支持向量机

Created	@March 28, 2022
Created by	
Tags	Home

算法原理

间隔与支持向量机

给定训练集样本$D=\{(x_1,y_1),...,(x_m,y_m)\},y_i\in\{-1,+1\}$,分类学习最基本的思想就是基于训练集D在样本空间找到一个划分超平面，将不同类别的样本分开。但是能够将样本分开的超平面可能有很多，要找到一个对训练样本局部扰动的“容忍”性最好的那个超平面。

从几何的角度，对于线性可分的数据集，支持向量机就是找距离正负样本都最远的超平面，相对于感知机，其解释唯一的，且不偏不倚，泛化性能好。

支持向量，support vector，两个异类支持向量到超平面的距离之和为

$$\gamma=\frac{2}{||\omega||}$$

这个距离被称为“间隔”（margin），**支持向量机就是要找到具有最大间隔**(maximum margin)的划分超平面，也就是找到约束的参数ω和b，使得γ最大，数学公式转化为：

$$max_{\omega,b}\frac{2}{||\omega||}$$

$$s.t.y_i(w'x_i+b)\geq 1,i=1,2,...,m.$$

为了**最大化间隔**，只需要最大化$||\omega||^{-1}$,等价于最小化$||\omega||^2$，所以支持向量机（Support Vector Machine， SVM)的基本型就是：

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

$$s.t. y_i(w'x_i + b) \geq 1, i = 1, 2, ..., m.$$

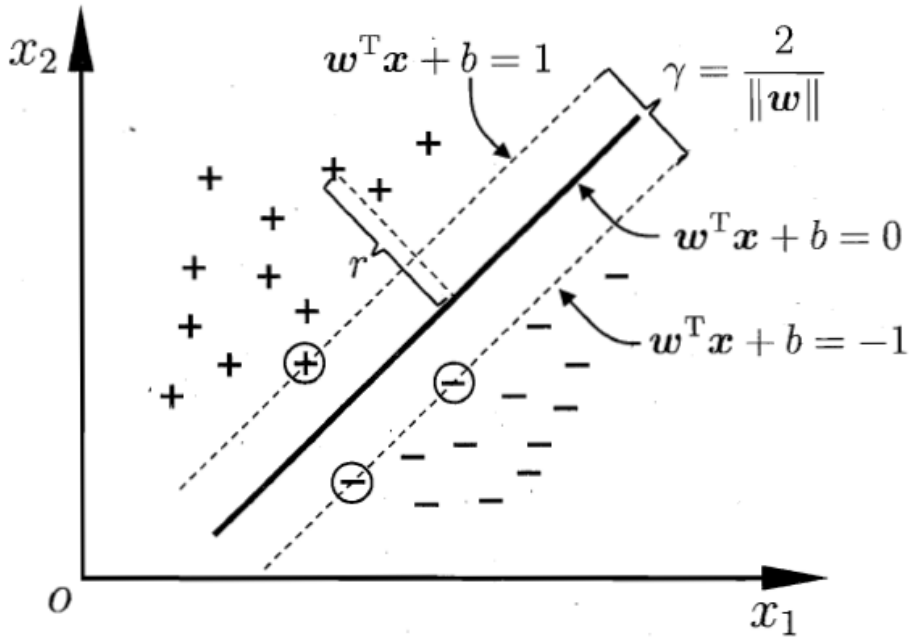


图 6.2　支持向量与间隔

超平面

n微空间的超平面（$\omega' x + b = 0, 其中 \omega, x \in R^n$）：

- 超平面方程不唯一
- 法向量ω和位移项b确定一个位移的超平面
- 法向量ω垂直于超平面，缩放ω和b时，若缩放倍数为负数会改变法向量的方向
- 法向量指向的那一半空间为正空间，另一半为负空间
- 任一点x到超平面的距离公式为$r = \frac{|\omega' x + b|}{||\omega||}$

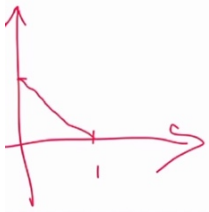
超平面

Datawhale

【证明】：对于任意一点$\boldsymbol{x}_0 = (x_1^0, x_2^0, ..., x_n^0)^{\mathrm{T}}$，设其在超平面$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0$上的投影点为$\boldsymbol{x}_1 = (x_1^1, x_2^1, ..., x_n^1)^{\mathrm{T}}$，则$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_1 + b = 0$，且向量$\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}$与法向量$\boldsymbol{w}$平行，因此

$$|\boldsymbol{w} \cdot \overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}| = |\|\boldsymbol{w}\| \cdot \cos\pi \cdot \|\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}\|| = \|\boldsymbol{w}\| \cdot \|\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}\| = \|\boldsymbol{w}\| \cdot r$$

