支持向量机

SVM(Support Vector Machines)

SVM

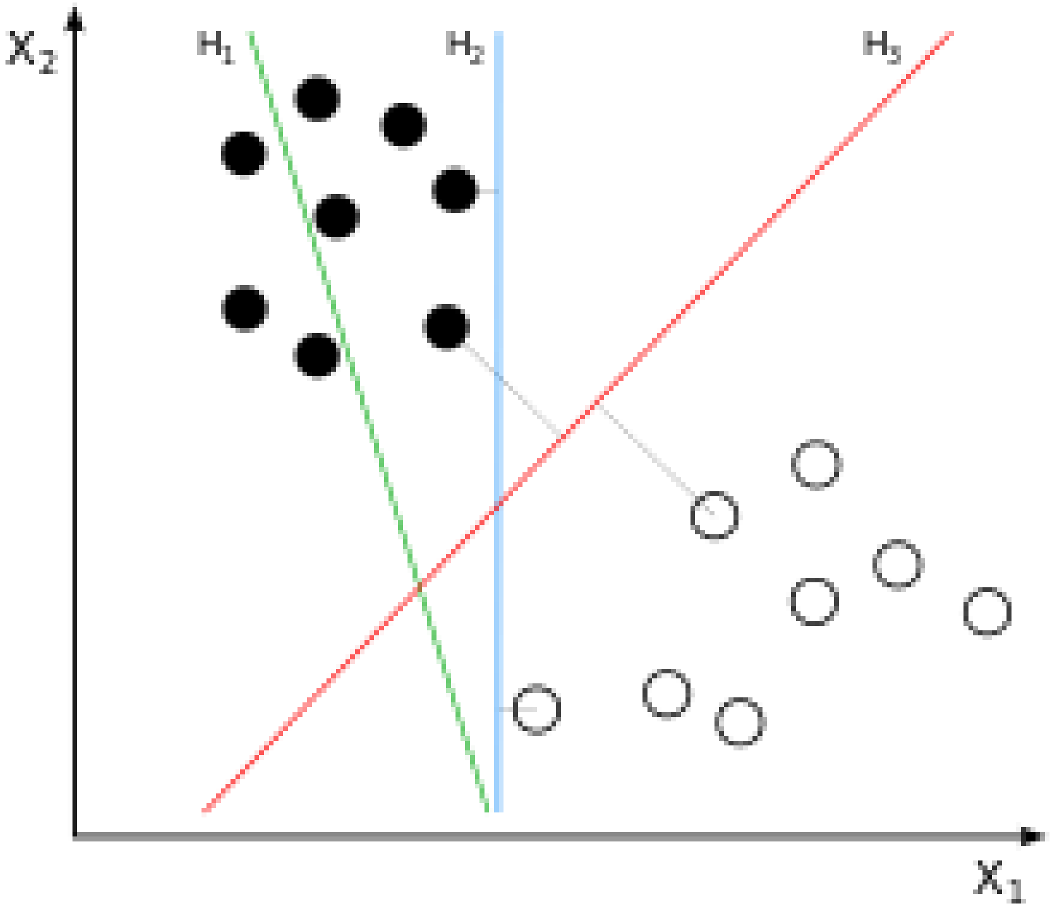
最早是由 Vladimir N. Vapnik 和 Alexey Ya. Chervonenkis 在1963年提出

目前的版本(soft margin)是由Corinna Cortes 和 Vapnik在1993年提出，并在1995年发表

深度学习（2012）出现之前，SVM被认为机器学习中近十几年来最成功，表现最好的算法

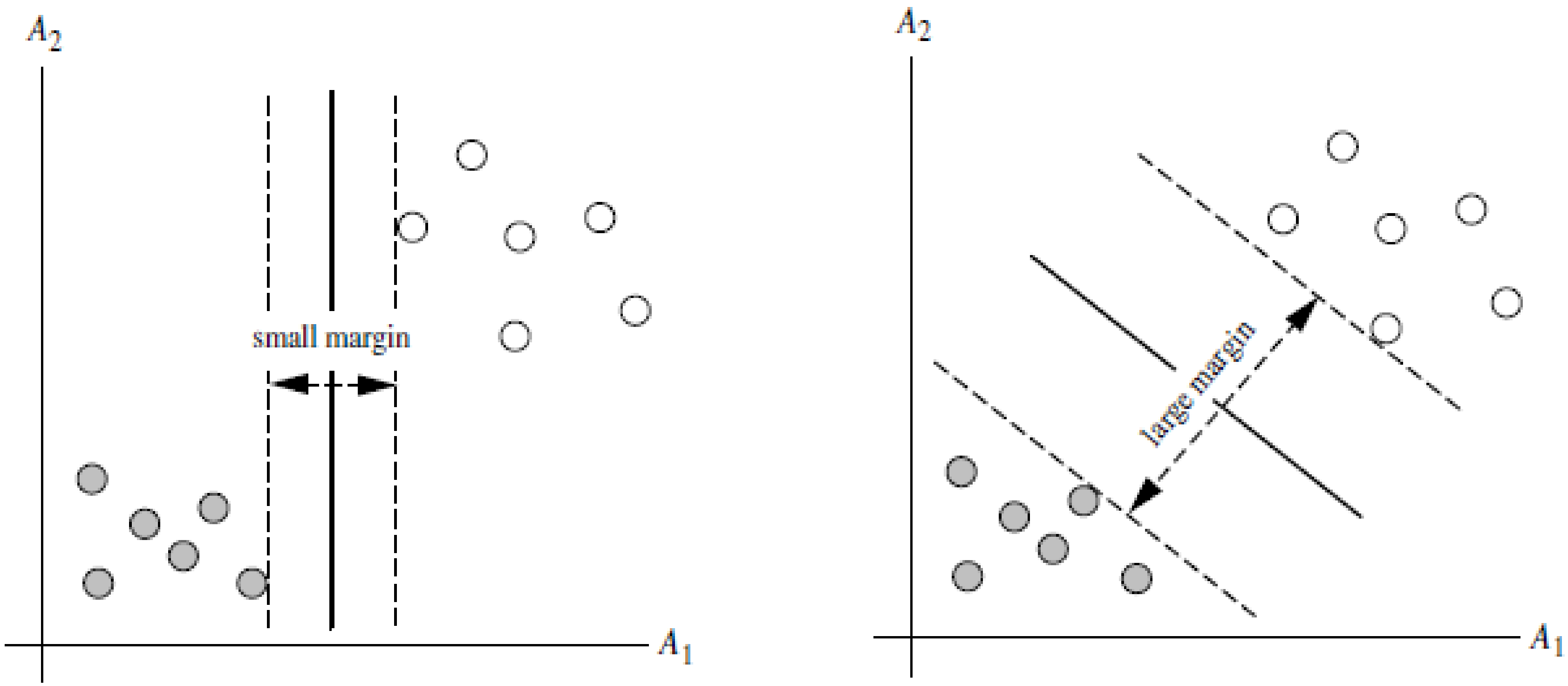
SVM

两个类别，黑白
红线为分界线好



SVM

SVM寻找区分两类的超平面（hyper plane),
使边际(margin)最大



向量内积

$$x = \begin{Bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{Bmatrix} \qquad y = \begin{Bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{Bmatrix}$$

