浙江大学数学科学学院

前沿数学专题讨论 PR02

19 Oct 2022

Submitted To:
张南松老师
22-23 秋冬学期

Submitted By :
吴凡
农业工程 + 统计学

Contents

1　概述

1.1　简介

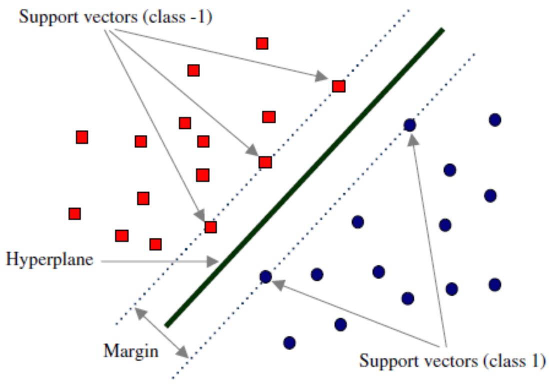
SVM，英文全称为 Support Vector Machine，中文名为支持向量机，由数学家 Vapnik 等人早在 1963 年提出。在深度学习兴起之前，SVM 一度风光无限，是机器学习近几十年来最为经典的，也是最受欢迎的分类方法之一。

1.2　为何叫支持向量机

SVM 的本质模型特征空间中最大化间隔的**线性分类器**，是一种二**分类模型**。

支持向量（Support vector）：离分类超平面（Hyper plane）最近的样本点。

其核心的理念是：支持向量样本会对识别的问题起关键作用

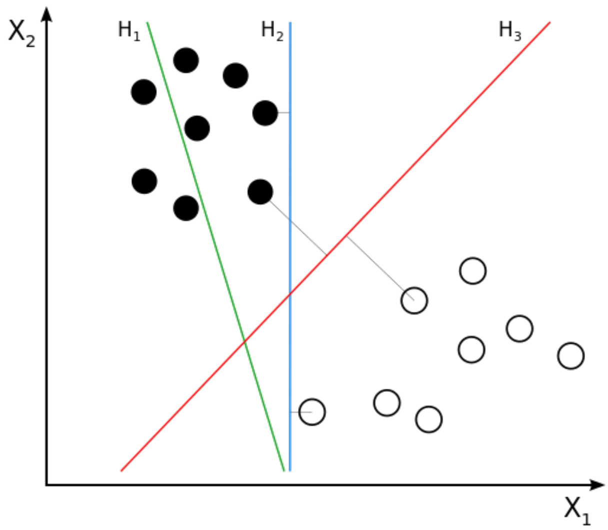


SVM 的学习策略就是间隔最大化。

1.3　直观理解

请看下图，图中有分别属于两类的一些二维数据点和三条直线。如果三条直线分别代表三个分类器的话，请问哪一个分类器比较好？

我们凭直观感受应该觉得答案是 H3。首先 H1 不能把类别分开，这个分类器肯定是不行的；H2 可以，但分割线与最近的数据点只有很小的间隔，如果测试数据有一些噪声的话可能就会被 H2 错误分类 (即对噪声敏感、泛化能力弱)。H3 以



较大间隔将它们分开，这样就能容忍测试数据的一些噪声而正确分类，是一个泛化能力不错的分类器。

对于支持向量机来说，数据点若是 p 维向量，我们用 p-1 维的超平面来分开这些点。但是可能有许多超平面可以把数据分类。最佳超平面的一个合理选择就是以最大间隔把两个类分开的超平面。因此，SVM 选择能够使离超平面最近的数据点的到超平面距离最大的超平面。

以上介绍的 SVM 只能解决线性可分的问题，为了解决更加复杂的问题，支持向量机学习方法有一些由简至繁的模型:

