泰坦尼克问题的 SVM 求解

刘浚嘉
上海交通大学机械与动力工程学院
118020910046，junjialiu@sjtu.edu.cn

摘要：本文利用支持向量机(SVM)方法解决 Kaggle 泰坦尼克幸存预测问题。其中，详细描述了该问题的凸二次规划模型的建立，应用拉格朗日乘子法以及对偶性原理将其转换为对偶问题，最后应用 KKT 条件求最优解，得到最优的分类超平面方程。本文使用 python 语言编写求解程序，首先使用 sklearn 库中的 svm 函数求解，之后又基于 SVM 的原理，重新实现了函数功能并进行求解。

一、题目

泰坦尼克幸存预测问题是 Kaggle（一个数据建模和数据分析竞赛平台）入门级比赛项目，也是机器学习的基础问题，问题描述如下：

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

题目给出了泰坦尼克号的乘员数据，以及他们的身份标签。根据这些数据，尝试找出乘员存亡与其身份之间的关系。在学习到的模型基础上，预测一名乘客能否幸免于泰坦尼克沉没。

二、模型建立

支持向量机，因其英文名为 Support Vector Machine，故一般简称 SVM，通俗来讲，它是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，其学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。

2.1 线性分类器

本题是典型的二分类问题，即给定一些数据点，它们分别属于两个不同的类，

现在要找到一个线性分类器把这些数据分成两类。如果用 x 表示数据点，用 y 表示类别（y 可以取 1 或者-1，分别代表两个不同的类），一个线性分类器的学习目标便是要在 n 维的数据空间中找到一个超平面（hyper plane）将数据点分为两类，这个超平面的方程可以表示为（ω^{T}中的 T 代表转置）：

$$\omega^T x + b = 0$$

下面举个简单的例子。如下图 1 所示，现在有一个二维平面，平面上有两种不同的数据，分别用圈和叉表示。由于这些数据是线性可分的，所以可以用一条直线将这两类数据分开，这条直线就相当于一个超平面，超平面一边的数据点所对应的 y 全是-1 ，另一边所对应的 y 全是 1。

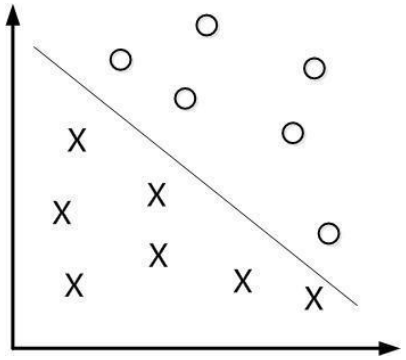


图 1 简单线性分类

这个超平面可以用分类函数$\mathrm{f(x)} = \omega^{\mathrm{T}}\mathrm{x} + \mathrm{b}$表示，当$f(x)$等于 0 的时候，x便是位于超平面上的点，而f(x)大于 0 的点对应 $\mathrm{y} = 1$ 的数据点，$f(x)$小于 0 的点对应$\mathrm{y} = -1$的点，如下图 2 所示：

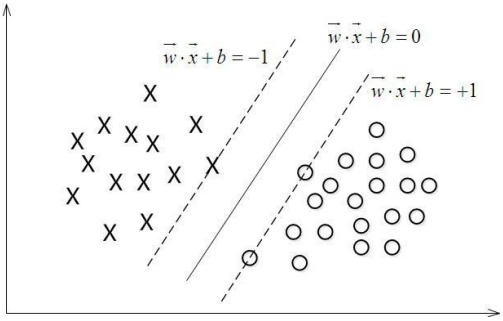


图 2 超平面与数据点之间的关系

线性分类问题的关键在于确定超平面的函数表达式。直观上讲，这个超平面应该是最适合分开两类数据的那一个。图 3 中列出了两种可能的超平面选择，其中黄色的色带表示超平面到训练样本的最短距离，可以理解为学习到的模型的容错性，很明显图 3(b)具有更大的最短距离，因此这个模型相较于图 3(a)有更好的鲁棒性。因此，判定“最适合”的标准就是超平面到两侧数据点的距离和最大，这就成为了一个最优化问题，优化的目标是找到使间隔最大的那个超平面。

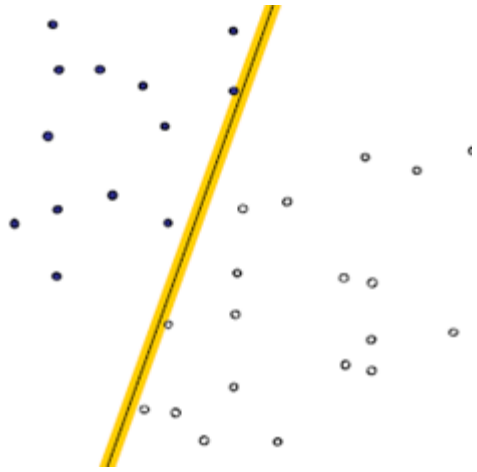


图 3(a) 第一种超平面

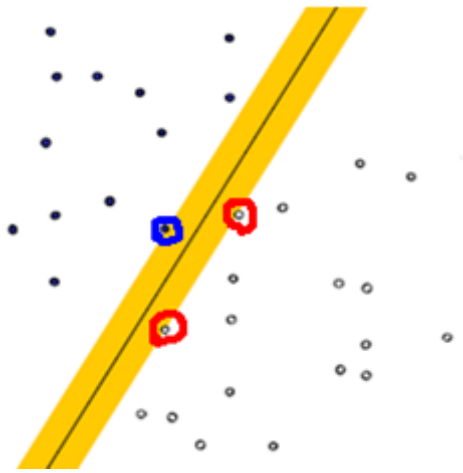


图 3(b) 第二种超平面

2.2 函数间隙和几何间隙

一个点距离分离超平面的远近可以表示分类预测的确信程度。在超平面$\omega^T x + b = 0$已经确定的前提下，$|\omega^T x + b|$能够相对地表示点x距离超平面的远近，通过观察$\omega^T x + b$的符号与类标记 y 的符号是否一致可判断分类是否正确，所以，可以用$y(\omega^T x + b)$来表示分类的正确性及确信度，即为函数间隙，下面给出其定义：

