# 2. 传统机器学习

xgsun@fudan.edu.cn
孙晓光

2025-2-27

# 提 纲

① 传统**机器学习**

**监督学习**

② 线性**回归**（一元、多元）、Logistic回归(分类)

③ 二**分类**、多分类

④ 决策树

⑤ 支持向量机 SVM

**无监督学习**

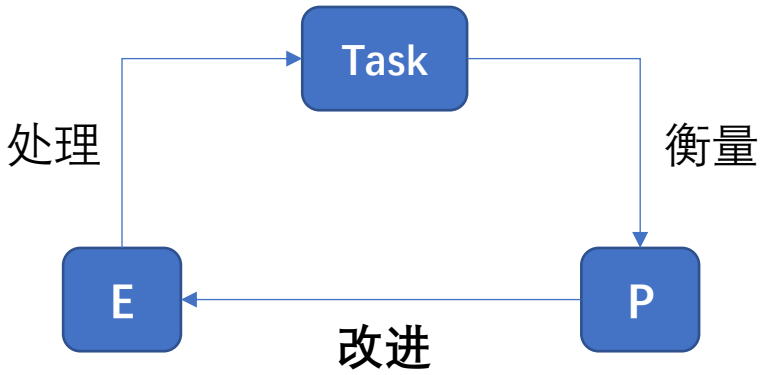
⑥ K均值聚类 （聚类）

⑦ 主成分分析 （降维）

# 机器学习 定义

- **机器学习**是这样的一个研究领域，它能让计算机不依赖于确定的编码指令来自主地学习工作。 1959, Arthur Samuel, 跳棋程序

- 一个计算机程序被称为**可以学习**，是指它能够针对某个任务Task和某个性能指标P，从经验E Experience中**学习**。这种学习的特点是，它在 Task上被 **P** 所衡量的性能，会随着经验 **E** 的增加而提高。 1998，Tom Mitshell 美国卡内基梅隆大学计算机学院院长





ML是AI 的一个核心分支，核心目标是通过**数据驱动**的方法，让计算机系统自动从经验(数据) 中学习规律和模式，从而完成特定任务或改进性能，而无需显式编程。

# 数据驱动的学习

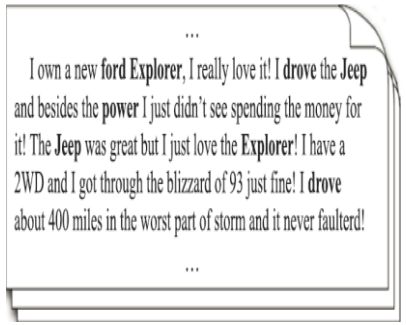
从**数据**中学习**知识** — 化繁为简

**图像数据**

$$f\left\{\begin{matrix} 81 & 116 & \cdots & 133 \\ 120 & 130 & \cdots & 126 \\ \vdots & \vdots & \ddots & \vdots \\ 145 & 178 & \cdots & 207 \\ 189 & 223 & \cdots & 215 \end{matrix}\right\}$$

- 猫
- 狗
- …

**类别分类**




**文本数据**

$$f\{jeep, money, \ldots\}$$

- 喜悦
- 愤怒
- … …

**情感分类**

- 从原始数据中提取**特征**
- 学习映射函数 $f(\cdot)$
- 映射函数$f$将原始数据映射到隐藏特征空间，即，寻找**数据**和**目标**之间的关系.

# ML 算法分类

按照训练数据是否存在**标签 label** (答案)

## 监督学习

- 标签：**连续**、**离散**

**回归** Regression — 连续

**分类** Classification — 离散

| | | | |
|---|---|---|---|
| 一元线性回归 | 多元线性回归 | 逻辑回归 | 二 (多) 分类 |
| 决策树 | 支持向量机 | 贝叶斯模型 | ..... |

## 无监督学习

| | | | |
|---|---|---|---|
| K均值聚类 | 高斯混合模型 | 主成分分析 | ..... |
| 聚 类 | | 降 维 | |

## 强化学习

# 有监督学习 *vs* 无监督学习

- 训练数据**有**标签 (答案)
- 最小化预测值与标签值间的误差
- 用于**回归**和**分类**。

  如，房价预测、图像分类、语音识别...

- 训练数据**无**标签 (答案)
- 发现数据的内在结构
- 用于**聚类**、**降维**和**异常检测**。

  如，客户分群、主成分分析...

✓ **特征提取**：二者都需要从数据中提取有用的特征。

✓ **结合使用**：如，先用无监督学习进行特征提取，后将提取的特征用于有监督学习的回归或分类。

# 预测模型

input predictor output example

- **二分类**
  Binary classification

$x \rightarrow f \rightarrow y \in \{+1, -1\}$ email → spam / not spam

- **多分类**
  Multiclass classification


$\rightarrow f \rightarrow$ cat, dog, ...

- **回归** Regression

$x \rightarrow f \rightarrow y \in \mathbb{R}$ location, year → housing price

- **结构化预测**
  Structured prediction

$x \rightarrow f \rightarrow y$为复杂物体 机器翻译：汉语 → 英语

# 判别模型、生成模型

**监督学习**方法

又可分为：

**判别方法**

$f($  $) \longrightarrow$ 人脸　　判别函数 $f(X)$

$P($ 人脸 $|$  $) = 0.99$　　条件概率分布 $P(Y|X)$

**生成方法**

$f($ 人脸 $) \longrightarrow$ 

所学到的模型：**判别模型**、**生成模型**。

# 2

# regression

# 回归

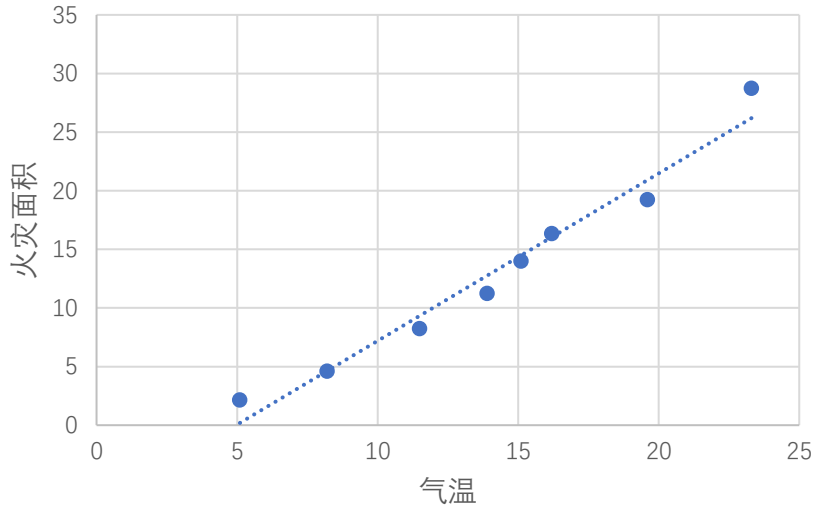
监督学习

# 一元线性回归

**回归分析**：分析不同变量之间的关系。

某地发生森林火灾的部分数据

| 气温 x | 5.1 | 8.2 | 11.5 | 13.9 | 15.1 | 16.2 | 19.6 | 23.3 |
|---|---|---|---|---|---|---|---|---|
| 火灾面积 y | 2.14 | 4.62 | 8.24 | 11.24 | 13.99 | 16.33 | 19.23 | 28.74 |




