R 机器学习

第 15 讲　支持向量机

张敬信

2022 年 10 月 27 日

哈尔滨商业大学

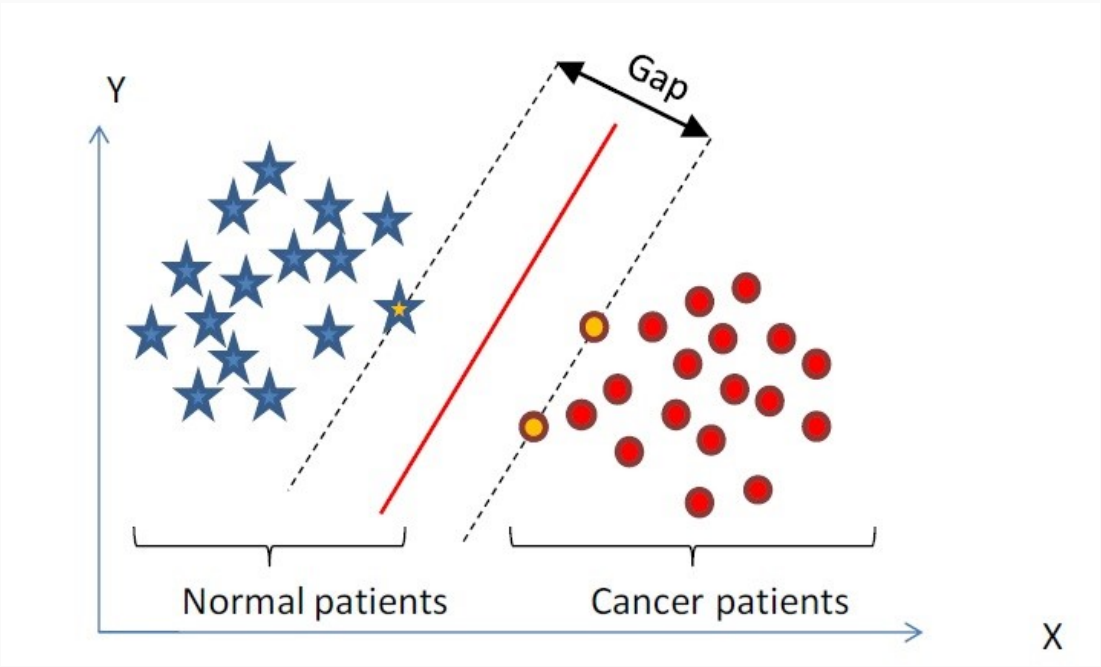
一. 支持向量机概述

支持向量机（Support Vector Machine）是一种经典的有监督机器学习算法，在深度神经网络兴起之前与神经网络并驾齐驱，最初是用于分类，改进后也可用于回归。

通俗来讲，经典 SVM 是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，其学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。

线性分类的引例

二维平面上有线性可分的两类点：蓝五角星表示正常病人，红点表示癌症病人。SVM 就是寻找一条直线（拓展到高维空间时就是超平面），使其能够尽可能的分割开两组数据。距离样本点近的直线虽然也可以分开两组数据，但不是最优的，因为这样的直线对噪声敏感度高，泛化能力较差，任何细微的变动（比如旋转一个小角度）都可能导致类别的改变。



可以选择分离两类数据的两个平行超平面，此时它们之间的距离达到尽可能的大，在这两个超平面范围内的区域称为间隔（margin），则最大间隔超平面就是位于它们正中间的超平面。

关于“支持向量”，通常数据集中有着无数的样本点，但 SVM 算法并非是以所有的样本点作为构建超平面的依据。可以构建的超平面有无穷多个，但具有最大间隔的超平面才是 SVM 要寻找的最优解，同时伴随产生两个平行超平面，即上图中的两条虚线，这两条虚线所穿过的样本点，就是 SVM 中的支持样本点，称为“支持向量”。实际上最优超平面的方向和位置完全取决于选择哪些样本作为支持向量。

最大间隔超平面可用线性泛函 $f(x) = 0$ 表示，则若 $f(x) < 0$, 则其类别归为 $y = -1$; 若 $f(x) > 0$, 则其类别归为 $y = 1$. 通常让 $f(\cdot)$ 在支持向量上取值 ± 1.

另外，对于线性不可分的数据，SVM 算法很巧妙地借助核函数将低维空间线性不可分转化为高维空间的线性可分，进而构造出最优分离超平面，从而把平面上本身不好分的非线性数据分开。核函数巧妙地实现了在低维上进行计算，并将实质上的分类效果表现在了高维上，从而避免了直接在高维空间中的复杂计算。

二. 函数间隔与几何间隔

1. 线性可分

设某特征空间上的训练数据集为

$$T = \{(x_1, y_1), \cdots, (x_m, y_m) : \ x_i \in \mathbb{R}^n, y_i \in \{1, -1\}\}$$

当 $y_i = 1$ 时，称所对应的样本为正例，当 $y_i = -1$ 时，对应的样本则为负例。

若如果存在某个超平面能够将数据集中的正例和负例完全正确地划分到超平面的两侧，则称数据集 T 为线性可分数据集。否则，则称为线性不可分。

SVM 的任务就是在 n 维数据空间中，寻找一个 $n-1$ 维超平面，其方程为 $f(x)=w^Tx+b=0$，相应的分类决策函数为

$$\text{sign}\big(f(x)\big)=\text{sign}\big(w^Tx+b\big)$$

其中，$w=(w_1,\cdots,w_n)^T$ 为法向量，它决定了超平面的方向；b 为截距项，它决定了超平面与原点之间的距离。

2. 函数间隔

通过观察 $w^T x_i + b$ 的符号与类标记 y_i 的符号是否一致，就可以判断分类是否正确。为此，定义超平面 (w, b) 关于任意样本点 (x_i, y_i) 的函数间隔为：

