# 机器学习导论
# 第五章

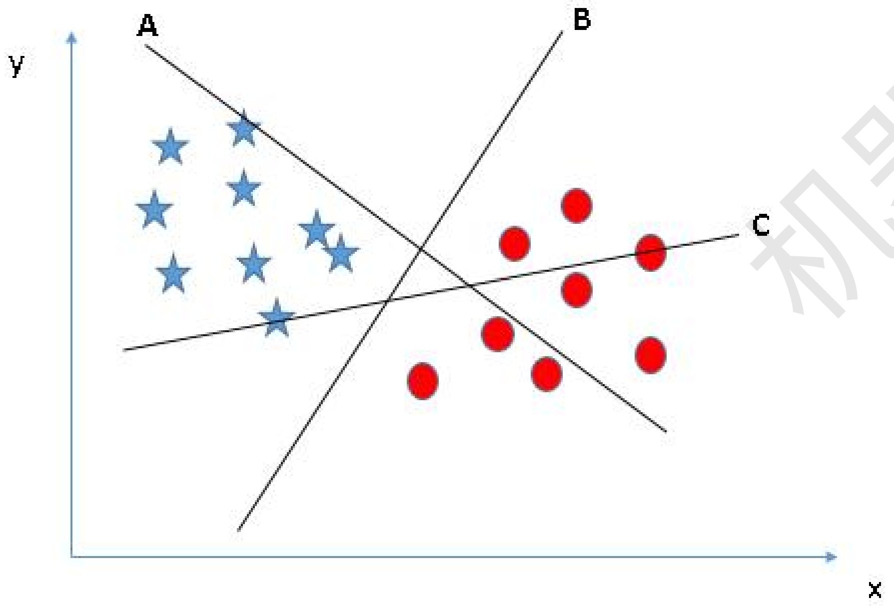
王小航

# 本章概要

- 支持向量机（Support Vector Machine, SVM）
  - 硬间隔支持向量机
  - 软间隔支持向量机
  - 非线性支持向量机
- 支持向量回归（Support Vector Regression，SVR）
- Scikit-Learn代码实现

# 硬间隔支持向量机

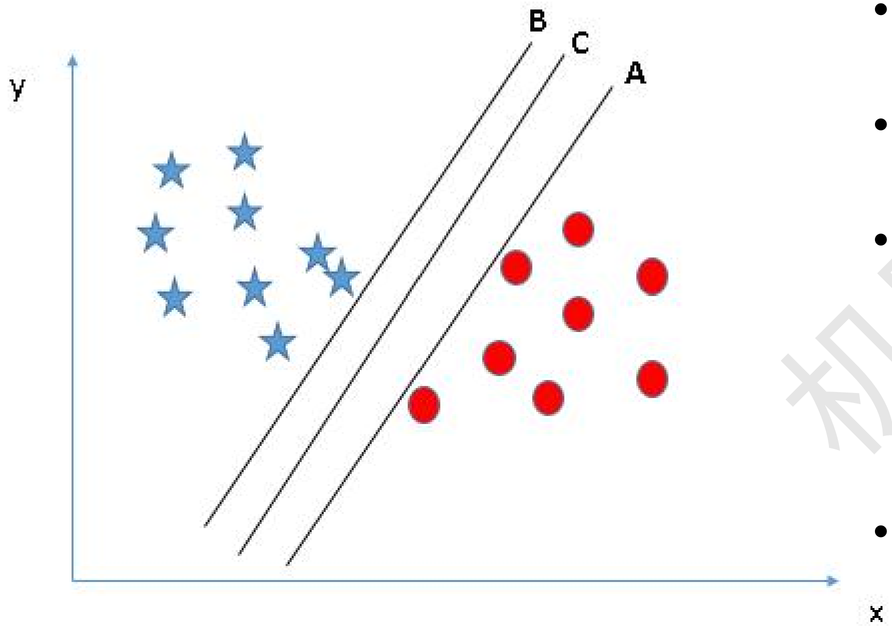
- 给定训练样本集$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m),\ y_i\in\{-1,\ +1\}$，分类学习最近本的想法就是基于训练集D在样本空间中找到一个划分超平面，将不同类别的样本分开。




对于图有A、B、C三个超平面，很明显应该选择超平面B，也就是说超平面首先应该能满足将两类样本点分开。

# 硬间隔支持向量机

- 能将训练样本分开的划分超平面可能有很多，应该如何选择?




- 直观上看，应该去找位于两类训练样本“正中间”的划分超平面，即红色的那个
- 因为该划分超平面对训练样本局部扰动的“容忍”性最好
- 例如，由于训练集的局限性或噪声的干扰，训练集外的样本可能比图中的训练样本更接近两个类目前的分隔界，在分类决策的时候就会出现错误，而超平面C受影响最小
- 也就是说超平面C所产生的分类结果是最鲁棒性的、是最可信的，对未见样本的泛化能力最强。

# 硬间隔支持向量机

- 在样本空间中，划分超平面可通过以下线性方程来描述：

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0$$

其中$\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$为法向量，决定了超平面的方向；$b$为位移项，决定了超平面与原点之间的距离，将超平面记为$(\boldsymbol{w}, b)$

