

课程号: 180206081100M1001H-01

## 第15章<sub>(第1讲)</sub>

# 支持向量机与核方法

## Support Vector Machine & Kernel Methods

向 世 明

smxiang@nlpr.ia.ac.cn

<https://people.ucas.ac.cn/~xiangshiming>

时空数据分析与学习课题组 (STDAL)

中科院自动化研究所 多模态人工智能系统国家重点实验室

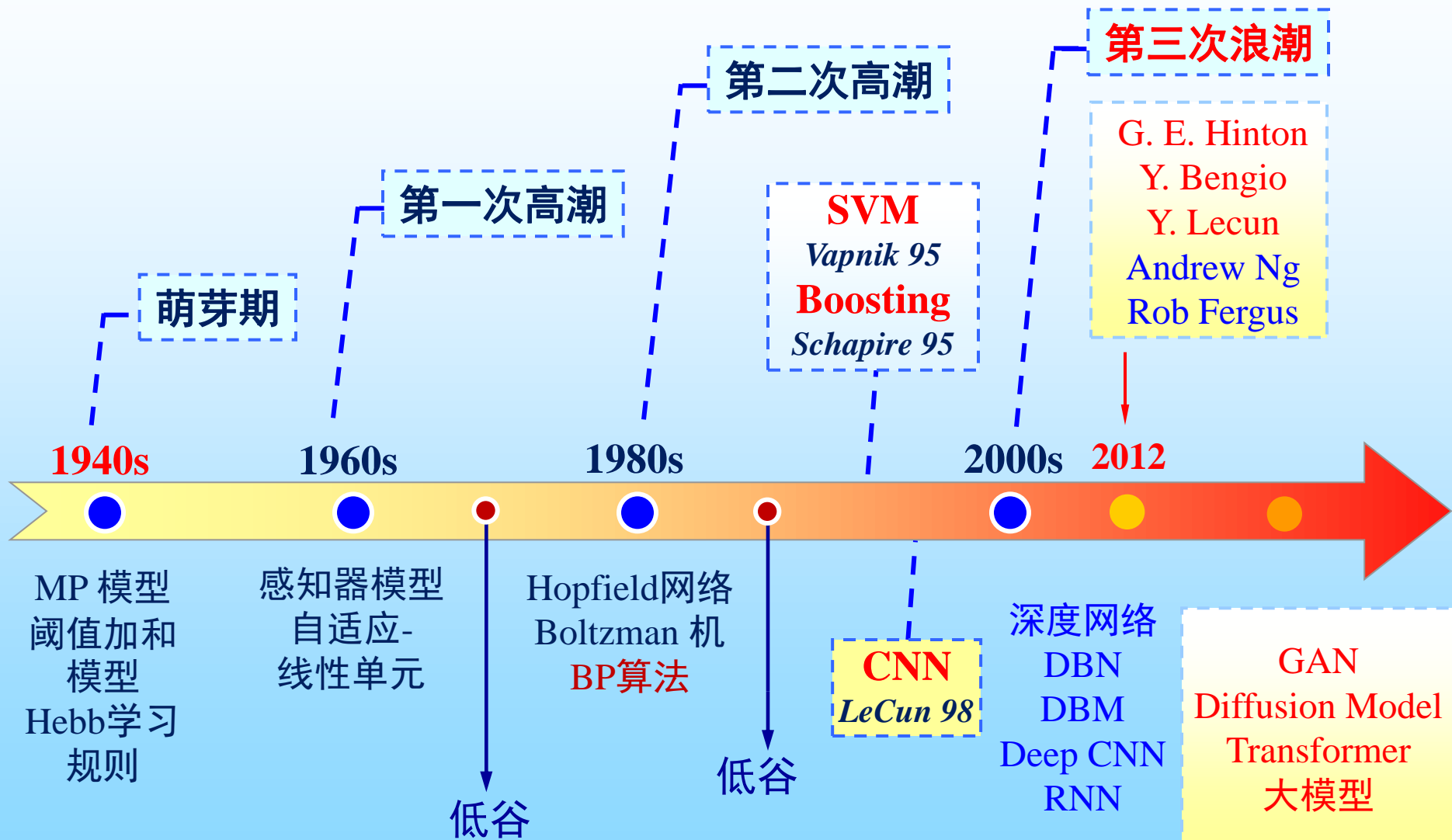
助教: 杨 奇 ( yangqi2021@ia.ac.cn )

张 涛 ( zhangtao2021@ia.ac.cn )

# History



大规模、开源、高性能计算



# History

Boser, Guyon, Vapnik, **A training algorithm for optimal margin classifiers**. COLT 1992.

Cortes, Vapnik, **Support-vector networks**. Machine Learning, **1995**.



- SVM is **a classifier** derived from statistical learning theory by Vapnik and Chervonenkis (万普尼克-切尔沃纳基斯)
- SVMs introduced by Boser, Guyon, Vapnik in **COLT-92**
- **Initially popularized** in the **NIPS community**, an important and active field of all machine learning research (Now named as NeurIPS) .
- Special issues of **Machine Learning Journal**, and **Journal of Machine Learning Research (JMLR)**.

# History (1995-2005, 2006-now)



第一个赌（正方）：Jackel: 1995-2000 五年时间从理论上弄懂神经网络

1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand quantitatively why big neural nets working on large databases are not so bad. (Understanding means that there will be clear conditions and bounds)

Vapnik bets (one fancy dinner) that Jackel is wrong.

赌约：one fancy dinner

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

\*\*\*\*\*

2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong

第二赌（反方）：Vapnik: 1995-2005 十年时间也无法弄清楚，没人使用神经网络

V. Vapnik

3/14/95

L. Jackel

3/14/95

Witnessed by Y. LeCun

3/14/95

见证人：Y. LeCun

1995年的赌约

Larry Jackel、Vladimir Vapnik和Yann LeCun

2000 年，他们在纽约的 Siam Garden 餐厅享用了这顿晚餐，Jackel 和 Vapnik 平分帐单

2017 NIPS (加州长滩)

Ali Rahimi (阿里·拉希米)

NIPS 2017 “Test of Time” Award

《Random Features for Large-Scale Kernel Machines》

将数据映射至随机的低维特征空间，  
然后再采用快速线性方法

时间检验奖



人工智能是新的电力，  
但今天的AI何尝又不是一种炼金术

Ali Rahimi 发表了精彩的论文解读和获奖演讲



机器学习已经成了炼金术 ?!

# 15.1 Related Theory (计算学习理论)

## 15.1.1 Structural Risk Minimization

- We want to get a low error rate on **unseen data** (始终的目标) .
  - This is called “**structural risk minimization**” (结构风险最小化)
  - Training error is “**empirical risk minimization**” (经验风险最小化)
- A Way to Choose the Best Model: It would be really helpful if we could get a guarantee:

$$\text{Test error rate} \leq \text{train error rate} + f(N, h, p)$$

where  $N$ : size of training set

$h$ : measure of the model complexity

$p$ : the probability that this bound fails (该界失败的概率)

We need  $p$  to allow for really unlucky training/test sets.

- 损失和模型复杂度通常是相反的。
  - Then we could choose the model complexity that minimizes the **bound on the test error rate**. (找到一个合适的中间点, 来平衡损失和模型复杂度)

## 15.1.2 VC Dimension

- Pick  $n$  data points, assign labels of “+” or “-” to them at random.
- If our model (e.g. a neural net with a certain number of hidden units) is powerful enough to learn any association of labels with the data, **its too powerful!**
- Characterize the power of a model by asking how many data points it can “shatter”.
- Hope to learn perfectly for all possible assignments of labels.
- This number of data points is called the **Vapnik-Chervonenkis dimension**.
- **VC dimension for a 2D plane?**
  - In 2-D, we can find a plane (i.e. a line) to deal with any labeling of three points. A 2-D hyperplane shatters 3 points

- 打散（对shatter的补充解释）

- 在“假设空间” $H$ 中，不同假设对于 $D$ 中示例赋予标记的结果可能相同，也可能不相同；
- 尽管 $H$ 可能包含无穷多个假设，但对其中示例赋予的可能结果数是有限的。对 $m$ 个示例，对两类分类问题，最多有 $2^m$ 个可能的结果。
- 对二分类问题来说， $H$ 中的假设对 $D$ 中示例赋予标记的每一种可能结果称为对 $D$ 的一种“对分”（dichotomy，二分法）。
- 若假设空间 $H$ 能实现示例集 $D$ 上的所有对分，则称示例集 $D$ 能被假设空间 $H$ “打散”（shattering）。

- VC维

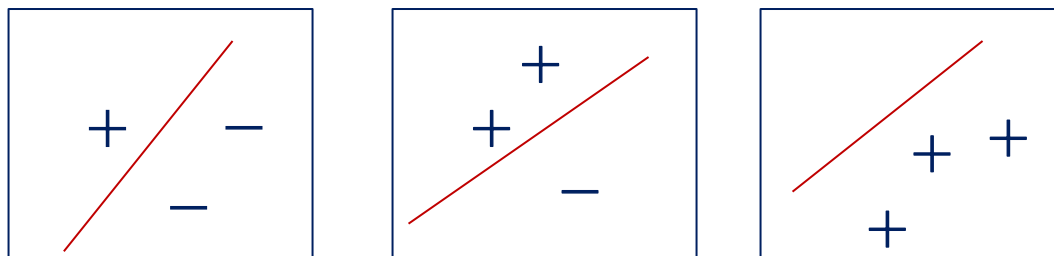
- 假设空间 $H$ 的VC维是能被 $H$ 打散的最大示例集的大小。
- $VC(H)=d$ 表明存在大小为 $d$ 的示例集能被假设空间 $H$ 打散。
- 通常按如下方式来计算VC维：
  - 若存在大小为 $d$ 的示例集能被 $H$ 打散，但不存在任何大小为 $d+1$ 示例集能被 $H$ 打散，则 $H$ 的VC维是 $d$ 。



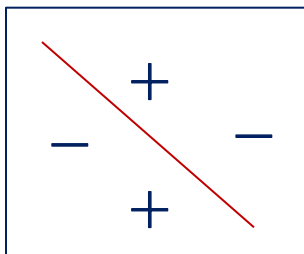
## 15.1.2 VC Dimension

- VC维的计算例子

- 二维平面上的线性划分：(VC维=3)



存在这样的集合，其 $2^3=8$ 种对分均可被线性划分实现



对任何含有四个示例的集合，其 $2^4=16$ 种对分中至少有一种不能被线性划分实现

## 15.1.2 VC Dimension

- The VC dimension of a hyperplane in 2-D is 3.
  - In  $k$  dimensions it is  $k+1$ .
  - It is just a **coincidence** that the VC dimension of a hyperplane is almost identical to the number of parameters it takes to define a hyperplane. (对超平面，其VC维与参数个数相同，只是一个巧合)
- A **sine wave has infinite VC dimension and only 2 parameters!**
  - By choosing the **phase and period** carefully we can shatter any random collection of one-dimensional data points

Let  $x_i=10^i$  where  $i$  ranges from 1 to  $n$ . The classifier  $y= \text{sign}(\sin(ax))$  can classify all  $x_i$  correctly for all possible combination of class labels on  $x_i$

$$f(x) = b \sin(ax)$$



## 15.1.2 VC Dimension

- VC维的计算例子

- 实数域中的区间 $[a, b]$ : 令 $H$ 表示实数域中所有闭区间构成的集合 $\{h_{[a,b]} : a, b \in \mathbb{R}, a \leq b\}$ 。  $X = \mathbb{R}$ 。对 $x \in \mathbb{R}$ , 令

$$\begin{cases} h_{[a,b]}(x) = +1, & \text{if } x \in [a,b] \\ h_{[a,b]}(x) = -1, & \text{otherwise} \end{cases}$$

设 $D = \{x_1=0.5, x_2=1.5\}$



$\{h_{[0,1]}, h_{[0,2]}, h_{[1,2]}, h_{[2,3]}\}$  可以均将 $D$ 打散。所以假设空间 $H$ 的VC维至少为2.

到底为多少呢?

## 15.1.2 VC Dimension

- The **VC-dimension of the nearest neighbor classifier is infinity**, because no matter how many points you have, you get perfect classification on training data
  - 1-NN  $\rightarrow$  K-NN: reduce VC dimension
- The higher the VC-dimension, the more flexible a classifier is （VC维度越高，分类器就越灵活）
- VC-dimension, however, is a theoretical concept
- VC-dimension of most classifiers, in practice, is difficult to be computed exactly （在实践中，大多数分类器的VC维数很难精确计算）
- Qualitatively, if we think a classifier is flexible, it probably has a high VC-dimension

## 15.1.2 VC Dimension

### The Probabilistic Guarantee:

$$E_{test} \leq E_{train} + \left( \frac{h + h \log(2N/h) - \log(p/4)}{N} \right)^{\frac{1}{2}}$$

置信水平:  $\log(1/\delta)$

where  $N$  = size of training set

$h$  = VC dimension of the model

$p$  = upper bound on probability that this bound fails

此边界失败的概率上限

✓ **Good generalization** → **large  $N$  and small  $h$**  (给出有关实践的启示)

- So, if we train models with different complexity, we should pick the one that minimizes this bound

# Large Margin and VC Dimension

- If we use a large set of non-adaptive features, we can often make the two classes linearly separable. (如果特征选择得很粗糙, 总能做到两类线性可分)
  - But if we just fit any separating plane, it will not generalize well to new cases. (可是, 对于复杂的情形, 假设过度适配一系列可分平面, 可以降低泛化性能。这就需要有一个准则)
- If we fit the separating plane that **maximizes the margin** (the minimum distance to any of the data points), we will get much better generalization. (换一种思路: 如果我们最大化各样本到分类面的最小距离)
  - Intuitively, by maximizing the margin we are **squeezing out all the surplus capacity** that came from using a high-dimensional feature space. (即, 最大化利用高维特征空间所带来的剩余容量)
- This can be justified by a whole lot of clever mathematics which shows that (一些数学证明表明):
  - **Large margin separators have lower VC dimension.**
  - Models with lower VC dimension have a smaller gap between the training and test error rates. (VC维数较低的模型在训练和测试错误率之间的差距较小)

## 15.1.3 Some Philosophy



- **Occam's Razor (奥卡姆剃刀原理)**
  - Entities should not be multiplied unnecessarily
  - 如无必要，勿增实体
  - All other things being equal, the simplest solution is the best



- **Vapnik's principle**
  - never to solve a problem that is more general than you actually need to solve : （永远不要试图去解决比你实际需要解决的更普遍的问题）



- **牛顿**
  - 如果某一原因既真又足以解释自然事物的特性，则我们不当接受比这更多的原因

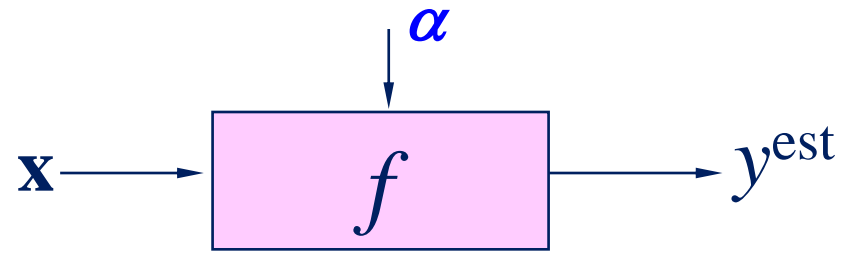
## 15.1.3 Some Philosophy

- **Simplest pattern classifier : 最简单的模式分类器?**
  - Binary two-class problem
  - Linear classifier
- **Which linear classifier?**
  - The large margin one
- **Extension for nonlinear case**
  - Any function can be a linear function if transformed to a high-dimension space
  - **Kernel method**: the inner product in high-dimension space
  - Almost all kernel methods (including SVM) adopt the linear classifier as its initial study objective
- **Extension for multi-class case**
  - 1 vs All, 1 vs 1 , ...



# 15.2 Hard-Margin SVM

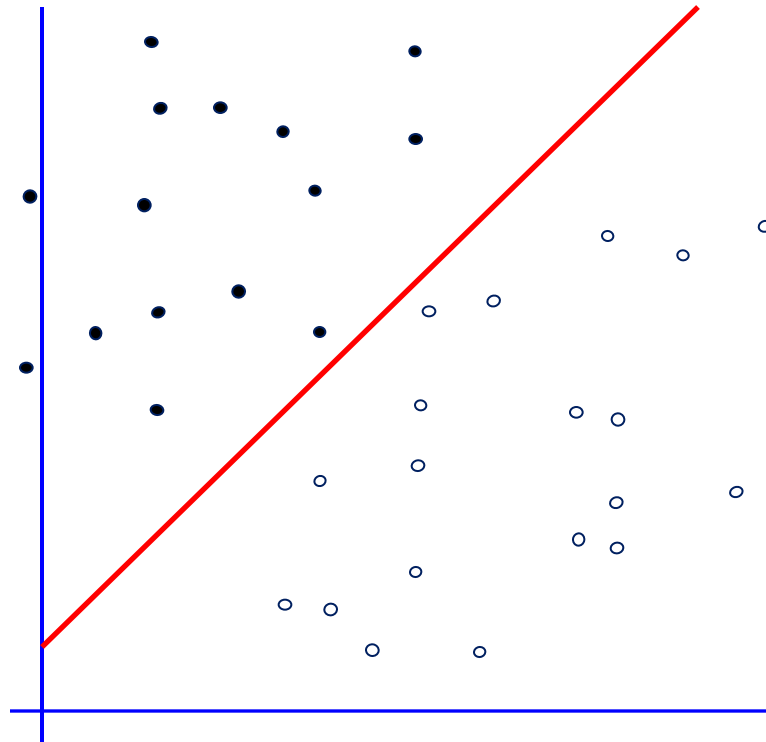
# Linear Classifiers



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

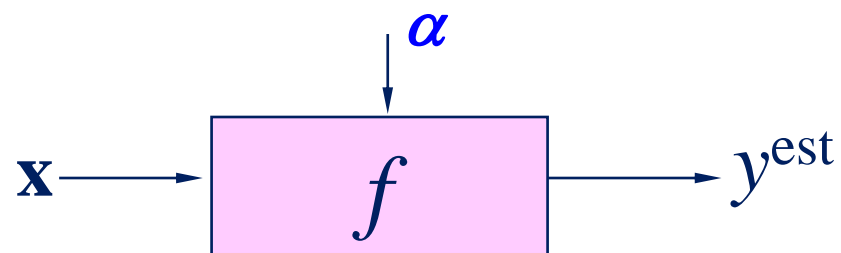
• : +1

○ : -1



How would you  
classify this data?

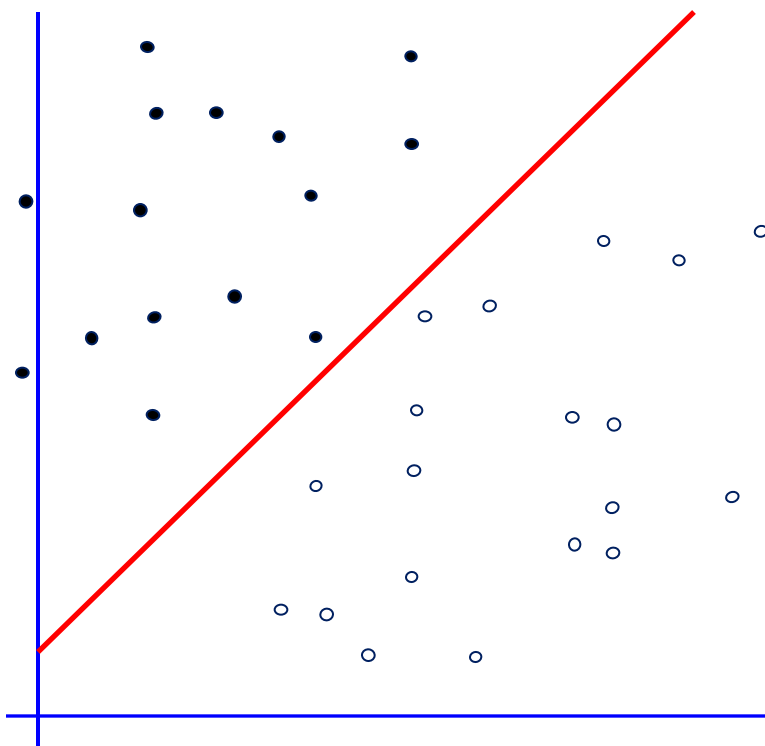
## 15.2.1 Linear Classifiers



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

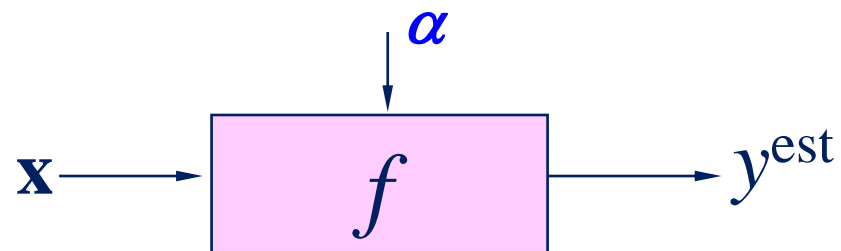
• : +1

○ : -1



How would you  
classify this data?

