逻辑回归

董峦 新疆农业大学

引言

当今社会，从传统的信用卡到花呗、京东白条等各类信用类金融产品层出不穷，人们向金融机构借贷解决了资金不足的燃眉之急，在超前消费中获得了享受，但还贷的紧迫感也给人造成了不小的压力。由于个人信用在当今社会十分重要，信用违约将给个人生活和发展带来很大影响，对金融机构来说信用违约是一种需要防范的风险。

本文针对一组信用卡还款记录数据预测何种情况可能发生违约，观察哪些因素可以用于预判信用违约。研究该问题有助于预判可能发生的风险从而正确运用干预措施。

数据集

在`UCI Machine Learning Repository`有这样一份数据集：**default of credit card clients Data Set**（https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients#），出自论文：`The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients` 。该论文分析了台湾某银行自2005年4月到9月30000个信用卡账户的账单和还款情况。

该数据集有25个特征，特征名称及数值含义如下：

- ID：序号（该字段对预测无益，将被丢弃）
- LIMIT_BAL：信用额度
- SEX：性别（1-男性，2-女性）
- EDUCATION：教育背景（1 = 研究生，2 = 本科， 3 = 高中，4 = 未定义。其它取值都将归为第4种）
- MARRIAGE：婚姻情况（1-已婚，2-单身，3-未定义。其它取值都将归为第3种）
- AGE：年龄
- PAY_0~PAY_6：过去6个月每月还款状态（PAY_0为2005年9月还款记录，PAY_1缺失，PAY_6为2005年4月的。-2代表未消费，-1代表按时还款，0表示使用循环信用（revolving credit），1至8的取值代表延期月数，其它取值未定义）
- BILL_AMT1~BILL_AMT6：过去6个月每月的账单
- PAY_AMT1~PAY_AMT6：过去6个月每月的还款额

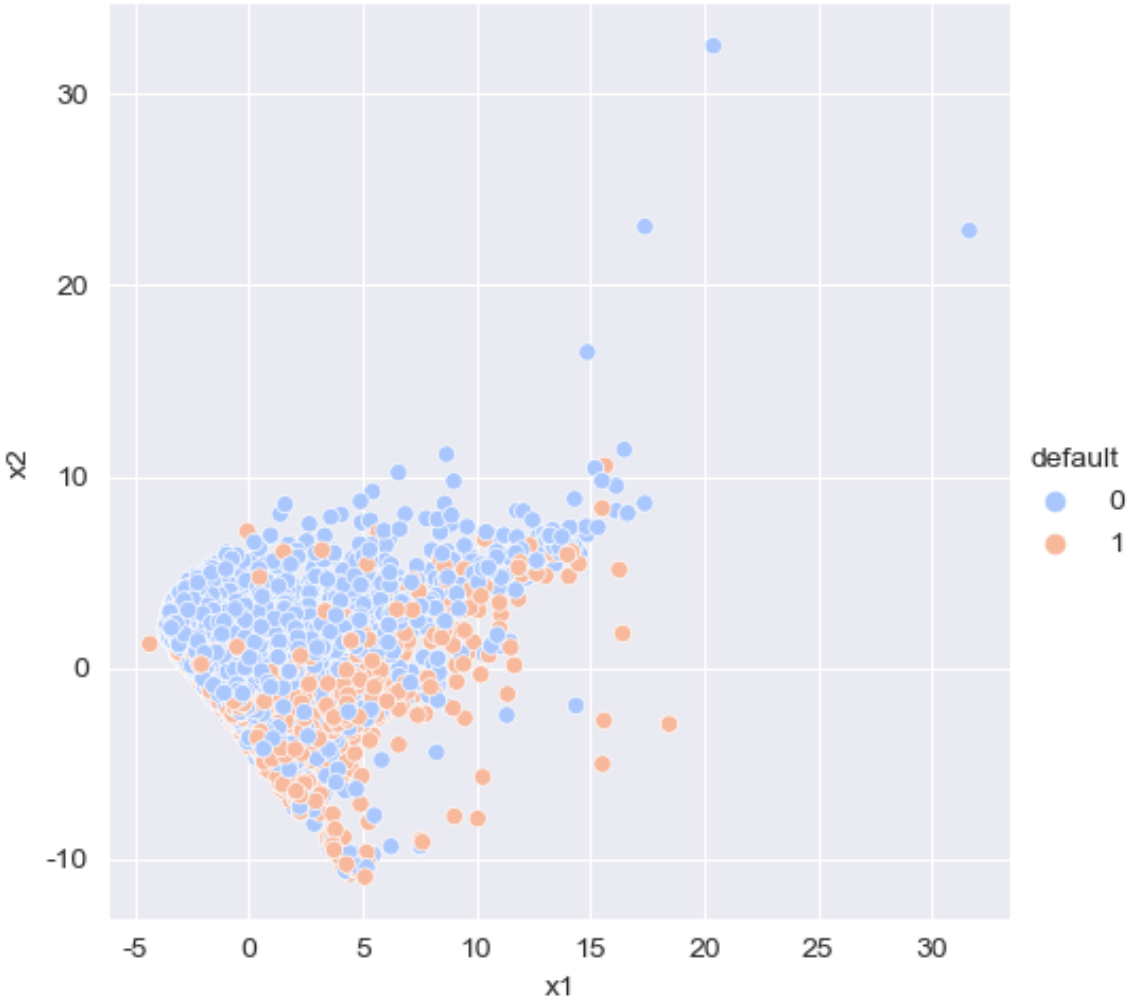
- default payment next month： 下个月是否违约（这是Ground Truth字段，1-违约，0-不违约）