以最小误差用数据拟合直线 $\hat{y} = ax + b$，求解出 $a$、$b$

$$\min_{a,b} \mathcal{L}(a,b) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - ax_i - b)^2 \qquad \text{最小二乘法}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} 2(y_i - ax_i - b)(-1) = 0$$

$$\Rightarrow \sum_{i=1}^{n}(y_i - ax_i - b) = 0$$

$$\Rightarrow b = \bar{y} - a\bar{x}$$

$$\frac{\partial \mathcal{L}}{\partial a} = \sum_{i=1}^{n} 2(y_i - ax_i - b)(-x_i) = 0$$

$$\Rightarrow \sum_{i=1}^{n}(y_i x_i - ax_i x_i - \bar{y}x_i + a\bar{x}x_i) = 0$$

$$\Rightarrow a = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i x_i - n\bar{x}^2}$$

$$a = \frac{x_1 y_1 + x_2 y_2 + \cdots + x_8 y_8 - 8\bar{x}\bar{y}}{x_1^2 + x_2^2 + \cdots + x_8^2 - 8\bar{x}^2} = 1.428$$

$$b = \bar{y} - a\bar{x} = -7.09$$

$$\hat{y} = 1.428x - 7.09$$

# 多元线性回归

某地发生森林火灾的部分数据

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 气温 x | 5.1 | 8.2 | 11.5 | 13.9 | 15.1 | 16.2 | 19.6 | 23.3 |
| (新增) | 风力 z | 4.5 | 5.8 | 4 | 6.3 | 4 | 7.2 | 6.3 | 8.5 |
| | 火灾面积 y | 2.14 | 4.62 | 8.24 | 11.24 | 13.99 | 16.33 | 19.23 | 28.74 |

$$f(x_i) = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{in} + w_0$$

$$f(x_i) = x_{1i} w_1 + x_{2i} w_2 + \cdots + x_{ni} w_n + w_0$$

$$f(x_i) = \boldsymbol{x_i^T w} + 1 \cdot w_0$$

$$\hat{\boldsymbol{y}} = \boldsymbol{X^T W}$$

$$\min_{\boldsymbol{W}} \mathcal{L} = (\boldsymbol{y} - \boldsymbol{X^T W})^T (\boldsymbol{y} - \boldsymbol{X^T W})$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = -2\boldsymbol{X}(\boldsymbol{y} - \boldsymbol{X^T W}) = 0$$

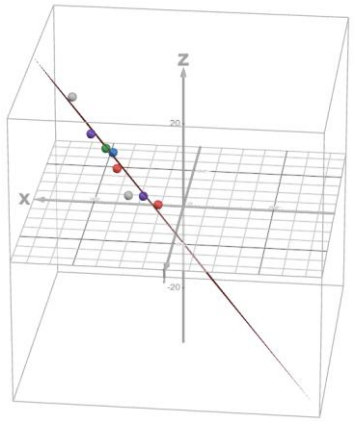
$$\Longrightarrow \quad \boldsymbol{W} = (\boldsymbol{X X^T})^{-1} \boldsymbol{X y}$$

$$\boldsymbol{X} = \begin{bmatrix} 5.1 & 8.2 & 11.5 & 13.9 & 15.1 & 16.2 & 19.6 & 23.3 \\ 4.5 & 5.8 & 4 & 6.3 & 4 & 7.2 & 6.3 & 8.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\boldsymbol{y} = [2.14 \;\; 4.62 \;\; 8.24 \;\; 11.24 \;\; 13.99 \;\; 16.33 \;\; 19.23 \;\; 28.74 \;]^T$$

$$\boldsymbol{W} = [1.312 \quad 0.626 \quad -9.103 \;]^T$$

$$\Rightarrow \quad \hat{y} = 1.312 x_1 + 0.626 x_2 - 9.103$$



# 代码1：二元线性回归

02-二元线性回归.ipynb

| 气温 x | 5.1 | 8.2 | 11.5 | 13.9 | 15.1 | 16.2 | 19.6 | 23.3 |
|---|---|---|---|---|---|---|---|---|
| 风力 z | 4.5 | 5.8 | 4 | 6.3 | 4 | 7.2 | 6.3 | 8.5 |
| 火灾面积 y | 2.14 | 4.62 | 8.24 | 11.24 | 13.99 | 16.33 | 19.23 | 28.74 |

```python
# 导入sklearn机器学习库中的线性回归模块(linear_model)
from sklearn import linear_model

# 训练数据
x = [[5.1, 4.5], [8.2, 5.8], [11.5, 4], [13.9, 6.3], [15.1, 4], [16.2, 7.2], [19.6, 6.3], [23.3, 8.5]]
y = [2.14, 4.62, 8.24, 11.24, 13.99, 16.33, 19.23, 28.74]

# 创建线性回归模型
model = linear_model.LinearRegression()
# 训练模型
model.fit(x, y)

# 预测
print("预测(10, 10): ", model.predict([[10, 10]]))

# 输出模型参数
print("斜率: ", model.coef_)       # 斜率
print("截距: ", model.intercept_) # 截距
```

一个类：创建、训练、预测 线性回归模型

$$\hat{y} = 1.312x_1 + 0.626x_2 - 9.103$$

输出结果：

预测(10, 10)：[10.28514756]

**斜率**：[**1.312**7219 **0.626**49508]

**截距**：**-9.102**525137839688

# 面向对象 程序设计

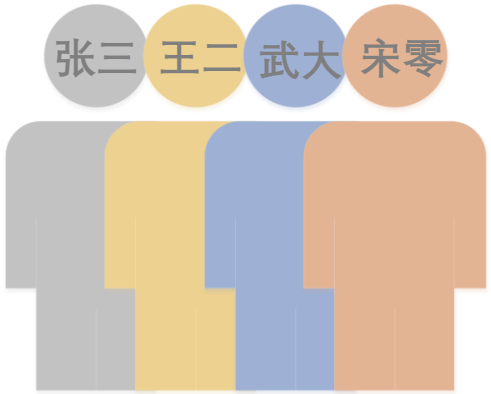
**对象** (object)：对客观事物的抽象。

- **属性**：对象的性质，决定了对象的**外观**。
- **方法**：对象的动作，决定了对象的**行为**。 (函数)

对象是类的具体表现

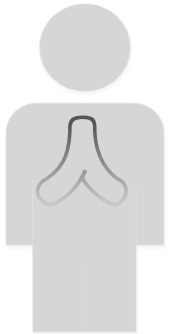

| 属性 | 姓名 | 性别 | 年龄 |
|---|---|---|---|
| | 李 四 | 男 | 20 |

| 方法 | 计算机 | 体育 | 音乐 |
|---|---|---|---|
| | 编程序 | 打篮球 | 弹吉他 |




**类** (class)：同类**对象**的抽象。

- **封装**：数据及其操作都组装在一起。信息隐蔽
- **继承**：代码可重用，对大型软件开发意义重大。



| 属性 | 姓名 | 性别 | 年龄 |
|---|---|---|---|
| | | | |

| 方法 | 计算机 | 体育 | 音乐 |
|---|---|---|---|
| | | | |

# 类 Class

```
class 类名():
    '''类的文档说明'''
    语句 1
    ......
    语句 n
```

封装对象的**属性**和**行为**的载体。

- **定义一个银行类**

类名要大写字母开头

```
class Banks():
    '''定义一个银行类'''   # 类的文档说明

    title = '上海银行'     # 定义一个属性

    def motto(self):       # 定义一个方法
        return '精诚至上，信义立行'
```

