南開大學
Nankai University

南　　开　　大　　学

计　算　机　学　院

机器学习实验报告

Lab2 回归分析

马溟怡

学号：2311061

专业：计算机科学与技术

2025 年 10 月 28 日

目录

一、 实验内容

(一) 任务 1：线性回归–最小二乘法

1. 代码

实现正规方程，返回 w 和 b

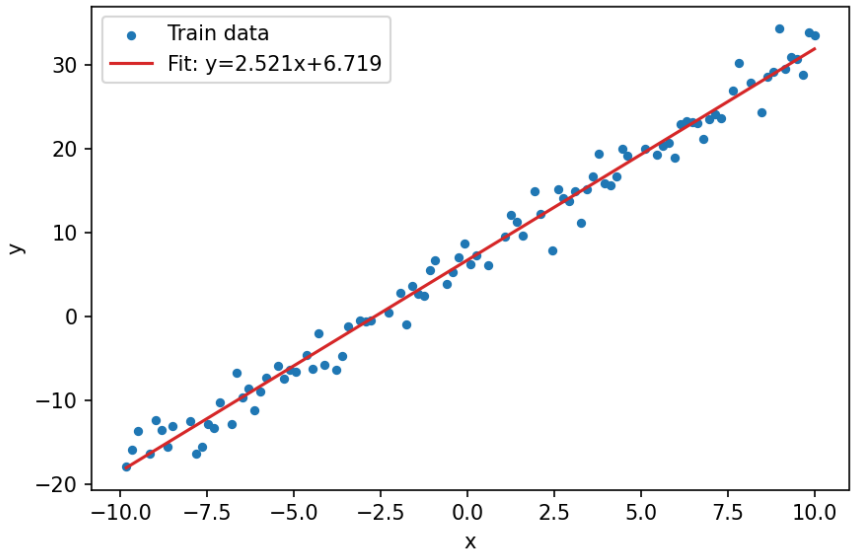
```python
def normal_equation_fit(x_train: np.ndarray, y_train: np.ndarray):
#     # YOUR CODE HERE
    # 1. 构造设计矩阵 Xb = [x, 1]
    # N x 1 的 x 向量
    N = x_train.shape[0]
    # 构造 N x 2 的矩阵: [x, 1]
    # 另一种常见的构造是 [1, x], 但模板提示 [x, 1]
    Xb = np.stack([x_train, np.ones(N)], axis=1)  # Xb 形状: N x 2

    # 确保 y_train 是 N x 1 矩阵以进行矩阵乘法, 尽管对于一维 numpy 向量也常自动工作
    y_mat = y_train.reshape(-1, 1)

    # 2. 计算 Xb^T Xb
    XTX = Xb.T @ Xb  # 2 x N @ N x 2 -> 2 x 2

    # 3. 计算 (Xb^T Xb)^-1
    # 使用 numpy.linalg.inv 求矩阵的逆
    XTX_inv = np.linalg.inv(XTX)

    # 4. 计算 Xb^T y
    XTy = Xb.T @ y_mat  # 2 x N @ N x 1 -> 2 x 1

    # 5. 计算最终参数 theta: theta = (Xb^T Xb)^-1 Xb^T y
    theta = XTX_inv @ XTy  # 2 x 2 @ 2 x 1 -> 2 x 1

    # 6. 提取 w 和 b
    # 由于 Xb = [x, 1], 所以 theta = [w, b]^T
    w = theta[0, 0]
    b = theta[1, 0]
    return w, b
```

2. 控制台输出结果

```
Train MSE: 3.3756    # 模型在用于训练的数据集上的平均误差。
Test MSE: 3.4544     # 模型在未参与训练的测试集上的平均误差。
5 new predictions:
  x=-12.00 -> y_pred=-23.531
  x=-3.00 -> y_pred=-0.844
  x=0.00 -> y_pred=6.719
  x=4.50 -> y_pred=18.063
```

```
  x=11.00 -> y_pred=34.449
# 使用训练得到的回归方程对构造的 5 个新测试样本点进行的预测结果，可以看到随着 x 值（自变量）的稳定增加，预测值 y（因变量）也稳定增加，符合线性模型的特性。
```

3. 训练数据的散点图和模型拟合的回归直线的可视化结果



4. 思考题：正规方程的求解核心与失败条件

思考题：正规方程的求解核心是哪一步？这一步在什么情况下可能会失败?