定义 1（函数间隙） 对于给定的训练数据集T和超平面(ω, b)，定义超平面(ω, b)关于样本点(x_i, y_i)的函数间隙为

$$\hat{\gamma}_i = y_i(\omega^T x_i + b)$$

定义超平面(ω, b)关于整个训练集T的函数间隔为超平面(ω, b)关于T中所有样本点(x_i, y_i)的函数间隔的最小值，即

$$\hat{\gamma} = \min_{i=1,\cdots,N} \hat{\gamma}_i$$

对函数间隔规范化，得到几何间隔，记作γ_i。

定义 2（几何距离） 对于给定的训练数据集T和超平面(ω, b)，定义超平面(ω, b)关于样本点(x_i, y_i)的几何间隔为

$$\gamma_i = y_i\left(\frac{\omega}{\|\omega\|} \cdot x_i + \frac{b}{\|\omega\|}\right)$$

其中，$\|\omega\|$是向量的L_2范数（norm）。定义超平面(ω, b)关于整个训练集T的函数间隔为超平面(ω, b)关于T中所有样本点(x_i, y_i)的函数间隔的最小值，即

$$\gamma = \min_{i=1,\cdots,N} \gamma_i$$

支持向量机学习的基本思想就是求解能够正确划分训练集并且几何间隔最大的分离超平面（本文主要叙述硬间隔最大化，至于软间隔最大化无非是引入了松弛变量和惩罚项，使模型容错性更强；对于非线性支持向量机所使用的核函数，其原理是对原空间数据进行非线性变换，将其映射到一个希尔伯特空间$\mathcal{H}$，使分类的超曲面对应于$\mathcal{H}$中的超平面，本文也不再赘述）。

2.3 模型建立

该问题可以表示为如下的最优化问题：

$$\max_{\omega,b} \gamma$$

$$s.t. \quad y_i\left(\frac{\omega}{\|\omega\|}\cdot x_i+\frac{b}{\|\omega\|}\right)\geq\gamma, \qquad i=1,\dots,N$$

根据函数间隔与几何间隔的关系，可以将其表示为函数间隔形式

$$\max_{\omega,b} \frac{\hat{\gamma}}{\|\omega\|}$$

$$s.t. \quad y_i(\omega^T x_i+b)\geq\hat{\gamma}, \qquad i=1,\dots,N$$

函数间隔$\hat{\gamma}$仅是一个常数，其取值并不影响最优化问题的解，此处$\hat{\gamma}$取 1。

将$max \to \min$，由于求$\frac{1}{\|\omega\|}$的最大值相当于求$\frac{1}{2}\|\omega\|^2$的最小值，所以上述目标函数等价为：

$$min \ \frac{1}{2}\|\omega\|^2$$

$$s.t. \quad y_i(\omega^T x_i+b)\geq 1, \qquad i=1,\dots,N$$

因为现在的目标函数是二次的，约束条件是线性的，所以它是一个凸二次规划问题。

2.4 拉格朗日乘子法

由于这个问题的特殊结构，可以通过拉格朗日对偶性变换到对偶变量 的优化问题，即通过求解与原问题等价的对偶问题得到原始问题的最优解，这就是线性可分条件下支持向量机的对偶算法，这样做的优点在于：一者对偶问题往往更容易求解；二者可以自然的引入核函数，进而推广到非线性分类问题。

拉格朗日乘子法的具体步骤就是对于每一个约束（不等式约束 g(x)、等式约束 h(x)）引入一个拉格朗日乘子（$u\geq 0$、$v\geq 0$），本问题中只有不等式约束，因此可将拉格朗日函数写作：

$$L(\omega,b,u)=f(\omega,b)-u^T g(x)+\sum_{i=1}^{N}u_i$$

$$=\frac{1}{2}\|\omega\|^2-\sum_{i=1}^{N}u_i(y_i(\omega^T x_i+b)-1)+\sum_{i=1}^{N}u_i$$

$$u_i\geq 0, \qquad i=1,2,\cdots,N$$

令

$$\theta(\omega)=\max_{u_i\geq 0} L(\omega,b,u)$$

由上可知，当某个约束条件不满足时，即存在$y_i(\omega^T x_i+b)<1$，那么显然有$\theta(\omega)=\infty$（因为u_i可以是任意大于等于 0 的数，此处令$u_i=\infty$）。而当所有约束条件都满足时，则最优值为$\frac{1}{2}\|\omega\|^2$，也就是目标函数中要最小化的值。因此最小化$\frac{1}{2}\|\omega\|^2$就等价于最小化$\theta(\omega)$。原目标函数就可以改写为：

$$\min_{\omega,b}\theta(\omega)=\min_{\omega,b}\max_{u_i\geq 0}L(\omega,b,u)=p^*$$

