Pattern Recognition

Tree-based methods
-Decision Tress Basics

Liang Wang

Center for Research on Intelligent Perception and Computing

Institute of Automation, Chinese Academy of Sciences

Recap of SVM

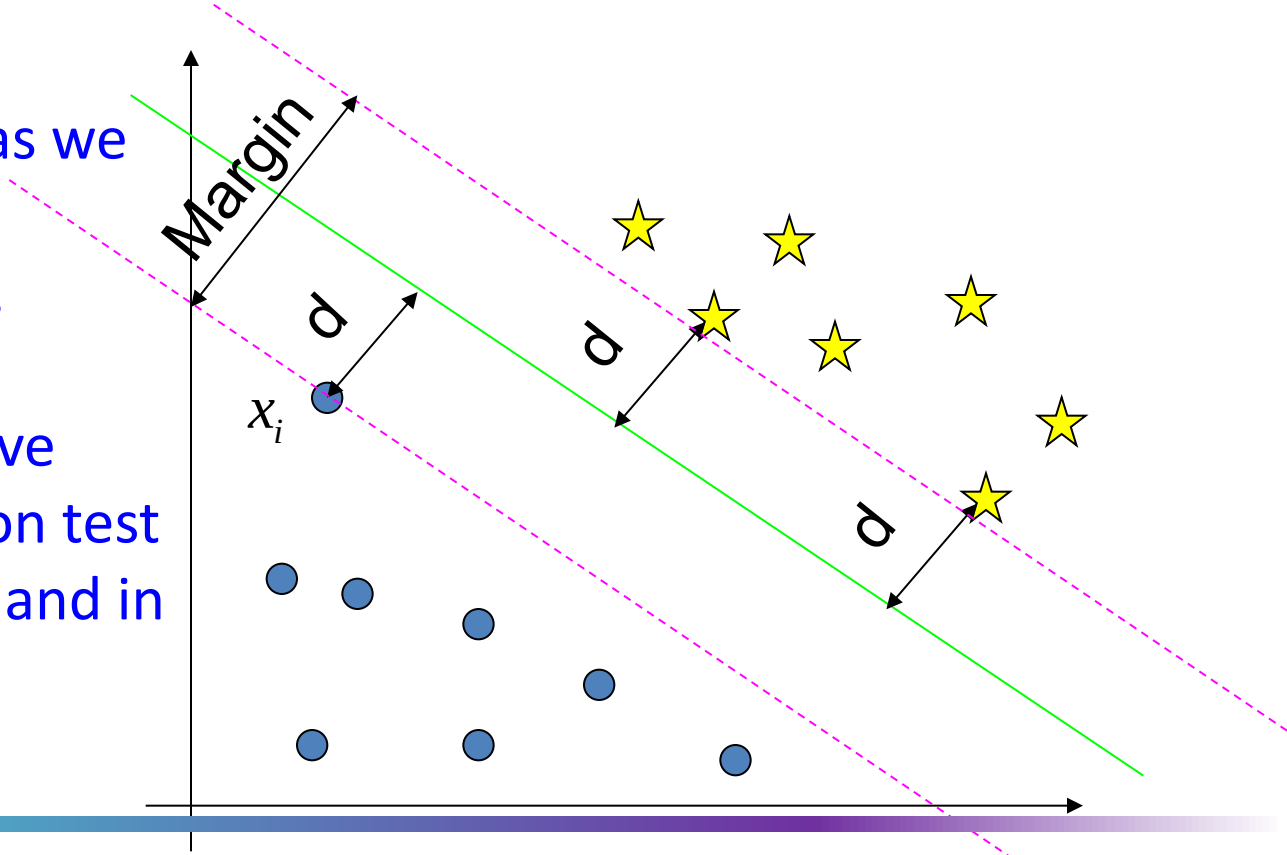
Choosing a separating hyperplane.
The SVM approach: Linear separable case

SVM's goal is to maximize the Margin which is twice the distance "d" between the separating hyperplane and the closest sample.

Why it is the best?

-Robust to outliners as we saw and thus strong generalization ability.

-It proved itself to have better performance on test data in both practice and in theory.



SVM : Linear separable case.
Finding the optimal hyperplane

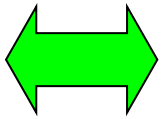
To find the optimal separating hyperplane , SVM aims to maximize the margin:

- Maximize $m = \frac{2}{\|w\|}$

such that:

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



Minimize $\frac{1}{2}\|w\|^2$

such that:

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

We transformed the problem into a form that can be efficiently solved. We got an optimization problem with a convex quadratic objective with only linear constrains and always has a single global minimum.

SVM : Linear separable case.
The optimization problem

So our primal optimization problem now:

$$\text{minimize} \quad L_p(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (x_i \cdot w + b) - 1)$$

$$s.t \quad \alpha_i \geq 0$$

We start solving this problem:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

SVM : Linear separable case.
Inroducing The Legrangin Dual Problem.

By substituting the above results in the primal problem and doing some math manipulation we get:

Lagrangian Dual Problem:

$$\text{maximaize } L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=0}^{n} \sum_{j=0}^{n} \alpha_i \alpha_j y_i y_j x_i^t x_j$$

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

$\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ are now our variables, one for each sample point x_i .

SVM : Linear separable case.
Finding “w” and “b” for the boundary $w^t x + b$

Using the KKT (Karush-Kuhn-Tucker) condition:

$$\forall i \qquad \alpha_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0$$

-We can calculate “b” by taking “ i” such that $\alpha_i > 0$:

Must be $y_i (w^t x_i + b) - 1 = 0 \Rightarrow b = \frac{1}{y_i} - w^t x_i = \underline{y_i - w^t x_i} \quad (y_i \in \{1, -1\})$

-Calculating “w” will be done using what we have found above : $w = \sum_i \alpha_i y_i x_i$

-Usually ,Many of the α_i are zero so the calculation of “w” has a low complexity.

Non Linear SVM:
Mapping the data to higher dimension cont'

-To solve a non linear classification problem with a linear classifier all we have to do is to substitute $\phi(x)$

Instead of x everywhere where x appears in the optimization problem:

$$\text{maximize } L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^t x_j \quad s.t\ \alpha_i \geq 0 \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

Now it will be:

$$\text{maximize } L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i^t) \phi(x_j) \quad s.t\ \alpha_i \geq 0 \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

The decision function will be: $g(x) = f(\phi(x)) = sign(w^t \cdot \phi(x) + b)$

The Kernel Trick:

Working in high dimensional space is computationally expensive.

But luckily the kernel trick comes to rescue:

If we look again at the optimization problem:

$$\text{maximize } L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i^t)\phi(x_j) \quad s.t\ \alpha_i \geq 0 \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

And the decision function:

$$f(\phi(x)) = sign(w^t \phi(x) + b) = sign(\sum_{i=1}^{n} \alpha_i y_i \phi(x_i^t)\phi(x) + b)$$

No need to know this mapping explicitly nor do we need to know the dimension of the new space, because we only use the dot product of feature vectors in both the training and test.

The Kernel Trick

A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Now we only need to compute $K(x_i, x_j)$ and we don't need to perform computations in high dimensional space explicitly. This is what is called the Kernel Trick.

Examples of Kernels

-Some common choices (the first two always satisfying Mercer's condition):

-Polynomial kernel $K(x_i, x_j) = (x_i^t x_j + 1)^p$

-Gaussian Radial Basis Function "RBF" (data is lifted to infinite dimension): $K(x_i, x_j) = \exp(-\frac{1}{2\sigma^2}\left\|x_i - x_j\right\|^2)$

-Sigmoidal : $K(x_i, x_j) = \tanh(kx_i \cdot x_j - \delta)$ (it is not a kernel for every k and δ).

-In fact, SVM model using a sigmoid kernel function is equivalent to a two-layer, feed-forward neural network.

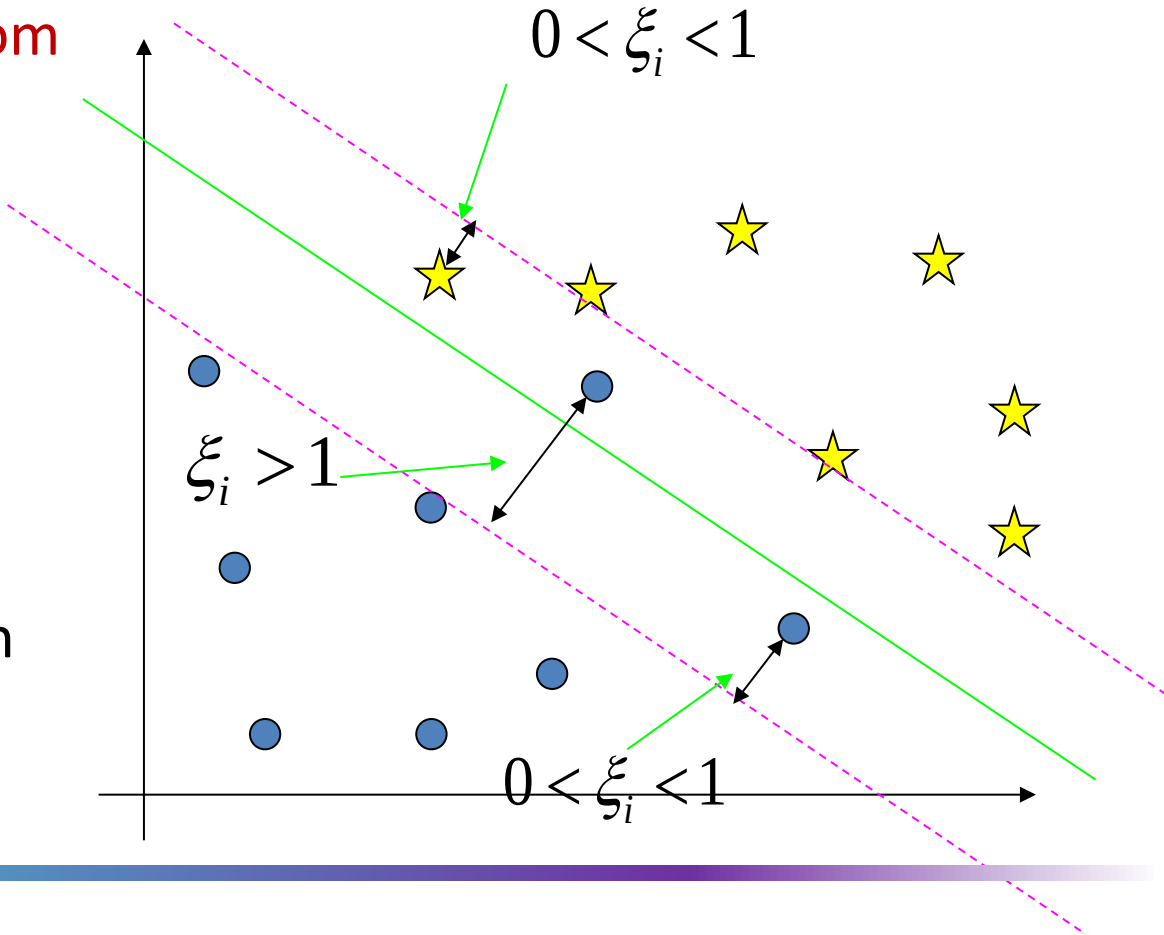
Soft Margin

-We allow "error" ξ_i in classification. We use "slack" variables $\xi_1, \xi_2, \ldots\ldots \xi_n$ (one for each sample).