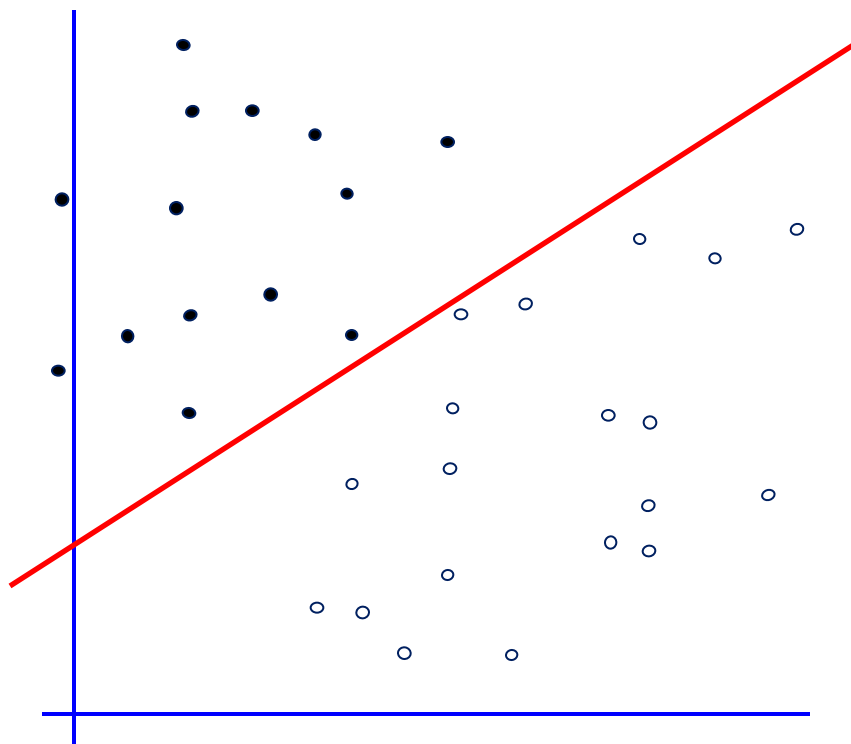
## 15.2.1 Linear Classifiers



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

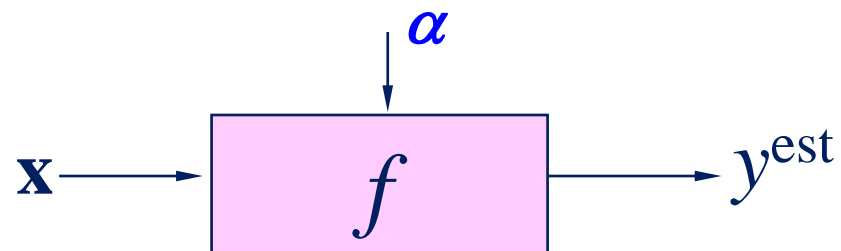
• : +1

○ : -1



How would you  
classify this data?

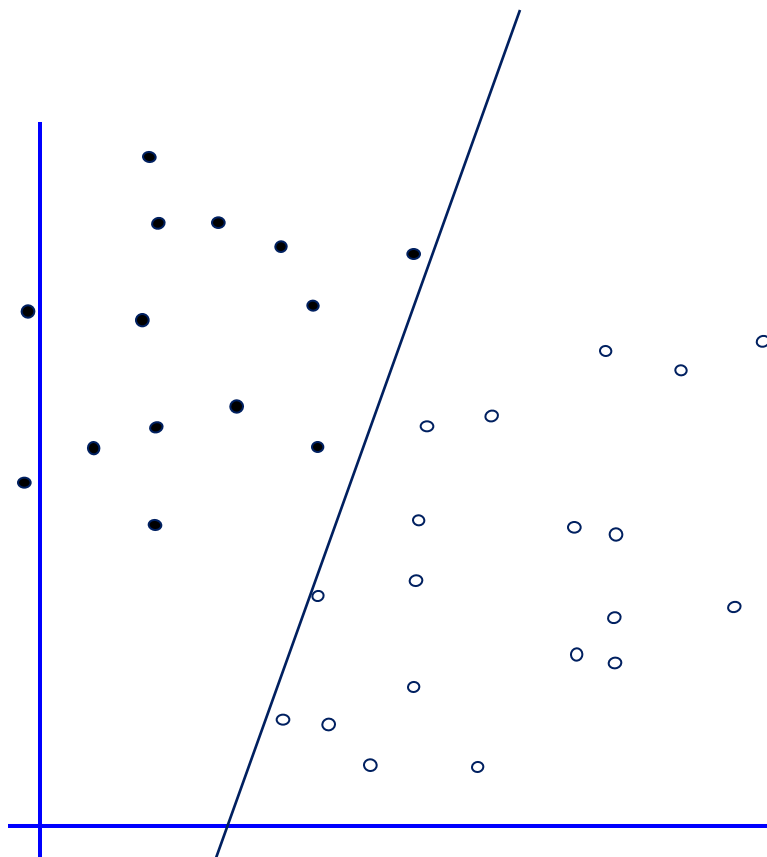
## 15.2.1 Linear Classifiers



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

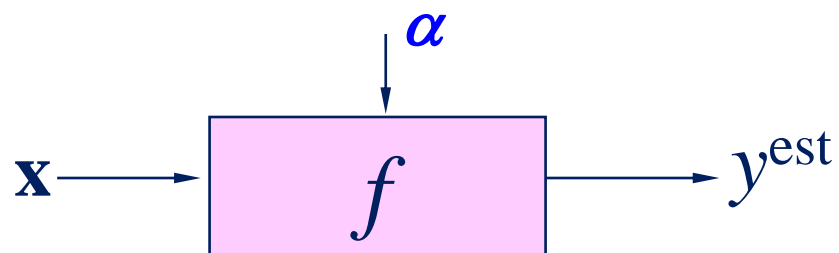
• : +1

◦ : -1



How would you  
classify this data?

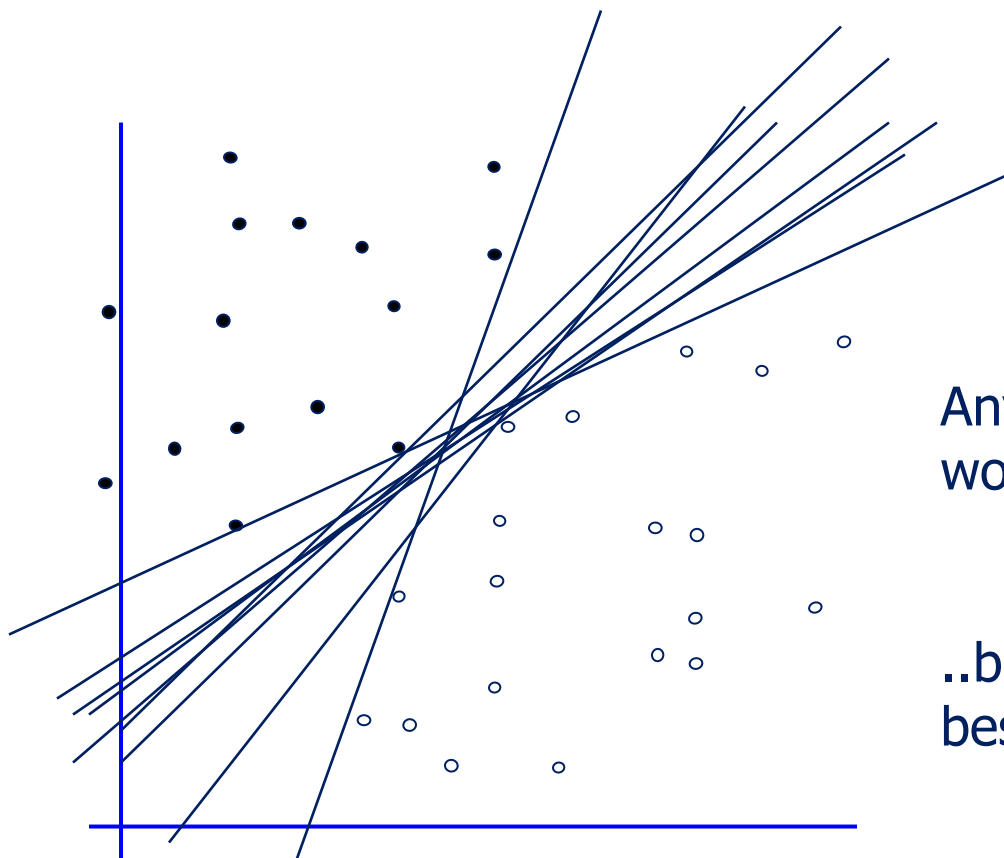
## 15.2.1 Linear Classifiers



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

• : +1

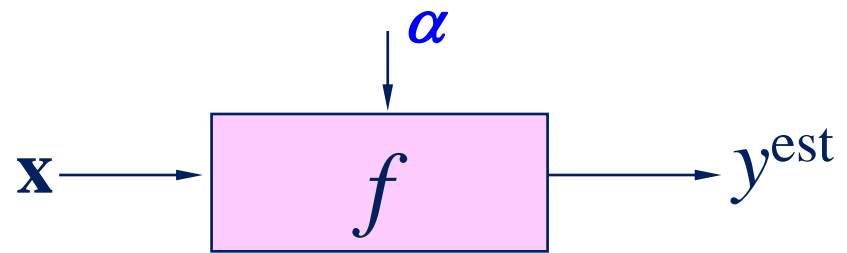
◦ : -1



Any of these  
would be fine..

..but which is  
best?

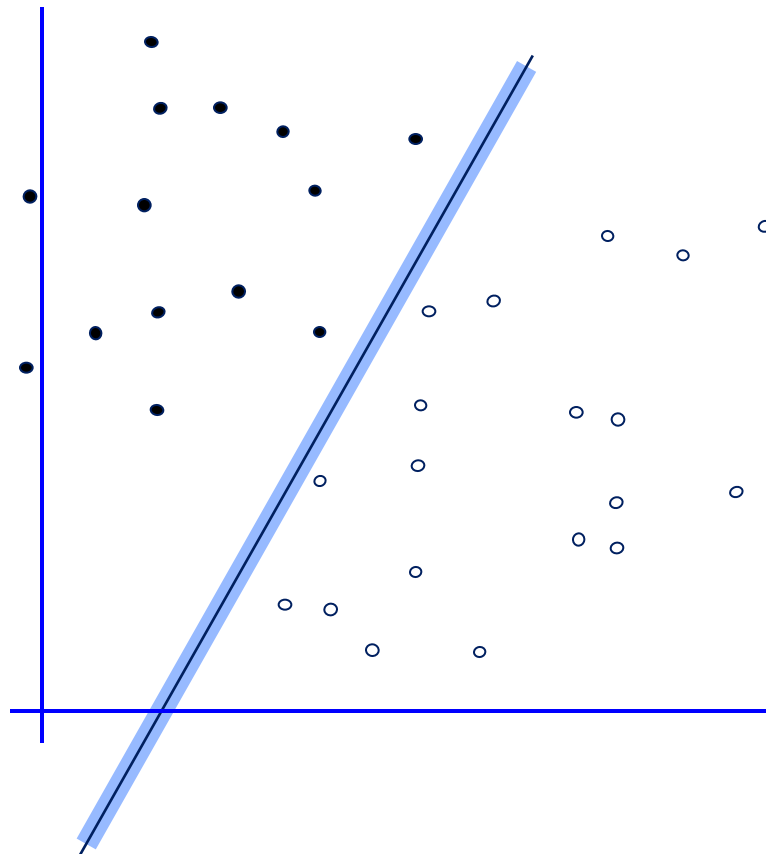
## 15.2.1 Linear Classifiers



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

• : +1

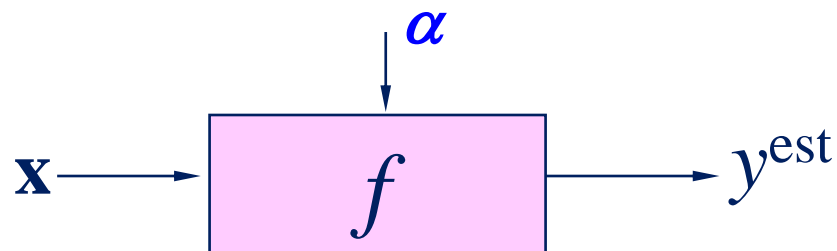
○ : -1



Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

（将线性分类器的间隔定义为在到达数据点之前可以增加的边界宽度）

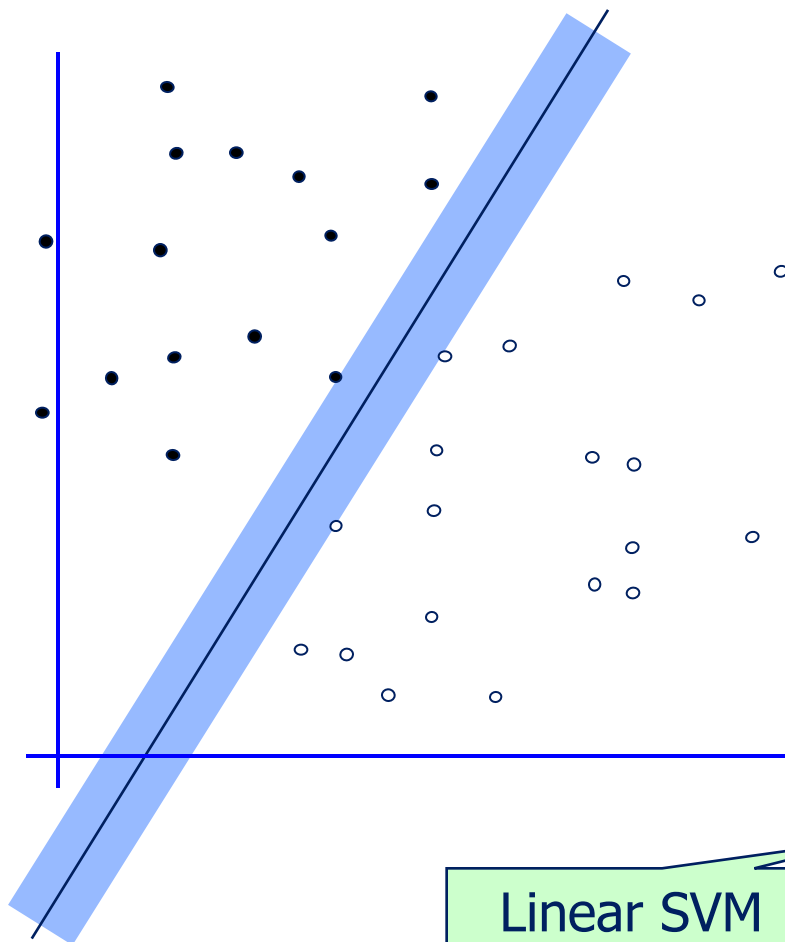
## 15.2.2 Maximum Margin



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

• : +1

○ : -1



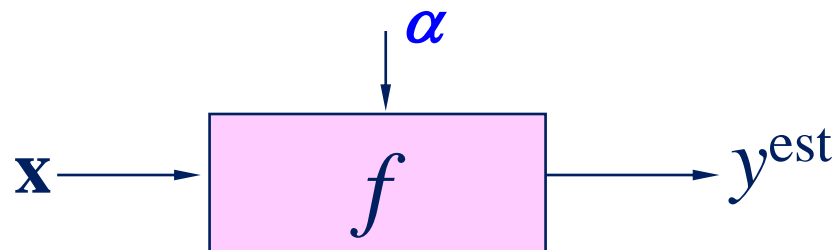
The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM



## 15.2.2 Maximum Margin

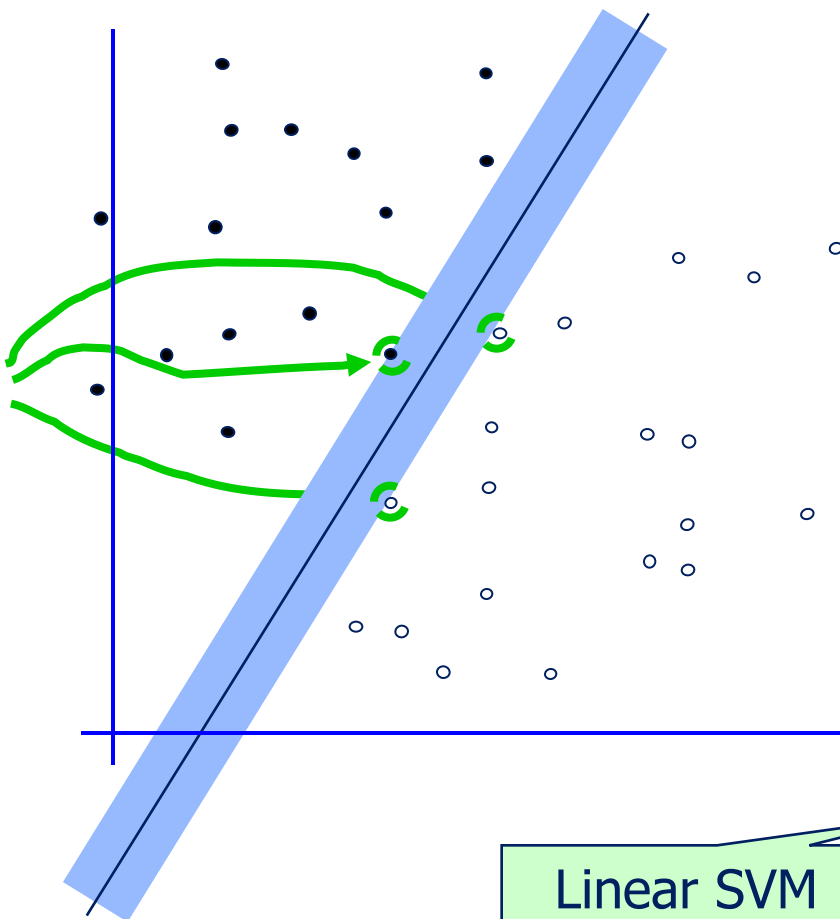


$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

• : +1

○ : -1

Support Vectors  
are those data  
points that the  
margin pushes  
up against

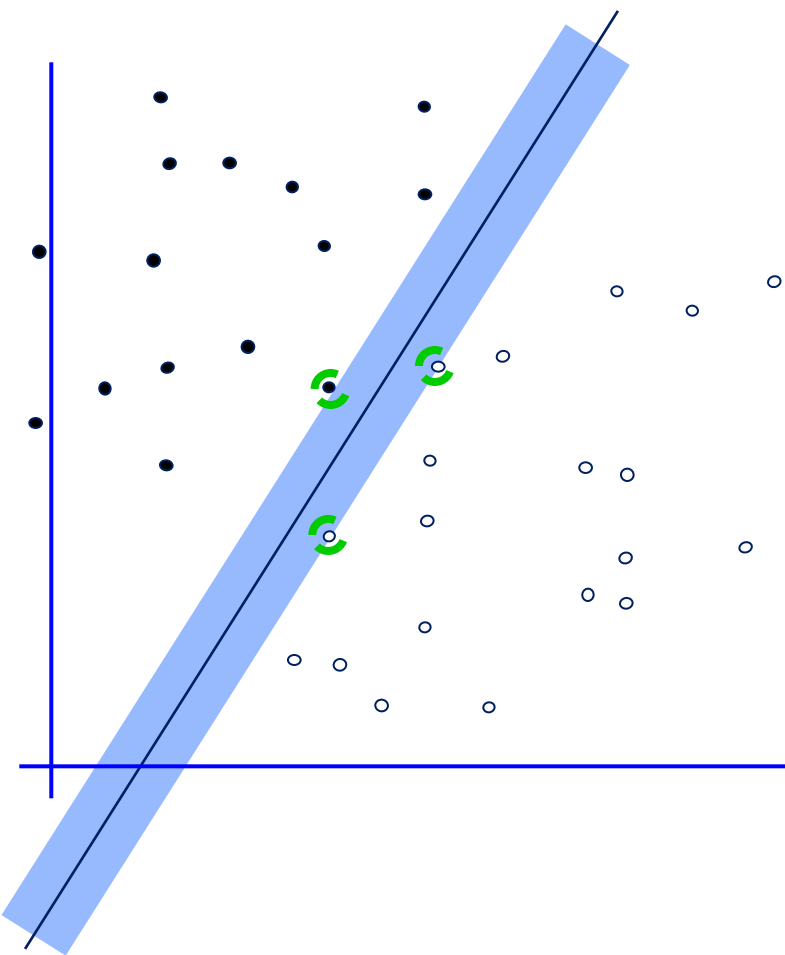


The maximum margin  
linear classifier is the  
linear classifier with  
the maximum margin.