$$\hat{\gamma}_i^f = y_i(w^T x_i + b) = y_i f(x_i)$$

进而定义超平面 (w, b) 关于训练数据集 T 的函数间隔为，超平面 (w, b) 关于 T 中所有样本点 (x_i, y_i) 的函数间隔的最小值：

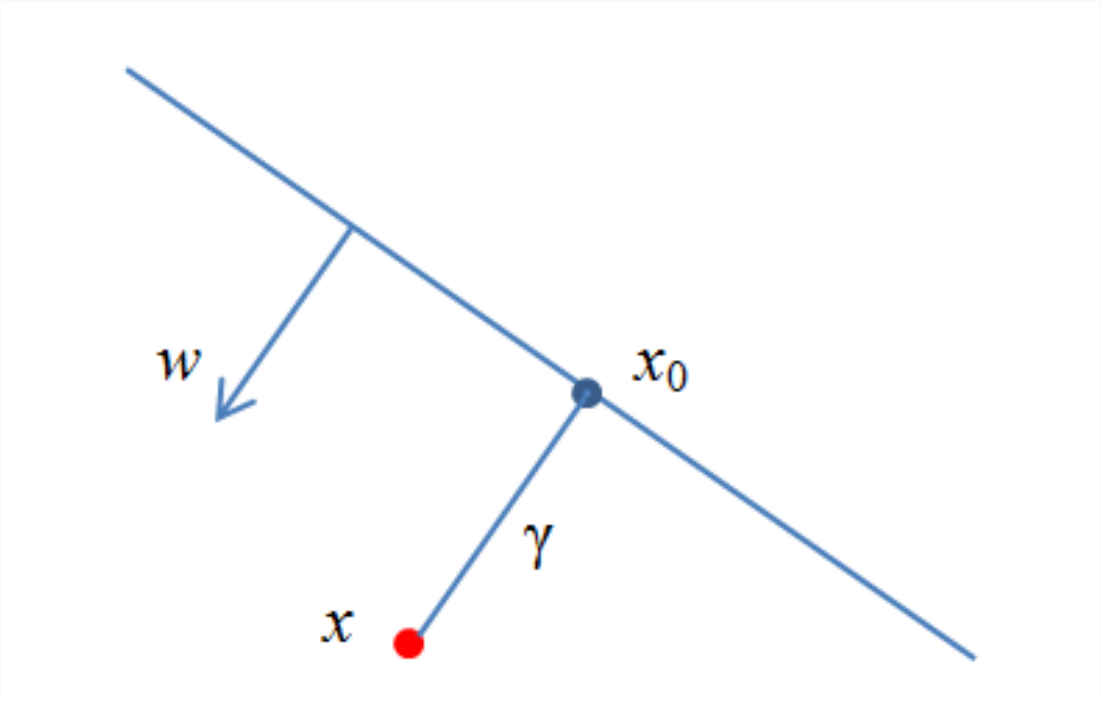
$$\hat{\gamma}_f = \min_{1 \le i \le m} \hat{\gamma}_i^f$$

但这样定义的函数间隔有一个缺陷：若成比例地改变 w 和 b，比如改成 $2w$ 和 $2b$, 则函数间隔的值 $f(x)$ 就变成了原来的 2 倍，而此时的超平面并没有改变。

3. 几何间隔

由于函数间隔的缺陷，就需要对分离超平面的法向量增加某些约束，比如规范化约束：$\|w\| = 1$. 为此，引入几何间隔（真正意义上的点到超平面的距离）。

对于样本点 x, 记其垂直投影到超平面上的点为 x_0, w 为超平面的法向量，用 γ 表示 x 到超平面的距离：



则有 $x = x_0 + \gamma \frac{w}{\|w\|}$, 其中 $\frac{w}{\|w\|}$ 为 w 的单位化，两边同乘以 w^T，注意到 $w^T w = \|w\|^2$, 则

$$w^T x = w^T x_0 + \gamma \frac{w^T w}{\|w\|} = w^T x_0 + \gamma \|w\|$$

又 x_0 是超平面上的点，即满足 $w^T x_0 + b = 0$, 则 $w^T x_0 = -b$. 从而

$$\gamma = \frac{w^T x + b}{\|w\|} = \frac{f(x)}{\|w\|}$$

类似地，若 x 在 w 反向一侧，则 $\gamma = \frac{-f(x)}{\|w\|}$. 综上，

$$\gamma = \frac{|w^T x + b|}{\|w\|} = \frac{|f(x)|}{\|w\|}$$

这就是直观上任意一点到超平面的距离公式。

为此，定义超平面 (w, b) 关于任意样本点 (x_i, y_i) 的几何间隔为：

$$\hat{\gamma}_i = y_i \frac{w^T x_i + b}{\|w\|} = y_i \frac{f(x_i)}{\|w\|} = \frac{\hat{\gamma}_f}{\|w\|}$$

进而定义超平面 (w, b) 关于训练数据集 T 的几何间隔为，超平面 (w, b) 关于 T 中所有样本点 (x_i, y_i) 的几何间隔的最小值：

$$\hat{\gamma} = \min_{1 \leq i \leq m} \hat{\gamma}_i$$

可见，实际上函数间隔就是 $|w^T x + b| = |f(x)|$, 几何间隔就是函数间隔除以 $\|w\|$, 几何间隔在缩放 w 和 b 时不会发生改变，它只随着超平面的变动而变动。所以，几何间隔才是判断超平面是否优良的标准。

1. 最大间隔分类器

对一个数据点进行分类，当超平面离数据点的 “几何间隔” 越大，分类的确信度也越大。所以，为了使得分类的确信度尽量高，需要让所选择的超平面能够最大化这个 “几何间隔” 值。于是最大间隔分类器的目标函数可以定义为：

$$\max \tilde{\gamma}, \quad \text{s.t.} \quad y_i(w^T x_i + b) = \tilde{\gamma}_i^f \geq \tilde{\gamma}_f, \quad i = 1, \cdots, m$$