其中，p^*表示上述问题的最优值，且和最初的问题是等价的。但是直接求解这个问题又过于麻烦，因为先是ω和b两个参数，而后又是不等式约束。因此，不妨把$\min_{\omega,b}$和$\max_{u_i\geq 0}$的位置交换一下，将上述问题转化为对偶问题：

$$\max_{u_i\geq 0}\min_{\omega,b}L(\omega,b,u)=d^*$$

根据对偶问题的弱对偶性可知，$d^*\leq p^*$。并由推论知，$d^*=p^*$时，ω^*,b^*和u^*分别为原问题和对偶问题的最优解。

（1） 求$\min_{\omega,b}L(\omega,b,u)$

对拉格朗日函数分别对ω,b求偏导数并令其等于 0。

$$\frac{\partial L(\omega,b,u)}{\partial\omega}=\omega-\sum_{i=1}^{N}u_iy_ix_i=0$$

$$\frac{\partial L(\omega,b,u)}{\partial b}=\sum_{i=1}^{N}u_iy_i=0$$

得

$$\omega=\sum_{i=1}^{N}u_iy_ix_i$$

$$\sum_{i=1}^{N}u_iy_i=0$$

将其代入拉格朗日函数，既得

$$L(\omega,b,u)=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}u_iu_jy_iy_j(x_i\cdot x_j)-\sum_{i=1}^{N}u_iy_i\left(\left(\sum_{j=1}^{N}u_jy_jx_j\right)\cdot x_i+b\right)+\sum_{i=1}^{N}u_i$$

$$=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}u_iu_jy_iy_j(x_i\cdot x_j)+\sum_{i=1}^{N}u_i$$

即

$$\min_{\omega,b}L(\omega,b,u)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}u_iu_jy_iy_j(x_i\cdot x_j)+\sum_{i=1}^{N}u_i$$

（2） 求$\max_{u_i\geq 0}\min_{\omega,b}L(\omega,b,u)$

$$\max_{u_i\geq 0}-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}u_iu_jy_iy_j(x_i\cdot x_j)+\sum_{i=1}^{N}u_i$$

$$s.t.\ \sum_{i=1}^{N} u_i y_i = 0$$

将目标函数转换为极小

$$\min_{u_i \geq 0} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} u_i u_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N} u_i$$

$$s.t.\ \sum_{i=1}^{N} u_i y_i = 0$$

由此可求得对偶问题的最优解u^*，根据对偶原理，可由u^*求得原问题的最优解ω^*, b^*：

$$\omega^* = \sum_{i=1}^{N} u_i^* y_i x_i$$

$$b^* = y_j - \sum_{i=1}^{N} u_i^* y_i (x_i \cdot x_j)$$

2.5 利用 KKT 条件求解

该最优化问题满足 KKT (Karush-Kuhn-Tucker) 条件，即得

$$\frac{\partial L(\omega^*, b^*, u^*)}{\partial \omega^*} = \omega^* - \sum_{i=1}^{N} u_i^* y_i x_i = 0$$

$$\frac{\partial L(\omega^*, b^*, u^*)}{\partial b^*} = -\sum_{i=1}^{N} u_i^* y_i = 0$$

$$u_i^* (y_i (\omega^* \cdot x_i + b^*) - 1) = 0$$

$$y_i (\omega^* \cdot x_i + b^*) - 1 \geq 0$$

$$u_i^* \geq 0, i = 1,2,\cdots,N$$

由此得

$$\omega^* = \sum_{i=1}^{N} u_i^* y_i x_i$$

$$b^* = y_j - \sum_{i=1}^{N} u_i^* y_i (x_i \cdot x_j)$$

故分离超平面可以写作

$$\sum_{i=1}^{N} u_i^* y_i (x \cdot x_i) + b^* = 0$$

2.6 支持向量

最后阐述一下支持向量的含义。所有坐落在间隔边界上的点被称作是 “支持向量”，如图 3(b)中用红色、蓝色小圆圈出来的点。

由 KKT 条件知，

$$u_i^*(y_i(\omega^* \cdot x_i + b^*) - 1) = 0$$

对于$u_i^* > 0$的样本x_i，有

$$y_i(\omega^* \cdot x_i + b^*) - 1 = 0$$

$$\omega^* \cdot x_i + b^* = \pm 1$$

即x_i一定间隔边界上，即为支持向量。

三、解题过程

本文使用 python 语言对所提供的数据集进行数据清洗，并利用 sklearn 中自带的 SVM 函数对该问题进行求解。在本节后半部分会详述 SVM 函数的内部结构，并独立完成 SVM 函数的编写以展示此类凸二次规划的解题方法。

3.1 数据标签

题目中提供的训练数据集 train.csv 含有 891 个乘客信息，每位乘客有 12 个属性，数据集截图如图 4：

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Floren	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2.	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily M	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vi	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achen	female	14	1	0	237736	30.0708		C

图 4 数据集截图

除了 Survived（表示是否获救）外，其他是乘客的信息：

- PassengerId ： 乘客 ID
- Pclass ： 乘客等级(1/2/3 等舱位)
- Name ： 乘客姓名
- Sex ： 性别
- Age ： 年龄
- SibSp ： 堂兄弟/妹个数
- Parch ： 父母与小孩个数
- Ticket ： 船票信息
- Fare ： 票价
- Cabin ： 客舱
- Embarked ： 登船港口