This is the simplest  
kind of SVM (Called an  
LSVM)

Linear SVM

# Why Maximum Margin?



1. **Intuitively this feels safest.**
2. **Empirically it works very well.**
3. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction), this gives us least chance of causing a misclassification. (如果在边界的位置上犯了一个小错误（比如，在垂直方向上受到了震动），那么造成错误分类的可能性就最小)
4. the model is immune to removal of any **non-support-vector data points**. (因此，适合使用LOOCV[留一法交叉验证]来作模型检验)
5. There is some theory (using VC dimension) that this is a good thing.

# 回忆点到平面之间的距离:

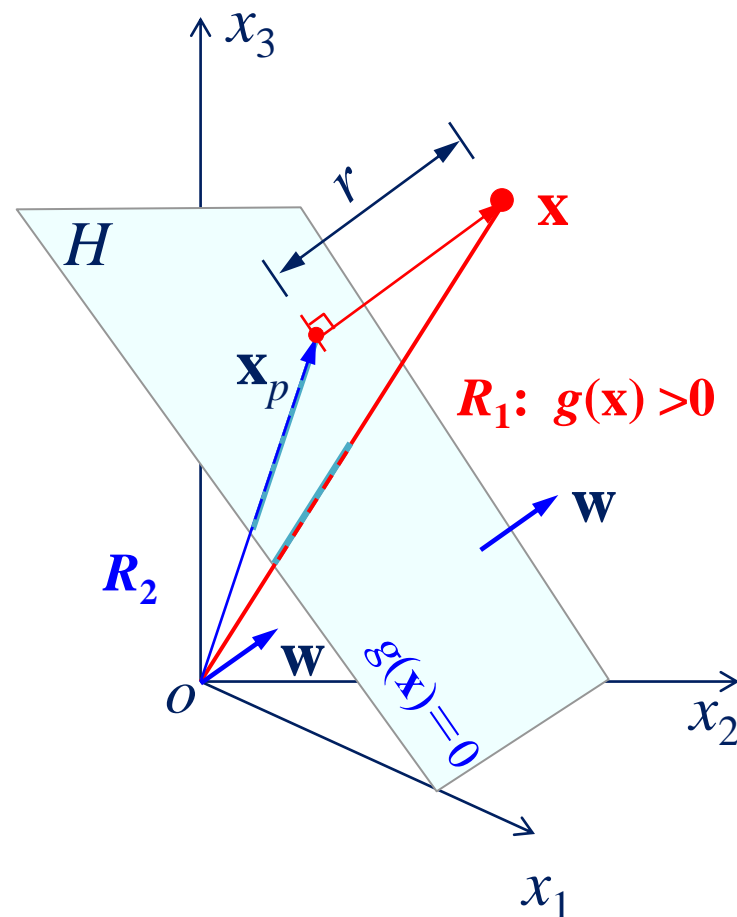
- 两类情形的决策面

- 决策面方程:  $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0$
- 对于任意样本  $\mathbf{x}$ , 将其向决策面内投影, 并写成两个向量之和:

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

其中,  $\mathbf{x}_p$  为  $\mathbf{x}$  在超平面  $H$  上的投影,  $r$  为点  $\mathbf{x}$  到超平面  $H$  的代数距离。如果  $\mathbf{x}$  在超平面正侧, 则  $r > 0$ ; 反之  $r < 0$ 。

即分类超平面



# 回忆点到平面之间的距离：

- 两类情形的决策面

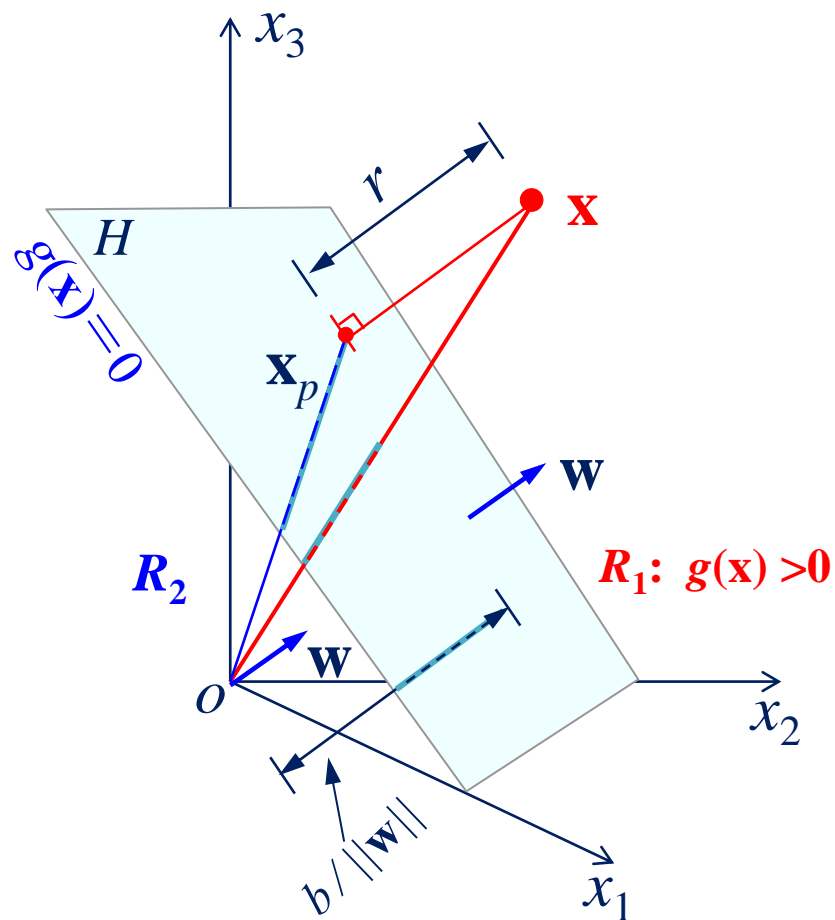
- 注意  $g(\mathbf{x}_p) = 0$ , 于是有：

$$g(\mathbf{x}) = \mathbf{w}^T \left( \mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b$$

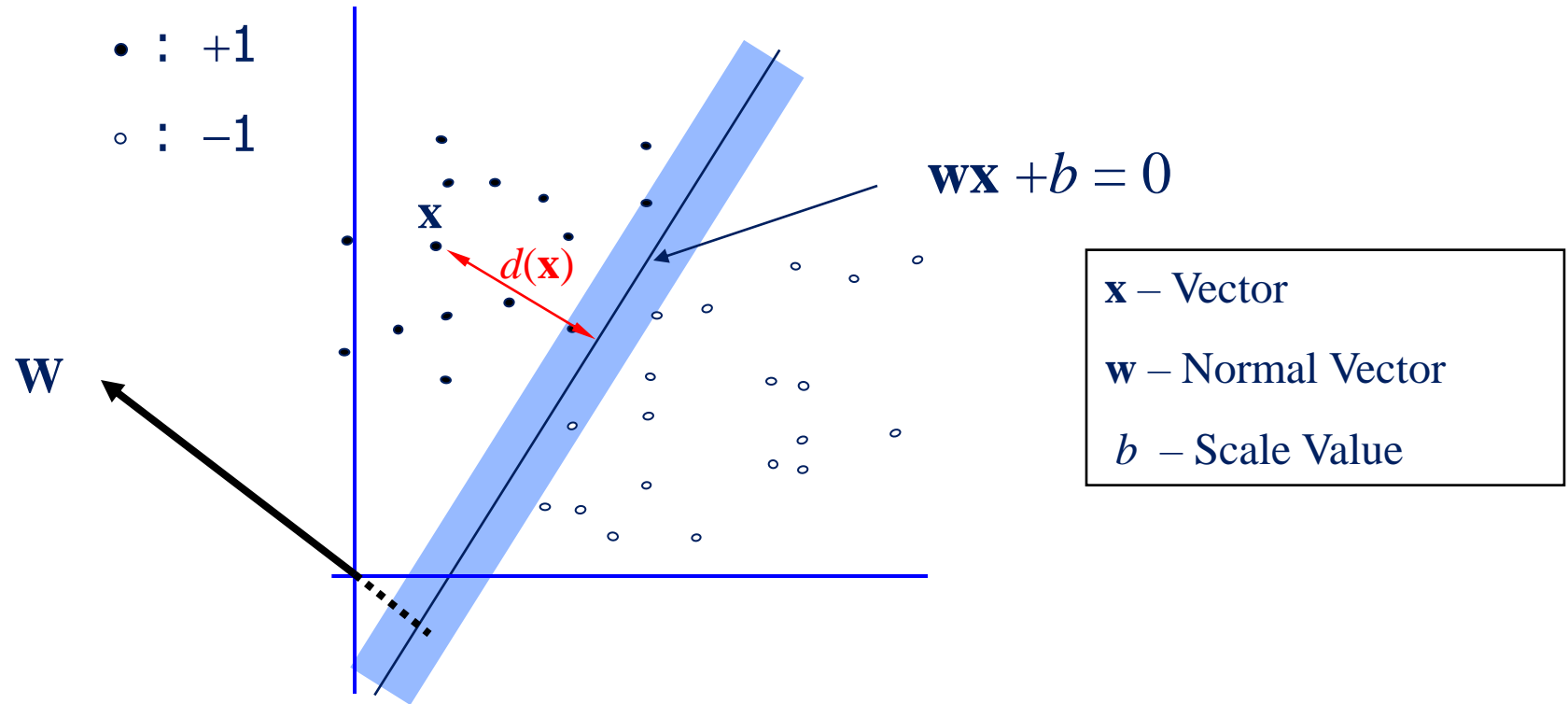
$$= r \|\mathbf{w}\|$$

$$\Rightarrow r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|} \quad (\text{符号距离})$$

此外，可得坐标原点到超平面的距离为： $b / \|\mathbf{w}\|$



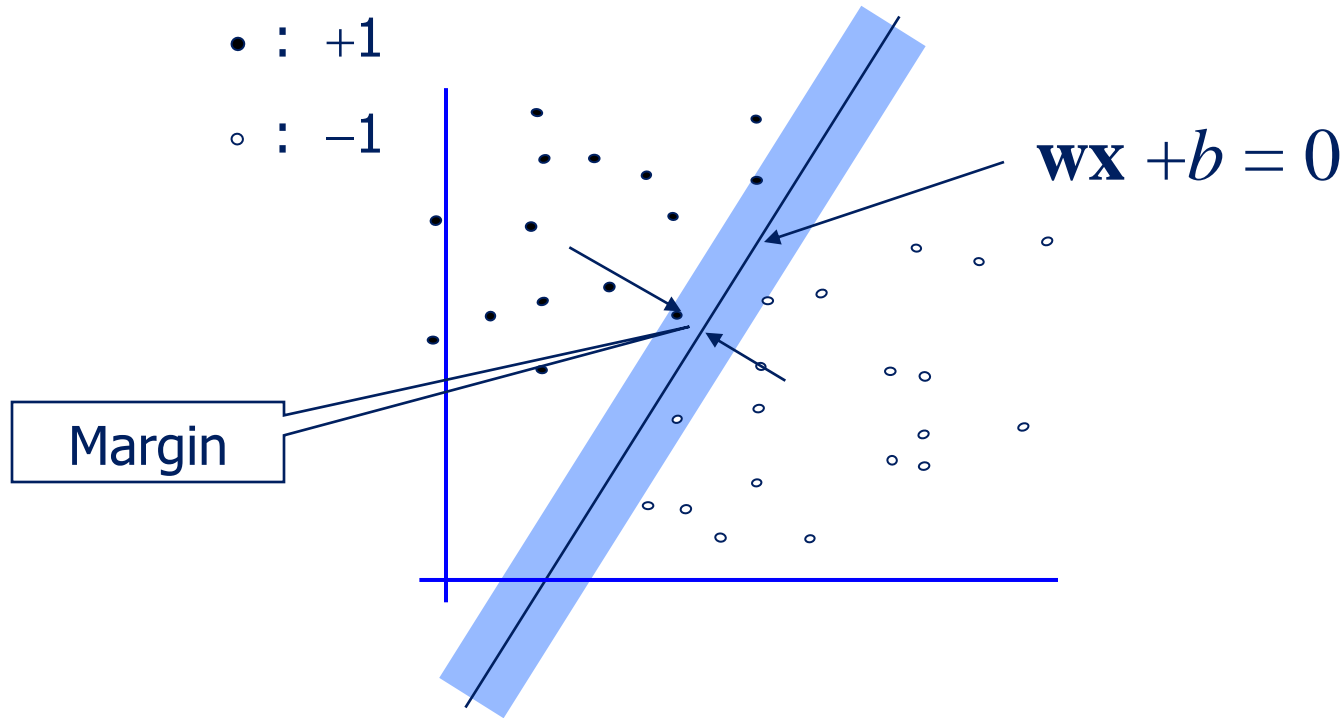
# Estimate Margin (Method 1)



- What is the distance expression for a point  $\mathbf{x}$  to a line  $\mathbf{w}\mathbf{x} + b = 0$ ?

$$d(\mathbf{x}) = \frac{|\mathbf{x} \cdot \mathbf{w} + b|}{\sqrt{\|\mathbf{w}\|_2^2}} = \frac{|\mathbf{x} \cdot \mathbf{w} + b|}{\sqrt{\sum_{i=1}^d w_i^2}}$$

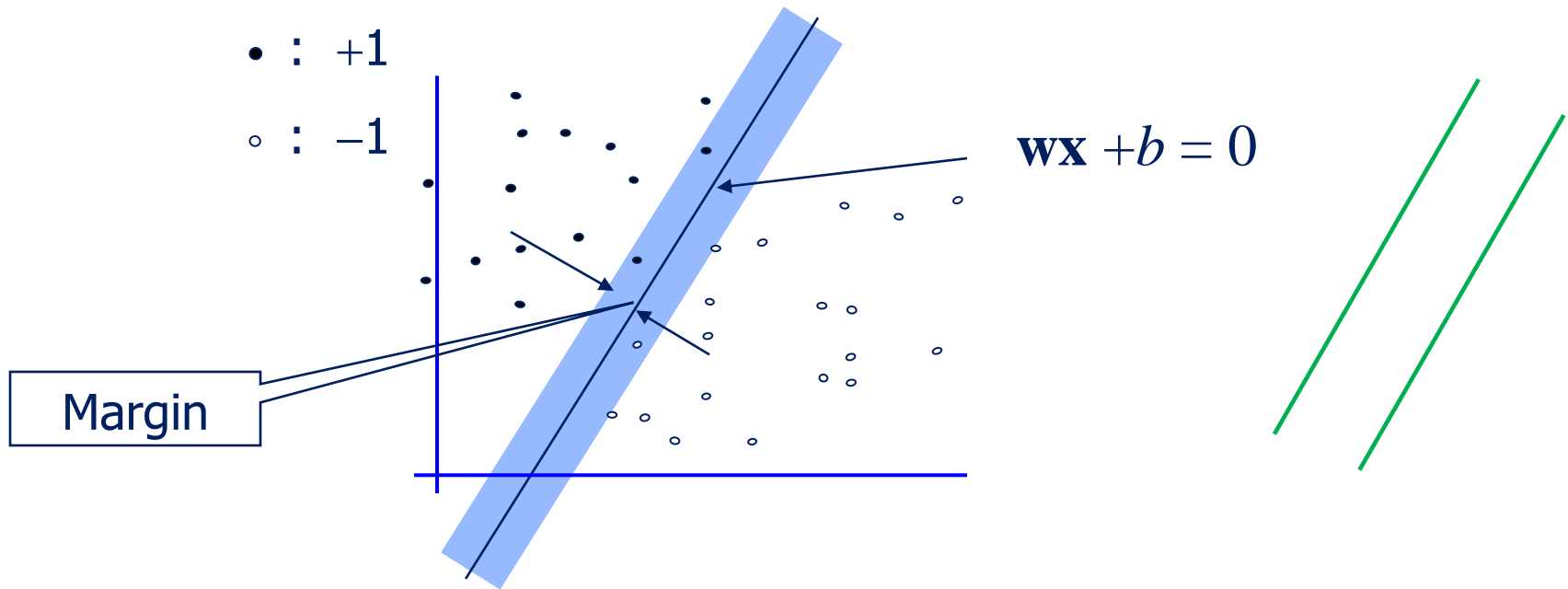
# Estimate Margin (Method 1)



- What is the expression for margin?

$$\text{margin} \equiv \arg \min_{\mathbf{x} \in D} \{d(\mathbf{x})\} = \arg \min_{\mathbf{x} \in D} \frac{|\mathbf{x} \cdot \mathbf{w} + b|}{\sqrt{\sum_{i=1}^d w_i^2}}$$

# Estimate Margin (Method 1)



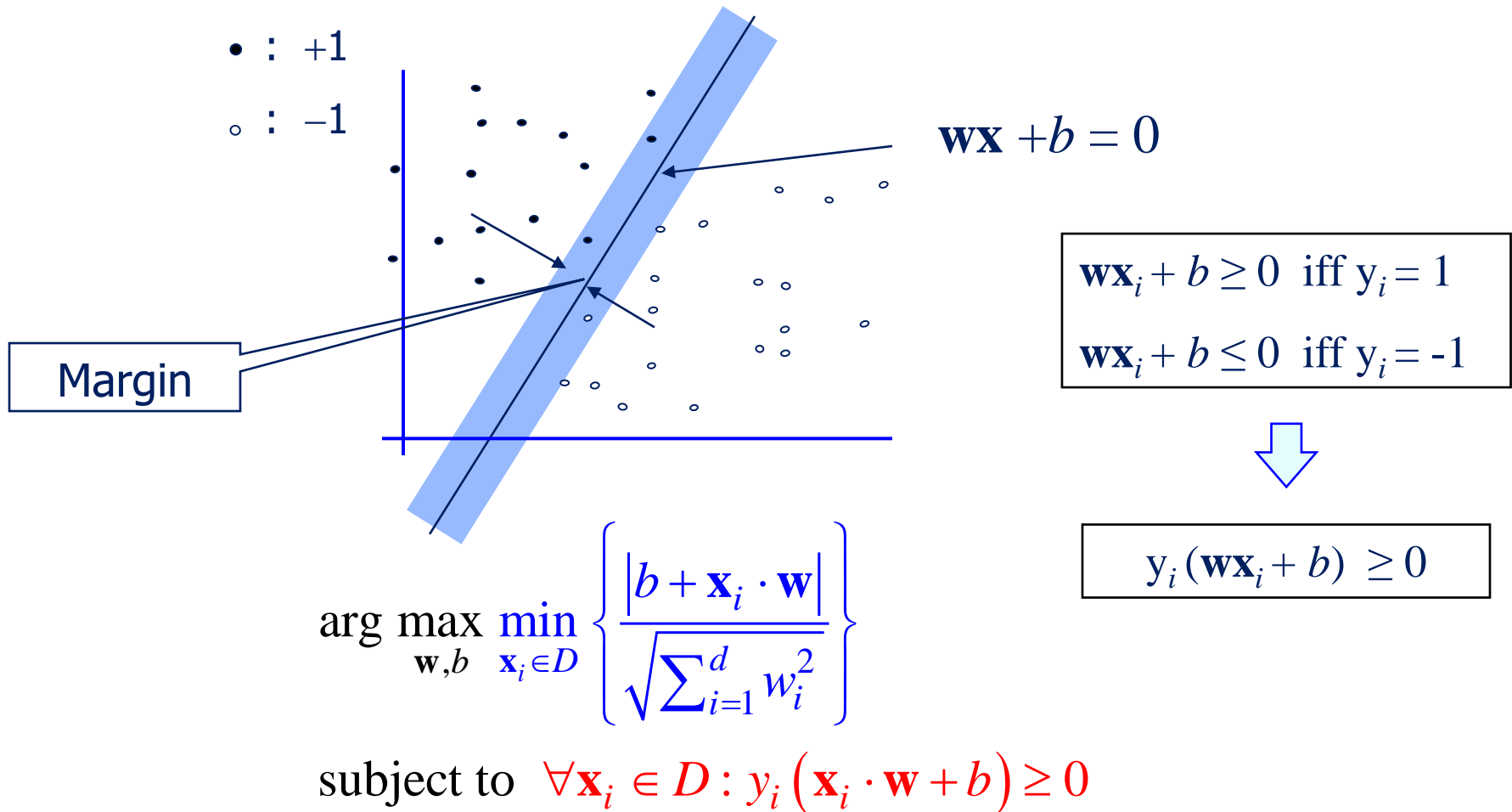
$$\arg \max_{\mathbf{w}, b} \text{margin}(\mathbf{w}, b, D)$$

