深度学习基础

实验一



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2022 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 实验介绍 3](#_Toc81357920)

[1.1 实验目的 3](#_Toc81357921)

[1.2 实验清单 3](#_Toc81357922)

[1.3 实验开发环境 4](#_Toc81357923)

[2 线性回归模拟实验 5](#_Toc81357924)

[2.1 实验介绍 5](#_Toc81357925)

[2.1.1 简介 5](#_Toc81357926)

[2.1.2 实验目的 5](#_Toc81357927)

[2.2 实验环境要求 5](#_Toc81357928)

[2.3 实验步骤 5](#_Toc81357929)

[2.3.1 导入MindSpore模块和辅助模块 5](#_Toc81357930)

[2.3.2 生成模拟数据 6](#_Toc81357931)

[2.3.3 建模 6](#_Toc81357932)

[2.3.4 使用模拟数据训练模型 6](#_Toc81357933)

[2.3.5 使用训练好的模型进行预测 7](#_Toc81357934)

[2.3.6 可视化 7](#_Toc81357935)

[3 鸢尾花二分类实验 9](#_Toc81357936)

[3.1 实验介绍 9](#_Toc81357937)

[3.1.1 简介 9](#_Toc81357938)

[3.1.2 实验目的 9](#_Toc81357939)

[3.2 实验环境要求 9](#_Toc81357940)

[3.3 实验总体设计 10](#_Toc81357941)

[3.4 实验过程 10](#_Toc81357942)

[3.4.1 数据准备 10](#_Toc81357943)

[3.4.2 数据读取与处理 11](#_Toc81357944)

[3.4.3 模型建立与训练 13](#_Toc81357945)

[3.4.4 模型评估 15](#_Toc81357946)

[3.5 实验小结 16](#_Toc81357947)

[3.6 创新设计 16](#_Toc81357948)

[4 红酒分类实验 17](#_Toc81357949)

[4.1 实验介绍 17](#_Toc81357950)

[4.1.1 简介 17](#_Toc81357951)

[4.1.2 实验目的 17](#_Toc81357952)

[4.2 实验环境要求 17](#_Toc81357953)

[4.3 实验过程 18](#_Toc81357954)

[4.3.1 数据准备 18](#_Toc81357955)

[4.3.2 数据读取与处理 19](#_Toc81357956)

[4.3.3 计算距离 21](#_Toc81357957)

[4.3.4 预测 22](#_Toc81357958)

[4.4 实验小结 22](#_Toc81357959)

[5 附录：ModelArts开发环境搭建 23](#_Toc81357960)

# 实验介绍

机器学习分为监督学习、无监督学习、半监督学习、强化学习。监督学习是指利用一组已知类别的样本调整分类器的参数，使其达到所要求性能的过程，也称为监督训练或有教师学习。分类和回归是监督学习中的两种典型任务。

本章实验涉及线性回归、逻辑回归、KNN回归等算法，通过不同算法的效果对比来加深对算法的理解。

本章实验难度为：初级。

初级实验：线性回归模拟实验、鸢尾花二分类实验、红酒分类实验

## 实验目的

本章实验分为回归与分类两部分，通过本章实验的学习，您将能够：

掌握使用MindSpore进行线性回归

掌握使用MindSpore进行逻辑回归

掌握使用MindSpore进行KNN分类

掌握回归与分类任务的区别与流程

掌握线性回归、逻辑回归、KNN分类等算法的原理与使用

掌握模型评估的方法

## 实验清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验** | **简述** | **难度** | **软件环境** | **开发环境** |
| 线性回归模拟实验 | 基于随机生成的数据，进行线性回归实验，实现对数据的拟合 | 初级 | MindSpore1.7 | ModelArts、PC 64bit |
| 鸢尾花二分类预测（分类） | 基于两个种类的鸢尾花数据，进行逻辑回归实验，实现鸢尾花的二分类预测 | 初级 | MindSpore1.7 | ModelArts、PC 64bit |
| 红酒分类实验 | 基于wine数据集，使用KNN算法实现不同品种的聚类 | 初级 | MindSpore1.7 | ModelArts、PC 64bit |

## 实验开发环境

MindSpore-1.7

若选择在华为云ModelArts上快速搭建开发环境，可参考文末附录：ModelArts开发环境搭建。

# 线性回归模拟实验

## 实验介绍

### 简介

线性回归（Linear Regression）是机器学习最经典的算法之一，具有如下特点：

自变量服从正态分布；

因变量是连续性数值变量；

自变量和因变量呈线性关系。

本实验主要介绍使用MindSpore在模拟数据上进行线性回归实验，分析自变量和因变量之间的线性关系，即求得一个线性函数。

### 实验目的

了解线性回归的基本概念和问题模拟；

了解如何使用MindSpore进行线性回归实验。

## 实验环境要求

MindSpore 1.7（MindSpore版本会定期更新，本指导也会定期刷新，与版本配套）；

华为云ModelArts：ModelArts是华为云提供的面向开发者的一站式AI开发平台，集成了昇腾AI处理器资源池，用户可以在该平台下体验MindSpore。环境搭建可参考文末附录。

## 实验步骤

### 导入MindSpore模块和辅助模块

代码：

import os

# os.environ['DEVICE\_ID'] = '0'

import numpy as np

import mindspore as ms

from mindspore import nn

from mindspore import context

#当前实验选择算力为Ascend，如果在本地体验，参数device\_target设置为"CPU”

context.set\_context(mode=context.GRAPH\_MODE, device\_target="Ascend")

