机器学习第六章支持向量机编程报告

162050127 颜劭铭

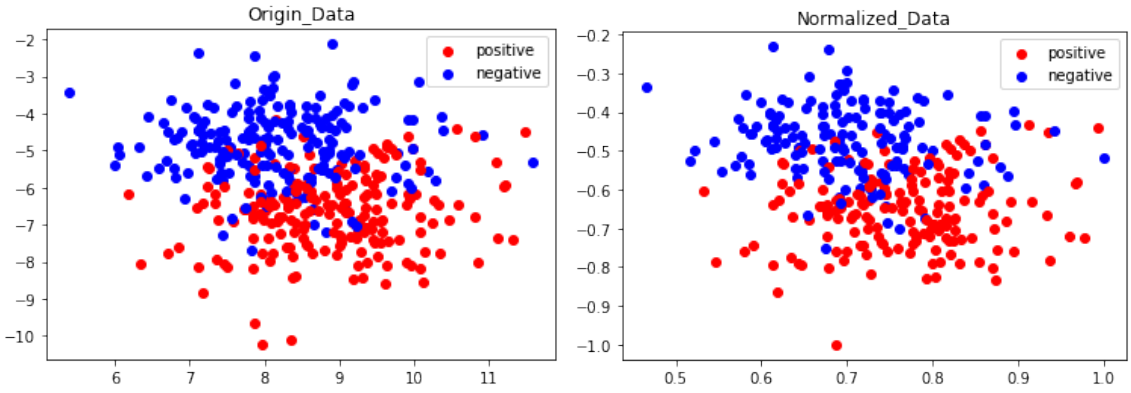
2022 年 5 月 24 日

1 用高斯核和线性核训练一个 SVM

1.1 数据处理

将 train_feature.csv 和 train_target.csv 文件中的数据处理为两个列表，其中 train_data 列表存储的是特征，并且是一个列表套列表的形式，train_feature 列表存储的是标签.

通过绘制原始的训练集数据图 1.a，我们可以发现，数据是线性不可分的，不适用于硬间隔的支持向量机，同时数据的变化太大，为了训练模型的效果更好，将数据归一化，得到图 1.b



(a) 原始数据图　　(b) 归一化数据图

图 1: 数据图

由于缺少验证集标签，所以为了方便后续验证，利用 train_test_split 函数，采用随机划分的方法，将训练集按 4:1 的比例划分成了训练集和验证集，但是因为这边并没有设置函数其中的参数 random_state，所以每一次划分出来的训练集和验证集都是随机的，所以准确率一直在变化.

1.2 公式原理

首先在我们的样本空间中，我们需要寻找用于划分的鲁棒性最强的超平面可以定义为：

$$w^T x + b = 0$$

由点到超平面的距离公式，可以得到我们的优化目标：

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2$$

而由图 1中的散点图，我们可以得知该问题需要使用软间隔的 SVM，软间隔的基本思想就是：允许模型犯错误，并不要求完全的线性可分，样本只在一定范围外记入错误，其他犯错的样本视而不见.

通过一系列的推导，我们可以推导出 SVM 模型的一般形式如下：

$$\min_f \Omega(f) + c\sum_{i=1}^{m} l(f(x_i), y_i)$$
$$\text{s.t.} \quad y_i(w^T x_i + b) \geqslant 1, i = 1, 2, \cdots, n$$

其中 Ω 指代结构风险，例如我们要求的间隔距离，c 指的是惩罚因子，l 指的是经验风险，用来描述模型与训练数据的契合程度，我们可以通过调整惩罚因子 c 来描述允许模型犯错误的程度，c 越小就是允许犯更多的错误.

1.3 线性核训练 SVM

通过调用 sklearn.svm 中的 LinearSVC 函数，并且调整惩罚因子，可以根据数据训练出线性核的 SVM，并对于划分的 xtest 特征验证集进行预测，与 ytest 标签进行对比得出验证集的准确度.