从上面的分析可以看出SEX，EDUCATION和MARRIAGE是类别型（categorical）特征，其它是数值型特征。下面载入数据并将类别型特征处理成 one-hot 形式

```
import pandas as pd

data = pd.read_csv('credit_card_default.csv', sep=',')
del self.data['ID'] # 删除 ID 列

# 利用Pandas的 get_dummies 方法把类别型字特征转换成 One-hot 形式
# 其它字段类似处理，处理完成后删除原特征
gender = pd.get_dummies(self.data.SEX, prefix='gender')
del self.data['SEX']

self.data.loc[(self.data.EDUCATION==0) | (self.data.EDUCATION==5) | \
              (self.data.EDUCATION==6), 'EDUCATION'] = 4
education = pd.get_dummies(self.data.EDUCATION, prefix='education')
del self.data['EDUCATION']

self.data.loc[(self.data.MARRIAGE==0), 'MARRIAGE'] = 3
marriage = pd.get_dummies(self.data.MARRIAGE, prefix='marriage')
del self.data['MARRIAGE']

# 把处理过的数据与原数据合并
self.data = pd.concat([gender, education, marriage, self.data], axis='columns')
```

训练集、验证集、测试集按 7:1:2 划分，划分方法及特征标准化与上一专题类似，这里不再赘述。这里谈一下对数据集的理解。从常识来看，额度、性别、教育程度、婚姻和年龄等反映的是一个人的收入水平，暗示着获得现金流的能力；账单反映了负债情况，联系着还贷压力；还款记录和数额反映了现金流出的情况和过往信用。抓住了还贷能力，负债和还贷记录这三个要点就能够大致预判违约情况了。那么数据分析支不支持这个假设呢？本文在模型测试阶段解答这个问题。

将训练集数据用PCA方法将至2维，可视化如下，可见数据具有成簇的特性，但正反例交错的情况也存在，这给分类造成较大影响。另外本数据集是非平衡数据集，正、反例各占22%和78%，即使全部判定为”不违约“也有78%的准确率，因此本文将介绍其它衡量分类器性能的指标。



模型

逻辑回归（Logistic Regression）中文翻译里虽然有逻辑字眼，但它实际是二分类模型。本文要预测在下个月是否会发生违约，这就是一个二分类问题（非此即彼）。

逻辑回归的模型如下所示，x_{ij} 是样本 x_i 的各特征，共 n 个特征。这些特征乘以权重 w_j 并加上偏置 b 后得到 z_i，最后在 z_i 上施加 Sigmoid 非线性函数得到预测值 y_i。

