

# SVM

赵耀

# 简介

- ▶ 一种监督式机器学习的模型
- ▶ 在文本分类、生物信息、语言识别、故障识别和预测等诸多领域有了成功的应用
- ▶ 优点：一定程度上克服了“维数灾难”和“过学习”等传统困难
- ▶ 缺点：数据量很大的时候速度比较慢

# 输入输出

- ▶ 输入：特征向量  $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
- ▶ 输出：分类标签（通常是1、-1）

# 问题表达：举例

病人编号	血压 ( 低压 ) $[x]_1$	胆固醇水平 $[x]_2$	有否心脏病 $y$
1	$[x]_1 = 73$	$[x]_2 = 150$	$y = -1$
2	$[x]_1 = 85$	$[x]_2 = 165$	$y = -1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	$[x]_1 = 110$	$[x]_2 = 190$	$y = 1$

心脏病人辅助诊断：

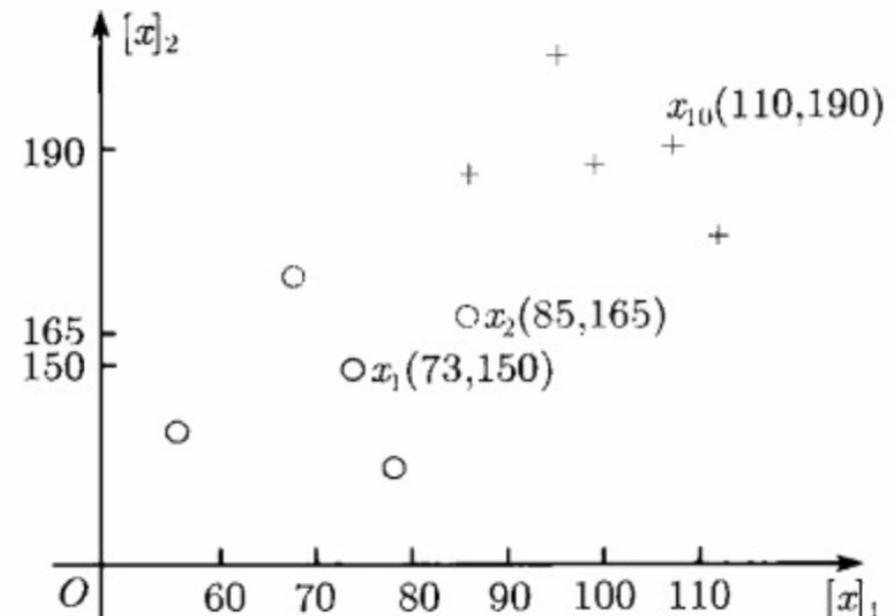
从上述例子可知，上述例子为输入为一个二维向量，十个样本点数据分别为：

$$x_1 = [73, 150]^T \quad y_1 = -1$$

$$x_2 = [85, 165]^T \quad y_2 = -1$$

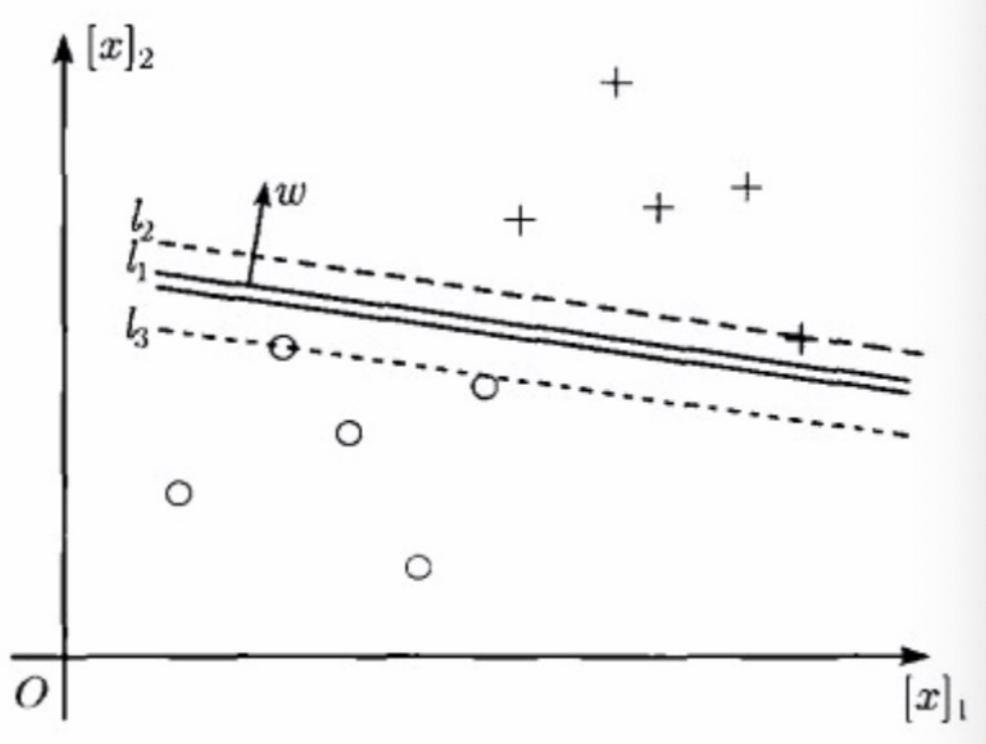
...