3.2 数据清洗

利用 Pandas 库，导入 csv 数据，我们可以了解数据集的基本情况。

```
In [1]: import pandas as pd

In [2]: train=pd.read_csv('/home/skylark/PythonProjects/SVM_Titanic/train.csv')

In [3]: test=pd.read_csv('/home/skylark/PythonProjects/SVM_Titanic/test.csv')

In [4]: train.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB

In [5]: train.describe()
```

Out[5]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

图 5 数据集基本情况

由图 5 可知，Age 和 Cabin 特征出现了数据缺失的情况，其中 Cabin 特征缺失较为严重，只有 204 条记录，可以考虑舍弃该特征。而 Age 特征可以通过 fillna() 函数填补缺失值，填充值为 Age 的平均值，并将其分为（1、2、3）三个等级，方便数据处理：

In [70]:
```
train['Age'].fillna(train['Age'].mean(),inplace=True)
age = np.zeros(len(train))
age[train['Age']<20] = 1
age[(train['Age']>=20)&(train['Age']<60)] = 2
age[(train['Age']>=60)] = 3
train['Age'] = age
train.head()
```

Out[70]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1.0	2.0	1	0	7.2500	3.0
1	1	1	0.0	2.0	1	0	71.2833	1.0
2	1	3	0.0	2.0	0	0	7.9250	3.0
3	1	1	0.0	2.0	1	0	53.1000	3.0
4	0	3	1.0	2.0	0	0	8.0500	3.0

此外，还需要对几个非数值类型的数据进行替换，包括 Sex、Embarked 两类特征。

In [67]:
```
sex = np.zeros(len(train))
sex[train['Sex']== 'male'] = 1
sex[train['Sex']== 'female'] = 0
train['Sex'] = sex

train.head()
```

Out[67]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1.0	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0.0	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	0.0	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0.0	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	1.0	35.0	0	0	373450	8.0500	NaN	S

In [68]:
```
train['Embarked'].fillna(train['Embarked'].mode()[0],inplace=True)
Embarked = np.zeros(len(train))
Embarked[train['Embarked']== 'C'] = 1
Embarked[train['Embarked']== 'Q'] = 2
Embarked[train['Embarked']== 'S'] = 3
train['Embarked'] = Embarked
train.head()
```

Out[68]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1.0	22.0	1	0	A/5 21171	7.2500	NaN	3.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0.0	38.0	1	0	PC 17599	71.2833	C85	1.0
2	3	1	3	Heikkinen, Miss. Laina	0.0	26.0	0	0	STON/O2. 3101282	7.9250	NaN	3.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0.0	35.0	1	0	113803	53.1000	C123	3.0
4	5	0	3	Allen, Mr. William Henry	1.0	35.0	0	0	373450	8.0500	NaN	3.0

删除数据集中的无用信息 PassengerId、Name、Ticket、Cabin 特征：

In [12]:
```
print(type(train))
dropping = ['PassengerId','Name', 'Ticket','Cabin']
train.drop(dropping,axis=1, inplace=True)
test.drop(dropping,axis=1, inplace=True)
train.head()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[12]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1.0	22.0	1	0	7.2500	3.0
1	1	1	0.0	38.0	1	0	71.2833	1.0
2	1	3	0.0	26.0	0	0	7.9250	3.0
3	1	1	0.0	35.0	1	0	53.1000	3.0
4	0	3	1.0	35.0	0	0	8.0500	3.0

3.3 数据分析

计算特征之间的相关性，并绘制热力图：