- 样本空间中任意点$\boldsymbol{x}$到超平面$(\boldsymbol{w}, b)$的距离可写为：

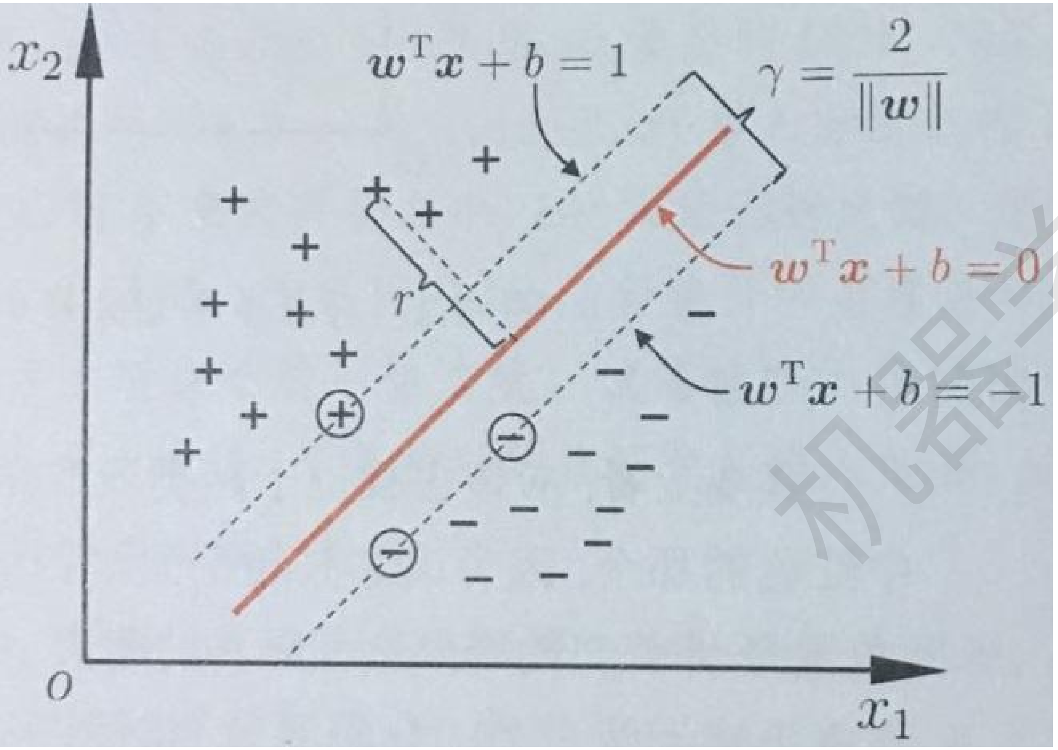
$$r = \frac{|\boldsymbol{w}^T\boldsymbol{x} + b|}{\|\boldsymbol{w}\|}$$

# 硬间隔支持向量机

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

$$\begin{cases} \boldsymbol{w}^T\boldsymbol{x}_i + b \geq +1, & y_i = +1 \\ \boldsymbol{w}^T\boldsymbol{x}_i + b \leq -1, & y_i = -1 \end{cases}$$

# 硬间隔支持向量机




$$\begin{cases} \boldsymbol{w}^T\boldsymbol{x}_i+b \geq +1, & y_i=+1 \\ \boldsymbol{w}^T\boldsymbol{x}_i+b \leq -1, & y_i=-1 \end{cases}$$

距离最佳超平面 最近的几个训练样本点使上式中的等号成立，它们被称为“支持向量”（support vector）

# 硬间隔支持向量机

- 记超平面$\boldsymbol{w}^T\boldsymbol{x}_i + b = +1$和$\boldsymbol{w}^T\boldsymbol{x}_i + b = -1$之间的距离为$\gamma$，该距离又被称为“间隔”（margin）：

$$\gamma = \frac{2}{\|\boldsymbol{w}\|}$$

- SVM的核心之一就是想办法将“间隔”$\gamma$最大化
- 也即

$$\max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|}$$
$$s.t.\ \ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1,\ \ \ i = 1,2,\ldots,m$$

# 硬间隔支持向量机

- 上式等价于

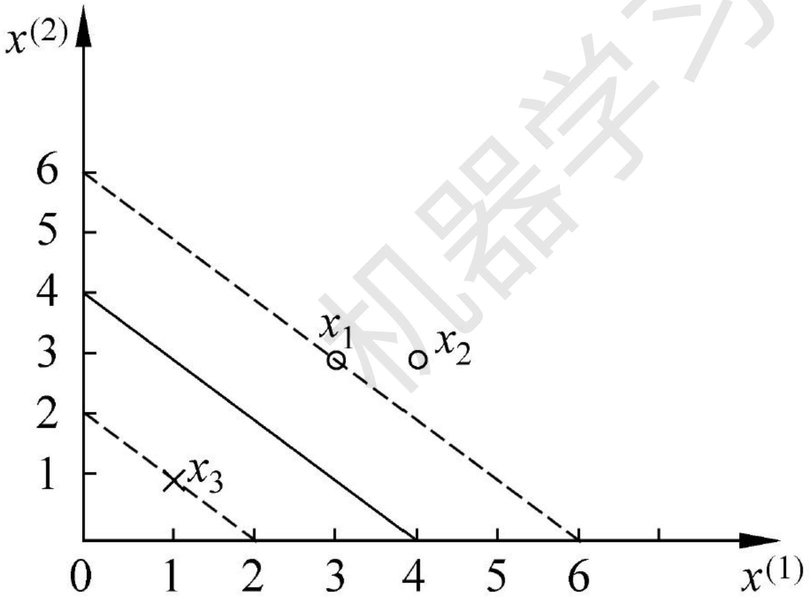
$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$s.t.\ \ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1,\quad i = 1,2,\ldots,m$$

这是支持向量基的基本型

# 硬间隔支持向量机

- 例：已知一个如图所示的训练数据集，其正例点是$x_1 = (3,3)^T$， $x_2 = (4,3)^T$负例点是$x_3 = (1,1)^T$，试求最大间隔分离超平面。