### 生成模拟数据

根据线性函数y = -5 \* x + 0.1生成模拟数据，并在其中加入少许扰动。

代码：

x = np.arange(-5, 5, 0.3)[:32].reshape((32, 1))

y = -5 \* x + 0.1 \* np.random.normal(loc=0.0, scale=20.0, size=x.shape)

### 建模

使用MindSpore提供的nn.Dense(1, 1)算子(https://www.mindspore.cn/api/zh-CN/0.2.0-alpha/api/python/mindspore/mindspore.nn.html#mindspore.nn.Dense)作为线性模型，其中(1, 1)表示线性模型的输入和输出皆是1维，即w是1x1的矩阵。算子会随机初始化权重w和偏置b。

y = w \* x + b

采用均方差（Mean Squared Error, MSE）作为损失函数。采用随机梯度下降（Stochastic Gradient Descent, SGD）对模型进行优化。

代码：

net = nn.Dense(1, 1)

loss\_fn = nn.loss.MSELoss()

opt = nn.optim.SGD(net.trainable\_params(), learning\_rate=0.01)

with\_loss = nn.WithLossCell(net, loss\_fn)

train\_step = nn.TrainOneStepCell(with\_loss, opt).set\_train()

### 使用模拟数据训练模型

代码：

for epoch in range(20):

loss = train\_step(ms.Tensor(x, ms.float32), ms.Tensor(y, ms.float32))

print('epoch: {0}, loss is {1}'.format(epoch, loss))

输出：

epoch: 0, loss is 191.03662

epoch: 1, loss is 137.56923

epoch: 2, loss is 99.519455

epoch: 3, loss is 72.41922

epoch: 4, loss is 53.13565

epoch: 5, loss is 39.367752

epoch: 6, loss is 29.570751

epoch: 7, loss is 22.636707

epoch: 8, loss is 17.659052

epoch: 9, loss is 14.094089

epoch: 10, loss is 11.589828

epoch: 11, loss is 9.775642

epoch: 12, loss is 8.497474

epoch: 13, loss is 7.5611076

epoch: 14, loss is 6.9201765

epoch: 15, loss is 6.452571

epoch: 16, loss is 6.100116

epoch: 17, loss is 5.856675

epoch: 18, loss is 5.6804886

epoch: 19, loss is 5.5560365

### 使用训练好的模型进行预测

训练一定的轮次后，得到的模型已经十分接近真实的线性函数了，使用训练好的模型进行预测。

代码：

wb = [x.asnumpy() for x in net.trainable\_params()]

w, b = np.squeeze(wb[0]), np.squeeze(wb[1])

print('The true linear function is y = -5 \* x + 0.1')

print('The trained linear model is y = {0} \* x + {1}'.format(w, b))

for i in range(-10, 11, 5):

print('x = {0}, predicted y = {1}'.format(i, net(ms.Tensor([[i]], ms.float32))))

输出：

The true linear function is y = -5 \* x + 0.1

The trained linear model is y = -4.842680931091309 \* x + 0.03442131727933884

x = -10, predicted y = [[49.714813]]

x = -5, predicted y = [[24.974724]]

x = 0, predicted y = [[0.23463698]]

x = 5, predicted y = [[-24.505451]]

x = 10, predicted y = [[-49.245537]]

### 可视化

模拟的样本数据、真实的线性函数和训练得到的线性模型，如下图所示：

代码：

from matplotlib import pyplot as plt

%matplotlib inline

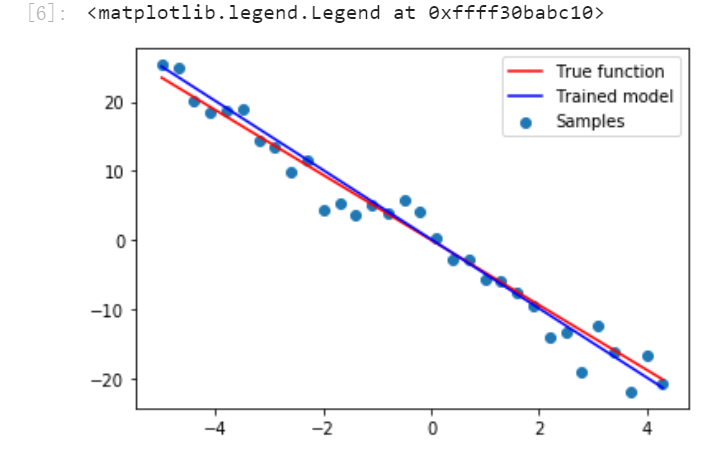
plt.scatter(x, y, label='Samples')

plt.plot(x, w \* x + b, c='r', label='True function')

plt.plot(x, -5 \* x + 0.1, c='b', label='Trained model')

plt.legend()

输出：



模拟数据展示

# 鸢尾花二分类实验

## 实验介绍

### 简介

逻辑回归（Logistic Regression）是机器学习最经典的算法之一，与线性回归有很多不同，这两种回归都属于广义线性回归（Generalized Linear Regression）的范畴。逻辑回归具有如下特点：