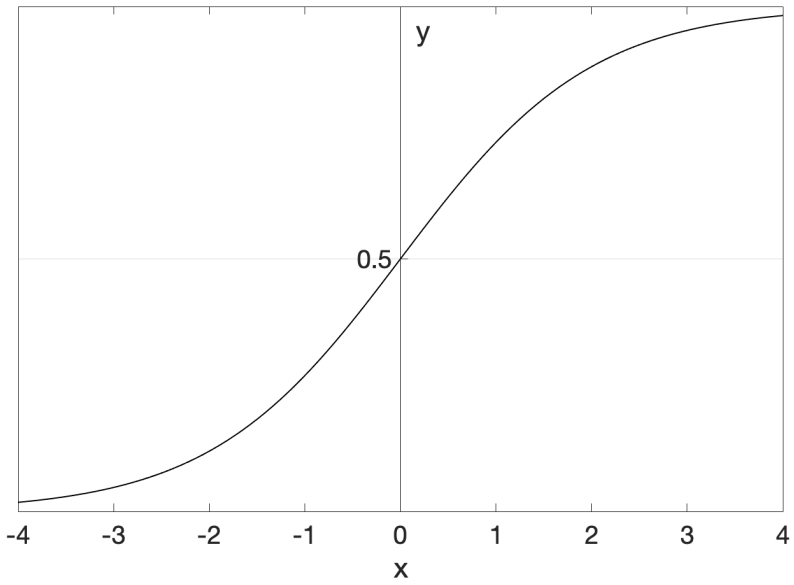
$$y_i = \frac{1}{e^{-z_i}+1}$$

$$\text{其中} z_i = \sum_{j=1}^{n} x_{ij}w_j + b$$

Sigmoid函数的表达式是

$$y = \frac{1}{e^{-x}+1}$$

其函数图像如下，可见该函数将 $(-\infty, \infty)$ 范围的数映射到 $(0,1)$，当 $x>0$ 时 $y>0.5$，如果正反例的判定阈值设为0.5的话，此时认为预测的类别为正,即 $\sum_{j=1}^{n} x_{ij}w_j + b > 0$ 时，样本 x_i 将被预测为正例。Sigmoid 函数的一个重要性质是 $y' = y(1-y)$。



由于 $y_i \in (0,1)$，所以将 y_i 视为 $P(\hat{y}_i = 1|x_i, \mathbf{w}, b)$，即给定 x_i 和模型参数时 $\hat{y}_i = 1$ 的概率，$\hat{y}_i$ 为样本的实际标签，在 $\{0,1\}$ 中取值。

损失函数

在上一节中 $P(\hat{y}_i = 1|x_i, \mathbf{w}, b) = y_i$，同理 $P(\hat{y}_i = 0|x_i, \mathbf{w}, b) = 1 - y_i$，以下技巧将两式合成一式

$$P(\hat{y}_i|x_i, \mathbf{w}, b) = y_i^{\hat{y}_i}(1 - y_i)^{1-\hat{y}_i}$$

如果将所有样本看做服从伯努利二项分布的独立同分布随机变量，则样本的似然（likelihood）定义为

$$L = \prod_{i=1}^{m} P(\hat{y}_i|x_i, \mathbf{w}, b)$$

机器学习的目标是求参数 $\mathbf{w}^*, b^*$ 使其满足：

$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w},b}{\operatorname{argmax}}(L)$$

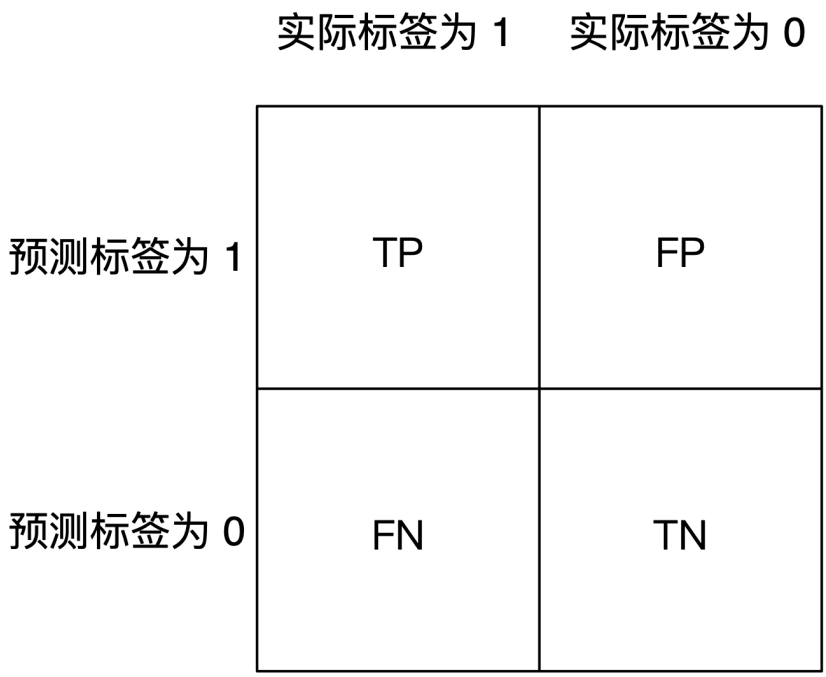
为了用梯度下降法求解该似然函数最大化问题，一般变通为求解负对数似然的最小化问题，负对数似然是：

