目录

SVM

【参考】

- 干货 | 从超平面到SVM（一）
- SVM（1）：理清分离超平面方程和法向量
- 支持向量机（SVM）的分析及python实现 使用 sklearn
- 支持向量机SVM通俗理解（python代码实现） 手动实现代码
- 机器学习算法与Python实践之（二）支持向量机（SVM）初级

【推荐】

- pluskid - 支持向量机: Maximum Margin Classifier
- pluskid - 支持向量机: Support Vector

超平面

超平面的公式可以写为如下形式：

$$\omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n + b = 0 \tag{1}$$

写成ω, $\mathbf{x}$矩阵的形式为：

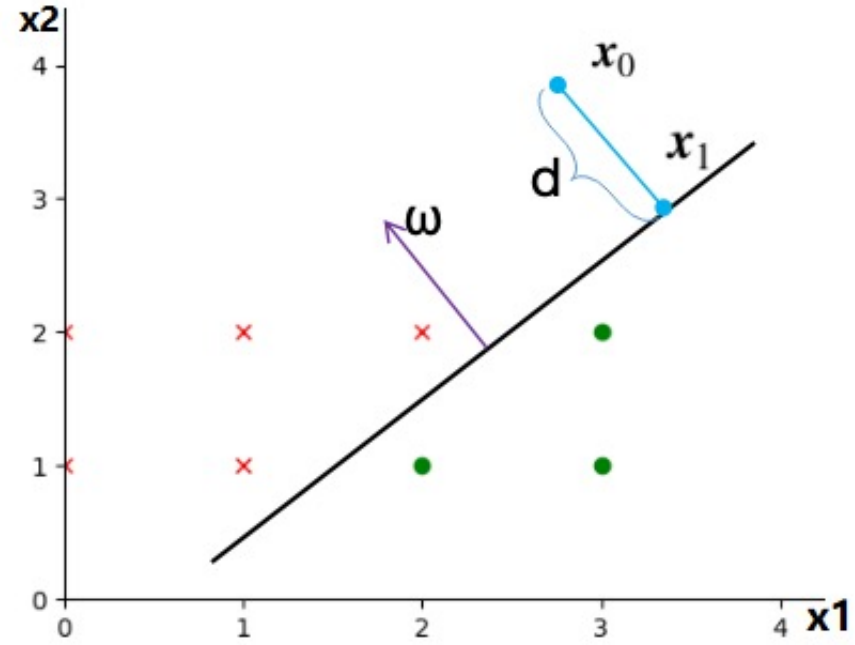
$$\omega^T \boldsymbol{x} = [\omega_1, \omega_2, \cdots, \omega_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n \tag{2}$$

因此 (1) 式又可以写成如下形式：

$$\omega^T \boldsymbol{x} + \mathbf{b} = \mathbf{0} \tag{3}$$

其中 $\omega = (\omega_1, \omega_2, \cdots, \omega_n)$ 为**法向量**，b 为截距，超平面由该方程唯一确定。

点到超平面距离的计算



设点 $\boldsymbol{x}_0$ 在超平面 S：

$\omega^T \boldsymbol{x} + \mathbf{b} = \mathbf{0}$ 上的投影为 $\boldsymbol{x}_1$，那么就有$\omega^T \boldsymbol{x_1} + \mathbf{b} = \mathbf{0}$，于是有：

$$\omega^1 x_1^1 + \omega^2 x_1^2 + \ldots + \omega^n x_1^n = -\boldsymbol{b} \tag{4}$$

令 $\boldsymbol{x}_0$ 在超平面 S 的距离为 d。其中$\boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{\omega}$ 都是 n 维向量。

由于向量$\overrightarrow{x_0 x_1}$ 与 S 平面的法向量$\boldsymbol{\omega}$平行，所以有：

$$|\boldsymbol{\omega} \cdot \overrightarrow{x_0 x_1}| = |\boldsymbol{\omega}||\overrightarrow{x_0 x_1}| = \sqrt{(\omega^1)^2 + \ldots + (\omega^1)^n}\, d = ||w|| d \tag{5}$$

又由于：

$$\begin{aligned}\boldsymbol{\omega}\cdot\overrightarrow{x_0x_1} &= \omega^1(x_0^1-x_1^1)+\omega^2(x_0^2-x_1^2)+\ldots+\omega^n(x_0^n-x_1^n)\\ &= \omega^1x_0^1+\omega^2x_0^2+\ldots+\omega^nx_0^n-(\omega^1x_1^1+\omega^2x_1^2+\ldots+\omega^nx_1^n)\\ &= \omega^1x_0^1+\omega^2x_0^2+\ldots+\omega^nx_0^n-(-\boldsymbol{b})\end{aligned}\tag{6}$$

由式子（5）和（6）可以推出：

$$||w||d=|\omega^1x_0^1+\omega^2x_0^2+\ldots+\omega^nx_0^n+\boldsymbol{b}|=|\boldsymbol{\omega}\cdot\boldsymbol{x}_0+\boldsymbol{b}|$$

即：

$$d=\frac{1}{||w||}|\boldsymbol{\omega}\cdot\boldsymbol{x}_0+\boldsymbol{b}|\tag{7}$$

线性可分 SVM

定义：（线性可分支持向量机）给定线性可分训练数据集，通过间隔最大化或等价地求解相应的凸二次规划问题得到的分离超平面为:

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

最大，相应的分类决策函数为：

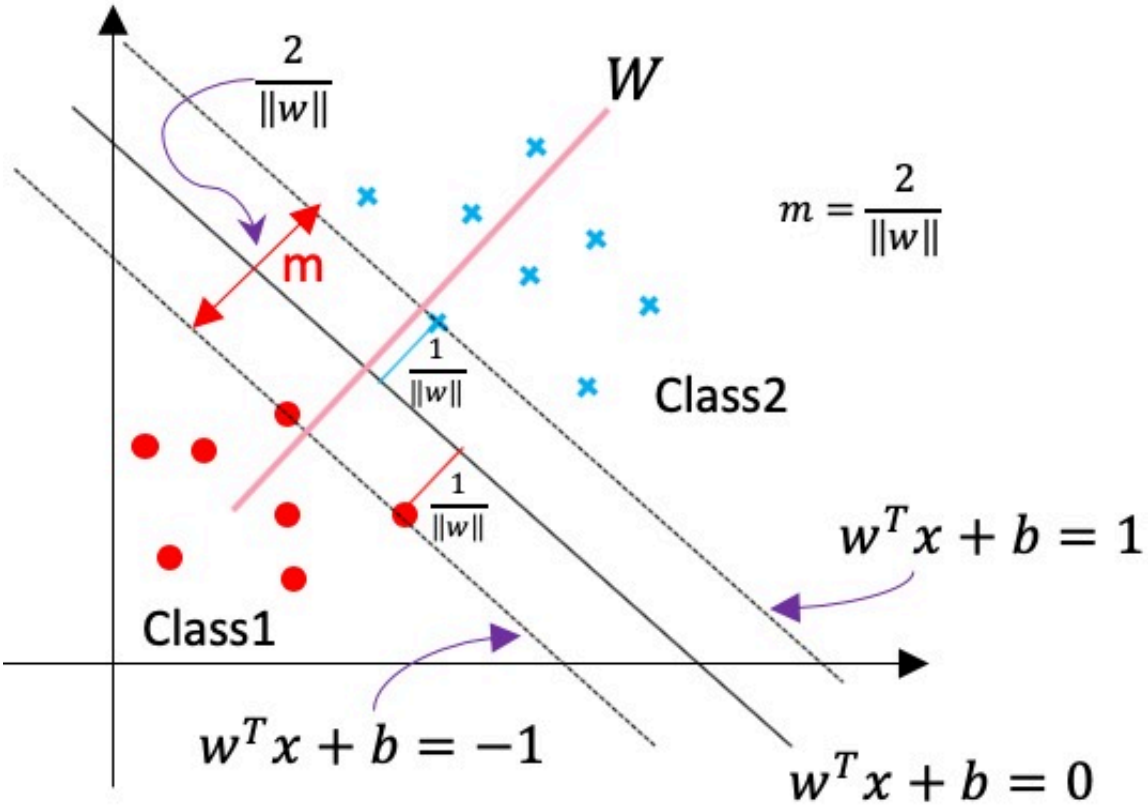
$$f(\boldsymbol{x})=sign(\omega^T\boldsymbol{x}+\mathbf{b})\tag{8}$$

称为线性可分支持向量机。

正如前面二维坐标系里面:

- 在直线上方的样本点带入超平面方程后使得: $\omega^T\boldsymbol{x}+\mathbf{b}>0$，y=1
- 在超平面下方的样本点带入后使得: $\omega^T\boldsymbol{x}+\mathbf{b}<0$ ， y=-1

$\omega^T \boldsymbol{x} + \mathbf{b} = 0$ 就是我们要寻找的分类超平面 。如下图：



H_1、H_2需要满足两个条件：

- （1）没有任何样本在这两个平面之间；
- （2）这两个平面的距离需要最大

假设超平面已经得到了，则 ω 值就是定值，即 $||\omega||$ 也是定的，假设某个样本点为 (x_0, y_0) 根据点到平面距离公式可知$\frac{|\omega^T \boldsymbol{x}_0 + \mathbf{b}|}{||\omega||}$在 $||\omega||$ 定的情况下，$|\omega^T \boldsymbol{x}_0 + \mathbf{b}|$越大，则点到超平面的距离就越大，所以$|\omega^T \boldsymbol{x}_0 + \mathbf{b}|$可以近似代替样本点到超平面的距离

