支持向量机实验报告

任俊彦　520021910546
电子信息与电气工程学院 自动化

目录

1 问题描述

1.1 问题设定

本次作业回归初心，重新拾起红葡萄酒数据集，设定任务为使用提供的酒品各物理化学成分指标建立软间隔支持向量机模型预测其酒品质量优劣。

1.2 问题分析

由于给定数据集的酒品质量为 $0-10$ 评分量化，故对于没有采用如 $one-against-one$ 等多分类策略的 SVM，需要先将评分二值化。对于该数据集，二值化有若干种方式，一种方式是将评分不小于最高分 60% 的酒品定义为“合格酒品”，将其它定义为低质量酒品。这种二值方式是否合理，需要观察已知数据的评分分布，该数据集酒品评分分布如下：

评分	0	1	2	3	4	5	6	7	8	9	10
样本数	0	0	0	10	53	681	638	199	18	0	0

对数据集酒品质量评分分布进行分析后，发现小于 6 分的有 744 个酒品样本，大于等于 6 分的有 855 个样本，故这种二值化方式在数据分布角度上是合理的。在求解问题前，将数据进行归一化并将酒品按 6 的品质阈值归类成 ± 1

2 数据清洗

在对数据代入模型求解前，需要处理空数据并剔除不合理数据。经分析，红酒数据集中不存在空数据且数据数值区间均合理。

3 算法策略

3.1 软间隔 SVM 的求解

SVM 的求解有多种方法，本次作业使用 SMO 算法求解其对偶问题。设数据集 $\{(x_i, y_i)\}_{i=1}^m$，K 为核矩阵，C 为惩罚系数，软间隔 SVM 对偶问题为

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_i\sum_j y_i\alpha_i K_{ij} - \sum_i \alpha_i \\ \text{s.t.} \quad & \sum_i y_i\alpha_i = 0 \\ & 0 \le \alpha_i \le C, \forall i = 1, 2, \dots m. \end{aligned}$$

SMO 算法采用每次单独优化两个对偶变量并迭代的策略求解对偶问题，每次求解为一个带约束二次规划问题，依据 KKT 条件进行推导，每次迭代优化策略如下：

$$\eta = K_{ii} - 2K_{ij} + K_{jj}$$

$$E_i = g(x_i) - y_i = \left(\sum_{j=1}^m \alpha_j y_j K(x_i, x_j) + b\right) - y_i$$

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} + y_j \frac{(E_i - E_j)}{\eta}$$

上述求得的 α_j^{new} 未考虑约束 $0 \leq \alpha_i \leq C$，为使其满足该约束，记

$$L = \begin{cases} \max\left(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}}\right), & y_i \neq y_j \\ \max\left(0, \alpha_j^{\text{old}} + \alpha_i^{\text{old}} - C\right), & y_i = y_j \end{cases} \quad H = \begin{cases} \min\left(C, C + \alpha_j^{\text{old}} - \alpha_i^{\text{old}}\right), & y_i \neq y_j \\ \min\left(C, \alpha_j^{\text{old}} + \alpha_i^{\text{old}}\right), & y_i = y_j \end{cases}$$

采用如下策略更新 α_j^{new}

$$\alpha_j^{\text{new}} = \begin{cases} H, & \alpha_j^{\text{new}} > H \\ \alpha_j^{\text{new}}, & L \leq \alpha_j^{\text{new}} \leq H \\ L, & \alpha_j^{\text{new}} < L \end{cases}$$

引入变量

$$\begin{aligned} b_i^{\text{new}} &= b^{\text{old}} - E_i - y_i K_{ii}\left(\alpha_i^{\text{new}} - \alpha_i^{\text{old}}\right) - y_j K_{ij}\left(\alpha_j^{\text{new,}} - \alpha_j^{\text{old}}\right) \\ b_j^{\text{new}} &= b^{\text{old}} - E_j - y_i K_{ij}\left(\alpha_i^{\text{new}} - \alpha_i^{\text{old}}\right) - y_j K_{jj}\left(\alpha_j^{\text{new}} - \alpha_j^{\text{old}}\right) \end{aligned}$$

并更新 α_i 和 b

$$\begin{aligned} \alpha_i^{\text{new}} &= \alpha_i^{\text{old}} + y_i y_j\left(\alpha_j^{\text{old}} - \alpha_j^{\text{new}}\right) \\ b^{\text{new}} &= (b_i^{\text{new}} + b_j^{\text{new}})/2 \end{aligned}$$

完成更新后另随机选两个对偶变量进行更新，并不断迭代，直至收敛。本次作业收敛条件设为在一轮迭代（将对偶变量随机排序后从前至后按步长为 1 依次取一对对偶变量，直至队尾）后，各对偶变量对 (α_i, α_j) 均满足：

$$|\alpha_i^{\text{new}} - \alpha_i^{\text{old}}| + |\alpha_j^{\text{new}} - \alpha_j^{\text{old}}| < 10^{-4}$$