1. **求解核心步骤：**正规方程 $\theta = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$ 的核心步骤是计算设计矩阵的转置与自身乘积矩阵 $\mathbf{X}^\mathsf{T}\mathbf{X}$ 的**逆矩阵** $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$。
2. **失败情况：**当 $\mathbf{X}^\mathsf{T}\mathbf{X}$ 矩阵**不可逆**时，求解步骤失败。这通常发生在以下两种情况:
   - **多重共线性（特征冗余）：**数据集中存在两个或多个特征之间具有高度线性相关性，导致矩阵的秩不足。
   - **特征数大于样本数：**样本数量 N 小于特征数量 D 时 $(N < D)$，即欠定系统，此时 $\mathbf{X}^\mathsf{T}\mathbf{X}$ 必然是奇异的。

(二) 任务 2：线性回归 - 梯度下降法

1. 思路

1. **参数初始化：**
   - 初始化模型参数（权重 $\mathbf{w}$ 和偏置 b），通常采用接近零的随机值。
2. **迭代训练（批量梯度下降 BGD）：**
   - 在设定的迭代次数内循环，每轮迭代执行以下步骤:
   - **前向计算：**计算当前参数下的预测值 $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b$。

- **计算损失：**计算训练集上的均方误差，并记录到历史列表中。
- **计算梯度：**利用**全部训练样本**计算损失函数对 $\mathbf{w}$ 和 b 的梯度 ∇J。
- **参数更新：**沿着梯度的反方向，根据学习率 η 更新参数：$\theta = \theta - \eta \nabla J$。

2. 代码

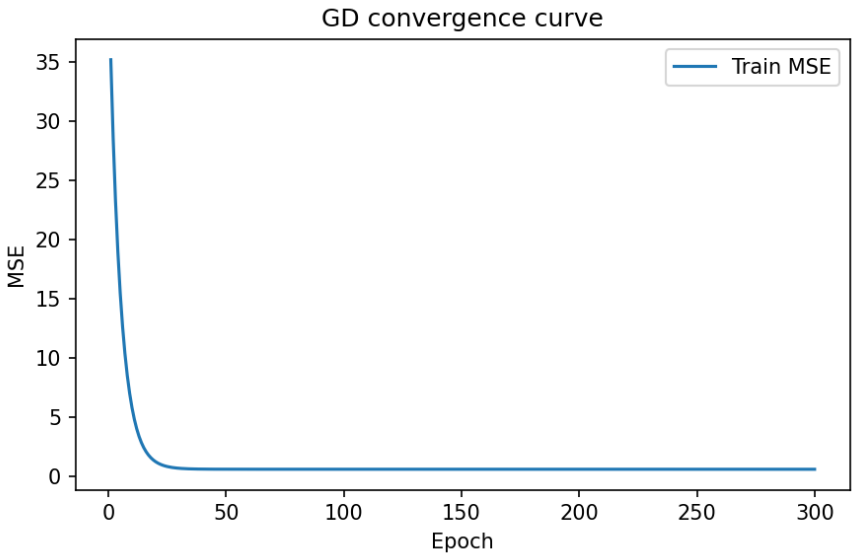
批量梯度下降（BGD）算法

```python
  # 目标：实现 BGD 或 SGD 训练线性回归，返回 (w, b, hist)
# 建议：BGD 每轮使用全部样本；SGD 可每次采样一个或小批量样本
def gd_train(X: np.ndarray, y: np.ndarray, lr=0.01, epochs=200):
    # YOUR CODE HERE
    # 维度：D（特征数）
    D = X.shape[1]
    # 样本数：N
    N = X.shape[0]

    # 1. 初始化参数（D x 1 权重向量，标量截距）
    # 使用随机初始化
    w = np.random.randn(D) * 0.01  # 确保 w 是 1D 数组 (D,)
    b = np.random.randn() * 0.01

    history_mse_list = []

    # 2. 迭代训练
    for epoch in range(epochs):
        # 3. 计算预测值 (N,)
        y_pred = X @ w + b

        # 4. 计算误差 (N,)
        error = y_pred - y

        # 5. 计算梯度（批量梯度下降 BGD）
        # 权重梯度 (D,)：(2/N) * X^T * error
        w_grad = (2 / N) * (X.T @ error)
        # 截距梯度（标量）：(2/N) * sum(error)
        b_grad = (2 / N) * np.sum(error)

        # 6. 更新参数
        w = w - lr * w_grad
        b = b - lr * b_grad

        # 7. 记录 MSE
        mse = np.mean(error ** 2)
        history_mse_list.append(mse)
    return w, b, history_mse_list
```