*Is this all?*

$$= \arg \max_{\mathbf{w}, b} \min_{\mathbf{x}_i \in D} d(\mathbf{x}_i)$$

$$= \arg \max_{\mathbf{w}, b} \min_{\mathbf{x}_i \in D} \frac{|b + \mathbf{x}_i \cdot \mathbf{w}|}{\sqrt{\sum_{i=1}^d w_i^2}}$$

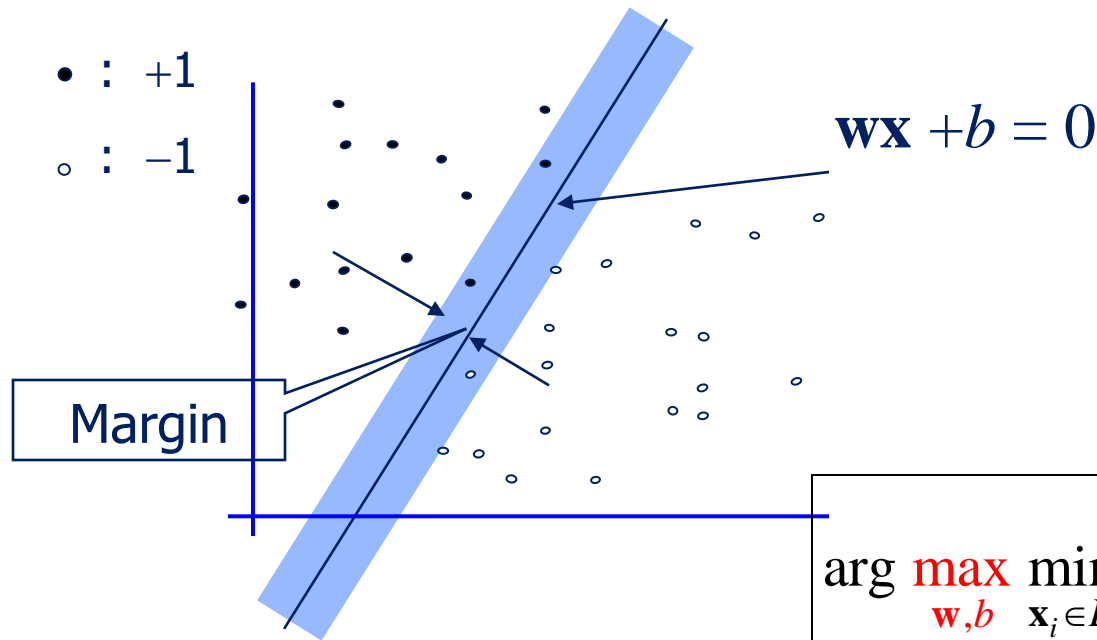
# Estimate Margin (Method 1)



- Min-max problem  $\rightarrow$  game problem



# Estimate Margin (Method 1)



$$\mathbf{w}\mathbf{x}_i + b \geq 0 \text{ iff } y_i = 1$$

$$\mathbf{w}\mathbf{x}_i + b \leq 0 \text{ iff } y_i = -1$$



$$y_i (\mathbf{w}\mathbf{x}_i + b) \geq 0$$

Strategy:

$$\forall \mathbf{x}_i \in D: |b + \mathbf{x}_i \cdot \mathbf{w}| \geq 1$$

$$\arg \max_{\mathbf{w}, b} \min_{\mathbf{x}_i \in D} \frac{|b + \mathbf{x}_i \cdot \mathbf{w}|}{\sqrt{\sum_{i=1}^d w_i^2}}$$

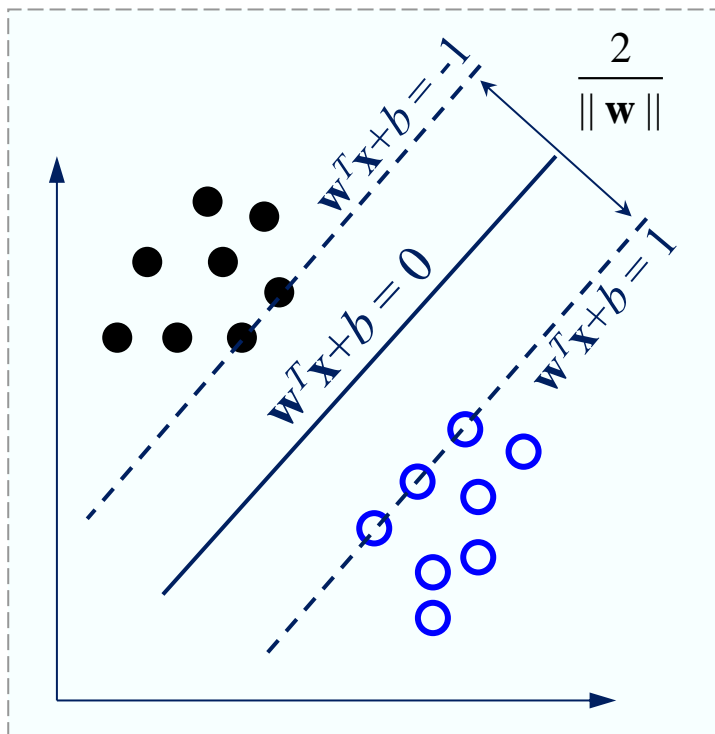
subject to  $\forall \mathbf{x}_i \in D: y_i (\mathbf{x}_i \cdot \mathbf{w} + b) \geq 0$



$$\arg \min_{\mathbf{w}, b} \sum_{i=1}^d w_i^2$$

subject to  $\forall \mathbf{x}_i \in D: y_i (\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$

# Estimate Margin (Method 1)



- ✓ 让负类样本点  $\mathbf{w}^T \mathbf{x} + b \leq -1$
- ✓ 让正类样本点  $\mathbf{w}^T \mathbf{x} + b \geq +1$
- ✓ 注意到负类标签  $y_i = -1$ , 正类标签  $y_i = +1$ , 综合起来, 有:

$$y_i (\mathbf{w}^T \mathbf{x} + b) \geq 1$$

总是可以做到的 (其一: 数值上)

Strategy:

$$\forall \mathbf{x}_i \in D: |b + \mathbf{x}_i \cdot \mathbf{w}| \geq 1$$

$$\begin{aligned} & \arg \max_{\mathbf{w}, b} \min_{\mathbf{x}_i \in D} \frac{|b + \mathbf{x}_i \cdot \mathbf{w}|}{\sqrt{\sum_{i=1}^d w_i^2}} \\ & \text{subject to } \forall \mathbf{x}_i \in D: y_i (\mathbf{x}_i \cdot \mathbf{w} + b) \geq 0 \end{aligned}$$



$$\begin{aligned} & \arg \min_{\mathbf{w}, b} \sum_{i=1}^d w_i^2 \\ & \text{subject to } \forall \mathbf{x}_i \in D: y_i (\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1 \end{aligned}$$

(其二, 考虑类别时的约束统一表示)

# Estimate Margin (Method 1)

- How does it come ? (总结一下)

$$\arg \max_{\mathbf{w}, b} \min_{\mathbf{x}_i \in D} \frac{|b + \mathbf{x}_i \cdot \mathbf{w}|}{\sqrt{\sum_{i=1}^d w_i^2}}$$

subject to  $\forall \mathbf{x}_i \in D : y_i (\mathbf{x}_i \cdot \mathbf{w} + b) \geq 0$

$$\operatorname{argmin}_{\mathbf{w}, b} \sum_{i=1}^d w_i^2$$

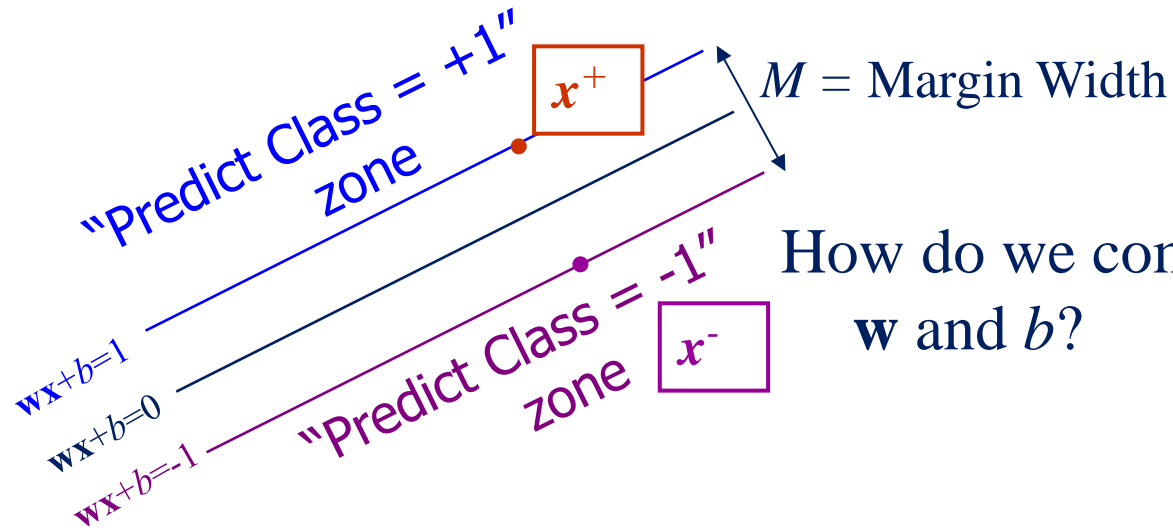
subject to  $\forall \mathbf{x}_i \in D : y_i (\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$

$$\forall \mathbf{x}_i \in D : |b + \mathbf{x}_i \cdot \mathbf{w}| \geq 1$$



$$\arg \min_{\mathbf{x}_i \in D} \frac{|b + \mathbf{x}_i \cdot \mathbf{w}|}{\sqrt{\sum_{i=1}^d w_i^2}} \geq \arg \min_{\mathbf{x}_i \in D} \frac{1}{\sqrt{\sum_{i=1}^d w_i^2}} = \frac{1}{\sqrt{\sum_{i=1}^d w_i^2}}$$

# Estimate Margin (Method 2)



How do we compute  $M$  in terms of  $\mathbf{w}$  and  $b$ ?

- Plus-plane =  $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane =  $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$
- What is the distance between these two planes?

## Estimate Margin (Method 2)

- Margin can also be defined as distance between two parallel lines.

Given 2 parallel lines with equations:

$$ax+by+c_1=0$$

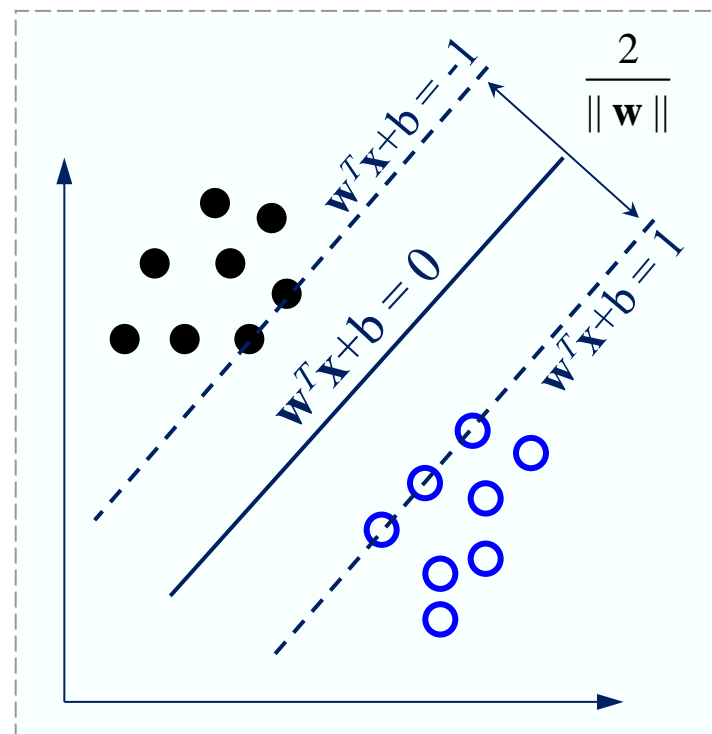
$$ax+by+c_2=0$$

The distance between them is given by

$$d = \frac{|c_2 - c_1|}{\sqrt{a^2 + b^2}}$$

$$\frac{2}{\|\mathbf{w}\|}$$

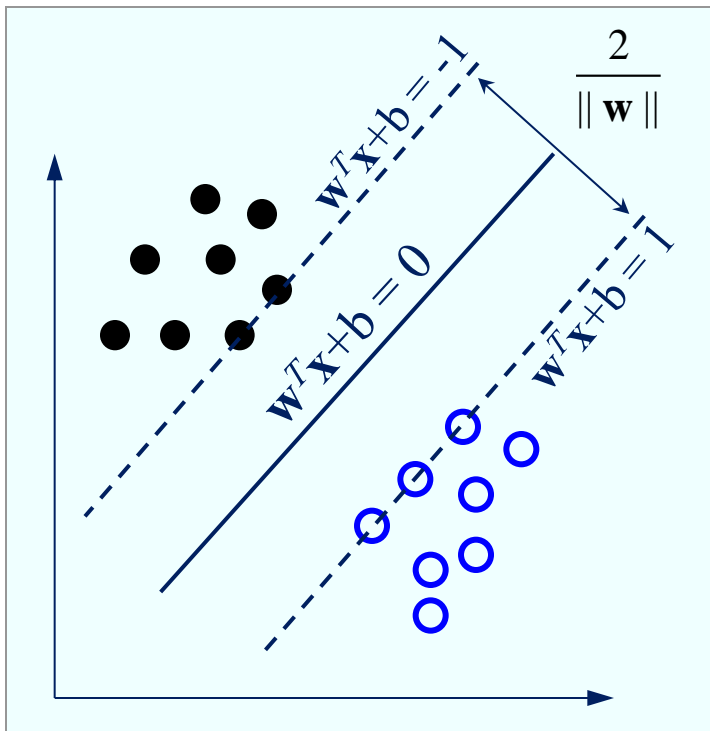
类间数据点间隔最大



## 15.2.3 线性可分支持向量机

- 学习模型

线性可分情形



给定训练集:

$$T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}, y_i \in \{+1, -1\}$$

任务: 估计最大间隔分类超平面

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

分类超平面:  $\mathbf{w}^T \mathbf{x} + b = 0$

分类决策函数:  $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$

## 15.2.3 线性可分支持向量机

定理1：对于线性可分数据集，最大间隔分类面存在且唯一。

存在性：

因为数据集线性可分，根据定义，一定存在 $(\mathbf{w}, b)$ 使得：

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

又因为数据集中既有正样本，又有负样本，所以 $\mathbf{w} \neq 0$ 。

所以存在解，且满足求解最大间隔分类面的最优化问题。

## 15.2.3 线性可分支持向量机

定理1：对于线性可分数据集，**最大间隔分类面存在且唯一。**

**唯一性：**

假设最优化问题存在两个最优解 $(w_1, b_1)$ 和 $(w_2, b_2)$ 。

$$\|w_1\| = \|w_2\| = c$$

$$\text{令： } w = \frac{1}{2}(w_1 + w_2), b = \frac{1}{2}(b_1 + b_2)$$

$$y_i(w^T x_i + b_i) = \frac{1}{2} y_i(w_1^T x_i + b_1) + \frac{1}{2} y_i(w_2^T x_i + b_2) \geq 1$$

$$\left. \begin{array}{l} \|w\| \geq c \\ \|w\| = \left\| \frac{1}{2} w_1 + \frac{1}{2} w_2 \right\| \leq \frac{1}{2} \|w_1\| + \frac{1}{2} \|w_2\| = c \end{array} \right\} \|w\| = c = \frac{1}{2} \|w_1\| + \frac{1}{2} \|w_2\|$$



## 15.2.3 线性可分支持向量机

定理1：对于线性可分数据集，最大间隔分类面存在且唯一。

唯一性（续）：

$$\left. \begin{aligned} w &= \frac{1}{2}(w_1 + w_2) \\ \|w\| &= \frac{1}{2}\|w_1\| + \frac{1}{2}\|w_2\| \end{aligned} \right\} w_1 = w_2 \quad \text{or} \quad w_1 = -w_2$$

若  $w_1 = -w_2$ ，则  $w = 0$

$$y_i(wx_i + b_i) \geq 1, i = 1, 2, \dots, n$$

矛盾！

$$w_1 = w_2 = w^*$$

## 15.2.3 线性可分支持向量机

定理1：对于线性可分数据集，**最大间隔分类面存在且唯一**。

**唯一性（续）：**

设 $x_{11}$ 和 $x_{12}$ 是正样本中分别对应于 $(w^*, b_1)$ 和 $(w^*, b_2)$ 使约束条件等号成立的点；  
 $x_{21}$ 和 $x_{22}$ 是负样本中分别对应于 $(w^*, b_1)$ 和 $(w^*, b_2)$ 使约束条件等号成立的点。

$$\begin{array}{l} \Rightarrow \left. \begin{array}{l} w^{*T} x_{11} + b_1 = 1 \\ w^{*T} x_{21} + b_1 = -1 \\ w^{*T} x_{12} + b_2 = 1 \\ w^{*T} x_{22} + b_2 = -1 \end{array} \right\} \begin{array}{l} b_1 = -\frac{1}{2}(w^{*T} x_{11} + w^{*T} x_{21}) \\ b_2 = -\frac{1}{2}(w^{*T} x_{12} + w^{*T} x_{22}) \end{array} \end{array}$$

## 15.2.3 线性可分支持向量机

定理1：对于线性可分数据集，最大间隔分类面存在且唯一。

唯一性（续）：

$$b_1 - b_2 = -\frac{1}{2} \underbrace{(w^{*T}(x_{11} - x_{12}))}_{=0} + \underbrace{w^{*T}(x_{21} - x_{22}))}_{=0}$$

$$\left. \begin{array}{l} w^{*T}x_{11} + b_1 = 1 \\ w^{*T}x_{12} + b_1 \geq 1 \end{array} \right\} w^{*T}(x_{11} - x_{12}) \leq 0$$

$$\left. \begin{array}{l} w^{*T}x_{11} + b_2 \geq 1 \\ w^{*T}x_{12} + b_2 = 1 \end{array} \right\} w^{*T}(x_{11} - x_{12}) \geq 0$$

$$w^{*T}(x_{11} - x_{12}) = 0$$

## 15.2.4 线性可分支持向量机求解

### 复习: Quadratic Programming

Find  $\arg \max_{\mathbf{u}} \quad c + \mathbf{d}^T \mathbf{u} + \frac{1}{2} \mathbf{u}^T \mathbf{R} \mathbf{u}$  ← Quadratic criterion

Subject to

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \dots + a_{1m}u_m &\leq b_1 \\ a_{21}u_1 + a_{22}u_2 + \dots + a_{2m}u_m &\leq b_2 \\ &\vdots \\ a_{n1}u_1 + a_{n2}u_2 + \dots + a_{nm}u_m &\leq b_n \end{aligned}$$

