神经网络
Neural Networks
第七章

核函数方法

史忠植

中国科学院计算技术研究所
http://www.intsci.ac.cn/

内容提要

- 7.1 概述
- 7.2 统计学习问题
- 7.3 学习过程的一致性
- 7.4 结构风险最小归纳原理
- 7.5 支持向量机
- 7.6 核函数
- 7.7 核主成分分析

支持向量机

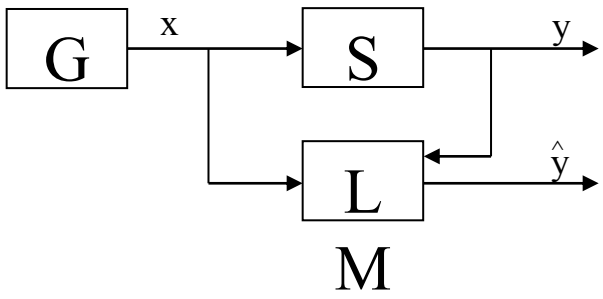
- SVM是一种基于统计学习理论的机器学习方法，它是由Boser,Guyon, Vapnik在COLT-92上首次提出，从此迅速发展起来
- Vapnik V N. 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York
- Vapnik V N. 1998. Statistical Learning Theory. Wiley-Interscience Publication, John Wiley&Sons, Inc
- 目前已经在许多智能信息获取与处理领域都取得了成功的应用。

统计学习理论

- 统计学习理论是小样本统计估计和预测学习的最佳理论。
- 假设输出变量Y与输入变量X之间存在某种对应的依赖关系,即一未知概率分布*P(X,Y)*，P(X,Y)反映了某种知识。学习问题可以概括为:根据*l*个独立同分布(independently drawn and identically distributed)的观测样本*train set*，

  (x1,y1),(x2,y2),…,(xn,yn)

函数估计模型



- 学习样本的函数:
  - 产生器 (G) generates observations x (typically in R^n), independently drawn from some fixed distribution $F(x)$
  - 训练器Supervisor (S) labels each input x with an output value y according to some fixed distribution $F(y/x)$
  - 学习机Learning Machine (LM) "learns" from an i.i.d. l-sample of (x, y)-pairs output from G and S, by choosing a function that best approximates S from a parameterised function class $f(x, \alpha)$, where α is in Λ the parameter set
- 关键概念: $F(x, y)$, an i.i.d. l-sample on F, functions $f(x, \alpha)$ and the equivalent representation of each f using its index α

期望风险

学习到一个假设$H=f(x, w)$ 作为预测函数,其中w是广义参数.它对$F(X,Y)$的期望风险$R(w)$是(即统计学习的实际风险)：

$$R(w)=\int L(y, f(x,w))dF(x,y)$$

其中，$\{f(x,w)\}$称作预测函数集，w为函数的广义参数。$\{f(x,w)\}$可以表示任何函数集。$L(y,f(x,w))$为由于用$f(x,w)$对y进行预测而造成的损失。不同类型的学习问题有不同形式的损失函数。

经验风险

而对*train set*上产生的风险$R_{emp}(w)$被称为经验风险(学习的训练误差):

$$R_{emp}(w)=\frac{1}{l}\sum_{i=1}^{l}L(y_i, f(x_i, w))$$

首先$R_{emp}(w)$和$R(w)$都是w的函数，传统概率论中的定理只说明了(在一定条件下)当样本趋于无穷多时$R_{emp}(w)$*将*在概率意义上趋近于$R(w)$，却没有保证使$R_{emp}(w)$最小的点也能够使$R(w)$ 最小(同步最小)。

经验风险

根据统计学习理论中关于函数集的推广性的界的结论，对于两类分类问题中的指示函数集$f(x, w)$的所有函数(当然也包括使经验风险员小的函数)，经验风险$R_{emp}(w)$和实际风险$R(w)$之间至少以不下于1-η(0≤η≤1)的概率存在这样的关系:

$$R(w) \le R_{emp}(w) + \phi(h/l)$$

VC维

VC维(Vapnik-Chervonenkis Dimension)。模式识别方法中VC维的直观定义是：对一个指示函数集，如果存在h个样本能够被函数集里的函数按照所有可能的2h种形式分开，则称函数集能够把h个样本打散。函数集的VC维就是它能打散的最大样本数目h。

$$\phi(h/l)=\sqrt{\frac{h(\ln(2l/h+1)-\ln(\eta/4)}{l}}$$

h是函数$H=f(x, w)$的VC维, l是样本数.

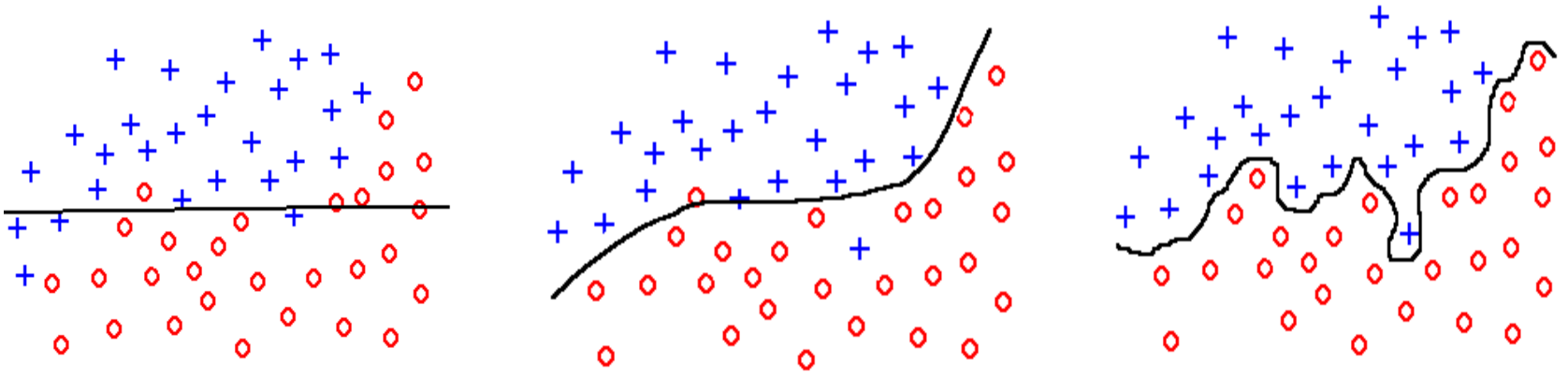
过学习

一般的学习方法(如神经网络)是基于 *Remp(w)* 最小,满足对已有训练数据的最佳拟和,在理论上可以通过增加算法（如神经网络）的规模使得*Remp(w)*不断降低以至为0。

但是,这样使得算法（神经网络）的复杂度增加, VC维h增加,从而*φ(h/l)*增大,导致实际风险*R(w)*增加,这就是学习算法的过拟合(Overfitting).

过学习 Overfitting and underfitting

Problem: how rich class of classifications q(**x**;**θ**) to use.



underfitting　　good fit　　overfitting

Problem of generalization: a small emprical risk R_{emp} does not imply small true expected risk R.

学习理论的四个部分

1. 学习过程的一致性理论

What are (necessary and sufficient) conditions for consistency (convergence of R_{emp} *to* R*) of a learning process based on the ERM Principle?*

2.学习过程收敛速度的非渐近理论

How fast is the rate of convergence of a learning process?

3. 控制学习过程的泛化能力理论

How can one control the rate of convergence (the generalization ability) of a learning process?

4. 构造学习算法的理论

How can one construct algorithms that can control the generalization ability?

结构风险最小化归纳原则 (SRM)

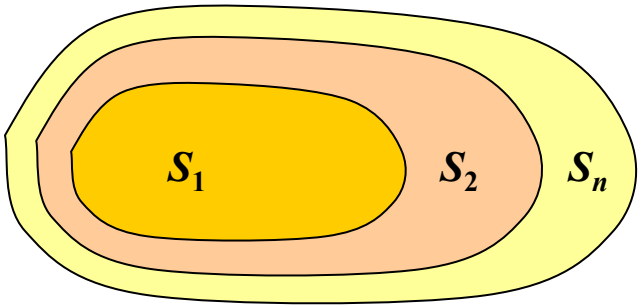
- **ERM** is intended for relatively **large samples** (large *l*/*h*)
  - Large *l*/*h* induces a small ε which decreases the the upper bound on risk
  - **Small samples?** Small empirical risk doesn't guarantee anything!
    …we need to minimise both terms of the RHS of the risk bounds
    1. The empirical risk of the chosen $\alpha \in \Lambda$
    2. An expression depending on the VC dimension of Λ

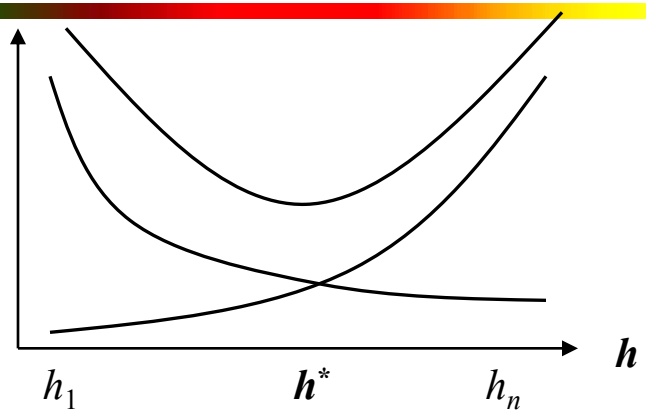
结构风险最小化归纳原则 (SRM)

- The Structural Risk Minimisation (SRM) Principle
  - Let $S = \{Q(z,\alpha), \alpha \in \Lambda\}$. An **admissible structure** $S_1 \subset S_2 \subset \ldots \subset S_n \subset \ldots \subset S$:
    - For each k, the VC dimension h_k of S_k is finite and $h_1 \le h_2 \le \ldots \le h_n \le \ldots \le h_S$
    - Every S_k is either is non-negative bounded, or satisfies for some (p, τ_k)

$$\sup_{\alpha \in \Lambda_k} \frac{\left(\int Q^p(z,\alpha) dF(z)\right)^{\frac{1}{p}}}{R(\alpha)} \le \tau_k, \quad p > 2$$

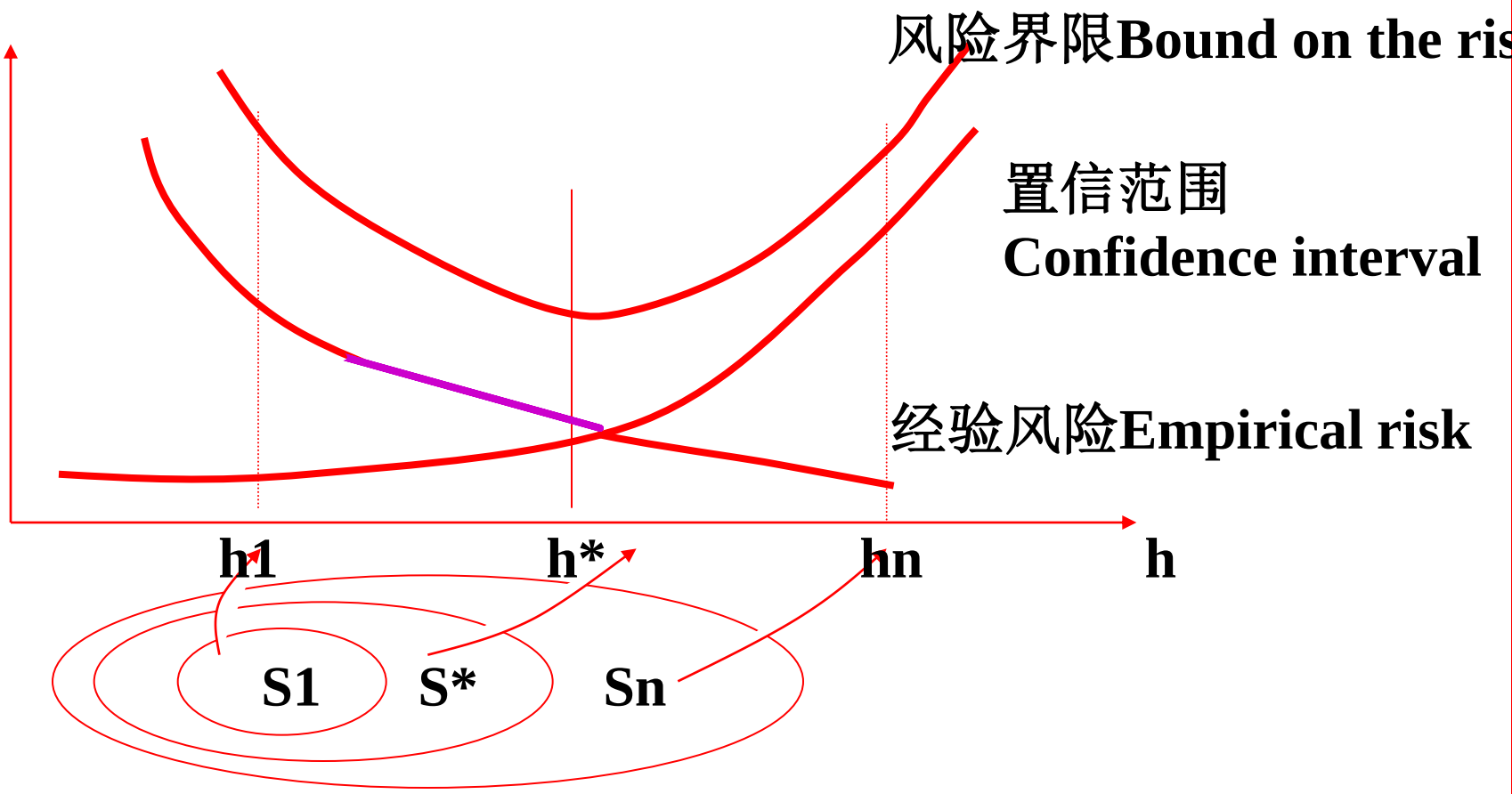
结构风险最小化归纳原则 (SRM)





