東南大學
SOUTHEAST UNIVERSITY

模式识别实验报告

专业: 人工智能

学号: 58122204

年级: 大二

姓名: 谢兴

签名:

时间:

目录

1 实验一 KNN Classification **3**

1.1 问题描述 . . . 3

1.2 概述 . . . 3

1.3 任务说明 . . . 3

1.4 实现步骤与流程 . . . 3

1.4.1 实验思路 . . . 3

1.4.2 数学模型 . . . 4

1.4.3 关键难点 . . . 5

1.4.4 算法描述 . . . 5

1.4.5 马氏距离梯度计算公式推导 . . . 5

1.5 实验结果与分析 . . . 8

1.5.1 数据集的部分可视化分析 . . . 8

1.5.2 实验结果的分析 . . . 8

1.6 MindSpore 学习使用心得体会 . . . 10

1.7 代码附录（数据加载可视化展示部分，具体见 knn.ipynb 文件） . . . 10

1.7.1 `knn.ipynb` . . . 10

1.7.2 `knn_mindspore.ipynb` . . . 10

2 实验二 Naïve Bayes Classification **12**

2.1 问题描述 . . . 12

2.1.1 概述 . . . 12

2.1.2 任务说明 . . . 12

2.2 实现步骤与流程 . . . 13

2.2.1 实验思路 . . . 13

2.2.2 数学模型 13
2.2.3 关键难点 13
2.2.4 算法描述 13
2.2.5 13
2.3 实验结果与分析 13
2.4 MindSpore 学习使用心得体会 13
2.5 代码附录 13

3 实验三 Neural Network Image Classification 15
3.1 问题描述 15
3.2 概述 15
3.3 任务说明 15
3.4 实现步骤与流程 16
3.4.1 实验思路 16
3.4.2 数学模型 16
3.4.3 关键难点 16
3.4.4 算法描述 16
3.5 实验结果与分析 16
3.6 MindSpore 学习使用心得体会 16
3.7 代码附录 16

4 心得体会 17

1 实验一 KNN Classification

1.1 问题描述

1.2 概述

利用 KNN 算法，对 Iris 鸢尾花数据集中的测试集进行分类。

1.3 任务说明

1. 利用欧式距离作为 KNN 算法的度量函数，对测试集进行分类。实验报告中，要求在验证集上分析近邻数 k 对 KNN 算法分类精度的影响。

2. 利用马氏距离作为 KNN 算法的度量函数，对测试集进行分类。

3. 基于 MindSpore 平台提供的官方模型库，对相同的数据集进行训练，并与自己独立实现的算法对比结果（包括但不限于准确率、算法迭代收敛次数等指标），并分析结果中出现差异的可能原因，给出使用 MindSpore 的心得和建议。

4.（加分项）使用 MindSpore 平台提供的相似任务数据集（例如，其他的分类任务数据集）测试自己独立实现的算法并与 MindSpore 平台上的官方实现算法进行对比，并进一步分析差异及其成因。

1.4 实现步骤与流程

1.4.1 实验思路

1. 导入必要的库，包括 `numpy`, `pandas`, `matplotlib`, `plotly` 和 `seaborn`；

2. 数据加载和基本信息显示；

   (a) 从 `data/train.csv` 文件中加载训练数据集

   (b) 显示数据集的前几行数据

(c) 显示数据集的描述性统计信息

(d) 显示数据集的基本信息，包括数据类型和缺失值情况

3. 数据可视化；

(a) 使用 `Seaborn` 绘制数据集的特征两两关系图，并按标签着色

(b) 使用 `Plotly` 绘制数据分布的饼图

(c) 分别绘制每个特征（萼片长度、萼片宽度、花瓣长度、花瓣宽度）的箱线图和直方图

4. 实现基于 Euiclidean 距离和基于 Mahalanobis 距离的 KNN 算法；

5. 数据处理和预测；

(a) 加载测试数据集并进行必要的类型转换和缺失值检查

(b) 使用预训练模型对测试数据进行预测，并将预测结果保存到 `CSV` 文件中

6. 最后对比欧式距离和马氏距离两种度量方式的分类效果和差异

(a) 比较多个预测结果文件 `task1_test_prediction.csv` 和 `task2_test_prediction.csv` 之间的差异，找出不同的行和列，并打印出不同值的位置

1.4.2 数学模型

KNN 算法的数学模型如下：

给定一个测试样本 x，KNN 算法通过计算 x 与训练集中所有样本之间的距离（常用欧氏距离），选择距离最近的 k 个样本，然后通过多数投票法决定 x 的类别。欧氏距离的计算公式为：

