昇腾创新实践课

MindSpore 基础操作
实验手册

华为技术有限公司

商标声明

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.huawei.com

目录

1 实验环境介绍

1.1 实验介绍

1.1.1 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/r1.3/index.html

1.1.2 实验环境介绍

实验、介绍、难度、软件环境、硬件环境：

表 1-1 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 MindSpore1.2	PC机

2 MindSpore 基础操作

2.1 实验介绍

2.1.1 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。

2.1.2 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。

2.1.3 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。

2.1.4 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

2.2 实验过程

2.2.1 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。

步骤 1 指定 MindSpore 数据类型

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

```
%matplotlib inline
# 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。

代码：

```
# 指定数据类型
a = 1
type(a)
b = Tensor(a, dtype.float64)
b.dtype
```

输出：

```
int
mindspore.float64
```

步骤 2 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=[3], dtype=Float64, value= [4.00000000e+000, 5.00000000e+000, 6.00000000e+000])
```

步骤 3 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
```

步骤 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]])
```

2.2.2 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。

步骤 1 加载 MNIST 数据集

下载 MNIST 数据集：
https://zhuanyejianshe.obs.cn-north-4.myhuaweicloud.com/chuangxinshijianke/cv-nlp/MNIST.zip

使用 mindspore.dataset.MnistDataset.map 映射函数，将数据操作映射到数据集。

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。

步骤 2 加载 NumPy 数据集

使用 mindspore.dataset.NumpySlicesDataset 映射函数，将数据映射为 numpy 数据。

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)}
```

2.2.3 全连接网络搭建

步骤 1 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.]]
```

步骤 2 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.]
```

步骤 3 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
```

2.2.4 模型训练与评估

步骤 1 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]
```

步骤 2 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)
```

步骤 3 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)
```

步骤 4 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
```

步骤 5 模型评估

model.eval

- valid_dataset：测试集

```
# model.eval(valid_dataset=test_dataset)    # test_dataset 是传入参数
```

2.3 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。