$$\begin{aligned} J &= -\sum_{i=1}^{m} log[P(\hat{y}_i|x_i, \mathbf{w}, b)] \\ &= -\sum_{i=1}^{m} log[y_i^{\hat{y}_i}(1 - y_i)^{1-\hat{y}_i}] \\ &= -\sum_{i=1}^{m} [\hat{y}_i log y_i + (1 - y_i) log(1 - \hat{y}_i)] \end{aligned}$$

本文以上述负对数似然为损失函数。之所以要把似然转化成对数似然，是为了把连乘式转变成连加式，且对数运算不会改变最值的位置。该损失函数是凸函数，存在全局最小值。该损失函数与交叉熵（cross entropy）损失函数形式上一致，因此在 PyTorch 中引用这个损失函数用的是 `from torch.nn import BCEWithLogitsLoss`

评价指标

逻辑回归模型是一个二分类模型，衡量分类器的性能首先要构造混淆矩阵（confusion matrix），如下



这是一个 2×2 表格，左上角填真阳性（True Positive, TP）结果，即实际标签为1预测标签也是1的样本数；右上角填假阳性（False Positive, FP）结果，即实际标签为0但预测标签为1的样本数；左下角是假阴性（False Negative, FN）结果，即实际标签为1但预测标签为0的样本数；右下角是真阴性（True Negative, TN）结果，即实际标签是0预测也是0的样本数。

由于预测值是一个介于 (0,1) 的实数，为了将其转换成 0 或 1，需要与一个**阈值**相比，比如阈值为 0.5 时，如果预测值大于 0.5 则认为预测的是 '1'，反之预测值小于 0.5 时认为预测的是'0'，代码类似下面这样

```
predict_label = (prediction > threshold).type(th.float32)
```

注意，阈值设定为 0.5 并不是最优的，下面将讨论如何设置阈值。

计算出混淆矩阵里各个数值后，就可以计算准确率（Accuracy）、查准率（Precision）、查全率（Recall）和F1值（F1 Score）这些指标了

$$
\begin{aligned}
Accuracy &= \frac{TP+TN}{TP+FP+FN+TN} \\
Precision &= \frac{TP}{TP+FP} \\
Recall &= \frac{TP}{TP+FN} \\
F_1 &= \frac{2PR}{P+R}
\end{aligned}
$$

F1 值中 P 指 Precision，R 指 Recall。

上述结果是在特定判定阈值下计算出来的，为了全面评价分类器性能需要另外的指标。观察分类器性能常常借助ROC（Receiver Operating Characteristic）曲线，ROC曲线横轴是 FPR（False Positive Rate），纵轴是TPR（True Positive Rate），定义如下

$$FPR = \frac{FP}{FP + TN}$$
$$TPR = \frac{TP}{TP + FN}$$

为了得到一组 FPR 和 TPR，先把预测值按从大到小排列，在最后补充一个0值。然后依次以排序后的预测值为判定阈值，计算混淆矩阵里各个数值进而计算 TPR 和 FPR。关键代码如下所示

```
import sys

# 排序
idx = th.argsort(pred, descending=True)
pred = pred[idx]
label = label[idx]

# 以预测值为判定阈值，补上0
thresholds = pred.tolist()
thresholds.append(0)

tpr = []
fpr = []

for thr in thresholds:
    TP, FP, TN, FN = confusion_matrix(pred, label, thr)
    tpr.append(TP / (TP + FN + sys.float_info.epsilon))
    fpr.append(FP / (TN + FP + sys.float_info.epsilon))
```

