ML Lab 2

SVM 支持向量机

王世烜

PB20151796

October 27, 2022

Part 1: 实验要求

本次实验要求我们完成 SVM 的模型实现，并根据 generate() 生成的数据进行训练和测试。

Part 2: 数据集介绍

本次实验选取的数据集来自 generate() 函数，它随机生成 [m,n] 维的线性可分数据，并且按一定的错标率将点进行错标，有关信息如下：

```
x[:5], y[:5]
✓ 0.6s

(array([[ 16.35598344,  -7.51488062,   7.04215491,   0.96109363,
           3.26704912],
        [-10.26505562,  10.56883694,   7.38140384,  -4.69291709,
           7.08620772],
        [ -1.8436832 ,  25.30074089,   8.46623682,   3.1899827 ,
          15.74825278],
        [  5.67981518,  -6.79453519,  -5.96566805,  15.38806096,
           2.86979085],
        [ -3.36887094,   9.55092809,  -3.83582814,  -0.22625917,
           5.26756843]]),
 array([[-1.],
        [ 1.],
        [ 1.],
        [-1.],
        [ 1.]]))
```

图 1: Examples of the data set

其中 y 是本次实验要进行预测的类别。

Part 3: 数据集划分

本实验采用 5 折交叉验证进行模型检验，即训练集与测试集比例为 $4:1$

具体实现方法为将数据集切成五份，每次取一份作为测试集，另外四份合并作为训练集。这样做的好处是随机划分，可以提高模型的泛化能力。

Part 4: 理论基础

4.1 软间隔 SVM 标准问题与对偶问题

软间隔 SVM 标准问题:

$$
\begin{aligned}
min_{w,b,\xi} \quad & \frac{1}{2}\boldsymbol{w^T w} + C\sum_{i=1}^{m}\xi_i \\
s.t. \quad & y_i(\boldsymbol{w^T \phi(x_i} + b)) > 1 - \xi_i, \quad i =, 2, \cdots, m \\
& \xi_i \geq 0, \quad i =, 2, \cdots, m
\end{aligned}
$$

使用合页损失的软间隔 SVM：

$$
min_{w,b} \quad \frac{1}{m}\sum_{i=1}^{m} max(0, 1 - y_i(\boldsymbol{w^T x_i} + b)) + \frac{\lambda}{2}||\boldsymbol{w}||^2
$$

是个无约束问题，可直接采用梯度下降法求解。
对偶问题：

$$
\begin{aligned}
min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x_i^T x_j} - \sum_{i=1}^{m}\alpha_i \\
s.t. \quad & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \cdots, m \\
& \sum_{i=1}^{m}\alpha_i y_i = 0
\end{aligned}
$$

4.2 优化方法

4.2.1 Pegasos 算法

解决 SVM 问题的困难在于其属于带约束的优化问题，为了解决此问题，Pegasos 算法采用了使用合页损失的 SVM 问题，即

$$
min_{w,b} \quad \frac{1}{m}\sum_{i=1}^{m} max(0, 1 - y_i(\boldsymbol{w^T x_i} + b)) + \frac{\lambda}{2}||\boldsymbol{w}||^2
$$

为了用梯度法优化参数, 应当将损失函数对参数 w, b 求导, 损失函数即上述最小化的目标函数:

$$
\begin{aligned}
\frac{\partial J(w,b)}{\partial w} &= -\frac{1}{m}\sum_{i=1}^{m}\amalg(y_i(\boldsymbol{w^T x_i} + b) < 1)y_i x_i + \lambda\boldsymbol{w} \\
\frac{\partial J(w,b)}{\partial b} &= -\frac{1}{m}\sum_{i=1}^{m}\amalg(y_i(\boldsymbol{w^T x_i} + b) < 1)
\end{aligned}
$$

其中，$\mathrm{II}(L)$ 为示性函数，内部语句成立函数值为 1，否则为 0
然后根据梯度下降法的原理:

$$\beta_{t+1} \leftarrow \beta_t - \alpha \nabla J(\beta)$$

即可进行迭代优化。

Algorithm 1 Pegasos

1: for each epoch do
2: $\quad \boldsymbol{w}^* = \boldsymbol{w} - \eta(-\frac{1}{m}\sum_{i=1}^{m} \mathrm{II}(y_i(\boldsymbol{w^T x_i} + b) < 1) y_i x_i + \lambda \boldsymbol{w})$
3: $\quad b^* = b - \eta(-\frac{1}{m}\sum_{i=1}^{m} \mathrm{II}(y_i(\boldsymbol{w^T x_i} + b) < 1))$
4: end for

4.2.2 DCD **算法**

DCD 基于坐标下降法求解软间隔线性 SVM 的对偶型:

$$\begin{aligned} min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j} - \sum_{i=1}^{m} \alpha_i \\ s.t. \quad & 0 \le \alpha_i \le C, \quad i = 1, 2, \cdots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