逻辑回归对自变量分布没有要求；

因变量是离散型变量，即分类变量；

逻辑回归分析的是因变量取某个值的概率与自变量的关系。

本实验主要介绍使用MindSpore在2分类数据集上进行逻辑回归实验，分析自变量和因变量（概率）之间的关系，即求得一个概率函数。

### 实验目的

了解逻辑回归的基本概念；

了解如何使用MindSpore进行逻辑回归实验。

## 实验环境要求

MindSpore1.7 （MindSpore版本会定期更新，本指导也会定期刷新，与版本配套）；

华为云ModelArts：ModelArts是华为云提供的面向开发者的一站式AI开发平台，集成了昇腾AI处理器资源池，用户可以在该平台下体验MindSpore。环境搭建可参考文末附录。

## 实验总体设计

## 实验过程

### 数据准备

下载数据

Iris数据集是模式识别最著名的数据集之一。数据集包含3类，每类50个实例，其中每个类都涉及一种鸢尾植物。 第一类与后两类可线性分离，后两类之间不能线性分离，所以本实验取前两类数据，做一个2分类数据集。

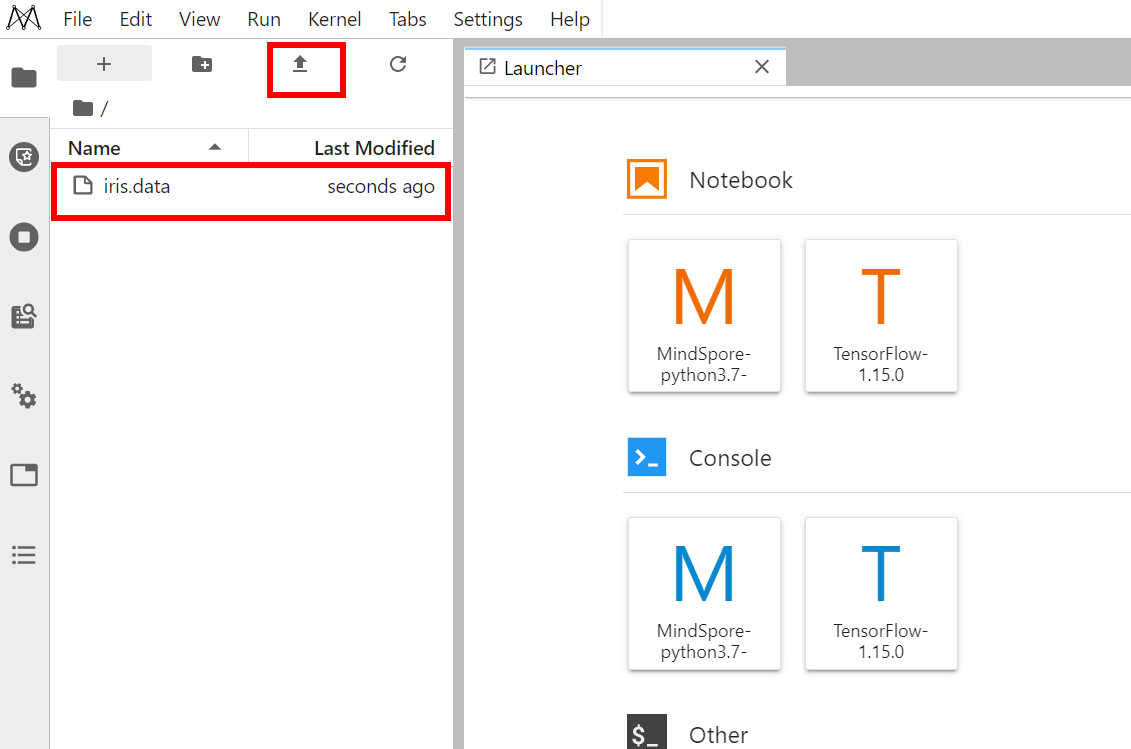
Iris数据集的官网：[Iris Data Set](http://archive.ics.uci.edu/ml/datasets/Iris)。

方式一，从Iris数据集官网下载[iris.data文件](http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data)。

方式二，从华为云OBS中下载[iris.data文件]( https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/MachineLearning/iris.data)。

上传数据到实验环境

在新建的notebook实验环境中，通过如图所示的“上传”按钮，然后选择自己本地已下载好的数据文件“iris.data”，将数据文件上传到实验环境中。



上传数据集到实验环境

### 数据读取与处理

导入MindSpore模块和辅助模块

import os

# os.environ['DEVICE\_ID'] = '6'

import csv

import numpy as np

import mindspore as ms

from mindspore import nn

from mindspore import context

from mindspore import dataset

from mindspore.train.callback import LossMonitor

from mindspore.common.api import ms\_function

from mindspore.ops import operations as P

#当前实验选择算力为Ascend，如果在本地体验，参数device\_target设置为"CPU”

context.set\_context(mode=context.GRAPH\_MODE, device\_target="Ascend")