- The SRM Principle *continued*
  - For given $z_1,\ldots,z_l$ and an admissible structure $S_1 \subset S_2 \subset \ldots \subset S_n \subset \ldots \subset S$, SRM chooses function $Q(z,\alpha_l^k)$ minimising R_{emp} in S_k for which the guaranteed risk (**risk upper-bound**) is minimal
  - Thus manages the unavoidable **trade-off** of *quality of approximation vs. complexity of approximation*

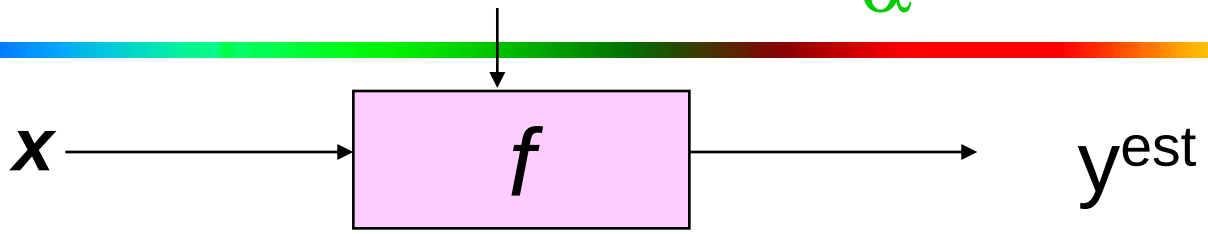
结构风险最小化归纳原则 (SRM)



支持向量机 SVM

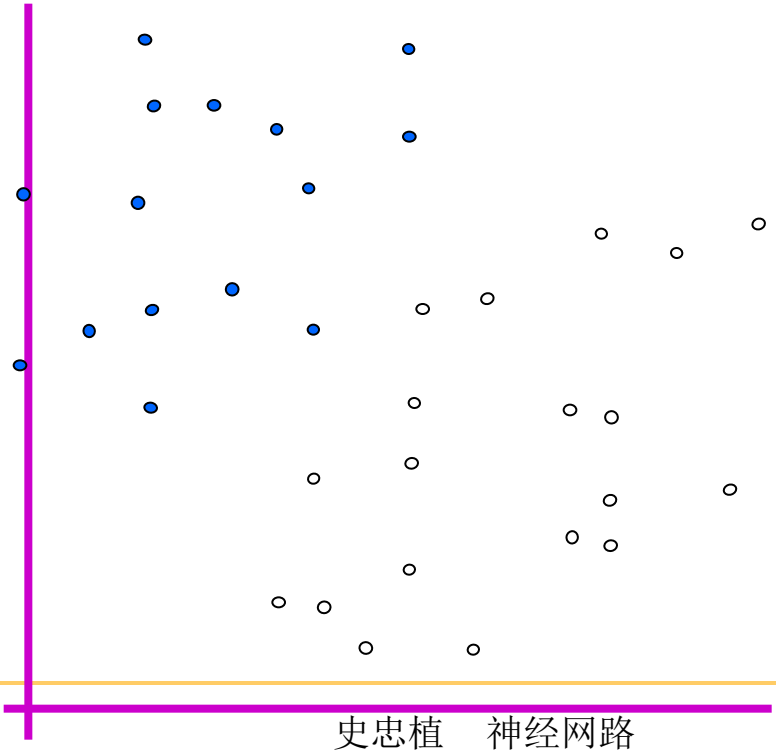
- SVMs are learning systems that
  - use a hyperplane of *linear functions*
  - in a high dimensional feature space — *Kernel function*
  - trained with a learning algorithm from optimization theory — *Lagrange*
  - Implements a learning bias derived from statistical learning theory — *Generalisation* SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis

线性分类器



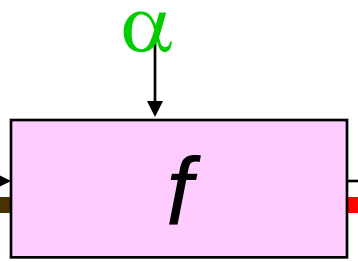
$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\boldsymbol{x} - b)$

- denotes +1
- denotes -1



How would you classify this data?

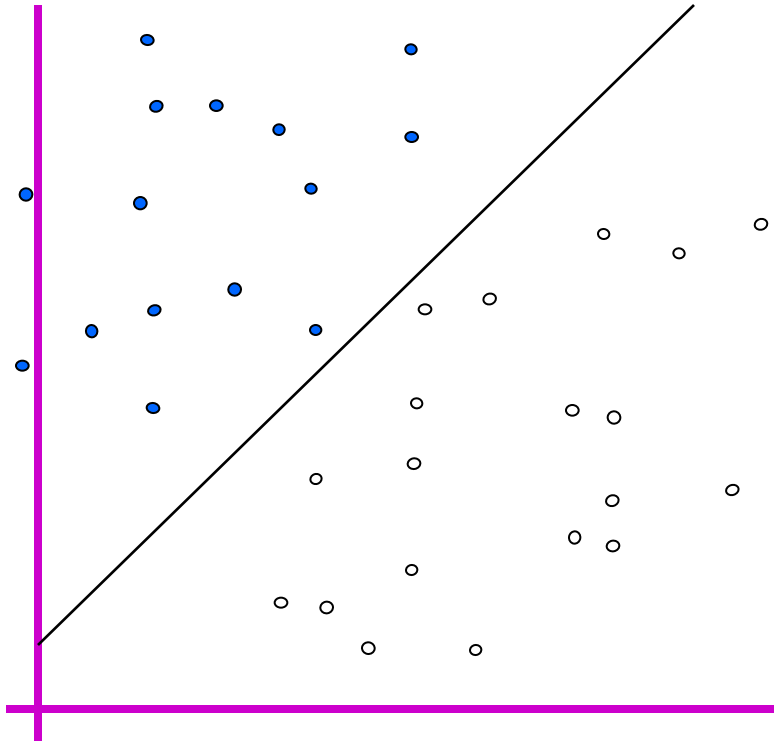
线性分类器



x → f → y^{est}

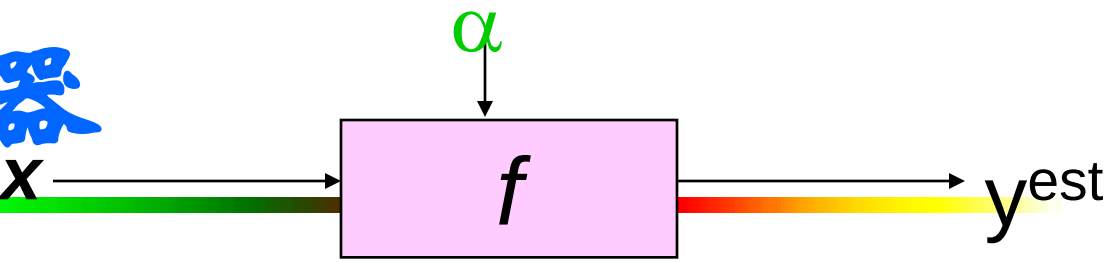
$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} - b)$

- denotes +1
- denotes -1



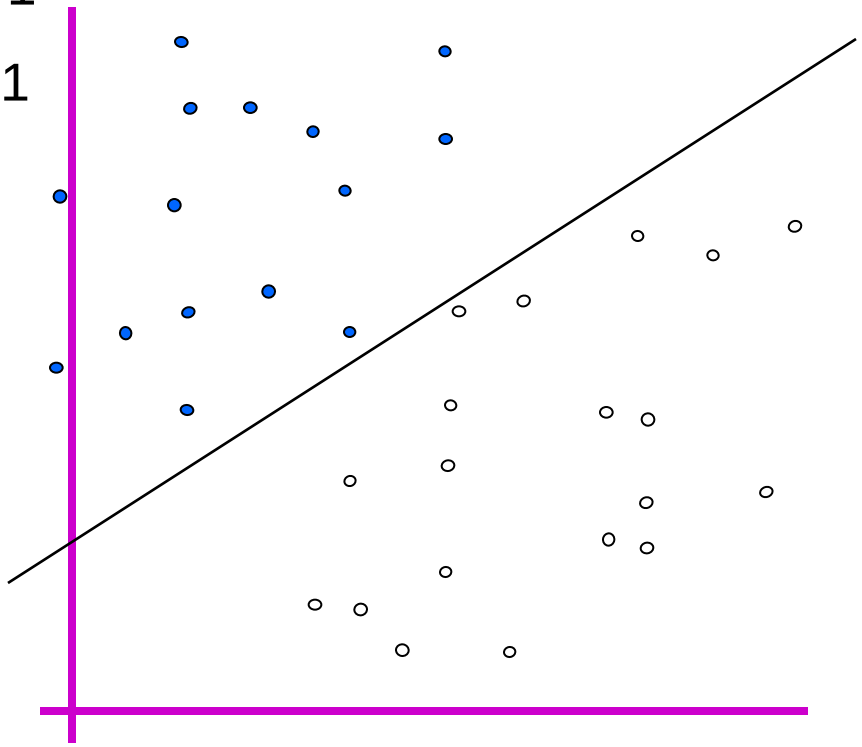
How would you classify this data?

