

# Machine Learning Note

## 目录

<b>1 模型评估与选择</b>	<b>1</b>
1.1 经验误差与过拟合 . . . . .	1
1.2 评估方法 . . . . .	1
1.2.1 留出法 . . . . .	1
1.2.2 交叉验证法 . . . . .	1
<b>2 支持向量机</b>	<b>2</b>
2.1 间隔与支持向量 . . . . .	2
2.2 对偶问题 . . . . .	3
2.3 核函数 . . . . .	5
2.4 软间隔与正则化 . . . . .	7
2.5 支持向量回归 . . . . .	11
2.6 核方法 . . . . .	14
2.7 支持向量机与多分类问题 . . . . .	14
2.7.1 Matlab LibSVM . . . . .	14
<b>3 决策树</b>	<b>20</b>
3.1 决策树的构造 . . . . .	20
3.1.1 信息增益 . . . . .	21
3.1.2 划分数据集 . . . . .	21
3.1.3 构建递归决策树 . . . . .	21
<b>4 Bagging 与随机森林</b>	<b>21</b>

## 1. 模型评估与选择

### 1.1 经验误差与过拟合

学习器在训练集上的误差称为“训练误差”或“经验误差”(empirical error), 在新样本上的误差称为“泛化误差”(generalization error).

“过拟合”: 训练样本学的太好, 泛化性能下降 /quad 是机器学习的关键障碍, 无法彻底避免

“欠拟合”: 是指对训练样本的一般性质尚未学好

### 1.2 评估方法

现实中要考虑时间开销, 存储开销, 可解释性等方面因素, 这里只考虑泛化误差.

#### 1.2.1 留出法

直接将数据集 D 划分为两个互斥的集合, 训练集 S 和测试集 T. 训练/测试集的划分要尽量保持数据分布的一致性.

单次使用留出法得到的估计结果不够稳定可靠, 一般采用多次随机划分, 重复试验取平均值作为评估结果.

#### 1.2.2 交叉验证法

先将数据集 D 划分成 k 个 (k 折) 大小相似的互斥子集, 每个子集都尽可能保持数据分布的一致性, 即从 D 中通过分层采样得到. 然后每次用  $k-1$  个子集的并集作为训练集, 余下子集作为测试集, 进行 k 次训练和测试, 最终返回的是 k 个测试结果的均值.

为减小因样本划分不同而引入的差别,k 折要随机用不同的划分重复 p 次, 最终评估结果是 p 次 k 折交叉验证结果的均值.

假定 D 包含 m 个样本, 若令  $k=m$ , 得到特例: 留一法 (Leave-One-Out, LOO)

## 2. 支持向量机

### 2.1 间隔与支持向量

机器学习中可以用于分类或者回归问题. 这里主要是分类.

超平面：二维空间的超平面是一条直线，三维空间的超平面是一个平面，而 N 维空间的超平面则是 N-1 维的仿射空间。

超平面性质：将空间分成两部分，一部分大于 0，一部分小于 0。

凸二次规划问题：<https://zhidao.baidu.com/question/281998847.html>

样本空间中划分超平面：

$$\omega^T \mathbf{x} + b = 0 \quad (2.1)$$

$\omega = \{\omega_1; \omega_2; \dots; \omega_d\}$  为法向量，决定超平面的方向； $b$  为位移项，决定超平面与原点之间的距离。

样本空间任一点  $\mathbf{x}$  到超平面  $(\omega, b)$  的距离：

$$r = \frac{|\omega^T \mathbf{x} + b|}{\|\omega\|} \quad (2.2)$$

设超平面  $(\omega, b)$  能将训练样本正确分类，即对于超平面  $(\mathbf{x}_i, y_i) \in D$ ，若  $y_i = +1$ ，则有  $\omega^T \mathbf{x} + b > 0$ ；若  $y_i = -1$ ， $\omega^T \mathbf{x} + b < 0$ 。另

$$\begin{cases} \omega^T \mathbf{x}_i + b \geqslant +1 & y_i = +1 \\ \omega^T \mathbf{x}_i + b \leqslant -1 & y_i = -1 \end{cases} \quad (2.3)$$

注：若存在超平面  $(\omega', b')$  能将训练样本正确分类，则总存在缩放变换  $\zeta \omega \mapsto \omega'$  和  $\zeta b \mapsto b'$  使式2.3成立。

如图1，距离超平面最近的几个样本点使式2.3的等号成立，每个样本点对应一个特征向量，他们被称为“支持向量”，两个异类支持向量到超平面的距离和：

$$\gamma = \frac{2}{\|\omega\|} \quad (2.4)$$

称为“间隔”(margin)(这里的“间隔”指几何间隔)。

欲找到具有“最大间隔”(maximum margin) 的划分超平面，即

$$\begin{aligned} & \max_{\omega, b} \quad \frac{2}{\|\omega\|} \\ & s.t. \quad y_i(\omega^T \mathbf{x}_i + b) \geqslant 1, \quad i = 1, 2, \dots, m. \end{aligned} \quad (2.5)$$