```
In [27]: f, ax = plt.subplots(figsize=(12, 10))
         plt.title('Pearson Correlation of Movie Features')
         sns.heatmap(abs(train.astype(float).corr()),linewidths=0.25,vmax=1.0, square=True, cmap="Y
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x272457983c8>
```

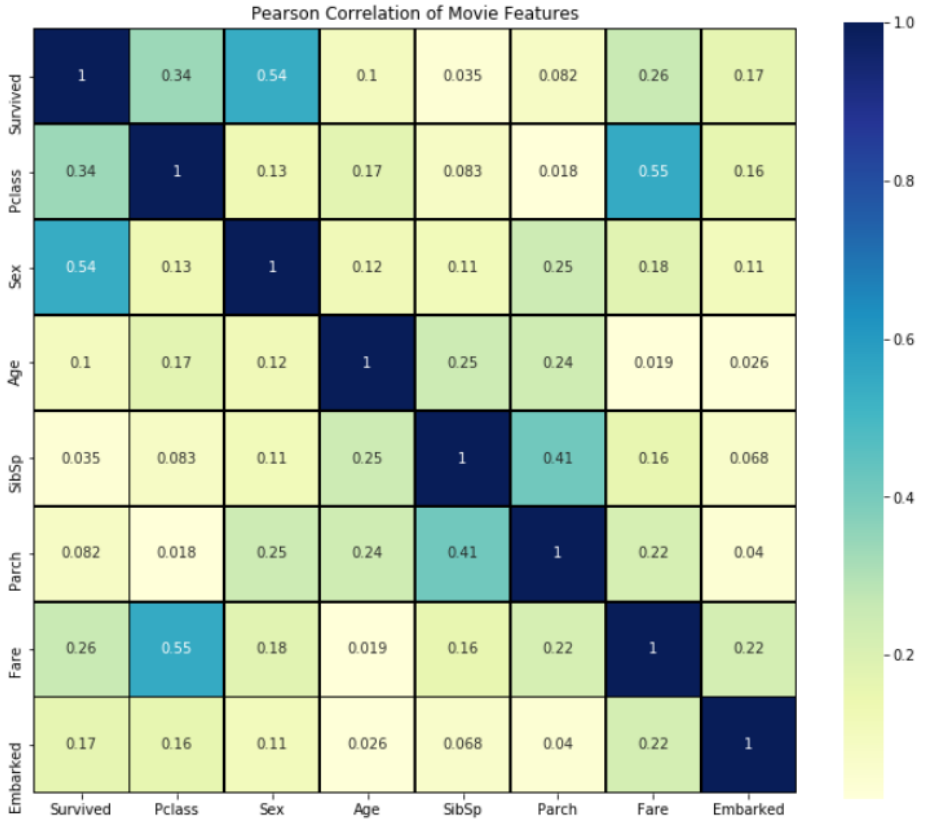


图 4 特征相关性热力图

上图中颜色越深表示特征之间的相关性越大，故 Sex、Pclass、Fare 对能否存活(Survived)影响较大。

根据 Sex 和 Pclass 特征，可以绘制出如下的关系图：

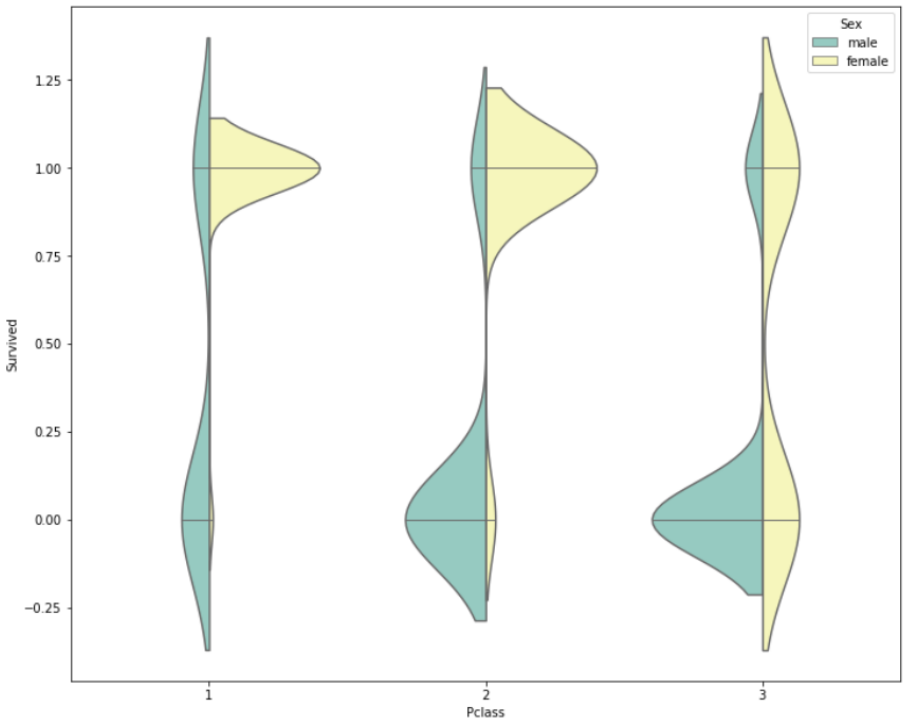


图 5 Sex 和 Pclass 特征与 Survived 关系的 Violin 图

上图直观地揭示了，有钱人以及女性更容易在泰坦尼克事故中幸存下来，而穷人以及男性等特征会增加其在事故中的死亡率。

3.4 建立 SVM 模型并预测

使用 sklearn 库中的 svm 函数，并将处理过的训练数据集分为 Survived 和其他两部分。

```
In [15]: from sklearn import svm
```

```
In [16]: train_y=train['Survived']
         train_x=train.drop('Survived',axis=1)
         print(train.head())
         print(train_x.head())
         print(train_y.head())
```

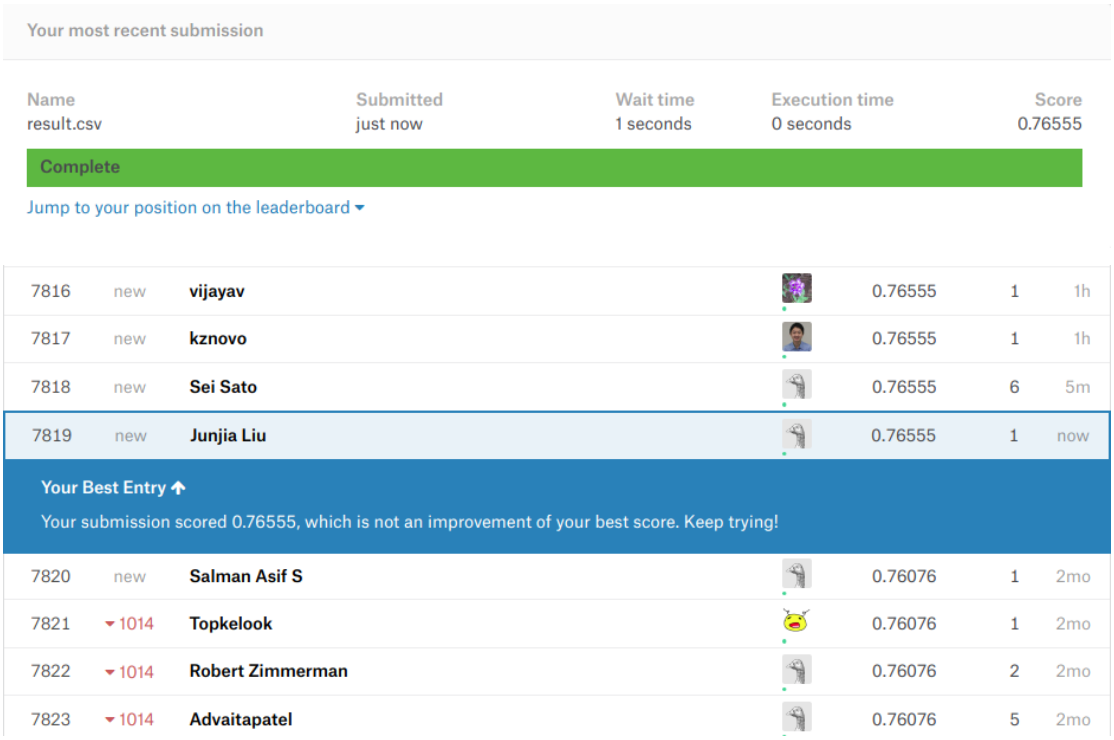
```
In [17]: model=svm.SVC(kernel='linear').fit(train_x, train_y)
```

此处用到了线性核$\kappa(x, x_i) = x \cdot x_i$，用于最基本的线性可分情况。对于核函数的选用，本文就不过多涉及。至此，我们已经训练出了分类模型，将其应用到测试数据集的预测中，注意测试数据集也需要相同方式的数据处理，此处不再赘述，详见附件程序。预测过程如下：

