Introduction to Data Science and Engineering Week 7

Author: Chenxu Han@ArcueidType

Student ID: 10225101440

Week 7 practices:

Chapter 11:

Practice 5:

Logistic回归是一种广泛用于二元分类问题的统计分析方法。虽然它的名称中含有“回归”，但实际上它是用于分类任务的。在这里，我们将深入探讨如何通过梯度下降法手动解决Logistic回归问题。我们首先从Logistic回归模型的基本原理谈起，然后讨论梯度下降法如何应用于优化问题。

Logistic 回归模型

1. **Sigmoid 函数**: Logistic回归使用Sigmoid函数作为激活函数，将线性回归结果映射到[0,1]区间，用于预测二元目标变量的概率。Sigmoid函数定义如下：

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

2. **模型假设**: Logistic回归模型假设目标变量y的对数几率是输入X的线性组合。这里，z代表线性组合：$z = w^T X + b$，其中w是权重向量，b是偏差项。

3. **损失函数(Loss Function)**: 我们使用交叉熵损失函数，关于该函数，我们其实可以验证出，**它不仅可以用来较精确的估计预测值和真实值的损失。** 由于它本身是一个**凸函数**，这样在梯度下降时，可以避免**局部最小值**的产生，以更好的获得最小的总损失。交叉熵损失函数如下：

$$L(w, b) = -y \log(\sigma(z)) - (1 - y) \log(1 - \sigma(z))$$

对于整个训练集，代价函数是所有训练样本损失的平均，即：

$$J(w, b) = \frac{1}{N} \sum_{i=1}^{N} L(\sigma(z), y)$$

梯度下降求解

梯度下降是一种迭代优化算法，用于最小化代价函数。基本思想是计算代价函数的梯度，并沿着减少最快的方向调整参数。

1. **计算梯度**: 首先，我们需要计算代价函数相对于模型参数的梯度。对于权重w和偏差b，梯度计算如下：

$$\frac{\partial L(\sigma(z), y)}{\partial w} = \frac{\partial L}{\sigma(z)} \frac{d\sigma(z)}{dz} \frac{\partial z(w, b)}{\partial w} = x(a - y)^T$$

故：

$$\frac{\partial L(\sigma(z), y)}{\partial b} = \frac{\partial L}{\sigma(z)} \frac{d\sigma(z)}{dz} \frac{\partial z(w, b)}{\partial b} = (a - y)^T$$

$$\frac{\partial J}{\partial w} = \frac{1}{N} \sum_{i=1}^{N} (\sigma(z^{(i)}) - y^{(i)}) x^{(i)}$$

$$\frac{\partial J}{\partial b} = \frac{1}{N} \sum_{i=1}^{N} (\sigma(z^{(i)}) - y^{(i)})$$

其中N是训练样本的数量。

2. **更新规则**: 然后，我们用以下规则更新参数：

$$w = w - \alpha \frac{\partial J}{\partial w}$$

$$b = b - \alpha \frac{\partial J}{\partial b}$$

其中，α是学习率，控制我们更新参数的步长。

3. **迭代优化**: 通过多次迭代过程，我们不断用计算出的梯度更新参数，直到代价函数收敛为止。

算法步骤总结

1. 初始化模型参数（通常初始化为0或小的随机值）。
2. 计算模型的预测输出。
3. 计算代价函数（损失）。
4. 计算代价函数的梯度。
5. 更新模型参数。
6. 重复步骤2-5，直到满足停止准则（例如，达到预定的迭代次数，或代价函数的改变非常小）。

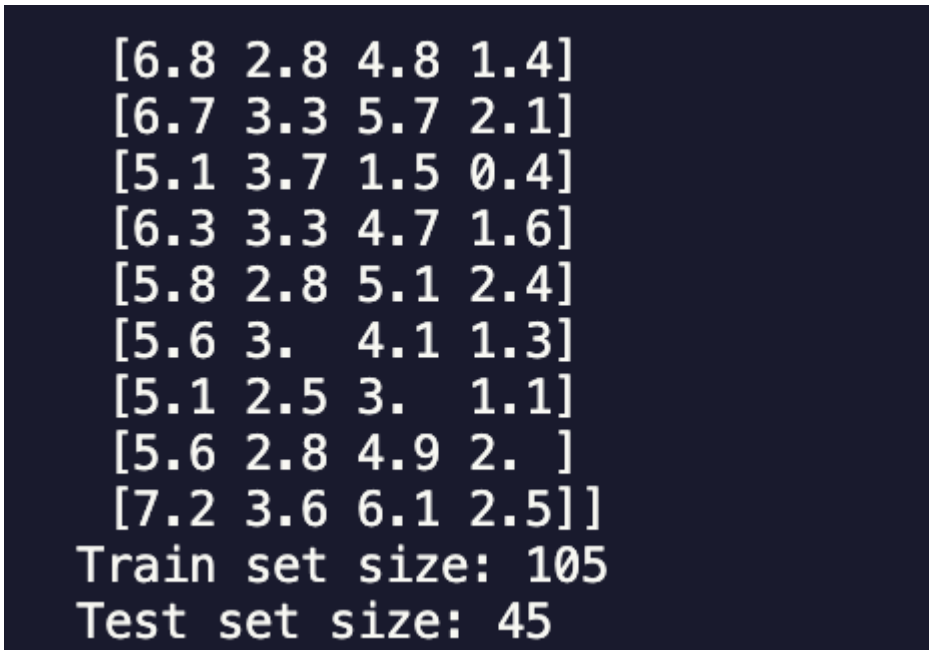
通过这个过程，模型最终学习到数据的权重参数，用于对新的输入数据进行预测。

> Codes in directories ./chap11 or ./chap12

Practice 6:

> Result is too long to be shown all!