$$x_{10} = [110, 190]^T \quad y_3 = 1$$

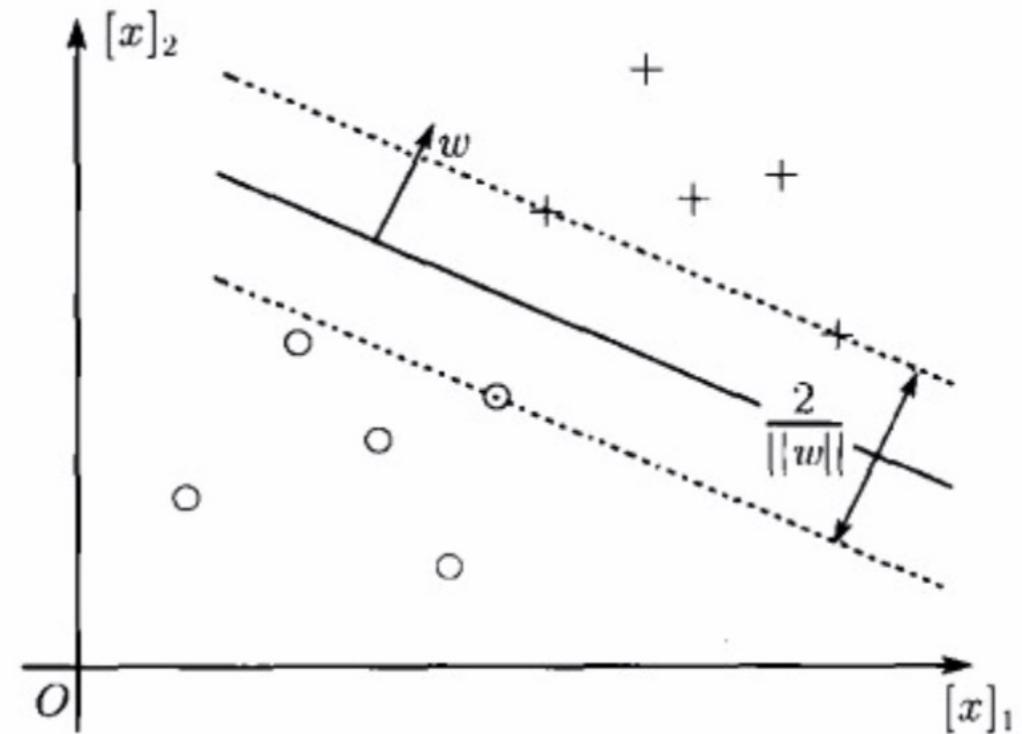


问题：来了一个新病人，测量完血压和胆固醇之后，预测一下其是否有心脏病。即  $y$  是 1 还是 -1

# 问题表达：可行解与最优解



多个可行解



最大间隔解

# 问题表达：凸优化问题

$$\begin{aligned} \max_{w,b} \quad & \frac{2}{\|w\|}, \\ \text{s.t.} \quad & \text{对所有使 } y_i = 1 \text{ 的下标 } i, \text{ 有 } (w \cdot x_i) + b \geq 1, \\ & \text{对所有使 } y_i = -1 \text{ 的下标 } i, \text{ 有 } (w \cdot x_i) + b \leq -1, \end{aligned}$$

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2, \\ \text{s. t.} \quad & y_i((w \cdot x_i) + b) \geq 1, i = 1, \dots, l, \end{aligned}$$

凸二次规划问题

对于线性可分的问题，有唯一的全局可行解

# 补充知识:二次规划

## B.2 二次规划

二次规划(Quadratic Programming, 简称 QP)是一类典型的优化问题, 包括凸二次优化和非凸二次优化. 在此类问题中, 目标函数是变量的二次函数, 而约束条件是变量的线性不等式.

假定变量个数为  $d$ , 约束条件的个数为  $m$ , 则标准的二次规划问题形如

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A} \mathbf{x} \leq \mathbf{b}, \end{aligned} \tag{B.12}$$

其中  $\mathbf{x}$  为  $d$  维向量,  $\mathbf{Q} \in \mathbb{R}^{d \times d}$  为实对称矩阵,  $\mathbf{A} \in \mathbb{R}^{m \times d}$  为实矩阵,  $\mathbf{b} \in \mathbb{R}^m$  和  $\mathbf{c} \in \mathbb{R}^d$  为实向量,  $\mathbf{A} \mathbf{x} \leq \mathbf{b}$  的每一行对应一个约束.

若  $\mathbf{Q}$  为半正定矩阵, 则式(B.12)目标函数是凸函数, 相应的二次规划是凸二次优化问题; 此时若约束条件  $\mathbf{A} \mathbf{x} \leq \mathbf{b}$  定义的可行域不为空, 且目标函数在此可行域有下界, 则该问题将有全局最小值. 若  $\mathbf{Q}$  为正定矩阵, 则该问题有唯一的全局最小值. 若  $\mathbf{Q}$  为非正定矩阵, 则式(B.12)是有多个平稳点和局部极小点的 NP 难问题.

常用的二次规划解法有椭球法(ellipsoid method)、内点法(interior point)、增广拉格朗日法(augmented Lagrangian)、梯度投影法(gradient projection) 等. 若  $\mathbf{Q}$  为正定矩阵, 则相应的二次规划问题可由椭球法在多项式时间内求解.

周志华《机器学习》 p.406

# 训练过程：步骤

获取训练数据

$$x_1 = [73, 150]^T \quad y_1 = -1$$

$$x_2 = [85, 165]^T \quad y_2 = -1$$

...

$$x_{10} = [110, 190]^T \quad y_{10} = 1$$

结合如下表达式构造问题：

$$\begin{array}{ll}\min_{w,b} & \frac{1}{2} \|w\|^2, \\ \text{s. t.} & y_i((w \cdot x_i) + b) \geq 1, \quad i = 1, \dots, 10.\end{array}$$

求解  $w_1$ 、 $w_2$  和  $b$ ，使得  $(w_1^2 + w_2^2)/2$  最小，同时满足：

$$-1 * (w_1 * 73 + w_2 * 150 + b) \geq 1$$

$$-1 * (w_1 * 85 + w_2 * 165 + b) \geq 1$$

...

$$1 * (w_1 * 110 + w_2 * 190 + b) \geq 1$$

有 10 个样本，那么就是 10 个约束。

求解  $w_1$ 、 $w_2$  和  $b$  的解

用  $w_1$ 、 $w_2$  和  $b$  的解构造分划超平面  $w_1x_1 + w_2x_2 + b = 0$ ，得出决策函数

Notice: 从二维推广到多维

# 训练过程：先说简单问题的求解

- ▶ 假设训练集有3个样本点， 分别为 $x_1 = (1,1)$   $y_1 = -1$ ,  $x_2 = (2,2)$   $y_2 = 1$ ,  $x_3 = (4,3)$ ,  $y_3 = 1$
- ▶ 求最大间隔分离超平面
- ▶ 预测测试点 $(0,0)$ 会被标记为1还是-1
- ▶ 答案:  $w_1 = w_2 = 1$   $b = -3$

# 训练过程：初始化W 以及 b

- ▶ 仔细观察上面的计算过程，可以发现，W和B取值的不同，决定了超平面的分隔程度好坏。
- ▶ 训练模型的目的，是找到一组合适的W和B的取值，使得输入值经过一系列变换之后能够达到最优分隔效果。
- ▶ SVM的学习过程就是学习怎么去控制权重矩阵
- ▶ 初始化可以将生成随机数作为w的初始值，b可以通过平移x消去。

```
def __init__(self, x, y, epochs=200, learning_rate=0.01):  
    self.x = np.c_[np.ones((x.shape[0])), x]  
    self.y = y  
    self.epochs = epochs  
    self.learning_rate = learning_rate  
    self.w = np.random.uniform(size=np.shape(self.x)[1], )
```

# 训练过程：损失函数和梯度下降

- ▶ 既然希望SVM的输出尽可能的接近真正想要预测的值，那么就可以通过比较当前网络的预测值与目标值，根据两者的差异情况来更新每一层的权重矩阵。
- ▶ 如何比较差异情况？ 损失函数(loss function)
- ▶ 如何更新权重矩阵？ 梯度下降(Gradient descend)