线性分类器



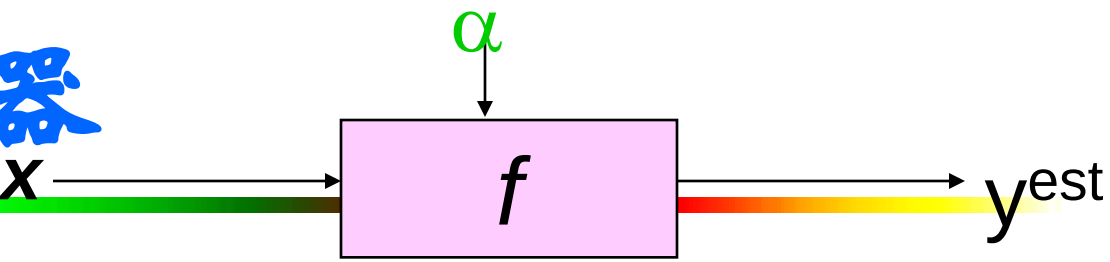
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1

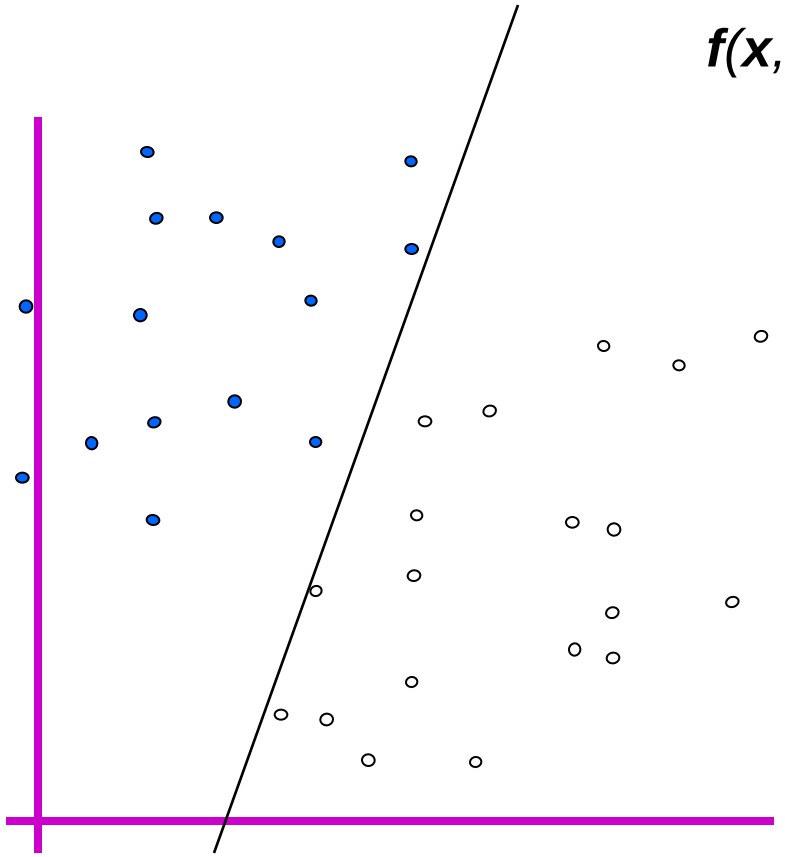


How would you classify this data?

线性分类器



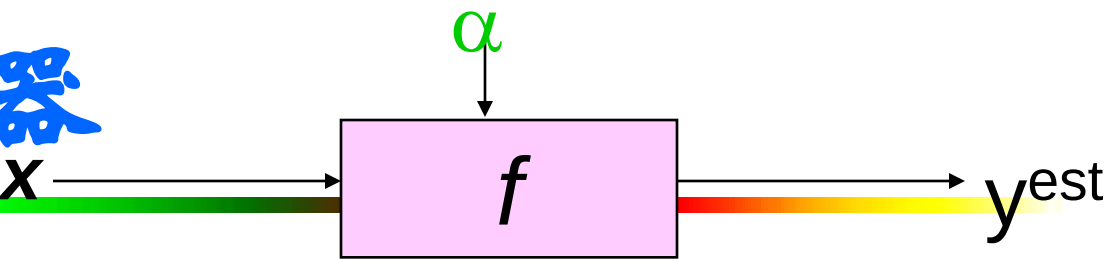
- denotes +1
- denotes -1



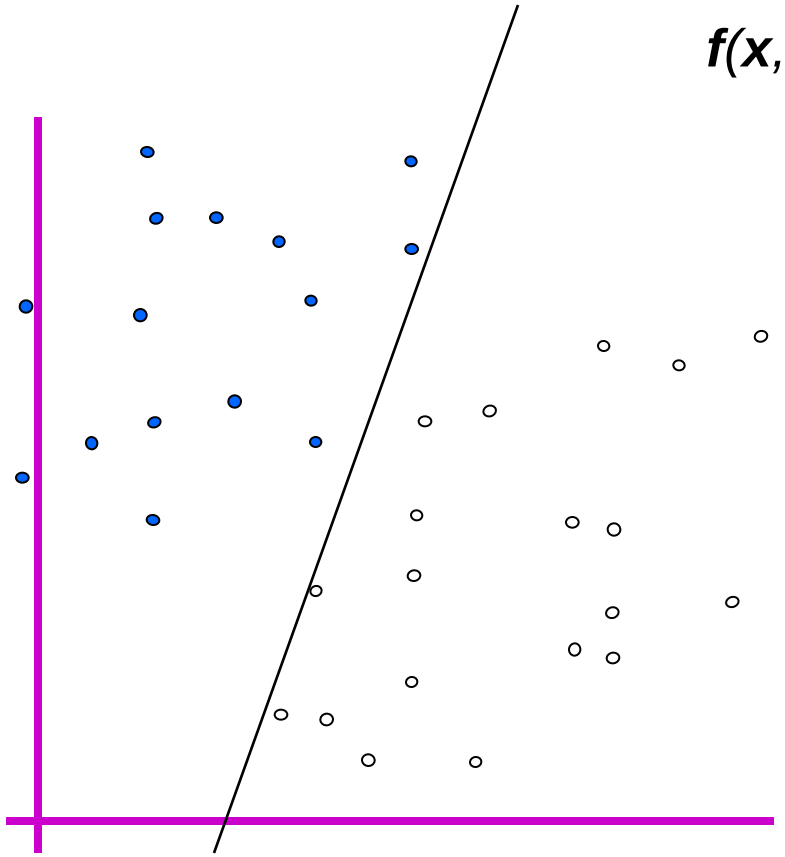
$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} - b)$

How would you classify this data?

线性分类器



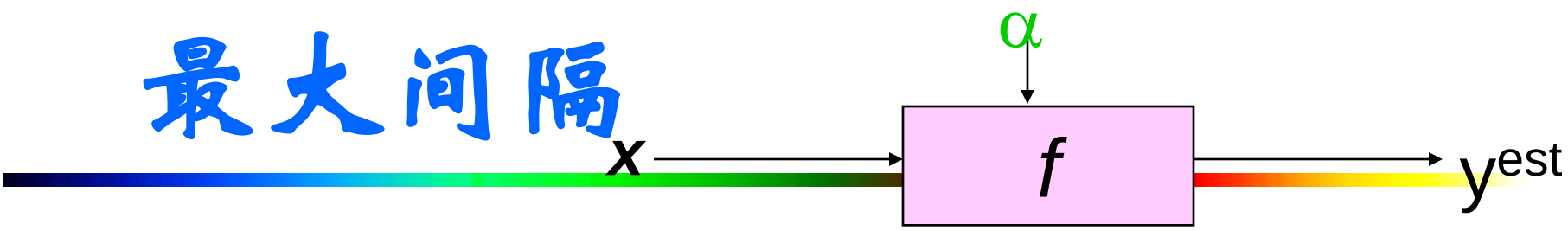
- denotes +1
- denotes -1



$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} - b)$

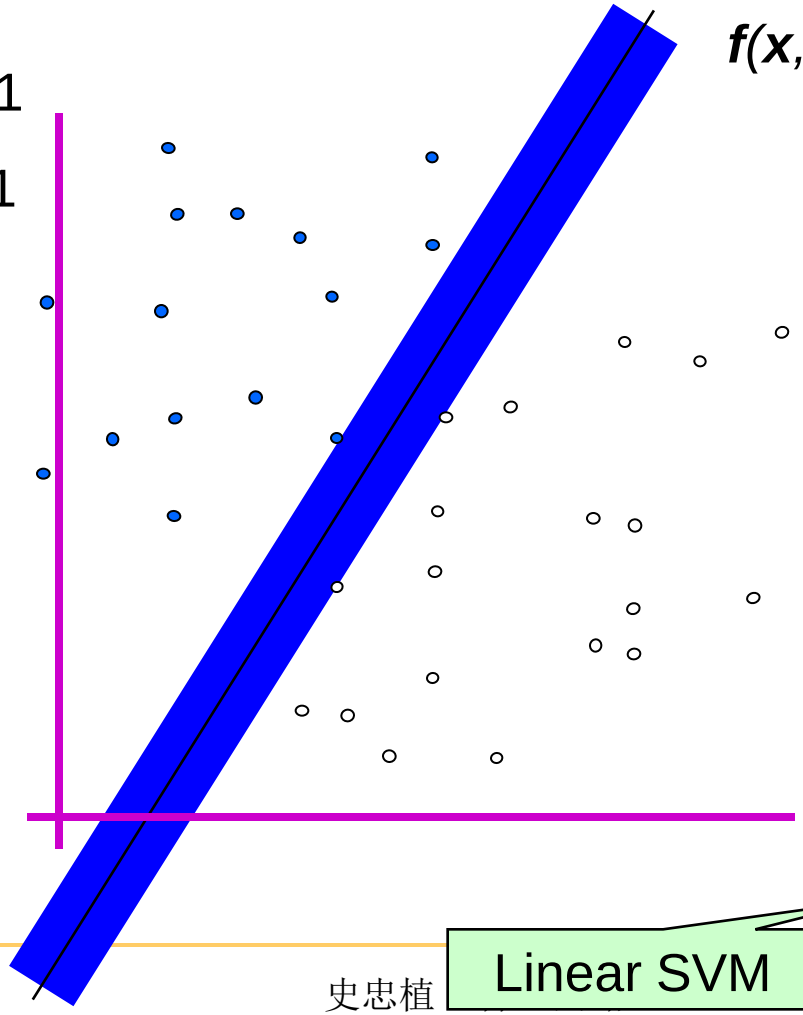
How would you classify this data?

最大间隔



$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} - b)$

- denotes +1
- denotes -1



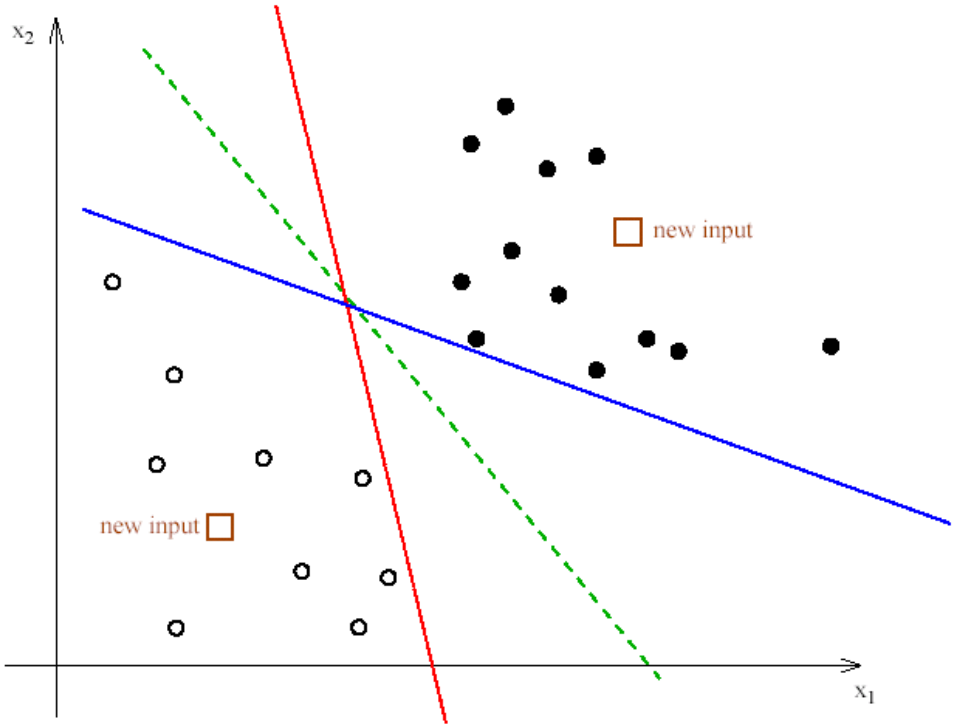
The maximum margin linear classifier is the linear classifier with the maximum margin.