} n additional linear inequality constraints

$$\begin{aligned} a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \dots + a_{(n+1)m}u_m &= b_{(n+1)} \\ a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \dots + a_{(n+2)m}u_m &= b_{(n+2)} \\ &\vdots \\ a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \dots + a_{(n+e)m}u_m &= b_{(n+e)} \end{aligned}$$

} e additional linear equality constraints

## 15.2.4 线性可分支持向量机求解

### Quadratic Programming for Linear SVM:

$$\{\vec{w}^*, b^*\} = \min_{\vec{w}, b} \sum_i w_i^2$$

subject to  $y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1$  for all training data  $(\vec{x}_i, y_i)$



$$\{\vec{w}^*, b^*\} = \operatorname{argmax}_{\vec{w}, b} \left\{ 0 + \vec{0} \cdot \vec{w} - \vec{w}^T \mathbf{I}_n \vec{w} \right\}$$

$$\left. \begin{array}{l} y_1 (\vec{w} \cdot \vec{x}_1 + b) \geq 1 \\ y_2 (\vec{w} \cdot \vec{x}_2 + b) \geq 1 \\ \dots \\ y_N (\vec{w} \cdot \vec{x}_N + b) \geq 1 \end{array} \right\} \text{inequality constraints}$$

# 15.3 Soft-Margin SVM

## 15.3.1 Noisy Data

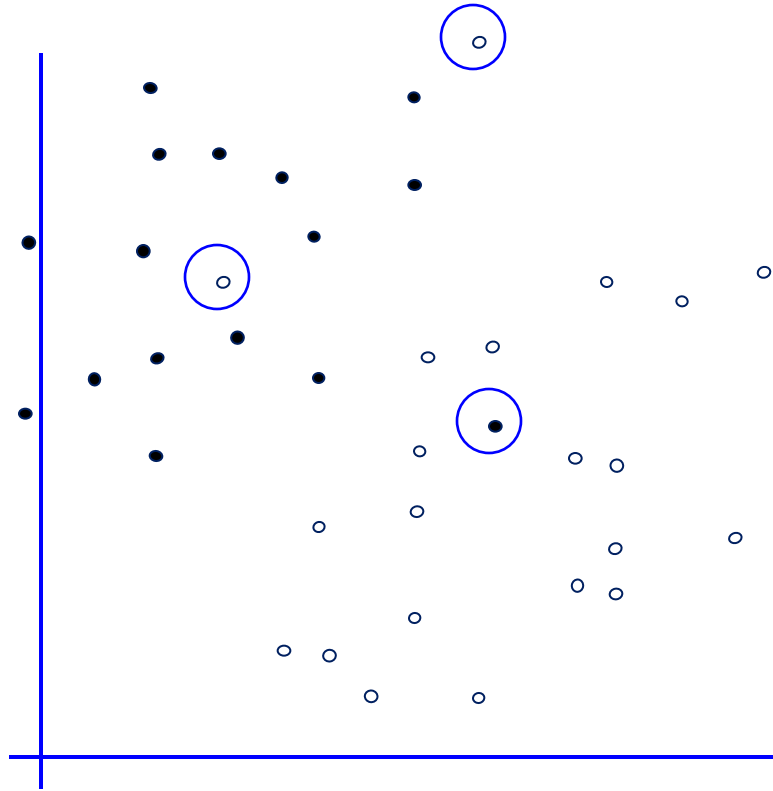
Not linear separable

This is going to be a problem!

What should we do?

• : +1

○ : -1



Idea 1:

Find minimum  $\mathbf{w} \cdot \mathbf{w}$ , while minimizing number of training set errors. (但这是一个NP难问题)

## 15.3.1 Noisy Data

This is going to be a problem!

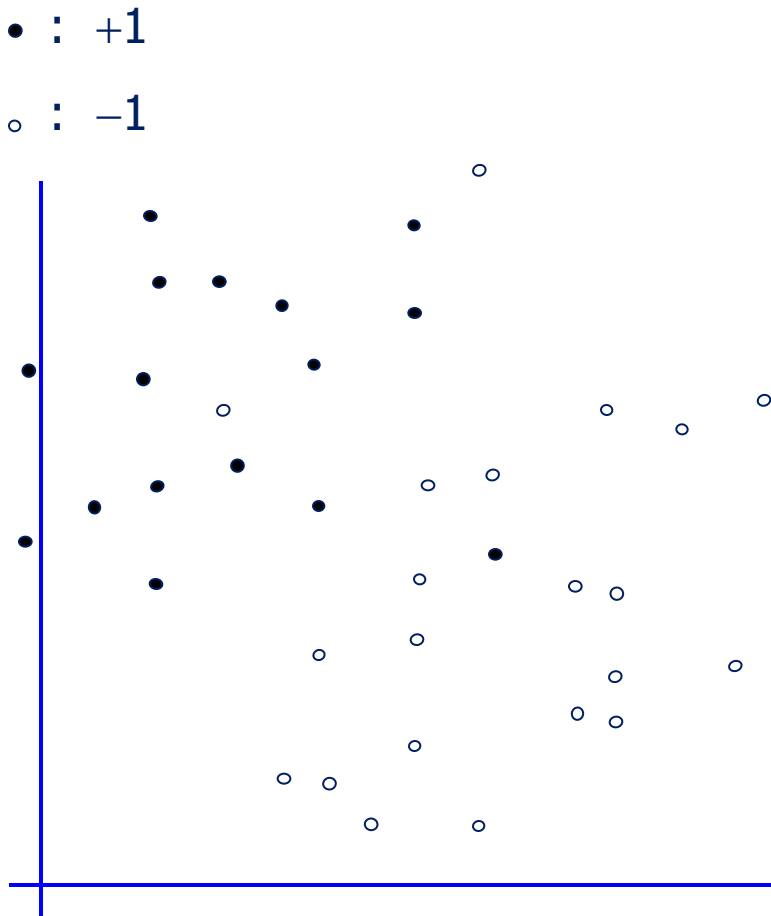
What should we do?

Idea 1.1:

Minimize

$$\mathbf{w} \cdot \mathbf{w} + C (\#train\ errors)$$

Tradeoff parameter



There's a serious practical problem that's about to make us reject this approach. Can you guess what it is?



## 15.3.1 Noisy Data

This is going to be a problem!

What should we do?

Idea 1.1:

Minimize

$$\mathbf{w} \cdot \mathbf{w} + C (\#train\ errors)$$

Tradeoff parameter

Can't be expressed as a Quadratic Programming problem.

Solving it may be too slow.

(Also, doesn't distinguish between disastrous errors and near misses)

us practical problem  
make us reject this  
you guess what it is?

So... any  
other  
ideas?

## 15.3.1 Noisy Data

This is going to be a problem!

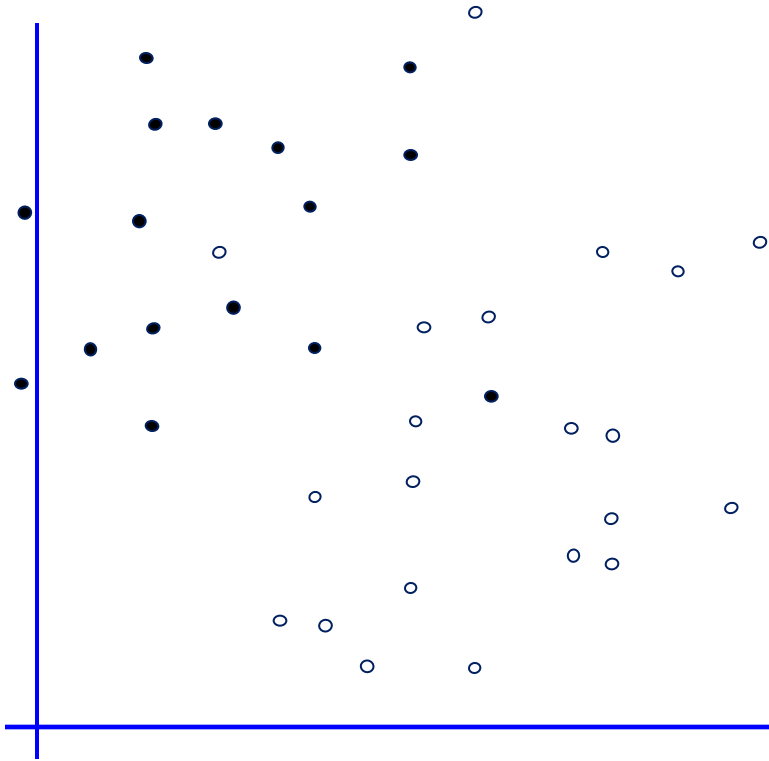
What should we do?

Idea 2.0:

Minimize

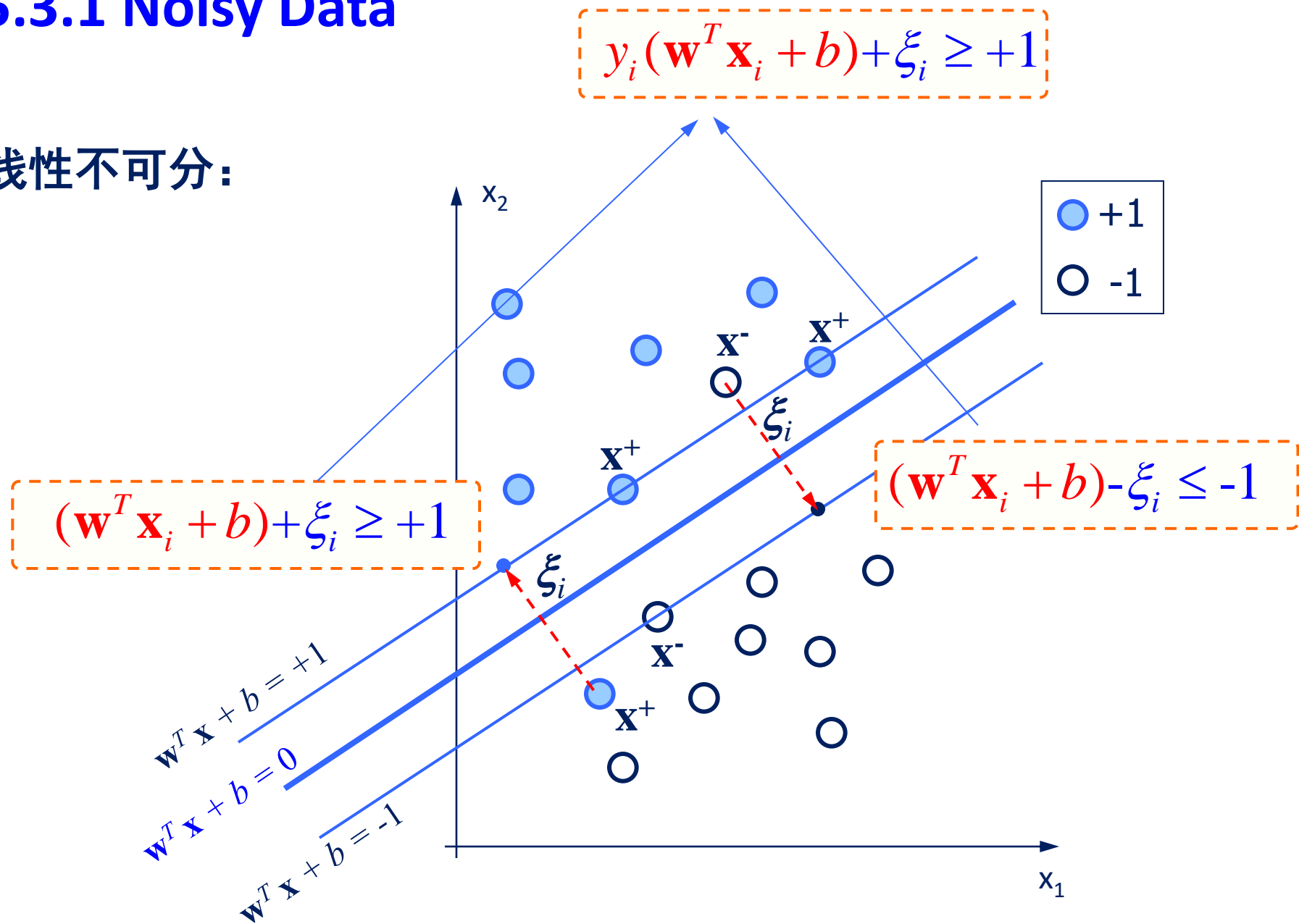
$\mathbf{w} \cdot \mathbf{w} + C$  (*distance of error points to their correct place*)

- denotes +1
- denotes -1



## 15.3.1 Noisy Data

线性不可分:



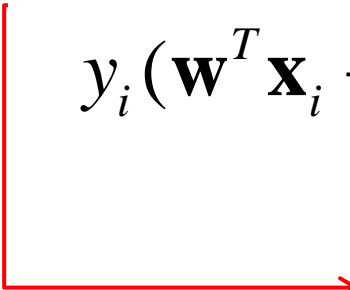
## 15.3.2 硬间隔与软间隔

线性可分情况：

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \longleftarrow \text{硬间隔}$$

对于线性不可分情况，对约束条件引入松弛变量，允许有少量样本落在两类分类间隔中间：

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \longleftarrow \text{软间隔}$$


$$\min \sum_{i=1}^n \xi_i$$

- Any problem with the above formulism?

What happens when  $\xi_i < 0$  ? (我们不希望)

# 15.3.3 线性不可分-学习模型（支持向量机）

Describe the Theory

Describe the Mistake

体现了表达能力

体现了经验风险

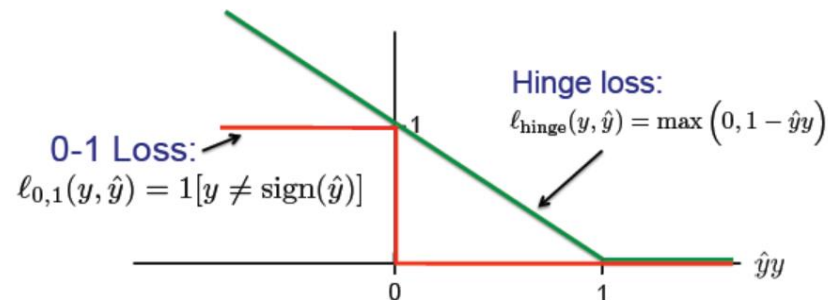
$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \\ & i = 1, 2, \dots, n \end{aligned}$$

$C$ : tradeoff parameter between error and margin; chosen by the user; large  $C$  means a higher penalty to errors

目标函数第一项表示使margin尽量大，第二项表示使误差分类点的个数尽量小。

# 合页损失

- 软间隔最大化：
  - More robust for outliers



Hinge loss upper bounds 0/1 loss!

It is the tightest convex upper bound on the 0/1 loss

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

$\longleftrightarrow \min_{\mathbf{w}, b}$

$$\sum_{i=1}^n [1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)]_+ + \lambda \|\mathbf{w}\|^2$$

$$[z]_+ = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases}$$

合页损失函数

**Hinge loss function!**

(证明参考“李航：统计学习方法，清华大学出版社”第7章第113~114页， 本课程不要求)

## Conclusion so far

- Which linear classifier  $\rightarrow$  large margin
- How to calculate margin  $\rightarrow \frac{2}{\|\mathbf{w}\|}$
- Hard margin SVM  $\rightarrow$  QP
- Not linear separable  $\rightarrow$  soft-margin SVM
- Still QP
- L2 norm regularization + hinge loss

# 15.4 Dual Problem



## 15.4.1 Lagrange Multipliers (复习)

*Minimize*  $f(x)$

*subject to* 
$$\begin{cases} a(x) \geq 0 \\ b(x) \leq 0 \\ c(x) = 0 \end{cases}$$

$$L(x, \alpha) = f(x) - \alpha_1 a(x) - \alpha_2 b(x) - \alpha_3 c(x)$$

$$\begin{cases} \alpha_1 \geq 0 \\ \alpha_2 \leq 0 \\ \alpha_3 \text{ is unconstrained} \end{cases}$$

We can recover the primal problem by maximizing the Lagrangian with respect to the Lagrange multipliers:

$$\max_{\alpha} L(x, \alpha) = \begin{cases} f(x), & \text{if } \begin{cases} a(x) \geq 0 \\ b(x) \leq 0 \\ c(x) = 0 \end{cases} \\ +\infty, & \text{otherwise} \end{cases}$$

So, the Primal problem can be changed into Dual problem

$$\min_x \boxed{\max_{\alpha} L(x, \alpha)} = \max_{\alpha} \boxed{\min_x L(x, \alpha)}$$

$Primal(x) \qquad \qquad \qquad Dual(\alpha)$

## 15.4.1 Lagrange Multipliers (复习)

- For a local minimum

$$\left\{ \begin{array}{ll} \text{Stationarity} & \nabla f(x^*) - \alpha_1 \nabla a(x^*) - \alpha_2 \nabla b(x^*) - \alpha_3 \nabla c(x^*) = 0 \\ \text{Primal feasibility} & \begin{cases} a(x^*) \geq 0 \\ b(x^*) \leq 0 \\ c(x^*) = 0 \end{cases} \\ \text{Dual feasibility} & \begin{cases} \alpha_1 \geq 0 \\ \alpha_2 \leq 0 \\ \alpha_3 \text{ is unconstrained} \end{cases} \\ \text{Complementary slackness} & \begin{cases} \alpha_1 a(x^*) = 0 \\ \alpha_2 b(x^*) = 0 \\ \alpha_3 c(x^*) = 0 \end{cases} \end{array} \right.$$

• 复习KKT条件 (数学知识点, 不要求):

KKT条件

原始问题

$$\begin{aligned} \min_{\mathbf{x} \in R^d} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & c_i(\mathbf{x}) \leq 0, \\ & i = 1, 2, \dots, k \\ & h_j(\mathbf{x}) = 0, \\ & j = 1, 2, \dots, l \end{aligned}$$

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$$

$$\nabla_{\boldsymbol{\alpha}} L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$$

$$\nabla_{\boldsymbol{\beta}} L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$$

$$\alpha_i c_i(\mathbf{x}) = 0, \quad i = 1, 2, \dots, k$$

$$c_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \dots, k$$

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

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \dots, l$$

$$L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{x}) + \sum_{i=1}^k \alpha_i c_i(\mathbf{x}) + \sum_{j=1}^l \beta_j h_j(\mathbf{x})$$

广义拉格朗日函数

## 15.4.2 支持向量机(对偶)

### • 对偶算法 (线性可分情形)

- ✓ 在约束最优化问题中, 常利用拉格朗日对偶性将原始问题转化为对偶问题进行求解
- ✓ 对偶算法往往容易求解
- ✓ 对偶算法可以推广到核学习

$$\min_{\mathbf{x} \in R^d} f(\mathbf{x})$$

$$s.t. \quad c_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \dots, k$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \dots, l$$

广义拉格朗日函数

$$L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{x}) + \sum_{i=1}^k \alpha_i c_i(\mathbf{x}) + \sum_{j=1}^l \beta_j h_j(\mathbf{x})$$

数学知识点

(拉格朗日对偶性)  $\Rightarrow$

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

$$s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, \\ i = 1, 2, \dots, n$$

原始问题

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

$$s.t. \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, n$$

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

对偶问题

证明见: 李航: 统计学习方法, 清华大学出版社, 2012 (第7章) (本课程不要求)

## 15.4.2 支持向量机(对偶)

- 对偶问题求解 (线性可分)

– (1) 求  $\min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha})$

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = \mathbf{0} \Rightarrow \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

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



$$\min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^n \alpha_i$$

## 15.4.2 支持向量机 (对偶)

– (2) 求对偶问题，即求  $\min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha})$  对  $\boldsymbol{\alpha}$  的极大

$$\max_{\boldsymbol{\alpha}} \quad -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^n \alpha_i$$

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

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

- ✓ This is a convex quadratic programming (QP) problem
- ✓ Global maximum of  $\alpha_i$  can always be found
  - well established tools for solving this optimization problem

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^n \alpha_i$$

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

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



## 15.4.2 支持向量机(对偶)

- 定理2 (线性可分)