读取Iris数据集，并查看部分数据

with open('iris.data') as csv\_file:

data = list(csv.reader(csv\_file, delimiter=','))

print(data[40:60]) # 打印部分数据

输出：

[['5.0', '3.5', '1.3', '0.3', 'Iris-setosa'], ['4.5', '2.3', '1.3', '0.3', 'Iris-setosa'], ['4.4', '3.2', '1.3', '0.2', 'Iris-setosa'], ['5.0', '3.5', '1.6', '0.6', 'Iris-setosa'], ['5.1', '3.8', '1.9', '0.4', 'Iris-setosa'], ['4.8', '3.0', '1.4', '0.3', 'Iris-setosa'], ['5.1', '3.8', '1.6', '0.2', 'Iris-setosa'], ['4.6', '3.2', '1.4', '0.2', 'Iris-setosa'], ['5.3', '3.7', '1.5', '0.2', 'Iris-setosa'], ['5.0', '3.3', '1.4', '0.2', 'Iris-setosa'], ['7.0', '3.2', '4.7', '1.4', 'Iris-versicolor'], ['6.4', '3.2', '4.5', '1.5', 'Iris-versicolor'], ['6.9', '3.1', '4.9', '1.5', 'Iris-versicolor'], ['5.5', '2.3', '4.0', '1.3', 'Iris-versicolor'], ['6.5', '2.8', '4.6', '1.5', 'Iris-versicolor'], ['5.7', '2.8', '4.5', '1.3', 'Iris-versicolor'], ['6.3', '3.3', '4.7', '1.6', 'Iris-versicolor'], ['4.9', '2.4', '3.3', '1.0', 'Iris-versicolor'], ['6.6', '2.9', '4.6', '1.3', 'Iris-versicolor'], ['5.2', '2.7', '3.9', '1.4', 'Iris-versicolor']]

抽取样本

取前两类样本（共100条），将数据集的4个属性作为自变量X。将数据集的2个类别映射为{0, 1}，作为因变量Y。

label\_map = {

'Iris-setosa': 0,

'Iris-versicolor': 1,

}

X = np.array([[float(x) for x in s[:-1]] for s in data[:100]], np.float32)

Y = np.array([[label\_map[s[-1]]] for s in data[:100]], np.float32)

样本可视化

取样本的前两个属性进行2维可视化，可以看到在前两个属性上两类样本是线性可分的。

from matplotlib import pyplot as plt

%matplotlib inline

plt.scatter(X[:50, 0], X[:50, 1], label='Iris-setosa')

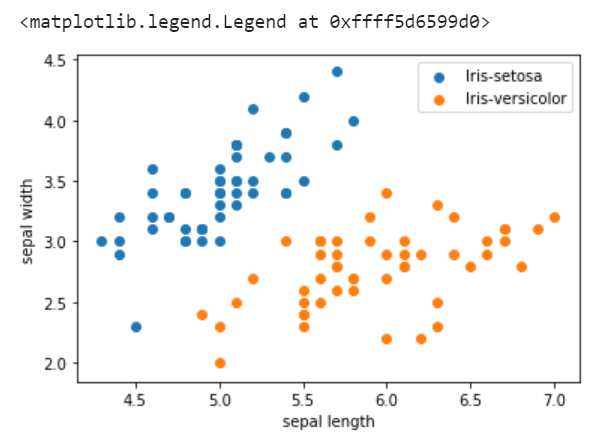
plt.scatter(X[50:, 0], X[50:, 1], label='Iris-versicolor')

plt.xlabel('sepal length')

plt.ylabel('sepal width')

plt.legend()

输出：



样本数据可视化

分割数据集

将数据集按8:2划分为训练集和验证集：

train\_idx = np.random.choice(100, 80, replace=False)

test\_idx = np.array(list(set(range(100)) - set(train\_idx)))

X\_train, Y\_train = X[train\_idx], Y[train\_idx]

X\_test, Y\_test = X[test\_idx], Y[test\_idx]

数据类型转换

使用MindSpore的GeneratorDataset接口将numpy.ndarray类型的数据转换为Dataset：

XY\_train = list(zip(X\_train, Y\_train))

ds\_train = dataset.GeneratorDataset(XY\_train, ['x', 'y'])

# ds\_train.set\_dataset\_size(80)

ds\_train = ds\_train.shuffle(buffer\_size=80).batch(32, drop\_remainder=True)

### 模型建立与训练

可视化逻辑回归函数

逻辑回归常用的联系函数是Sigmoid（S形函数），Sigmoid函数如下图所示，可以将连续值映射到{0, 1}，同时也是单调可微的。

coor\_x = np.arange(-10, 11, dtype=np.float32)

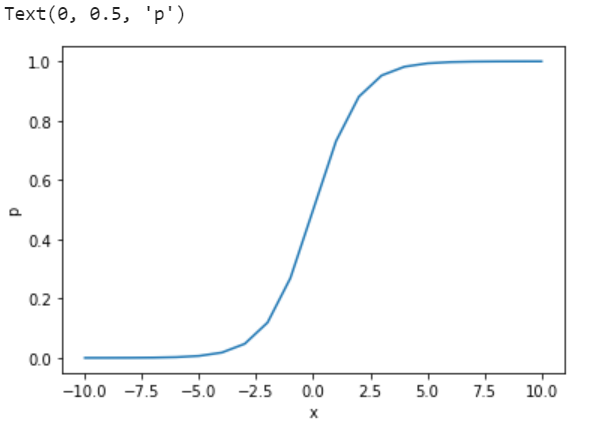
coor\_y = nn.Sigmoid()(ms.Tensor(coor\_x)).asnumpy()

plt.plot(coor\_x, coor\_y)

plt.xlabel('x')

plt.ylabel('p')

