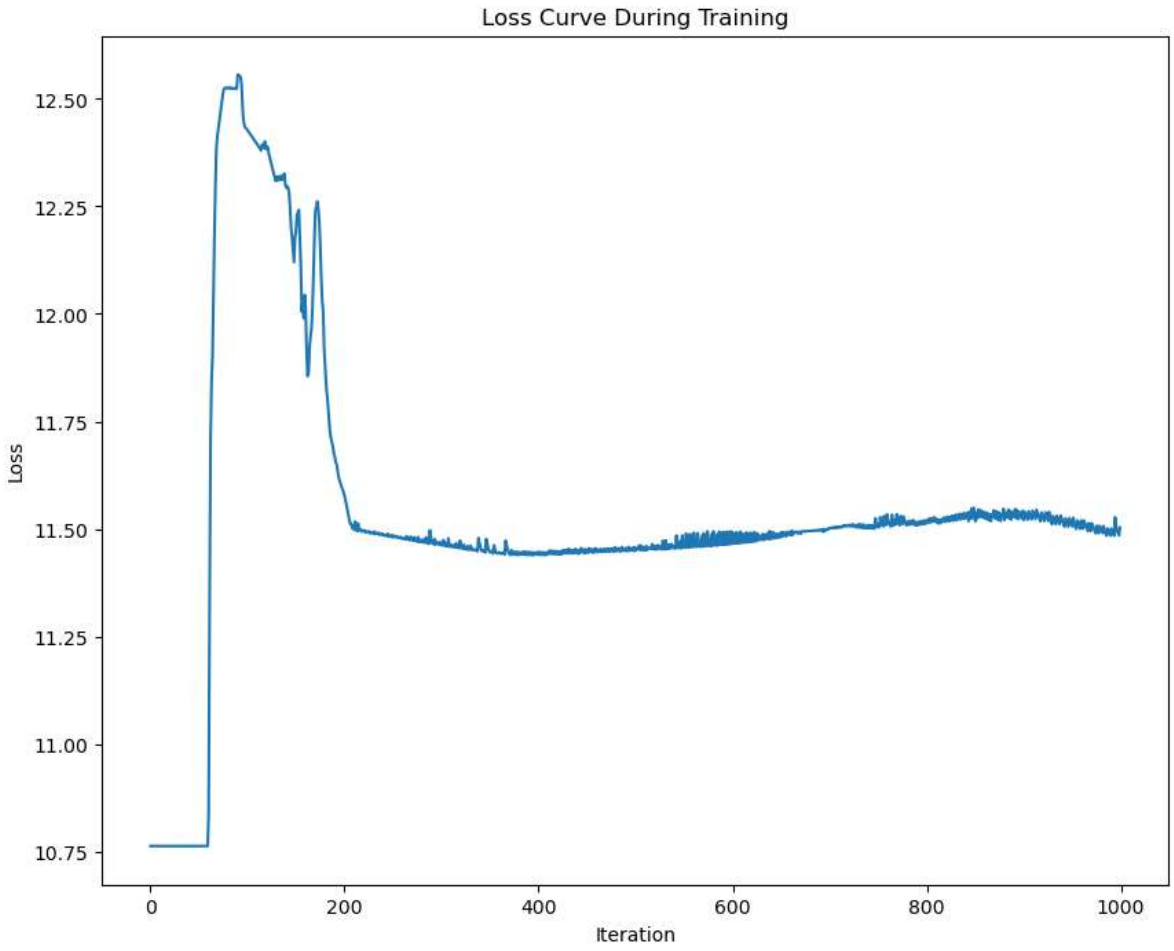
Lab1 逻辑回归

- 姓名：吴欣怡
- 学号：PB21051111

实验要求提交的材料

一次训练中的loss曲线：



不同参数的效果比较

序号\参数	lr	gamma	penalty	accuracy
1	0.0001	0.1	l1	0.7258
2	0.0001	0.1	l2	0.7258
3	0.001	0.4	l1	0.7258
4	0.001	0.4	l2	0.7258
5	0.005	0.4	l1	0.71
6	0.005	0.4	l2	0.71

最佳准确率

0.7258

Logistic.py

实验分析

需要实现fit()函数和predict()函数

实验过程

fit函数中，输入学习率，收敛性范围，最大迭代次数等参数，输出每次迭代中的损失函数。

首先理解框架中的内容，若要考虑截距项，则在X的最左侧增加一列全1列向量。

把权重初始化为全1向量，在每次迭代中，依次计算X和权重点乘，代入sigmoid预测概率值，并且根据损失函数、梯度函数的公式计算出损失函数、梯度函数的初始情况。

然后考虑l1和l2正则化这两种情况会对梯度和损失函数造成影响。

```
    def fit(self, X, y, lr=0.00001, tol=1e-7, max_iter=1000):
        if self.fit_intercept:
            X = np.c_[np.ones(X.shape[0]), X]

        # Initialize coefficients
        self.coef_ = np.ones(X.shape[1])
        # List to store loss values at each iteration
        losses = []

        n_samples = X.shape[0]

        for iteration in range(int(max_iter)):
            linear_output = np.dot(X, self.coef_)
            y_pred = self.sigmoid(linear_output)
            y_pred=np.array(y_pred)
            y_pred = np.clip(y_pred, 1e-15, 1 - 1e-15)
            loss = -np.mean(y * np.log(y_pred) + (1 - y) * np.log(1 - y_pred))
            y_pred = np.where(y_pred >= 0.5, 1, 0)
            gradient = np.dot(X.T,(y_pred - y)) / n_samples # Calculate the
gradient

            if self.penalty == "l1":
                gradient[1:] =
gradient[1:]+self.gamma*np.sign(self.coef_[1:])/(2*n_samples)
                loss = loss + np.sum(np.abs(self.coef_[1:])) * self.gamma / (2 *
n_samples)
            elif self.penalty == "l2":
                gradient[1:] = gradient[1:]+self.gamma*self.coef_[1:]/n_samples
                loss = loss + np.sum(self.coef_[1:]**2) * self.gamma / (2 *
n_samples)
            losses.append(loss)
            #norm_diff = np.linalg.norm(gradient)
            if abs(lr*gradient) < tol:
                break
            self.coef_ = self.coef_ - lr* gradient
```

```
        print(losses)
        return losses
```

predict()函数中，只需要代入sigmoid函数并把结果返回即可：

```
    def predict(self, X):

        if self.fit_intercept:
            X = np.c_[np.ones(X.shape[0]), X]

        linear_output = np.dot(X, self.coef_)
        probs = self.sigmoid(linear_output)  #使用sigmoid函数
        return probs
```

Loan.ipynb

数据清洗

对于数值类属性采用均值填补缺失值，对于多分类属性采用众数来填补缺失值。

划分数据集为测试集和训练集

将每条数据随机编号并且按照9：1的比例划分所有索引，据此划分实现随机划分数据集。拆分为需要验证的属性数据y和其它数据X，转成numpy数组。

训练和验证准确率

训练并画出loss关于迭代次数变化的曲线。由于predict函数得到的是概率，所以按照概率0.5为分界线来判断预测结果为正例还是反例。在测试集上使用predict，结合一个判断准确率的函数（按照预测结果和实际结果对应位置相同个数/总数据数得到准确率）。使用不同参数代入，比较准确率。

实验总结

对于l1和l2部分需要怎么参考公式写出表达式思考了很久，让我对numpy的使用方法有了更好的掌握。期间维度不匹配等问题也困扰了我很久，但好在最后都解决了。