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

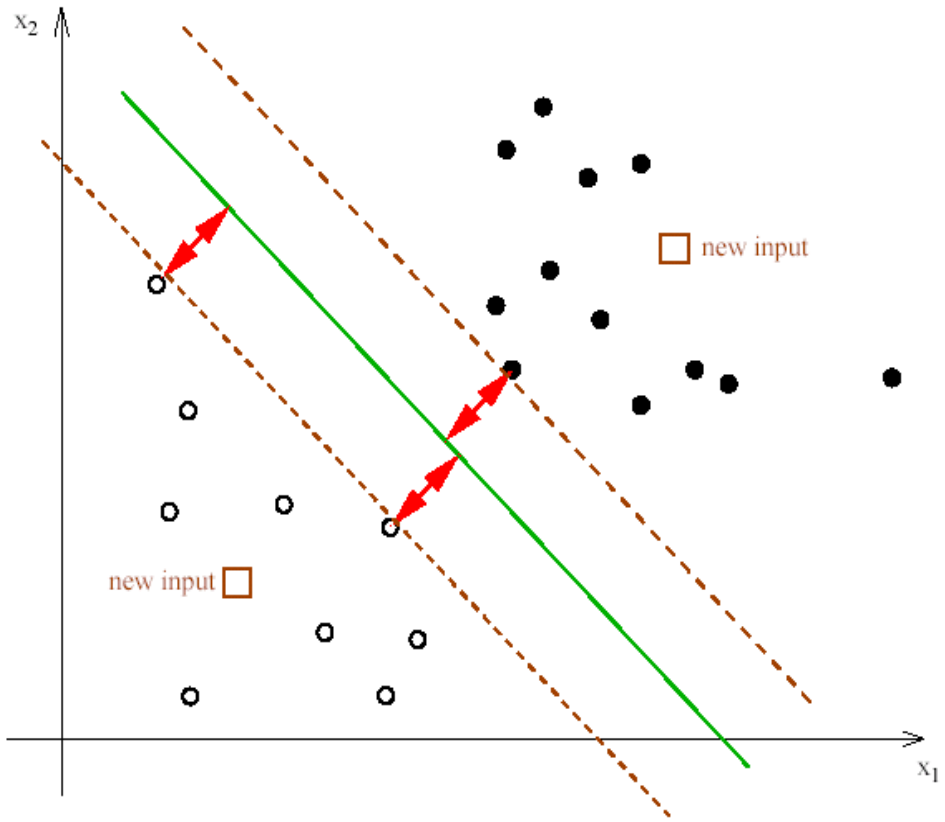
Linear SVM

分类超平面

- Training set: $(\mathbf{x}_i, y_i)$, i=1,2,…N; $y_i \in \{+1,-1\}$
- Hyperplane: $\mathbf{wx}+b=0$
  - This is fully determined by $(\mathbf{w},b)$

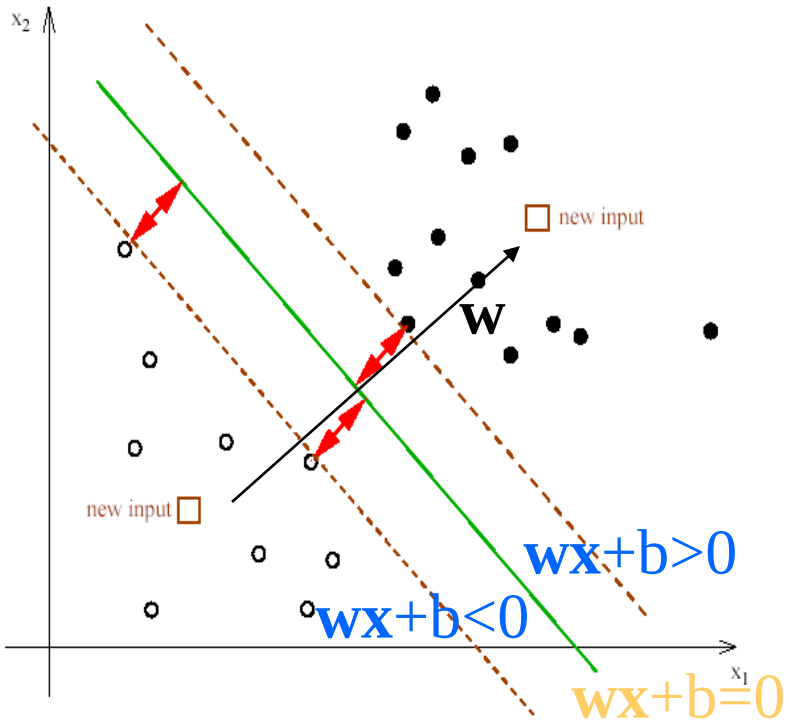


最大间隔



According to a theorem from Learning Theory, from all possible linear decision functions the one that maximises the margin of the training set will minimise the generalisation error.

最大间隔原则



Note1: decision functions (**w**,b) and (c**w**, cb) are the same

Note2: but margins as measured by the outputs of the function **x**→**wx**+b are not the same if we take (**cw**, cb).

Definition: *geometric margin*: the margin given by the *canonical decision function*, which is when c=1/||**w**||

Strategy:

1) we need to maximise the geometric margin! (cf result from learning theory)

2) subject to the constraint that training examples are classified correctly

最大间隔原则

According to Note1, we can demand the function output for the nearest points to be +1 and –1 on the two sides of the decision function. This removes the scaling freedom.

Denoting a nearest positive example x_+ and a nearest negative example x_-, this is

$$\mathbf{w}\mathbf{x}_+ + b = +1 \text{ and } \mathbf{w}\mathbf{x}_- + b = -1$$

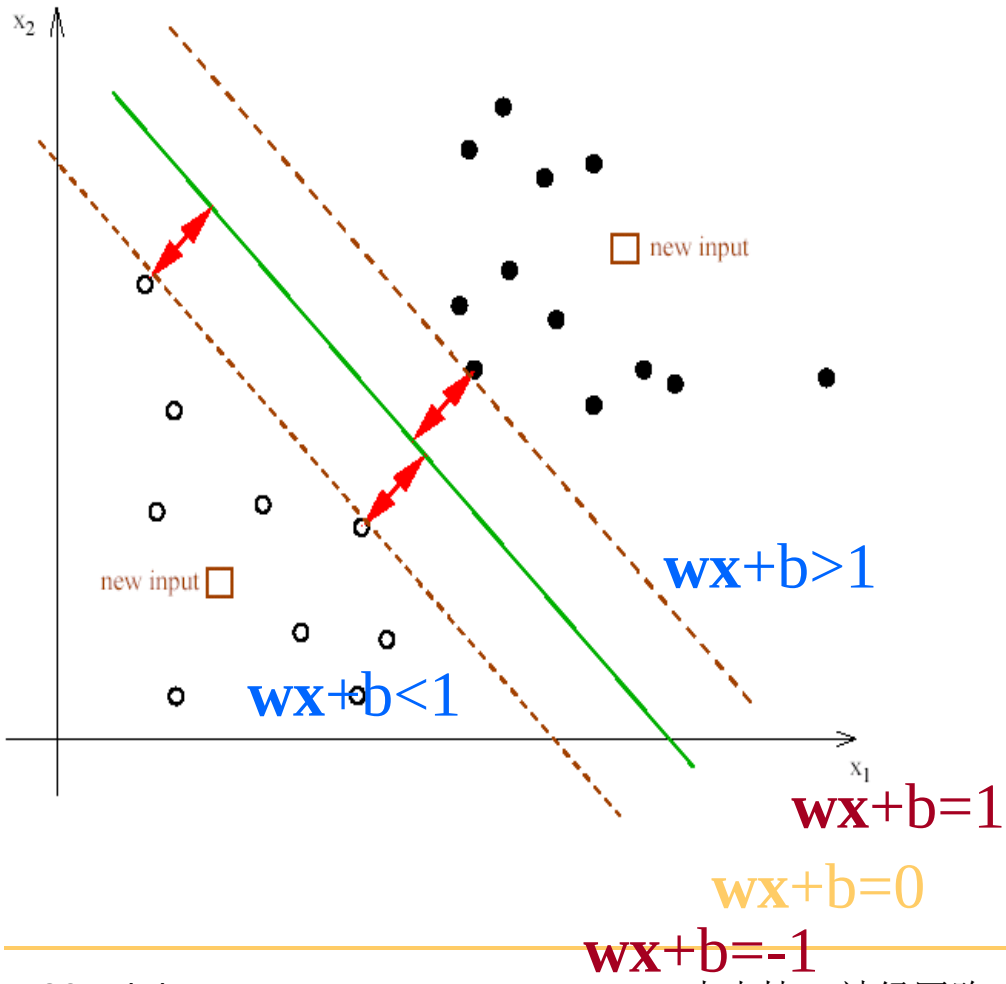
Computing the geometric margin (that has to be maximised):

$$\frac{1}{2}\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\mathbf{x}_+ + \frac{b}{\|\mathbf{w}\|} - \frac{\mathbf{w}}{\|\mathbf{w}\|}\mathbf{x}_- - \frac{b}{\|\mathbf{w}\|}\right) = \frac{1}{2\|\mathbf{w}\|}(\mathbf{w}\mathbf{x}_+ + b - \mathbf{w}\mathbf{x}_- - b) = \frac{1}{\|\mathbf{w}\|}$$

And here are the constraints:

$$\left.\begin{array}{l} \mathbf{w}\mathbf{x}_i + b \geq +1 \quad \text{for } y_i = +1 \\ \mathbf{w}\mathbf{x}_i + b \leq -1 \quad \text{for } y_i = -1 \end{array}\right\} \Leftrightarrow y_i(\mathbf{w}\mathbf{x}_i + b) - 1 \geq 0 \quad \text{for all } i$$

Maximum margin – summing up



Given a linearly separable training set $(\mathbf{x}_i, y_i)$, $i=1,2,\ldots N$; $y_i \in \{+1,-1\}$

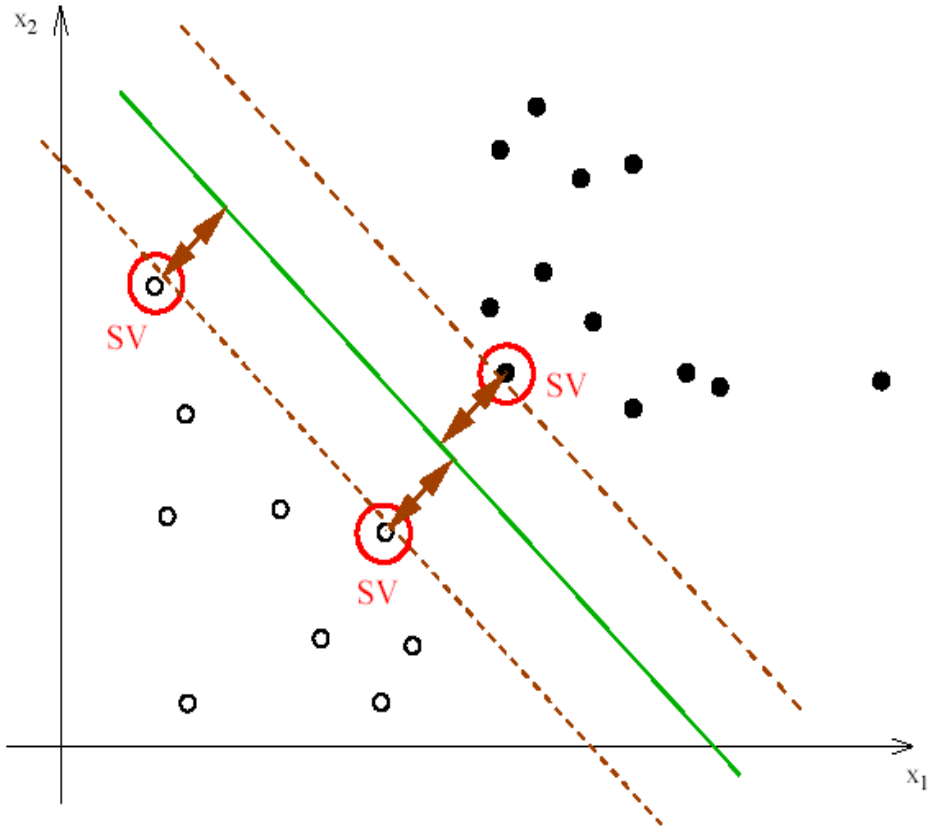
Minimise $\|w\|^2$

Subject to

$$y_i(\mathbf{w}\mathbf{x}_i + b) - 1 \geq 0, \quad i = 1,..N$$

→ This is a quadratic programming problem with linear inequality constraints. There are well known procedures for solving it ☺

支持向量



The training points that are nearest to the separating function are called support vectors.