$b$  通过约束隐式地影响  $\omega$ ，所以间隔与  $b$  和  $\omega$  都有关。

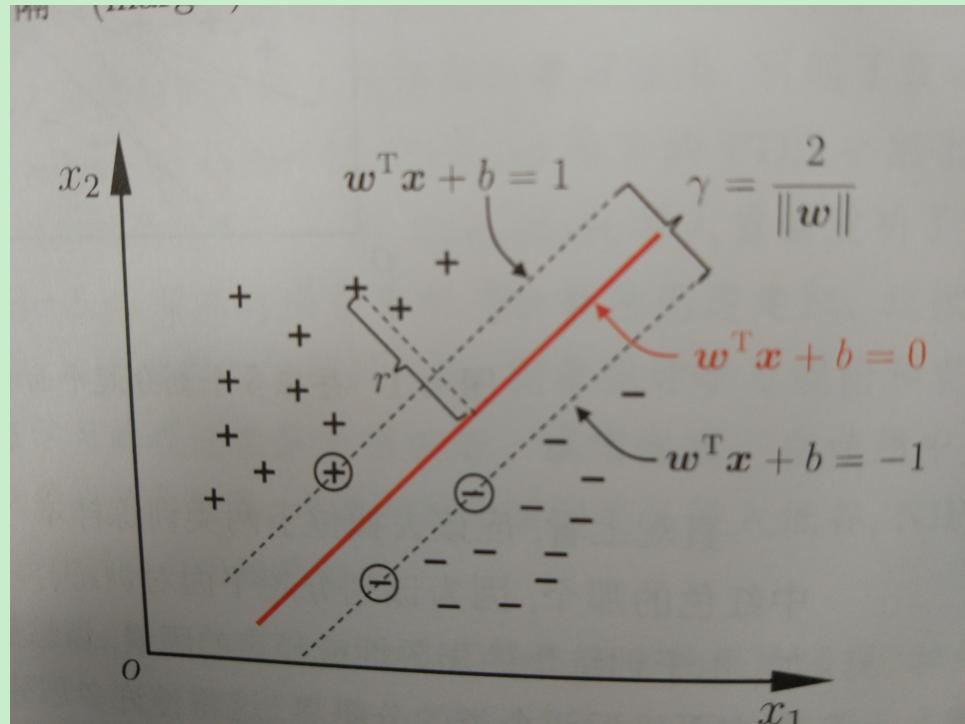


图 1: 支持向量与间隔

式2.5可以重写为:

$$\begin{aligned} & \min_{\omega, b} \frac{\|\omega\|^2}{2} \\ \text{s.t. } & y_i(\omega^T x_i + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned} \tag{2.6}$$

即支持向量机 (SVM) 的基本型.

## 2.2 对偶问题

希望通过解式2.6得到大间隔划分超平面对应的模型

$$f(x) = \omega^T x + b \tag{2.7}$$

对式2.6使用拉格朗日乘子法可得到其“对偶问题”(线性规划有一个有趣的特性，就是任何一个求极大的问题都有一个与其匹配的求极小的线性规划问题). 对偶问题可以更高效地求解模型参数  $\omega, b$ , 对2.6每条约束添加拉格朗日乘子  $\alpha_i \geq 0$ , 拉格朗日函数可写为

$$L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}) = \frac{\|\boldsymbol{\omega}\|^2}{2} + \sum_{i=1}^m \alpha_i (1 - y_i (\boldsymbol{\omega}^T \mathbf{x}_i + b)) \quad (2.8)$$

$\boldsymbol{\alpha} = (\alpha_1; \alpha_2; \dots; \alpha_m)$ , 令  $L(\boldsymbol{\omega}, b, \boldsymbol{\alpha})$  对  $\boldsymbol{\omega}$  和  $b$  的偏导为 0 得

$$\boldsymbol{\omega} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \quad (2.9)$$

$$0 = \sum_{i=1}^m \alpha_i y_i \quad (2.10)$$

2.9代入2.8, 考虑2.10的约束, 就可得到2.6的对偶问题

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \\ & \alpha_i \geq 0, \quad i = 1, 2, \dots, m \end{aligned} \quad (2.11)$$

(这里的对偶问题式子里没出现  $b$ , 貌似因为式2.6里显示间隔只和  $\boldsymbol{\omega}$  有关, 所以这里把  $b$  那项忽略了)

解出  $\boldsymbol{\alpha}$ , 求出  $\boldsymbol{\omega}$  与  $b$  即可得到模型

$$f(\mathbf{x}) = \boldsymbol{\omega}^T \mathbf{x} + b = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b \quad (2.12)$$

因式2.6中有不等式约束, 上述过程需满足 KKT(Karush-Kuhn-Tucker) 条件, 即要求

$$\begin{cases} \alpha_i \geq 0; \\ y_i f(\mathbf{x}_i) - 1 \geq 0; \\ \alpha_i (y_i f(\mathbf{x}_i) - 1) = 0. \end{cases} \quad (2.13)$$

(关于 KKT 条件和拉格朗日乘子的详细解释: <http://blog.csdn.net/xianlingmao/article/details/7919597>)

于是, 对于任意训练样本  $(\mathbf{x}_i, y_i)$ , 总有  $\alpha_i = 0$  或  $y_i f(\mathbf{x}_i) = 1$ . 若  $\alpha_i = 0$ , 则该样本不会在式2.12中出现, 也就不会对  $f(\mathbf{x})$  产生影响; 若  $\alpha_i > 0$ , 则必有  $y_i f(\mathbf{x}_i) = 1$ , 所对应的样本点位于最大间隔边界

上, 是一个支持向量. 因此, 训练完成后, 大部分的训练样本都不需要保留, 最终模型仅与支持向量有关. 此外, 求解式2.11, 可以用 SMO 算法.(见 svm 程序部分)

Sequential Minimal Optimization (SMO) 基本思路: 先固定  $\alpha_i$  以外的所有参数, 然后求  $\alpha_i$  上的极值. 由于存在约束  $\sum_{i=1}^m \alpha_i y_i = 0$ , 若固定  $\alpha_i$  以外的其他变量,  $\alpha_i$  可由其他变量导出.

在参数初始化后, SMO 不断执行如下两个步骤直至收敛:

- 选取一对需更新的变量  $\alpha_i$  和  $\alpha_j$  (为了满足约束条件  $\sum_{i=1}^m \alpha_i y_i = 0$ , 所以每次需更新一对)
- 固定  $\alpha_i$  和  $\alpha_j$ , 求解式2.11获得更新后的  $\alpha_i$  和  $\alpha_j$ .