若令函数间隔 $\hat{\gamma}_f = 1$（不影响目标函数的优化），则几何间隔 $\hat{\gamma} = \frac{1}{\|w\|}$，于是上述目标函数可转化为：

$$\max \frac{1}{\|w\|}, \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, \cdots, m \tag{1}$$

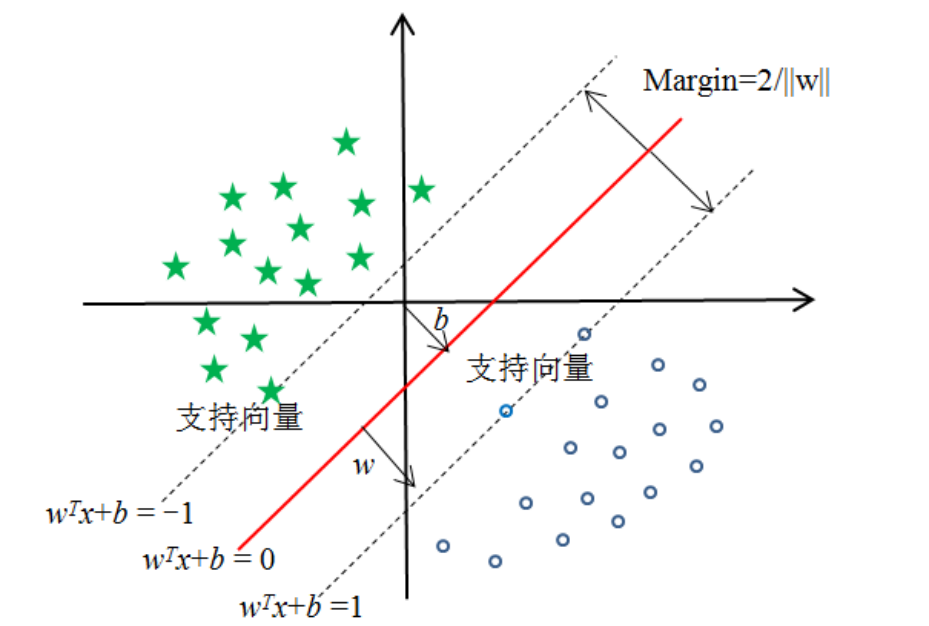


图 1: 最大几何间隔的超平面

如图所示，中间的实线便是寻找到的最优超平面，其到两条虚线的距离相等，这个距离便是几何间隔 $\hat{\gamma}$, 两条虚线之间的距离等于 $2\hat{\gamma}$, 而虚线上的点则是支持向量。由于这些支持向量刚好在边界上，所以它们满足 $y(w^Tx+b)=\pm 1$, 而对于所有不是支持向量的点，显然有 $y(w^Tx+b)>1$ 和 $y(w^Tx+b)<-1$.

2. 模型的对偶转化

由模型（1），要求 $\frac{1}{\|w\|}$ 的最大值，相当于求 $\frac{1}{2}\|w\|^2$ 的最小值，于是可等价地表示为：

$$\min \frac{1}{2}\|w\|^2, \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, \cdots, m \tag{2}$$

这是凸二次规划问题，在一定的约束条件下，使目标最优、损失最小。

对于模型（2），可以通过求解对偶问题得到最优解，这就是线性可分支持向量机的对偶算法，这样做的优点在于：

- 对偶问题往往更容易求解；
- 可以自然地引入核函数，进而推广到非线性分类问题。

先利用拉格朗日乘子法，通过给每一个约束条件加上一个拉格朗日乘子 α, 定义拉格朗日函数：

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i(y_i(w^Tx_i+b)-1)$$

再令

$$\theta(w)=\max_{\alpha_i\geq 0}L(w,b,\alpha)$$

则容易验证：

(1) 通过调整 α 的大小，可以使 $L(w,b,\alpha)$ 达到最大；

(2) 当约束条件不满足时（例如某 $y_i(w^Tx_i+b)<1$），则一定可以调整 α 使得 $\theta(w)=\infty$（只要令 $\alpha_i=\infty$ 即可）；

(3) 当所有约束条件都满足时，若 $L(w,b,\alpha)$ 达到最大，就有 $\theta(w)=\frac{1}{2}\|w\|^2$.

于是，目标函数就可等价地改写为：

$$\min_{w,b} \theta(w) = \min_{w,b} \max_{\alpha_i \geq 0} L(w, b, \alpha) = p^* \tag{3}$$

这里用 p^* 表示该问题的最优值。需要先求约束条件下不等式的最大值，再求关于 w, b 的最小值，这仍是难以求解的问题。若将顺序调换一下，先求解关于 w, b 的最小值，再求约束条件下的最大值，即

$$\max_{\alpha_i \geq 0} \min_{w,b} L(w, b, \alpha) = d^* \tag{4}$$

模型 (4) 是模型 (3) 的对偶问题，它更容易求解，其最优值用 d^* 表示。对偶问题有这样结论：

- **若原始问题和对偶问题都有最优值，则弱对偶性质成立**：

$$d^* = \max_{\alpha_i \geq 0} \min_{w,b} L(w, b, \alpha) \leq \min_{w,b} \theta(w) = \min_{w,b} \max_{\alpha_i \geq 0} L(w, b, \alpha) = p^*$$

大致相当于最小值中最大的那个必然要小于等于最大值中最小的那个。

于是，若找到能够使该不等式恰好取到等号的条件，则 d^* 就是要找的最优值 p^*.