— 设  $\mathbf{a}^* = [\alpha_1^*, \alpha_1^*, \dots, \alpha_n^*]^T \in R^n$  是对偶问题的解, 则至少存在一个下标  $j$ , 使得  $\alpha_j^* > 0$ , 可按下式求得原始问题的最优解:

$$\mathbf{w}^* = \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i, \quad b^* = y_j - \sum_{i=1}^n \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\mathbf{w}^* \cdot \mathbf{x} + b^* = 0$$

- 分类超平面:

- 分类决策函数:  $f(\mathbf{x}) = \text{sign}(\mathbf{w}^* \cdot \mathbf{x} + b^*) = \text{sign}\left(\sum_{i=1}^n \alpha_i^* y_i (\mathbf{x} \cdot \mathbf{x}_i) + b^*\right)$

结论: 对线性可分情形 最优解  $b^*$  是唯一的。

李航: 统计学习方法, 清华大学出版社, 2012 (第7章)

• 复习KKT条件 (数学知识点, 不要求):

KKT条件

原始问题

$$\begin{aligned} \min_{\mathbf{x} \in R^d} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & c_i(\mathbf{x}) \leq 0, \\ & i = 1, 2, \dots, k \\ & h_j(\mathbf{x}) = 0, \\ & j = 1, 2, \dots, l \end{aligned}$$

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$$

$$\nabla_{\boldsymbol{\alpha}} L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$$

$$\nabla_{\boldsymbol{\beta}} L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 0$$

$$\alpha_i c_i(\mathbf{x}) = 0, \quad i = 1, 2, \dots, k$$

$$c_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \dots, k$$

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

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \dots, l$$

$$L(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{x}) + \sum_{i=1}^k \alpha_i c_i(\mathbf{x}) + \sum_{j=1}^l \beta_j h_j(\mathbf{x})$$

广义拉格朗日函数



## 15.4.2 支持向量机(对偶)

- KKT条件:

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

$$\alpha_i^* \geq 0$$

$$y_i(\mathbf{w}^{*T} \mathbf{x}_i + b^*) - 1 \geq 0$$

$$\alpha_i^* (y_i(\mathbf{w}^{*T} \mathbf{x}_i + b^*) - 1) = 0$$

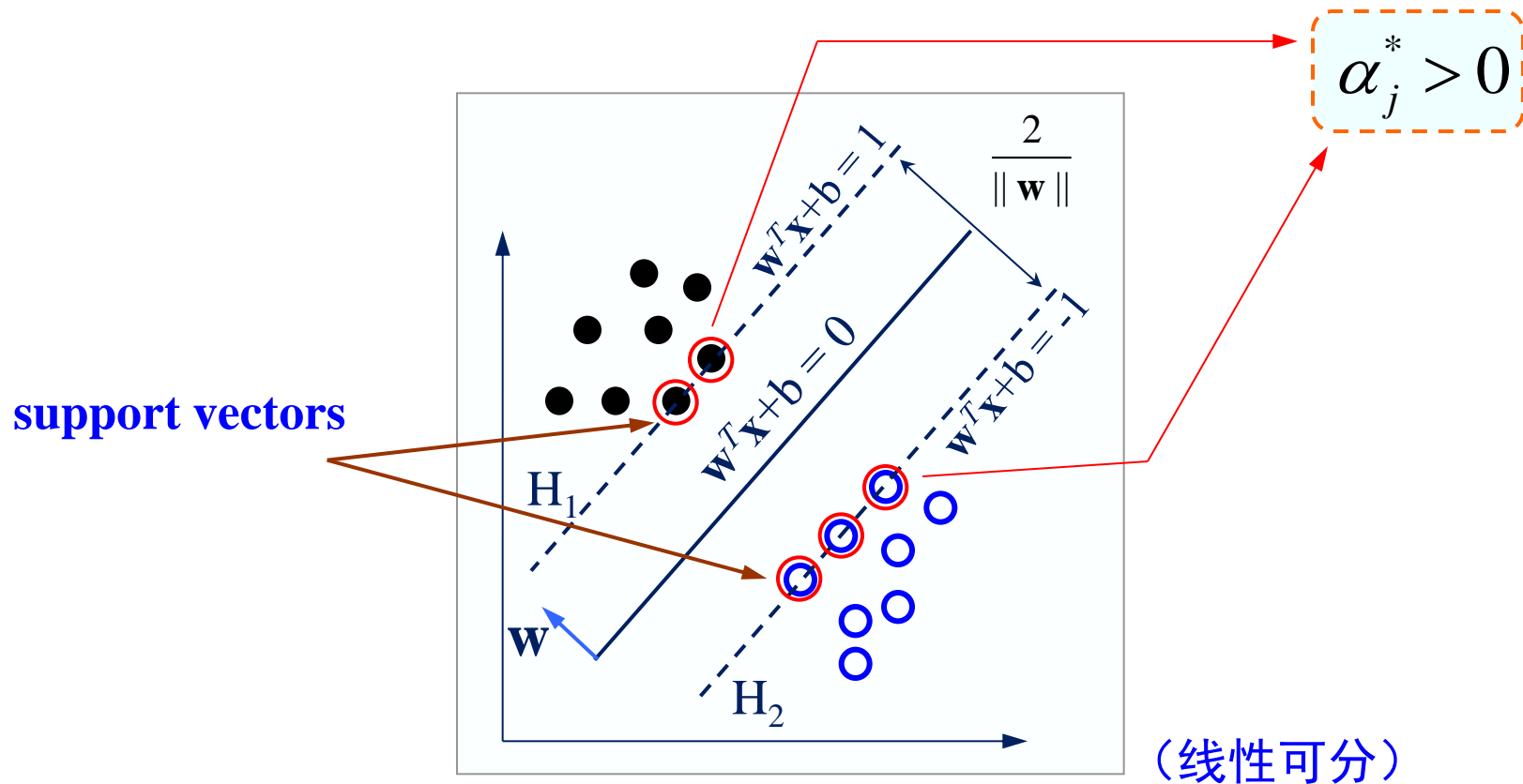
KKT条件!

$$\alpha_i^* = 0, \quad y_i(\mathbf{w}^{*T} \mathbf{x}_i + b^*) \geq 1$$

$$\alpha_i^* > 0, \quad y_i(\mathbf{w}^{*T} \mathbf{x}_i + b^*) = 1$$

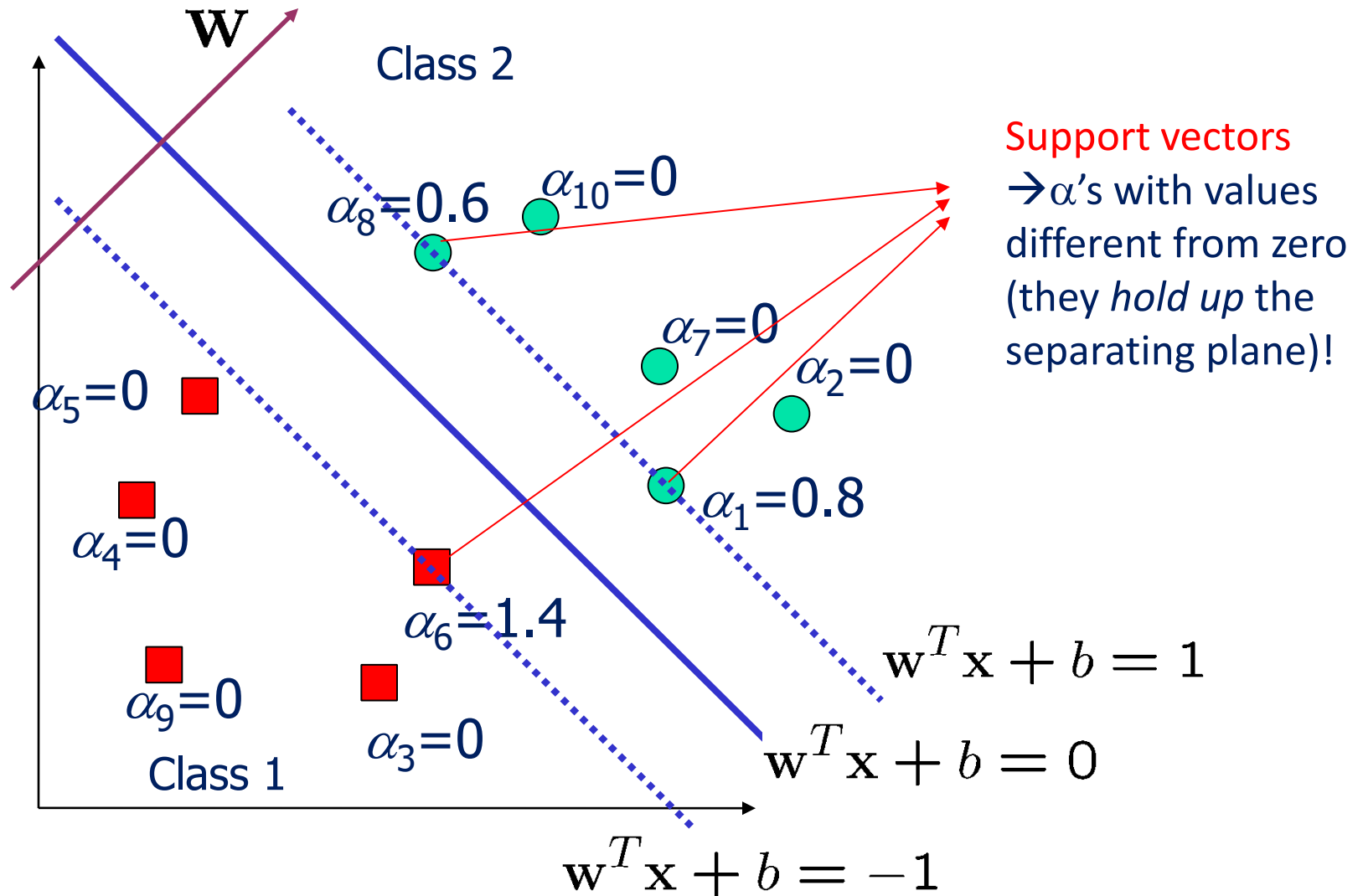
支持向量

✓ 使等式成立的点为**支持向量**:  $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$



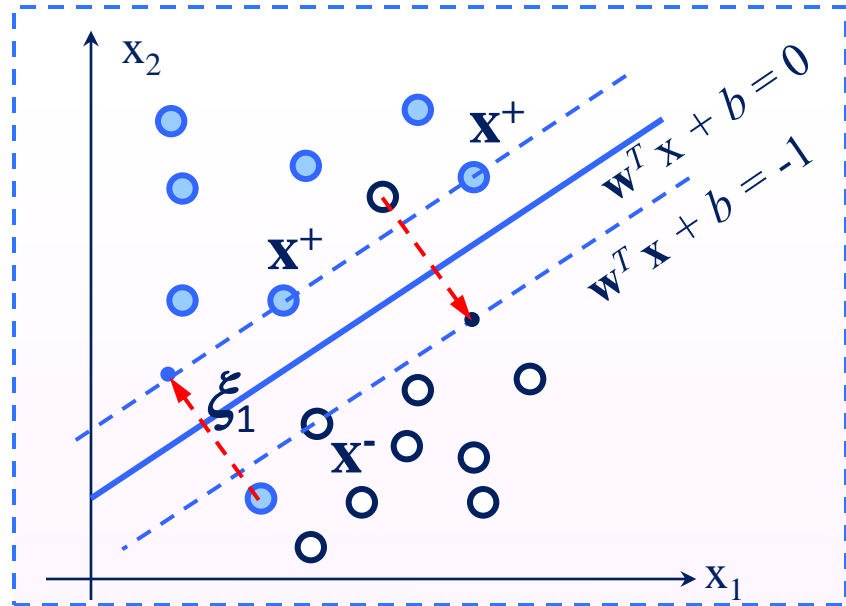
所有样本中，“支持向量”到分类面的几何距离最小。

# Support Vectors (Hard-Margin Case)



软间隔最大化（线性不可分）：

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \\ & i = 1, 2, \dots, n \end{aligned} \quad \text{原始问题}$$



↓ (广义拉格朗日函数)

↓

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

↓

拉格朗日对偶

$$\max_{\substack{\alpha \geq 0 \\ w, b, \xi \\ \mu \geq 0}} \min L(w, b, \xi, \alpha, \mu)$$

$$\min_{\alpha} \quad \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^n \alpha_i$$

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

对偶问题

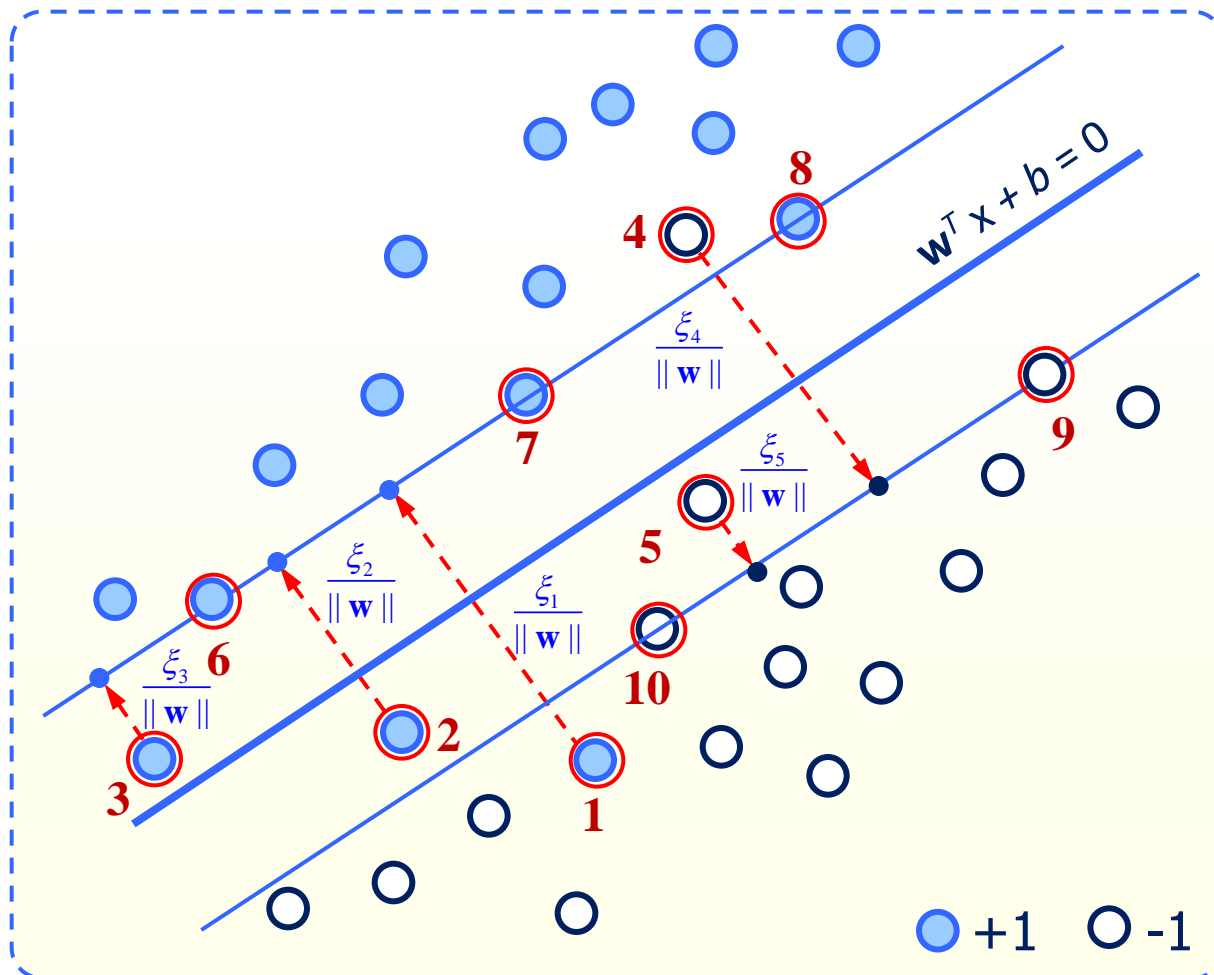
$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, n$$

**软间隔支持向量：注意  $\alpha_j^* > 0$  的样本点均称为支持向量！**

图中，第一个点到其正确边界的距离为： $\frac{\xi_1}{\|\mathbf{w}\|}$ ，其它类推。

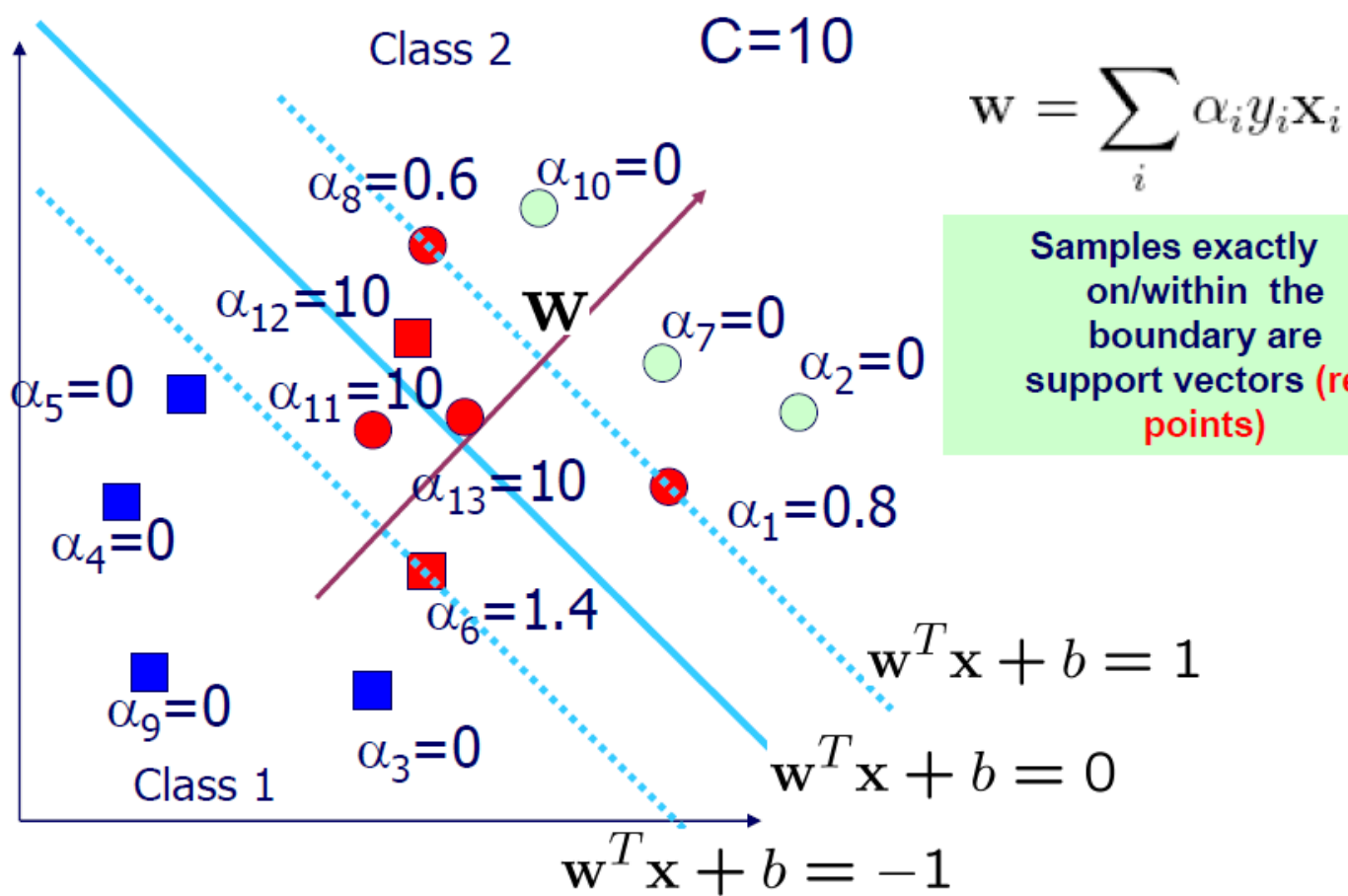
- 1:  $\alpha^* = C, \xi_1 > 1$
- 2:  $\alpha^* = C, \xi_2 > 1$
- 3:  $\alpha^* = C, 0 < \xi_3 < 1$
- 4:  $\alpha^* = C, \xi_4 > 1$
- 5:  $\alpha^* = C, 0 < \xi_5 < 1$
- 6:  $0 < \alpha^* < C, \xi_6 = 0$
- 7:  $0 < \alpha^* < C, \xi_7 = 0$
- 8:  $0 < \alpha^* < C, \xi_8 = 0$
- 9:  $0 < \alpha^* < C, \xi_9 = 0$
- 10:  $0 < \alpha^* < C, \xi_{10} = 0$