并且由于默认分类正确的话 $\omega^T \boldsymbol{x}_0 + \mathbf{b}$ 正好与 y_0 同号，所以又可以用$y_0(\omega^T \boldsymbol{x}_0 + \mathbf{b})$来表示分类正确，结合二者，$y_0(\omega^T \boldsymbol{x}_0 + \mathbf{b})$可以同时表示分类正确和样本点到超平面的距离。

函数间隔

可参考 支持向量机: Maximum Margin Classifier中的关于函数间隔与几何间隔的论述

为了计算所有样本点到该超平面的距离并且找到最近的样本点，可以定义如下函数间隔：

函数间隔：对于给定的训练数据集 T 和超平面 (ω, b)，可以定义超平面(ω, b)与样本点 (x_i, y_i)的`函数间隔(functional margin)`为：

$$\hat{r}_i = y_i(\omega^T \boldsymbol{x}_i + \mathbf{b}) \tag{9}$$

为了找到最小的那几个样本点（即支持向量），于是可以求出所有样本点距离里面最小的那个：

$$r_{min} = min(\hat{r}_i) \tag{10}$$

不过上面函数间隔存在一个问题就是，假如 ω 和 b 同时成比例变化，超平面方程没有变化，但是函数间隔会变成原来的同比例倍数，为了解决这个问题，又引入了`几何间隔(geometrical margin)`，即

$$\tilde{r}_i = y_i(\frac{\boldsymbol{\omega}^T}{||\omega||}\boldsymbol{x}_i + \frac{\boldsymbol{b}}{||\omega||}) \tag{11}$$

这样就保证了，如果 ω 和 b 成比例变化超平面也会变化。 同时可以看到函数间隔与几何间隔差了一个 $||w||$ 缩放因子。

最大化间隔

在正确分类的前提下，最大化离超平面最近的点到超平面的距离。找到最小间隔的数据点，然后将其最大化并求出参数w,b。

$$arg\max_{\omega,b}\{\min_{n}(y_i \cdot (\boldsymbol{\omega}^T\boldsymbol{x}_i + \boldsymbol{b})) \cdot \frac{1}{||\omega||}\}$$

后面的步骤就和前面说的寻找最好分割直线一样了，如果可以找到一个超平面，使得最近的支持向量离此超平面的距离尽量大。即:

$$\begin{cases} \max_{\omega,b}(\tilde{r}) \\ s.t.\ y_i(\frac{\omega}{||\omega||}x_i + \frac{b}{||\omega||}) \geq \tilde{r},\ i = 1,2,\ldots,n \end{cases}$$
$$\Rightarrow \begin{cases} \max_{\omega,b}(\frac{\hat{r}}{||\omega||}) \\ s.t.\ y_i(\omega x_i + b) \geq \hat{r},\ i = 1,2,\ldots,n \end{cases}$$

其中 $\hat{r} = \tilde{r}\|w\|$，根据前面的讨论，即使在超平面固定的情况下，$\hat{r}$ 的值也可以随着 $\|w\|$ 的变化而变化。由于我们的目标是确定超平面，因此可以将无关的变量固定下来，固定的方式有两种：

- 一种是固定 $\|w\|$ ，当我们找到最优的 $\tilde{r}$ 时，$\hat{r}$ 也可以随着固定
- 第二种是反过来固定 $\hat{r}$ ，此时 $\|w\|$ 也可以根据最优的 $\tilde{r}$ 得到。

为了方便推导和优化，通常选择第二种方式，即令 $\hat{r} = 1$，这样就得到了目标函数的最终形式：

$$\begin{aligned} &\min_{w,b}\frac{1}{2}\|w\|^2 \\ &s.t.\ y_i(w \cdot x_i + b) \geq 1, \forall x_i \end{aligned} \tag{12}$$

到了式（12）就可以很明显地看出来，它是一个凸优化问题，或者更具体地说，它是一个二次优化问题——目标函数是二次的，约束条件是线性的。凸二次规划函数优化，转为更高效的拉格朗日对偶性（其实就是将约束条件融合到目标函数中去）。

对偶性

在约束最优化问题中，常常利用拉格朗日对偶性将原始问题转换为对偶问题，通过求解对偶问题而得到原始问题的解。。假设我们的优化问题是：

$$\begin{aligned} &\min f(x) \\ &s.t.\ h_i(x) = 0, i = 1,2,\ldots,n \end{aligned}$$

这是个带等式约束的优化问题。我们引入拉格朗日乘子，得到拉格朗日函数为：

$$L(\mathbf{x}, \boldsymbol{\alpha}) = f(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_n h_n(x)$$

然后我们将拉格朗日函数对x求极值，也就是对x求导，导数为0，就可以得到α关于x的函数，然后再代入拉格朗日函数就变成：

$$\max W(\boldsymbol{\alpha}) = L(\mathbf{x}(\boldsymbol{\alpha}), \boldsymbol{\alpha})$$

这时候，带等式约束的优化问题就变成只有一个变量$\boldsymbol{\alpha}$（多个约束条件就是向量）的优化问题，这时候的求解就很简单了。同样是求导另其等于0，解出$\boldsymbol{\alpha}$即可。需要注意的是，我们把原始的问题叫做primal problem(原始问题)，转换后的形式叫做dual problem（对偶问题）。需要注意的是，**原始问题是最小化，转化为对偶问题后就变成了求最大值了**。对于不等式约束，其实是同样的操作。简单地来说，通过给每一个约束条件加上一个 Lagrange multiplier（拉格朗日乘子），我们可以将约束条件融和到目标函数里去，这样求解优化问题就会更加容易。

SVM 的对偶问题

对于SVM，前面提到，其primal problem是以下形式：

$$\begin{aligned} & min \frac{1}{2}\|w\|^2 \\ & s.t.\ y_i(w \cdot x_i + b) \geq 1, \forall x_i \end{aligned}$$

同样的方法引入拉格朗日乘子，我们就可以得到以下拉格朗日函数：

$$\mathcal{L}(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{N} \alpha_i \left(1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b)\right) \tag{13}$$

然后对 $\mathcal{L}(\boldsymbol{w}, b, \alpha)$ 分别求$\mathbf{w}$和b的极值。也就是 $\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 对$\mathbf{w}$和b的梯度为0：

- $\frac{\partial L}{\partial \boldsymbol{w}} = 0$
- $\frac{\partial L}{\partial b} = 0,$

还需要满足α>=0。求解这里导数为0的式子可以得到：

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i} \tag{14}$$

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

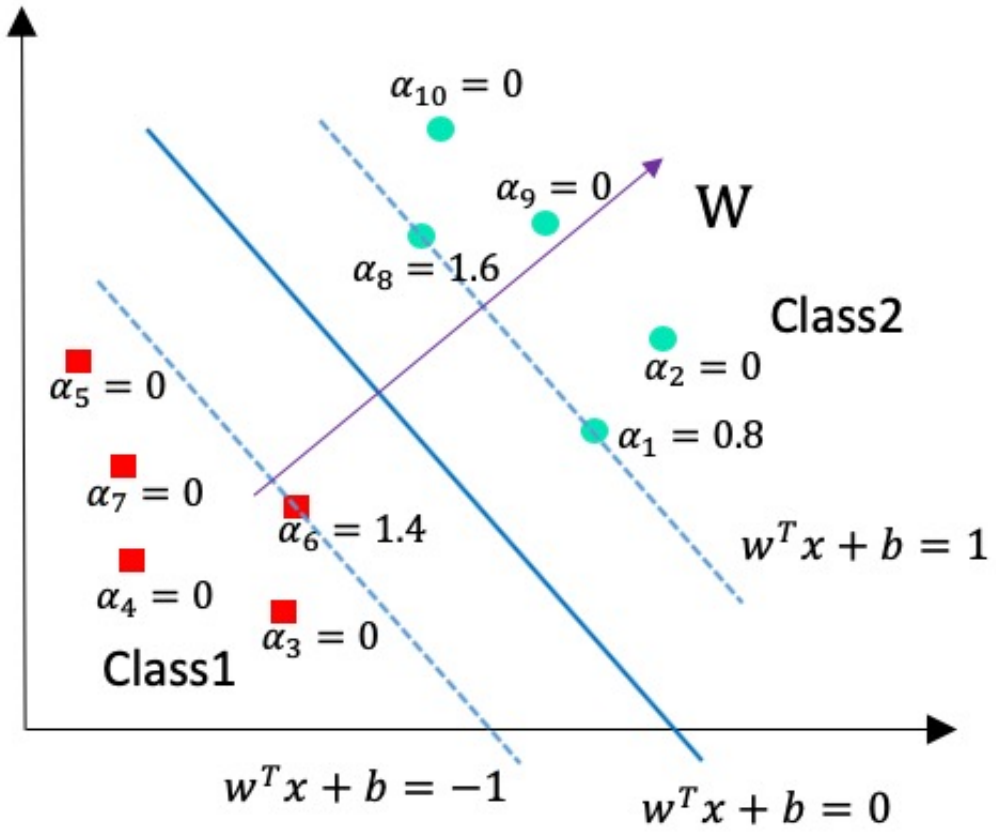
然后再代入拉格朗日函数后，就变成：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,y=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{15}$$
$$s.t.\ \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0$$