SMO 的变量选取方式: 使选取的两变量所对应样本之间的间隔最大.

偏移项  $b$  的确定: 对任意支持向量  $(\mathbf{x}_s, y_s)$  都有  $y_s f(\mathbf{x}_s) = 1$ , 即

$$y_s \left( \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s + b \right) = 1 \quad (2.14)$$

$S = \{i \mid \alpha_i > 0, i = 1, 2, \dots, m\}$  为所有支持向量的下标集. 理论上, 可选任意支持向量并通过式2.14获得  $b$ , 实际中:

$$b = \frac{1}{|S|} \sum_{s \in S} \left( y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s \right) \quad (2.15)$$

## 2.3 核函数

不是所有的原始样本空间都存在能正确划分两类样本的超平面. 可将样本从原始空间映射到一个更高维的特征空间. 如果原始空间是有限维, 那么一定存在一个高维特征空间使样本可分.

令  $\Phi(\mathbf{x})$  表示将  $\mathbf{x}$  映射后的特征向量, 在特征空间中划分超平面所对应的模型为

$$f(\mathbf{x}) = \boldsymbol{\omega}^T \Phi(\mathbf{x}) + b \quad (2.16)$$

对偶问题什么的和之前都差不多

$$\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \\
\text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \\
& \alpha_i \geq 0, \quad i = 1, 2, \dots, m
\end{aligned} \tag{2.17}$$

特征空间维数可能很高, 直接计算  $\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$  通常困难, 设想一个函数:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \tag{2.18}$$

即  $\mathbf{x}_i$  和  $\mathbf{x}_j$  在特征空间的内积等于它们在原始样本空间中通过“核函数” $\kappa(\cdot, \cdot)$ 计算的结果. 式2.17重写为

$$\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\
\text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \\
& \alpha_i \geq 0, \quad i = 1, 2, \dots, m
\end{aligned} \tag{2.19}$$

求解后得到

$$\begin{aligned}
f(\mathbf{x}) &= \boldsymbol{\omega}^T \Phi(\mathbf{x}) + b \\
&= \sum_{i=1}^m \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b \\
&= \sum_{i=1}^m \alpha_i y_i \kappa(\mathbf{x}, \mathbf{x}_i) + b
\end{aligned} \tag{2.20}$$

式2.20显示出模型最优解就可通过训练样本的核函数展开, 这一表达式也称“支持向量展示”.

**定理 6.1 (核函数)** 令  $\chi$  为输入空间,  $\kappa(\cdot, \cdot)$  是定义在  $\chi \times \chi$  上的对称函数, 则  $\kappa$  是核函数当且仅当对于任意数据  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , “核矩阵” $K$  总是半正定的. 定理 6.1 表明, 只要一个对称函数所对应的核矩阵半正定, 它就能作为核函数使用. 事实上, 对于一个半正定核矩阵, 总能找到一个与之对

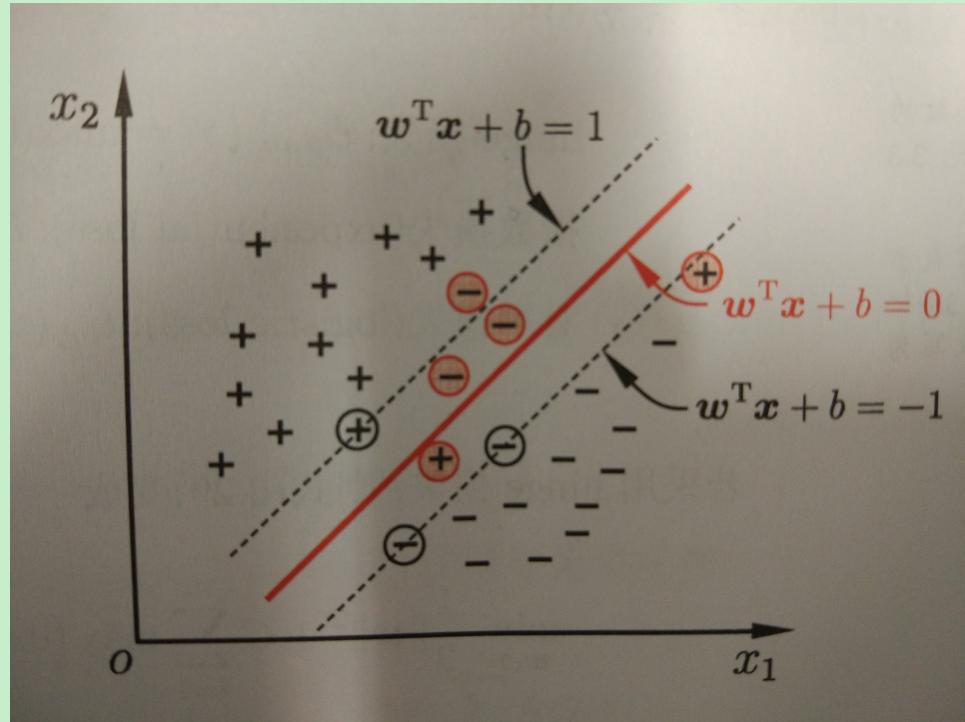


图 2: 软间隔

应的映射  $\Phi$

常用核函数:

线性核,  $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$

多项式核,  $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$

高斯核 (亦称 RBF 核),  $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

拉普拉斯核, Sigmoid 核, 还有函数组合的核.

## 2.4 软间隔与正则化

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, 可分的结果也许是过拟合造成的.