3. K.K.T. 条件

一般地，最优化数学模型可表示为如下标准形式：

$$\begin{aligned} & \min f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \cdots, p \\ & h_j(x) = 0, \quad j = 1, \cdots, q \\ & x \in D \subset \mathbb{R}^n \end{aligned}$$

凸优化问题是指优化问题的目标函数和定义域都是凸集，最大的好处就是其局部最优解就是全局最优解。

对于上述的优化问题，若最小值点 x^* 满足：

(i) $g_i(x^*) \leq 0,\ i = 1, \cdots, p;\ h_j(x^*) = 0,\ j = 1, \cdots, q;$

(ii) $\nabla f(x^*) + \sum_{i=1}^{p} \lambda_i \nabla h_i(x^*) + \sum_{j=1}^{q} \mu_j \nabla g_j(x^*) = 0,\ \lambda_i \neq 0,\ \mu_j \geq 0,\ \mu_j g_j(x^*) = 0,$ 其中 ∇ 为梯度算子；

则称为满足 K.K.T. 条件。K.K.T. 条件是让非线性规划问题能有最优解的一个充要条件。

可以证明，前文的优化模型满足 K.K.T. 条件，即若找到参数值 w^*, b^*, α^* 满足了 K.K.T. 条件，就有 $p^* = d^*$.

- 第 1 步，先固定 α, 求 $L(w, b, \alpha)$ 关于 w, b 的最小值。

分别对 w, b 求偏导，并令其等于 0, 可求得

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i, \qquad \sum_{i=1}^{m} \alpha_i y_i = 0$$

再代入 $L(w, b, \alpha)$ 得到

$$L(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- 第 2 步，求对 α 的极大值，即关于对偶问题的最优化问题。

其数学表示为

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \cdots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

求这个拉格朗日乘子 α，用到了 SMO 高效优化算法（略）。

- 第 3 步，利用前面求出最优的 $\alpha = [\alpha_1, \cdots, \alpha_m]^T$, 根据

$$w^* = \sum_{i=1}^{m} \alpha_i y_i x_i, \qquad b^* = -\frac{\max\limits_{i:\, y_i=-1} w^{*T} x_i + \min\limits_{i:\, y_i=1} w^{*T} x_i}{2}$$

即可求出 w, b, 最终得到分离超平面 $w^T x + b = 0$ 和分类函数:

$$f(x) = w^T x + b = \left(\sum_{i=1}^{m} \alpha_i y_i x_i \right)^T x + b \tag{5}$$

于是，对一个数据点 x 进行分类，实际上就是通过把 x 代入到上式，计算出结果然后根据其正负号来进行类别划分。

到目前为止，SVM 还比较弱，只能处理线性的情况，下面将引入核函数，进而推广到非线性分类问题。

四. 核函数与非线性 SVM

对于非线性的情况，SVM 的处理方法是选择一个核函数，通过将数据映射到高维空间，来解决在原始空间中线性不可分的问题。

具体来说，在线性不可分的情况下，支持向量机首先在低维空间中完成计算，然后通过核函数将输入空间映射到高维特征空间，最终在高维特征空间中构造出最优分离超平面，从而把平面上本身不好分的非线性数据分开。

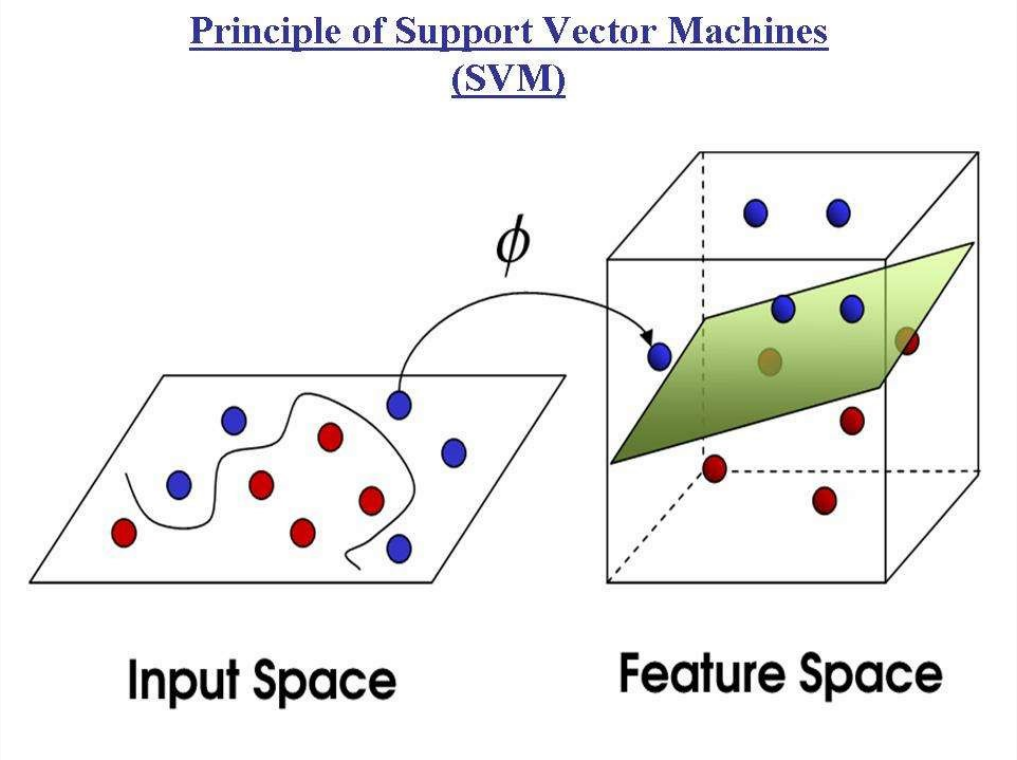
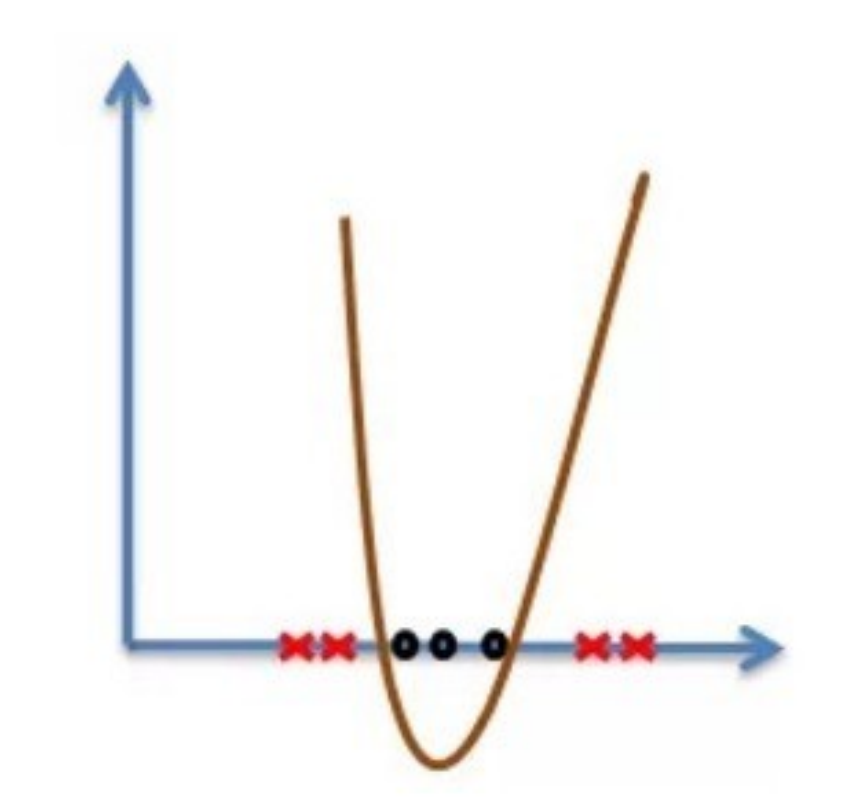