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$

马氏距离的计算公式为：

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^T \Sigma^{-1} (x_1 - x_2)}$$

其中，Σ 为协方差矩阵。

1.4.3 关键难点

1. 如何高效地计算欧氏距离。

2. 如何在较大的数据集上进行快速的邻居搜索。

3. 如何处理训练数据和测试数据的维度一致性问题。

4. 如何在分类时处理类别不均衡的问题。

1.4.4 算法描述

基于 Euiclidean 距离和 Mahalanobis 距离的 KNN 算法的伪代码分别见算法 1 和算法 2。

1.4.5 马氏距离梯度计算公式推导

假设我们有训练数据集 $\{(x_i, y_i)\}_{i=1}^n$，其中 $x_i \in \mathbb{R}^d$ 为样本特征，y_i 为样本类别。为了优化马氏距离下的 KNN 算法，我们需要学习一个矩阵 $A \in \mathbb{R}^{e \times d}$，使得同类样本之间的距离最小化。马氏距离的计算公式为：

$$d_M(x_i, x_j) = \sqrt{(x_i - x_j)^\top A^\top A (x_i - x_j)} \tag{1}$$

为了优化矩阵 A，我们使用如下的目标函数：

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} p_{ij} \cdot d_M^2(x_i, x_j) \tag{2}$$

Algorithm 1 K-Nearest Neighbors Based on Euclidean Distance

初始化 KNN 分类器，邻居数为 k
procedure 拟合 (X_train, y_train)
    存储训练数据和标签
end procedure
procedure 预测 (X)
    for 每个测试数据 x **do**
        计算 x 与所有训练样本之间的欧氏距离
        对距离进行排序，选择最近的 k 个邻居
        对这 k 个邻居的标签进行多数投票
        将多数投票结果赋予 x
    end for
    return 预测的标签
end procedure

其中，$\mathcal{N}(i)$ 表示与 x_i 同类的样本索引集合，p_{ij} 为权重，定义为：

$$p_{ij} = \frac{\exp(-d_M^2(x_i, x_j))}{\sum_{k \in \mathcal{N}(i)} \exp(-d_M^2(x_i, x_k))} \tag{3}$$

首先，我们对 $d_M^2(x_i, x_j)$ 进行展开：

$$d_M^2(x_i, x_j) = (x_i - x_j)^\top A^\top A (x_i - x_j) \tag{4}$$

为了计算梯度 $\nabla_A \mathcal{L}$，我们需要对 $\mathcal{L}$ 关于 A 求导：

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} p_{ij} (x_i - x_j)^\top A^\top A (x_i - x_j) \tag{5}$$

对 A 求导时，需要使用链式法则：

$$\frac{\partial \mathcal{L}}{\partial A} = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \left(\frac{\partial p_{ij}}{\partial A} (x_i - x_j)^\top A^\top A (x_i - x_j) + p_{ij} \frac{\partial ((x_i - x_j)^\top A^\top A (x_i - x_j))}{\partial A} \right) \tag{6}$$

Algorithm 2 K-Nearest Neighbors Based on Mahalanobis Distance

初始化 KNN 分类器，邻居数为 k，矩阵 A 的维度为 e，学习率为 η，最大迭代次数为 max_iter
procedure 拟合 (X_train, y_train)
    存储训练数据和标签
    初始化矩阵 A 为随机值
    for 迭代次数 $iteration = 1, 2, \ldots, max\_iter$ **do**
        初始化梯度矩阵 ∇A 为零
        for 每个训练样本 x_i **do**
            获取与 x_i 同类的样本索引 $same\_class\_indices$
            for 每个同类样本 x_j **do**
                if $i == j$ **then**
                    跳过
                end if
                计算 p_{ij} 值
                计算样本差异 $diff = x_i - x_j$
                更新梯度 $\nabla A += 2 \cdot p_{ij} \cdot (A \cdot diff) \cdot diff^T$
            end for
        end for
        按学习率更新矩阵 A: $A = A - \eta \cdot \nabla A / n$
    end for
end procedure
procedure 预测 (X)
    for 每个测试数据 x **do**
        计算 x 与所有训练样本之间的马氏距离
        对距离进行排序，选择最近的 k 个邻居
        对这 k 个邻居的标签进行多数投票
        将多数投票结果赋予 x
    end for
    return 预测的标签
end procedure

首先计算 p_{ij} 对 A 的导数。由于 p_{ij} 包含在指数函数内，我们得到：

$$\frac{\partial p_{ij}}{\partial A} = p_{ij} \left(- \sum_{k \in \mathcal{N}(i)} p_{ik} \cdot 2(x_i - x_k)^\top A^\top \cdot (x_i - x_k) + 2(x_i - x_j)^\top A^\top \cdot (x_i - x_j) \right) \tag{7}$$