ξ_i is the deviation error from ideal place for sample i:

-If $0 < \xi_i < 1$ then sample i is on the right side of the hyperplane but within the region of the margin.

-If $\xi_i > 1$ then sample i is on the wrong side of the hyperplane.



Soft Margin: The primal optimization problem

-We change the constrains to $y_i(w^t x_i + b) \geq 1 - \xi_i \quad \forall i \quad \xi_i \geq 0$

instead of $y_i(w^t x_i + b) \geq 1 \quad \forall i$

Our optimization problem now is:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{such that:} \quad y_i(w^t x_i + b) \geq 1 - \xi_i \quad \forall i \quad \xi_i \geq 0$$

$C > 0$ is a constant. It is a kind of penalty on the term $\sum_{i=1}^{n}\xi_i$.

It is a tradeoff between the margin and the training error. It is a way to control overfitting along with the maximum margin approach[1].

Soft Margin: The Dual Formulation.

Our dual optimization problem now is:

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

Such that:

-We can find "w" using : $0 \le \alpha_i \le C \quad \forall i \quad and \quad \sum_{i=1}^{n} \alpha_i y_i = 0$

-To compute "b" we take any $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ and solve for "b".

$$\alpha_i [y_i (w^t x_i + b) - 1] = 0 \qquad 0 < \alpha_i < C$$

$$\alpha_i = 0 \Rightarrow y_i (w^T x_i + b) > 1$$

$$0 < \alpha_i < C \Rightarrow y_i (w^T x_i + b) = 1$$

$$\alpha_i = C \Rightarrow y_i (w^T x_i + b) < 1 \quad (\text{points } with\ \xi_i > 0)$$

Which value for "C" should we choose?

Soft Margin: The “C” Problem

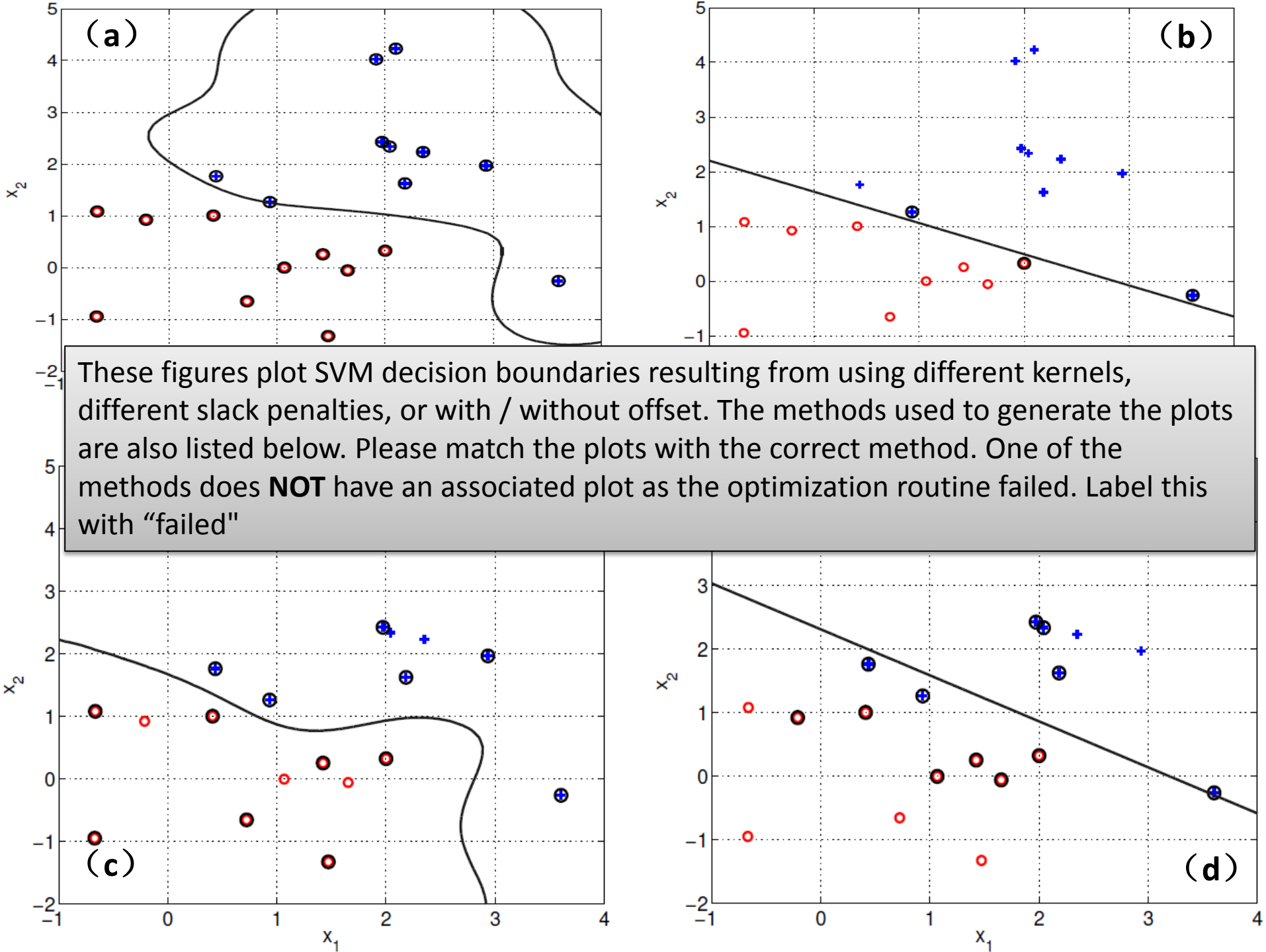
-“C” plays a major role in controlling “ overfitting”.

-Finding the “Right” value for “C” is one of the major problems of SVM:

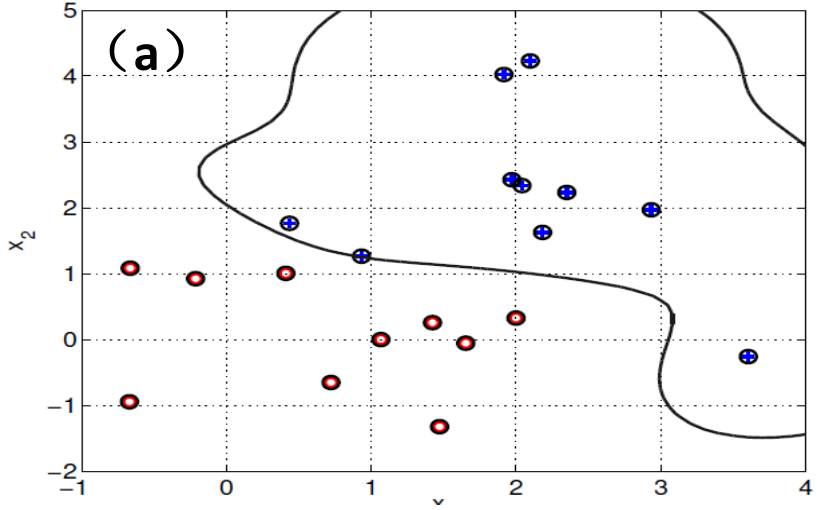
-Larger C → less training samples that are not in ideal position (which means less training error that affects positively the Classification Performance (CP)) But smaller margin (affects negatively the (CP)).C large enough may lead us to overffiting (too much complicated classifier that fits only the training set)

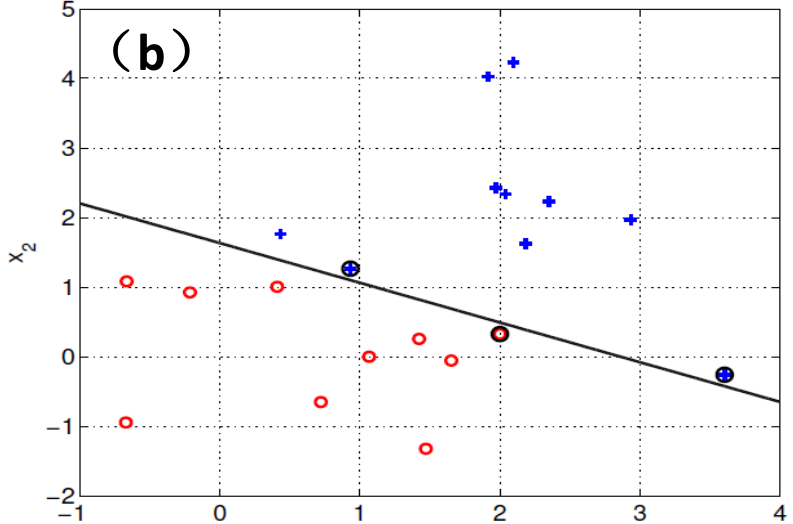
-Smaller C → more training samples that are not in ideal position (which means more training error that affects negatively the Classification Performance (CP)) But larger Margin (good for (CP)). C small enough may lead to underffiting (naïve classifier)

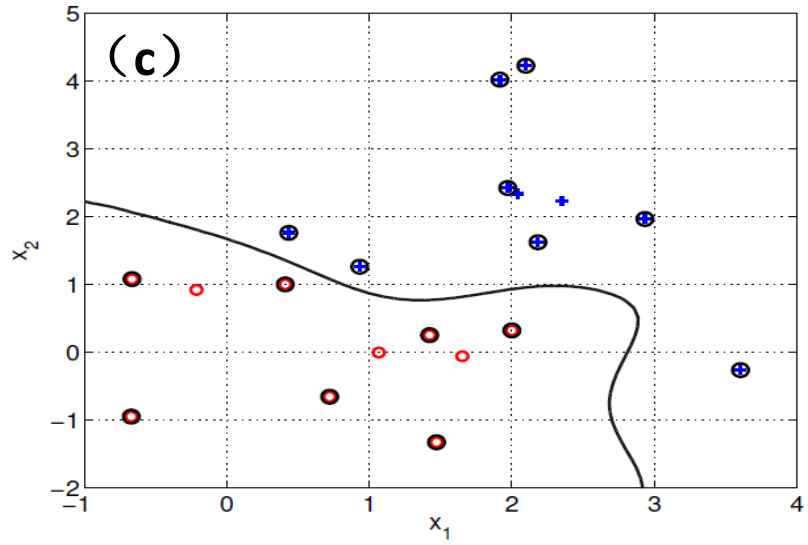
Now try to solve the following SVM problem