这个就是dual problem（如果我们知道α，我们就知道了$\boldsymbol{w}$。反过来，如果我们知道w，也可以知道α）。这时候我们就变成了求对α的极大，即是关于对偶变量α的优化问题（没有了变量$\boldsymbol{w}$，b，只有α）。当求解得到最优的α*后，就可以同样代入到上面的公式，导出*w和b*了，最终得出分离超平面和分类决策函数。也就是训练好了SVM。那来一个新的样本x后，就可以这样分类了：

$$\begin{aligned} f(\boldsymbol{x}) &= sign(\boldsymbol{w} \cdot \boldsymbol{x} + b) \\ &= sign((\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i) \cdot \boldsymbol{x} + b) \\ &= sign(\sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + b) \end{aligned} \tag{16}$$

在这里，其实很多的α_i都是0，也就是说w只是一些少量样本的线性加权值。这种“稀疏”的表示实际上看成是KNN的数据压缩的版本。也就是说，以后新来的要分类的样本首先根据w和b做一次线性运算，然后看求的结果是大于0还是小于0来判断正例还是负例。现在有了α_i，我们不需要求出 w，只需将新来的样本和训练数据中的所有样本做内积和即可。那有人会说，与前面所有的样本都做运算是不是太耗时了？其实不然，我们从KKT条件中得到，只有支持向量的 α_i 不为0，其他情况α_i都是0。因此，我们只需求新来的样本和支持向量的内积，然后运算即可。这种写法为下面要提到的核函数（kernel）做了很好的铺垫。如下图所示：



SVM 优缺点

优点：

- 高维空间有效
- 在维度远远大于样本数时仍然有效
- 在决策函数（称为支持向量）只使用训练样本点的一小部分，因此在内存上也比较有效率
- 多样性：可以为决策函数指定不同的核函数（kernel function），既提供了通用的核，也可以自定义核

缺点：

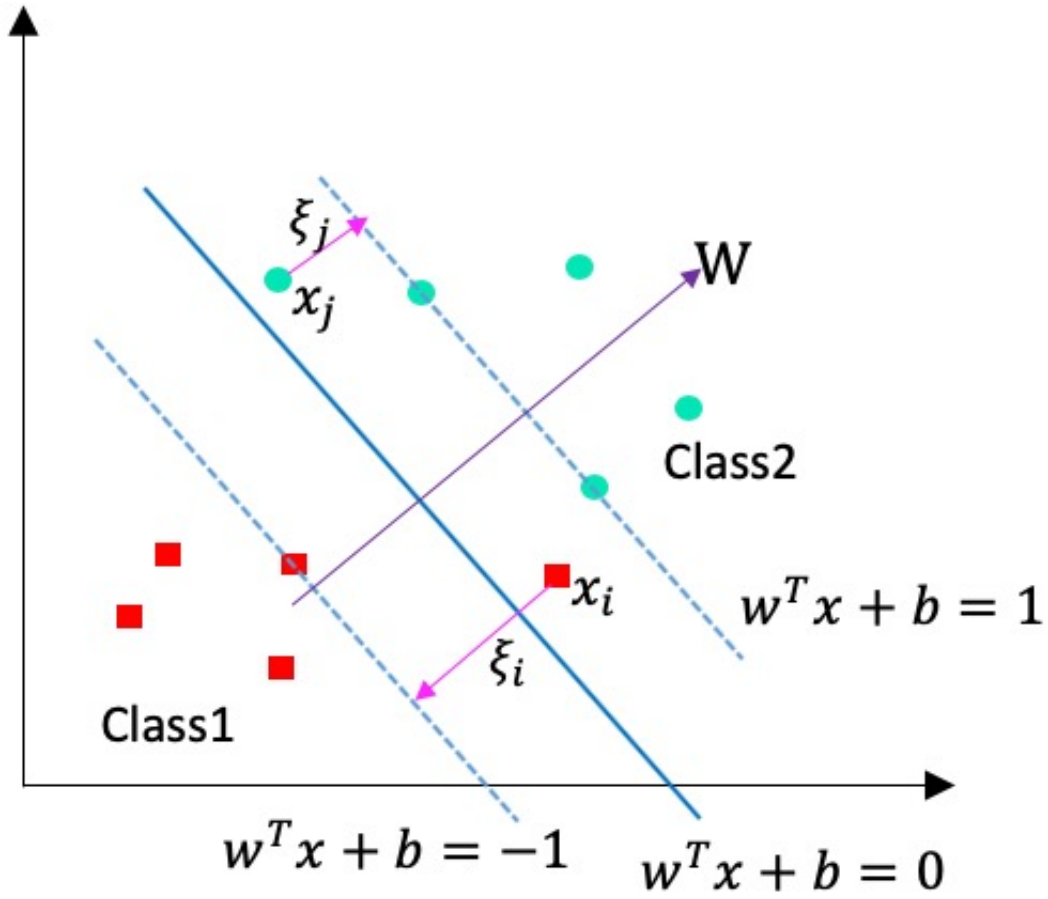
- 在特征数量远大于样本数量是，为了避免过拟合核函数和正则化项非常重要
- SVM 不支持直接的概率评估，而是通过代价昂贵的五折交叉样本计算而来

软间隔与松弛向量

软件间隔

在前面的讨论中，我们一直假设训练样本在样本空间中是**线性可分**的，即存在一个超平面能见不同类的样本完全划分开。但在现实任务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，退一步说，即便恰好找到了某个核函数是训练集在特征空间中线性可分，也很难确定这个貌似线性可分的结果不是由于过拟合造成的。

缓解该问题的一个方法是**允许支持向量机在一些样本上出错**，为此就需要引入**软间隔（soft margin）**的概念，如下图：



即，在前面介绍的支持向量机形式是要求所有样本均满足式（12）的约束条件，即所有的样本都必须能划分正确，这称为**硬间隔（hard margin）**，而软间隔则是允部分样本不满足约束

$$y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b) \geq 1 \tag{17}$$

在最化大间隔的同时，不满足约束的样本应尽可能的少，于是优化的目标就可以写成：

$$\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \ell_{0/1}\left(y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b) - 1\right) \tag{18}$$

其中 $C > 0$ 是一个常数，$\ell_{0/1}$ 是`0/1损失函数`：

$$\ell_{0/1}(z)\begin{cases} 1, & if\ z < 0 \\ 0, & otherwise \end{cases}$$

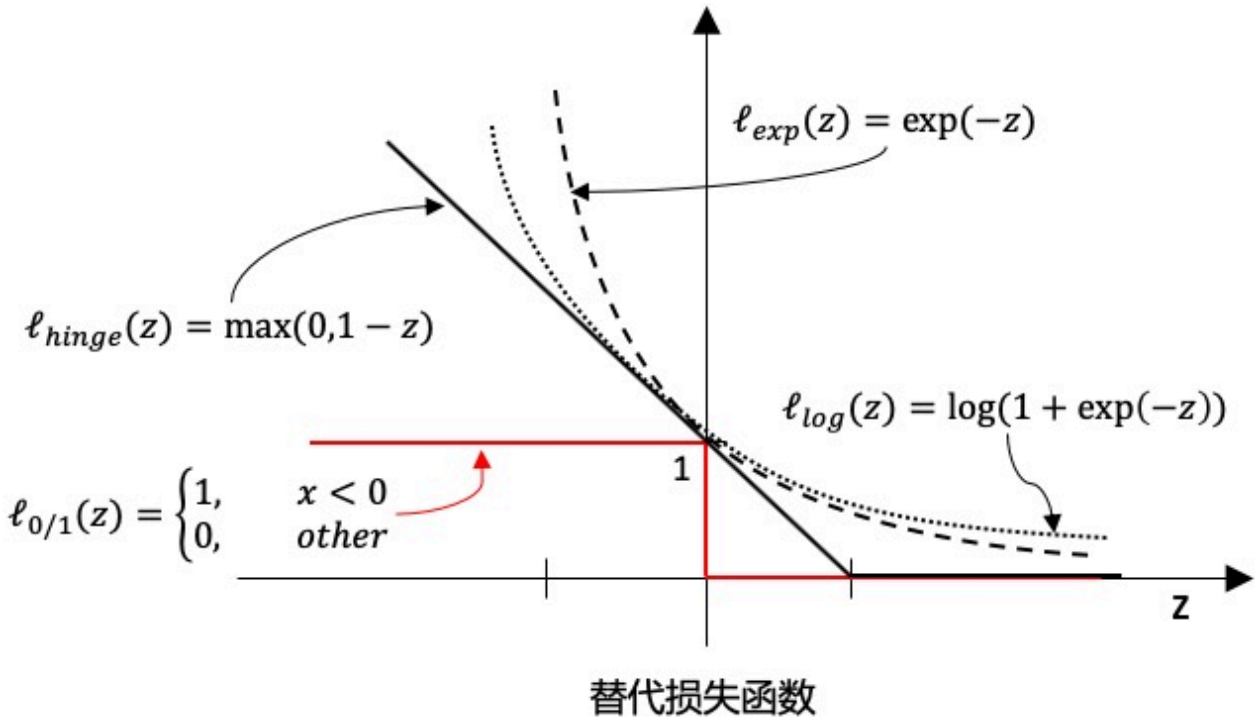
当 C 为无穷大时，式（18）会迫使所有样本满足约束（17），于是式（18）等价于式（12）；当 C 取有限值时，式（18）允许一些样本不满足约束。

