PRML Course Report

学号	姓名	班级	邮箱
2021302488	郑安玮	10042101	nekozo0315@163.com

Task 1 多项式回归

1 任务描述

多项式回归是一种回归分析形式，在这种形式中，自变量 (x) 和因变量 (y) 之间的关系被建模为 (n) 阶多项式。使用机器学习的方法来创建一个多项式回归模型，该模型可以根据给定的数据集预测结果。数据集由自变量 (x) 和因变量 (y) 组成，你的任务是找到一个多项式，能最好地描述 (x) 和 (y) 之间的关系。

2 数据集描述

本实验选取数据集包含125个样本点，每个样本具有一个自变量(x)和一个因变量(y)。数据集根据4:1的比例划分为训练集和测试集。

3 数学分析

多项式回归是一种回归分析方法，他通过多项式函数来拟合数据集，适用于数据和一个或多个非线性关系的情况。多项式回归模型的形式一般如下：

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_n x^n + \epsilon$$

在多项式回归模型中，自变量x的不同次方 x^2, x^3 被作为新的特征加入到模型，通过训练数据，使用普通最小二乘法(OLS)等算法来估计回归系数 $\beta_0, \beta_1, \beta_2 \ldots \beta_n, \epsilon$,从而捕捉数据中的非线性关系。

4 算法分析

多项式回归的实现比较简单，本次作业使用Python语言中的`scikit-learn`库进行，代码在`Jupiter`环境支持下运行。

多项式回归的实现步骤主要如下：

> 1 数据预处理：读取数据集，将自变量扩展为多项式特征，即 $x, x^2, x^3 \ldots x^n$
> 2 模型训练：将训练用的扩展自变量与因变量提供给模型进行拟合，估计回归系数
> 3 模型评估：使用验证用数据评估模型性能，计算均方误差MSE等
> 4 预测：使用训练好的模型对测试用数据进行预测

5 代码实现与分析

5.1 数据集提取和预处理

数据存储在csv文件中，因此导入pandas库来提取数据,由于涉及特征扩展，需要将自变量与因变量转换为numpy数组，并将自变量进行维度扩展，从而使用sklearn.preprocessing中的PolynomialFeatures方法进行计算。

```
import pandas as pd
import numpy as np

trn_path = "data/Data/poly_reg/train_dataset.csv"
val_path = "data/Data/poly_reg/test_dataset.csv"

# 训练集数据处理
df_trn = pd.read_csv(trn_path, header=0)
x_trn = np.array(df_trn['x'].values).reshape(-1, 1)
y_trn = np.array(df_trn['y'].values)

# 验证集数据处理，与训练集处理方式保持一致
df_val = pd.read_csv(val_path, header=0)
x_val = np.array(df_val['x'].values).reshape(-1, 1)
y_val = np.array(df_val['y'].values)
```

接下来设置多项式阶数，对自变量进行特征转换。多项式阶数可根据需求改变，从而测试不同阶数对数据的拟合程度。

```
from sklearn.preprocessing import PolynomialFeatures

# 多项式阶数
degree = 2  # 测试不同的阶数对数据的拟合程度
# 转换器
poly = PolynomialFeatures(degree=degree)

# 转换特征
x_trn_poly = poly.fit_transform(x_trn)
# print(x_trn)
# print(x_trn_poly)
x_val_poly = poly.transform(x_val)
```

5.2 模型训练

作业用模型选择sklearn.linear_model中的LinearRegression，通过fit()方法进行模型训练，同时计算时间戳差值，观察训练时间。

```
from sklearn.metrics import mean_squared_error
import time

y_pred = model.predict(x_val_poly)
mse = mean_squared_error(y_val, y_pred)
print(f"MSE: {mse}")
```

5.3 模型评估与预测

本次作业并未设置验证集，直接使用测试集作为验证数据进行预测，并计算均方误差(MSE)。

```
from sklearn.metrics import mean_squared_error

y_pred = model.predict(x_val_poly)
mse = mean_squared_error(y_val, y_pred)
print(f"MSE: {mse}")
```