- **调用银行类**

```
aBank = Banks()          # 创建一个实例对象
print(aBank.title)       # 调用实例属性
print(aBank.motto())     # 调用实例方法
```

```
上海银行
精诚至上，信义立行
```

# LinearRegression 类

```python
# 导入sklearn机器学习库中的线性回归模块(linear_model)
from sklearn import linear_model

# 训练数据
x = [[5.1, 4.5], [8.2, 5.8], [11.5, 4], [13.9, 6.3], [15.1, 4], [16.2, 7.2], [19.6, 6.3], [23.3, 8.5]]
y = [2.14, 4.62, 8.24, 11.24, 13.99, 16.33, 19.23, 28.74]

# 创建线性回归模型
model = linear_model.LinearRegression()

# 训练模型
model.fit(x, y)

# 预测
print("预测(10, 10): ", model.predict([[10, 10]]))

# 输出模型参数
print("斜率: ", model.coef_)        # 斜率
print("截距: ", model.intercept_)   # 截距
print("方差: ", model.score(x, y)) # 方差
```

**LinearRegression 类**

创建、训练、预测、评估线性回归模型:

**方法**:

- fit(X, y): 训练模型
- predict(X): 预测模型
- score(X, y): 模型的方差

**属性**:

- coef_: 模型的斜率
- intercept_: 模型的截距

# scikit-Learn 机器学习库

**数据集** 鸢尾花、波士顿房价、……

sklearn.datasets

**数据预处理** 标准化、归一化、编码、……

sklearn.preprocessing

**模型训练**

- **有监督学习**：回归、分类
- **无监督学习**：聚类、降维

sklearn.ensemble

**模型优化**

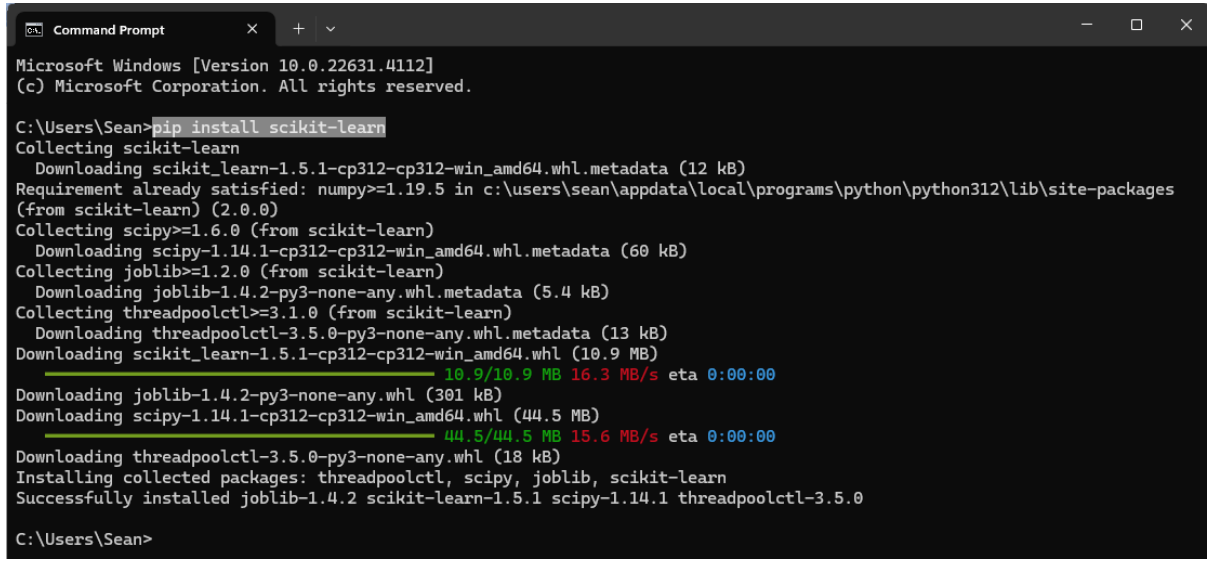
- **集成方法**：随机森林、梯度提升机
- **超参数调优**：随机搜索、网格搜索

**模型评估**

- **分类指标**：准确率、召回率、$F1$值
- **回归指标**：均方差、R平方

sklearn.metrics

安装： > **pip3 install scikit-learn**



# scikit-Learn 自带标准数据集

- **load_iris()　鸢尾花数据集**　　(三分类)

  150个样本，对应3种花儿，每种有50个样本，4个自变量

- **load_wine()** 红酒数据集　　(三分类)

  178个样本，对应3种档次：59、71、48个样本，13个自变量

- **load_breast_cancer()** 乳腺癌数据集 (二分类)

  569个病人，对应2种：恶性(1) / 良性(0)，30个生理指标

- **load_digits()** 手写数字数据集　　(图像分类)

  1797个样本，对应0~9十种，8x8点阵

- **load_diabetes()** 糖尿病数据集　　(回归)

  442行数据，10个属性：年龄、性别、体质指标、血压、S1~S6指标

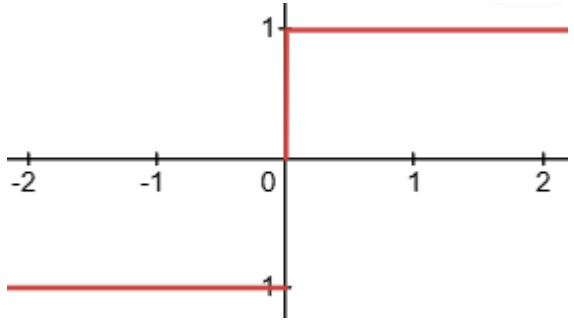
- **load_linnerud()** 体能数据集　　(多变量回归)

# Logistic 回归 Logistic Regression

将线性回归的连续输出映射到概率值（0~1之间），用于**二分类**问题

Sigmoid( ):