线性核指的是：

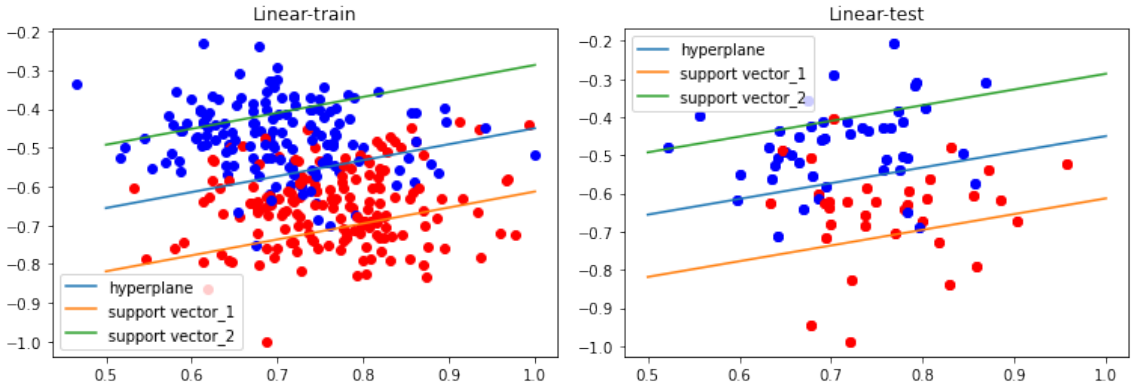
$$k\left(x_i, x_j\right) = x_i^T x_j$$

通过不断地调整惩罚因子，我最终选择惩罚因子为 10，并根据 sklearn.svm 中封装的函数 coef_ 和 intercept_ 得到方向向量 w 和截距项 b，并通过数学公式我们可以得到超平面和支撑向量的函数表达式，并通过 plt.plot 对其进行绘制.

下面给出计算得到的超平面与支撑向量的函数表达式：

$$
\begin{aligned}
\text{超平面:} \quad & y = -\frac{w[0]}{w[1]}x - \frac{b}{w[1]} \\
\text{支撑向量 1:} \quad & y = -\frac{w[0]}{w[1]}x - \frac{1-b}{w[1]} \\
\text{支撑向量 2:} \quad & y = -\frac{w[0]}{w[1]}x - \frac{-1-b}{w[1]}
\end{aligned}
$$

为了方便观察效果我绘制了训练集和验证集的散点图以及线性核 SVM 的超平面和支撑向量，如图 2.a 和图 2.b 所示，以及其中一次的验证结果，如图 3所示，不过要说明的是这个验证集准确度具有偶然性，大部分时候准确度稳定在 85% 上下..



(a) 训练集线性核　　(b) 验证集线性核

图 2: 线性核 SVM 训练集和验证集可视化

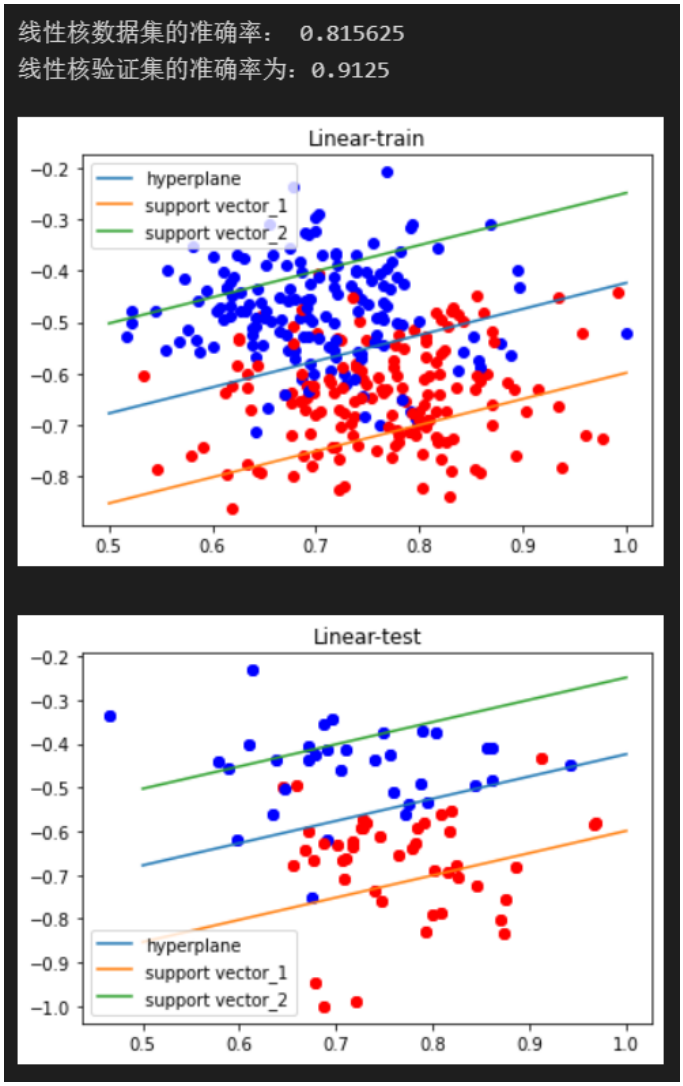


图 3: 线性核 SVM 结果

1.4 高斯核训练 SVM

通过调用 sklearn.svm 中的 SVC 函数，并且调整惩罚因子为 5，kernel 参数为 $'rbf'$，可以根据数据训练出线性核的 SVM，并对于划分的 xtest 特征验证集进行预测，与 ytest 标签进行对比得出验证集的准确度.