然后计算 $(x_i - x_j)^\top A^\top A (x_i - x_j)$ 对 A 的导数：

$$\frac{\partial ((x_i - x_j)^\top A^\top A (x_i - x_j))}{\partial A} = 2A(x_i - x_j)(x_i - x_j)^\top \tag{8}$$

将以上结果代入梯度公式中，我们得到：

$$\begin{aligned} \nabla_A \mathcal{L} = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \Bigg[p_{ij} \Bigg(& - \sum_{k \in \mathcal{N}(i)} p_{ik} \cdot 2(x_i - x_k)^\top A^\top \cdot (x_i - x_k) \\ & + 2(x_i - x_j)^\top A^\top \cdot (x_i - x_j) \Big) + 2 p_{ij} A (x_i - x_j)(x_i - x_j)^\top \Bigg] \end{aligned} \tag{9}$$

整理后得到最终的梯度公式：

$$\nabla_A \mathcal{L} = 2 \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} p_{ij} \left[A(x_i - x_j)(x_i - x_j)^\top - \sum_{k \in \mathcal{N}(i)} p_{ik} A (x_i - x_k)(x_i - x_k)^\top \right] \tag{10}$$

1.5 实验结果与分析

1.5.1 数据集的部分可视化分析

1. `train.csv` 文件中训练数据的 pairplot 图如图 1所示。

2. 训练数据的分布情况如图 2所示，可以看出这三类鸢尾花的数据分布比例是不完全一致的，但三类的数据量大致相同。

3.

