

《基于 MindSpore 的鸢尾花分 类实验》



华为技术有限公司

版权所有 © 华为技术有限公司 2024 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



HUAWEI 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 鸢尾花二分类实验 | 2 |
| 1.1 实验介绍 | 2 |
| 1.1.1 简介 | 2 |
| 1.1.2 实验目的 | 2 |
| 1.2 实验环境搭建 | 2 |
| 1.3 实验总体设计 | 3 |
| 1.4 实验过程 | 3 |
| 1.4.1 数据准备 | 3 |
| 1.4.2 数据读取与处理 | 4 |
| 1.4.3 模型建立与训练 | 6 |
| 1.4.4 模型评估 | 8 |
| 1.5 实验小结 | 9 |
| 1.6 创新设计 | 9 |



1 鸢尾花二分类实验

1.1 实验介绍

1.1.1 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。

1.1.2 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；

1.2 实验环境搭建

预备知识：

- 熟练使用 Python。
- 具备一定的机器学习理论知识，如逻辑回归、损失函数、优化器，训练策略等。

实验环境：

- MindSpore 2.2（ MindSpore 版本会定期更新，本指导也会定期刷新，与版本配套）；
- 本案例支持 win_x86 和 Linux 系统，CPU/GPU/Ascend 均可运行。

本实验在华为云 ModelArts 上进行，开发环境搭建可参考如下手册：ModelArts 云环境环境搭建指南。



1.3 实验总体设计



1.4 实验过程

1.4.1 数据准备

步骤 1 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.data 数据文件下载链接：<https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/MachineLearning/iris.data>

每个样本含有 4 个数值属性和一个类别属性：

4 个数值属性：sepal length；sepal width；petal length；petal width。

一个类别属性，属性值如下：

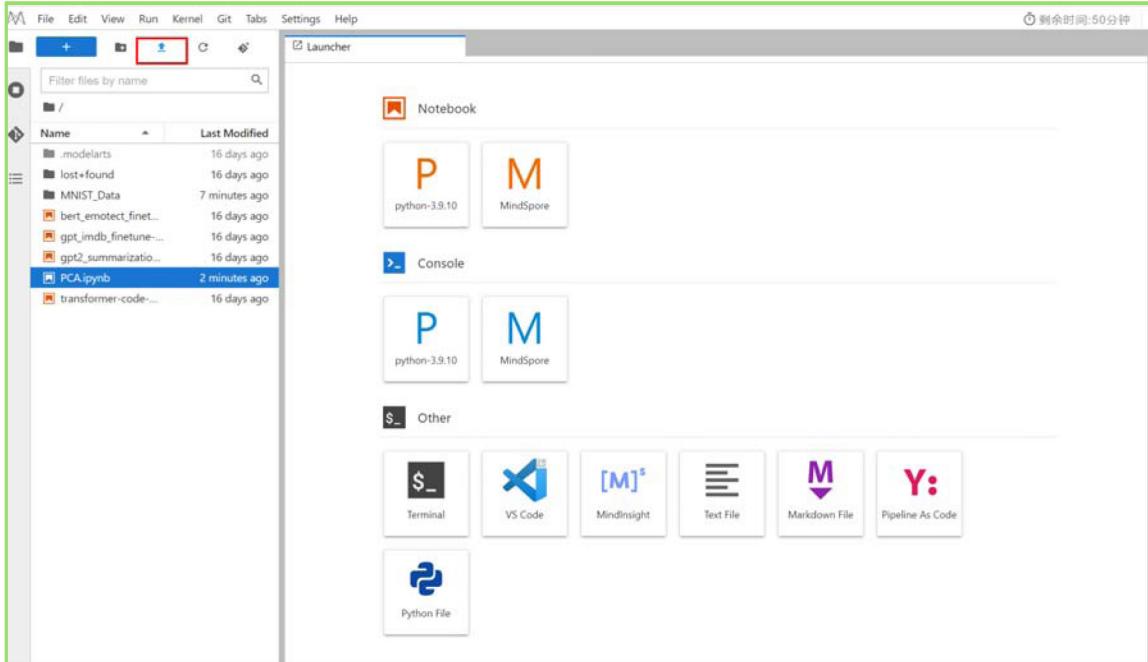
- Iris Setosa
- Iris Versicolour
- Iris Virginica