高斯核指的是：

$$k\left(x_i, x_j\right)=\exp\left(-\frac{\left\|x_i-x_j\right\|^2}{2\delta^2}\right)\ \delta>0 \text{ 为高斯核的带宽 (width)}$$

而对于高斯核我们可以对其进行泰勒展开为：

$$\exp(a)=\sum_{i=0}^{\infty}\frac{a^i}{i!}$$

因此高斯核可以把数据拓展到无限维，在无限维的时候数据是更有可能可分的，所以我认为高斯核的训练效果会比线性核训练的 SVM 效果好一点.

但是由于无限维我不知道该怎么画图，所以这里就只展示了高斯核训练出的 SVM 的数据集准确度和验证集准确度以及支撑向量的个数，如图 4所示，不过要说明的是这个验证集准确度具有偶然性，大部分时候准确度稳定在 85% 上下.

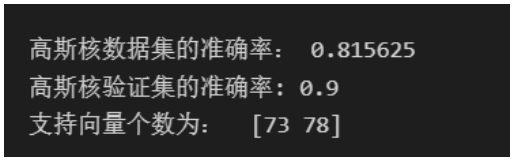


图 4: 高斯核 SVM 结果

2 与 BP 神经网络对比

SVM 与神经网络最大的一个不同点的话应该在于，比如说对于一个线性可分的问题，我们要求找出一个超平面用以将两种样本分类开。**由于我们可以找到无数条线能把这两类完全分开的，如果运用 BP 神经网络进行求解的话，我们会得到其中一个可行解，不一定是最优解，并且每次使用的初始点不一样的话，得到的结果不一定一样。而如果运用 SVM 做这个线性可分的问题的话，我们一定可以得到最优解的超平面，也就是在所有可行解中寻找一个最优解，因此我们可以用优化思想来理解 SVM。**

BP 神经网络是一种鲁棒性很强，具有非常强的非线性拟合能力的一种算法，并且在大样本上表现较好，但是它不具有可解释性，比如说用来解决分类问题的时候，它只能给出验证集的标签，也就是分类结果，不能像线性核 SVM 一样给出它的超平面方程，支持向量。

而相比起来 SVM 是一种有坚实理论基础的新颖的小样本学习方法，而其中它的支持向量决定了最终结果, 因为支持向量可以帮助我们可以抓住关键样本、“剔除”大量冗余样本, 而且注定了 SVM 不但算法简单, 而且具有较好的鲁棒性. 这种鲁棒性主要体现在:

①增、删非支持向量样本对模型没有影响;

②支持向量样本集具有一定的鲁棒性;

同时，SVM 具有较好的可解释性，支持向量，超平面方程便于我们理解 SVM 的训练结果，但是由于 SVM 是一个优化问题，求解的话需要涉及到矩阵的运算，当样本数量过大时，计算开销会非常大，所以说它较适用于小样本.

3 心得体会

编程实现 SVM 和编程实现单层感知机是两种不一样的体验，对于单层感知机的实现，由于对于他的数学原理了解较为清楚，所以一方面是锻炼自己的编程能力，另一方面是对于整个感知机涉及到的数学原理的再一次梳理，有了一个更加深刻的印象。

而实现 SVM 由于使用 sklearn 调库实现，所以对于底层代码并没有了解得非常多，也就是其中的数学原理，我知道 SVM 是通过将问题转化为一个优化问题的对偶形式，利用拉格朗日乘子法将其逐步迭代到满足 KKT 条件为止，最终求得所需参数。但是这次调库实现的话方便调参，因为惩罚因子 C 是一个经验参数，需要通过多次调整观察实验效果，并且可以多撰写一些可视化代码，通过观察可视化效果，也对于 SVM 的效果有了一个更深刻的了解。总的来说，是一次效果不错的编程体验，第一次实现了简单的 SVM，并且感受到了调库的方便。