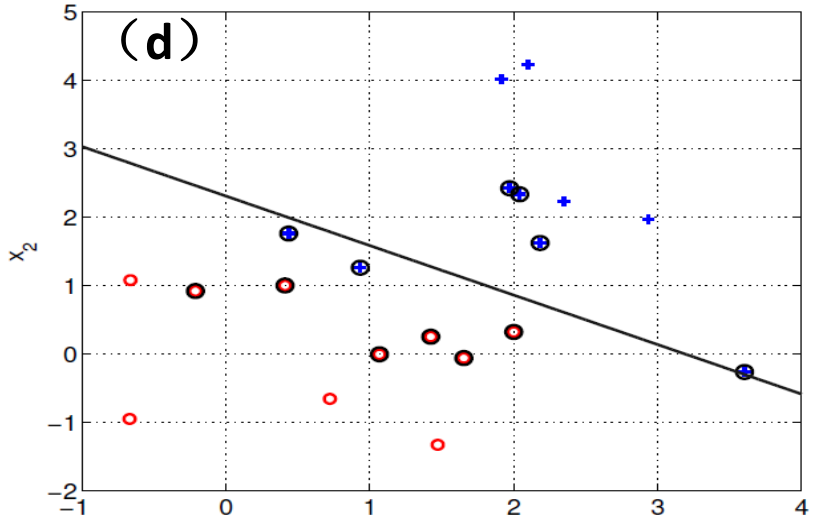


These figures plot SVM decision boundaries resulting from using different kernels, different slack penalties, or with / without offset. The methods used to generate the plots are also listed below. Please match the plots with the correct method. One of the methods does **NOT** have an associated plot as the optimization routine failed. Label this with “failed"







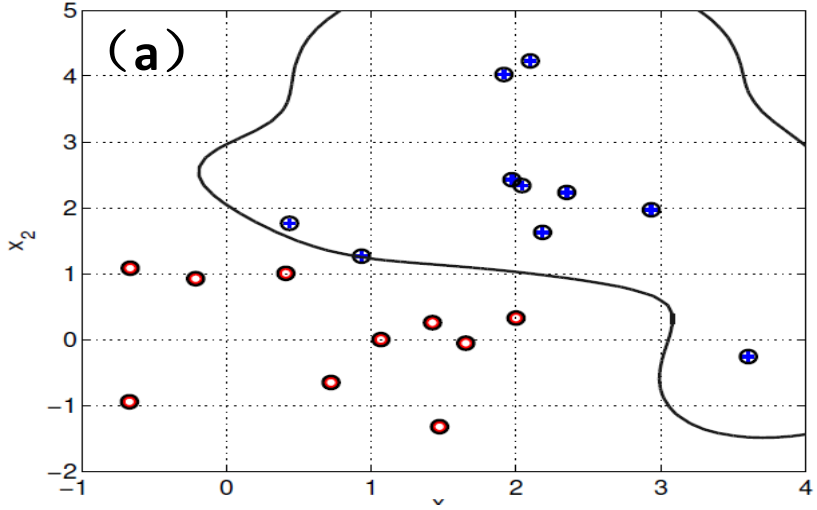


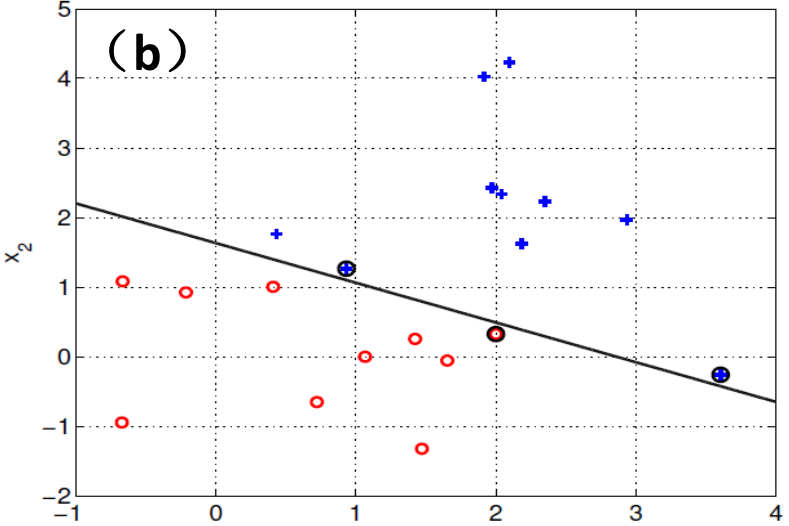
$$\min \frac{1}{2} || w ||_2^2 + C \sum_{i=1}^{m} \xi_i$$

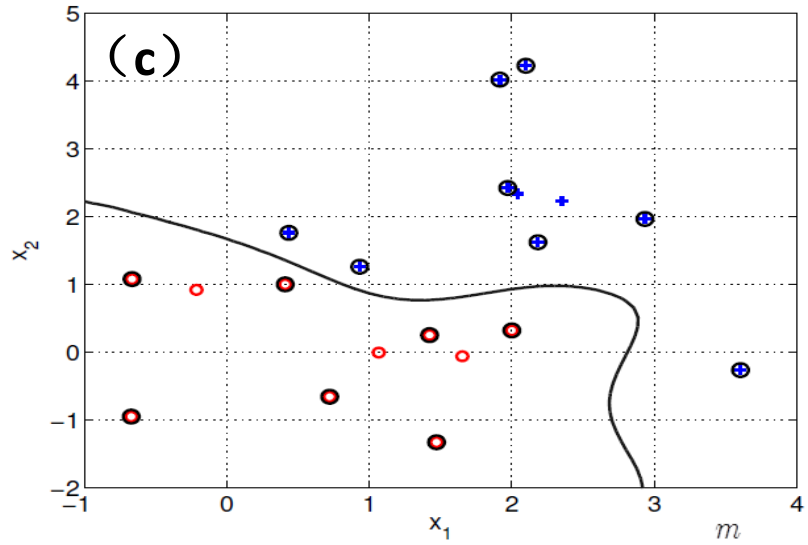
$$s.t.\ \xi_i \geq 0,\ y_i(w^T x_i + b) \geq 1 - \xi_i,\ i = 1, 2, \ldots, m$$

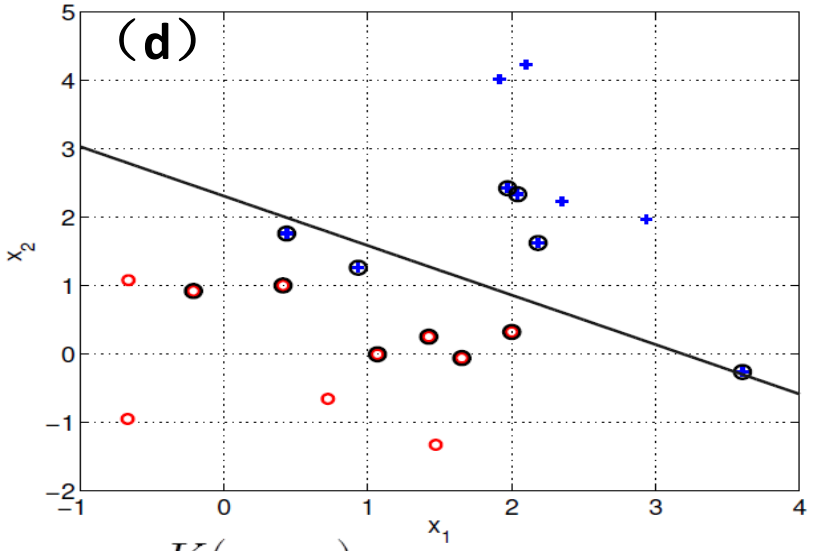
$$where\ C = 0.1$$

The above method matches: **d** (Choose from Plot a, b, c, d or "failed")







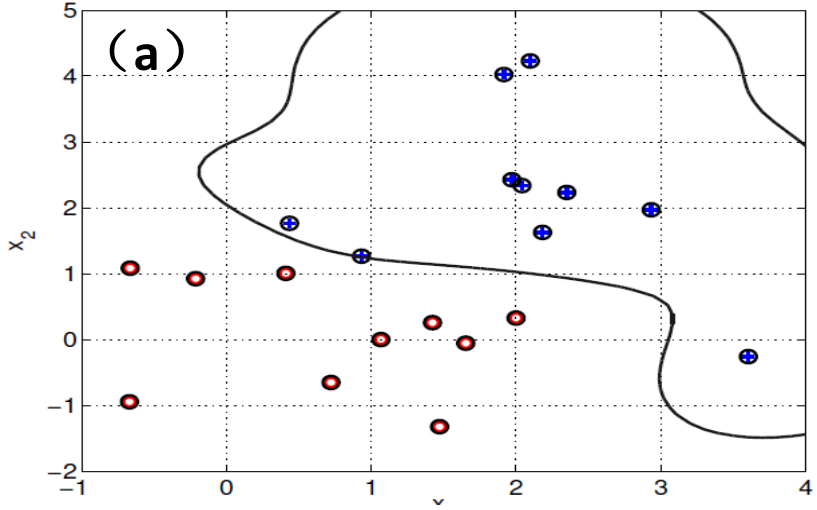


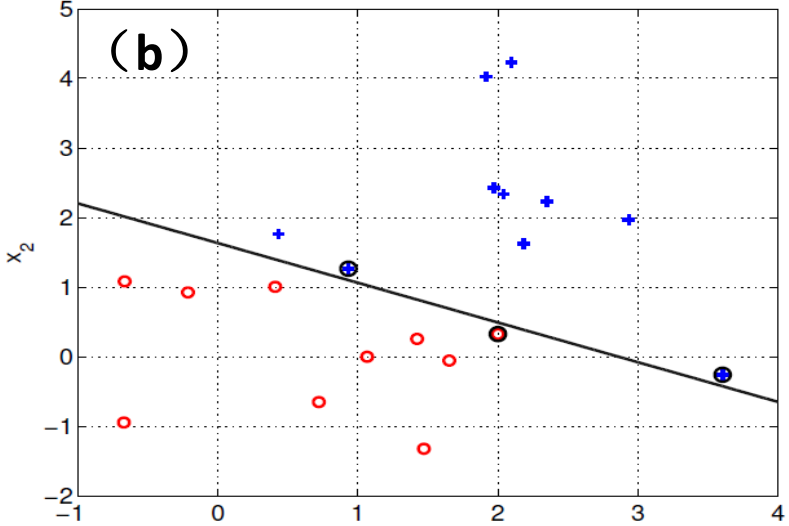
$$\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