为了用一个标量定量描述分类器性能可以计算 ROC曲线下面积，该值记为 **AUC** （Area Under ROC Curve）,本文利用 scikit learn 里的工具计算该值

```
from sklearn import metrics

auc = metrics.auc(fpr, tpr)
```

最后介绍怎样获得最佳判定阈值，本文利用 G-Mean（geometric mean），G-Mean定义为

$$gmean = \sqrt{TPR(1 - FPR)}$$

让 G-Mean 取得最大值的阈值是最佳判定阈值。

优化

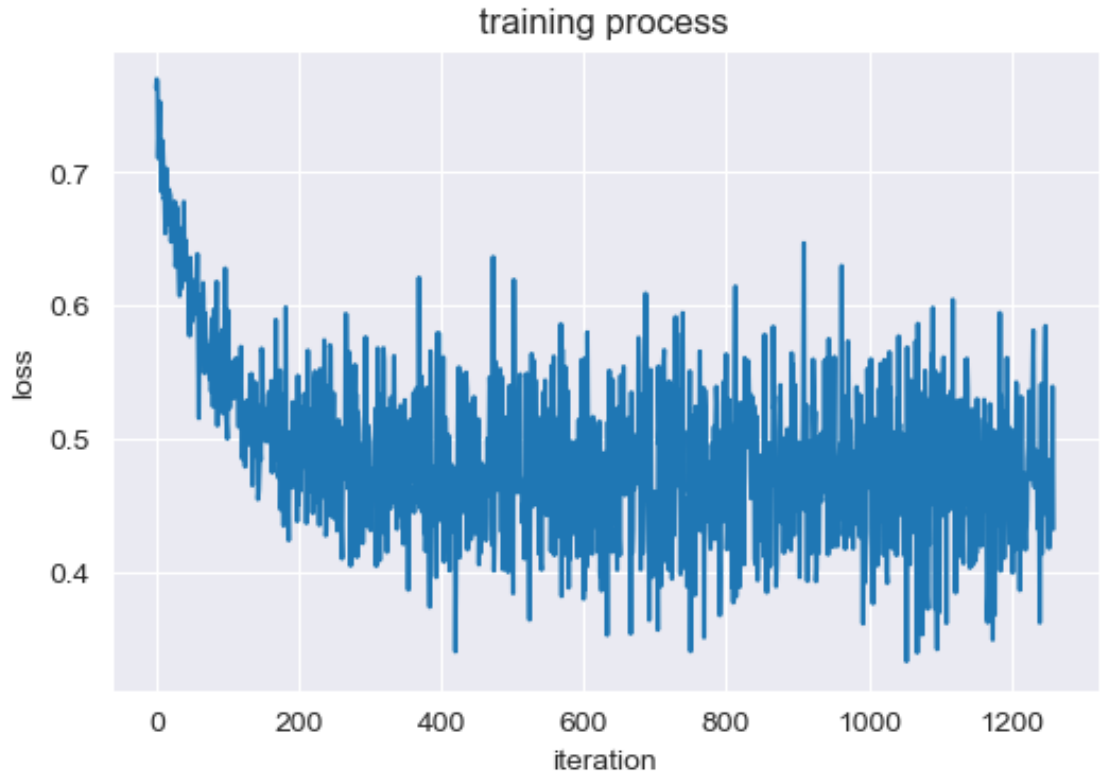
优化器的代码结构与上一专题的类似，如下所示。主要由这样几个部分组成：首先是初始化若干容器，以便保存训练中产生的中间结果；然后是设定 dataloader 和 优化器；接着是一个循环结构，在此循环中有前馈（forward）、回馈（backward）和更新（update）三个基础步骤，其间在验证集上计算损失或某些指标，从而检验训练是否有效以及观察哪种超参数对训练有益。

以下代码有两处实现技巧，第一是在 `with th.no_grad()` 语境下做模型推断（inference），当不需要计算梯度的时候这样做能够减少计算量；第二是在每一轮迭代完成后缩减学习率，这是一种最简单的学习率退火机制。

