基于RNN的行为评分卡模型实战

原创 梅子行 大数据风控与机器学习 2019-11-07

背景：因风控场景下有大量的具有时间先后顺序的数据。近期使用的较为广泛的一种建模方式，就是先使用一个针对时序数据进行建模的模型，输出时序预测分数，然后将分数应用于评分卡中作为一个特征。

应用场景：主要应用于时序数据较多的 B 卡中。

数据：复贷超过3次的B卡客户，以月份为切片构造基础特征向量，历史数据超过2年。

数据扩充：由于负样本不足5万。为得到更好的模型效果，使用pd5代替M1+。将观察期介于2~3年的客户首尾相连，进行随机截断，以获取更多的24维向量客户。再将观察期介于1~2年之间的客群首尾相连，通过prophet算法前向延伸，作为辅助训练样本。

选用模型：2层LSTM模型（示例，目前最深不超过6层）

深度学习框架：pytorch

使用方法：特征带入 LSTM 中输出 score 或隐层参数，分别作为单维度特征，一同带入逻辑回归中进行建模。

环境加载。

```
import torch
import torch.nn as nn
import random
from sklearn.model_selection import train_test_split
import torchvision.transforms as transforms
import torchvision.datasets as dsets
from torch.autograd import Variable
```

数据加载。

```
random_st = random.choice(range(10000))
trainB, testB = train_test_split(trainB,test_size=0.25,
                                 random_state=random_st)

train_data = MyDataset(trainB)
test_data = MyDataset(testB)

train_loader = torch.utils.data.DataLoader(train_data, batch_size=50,
                                           shuffle=True, num_workers=0)
test_loader = torch.utils.data.DataLoader(test_data, batch_size=25,
                                          shuffle=False, num_workers=0)
```

定义网络。

```
#搭建LSTM网络
class Rnn(nn.Module):
    def __init__(self, in_dim, hidden_dim, n_layer, n_class):
        super(Rnn, self).__init__()
        self.n_layer = n_layer
        self.hidden_dim = hidden_dim
        self.LSTM = nn.LSTM(in_dim, hidden_dim,
                                    n_layer,batch_first=True)
        self.linear = nn.Linear(hidden_dim,n_class)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
```

```
        x = x.sum(dim = 1)
        out, _ = self.LSTM(x)
        out = out[:, -1, :]
        out = self.linear(out)
        out = self.sigmoid(out)
        return out
```

指定参数。

```
#指定网络参数。

#28个特征，42个月切片，2个隐层，2分类
model = Rnn(28,42,2,2)
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)

#使用二分类对数损失函数
criterion = nn.SoftMarginLoss(reduction='mean')
opt = torch.optim.Adam(model.parameters())
total_step = len(train_loader)
total_step_test = len(test_loader)
num_epochs = 50
```