$$\begin{aligned}\boldsymbol{w} \cdot \overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0} &= w_1(x_1^0 - x_1^1) + w_2(x_2^0 - x_2^1) + ... + w_n(x_n^0 - x_n^1) \\ &= w_1x_1^0 + w_2x_2^0 + ... + w_nx_n^0 - (w_1x_1^1 + w_2x_2^1 + ... + w_nx_n^1) \\ &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_0 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_1 \\ &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_0 + b\end{aligned}$$



$$|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_0 + b| = \|\boldsymbol{w}\| \cdot r \Rightarrow r = \frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b|}{\|\boldsymbol{w}\|}$$

内积就是点乘，向量点乘 a·b = ||a|| · cosθ · ||b||

几何间隔

对于给定的数据集D超平面$\omega' x + b = 0$ 定义数据集D中的任意一个样本点$\{(x_1, y_1), ..., (x_m, y_m)\}, y_i \in \{-1, +1\}$, 关于超平面的几何间隔为：

$$\gamma_i = \frac{y_i(\omega' x_i + b)}{||\omega||}$$

正确分类时，$\gamma_i > 0$，几何间隔此时也等价于点到超平面的距离

没有正确分类时，$\gamma_i < 0$

在给定数据就和超平面下，定义数据集关于超平面的几何间隔为：微数据D中所有样本点的几何间隔的最小值，也就是$\gamma = min_{(i=1,2,...m)}\gamma_i$h

核函数

假设训练样本是线性可分的，即存在一个划分超平面能够将训练样本正确分类，现实任务中可能并不存在这样的超平面。对于这样的问题，可以将样本从原始空间映射到一个更高维的特征空间，使得样本在这个特征空间内线性可分。**如果原始空间是有限维，那一定存在一个高位特征空间线性划分样本。**

核函数是一个设想能够计算高维空间样本映射到特征空间之后的特征样本的内积的函数。通常用$k(\cdot)$表示核函数。**只要一个对称函数所对应的核矩阵是半正定，它就能作**

为核函数使用。

表 6.1 列出了几种常用的核函数.

表 6.1 常用核函数

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	$d \geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	tanh 为双曲正切函数, $\beta > 0, \theta < 0$

软间隔

现实中很难确定合适的核函数使得训练样本在特征空间中线性可分，即使找到了，也很难断定这个貌似线性可分的结果不是由于过拟合造成的。于是有了允许向量机在一些样本上出错的缓解方法。这个就是“软间隔soft margin”的概念

surrogate loss:替代损失函数 $l_{0/1}$是0/1损失函数。

hinge 损失：$l_{hinge}(z) = max(0, 1 - z)$
指数损失(exponential loss): $l_{exp}(z) = exp(-z)$
对率损失(logistic loss): $l_{log}(z) = log(1 + exp(-z))$

松弛变量（slack variables） $\xi_i \geq 0$

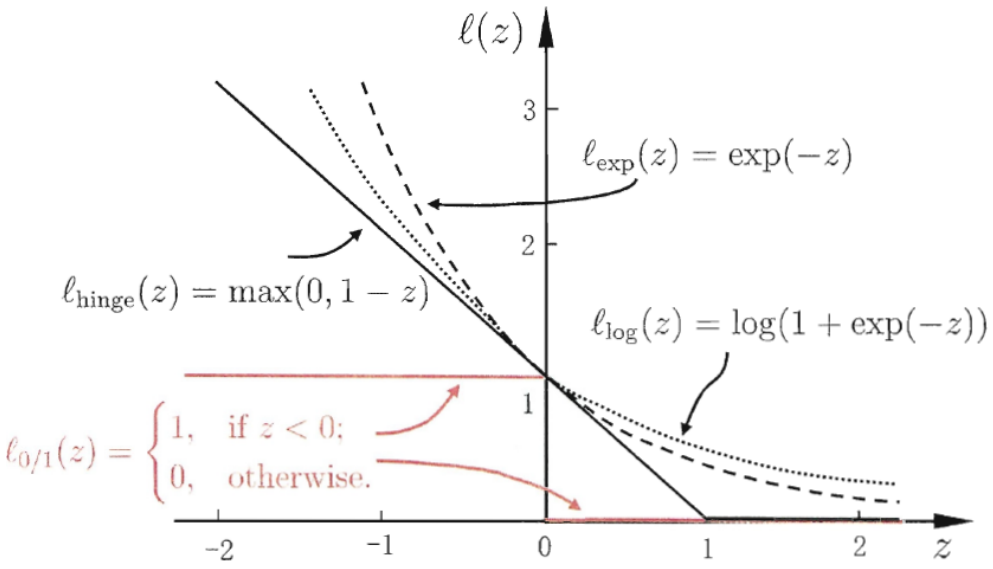


图 6.5 三种常见的替代损失函数: hinge损失、指数损失、对率损失

支持向量机

▼ 模型：给定线性可分数据集X，支持向量机模型希望求得数据集X关于超平面的几何间隔γ达到最大的那个超平面，然后套上一个sign函数实现分类功能：

$$y = sign(\omega' x + b) = \begin{cases} 1, & \omega' x + b > 0 \\ -1, & \omega' x + b < 0 \end{cases}$$