图 2: 核函数转化特征示意图

如图所示的二维空间中很难建立一个最大间隔超平面来把数据划分开来，但是若恰当地映射到三维空间中，这样在二维空间中的线性不可分也就巧妙地转化为三维空间中的线性可分情况了。

简单示例



对于本例，显然直线是无法有效地区分开它们的，而当换用二次曲线后，问题就迎刃而解了。

这条二次曲线具体表述为：

$$f(x) = w_1 x^2 + w_2 x + w_3 + b = w^T \begin{bmatrix} x^2 \\ x \\ C \end{bmatrix} + b = w^T \phi(x) + b$$

这样，相当于找到一个从输入空间到某特征空间的非线性映射 ϕ, 把一维的 x 映射到 $[x^2, x, C]$ 上。

回顾线性可分情形，推导出来的用于预测的分类函数，即 (5) 式，可写为内积形式：

$$f(x) = w^T x + b = \left(\sum_{i=1}^{m} \alpha_i y_i x_i \right)^T x + b = \sum_{i=1}^{m} \alpha_i y_i \langle x_i, x \rangle + b$$

由于映射后 $x \to \phi(x)$, 相应的分类函数就变为

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

貌似问题已经解决了：拿到非线性数据，就找一个非线性映射，然后一股脑把原来的数据映射到新空间中，再做线性 SVM 即可。

但事实没有这么简单，比如对二维空间做映射，生成的新空间是原始空间中一阶和二阶的组合，这样会得到 5 个维度；若原始空间是三维，生成的新空间自然是原始空间中一阶至三阶的组合，并因此而得到 19 个维度 …… 以此类推，高维空间通过映射得到的新空间的维度是呈爆炸性增长的（即维数灾难），这就给映射的计算带来了非常大的困难。

好在有 “核函数” 这一利器，它在处理非线性数据的同时将极大的简化计算过程。

核函数技术

若原始特征内积为 $\langle x,\, y\rangle$, 映射后为 $\langle \phi(x),\, \phi(y)\rangle$, 那么定义核函数为: $K(x,y) = \langle \phi(x),\, \phi(y)\rangle$.

核函数的作用在于，简化了映射空间的内积计算，可以让 x 和 y 不用通过映射到高维空间再计算内积，而是直接在低维空间就计算了（即避开了 x 映射到 $\phi(x)$, y 映射到 $\phi(y)$ 的中间步骤)。

以前例为例，设 $x = [x_1, x_2]^T$, $\phi(x) = [x^2, x, c]^T$, 任取 $C = 1/2$. 令核函数 $K(x,y) = (x^T y + 1/2)^2$. 可推得

$$\begin{aligned} K(x,y) = \cdots &= \langle [x^2, x, 1/2]^T, [y^2, y, 1/2]^T \rangle \\ &= \langle \phi(x),\, \phi(y)\rangle \end{aligned}$$

可见，核函数在低维计算的结果完全等价于这两个变量高维映射后的内积，这样就避开了直接在高维空间中进行计算。这就解决了映射的维数灾难问题，甚至能处理无穷维度的情况。

因此，核函数能简化映射空间中的内积运算——刚好“碰巧”的是，在 SVM 里需要计算的地方数据向量总是以内积的形式出现的。比如前文的分类函数可表示为：

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b$$

前文的对偶最优化问题可表示为：

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \cdots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

这样以来，计算的问题就算解决了，避开了直接在高维空间中进行计算，而结果却是等价的！当然，因为上面的例子非常简单，所以可以手工构造出对应于 $\phi(\cdot)$ 的核函数。但核函数不是固定的，对于任意一个映射，想要构造出对应的核函数是比较困难的。

常用的核函数

核函数是需要调参的超参数，常用的核函数有：

(1) 线性核：$K(x_i, x_j) = \langle x_i, x_j \rangle$, 实际上就是原始空间中的内积；

(2) 多项式核：$K(x_i, x_j) = \left(\gamma \langle x_i, x_j \rangle + c_0\right)^d$, 前面的示例就是多项式核。该特征空间的维度是 C_{m+d}^d，其中 m 是原始空间的维度；

(3) 径向基核（高斯核）：$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$, 若 γ 选的很小，高次特征上的权重实际上衰减得非常快（相当于一个低维子空间）；若 γ 选的很大，则可以将任意的数据映射为线性可分，但随之而来的可能是非常严重的过拟合。总的来说，通过调控参数 γ, 径向基核具有相当高的灵活性，也是使用最广泛的核函数之一。

(4) Sigmoid 核：$\tanh(\gamma \langle x_i, x_j \rangle + c_0)$.