向量内积：$x \cdot y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$

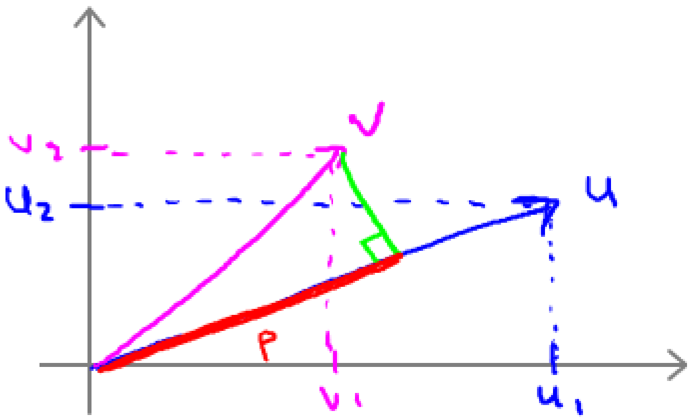
向量内积：$x \cdot y = \|x\| \|y\| \cos(\theta)$

范数：$\|x\| = \sqrt{x \cdot x} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

当$\|x\| \neq 0, \|\mathrm{y}\| \neq 0$时，可以求余弦相似度：

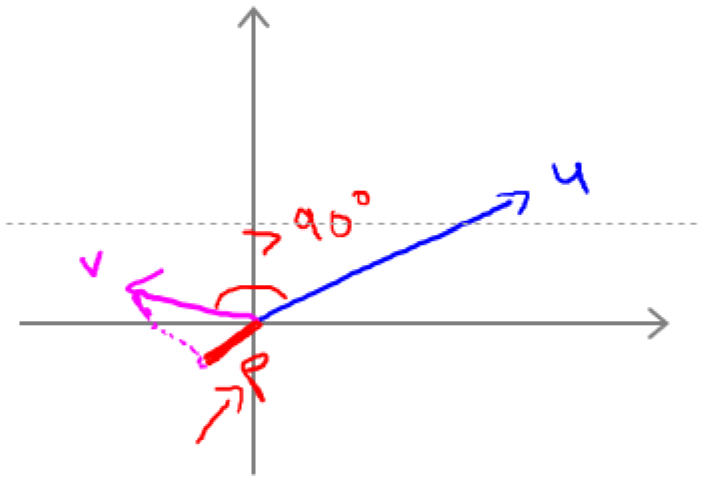
$$cos\theta = \frac{x \cdot y}{\|x\| \|y\|}$$

