人工智能实验：实验二

—— 回归模型

3190102060　黄嘉欣

一、回归概述

假设现在有一些数据点，我们用一条线对这些点进行拟合（该线称为最佳拟合直线），这个拟合过程就称作回归。在统计学中，回归分析指的是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。回归分析按照涉及的变量的多少，分为一元回归和多元回归分析；按照因变量的多少，可分为简单回归分析和多重回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。在本次实验中，我们将主要学习线性回归、多项式回归和逻辑回归三种算法。

二、线性回归

① 算法概述

线性回归模型对输入特征加权求和，再加上一个偏置项的常数，以此进行预测，其数学表达式为：$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$，其中，$\hat{y}$为预测的值，$n$为特征数，$x_i$为第$i$个特征值，而$\theta_i$为第$i$个模型参数（包含偏置项$\theta_0$及权重$\theta_1$、$\theta_2$等）。将上述式子写为向量的形式为：$\hat{y} = \boldsymbol{\theta} \cdot \mathbf{x}$。当已知结果$y$时，我们可以利用最小二乘法反求出模型参数向量为：$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y}$，从而完善模型，利用该$\boldsymbol{\theta}$实现预测。

② 实验题目

生成直线$y = 4 + 3x$在$x \in [0,2]$上的随机数据，并附加高斯噪声。对其进行线性回归，求出$\boldsymbol{\theta}$，利用此回归模型对$x = 0$和$x = 2$两点进行预测，最后利用 plot 完成随机数据和预测结果的可视化。

③ 实验结果与分析

如图2.1，根据随机数据求出的$\boldsymbol{\theta} = (4.152, 2.837)$，故回归模型为$y = 4.152 + 2.837x$，与真实的直线方程相接近，预测出的两点分别为$(0, 4.152321)$和$(2, 9.825961)$。可以看到，回归直线基本上处于随机数据的中心，而预测点的位置也比较合理，拟合结果较好。

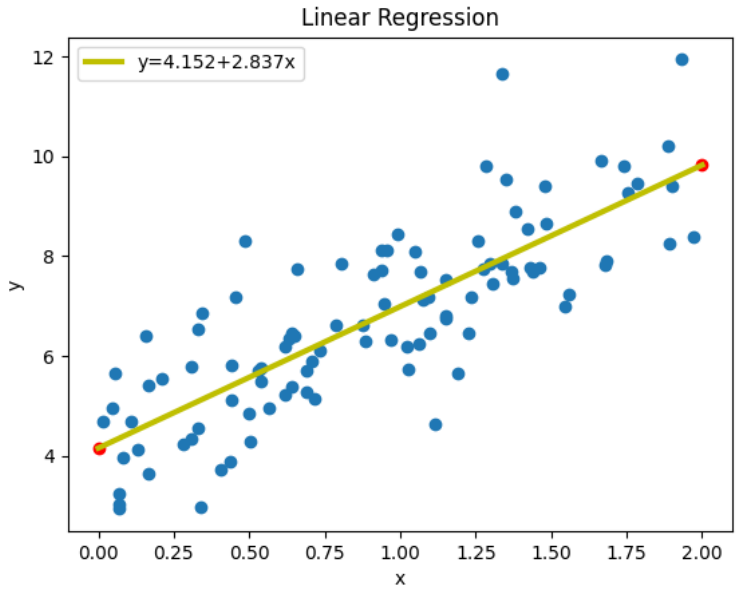

图 2.1 线性回归数据可视化

三、多项式回归

① 算法概述

当数据不能用直线描述时，我们仍然可以用线性模型拟合非线性数据，即利用多项式回归，将每个特征的幂次方添加为一个新的特征，然后在此扩展特征集上训练一个线性模型。利用此回归模型，我们便同样可以完成结果的预测。

② 实验题目

生成二次曲线$y = 2 + x + 0.5x^2$在$x \in [-3,3]$上的随机数据，并附加高斯噪声。利用 PolynomialFeatures 类来转换训练数据，将训练集中每个特征的平方添加为新特征；再调用 Scikit-Learn 中的 LinearRegression 模块，对上述产生的二级多项式数据做线性回归，最后可视化回归模型和数据。

③ 实验结果与分析

如图 3.1，拟合得到的模型为：$y = 1.81 + 1.02x + 0.526x^2$，与实际的多项式方程差异不大。回归得到的二次曲线基本上处于随机数据的中心，实验效果较好。