10个支持向量



图中带红色圆圈的均表示支持向量

## Support Vectors (Soft-Margin Case)



## 15.4.2 支持向量机(对偶)

- 软间隔支持向量

- 支撑面以外（两个类边界以外）的样本点。均有： $\alpha^* = 0$

- 支持向量： $\alpha^* > 0$

- 包含位于边界上的点，两个类边界以内的，以及错分点（边界以外）。

- 位于类边界上的点其对应的拉格朗日乘子可能有如下三种情形：

- $\alpha^* = 0$  (正好不是支持向量);

- $0 < \alpha^* < C$ ;

- $\alpha^* = C$

- 模型求解：序列最小最优算法（暂时略）

## 15.3.3 支持向量机解的存在性

- 定理3

- 设  $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_1^*, \dots, \alpha_n^*]^T \in R^n$  是对偶问题的解，则至少存在一个下标  $j$ ，有  $0 < \alpha_j^* < C$ ，可按下式求得原始问题的最优解：

$$\mathbf{w}^* = \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i, \quad b^* = y_j - \sum_{i=1}^n \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}_j)$$

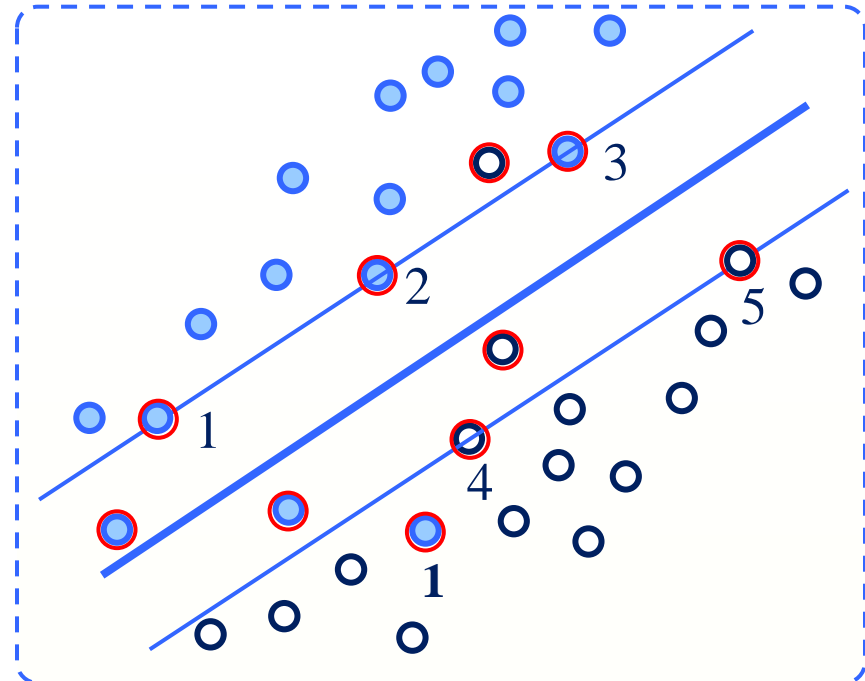
- 分类超平面： $\mathbf{w}^* \cdot \mathbf{x} + b^* = 0$
- 分类决策函数： $f(\mathbf{x}) = \text{sign}(\mathbf{w}^* \cdot \mathbf{x} + b^*) = \text{sign}\left(\sum_{i=1}^n \alpha_i^* y_i (\mathbf{x} \cdot \mathbf{x}_i) + b^*\right)$



### 15.3.3 支持向量机解的存在性

- 偏置 $b$ 的确定

- 不唯一



$$b^* = y_j - \sum_{i=1}^n y_i \alpha_i^* (\mathbf{x}_i \cdot \mathbf{x}_j), \quad 0 < \alpha_j^* < C$$

在所有符合条件的样本上计算一个 $b^*$ ，然后取平均：

编程技巧：

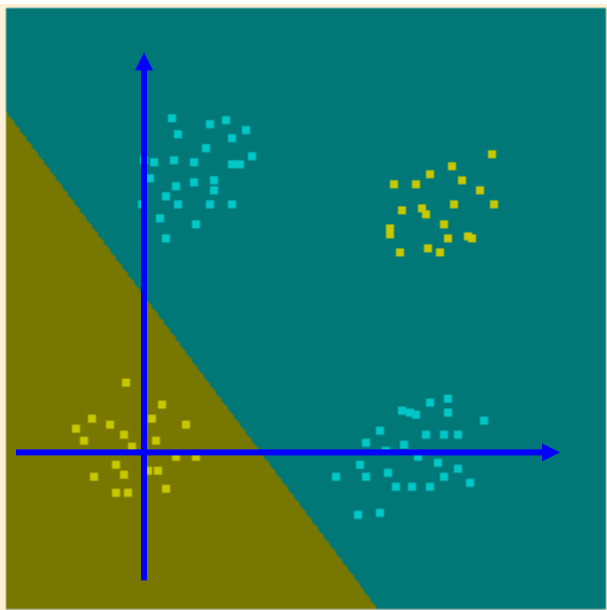
$$b^* = \frac{1}{|\{\alpha_k^* : 0 < \alpha_k^* < C\}|} \sum_{0 < \alpha_k^* < C} (y_k - \sum_{i=1}^n \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x}_j)$$

$|\{\cdot\}|$ : 表示集合的基，也就是集合元素的个数

# 15.5 Kernel Methods

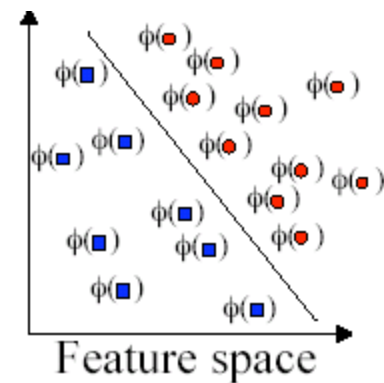
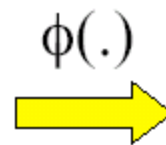
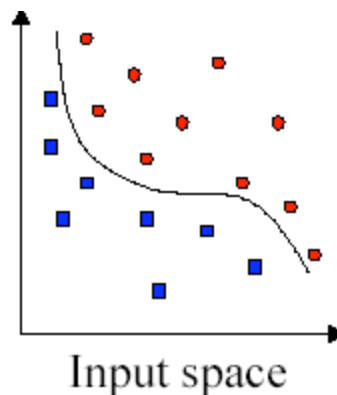
## 15.5.1 特征变换 (Feature Transformation)

- The problem is non-linear
- Find some trick to transform the input
- Linear separable after Feature Transformation
- What Features should we use ?



XOR Problem

Basic Idea :

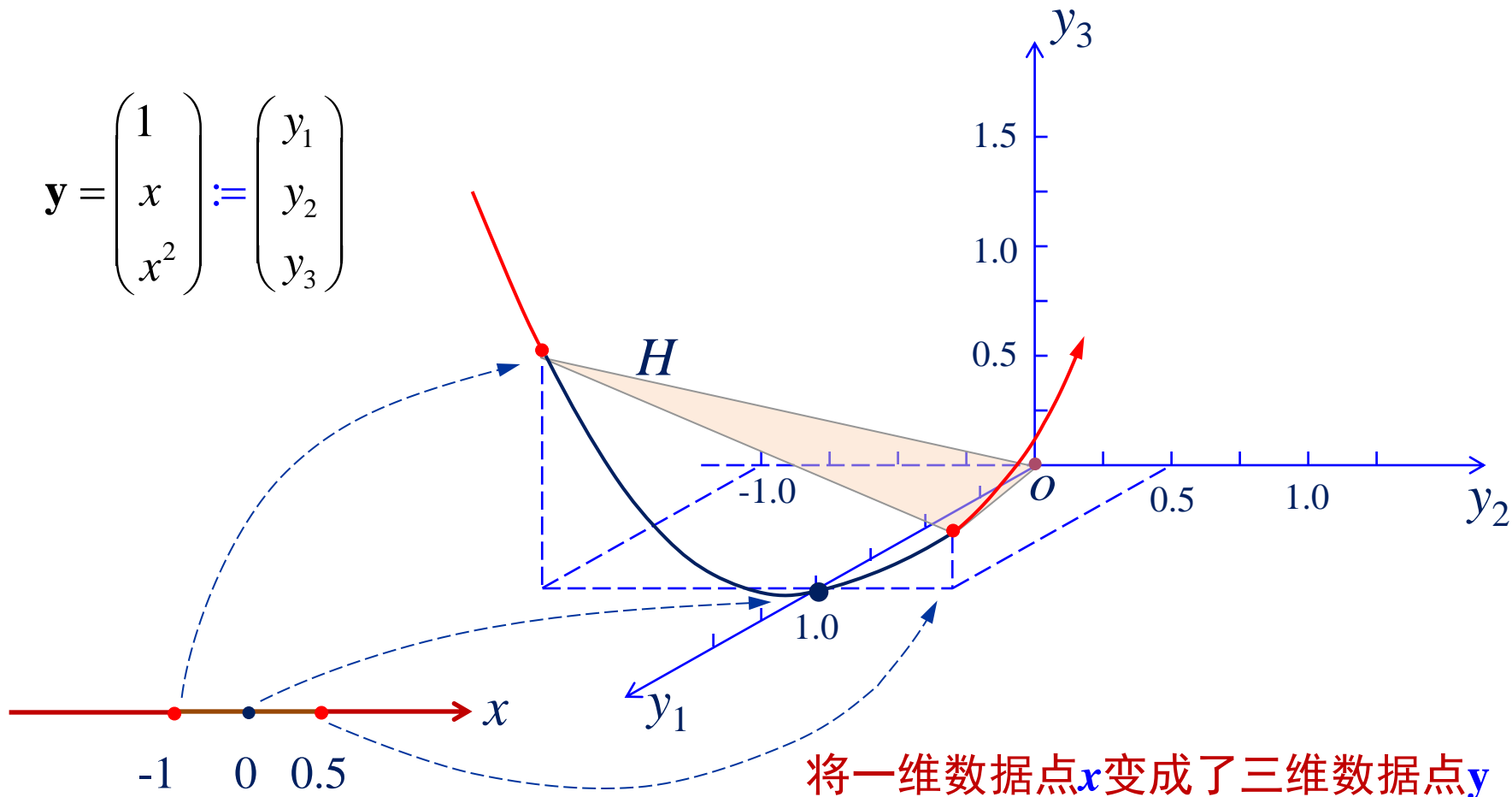


# 15.5.1 特征变换

## 回顾：广义线性判别函数的例子

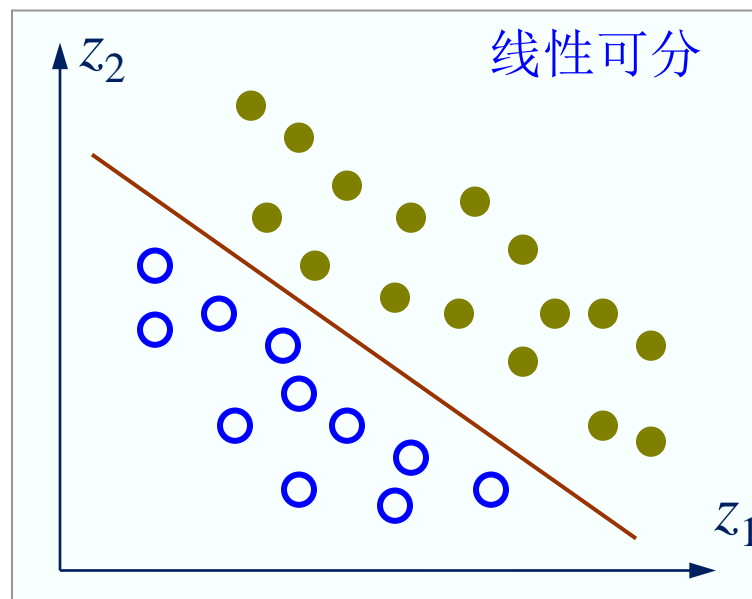
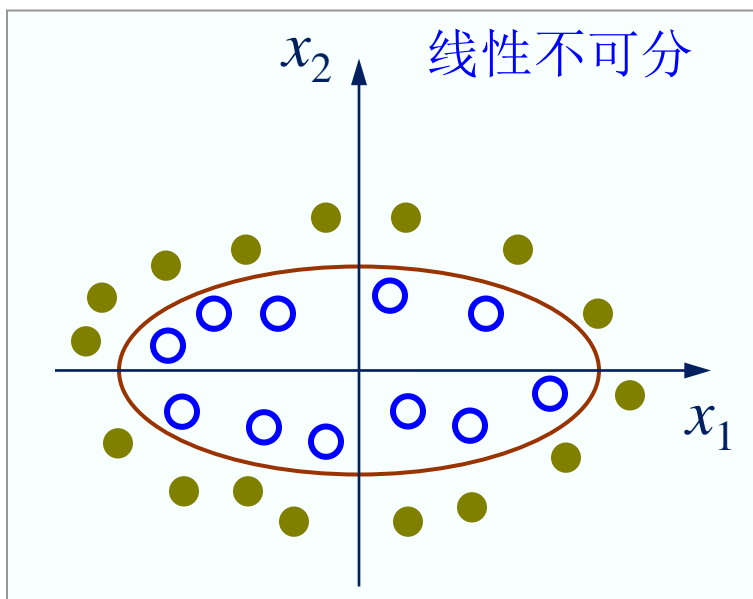
例子

- 设有一维样本空间 $X$ ，我们期望如果  $x < -1$  或者  $x > 0.5$ ，则  $x$  属于第一类 $\omega_1$ ；如果  $-1 < x < 0.5$ ，则属于第二类 $\omega_2$ ，请设计一个判别函数  $g(x)$ 。
- 决策函数： $g(x) = (x-0.5)(x+1)$
- 决策规则： $g(x) > 0$ ,  $x$  属于 $\omega_1$ ； $g(x) < 0$ ,  $x$  属于 $\omega_2$



# 15.5.1 特征变换

## 非线性分类问题



椭圆:  $w_1 x_1^2 + w_2 x_2^2 + b = 0$



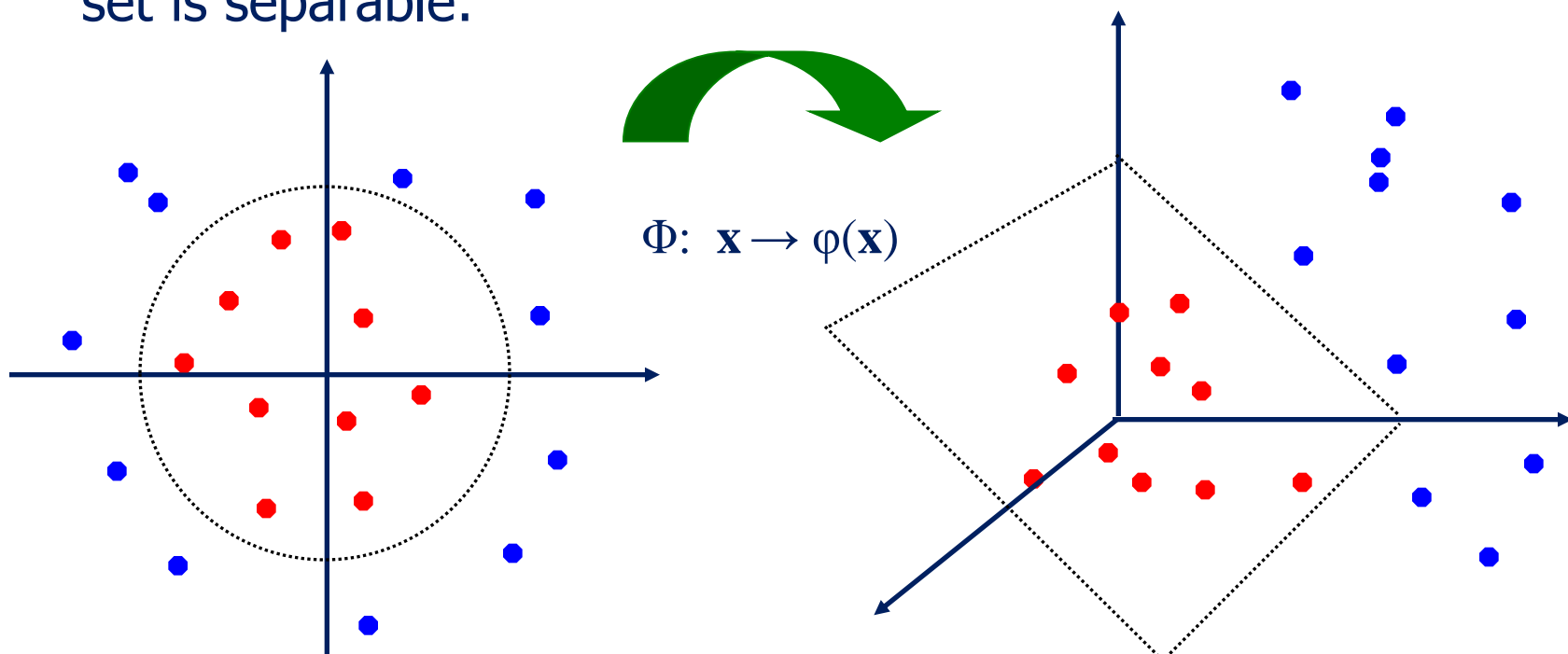
直线:  $w_1 z_1 + w_2 z_2 + b = 0$

变换:  $\mathbf{z} = \phi(\mathbf{x}) = ((x_1)^2, (x_2)^2)^T$

## 15.5.1 特征变换

### Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:




$$\Phi: R^2 \rightarrow R^3$$

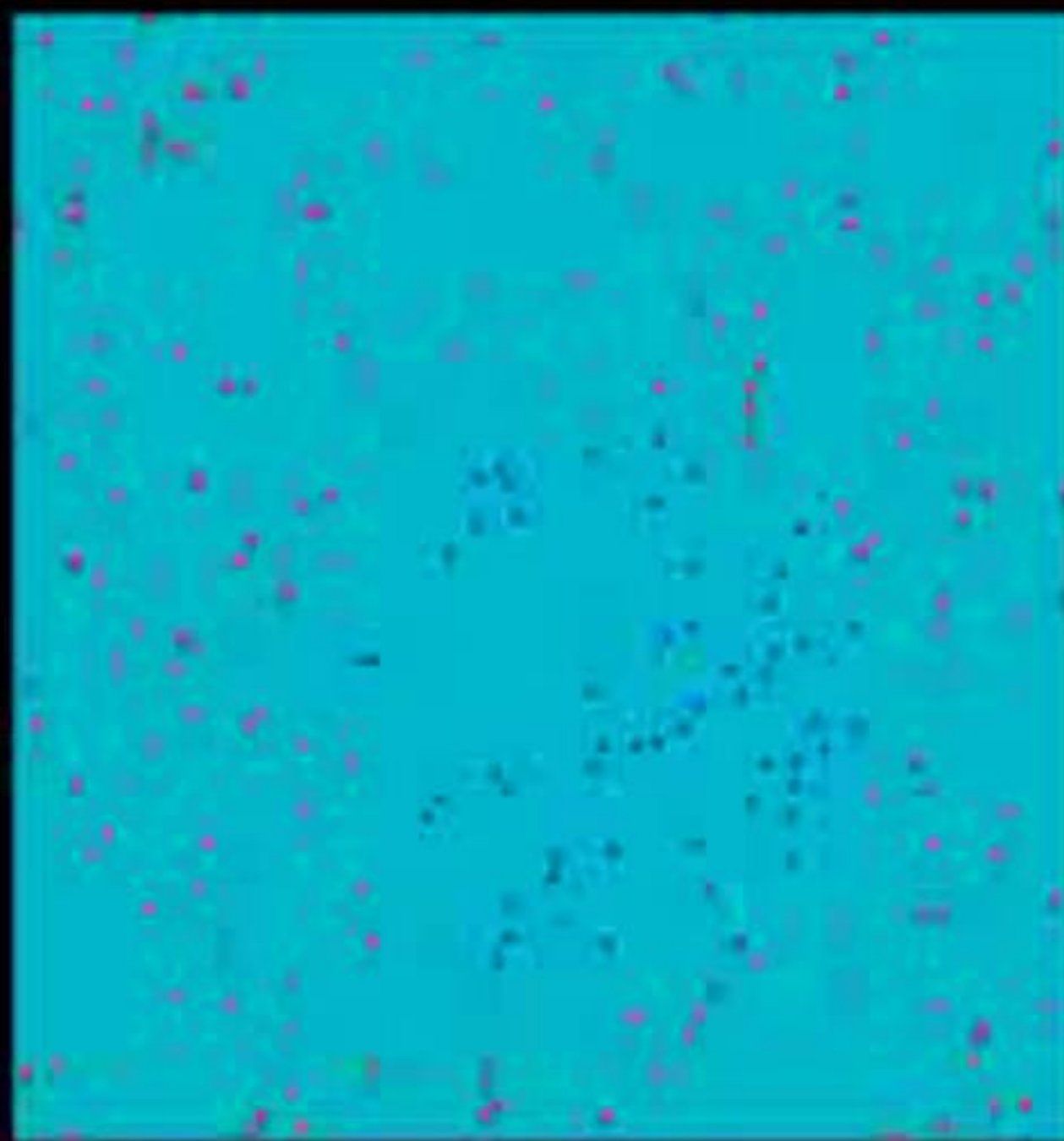
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

## 15.5.2 核技巧

- 用线性方法解决非线性问题
  - 第一步，使用一个变换将原空间中的数据映射到新空间
  - 第二步，在新空间里用线性分类学习方法从训练中学习一个分类模型



核技巧就是属于这样的方法！





## 15.5.2 核技巧

Note that data only appears as dot products

Recall:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j x_i x_j \\ \text{subject to} \quad & C \geq \alpha_i \geq 0, \sum_{i=1}^N \alpha_i y_i = 0 \end{aligned}$$

Since data is only represented as **dot products**, we need **not do the mapping explicitly**.

