山东大学

信息科学与工程学院

2022－2023 学年第一学期

实 验 报 告

课程名称：信息基础 II

专 业 班 级 崇新学堂

学 生 学 号

学 生 姓 名

课 程 报 告 支持向量机

1. 作业要求

有 2 类二维空间点，A 类和 B 类。
A 类以(0, 0)为中心、(1, 0; 0, 1)为协方差矩阵的二维高斯分布；
B 类以(0, 0)为中心、(5, 0; 0, 5)为协方差矩阵的二维高斯分布；
随机生成 50 个 A 类点， 50 个 B 类点， 并用 SVM 的方法进行分类
请解释你的 SVM 的方法，提供程序和描述结果 （不能调用机器学习库，
可以调用优化求解器, 选择合适的 kernel)

2. 理论部分

2.1 核函数

SVM 用于求解线性可分问题，当然 SVM 的真正强大之处在于其可以利用核函数解决非线性问题，而核函数的作用在于将低维特征映射到高维空间内，这些数据在高维空间内是线性可分的。我们常用的核函数为高斯核函数，如公式（1）所示。

$$k(x,x') = \exp(\frac{-\| x-x' \|^2}{2\sigma^2}) \tag{1}$$

一般选取高斯核函数满足的情况为：**原始特征维度较低（如二维的），训练样本数量远大于原始特征维度。**本次作业中条件非常符合，因此在本次作业编程中，核函数选择高斯核函数而不选择线性核函数。

核函数的具体映射过程：以样本 $x^{(i)}$ 为例进行映射，在本次实验中总共有 100 个训练样本，因此经过核函数映射，$x^{(i)}$ 会被映射为一个具有 100 个维度的特征向量 $f^{(i)} = [f_1^{(i)}, f_2^{(i)}, \cdots f_N^{(i)}]$，具体的映射函数为 $f_j^{(i)} = \exp(\frac{-\| x^{(i)} - x^{(j)} \|^2}{2\sigma^2})$，根据这个规则便可以完成映射工作。

高斯核函数存在着一个参数 σ 选择的问题，当 σ 较大时会出现较高的偏差和较低的方差，而当 σ 较小时，会出现较低的偏差和较高的方差。

2.2 硬间隔支持向量机（SVM）

支持向量机的主要任务是，寻找到一条直线，使得这条直线可以将两类数据区分开来，并且满足距离直线最近的点到直线的距离是最大的。从而我们首先给出支持向量机的决策方程，如公式（2）所示：

$$h_\theta(x) = w^T x + b \quad \begin{cases} h_\theta(x) > 0 & y = 1 \\ h_\theta(x) < 0 & y = -1 \end{cases} \tag{2}$$

而 SVM 最终的优化目标为距离直线最近的点到直线的距离最大，首先在此引入点到平面的距离公式，如公式（3）

$$r = \frac{y_i \cdot (w^T x + b)}{\| w \|} \tag{3}$$

通过化简可以获得我们的优化目标函数变为：

$$\arg\max_{w,b}\frac{1}{2}w^{T}w \quad st.y_i\cdot(w^{T}x+b)\geq 1 \tag{4}$$

对于该目标函数采用拉格朗日优化法，从而可以获得：

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^{2}-\sum_{i=1}^{n}\alpha_i[y_i(w^{T}x+b)-1] \tag{5}$$

之后通过对拉格朗日函数求偏导，令偏导数为 0 可获得以下约束条件：

$$\begin{cases}\dfrac{\partial L}{\partial w}=0\Rightarrow \quad w=\sum_{i=1}^{n}\alpha_i y^{(i)}x^{(i)}\\ \dfrac{\partial L}{\partial b}=0\Rightarrow \quad \sum_{i=1}^{n}\alpha_i y^{(i)}=0\end{cases} \tag{5}$$

将其代入拉格朗日方程之后可以获得以下结果：

$$\min_{\alpha}\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{n}\alpha_i \tag{6}$$

约束条件如下：

$$\begin{cases}\sum_{i=1}^{n}\alpha_i y_i=0\\ \alpha_i\geq 0\end{cases} \tag{7}$$