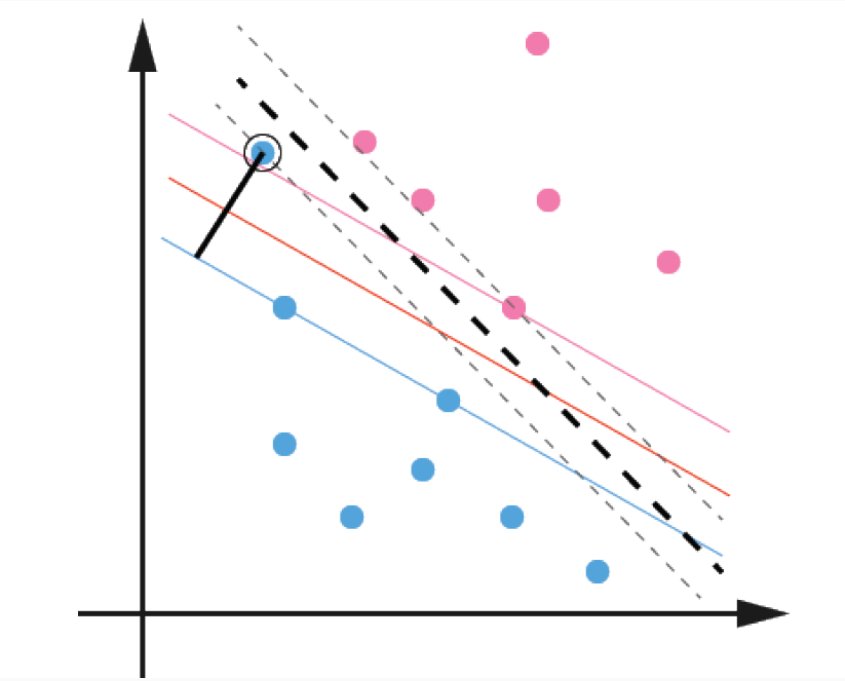
另外，还有拉普拉斯核。针对具体情况，有的核函数效果很好，有的比较差，没有哪一个核函数是完美的，需要进行不断的试验和甄别。不过总体来看，径向基核是不会出太大偏差的一种。

Mercer 定理给出了判断一个核函数是否有效的充要条件：

$$K(x_i, x_j) \text{ 是有效核函数} \iff \text{核函数矩阵 } K \text{ 是对称半正定的}$$

五. 使用松弛变量处理异常值

若数据有噪声，即偏离正常位置很远的数据点，称为异常值，在线性 SVM 模型里，异常值的存在有可能造成很大的影响，因为超平面本身就是只有少数几个支持向量组成的，如果这些支持向量里又存在异常值的话，其影响就很大了。



若没有那个带黑圈的异常值点，分割超平面是很好的，但它的存在导致分割超平面被挤歪了。若把该异常值点移动回来，就可以避免这种情况。

数学上的做法就是，给约束条件加上可以“容忍”的松弛变量，将原约束条件变为：

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \qquad i = 1, ..., m$$

其中，ξ_i 为松弛变量，表示数据点 x_i 允许偏离的函数间隔量。当然，不能任意大地偏离，为此，给原目标函数也增加一项，使得这些 ξ_i 的总和也要尽量小：

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$

其中，参数 C 是正则项的惩罚，用来控制目标函数中“寻找几何间隔最大的超平面”与“保证数据点偏差量最小”的权重，是需要调参的超参数。

于是，原优化模型 (2) 变为：

$$\begin{aligned} \min\ & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.}\ \ & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \cdots, m \\ & \xi_i \geq 0, \quad i = 1, \cdots, m \end{aligned}$$

利用同样的方法求解，得到对偶问题的最优化模型：

$$\begin{aligned} \max_{\alpha}\ & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ \text{s.t.}\ \ & 0 \leq \alpha_i \leq C, \quad i = 1, \cdots, m \\ & \sum_{i=1}^{m}\alpha_i y_i = 0 \end{aligned}$$

对比前式，唯一的区别就是现在对偶变量 α 多了一个上限 C.

进一步，核函数化的非线性 SVM 形式也是一样的，只要把 $\langle x_i, x_j \rangle$ 换成 $K(x_i, x_j)$ 即可。

以上建立起来一个完整的可以处理线性和非线性并能容忍噪声和异常值的支持向量分类理论。

六. 支持向量回归

支持向量分类的损失函数是 Hinge（合页）损失：

$$L(y, f(x)) = \max\{0, 1 - y \cdot f(x)\}$$

若将 Hinge 损失换成 ε-不敏感损失：

$$L(y, f(x)) = \max\{0, |y - f(x)| - \varepsilon\}$$

则得到支持向量回归模型。

SVM 最重要的推广有两个：一是推广到 ε-SVM 包括分类和回归，其缺陷是很难选择恰当的 ε. 二是 ν-SVR，它可以通过自动调整参数 ε 和 ε 不敏感损失函数来控制训练误差，并能控制支持向量的个数。

ν-SVR 模型可表示为：

$$
\begin{aligned}
&\min \frac{1}{2} w^T w + C\left(\nu\varepsilon + \frac{1}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*)\right) \\
\text{s.t.} \quad &\left(w^T\phi(x_i) + b\right) - y_i \leq \varepsilon + \xi_i, \\
&y_i - \left(w^T\phi(x_i) + b\right) - y_i \leq \varepsilon + \xi_i^*, \\
&\xi_i,\, \xi_i^* \geq 0, \quad i = 1, \cdots, m,\, \varepsilon \geq 0
\end{aligned}
$$