# 硬间隔支持向量机

- 根据训练数据集构造约束最优化问题：

$$\min_{w,b} \frac{1}{2}(w_1^2 + w_2^2)$$
$$s.t.\ \ 3w_1 + 3w_2 + b \geq 1$$
$$4w_1 + 3w_2 + b \geq 1$$
$$-w_1 - w_2 - b \geq 1$$

求得此最优化问题的解$w_1 = w_2 = \frac{1}{2}$，$b_1 = -2$。于是最大间隔分离超平面为

$$\frac{1}{2}x^{(1)} + \frac{1}{2}x^{(2)} - 2 = 0$$

其中$x_1 = (3,3)^T$和$x_3 = (1,1)^T$为支持向量。

# 硬间隔支持向量机

- 上一页PPT的公式是一个凸二次规划问题，我们可以将该凸二次规划问题通过拉格朗日对偶性来解决
- 对上一页PPT的公式的每条约束添加拉格朗日乘子$\alpha_i \geq 0$，则该问题的拉格朗日函数可写为：

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

- 令$\mathrm{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})$对$\boldsymbol{w}$和$b$的偏导为0可得：

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$$

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

# 硬间隔支持向量机

- 可得支持向量基的基本型的对偶问题：

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

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

$$\alpha_i \geq 0, \quad i = 1,2,\ldots,m$$

- 解出$\boldsymbol{\alpha}$后，求出$\boldsymbol{w}$和$b$即可得到模型

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

# 硬间隔支持向量机

- 例：与上例的训练数据集相同，其正例点是$x_1 = (3,3)^T$，$x_2 = (4,3)^T$负例点是$x_3 = (1,1)^T$，试用算法求线性可分支持向量机。

# 硬间隔支持向量机

- 根据所给数据，对偶问题是

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j - \sum_{i=1}^{m}\alpha_i$$

$$= \frac{1}{2}(18\alpha_1^2 + 25\alpha_2^2 + 2\alpha_3^2 + 42\alpha_1\alpha_2 - 12\alpha_1\alpha_3 - 14\alpha_2\alpha_3) - \alpha_1 - \alpha_2 - \alpha_3$$

$$s.t.\ \ \alpha_1 + \alpha_2 + \alpha_3 = 0$$

$$\alpha_i \geq 0,\ \ i = 1,2,3$$

- 将$\alpha_3 = \alpha_1 + \alpha_2$代入目标函数并记为：

$$s(\alpha_1, \alpha_2) = 4\alpha_1^2 + \frac{13}{2}\alpha_2^2 + 10\alpha_1\alpha_2 - 2\alpha_1 - 2\alpha_2$$

- 对$\alpha_1, \alpha_2$求偏导数并令其为0，可得$s(\alpha_1, \alpha_2)$在点$(\frac{3}{2}, -1)$取极值，但该点不满足约束条件$\alpha_2 \geq 0$，所以最小值应在边界上达到。

# 硬间隔支持向量机

- 当$\alpha_1=0$，最小值$s\left(0,\frac{2}{13}\right)=-\frac{2}{13}$； $\alpha_2=0$，最小值$s\left(\frac{1}{4},0\right)=-\frac{1}{4}$。于是$s(\alpha_1,\alpha_2)$在$\alpha_1=\frac{1}{4}$, $\alpha_2=0$达到最小，此时$\alpha_3=\alpha_1+\alpha_2=\frac{1}{4}$
- 计算得

$$w_1=w_2=\frac{1}{2}$$
$$b=-2$$

- 分离超平面为

$$\frac{1}{2}x^{(1)}+\frac{1}{2}x^{(2)}-2=0$$

- 分类决策函数为

$$f(\boldsymbol{x})=sign(\frac{1}{2}x^{(1)}+\frac{1}{2}x^{(2)}-2)$$

# 硬间隔支持向量机

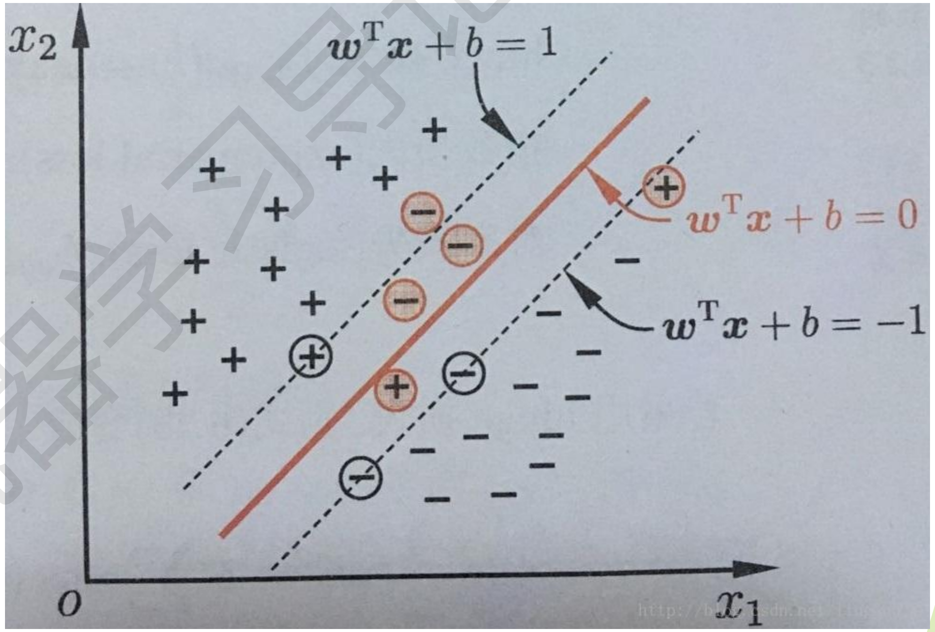
- 在式中有不等式约束，因此上述过程满足Karush-Kuhn-Tucker(KKT)条件：

$$\begin{cases} \alpha_i \geq 0 \\ y_i f(\boldsymbol{x}_i) - 1 \geq 0 \\ \alpha_i (y_i f(\boldsymbol{x}_i) - 1) = 0 \end{cases}$$