缓解的一个办法是允许支持向量机在一些样本上出错. 为此, 引入图2“软间隔”(soft margin) 概念.

支持向量机形式是要求所有样本均满足约束2.3, 即所有样本都必须划分正确, 称“硬间隔”(hard margin). 软间隔允许某些样本不满足约束 (不满足约束的样本应尽可能少)

$$y_i(\omega^T \mathbf{x}_i + b) \geq 1 \quad (2.21)$$

优化目标为

$$\min_{\omega, b} \quad \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^m l_{0/1}(y_i(\omega^T \mathbf{x}_i + b) - 1) \quad (2.22)$$

(原来的是那个  $\min_{\omega, b} \frac{1}{2} \|\omega\|^2$ )

$C > 0$  是常数,  $l_{0/1}$  是“0/1 损失函数”

$$l_{0/1}(z) = \begin{cases} 1 & \text{if } z < 0; \\ 0 & \text{otherwise} \end{cases} \quad (2.23)$$

当  $C$  为无穷大时, 式2.22迫使所有样本均满足约束2.21(不满足的话, 2.22会无穷大), 2.22等价于2.6;  $C$  取有限值时, 2.22允许一些样本不满足约束.

$l_{0/1}$  非凸, 非连续, 数学性质不太好, 使式2.22不易直接求解 (??), 常用其他函数替代之, 称“替代损失”. 替代损失函数一般具有较好的数学性质, 如通常是凸的连续函数且是  $l_{0/1}$  的上界 (这个也算优点????!!!). 三种常用替代损失函数 (图3):

hinge 损失:

$$l_{\text{hinge}}(z) = \max(0, 1 - z); \quad (2.24)$$

指数损失 (exponential loss):

$$l_{\text{exp}}(z) = \exp(-z); \quad (2.25)$$

对率损失 (logistic loss):

$$l_{\text{log}}(z) = \log(1 + \exp(-z)); \quad (2.26)$$

若采用 hinge 损失, 式2.22变成

$$\min_{\omega, b} \quad \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^m \max(0, 1 - y_i(\omega^T \mathbf{x}_i + b)) \quad (2.27)$$

引入“松弛变量”(slack variables)  $\xi_i \geq 0$ , 式2.27重写为

$$\begin{aligned} \min_{\omega, b, \xi_i} \quad & \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y_i(\omega^T \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \quad i = 1, 2, \dots, m. \end{aligned} \quad (2.28)$$

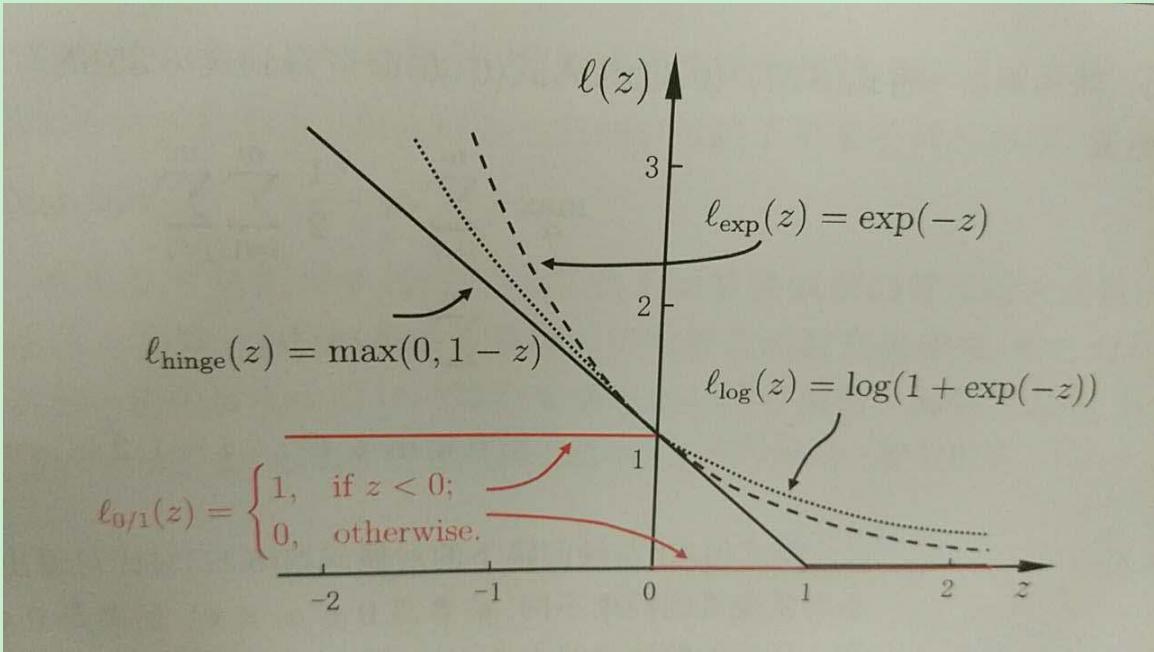


图 3: 三种常见的替代损失函数:hinge 损失, 指数损失, 对率损失

这就是常用的”软间隔支持向量机”.

式2.28中每个样本都有一个对应的松弛变量, 用以表征该样本不满足约束2.21的程度. 与2.6相似, 这也是一个凸二次规划问题. 通过拉格朗日乘子法可得到式2.28的拉格朗日函数

$$L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) = \frac{1}{2} \|\boldsymbol{\omega}\|^2 + C \sum_{i=1}^m \xi_i + \sum_{i=1}^m \alpha_i (1 - \xi_i - y_i (\boldsymbol{\omega}^T \mathbf{x}_i + b)) - \sum_{i=1}^m \mu_i \xi_i \quad (2.29)$$

$\alpha_i \geq 0, \mu_i \geq 0$  是拉格朗日乘子.