3. 控制台输出结果

```
[Template] Implement gd_train(X_train, y_train) returning w, b, history.
Train MSE: 0.5636
Test MSE:  0.5744
```

4. 训练数据的散点图和模型拟合的回归直线的可视化结果



(三) 任务 3：超参数调优 - 学习率分析

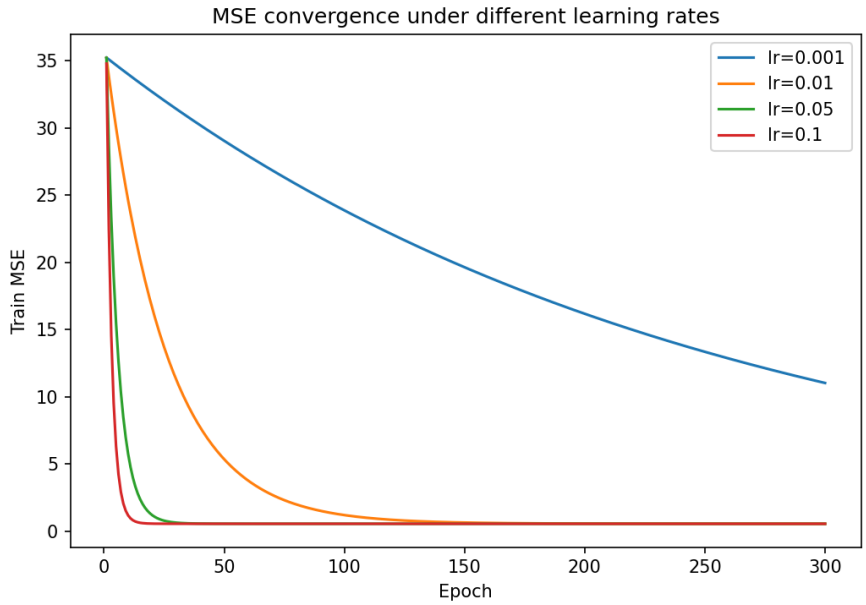
1. 代码

代码与任务 2 相同，不再赘述

2. 控制台输出结果

```
[Template] Implement gd_train(X_train, y_train) to compare learning rates.
lr=0.001 -> Train MSE=10.9926, Test MSE=11.0964
lr=0.010 -> Train MSE=0.5657, Test MSE=0.5780
lr=0.050 -> Train MSE=0.5636, Test MSE=0.5745
lr=0.100 -> Train MSE=0.5629, Test MSE=0.5718
```

3. 训练数据的散点图和模型拟合的回归直线的可视化结果



4. 实验结果分析

表 1: 任务 3a: 不同学习率下的梯度下降性能比较

学习率 (η)	训练误差 (Train MSE)	测试误差 (Test MSE)	总结与解释
0.001	10.9926	11.0964	**收敛太慢 / 欠收敛**：误差远高于其他模型，表明在 300 个 Epoch 内，参数更新步长太小，未能有效接近最小值。
0.010	0.5657	0.5780	**收敛速度慢**：收敛到了合理的最小值，但相比更大的学习率，收敛过程缓慢。
0.050	0.5636	0.5745	**表现良好**：误差较低，收敛速度适中，性能较稳定。
0.100	0.5629	0.5718	**最佳学习率**：在测试集中获得了最低的 MSE，表明它在确保收敛的同时，达到了较快的学习速度和更优的参数解。

(四) 任务 4：正则化 - 岭回归

1. 实验思路

本任务旨在通过实现岭回归的**解析解**，为线性回归模型引入 $L2$ 正则化，以提高模型的泛化能力并解决多重共线性问题。

1. **数据准备：**

- 构造扩展设计矩阵 $\mathbf{X}_{\text{ext}}$：在 $\mathbf{X}_{\text{train}}$ 基础上追加一列全 1 的常数列以包含偏置 b。

2. **岭回归求解：**

- 设定正则化参数 λ（例如 $\lambda = 1.0$）。
- 利用岭回归的解析解一步计算出参数向量 θ：

$$\theta = (\mathbf{X}_{\text{ext}}^{\mathsf{T}}\mathbf{X}_{\text{ext}} + \lambda\mathbf{I})^{-1}\mathbf{X}_{\text{ext}}^{\mathsf{T}}\mathbf{y}$$

其中 $\mathbf{I}$ 为单位矩阵，$\lambda\mathbf{I}$ 项确保了矩阵的可逆性。

3. **结果评估：**

- 将 θ 分解为权重 $\mathbf{w}$ 和偏置 b。
- 计算模型在训练集和测试集上的 MSE，并与普通线性回归（任务 2）的性能进行对比，验证正则化对测试误差的改善效果。