- 于是，对任意训练样本$(\boldsymbol{x}_i, y_i)$，总有$\alpha_i = 0$或$y_i f(\boldsymbol{x}_i) = 1$。若$\alpha_i = 0$，则该样本不会在$f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b$的求和中出现，也就不会对$f(\boldsymbol{x})$有任何影响；若$\alpha_i > 0$，则必有$y_i f(\boldsymbol{x}_i) = 1$，所对应的样本点位于最大间隔边界上，是一个支持向量。
- 这写实处支持向量机的一个重要性质：训练完成后，大部分的训练样本不需要保留，最终模型仅与支持向量有关。
- 上例中的$\alpha_1 = \alpha_3 = \frac{1}{4}$，对应的实例点$x_1$，$x_3$是支持向量。

# 软间隔支持向量机

- 在现实任务中很难找到一个超平面将不同类别的样本完全划分开，即很难找到合适的核函数使得训练样本在特征空间中线性可分。
- 退一步说，即使找到了一个可以使训练集在特征空间中完全分开的核函数，也很难确定这个线性可分的结果是不是由于过拟合导致的。
- 解决该问题的办法是在一定程度上运行SVM在一些样本上出错，为此引入了“软间隔”（soft margin）的概念



# 软间隔支持向量机

- 硬间隔支持向量机要求所有的样本均被最佳超平面正确划分，而软间隔支持向量机允许某些样本点不满足间隔大于等于1的条件$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$
- 在最大化间隔的时候也要限制不满足间隔大于等于1的样本的个数使之尽可能的少
- 引入一个惩罚系数 $C > 0$，并对每个样本点$(x_i, y_i)$引入一个松弛变量（slack variables）$\xi \geq 0$

# 软间隔支持向量机

$$\min_{\boldsymbol{w},b} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i\right)$$

$$s.t.\ \ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i, \quad i = 1,2,\ldots,m$$

$$\xi_i \geq 0$$

- 上式中约束条件改为$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$，表示间隔加上松弛变量大于等于1
- 优化目标改为$\min_{\boldsymbol{w},b} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i\right)$，表示对每个松弛变量都要有一个代价损失$C\xi_i$，$C$越大对误分类的惩罚越大、$C$越小对误分类的惩罚越小。

# 软间隔支持向量机

- 假设求解软间隔支持向量机间隔最大化问题得到的最佳超平面$\boldsymbol{w}^T\boldsymbol{x}+b=0$，对应的分类决策函数为$\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}+b)$，称为软间隔支持向量机
- 利用拉格朗日乘子法可得到上式的拉格朗日函数：

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

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

# 软间隔支持向量机

- 令$\mathrm{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu})$分别对$\boldsymbol{w}, b, \boldsymbol{\xi}$求偏导并将偏导数为零可得：

$$\begin{cases} \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \\ \sum_{i=1}^{m} \alpha_i y_i = 0 \\ \mathrm{C} = \alpha_i + \mu_i \end{cases}$$

# 软间隔支持向量机

- 可得到对偶问题：

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

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

$$0 \le \alpha_i \le C, \quad i = 1,2,\dots,m$$

- 对比软间隔支持向量机的对偶问题和硬间隔支持向量机的对偶问题可发现二者的唯一差别就在于对偶变量的约束不同，软间隔支持向量机对对偶变量的约束是 $0 \le \alpha_i \le C$，硬间隔支持向量机对对偶变量的约束是 $0 \le \alpha_i$，于是可采用和硬间隔支持向量机相同的解法求解。

# 软间隔支持向量机

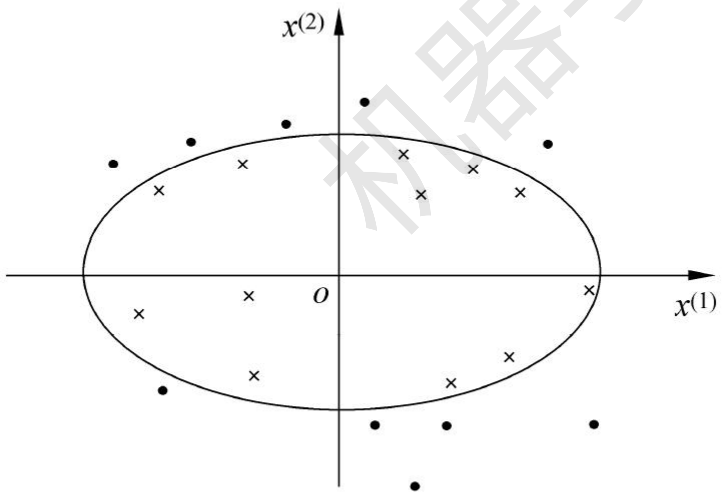
- 对于软间隔支持向量机，KKT条件要求：

$$\begin{cases} \alpha_i \geq 0, \mu_i \geq 0 \\ y_i f(\boldsymbol{x}_i) - 1 + \xi_i \geq 0 \\ \alpha_i (y_i f(\boldsymbol{x}_i) - 1 + \xi_i) = 0 \\ \xi_i \geq 0,\ \mu_i \xi_i = 0 \end{cases}$$