令  $L$  对  $\boldsymbol{\omega}, b, \xi_i$  的偏导为零可得

$$\boldsymbol{\omega} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i, \quad (2.30)$$

$$0 = \sum_{i=1}^m \alpha_i y_i, \quad (2.31)$$

$$C = \alpha_i + \mu_i \quad (2.32)$$

式2.30-2.32代入式2.29可得到式2.28的对偶问题

$$\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\
\text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \\
& 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, m.
\end{aligned} \tag{2.33}$$

同样可采用 SMO 优化算法, 核函数方法等解式2.33.

对软间隔支持向量机, KKT 条件:

$$\left\{
\begin{array}{l}
\alpha_i \geq 0, \quad \mu_i \geq 0, \\
y_i f(\mathbf{x}_i) - 1 + \xi_i \geq 0, \\
\alpha_i (y_i f(\mathbf{x}_i) - 1 + \xi_i) = 0, \\
\xi_i \geq 0, \quad \mu_i \xi_i = 0
\end{array}
\right. \tag{2.34}$$

若  $\alpha_i = 0$ , 则该样本不会对  $f(\mathbf{x})$  有任何影响; 若  $\alpha_i > 0$ , 必有  $y_i f(\mathbf{x}_i) = 1 - \xi_i$ , 即该样本是支持向量: 由式2.32可知, 若  $\alpha_i < C$ , 则  $\mu_i > 0$ , 进而  $\xi_i = 0$ , 该样本恰在最大间隔边上; 若  $\alpha_i = C$ , 有  $\mu_i = 0$ , 此时若  $\xi_i \leq 1$  则该样本落在最大间隔内部, 若  $\xi_i > 1$  则样本被错误分类. 因此, 软间隔支持向量机的最终模型仅与支持向量有关, 即采用 hinge 损失函数仍保持了稀疏性 (图中 hinge 损失函数有块“平坦”的零区域, 使得 svm 的解具有稀疏性).

式2.22的 0/1 损失函数换成别的替代损失函数可得到其他学习模型. 不管用什么损失函数, 模型具有一个共性: 目标的第一项用来描述划分超平面的“间隔”大小, 另一项用来表述训练集上的误差 (参照2.22), 更一般的形式:(“正则化”问题)

$$\min_f \quad \Omega(f) + C \sum_{i=1}^m l(f(\mathbf{x}_i), y_i) \tag{2.35}$$

$\Omega(f)$  为“结构风险”(structural risk), 用于描述模型  $f$  的某些性质;  $\sum_{i=1}^m l(f(\mathbf{x}_i), y_i)$  称为“经验风险”(empirical risk), 描述与训练数据的契合程度;  $C$  用于对二者进行折中. 从经验风险最小化角度来看,  $\Omega(f)$  表达了我们希望获得具有何种性质的模型 (例如希望获得复杂度较小的模型), 这为引入领域知识和用户意图提供了途径; 另一方面, 该信息有助于削减假设空间, 从而降低了最小化训练误差的过拟合风险 (??why?). 从这个角度说, 式 u2.35 称为“正则化”(regularization) 问题,  $\Omega(f)$  称为正则化项,  $C$  为正则化常数.  $L_p$  范数 (norm) 是常用的正则化项.

软间隔“软”在哪儿?

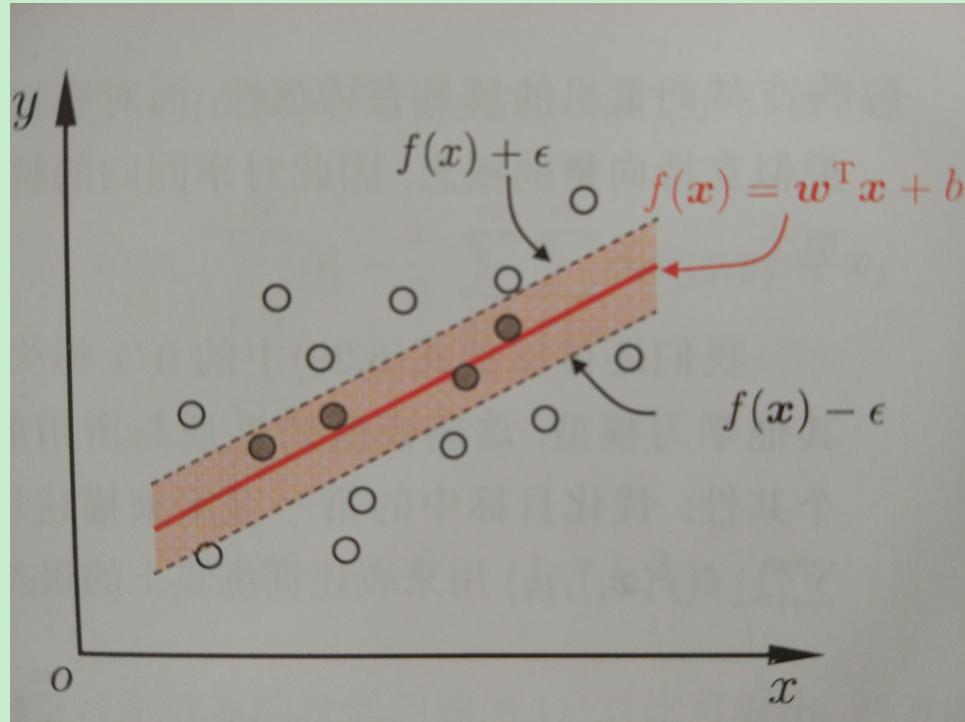


图 4: 支持向量回归, 落入橙色区域的样本不计算损失

## 2.5 支持向量回归

给定训练样本  $D = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m), y_i \in R$ , 希望学得一个形如式2.7的回归模型, 使得  $f(\mathbf{x})$  与  $y$  尽可能接近,  $\omega$  和  $b$  是待确定的模型参数.