$$y = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(w^T x+b)}}$$

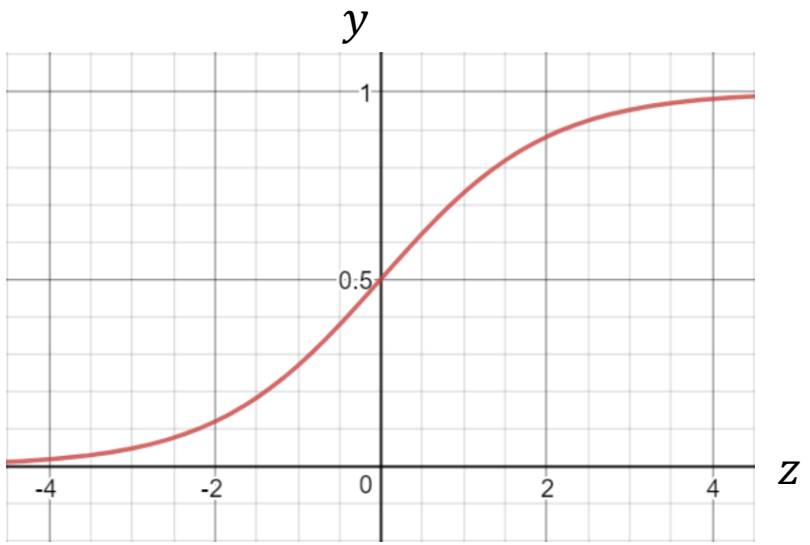



Sign()在$x = 0$处不连续，不可导，无法用梯度下降法优化参数。

- 单调递增函数
- 值域(0, 1)，输出可作概率值

  $\geq 0.5$ 正类别

  $< 0.5$ 负类别
- 对输入 $z$ 取值范围没有限制



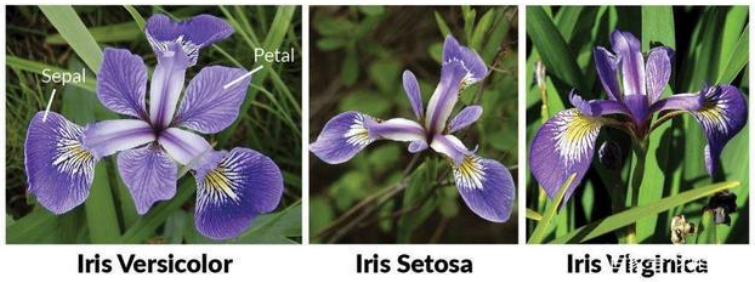
# 代码2： Logistic 回归

02-Logistic回归.ipynb

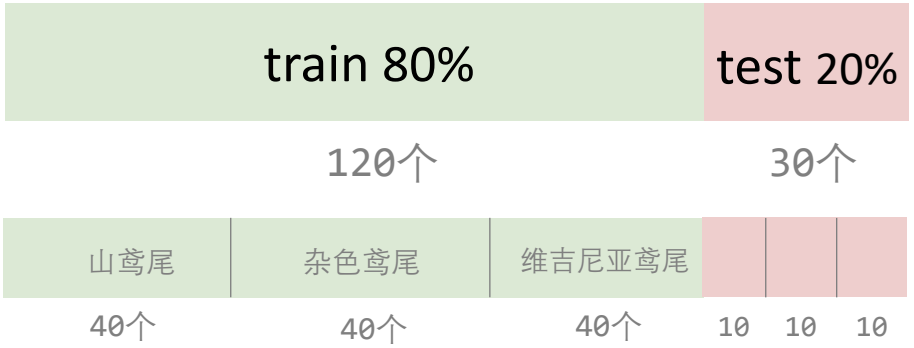
```
# <1> 提取数据集
from sklearn import datasets

# 加载 yuān鸢尾花(iris)数据集
iris = datasets.load_iris()

# 花萼的长、宽、花瓣的长、宽
X = iris.data
# 分类 [0-setosa山鸢尾, 1-versicolor杂色鸢尾, 2-virginica维吉尼亚鸢尾]
y = iris.target

# <2> 数据集划分
from sklearn.model_selection import train_test_split

# 将数据集分为： 训练集、测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size    = 0.2,  # 测试集占20%，隐含 训练集占80%
    random_state = 1,    # 保证每次运行程序时划分结果一致
    shuffle      = True, # 分割前对数据集进行洗牌
    stratify     = y   ) # 训练集和测试集中各类别样本的比例
```

共有150个样本、四个特征和一个类别标签

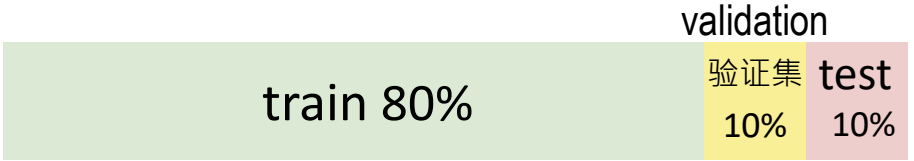
| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | 0 |




**留出法** (Hold-out)




模型容易偏向于训练集中出现次数多的样本




竞赛

**数据集简单随机划分**：

将数据集随机地分配到训练集、验证集和测试集。

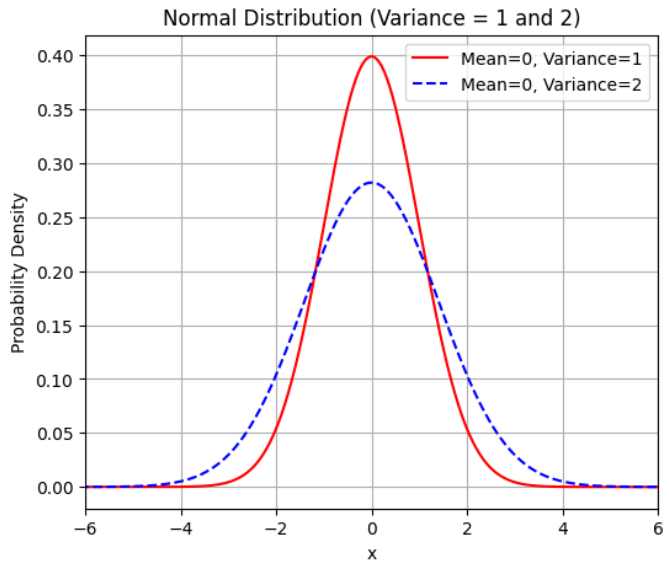
# 代码2： Logistic 回归 -2

02-Logistic回归.ipynb

共有150个样本、四个特征和一个类别标签

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | 0 |

```
[[-0.91782384  1.5241518  -1.28391928 -1.0458439 ]
 [-1.51445906  0.32717395 -1.33998562 -1.30595015]
 [ 1.23006294  0.08777838  0.73446911  1.42516552]
 [-0.44051567 -1.10919948  0.3420047  -0.00541888]
 [-0.32118863 -0.39101277 -0.10652605  0.12463425]]
```



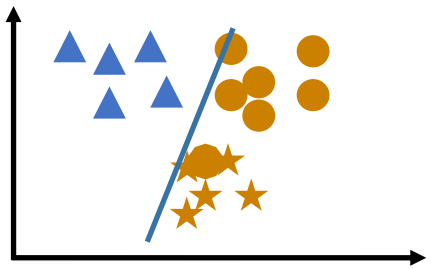

```python
# <3> 特征标准化
from sklearn.preprocessing import StandardScaler

# 标准化数据，使得数据符合标准正态分布，均值为0，方差为1
ss = StandardScaler()

# 标准化训练数据、测试数据
X_train_std = ss.fit_transform(X_train)
X_test_std  = ss.fit_transform(X_test)
```

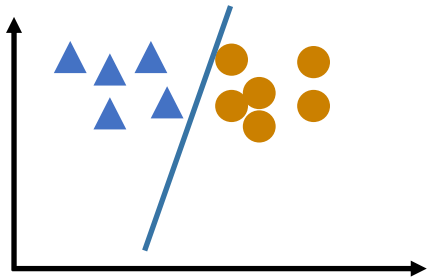
In the multiclass case, the training algorithm uses the one-vs-rest (**OvR**). 本例要分为3类花

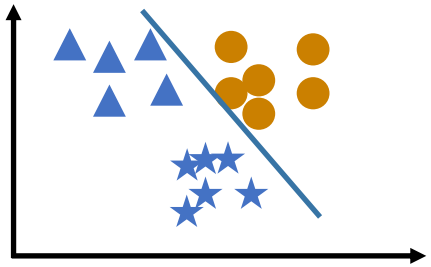
```python
# <4> 训练模型、预测
from sklearn.linear_model import LogisticRegression

L_model = LogisticRegression()  # 创建逻辑回归模型

L_model.fit(X_train_std, y_train) # 在训练集上进行训练

y_pred = L_model.predict(X_test_std) # 训练后预测

# <5> 计算模型准确率
from sklearn.metrics import accuracy_score

# 准确率
acc = accuracy_score(y_pred, y_test)
print(f"准确率: {acc :.2f}")
```