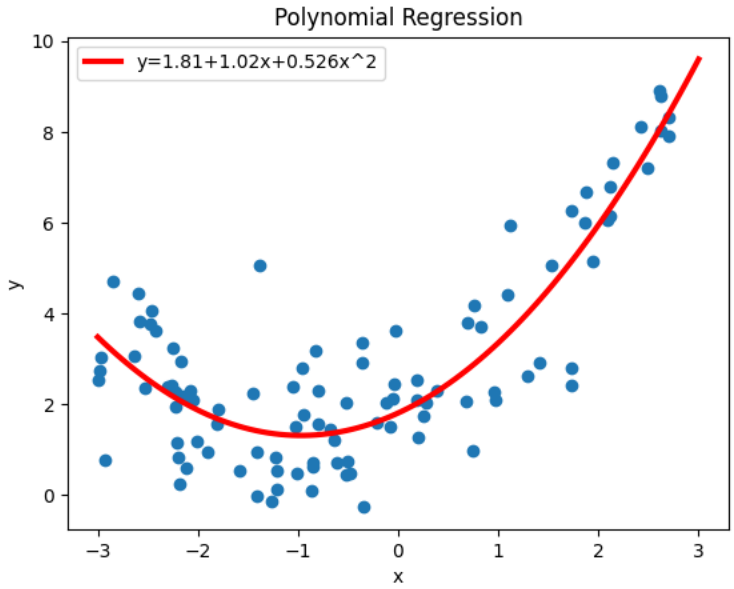

图 3.1 多项式回归数据可视化

四、逻辑回归

① 算法概述

逻辑回归用于估算一个实例属于某个特定类别的概率，如一封电子邮件属于垃圾邮件的概率是多少。当预估的概率超过 50%，此回归模型会预测该实例属于某类别（标记为 1）；否则预测为不是（标记为 0）。因此，逻辑回归模型相当于一个二元分类器，其概率估计用向量表示为：$\hat{p} = \sigma(\mathbf{x}^T\boldsymbol{\theta})$，其中$\sigma(t)$为 sigmoid 函数，其数学表达式为：$\sigma(t) = \frac{1}{1+e^{-t}}$。对预测结果，采用硬判决，即：$\hat{y} = \begin{cases} 0, & \hat{p} < 0.5 \\ 1, & \hat{p} \geq 0.5 \end{cases}$。注意到当$t < 0$时，$\sigma(t) < 0.5$，而当$t \geq 0$时，$\sigma(t) \geq 0.5$。因此，当$\mathbf{x}^T\boldsymbol{\theta}$为正时，逻辑回归模型预测 1；当$\mathbf{x}^T\boldsymbol{\theta}$为负时，逻辑回归模型预测 0。

② 实验题目

sklearn 的官方鸢尾植物逻辑回归实例：150 朵、三个品种，数据集中包含有花的萼片及花瓣长度和宽度。加载该数据，通过 sklearn.linear_model 中的 LogisticRegression 模块，以花瓣宽度为特征，创建一个分类器，用于检测维吉亚鸢尾花，并可视化该模型检测花瓣宽度在 0-3cm 之间的鸢尾花概率。

③ 实验结果与分析

如图 4.1，为概率估计$\hat{p}$的图像。根据逻辑回归模型的 predict_proba 方法，当花瓣宽度大于 2.0cm 时，有很大的概率可认为其为维吉亚鸢尾花；当花瓣宽度小于 1.2cm 时，

有很大的概率可认为其不是维吉亚鸢尾花。两条曲线约在宽度为 1.6cm 处产生了交叉。利用 predict 方法，能够得到图 4.2 所示的$\hat{y}$的预测图像。可以发现，当花瓣宽度大于 1.6cm 时，分类器输出为 1，即认为该花朵是维吉亚鸢尾花；否则输出为 0，认为它不是维吉亚鸢尾花，与图 4.1 相吻合。