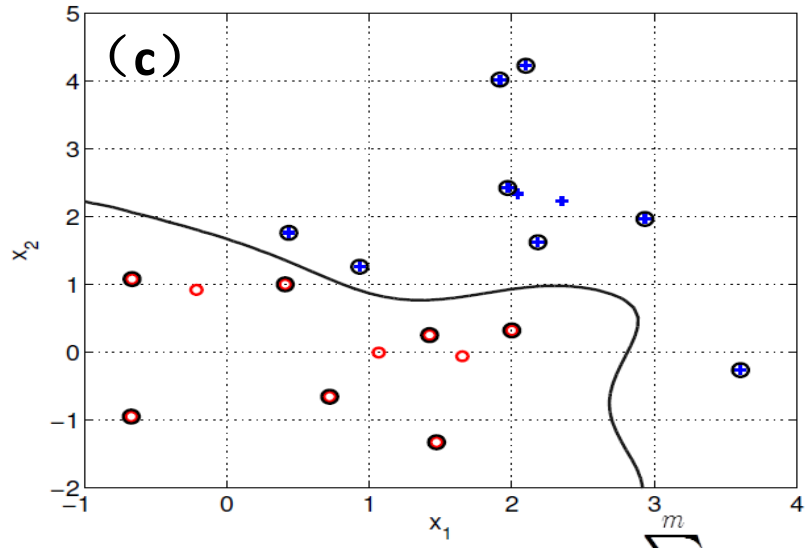
$$s.t. \;\; \alpha_i \geq 0, \;\; i = 1, 2, \ldots, m$$

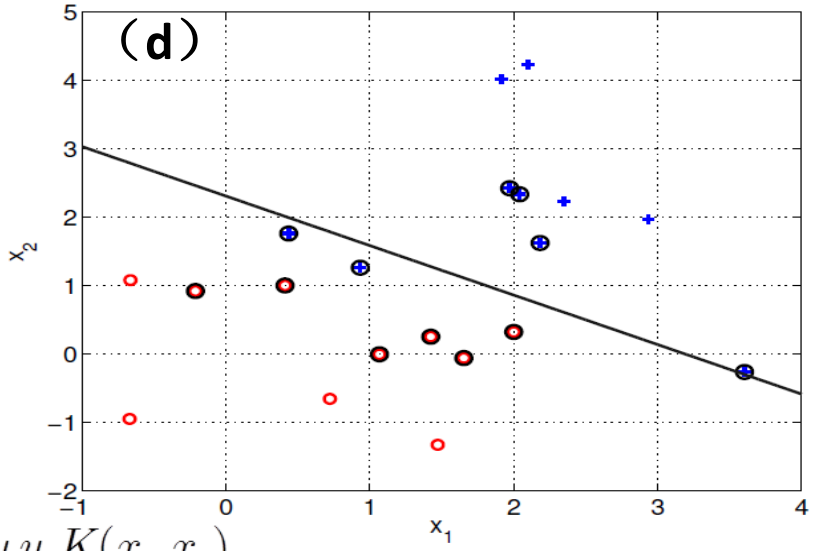
$$where \; K(x_i, x_j) = x_i^T x_j$$

The above method matches: **failed** (Choose from Plot a, b, c, d or "failed")







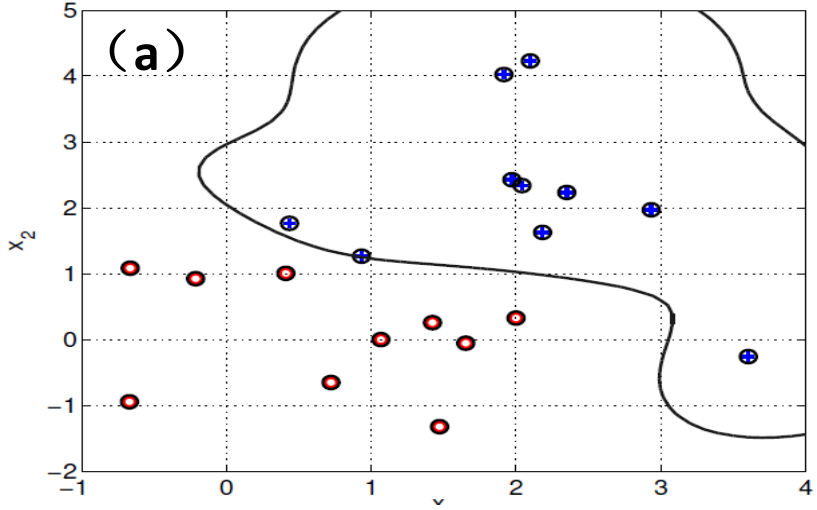


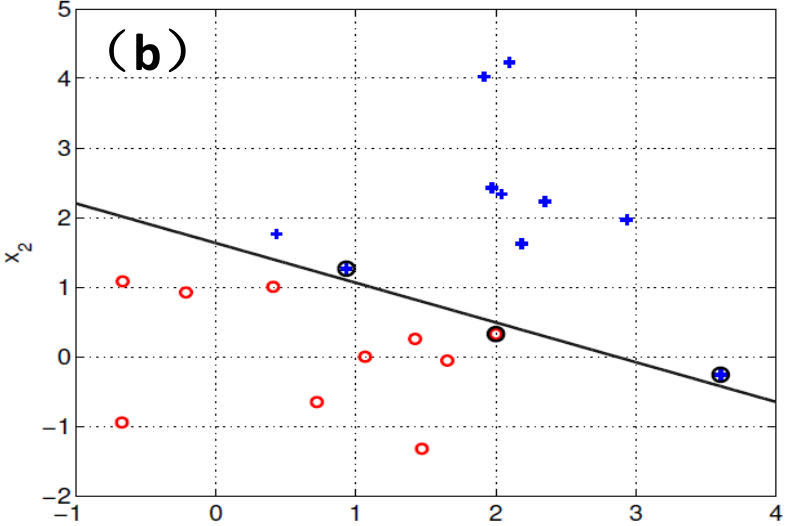
$$\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

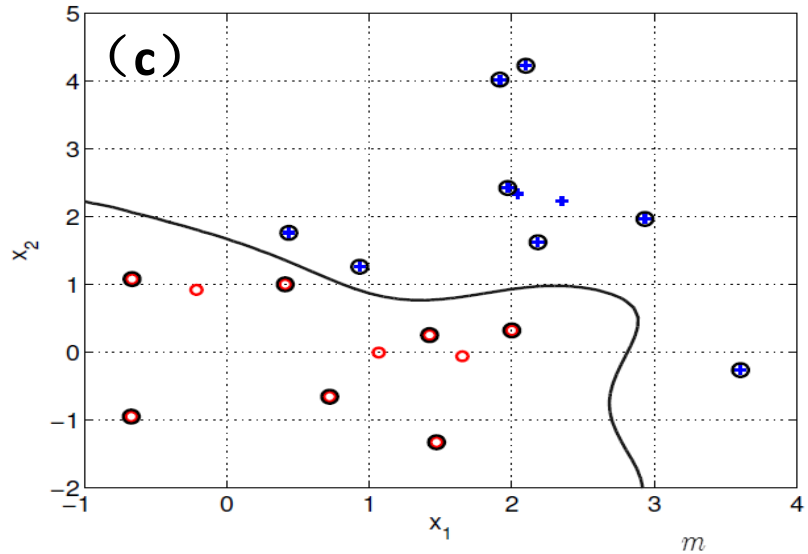
$$s.t.\ \ \alpha_i \geq 0,\ \ i = 1, 2, \ldots, m, \sum_{i=1}^{m} \alpha_i y_i = 0$$

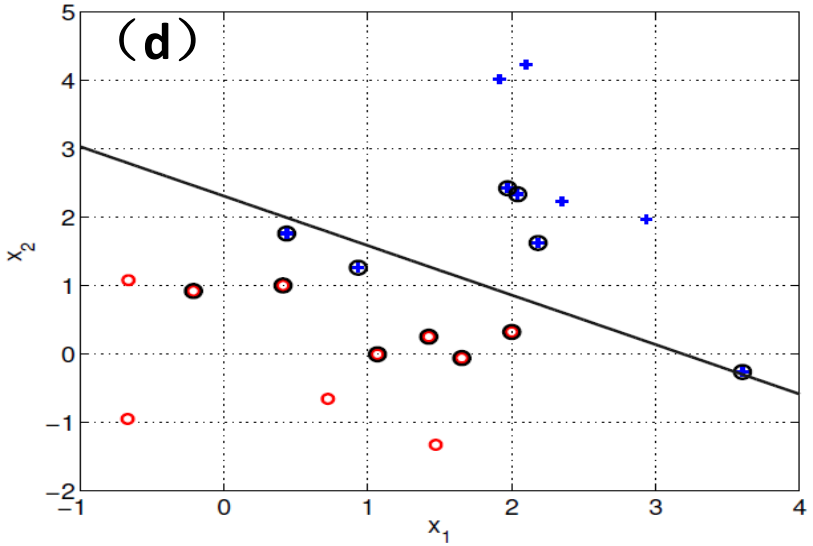
$$where\ K(x_i, x_j) = x_i^T x_j$$

The above method matches: b (Choose from Plot a, b, c, d or "failed")