向量内积



夹角小于九十度

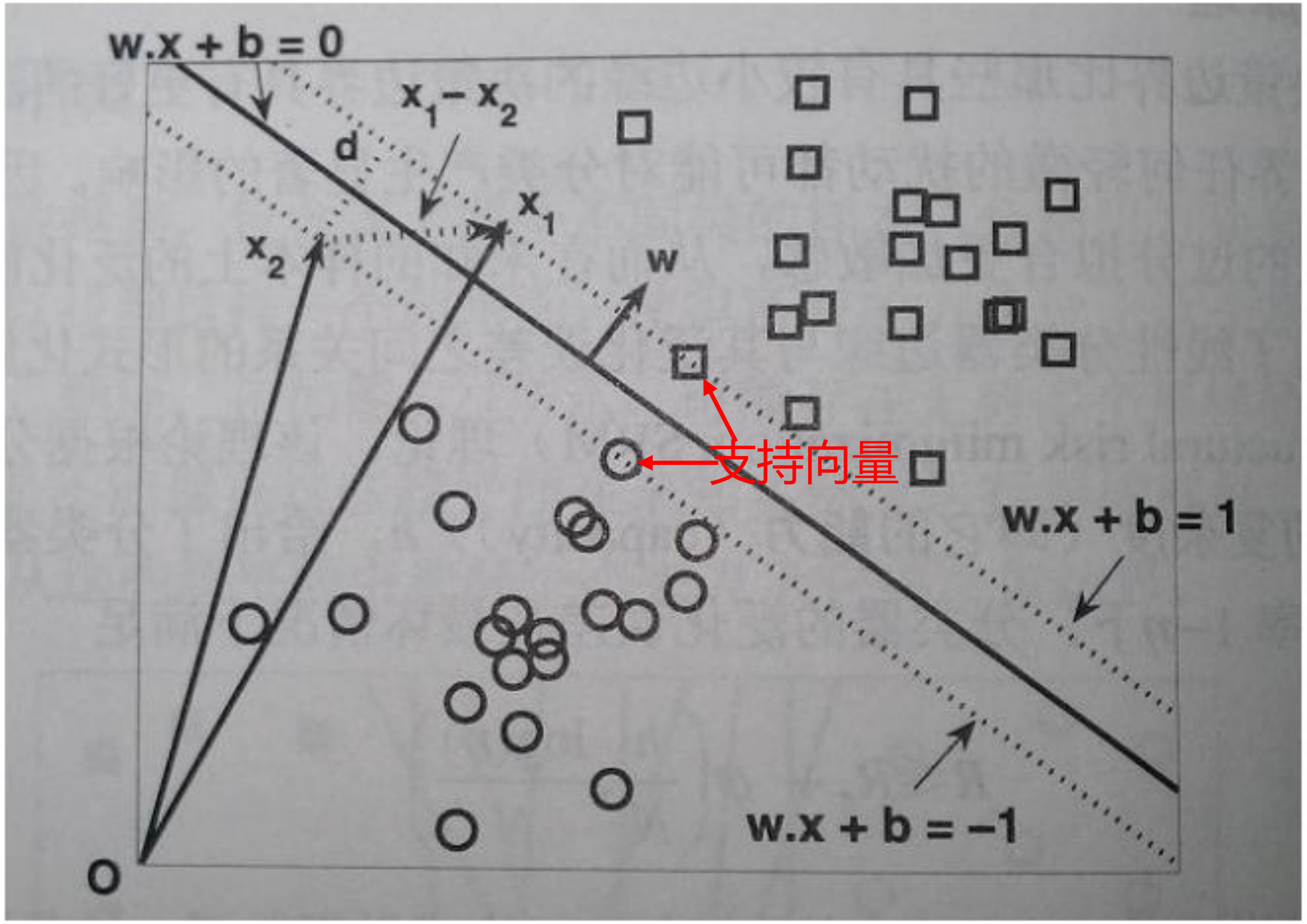
向量内积：$v \cdot u > 0$



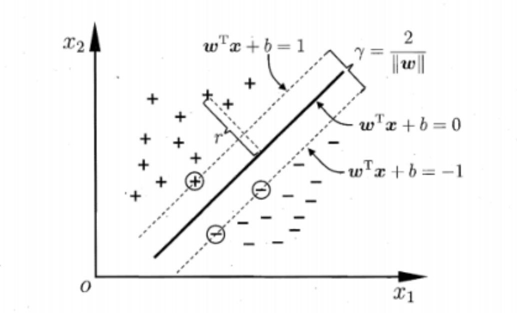
向量内积：$v \cdot u < 0$

SVM分类

支持向量，和虚线相切的点，可能一个或多个，



一些推导



划分超平面可以定义为一个线性方程：$w^T X + b = 0$，其中：

- $w = \{w_1; w_2; \ldots; w_d\}$ 是一个法向量，决定了超平面的方向，d 是特征值的个数
- X 为训练样本
- b 为位移项，决定了超平面与原点之间的距离

$$w \cdot x + b = 1$$
$$w \cdot x + b = -1$$

$$w \cdot x + b = 2$$
$$w \cdot x + b = -3$$

调整w和b可以调成1和-1

2*u+v=1

-3*u+v=-1

u=-2/5

v=1/5

X_i 为支持向量点的特征值;

$$w \cdot x_1 + b = 1$$
$$w \cdot x_2 + b = -1$$

$$w \cdot (x_1 - x_2) = 2$$

和图对应

$$\|w\| \|(x_1 - x_2)\| \cos(\theta) = 2$$
$$\|w\| *d=2$$
$$d = \frac{2}{\|w\|}$$

系数

d相当于两条线的距离
要距离大揑

d越大越好

转化为凸优化问题

y_i 是支持向量点X_i的类别标记（class label），比如+1还是-1；

等于在虚线上，是支持向量

$$\left.\begin{array}{l} w\cdot x+b\geq 1\text{，则分类y=1} \\ w\cdot x+b\leq -1\text{，则分类y=-1} \end{array}\right\} \quad y(w\cdot x+b)\geq 1$$

求$d=\frac{2}{\|w\|}$最大值，

也就是求$min\frac{\|w\|^2}{2}$

要求这个的最小值，还要满足约束条件
属于第三种，要最小值要取等号，
还是第二类，拉格朗日

凸优化问题

1.无约束优化问题：

$$\min f(\boldsymbol{x})$$

-费马定理 求导，导为0

2.带等式约束的优化问题：

-拉格朗日乘子法：

$$\min f(\boldsymbol{x})$$

$$s.t.\ h_i(\boldsymbol{x}) = 0, \qquad i = 1,2,\cdots n$$

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \sum_{i=1}^{n} \lambda_i h_i(\boldsymbol{x})$$

3.带不等式约束的优化问题：

-KKT条件

$$\min f(\boldsymbol{x})$$

$$s.t.\ h_i(\boldsymbol{x}) = 0, \qquad i = 1,2,\cdots,n$$

$$g_i(\boldsymbol{x}) \le 0, \qquad i = 1,2,\cdots,k$$

$$\mathcal{L}(x, \lambda, v) = f(x) + \sum_{i=1}^{k} \lambda_i g_i(x) + \sum_{i=1}^{n} v_i h_i(x)$$

广义拉格朗日乘子法

满足约束条件同时使得代价函数的值最小

目标函数　α拉格朗日乘子　约束条件

带约束的代价函数 $L(w,b,a)=\boxed{\frac{1}{2}\|w\|^2}-\sum_{i=1}^{n}\alpha_i\boxed{(y_i(w^Tx_i+b)-1)}$

目的还是求这个式子的最小值

代价函数，要求它的最小值

$$\frac{\partial L}{\partial w}=0\rightarrow w=\sum_{i=1}^{n}\alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b}=0\rightarrow \sum_{i=1}^{n}\alpha_i y_i=0$$

跟岭回归和LASSO类似

岭回归代价函数：

$$J(\theta)=\frac{1}{2m}\left[\sum_{i=1}^{m}\left(\overset{\text{预测值}}{h_\theta\left(x^{(i)}\right)}-\underset{\text{真实值}}{y^{(i)}}\right)^2+\lambda\overset{\text{约束条件}}{\sum_{j=1}^{n}}\theta_j^2\right]$$