# 训练过程：损失函数(loss function)

- ▶ 损失函数，又称为误差函数（error function），也有叫cost function。是用于衡量预测值和目标值差异的函数。
- ▶ loss function的输出值越高表示差异性越大，那么SVM的训练目标就是尽可能的缩小差异
- ▶ 例子： $\mathcal{L}(\mathbf{w}, b) = \sum_{i=1}^n \max(0, 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b))$ ,

```
def get_loss(self, x, y):
    loss = max(0, 1 - y * np.dot(x, self.w))
    return loss
```

用当前的 $w$ 和 $b$ 值算出来的 $1 - y * np.dot(x, self.w)$ 的值，如果 $<0$ ，表示符合标记信息，损失即为0；如果大于0，表示预测不准，损失即为该值。

# 梯度下降(Gradient descend)

- ▶ 梯度下降：通过使loss值向当前点对应梯度的反方向不断移动，来降低loss。一次移动多少是由学习速率（learning rate）来控制的。

$$\mathbf{w} = \mathbf{w} - \alpha \frac{\partial \mathcal{L}(\mathbf{w}, b)}{\partial \mathbf{w}}$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, b)}{\partial \mathbf{w}} = \begin{cases} -y_i * x_i, & \text{if } 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

```
def cal_sgd(self, x, y, w):
    if y * np.dot(x, w) < 1:
        w = w - self.learning_rate * (-y * x)
    else:
        w = w
    return w
```

学习速率

```
def train(self):
    for epoch in range(self.epochs):
        randomize = np.arange(len(self.x))
        np.random.shuffle(randomize)
        x = self.x[randomize]
        y = self.y[randomize]
        loss = 0
        for xi, yi in zip(x, y):
            loss += self.get_loss(xi, yi)
            self.w = self.cal_sgd(xi, yi, self.w)
    print('epoch: {} loss: {}'.format(epoch, loss))
```

# 随机梯度下降

- ▶ 上页公式中，梯度下降时，每次计算损失函数都是在整个数据集上计算的，并进一步去算 $w$ ，在样本很大的情况下，这样做极为耗时，一般实验中会采用随机梯度下降的方法，从整个数据集中随机抽取一部分数据，用这一部分数据去计算损失函数和更新梯度。代码也是用随机梯度下降实现的。

# 训练过程：预测

```
def predict(self, x):
    x_test = np.c_[np.ones((x.shape[0])), x]
    return np.sign(np.dot(x_test, self.w))
```

# 问题

- ▶ 上述损失函数的问题：

$$\mathcal{L}(\mathbf{w}, b) = \sum_{i=1}^n \max(0, 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)),$$

- ▶ 其损失函数并不能达到最优分隔效果
- ▶ 其目标是找到一个可行解，而不是最优解

# Pegasos 算法

- ▶ 见 [pegasos.pdf](#)

# 对偶算法

- ▶ 对偶算法是指为了求解主问题(转为拉格朗日函数)，应用拉格朗日对偶性，通过求解对偶问题得到原始问题的最优解

为什么要这么做？

- 1、当原问题的维度比较高的时候，对偶问题更容易求解
- 2、自然引入核函数，方便推广到非线性问题

# 原问题转化为对偶问题

$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)^\top$  为拉格朗日乘子向量

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, N$$

$$\begin{aligned} & \min_{w,b} \quad \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y_i(w \cdot x_i + b) - 1 \geq 0, \quad i = 1, 2, \dots, N \end{aligned}$$

$$\begin{aligned} L(w, b, \alpha) &= \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i y_i (w \cdot x_i + b) + \sum_{i=1}^N \alpha_i \\ & \max_{\alpha} \min_{w,b} L(w, b, \alpha) \end{aligned}$$

为了得到对偶问题的解，需要先求  $L(w, b, \alpha)$  对  $w, b$  的极小，再求对  $\alpha$  的极大.

# 原问题转化为对偶问题

先求  $L(w, b, \alpha)$  对  $w, b$  的极小

将拉格朗日函数  $L(w, b, \alpha)$  分别对  $w, b$  求偏导数并令其等于 0.

$$\nabla_w L(w, b, \alpha) = w - \sum_{i=1}^N \alpha_i y_i x_i = 0$$

$$\nabla_b L(w, b, \alpha) = \sum_{i=1}^N \alpha_i y_i = 0$$

得

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

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

代入

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

$$\max_{\alpha} \min_{w, b} L(w, b, \alpha)$$

$$\begin{aligned} L(w, b, \alpha) &= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^N \alpha_i y_i \left( \left( \sum_{j=1}^N \alpha_j y_j x_j \right) \cdot x_i + b \right) + \sum_{i=1}^N \alpha_i \\ &= -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^N \alpha_i \end{aligned}$$

再求对  $\alpha$  的极大

$$\max_{\alpha} -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^N \alpha_i$$

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

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, N$$

# 补充阅读：拉格朗日乘子法

- ▶ 参考周志华《机器学习》p403~408
- ▶  $d^* \leq p^*$ , 因为 $p$ 是先求最大的一块区域然后在这块区域求最小,  $d$ 是先求最小的一块区域然后在这块区域求最大, 这个式子表示最大里面的最小, 总会比最小里面的最大要大(或等于)。
- ▶ 如果 $d^* = p^*$ , 称为弱对偶, 对于所有优化问题都成立, 这个时候我们可以得到原始问题的一个下界。
- ▶ 如果 $d^* = p^*$ , 称为强对偶, 满足某些条件才成立, 这时可以用强对偶问题替代原始问题。那么满足什么样的条件可以得到强对偶呢?
- 如果原问题是一个凸优化, 并且 $g(x)$ 是一个凸函数不等式约束 $g_i(x) \leq 0$ 是严格可行的, 即存在 $x$ 对所有 $i$ 有 $g_i(x) < 0$ , 那么强对偶成立。这里需要注意的是, 这里的条件只是强对偶成立的一种情况, 对于非凸的问题也有可能是强对偶。
- 强对偶成立时, 将拉格朗日函数分别对原变量 $x$ 和对偶变量 $\alpha$ 和 $\beta$ 分别求导, 令导数等于零, 并同时满足KKT条件, 即可求解对偶问题的解, 也就求得了原问题的解。

## 补充阅读: KKT

- ▶ <http://www.csc.kth.se/utbildning/kth/kurser/DD3364/Lectures/KKT.pdf>

# KKT条件

$$\alpha_i^*(y_i(w^* \cdot x_i + b^*) - 1) = 0, \quad i = 1, 2, \dots, N$$

$$y_i(w^* \cdot x_i + b^*) - 1 \geq 0, \quad i = 1, 2, \dots, N$$

$$\alpha_i^* \geq 0, \quad i = 1, 2, \dots, N$$

结合：

$$\nabla_w L(w^*, b^*, \alpha^*) = w^* - \sum_{i=1}^N \alpha_i^* y_i x_i = 0$$

$$\nabla_b L(w^*, b^*, \alpha^*) = -\sum_{i=1}^N \alpha_i^* y_i = 0$$

$\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)^T$  为对偶问题的解

则原始问题的解可按照如下公式：

$$w^* = \sum_{i=1}^N \alpha_i^* y_i x_i$$

$$b^* = y_j - \sum_{i=1}^N \alpha_i^* y_i (x_i \cdot x_j)$$

## 对偶问题求解步骤(1)(2)

(1) 给定训练集  $T = \{(x_1, y_1), \dots, (x_l, y_l)\} \in (R^n \times \mathcal{Y})^l$ , 其中  $x_i \in R^n, y_i \in \mathcal{Y} = \{1, -1\}, i = 1, \dots, l$ ;

