函数，用于分类模型。当标签数据不是one-hot编码形式时，需要输入参数sparse为True。

mindspore.nn.SoftmaxCrossEntropyWithLogits

代码：

import mindspore.nn as nn

# 交叉熵损失函数

loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True)

np.random.seed(0)

logits = Tensor(np.random.randint(0, 9, [1, 10]), mindspore.float32)

print(logits)

labels\_np = np.ones([1,]).astype(np.int32)

labels = Tensor(labels\_np)

print(labels)

output = loss(logits, labels)

print(output)

输出：

[[5. 0. 3. 3. 7. 3. 5. 2. 4. 7.]]

[1]

[7.868383]

优化器

深度学习优化算法大概常用的有SGD、Adam、Ftrl、lazyadam、Momentum、RMSprop、Lars、Proximal\_ada\_grad和lamb这几种。

动量优化器

mindspore.nn.Momentum

代码：

# optim = nn.Momentum(params, learning\_rate=0.1, momentum=0.9, weight\_decay=0.0)

模型编译

mindspore.Model

network：神经网络

loss\_fn：损失函数

optimizer：优化器

metrics：评估指标

代码：

from mindspore import Model

# 定义神经网络

net = nn.Dense(in\_channels=3, out\_channels=3, weight\_init=1)

# 定义损失函数

loss = nn.SoftmaxCrossEntropyWithLogits()

# 定义优化器

optim = nn.Momentum(params=net.trainable\_params(), learning\_rate=0.1, momentum=0.9)

# 模型编译

model = Model(network = net, loss\_fn=loss, optimizer=optim, metrics=None)

模型训练

model.train

epoch：训练次数

train\_dataset ：训练集

代码：

# model.train(epoch=10, train\_dataset=train\_dataset) # train\_dataset 是传入参数

模型评估

model.eval

valid\_dataset：测试集

# model.eval(valid\_dataset=test\_dataset) # test\_dataset 是传入参数

## 实验总结

本实验介绍了MindSpore的数据结构与类型，以及MindSpore搭建神经网络用到的基础模块，让学员学会如何加载数据集，搭建神经网络，训练和评估模型等，从易到难，由浅入深，让学员熟悉MindSpore的基础用法，掌握MindSpore开发的简单流程。