中国地质大学(武汉)

机器学习第四次作业

姓　　名：叶宇涛

专　　业：计算机科学与技术

学　　号：20191000595

指导老师：刘超

1 目录

6 支持向量机

6.2 在西瓜数据集 3.0α上分别用线性核和高斯核训练一个 SVM.

6.2.1 编程题目理解

题目要求我们利用西瓜数据集 3.0α,分别利用不同的核函数：线性核以及高斯核，训练一个支持向量机。其中，给出西瓜数据集的内容如下：

编号	密度	含糖率	好瓜
1	0.697	0.46	1
2	0.774	0.376	1
3	0.634	0.264	1
4	0.608	0.318	1
5	0.556	0.215	1
6	0.403	0.237	1
7	0.481	0.149	1
8	0.437	0.211	1
9	0.666	0.091	0

10	0.243	0.267	0
11	0.245	0.057	0
12	0.343	0.099	0
13	0.639	0.161	0
14	0.657	0.198	0
15	0.36	0.37	0
16	0.593	0.042	0
17	0.719	0.103	0

表格 6.2.1-1 西瓜数据集 3.0α

而对于支持向量机 SVM，基本的想法就是求解能够正确划分训练数据集，并且使几何间隔最大的分离超平面，是一种二分类模型。本题需要通过两个核函数，将原本线性不可分的点投影至高维空间，转化为线性可分，来区分密度、含糖率不同的好瓜以及坏瓜。

6.2.2 原理阐述

硬间隔问题：

SVM 学习的基本原理是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示，$wx+b=0$即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。

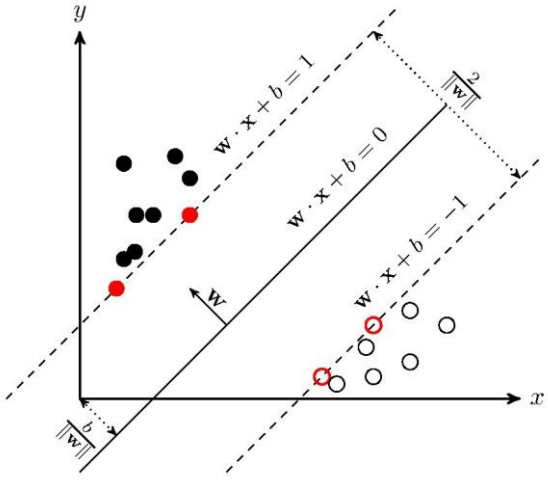

图 6.2.2-1 SVM 示例图

因此，求解最大超平面分割问题就可以转化为约束优化问题：

$$\min_{w,b}\frac{1}{2}\|w\|^2$$

$$s.t.\, y_i(w_i x+b)\geq 1, i=1,2,\ldots,N$$

此时，用有约束的拉格朗日乘子法求解上式：

$$L(w,b,a)=\frac{1}{2}||w||^2-\sum_{i=1}^{N}a_i(y_i(wx_i+b)-1)$$

其中，α_i为拉格朗日乘子，并且$\alpha_i \geq 0$。因此，需要求出w，b带入L中就可以求出最大值。

而求最大值，通常采用求偏导的方式。即对w，b求偏导为 0，可分别得出下式：

$$w=\sum_{\{i=1\}}^{N}a_i y_i x_i$$

$$\sum_{i=1}^{N}a_i\, y_i=0$$

带入到L中，消去w，b有：

$$\max_a -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}a_i\, a_j y_i y_j(x_i y_j)+\sum_{i=1}^{N}a_i$$

该式即为对偶问题，我们需要的就是优化a_i的值，在满足kkt条件之后，用 SMO 算法求解凸二次规划问题，可以求出a_i。

$$y_i(w^T x_i+b)-1=0$$

$$y_i y_i(w^T x_i+b)-y_i=0$$

$$\hat{b}=y_i-w^T x_i$$

最终，可以求出模型$f(x)$:

$$f(x)=w^T x+b=\sum_{i=1}^{m}\alpha_i\, y_i x_i^T x+b$$

对于 SMO 算法，这里太难了，个人没有看懂，于是对于算法的实现，采取调库的方式进行。

软间隔问题：

由于训练集中常常出现错误的数据，因此，难免会出现分类错误的情况，SVM 中给出软间隔的定义，目的是允许分类中出现一些错误，保证 SVM 的鲁棒性。

SVM 把把分类错误的个数作为损失函数，有：

$$Loss=\sum_{i=1}^{N}I\{y_i(w^T x_i+b)<1\}$$

令$z=y_i(w^T x_i+b)<1$作为分类正确，则计数为 1，损失函数显见为 0-1 损失，这是一个非凸的函数，在求解的过程中，存在很多的不足，通常在实际的使用中将 0-1 损失函数作为一个标准.