利用拉格朗日函数的固定解法，将限制条件代入，之后求取偏导数为 0 的点，看是否满足限制条件，若不满足则在边界上进行寻找，即可完成问题的求解。

2.3 软间隔支持向量机

为了避免异常点点和离群点的影响，在此介绍一种软间隔支持向量机，与硬间隔支持向量机不同之处在于引入了松弛因子 ξ_i，公式如下所示：

$$y_i(w^{T}x_i)\geq 1-\xi_i \tag{8}$$

从而出现了新的目标函数：

$$\min_{w,b}\frac{1}{2}\|w\|^{2}+C\sum_{i=1}^{n}\xi_i \tag{9}$$

需要注意的是参数 C，当 C 的数值很大的时候，那么就要求 SVM 需要严格分类，不能有错，而当 C 的数值很小的时候，就可以容忍更多的错误。

对于软间隔向量机，本次作业中采用佩加索斯（Pegasos）解法实现优化，具体解法过程如下。我们使用以下损失函数来表示松弛变量：

$$\xi_i=\begin{cases}0 & if \quad y_i(w^{T}x_i+b)\geq 1\\ 1-y_i(w^{T}x_i+b) & if \quad y_i(w^{T}x_i+b)<1\end{cases} \tag{10}$$

则可以将目标函数简化为:

$$\min_{w,b}\frac{1}{2}\|w\|^2+C\sum_{i=1}^{n}\max(0,1-y_i(w^Tx_i+b)) \tag{11}$$

之后为了计算方便性我们令 C=1/n*lamda,从而得到的目标函数为:

$$\min_{w,b}\frac{\lambda}{2}\|w\|^2+\frac{1}{n}\sum_{i=1}^{n}\max(0,1-y_i(w^Tx_i+b)) \tag{12}$$

那么对于一个随机选择的样本点，目标函数变为:

$$f(w,b)=\frac{\lambda}{2}\|w\|^2+\max(0,1-y_i(w^Tx_i+b)) \tag{13}$$

针对该目标函数我们使用佩加索斯算法，可以使用梯度下降法来求解该目标函数:

当 $y_i(w^Tx_i+b)<1$ 时:

$$\nabla f=\begin{bmatrix}\frac{\partial f}{\partial w}\\ \frac{\partial f}{\partial b}\end{bmatrix}=\begin{bmatrix}\lambda w-y_i\cdot x_i\\ -y_i\end{bmatrix} \tag{14}$$

当 $y_i(w^Tx_i+b)>1$ 时:

$$\nabla f=\begin{bmatrix}\frac{\partial f}{\partial w}\\ \frac{\partial f}{\partial b}\end{bmatrix}=\begin{bmatrix}\lambda w\\ 0\end{bmatrix} \tag{15}$$

此时我们可以通过梯度下降的方式来优化参数:

当 $y_i(w^Tx_i+b)<1$ 时:

$$\begin{bmatrix}w_{t+1}\\ b_{t+1}\end{bmatrix}=\begin{bmatrix}w_t\\ b_t\end{bmatrix}-\eta_t\begin{bmatrix}\lambda w_t-y_ix_i\\ -y_i\end{bmatrix} \tag{16}$$

当 $y_i(w^Tx_i+b)>1$ 时:

$$\begin{bmatrix}w_{t+1}\\ b_{t+1}\end{bmatrix}=\begin{bmatrix}w_t\\ b_t\end{bmatrix}-\eta_t\begin{bmatrix}\lambda w_t\\ 0\end{bmatrix} \tag{17}$$

其中 η_t 为学习率 $\eta_t=1/\lambda t$，其中 t 为训练轮次，它的作用是控制参数优化的步长，训练次数越大，步长越短。这便是佩加索斯算法。本次作业主要使用佩加索斯算法进行实现，而没有使用现成的优化函数。

3. 代码实现

3.1 数据产生部分

利用高斯随机数产生函数和 for 循环创建了两类高斯随机点和标签值。

```
def Gauss_random(mean1, cov1, mean2, cov2, number):
    np.random.seed(12)
    x1 = np.random.multivariate_normal(mean1, cov1, number)
    label1 = np.array(np.array(ones((number, 1))))
    x2 = np.random.multivariate_normal(mean2, cov2, number)
    label2 = np.array(np.array(ones((number, 1))))
    for i in range(len(label2)):
        label2[i][0] = -1
    return x1, label1, x2, label2
```