(2) 构造并求解凸二次规划问题

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j (x_i \cdot x_j) \alpha_i \alpha_j - \sum_{j=1}^l \alpha_j , \\ \text{s. t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\ & \alpha_i \geq 0, \quad i = 1, \dots, l , \end{aligned}$$

得解  $\alpha^* = (\alpha_1^*, \dots, \alpha_l^*)^T$ ;

## 对偶问题求解步骤(3)(4)

(3) 计算  $w^* = \sum_{i=1}^l \alpha_i^* y_i x_i$ . 选取  $\alpha^*$  的一个正分量  $\alpha_j^*$ , 据此计算

$$b^* = y_j - \sum_{i=1}^l \alpha_i^* y_i (x_i \cdot x_j);$$

(4) 构造分划超平面  $(w^* \cdot x) + b^* = 0$ , 由此求得决策函数

$$f(x) = \text{sgn}(g(x)),$$

其中

$$g(x) = (w^* \cdot x) + b^* = \sum_{i=1}^l y_i \alpha_i^* (x_i \cdot x) + b^*.$$

# 对偶问题计算示例

- ▶ 假设训练集有3个样本点， 分别为 $x_1 = (1,1)$   $y_1 = -1$ ,  $x_2 = (3,3)$   $y_2 = 1$ ,  $x_3 = (4,3)$ ,  $y_3 = 1$
- ▶ 写出对偶问题的形式，通过求对偶问题求最大间隔分离超平面

# 线性支持向量机问题的表达

- ▶ 实际的数据很难满足完全线性可分的情况
- ▶ 引入松弛变量：允许有不满足约束条件的训练点存在

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i , \\ \text{s. t.} \quad & y_i((w \cdot x_i) + b) \geq 1 - \xi_i , \quad i = 1, \dots, l , \\ & \xi_i \geq 0 , \quad i = 1, \dots, l , \end{aligned}$$

# 原问题 -> 对偶问题

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l \alpha_i (y_i((w \cdot x_i) + b) - 1 + \xi_i) - \sum_{i=1}^l \beta_i \xi_i ,$$

其中  $\alpha = (\alpha_1, \dots, \alpha_l)^T$  和  $\beta = (\beta_1, \dots, \beta_l)^T$  均为 Lagrange 乘子向量.

原问题

$$\begin{aligned} \min_{w, b, \xi} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i , \\ \text{s. t.} \quad & y_i((w \cdot x_i) + b) \geq 1 - \xi_i , \quad i = 1, \dots, l , \\ & \xi_i \geq 0 , \quad i = 1, \dots, l , \end{aligned}$$

对偶问题

$$\begin{aligned} \max_{\alpha, \beta} \quad & -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) + \sum_{j=1}^l \alpha_j , \\ \text{s. t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\ & C - \alpha_i - \beta_i = 0 , \quad i = 1, \dots, l , \\ & \alpha_i \geq 0 , \quad i = 1, \dots, l , \\ & \beta_i \geq 0 , \quad i = 1, \dots, l \end{aligned}$$

# 对偶问题：最大化写成最小化

$$\begin{aligned}
 \max_{\alpha, \beta} \quad & -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) + \sum_{j=1}^l \alpha_j , \\
 \text{s. t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\
 & C - \alpha_i - \beta_i = 0, \quad i = 1, \dots, l , \\
 & \alpha_i \geq 0, \quad i = 1, \dots, l , \\
 & \beta_i \geq 0, \quad i = 1, \dots, l
 \end{aligned}$$

$$\begin{aligned}
 \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j (x_i \cdot x_j) \alpha_i \alpha_j - \sum_{j=1}^l \alpha_j , \\
 \text{s. t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\
 & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, l .
 \end{aligned}$$

$\alpha^* = (\alpha_1^*, \dots, \alpha_l^*)^T$  为任一解，则：

$$w^* = \sum_{i=1}^l \alpha_i^* y_i x_i ,$$

$$b^* = y_j - \sum_{i=1}^l \alpha_i^* y_i (x_i \cdot x_j) .$$

# KKT条件

$$\alpha_i^*(y_i(w^* \cdot x_i + b^*) - 1 + \xi_i^*) = 0$$

$$\beta_i^* \xi_i^* = 0$$

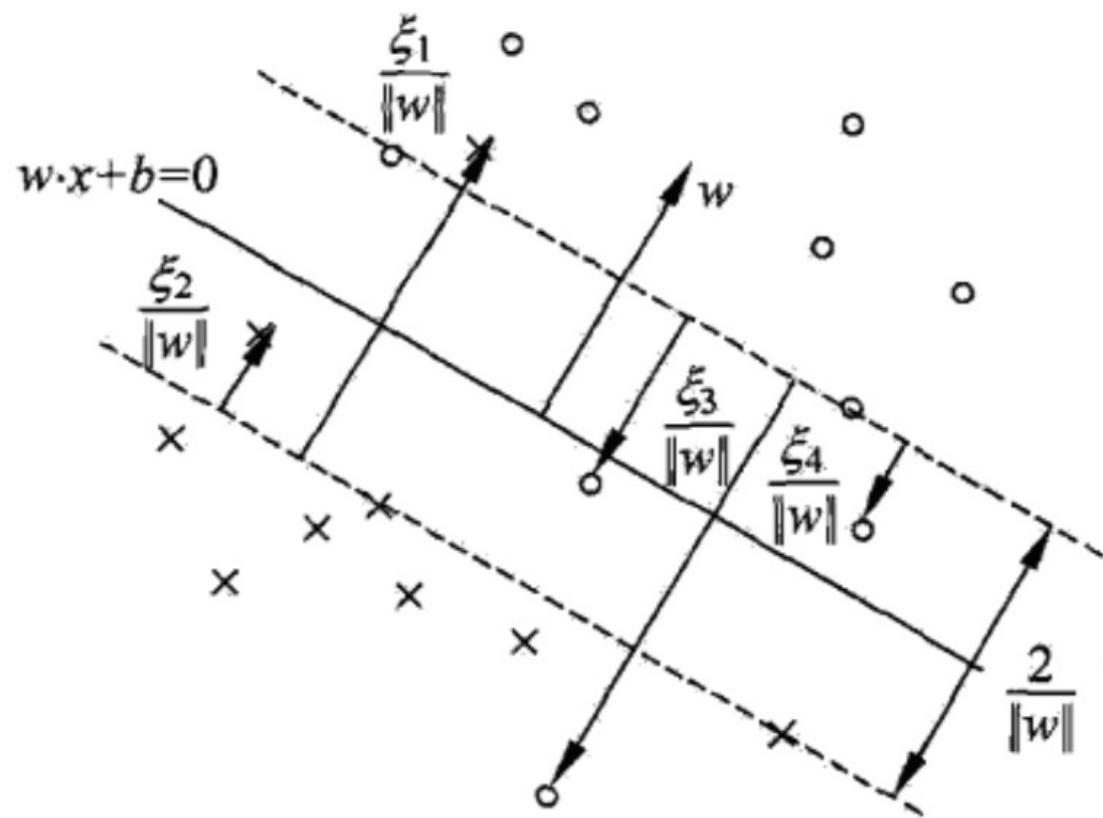
$$y_i(w^* \cdot x_i + b^*) - 1 + \xi_i^* \geq 0$$