5.4 结果可视化

为了直观观察训练数据点、预测点分布以及回归曲线，导入matplotlib.pyplot进行绘图，绘图过程不作分析。

```
import matplotlib.pyplot as plt

x_range = np.linspace(x_trn.min(), x_trn.max(), 500).reshape(-1, 1)
x_range_poly = poly.transform(x_range)
y_range_pred = model.predict(x_range_poly)

plt.scatter(x_val, y_val, color='blue', label='Validation point')
plt.scatter(x_trn, y_trn, color='red', label='Training point')
# plt.plot(x_val, y_pred, color='orange')
plt.plot(x_range, y_range_pred, color='black', label=f'Degree {degree} polynomial regression')

plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.savefig('result.png')
plt.show()
```

使用coef_与intercept_方法获取多项式回归系数和截距项，通过格式化print输出回归曲线表达式。

```
# 获取模型的系数
coefficients = model.coef_
intercept = model.intercept_

# 构造多项式表达式字符串
polynomial_expression = f"{intercept}"  # 截距项

for i, coef in enumerate(coefficients, start=0):
    term = f"{coef:.3f}*x^{i}" if coef != 0 else ''  # 去除系数为0的项
    polynomial_expression += (' + ' + term if term else '')

print(f"多项式回归模型对应的多项式表达式为：{polynomial_expression}")
```

5.5 填写测试结果

同样使用pandas库，过程简单，不做分析。

```
# 填写测试集 test.csv
test_csv = pd.read_csv('test.csv', header=0)
test_x = np.array(test_csv['x'].values).reshape(-1, 1)
test_x_poly = poly.transform(test_x)
# print(test_x)

pred_y = model.predict(test_x_poly).tolist()
# print(pred_y)
results = pd.DataFrame({'x':test_csv['x'].values, 'y':pred_y})
results.to_csv('results.csv', index=False)
```

6 运行测试与结果展示

6.1 不同多项式阶数下的MSE

- degree = 2
  - MSE: 0.48883668754145754
- degree = 4
  - MSE:0.48645712023557885

```python
from sklearn.metrics import mean_squared_error

y_pred = model.predict(x_val_poly)
mse = mean_squared_error(y_val, y_pred)
print(f"MSE: {mse}")
```

[11] ✓ 0.0s

```
MSE: 0.48645712023557885
```

- degree = 8
  - MSE:0.5045193570740093

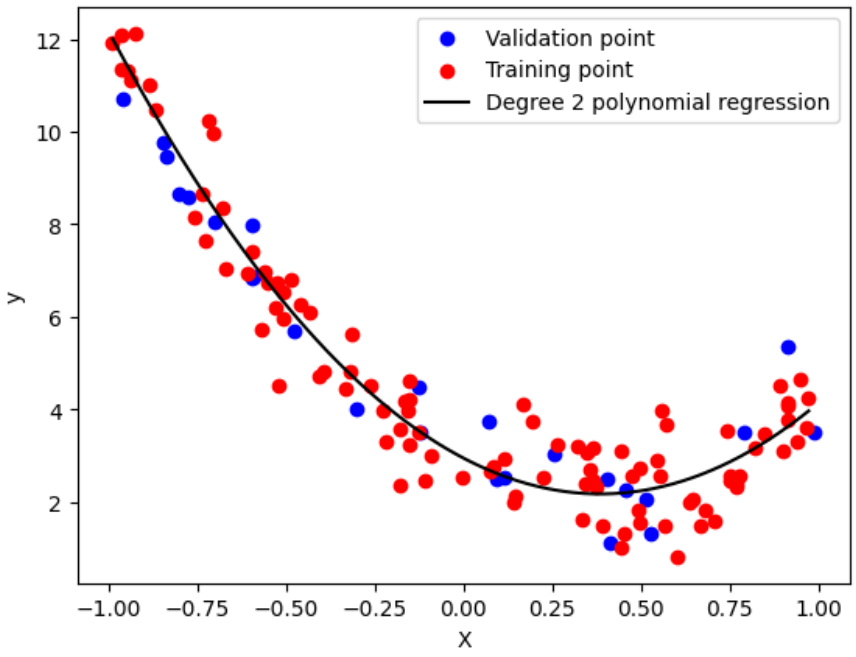
```python
from sklearn.metrics import mean_squared_error

y_pred = model.predict(x_val_poly)
mse = mean_squared_error(y_val, y_pred)
print(f"MSE: {mse}")
```