本质上仍然是在求一个超平面。当超平面没有正确划分正负样本室，集合间隔最小的为误分类点，所以γ<0,当超平面划分正确时，γ≥0，且越靠近中央，γ越大。

▼ 策略：给定线性可分的数据集X， 设X中级和间隔最小样本为(x_{min}, y_{min})，那么支持向量机找超平面的过程可以转化为带约束条件的优化问题.

$max\ \gamma\ s.t.\ \gamma_i \geq \gamma, i = 1, 2, ..., m$

最终优化问题就转化为了凸优化问题：

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

$$s.t.\ 1 - y_i(w' x_i + b) \leq 0, i = 1, 2, ..., m$$

在支持向量机通常采用拉格朗日对偶来求解这个凸优化问题。目标函数$\frac{1}{2}||w||^2$是关于ω的凸函数，不等式约束$1 - y_i(w' x_i + b)$也是关于ω的凸函数，所以说支

持向量是一个凸函数问题。

正定矩阵/半正定矩阵都是凸函数

对偶问题Dual Problem

凸二次规划（convex quadratic programming)

> 对于一般约束优化问题：
>
> $min\ f(x)$
>
> $s.t.\ g_i(x) \leq 0, i = 1, 2, ...m$
> $\quad\quad h_j(x) = 0, j = 1, 2, ..., n$
>
> 上述的优化问题的定义域$D = dom\ f \cap \cap_{i=1}^{m}\ dom\ g_i \cap \cap_{j=1}^{n}\ dom\ h_j$，可行集为$\tilde{D} = \{x|x \in D, g_i(x) \leq 0, h_j(x) = 0\}$，显然$\tilde{D}$是$D$的子集。最优解$p^* = min\{f(\tilde{x})\}$。有拉格朗日函数的定义可知上述优化问题的拉格朗日函数为
>
> $$L(x, \mu, \lambda) = fa(x) = \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{j=1}^{n} \lambda_j h_j(x)$$
>
> 其中μ、λ为拉格朗日乘子。
>
> > 拉格朗日对偶函数$\Gamma(\mu, \lambda)$，只和μ、λ有关。是$L(x, \mu, \lambda)$关于x的下确界。
> >
> > 对偶函数有如下重要性质：
> >
> > - 无论上述优化问题是否是凸优化问题，其对偶函数恒为凹函数

> - 当μ≥0时，$\Gamma(\mu,\lambda)$构成了上述优化问题最优值p^*的下界，也就是$\Gamma(\mu,\lambda) \leq p^*$（证明略）。

SVM的核技巧

核技巧是为了解决线性SVM无法进行多分类以及SVM在一些线性不可分的情况下无法分类的情况。核技巧就是将数据用一个核函数进行转换，最终得到结果。

▼ **sklearn-SVM参数，kernel特征选择**

▼ kernel：核函数选择，字符串类型，可选的
有“linear”，“poly”，“rbf”，“sigmoid”，“precomputed”以及自定义的核函数，默认选择是“rbf”。

▼ 各个核函数介绍如下：

▼ “linear”：线性核函数，最基础的核函数，计算速度较快，但无法将数据从低维度演化到高维度

▼ “poly”：多项式核函数，依靠提升维度使得原本线性不可分的数据变得线性可分

▼ “rbf”：高斯核函数，这个可以映射到无限维度，缺点是计算量比较大

▼ “sigmoid”：Sigmoid核函数，对，就是逻辑回归里面的那个Sigmoid函数，使用Sigmoid的话，其实就类似使用一个一层的神经网络

▼ “precomputed”：提供已经计算好的核函数矩阵，sklearn不会再去计算，这个应该不常用“自定义核函数”：

▼ sklearn会使用提供的核函数来进行计算说这么多，那么给个不大严谨的推荐吧样本多，特征多，二分类，选择线性核函数样本多，特征多，多分类，多项式

核函数样本不多，特征多，二分类/多分类，高斯核函数样本不多，特征不多，二分类/多分类，高斯核函数

▼ 当然，正常情况下，一般都是用交叉验证来选择特征，上面所说只是一个较为粗浅的推荐。