$$\xi_i^* \geq 0$$

$$\alpha_i^* \geq 0$$

$$\beta_i^* \geq 0 \quad i=1,2,\dots,I$$

# 软间隔的支持向量



软间隔的支持向量  $x_i$  或者在间隔边界上，或者在间隔边界与分离超平面之间，或者在分离超平面误分一侧。若  $\alpha_i^* < C$ ，则  $\xi_i = 0$ ，支持向量  $x_i$  恰好落在间隔边界上；若  $\alpha_i^* = C$ ， $0 < \xi_i < 1$ ，则分类正确， $x_i$  在间隔边界与分离超平面之间；若  $\alpha_i^* = C$ ， $\xi_i = 1$ ，则  $x_i$  在分离超平面上；若  $\alpha_i^* = C$ ， $\xi_i > 1$ ，则  $x_i$  位于分离超平面误分一侧。

# 软间隔算法步骤

- (1) 给定训练集  $T = \{(x_1, y_1), \dots, (x_l, y_l)\} \in (R^n \times \mathcal{Y})^l$ , 其中  $x_i \in R^n, y_i \in \mathcal{Y} = \{1, -1\}, i = 1, \dots, l$ ;
- (2) 选择适当的惩罚参数  $C > 0$ ;
- (3) 构造并求解凸二次规划问题

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j (x_i \cdot x_j) \alpha_i \alpha_j - \sum_{j=1}^l \alpha_j , \\ \text{s. t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, l , \end{aligned}$$

得解  $\alpha^* = (\alpha_1^*, \dots, \alpha_l^*)^T$ ;

# 软间隔算法步骤

(4) 计算  $b^*$ : 选取位于开区间  $(0, C)$  中的  $\alpha^*$  的分量  $\alpha_j^*$ , 据此计算

$$b^* = y_j - \sum_{i=1}^l y_i \alpha_i^* (x_i \cdot x_j);$$

(5) 构造分划超平面  $(w^* \cdot x) + b^* = 0$ , 由此求得决策函数

$$f(x) = \text{sgn}(g(x)),$$

其中

$$g(x) = \sum_{i=1}^l y_i \alpha_i^* (x_i \cdot x) + b^*.$$

# 硬间隔VS软间隔

- ▶ 软间隔算法同样适用于数据完全可分的情况
- ▶ 在数据完全可分时，硬间隔与软间隔的结果不一定相同
- ▶ C的约束作用：可以避免某“野点”严重影响分隔效果

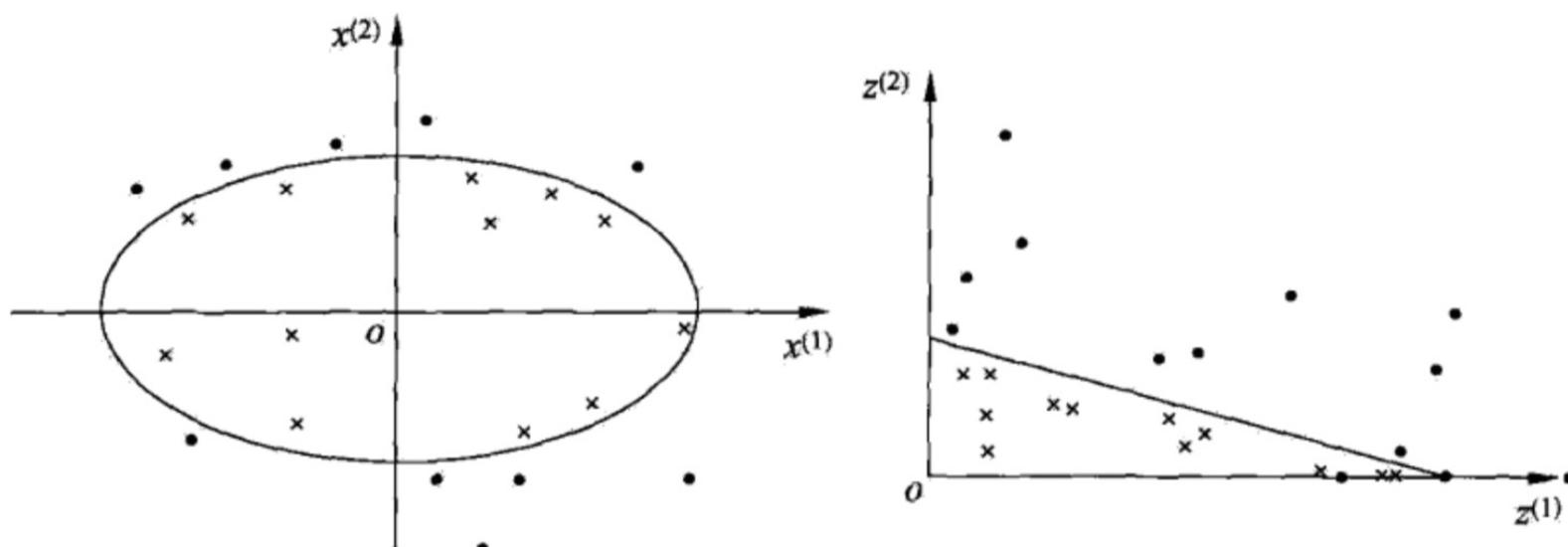
# 非线性支持向量机

- ▶ 关键是引入核函数

核技巧应用到支持向量机，其基本想法就是通过一个非线性变换将输入空间（欧氏空间  $\mathbf{R}^n$  或离散集合）对应于一个特征空间（希尔伯特空间  $\mathcal{H}$ ），使得在输入空间  $\mathbf{R}^n$  中的超曲面模型对应于特征空间  $\mathcal{H}$  中的超平面模型（支持向量机）。这样，分类问题的学习任务通过在特征空间中求解线性支持向量机就可以完成。