What is the output of our decision function for these points?

分类问题的数学表示

已知： 训练集包含 l 个样本点：

$$T=\{(x_1,y_1),\cdots,(x_l,y_l)\}\in(X\times y)^l$$

说明： $x_i\in X=R^n$ 是输入指标向量，或称输入，或称模式，其分量称为特征，或属性，或输入指标；

$y_i\in y=\{1,-1\}$ 是输出指标，或输出.

问题： 对一个新的模式 x，推断它所对应的输出 y 是1还是-1.

实质： 找到一个把 R^n 上的点分成两部分的规则.

2维空间上的分类问题) $\Longrightarrow$ n维空间上的分类问题.

分类学习方法

根据给定的训练集 $T=\{(x_1,y_1),\cdots,(x_l,y_l)\}\in(X\times Y)^l$

其中，$x_i\in X=R^n, y_i\in Y=\{1,-1\}, i=1,\cdots,l$，寻找 $X=R^n$ 上的一个实值函数 $g(x)$，用决策函数

$$f(x)=\mathrm{sgn}(g(x))$$

判断任一模式 x 对应的 y 值.

可见，分类学习机——构造决策函数的方法(算法)，

两类分类问题 $\Rightarrow$ 多类分类问题

线性分类学习机 $\Rightarrow$ 非线性分类学习机

分类学习方法

SVM分类问题大致有三种：线性可分问题、近似线性可分问题、线性不可分问题。

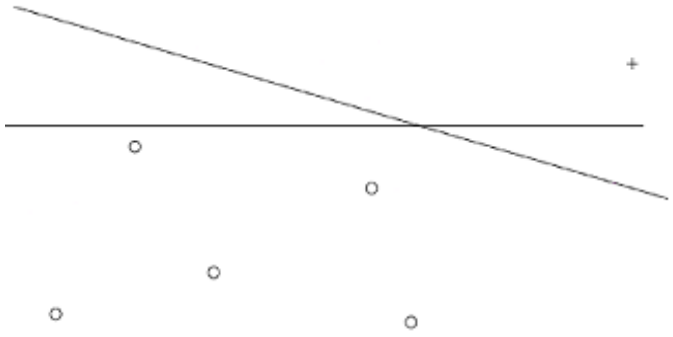


图 1.1.2

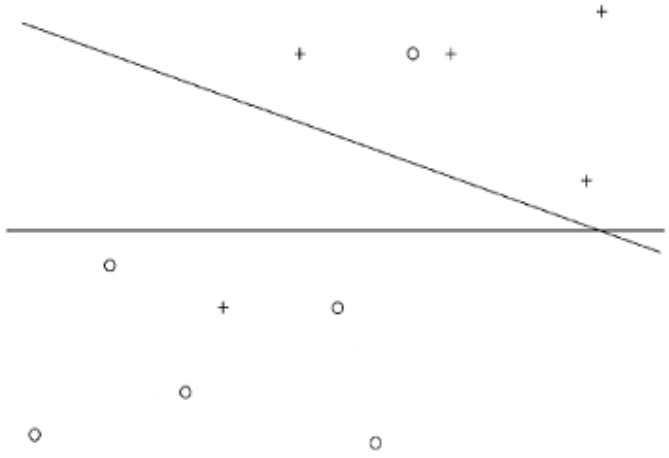


图 1.1.3

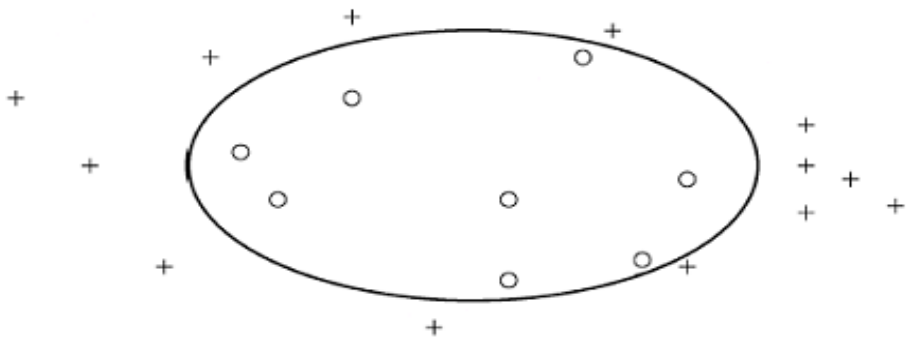


图 1.1.4

最大间隔法的直观导出

考虑 R^2 上的线性可分的分类问题.

这里有许多直线 $(w\cdot x)+b=0$ 能将两类点正确分开.

如何选取 w 和 b？

简单问题：设法方向 w 已选定，如何选取 b ？

解答：w 选定 → 平行直线 → 极端直线 l_2 和 l_3

→ 取 l_2 和 l_3 的中间线为分划直线

如何选取 w ？

对应一个 w，有极端直线 $l_2=l_2(w)$ $l_3=l_3(w)$，称 l_2 和 l_3 之间的距离为“间隔”.显然应选使“间隔”最大的 w。

数学语言描述

给定适当的法方向 $\tilde{w}$ 后，这两条极端直线 k_1, k_2 可表示为

$$(\tilde{w}\cdot x)+\tilde{b}=k_1,(\tilde{w}\cdot x)+\tilde{b}=k_2$$

调整 $\tilde{b}$，使得

$$(\tilde{w}\cdot x)+\tilde{b}=k,\ (\tilde{w}\cdot x)+\tilde{b}=-k$$

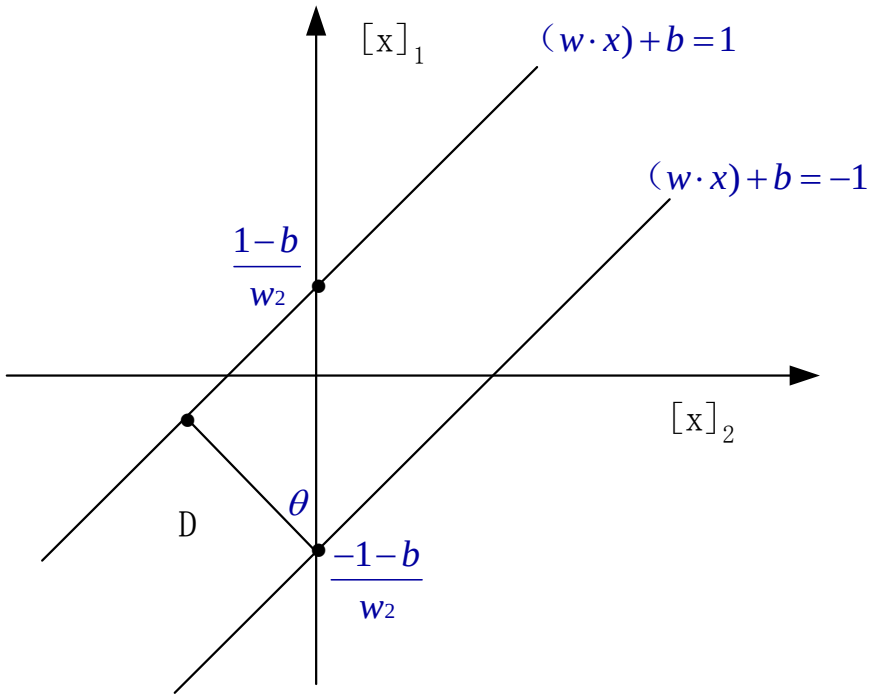
令 $w=\frac{\tilde{w}}{k}, b=\frac{\tilde{b}}{k}$，则两式可以等价写为

$$(w\cdot x)+b=1,\ (w\cdot x)+b=-1$$

与此相应的分划直线表达式：$(w\cdot x)+b=0$

如何计算分划间隔？

考虑2维空间中极端直线之间的间隔情况



$$\frac{\sqrt{\left(\frac{1-b}{w_2}-\frac{-1-b}{w_2}\right)^2-D^2}}{D}=\left|\frac{w_1}{w_2}\right|$$

求出两条极端直线的距离：

$$D=\frac{2}{\sqrt{{w_1}^2+{w_2}^2}}=\frac{2}{\|w\|}$$

分划直线表达式为 $(w\cdot x)+b=0$ “间隔”为 $\frac{2}{\|w\|}$

极大化“间隔”的思想导致求解下列对变量 w 和 b 的最优化问题

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2, \tag{1.2.1}$$

原始问题

$$\text{s.t.} \quad y_i((w\cdot x_i)+b)\geq 1, i=1,\cdots,l \tag{1.2.2}$$

说明：只要我们求得该问题的最优解 w^*,b^*，从而构造分划超平面 $(w^*\cdot x)+b^*=0$，求出决策函数 $f(x)=\text{sgn}((w^*\cdot x)+b^*)$。

上述方法对一般 R^n 上的分类问题也适用.

$$\text{Margin} = \frac{2}{\|\mathbf{W}\|} \qquad \text{.....(1)}$$

H1平面： $\mathbf{W} \bullet \mathbf{X}_1 + b \geq 1$

H2平面： $\mathbf{W} \bullet \mathbf{X}_2 + b \leq -1$

$$y_i[(\mathbf{W} \bullet \mathbf{X}_i) + b] - 1 \geq 0 \qquad \text{.....(2)}$$

求解原始问题

为求解原始问题，根据最优化理论，我们转化为对偶问题来求解

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \left(x_i \cdot x_j\right) - \sum_{j=1}^{l} \alpha_j$$

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

$$\alpha_i \geq 0, i = 1 \cdots l$$

对偶问题

α_i 为原始问题中与每个约束条件对应的Lagrange乘子。这是一个不等式约束条件下的二次函数寻优问题，存在唯一解 α^*

线性可分问题

根据最优解 $\alpha^* = (a_1^*, \cdots, a_l^*)^T$

计算 $w^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i$，选择 a^* 的一个正分量 α_j^*，并据此计算

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

构造分划超平面 $(w^* \cdot x) + b^* = 0$，决策函数 $f(x) = \mathrm{sgn}((w^* \cdot x) + b^*)$

事实上，α^* 的每一个分量 α_i^* 都与一个训练点相对应。而分划超平面仅仅依赖于 α_i^* 不为零的训练点 (x_i, y_i)，而与对应于 α_i^* 为零的那些训练点无关。

称 α_i^* 不为零的这些训练点的输入 x_i 为**支持向量**(SV)

近似线性可分问题

不要求所有训练点都满足约束条件 $y_i\left((w\cdot x_i)+b\right)\geq 1$，为此对第 i 个训练点 (x_i,y_i) 引入**松弛变量**(Slack Variable) $\xi_i\geq 0$，把约束条件放松到 $y_i\left((w\cdot x_i)+b\right)+\xi_i\geq 1$。（即“软化”约束条件）

$\xi=(\xi_1,\cdots\xi_l)^T$ 体现了训练集被错分的情况，可采用 $\sum_{i=1}^{l}\xi_i$ 作为一种度量来描述错划程度。

两个目标： 1. 间隔 $\frac{2}{\|w\|}$ 尽可能大 2. 错划程度 $\sum_{i=1}^{l}\xi_i$ 尽可能小

显然，当 ξ_i 充分大时，样本点 (x_i,y_i) 总可以满足以上约束条件。然而事实上应避免 ξ_i 太大，所以需在目标函数对 ξ 进行惩罚

近似线性可分问题

因此，引入一个**惩罚参数** $C>0$ ，新的目标函数变为:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i$$

$$s.t \quad y_i((w\cdot x_i)+b) \geq 1-\xi_i, i=1,\cdots l$$

$$\xi_i \geq 0, i=1,\cdots l$$

$\sum_{i=1}^{l}\xi_i$ 体现了经验风险，而 $\|w\|$ 则体现了表达能力。所以惩罚参数 C 实质上是对经验风险和表达能力匹配一个裁决。当 $C\to\infty$ 时，近似线性可分SVC的原始问题退化为线性可分SVC的原始问题。