- **线性可分 SVM**: 当训练数据线性可分时，通过硬间隔 (hard margin，什么是硬、软间隔下面会讲) 最大化可以学习得到一个线性分类器，即硬间隔 SVM，如上图的的 H3。
- **线性 SVM**: 当训练数据不能线性可分但是可以近似线性可分时，通过软间隔 (soft margin) 最大化也可以学习到一个线性分类器，即软间隔 SVM。
- **非线性 SVM**: 当训练数据线性不可分时，通过使用核技巧 (kernel trick) 和软间隔最大化，可以学习到一个非线性 SVM。

2 线性可分 SVM——hard margin

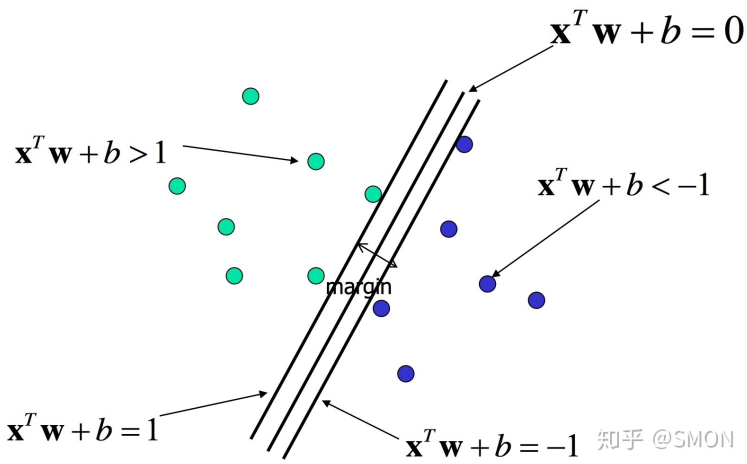
‘

考虑如下形式的线性可分的训练数据集：$(X_1, y_1), (X_2, y_2), \cdots, (X_N, y_N)$

$y_i \in \{+1, -1\}$,$y_i = +1$ 时表示 X_i 属于正类别，$y_i = -1$ 时表示 X_i 属于负类别。

2.1 超平面与间隔

超平面方程为 $W^T X + b = 0$，可以规定法向量指向的一侧为正类, 另一侧为负类。下图画出了三个平行的超平面，法方向取左上方向。



为了找到最大间隔超平面，我们可以先选择分离两类数据的两个平行超平面，使得它们之间的距离尽可能大。在这两个超平面范围内的区域称为“间隔 (margin)”，最大间隔超平面是位于它们正中间的超平面。这个过程如上图所示。

2.2 数学模型

判别模型：$y_i = sign(W^T X_i + b)$

模型判别正确 $\Leftrightarrow \begin{cases} W^T X_i + b > 0, y_i = +1 \\ W^T X_i + b < 0, y_i = -1 \end{cases} \Leftrightarrow y_i(W^T X_i + b) > 0, i =$

$1, 2, \ldots, N$

margin(w,b)=min_{w,b,X_i}distance(w,b,X_i)= $min_{w,b,X_i} \frac{\left|w^T X_i + b\right|}{\|w\|}$

SVM 的中心思想：找到最大间隔超平面，即使 margin 取到最大值。则优化问题可用如下数学模型表示：

$$\begin{cases} max_{w,b} \frac{1}{\|w\|} \\ s.t. y_i(w^T X_i + b) \geq 1, i = 1, 2, \ldots, N \end{cases} \tag{1}$$

$$\Leftrightarrow \begin{cases} min_{w,b} \frac{1}{2} w^T w \\ s.t. 1 - y_i(w^T X_i + b) \leq 0, i = 1, 2, \ldots, N \end{cases} \tag{2}$$

问题转化为纯粹的凸优化问题，用拉格朗日乘子法求最优解。

$$L(w, b, \lambda) = \frac{1}{2} w^t w + \sum_{i=1}^{N} \lambda_i \left[1 - y_i(w^t X_i + b)\right] \tag{3}$$

$$\Rightarrow \begin{cases} min_{w,b} max_{\lambda} L(w, b, \lambda) \\ s.t. \lambda_i \geq 0, i = 1, 2, \ldots, N \end{cases} \tag{4}$$