- 同硬间隔支持向量机类似，对任意训练样本 $(\boldsymbol{x}_i, y_i)$，总有 $\alpha_i = 0$或 $y_i f(\boldsymbol{x}_i) - 1 + \xi_i = 0$，若 $\alpha_i = 0$，则该样本不会对最佳决策面有任何影响；若 $\alpha_i > 0$则必有 $y_i f(\boldsymbol{x}_i) = 1 - \xi_i$，也就是说该样本是支持向量。
- 若 $\alpha_i < C$则$\mu_i > 0$进而有 $\xi_i = 0$，即该样本处在最大间隔边界上；若 $\alpha_i = C$则 $\mu_i = 0$此时如果$\xi_i \leq 1$则该样本处于最大间隔内部，如果 $\xi_i > 1$则该样本处于最大间隔外部即被分错了。由此也可看出，软间隔支持向量机的最终模型仅与支持向量有关。

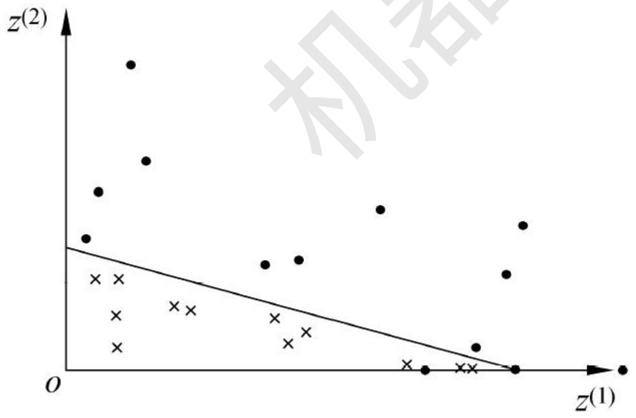
# 非线性支持向量机

- 对解线性分类问题，线性分类支持向量机是一种非常有效的方法。但是，有时分类问题是非线性的，这时可以使用非线性支持向量机
- 如图，是一个分类问题。由图可见，无法用直线（线性模型）将正负实例正确分开，但可以用一条椭圆曲线（非线性模型）将它们正确分开



# 非线性支持向量机

- 非线性问题往往不好求解，所以希望能用解线性分类问题的方法解决这个问题.
- 所采取的方法是进行一个非线性变换，将非线性问题变换为线性问题，通过解变换后的线性问题的方法求解原来的非线性问题.
- 对上图所示的例子，通过变换，将左图中椭圆变换成右图中的直线，将非线性分类问题变换为线性分类问题.



# 非线性支持向量机

- 设原空间为$\mathcal{X} \subset R^2$，$\boldsymbol{x} = \left(x^{(1)}, x^{(2)}\right)^T \in \mathcal{X}$，新空间为$\mathcal{Z} \subset R^2$， $\boldsymbol{z} = \left(z^{(1)}, z^{(2)}\right)^T \in \mathcal{Z}$。
- 定义从原空间到新空间的变换（映射）：

$$z = \phi(\boldsymbol{x}) = \left(\left(x^{(1)}\right)^2, \left(x^{(2)}\right)^2\right)^T$$

经过变换$z = \phi(x)$，原空间$\mathcal{X} \subset R^2$变换为新空间$\mathcal{Z} \subset R^2$，原空间中的点相应地变换为新空间中的点，原空间中的椭圆

$$w_1\left(x^{(1)}\right)^2 + w_2\left(x^{(2)}\right)^2 + b = 0$$

变换成为新空间中的直线

$$w_1 z^{(1)} + w_2 z^{(2)} + b = 0$$

# 非线性支持向量机

- 将$\phi(\boldsymbol{x})$表示将$\boldsymbol{x}$映射后的特征向量，在特征空间划分超平面所对应的模型可表示为

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) + b$$

- 优化问题为：

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$s.t. \ \ y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1, \quad i = 1,2,\dots,m$$

# 非线性支持向量机

- 其对偶问题为

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

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

$$\alpha_i \geq 0,\ \ i = 1,2, \ldots, m$$

- 引入核函数（kernel function）：

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

# 非线性支持向量机

- 对偶问题可写为：

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

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

$$\alpha_i \geq 0, \quad i = 1,2, \ldots, m$$

- 求解后可得到：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b = \sum_{i=1}^{m} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + b$$