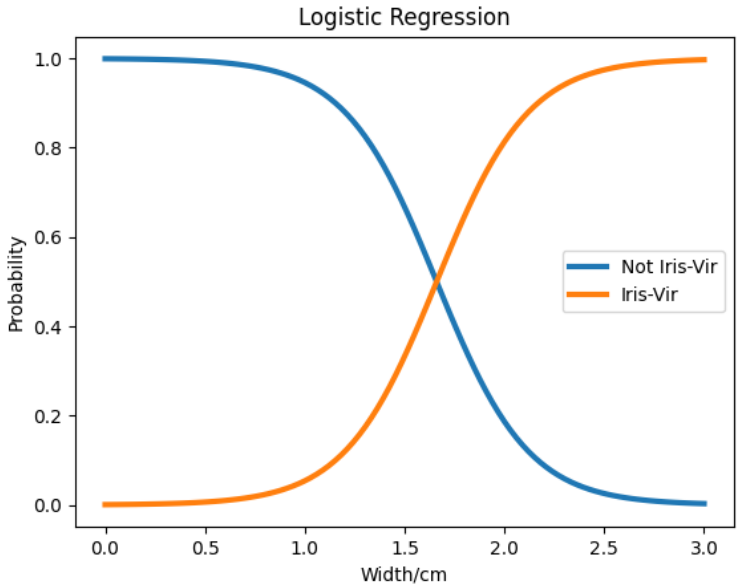

图 4.1 花瓣宽度与花类概率可视化

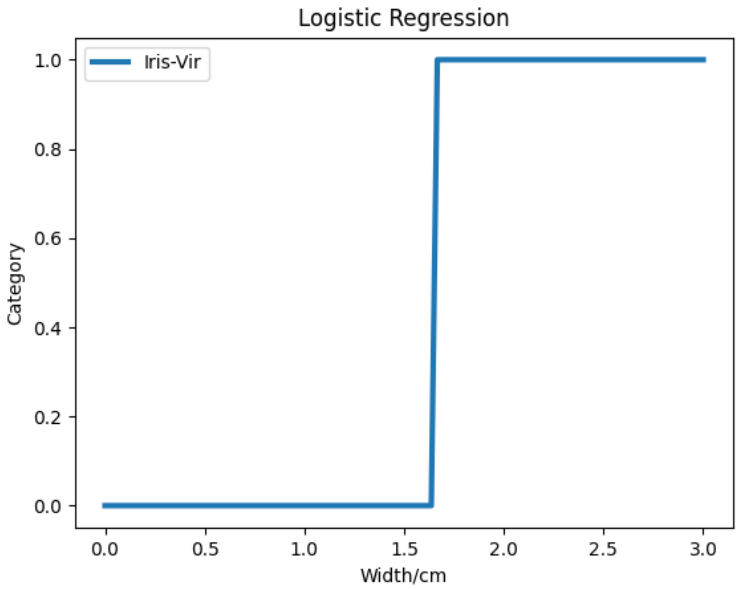

图 4.2 花瓣宽度与花类可视化

五、附录：实验 Python 代码（可见 src 目录）

① linear.py

```
from numpy import *
import matplotlib.pyplot as plt

X = 2*random.rand(100,1)
y = 4+3*X+random.randn(100,1)                                    # 高斯噪声

biasX = ones((100,1))
X_Mat = c_[biasX,X]                                               # 构造X矩阵
# print(X_Mat)

# 做线性回归
X_Mat_T = X_Mat.transpose()                                       # 转置
theta = dot(dot(linalg.inv(dot(X_Mat_T,X_Mat)),X_Mat_T),y)        # 矩阵乘法
b = round(float(theta[0]),3)                                      # 截距
k = round(float(theta[1]),3)                                      # 斜率

# 预测
x1 = [1,0]                                                        # x = 0
x2 = [1,2]                                                        # x = 2
y1 = dot(x1,theta)
y2 = dot(x2,theta)
print("(0,%f)" %y1)
print("(2,%f)" %y2)

# 可视化
plt.scatter(X, y)
X_axis = linspace(0,2,100)                                        # 回归模型
X_axis_Mat = c_[biasX,X_axis]
y_predict = dot(X_axis_Mat,theta)
l1, = plt.plot(X_axis,y_predict,linewidth=3,c='y')
plt.legend(handles=[l1],labels=['y='+str(b)+'+'+str(k)+'x'],loc='best')
plt.scatter([0],[y1],c='r')                                       # 预测结果x = 0
plt.scatter([2],[y2],c='r')                                       # 预测结果x = 2
plt.xlabel('x')
plt.ylabel('y')
plt.title("Linear Regression")
plt.show()
```

② polynomial.py

```
from numpy import *
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

m = 100
X = 6*random.rand(m,1)-3
y = 0.5*X**2+X+2+random.randn(m,1)
poly_features = PolynomialFeatures(degree=2,include_bias=False)
X_poly = poly_features.fit_transform(X)                  # 平方项
# print(X_poly[:,0])
# print(X_poly[:,1])

# 线性回归
model = LinearRegression()                               # 导入模型
model.fit(X_poly,y)                                      # 训练模型
k1 = round(model.coef_[0][0],3)                          # 一次项系数
k2 = round(model.coef_[0][1],3)                          # 二次项系数
```

```
b = round(model.intercept_[0],3)                    # 截距
# print("k: %f" % k)
# print("b: %f" % b)

# 可视化
plt.scatter(X,y)                                    # 数据
X_axis = linspace(-3,3,100)                         # 回归模型
y_predict = b+k1*X_axis+k2*X_axis**2
l1, = plt.plot(X_axis,y_predict,linewidth=3,c='r')
plt.legend(handles=[l1],labels=['y='+str(b)+'+'+str(k1)+'x'+'+'+str(k2)+'x^2'],loc='best')
plt.xlabel('x')
plt.ylabel('y')
plt.title("Polynomial Regression")
plt.show()
```

③ logistic.py

```
from numpy import *
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt

# 加载数据
iris = datasets.load_iris()
X = iris["data"][:,3:]
y = (iris["target"]==2).astype(int)

# 分类器
classifier = LogisticRegression()                                   # 导入模型
classifier.fit(X,y)                                                 # 训练模型
width = linspace(0,3,100)                                           # 预测
probability = classifier.predict_proba(reshape(width,[len(width),1]))
# probability = classifier.predict(reshape(width,[len(width),1]))  # 预测类别

# 可视化
l1, = plt.plot(width,probability[:,0],linewidth=3)
l2, = plt.plot(width,probability[:,1],linewidth=3)
# l1, = plt.plot(width,probability,linewidth=3)
plt.legend(handles=[l1,l2],labels=["Not Iris-Vir","Iris-Vir"],loc='best')
# plt.legend(handles=[l1],labels=["Iris-Vir"],loc='best')
plt.xlabel('Width/cm')
plt.ylabel('Probability')
# plt.ylabel('Category')
plt.title("Logistic Regression")
plt.show()
```