Result:



Practice 7:

> `random_state` is set to `42`

Result:

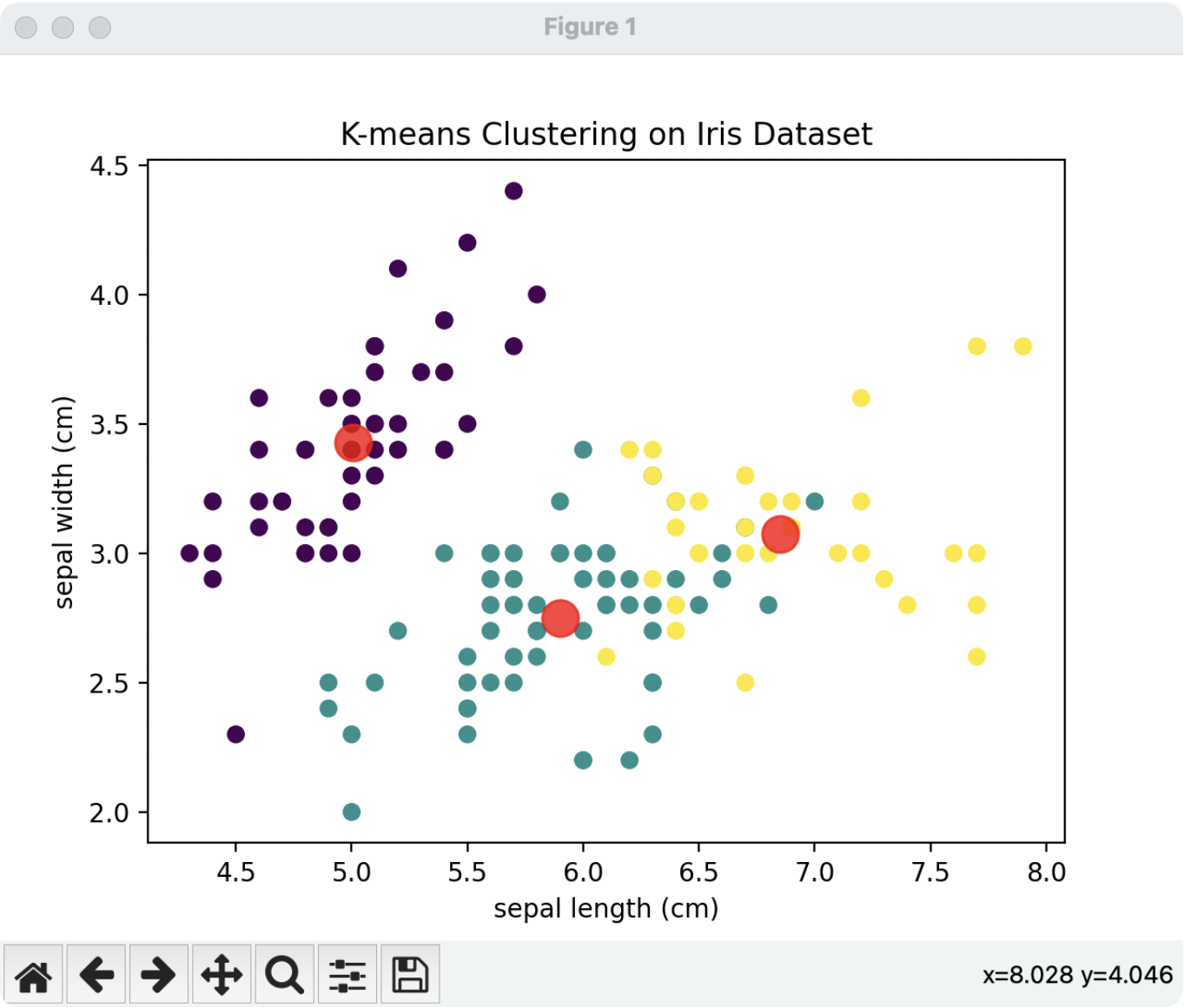
Practice 8:

Result:

```
Centres of each dimension relatively: [5.84333333 3.05733333 3.758      1.19933333]

Average distance to centre point: 4.5424706666666665
```

Practice 9:

Result:



Chapter 12:

Practice 4:

Result:

learning:

