CSDM-NJUSME

南京大学工程管理学院2020年春

研究生数据挖掘课程作业2

一、作业说明

以第一次作业清洗后的数据构建SVM，并求解，计算查全率、查准率和KS系数。

用Python 3实现，IDE是PyCharm，展示代码运行结果和分析的部分使用PyCharm内嵌的Jupyter Notebook。

二、文件说明：

1. 文件结构

```
$ tree
.
├── README.md（文件说明和运行说明）
├── data
│   └── preprocess.csv（第一次作业生成的数据文件，包括样本特征和标签）
├── metrics
│   └── metrics.py（计算查全率、查准率、ks得分，绘制ks曲线）
├── model
│   ├── model.ipynb（自己实现版本，比较了有无核函数、不同核函数、不同惩罚系数下的模型效果）
│   └── model_sklearn.ipynb（sklearn调包版本，比较了不同阈值和不同核函数下的模型效果）
├── model.pdf（model.ipynb生成的pdf文件，方便查看）
├── model_sklearn.pdf（model_sklearn.ipynb生成的pdf文件，方便查看）
└── smo
    ├── smo_platt_no_kernel.py（没有核函数的完整smo实现）
    ├── smo_platt_with_kernel.py（加了核函数的完整smo实现）
    └── smo_simple.py（简化版本的smo实现）
```

2. 算法实现

我完成的：带软间隔（拉格朗日松弛）、核函数、对偶处理后的支持向量机学习算法。

原理是根据老师ppt和[1]的7.4节，实现的具体细节和优化参考了[2]的第六章。

- `smo_simple.py`和`smo_platt_no_kernel.py`用了对偶、拉格朗日松弛，不带核函数。
  区别在于前者在具体实现时SMO部分的子问题的外层循环α_1选取方式做了简化。

求解的对偶问题如下，用SMO算法启发式地求出变量α的最优解α^*。

$$
\begin{aligned}
\min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left(x_i \cdot x_j\right) - \sum_{i=1}^{N} \alpha_i \\
\text{s.t.} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& 0 \leqslant \alpha_i \leqslant C, \quad i = 1, 2, \cdots, N
\end{aligned}
$$

然后求出原问题的最优解w^*和b^*，然后预测样本x标签为$f(x) = \text{sign}(w^* \cdot x + b^*)$。

- `smo_platt_with_kernel.py`用了对偶、拉格朗日松弛，带核函数。

求解的对偶问题如下，用SMO算法启发式地求出对偶问题的最优解w^*，然后求出原问题的最优解b^*，然后预测样本x标签为$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i^* y_i K\left(x, x_i\right) + b^*\right)$。

$$
\begin{aligned}
\min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K\left(x_i, x_j\right) - \sum_{i=1}^{N} \alpha_i \\
\text{s.t.} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& 0 \leqslant \alpha_i \leqslant C, \quad i = 1, 2, \cdots, N
\end{aligned}
$$

3. 指标计算

- `metrics.py`

  根据算法求出的预测标签和真实标签计算TP、TN、FP、FN，然后求出recall和precision。

  求KS要麻烦些，先按照预测的概率值从大到小排列，然后自己根据判定比例来假设预测标签，比如判定比例为1/1412时，假设的预测标签就只有预测概率值最大的样本是违约，其他样本都是不违约。接着根据假设的预测标签和真实标签计算TP、TN、FP、FN，然后求出真正例率、假正例率和二者差（ks值），这样不断假设预测标签，得到每个判定比例下的真正例率、假正例率和二者差（ks值），然后就可以绘制ks曲线，并得到最大ks值和其对应的判定比例了。

4. 保存预测结果

保存了三个结果最好模型的每个样本的预测违约概率和预测标签。

在信用评分，我们认为应该查全率最重要，所以将查全率最高的对应预测结果进行保存，rbf、阈值0.4。

5. 入口文件

- `model.ipynb` 对自己实现版本的SMO算法进行比较试验，指标是recall、precision、ks和运行时间。目的是查看核函数的效果，以及rbf内核的smo在惩罚系数取0.6、6、60时的表现。
- `model_sklearn.ipynb`

  对调包版本进行比较试验，指标和上面相同，主要是看linear核和rbf核的不同效果，以及分类时对概率值设置的阈值的效果。总的结论：rbf核在这个数据集上分类效果好；rbf核下，低阈值的recall值更高，高阈值的precision值更高。

- 对比自己实现的和sklearn提供的，发现运行时间差距巨大，人家用的LIBSVM是用C语言高效实现的。得分效果上，自己实现的rbf核版本调参后的指标得分和sklearn差不多。

	recall	**precision**	**ks**	**time(s)**
sklearn	**0.80**	0.95	0.78	**0.23**
Ours	0.79	**0.98**	**0.82**	226.52

6. 其他

- 没画图的原因：我们的变量经过整理分类离散化后，每个变量的取值小于等于5，二维图画散点图最多也就5*5个点，其上分布着1400个样本点，大量重合。我尝试绘制了，发现看不出什么规律，效果不好。
- 整体过程：先用jupyter方便查看每个函数效果是否正确，或者直接写py文件，把确定正确的函数功能封装起来。最后的model不方便调试，只用来查看对比图和变量的，是最后一步展示。

三、文件执行说明：

1. 输入下列指令创建环境`ml`，同时也安装好了依赖包

```
conda create --name ml python=3 scikit-learn pandas numpy matplotlib
```

2. 进入环境`ml`

```
conda activate ml
```

3. 从根目录进入`/model`文件夹，分别执行`model.ipynb`和`model_sklearn.ipynb`文件，每个文件都要从上到下逐个单元格执行，即可复现已有的运行结果。

   如果不需要复现，可直接查看根目录下的`model.pdf`和`model_sklearn.pdf`，里面有代码运行结果和辅助的分析。

参考文献：

[1] 李航, 统计学习方法. 清华大学出版社, 2012.
[2] P. Harrington, 机器学习实战. 人民邮电出版社, 2013.