机器学习概论 实验报告

Lab2: SVM

2020 年 12 月 4 日

1 理论基础

1.1 基本原理

- **支持向量机**(support vector machine, SVM)是一种二分类模型, 它的基本模型是定义在特征空间上的间隔最大的线性分类器, 间隔最大使它有别于感知机. SVM 还包括核技巧, 这使它成为实质上的非线性分类器. SVM 的学习算法就是求解凸二次规划的最优化算法.
- **基本思想**: 求解能够正确划分数据集并且几何间隔最大的分离超平面.

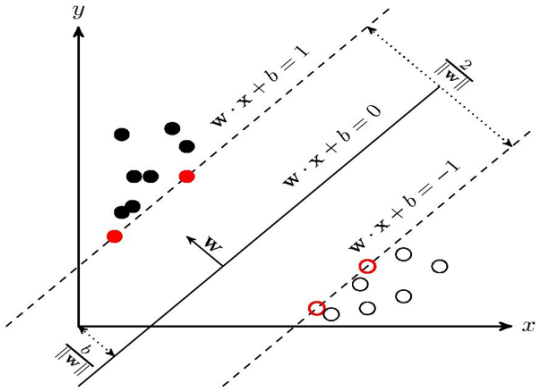

图 1: SVM 图解

- SVM 的类别
  - **线性支持向量机**: 就是上述求解凸二次优化问题的模型
  - **近似线性支持向量机**: 当数据集不能严密可分时, 我们可以引入松弛因子, 这样就能保证点被超平面"分开"
  - **非线性支持向量机**: 有了核技巧的帮助, 就可以将 SVM 推广到非线性情况.

2 数学基础

根据数据集的情况, 我决定选用 **近似线性支持向量机**.

2.1 优化目标

在利用软间隔的情况下, 最优化目标变为:

$$\min_{\boldsymbol{\omega},b} \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{m} l_{hinge}(y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b) - 1)$$

其中 l_{hinge}是 hinge损失函数:

$$l_{hinge}(z) = \max(0, 1 - z)$$

因此最终优化目标可以写为:

$$\min_{\boldsymbol{\omega},b} \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{m} \xi_i$$

$$s.t. \quad \begin{cases} y_i(\boldsymbol{\omega}\boldsymbol{x}_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0, i = 1, 2, \cdots, m \end{cases}$$

这里

$$\xi_i = \max(0, 1 - y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b))$$

2.2 转化为对偶问题

可以写出拉格朗日函数:

$$\begin{aligned} L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) =& \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{m} \xi_i \\ &+ \sum_{i=1}^{m} \alpha_i(1 - \xi_i - y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b)) - \sum_{i=1}^{m} \mu_i\xi_i \end{aligned}$$

这里 $\alpha_i, \mu_i \geq 0$.
让 $L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu})$ 对$\boldsymbol{\omega}, b, \xi_i$ 求偏导并带回, 可以得到对偶问题:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j$$

$$s.t. \quad \begin{cases} \sum\limits_{i=1}^{m} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \ i = 1, 2, \cdots, m \end{cases}$$

而通过α_i能够求出ω:

$$\boldsymbol{\omega} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$$

也能求出b:

$$b = \frac{1}{|S|}\sum_{s\in S}\left(y_s(1 - \xi_i) - \sum_{i\in S} \alpha_i y_i \boldsymbol{x}_i^T\boldsymbol{x}_s\right)$$

2.3 KKT条件条件

其相应 KKT 条件为:

$$\begin{cases} \alpha_i \geq 0, \qquad \mu_i \geq 0 \\ y_i f(\boldsymbol{x}_i) - 1 + \xi_i \geq 0 \\ \alpha_i(y_i f(\boldsymbol{x}_i) - 1 + \xi_i) = 0 \\ \xi_i \geq 0, \ \mu_i\xi_i = 0 \end{cases}$$

根据 KKT 条件我们能够知道,

- 当 $\alpha_i = 0$ 时, 该样本对 $f(\boldsymbol{x})$ 不会有任何影响,
- 当$C > \alpha_i > 0$ 时, 必然有 $y_i f(\boldsymbol{x}_i) = 1 - \xi_i = 1$, 即该样本是支持向量
- 当 $\alpha = C$, 则$\mu_i = 0$, 此时若$\xi_i \leq 1$, 则样本落在最大间隔内部; 若$\xi_i > 1$, 则样本被错误分类.