## 引入映射函数示例：

$$w_1(x^{(1)})^2 + w_2(x^{(2)})^2 + b = 0$$
$$z = \phi(x) = ((x^{(1)})^2, (x^{(2)})^2)^T$$
$$w_1 z^{(1)} + w_2 z^{(2)} + b = 0$$



# 引入映射函数后原始问题的表达

$$T = \{(x_i, y_i), i = 1, \dots, l\} \in (R^n \times \mathcal{Y})^l,$$

其中  $x_i \in R^n, y_i \in \mathcal{Y} = \{-1, 1\}, i = 1, \dots, l.$

$$\begin{aligned}\Phi : \quad & R^n \rightarrow \mathcal{H}, \\ & x \rightarrow \mathbf{x} = \Phi(x)\end{aligned}$$

$$T_\Phi = \{(\mathbf{x}_i, y_i), i = 1, \dots, l\} \in (\mathcal{H} \times \mathcal{Y})^l,$$

其中  $\mathbf{x}_i = \Phi(x_i) \in \mathcal{H}, y_i \in \mathcal{Y} = \{-1, 1\}, i = 1, \dots, l.$

原始问题:

$$\begin{aligned}\min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l \xi_i, \\ \text{s.t.} \quad & y_i((\mathbf{w} \cdot \Phi(x_i)) + b) \geq 1 - \xi_i, \quad i = 1, \dots, l, \\ & \xi_i \geq 0, \quad i = 1, \dots, l.\end{aligned}$$

# 原问题 -> 对偶问题

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l \alpha_i (y_i (\mathbf{w} \cdot \Phi(x_i)) + b) - 1 + \xi_i - \sum_{i=1}^l \beta_i \xi_i ,$$

其中  $\alpha = (\alpha_1, \dots, \alpha_l)^T$  和  $\beta = (\beta_1, \dots, \beta_l)^T$  均为 Lagrange 乘子向量.

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l \xi_i , \\ \text{s.t.} \quad & y_i ((\mathbf{w} \cdot \Phi(x_i)) + b) \geq 1 - \xi_i , \quad i = 1, \dots, l , \\ & \xi_i \geq 0 , \quad i = 1, \dots, l . \end{aligned}$$



$$\begin{aligned} \max_{\alpha, \beta} \quad & -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (\Phi(x_i) \cdot \Phi(x_j)) + \sum_{j=1}^l \alpha_j , \\ \text{s.t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\ & C - \alpha_i - \beta_i = 0 , \quad i = 1, \dots, l , \\ & \alpha_i \geq 0 , \quad i = 1, \dots, l , \\ & \beta_i \geq 0 , \quad i = 1, \dots, l \end{aligned}$$

# 对偶问题：最大化写成最小化

$$\begin{aligned}
 \max_{\alpha, \beta} \quad & -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (\Phi(x_i) \cdot \Phi(x_j)) + \sum_{j=1}^l \alpha_j , \\
 \text{s.t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\
 & C - \alpha_i - \beta_i = 0, \quad i = 1, \dots, l , \\
 & \alpha_i \geq 0, \quad i = 1, \dots, l , \\
 & \beta_i \geq 0, \quad i = 1, \dots, l
 \end{aligned}$$

$$\begin{aligned}
 \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_{j=1}^l \alpha_j , \\
 \text{s.t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\
 & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, l .
 \end{aligned}$$

$\alpha^* = (\alpha_1^*, \dots, \alpha_l^*)^T$  为任一解，则：

$$w^* = \sum_{i=1}^l \alpha_i^* y_i x_i$$

$$b^* = y_j - \sum_{i=1}^l y_i \alpha_i^* (x_i \cdot x_j).$$

# 非线性分类算法步骤

- (1) 给定训练集  $T = \{(x_1, y_1), \dots, (x_l, y_l)\} \in (R^n \times \mathcal{Y})^l$ , 其中  $x_i \in R^n, y_i \in \mathcal{Y} = \{1, -1\}, i = 1, \dots, l$ ;
- (2) 选取适当的从空间  $R^n$  到 Hilbert 空间的变换  $\Phi : x = \Phi(x)$  以及惩罚参数  $C > 0$ ;
- (3) 构造并求解凸二次规划问题

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_{j=1}^l \alpha_j , \\ \text{s.t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\ & 0 \leq \alpha_i \leq C , \quad i = 1, \dots, l , \end{aligned}$$

得解  $\alpha^* = (\alpha_1^*, \dots, \alpha_l^*)^T$ ;

# 非线性分类算法步骤

(4) 计算  $b^*$ : 选取位于开区间  $(0, C)$  中的  $\alpha^*$  的分量  $\alpha_j^*$ , 据此计算

$$b^* = y_j - \sum_{i=1}^l y_i \alpha_i^* (\Phi(x_i) \cdot \Phi(x_j));$$

(5) 构造决策函数

$$f(x) = \text{sgn}(g(x)),$$

其中

$$g(x) = \sum_{i=1}^l y_i \alpha_i^* (\Phi(x_i) \cdot \Phi(x)) + b^*.$$

# 非线性分类 vs 软间隔

计算  $b^*$

$$(\Phi(x_i) \cdot \Phi(x_j))$$

$$(x_i \cdot x_j)$$

决策函数

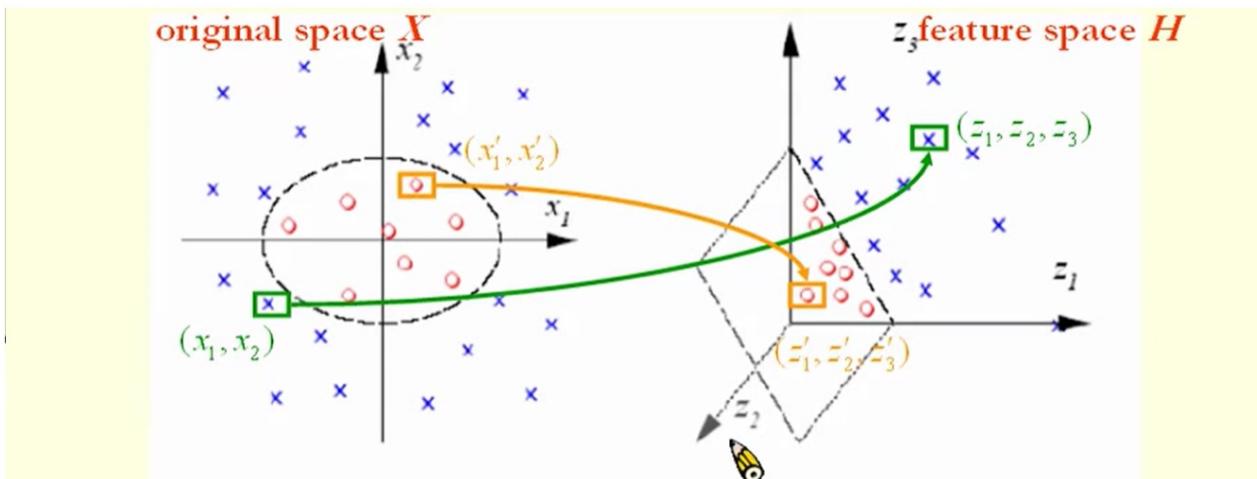
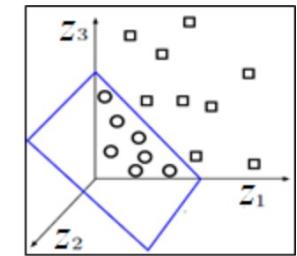
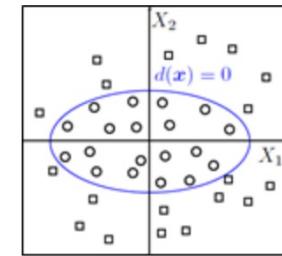
$$(\Phi(x_i) \cdot \Phi(x))$$