```
In [24]: predictions = model.predict(test)
```

```
In [25]: test_rare=pd.read_csv('test.csv')
         submission = pd.DataFrame({ 'PassengerId': test_rare['PassengerId'],
                                     'Survived': predictions })
         submission.to_csv("result.csv", index=False)
```

预测结果被存入 result.csv 文件中，将其提交到 Kaggle 上看一下正确率：



很遗憾，得分 0.76555，排名 7819 名。由此可见，SVM 对于这道题来说可能并不是个好办法，但这就不是本文的重点了

3.5 SVM 的 python 实现

本程序省略了数据处理部分，直接使用了上面 IPython 程序处理后的 train_new.csv 和 test_new.csv 数据集。由于原题给出的测试数据集(test.csv)实际上是预测数据集，缺少标签。因此在本程序的将原训练数据集按传统方法分为 7:3 两部分，30%的训练数据不参加训练而是应用到测试环节，用于观察模型是否过拟合。具体思路见下面代码及注释 ：

```
1.  # -*- coding: utf-8 -*-
2.  """
3.  Pycharm
4.
5.  by Junjia Liu
6.
7.  """
8.
9.  from numpy import *
10. import pandas as pd
11.
12. def loadDataSet(filename): #读取数据
13.
14.     f=pd.read_csv(filename)
15.     # train=f[ :630, :]
16.     # test=f[631: , :]
```

```
    labelMat=f['Survived']
    labelMat.replace(0, -1, inplace=True)
    dataMat=f.drop('Survived',axis=1)
    return dataMat,labelMat #返回数据特征和数据类别

def selectJrand(i,m): #在 0-m 中随机选择一个不是 i 的整数
    j=i
    while (j==i):
        j=int(random.uniform(0,m))
    return j

#最优解剪辑，《统计学习方法》p127 7.108
def clipAlpha(aj,H,L):  #保证 a 在 L 和 H 范围内（L <= a <= H）
    if aj>H:
        aj=H
    if L>aj:
        aj=L
    return aj

def kernelTrans(X, A, kTup): #核函数，输入参数,X:支持向量的特征树；A：某一行特征数据；kTup：('lin',k1)核函数的类型和参数
    m,n = shape(X)
    K = mat(zeros((m,1)))
    if kTup[0]=='lin': #线性函数
        K = X * A.T
    elif kTup[0]=='rbf': # 径向基函数(radial bias function)
        for j in range(m):
            deltaRow = X[j,:] - A
            K[j] = deltaRow*deltaRow.T
        K = exp(K/(-1*kTup[1]**2)) #返回生成的结果
    else:
        raise NameError('Houston We Have a Problem -
- That Kernel is not recognized')
    return K


#定义类，方便存储数据
class optStruct:
    def __init__(self, dataMatIn, classLabels, C, toler, kTup):  # 存储各类参数
        self.X = dataMatIn  #数据特征
        self.labelMat = classLabels #数据类别
        self.C = C #软间隔参数 C，参数越大，非线性拟合能力越强
        self.tol = toler #停止阀值
```

```
58.         self.m = shape(dataMatIn)[0] #数据行数
59.         self.alphas = mat(zeros((self.m,1)))
60.         self.b = 0 #初始设为 0
61.         self.eCache = mat(zeros((self.m,2))) #缓存
62.         self.K = mat(zeros((self.m,self.m))) #核函数的计算结果
63.         for i in range(self.m):
64.             self.K[:,i] = kernelTrans(self.X, self.X[i,:], kTup)
65.
66.
67. def calcEk(oS, k): #计算 Ek（参考《统计学习方法》p127 公式 7.105）
68.     gXk = float(multiply(oS.alphas,oS.labelMat).T*oS.K[:,k] + oS.b) #7.104
69.     # print(float(multiply(oS.alphas,oS.labelMat).T*oS.K[:,k]))
70.     Ek = gXk - float(oS.labelMat[k])
71.     return Ek
72.
73. #随机选取 aj，并返回其 E 值
74. def selectJ(i, oS, Ei):
75.     maxK = -1
76.     maxDeltaE = 0
77.     Ej = 0
78.     oS.eCache[i] = [1,Ei]
79.     validEcacheList = nonzero(oS.eCache[:,0].A)[0]  #返回矩阵中的非零位置的行
    数
80.     if (len(validEcacheList)) > 1:
81.         for k in validEcacheList:
82.             if k == i:
83.                 continue
84.             Ek = calcEk(oS, k)
85.             deltaE = abs(Ei - Ek)
86.             if (deltaE > maxDeltaE): #返回步长最大的 aj
87.                 maxK = k
88.                 maxDeltaE = deltaE
89.                 Ej = Ek
90.         return maxK, Ej
91.     else:
92.         j = selectJrand(i, oS.m)
93.         Ej = calcEk(oS, j)
94.     return j, Ej
95.
96.
97. def updateEk(oS, k): #更新 os 数据
98.     Ek = calcEk(oS, k)
99.     oS.eCache[k] = [1,Ek]
100.
```