其中，$0 < \nu < 1, C$ 为正则化参数，训练向量 x_i 通过核函数 $\phi(\cdot)$ 映射到更高维空间，ε-非敏感损失函数意味着，若 $w^T\phi(x)$ 落在 $y \pm \varepsilon$ 范围内，则不考虑损失。

相应的对偶问题为：

$$\begin{aligned} &\min \frac{1}{2}(\alpha-\alpha^*)^T Q(\alpha-\alpha^*)+y^T(\alpha-\alpha^*) \\ \text{s.t.}\quad &e^T(\alpha-\alpha^*)=0,\ e^T(\alpha+\alpha^*)\le C\nu, \\ &0\le \alpha_i,\ \alpha_i^*\le C/m,\quad i=1,\cdots,m \end{aligned}$$

其中，$Q_{ij}=\phi(x_i)^T\phi(x_i)$ 为核，e 为全 1 向量，则逼近函数为：

$$f(x)=\sum_{i=1}^{m}(\alpha_i^*-\alpha_i)\phi(x_i)^T\phi(x_i)+b$$

关于 ν-SVR 的核函数的选择，最常用的是径向基核函数。

七. 支持向量分类案例

mlr3verse 提供了 `classif.svm` 和 `regr.svm` 学习器分别用于支持向量分类和支持向量回归，它们是 `e1071::svm()` 函数的封装。

以来自 UCI 网站的美国威斯康星州乳腺癌诊断数据集为例，来演示支持向量分类。该数据集共包含 569 个观测和 32 个变量，其中变量 diagnosis 为因变量，特征包括半径、平滑度、分形维数的均值、标准差等。

1. 准备数据

- 读取数据，删除 id 列，修正不合法的列名：

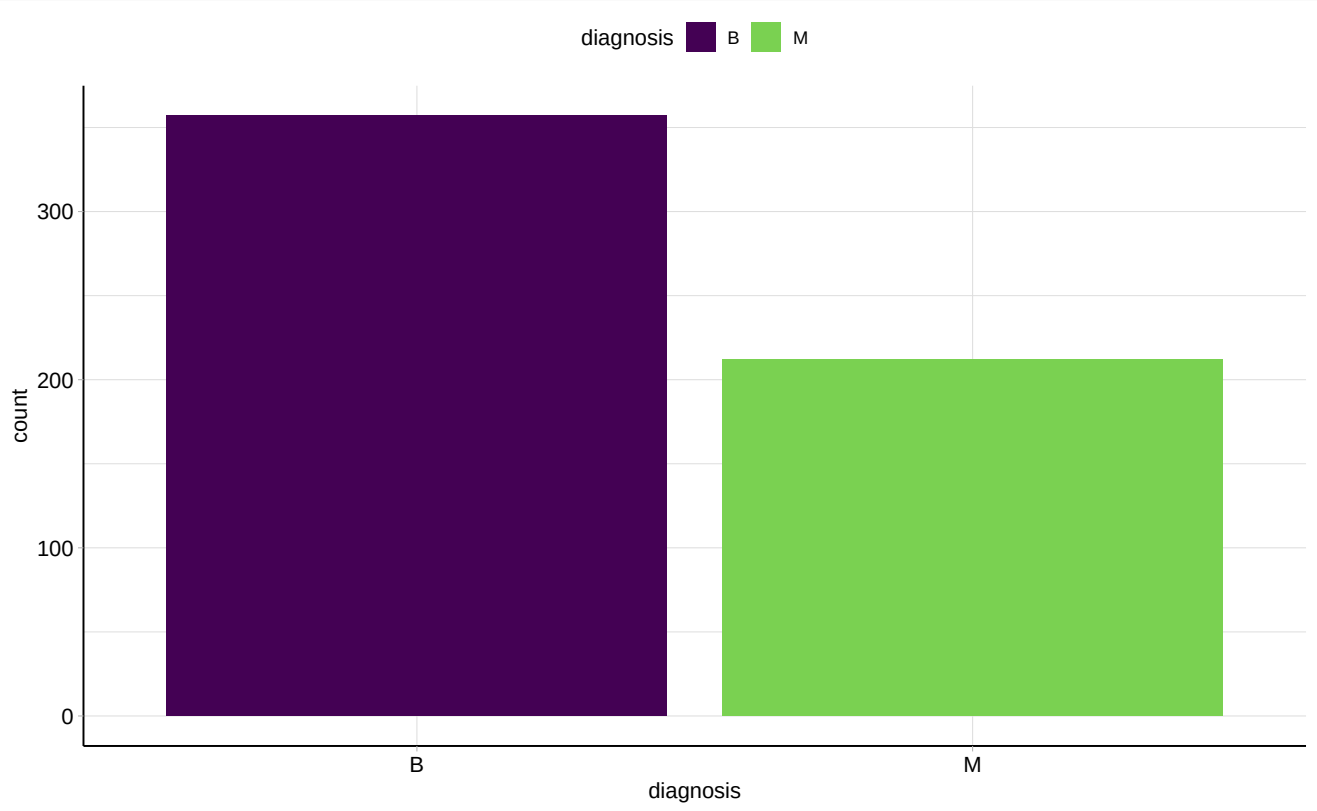
```
library(mlr3verse)
library(tidyverse)

## 支持向量分类
df = read_csv("datas/breast-cancer.csv") %>%
  select(-id) %>%
  mutate(diagnosis = factor(diagnosis)) %>%
  set_names(str_replace(names(.), " ", "_"))
head(df, 2)
#> # A tibble: 2 x 31
#>   diagnosis radius_mean texture_mean perimeter_mean area_m
#>   <fct>           <dbl>        <dbl>          <dbl>     <d
#> 1 M                18.0         10.4           123.      1
#> 2 M                20.6         17.8           133.      1
```