替代损失

$\ell_{0/1}$ 损失函数非凸、非连续，数学形式不是特别好，使用式（18）不容易直接求解，于是通常使用一些函数替代$\ell_{0/1}$ 损失函数，称为`替代损失（surrogate loss）`。替代函数一般由较好的数学性质，如通常是凸的连续函数且是$\ell_{0/1}$ 的上界，常用的替代函数有：

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

四个损失函数的对比图如下：



Hinge 损失与松弛变量

若采用 hinge 损失函数，那么式（18）就变成了：

$$\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \max\left(0, 1-y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b)\right) \tag{19}$$

引入`松弛变量（slack variables）` ξ_i，他对应了数据点 $\boldsymbol{x}_i$ 允许偏离的`函数间隔(functional margin)`的量，如果我们运行 ξ_i 任意大的话，那任意的超平面都是符合条件的了。所以，我们在原来的目标函数后面加上一项，使得这些 ξ_i 的总和也要最小，因此可以将上式重写为：

$$\begin{aligned} &\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \xi_i \\ &s.t.\ \ y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \\ &\qquad \xi_i \geq 0,\ i = 1, 2, 3, \ldots, m \end{aligned} \tag{20}$$

这就是常用的`软间隔支持向量机`。其中 C 是一个确定好的参数，用于控制目标函数中两项（“寻找 margin 最大的超平面”和“保证数据点偏差量最小”）之间的平衡，ξ 则是需要优化的变量之一。

约束条件也可以写成如下形式，但没有上面的简洁：

$$\begin{cases} \boldsymbol{\omega}^T \boldsymbol{x}_i + b \geq 1 - \xi_i, & y_i = 1 \\ \boldsymbol{\omega}^T \boldsymbol{x}_i + b \leq -1 + \xi_i, & y_i = -1 \\ \xi_i \geq 0, & \forall i \end{cases}$$

引入非负参数ξ_i后，就允许某些样本点的函数间隔小于1，即在最大间隔区间里面，或者函数间隔是负数，即样本点在对方的区域中。而放松限制条件后，我们需要重新调整目标函数，以对离群点进行处罚，目标函数后面加上的第二项就表示离群点越多，目标函数值越大，而我们要求的是尽可能小的目标函数值。这里的 C 是离群点的权重，是一个事先确定好的常量，C越大表明离群点对目标函数影响越大，也就是越不希望看到离群点。这时候，间隔也会很小。我们看到，目标函数控制了离群点的数目和程度，使大部分样本点仍然遵守限制条件。

拉格朗日对偶形式

【附加】

- pluskid - 支持向量机：Duality

显然，式（20）的每一个样本都对应一个松弛变量，用以表征该样本不满足约束（17）的程度。这仍然是一个二次规划问题，于是亦可以通过拉格朗日乘子法得到（20）的拉格朗日函数：

$$\begin{aligned} \mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) = & \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{m}\xi_i \\ & + \sum_{i=1}^{m}\alpha_i\left(1 - \xi_i - y_i(\boldsymbol{\omega}^T\boldsymbol{x}_i + b)\right) - \sum_{i=1}^{m}\mu_i\xi_i \end{aligned} \tag{21}$$

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

令 $\mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu})$ 对$\boldsymbol{\omega}, b, \boldsymbol{\xi}$ 的偏导为零可以得到：

$$\begin{aligned} \boldsymbol{\omega} &= \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i \\ 0 &= \sum_{i=1}^{m}\alpha_i y_i \\ C &= \alpha_i + \mu_i \end{aligned} \tag{22}$$

将式（22）带入式（21）可以得到式（20）其对偶问题：

$$\begin{aligned} & \max_{\boldsymbol{\alpha}} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j \\ & s.t. \sum_{i=1}^{m}\alpha_i y_i = 0 \\ & \quad 0 \leq \alpha_i \leq C,\ i = 1, 2, \ldots, m \end{aligned} \tag{23}$$

此时，我们发现没有了参数ξ_i，与之前`硬间隔`模型唯一不同在于α_i又多了$\alpha_i \leq C$的限制条件。

KKT 条件

软间隔支持向量机的 KKT 条件要求：

$$\begin{cases} \alpha_i \geq 0, \ \mu_i \geq 0 \\ y_i f(\boldsymbol{x}_i) - 1 + \xi_i \geq 0 \\ \alpha_i (y_i f(\boldsymbol{x}_i) - 1 + \xi_i) = 0 \\ \xi_i \geq 0, \ \mu_i \xi_i = 0 \end{cases} \tag{24}$$

于是，对于任意训练样本$(\boldsymbol{x}_i, y_i)$，总有 $\alpha_i = 0$ 或 $y_i f(\boldsymbol{x}_i) = 1 - \xi_i$.

若 $\alpha_i = 0$ 则该样本不会对 $f(\boldsymbol{x})$有任何影响；
若$\alpha_i > 0$ 则必有$y_i f(\boldsymbol{x}_i) = 1 - \xi_i$，即该样本是**支持向量**：由式（22）第三项可知，

- 若 $\alpha_i < C$，则$\mu_i > 0$，再有 KKT 约束条件的第四项，可知 $\xi_i = 0$，即该样本恰在**最大间隔边界**上
- 若 $\alpha_i = C$，则有$\mu_i = 0$，此时若 $\xi_i \leq 1$ 则样本落在最大间隔的内部，若$\xi_i > 1$ 则样本被错误的分类。

由此可以看出，软间隔支持向量机的最终模型仅与支持向量有关，即通过采用 hinge 损失函数仍然保持了稀疏性。

结构化风险与经验风险

如果使用 对率损失函数 ℓ_{log} 来替代 0/1 损失函数，则几乎就得到了对率回归模型。实际上，支持向量机与对率回归的优化目标相近，通常情形下性能也相当，对率回归的优势在于输出具有自然的概率意义，即在给出预测标记的同时也给出了概率，而支持向量机的输出不具有概率意义，欲得到概率输出需要进行特殊的处理。此外对率回归能直接用于多分类任务，支持向量机为此则需要进行推广。

另外从上图可以看出 hinge 损失有一块 **平坦**的零区域，这使得支持向量机的解具有稀疏性，而对率损失是光滑的单调递减函数，不能导出类似的支持向量的概念，因此对率回归的解依赖于更多的训练样本，其预测开销会更大。

替换成不同的损失函数会得到不同的学习模型，这些模型的性质与所替代的函数直接相关，但他们有一个共性：**优化目标中的第一项用来描述划分超平面的`间隔`大小，另一项$\sum_{i=1}^{m} \ell(f(\boldsymbol{x}_i), y_i)$描述训练集上的误差**，其一般形式为：

$$\min_f \Omega(f) + C \sum_{i=1}^{m} \ell(f(\boldsymbol{x}_i), y_i) \tag{25}$$

其中，$\Omega(f)$ 称为`结构风险（structural risk）`，用于描述模型 f 的某些性质；
$\sum_{i=1}^{m} \ell(f(\boldsymbol{x}_i), y_i)$ 称为`经验风险（empirical risk）`，用于描述模型与训练集的契合程度；
C 是两者的折中。

从经验风险最小化的角度来看，$\Omega(f)$ 表述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），这位引入领域知识和用于意图提供了途径；

另一方面该信息有助于削减假设空间，从而较低了最小化训练误差的过拟合风险，从这个角度来说式（25）称为`正则化问题（regularization）`问题，$\Omega(f)$ 称为`正则化项`，C 称为`正则化常数`。

L_p范数（norm）是常用的正则化项，其中$\boldsymbol{L}_2$ 范数 $\|\boldsymbol{\omega}\|_2$ 倾向于 $\boldsymbol{\omega}$ 的分量取值尽量均衡，即非零量个数尽可能的稠密，而 $\boldsymbol{L}_0$ 范数 $\|\boldsymbol{\omega}\|_0$ 和 $\boldsymbol{L}_1$ 范数 $\|\boldsymbol{\omega}\|_1$ 则倾向于 $\boldsymbol{\omega}$ 的分量尽量系数，即非零量尽可能的少。

核函数

【参考】

- 关于核函数的一些思考

向量的内积与外积意义

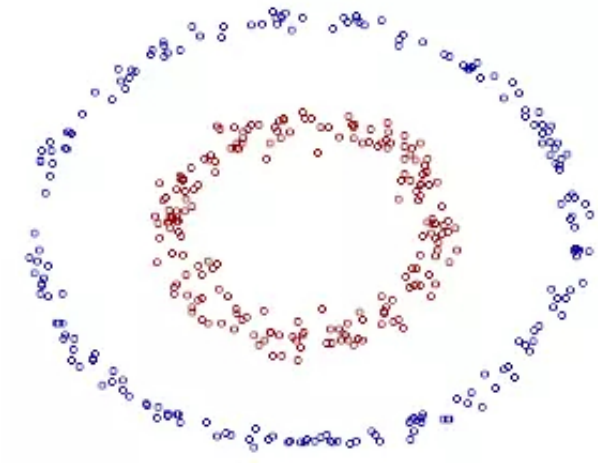
> 首先“内”“外”之分还是挺形象的，内积的结果是定义在空间里的（一个双线性函数，结果是一个数），外积的结果则不是定义在空间里，是定义在另外的空间的（至于为什么欧氏三维空间的两个向量外积可以定义在三维空间里，就是因为C（3,2）=3，导致外积空间和三维空间同构，其实结果是把外积空间“嵌入”了三维空间而已。）
> 也可以这样理解“内外”：两个向量的内积只需要知道它们生成的平面的性质就可以确定了，而外积必须在这个平面外才有意义，否则就是一个数（还可以用内积表示）。
> 来自知乎：https://www.zhihu.com/question/26661955