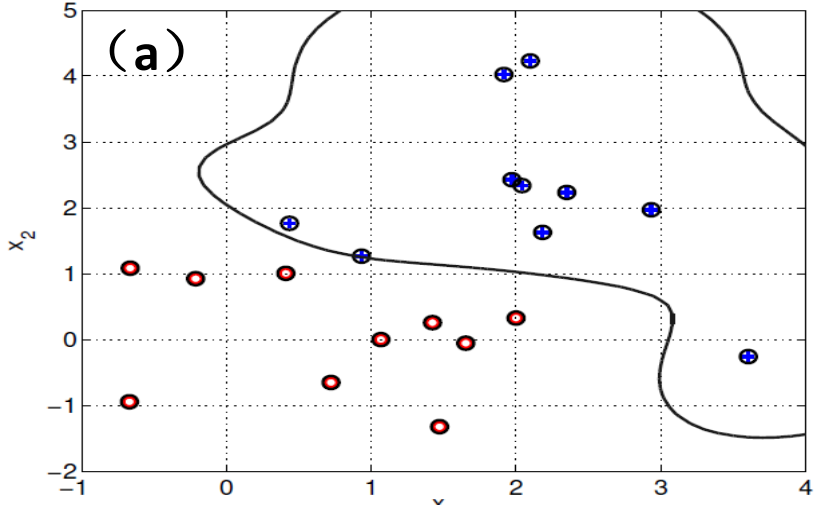


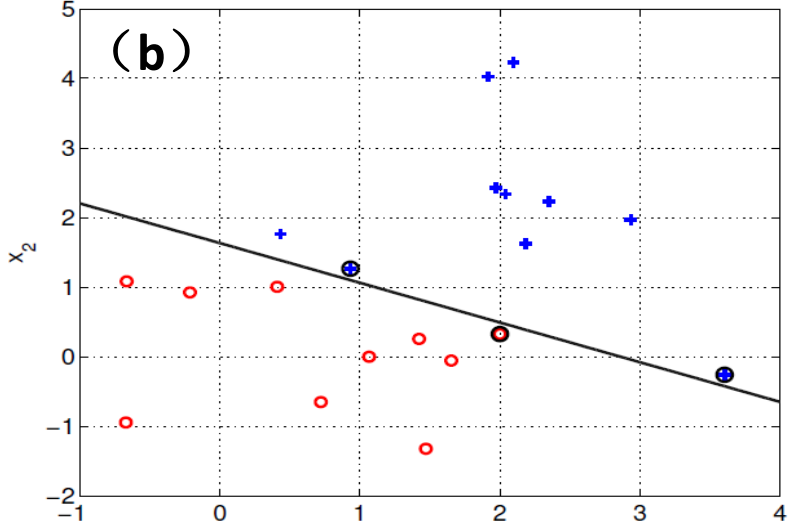
$$\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

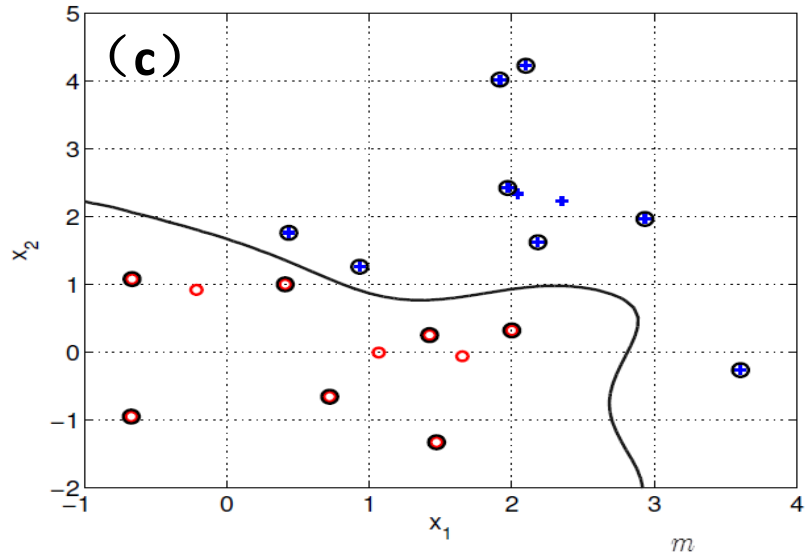
$$s.t. \;\; \alpha_i \geq 0, \;\; i = 1, 2, \ldots, m,$$

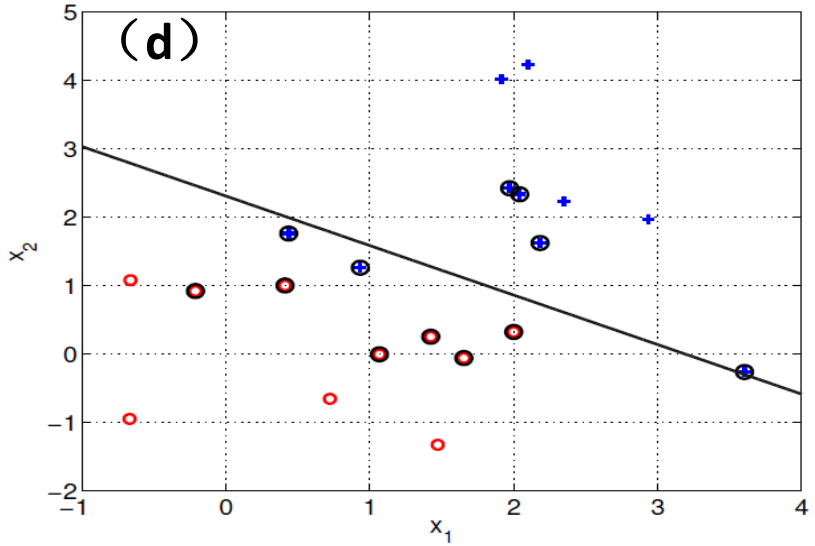
$$where \; K(x_i, x_j) = \exp(- || x_i - x_j ||^2)$$

The above method matches: c (Choose from Plot a, b, c, d or "failed")









$$\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$s.t.\ \ 0.1 \geq \alpha_i \geq 0, \quad i = 1, 2, \ldots, m,$$

$$where\ K(x_i, x_j) = \exp(- \mid\mid x_i - x_j \mid\mid^2)$$

The above method matches: **a** (Choose from Plot a, b, c, d or “failed”)

Provide two reasons for why the method you had linked to Figure (a) could have generated the figure.

1. The complex, enclosing shape of the decision boundary suggests the RBF kernel was used.

2. The upper bound on each α_i in method 5 corresponds to permitting slack in the primal formulation. In contrast to Figure (c), we can tell that slack is used in Figure (a) because:
there are more support vectors.
there is a training example lying essentially on the decision boundary.

As discussed above, one of the methods (1~5) failed because the optimization routine couldn't find a solution. Briefly explain why this method failed on this data.

Method 2 attempts to find a linear classifier through the origin with no slack permitted. The optimization failed because the data are not separable with such a classifier.

Roadmap for tree-based methods

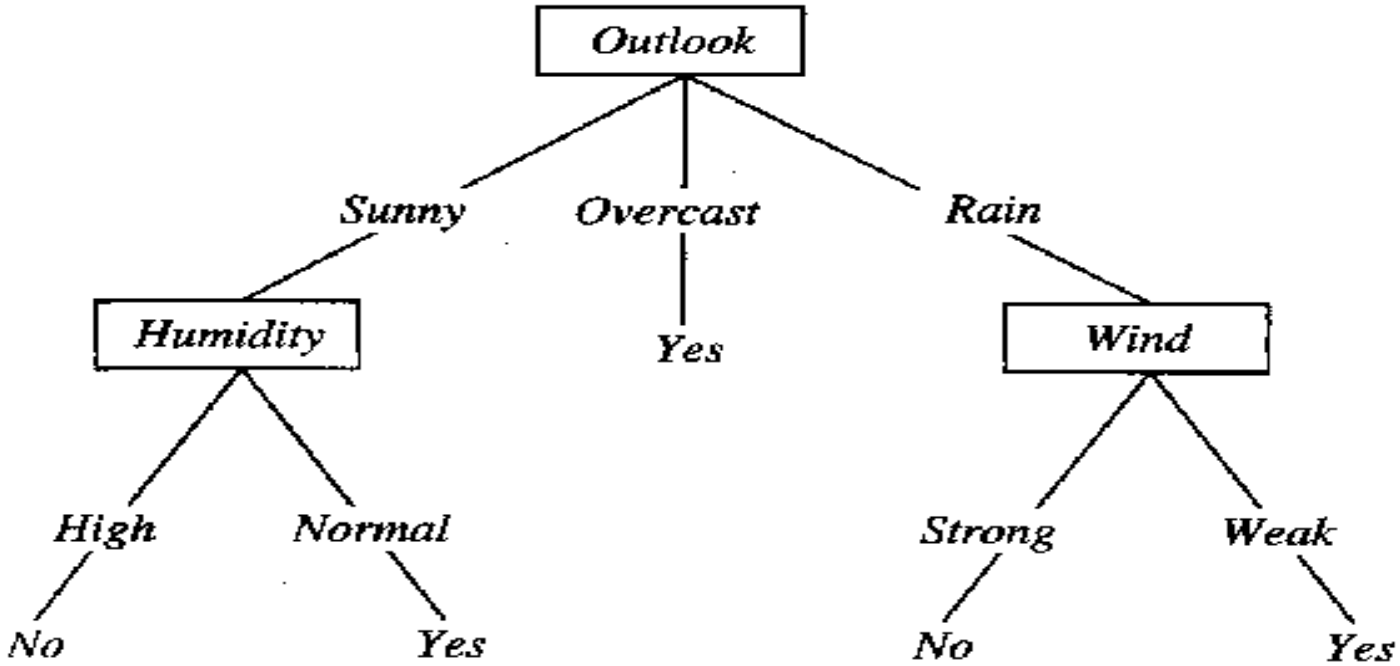
- Decision Tree basics (Today)
- Random Forests (RF)