2. 创建任务

```
task = as_task_classif(df, target = "diagnosis")
task
#> <TaskClassif:df> (569 x 31)
#> * Target: diagnosis
#> * Properties: twoclass
#> * Features (30):
#>   - dbl (30): area_mean, area_se, area_worst, compactness_
#>     compactness_se, compactness_worst, concave_points_mean
#>     concave_points_se, concave_points_worst, concavity_mea
#>     concavity_se, concavity_worst, fractal_dimension_mean,
#>     fractal_dimension_se, fractal_dimension_worst, perimet
#>     perimeter_se, perimeter_worst, radius_mean, radius_se,
#>     radius_worst, smoothness_mean, smoothness_se, smoothne
#>     symmetry_mean, symmetry_se, symmetry_worst, texture_me
#>     texture_se, texture_worst
```

- 可视化任务

```
autoplot(task)
```



3. 选择学习器

```
learner = lrn("classif.svm",
              type = "C-classification")     # 需要 e1071 包
learner
#> <LearnerClassifSVM:classif.svm>
#> * Model: -
#> * Parameters: type=C-classification
#> * Packages: mlr3, mlr3learners, e1071
#> * Predict Types:  [response], prob
#> * Feature Types: logical, integer, numeric
#> * Properties: multiclass, twoclass
```

4. 划分训练集测试集

- 做留出 (holdout) 重抽样，70% 作为训练集，其余 30% 作为测试集
- 为了保持训练集、测试集的因变量数据具有相似的分布，采用分层抽样方法
- 用 partition() 函数对任务做划分，默认按因变量分层，取出训练集索引和测试集索引

```
set.seed(123)
split = partition(task, ratio = 0.7)
# 默认 stratify = TRUE
```

5. 超参数调参

- 查看学习器的超参数集，其中重要超参数是正则化参数 `cost`, 核函数 `kernel`, 核函数参数 `gamma`.

```
learner$param_set
#> <ParamSet>
#>                     id    class lower upper nlevels
#>  1:           cachesize ParamDbl  -Inf   Inf     Inf
#>  2:     class.weights ParamUty    NA    NA     Inf
#>  3:               coef0 ParamDbl  -Inf   Inf     Inf
#>  4:                cost ParamDbl     0   Inf     Inf
#>  5:               cross ParamInt     0   Inf     Inf
#>  6: decision.values ParamLgl    NA    NA       2
#>  7:              degree ParamInt     1   Inf     Inf
#>  8:             epsilon ParamDbl     0   Inf     Inf
#>  9:              fitted ParamLgl    NA    NA       2
#> 10:               gamma ParamDbl     0   Inf     Inf   <NoDefa
```

设置搜索空间和自动调参器

```
search_space = ps(
  cost = p_dbl(lower = 0.1, upper = 10),
  gamma = p_dbl(lower = 0, upper = 5),
  kernel = p_fct(c("polynomial", "radial")))
at = auto_tuner(
  method = "grid_search",
  learner = learner,
  resampling = rsmp("cv", folds = 3),
  measure = msr("classif.acc"),
  search_space = search_space,
  term_evals = 20,
  resolution = 10)
```

- 在训练集上启动调参过程，并启用并行化加速

```
future::plan("multicore")    # 启动并行化
set.seed(1)
at$train(task, row_ids = split$train)
#> INFO  [11:26:24.926] [bbotk] Starting to optimize 3 parame
#> INFO  [11:26:24.955] [bbotk] Evaluating 1 configuration(s)
#> INFO  [11:26:24.982] [mlr3] Running benchmark with 3 resam
#> INFO  [11:26:25.005] [mlr3] Applying learner 'classif.svm'
#> INFO  [11:26:25.040] [mlr3] Applying learner 'classif.svm'
#> INFO  [11:26:25.070] [mlr3] Applying learner 'classif.svm'
#> INFO  [11:26:25.096] [mlr3] Finished benchmark
#> INFO  [11:26:25.132] [bbotk] Result of batch 1:
#> INFO  [11:26:25.133] [bbotk]  cost gamma     kernel classi
#> INFO  [11:26:25.133] [bbotk]   7.8  4.44 polynomial
#> INFO  [11:26:25.133] [bbotk]
#> INFO  [11:26:25.133] [bbotk]  a67fae53-08fe-4c24-bb32-ccd3
#> INFO  [11:26:25.135] [bbotk] Evaluating 1 configuration(s)
```

- 查看最优超参数

```
at$tuning_result
#>    cost gamma     kernel learner_param_vals  x_domain clas
#> 1:  7.8  4.44 polynomial          <list[4]> <list[3]>
```

- 用调出的最优参数更新学习器的参数集，然后训练模型

```
learner$param_set$values =
  at$tuning_result$learner_param_vals[[1]]
learner$train(task, row_ids = split$train)
```

6. 模型预测与评估

```
predictions = learner$predict(task, row_ids = split$test)
predictions$score(msr("classif.acc"))
#> classif.acc
#>       0.947
```

7. 预测新数据

```
newdata =df[1:5, -1]
learner$predict_newdata(newdata)
#> <PredictionClassif> for 5 observations:
#>  row_ids truth response
#>        1  <NA>        M
#>        2  <NA>        M
#>        3  <NA>        M
#>        4  <NA>        M
#>        5  <NA>        M
```

主要参考文献

[1] mlr3book. 2021. https://mlr3book.mlr-org.com/

[2] July, 支持向量机通俗导论：理解 SVM 的三层境界. 结构之法算法之道 blog: https://blog.csdn.net/v_july_v/article/details/7624837. 2016.

[3] 刘顺祥. 从零开始学 Python 数据分析与挖掘. 清华大学出版社. 2018.

[4] Jason. 知乎专栏：数据科学笔记本. Learn R | SVM of Data Mining, 2017.

[5] Schölkopf, Bernhard, Smola, Alex J.,Williamson, Robert C., & Bartlett, Peter L. New support vector algorithms. Neural Computation, 2000, 12(5): 1207–1245.

[6] Bin Gua, Victor S. Sheng. Incremental learning for v-Support Vector Regression. Neural Networks 2015, 67: 140–150.