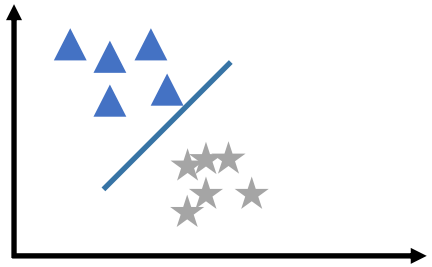
运行结果:
准确率: 0.97

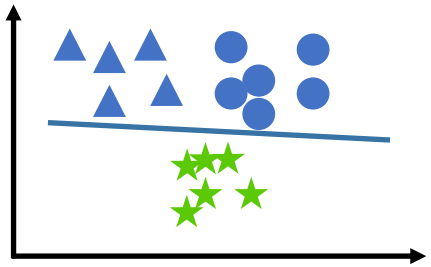
# 多分类

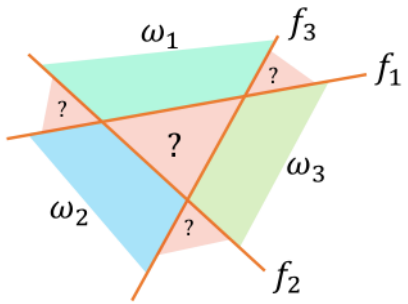


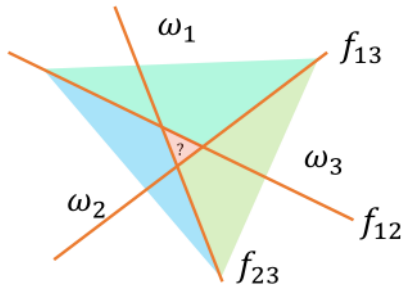


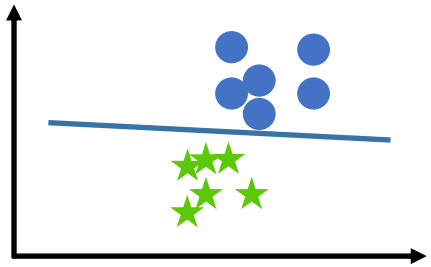








**One Vs Rest** 一对其余

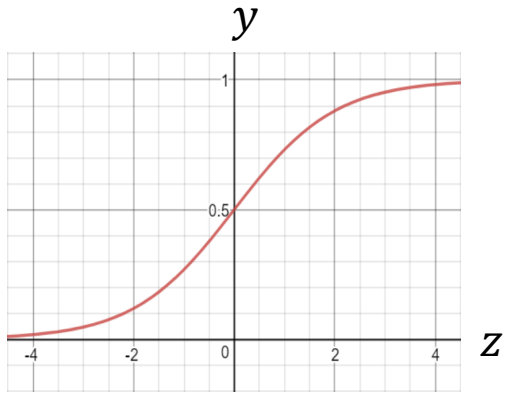
**One Vs One** 一对一

# Sigmoid( )、Softmax( )

- **Sigmoid( ) 函数:** 二分类 $\boldsymbol{w} = [\boldsymbol{w}_0, \boldsymbol{w}_1]$

$$\mathbb{P}(y=1|z) = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z} = \frac{e^{\boldsymbol{w}_1^T x}}{e^{\boldsymbol{w}_0^T x} + e^{\boldsymbol{w}_1^T x}}$$

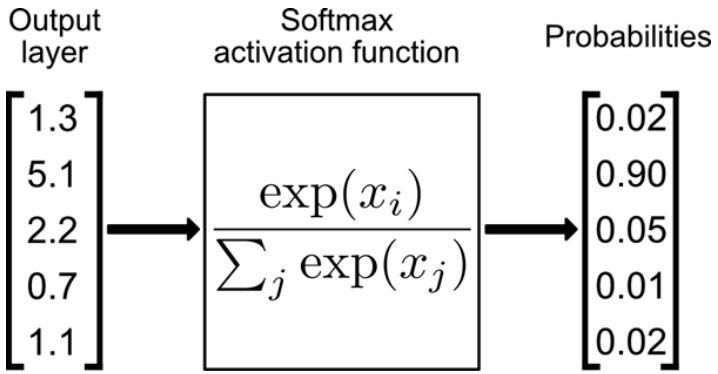
$$\mathbb{P}(y=0|z) = 1 - \sigma(z) = \frac{1}{1+e^z} = \frac{e^{\boldsymbol{w}_0^T x}}{e^{\boldsymbol{w}_0^T x} + e^{\boldsymbol{w}_1^T x}}$$

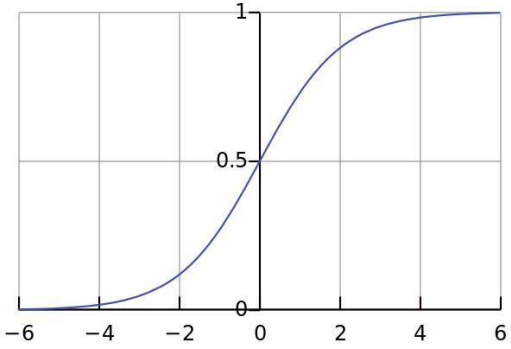



- **Softmax( ) 函数:** 多分类 $\boldsymbol{w} = [\boldsymbol{w}_0, \boldsymbol{w}_1, \cdots, \boldsymbol{w}_{K-1}]$

$$\mathbb{P}(y=K-1|x) = \frac{e^{\boldsymbol{w}_{K-1}^T x}}{\sum_{i=0}^{K-1} e^{\boldsymbol{w}_i^T x}}$$

$$\mathbb{P}(y=0 \mid x) = \frac{e^{\boldsymbol{w}_0^T x}}{\sum_{i=0}^{K-1} e^{\boldsymbol{w}_i^T x}}$$