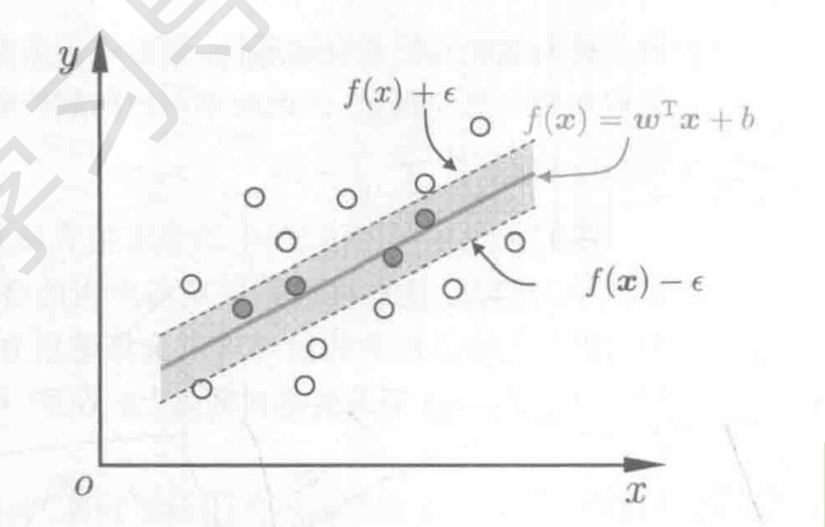
# 非线性支持向量机

## 常用核函数

| 名称 | 表达式 | 参数 |
|---|---|---|
| 线性核 | $\kappa(\vec{x_i},\vec{x_j})=\vec{x_i}^T\vec{x_j}$ | |
| 多项式核 | $\kappa(\vec{x_i},\vec{x_j})=(\vec{x_i}^T\vec{x_j})^n$ | $n\geq 1$为多项式的次数 |
| 高斯核(RBF) | $\kappa(\vec{x_i},\vec{x_j})=exp(-\frac{\|\vec{x_i}-\vec{x_j}\|^2}{2\sigma^2})$ | $\sigma>0$为高斯核的带宽 |
| 拉普拉斯核 | $\kappa(\vec{x_i},\vec{x_j})=exp(-\frac{\|x_i-x_j\|}{\sigma})$ | $\sigma$>0 |
| Sigmoid核 | $\kappa(\vec{x_i},\vec{x_j})=tanh(\beta\vec{x_i}^T\vec{x_j}+\theta)$ | thah为双曲正切函数 |

# 支持向量回归

- 给定训练样本集$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$，$y_i \in \mathbb{R}$，希望学得一个回归模型$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$，使得$f(\boldsymbol{x})$与$y$尽可能接近，$\boldsymbol{w}$和$b$是待确定的模型参数
- 对样本$(\boldsymbol{x}, y)$，传统回归模型基于模型输出$f(\boldsymbol{x})$与真实输出$y$之间的差别来计算损失，当且仅当$f(\boldsymbol{x})$与$y$完全相同时，损失才为0
- 支持向量回归假设我们能容忍$f(\boldsymbol{x})$与$y$之间最多有$\epsilon$的偏差，即仅当$f(\boldsymbol{x})$与$y$之间的差别绝对值大于$\epsilon$时才计算损失。



# 支持向量回归

- SVR的问题可形式化为

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

- $l_\epsilon$是$\epsilon$不敏感损失函数：

$$l_\epsilon(z) = \begin{cases} 0, & if\ |z| \le \epsilon \\ |z| - \epsilon, & otherwise \end{cases}$$

# Scikit-Learn代码实现

- 线性支持向量机

```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

iris = datasets.load_iris()
X = iris["data"][:, (2, 3)]  # petal length, petal width
y = (iris["target"] == 2).astype(np.float64)  # Iris virginica

svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge")),
    ])

svm_clf.fit(X, y)

>>> svm_clf.predict([[5.5, 1.7]])
array([1.])
```

# Scikit-Learn代码实现

- 非线性支持向量机（多项式内核）

让我们在卫星数据集上进行测试：这是一个用于二元分类的小数据集，其中数据点的形状为两个交织的半圆

```
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline

X, y = make_moons(n_samples=100, noise=0.15)


from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

超参数coef0控制的是模型受高阶多
项式还是低阶多项式影响
的程度

# Scikit-Learn代码实现

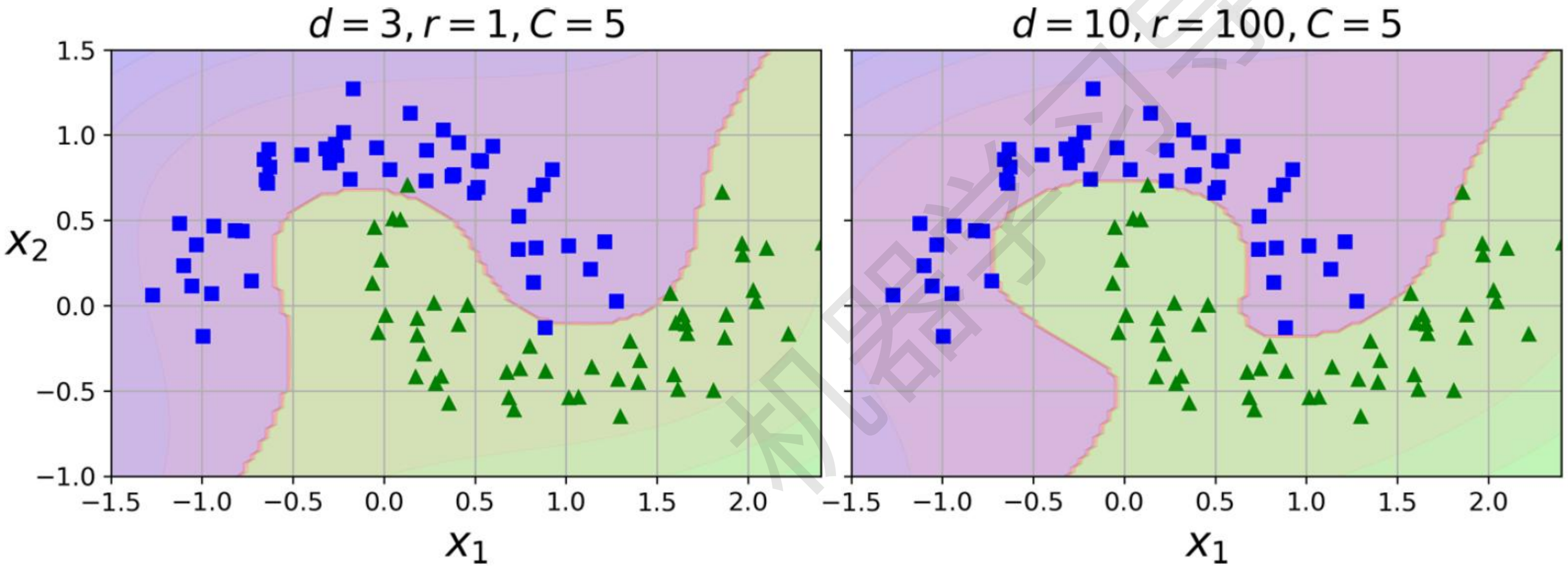



*Figure 5-7. SVM classifiers with a polynomial kernel*

# Scikit-Learn代码实现

非线性支持向量机（高斯RBF内核）