支持向量回归 (Support Vector Regression, SVR) 仅当  $f(\mathbf{x})$  与  $y$  之间的差别绝对值大于  $\epsilon$  时才计算损失. 如图4.

SVR 问题可形式化为

$$\min_{\omega, b} \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^m l_\epsilon(f(\mathbf{x}_i) - y_i) \quad (2.36)$$

$C$  为正则化常数,  $l_\epsilon$  是  $\epsilon$ -不敏感损失

$$l_\epsilon(z) = \begin{cases} 0, & \text{if } |z| \leq \epsilon; \\ |z| - \epsilon, & \text{otherwise} \end{cases} \quad (2.37)$$

引入松弛变量  $\xi_i, \hat{\xi}_i$ , 式2.36可重写为

$$\begin{aligned}
& \min_{\omega, b, \xi_i, \hat{\xi}_i} \quad \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^m (\xi_i + \hat{\xi}_i) \\
& \text{s.t.} \quad f(\mathbf{x}_i) - y_i \leq \epsilon + \xi_i \\
& \quad y_i - f(\mathbf{x}_i) \leq \epsilon + \hat{\xi}_i \\
& \quad \xi_i \geq 0, \quad \hat{\xi}_i \geq 0, \quad i = 1, 2, \dots, m.
\end{aligned} \tag{2.38}$$

通过引入拉格朗日乘子  $\mu_i \geq 0, \hat{\mu}_i \geq 0, \alpha_i \geq 0, \hat{\alpha}_i \geq 0$ , 由拉格朗日乘子法可得到式 2.38 的拉格朗日函数

$$\begin{aligned}
& L(\omega, b, \alpha, \hat{\alpha}, \xi, \hat{\xi}, \mu, \hat{\mu}) \\
& = \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^m (\xi_i + \hat{\xi}_i) - \sum_{i=1}^m \mu_i \xi_i - \sum_{i=1}^m \hat{\mu}_i \hat{\xi}_i \\
& + \sum_{i=1}^m \alpha_i (f(\mathbf{x}_i) - y_i - \epsilon - \xi_i) + \sum_{i=1}^m \hat{\alpha}_i (y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i)
\end{aligned} \tag{2.39}$$

将式 2.7 代入, 令  $L()$  对  $\omega, b, \xi_i, \hat{\xi}_i$  的偏导为 0 可得

$$\omega = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \mathbf{x}_i, \tag{2.40}$$

$$0 = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i), \tag{2.41}$$

$$C = \alpha_i + \mu_i \tag{2.42}$$

$$C = \hat{\alpha}_i + \hat{\mu}_i \tag{2.43}$$

将式 2.40 - 2.43 代入式 2.39, 即可得到 svr 的对偶问题

$$\begin{aligned}
 & \max_{\alpha, \hat{\alpha}} \quad \sum_{i=1}^m y_i (\hat{\alpha} - \alpha_i) - \epsilon(\hat{\alpha}_i + \alpha_i) \\
 & \quad - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j) \mathbf{x}_i^T \mathbf{x}_j \\
 & \text{s.t.} \quad \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) = 0 \\
 & \quad 0 \leq \alpha_i, \hat{\alpha}_i \leq C
 \end{aligned} \tag{2.44}$$

上述过程需满足 KKT 条件:

$$\left\{
 \begin{array}{l}
 \alpha_i(f(\mathbf{x}_i) - y_i - \epsilon - \xi_i) = 0, \\
 \hat{\alpha}_i(y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i), \\
 \alpha_i \hat{\alpha}_i = 0, \xi_i \hat{\xi}_i = 0, \\
 (C - \alpha_i)\xi_i = 0, (C - \hat{\alpha}_i)\hat{\xi}_i = 0.
 \end{array}
 \right. \tag{2.45}$$

当且仅当  $f(\mathbf{x}_i) - y_i - \epsilon - \xi_i = 0$  时  $\alpha_i$  能取非零值, 当且仅当  $y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i = 0$  时  $\hat{\alpha}_i$  能取非零值. 即, 仅当样本  $(\mathbf{x}_i, y_i)$  不落入  $\epsilon$ -间隔带中, 相应的  $\alpha_i$  和  $\hat{\alpha}_i$  才能取非零值. 此外, 约束  $f(\mathbf{x}_i) - y_i - \epsilon - \xi_i = 0$  和  $y_i - f(\mathbf{x}_i) - \epsilon - \hat{\xi}_i = 0$  不能同时成立 (???), 因此  $\alpha_i$  和  $\hat{\alpha}_i$  至少有一个是零.

式2.40代入式2.7, 则 SVR 的解形如

$$f(\mathbf{x}) = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \mathbf{x}_i^T \mathbf{x} + b \tag{2.46}$$

能使式2.46中的  $\hat{\alpha}_i - \alpha_i \neq 0$  的样本即为 SVR 的支持向量??? 它们必落在  $\epsilon$ -间隔带之外 (落在间隔带中的样本  $\alpha$  都是零). SVR 的支持向量仅是训练样本的一部分, 其解仍具有稀疏性.

由式2.45, 对每个样本 (

$\mathbf{x}_i, y_i$ ) 都有  $(C - \alpha_i)\xi_i = 0$  且  $\alpha_i(f(\mathbf{x}_i) - y_i - \epsilon - \xi_i) = 0$

. 故若  $0 < \alpha_i < C$ , 必有  $\xi_i = 0$ , 进而

$$b = y_i + \epsilon - \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \mathbf{x}_i^T \mathbf{x} \tag{2.47}$$

在求解式2.44得到  $\alpha_i$  后, 可任意选取满足  $0 < \alpha_i < C$  的样本通过式2.47求  $b$ . 实践中常选取多个 (或所有) 满足条件  $0 < \alpha_i < C$  的样本求解  $b$  后取平均值, 更鲁棒.