引入本拉格朗日方程的对偶方程。我们称上式所述问题为原始问题 (primal problem), 可以应用拉格朗日乘子法构造拉格朗日函数 (Lagrange function) 再通过求解其对偶问题 (dual problem) 得到原始问题的最优解。转换为对偶问题来求解的原因是:

1. 对偶问题更易求解，由下文知对偶问题只需优化一个变量且约束条件更简单；
2. 能更加自然地引入核函数，进而推广到非线性问题。

对偶问题：

$$\begin{cases} max_{\lambda} min_{w,b} L(w, b, \lambda) \\ s.t. \lambda_i \geq 0, i = 1, 2, \ldots, N \end{cases} \tag{5}$$

$$\frac{\partial L}{\partial b}=\frac{\partial \frac{1}{2}w^t w+\sum_{i=1}^{N}\lambda_i\left[1-y_i(w^t X_i+b)\right]}{\partial b}=-\sum_{i=1}^{N}\lambda_i y_i=0 \tag{6}$$

$$\Rightarrow \sum_{i=1}^{N}\lambda_i y_i=0 \tag{7}$$

带入原式 $L(w,b,\lambda)$ 中，则

$$L(w,b,\lambda)=\frac{1}{2}w^t w+\sum_{i=1}^{N}\lambda_i-\sum_{i=1}^{N}\lambda_i y_i w^T X_i \tag{8}$$

$$\frac{\partial L}{\partial w}=\frac{1}{2}w-\sum_{i=1}^{N}\lambda_i y_i X_i=0 \tag{9}$$

$$\Rightarrow w^*=\sum_{i=1}^{N}\lambda_i y_i X_i \tag{10}$$

将 w^* 带入原式 $L(w,b,\lambda)$ 中，

$$\begin{array}{l} L(w,b,\lambda)=-\frac{1}{2}(\sum_{i=1}^{N}\lambda_i y_i X_i)^T(\sum_{j=1}^{N}\lambda_j y_j X_j)+\sum_{i=1}^{N}\lambda_i \\ =-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j X_i^T X_j+\sum_{i=1}^{N}\lambda_i \end{array} \tag{11}$$

因而，可通过逐项求导，求出 λ_i 的值。

3 算法实例

3.1 鸢尾花分类问题背景

一名植物学爱好者收集了鸢尾花的一些测量数据：花瓣的长度和宽度以及花萼的长度和宽度。他还有一些鸢尾花分类的测量数据，这些花已经被植物学专家鉴定为属于 versicolor 、setosa 或 virginica 三个品种之一。