(广义)线性支持向量分类机算法

1. 设已知训练集 $T=\{(x_1,y_1),\cdots,(x_l,y_l)\}\in(X\times y)^l$，其中 $x_i\in X=R^n, y_i\in y=\{1,-1\}, i=1,\cdots,l$

2. 选择适当的惩罚参数 $C>0$，构造并求解最优化问题

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \left(x_i \cdot x_j\right) - \sum_{j=1}^{l}\alpha_j$$

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

$$0 \le \alpha_i \le C, i=1,\cdots l$$

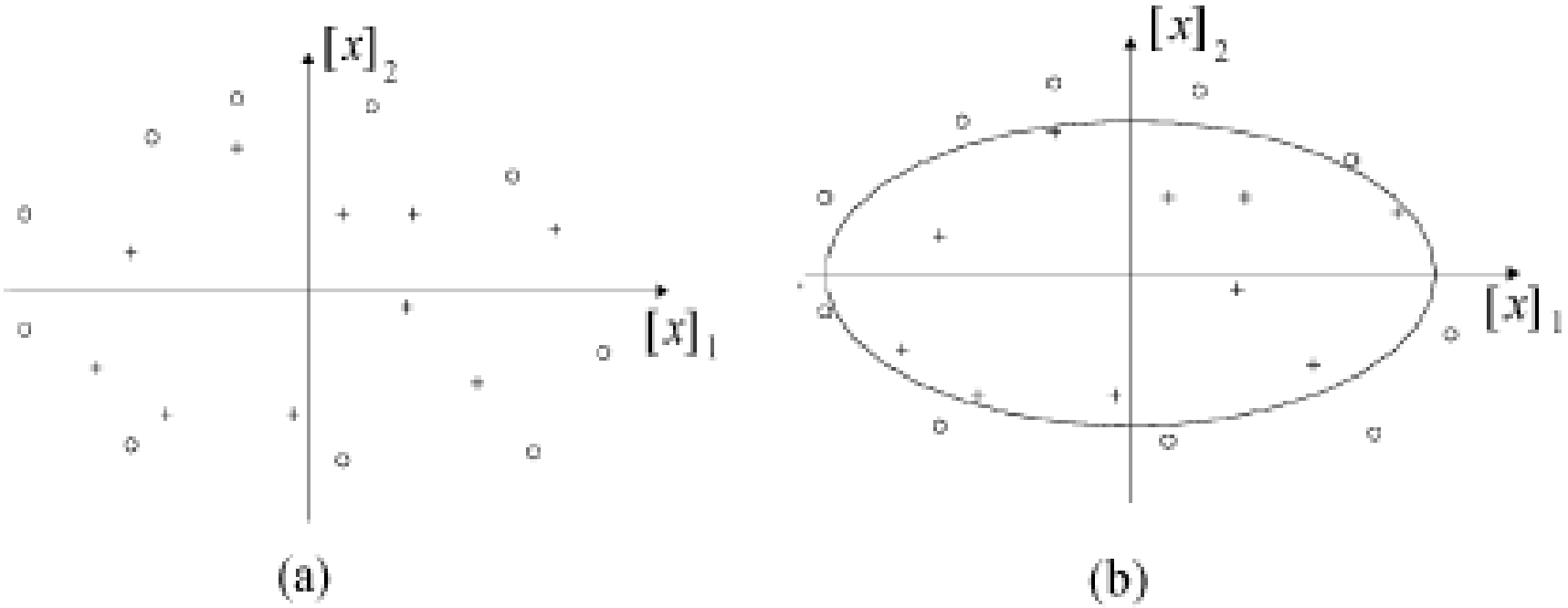
求得 $a^*=(a_1^*,\cdots a_l^*)^T$

3. 计算 $w^*=\sum_{i=1}^{l} y_i \alpha_i^* x_i$，选择 α^* 的一个分量 $0<\alpha_j^*<C$，并据此计算出 $b^*=y_j-\sum_{i=1}^{l} y_i a_i^* \left(x_i \cdot x_j\right)$

4. 构造分划超平面 $(w^*\cdot x)+b^*=0$，决策函数 $f(x)=\mathrm{sgn}((w^*\cdot x)+b^*)$

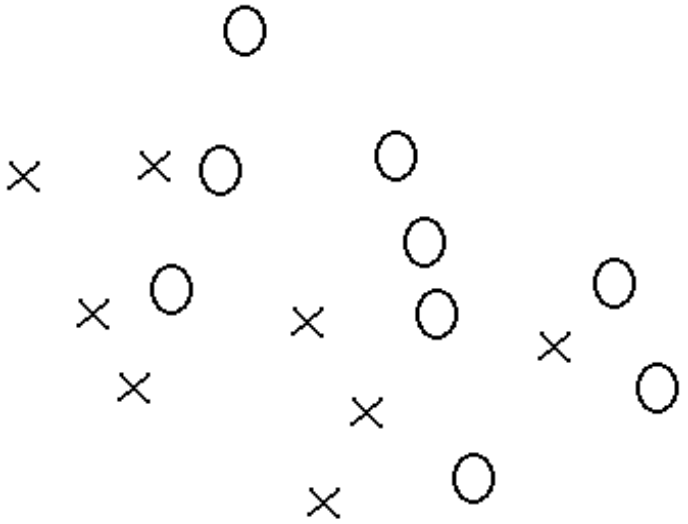
非线性分类

例子:



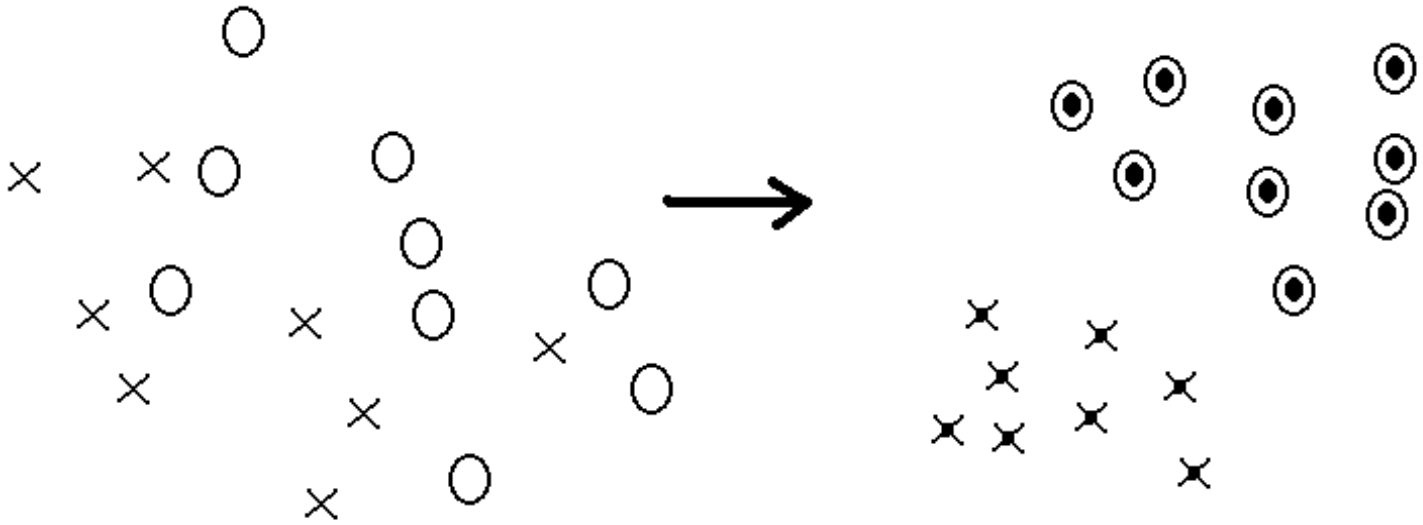
Non-linear Classification

What can we do if the boundary is nonlinear ?



Idea: transform the data vectors to a space where the separator is linear

Non-linear Classification

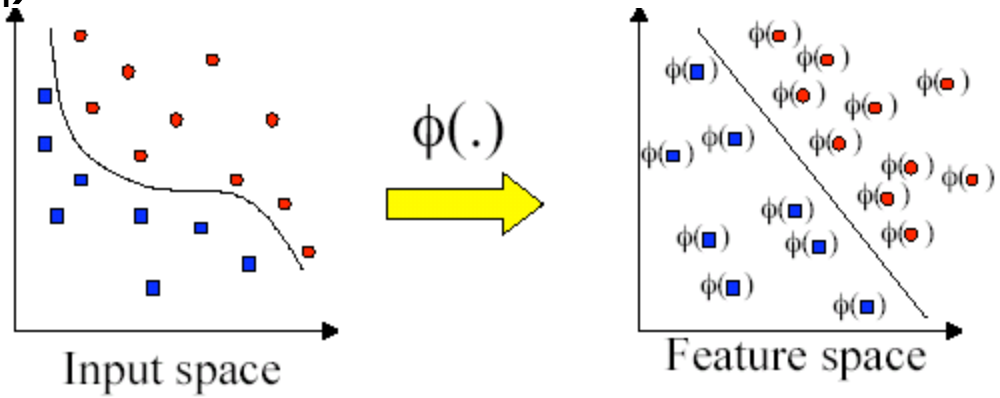


The transformation many times is made to an infinite dimensional space, usually a function space.

Example: $x \rightarrow \cos(u^T x)$

Non-linear SVMs

- Transform $\mathbf{x} \rightarrow \phi(\mathbf{x})$
- The linear algorithm depends only on $\mathbf{x}\mathbf{x}_i$, hence transformed algorithm depends only on $\phi(\mathbf{x})\phi(\mathbf{x}_i)$
- Use kernel function $K(\mathbf{x}_i,\mathbf{x}_j)$ such that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x})\phi(\mathbf{x}_i)$



非线性分类

设训练集 $T=\{(x_i,y_i),i=1,\cdots l\}$，其中 $x_i=([x_i]_1,[x_i]_2)^T, y_i\in\{1,-1\}$
假定可以用 $([x]_1,[x]_2)$ 平面上的二次曲线来分划：

$$[w]_1+2[w]_2[x]_1+2[w]_3[x]_2+2[w]_4[x]_1[x]_2+[w]_5[x]_1^2+[w]_6[x]_2^2+b=0$$

现考虑把2维空间 $x=([x]_1,[x]_2)^T$ 映射到6维空间的变换

$$\phi(x)=(1,\sqrt{2}[x]_1,\sqrt{2}[x]_2,\sqrt{2}[x]_1[x]_2,[x]_1^2,[x]_2^2)^T \qquad (1)$$

上式可将2维空间上二次曲线映射为6维空间上的一个超平面：

$$[w]_1[X]_1+\sqrt{2}[w]_2[X]_2+\sqrt{2}[w]_3[X]_3+\sqrt{2}[w]_4[X]_4+[w]_5[X]_5+[w]_6[X]_6+b=0$$

非线性分类

可见，只要利用变换，把 x 所在的2维空间的两类输入点映射到 X 所在的6维空间，然后在这个6维空间中，使用线性学习机求出分划超平面：

$$(w^* \cdot x) + b^* = 0, \quad 其中 w^* = ([w^*]_1, \cdots [w^*]_6)^T$$

最后得出原空间中的二次曲线：

$$[w^*]_1 + 2[w^*]_2[x]_1 + 2[w^*]_3[x]_2 + 2[w^*]_4[x]_1[x]_2 + [w^*]_5[x]_1^2 + [w^*]_6[x]_2^2 + b = 0$$

怎样求6维空间中的分划超平面？(线性支持向量分类机)

非线性分类

需要求解的最优化问题

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \left(\phi(x_i)\cdot\phi(x_j)\right) - \sum_{j=1}^{l}\alpha_j$$