为了使用坐标下降，需要去掉 $\sum_{i=1}^{m} \alpha_i y_i = 0$ 这个约束，具体做法是将 $\boldsymbol{w^T x_i} + b$ 写作 $\boldsymbol{w^T x_i}$, 即将偏置 b 并入到 w 中，并且在 X 中添加一列 1，从而得到新的对偶问题:

$$\begin{aligned} min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j} - \sum_{i=1}^{m} \alpha_i \\ s.t. \quad & 0 \le \alpha_i \le C, \quad i = 1, 2, \cdots, m \end{aligned}$$

接下来进行坐标下降，每一次迭代时，需要对当前的 α_u 进行优化，从优化目标中拆分得到有关 α 的项，以及无关的项，将 α_u 视为变量，其他无关项视为常量，对 α_u 进行优化，经过一系列运算后得到以下公式:

$$\alpha_u^* = \alpha_u - \frac{y_u \boldsymbol{w^T x_u - 1}}{\boldsymbol{x_u^T x_u}}$$

由于约束的关系 α_u 不能直接更新，新的 $\hat{\alpha}_u^*$ 如下:

$$\hat{\alpha}_u^* = min(max(\alpha_u^*, 0), C)$$

并由此推出 $\boldsymbol{w}$ 的更新公式:

$$\hat{w}^* = \hat{w} + (\hat{\alpha}_u^* - \alpha_u) y_u \boldsymbol{x}_u$$

Part 5: 实验结果

数据维度 20×10000 错标率 0.037

5.1 总体对比

对于数据集进行了 5 **折交叉验证**，得到如下结果:

表 1: 5×10000 五折交叉验证准确率

模型	**第 1 折**	**第 2 折**	**第 3 折**	**第 4 折**	**第 5 折**
梯度下降	0.9445	0.9525	0.9515	0.95	0.9435
DCD	0.9535	0.9565	0.965	0.957	0.9435
sklearn LinearSVC	0.958	0.9555	0.966	0.957	0.9465

表 2: Results of 5 fold cross Validation

模型	**测试集准确率（平均）**	**迭代轮数（最大）**	**参数**	**训练时间（1 次）**
梯度下降	0.9484	500	lr=1/epoch,$\lambda = 0.01$	1m8.3s
DCD	0.9551	10^7	$C = 0.02$	1m19.7s
sklearn LinearSVC	0.9566	10^7	C=1	24.6s

注：对于梯度下降法，学习率设为 $1/epoch$ 这样的目的是防止出现震荡，但是这样做的缺点就是当轮数较大时步长太小，收敛过慢，解决办法是限制，当学习率小于某个给定值时，保持其不变：

```
while True:
    wGrad, bGrad = self.grad(X, y)
    self.w -= LR * wGrad
    self.b -= LR * bGrad
    self.epoch += 1
    if LR > 0.002:
        LR = lr / self.epoch
    else:
        LR=LR
```

5.2 具体参数对比

为提高运行速度改用维度 5×1000 错标率 0.028 进行对比

由教材 P_{130} 的知识可知，对于软间隔支持向量机：

$$
\begin{aligned}
min_{w,b,\xi} \quad & \frac{1}{2}\boldsymbol{w^T w} + C\sum_{i=1}^{m}\xi_i \\
s.t. \quad & y_i(\boldsymbol{w^T \phi(x_i} + b)) > 1 - \xi_i, \quad i =, 2, \cdots, m \\
& \xi_i \geq 0, \quad i =, 2, \cdots, m
\end{aligned}
$$

C 的值越小，容忍样本不满足约束条件的能力越强，软间隔程度越大，下面来验证本条性质：

梯度下降

对于梯度下降来说，式子

$$
min_{w,b} \quad \frac{1}{m}\sum_{i=1}^{m} max(0, 1 - y_i(\boldsymbol{w^T x_i} + b)) + \frac{\lambda}{2}||\boldsymbol{w}||^2
$$

中的 $\lambda \propto \frac{1}{C}$，即 λ 越大，软间隔的程度就越大，而对于本实验数据而言，过大的软间隔间隔显然不是一个好的模型，所以从理论上来说，准确率应该会降低。

实验结果：

表 3: 参数 λ 的比较

$\boldsymbol{\lambda}$	0.01	0.1	1	10	100	1000
准确率	0.97	0.965	0.86	0.635	0.55	0.635

从结果上来看，符合我们的预期。

DCD **方法**

对于 DCD 方法来说参数 C 就与教材中的 C 含义相同，对此参数做同上测试：

表 4: 参数 λ 的比较

$\boldsymbol{C}$	0.01	0.1	1	10	100	1000
准确率	0.975	0.975	0.975	0.97	0.88	0.83

从中选择最好的作为模型参数储存下来。

5.3 可视化

由于高维的点与超平面无法画出，本次实验进行了对于 dim=[2,1000] 的一组随机生成的数据进行了测试，并采取交叉验证的方法得到以下图像（错标率 0.036）：