3.2 高斯核函数

高斯核函数将 100x2 的特征矩阵映射为 100x100 的特征矩阵，其中方差值选择为 1，用于后续 SVM 分析。

```
def Gauss_Kernel(X):
    F = np.array(zeros((len(X), len(X))))              #定义一个方阵，维度为样本数量
    for i in range(len(X)):
        for j in range(len(X)):
            B = 0                                      #用于存储中间结果
            for a in range(len(X[0])):
                B -= ((X[i][a]-X[j][a])**2)/2          #对于高斯核函数，方差值选取为1
            F[i][j] = math.e**(B)
    return F
```

3.3 决策函数

决策函数利用 np.dot()函数即可实现。

```
def Predit(x,b,w):
    return x.dot(w)+b
```

3.4 佩加索斯函数

佩加索斯函数使用随机梯度下降实现，迭代次数设置为 n,每次迭代过程中，随机选择一个样本求取梯度进行迭代。

```
def Pegasos(F,Y,lamda):
    n=100000                                           #迭代次数
    row,col = np.shape(F)
    w = np.array(ones((len(F[0]),1)))                  #定义w矩阵初始值
    b = 0                                              #定义b初始值
    for i in range(1,n+1):
        r = random.randint(0,row-1)                    #随机梯度下降时，随机选取一个样本的梯度
        eta = 1.0/(lamda*i)                            #定义学习率，随着学习次数增加，学习率逐渐下降
        predit = Predit(F[r],b,w)                      #针对随机选中的样本进行预测
        if Y[r][0]*predit[0] > 1:                      #进行梯度优化
            w = w - w/i
        else:
            w = w - w/i
            for j in range(len(F[0])):
                w[j][0] -= eta*Y[r][0] * F[r][j]
            b += eta*Y[r][0]
    return w,b                                         #返回最终迭代后的w和b
```

3.5 绘图函数

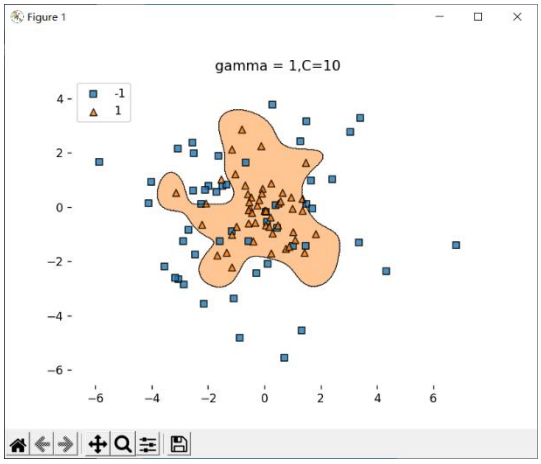
```
def Plot(x1,x2,y1,y2):
    x = []
    y = []
    for i in range(len(x1)):
        x.append(x1[i])
        y.append(y1[i][0])
    for i in range(len(x2)):
        x.append(x2[i])
        y.append(y2[i][0])
    x11 = []
    x12 = []
    x21 = []
    x22 = []
    for i in range(len(y)):
        if y[i]>0:
            x11.append(x[i][0])
            x12.append(x[i][1])
        else:
            x21.append(x[i][0])
            x22.append(x[i][1])
    plt.scatter(x11, x12, color='red')
    plt.scatter(x21, x22, color='blue')
    plt.show()
```