数据集主要属性：

	萼片长度	萼片宽度	花瓣长度	花瓣宽度	类型
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

每条记录包含 5 项基本信息：花萼的长度、花萼的宽度、花瓣的长度、花瓣的宽度以及鸢尾花的类别。

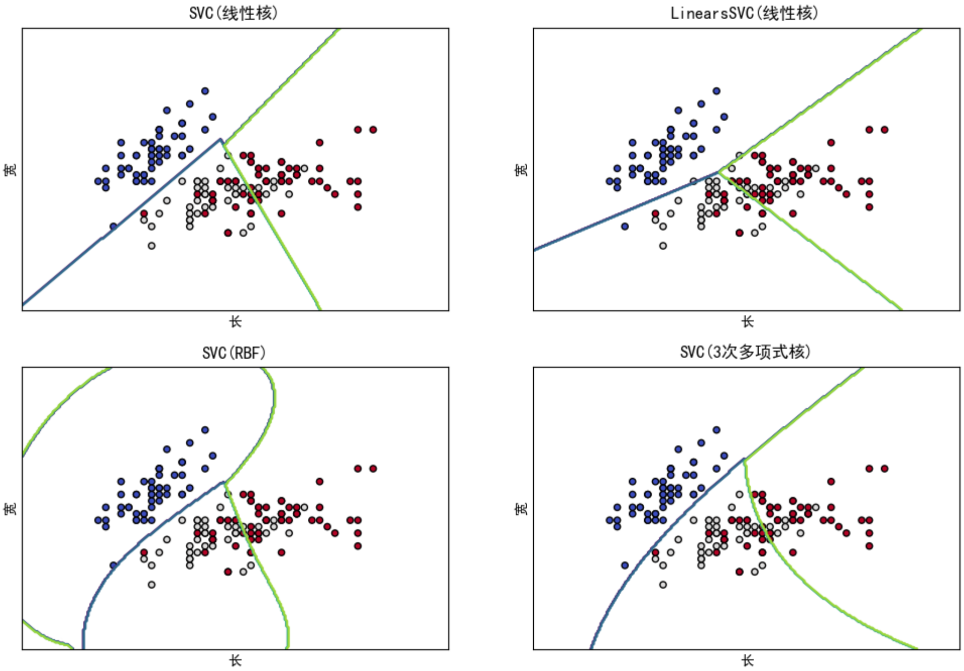
3.2 SVM 分类

现采用 Python sklearn 中的 SVM 模块对鸢尾花数据集进行分类。

采用的 SVM 分类方法分别为:

- SVC(线性核)
- Linears(线性核)
- SVC(RBF)
- SVC(3 次多项式核)

程序分类可视化结果如下：



3.3 代码

```
import numpy as np
import matplotlib.pyplot as pl
from sklearn import svm,datasets
from pylab import mpl

mpl.rcParams['font.sans-serif']='SimHei' #画图正常显示中文
mpl.rcParams['axes.unicode_minus']=False #决绝保存图像是负号‘-’显示方
                                          块的问题

def make_meshgrid(x,y,h=.02):
    """准备用于绘图的网格点
    参数
    ——————————
    x:x轴数据点
    y: y轴数据点
```

```
    h: 间隔距离
    返回值
    ———————————
    xx, yy: ndarray
    """
    x_min,x_max=x.min()-1,x.max()+1
    y_min,y_max=y.min()-1,y.max()+1
    xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
                      np.arange(y_min,y_max,h))
    return xx,yy


def plot_contours(ax, clf, xx, yy, **params):
    """绘制分类器边界
    参数
    —————————
    ax: matplotlib.axes对象
    xx: 网格点
    yy: 网格点
    params: 控制绘图的其它字典
    """

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contour(xx, yy, Z, **params)
    return out


# 载入鸢尾花数据集
iris = datasets.load_iris()
# 为了后面方便绘图, 这里只使用二个特征
X = iris.data[:, :2]
y = iris.target

C = 1.0
# 备用的各种模型设置
models = (svm.SVC(kernel='linear', C=C),
          svm.LinearSVC(C=C),
```

```
          svm.SVC(kernel='rbf', gamma=0.7, C=C),
          svm.SVC(kernel='poly', degree=3, C=C))
# 训练模型
models = (clf.fit(X, y) for clf in models)
# 各模型标题
titles = (u'SVC(线性核)',
          u'LinearsSVC(线性核)',
          u'SVC(RBF)',
          u'SVC(3次多项式核)')

# 把整个图划分成2*2网格
fig, sub = pl.subplots(2, 2, figsize=(12, 8))
pl.subplots_adjust(wspace=0.2, hspace=0.2)

X0, X1 = X[:, 0], X[:, 1]
xx, yy = make_meshgrid(X0, X1)

for clf, title, ax in zip(models, titles, sub.flatten()):
    plot_contours(ax, clf, xx, yy, alpha=0.8)
    ax.scatter(X0, X1, c=y, cmap=pl.cm.coolwarm, s=20, edgecolors='k')
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xlabel(u'长')
    ax.set_ylabel(u'宽')
    ax.set_xticks(())  # 不显示坐标
    ax.set_yticks(())  # 不显示坐标
    ax.set_title(title)

pl.show()
```