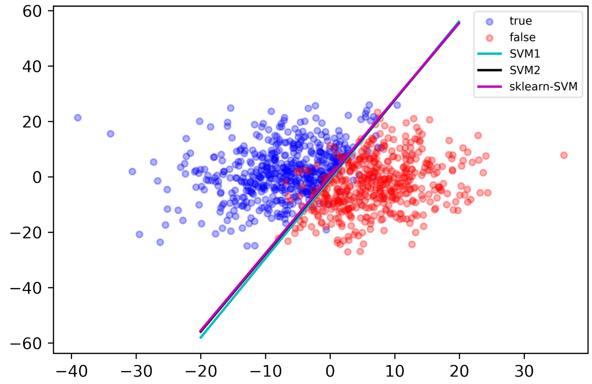

图 2: 第 1 折

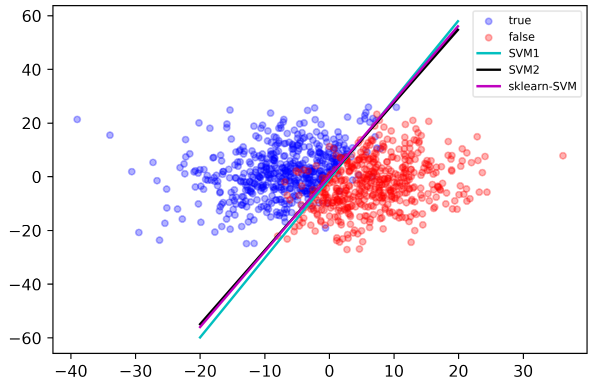

图 3: 第 2 折

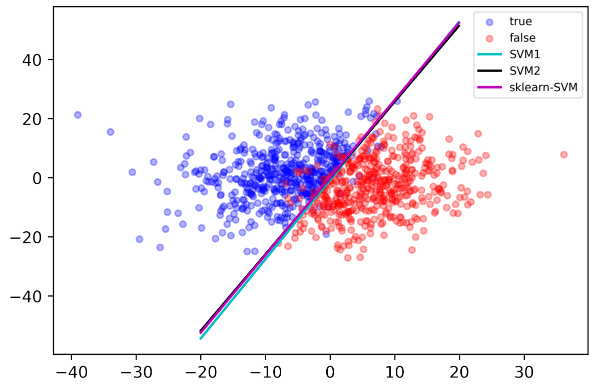

图 4: 第 3 折

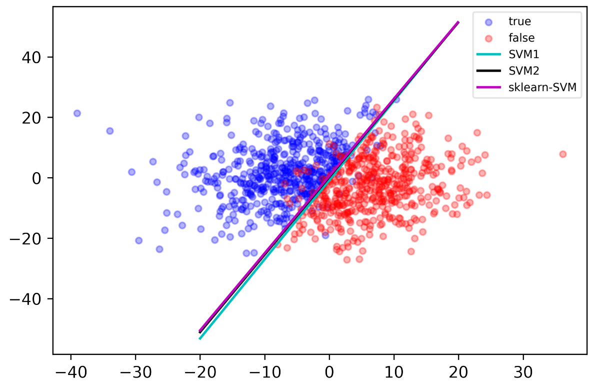

图 5: 第 4 折

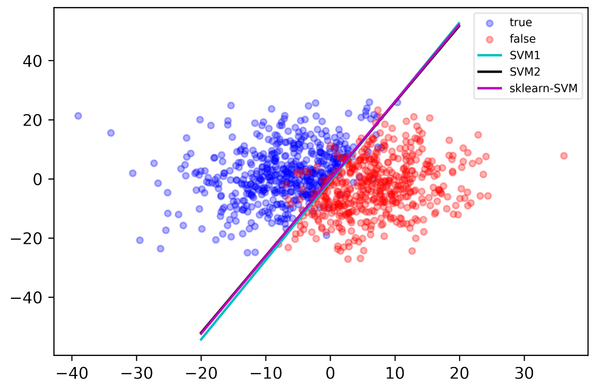

图 6: 第 5 折

三种方法的准确率对比：

表 5: 2×1000 五折交叉验证准确率

模型	**第** 1 **折**	**第** 2 **折**	**第** 3 **折**	**第** 4 **折**	**第** 5 **折**
梯度下降	0.96	0.95	0.94	0.95	0.925
DCD	0.97	0.97	0.945	0.95	0.93
sklearn LinearSVC	0.975	0.965	0.945	0.955	0.93

可见，在二维问题上，三种方法所得的超平面几乎完全一致，效果较好。

5.3 实验总结

本次实验完成了软间隔 SVM 的梯度下降及 DCD 方法求解。在实验过程中遇到了以下问题：

- 模型需要迭代很多次，消耗时间长
- 调参时比较盲目，无法快速找到合适参数
- 对于 SVM 模型原理，不能弄清楚各个参数的含义

解决办法：

- 熟练运用 numpy 的矩阵乘法，能极大提高训练计算速度（DCD 方法无法使用，因为是每条数据进行一次更新）
- 写程序帮我调
- 重新手动推导一遍 SVM 以加深理解。