$$(x_i \cdot x)$$

# 一个例子

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$



$$\begin{aligned}
 & \langle \phi(x_1, x_2), \phi(x'_1, x'_2) \rangle = \langle (z_1, z_2, z_3), (z'_1, z'_2, z'_3) \rangle = \langle (x_1^2, \sqrt{2}x_1x_2, x_2^2), (x'^1_1, \sqrt{2}x'_1x'_2, x'^2_2) \rangle \\
 & = x_1^2 x'^2_1 + 2x_1 x_2 x'_1 x'_2 + x_2^2 x'^2_2 = (x_1 x'_1 + x_2 x'_2)^2 = (\langle x, x' \rangle)^2 := \kappa(x, x') \leftarrow \text{kernel function}
 \end{aligned}$$

# 核函数的定义

$$\begin{aligned}\Phi : \quad & R^n \rightarrow \mathcal{H}, \\ & x \mapsto \Phi(x),\end{aligned}$$

$$K(x, x') = (\Phi(x) \cdot \Phi(x')) , \text{ 其中 } (\cdot) \text{ 表示空间 } \mathcal{H} \text{ 中的内积.}$$

核函数的使用：避免计算映射函数的内积，简化计算

# 非线性支持向量机步骤

- (1) 给定训练集  $T = \{(x_1, y_1), \dots, (x_l, y_l)\} \in (R^n \times \mathcal{Y})^l$ , 其中  $x_i \in R^n, y_i \in \mathcal{Y} = \{1, -1\}, i = 1, \dots, l$ ;
- (2) 选取适当的核函数  $K(x, x')$  以及惩罚参数  $C > 0$ ;
- (3) 构造并求解凸二次规划问题

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j K(x_i, x_j) \alpha_i \alpha_j - \sum_{j=1}^l \alpha_j , \\ \text{s.t.} \quad & \sum_{i=1}^l y_i \alpha_i = 0 , \\ & 0 \leq \alpha_i \leq C , \quad i = 1, \dots, l , \end{aligned}$$

得解  $\alpha^* = (\alpha_1^*, \dots, \alpha_l^*)^T$ ;

# 非线性支持向量机步骤

(4) 计算  $b^*$ : 选取位于开区间  $(0, C)$  中的  $\alpha^*$  的分量  $\alpha_j^*$ , 据此计算

$$b^* = y_j - \sum_{i=1}^l y_i \alpha_i^* K(x_i, x_j);$$

(5) 构造决策函数

$$f(x) = \text{sgn}(g(x)),$$

其中

$$g(x) = \sum_{i=1}^l y_i \alpha_i^* K(x_i, x) + b^*.$$

# 支持向量机的特点

- ▶ 问题规模的转化：对 $n$ 维向量转化为对偶问题变量个数 $|I|$ 的求解
- ▶ 核函数的使用：避免计算映射函数的内积，简化计算
- ▶ 稀疏性：只有部分训练点对决策函数起作用。对于大型问题计算有重要意义

# 如何选取核函数

- ▶ 相似程度：需要有关领域的专门知识、了解问题的特点

- 距离

$$\Phi(x) = x \quad K(x, x') = (\Phi(x) \cdot \Phi(x')) = (x \cdot x')$$

- 长度

$$\Phi(x) = \|x\| \quad K(x, x') = (\Phi(x) \cdot \Phi(x')) = \|x\| \|x'\|$$

- 幅角

$$\Phi(x) = \frac{x}{\|x\|} \quad K(x, x') = (\Phi(x) \cdot \Phi(x')) = \frac{(x \cdot x')}{\|x\| \|x'\|}$$

- ▶ 当样特征维度不高时，样本数量也不多时，考虑用高斯核函数
- ▶ 当样本的特征很多时，特征的维数很高，这是往往样本线性可分，可考虑用线性核函数的SVM或LR
- ▶ 当样本的数量很多，但特征较少时，可以手动添加一些特征，使样本线性可分，再考虑用线性核函数的SVM或LR
- ▶ 交叉验证，试用不同的核函数，看哪个效果好
- ▶ 混合核函数