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。

利用低维的输入空间，将其转换为高维空间，即它将不可分离的问题转化为可分离问题，这些函数称为核。

线性不可分

假如我们有以下非线性可分的数据集：



二维空间中，每个数据点都可用一个二维向量$(x_1, x_2)^T$来表示，我们可以用一个椭圆形状的超平面在该2维空间中对数据集进行分类，我们写出椭圆的一般方程：

$$w_1 x_1 + w_2 x_2^2 + w_3 x_2 + w_4 x_2^2 + w_5 x_1 x_2 + w_6 = 0$$

令：

$$\sum_{i=1}^{5} w_i z_i + w_6 = 0$$

其中：$z_1 = x_1, z_2 = x_1^2, z_3 = x_2, z_4 = x_2^2, z_5 = x_1 x_2$ ，就会发现，2维向量x被映射成另一个5维向量z后，分类超平面是一个线性超平面，数据点变得线性可分。也即是下面的变换：

$$z = \begin{pmatrix} x_1 \\ x_1^2 \\ x_2 \\ x_2^2 \\ x_2^2 \\ x_1 x_2 \end{pmatrix} = \phi(x) = \phi\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right]$$

低维到高维映射

也就是说，数据集在二维空间中线性不可分，若想实现线性可分，须把该数据集映射到一个5维空间中. 考虑SVM的的原始优化问题：

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$s.t.\, y_i(w \cdot x_i + b) \geq 1, \forall x_i$$

上式中的x_i对应数据集中的样本点，现在在二维空间中。我们要实现该数据集线性可分，需要把每个点都映射到5维空间中去。也就变成了下式：

$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$s.t.\, y_i(w \cdot \phi(x_i) + b) \geq 1, \forall x_i$$

根据上式可以推知，对于一个线性不可分的数据集，我们只要把xi替换成相应的映射后的点就可以了。所以，原来二维空间中的分类决策函数：

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i (x \cdot x_i) + b\right) \tag{B1}$$

也就变成了5维空间中的分类决策函数：

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i \left(\phi(x) \cdot \phi(x_i)\right) + b\right) \tag{B2}$$

x的映射函数已知，所以我们就能轻松根据上式得到决策超平面了。

核函数

看似问题到这里就结束了，但是，考虑到本例中只是实现二维空间中数据的线性可分，就把数据映射到了5维空间。想像如果数据本身的维度就很高，那映射后的空间维度会更高，甚至是无限维！我们该怎么求这个映射函数呢？即便知道了这个映射函数，也没法算啊，因为它是无限维的。所以，**我们应该找一个合适的二元函数，它的输入是原空间的两个向量，它的输出是映射到高维空间的两个向量的内积**！给这个合适的二元函数起个霸气的名字，就叫做**核函数**。

为什么这样定义？对照 B2 式中的两个映射函数的内积来思考一下：我们要求出 B2 式两个映射函数的内积，所以要构造一个二元函数，它的输入就是原二维空间中的 x 和 x_i 两个向量，它的输出就是映射到5维空间的两个向量的内积:

$$\left(x \cdot x_i\right) \dashrightarrow f \dashrightarrow \left(\phi(x) \cdot \phi(x_i)\right)$$

这样，我们就避免了求映射函数，只通过一个核函数就可以在低维空间完成高维空间中才能完成的事！

先考虑一个简单的例子：现在有两个二维空间中的数据点和二元函数：

$$\begin{aligned} x &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ y &= \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \\ f(x, y) &= (x \cdot y)^2 \end{aligned} \tag{B3}$$

把x,y代表 B3 式，解之得：

$$\begin{aligned} f(x, y) &= (x \cdot y)^2 \\ &= (x_1 y_1 + x_2 y_2)^2 \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 \\ &= \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2\right) \cdot \left(y_1^2, y_2^2, \sqrt{2} y_1 y_2\right) \\ &= p \cdot q \end{aligned} \tag{B4}$$

最后的函数值竟然等于两个向量 p 和 q 的内积，而且两个向量分别是二维空间数据点 x 和 y 在三维空间中的映射！想到刚才定义的核函数，我们很容易想到，f(x,y)就是一个核函数。它给出了一个二维的表达式，使得x,y代入即可求值，而不再需要先把x,y映射成3维空间中的向量p,q，再求内积。

这也正是我们定义核函数的目的，即**虽然没有显式地给出原空间中向量的映射函数，但却达到了可以在原空间中计算映射后的向量内积的目的！**

回到我们刚才讨论的问题，对于B4 式，假设我不知道这个5维的映射是啥，但我要求映射后向量的内积，所以我要构造一个核函数$K\left(x, x_i\right)$ 来代替映射后向量的内积，即可得到下面的决策分类面：

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b\right) \tag{B5}$$

不同的核函数

那么问题来了：

- 应该怎样构造这个核函数呢?

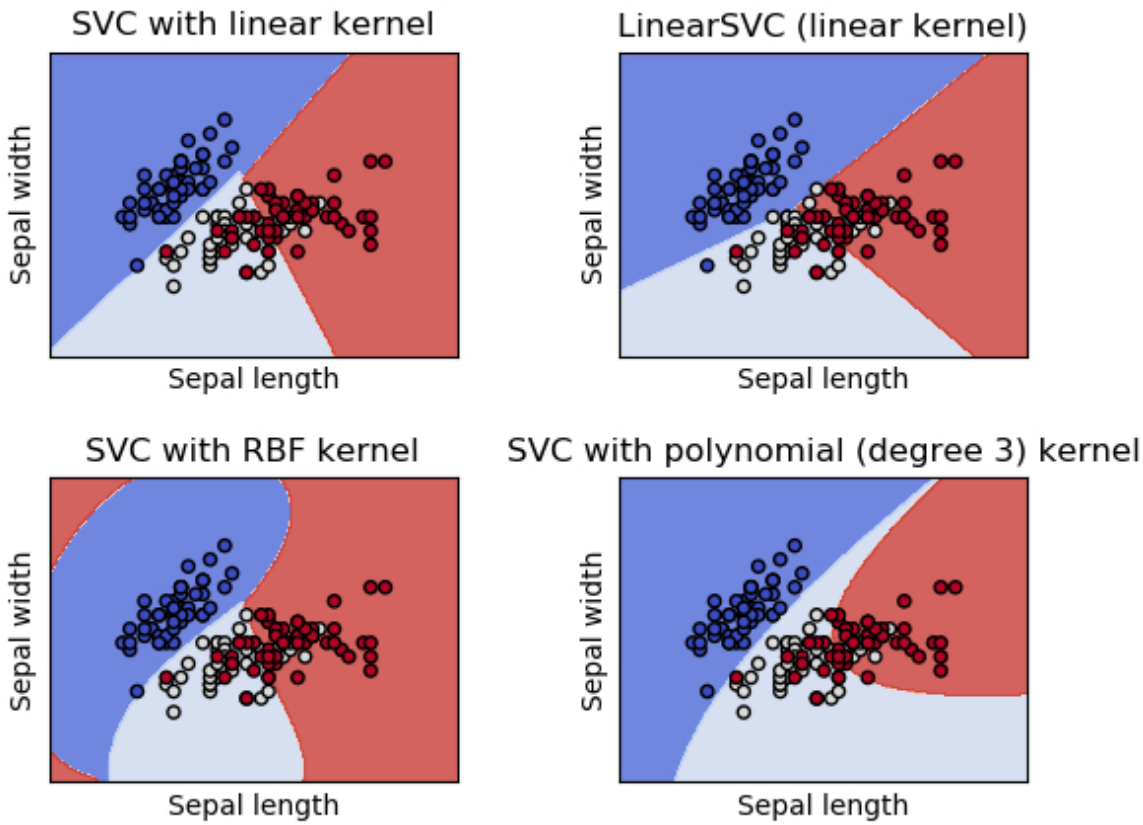
有几个经典的核函数可供选用，如：

- 线性核函数：$k(x_i, x_j) = x_i^T x_j$
- 多项式核函数：$k(x_i, x_j) = (x_i^T x_j)^d$ 或者 $(\langle x_1, x_2 \rangle + R)^d \qquad d \geq 0$
- 高斯核函数(RBF ,径向基函数)：$k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \qquad \sigma > 0$
- 拉普拉斯核函数：$k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|}{2\sigma}\right) \qquad \sigma > 0$
- Sigmoid核函数：$k(x_i, x_j) = tanh\left(\beta x_i^T x_j + \theta\right) \qquad \beta > 0, \theta < 0$