经测试，此迭代方式虽然没有在一轮中遍历所有对偶变量对，但其收敛结果与遍历方式基本一致。

3.2 核方法

使用核方法可将原数据映射至核空间，解决线性不可分的问题，提升分类性能。SVM 由于其对偶问题的特殊形式，实现核方法只需设计满足要求的内积矩阵。本次作业完成了多项式核函数及高斯核函数（RBF 核）的实现。当多项式次数为 1 时，为线性核函数，即未施核方法的情形。

3.2.1 核矩阵的计算

对于数据较多的数据集，计算核矩阵非常费时。对于多项式核矩阵，可以直接使用矩阵乘积计算，其效率较高；而对于 RBF 核，由于其形式难以写成矩阵运算，可能会考虑使用双循环来利用计算式计算矩阵中的每一个元素，但这将导致求解效率极低；即使考虑对称性只计算一半的元素，其求解效率亦极低，计算本次作业所用数据需用接近 10 秒的时间，这无疑是不可接受的。利用 numpy 中 array 的广播机制可快速简洁地计算 RBF 核，可在 0.5 秒内完成 RBF 核的计算，详见 *algorithm.SVM._Getkernel*.

3.2.2 核方法下的预测

使用核方法后，对于新数据，与训练数据重新计算核矩阵 $K^{'}$ 并由式

$$\hat{y_i} = sign\left(\sum_{j \in \mathrm{S}} \alpha_i y_i K_{ij}^{'} + b\right)$$

得到预测类别，其中 S 为支持向量索引集，即 $\mathrm{S} = \{j \mid \alpha_j \neq 0\}$.

3.3 五倍交叉验证

评估模型表现使用 K 倍交叉验证 (K-Fold Cross Validation)，其会得到 K 个模型，K 个模型分别在验证集中评估结果，最后的准确率取平均即得到交叉验证准确率，本次准确率设置为

$$acc = \frac{\sum_{i=1}^{m} I(\hat{y_i} = y_i)}{m}$$

交叉验证有效利用了有限的数据，并且评估结果能够尽可能接近模型在测试集上的表现，可以做为模型优化的指标使用。本次作业中 K 默认选 5，代码中可调，详见 *main.py*.

4 代码实现

4.1 代码结构

软间隔 SVM 的类实现在 *algorithm.py* 中，*main.py* 给出示例程序，可以直接运行得到本文中所有展示的结果，欢迎尝试。

4.2 类 API

SVM 类接口定义如下：

1. 创建实例：mySVM = SVM $(X, Y, C, ker =' poly', kpara = 1)$
2. 求解模型：mySVM.fit ()
3. 获得分类结果：mySVM.Inference $(data)$ -> array: class label {-1,1}

其中 ker 为核方法类型，$'poly'$ 为多项式核，此时 $kpara$ 为多项式次数，当 $kpara = 1$ 时为线性核；$ker =' gaussian'$ 时为高斯核，此时 $kpara$ 为高斯核的标准差。输入 $X.shape$ 应为 $(n\_component, n\_feature)$，Y 应为长度为 $n\_component$ 的一维数组。

关于类的具体实现，请参见 *algorithm.py*，文件里写有详细的注释，此处不赘述。

5 模型表现

5.1 核的选取

5.1.1 多项式核

将惩罚系数 C 设为 1，采用五倍交叉验证，分别测试多项式核与高斯核在该数据集上的表现。在 $1-6$ 间以 0.2 的步长测试多项式核次数对准确率的影响。

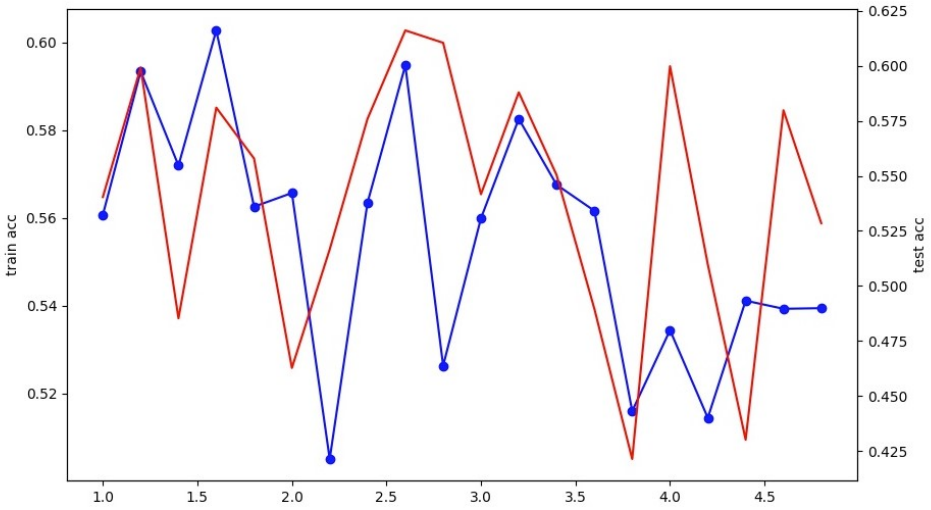

图 1: 不同次数多项式核与准确率的关系

其中蓝线为训练集，红线为测试集，横坐标为多项式核次数。可以发现，多项式核次数的改变与准确率并无明显关系，多项式核在该任务下的表现普遍不佳，最高准确率仅在 60% 左右。