```
#首先检验 ai 是否满足 KKT 条件，如果不满足，随机选择 aj 进行优化，更新 ai,aj,b 值
def innerL(i, oS): #输入参数 i 和所有参数数据
    Ei = calcEk(oS, i) #计算 E 值
    # print(float(oS.labelMat[i]*Ei)+oS.alphas[i])
    if ((oS.labelMat[i]*Ei < -oS.tol) and (oS.alphas[i] < oS.C)) or ((oS.labelMat[i]*Ei > oS.tol) and (oS.alphas[i] > 0)): #检验这行数据是否符合 KKT 条件 参考《统计学习方法》p128 公式 7.111-113
        j,Ej = selectJ(i, oS, Ei) #随机选取 aj，并返回其 E 值
        alphaIold = oS.alphas[i].copy()
        alphaJold = oS.alphas[j].copy()
        if (oS.labelMat[i] != oS.labelMat[j]): #以下代码的公式参考《统计学习方法》p126 下侧，L、H 取值计算
            L = max(0, oS.alphas[j] - oS.alphas[i])
            H = min(oS.C, oS.C + oS.alphas[j] - oS.alphas[i])
        else:
            L = max(0, oS.alphas[j] + oS.alphas[i] - oS.C)
            H = min(oS.C, oS.alphas[j] + oS.alphas[i])
        if L==H:
            # print("Error: L==H")
            return 0
        eta = 2.0 * oS.K[i,j] - oS.K[i,i] - oS.K[j,j]   #参考《统计学习方法》p127 公式 7.107
        if eta >= 0:
            # print("Error: eta>=0")
            return 0
        oS.alphas[j] -= oS.labelMat[j]*(Ei - Ej)/eta #参考《统计学习方法》p127 公式 7.106，未经剪辑的情况下沿着约束方向更新 alpha[j]
        oS.alphas[j] = clipAlpha(oS.alphas[j],H,L) #参考《统计学习方法》p127 公式 7.108，对新 alpha[j]进行剪辑
        updateEk(oS, j)
        if (abs(oS.alphas[j] - alphaJold) < oS.tol): #alpha 变化大小阀值（自己设定）
            # print("Error: j not moving enough")
            return 0
        oS.alphas[i] += oS.labelMat[j]*oS.labelMat[i]*(alphaJold - oS.alphas[j])#参考《统计学习方法》p127 公式 7.109， 由 alpha[j]求得 alpha[i]
        updateEk(oS, i) #更新数据
        #以下求解 b 的过程，参考《统计学习方法》p129 公式 7.115-7.116
        b1 = oS.b - Ei- oS.labelMat[i]*(oS.alphas[i]-alphaIold)*oS.K[i,i] - oS.labelMat[j]*(oS.alphas[j]-alphaJold)*oS.K[i,j]
        b2 = oS.b - Ej- oS.labelMat[i]*(oS.alphas[i]-alphaIold)*oS.K[i,j]- oS.labelMat[j]*(oS.alphas[j]-alphaJold)*oS.K[j,j]
        if (0 < oS.alphas[i]<oS.C):
```

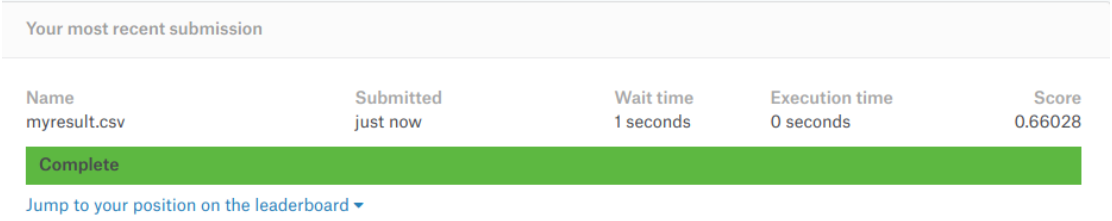
```
134.             oS.b = b1
135.         elif (0 < oS.alphas[j]<oS.C):
136.             oS.b = b2
137.         else:
138.             oS.b = (b1 + b2)/2.0
139.         return 1
140.     else:
141.         # print("Error:")
142.         return 0
143.
144.
145. #SMO 函数，用于快速求解出 alpha
146. def smoP(dataMatIn, classLabels, C, toler, maxIter,kTup=('lin', 0)): # 输 入参数：数据特征，数据类别，参数 C，阀值 toler，最大迭代次数，核函数（默认线性核）
147.     oS = optStruct(mat(dataMatIn),mat(classLabels).transpose(),C,toler, kTup)
148.     iter = 0
149.     entireSet = True
150.     alphaPairsChanged = 0
151.     while (iter < maxIter) and ((alphaPairsChanged > 0) or (entireSet)):
152.         alphaPairsChanged = 0
153.         if entireSet:
154.             for i in range(oS.m): #遍历所有数据
155.                 alphaPairsChanged += innerL(i,oS)
156.                 #print("fullSet, iter: %d i:%d, pairs changed %d" % (iter,i,alphaPairsChanged)) #显示第多少次迭代，那行特征数据使 alpha 发生了改变，这次改变了多少次 alpha
157.             iter += 1
158.         else:
159.             nonBoundIs = nonzero((oS.alphas.A > 0) * (oS.alphas.A < C))[0]