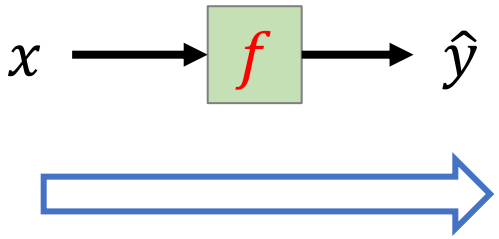
# 3

# 分类

监督学习

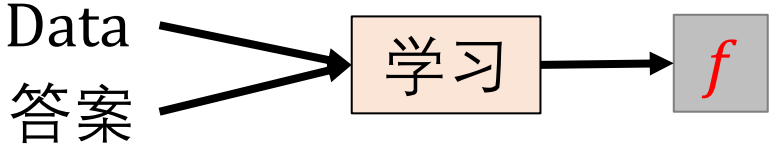
# 机器学习范式

(1) 模型

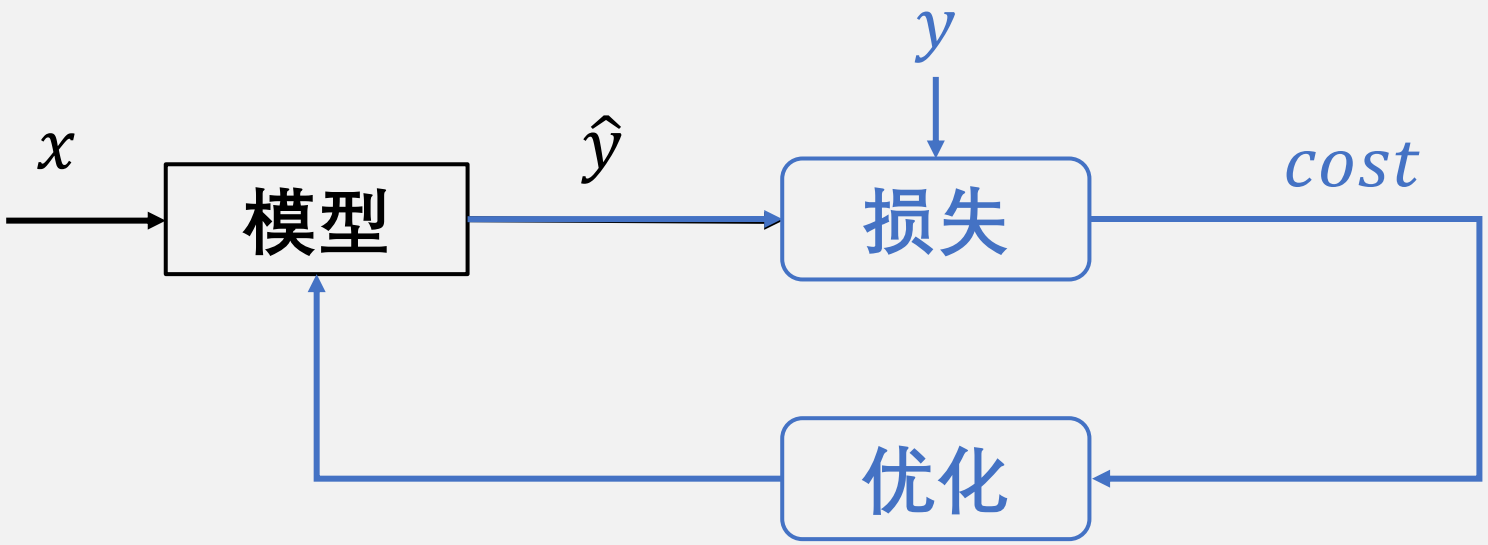



(2) 推理

(3) 学习




三要素:



# 例1：垃圾邮件分类

**输入**：$input = \text{email message}$

**特征向量 $x$**：

length>10 : 1
fracOfAlpah : 0.8
contains_@ : 1
endsWith_.org : 1
endsWith_.com : 0

$$\begin{bmatrix} 1 \\ 0.8 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

**权重向量 $w$**：

length>10 : -1.2
fracOfAlpah : 0.6
contains_@ : 3
endsWith_.org : 2.2
endsWith_.com : 1.4

$$\begin{bmatrix} -1.2 \\ 0.6 \\ 3 \\ 2.2 \\ 1.4 \end{bmatrix}$$

$$Score = \boldsymbol{w}^T \boldsymbol{x}$$

$$= -1.2(1) + 0.6(0.8) + 3(1) + 2.2(1) + 1.4(0)$$

$$= 4.48$$

预测值：

$$\hat{y} = sign(\boldsymbol{w}^T \boldsymbol{x}) = \begin{cases} +1 & if\ \boldsymbol{w}^T \boldsymbol{x} \geq 0 \\ -1 & otherwise \end{cases}$$

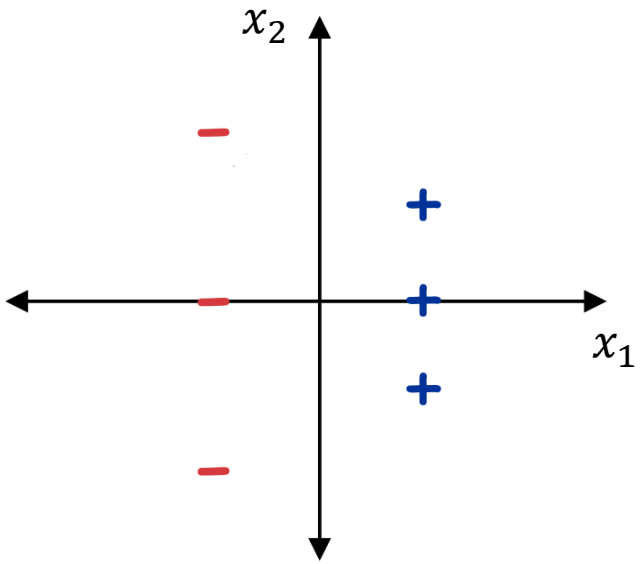
符号函数

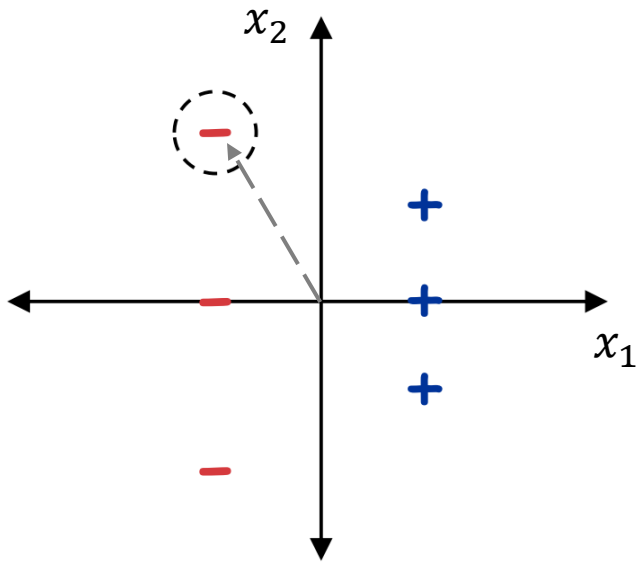
**输出**：$\hat{y} \in \{+1, -1\}$

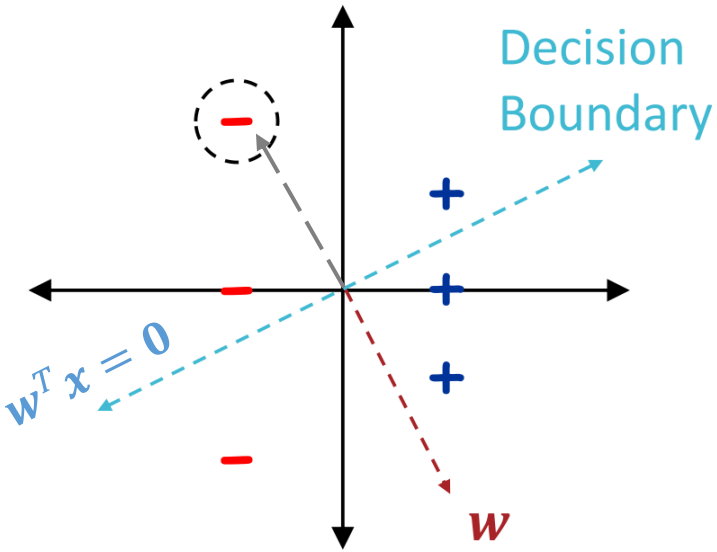
**目标**：求 $f$　　$x \longrightarrow f \longrightarrow \hat{y}$

预测不正确怎么办？

# 例2：调整权重 $w$




| $x_1$ | $x_2$ | $y$ | $\hat{y}$ | 正确 |
|---|---|---|---|---|
| -1 | 2 | -1 | **+1** | No |