决策树的一些基本概念

决策树（Decision Tree）

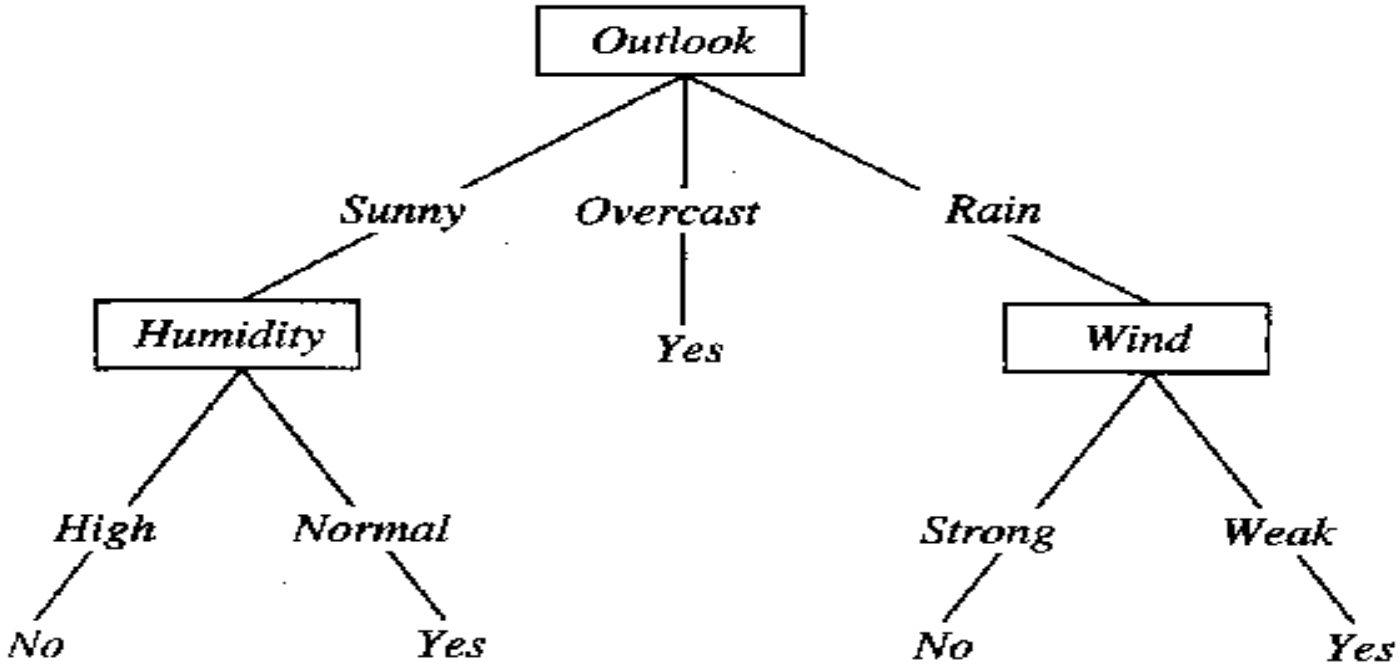
A decision tree is a tree in which

- **each *branch node* represents a choice between a number of alternatives**
- **each *leaf node* represents a classification or decision**



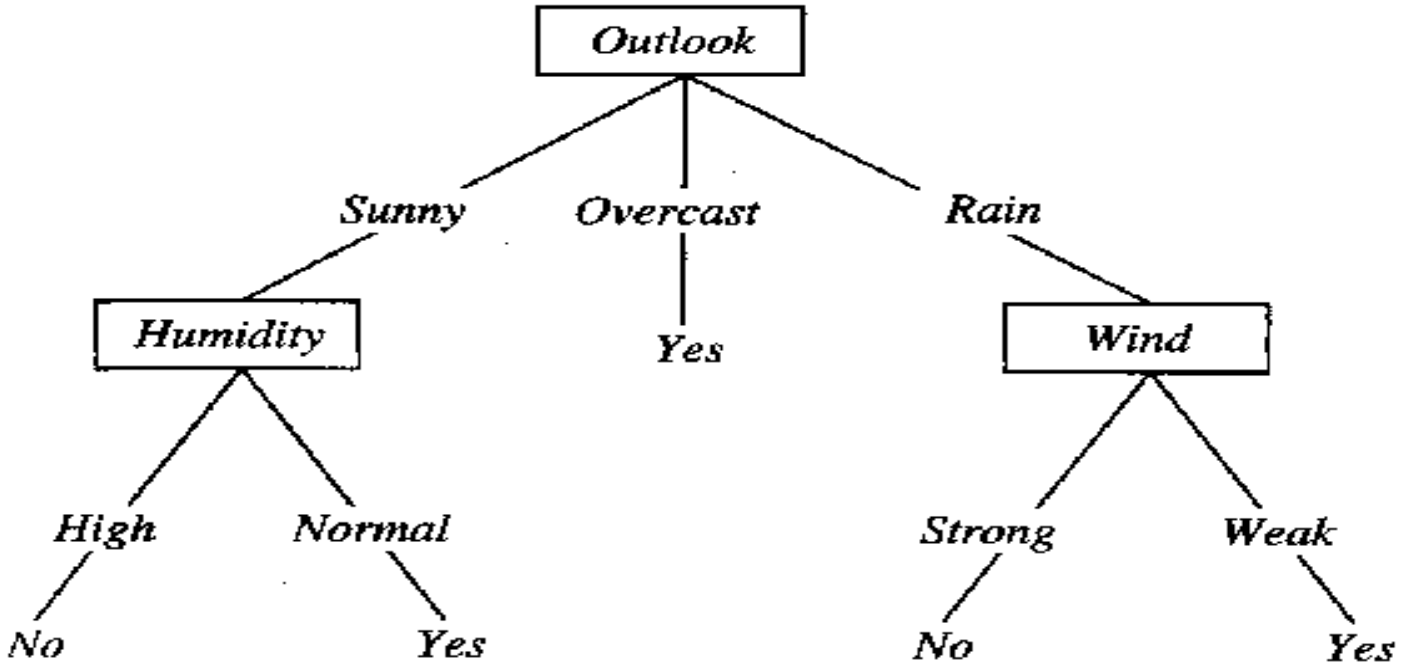
节点 （Node）

每一个节点表示对样本实例的某个属性值进行测试，该节点后相连接的各条路径上的值表示该属性的可能取值（二叉，三叉， ...）



叶子 （Leaf）

每一个叶子产生一条规则，规则由根到该叶子的路径上所有节点的条件，规则的结论是叶子上标注的结论（决策，分类，判断）



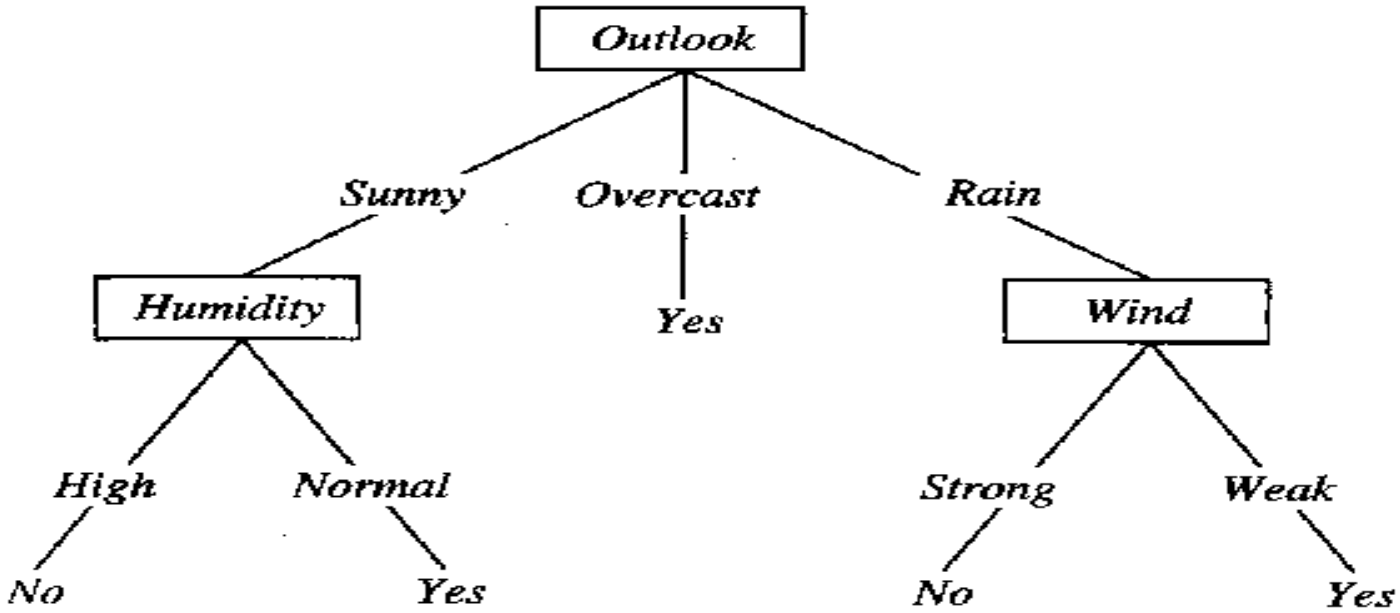
决策树所产生的规则

决策树代表实例属性值约束的合取（交集）的析取（并集）式。从树根到树叶的每一条路径对应一组属性测试的合取，树本身对应这些合取的析取

$$(Outlook = sunny \wedge Humidity = normal)$$

$$\vee (Outlook = overcast)$$

$$\vee (Outlook = rain \wedge Wind = weak)$$



举例：根据身高/发色/眼睛判断东西方人

(tall, blond, blue) w

(short, silver, blue) w

(short, black, blue) w

(tall, blond, brown) w

(tall, silver, blue) w

(short, blond, blue) w

(short, black, brown) e

(tall, silver, black) e

(short, black, brown) e

(tall, black, brown) e

(tall, black, black) e

(short, blond, black) e

w: west e: east

Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]

- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse
    - “Iterative Dichotomizer” (ID3)
    - C4.5 (ID3+improvements)

一些预备知识

信息论相关

信息论相关：熵与不确定性

在信息论与概率统计中，熵（entropy）是表示随机变量不确定性的度量．设 X 是一个取有限个值的离散随机变量，其概率分布为

$$P(X=x_i)=p_i, \quad i=1,2,\cdots,n$$

则随机变量 X 的熵定义为

$$H(X)=-\sum_{i=1}^{n} p_i \log p_i \tag{5.1}$$

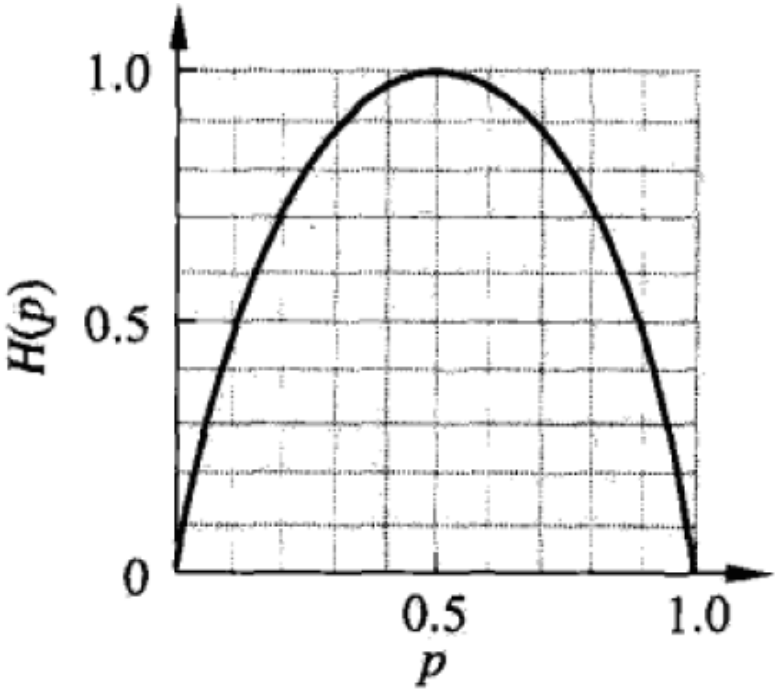
信息论相关：熵与不确定性

当随机变量只取两个值，例如 1，0 时，即 X 的分布为

$$P(X=1)=p\,,\quad P(X=0)=1-p\,,\quad 0\leqslant p\leqslant 1$$

熵为

$$H(p)=-p\log_2 p-(1-p)\log_2(1-p)$$



分布为贝努利分布时熵与概率的关系

信息论相关：条件熵

设有随机变量(X,Y)，其联合概率分布为

$$P(X=x_i,Y=y_j)=p_{ij}\,,\quad i=1,2,\cdots,n\,;\quad j=1,2,\cdots,m$$

条件熵$H(Y|X)$表示在已知随机变量X的条件下随机变量Y的不确定性. 随机变量X给定的条件下随机变量Y的条件熵（conditional entropy）$H(Y|X)$，定义为X给定条件下Y的条件概率分布的熵对X的数学期望

$$H(Y|X)=\sum_{i=1}^{n}p_iH(Y|X=x_i) \tag{5.5}$$

这里，$p_i=P(X=x_i)$，$i=1,2,\cdots,n$.