输出：



Sigmoid函数图示

建模

使用MindSpore提供的nn.Dense(4, 1)算子（https://www.mindspore.cn/api/zh-CN/0.2.0-alpha/api/python/mindspore/mindspore.nn.html#mindspore.nn.Dense)作为线性部分，其中(4, 1)表示每个样本的输入是含4个元素的向量，输出是含1个元素的向量，即W是1x4的矩阵。算子会随机初始化权重W和偏置b。使用SigmoidCrossEntropyWithLogits算子(https://www.mindspore.cn/api/zh-CN/0.3.0-alpha/api/python/mindspore/mindspore.ops.operations.html?#mindspore.ops.operations.SigmoidCrossEntropyWithLogits)作为非线性部分：

对于每个样本N\_i，模型的计算方式如下：

其中，是1D Tensor（含4个元素），是1D Tensor（含1个元素），是真实类别（2个类别{0, 1}中的一个），是1D Tensor（含1个元素，表示属于类别1的概率，值域为[0, 1]），loss是标量。

# 自定义Loss

class Loss(nn.Cell):

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

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

self.sigmoid\_cross\_entropy\_with\_logits = P.SigmoidCrossEntropyWithLogits()

self.reduce\_mean = P.ReduceMean(keep\_dims=False)

def construct(self, x, y):

loss = self.sigmoid\_cross\_entropy\_with\_logits(x, y)

return self.reduce\_mean(loss, -1)

net = nn.Dense(4, 1)

loss = Loss()

opt = nn.optim.SGD(net.trainable\_params(), learning\_rate=0.003)

模型训练

使用2分类的Iris数据集对模型进行几代（Epoch）训练：

代码：

model = ms.train.Model(net, loss, opt)

model.train(10, ds\_train, callbacks=[LossMonitor(per\_print\_times=ds\_train.get\_dataset\_size())], dataset\_sink\_mode=False)

输出：

epoch: 1 step: 2, loss is 0.6513498

epoch: 2 step: 2, loss is 0.5780734

epoch: 3 step: 2, loss is 0.5097989

epoch: 4 step: 2, loss is 0.4643281

epoch: 5 step: 2, loss is 0.41714883

epoch: 6 step: 2, loss is 0.39717123

epoch: 7 step: 2, loss is 0.3307726

epoch: 8 step: 2, loss is 0.29960862

epoch: 9 step: 2, loss is 0.29403976

epoch: 10 step: 2, loss is 0.27250308

### 模型评估

然后计算模型在测试集上精度，测试集上的准确率达到了1.0左右，即逻辑回归模型学会了区分2类鸢尾花。

代码：

x = model.predict(ms.Tensor(X\_test)).asnumpy()

pred = np.round(1 / (1 + np.exp(-x)))

correct = np.equal(pred, Y\_test)

acc = np.mean(correct)

print('Test accuracy is', acc)

输出：

Test accuracy is 1.0

## 实验小结

本实验使用MindSpore实现了逻辑回归，用来解决2分类问题。在Iris数据集上进行训练后，所得的模型可以很好的表示每个样本类别y和属性x的关系。

## 创新设计

请使用Softmax函数作为概率映射函数，对完整的Iris数据集实现多分类任务。

# 红酒分类实验

## 实验介绍

### 简介

K近邻算法（K-Nearest-Neighbor, KNN）是一种用于分类和回归的非参数统计方法，是机器学习最基础的算法之一。KNN也是有监督学习中的分类算法。KNN的三个基本要素：

K值，一个样本的分类是由K个邻居的“多数表决”确定的。K值越小，容易受噪声影响，反之，会使类别之间的界限变得模糊。

距离度量，反映了特征空间中两个样本间的相似度，距离越小，越相似。常用的有Lp距离（p=2时，即为欧式距离）、曼哈顿距离、海明距离等。

分类决策规则，通常是多数表决，或者基于距离加权的多数表决（权值与距离成反比）。

本实验主要介绍使用MindSpore在部分wine数据集上进行KNN实验。

### 实验目的

了解KNN的基本概念；

了解如何使用MindSpore进行KNN实验。

## 实验环境要求

MindSpore1.7 （MindSpore版本会定期更新，本指导也会定期刷新，与版本配套）；

华为云ModelArts：ModelArts是华为云提供的面向开发者的一站式AI开发平台，集成了昇腾AI处理器资源池，用户可以在该平台下体验MindSpore。环境搭建可参考文末附录。

## 实验过程

### 数据准备

下载数据

Wine数据集是模式识别最著名的数据集之一，Wine数据集的官网：[Wine Data Set](http://archive.ics.uci.edu/ml/datasets/Wine)。这些数据是对来自意大利同一地区但来自三个不同品种的葡萄酒进行化学分析的结果。数据集分析了三种葡萄酒中每种所含13种成分的量。这些13种属性如下表所示:

|  |  |  |
| --- | --- | --- |
| 序号 | 成分（英文） | 成分（中文） |
| 1 | Alcohol | 酒精 |
| 2 | Malic acid | 苹果酸 |
| 3 | Ash | 灰 |
| 4 | Alcalinity of ash | 灰的碱度 |
| 5 | Magnesium | 镁 |
| 6 | Total phenols | 总酚 |
| 7 | Flavanoids | 类黄酮 |
| 8 | Nonflavanoid phenols | 非黄酮酚 |
| 9 | Proanthocyanins | 原花青素 |
| 10 | Color intensity | 色彩强度 |
| 11 | Hue | 色调 |
| 12 | OD280/OD315 of diluted wines | 稀释酒的OD280/OD315 |
| 13 | Proline | 脯氨酸 |