但是 0-1 损失函数不连续非凸，性质不好，西瓜书中，给了 hinge 损失函数作为替代。

$$Loss\ =\ max\,\{0,1-yi(w^T xi+b\}$$

与硬间隔一样，用拉格朗日乘子法转换，求偏导，得出对偶问题，经对比发现，与硬间隔的区别是对偶变量的约束不同：前者$0 \leqslant \alpha i \leqslant C$，后者是$0 \leqslant \alpha i$。

$$\xi_i = 1 - y_i(w^T x_i + b)$$

根据之前推导的公式：

$$f(x) = w^T x + b = \sum_{i=1}^{m} \alpha_i y_i x_i^T x + b$$

以及 KKT 条件：

$$\begin{cases} C \geqslant \alpha_i \geqslant 0, \quad \mu_i \geqslant 0 \\ y_i f(x_i) - 1 + \xi_i \geqslant 0 \\ \alpha_i (y_i f(x_i) - 1 + \xi_i) = 0 \\ \xi_i \geqslant 0, \mu_i \xi_i = 0 \end{cases}$$

可以得出下面分析：

1) 当$\alpha_i = 0$时，$y_i f(x_i) - 1 + \xi_i$可等于零，也可以大于 0。说明：边界外的样本权重必为零，即对模型没有影响。
2) 当$\alpha_i > 0$时，$y_i f(x_i) - 1 + \xi_i$必为零，说明边界上的权重不为零，即模型建模主要取决于支持向量。
3) 当$\alpha_i = C$时，则μi必为零(根据公式：$C = \alpha_i + \mu_i$)，则ξi取值任意。
4) $\xi i > 1$：错误分类。
5) $\xi i \leqslant 1$：样本落在最大间隔的内部。
6) 当$\alpha_i < C$时，μ_i必大于零(根据公式：$C = \alpha_i + \mu_i$)，进一步必有$\xi_i = 0$。即样本就在最大分类边界上。

简单来说，软间隔就是给 SVM 一些缓冲空间，允许一些错误的分类。其中，支持向量中错误分类为 0，并且，很明显看出，C 越大α_i可以取得的值越大，即建模的时候越看重支持向量，越不能分错，因此，C 的取值很重要。

核函数：

核函数是为了解决线性不可分时候计算困难的情况。在低维空间中，分类平面可能无法正确区别一些线性不可能情况，此时，利用非线性函数将低维样本映射至高维空间中，可转换为线性可分的情况。但是，高维空间的数据计算很困难，于是，出现了核函数来简化高维运算量。

将相近的点映射到一个空间，将远的点分到另一个空间，核函数的诀窍在于解决了映射后高维空间中样本距离$\| \Phi(x_i) - \Phi(x_j) \|$的计算，但又不显式地展示出映射函数$\Phi(\cdot)$。

线性核：

让转换函数为$\Phi(x) = x$，得到线性核函数，则两个向量的点积为：

$$k(x, x') = x^T x'$$

当不需要在特征空间进行运算时，可以用线性核函数。如原始数据已经是高维的、可比较的，并且在输入空间线性可分。其实就是指样本点本来就是线性可分，不做任何映射。但

是在西瓜数据集中，并不是线性可分的，这点可以从西瓜数据集的分布图可以看出，无法用线性函数区分好瓜坏瓜。于是，需要映射到高维空间。

高斯核：

此时，就出现了高斯核。高斯核是机器学习中比较普遍的核函数，将低维映射至无限维空间中，公式如下：

$$k(x,x') = \exp\left(-\frac{1}{2}\sum_{j=1}^{p}\frac{\left(x_i - x_j'\right)^2}{\sigma_j^2}\right)$$

6.2.3 算法设计思路

由于该算法中 SMO 算法没有看懂，个人实现比较困难，所以算法设计中采取调库的方式进行。Sklearn 库中自带有 SVM 模型，可以帮助我们训练。因此，算法设计思路如下：