初始权重：$\boldsymbol{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

更新权重：$\boldsymbol{w} \Leftarrow \boldsymbol{w} - \boldsymbol{x}^{(1)}$

$$\boldsymbol{w} \Leftarrow \boldsymbol{w} + y^{(1)}\boldsymbol{x}^{(1)}$$

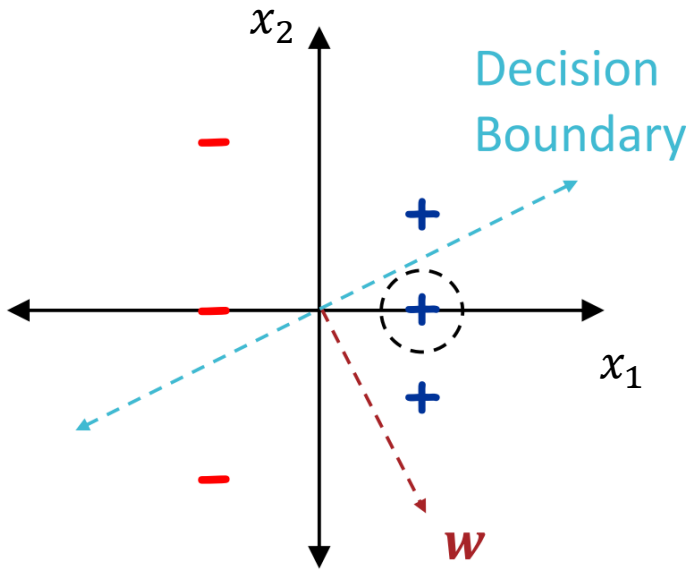
输入值：$\boldsymbol{x}^{(1)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

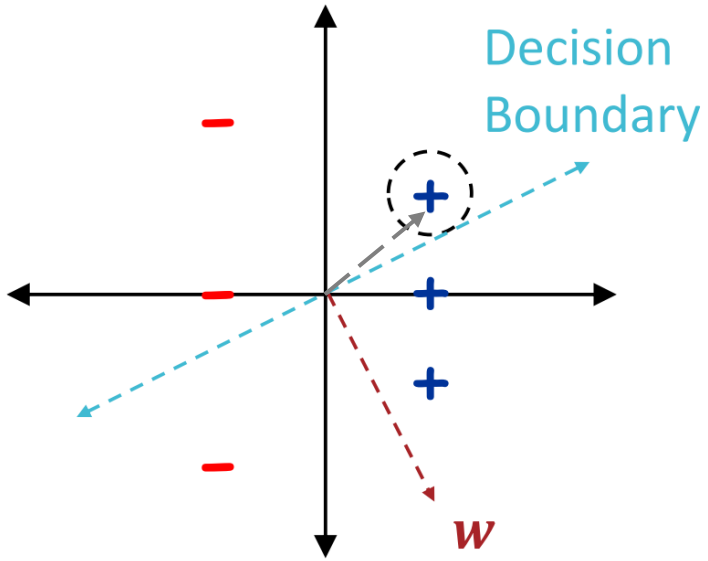
**答案**值：$y^{(1)} = -1$

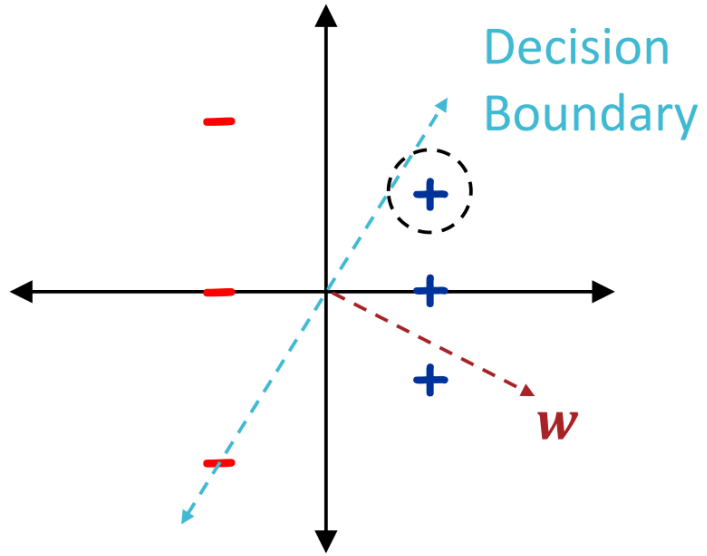
$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 1\begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

**预测**值：$\hat{y} = sign(\boldsymbol{w}^T\boldsymbol{x}) = sign\left(\begin{bmatrix} 0 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix}\right) = +1$

# 例2：调整权重 $\boldsymbol{w}$ (2)



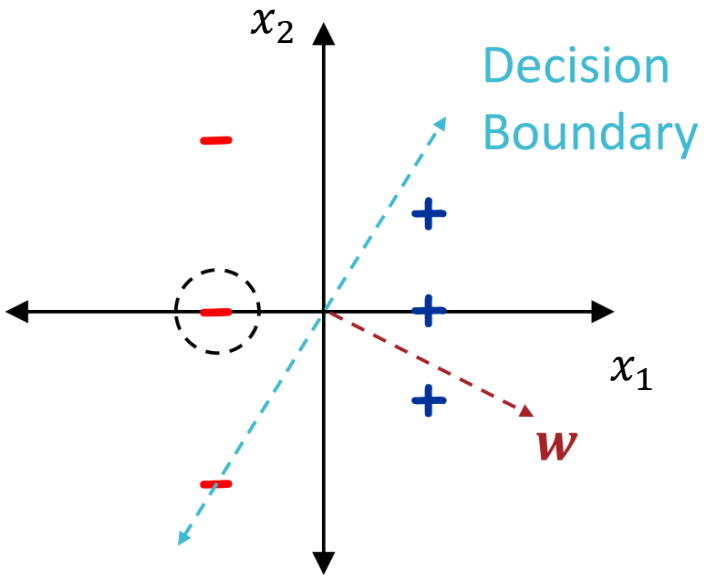

| $x_1$ | $x_2$ | $y$ | $\hat{y}$ | 正确 |
|---|---|---|---|---|
| -1 | 2 | -1 | **+1** | No |
| 1 | 0 | +1 | **+1** | Yes |
| 1 | 1 | +1 | **-1** | No |

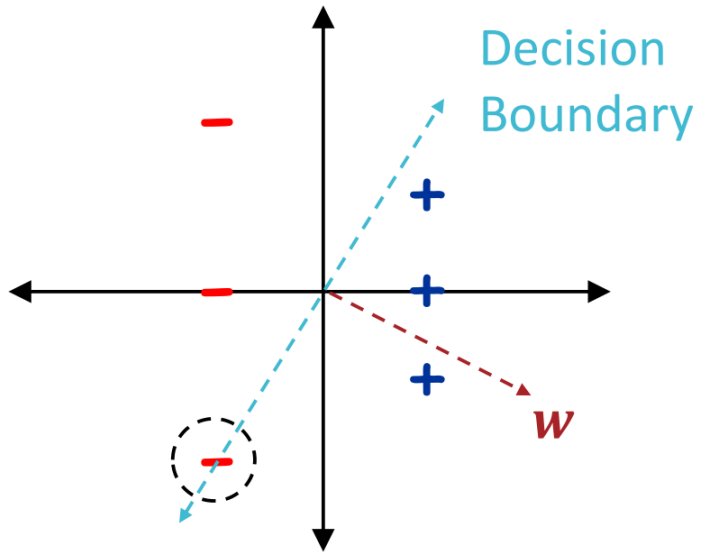
更新权重：$\boldsymbol{w} \Leftarrow \boldsymbol{w} + \boldsymbol{x}^{(3)}$