考虑特征映射 (核方法), SVR 可表示为

$$f(\mathbf{x}) = \sum_{i=1}^m (\hat{\alpha}_i - \alpha_i) \kappa(\mathbf{x}, \mathbf{x}_i) + b \quad (2.48)$$

## 2.6 核方法

## 2.7 支持向量机与多分类问题

### 1. 一对多法

one-versus-rest(1-v-r SVMs), 训练时把某个类别的样本归为一类, 其他剩余的样本归为另一类, 这样  $k$  个类别的样本就构造出  $k$  个 SVM. 分类时将未知样本分类为具有最大分类函数值的那类?

### 2. 一对一法

libsvm 中的多类分类是根据一对一 (one-versus-one, 1-v-1SVMs) 法实现的, 做法是在任意两类样本之间设计一个 SVM, 因此  $k$  个类别的样本就需要设计  $k(k-1)/2$  个 SVM, 当对一个未知样本进行分类时, 最后得票对多的类别即为该未知样本的类别.

### 3. 层次支持向量机 (H-SVMs)

首先将所有类别分成两个子类, 再将子类分成两个次级子类, 循环直到得到一个单独的类为止.

#### 2.7.1 Matlab LibSVM

参考网址:

<http://www.matlabsky.com/forum-viewthread-tid-12379-fromuid-18677.html>

<http://www.matlabsky.com/thread-12649-1-1.html>

<http://www.ilovematlab.cn/thread-80099-1-1.html>

用 SVM (libsvm, lssvm, hssvm) 等等进行分类预测, 大致上要有三个步骤: 1. 训练, 2. 测试, 3. 预测

1. **训练** 用训练数据集, 不管采用那种寻优方式, 得到相对的最优参数, 训练模型。

2. **测试** 就是用刚刚得到的模型, 对测试数据进行测试, 此时测试数据集的 label 是已知的, 这主要是用来对刚刚的模型的检测, 或者是对参数的检测、或者是对模型的泛化能力的检测。此时会得到一个准确率, 这时的准确率是有用的, 是有实际意义的, 因为原来的 label 是已知的。

**3. 预测** 预测就是对未知类别的样本在测试确定了模型有好的泛化能力的情况下进行预测分类，这步才是真的预测能力功能的实现。此时数据的 label 随便给了，这样是为了满足 libsvm 对数据格式的要求。此时也会得到一个准确率，但是这个是没有实际意义的，因为原始的 label 是随便给的，没有意义，我们只是关心的是最后得到的类别号——达到分类的目的。

先用 libsvm 工具箱本身带的测试数据 heart\_scale 来实际进行实例测试，

```

1 %% HowToClassifyUsingLibsvm
2 % by faruto @ faruto's Studio-
3 % http://blog.sina.com.cn/faruto
4 % Email:faruto@163.com
5 % http://www.MATLABsky.com
6 % http://www.mfun.la
7 % http://video.ourmatlab.com
8 % last modified by 2010.12.27
9 %% a little clean work
10 tic;
11 close all;
12 clear;
13 clc;
14 format compact;
15 %%
16
17 % 首先载入数据
18 load heart_scale;
19 data = heart_scale_inst;
20 label = heart_scale_label;
21
22 % 选取前200个数据作为训练集合，后70个数据作为测试集合
23 ind = 200;
24 traindata = data(1:ind,:);
25 trainlabel = label(1:ind,:);
26testdata = data(ind+1:end,:);
27 testlabel = label(ind+1:end,:);
28
29 % 利用训练集合建立分类模型
30 model = svmtrain(trainlabel,traindata,'-s 0 -t 2 -c 1.2 -g 2.8');
```

```

31
32 % 分类模型model解密
33 model
34 Parameters = model.Parameters
35 Label = model.Label
36 nr_class = model.nr_class
37 totalSV = model.totalSV
38 nSV = model.nSV
39
40 % 利用建立的模型看其在训练集合上的分类效果
41 [ptrain,acctrain] = svmpredict(trainlabel,traindata,model);
42
43 % 预测测试集合标签
44 [ptest,acctest] = svmpredict(testlabel,testdata,model);
45
46 %%
47 toc;

```

运行结果:

```

1 model =
2     Parameters: [5x1 double]
3         nr_class: 2
4         totalSV: 197
5             rho: 0.0583
6             Label: [2x1 double]
7             ProbA: []
8             ProbB: []
9             nSV: [2x1 double]
10            sv_coef: [197x1 double]
11            SVs: [197x13 double]
12 Parameters =
13     0
14    2.0000
15    3.0000
16    2.8000
17     0

```

```

18 Label =
19      1
20      -1
21 nr_class =
22      2
23 totalSV =
24      197
25 nSV =
26      89
27      108
28 Accuracy = 99.5% (199/200) (classification)
29 Accuracy = 68.5714% (48/70) (classification)
30 Elapsed time is 0.040873 seconds.
31 >>

```

model = svmtrain(trainlabel,traindata,'-s 0 -t 2 -c 1.2 -g 2.8'); 中参数的意义:

**-s** SVM 类型: SVM 设置类型 (默认 0)

0 – C-SVC

1 – v-SVC

2 – 一类 SVM

3 – e-SVR

4 – v-SVR

**-t** 核函数类型: 核函数设置类型 (默认 2)

0 – 线性

1 – 多项式

2 – RBF 函数

3 – sigmoid

**-g r(gama):** 核函数中的 gamma 函数设置 (针对多项式/rbf/sigmoid 核函数)

**c cost:** 设置 C-SVC, e-SVR 和 v-SVR 的参数 (损失函数)(默认 1)

...

