人工智能实验报告 实验7-1

姓名:武珂晗 学号:22336249

一.实验题目

购房预测分类任务

二.实验内容

`data.csv` 数据集包含三列共400条数据，其中第一列 `Age` 表示用户年龄，第二列 `EstimatedSalary` 表示用户估计的薪水，第三列 `Purchased` 表示用户是否购房。
根据用户的年龄以及估计的薪水，利用感知机学习算法预测用户是否购房，并画出数据可视化图、loss曲线图，计算模型收敛后的分类准确率。

1.算法原理

感知机学习算法 的原理是基于线性分类器，通过不断迭代更新权重和偏置，使得模型能够找到能够正确划分数据的超平面。具体实现如下：

1. 初始化权重向量和偏置。
2. 遍历训练数据集，对每个样本计算预测值，并根据预测值与真实值之间的差异更新权重和偏置。
3. 不断迭代直到模型收敛或达到预设的迭代次数。

单层感知机
单层感知机（Perceptron）是一种基本的线性分类器，可以用于逻辑回归任务。其接收多个输入信号，并产生一个输出信号。

考虑输入向量为x，权重向量为ω，偏置为b，则感知机的输出y定义为：

$$y = \phi(\omega^T x + b)$$

其中，ϕ是一个激活函数，根据$\omega^T x + b$的正负来决定输出 0 或 1。

数据标准化和归一化

- 归一化：数据归一化的目的是使得各个特征对目标（输出）变量的影响一致，会将特征数据进行伸缩变化，所以数据归一化是会改变特征数据分布的。
  将数据映射到[0,1]区间

$$x^* = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Z-Score：数据标准化为了不同特征之间具备可比性，经过标准化变换之后的特征数据分布没有发生改变
  - 当数据特征取值范围或单位差异较大时，最好是做一下标准化处理
  - 处理后的数据均值为0，方差为1

2.关键代码展示

单层感知机算法实现如下：

```
class Perceptron:
    #初始化感知器的权重、偏置和学习率
    def __init__(self, input_dim, lr=0.3):
        self.weights = np.random.randn(input_dim, 1)
        self.bias = np.random.randn(1)
        self.lr = lr    #学习率
        return

    #接收输入 x，计算加权和并通过激活函数得到预测结果
    def forward(self, x):
        z = np.dot(x, self.weights) + self.bias
        return 1 / (1 + np.exp(-z))

    #损失函数，接收预测值 y_pred 和真实值y，计算二者之间的均方误差
    def loss(self, y_pred, y):
        return np.mean((y_pred - y) ** 2)

    #根据损失计算权重和偏置的梯度，更新权重和偏置
    def backprop(self, x, y):
        y_pred = self.forward(x)
        error = y_pred - y
        d_weights = np.dot(x.T, 2 * error * y_pred * (1 - y_pred)) / len(y)
        d_bias = np.sum(2 * error * y_pred * (1 - y_pred)) / len(y)
        self.weights -= self.lr * d_weights
        self.bias -= self.lr * d_bias
        return
```

sigmoid 函数:

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))
```

画出散点图:

```
def plot_points(data, model):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

    X = data.iloc[:, :-1]
    X0 = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
    Y = data.iloc[:, -1]

    ax1.scatter(X.iloc[:, 0][Y == 0], X.iloc[:, 1][Y == 0], c='blue', label='Not Purchased')
    ax1.scatter(X.iloc[:, 0][Y == 1], X.iloc[:, 1][Y == 1], c='red', label='Purchased')
    ax1.set_xlabel('Age')
    ax1.set_ylabel('Salary')
    ax1.legend()

    ax2.scatter(X0.iloc[:, 0][Y == 0], X0.iloc[:, 1][Y == 0], c='blue', label='Not Purchased')
    ax2.scatter(X0.iloc[:, 0][Y == 1], X0.iloc[:, 1][Y == 1], c='red', label='Purchased')
    w1, w2 = model.weights
    b = model.bias
    x_line =  np.array([X0.iloc[:,0].min(),X0.iloc[:,0].max()])
    y_line = (-w1 / w2) * x_line - (b / w2)
    ax2.plot(x_line, y_line, color='purple', label='Decision Boundary')
    ax2.set_title('Predictions')
    ax2.set_xlabel('Age')
```