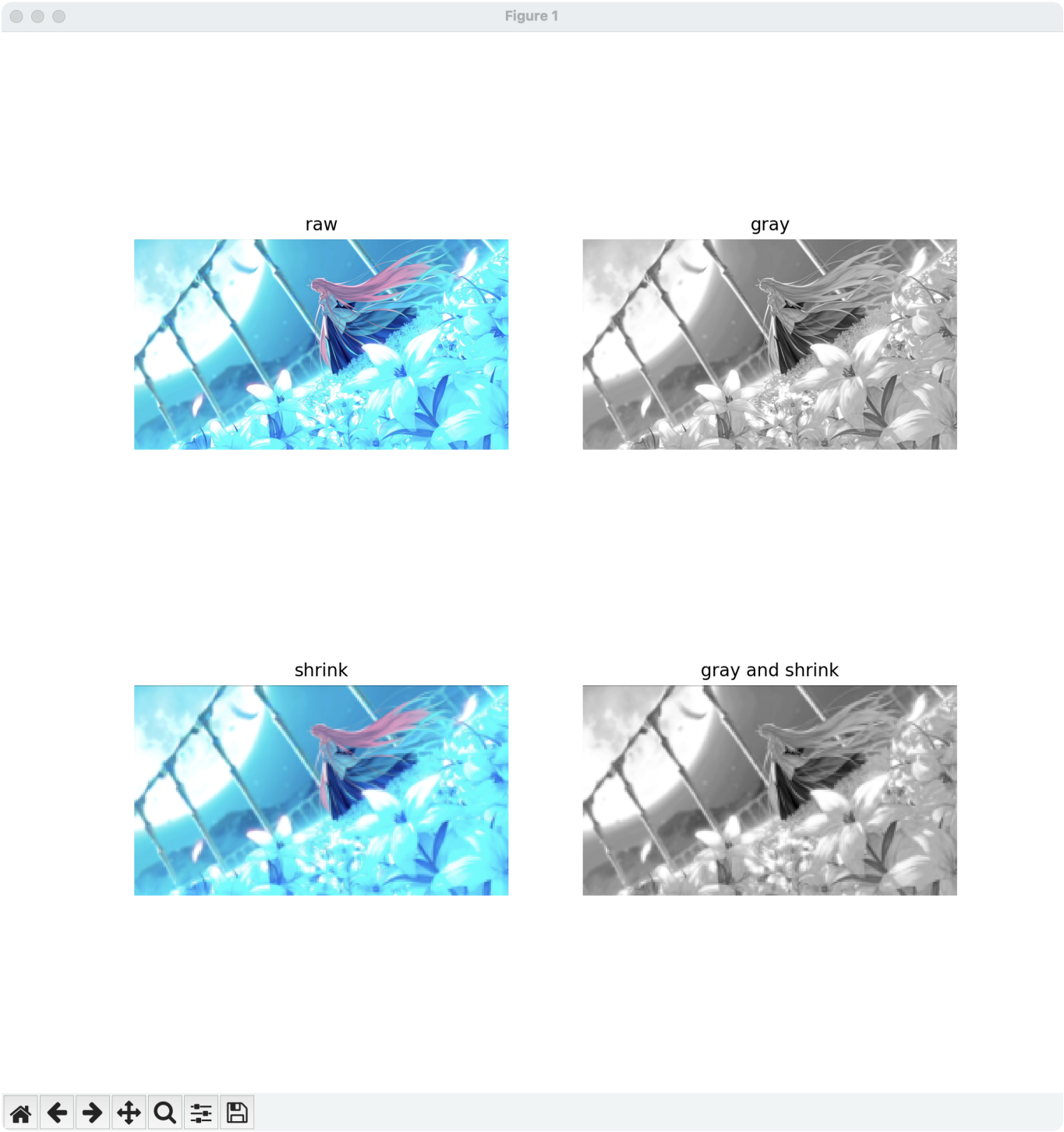
```
Epoch 999
-------------------------------
Test Error:
 Accuracy: 97.8%, Avg loss: 0.183603

Epoch 1000
-------------------------------
Test Error:
 Accuracy: 97.8%, Avg loss: 0.120632

Done!
```

predicting:

```
Using mps device
Predicted: "versicolor", Actual: "versicolor"
```

Practice 5:

Result:



Practice 7:

Result:

```
LeNetFive(
  (conv1): Sequential(
    (0): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (conv2): Sequential(
    (0): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc1): Sequential(
    (0): Linear(in_features=400, out_features=120, bias=True)
    (1): ReLU()
  )
  (fc2): Sequential(
    (0): Linear(in_features=120, out_features=84, bias=True)
    (1): ReLU()
  )
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

Practice 8:

Result:

learning:

```
Epoch 10
-------------------------------
loss: 0.032439 [   64/60000]
loss: 0.069483 [ 6464/60000]
loss: 0.014526 [12864/60000]
loss: 0.012085 [19264/60000]
loss: 0.185118 [25664/60000]
loss: 0.028057 [32064/60000]
loss: 0.023757 [38464/60000]
loss: 0.027807 [44864/60000]
loss: 0.078107 [51264/60000]
loss: 0.024016 [57664/60000]
Test Error:
 Accuracy: 98.1%, Avg loss: 0.057701

Done!
```

predicting:

```
Actual: 6 Prediction: 6
```