其实还有很多，详细的见 <http://www.matlabsky.com/thread-12380-1-1.html>

model 参数解密：还是 libsvm-mat 自带的 heart\_scale.mat 数据（270\*13 的一个属性矩阵，共有 270 个样本，每个样本有 13 个属性）

```

1 %% ModelDecryption
2 % by faruto @ faruto's Studio-
3 % http://blog.sina.com.cn/faruto
4 % Email:faruto@163.com
5 % http://www.MATLABsky.com
6 % http://www.mfun.la
7 % http://video.ourmatlab.com
8 % last modified by 2011.01.06
9 %% a little clean work
10 tic;
11 close all;
12 clear;
13 clc;
14 format compact;
15 %%
16 % 首先载入数据
17 load heart_scale;
18 data = heart_scale_inst;
19 label = heart_scale_label;
20 % 建立分类模型
21 model = svmtrain(label,data,'-s 0 -t 2 -c 1.2 -g 2.8');
22 model
23 % 利用建立的模型看其在训练集合上的分类效果
24 [PredictLabel,accuracy] = svmpredict(label,data,model);
25 accuracy
26 %%
27 toc;
```

运行结果：

```

1 model =
2     Parameters: [5x1 double]
```

```

3      nr_class: 2
4      totalSV: 259
5          rho: 0.0514
6      Label: [2x1 double]
7      ProbA: []
8      ProbB: []
9      nSV: [2x1 double]
10     sv_coef: [259x1 double]
11     SVs: [259x13 double]
12 Accuracy = 99.6296% (269/270) (classification)
13 accuracy =
14     99.6296
15     0.0148
16     0.9851
17 Elapsed time is 0.040155 seconds.

```

**model.Parameters** 参数意义从上到下依次为：

**-s** Svm 类型

**-t** 核函数类型

**-d** degree: 核函数中的 degree 设置 (针对多项式核函数)(默认 3)

**-g** r(gamma)

**-r** coef0: 核函数中的 coef0 设置 (针对多项式/sigmoid 核函数)((默认 0))

Libsvm 中参数设置可以按照 SVM 的类型和核函数所支持的参数进行任意组合，如果设置的参数在函数或 SVM 类型中没有也不会产生影响，程序不会接受该参数；如果应有的参数设置不正确，参数将采用默认值。

**model.Label** 表示数据集中类别的标签都有什么

**model.nr\_class** 表示数据集中有多少类别

**model.totalSV** 总共的支持向量的数目，这里有 259 个

**model.nSV** 每类样本的支持向量的数目，这里的顺序是和 **model.Label** 对应的

`model.ProbA & model.ProbB` 使用`-b` 参数时才能用到，用于概率估计

`model.sv_coef` 矩阵，这里承装的是 259 个支持向量在决策函数中的系数

`model.SVs` 是一个 259\*13 的稀疏矩阵，承装的是 259 个支持向量

`model.rho` 是决策函数中的常数项的相反数 (-b)

通过`-s 0` 参数 (C-SVC 模型) 得到的最终的分类决策函数的表达式是

$$p_{label} = \text{sgn}\left(\sum_{j=1}^n w_j K(x_j, x)\right) + b$$

`model.sv_coef(i)` 对于每一个 i: `wi = model.sv_coef(i);` 支持向量的系数 (一个标量数字) . `xi = model.SVs(i,:)` 支持向量 (1\*13 的行向量) , 这里的 `wi` 是  $\alpha_i y_i$

## 3. 决策树

类似于只能根据判断是与否来一直划分，直至最终确定事物类别的流程图。决策树的一个重要任务是为了数据中所蕴含的知识信息，因此决策树可以使用不熟悉的数据集合，并从中提取一系列规则，在这些机器根据数据集创建规则时，就是机器学习的过程。

**优点：**计算复杂度不高，输出结果易于理解，对中间值的缺失不敏感，可以处理不相关特征数据。

**缺点：**可能会产生过度匹配的问题

**适用数据类型：**数值型和标称型

### 3.1 决策树的构造

构造决策树需要解决的第一个问题：当前数据集上哪个特征在划分数据分类时起决定性作用。为找到决定性的特征，必须评估每个特征。完成测试后，原始数据集被划分为几个数据子集。这些数据子集会分布在第一个决策点的所有分支上。如果某个分支下的数据属于同一类型，则已经正确地划分数据分类。无需进一步分割。如果数据子集内的数据不属于同一类型，则需重复划分数据子集的过程。

创建分支函数 `createBranch()` 的伪代码：

```

1 检测数据集中的每个子项是否属于同一分类;
2     if so return 标签;
3     else
4         寻找划分数据集的最好特征

```

```
5      划分数据集  
6      创建分支节点  
7      for 每个划分的子集  
8          调用函数createBranch并增加返回结果到分支节点中  
9      return 分支节点
```

### 3.1.1 信息增益

划分数据的最大原则是：将无序的数据变得更加有序。

在划分数据集之前之后信息发生的变化成为信息增益，计算每个特征值划分数据集获得的信息增益，获得信息增益最高的特征就是最好的选择。

### 3.1.2 划分数据集

对每个特征划分数据集的结果计算一次信息熵，然后判断按照哪个特征划分数据集是最好的划分方式。

### 3.1.3 构建递归决策树

递归函数的第一个停止条件是所有的类别标签完全相同，则直接返回该类别标签。递归函数的第二个停止条件是使用完了所有的特征，仍不能将数据集划分成仅包含唯一类别的数组。挑选出现次数最多的类别作为返回值。

## 4. Bagging 与随机森林