1.5.2 实验结果的分析

1. 对于基于欧氏距离的 KNN 算法，当 $k = 3$ 时，测试集的准确率为 93.33%；

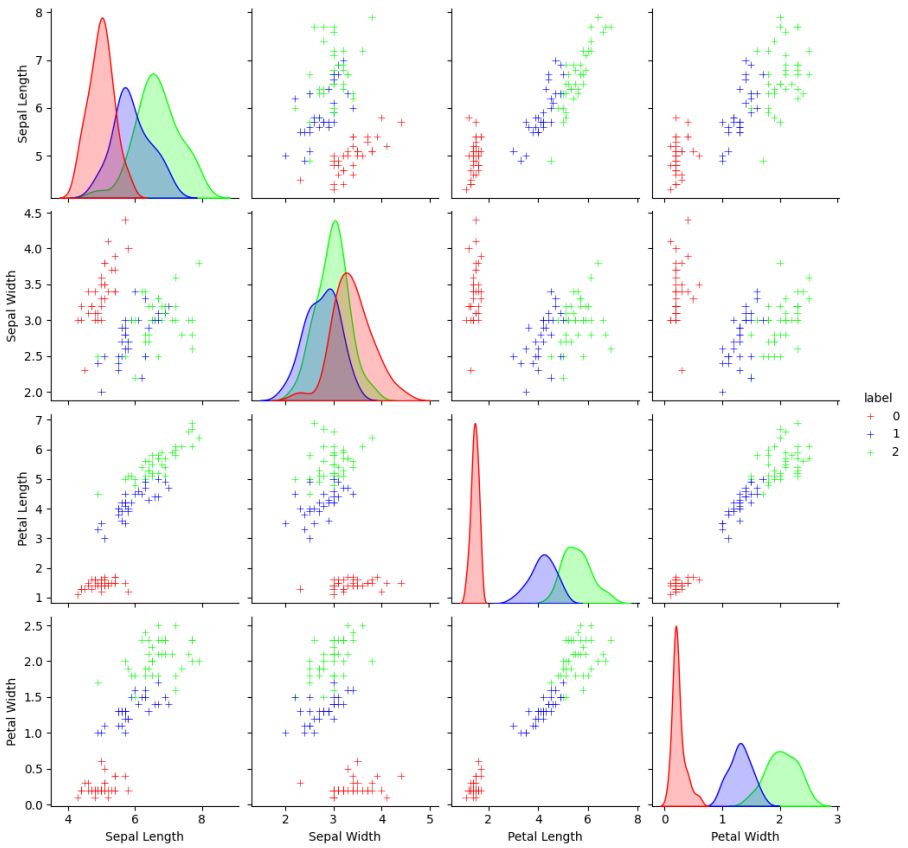

图 1: train.csv 训练数据的 pairplot 图

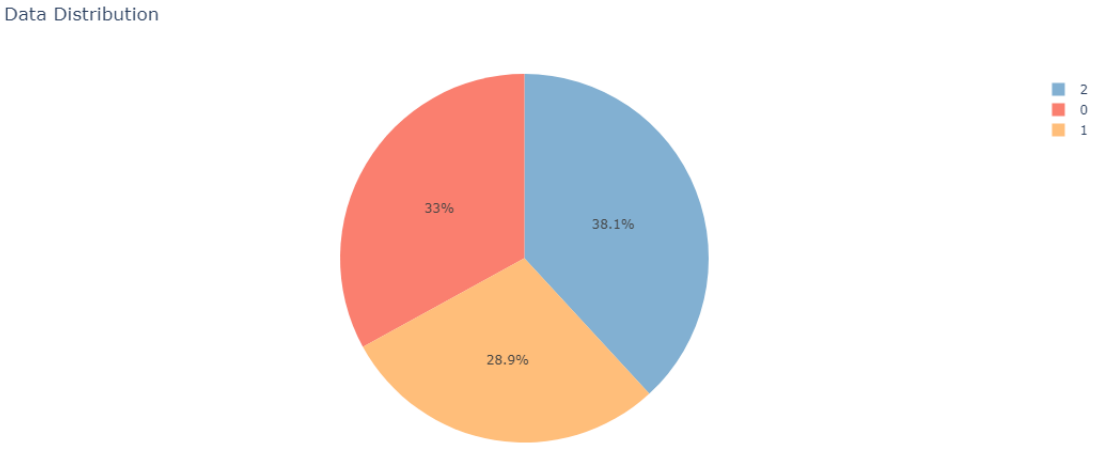

图 2: train.csv 训练数据分布比例

2. 对于基于马氏距离的 KNN 算法，当 $k = 3$ 时，测试集的准确率为 93.33%；

3. 将 k 从 1 到 50 遍历，分析出基于欧氏距离的 KNN 算法对 Iris 数据集进行分类的准确率随 k 的变化情况，如图 3所示。

显然，基于欧式距离的最佳 k 值为 5 或 27 或 29，此时的准确率最高，为 100%，说明 $k = 5$，$k = 27$，$k = 29$ 时能完全正确的将 Iris 数据集中三类鸢尾花进行分类。