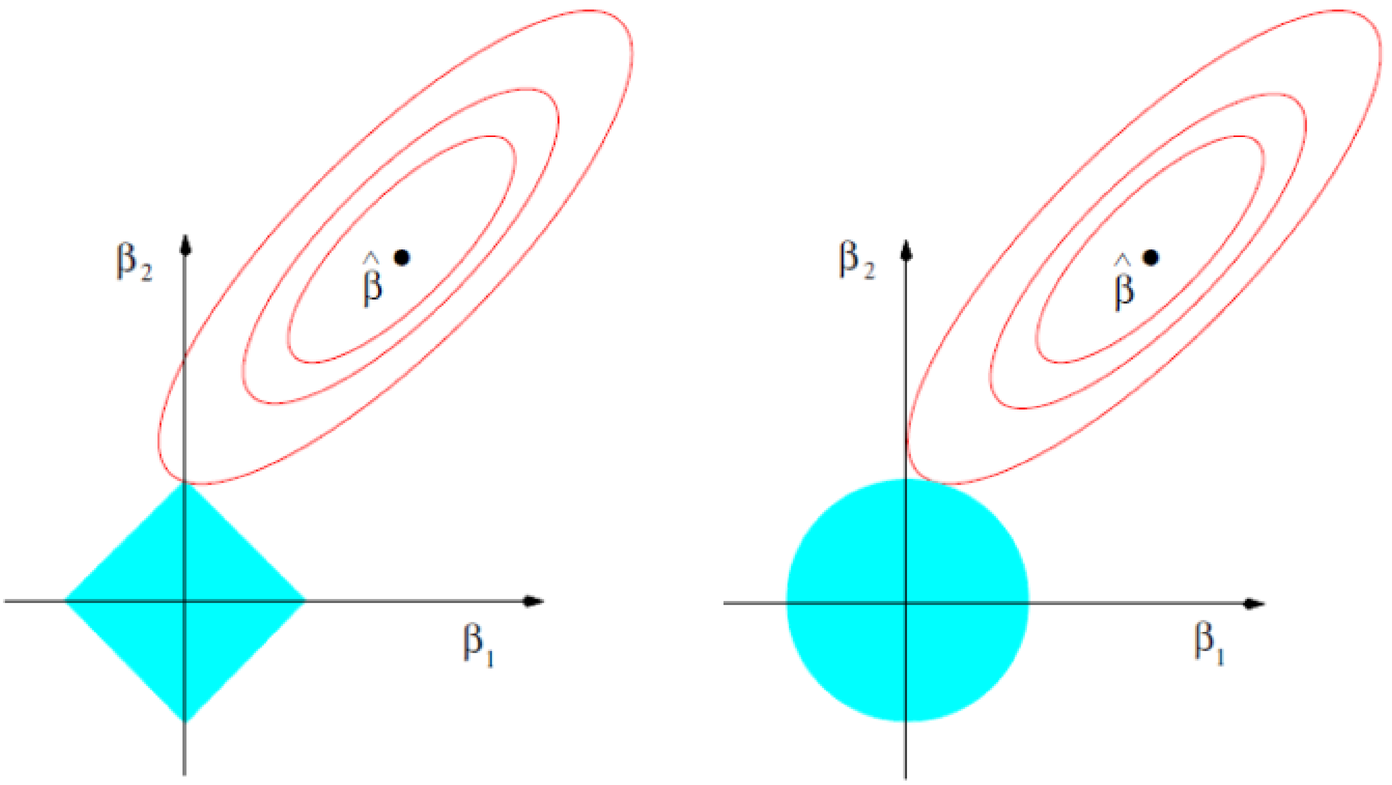
λ的值可以用于限制$\sum_{j=1}^{n}\theta_j^2\leq t$

LASSO代价函数：

$$J(\theta)=\frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2+\lambda\sum_{j=1}^{n}\left|\theta_{\mathrm{j}}\right|\right]$$

λ的值可以用于限制$\sum_{j=1}^{n}\left|\theta_{\mathrm{j}}\right|\leq t$

跟岭回归和LASSO类似



LASSO 岭回归

Karush-Kuhn-Tucker最优化条件(KKT条件)

拉格朗日乘子法的一种推广，可以处理有不等号的约束条件。

$$\min f(\boldsymbol{x})$$

$$s.t.\ h_i(\boldsymbol{x}) = 0, \qquad i = 1,2,\cdots,n$$

$$g_i(\boldsymbol{x}) \leq 0, \qquad i = 1,2,\cdots,k$$

$$\mathcal{L}(x,\lambda,v) = f(x) + \sum_{i=1}^{k} \lambda_i g_i(x) + \sum_{i=1}^{n} v_i h_i(x)$$

进一步简化为对偶问题

只考虑阿尔法，这个式子有最大值

阿尔法越大这个前面有减号的式子越小

阿尔法也大于等于0

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

这部分大于等于0

上述问题可以改写成：

先只考虑阿尔法，找后面式子的最大值，再只考虑w,b找L的最小值

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

可以等价为下列对偶问题：有条件才可换成对偶，SVM正好可以

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

先求L关于w,b的最小值，再求阿尔法的最大值

进一步简化为对偶问题

$$\inf_{w,b} L(w,b,\alpha) = \frac{1}{2}w^T w + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}a_i y_i w^T x_i - \sum_{i=1}^{m}a_i y_i b$$

这个为0了

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

这一页就相当于求L关于w,b的最小值

$$\frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{n}\alpha_i y_i x_i$$

这两个代回L

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\inf_{w,b} L(w,b,\alpha) = -\frac{1}{2}w^T\sum_{i=1}^{m}a_i y_i x_i + \sum_{i=1}^{m}\alpha_i$$

$$= -\frac{1}{2}(\sum_{i=1}^{m}a_i y_i x_i)^T\sum_{i=1}^{m}a_i y_i x_i + \sum_{i=1}^{m}\alpha_i$$

把w和b消除了

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

这个和那个1对应

这个和平方-四个乘对应

进一步简化为对偶问题

这一页要求前一页式子关于阿尔法的最大值

$$\max_{\alpha_i \geq 0} \min_{\boldsymbol{w},b} \mathcal{L}(\boldsymbol{w},b,\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{k} \alpha_i \alpha_j y_i y_j (x_i)^T x_j \right]$$

还要满足约束条件

$$s.t. \sum_{i=1}^{k} \alpha_i y_i = 0, \qquad \alpha_i \geq 0, i = 1,2,\cdots,n$$

$$\min_{\boldsymbol{\alpha}} \left[\frac{1}{2} \sum_{i,j=1}^{k} \alpha_i \alpha_j y_i y_j (x_i)^T x_j - \sum_{i=1}^{k} \alpha_i \right] = \min_{\boldsymbol{\alpha}} \left[\frac{1}{2} \sum_{i,j=1}^{k} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{k} \alpha_i \right]$$

$$s.t. \sum_{i=1}^{k} \alpha_i y_i = 0, \qquad \alpha_i \geq 0, i = 1,2,\cdots,n$$