```
rbf_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ])
rbf_kernel_svm_clf.fit(X, y)
```

# Scikit-Learn代码实现

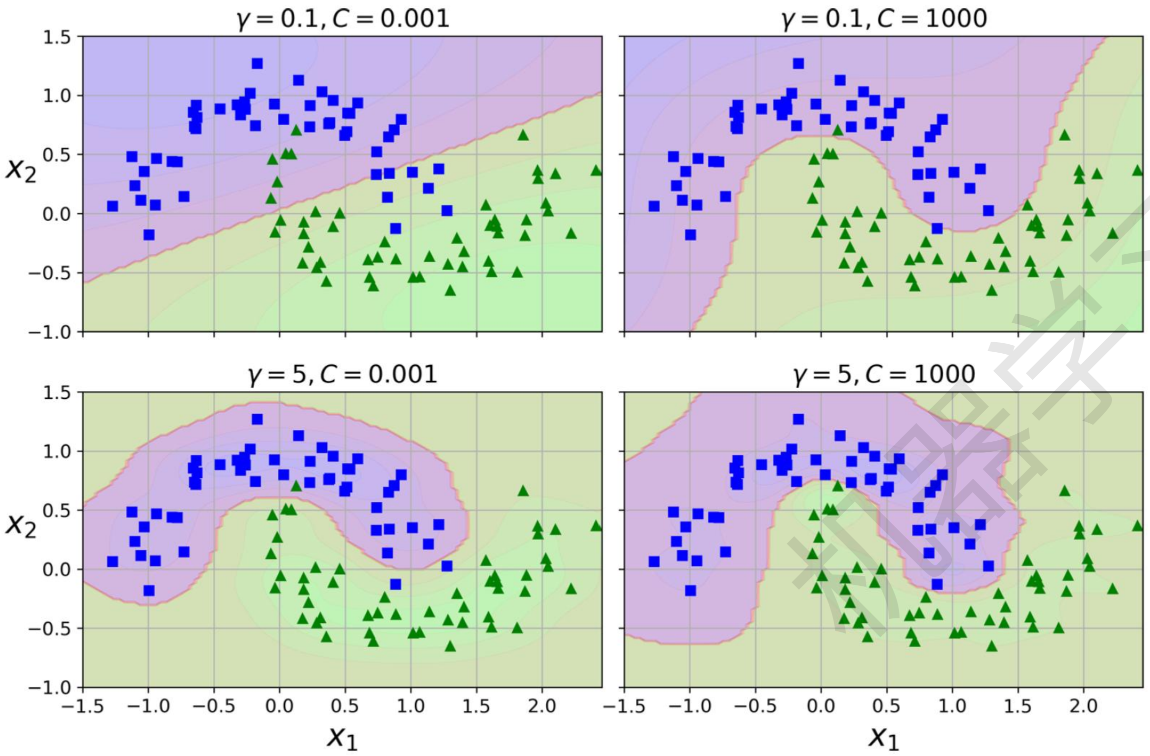



*Figure 5-9. SVM classifiers using an RBF kernel*

- 增加γ值会使钟形曲线变得更窄（左图），因此每个实例的影响范围随之变小：决策边界变得更不规则，开始围着单个实例绕弯。
- 反过来，减小γ值使钟形曲线变得更宽，因而每个实例的影响范围增大，决策边界变得更平坦。所以γ就像是一个正则化的超参数：模型过拟合，就降低它的值，如果欠拟合则提升它的值

# Scikit-Learn代码实现

- 支持向量回归

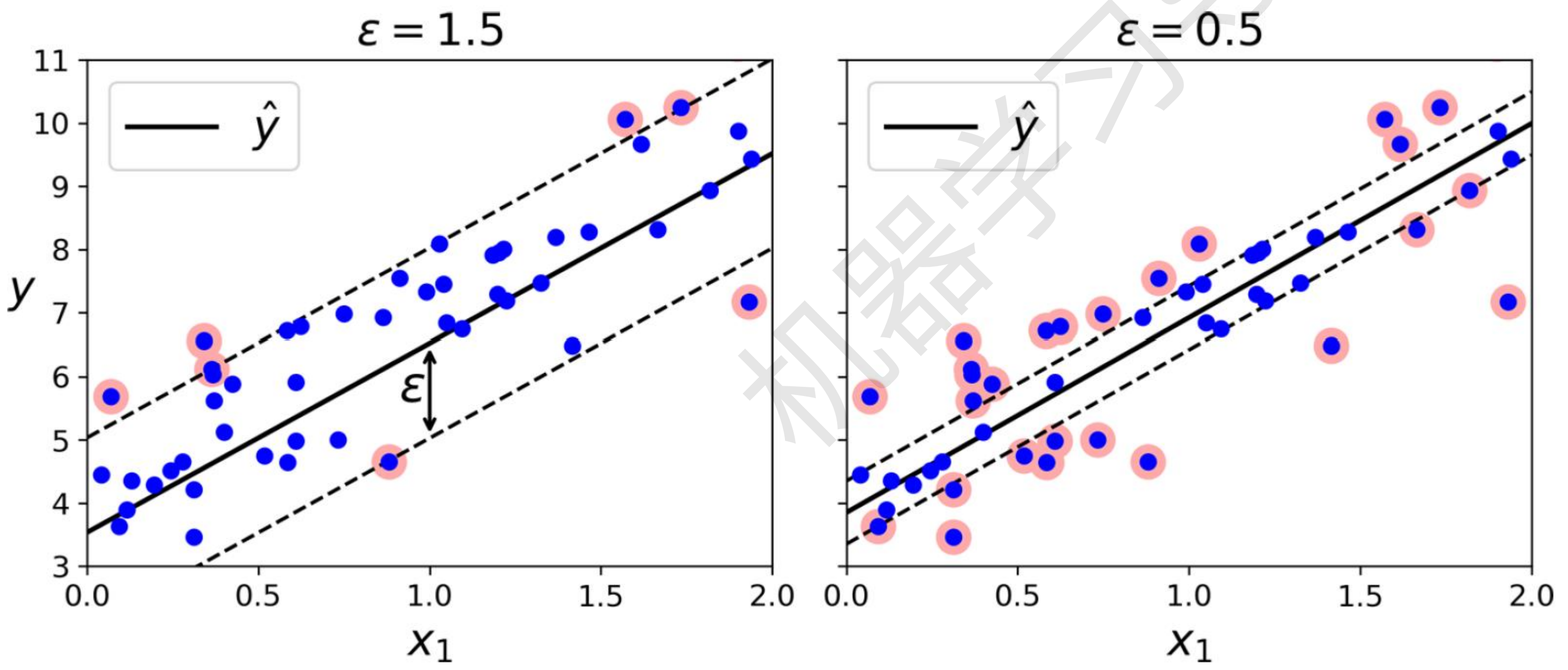



*Figure 5-10. SVM Regression*

# Scikit-Learn代码实现

- 支持向量回归

**线性**