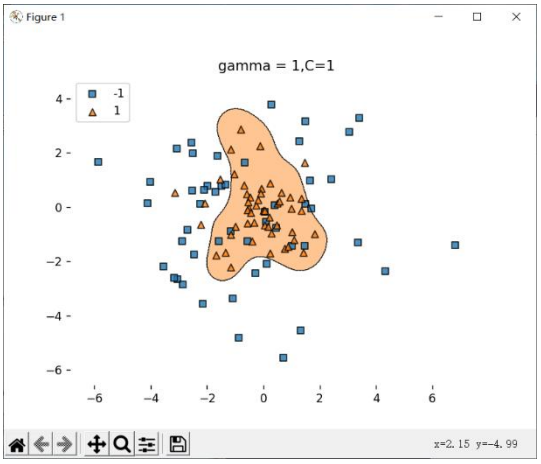
4. 实验结果

通过运行代码发现其分类正确率可以达到 99%

```
(100, 100)
正确率为： 0.99
```

C 的影响：



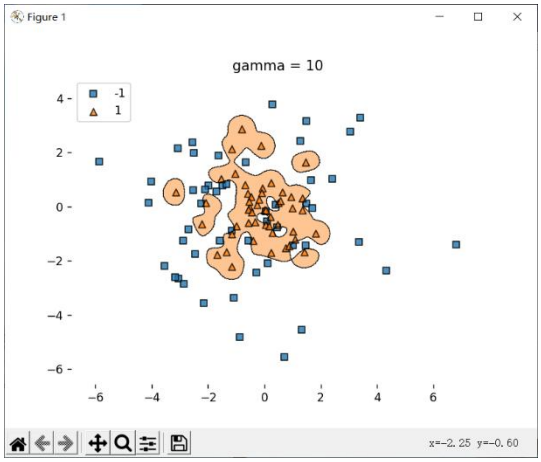


C=10　　C=1

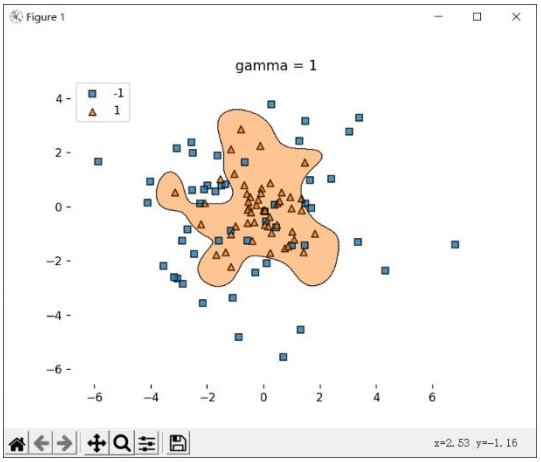
由上图可知，结合（9）的结论，当 C 的取值比较大时，更容易收到异常点及离群点的影响，可以看到以上左图，对于错误的包容程度低。而当 C 的取值比较小的时候，对于错误的包容度高一些，因此边界我范围更广一些。

高斯核函数的方差值的影响：

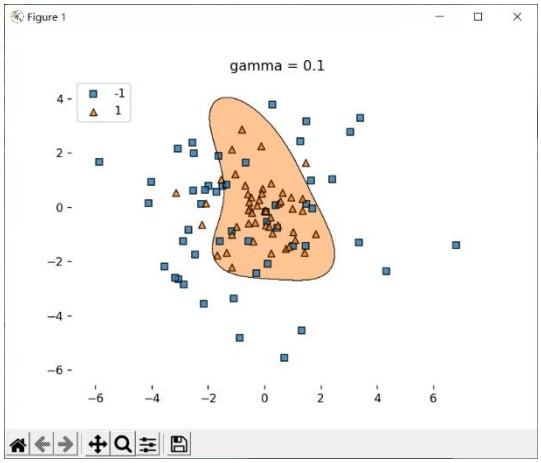
获得的边界结果如下图：



Gamma=10



Gamma=1



Gamma=0.1

根据以上的结果，结合（1）中的内容，gamma 的数值越小会出现较低的偏差和较大的方差，而当 gamma 的数值越大，会出现较大的偏差和较小的方差。

5. 实验心得

此次作业中的知识在上课的时候听得有一些模糊，在课后看了很多的机器学习的课程，逐渐明白了内层的逻辑，对我来说的主要难点在于两个地方，其一是核函数，其二是拉格朗日的优化过程。核函数通过观看吴恩达老师的课程，明白了和函数的内在本质，并且能够在不同的情况下选择不同的核函数。而数学优化过程利用拉格朗日进行优化，而实际编程也有现成的包，但在实际编程的时候我选择了使用随机梯度下降法实现佩加索斯算法，可以在较低的时间复杂度内完成计算，并且最后的分类效果表现也很优异。通过此次实验，更透彻地理解了 SVM 的核心，收获良多。辛苦老师、学长、学姐。