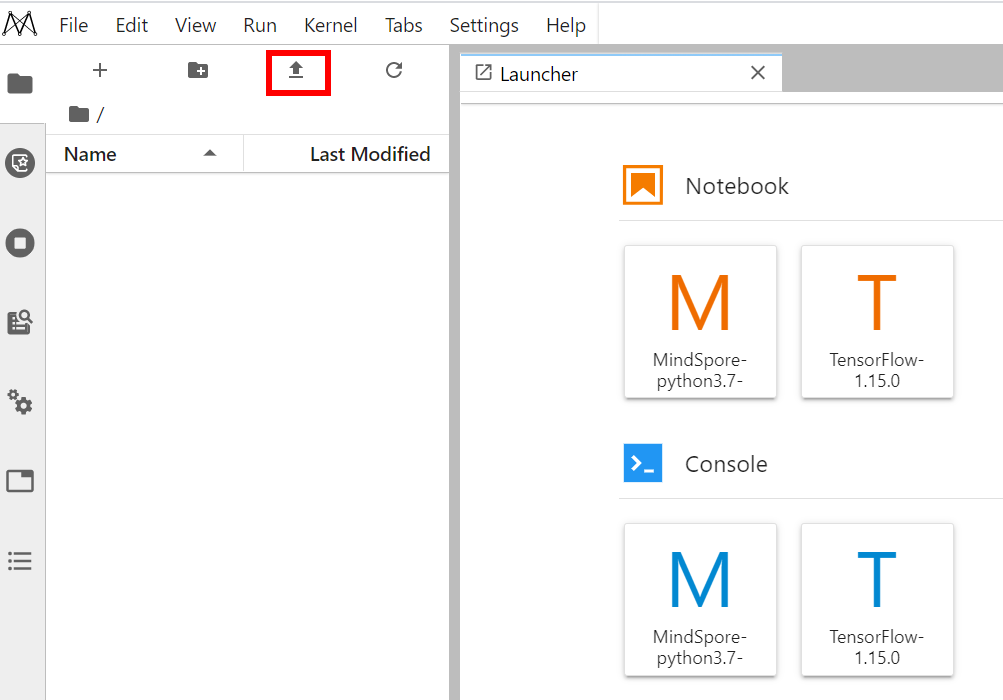
葡萄酒成分表

方式一：从Wine数据集官网下载[wine.data文件](http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data)。

方式二：从华为云OBS中下载[wine.data文件]( https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/MachineLearning/wine.data)。

上传数据到实验环境

在新建的notebook实验环境中，通过如图所示的“上传”按钮，然后选择自己本地已下载好的数据文件“wine.data”，将数据文件上传到实验环境中。



上传数据集到实验环境

### 数据读取与处理

导入MindSpore模块和辅助模块

import os

# os.environ['DEVICE\_ID'] = '4'

import csv

import numpy as np

import mindspore as ms

from mindspore import context

from mindspore import nn

from mindspore.ops import operations as P

from mindspore.ops import functional as F

context.set\_context(device\_target="Ascend")

读取Wine数据集wine.data，并查看部分数据。

输入：

with open('wine.data') as csv\_file:

data = list(csv.reader(csv\_file, delimiter=','))

print(data[56:62]+data[130:133]) # 打印部分数据

输出：

[['1', '14.22', '1.7', '2.3', '16.3', '118', '3.2', '3', '.26', '2.03', '6.38', '.94', '3.31', '970'], ['1', '13.29', '1.97', '2.68', '16.8', '102', '3', '3.23', '.31', '1.66', '6', '1.07', '2.84', '1270'], ['1', '13.72', '1.43', '2.5', '16.7', '108', '3.4', '3.67', '.19', '2.04', '6.8', '.89', '2.87', '1285'], ['2', '12.37', '.94', '1.36', '10.6', '88', '1.98', '.57', '.28', '.42', '1.95', '1.05', '1.82', '520'], ['2', '12.33', '1.1', '2.28', '16', '101', '2.05', '1.09', '.63', '.41', '3.27', '1.25', '1.67', '680'], ['2', '12.64', '1.36', '2.02', '16.8', '100', '2.02', '1.41', '.53', '.62', '5.75', '.98', '1.59', '450'], ['3', '12.86', '1.35', '2.32', '18', '122', '1.51', '1.25', '.21', '.94', '4.1', '.76', '1.29', '630'], ['3', '12.88', '2.99', '2.4', '20', '104', '1.3', '1.22', '.24', '.83', '5.4', '.74', '1.42', '530'], ['3', '12.81', '2.31', '2.4', '24', '98', '1.15', '1.09', '.27', '.83', '5.7', '.66', '1.36', '560']]

取三类样本（共178条），将数据集的13个属性作为自变量X。将数据集的3个类别作为因变量Y。

X = np.array([[float(x) for x in s[1:]] for s in data[:178]], np.float32)

Y = np.array([s[0] for s in data[:178]], np.int32)