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

$$0 \le \alpha_i \le C, i = 1, \cdots l$$

其中

$$\phi(x_i) = (1, \sqrt{2}[x_i]_1, \sqrt{2}[x_i]_2, \sqrt{2}[x_i]_1[x_i]_2, [x_i]_1^2, [x_i]_2^2)^T$$

$$\phi(x_j) = (1, \sqrt{2}[x_j]_1, \sqrt{2}[x_j]_2, \sqrt{2}[x_j]_1[x_j]_2, [x_j]_1^2, [x_j]_2^2)^T$$

$$(\phi(x_i)\cdot\phi(x_j)) = 1 + 2[x_i]_1[x_j]_1 + 2[x_i]_2[x_j]_2 + 2[x_i]_1[x_i]_2[x_j]_1[x_j]_2 + [x_i]_1^2[x_j]_1^2 + [x_i]_2^2[x_j]_2^2 \quad (2)$$

非线性分类

线性分划—>非线性分划

代价：2维空间内积—>6维空间内积

在求得最优化问题的解 $\alpha^* = (\alpha_1^*, \cdots \alpha_l^*)^T$ 后，得到分划超平面

$$(w^* \cdot x) + b^* = 0$$

其中

$$w^* = \sum_{i=1}^{l} y_i \alpha_i^* \phi(x_i),\ j \in \{j \mid 0 < \alpha_j^* < C\}$$

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

最后得到决策函数

$$f(x) = \mathrm{sgn}((w^* \cdot \phi(x)) + b^*)$$

或

$$f(x) = \mathrm{sgn}(\sum_{i=1}^{l} y_i \alpha_i (\phi(x_i) \cdot \phi(x)) + b^*)$$

非线性分类

为此，引进函数 $K(x_i, x_j) \square ((x_i \cdot x_j)+1)^2$

有 $K(x_i, x_j) = (([x_i]_1[x_j]_1 + [x_i]_2[x_j]_2 + 1)^2$

$$= [x_i]_1^2[x_j]_1^2 + [x_i]_2^2[x_j]_2^2 + 1 + 2[x_i]_1[x_j]_1[x_i]_2[x_j]_2$$

$$+ 2[x_i]_1[x_j]_1 + 2[x_i]_2[x_j]_2 \qquad (3)$$

比较(2)和(3)，可以发现

$$(\phi(x_i) \cdot \phi(x_j)) = K(x_i, x_j) = ((x_i \cdot x_j)+1)^2$$

这是一个重要的等式，提示6维空间中的内积 $(\phi(x_i) \cdot \phi(x_j))$ 可以通过计算 $K(x_i, x_j)$ 中2维空间中的内积 $(x_i \cdot x_j)$ 得到。

实现非线性分类的思想

给定训练集后，决策函数仅依赖于 $K(x_i, x_j) = ((x_i \cdot x_j)+1)^2$

而不需要再考虑非线性变换 $\phi(x)$

如果想用其它的非线性分划办法，则可以考虑选择其它形式的函数 $K(x_i, x_j)$，一旦选定了函数，就可以求解最优化问题

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K\left(x_i, x_j\right) - \sum_{j=1}^{l} \alpha_j$$

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

$$0 \le \alpha_i \le C, i = 1, \cdots l$$

得 $\alpha^* = (\alpha_1^*, \cdots \alpha_l^*)^T$，而决策函数

实现非线性分类的思想

决策函数 $$f(x)=\mathrm{sgn}(\sum_{i=1}^{l} y_i\alpha_i K(x_i,x))+b^*)$$

其中 $$b^*=y_j-\sum_{i=1}^{l} y_i\alpha_i K(x_i,x_j) \quad j\in\{j\,|\,0<\alpha_j^*<C\}$$

$K(x_i,x)$ — 核函数

核函数(核或正定核)定义

设χ 是 R^n 中的一个子集。称定义在$\chi\times\chi$ 上的函数$K(x,x')$是核函数(正定核或核)，如果存在着从 χ 到某一个*Hilbert*空间 H 的映射

$$\phi: \begin{array}{c} \chi \to \mathrm{H} \\ x \mapsto \phi(x) \end{array}$$

使得 $$K(x,x') = (\phi(x)\cdot\phi(x'))$$

其中 $(\cdot)$ 表示 *Hilbert*中的内积

核函数的选择

目前研究最多的核函数主要有三类：

- 多项式内核

$$K(x, x_i) = [(x \cdot x_i) + c]^q$$ 得到q 阶多项式分类器

- 径向基函数内核RBF

$$K(x, x_i) = \exp\{-\frac{|x - x_i|^2}{\sigma^2}\}$$

每个基函数中心对应一个支持向量，它们及输出权值由算法自动确定

- Sigmoind内核

$$K(x, x_i) = \tanh(\upsilon(x \cdot x_i) + c)$$

包含一个隐层的多层感知器，隐层节点数是由算法自动确定

多项式内核

$$K(x, y) = \langle x \cdot y \rangle^d$$

– The kind of kernel represents the inner product of two vector(point) in a feature space of $\binom{n+d-1}{d}$ dimension.

– For example

$$(\mathbf{x} \cdot \mathbf{y})^2 = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2)(y_1^2, y_2^2, \sqrt{2}\, y_1 y_2)^{\mathsf{T}} = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})),$$

SVM寻优算法

传统的利用二次型优化技术解决对偶问题时：

■ 需要计算存储核函数矩阵。当样本点数较大时，需要很大的存储空间。例如：当样本点超过4000时，存储核函数矩阵就需要多达128兆内存；

■ SVM在二次型寻优过程中要进行大量的矩阵运算，通常寻优算法占用了算法时间的主要部分。

—

Edgar Osuna(Cambridge,MA)等人在IEEE NNSP'97发表了An Improved Training Algorithm for Support Vector Machines ,提出了SVM的分解算法，即将原问题分解为若干个子问题，按照某种迭代策略，通过反复求解子问题，最终使得结果收敛于原问题的最优解。

SVM寻优算法

根据子问题的划分和迭代策略的不同，大致分为：

1. 块算法(Chunking Algorithm)：

考虑去掉Lagrange乘子等于零的训练样本不会影响原问题的解，采用一部分样本构成工作样本集进行训练，移除其中的非支持向量，并把训练结果对剩余样本进行检验，将不符合KKT条件的样本与本次结果的支持向量合并成为一个新的工作集。然后重新训练，如此重复获得最优结果。

例如：基于这种思路的 *SMO*算法。

SVM寻优算法

块算法(Chunking Algorithm)：

SMO使用了块与分解技术，而SMO算法则将分解算法思想推向极致，每次迭代仅优化两个点的最小子集，其威力在于两个数据点的优化问题可以获得解析解，从而不需要将二次规划优化算法作为算法一部分。尽管需要更多的迭代才收敛，但每个迭代需要很少的操作，因此算法在整体上的速度有数量级的提高。另外，算法其他的特征是没有矩阵操作，不需要在内存中存储核矩阵。

SVM寻优算法

SMO算法每次迭代时，在可行的区域内选择两点，最大化目标函数，从而优化两个点的最小子集。无论何时，当一个乘子被更新时，调整另一个乘子来保证线性约束条件成立，保证解不离开可行区域。每步SMO选择两个参数优化，其他参数固定，可以获得解析解。尽管需要更多的迭代才收敛，但每个迭代需要很少的操作，因此算法在整体上的速度有数量级的提高。另外，算法其他的特征是没有矩阵操作，不需要在内存中存储核矩阵。

SVM寻优算法

类别名称	测试样本数	错误分类数	准确度（%）
政治	146	4	97.26
军事	83	0	100
经济	137	3	97.81
法律	32	2	93.75
农业	106	2	98.11
体育	90	1	98.89
卫生	34	1	97.06
工业	87	2	97.70
科技	111	2	98.20
交通	40	1	97.50
生活	91	1	98.90
宗教	3	0	100
天气	24	2	91.67
合计	984	21	97.87

SVM寻优算法

SMO算法核缓存算法

SMO算法在每次迭代只选择两个样本向量优化目标函数，不需要核矩阵。虽然没有核矩阵操作，但仍需要计算被选向量和训练集中所有样本向量的核函数，计算次数为2n（n为训练集中的样本数）。如果训练集中的样本选取有误，在噪声比较多的情况下，收敛会很慢，迭代次数很多，则核函数的计算量也是非常可观的，SMO 算法的优点就完成失去了。同时，考虑到文本分类的文本向量一般维数比较大，核函数的计算将会非常耗时，尤其在高价多项式核和高斯核等核函数的计算中表现更加明显。

SVM寻优算法

SMO算法核缓存算法

在内存中为SMO算法核函数开辟n行m列的核矩阵空间。其中：n为训练集中的样本数；m是为可调节参数，根据实际的内存大小进行调整，每列存放训练集中某个样本向量与训练集中所有样本向量的核函数计算结果列表。在核矩阵列头生成m个节点的双向循环链表队列，每个节点指向核矩阵的列，通过双向循环链表队列实现核矩阵中的核函数列唤入唤出操作。同时，为了实现样本向量的核函数列的快速查找，为每个训练样本向量设计了快速索引列表，通过索引列表判断该训练样本向量的核函数列是否在核矩阵中，并确定在哪一列。

SVM寻优算法

- 选择一个训练集，通过调整核缓冲参数的大小，记录不同核缓存大小情况下训练时间，结果如下表:

核缓存大小(Mb)	训练样本数	核矩阵	迭代次数	训练时间(M:S)
1	5624	5624*23	40726	7:06
10	5624	5624*233	40726	3:50
20	5624	5624*466	40726	2:41
30	5624	5624*699	40726	1:56
40	5624	5624*932	40726	1:29
50	5624	5624*1165	40726	1:23
60	5624	5624*1398	40726	1:08
70	5624	5624*1631	40726	1:05
80	5624	5624*1864	40726	1:04
90	5624	5624*2097	40726	1:07
100	5624	5624*2330	40726	1:37
250	5624	5624*5624	40726	1:12

SVM寻优算法

通过引入核缓存机制，有效的改进了SMO算法，提高了文本分类的训练速度。在核缓存机制中采用简单的hash查找算法和队列FILO算法，有效提高了核矩阵查找和唤入唤出操作的效率。设置核矩阵列参数，通过调节列参数，可以灵活的根据系统运行情况调整训练的时间和空间开销，避免因系统空间开销过大使系统运行效率下降，反而影响训练速度。

SVM寻优算法

活动向量集选择算法

当训练样本数非常大的时候，如果系统能够提供的核缓冲大小很有限，那么能够同时保存在核缓冲中训练样本的核函数数目在训练样本数中所占比例将非常的小。在训练过程中，训练样本在核缓冲中的核函数命中率将显著下降，导致核缓冲中的核函数被频繁的唤入唤出，而每执行一次唤入唤出操作将引起系统重新计算训练样本的核函数，核缓存的作用被很大程度的削弱了。如果出现这样的情况，要么增加系统的存储空间；要么减少训练样本数，才能提高系统的训练速度。为解决训练样本数多，系统内存空间小的矛盾，本文通过活动向量集选择算法，比较好地解决了这个问题。

SVM寻优算法

活动向量集选择算法

- 算法的主要思想是：定期检查训练样本集，在收敛前预先确定训练样本集中一些边界上的点（alpha=0，或者alpha=C）是否以后不再被启发式选择，或者不再被判定为最有可能违例，如果存在这样的点，将它们从训练样本集中剔除出去，减少参加训练的样本数。该算法基于如下的认识：经过多次迭代后，如果样本的拉格朗日乘子一直为0，该点被当前估计的支持向量集所确定的超平面区分得很开，即使以后支持向量集发生变化，该点也不会是最靠近超平面的点，则可以确定该样本不是支持向量；经过多次迭代后，如果样本的拉格朗日乘子一直为非常大的C常数，即使以后支持向量集发生变化，该点也不会远离超平面，则可以确定该样本是上边界处的支持向量

SVM寻优算法

活动向量集选择算法

•这样就可以在SMO算法收敛前，提前将边界上的点从训练样本集中剔除，逐渐缩小参加训练的活动样本集，从而减少SMO算法对核缓存空间的要求，提高训练速度。

•训练开始前，训练活动集样本初始化为全部训练样本。每经过一定次数的迭代（比如迭代1000次），如果算法还没有收敛，应检查活动集中的向量，检查是否有训练样本可以不参加迭代运算。

•检查完当前活动向量集中所有样本后，产生了新的活动向量集。如果新的活动向量集的样本数减少一成以上（含一成），则可以收缩当前活动向量集，用新的活动向量集替换当前活动向量集。当活动向量集的样本数减少到一定的程度，对核缓存空间的要求不是很大的时候，继续减少训练样本对训练速度的提高就非常有限了，这时就没有必要再减少训练样本了。