```
    ax2.set_ylabel('Salary')
    ax2.legend()

    plt.show()
```

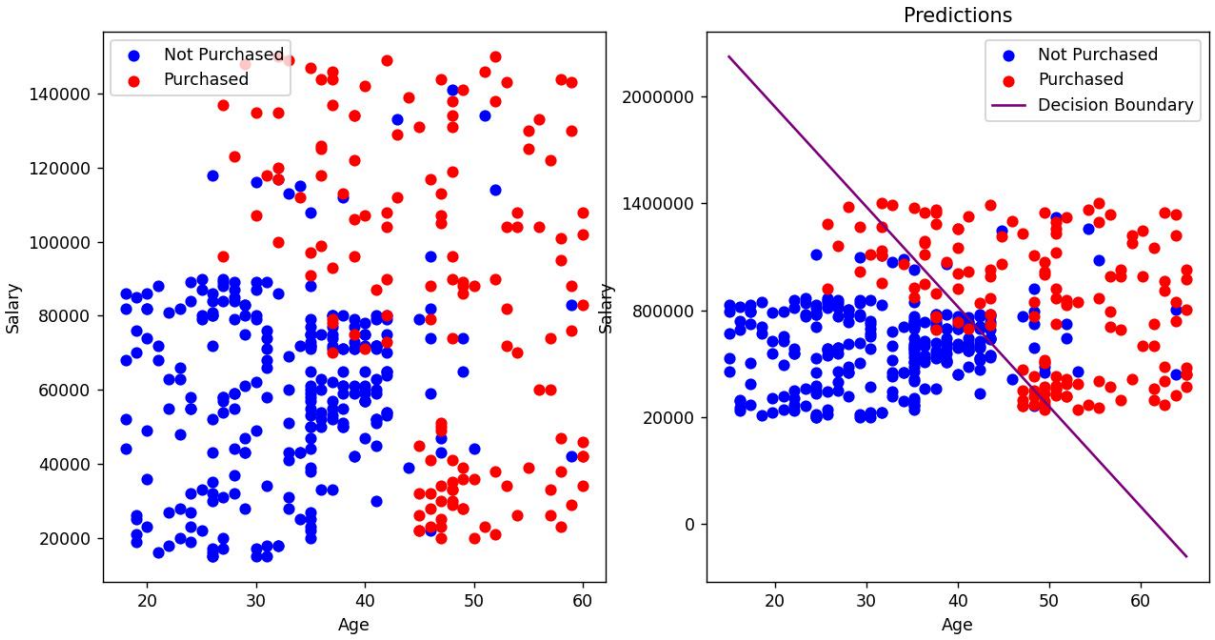
3.创新点&优化

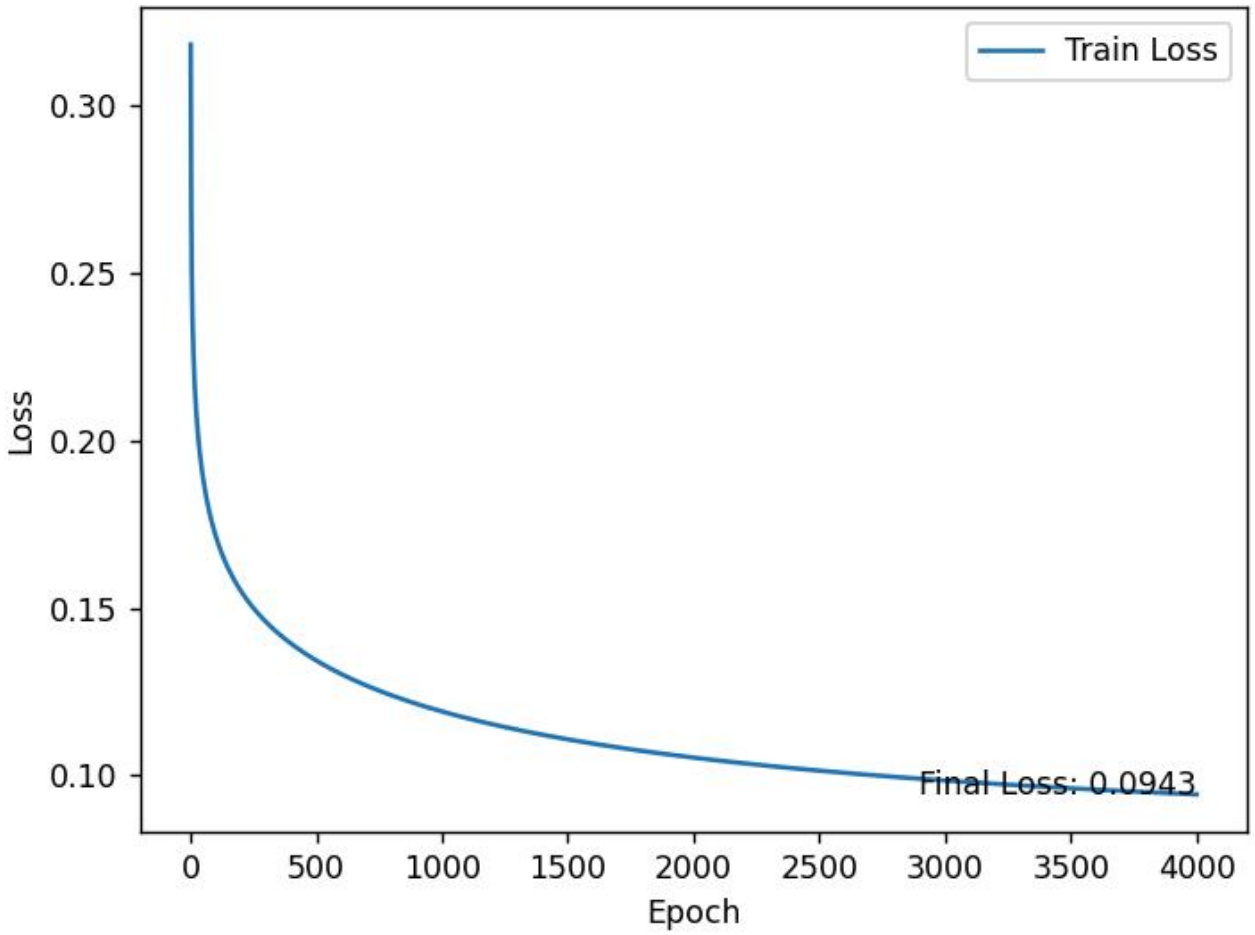
- 在感知机模型的初始化部分，使用了 `np.random.randn()` 来初始化权重向量，这样做有助于提高模型的收敛速度和准确性，因为随机初始化可以防止模型陷入局部最优解。
- 在损失函数和反向传播部分，使用了向量化的方法来计算梯度，可以提高代码的运行效率，加快模型训练的速度。
- 使用 `sigmoid` 函数代替原始的 `sign` 函数作为激活函数，将输入的实数映射到0到1之间的概率值，获得一条平滑的曲线。

三.实验结果及分析

1.实验结果展示示例

数据可视化图和loss曲线图如下：





2.评测指标展示及分析

为确保模型收敛，不断更改迭代次数如下：

```
Epoch:   500 Loss: 0.134315
Epoch:  1000 Loss: 0.119065
Epoch:  1500 Loss: 0.110742
Epoch:  2000 Loss: 0.105287
Epoch:  2500 Loss: 0.101381
Epoch:  3000 Loss: 0.098432
Epoch:  3500 Loss: 0.096126
Epoch:  4000 Loss: 0.094278
The accuracy is: 84.50%
```

经过模型训练后，得到的分类准确率为84.50%。

四.参考资料(可选)

- https://blog.csdn.net/qq_42642142/article/details/120778439