信息增益（information gain）表示得知特征X的信息而使得类Y的信息的不确定性减少的程度.

信息论相关：信息增益（互信息）

定义 5.2（信息增益） 特征 A 对训练数据集 D 的信息增益 $g(D,A)$，定义为集合 D 的经验熵 $H(D)$ 与特征 A 给定条件下 D 的经验条件熵 $H(D|A)$ 之差，即

$$g(D,A)=H(D)-H(D|A) \tag{5.6}$$

一般地，熵 $H(Y)$ 与条件熵 $H(Y|X)$ 之差称为互信息（mutual information）. 决策树学习中的信息增益等价于训练数据集中类与特征的互信息.

信息论相关：信息增益算法

根据信息增益准则的特征选择方法是：对训练数据集（或子集）D，计算其每个特征的信息增益，并比较它们的大小，选择信息增益最大的特征.

算法 5.1（信息增益的算法）

输入：训练数据集 D 和特征 A；

输出：特征 A 对训练数据集 D 的信息增益 $g(D,A)$.

（1）计算数据集 D 的经验熵 $H(D)$

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log_2\frac{|C_k|}{|D|} \tag{5.7}$$

（2）计算特征 A 对数据集 D 的经验条件熵 $H(D|A)$

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|} \tag{5.8}$$

（3）计算信息增益

$$g(D,A)=H(D)-H(D|A) \tag{5.9}$$ ■

信息论相关：信息增益比

信息增益值的大小是相对于训练数据集而言的，并没有绝对意义．在分类问题困难时，也就是说在训练数据集的经验熵大的时候，信息增益值会偏大．反之，信息增益值会偏小．使用信息增益比（information gain ratio）可以对这一问题进行校正．这是特征选择的另一准则．

定义 5.3（信息增益比） 特征 A 对训练数据集 D 的信息增益比 $g_R(D,A)$ 定义为其信息增益 $g(D,A)$ 与训练数据集 D 的经验熵 $H(D)$ 之比：

$$g_R(D,A)=\frac{g(D,A)}{H(D)} \tag{5.10}$$

信息增益计算举例

例 5.2 对表 5.1 所给的训练数据集 D，根据信息增益准则选择最优特征.

表 5.1 贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

例 5.1[①] 表 5.1 是一个由 15 个样本组成的贷款申请训练数据. 数据包括贷款申请人的 4 个特征（属性）：第 1 个特征是年龄，有 3 个可能值：青年，中年，老年；第 2 个特征是有工作，有 2 个可能值：是，否；第 3 个特征是有自己的房子，有 2 个可能值：是，否；第 4 个特征是信贷情况，有 3 个可能值：非常好，好，一般. 表的最后一列是类别，是否同意贷款，取 2 个值：是，否.

表 5.1 贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

信息增益计算举例

解 首先计算经验熵 $H(D)$.

$$H(D)=-\frac{9}{15}\log_2\frac{9}{15}-\frac{6}{15}\log_2\frac{6}{15}=0.971$$

然后计算各特征对数据集 D 的信息增益. 分别以 A_1，A_2，A_3，A_4 表示年龄、有工作、有自己的房子和信贷情况 4 个特征，则

（1）

$$\begin{aligned}g(D,A_1)&=H(D)-\left[\frac{5}{15}H(D_1)+\frac{5}{15}H(D_2)+\frac{5}{15}H(D_3)\right]\\&=0.971-\left[\frac{5}{15}\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)\right.\\&\quad\left.+\frac{5}{15}\left(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}\right)+\frac{5}{15}\left(-\frac{4}{5}\log_2\frac{4}{5}-\frac{1}{5}\log_2\frac{1}{5}\right)\right]\\&=0.971-0.888=0.083\end{aligned}$$

这里 D_1，D_2，D_3 分别是 D 中 A_1（年龄）取值为青年、中年和老年的样本子集. 类似地，

信息增益计算举例

(2)

$$g(D,A_2)=H(D)-\left[\frac{5}{15}H(D_1)+\frac{10}{15}H(D_2)\right]$$

$$=0.971-\left[\frac{5}{15}\times 0+\frac{10}{15}\left(-\frac{4}{10}\log_2\frac{4}{10}-\frac{6}{10}\log_2\frac{6}{10}\right)\right]=0.324$$

(3)

$$g(D,A_3)=0.971-\left[\frac{6}{15}\times 0+\frac{9}{15}\left(-\frac{3}{9}\log_2\frac{3}{9}-\frac{6}{9}\log_2\frac{6}{9}\right)\right]$$

$$=0.971-0.551=0.420$$

(4)

$$g(D,A_4)=0.971-0.608=0.363$$

最后，比较各特征的信息增益值．由于特征 A_3（有自己的房子）的信息增益值最大，所以选择特征 A_3 作为最优特征．■

ID3学习算法

ID3算法

ID3是Quinlan于1986年提出的，它的提出开创了决策树算法的先河，而且是国际上最早的决策树方法，在该算法中，引入了信息论中熵的概念，利用分割前后的熵来计算信息增益，作为判别能力的度量。

ID3基本思想

ID3 算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归地构建决策树．具体方法是：从根结点（root node）开始，对结点计算所有可能的特征的信息增益，选择信息增益最大的特征作为结点的特征，由该特征的不同取值建立子结点；再对子结点递归地调用以上方法，构建决策树；直到所有特征的信息增益均很小或没有特征可以选择为止．最后得到一个决策树．ID3 相当于用极大似然法进行概率模型的选择．

ID3算法

输入：训练数据集 D，特征集 A，阈值 ε；

输出：决策树 T.

ID3算法举例

例 5.3 对表 5.1 的训练数据集，利用 ID3 算法建立决策树.

表 5.1 贷款申请样本数据表

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

ID3算法举例

解 利用例 5.2 的结果，由于特征 A_3（有自己的房子）的信息增益值最大，所以选择特征 A_3 作为根结点的特征．它将训练数据集 D 划分为两个子集 D_1（A_3 取值为“是”）和 D_2（A_3 取值为“否”）．由于 D_1 只有同一类的样本点，所以它成为一个叶结点，结点的类标记为“是”.

对 D_2 则需从特征 A_1（年龄），A_2（有工作）和 A_4（信贷情况）中选择新的特征．计算各个特征的信息增益：

$$g(D_2, A_1) = H(D_2) - H(D_2 \mid A_1) = 0.918 - 0.667 = 0.251$$
$$g(D_2, A_2) = H(D_2) - H(D_2 \mid A_2) = 0.918$$
$$g(D_2, A_4) = H(D_2) - H(D_2 \mid A_4) = 0.474$$

选择信息增益最大的特征 A_2（有工作）作为结点的特征．由于 A_2 有两个可能取值，从这一结点引出两个子结点：一个对应“是”（有工作）的子结点，包含 3 个样本，它们属于同一类，所以这是一个叶结点，类标记为“是”；另一个是对应“否”（无工作）的子结点，包含 6 个样本，它们也属于同一类，所以这也是一个叶结点，类标记为“否”.

ID3算法举例

这样生成一个如图 5.5 所示的决策树．该决策树只用了两个特征（有两个内部结点）．

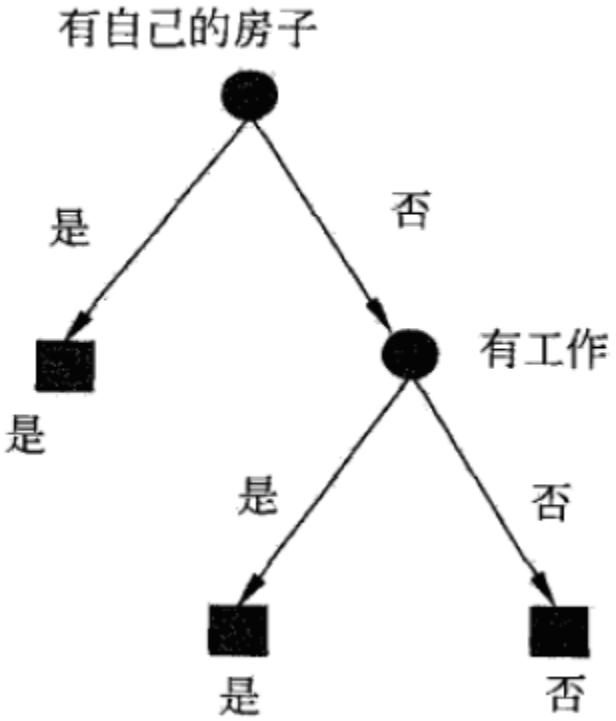


图 5.5　决策树的生成

ID3 算法只有树的生成，所以该算法生成的树容易产生过拟合．

C4.5 算法

C4.5 算法与 ID3 算法相似，C4.5 算法对 ID3 算法进行了改进．C4.5 在生成的过程中，用信息增益比来选择特征．

C4.5 生成算法

输入：训练数据集 D，特征集 A，阈值 ε；

输出：决策树 T.

（1）如果 D 中所有实例属于同一类 C_k，则置 T 为单结点树，并将 C_k 作为该结点的类，返回 T；

（2）如果 $A=\varnothing$，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类，返回 T；

（3）否则，按式(5.10)计算 A 中各特征对 D 的信息增益比，选择信息增益比最大的特征 A_g；

（4）如果 A_g 的信息增益比小于阈值 ε，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类，返回 T；

（5）否则，对 A_g 的每一可能值 a_i，依 $A_g=a_i$ 将 D 分割为子集若干非空 D_i，将 D_i 中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树 T，返回 T；

（6）对结点 i，以 D_i 为训练集，以 $A-\{A_g\}$ 为特征集，递归地调用步(1)～步(5)，得到子树 T_i，返回 T_i.