进一步简化为对偶问题

由此可以求出最优解α^*，求出该值后将其带入可以得到：

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

$$b^* = y_i - (w^*)^T x_i$$

SMO算法

Microsoft Research的John C. Platt在1998年提出
针对线性SVM和数据稀疏时性能更优

X_i 为支持向量点的特征值；

$$\min_{\alpha}\left[\frac{1}{2}\sum_{i,j=1}^{k}\alpha_i\alpha_j y_i y_j (x_i)^T x_j-\sum_{i=1}^{k}\alpha_i\right]=\min_{\alpha}\left[\frac{1}{2}\sum_{i,j=1}^{k}\alpha_i\alpha_j y_i y_j (x_i\cdot x_j)-\sum_{i=1}^{k}\alpha_i\right]$$

xi是坐标，是样本xi的坐标，yi是标签+1或-1

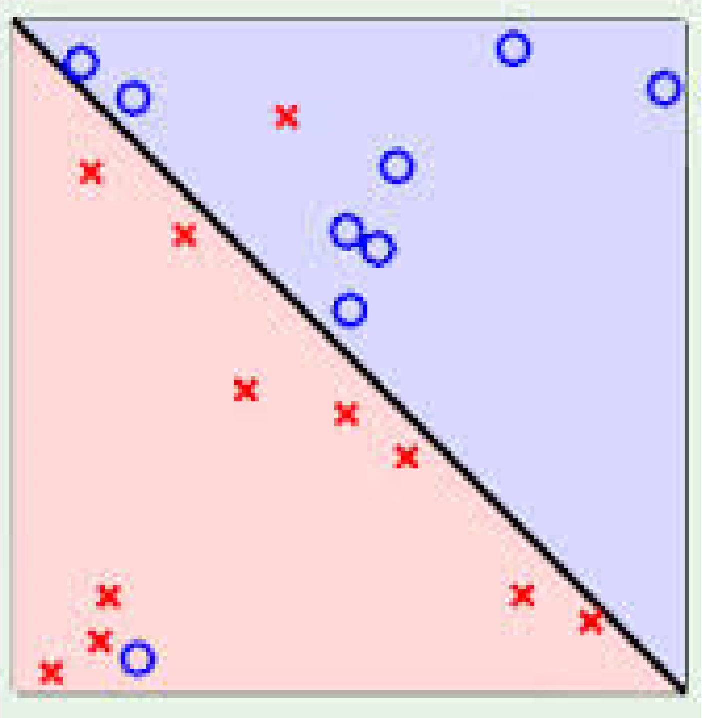
$$s.t.\sum_{i=1}^{k}\alpha_i y_i=0,\qquad \alpha_i\geq 0, i=1,2,\cdots,n$$

s.t.,C⩾α_i⩾0,i=1,⋯,n

基本思路是先根据约束条件随机给α赋值。然后每次选取两个α，调节这两个α使得目标函数最小。然后再选取两个α，调节α使得目标函数最小。以此类推

迭代 逼近

线性不可分的情况



松弛变量与惩罚函数

损失函数(Loss Function)：
是定义在单个样本上的，用来评价模型的预测值和真实值不一样的程度，指一个样本的误差。

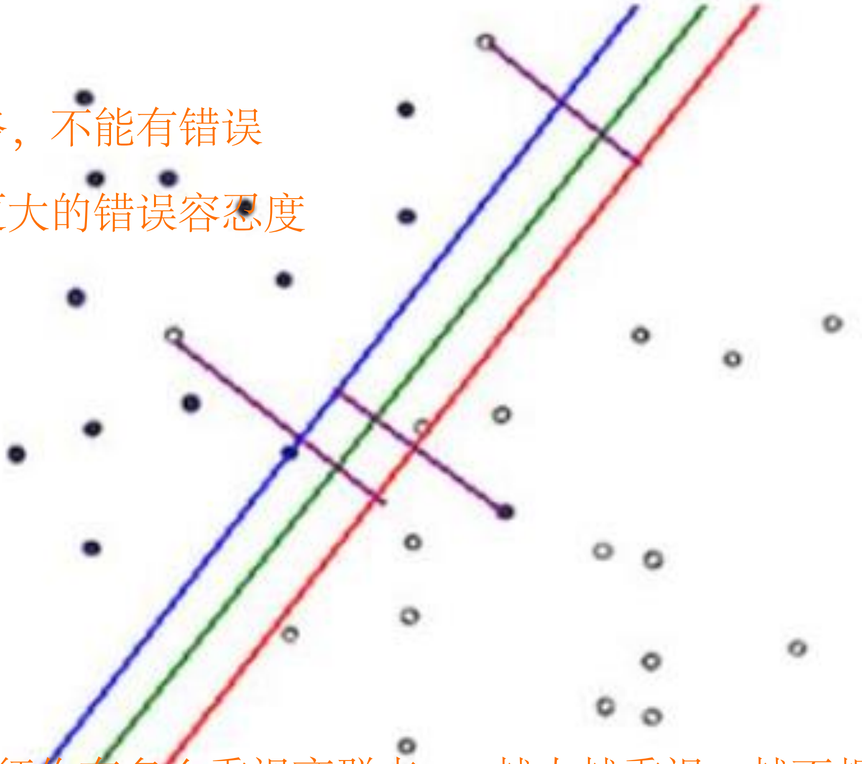
代价函数(Cost Function)：
定义在整个训练集上的，是指所有样本误差的平均，也就是所有损失函数值的平均。

目标函数(Object Function)：
指最终需要优化的函数，一般来说是经验风险加结构风险（代价函数+正则化项），正则化项指惩罚项，做矫正作用。

以红线为例，要e=1,则，红线与绿线重合
e=2,红线与蓝线重合

c越大，分类越严格，不能有错误

c越小，意味着有更大的错误容忍度



margin边际

$$y_i(w_i \cdot x_i + b) \geq 1 - \varepsilon_i, \varepsilon_i \geq 0$$

松弛变量

约束条件没有体现错误分类的点要尽量接近分类边界

目标函数　　　　惩罚函数

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

C人为调节因子

使得分错的点越少越好，距离分类边界越近越好

C:惩罚因子。C表征你有多么重视离群点，C越大越重视，越不想丢掉它们。
C值大时对误差分类的惩罚增大，C值小时对误差分类的惩罚减小。
当C越大，趋近无穷的时候，表示不允许分类误差的存在，margin越小，容易过拟合；
当C趋于0时，表示我们不再关注分类是否正确，只要求margin越大，容易欠拟合。

线性不可分情形下的对偶问题

$$\min_{\alpha}\left[\frac{1}{2}\sum_{i,j=1}^{k}\alpha_i\alpha_j y_i y_j (x_i)^T x_j - \sum_{i=1}^{k}\alpha_i\right] = \boxed{\min_{\alpha}\left[\frac{1}{2}\sum_{i,j=1}^{k}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{k}\alpha_i\right]}$$