当然你也可以自己构造核函数，但也不是任意一个函数就可以当做核函数的，需要满足Mercer条件。

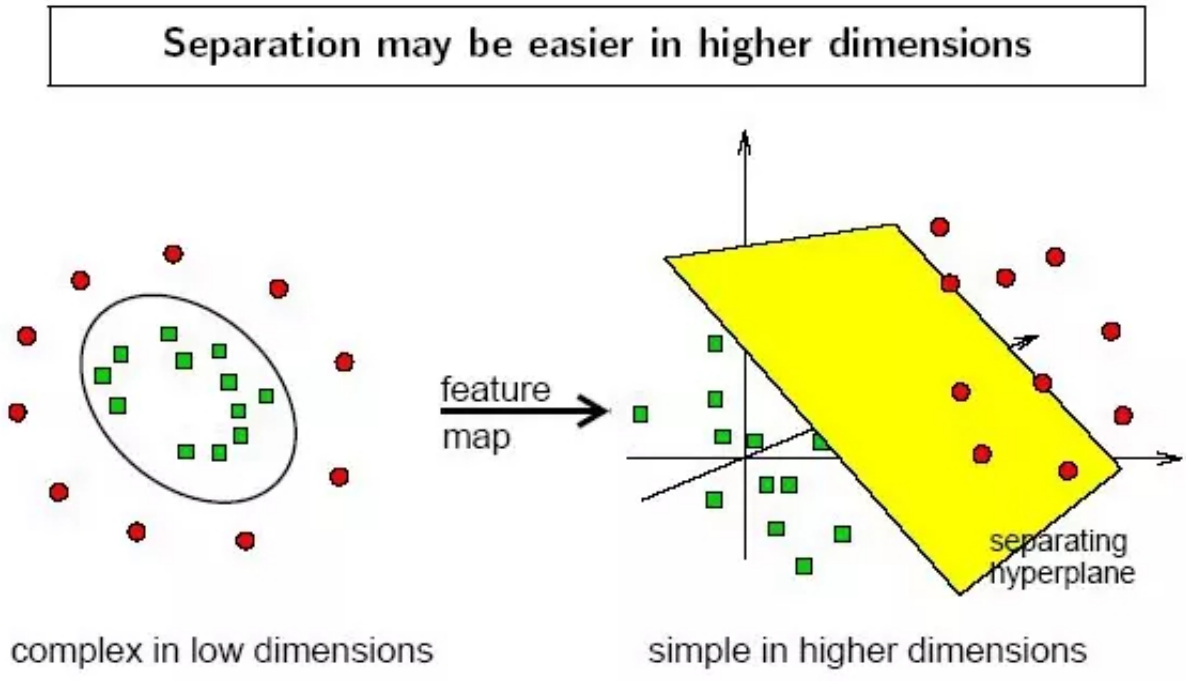
高斯核可以把原始空间映射到无穷维空间，不过，如果 σ 选得很大的话，高次特征上的权重实际上衰减得非常快，所以实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果 σ 选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调控参数 σ ，高斯核实际上具有相当高的灵活性，也是使用最广泛的核函数之一

不同的核函数在 SVM 差生的分类边界：Classification



- 你咋知道你构造的这个核函数就一定能使数据在高维空间中线性可分呢?

我没专门研究过，我也不知道自己构造的核函数一定能把数据点在高维空间中线性地分开。但是你可以选择上面提到的常用的核函数，选择合适的参数，就能使得原空间中的数据点在高维空间中变得线性可分。常用的核函数把原空间的数据映射到了高维空间中，使得数据变得"更容易"线性可分，考虑下图所示的例子：



二维空间中的点只能用非线性的超平面才能分开，但把数据映射到高维空间中，就可以用一个线性的平面给分开了。虽然更高维的画面无法脑补，但是可以参考我们在文章开始举的椭圆方程的例子。即空间的维度越高，数据越容易线性可分。

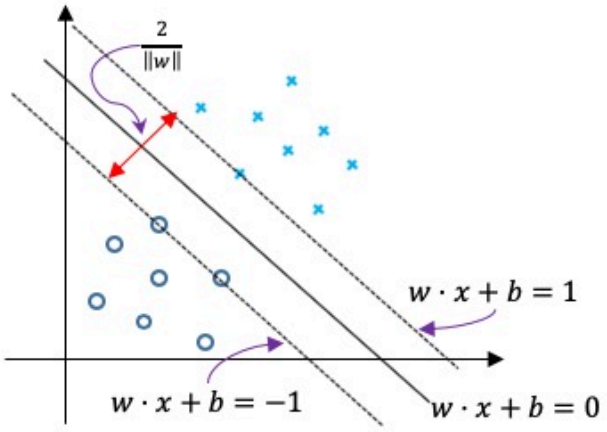
SVM 回归（SVR）

【参考】

- sklearn - SVM Regression
- 知乎 - 支持向量机svc和svr回归和分类具体的区别在于哪里呢？感觉不是很明确？
- stackexchange - How does support vector regression work intuitively?
- A tutorial on support vector regression

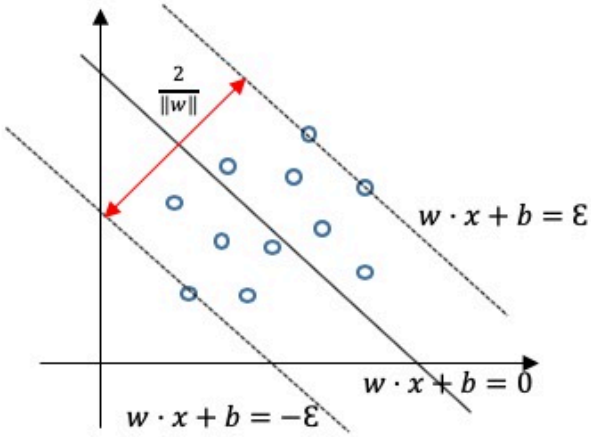
简介

直观上来讲 SVM 分类（SVC Support Vector Classification）与 SVR（Support Vector Regression）图形上的区别如下：



使得到超平面最近的样本点的距离最大

SVC

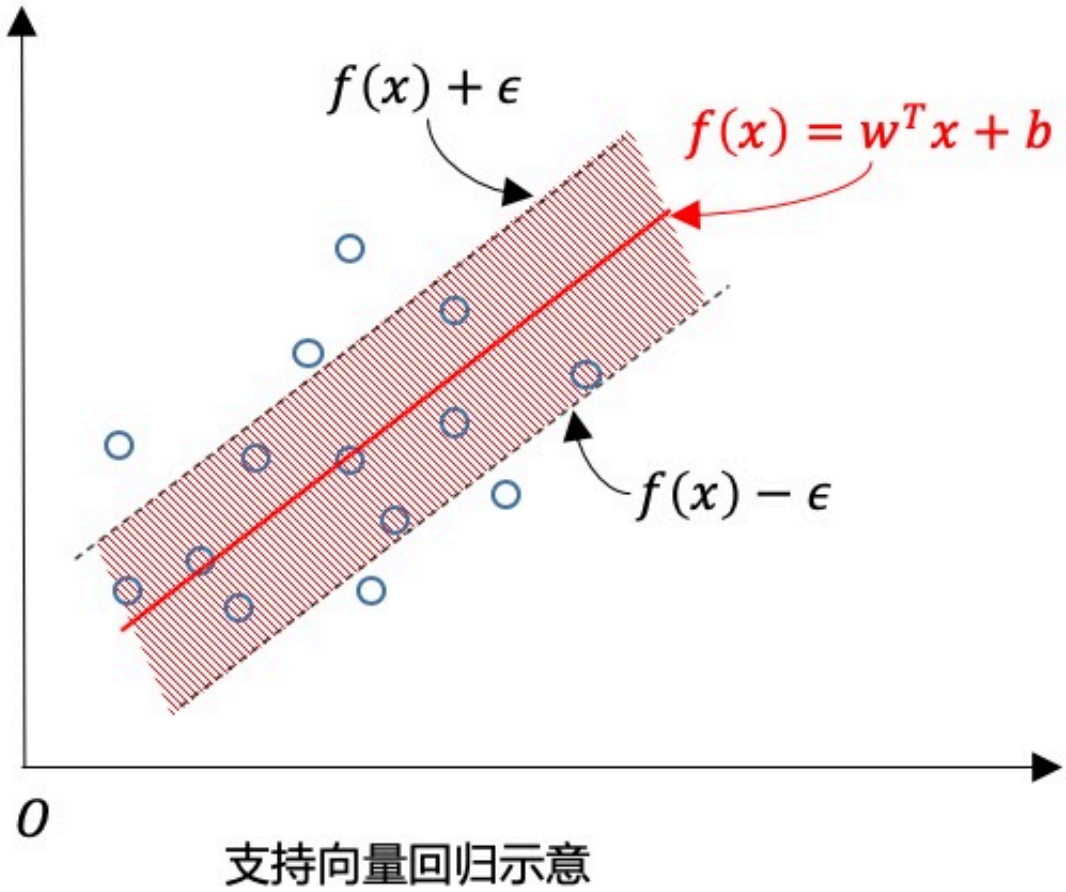


使得到超平面最远的样本点的距离最小

SVR

分类是找一个平面，使得**边界上的点**到平面的距离最远，回归是让**每个点**到回归线的距离最小。

对于样本(x, y)，传统的回归模型通常直接输出 $f(x)$ 与真实输出 y 之间的差别来计算损失，当且仅当 $f(x)$ 与 y 完全相同时，损失才是零。与此不同 SVR 假设我们能容忍 $f(x)$ 与 y 之间最多有 ϵ 的偏差，即仅当 $f(x)$ 与 y 之间的差别绝对值大于 ϵ 时才计算损失。这相当于以 $f(x)$ 为中心 构建一个宽度为 2ϵ 的间隔带，若样本落入此间隔带，则认为是预测正确的，如下图：