取样本的某两个属性进行2维可视化，可以看到在某两个属性上样本的分布情况以及可分性。

from matplotlib import pyplot as plt

%matplotlib inline

attrs = ['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols',

'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue',

'OD280/OD315 of diluted wines', 'Proline']

plt.figure(figsize=(10, 8))

for i in range(0, 4):

plt.subplot(2, 2, i+1)

a1, a2 = 2 \* i, 2 \* i + 1

plt.scatter(X[:59, a1], X[:59, a2], label='1')

plt.scatter(X[59:130, a1], X[59:130, a2], label='2')

plt.scatter(X[130:, a1], X[130:, a2], label='3')

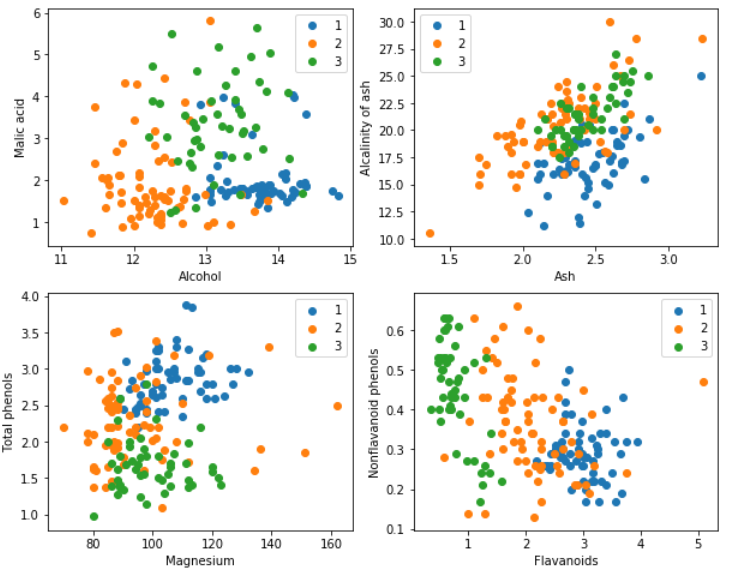
plt.xlabel(attrs[a1])

plt.ylabel(attrs[a2])

plt.legend()

plt.show()

输出：



查看数据分布

将数据集按128:50划分为训练集（已知类别样本）和验证集（待验证样本）：

train\_idx = np.random.choice(178, 128, replace=False)

test\_idx = np.array(list(set(range(178)) - set(train\_idx)))

X\_train, Y\_train = X[train\_idx], Y[train\_idx]

X\_test, Y\_test = X[test\_idx], Y[test\_idx]

### 计算距离

利用MindSpore提供的tile, suqare, ReduceSum, sqrt, TopK等算子，通过矩阵运算的方式同时计算输入样本x和已明确分类的其他样本X\_train的距离，并计算出top k近邻。

class KnnNet(nn.Cell):

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

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

self.tile = P.Tile()

self.sum = P.ReduceSum()

self.topk = P.TopK()

self.k = k

def construct(self, x, X\_train):

# Tile input x to match the number of samples in X\_train

x\_tile = self.tile(x, (128, 1))

square\_diff = F.square(x\_tile - X\_train)

square\_dist = self.sum(square\_diff, 1)

dist = F.sqrt(square\_dist)

# -dist mean the bigger the value is, the nearer the samples are

values, indices = self.topk(-dist, self.k)

return indices

def knn(knn\_net, x, X\_train, Y\_train):

x, X\_train = ms.Tensor(x), ms.Tensor(X\_train)

indices = knn\_net(x, X\_train)

topk\_cls = [0]\*len(indices.asnumpy())

for idx in indices.asnumpy():

topk\_cls[Y\_train[idx]] += 1

cls = np.argmax(topk\_cls)

return cls

### 预测

在验证集上验证KNN算法的有效性，取k = 5，验证精度接近80%，说明KNN算法在该3分类任务上有效，能根据酒的13种属性判断出酒的品种。

acc = 0

knn\_net = KnnNet(5)

for x, y in zip(X\_test, Y\_test):

pred = knn(knn\_net, x, X\_train, Y\_train)

acc += (pred == y)

print('label: %d, prediction: %s' % (y, pred))

print('Validation accuracy is %f' % (acc/len(Y\_test)))

输出：

label: 1, prediction: 1

label: 3, prediction: 2

label: 3, prediction: 3

label: 1, prediction: 1

……

Validation accuracy is 0.800000

## 实验小结

本实验使用MindSpore实现了KNN算法，用来解决3分类问题。取wine数据集上的3类样本，分为已知类别样本和待验证样本，从验证结果可以看出KNN算法在该任务上有效，能根据酒的13种属性判断出酒的品种。