■

C4.5 algorithm

C4.5 is an extension of the basic ID3 algorithm designed by Quinlan to address the following issues not dealt with by ID3:

- Avoiding overfitting the data
- Determining how deeply to grow a decision tree.
- Reduced error pruning.
- Rule post-pruning.
- Handling continuous attributes. e.g., temperature
- Choosing an appropriate attribute selection measure.
- Handling training data with missing attribute values.
- Handling attributes with differing costs.
- Improving computational efficiency.

Classification and Regression Trees
(CART or C&RT)

CART

分类与回归树（classification and regression tree，CART）模型由 Breiman 等人在 1984 年提出，是应用广泛的决策树学习方法．CART 同样由特征选择、树的生成及剪枝组成，既可以用于分类也可以用于回归．以下将用于分类与回归的树统称为决策树．

CART 是在给定输入随机变量 X 条件下输出随机变量 Y 的条件概率分布的学习方法．CART 假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支．这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布．

CART算法组成

（1）决策树生成：基于训练数据集生成决策树，生成的决策树要尽量大；

（2）决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树，这时用损失函数最小作为剪枝的标准.

CART生成：回归树

决策树的生成就是递归地构建二叉决策树的过程.对回归树用平方误差最小化准则，对分类树用基尼指数（Gini index）最小化准则，进行特征选择，生成二叉树.

假设 X 与 Y 分别为输入和输出变量，并且 Y 是连续变量，给定训练数据集

$$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$

考虑如何生成回归树.

算法 5.5（最小二乘回归树生成算法）

输入：训练数据集 D；

输出：回归树 $f(x)$.

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 j 与切分点 s，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{5.21}$$

遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使式 (5.21) 达到最小值的对 (j,s).

（2）用选定的对 (j,s) 划分区域并决定相应的输出值：

$$R_1(j,s)=\{x\mid x^{(j)}\leqslant s\},\quad R_2(j,s)=\{x\mid x^{(j)}>s\}$$

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i,\quad x\in R_m,\quad m=1,2$$

（3）继续对两个子区域调用步骤 (1)，(2)，直至满足停止条件.

（4）将输入空间划分为 M 个区域 $R_1,R_2,\cdots,R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

CART生成：分类树

分类树用基尼指数选择最优特征，同时决定该特征的最优二值切分点.

定义 5.4（基尼指数） 分类问题中，假设有 K 个类，样本点属于第 k 类的概率为 p_k，则概率分布的基尼指数定义为

$$\text{Gini}(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2 \tag{5.22}$$

对于二类分类问题，若样本点属于第 1 个类的概率是 p，则概率分布的基尼指数为

$$\text{Gini}(p)=2p(1-p) \tag{5.23}$$

对于给定的样本集合 D，其基尼指数为

$$\text{Gini}(D)=1-\sum_{k=1}^{K}\left(\frac{|C_k|}{|D|}\right)^2 \tag{5.24}$$

这里，C_k 是 D 中属于第 k 类的样本子集，K 是类的个数.

CART生成：分类树

如果样本集合 D 根据特征 A 是否取某一可能值 a 被分割成 D_1 和 D_2 两部分，即

$$D_1 = \{(x, y) \in D \mid A(x) = a\}, \quad D_2 = D - D_1$$

则在特征 A 的条件下，集合 D 的基尼指数定义为

$$\text{Gini}(D, A) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2) \tag{5.25}$$

基尼指数 $\text{Gini}(D)$ 表示集合 D 的不确定性，基尼指数 $\text{Gini}(D, A)$ 表示经 $A = a$ 分割后集合 D 的不确定性．基尼指数值越大，样本集合的不确定性也就越大，这一点与熵相似．

CART生成算法

输入：训练数据集 D，停止计算的条件；

输出：CART 决策树.

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树：

（1）设结点的训练数据集为 D，计算现有特征对该数据集的基尼指数. 此时，对每一个特征 A，对其可能取的每个值 a，根据样本点对 $A=a$ 的测试为“是”或“否”将 D 分割成 D_1 和 D_2 两部分，利用式（5.25）计算 $A=a$ 时的基尼指数.

（2）在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点. 依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去.

（3）对两个子结点递归地调用（1），（2），直至满足停止条件.

（4）生成 CART 决策树.

算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值（样本基本属于同一类），或者没有更多特征.

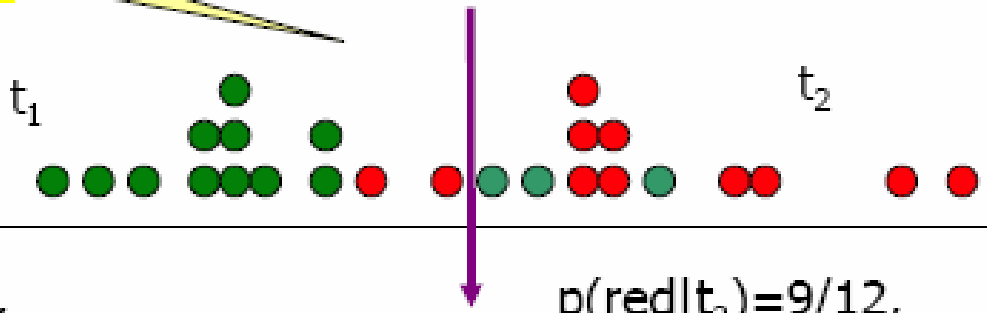
基尼指数

- $gini(t)=1-\Sigma_j p^2(j|t)$

数据混杂度

p(red|t)=11/25,
p(green|t)=14/25,
gini(t)=1-0.194-0.314=0.492

一个划分



p(red|t_1)=2/13,
p(green|t_1)=11/13,
gini(t_1)=1 – 0.024 – 0.726 = 0.25

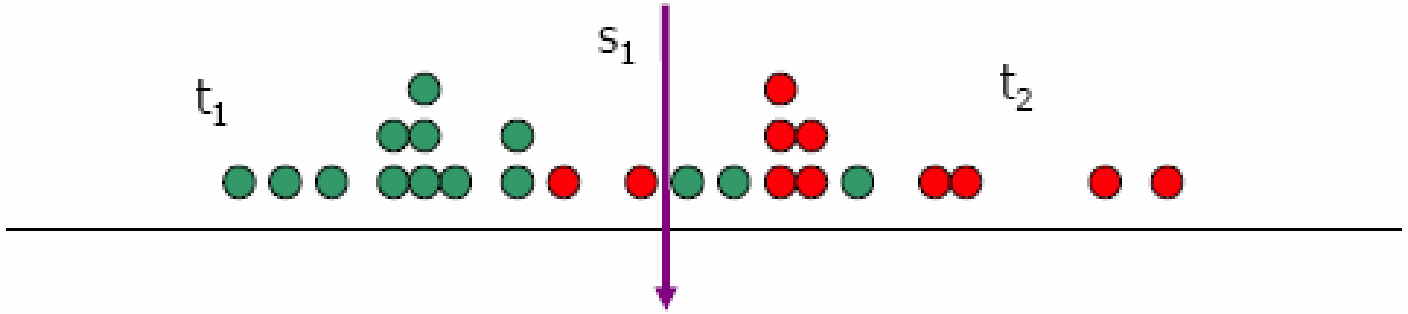
p(red|t_2)=9/12,
p(green|t_2)=3/12,
gini(t_2)=1 – 0.563 – 0.063 = 0.374

Weighted average = 0.25*(13/25) +0.374*(12/25) = 0.31

划分的优度

基尼指数的变化量：

$$= \text{gini}(t) - (n_1/n)*\text{gini}(t_1) - (n_2/n)*\text{gini}(t_2)$$



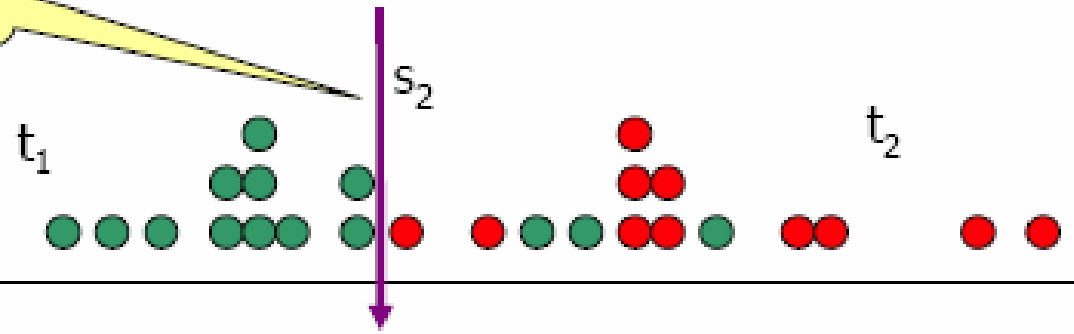
$$\text{Improvement}(s_1) = 0.492 - 0.31 = 0.182$$

$$gini(t) = 1 - \Sigma_j p^2(j|t)$$

数据混杂度

p(red|t)=11/25,
p(green|t)=14/25,
gini(t)=1-0.194-0.314=0.492

另一个划分



$p(red|t_1)=0/11$,
$p(green|t_1)=11/11$,
$gini(t_1) = 1 - 0.0 - 1.0 = 0.0$

$p(red|t_2)=11/14$,
$p(green|t_2)=3/14$,
$gini(t_2) = 1 - 0.617 - 0.046 = 0.337$

Weighted average = 0.0*(11/25) +0.337*(14/25) = 0.189

$Improvement(s_2) = 0.492 - 0.189 = 0.303$

$Improvement(s_1) = 0.492 - 0.31 = 0.182$

s_2 是更好的划分

基尼指数划分的特点

- 基尼指数关注的目标变量里面最大的类，它试图找到一个划分把它和其它类别区分开来。
- 完美的系列划分将会得到k个纯粹的子节点，每一个节点对应目标变量的一个类别。
- 如果误分类代价因素被加入，基尼指数试图把代价最大的类别区分开来。

剪枝

在终止条件被满足，划分停止之后，下一步是剪枝：

- 给树剪枝就是剪掉“弱枝”，弱枝指的是在验证数据上误分类率高的树枝
- 为树剪枝会增加训练数据上的错误分类率，但精简的树会提高新记录上的预测能力
- 剪掉的是最没有预测能力的枝

Next lecture

- Random Forests (随机森林)

Thank you!

CRIPAC
智能感知与计算研究中心
Center for Research on Intelligent Perception and Computing

Questions?