5.1.2 RBF 核

选取核为高斯核，分别测试标准差为 $0.01, 0.1, 1, 5, 10, 20, 50$ 时高斯核在该任务上的表现，得到结果如下：

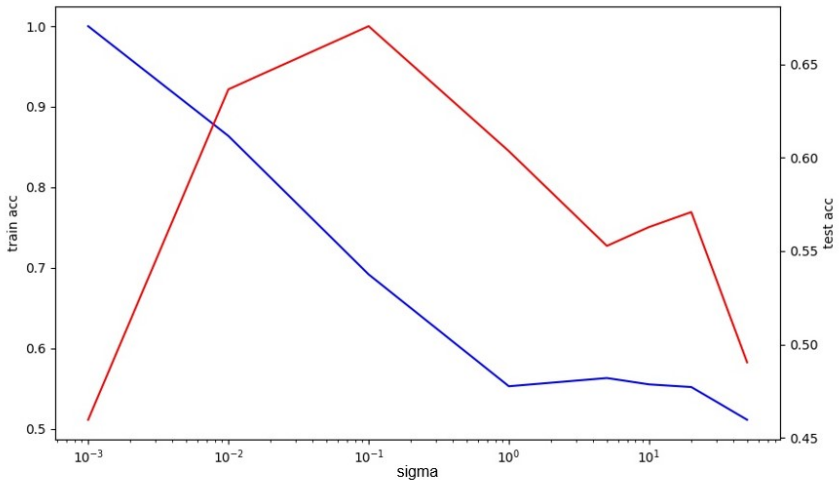

图 2: 不同标准差 RBF 核与准确率的关系

可以发现，当方差较小时，高斯核的拟合能力极强，在标准差为 0.001 时训练集准确率达到 100%，但导致了过拟合。随着标准差增大，训练集准确率降低，而验证集准确率提升，标准差为 0.01 时训练集及验证集准确率均较高，在进一步精确搜索后，在后续的实验中将高斯核的标准差设为 0.02.

5.2 惩罚系数 C 与准确率

选取核为标准差为 0.1 的高斯核，分别测试 C 为 $0.01, 0.1, 1, 5, 10, 20, 50$ 时的准确率，得到结果如下：

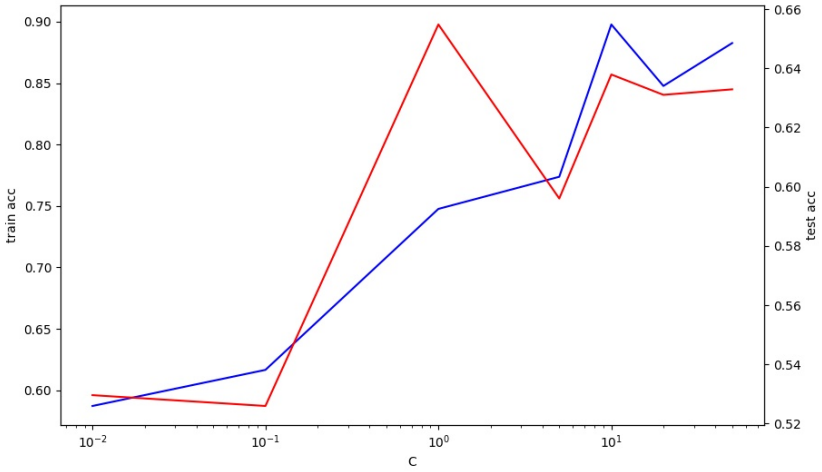

图 3: RBF 核下 C 与准确率的关系

可以发现，准确率总体随惩罚系数 C 的增大而增大，与该参数的意义相符。当 C 为 10 时，训练集测试集准确率均较高，故在模型最终求解中选取 $C = 10$.

5.3 求解结果

选取高斯 (RBF) 核，σ 设为 0.02，C 设定为 10，选取不同的高低质量酒品阈值 (5,6,7)，得到结果如下：

品质阈值	交叉验证准确率
5	0.94
6	0.65
7	0.85

多次测试后发现，当高低质量酒品阈值设定为 5 或 7 时，软间隔 SVM 在该数据集上能较好地对高低质量酒品进行划分，而当阈值设定为 6 时，多项式核与高斯核在该数据集上表现均不理想。这表明酒品质量评分为 5 和 6 的两类酒品较难区分，若欲取得更好的划分效果，应当考虑更加充分地对数据进行分析处理，并考虑充分运用酒品质量的先验知识，设计更符合其数据分布的核进行分类，或建立更精细的模型进行求解，以取得期望的效果。