```
from sklearn.svm import LinearSVR

svm_reg = LinearSVR(epsilon=1.5)
svm_reg.fit(X, y)
```

**非线性**

```
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1)
svm_poly_reg.fit(X, y)
```

# Scikit-Learn代码实现

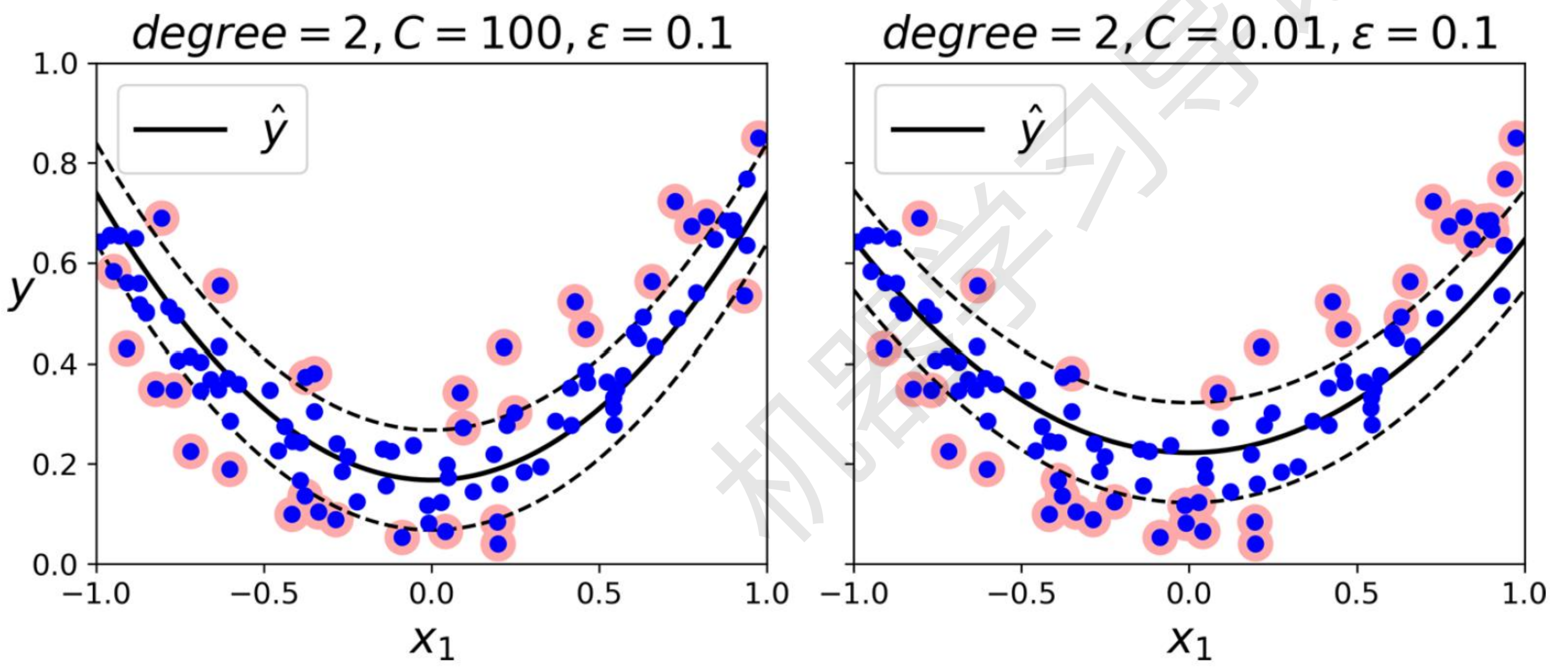



*Figure 5-11. SVM Regression using a second-degree polynomial kernel*