160.             for i in nonBoundIs: #遍历非边界的数据
161.                 alphaPairsChanged += innerL(i,oS)
162.                 # print("non-bound, iter: %d i:%d, pairs changed %d" % (iter,i,alphaPairsChanged))
163.             iter += 1
164.         if entireSet:
165.             entireSet = False
166.         elif (alphaPairsChanged == 0):
167.             entireSet = True
168.         print("iteration number: %d" % iter)
169.     return oS.b,oS.alphas
170.
171. def testRbf(data_train, data_predict):
```

```
172.     dataArr,labelArr = loadDataSet(data_train) #读取训练数据
173.
174.     #因为题目中提供的 test 数据是没有标签的，所以测试集就按照 3:7 的比例从训练数据集中分出
175.     dataArr_train=dataArr.values[ :630, :]
176.     labelArr_train=labelArr.values[ :630]
177.     dataArr_test=dataArr.values[631: , :]
178.     labelArr_test=labelArr.values[631: ]
179.
180.     b,alphas = smoP(dataArr_train, labelArr_train, 20, 0.0001, 12, ('rbf', 0.2)) #通过 SMO 算法得到 b 和 alpha
181.     datMat=mat(dataArr_train)
182.     labelMat = mat(labelArr_train).transpose()
183.     svInd=nonzero(alphas)[0]  #选取不为 0 数据的行数（也就是支持向量）
184.     sVs=datMat[svInd] #支持向量的特征数据
185.     labelSV = labelMat[svInd] #支持向量的类别（1 或-1）
186.     print("there are %d Support Vectors" % shape(sVs)[0]) #打印出共有多少的支持向量
187.     m,n = shape(datMat) #训练数据的行列数
188.     errorCount = 0
189.     for i in range(m):
190.         kernelEval = kernelTrans(sVs,datMat[i,:], ('rbf', 0.2)) #将支持向量转化为核函数
191.         predict=kernelEval.T * multiply(labelSV,alphas[svInd]) + b  #这一行的预测结果（代码来源于《统计学习方法》p133 里面最后用于预测的公式）注意最后确定的分离平面只有那些支持向量决定。
192.         # print(sign(predict), sign(labelArr_train[i]))
193.         if sign(predict) != sign(labelArr_train[i]): #sign 函数 -1 if x < 0, 0 if x==0, 1 if x > 0
194.             errorCount += 1
195.     print("the training error rate is: %f" % (float(errorCount)/m)) #打印出错误率
196.
197.
198.     errorCount_test = 0
199.     datMat_test=mat(dataArr_test)
200.     labelMat_test = mat(labelArr_test).transpose()
201.     m,n = shape(datMat_test)
202.     for i in range(m): #在测试数据上检验错误率
203.         kernelEval = kernelTrans(sVs,datMat_test[i,:], ('rbf', 0.2))
204.         predict=kernelEval.T * multiply(labelSV,alphas[svInd]) + b
205.         if sign(predict) != sign(labelArr_test[i]):
206.             errorCount_test += 1
207.     print("the test error rate is: %f" % (float(errorCount_test)/m))
```

```
208.
209.     #Prediction
210.     datMat_predict=mat(data_predict)
211.     m,n=shape(datMat_predict)
212.     predictions=zeros(m)
213.     for i in range(m):
214.         kernelEval = kernelTrans(sVs, datMat_predict[i, :], ('rbf', 0.2))
215.         predictions[i] = kernelEval.T * multiply(labelSV, alphas[svInd]) +
    b
216.     avg_predict=average(predictions)
217.     for i in range(m):
218.         if predictions[i] >= avg_predict:
219.             predictions[i] = 1
220.         else:
221.             predictions[i] = 0
222.     predictions.astype('int')
223.     test_rare = pd.read_csv('test.csv')
224.     submission = pd.DataFrame({'PassengerId': test_rare['PassengerId'],
225.                                'Survived': predictions}).astype('int')
226.     submission.to_csv("myresult.csv", index=False)
227.
228. #主程序, 直接应用 IPython 程序处理过的数据集 train_new 和 test_new
229. def main():
230.     filename_traindata='train_new.csv'
231.     filename_predictdata=pd.read_csv('test_new.csv')
232.     testRbf(filename_traindata, filename_predictdata)
233.
234. if __name__=='__main__':
235.     main()
```

经过多次调参，在使用高斯核函数、软间隔系数$C = 20$且误差$\varepsilon = 0.0001$的情况下，得到了较为理想的模型。应用该模型预测 test_new.csv 数据集，并将结果处理后，提交到 Kaggle，结果如下：



这一次的成绩为 0.66028，较上次还有所下降。但重要的是，本程序基于支持向量机的原理实现了与 sklearn 库中 svm 函数等同的功能，也是本文创新所在。

参考文献

[1] 李航. 统计学习方法[J]. 2012.