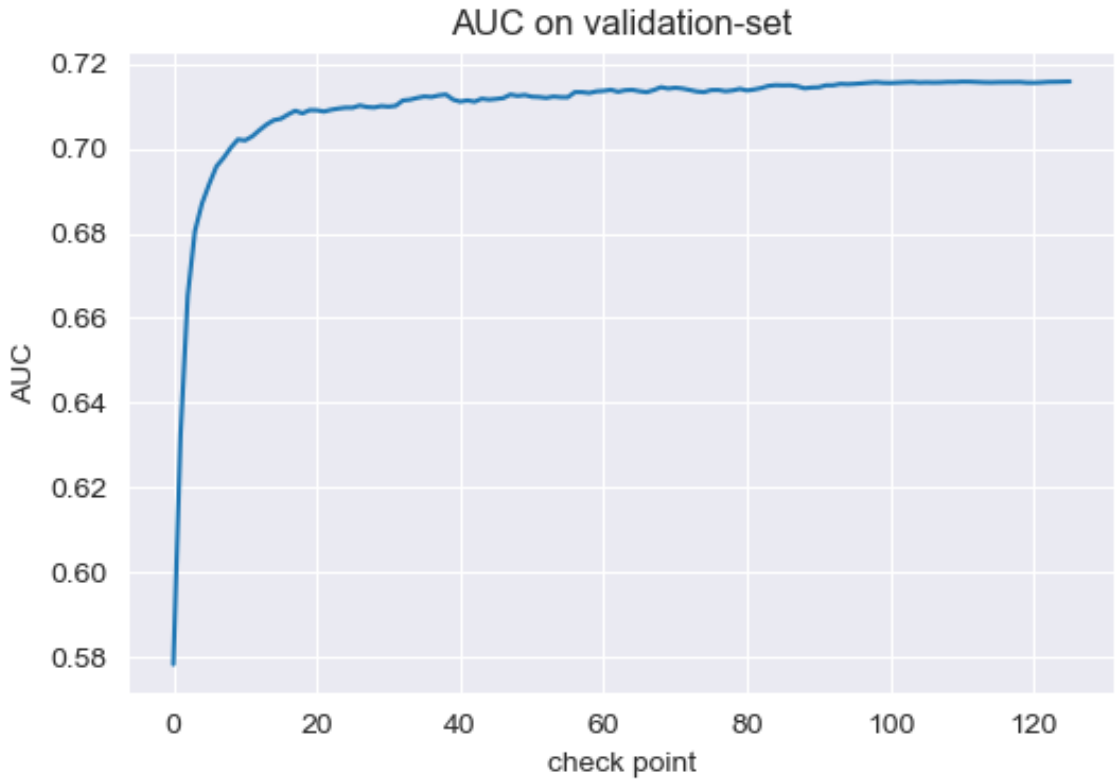
```
def trainer(trainset, valset, model, config, save_snapshot=False):
    trainloss = []
    auc_list  = []
    snapshots = [] # 保存模型快照

    train_loader = DataLoader(trainset, batch_size=config['batchsize'], shuffle=True)
    val_feature, val_label = valset[:]
    optimizer = optim.SGD(model.parameters(), lr=config['lr']) # 设定优化器

    for epoch in range(config['epoches']):

        iteration = 0
        for feature, label in train_loader:
            # forward step
            prediction = model(feature)
            loss = bce(prediction, label) # 计算损失

            trainloss.append(loss.item())

            iteration += 1
            if iteration % 10 == 0:
                # 当不需要计算梯度的时候用下面方法做模型推断
                with th.no_grad():
                    val_prediction = th.sigmoid(model(val_feature))

                accuracy, precision, recall, F1 = classification_metric(\
                    val_prediction.flatten(), val_label.flatten(), 0.5)

                auc, _,_,_ = roc_metric(val_prediction.detach().flatten(), \
                                        val_label.flatten())
                auc_list.append(auc)
                print(f'epoch: {epoch} iteration: {iteration:3d}, auc: {auc:.3f} \
                        accuracy: {accuracy:.3f}, precision: {precision: .3f}, \
                        recall: {recall:.3f}, F1: {F1:.2f}')

            # backward step
            optimizer.zero_grad()
            loss.backward()

            # update parameters
            optimizer.step()

            # save snapshot
            if save_snapshot:
                snapshots.append(copy.deepcopy(model.state_dict()))

        # 学习率每一轮递减
        for g in optimizer.param_groups:
            g['lr'] *= 0.7
```

```
    return trainloss, auc_list, snapshots
```