# 常用核函数

► 线性核：即线性可分的情况     $K(x, x') = (\Phi(x) \cdot \Phi(x')) = (x \cdot x')$

► 多项式核函数     $K(x, z) = (x \cdot z + 1)^p$

► 高斯核函数     $K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$

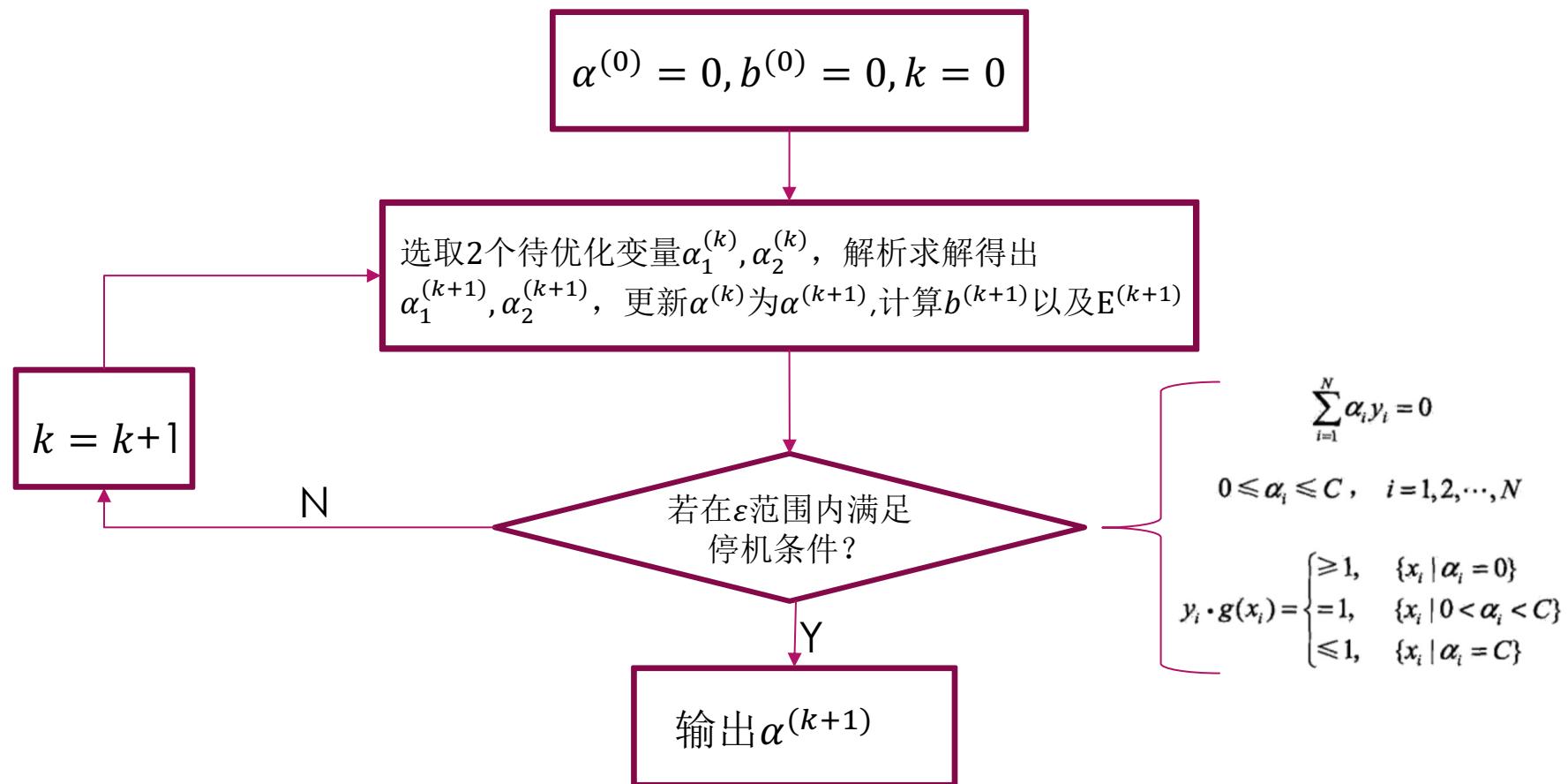
# SMO算法

- ▶ 求解凸二次规划问题的一种快速算法。
- ▶ ps: 解决凸二次规划问题，有许多最优化算法可以使用，然而当训练样本容量很大时，这些算法往往变得非常低效。

# SMO算法的基本思路

- ▶ 目标：让所有变量的解都满足此最优化问题的KKT条件，如果满足，则表示找到最优解。
- ▶ 过程：选择两个变量，固定其他变量（此时其他变量作为常量出现），针对这两个变量构建一个二次规划问题。这个两个变量的二次规划问题可以用解析解求解，计算速度很快。将求出的解更新现有值，可以使得原始的二次规划问题的目标函数值变得更小。重复这个过程，直到达到目标。
- ▶ 如何选择两个变量 (1) 首先选择违反KKT条件最严重的（自变量）； (2) 另一个由约束条件自动确定。

# SMO算法流程



# 如何选择变量： 变量1

- ▶ 第一个变量的选择：违反KKT条件最严重的样本点。首先遍历检查所有满足  $0 < \alpha_i < C$  的样本点；其次再遍历整个训练集，检验其是否满足KKT的条件。

$$\alpha_i = 0 \Leftrightarrow y_i g(x_i) \geq 1$$

$$0 < \alpha_i < C \Leftrightarrow y_i g(x_i) = 1$$

$$\alpha_i = C \Leftrightarrow y_i g(x_i) \leq 1$$

$$g(x_i) = \sum_{j=1}^N \alpha_j y_j K(x_i, x_j) + b$$

# 如何选择变量： 变量2

## ▶ 第二个变量的选择：

- 使 $|E_1 - E_2|$ 最大，算出所有 $E_i$ ，视 $E_1$ 的正负去选择最小还是最大的 $E_i$ （ $E_i$ 的计算）
- 如果上述选择不能使目标函数有足够的下降，则使用启发式规则继续选择（1）间隔边界的支撑向量点，依次将其对应变量试用；（2）如无合适，遍历整个数据集。如无法找到合适的，放弃当前变量1，重新找其他的变量1

$$E_i = g(x_i) - y_i = \left( \sum_{j=1}^N \alpha_j y_j K(x_j, x_i) + b \right) - y_i, \quad i=1,2$$

# 如何求解两个变量的二次规划问题

$$\alpha_2^{\text{new,unc}} = \alpha_2^{\text{old}} + \frac{y_2(E_1 - E_2)}{\eta}$$
$$\eta = K_{11} + K_{22} - 2K_{12}$$
$$\alpha_2^{\text{new}} = \begin{cases} H, & \alpha_2^{\text{new,unc}} > H \\ \alpha_2^{\text{new,unc}}, & L \leq \alpha_2^{\text{new,unc}} \leq H \\ L, & \alpha_2^{\text{new,unc}} < L \end{cases}$$
$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2 (\alpha_2^{\text{old}} - \alpha_2^{\text{new}})$$
$$E_i = g(x_i) - y_i = \left( \sum_{j=1}^N \alpha_j y_j K(x_j, x_i) + b \right) - y_i, \quad i=1,2$$
$$y_1 \neq y_2$$
$$L = \max(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}}), \quad H = \min(C, C + \alpha_2^{\text{old}} - \alpha_1^{\text{old}})$$
$$y_1 = y_2$$
$$L = \max(0, \alpha_2^{\text{old}} + \alpha_1^{\text{old}} - C), \quad H = \min(C, \alpha_2^{\text{old}} + \alpha_1^{\text{old}})$$

## 更新b和E

$$b_1^{\text{new}} = -E_1 - y_1 K_{11} (\alpha_1^{\text{new}} - \alpha_1^{\text{old}}) - y_2 K_{21} (\alpha_2^{\text{new}} - \alpha_2^{\text{old}}) + b^{\text{old}}$$

$$b_2^{\text{new}} = -E_2 - y_1 K_{12} (\alpha_1^{\text{new}} - \alpha_1^{\text{old}}) - y_2 K_{22} (\alpha_2^{\text{new}} - \alpha_2^{\text{old}}) + b^{\text{old}}$$

当  $0 < \alpha_1^{\text{new}} < C$  时,  $0 < \alpha_2^{\text{new}} < C$ ,  $b^{\text{new}} = b_1^{\text{new}} = b_2^{\text{new}}$

其他情况:

$$b^{\text{new}} = (b_1^{\text{new}} + b_2^{\text{new}})/2$$

$$E_i^{\text{new}} = \sum_s y_j \alpha_j K(x_i, x_j) + b^{\text{new}} - y_i$$