[8] ✓ 0.0s

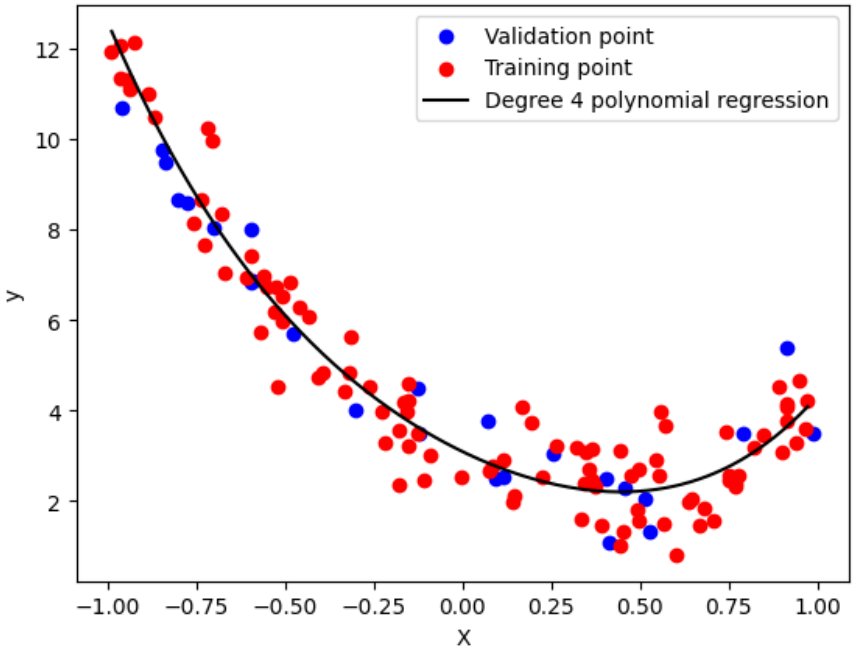
```
MSE: 0.5045193570740093
```

6.2 不同多项式阶数下的图像

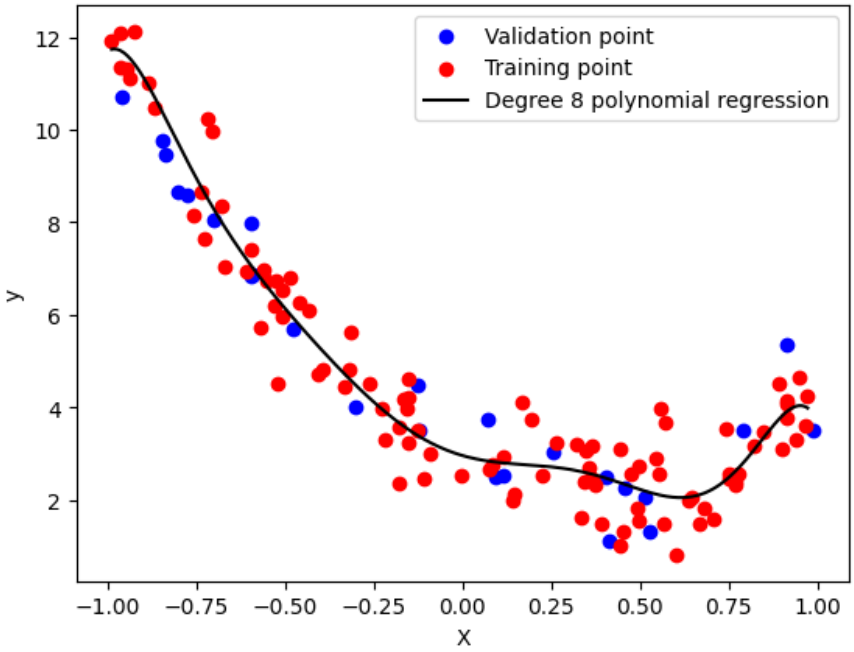
- degree = 2



- degree = 4



- degree = 8



6.3 不同多项式阶数下的回归曲线表达式

- degree = 2
  - $2.9408127516015634 + -4.009x^1 + 5.207x^2$
- degree = 4
  - $3.088229332443209 + -3.800x^1 + 3.916x^2 + -0.325x^3 + 1.442x^4$
- degree = 8
  - $2.954138666711126 + -2.215x^1 + 8.733x^2 + -10.212x^3 + -26.209x^4 + 15.557x^5 + 50.248x^6 + -7.126x^7 + -28.024x^8$