求此式的最小值

$$s.t.\sum_{i=1}^{k}\alpha_i y_i = 0, \qquad \alpha_i \geq 0, i = 1,2,\cdots,n$$

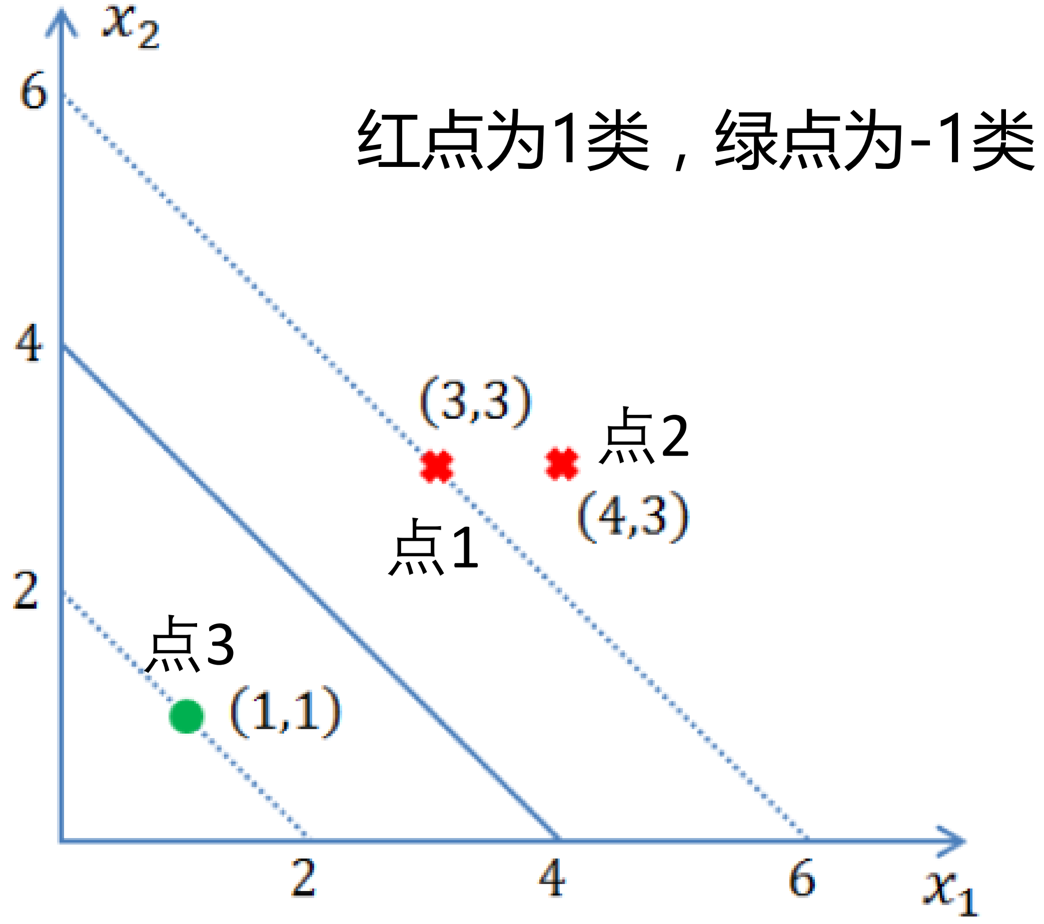
y是标签

s.t.,C⩾α_i⩾0,i=1,⋯,n

同时满足这两个式子

前面一页的C

SVM例子



SVM例子

两个向量a = [a1, a2,⋯, an]和b = [b1, b2,⋯, bn]的点积定义为:
a · b=a1b1+a2b2+……+anbn。

可知目标函数为

$$\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}), \quad s.t. \alpha_1 + \alpha_2 - \alpha_3 = 0, \quad \alpha_i \geq 0, i = 1,2,3$$

其中

此式中3平方+3平方
xi,xj为横纵坐标

求内积

$$f(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{3} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{3} \alpha_i$$

y是标签

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

4平方+3平方　　3*4+3*3的和*2，有 点1 点2 和 点2 点1

然后，将$\alpha_3 = \alpha_1 + \alpha_2$带入目标函数，得到一个关于$\alpha_1$和$\alpha_2$的函数

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

SVM例子

对α_1和α_2求偏导数并令其为0，易知$s(\alpha_1,\alpha_2)$在点$(1.5,-1)$处取极值。而该点不满足$a_i \geq 0$的约束条件，于是可以推断最小值在边界上达到。经计算当$\alpha_1=0$时，$s(\alpha_1=0,\alpha_2=2/13)=-0.1538$；当$\alpha_2=0$时，$s(\alpha_1=1/4,\alpha_2=0)=-0.25$。于是$s(\alpha_1,\alpha_2)$在$\alpha_1=1/4$，$\alpha_2=0$时取得最小值，此时亦可算出$\alpha_3=\alpha_1+\alpha_2=1/4$。因为$\alpha_1$和$\alpha_3$不等于0，所以对应的点$x_1$和$x_3$就应该是支持向量。

SVM例子

在数学和计算机运算中，其功能是取某个数的符号（正或负）：

当x>0，sign(x)=1;

当x=0，sign(x)=0;

当x<0， sign(x)=-1；

进而可以求得

斜率 $$\boldsymbol{w}^* = \sum_{i=1}^{3} \alpha_i^* y_i x_i = \frac{1}{4} \times (3,3) - \frac{1}{4} \times (1,1) = \left(\frac{1}{2}, \frac{1}{2}\right)$$