2. 代码

岭回归算法

```python
def ridge_fit_closed_form(X: np.ndarray, y: np.ndarray, lam: float) -> np.ndarray:
    # YOUR CODE HERE: 返回包含 w 和 b 的向量 theta
    # 确保 y 是 N x 1 向量
    y_mat = y.reshape(-1, 1)

    # 矩阵维度: (N x D_ext)
    N, D_ext = X.shape

    # 1. 计算 X^T X (D_ext x D_ext)
    XTX = X.T @ X

    # 2. 构造正则化项 I (D_ext x D_ext)
    # 注意: 岭回归正则化通常不应用于偏置项（即常数列对应的参数b）。
    # 但是，如果直接使用闭式解的矩阵形式，为了保证矩阵可逆性，通常会对所有项加正则化。
    # 为了简化，我们遵循模板提示，对所有 D_ext 个参数都进行正则化（包括偏置b）。
    I = np.eye(D_ext)

    # 如果想排除偏置项b (即最后一列)，可以设置 I 的最后一行的最后一个元素为 0:
    # I[-1, -1] = 0

    lam_I = lam * I

    # 3. 计算 (X^T X + I)
    denom = XTX + lam_I

```

```
    # 4. 计算 (X^T X + I)^(-1)
    # 使用 numpy.linalg.inv 求矩阵的逆
    denom_inv = np.linalg.inv(denom)

    # 5. 计算 X^T y
    XTy = X.T @ y_mat

    # 6. 计算最终参数 theta:  = (X'X + I)^(-1) X'y
    theta = denom_inv @ XTy  # (D_ext x D_ext) @ (D_ext x 1) -> (D_ext x 1)

    return theta.flatten()  # 返回 (D_ext,) 的 1D 数组
```

3. 控制台输出结果

```
[Template] Implement ridge_fit_closed_form(X_train, y_train, lam).
lambda=1.0 -> Train MSE: 0.5627
lambda=1.0 -> Test MSE:  0.5696
```