Introduce a Kernel Function  $K$  such that:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

**Kernel function** – a function that can be applied **to pairs of input data to evaluate dot products in some corresponding feature space**

# Example Transform

- Consider the following transformation

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

- Define the kernel function  $K(\mathbf{x}, \mathbf{y})$  as

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \right\rangle = (1 + x_1y_1 + x_2y_2)^2 \quad := \quad K(\mathbf{x}, \mathbf{y})$$

- The inner product  $\phi(\cdot) \cdot \phi(\cdot)$  can be computed by  $K$  without going through the map  $\phi(\cdot)$  explicitly!!!

# Examples of Kernel Function

- Polynomial kernel with degree  $d$

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel with width  $\sigma$

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$$

– Closely related to radial basis function neural networks

- Sigmoid with parameter  $\kappa$  and  $\theta$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} + \theta)$$

– It does not satisfy the Mercer condition on all  $\kappa$  and  $\theta$

- Research on different kernel functions in different applications is very active

## 15.5.2 核技巧

### Modification Due to Kernel Function:

- Change **all inner products to kernel functions**
- For training,

Original

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

With kernel  
function

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

## 15.5.2 核技巧

### Modification Due to Kernel Function:

- For testing, the new data  $\mathbf{z}$  is classified as *class 1* if  $f \geq 0$  and as *class 2* if  $f < 0$

Original

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

## 15.5.2 核技巧

### Modification Due to Kernel Function:

- Find the bias  $b$

Original 
$$b = y_i - \sum_{j=1}^s \alpha_j y_j x_j^T x_i \quad \forall 0 < \alpha_i < C$$

With kernel function 
$$b = y_i - \sum_{j=1}^s \alpha_j y_j k(x_j, x_i) \quad \forall 0 < \alpha_i < C$$

## KSVM小结:

- 从对偶问题直接实现SVM核化—训练

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



$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^n \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \quad i = 1, 2, \dots, n \end{aligned}$$

## KSVM小结:

- 预测 (对新数据)

$$f(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^n \alpha_i^* y_i (\mathbf{x} \cdot \mathbf{x}_i) + b^* \right), \quad b^* = y_j - \sum_{i=1}^n \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}_j)$$



$$f(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^n \alpha_i^* y_i K(\mathbf{x} \cdot \mathbf{x}_i) + b^* \right), \quad b^* = y_j - \sum_{i=1}^n \alpha_i^* y_i K(\mathbf{x}_i \cdot \mathbf{x}_j)$$



## Example

- Suppose we have five 1D data points  
–  $x_1=1, x_2=2, x_3=4, x_4=5, x_5=6$ , with 1, 2, 5 as class 1; and  
3, 4 as class 2:  $y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1$
- We use the polynomial kernel of degree 2  
–  $K(x,y) = (xy+1)^2$   
–  $C$  is set to 100
- We first find  $\alpha_i$  ( $i=1, \dots, 5$ ) by

$$\max. \quad \sum_{i=1}^5 \alpha_i - \frac{1}{2} \sum_{i=1}^5 \sum_{j=1}^5 \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \quad \sum_{i=1}^5 \alpha_i y_i = 0$$

## Example

- By using a QP solver, we get

$$\alpha_1=0, \alpha_2=2.5, \alpha_3=0, \alpha_4=7.333, \alpha_5=4.833$$

–Verify that the constraints are indeed satisfied

–The support vectors are  $\{x_2=2, x_4=5, x_5=6\}$

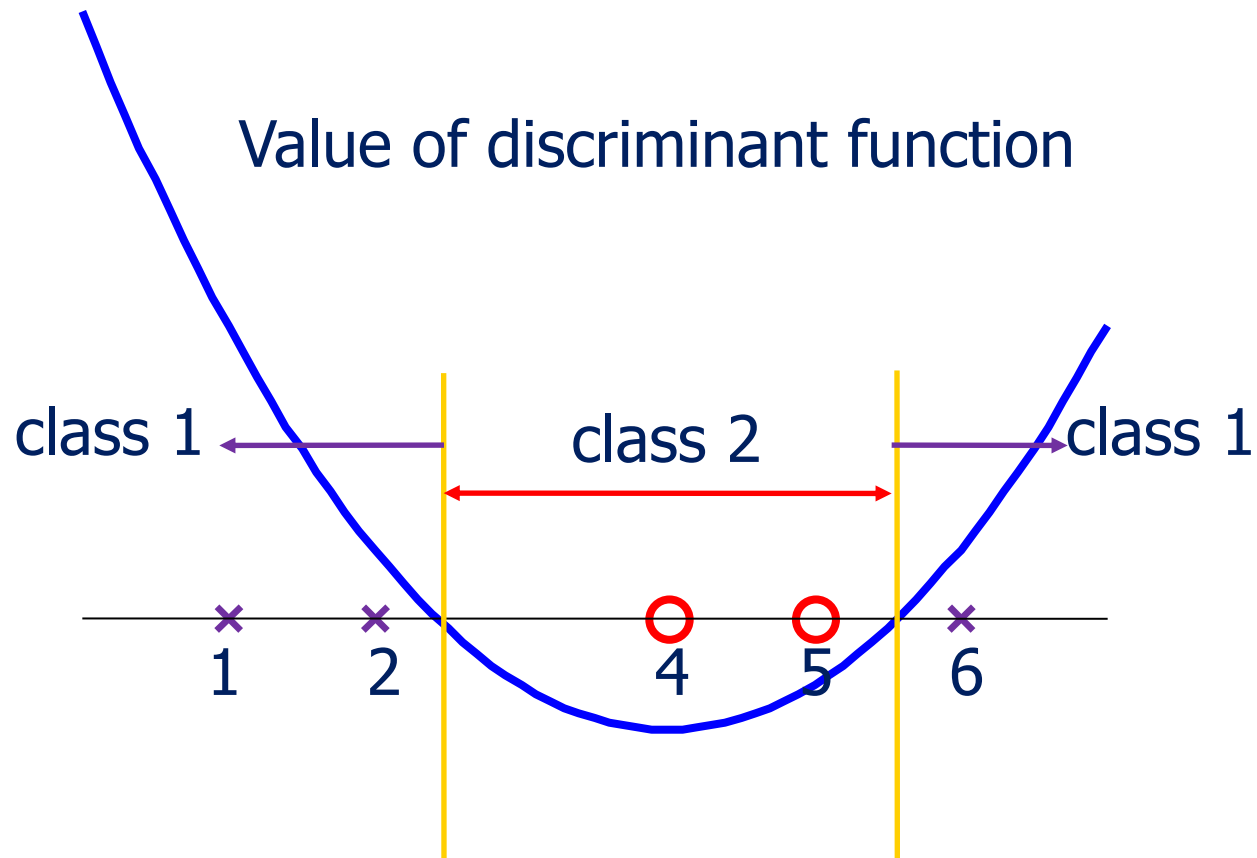
- The discriminant function is

$$\begin{aligned} f(y) &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

- $b$  is recovered by solving  $f(2)=1$  or by  $f(5)=-1$  or by  $f(6)=1$ ,  
as  $x_2, x_4, x_5$  lie on  $y_i(\mathbf{w}^T \phi(z) + b) = 1$  and all give  $b=9$

$$\Rightarrow f(y) = 0.6667x^2 - 5.333x + 9$$

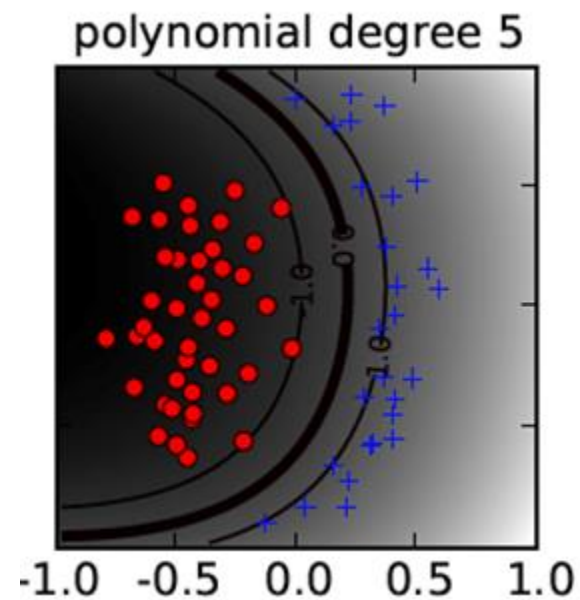
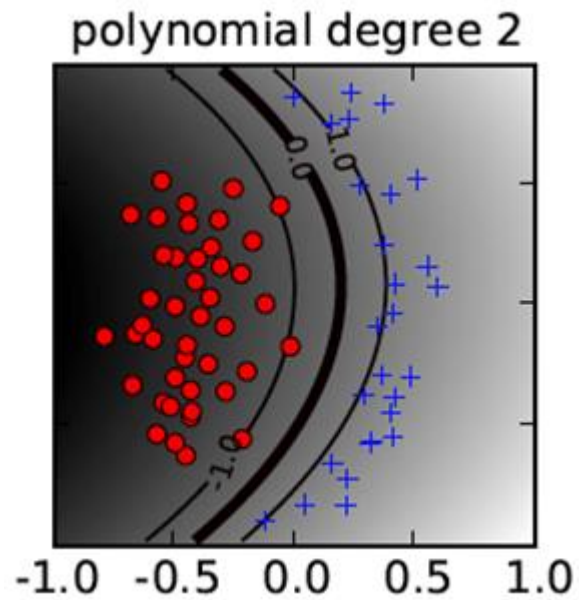
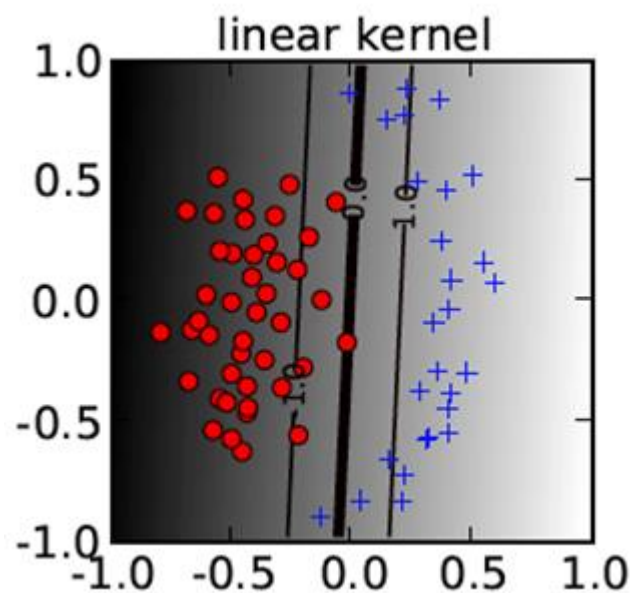
## Example



## 15.5.3 Choosing Kernel Functions

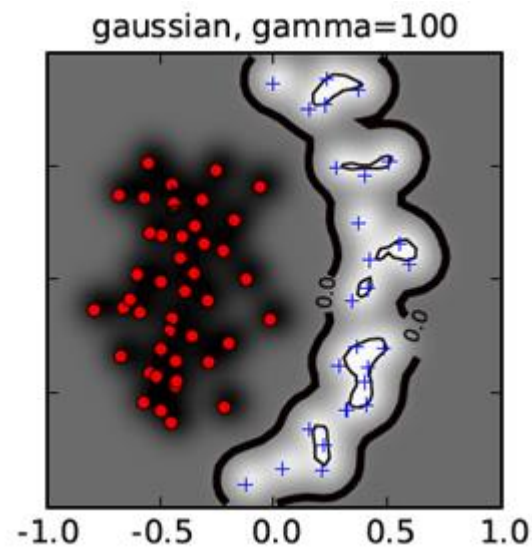
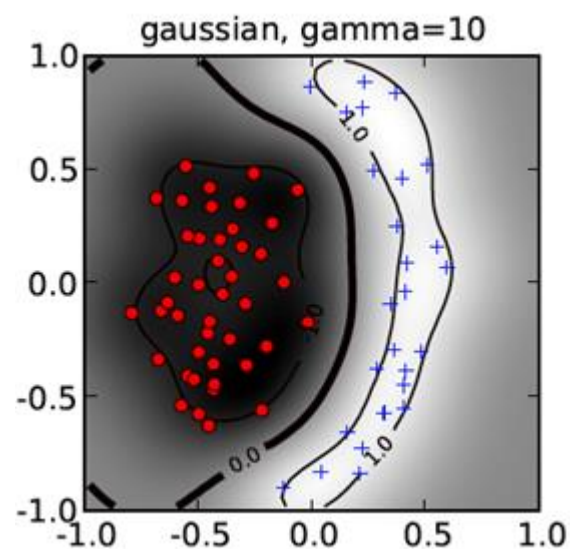
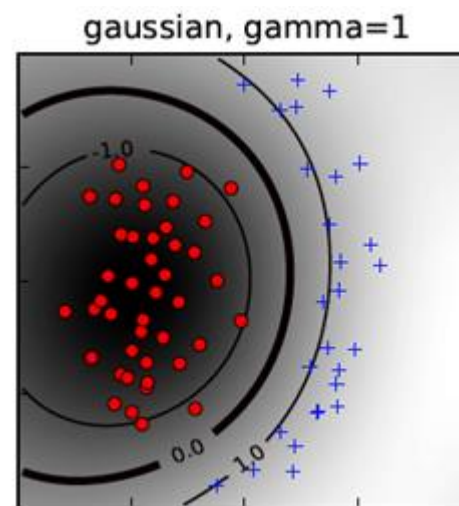
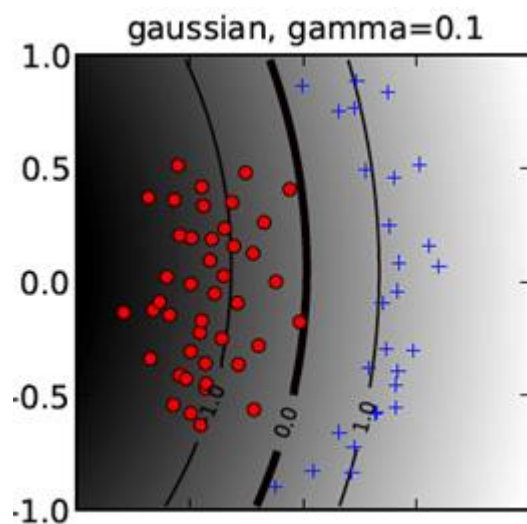
- Probably the most tricky part of using SVM:
  - The kernel function is important because it creates the kernel matrix, which summarizes all the data
  - Many principles have been proposed (diffusion kernel, **Fisher kernel**, string kernel, ...)
  - There is even research to estimate the kernel matrix from available information
  - **Multiple Kernel Learning**
  - In practice, a **low degree polynomial kernel or RBF kernel with a reasonable width** is a good initial try
  - Note that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

# Polynomial Kernel



# RBF Kernel

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$$



# 15.6 Conclusion and Discussion

## 15.6.1 Steps in SVM

- Prepare data matrix  $\{(\mathbf{x}_i, \mathbf{y}_i)\}$
- Select a Kernel function
- Select the error parameter  $C$
- “Train” the system (to find all  $\alpha_i$  and  $b$ )
- New data can be classified using  $\alpha_i$  and Support Vectors



## 15.6.2 Weakness

- **Training (Testing) is quite slow compared to NN**
  - Because of Constrained Quadratic Programming
- **Essentially a binary classifier**
  - However, there are some tricks to evade this.
- **Very sensitive to noise. Why?**
  - A few off data points can completely throw off the algorithm
- **Biggest Drawback: The choice of Kernel function.**
  - There is no “set-in-stone” theory for choosing a kernel function for any given problem (still in research...)
  - Once a kernel function is chosen, there is only ONE modifiable parameter, the error penalty  $C$ .

## 15.6.3 Strengths

- **Training is relatively easy**
  - don't have to deal with local minimum like in ANN
  - SVM solution is always global and unique
- **Unlike NN, doesn't suffer from “curse of dimensionality”**
  - How? Why? We have infinite dimensions?!
  - Maximum Margin Constraint: DOT-PRODUCTS!
- **Less prone to overfitting**
- **Simple, easy to understand geometric interpretation.**
  - No large networks to mess around with.

## 15.6.4 Kernelize Logistic Regression (LR)

$$p(y | \vec{x}) = \frac{1}{1 + \exp(-y\vec{x} \cdot \vec{w})}$$

$$l_{reg}(\vec{\alpha}) = \sum_{i=1}^N \log \frac{1}{1 + \exp(-y\vec{x} \cdot \vec{w})} - c \sum_{k=1}^N w_k^2$$

**What is the difference between SVM and LR?**

- ✓ SVM: L2 norm regularization + **hinge loss**
- ✓ LR: L2 norm regularization + **sigmoid probability (entropy loss)**

Merits and drawbacks?

How can we introduce the **nonlinearity** into the logistic regression?

## 15.6.4 Kernelize Logistic Regression (LR)

$$\vec{x} \rightarrow \vec{\phi}(\vec{x}), \vec{w} = \sum_{i=1}^N \alpha_i \vec{\phi}(\vec{x}_i)$$

$$K(\vec{w}, \vec{x}) = \sum_{i=1}^N \alpha_i K(\vec{x}_i, \vec{x})$$

$$p(y | \vec{x}) = \frac{1}{1 + \exp(-yK(\vec{x}, \vec{w}))} = \frac{1}{1 + \exp\left(-y \sum_{i=1}^N \alpha_i K(\vec{x}_i, \vec{x})\right)}$$

$$l_{reg}(\vec{\alpha}) = \sum_{i=1}^N \log \frac{1}{1 + \exp\left(-y_i \sum_{j=1}^N \alpha_j K(\vec{x}_j, \vec{x}_i)\right)} - c \sum_{i,j=1}^N \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j)$$

- Representation Theorem
- **Kernelization** of many algorithms (PCA, LDA ...)
  - From linear to non-linear without changing the algorithm

**Thank All of You!**  
**(Questions?)**

**向世明**

**smxiang@nlpr.ia.ac.cn**

**peopleucas.ac.cn/~xiangshiming**

**时空数据分析与学习课题组(STDAL)**

**中科院自动化研究所·模式识别国家重点实验室**