作业五： 2D SVM

实验目的： 通过二次规划实现 SVM 方法。

实验过程：

SVM 作为一个比较流行的二类分类器，可以利用分割超平面将数据分为两类，在实现中其求解过程是用 SMO 算法进行二次规划求解的，对目标函数加使用拉格朗日乘子之后变为对偶函数最后的优化问题：

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

要解决的是在参数上求最大值 W 的问题，至于 x_i 和 y_i 都是已知数。C 由我们预先设定，也是已知数。

SMO算法的目标是求出一系列的alpha和b，一旦求出了这些alpha，就很容易计算出权重向量w，并得到分割超平面，从而对数据进行分类。其工作原理是每次循环中选择两个alpha进行优化处理，一旦找到一对合适的alpha，则增大其中一个同时减小另一个。“合适”表示两个alpha必须满足两个条件，一是必须要在间隔边界之外；二是还没有进行过区间化处理或者不在边界上。

算法主要步骤如下：

```
创建一个alpha向量并初始化
当迭代次数小于最大迭代次数时（外循环）
    对数据集中的每个数据向量（内循环）：
        如果该数据向量可以被优化：
            随机选择另外一个数据向量
            同时优化这两个向量
            如果两个向量都不能被优化，退出内循环
    如果所有向量都没被优化，增加迭代数目，继续下一次循环
```

当获得 alpha 值之后，则可以基于这些值得到超平面，这其中包括了 w 的计算：

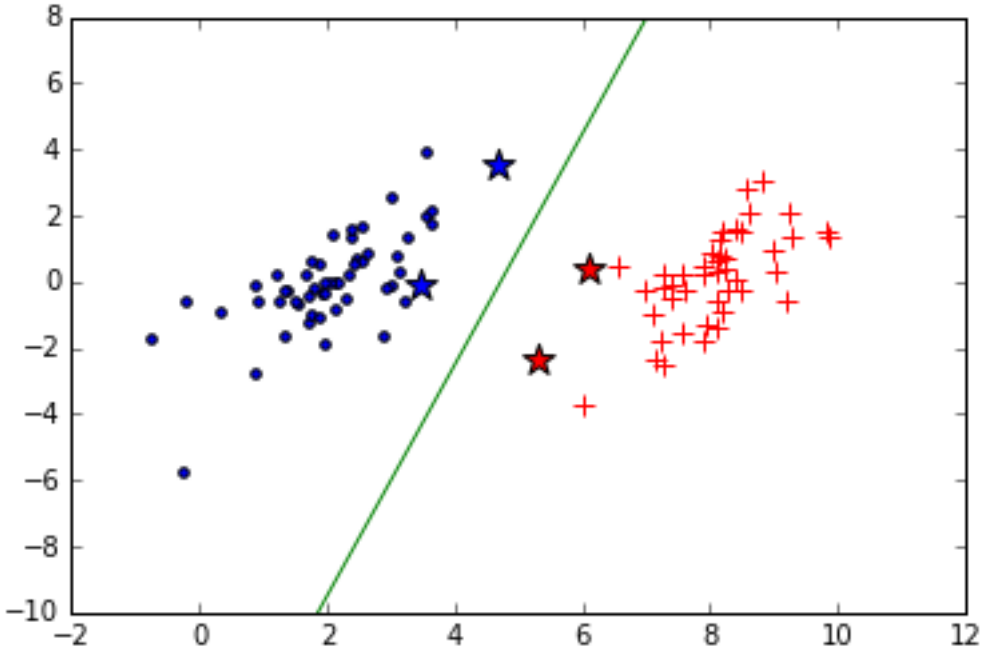
```
'''
计算 w 并画出分割面
'''
def calcWs(alphas,dataArr,classLabels,b):
```

```
        X=np.mat(dataArr)
        #labelMat=np.mat(classLabels).transpos()
        tmp=np.mat(classLabels)
        labelMat=np.transpose(tmp)
        m,n=np.shape(X)
        w=np.zeros((n,1))
        for i in range (m):
            w+=np.multiply(alphas[i]*labelMat[i],X[i,:].T)
        min_x=min(X[:,0])[0,0]
        max_x=max(X[:,0])[0,0]
        y_min_x=float(-b-w[0]*min_x)/w[1]
        y_max_x=float(-b-w[0]*max_x)/w[1]
        plt.axis([0, 10, 0, 10])
        plt.axis([-2, 12, -10, 8])
        plt.plot([min_x,max_x],[y_min_x,y_max_x],'-g')
```

最后便可以对实验数据进行分类，并在图中将点及支持向量展示出来：

```
for i in range(100):
    if alphas[i] > 0.0:
        print alphas[i]
        if  labelArr[i] == 1.0:
            plt.scatter(dataArr[i][0],dataArr[i][1],s=150,marker='*',c='r')
            print dataArr[i][0],dataArr[i][1]
        if  labelArr[i] == -1.0:
            plt.scatter(dataArr[i][0],dataArr[i][1],s=150,marker='*',c='b')
            print dataArr[i][0],dataArr[i][1]
    else:
        if labelArr[i] == 1:
            plt.scatter(dataArr[i][0],dataArr[i][1],s=50,marker='+',c='r')
        else:
            plt.scatter(dataArr[i][0],dataArr[i][1],s=50,marker='.',c='b')
```

实验结果：



实验小结： 从实验结果中可以看出，数据被很好地分成了两类，绿线表示分割超平面，星型点为支持向量。