SVM寻优算法

2. 固定工作样本集 **(Osuna et al.)**：

将工作样本集的大小固定在算法速度可以容忍的限度内，迭代过程选择一种合适的换入换出策略，将剩余样本中的一部分与工作样本集中的样本进行等量交换，即使支持向量的个数超过工作样本集的大小，也不改变工作样本集的规模，而只对支持向量中的一部分进行优化。

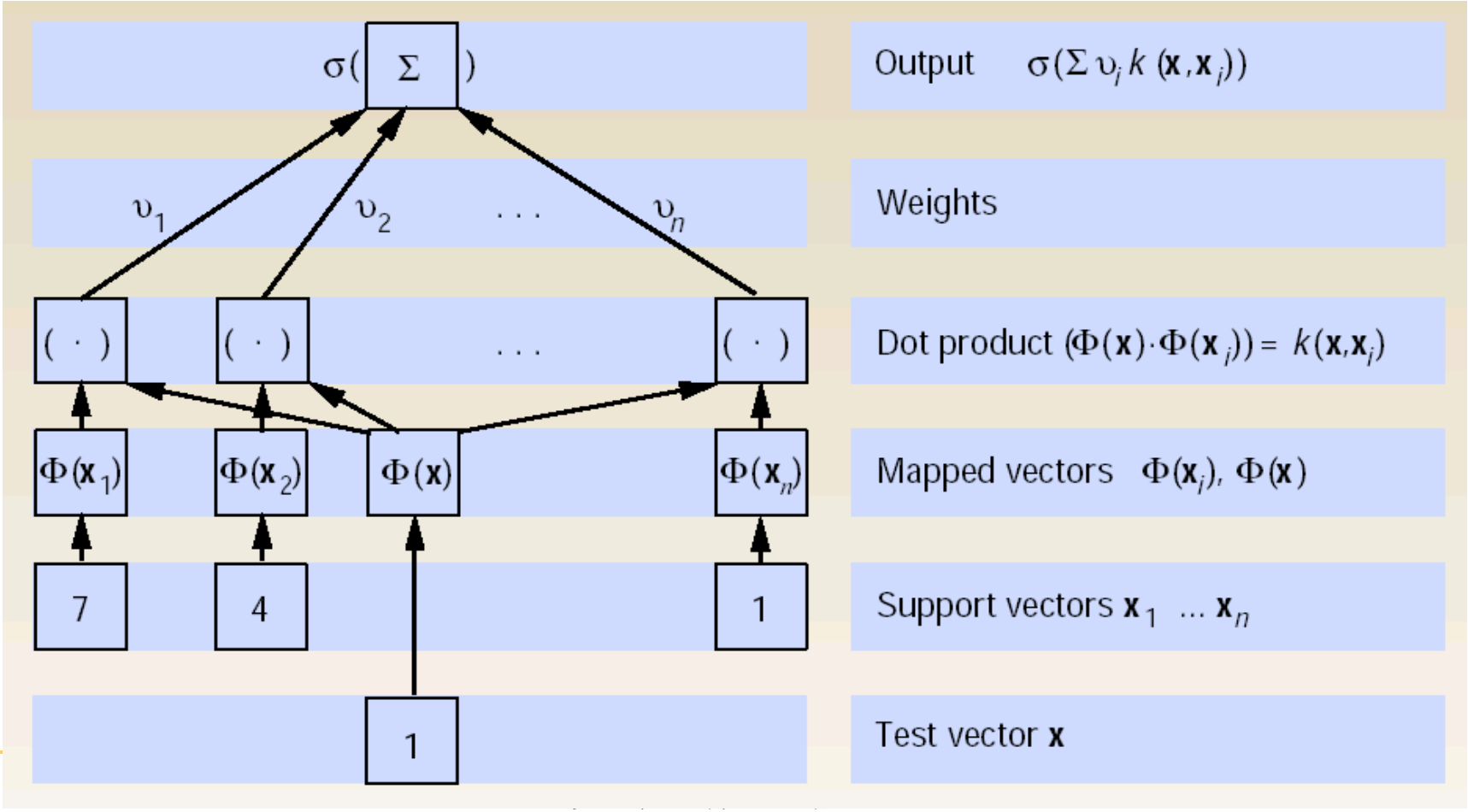
例如：*SVM* ^{Light} 算法

SVM applications

- Pattern recognition
  - o Features: words counts
- DNA array expression data analysis
  - o Features: expr. levels in diff. conditions
- Protein classification
  - o Features: AA composition

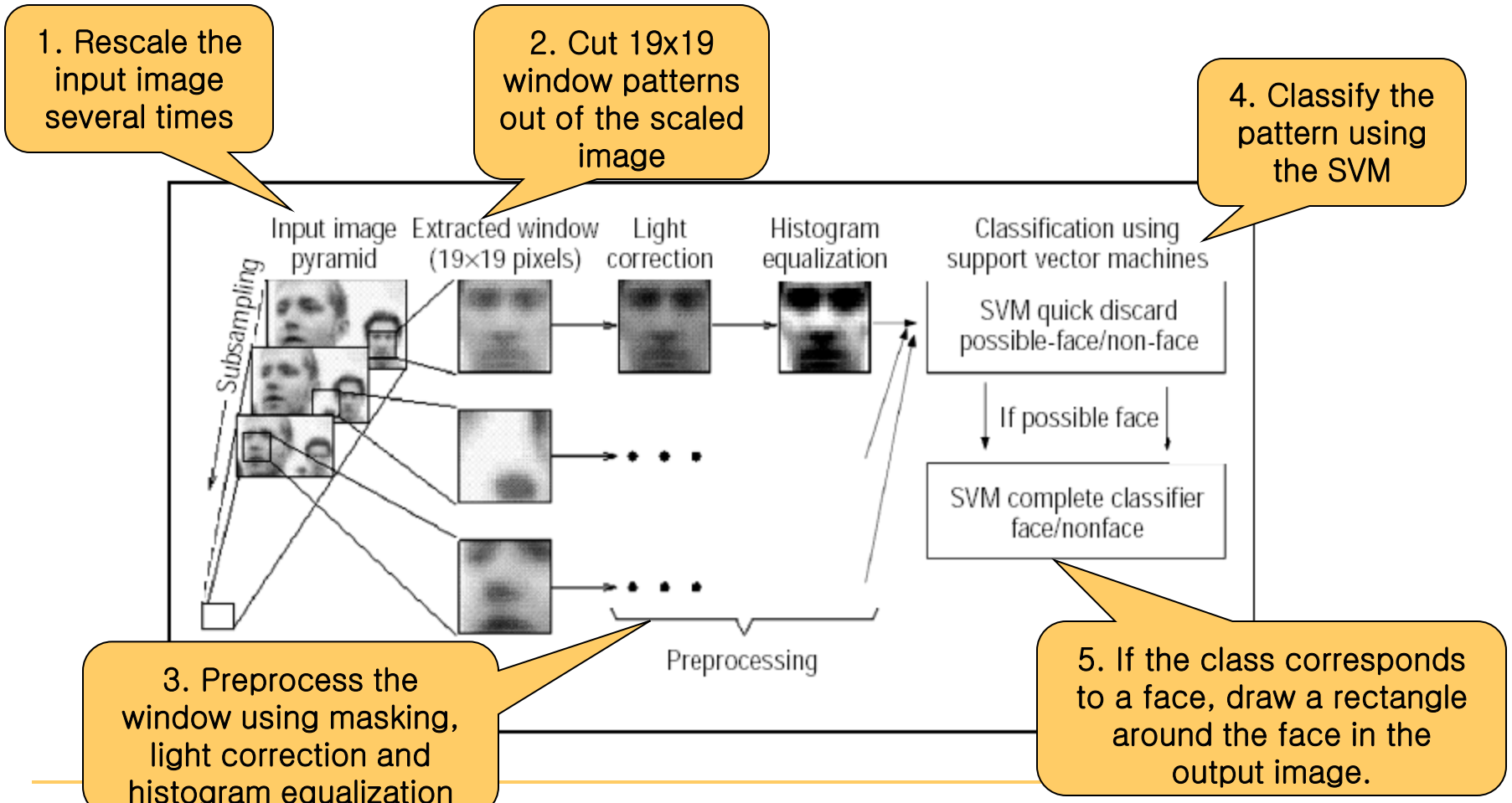
Handwritten Digits Recognition

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{\ell} v_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b\right).$$



Applying SVMs to Face Detection

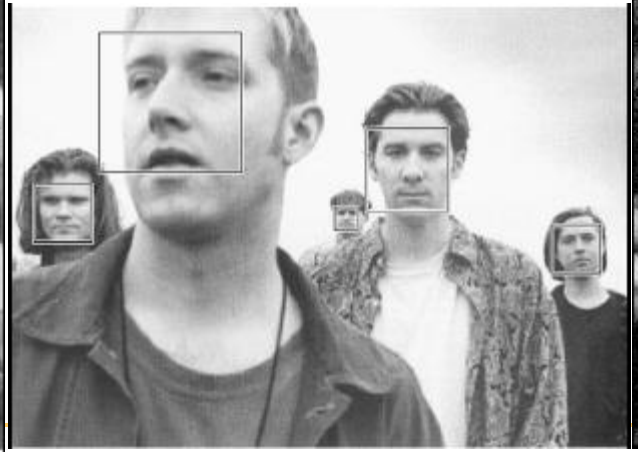
- The SVM face-detection system



Applying SVMs to Face Detection

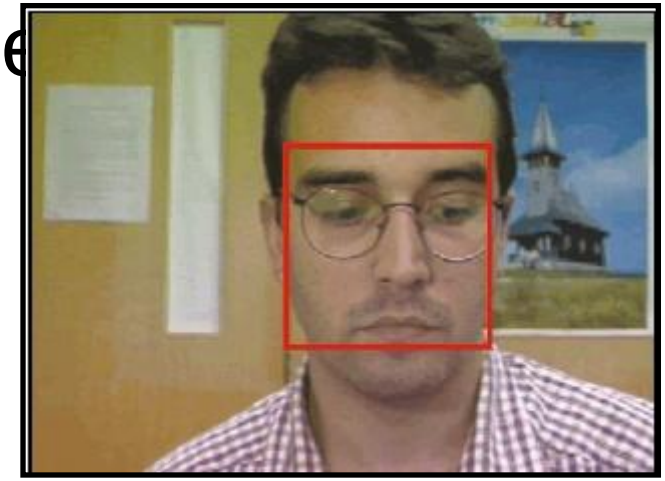
- Experimental results on static images
  - Set A: 313 high-quality, same number of faces

	Test set A		Test set B	
	Detect rate (%)	False alarms	Detect rate (%)	False alarms
SVM	97.1	4	74.2	20
Sung	94.6	2	74.2	11

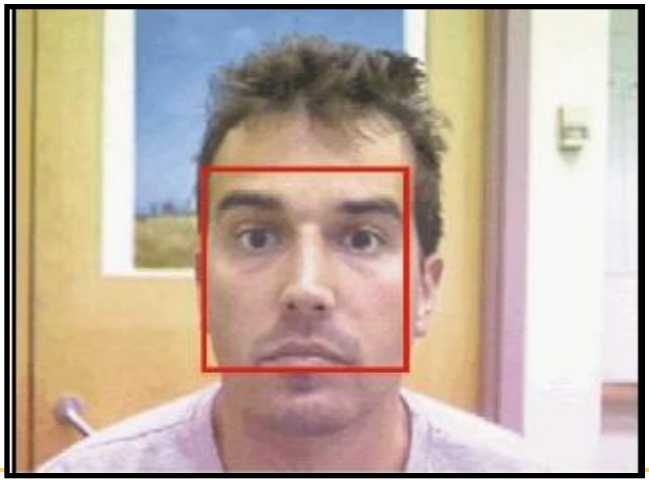


Applying SVMs to Face Detection

An example of the skin detection module implemented using SVMs



Face Detection on the PC-based Color Real Time System



Thank You

Intelligence Science

http://www.intsci.ac.cn/