即$w_1 = w_2 = 0.5$。进而有 用的是阿尔法不为0的样本，不为0才是支持向量

截距 $$b^* = 1 - (w_1, w_2) \cdot (3,3) = -2$$

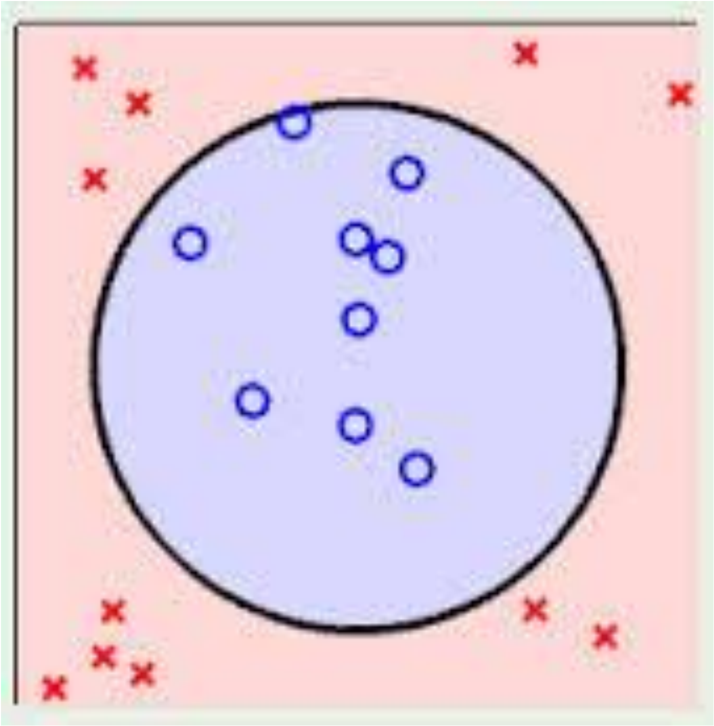
因此最大间隔分类超平面为

$$\frac{1}{2} x_1 + \frac{1}{2} x_2 - 2 = 0$$

分类决策函数为

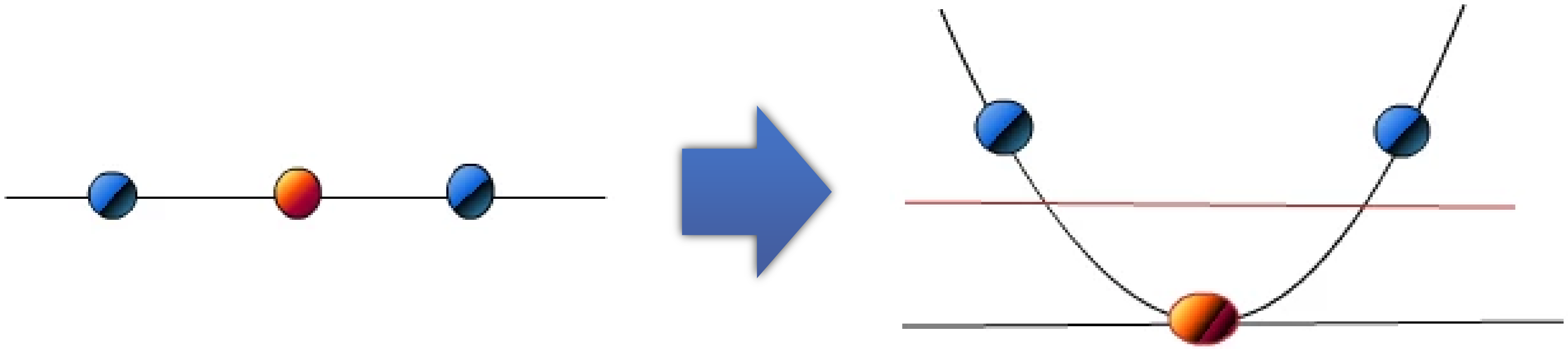
$$f(\boldsymbol{x}) = sign\left(\frac{1}{2} x_1 + \frac{1}{2} x_2 - 2\right)$$