3 更新算法

3.1 Gradient Descent 算法(GD)

这个算法的理论基础上一次报告已经介绍过了, 这里再展示一下算法流程:

Algorithm 1 GD

Require: 训练的 epochs M; 初始化 $\beta = (w, b)$, 学习率 α

for 每个 epoch **do**
  for 每个训练样本 x_i **do**
    计算误差 $e = 1 - y_i(w \cdot x_i + b)$
    if $e > 0$ **then**
      $w = (1 - \eta)w + \eta C x_i y_i$
      $b = b + \eta C y_i$

3.2 Stochastic Gradient Descent 算法(SGD)

这个算法的理论基础上一次报告已经介绍过了, 这里再展示一下算法流程:

Algorithm 2 SGD

Require: 训练的 epochs M; 初始化 $\beta = (w, b)$, 学习率 α

for 每个 epoch **do**
  for 每个训练样本 x_i **do**
    按这个公式计算所有数据的误差 $e_i = 1 - y_i(w \cdot x_i + b)$
    取误差最大的一项 $i = \arg\max_i e_i$
    if $e_i \leq 0$ **then**
      break
    $w = (1 - \eta)w + \eta C x_i y_i$
    $b = b + \eta C y_i$

3.3 SMO算法

3.3.1 选取待优化的两个参数

- 选取α_i:
  1. 首先选取违反 $0 < \alpha_i < C \Rightarrow y_i f(\boldsymbol{x}_i) = 1$ 的点
  2. 其次选择违反 $\alpha_i = 0 \Rightarrow y_i f(\boldsymbol{x}_i) \geq 1$ 的点和 $\alpha_i = C \Rightarrow y_i f(\boldsymbol{x}_i) \leq 1$ 的点.
- 选取α_j: 第二个变量的选择应当使得目标函数值减小最快

3.3.2 更新两个参数

- 求出

$$\alpha_j^{newunc} = \alpha_j^k + \frac{y_j(E_i - E_j)}{K_{ii} + K_{jj} - 2K_{12}}$$

- 为了保证其界限, 需要如下操作:

$$\alpha_j^{k+1} = \begin{cases} H & \alpha_j^{newunc} > H \\ \alpha_j^{newunc} & L \le \alpha_j^{newunc} \le H \\ L & \alpha_j^{newunc} < L \end{cases}$$

- 进而可以求出 α_i^{k+1}

3.3.3 更新b

- 按如下公式更新b:

$$b^{new} = \frac{b_i^{new} + b_j^{new}}{2}$$

其中

$$b_i^{new} = -E_i - y_i K_{ii}(\alpha_i^{new} - \alpha_i^{old}) - y_j K_{ji}(\alpha_j^{new} - \alpha_j^{old}) + b^{old}$$
$$b_j^{new} = -E_j - y_i K_{ij}(\alpha_i^{new} - \alpha_i^{old}) - y_j K_{jj}(\alpha_j^{new} - \alpha_j^{old}) + b^{old}$$

4 实验结果

4.1 总体对比

模型/算法	数据集	训练集准确度	测试集准确度	迭代次数
GD	s-svm	0.9571	0.9714	30
SGD	s-svm	0.9285	0.9714	100000
SMO	svm	0.7571	0.8642	200
sklearn(与SMO相同条件)	svm	0.7142	0.6667	-

可以看到就第一个简单的数据集(s-svm)而言, GD 算法 和 SGD 算法都能够达到非常好的表现. 而对于比较奇怪的数据集(svm)来说, GD/SGD算法表现不佳(未展示), 而 SMO 算法 则表现得好一些, 甚至好于同等条件下的sklearn(但未进行较好调参).

4.2 GD 算法实验结果

- 经过调整参数, 可以得到在**学习率为 0.01, epoch为30时**, 能够在**训练集上达到 0.9571 的准确率**, 在**测试集上达到 0.9714 的准确率**.

- 训练结果可视化:

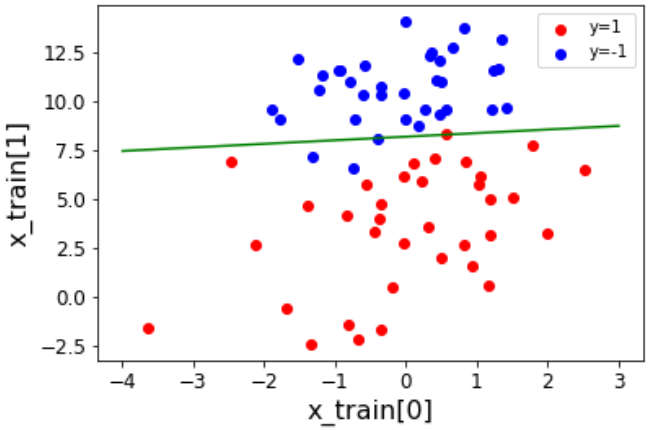

图 2: GD算法在训练集上的表现

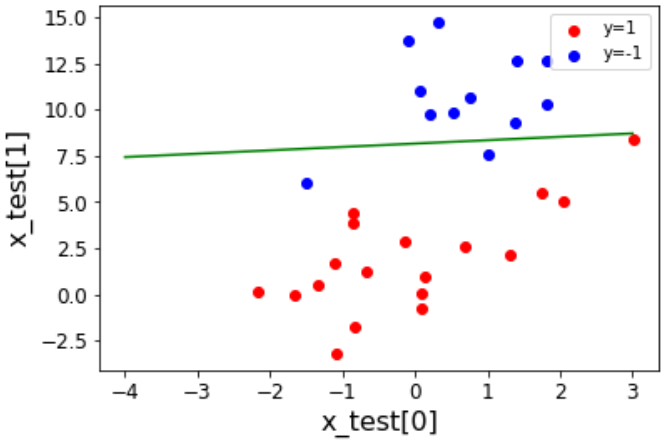

图 3: GD算法在测试集上的表现

4.3 SGD 算法实验结果

- 经过调整参数, 可以得到在**学习率为 0.001, epoch为 100000 时**, 能够在**训练集上达到 0.9285 的准确率**, 在**测试集上达到 0.9714 的准确率**.
- 训练结果可视化:

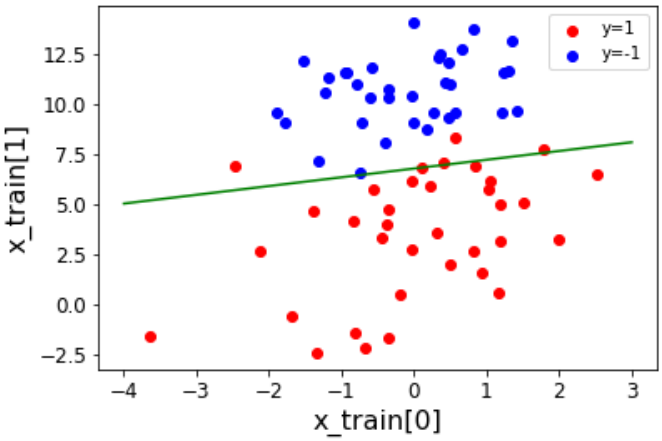

图 4: SGD算法在训练集上的表现

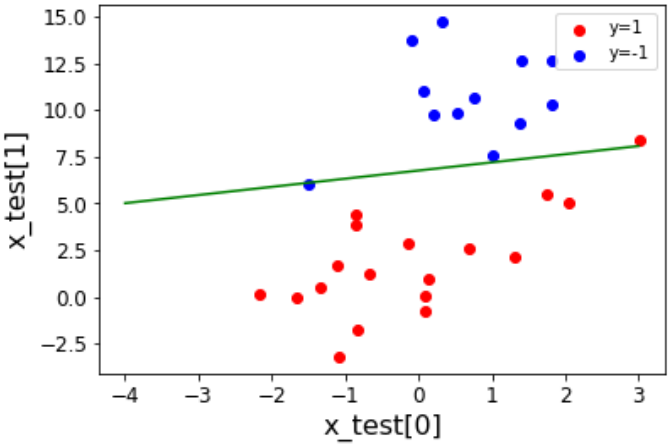

图 5: SGD算法在测试集上的表现

4.4 SMO 算法实验结果

- 经过调整参数, 可以得到在**epoch为 200 时**, 能够在**训练集上达到 0.7571 的准确率**, 在**测试集上达到 0.8642 的准确率**.
- 训练结果可视化:

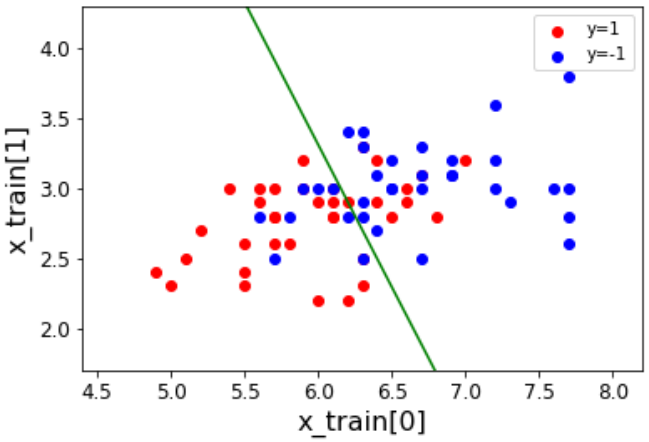

图 6: SMO 算法在训练集上的表现

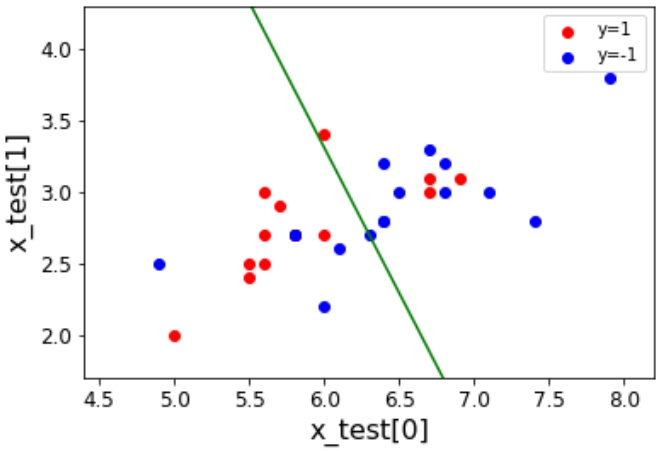

图 7: SMO 算法在测试集上的表现

5 实验总结

本次实验和上次实验中均发现对于数据集小的情况下, GD 算法 往往能够优于 SGD 算法(些微). 而本次实验中的 SMO 算法 对于小数据集也颇有杀鸡而用牛刀之意. 因此, 针对数据集不同, 我们应该选择合适的方法.