经过训练，模型训练损失如下（在随机梯度优化方法下，训练损失非单调下降，但总体是下降趋势）



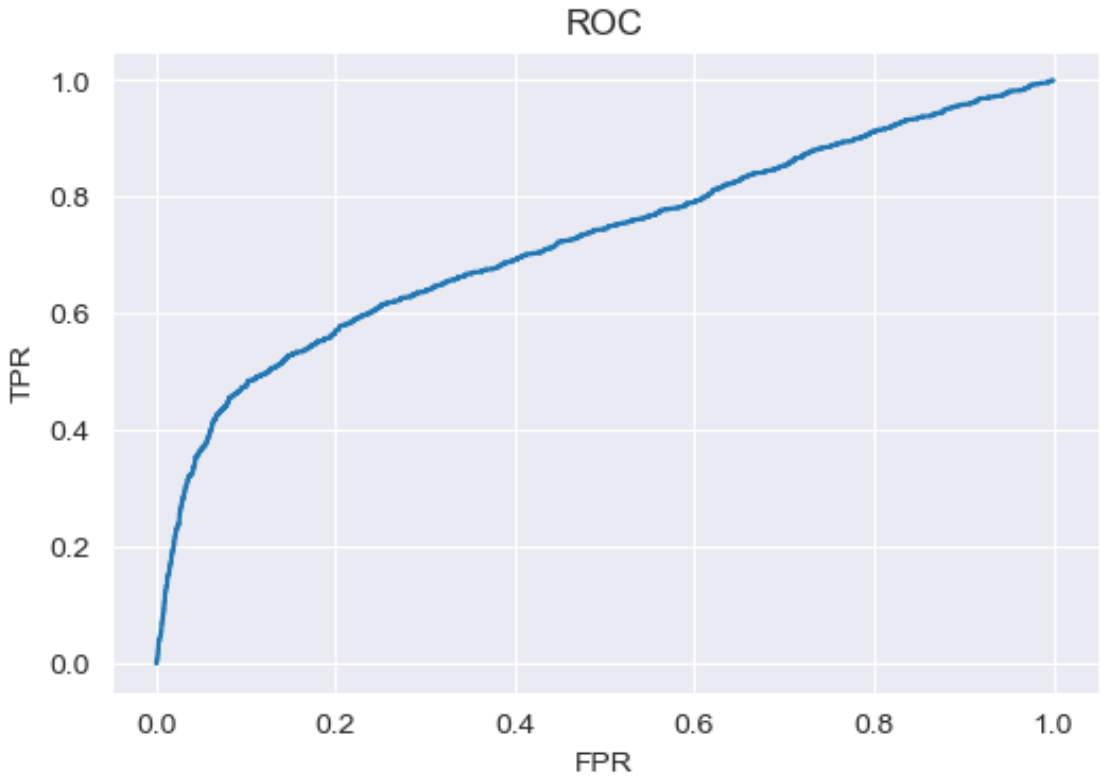
验证集上的 AUC 如下



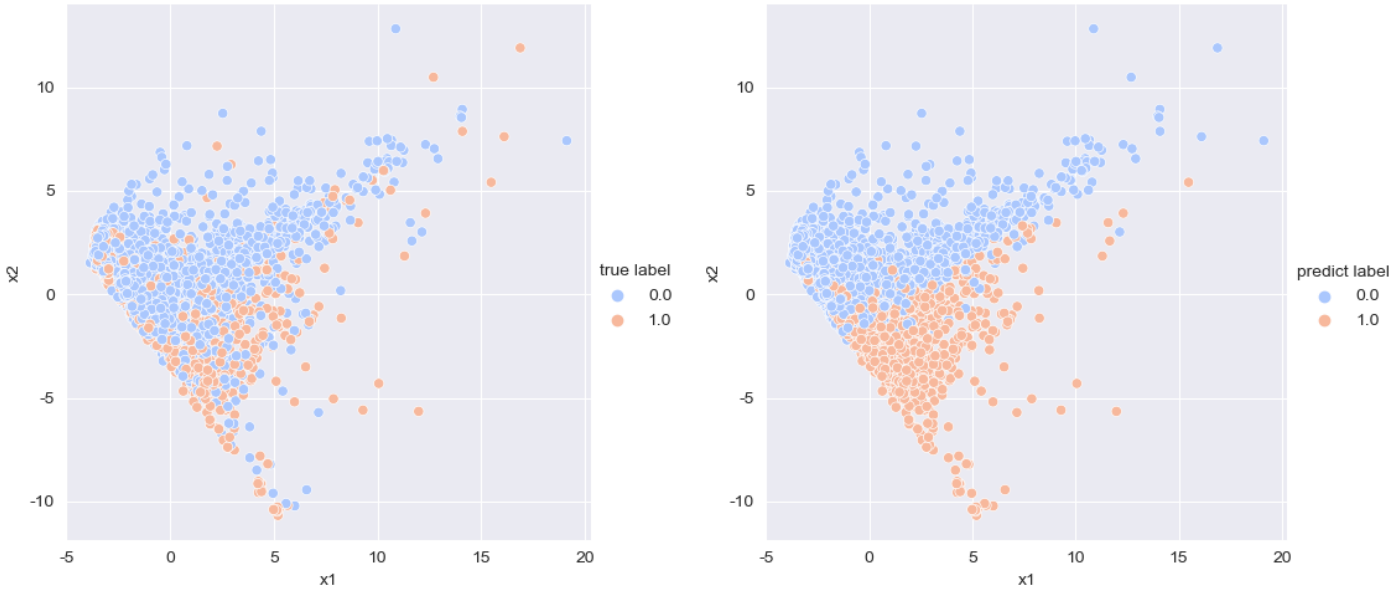
从两个曲线的形态看，模型的训练是有效的。

测试

测试集上的 ROC 如图所示



AUC是 0.72，用 G-Mean方法计算出的最佳分类阈值是 0.25，在该阈值下分类精确率是 0.72，查准率是 0.40， 查全率是 0.62， F1值是 0.49。把测试集数据可视化如下所示



左图和右图分别是实际标签和预测标签的可视化结果。从分类器性能指标和可视化结果来看模型的分类性能是差强人意的，由于逻辑回归模型仍然是一个线性模型，所以在面对左图所示的数据分布时是难以胜任分类任务的。

观察模型权重，如下所示。性别方面：男性较女性更易于违约；教育程度方面：研究生学历较其他学历更易违约；婚姻方面：已婚的较其它人群更易违约；额度方面：额度小的更易违约；PAY_0 和 PAY_2 权重较大，说明较近两个月如果延期了，下个月更易于违约；BILL_AMT1 较大且是负值，说明当月账单越小下个月更易违约；PAY_AMT1 ~ PAY_AMT6 均是负数，说明之前偿还的越多下个月越不易于违约。

分析权重可以印证一些猜想，一些结论也常常有违常识，比如已婚的人士按理说财务状况应该较未婚时更稳定，但从权重看已婚的更易于违约，实际是否如此以及该结果背后的原因需要进一步调查研究。

```
gender_1        0.028
gender_2        -0.014
education_1     0.024
education_2     -0.022
education_3     -0.035
education_4     -0.097
marriage_1      0.068
marriage_2      -0.004
marriage_3      -0.010
LIMIT_BAL       -0.085
AGE             0.067
PAY_0           0.508
PAY_2           0.170
PAY_3           0.068
PAY_4           0.049
PAY_5           0.037
PAY_6           0.039
BILL_AMT1       -0.122
BILL_AMT2       0.085
BILL_AMT3       0.032
BILL_AMT4       -0.047
BILL_AMT5       -0.069
BILL_AMT6       -0.015
PAY_AMT1        -0.119
PAY_AMT2        -0.088
PAY_AMT3        -0.019
PAY_AMT4        -0.038
PAY_AMT5        -0.040
PAY_AMT6        -0.079
```

小结

本文数据集是一个非平衡数据集，正反例样本的数量较为悬殊，这类问题的一个处理办法是使用平衡交叉熵或者 **Focal Loss**（https://arxiv.org/abs/1708.02002）。

在机器学习算法的编程实现上，Dataset 类、训练器和评价指标是花费精力较多的部分。对于创新来说模型和损失函数是发挥创造性的地方。

思考

在理解信用卡系统运行原理的基础上，能否在特征工程（feature engineering）上做文章能够提高数据的线性可分性?