非线性的情况

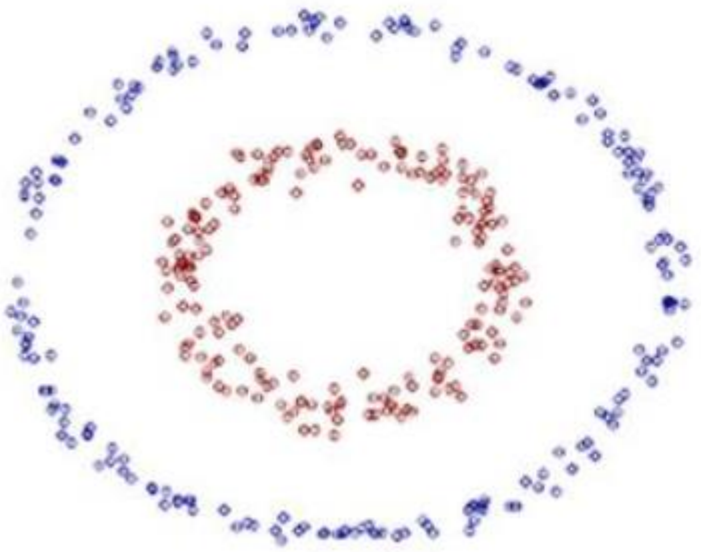


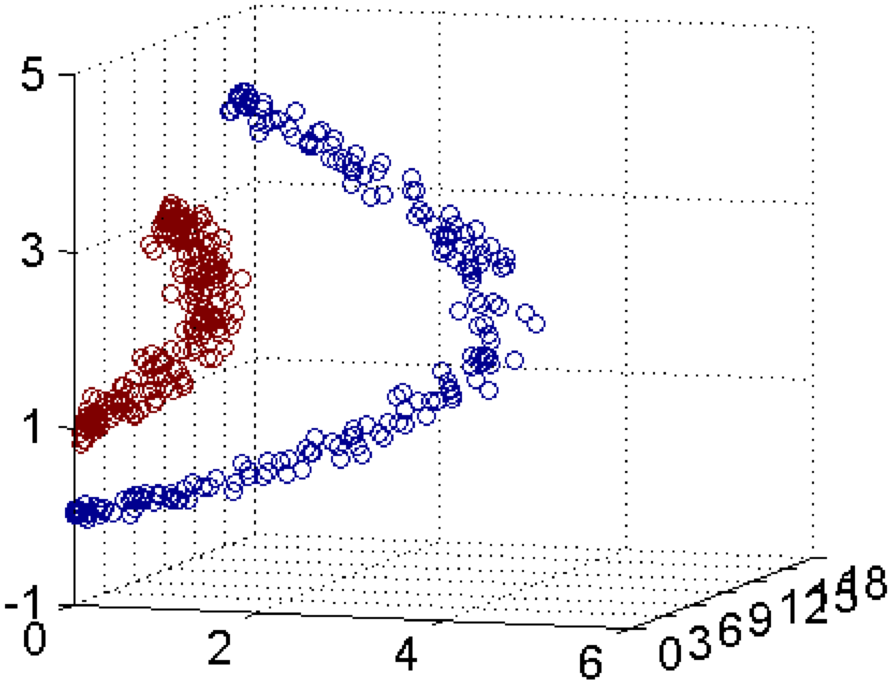
非线性的情况

把低维空间的非线性问题映射到
高维空间，变成求解线性问题

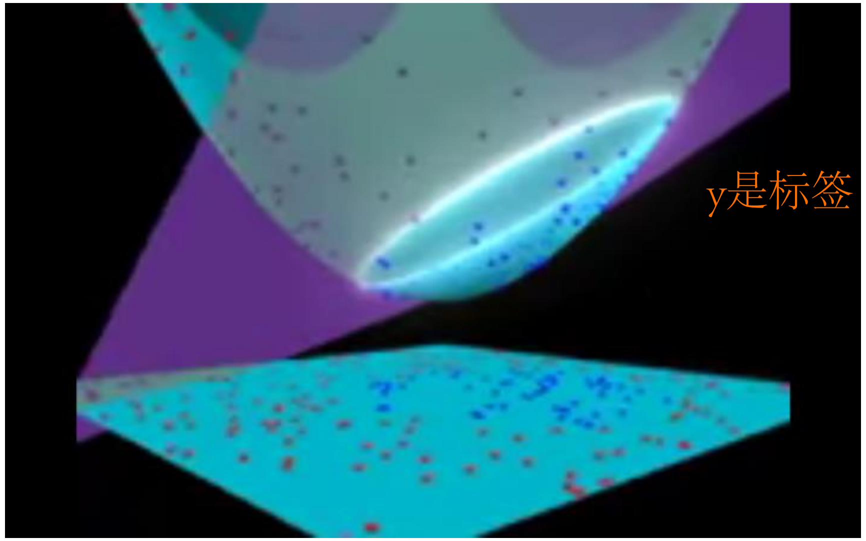


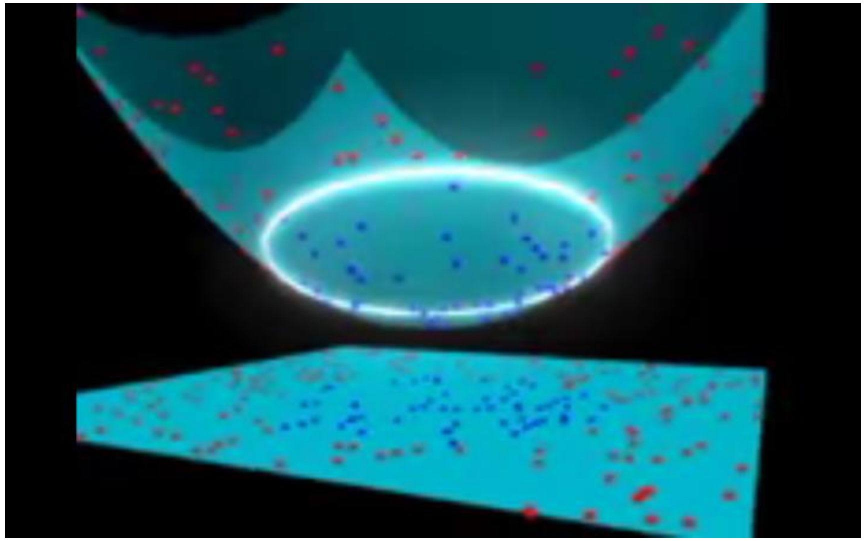
非线性的情况





非线性的情况





映射举例

3维输入向量： $X = (x_1, x_2, x_3)$

转化到6维空间 Z 中去：

$$\phi_1(X) = x_1,\ \phi_2(X) = x_2,\ \phi_3(X) = x_3,\ \phi_4(X) = (x_1)^2,\ \phi_5(X) = x_1x_2,\ \text{and}\ \phi_6(X) = x_1x_3.$$

新的决策超平面：$d(Z) = \mathbf{WZ} + b$，其中W和Z是向量，这个超平面是线性的，解出W和b之后，并且带回原方程：

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned}$$

存在的问题

$$min_{\alpha}\left[\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j\right],\sum_{i=1}^{k}\alpha_i y_i=0\,,\mathrm{C}\geq\alpha_i\geq 0$$

$$min_{\alpha}\left[\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \emptyset(x_i)^T\emptyset(x_j)\right],\sum_{i=1}^{k}\alpha_i y_i=0\,,\mathrm{C}\geq\alpha_i\geq 0$$

1.维度灾难
红色的地方要使用映射后的样本向量做内积
假如最初的特征是n维的，我们把它映射到n^2维，然后再计算。这样需要的时间从原来的的O(n)，变成了O(n^2)

2.如何选择合理的非线性转换？

引入核函数

注意RBF公式高斯核函数里面的sigma和gamma的关系如下：

$$k(x,z)=\exp(-\frac{d(x,z)^2}{2*\sigma^2})=\exp(-gamma\cdot d(x,z)^2)\Rightarrow gamma=\frac{1}{2\cdot\sigma^2}$$

我们可以构造核函数使得运算结果等同于非线性映射，同时运算量要远远小于非线性映射。

$$\underbrace{K(X_i,X_j)}_{\text{核函数}}=\underbrace{\phi(X_i)\cdot\phi(X_j)}_{\text{非线性映射再求内积}}$$

h 次多项式核函数：$K(X_i,X_j)=(X_i\cdot X_j+1)^h$

高斯径向基函数核函数：$K(X_i,X_j)=e^{-\|X_i-X_j\|^2/2\sigma^2}$

S 型核函数：$K(X_i,X_j)=\tanh(\kappa X_i\cdot X_j-\delta)$

gamma: 是’rbf’，’poly’和’sigmoid’的核系数且gamma的值必须大于0。
随着gamma的增大，存在对于测试集分类效果差而对训练分类效果好的情况，并且容易泛化误差出现过拟合。

核函数举例

假设定义两个向量： x = (x1, x2, x3); y = (y1, y2, y3)
定义高维映射方程：f(x) = (x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)
假设x = (1, 2, 3)，y = (4, 5, 6).
f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)
f(y) = (16, 20, 24, 20, 25, 36, 24, 30, 36)
求内积<f(x), f(y)> = 16 + 40 + 72 + 40 + 100+ 180 + 72 + 180 + 324 = 1024

内积
定义核函数：K(x,y) = (<f(x), f(y)>)^2
K(x,y) = (4 + 10 + 18)^2 = 1024
同样的结果，使用核方法计算容易得多。

SVM优点

- 训练好的模型的算法复杂度是由支持向量的个数决定的，而不是由数据的维度决定的。所以SVM不太容易产生overfitting过拟合
- SVM训练出来的模型完全依赖于支持向量(Support Vectors), 即使训练集里面所有非支持向量的点都被去除，重复训练过程，结果仍然会得到完全一样的模型。
- 一个SVM如果训练得出的支持向量个数比较小，SVM训练出的模型比较容易被泛化。