（五） 拓展任务：模型选择 - 多项式回归

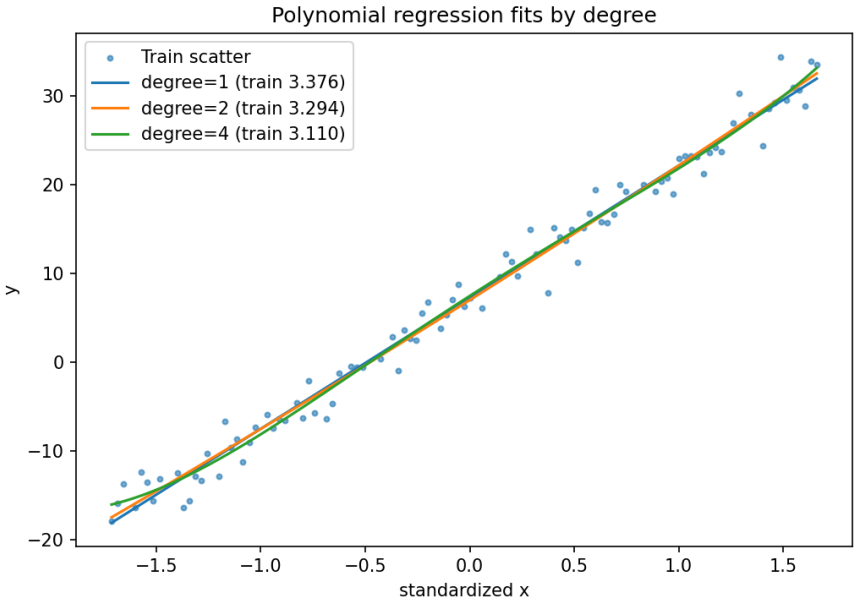
1. 代码

多项式回归

```
# 目标: 实现 poly_features(x, degree) 与闭式解 fit_closed_form(X, y)
def poly_features(x: np.ndarray, degree: int) -> np.ndarray:
    # YOUR CODE HERE: 返回形如 [1, x, x^2, ..., x^degree]
    # 构造特征矩阵 X。结果形如 [x^0, x^1, x^2, ..., x^degree]
    # X_poly 形状为 (N, degree + 1), 其中 N 为样本数
    X_poly = np.ones((x.shape[0], degree + 1))

    # 从 i=1 开始填充 x^1, x^2, ..., x^degree
    for i in range(1, degree + 1):
        X_poly[:, i] = x ** i

    return X_poly


def fit_closed_form(X: np.ndarray, y: np.ndarray) -> np.ndarray:
    # YOUR CODE HERE: 返回 theta = (X^T X)^{-1} X^T y; 建议使用 pinv 增稳
    # 确保 y 是 N x 1 向量
    y_mat = y.reshape(-1, 1)

    # 使用 np.linalg.pinv (伪逆) 求解: theta = (X^T X)^-1 X^T y
    # 伪逆在矩阵不可逆或接近奇异时更稳定。
    XTX_pinv = np.linalg.pinv(X.T @ X)

    # 计算最终参数 theta
```

```
    theta = XTX_pinv @ X.T @ y_mat

    return theta.flatten()  # 返回一维向量
```

2. 训练数据的散点图和模型拟合的回归直线的可视化结果



3. 实验结果分析

表 2: 多项式回归不同阶数下的模型性能比较

阶数 (Degree)	模型类型	训练误差 (Train MSE)	曲线观察	模型诊断
1	线性回归	3.376	拟合曲线是直线。在数据弯曲处（特别是左右两端），曲线与散点偏离较大。	**欠拟合**：模型过于简单，无法捕捉数据中明显的非线性趋势。
2	二次回归	3.294	拟合曲线是平滑的抛物线，弯曲方向与数据点的整体趋势非常吻合。	**拟合效果最好**：曲线走势与数据分布趋势一致，且训练误差明显低于 $d = 1$。
4	四次回归	3.110	拟合曲线与 $d = 2$ 的曲线非常接近，但局部可能更贴近一些噪点。	**略微复杂**：训练误差最低，但拟合曲线与 $d = 2$ 差距不大。高阶项作用有限。

二、 实验总结

本次实验成功从零开始实现了线性回归的核心算法，并深入探究了影响模型性能的多个关键因素，包括求解方法、超参数选择、正则化和模型复杂度。

(一) 求解方法的对比与掌握

实验中对比了两种主要的求解策略：

1. **解析法（任务 1）：**利用正规方程 ($\theta = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$) 实现了参数的一**步到位**计算。该方法保证了找到最小二乘意义下的最优解，但也突显了其在 $\mathbf{X}^\mathsf{T}\mathbf{X}$ 矩阵不可逆（如多重共线性）时的局限性。

2. **迭代法（任务 2）：**实现了批量梯度下降（BGD）算法，通过不断迭代沿着损失函数（MSE）的负梯度方向更新参数。该方法在处理大规模、多特征数据集时展现出更高的计算效率和灵活性。

(二) 优化与超参数调优

在迭代求解器（梯度下降）的基础上，实验分析了超参数**学习率** (η) 对收敛过程的影响（任务 3）：

- **学习率过小** ($\eta = 0.001$) 导致模型收敛速度极慢，在预设迭代次数内出现欠收敛（Train MSE 10.9926）。
- **学习率适中** ($\eta = 0.100$) 实现了最快的收敛速度，并达到了最低的测试误差 (Test MSE 0.5718)，验证了学习率是平衡收敛速度和稳定性的关键。

(三) 模型改进与正则化

为了解决过拟合和特征共线性问题，实验引入了 $L2$ **正则化**（任务 4）：

- **岭回归**通过在损失函数中增加对权重大小的惩罚 ($\lambda \sum w_i^2$) 来收缩模型参数，从而提升模型的泛化能力。
- 在 $\lambda = 1.0$ 下，岭回归模型获得了优于普通线性回归的测试误差 (Test MSE 0.5696)，证实了 $L2$ 正则化在提高模型稳定性方面的有效性。

(四) 模型复杂度与选择

通过多项式回归拓展任务，实验探讨了模型**复杂度**对拟合效果的影响：

- **低阶模型** ($d = 1$) 表现为欠拟合，未能捕捉数据的非线性趋势（Train MSE 3.376）。
- **中阶模型** ($d = 2$) 拟合效果最佳（Train MSE 3.294），曲线完美贴合了数据底层模式。
- **高阶模型** ($d = 4$) 虽然训练误差最低（Train MSE 3.110），但其曲线已开始略微偏离趋势，预示着模型复杂度过高可能导致过拟合的风险。

综上，本次实验不仅从代码层面实现了线性回归及其变种，更从原理上理解了求解方法、超参数选择、正则化和模型复杂度对回归模型性能的决定性作用。