概括统计：

| Attribute | Min | Max | Mean | SD | Class Correlation |
|---------------|-----|-----|------|------|-------------------|
| sepal length: | 4.3 | 7.9 | 5.84 | 0.83 | 0.7826 |
| sepal width: | 2.0 | 4.4 | 3.05 | 0.43 | -0.4194 |
| petal length: | 1.0 | 6.9 | 3.76 | 1.76 | 0.9490 (high!) |

| | | | | | |
|--------------|-----|-----|------|------|----------------|
| petal width: | 0.1 | 2.5 | 1.20 | 0.76 | 0.9565 (high!) |
|--------------|-----|-----|------|------|----------------|

实验开始前，在云环境中，请通过下图所示的上传按钮，将下载的数据上传至 ModelArts 实验平台。



1.4.2 数据读取与处理

步骤 1 导入 MindSpore 模块和辅助模块

```
import os
import csv
import numpy as np

import mindspore as ms
from mindspore import nn, ops, ms_function, dataset, Tensor
from mindspore.train import LossMonitor
from mindspore.ops import SigmoidCrossEntropyWithLogits
ms.set_context(mode=ms.GRAPH_MODE, device_target="Ascend")
```

步骤 2 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']]
```

步骤 3 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)
```

步骤 4 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()
plt.show()
```

输出：

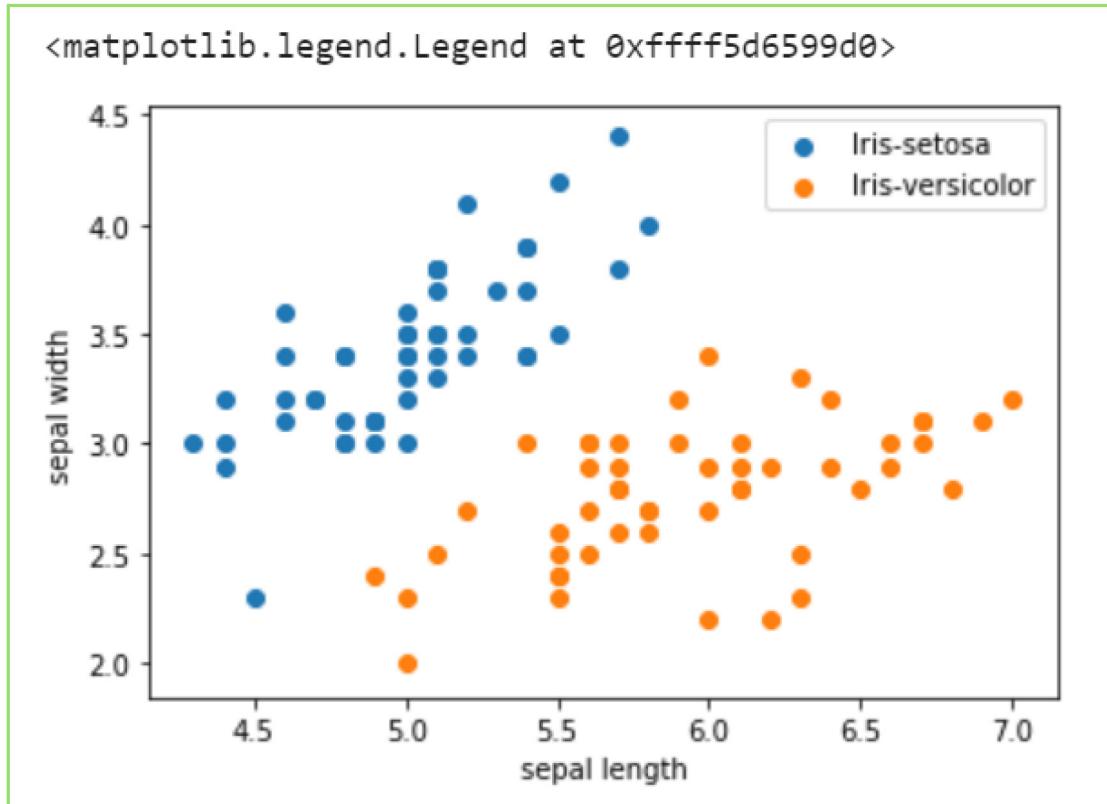


图1-1 样本数据可视化

步骤 5 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]
```

