SVM

聂欣雨

2019 年 11 月 5 日

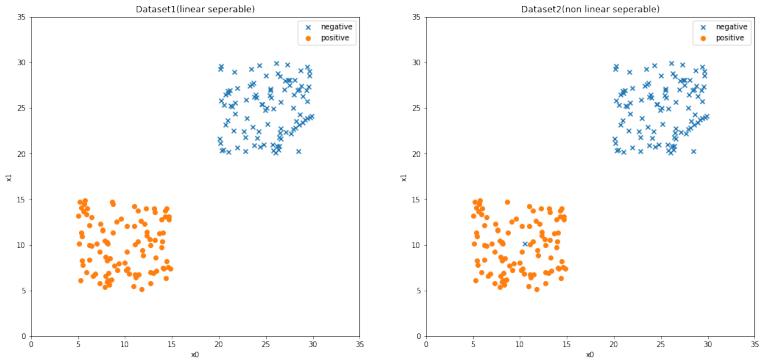
1 准备数据

1.1 线性可分数据集

使用 numpy 库的 rand 方法，制作线性可分数据集 $D1$。令位于直线 y=17 下方的点为正样本，上方的点为副样本，得到线性可分数据如图 1(a) 所示。

1.1.1 线性不可分数据集

在线性可分数据集的基础上制作线性不可分数据集 $D2$，将 $D1$ 中的第一个点设置为 [10.5180034749718639, 10.11005192452767676]，并修改它的标签为-1，得到线性不可分数据，结果如图 1(b) 所示。



(a) 线性可分数据集　　(b) 线性不可分数据集

图 1: SVM 数据集

2 二次规划求解

使用 cvxopt 作为拉格朗日对偶问题的求解器。该求解器能够在一定时间内得到形如公式 (1) 的二次规划问题。

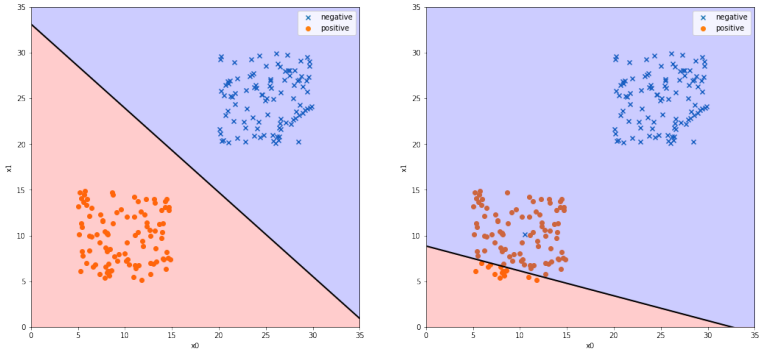
$$\min_x \frac{1}{2} x^T P x + q^T \tag{1}$$

subject to

$$Gx <= h \tag{2}$$

$$Ax = b \tag{3}$$

因此，只需要计算出 P、q、G、h、A、b 之后，调用 cvxopt.solvers.qp 方法即可解得最优的 α。将其带入 KKT 条件中得到相应的 w 和 b。结果如图 2 所示。



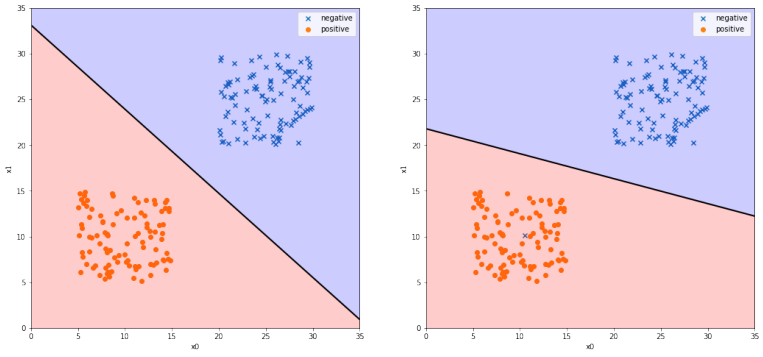
(a) 线性可分数据集　　(b) 线性不可分数据集

图 2: 二次规划求解结果

可以发现，对于线性可分的数据集，cvxopt 可以得到能够正确划分所有样本的结果；而对于线性不可分数据集，cvxopt 求解器得到的结果并不能令人满意。

3 sklearn

机器学习库 sklearn 提供了 SVM 分类器 SVC 类。通过调用 SVC 类实例的 fit 方法，可以轻松地从数据中学习得到一个 SVM 分类器。分类结果如图 3 所示。



(a) 线性可分数据集　　(b) 线性不可分数据集

图 3: sklearn SVM 分类结果

4 梯度下降法

可以使用梯度下降法求解 SVM 问题，令 $L(w,b)$ 为整体损失，θ 为正则化参数，按公式 (4) 设计损失函数。

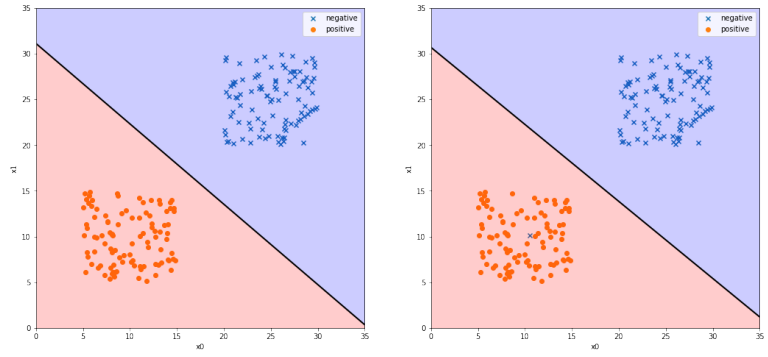
$$L(w,b) = max(0, 1 - y(wx+b)) + \frac{\theta}{2}\|w\|^2 \tag{4}$$

则

$$dw = \begin{cases} \theta w, & y(wx+b) \geqslant 1 \\ \theta w - yx, & y(wx+b) < 1 \end{cases} \tag{5}$$

$$db = \begin{cases} 0, & y(wx+b) \geqslant 1 \\ -y, & y(wx+b) < 1 \end{cases} \tag{6}$$

可以发现，求出的导数与 PLA 算法十分相似，不同之处仅在于 SVM 多了一个正则项。设置 $epochs = 3000$，学习率为 1e-3，$\theta = 0.2$，得到图 4 所示结果。



(a) 线性可分数据集　　(b) 线性不可分数据集

图 4: 梯度下降方法结果

5 遇到困难与解决方法

在实现 SVM 算法的过程中主要遇到了这些困难:

- cvxopt 无法得到最优的结果
  在刚开始实验时，使用了上一次 PLA 实验所用的数据集，发现 cvxopt 无法得到正确的结果。与 sklearn 所得结果对比发现 w 相同而 b 不同。由于不了解 cvxopt 库内部的具体求解过程，因此不明白造成错误的原因。在尝试放大数据集尺度，增大类别间间隔后问题解决。

- 梯度下降法求解速度慢
  一开始我使用循环实现梯度下降法，发现速度非常慢，运行 3000 个 epoch 需要一分钟左右时间。在参考 cs231n assignment1 后，使用向量化方法，通过矩阵运算实现了 batch 操作。设置 batch_size 为 32，运行时间下降到了 1.4 秒。

6 实验代码

实验代码上传到了 github，地址为: `https://github.com/CSU-NXY/NUDT_ML_course_assignments`