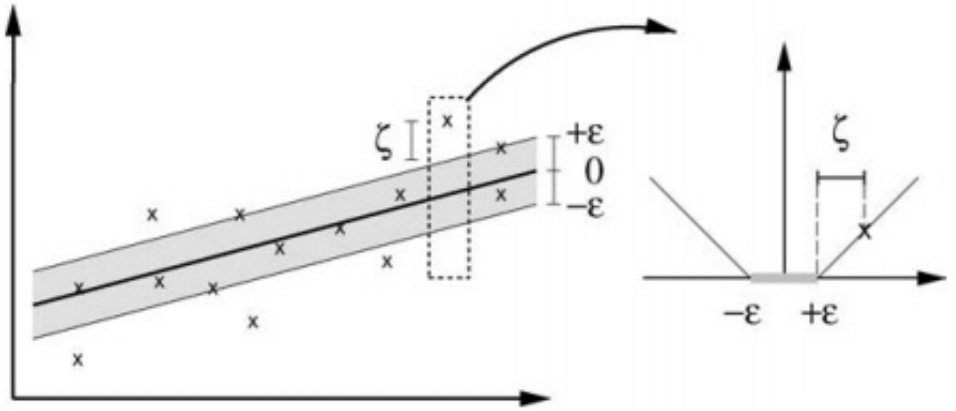


支持向量回归示意

数学形式

【参考】

- 简书 - SVM系列十三讲--支持向量回归机SVR
- 个站 - SVR，Support Vector Regression，支持向量回归

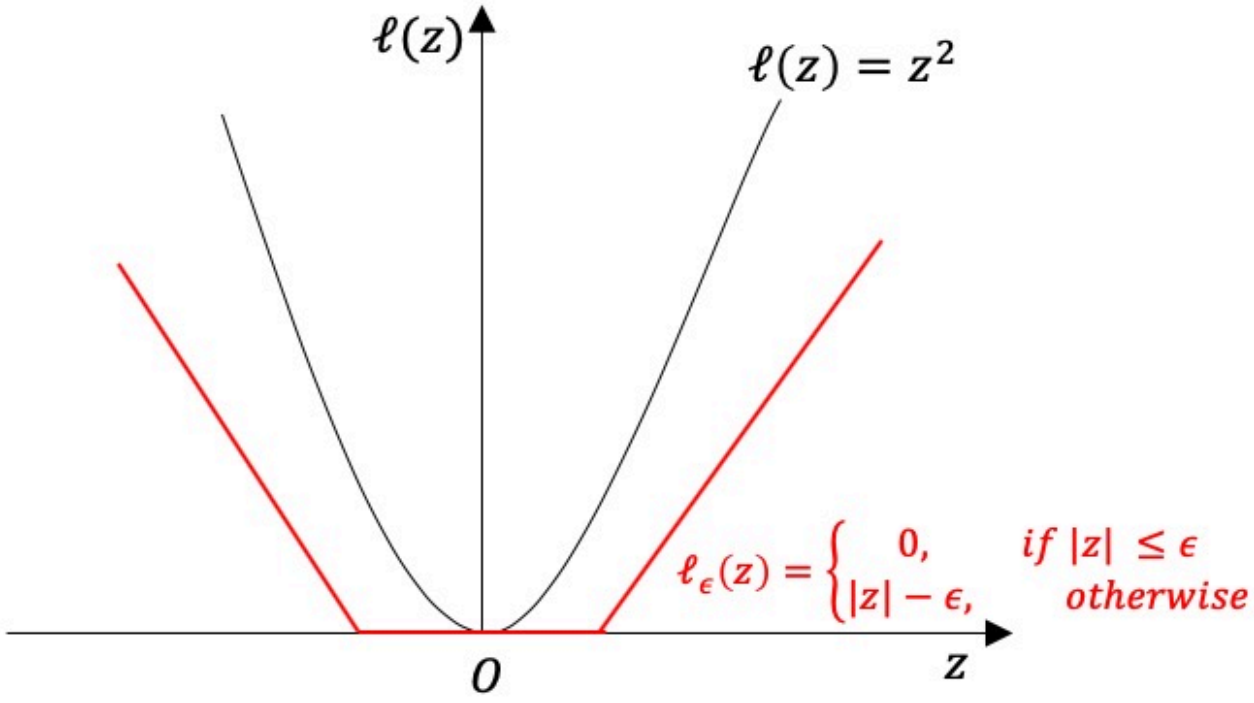


于是 SVR 问题可以形式化为：

$$\min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m} \ell_\epsilon(f(\boldsymbol{x}_i) - y_i) \tag{C1}$$

其中 C 正则化常数，ℓ_ϵ 是下图的 `ε–不敏感损失`（ε-insensitive loss）函数：

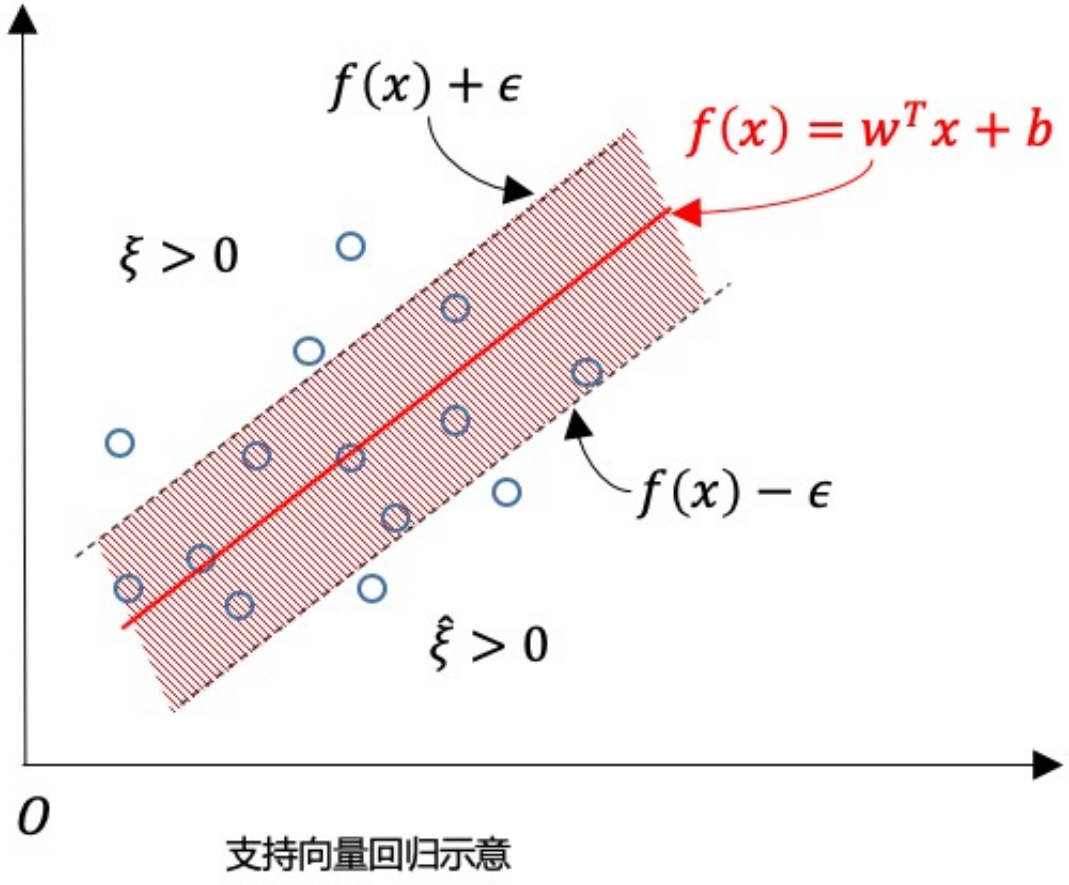
$$\ell_\epsilon(z) = \begin{cases} 0, & if\ |z| \le \epsilon \\ |z| - \epsilon, & otherwise \end{cases} \tag{C2}$$



ϵ-不敏感损失函数

引入松弛变量ξ_i和$\hat{\xi}_i$(间隔两侧的松弛程度有可能不同)，可以将式（C1）重写为：

$$\begin{aligned} \min_{\omega,b} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i) \\ s.t.\ f(\boldsymbol{x}_i) - y_i \le \epsilon + \xi_i \\ y_i - f(\boldsymbol{x}_i) \le \epsilon + \hat{\xi}_i \\ \xi_i > 0\ \hat{\xi}_i > 0\ i = 1, 2, 3...m \end{aligned} \tag{C3}$$