步骤 6 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'], shuffle=True)
# ds_train.set_dataset_size(80)
ds_train = ds_train.shuffle(buffer_size=80).batch(32, drop_remainder=True)
```

1.4.3 模型建立与训练

步骤 1 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')
plt.show()
```

输出：

Text(0, 0.5, 'p')

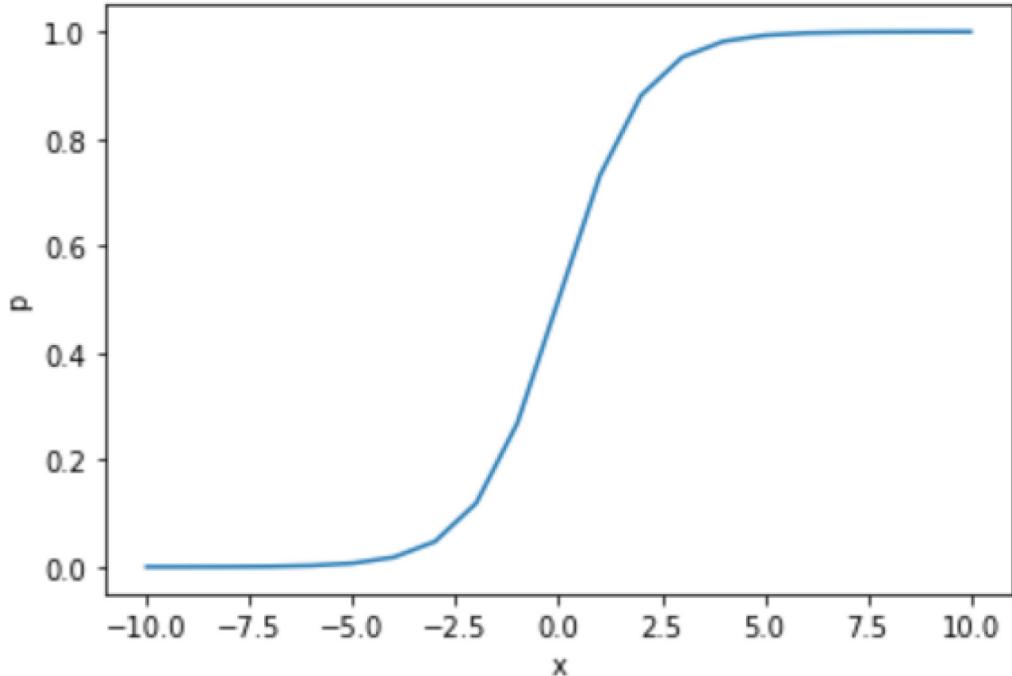


图1-2 Sigmoid 函数图示

步骤 2 建模

使用 MindSpore 提供的 nn.Dense 算子作为线性部分，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 1×4 的矩阵。算子会随机初始化权重 W 和偏置 b 。使用 SigmoidCrossEntropyWithLogits 算子作为非线性部分：

对于每个样本 N_i ，模型的计算方式如下：

$$\begin{aligned} z_i &= w x_i + b \\ p_i &= \text{sigmoid}(z_i) = \frac{1}{1 + e^{-z_i}} \\ loss &= \frac{1}{n} \sum_{i=1}^n (y_i * \ln(p_i) + (1 - y_i) * \ln(1 - p_i)) \end{aligned}$$



其中， x_i 是 1D Tensor（含 4 个元素）， z_i 是 1D Tensor（含 1 个元素）， y_i 是真实类别（2 个类别 {0, 1} 中的一个）， p_i 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 = 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)
```

步骤 3 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
```

1.4.4 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
```

1.5 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。

1.6 创新设计

请使用 Softmax 函数作为联系函数，对完整的 Iris 数据集实现多分类任务。