理解损失函数和优化过程在一个简单模型上的应用

问题：

一个房产中介，有历史上为客户卖房的价格数据，即不同面积的房子分别卖了多少钱。他们希望能够根据历史数据做一个模型，这样当新客户来咨询房价的时候，报了房屋面积后，就能给用户提供一个比较靠谱的可能的售价。

问题数学转化

面积用 x 表示，售价用 y 表示。

售价和面积的关系使用函数：y = m.x + b 表示。

并且有历史数据

房屋面积（平方米） x_i	房屋价格（万元） y_i
50	300
70	500
80	600
100	800

注意：实际房屋的很多其他属性也会和房价有关，比如楼层，位置等，为了简化问题，只考虑面积。

所以问题转化为，如何根据历史数据，确定模型参数 m/b 的值（即得到的参数 m/b 的值能够让模型和历史数据最契合）

方法：

Step1：定义一个损失函数

求解最合适的模型参数 w，b 的方法就是**定义一个损失函数 Loss function，并且通过求解 Loss function 的最小值，来确定 Loss 最小时候的 w/b**。（*Loss 最小就是模型通过 x 预测出的 y 能够跟实际的 yi 差别最小，即模型的曲线最能描述实际的数据，这时得到的参数就是能让模型曲线最贴合训练数据的参数值*）。

MSE（均方误差，也是一种最小二乘）是一种比较常用的损失函数。

$$L(m,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (m \cdot x_i + b))^2$$

y_i 是第 i 个数据点的实际房价，x_i 是第 i 个数据点的房屋面积，n 是数据点的总数。$m.x_i + b$ 是通过模型对第 i 个数据点的房价预测值。*当 Loss 最小就是实际房价与预测房价差别最小的时候。*

将把全部历史数据(x_i y_i)带入进去，Loss 函数就表示为一个以 m/b 为自变量的函数。

$$L(m,b) = \frac{1}{4}[(300 - (m \cdot 50 + b))^2 + (500 - (m \cdot 70 + b))^2 + (600 - (m \cdot 80 + b))^2 + (800 - (m \cdot 100 + b))^2]$$

Step2:优化求解，即求 Loss 最小值

给 m/b 设置一个随机值，并求解 L 的梯度（对于 m 和 b 的偏导数）

设初始值：m=0，b=0 （设置其他值也可以）

求初始梯度：

$$\frac{\partial L}{\partial m}=\frac{1}{4}\sum_{i=1}^{4}-2x_i(y_i-(0\cdot x_i+0))=\frac{1}{4}\sum_{i=1}^{4}-2x_iy_i=-\frac{1}{2}(50\cdot 300+70\cdot 500+80\cdot 600+100\cdot 800)$$

$$\frac{\partial L}{\partial m}=-\frac{1}{2}(15000+35000+48000+80000)=-89000$$

$$\frac{\partial L}{\partial b}=\frac{1}{4}\sum_{i=1}^{4}-2(y_i-(0\cdot x_i+0))=-\frac{1}{2}(300+500+600+800)=-1100$$

根据梯度更新参数的值：

$$m=m-\alpha\cdot\frac{\partial L}{\partial m}$$
$$b=b-\alpha\cdot\frac{\partial L}{\partial b}$$

其中 α 叫学习率，是梯度下降中用于更新参数的步长。选择一个合适的学习率的值，太小会导致收敛速度慢（即需要很多步骤才能到达 $\frac{\partial L}{\partial m}$ =0，太大可能会导致过拟合或不收敛，即把=0 的点跳过去了，导致算不出来）

在我们的例子中：

假设学习率 $\alpha=0.01$：

$$m=0-0.01\cdot(-89000)=890$$
$$b=0-0.01\cdot(-1100)=11$$

使用 m，b 的新值重复 Step2 求梯度，并更新参数值的过程。直到计算的梯度接近零或达到设定的迭代次数限制。

当停止迭代时的参数 m，b 的值就是模型最优的时候的参数值。(m = 7.51, b = -2.08)

于是我们就得到了一个预测房价的模型：y=7.51x – 2.08

Python 代码

```
import numpy as np
import matplotlib.pyplot as plt

# 训练数据
X = np.array([50, 70, 80, 100])
y = np.array([300, 500, 600, 800])

# 模型参数初始化
m = 0
b = 0

# 超参数
learning_rate = 0.0001
epochs = 1000

# 存储损失值用于绘图
losses = []

# 训练模型
for epoch in range(epochs):
    # 前向传播
    y_pred = m * X + b

    # 计算损失 (MSE)
    loss = np.mean((y_pred - y) ** 2)
    losses.append(loss)

    # 计算梯度
    dm = -2 * np.mean(X * (y - y_pred))
    db = -2 * np.mean(y - y_pred)

    # 更新参数
    m = m - learning_rate * dm
    b = b - learning_rate * db

# 打印最终结果
print(f"Final parameters: m = {m:.2f}, b = {b:.2f}")
print(f"Final loss: {losses[-1]:.2f}")

# 绘制结果
plt.figure(figsize=(12, 5))

# 绘制损失曲线
```

```
plt.subplot(1, 2, 1)
plt.plot(losses)
plt.title('Loss over epochs')
plt.xlabel('Epoch')
plt.ylabel('MSE Loss')

# 绘制拟合直线
plt.subplot(1, 2, 2)
plt.scatter(X, y, color='blue', label='Data points')
plt.plot(X, m * X + b, color='red', label='Fitted line')
plt.title('Linear Regression Fit')
plt.xlabel('Area')
plt.ylabel('Price')
plt.legend()

plt.tight_layout()
plt.show()
```