支持向量回归示意

拉格朗日对偶形式

通过引入$\mu_i \geq 0, \hat{\mu}_i \geq 0, \alpha_i \geq 0, \hat{\alpha}_i \geq 0$，有拉格朗日乘子可以得到式(C3) 的拉格朗日函数：

$$
\begin{aligned}
&L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}, \boldsymbol{\mu}, \hat{\boldsymbol{\mu}}) \\
&= \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{m}(\xi_i + \hat{\xi}_i) - \sum_{i=1}^{m}\mu_i\xi_i - \sum_{i=1}^{m}\hat{\mu}_i\hat{\xi}_i \\
&+ \sum_{i=1}^{m}\alpha_i\left(f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i\right) \\
&+ \sum_{i=1}^{m}\hat{\alpha}_i\left(y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i\right)
\end{aligned}
\tag{C4}
$$

将 $f(\boldsymbol{x}_i) = \boldsymbol{w}^T\boldsymbol{x} + b$带入上式，并令$L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}, \boldsymbol{\xi}, \hat{\boldsymbol{\xi}}, \boldsymbol{\mu}, \hat{\boldsymbol{\mu}})$ 对$\omega, b, \xi_i, \hat{\xi}_i$ 的偏导为零，得到：

$$
\begin{aligned}
\boldsymbol{\omega} &= \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)\boldsymbol{x}_i \\
0 &= \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) \\
C &= \alpha_i + \mu_i \\
C &= \hat{\alpha}_i + \hat{\mu}_i
\end{aligned} \tag{C5}
$$

将式（C5）带入式（C4）可以得到 SVR 的对偶问题：

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} \ & \sum_{i=1}^{m} y_i(\hat{\alpha}_i - \alpha_i) - \epsilon(\hat{\alpha}_i + \alpha_i) \\
& - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\boldsymbol{x}_i^T\boldsymbol{x}_j \\
s.t. \ & \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) = 0 \\
& 0 \le \alpha_i,\ \hat{\alpha}_i \le C
\end{aligned} \tag{C6}
$$

KKT 与最终决策函数

上述过程满足的 KKT 条件为：

$$
\begin{cases}
\alpha_i\left(f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i\right) = 0 \\
\hat{\alpha}_i\left(y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i\right) = 0 \\
\alpha_i\hat{\alpha}_i = 0,\ \xi_i\hat{\xi}_i = 0 \\
(C - \alpha_i)\xi_i = 0,\ (C - \hat{\alpha}_i)\hat{\xi}_i = 0
\end{cases} \tag{C7}
$$

可以看出，当且仅当 $f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i = 0$时，$\alpha_i$ 能取非零值，当且仅当，$y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i = 0$ 时 $\hat{\alpha}_i$ 能取非零值。换言之，仅当样本$(\boldsymbol{x}_i, y_i)$ 不落入 ε-间隔带中，相应的 α_i 和 $\hat{\alpha}_i$ 才能取非零值。此外，约束$f(\boldsymbol{x}_i) - y_i - \epsilon - \xi_i = 0$ 与 $y_i - f(\boldsymbol{x}_i) - \epsilon - \hat{\xi}_i = 0$ 不能同时成立，因此α_i 和 $\hat{\alpha}_i$中至少有一个为零。

将式（C5）第一项带入决策函数，可得最终的决策函数为：

$$
f(\boldsymbol{x}) = \sum_{i=1}^{n}(\hat{\alpha}_i - \alpha_i)\boldsymbol{x}_i^T\boldsymbol{x}_j + b \tag{C8}
$$

能使上式中$\hat{\alpha}_i - \alpha_i \neq 0$ 成立的样本即为 SVR 的支持向量，他们必然落在`ε-间隔带`之外。显然 SVR 的支持向量仅是训练样本的一部分，即其解仍然具有稀疏性。

由 KKT 条件可以看出，对于每个样本 $(\boldsymbol{x}_i, y_i)$ 都有 $(C-\alpha_i)\xi_i=0$ 且 $\alpha_i\left(f(\boldsymbol{x}_i)-y_i-\epsilon-\xi_i\right)=0$，于是在得到 α_i 之后，若$0<\alpha_i<C$ 则必有 $\xi_i=0$，继而有：

$$b=y_i+\epsilon-\sum_{i=1}^{n}(\hat{\alpha}_i-\alpha_i)\boldsymbol{x}_i^T\boldsymbol{x}_j \tag{C9}$$

因此，若求解式（C6）得到 α_i 后，理论上说可以任意选取满足 $0<\alpha_i<C$ 的样本，通过式（C9）求得 b。在实践中采用一种更鲁棒的办法：选择多个（或所有）满足条件 $0<\alpha_i<C$ 的样本求解 b 后去平均值。

核函数的形式最终的决策函数为：

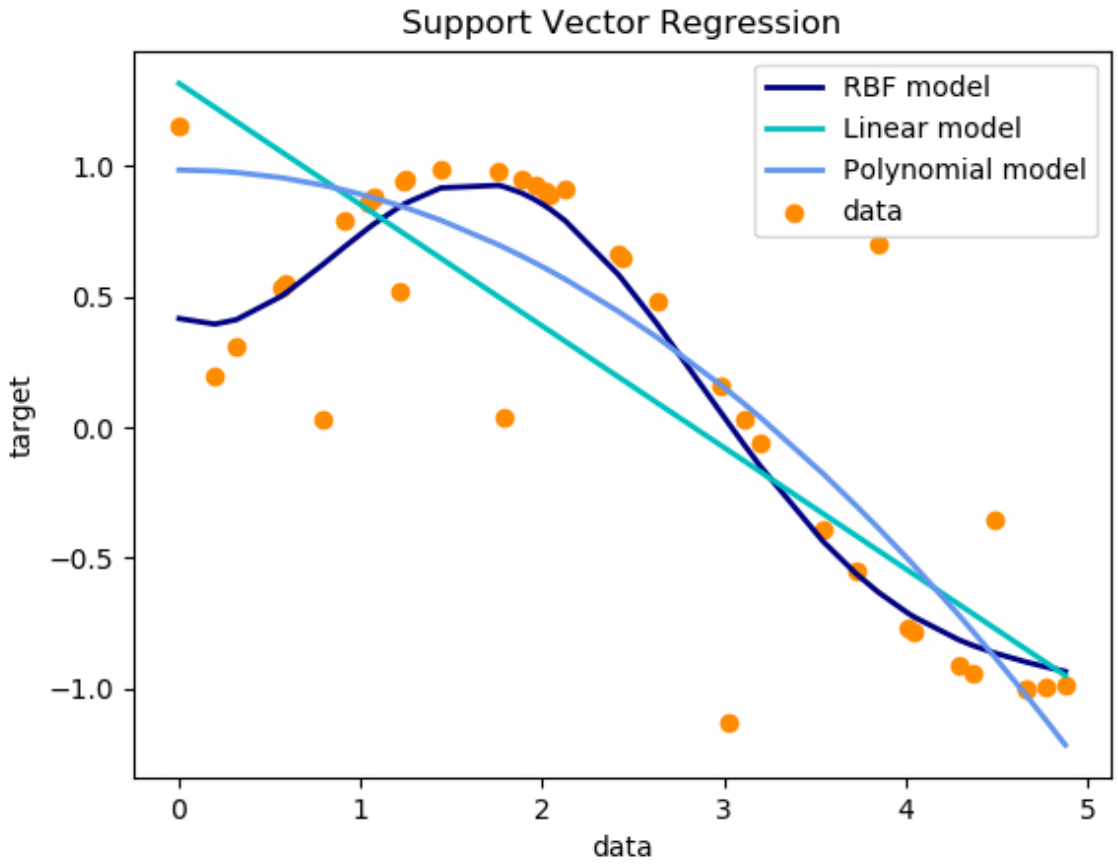
$$f(\boldsymbol{x})=\sum_{i=1}^{n}(\hat{\alpha}_i-\alpha_i)\kappa(\boldsymbol{x},\boldsymbol{x}_i)+b \tag{C9}$$

其中 $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$为核函数。

不同核的效果

【参考】

- sklearn - Support Vector Regression (SVR) using linear and non-linear kernels



基于 Sklearn 的实践建议

【参考】

- sklearn - Tips on Practical Use
- 避免数据拷贝
- 核缓存的大小：对于 `SCV`、`SVR`、`NuSVC` 和 `NuSVR`，核函数缓存的大小对于大型问题的运行时间有着非常大的影响。如果有足够多的内存，建议把`cache_size`的大小设置的尽可能的大。
- 设置 C：1 是一个合理的默认选择，如果有较多噪点数据，你应该较少 `C` 的大小。
- SVM 算法不是尺度不变，因此强烈建议缩放你的数据。如将输入向量 X 的每个属性缩放到[0,1] 或者 [-1,1]，或者标准化为均值为 0 方差为 1 。另外，在测试向量时也应该使用相同的缩放，已获得有意义的结果。
- 对于`SVC`，如果分类的数据不平衡（如有很多的正例很少的负例），可以设置`class_weight='balanced'`，或者尝试不同的惩罚参数 `C`
- 底层实现的随机性：`SVC`和`NuSVC`的底层实现使用了随机数生成器，在概率估计时混洗数据（当 `probability` 设置为 `True`），随机性可以通过 `random_state` 参数控制。如果 `probability` 设置为`False`，这些估计不是随机的，`random_state` 对结果不在有影响。
- 使用 `L1` 惩罚来产生稀疏解