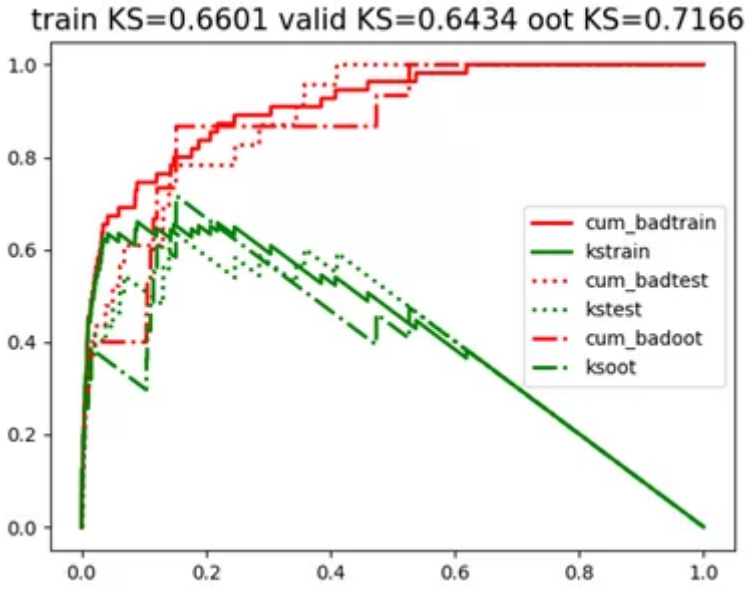
训练网络。

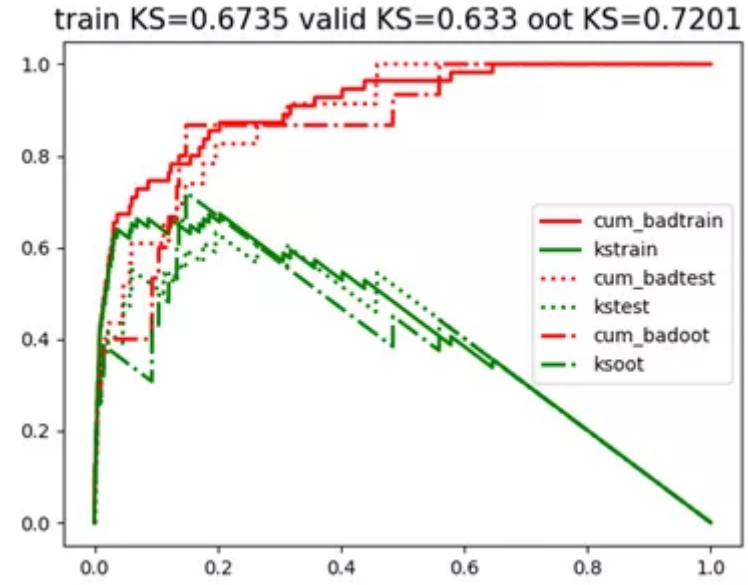
```
#指定网络参数。

#28个特征，42个月切片，2个隐层，2分类
model = Rnn(28,42,2,2)
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)

#使用二分类对数损失函数
criterion = nn.SoftMarginLoss(reduction='mean')
opt = torch.optim.Adam(model.parameters())
total_step = len(train_loader)
total_step_test = len(test_loader)
num_epochs = 50
```

```
训练网络。

#训练得到LSTM模型并计算单模型的KS值和AUC值。
for epoch in range(num_epochs):
    train_label = []
    train_pred = []
    model.train()
    for i, (B_features, labels) in enumerate(train_loader):
        B_features = B_features.to(device)
        labels = labels.to(device)
        #网络训练
        out = model(B_features)
        loss = criterion(out, labels)
        opt.zero_grad()
        loss.backward()
        opt.step()
        #每一百轮打印一次
        if i%100 == 0:
            print('train epoch: {}/{}, round: {}/{},
                                loss: {}'.format(epoch + 1, num_epochs,
                                                 i + 1, total_step, loss))
        #真实标记和预测值
        train_label.extend(labels.cpu().numpy().flatten().tolist())
        train_pred.extend(out.detach().cpu().numpy().flatten().tolist())
    #计算真正率和假正率
    fpr_lm_train, tpr_lm_train, _ = roc_curve(np.array(train_label),
                                              np.array(train_pred))
    #计算KS和AUC
    print('train epoch: {}/{}, KS: {}, ROC: {}'.format(
        epoch + 1, num_epochs,abs(fpr_lm_train - tpr_lm_train).max(),
               metrics.auc(fpr_lm_train, tpr_lm_train)))

    test_label = []
    test_pred = []

    model.eval()
    #计算测试集上的KS值和AUC值
    for i, (B_features, labels) in enumerate(test_loader):

        B_features = B_features.to(device)
```

```
        labels = labels.to(device)
        out = model(B_features)
        loss = criterion(out, labels)

        #计算KS和AUC
        if i%100 == 0:
            print('test epoch: {}/{}, round: {}/{},
                                loss: {}'.format(epoch + 1, num_epochs,
                                i + 1, total_step_test, loss))
        test_label.extend(labels.cpu().numpy().flatten().tolist())
        test_pred.extend(out.detach().cpu().numpy().flatten().tolist())

    fpr_lm_test, tpr_lm_test, _ = roc_curve(np.array(test_label),
                                                    np.array(test_pred))

    print('test epoch: {}/{}, KS: {}, ROC: {}'.format(
                                                        epoch + 1, num_epoch
                                                          abs(fpr_lm_test -
```

最终模型结果。





阅读原文