# 附录：ModelArts开发环境搭建

* ModelArts平台：Mindspore-1.7

进入ModelArts

在[华为云](https://www.huaweicloud.com/)主页搜索Modelarts，点击“AI开发平台ModelArts”中的“进入控制台”。

图形用户界面, 文本, 应用程序

描述已自动生成

选择训练作业

选择“北京四”地区，在左侧下拉框中点击“开发环境”中的“Notebook”：

图形用户界面, 应用程序

描述已自动生成

创建Notebook

点击创建按钮来创建一个新的Notebook，选择如下配置：

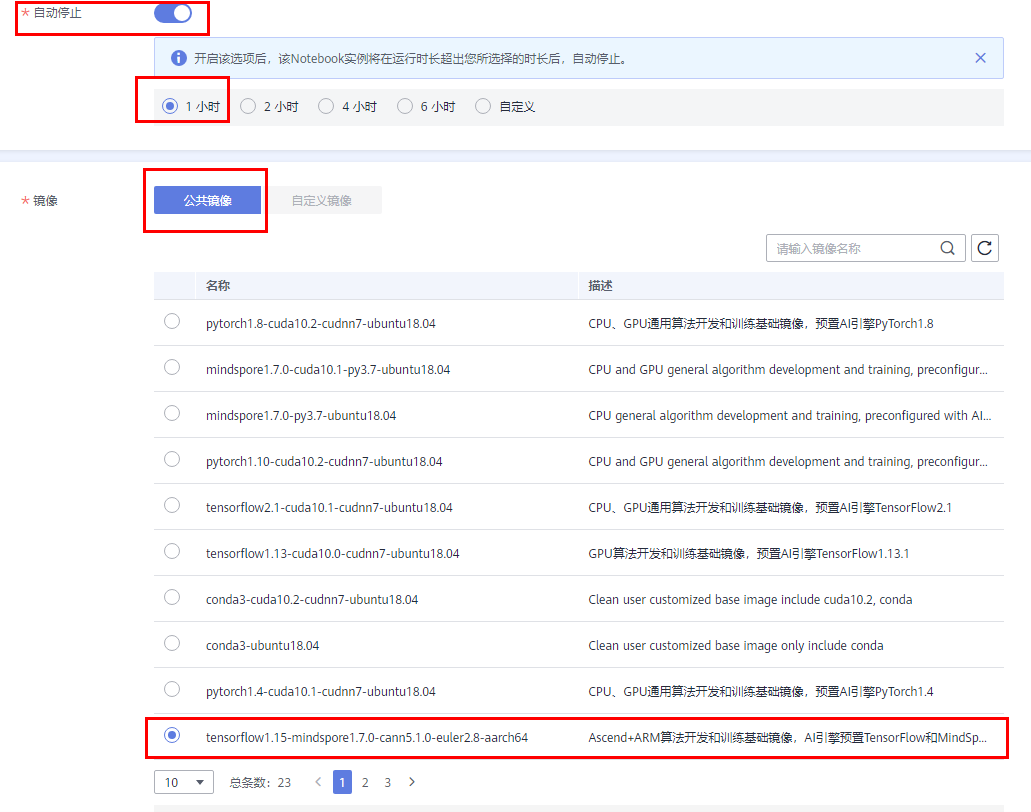
名称：自定义。

**自动停止：建议1小时。**

工作环境：Ascend+ARM算法开发和训练基础镜像。

规格：Ascend: 1\*Ascend910|CPU: 24核 96GB。

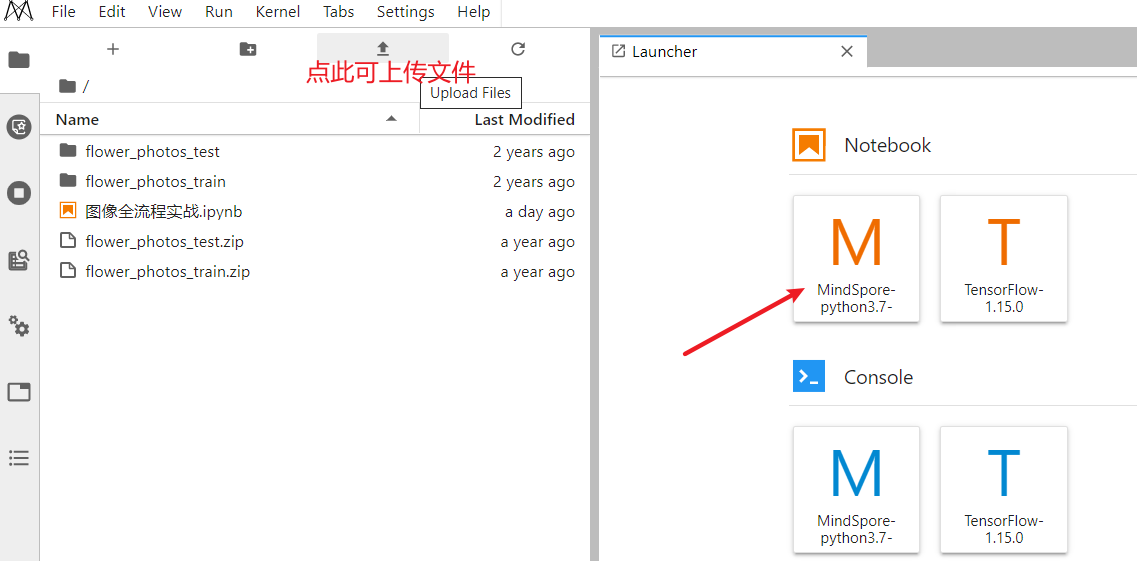
存储配置：默认存储。

点击“下一步”，确认规格如下后选择提交：



启动Notebook进入开发环境

当Notebook状态变为“运行中”时，点击右侧“打开”按钮打开Notebook。打开后选择右侧“MindSpore”相关的 Notebook环境：



停止实验环境

试验完成之后请及时停止实验环境，避免资源浪费，如下图：