1) 数据统一格式化，转化为 narray 数据结构，将$y = 0$转换为$y = -1$，方便分类。
2) 初始化各类参数，设置迭代次数为 1000 次。
3) 调用 sklearn 中的 SVM 库，分别用线性核以及高斯核进行分类。
4) 打印出预测值、真实值、支持向量，进行对比。
5) 画图，分别画出正例、负例、支持向量、超分类平面，用不同样例表示。

6.2.4 实验流程、测试结果及分析

测试结果：

线性核:

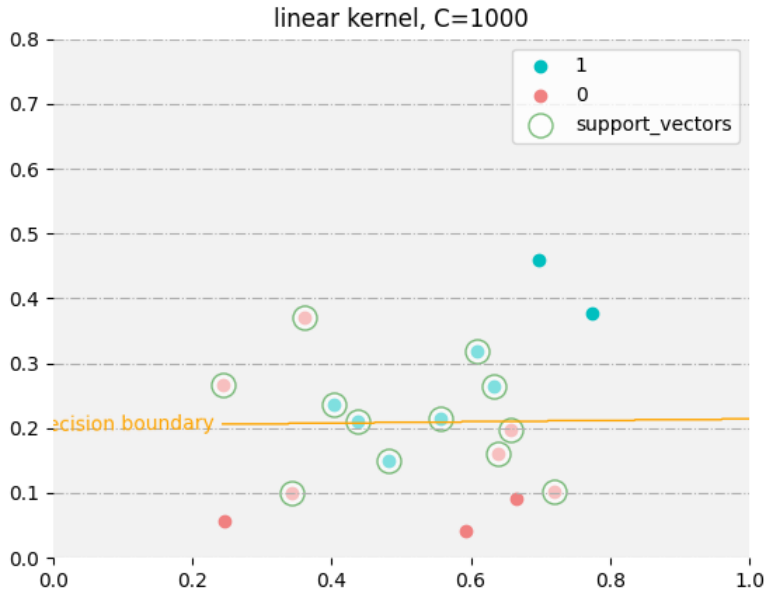

图 6.2.4-1 线性核拟合的 SVM

```
线性核：
预测值： [ 1  1  1  1  1  1 -1  1 -1  1 -1 -1 -1 -1  1 -1 -1]
真实值： [ 1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
支持向量： [ 9 11 12 13 14 16  2  3  4  5  6  7]
```

图 6.2.4-2 线性核结果

高斯核：

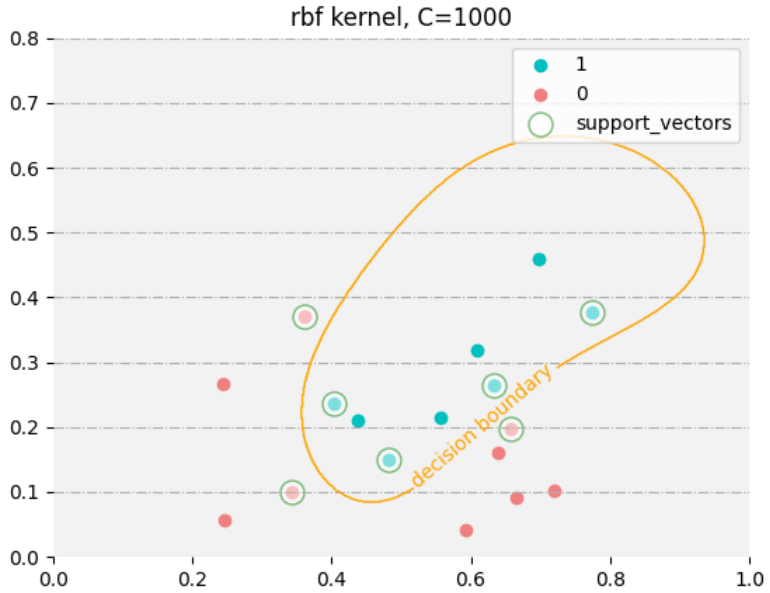

图 6.2.4-3 高斯核拟合的 SVM

```
高斯核：
预测值： [ 1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
真实值： [ 1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
支持向量： [11 13 14  1  2  5  6]
```

图 6.2.4-4 高斯核结果

结果分析：

由于西瓜数据集二维上不是线性可分，因此，用线性核无法完美区分所有点数据点，总会有一两个样本点分类错误。这也就是为什么线性核用的很少的原因。而高斯核划定的超平面可以恰好将样本点分隔开，实现了比较好的效果。