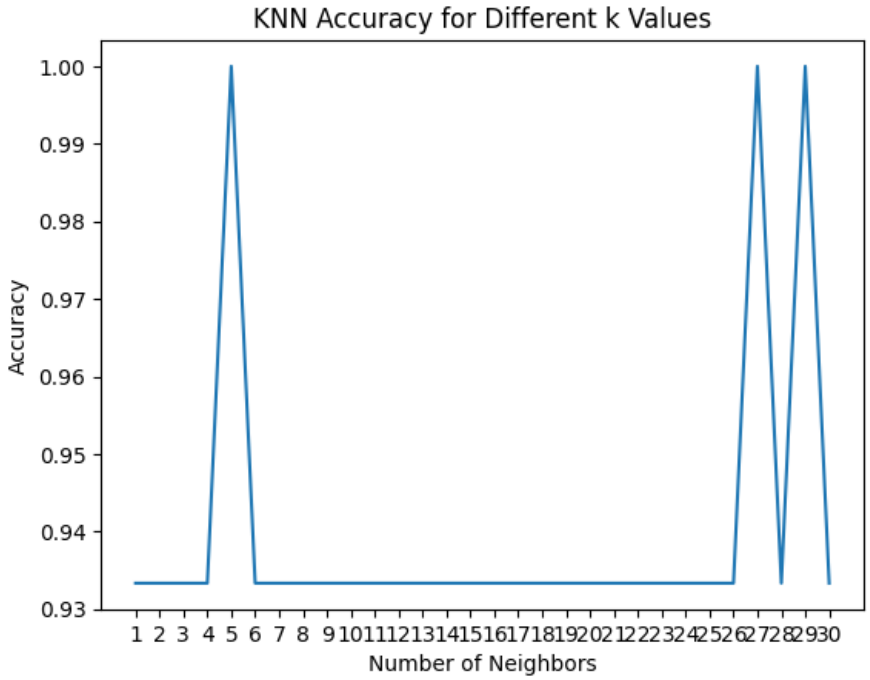

图 3: 分类准确率随 k 的变化情况

1.6 MindSpore 学习使用心得体会

1.7 代码附录（数据加载可视化展示部分，具体见 **knn.ipynb** 文件）

1.7.1 knn.ipynb

```
# 实验一代码
```

1.7.2 knn_mindspore.ipynb

```
1 # 实验一代码
```

2 实验二 Naïve Bayes Classification

2.1 问题描述

2.1.1 概述

利用朴素贝叶斯算法，对 MNIST 数据集中的测试集进行分类。

2.1.2 任务说明

1. 在课程学习中同学们已经学习了贝叶斯分类理论并掌握了其基本原理，即利用贝叶斯公式

$$p(\omega_j|x) = \frac{p(x|\omega_j)p(\omega_j)}{p(x)}$$

对 $p(\omega_j|x)$ 作出预测。由于 $p(x)$ 为一固定值，所以一般不在计算过程中求得 $p(x)$ 的具体值。在实际运用中，为了方便计算，通常假设数据特征之间相互独立，即

$$p(x|\omega_j) = p(x_1|\omega_j) \cdot p(x_2|\omega_j) \cdots p(x_d|\omega_j), \quad x \in \mathbb{R}^d,$$

这便是著名的朴素贝叶斯算法。

2. MNIST 数据集本身以二进制形式保存，所以首先需要选择合适的编程语言编写读写二进制数据的程序完成对图片、标记信息的初步提取工作。读取了图片信息后，发现每个像素点的值在 [0,1] 区间内，这是图像压缩后的结果，所以可以先将像素值乘以 255 再取整，得到每一个点的灰度值。将图像二值化，得到可以用于分类的 28×28 个特征向量以及对应的标签数据，之后便可以交由贝叶斯分类器进行学习。

3. 基于 MindSpore 平台提供的官方模型库，对相同的数据集进行训练，并与自己独立实现的算法对比结果（包括但不限于准确率、算法迭代收敛次数等指标），并分析结果中出现差异的可能原因，给出使用 MindSpore 的心得和建议。

4.（加分项）使用 MindSpore 平台提供的相似任务数据集（例如，其他的分类任务数据集）测试自己独立实现的算法并与 MindSpore 平台上的官方实现算法进行对比，并进一步分析差异及其成因。

2.2 实现步骤与流程

2.2.1 实验思路

2.2.2 数学模型

2.2.3 关键难点

2.2.4 算法描述

朴素贝叶斯算法实现的伪代码如算法 3 所示。

2.2.5

2.3 实验结果与分析

2.4 MindSpore 学习使用心得体会

2.5 代码附录

```
# 实验二代码
```

Algorithm 3 Optimized Multinomial Naive Bayes

Input: 平滑参数 α
Initialize:
    类别数 $n\_classes$
    特征数 $n\_features$
    类别计数 $class\_count$
    特征计数 $feature\_count$
    类别对数先验 $class\_log\_prior$
    特征对数概率 $feature\_log\_prob$
procedure 拟合 (X, y)
    获取唯一类别 $classes = \text{np.unique}(y)$
    初始化类别计数 $class\_count$ 和特征计数 $feature\_count$
    for 每个类别 $c \in classes$ **do**
        获取属于类别 c 的样本 X_c
        更新类别计数 $class\_count[c]$
        更新特征计数 $feature\_count[c, :]$
    end for
    计算类别对数先验 $class\_log\_prior$
    计算特征对数概率 $feature\_log\_prob$
end procedure
procedure 预测 (X)
    计算对数似然 $log\_likelihood = X \times feature\_log\_prob^T$
    计算对数后验概率 $log\_posterior = log\_likelihood + class\_log\_prior$
    返回类别 $classes[\text{np.argmax}(log\_posterior, axis = 1)]$
end procedure

3 实验三 Neural Network Image Classification

3.1 问题描述

3.2 概述

利用神经网络算法，对 CIFAR 数据集中的测试集进行分类。

3.3 任务说明

1. 基于神经网络模型及 BP 算法，根据训练集中的数据对你设计的神经网络模型进行训练，随后对给定的打乱的测试集中的数据进行分类。

2. 基于 MindSpore 平台提供的官方模型库，对相同的数据集进行训练，并与自己独立实现的算法对比结果（包括但不限于准确率、算法迭代收敛次数等指标），并分析结果中出现差异的可能原因。

3.（加分项）使用 MindSpore 平台提供的相似任务数据集（例如，其他的分类任务数据集）测试自己独立实现的算法并与 MindSpore 平台上的官方实现算法进行对比，并进一步分析差异及其成因。

3.4 实现步骤与流程

3.4.1 实验思路

3.4.2 数学模型

3.4.3 关键难点

3.4.4 算法描述

3.5 实验结果与分析

3.6 MindSpore 学习使用心得体会

3.7 代码附录

```
# 实验三代码
```

4　心得体会