根据图片也可以看出，线性核划分的超平面有三四个点分类错误，高斯核就可以刚好区分开所有的样本点。当然，也可能是样本点数据量不够大的原因。

6.2.5 代码结构，核心代码简要分析

训练代码以及注释：

```python
clf_rbf = svm.SVC(C=C)#创建SVC分类器
clf_rbf.fit(X, y.astype(int))#训练模型
print('高斯核：')#输出模型信息
print('预测值：', clf_rbf.predict(X))#输出预测值
print('真实值：', y.astype(int))#输出真实值
print('支持向量：', clf_rbf.support_)#输出支持向量

print('-' * 40)#输出分隔符
clf_linear = svm.SVC(C=C, kernel='linear')#创建SVC分类器
clf_linear.fit(X, y.astype(int))#训练模型
print('线性核：')#输出模型信息
print('预测值：', clf_linear.predict(X))#输出预测值
print('真实值：', y.astype(int))#输出真实值
print('支持向量：', clf_linear.support_)#输出支持向量
```

图 6.2.5-1 训练代码以及注释

画图代码以及注释：

```python
def plt_support_(clf, X_, y_, kernel, c):
    pos = y_ == 1#获取正例
    neg = y_ == -1#获取负例
    ax = plt.subplot()#创建子图

    x_tmp = np.linspace(0, 1, 600)#设置x轴范围
    y_tmp = np.linspace(0, 0.8, 600)#设置y轴范围

    X_tmp, Y_tmp = np.meshgrid(x_tmp, y_tmp)#设置网格点

    Z_rbf = clf.predict(#计算预测值
        np.c_[X_tmp.ravel(), Y_tmp.ravel()]).reshape(X_tmp.shape)

    # ax.contourf(X_, Y_, Z_rbf, alpha=0.75)
    cs = ax.contour(X_tmp, Y_tmp, Z_rbf, [0], colors='orange', linewidths=1)#绘制超
平面
    ax.clabel(cs, fmt={cs.levels[0]: 'decision boundary'})#设置超平面标签

    set_ax_gray(ax)#设置子图属性

    ax.scatter(X_[pos, 0], X_[pos, 1], label='1', color='c')#绘制正例
    ax.scatter(X_[neg, 0], X_[neg, 1], label='0', color='lightcoral')#绘制负例

    ax.scatter(X_[clf.support_, 0], X_[clf.support_, 1], marker='o',c='w',
edgecolors='g', s=150,
               label='support_vectors',alpha=0.5)#绘制支持向量

    ax.legend()#绘制图例
    ax.set_title('{} kernel, C={}'.format(kernel, c))#设置标题
    plt.show()
```

图 6.2.5-2 画图代码以及注释

6.2.6 本次实验解决的主要问题，主要收获

这次的模型来说，比前面三个实验要困难许多。通俗的原理并不是很难，困难的地方在于数学公式的推导，以及各类公式的使用，都对我的数学功底是一大考验。尤其在拉格朗日乘子法的推导，以及相应对偶问题的推导，都需要我查阅大量资料，才勉强看懂。遇到了很多数学公式推导上面的问题，对 SMO 算法没有完全掌握，对其的了解仅限于优化 SVM 的目标函数。并且，太过于困难，时间比较仓促，导致最后没有手写实现 SVM，是比较遗憾的事情。

主要收获在于对于 SVM 的原理的理解，以及对转换问题思路有了更加清晰的认知，遇到一些比较困难的优化问题的时候，常常可以转化其为对偶问题。并且，对于核函数有了全新的认知，刚开始认为核函数只是将低维样本映射到高维空间的函数。查阅相关资料发现，这种认知是错误的，映射函数是单独的一类函数，核函数存在的意义在于简化高维空间的运算，而不是映射函数。

对于 SVM 的推导过程有了比较清楚的认识，相信在日后的学习过程中，还会接触到

SVM 相关的内容。

6.2.7 编码及内容撰写中的参考来源

【1】 https://blog.csdn.net/mengjizhiyou/article/details/103437423 ， 核 函 数 (Kernel function)(举例说明，通俗易懂)

【2】 https://blog.csdn.net/c406495762/article/details/78072313#2smo%E7%AE%97%E6%B3%95, Python3《机器学习实战》学习笔记（八）：支持向量机原理篇之手撕线性 SVM