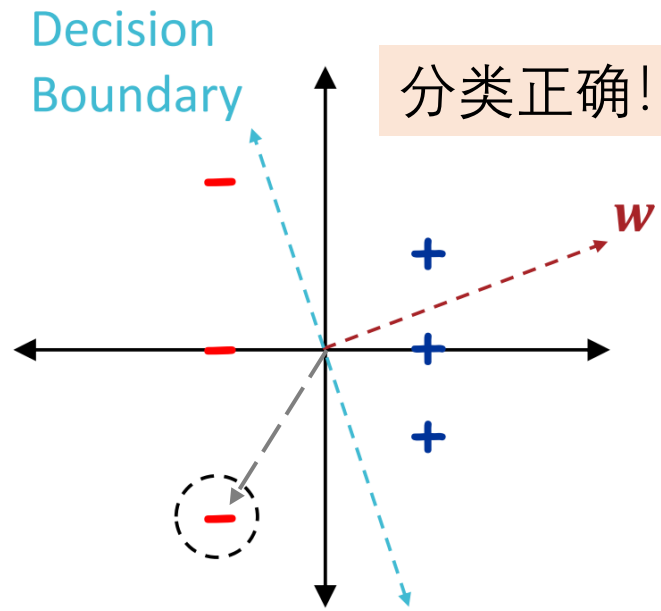
$$\boldsymbol{w} \Leftarrow \boldsymbol{w} + y^{(3)}\boldsymbol{x}^{(3)}$$

$$= \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

# 例2：调整权重 $\boldsymbol{w}$ (3)




分类正确!

| $x_1$ | $x_2$ | $y$ | $\hat{y}$ | 正确 |
|---|---|---|---|---|
| -1 | 2 | -1 | **+1** | No |
| 1 | 0 | +1 | **+1** | Yes |
| 1 | 1 | +1 | **-1** | No |
| -1 | 0 | -1 | **-1** | Yes |
| -1 | -2 | -1 | **+1** | No |

权重更新： $\boldsymbol{w} \Leftarrow \boldsymbol{w} + y^{(5)}\boldsymbol{x}^{(5)}$

$$= \begin{bmatrix} 2 \\ -1 \end{bmatrix} - 1 \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

# 调整权重 $\boldsymbol{w}$ 小结

对sign() 激活函数，如果预测正确，则不调整权重；否则调整权重：$\boldsymbol{w} \Leftarrow \boldsymbol{w} + y^{(i)}\boldsymbol{x}^{(i)}$

错误分类：
$$\begin{cases} \text{当 } \boldsymbol{w}^T\boldsymbol{x}^{(i)} > 0 \text{ 时}, \quad y^{(i)} = -1 \\ \text{当 } \boldsymbol{w}^T\boldsymbol{x}^{(i)} < 0 \text{ 时}, \quad y^{(i)} = +1 \end{cases} \longrightarrow -y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}\right) > 0$$

**损失函数**：
$$\mathcal{L}(\boldsymbol{w}) = -\sum_{i=1}^{n} y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)$$

将所有的错误累加

利用**梯度下降法**优化损失函数：$\dfrac{d\mathcal{L}(\boldsymbol{w})}{d\boldsymbol{w}} = -y^{(i)}\boldsymbol{x}^{(i)}$　　对每一次错误分类样本

减少损失

$$\boldsymbol{w} \Leftarrow \boldsymbol{w} - \frac{d\mathcal{L}(\boldsymbol{w})}{d\boldsymbol{w}} = \boldsymbol{w} + y^{(i)}\boldsymbol{x}^{(i)}$$

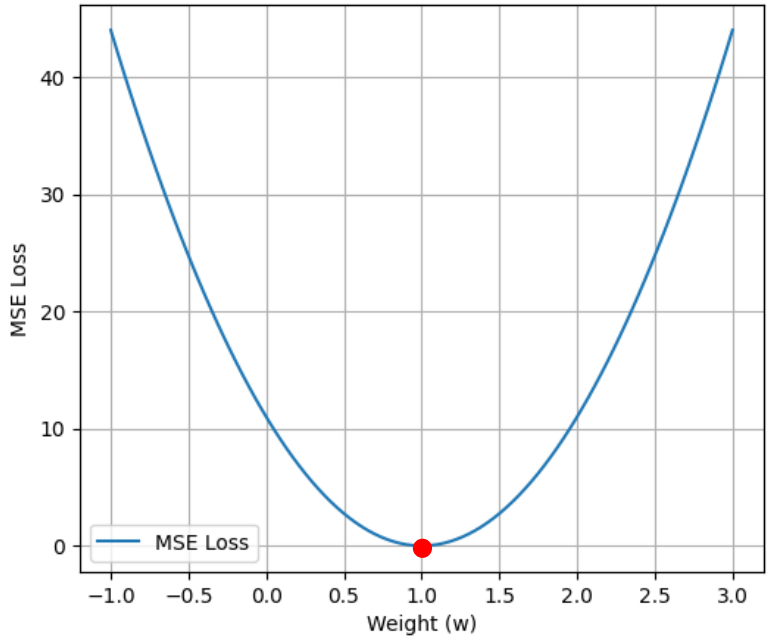
# 损失函数 loss function

计算模型预测、真实标签间的差异。

**均方差**损失函数:

**M**ean-**S**quare **E**rror

$$\mathcal{L} = \frac{1}{2n}(\boldsymbol{y} - \hat{\boldsymbol{y}})^2 = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$




$\boldsymbol{y}$：真实答案向量

$\hat{\boldsymbol{y}}$：预测输出向量　$\hat{\boldsymbol{y}} = f(\boldsymbol{w}^T\boldsymbol{x})$

$i$：第 $i$ 个数据

$n$：训练样本的个数($n$ >0的整数)

2：没有特别的含义

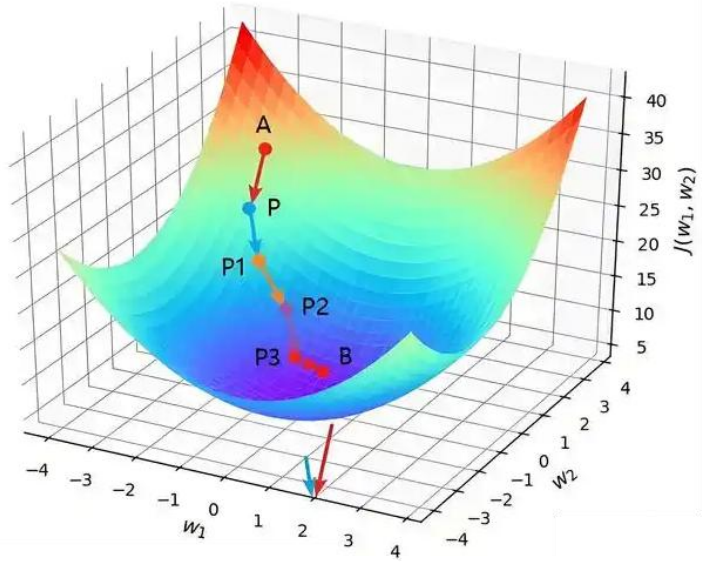
# 导数、偏导数、方向导数、梯度

- **导　数**：一元函数中，函数 $y = f(x)$ 在某一点处沿 $x$ 轴正方向的变化率。

- **偏导数**：多元函数中，函数 $y = f(x_0, x_1, \dots, x_n)$ 在某一点处沿某一坐标轴 $(x_0, x_1, \dots, x_n)$ 正方向的变化率。

- **方向导数**：函数在特定方向上的变化率。

- **梯度**：对于多元函数上的某一点，有很多方向导数，梯度的方向是函数在某一点增长最快的方向。

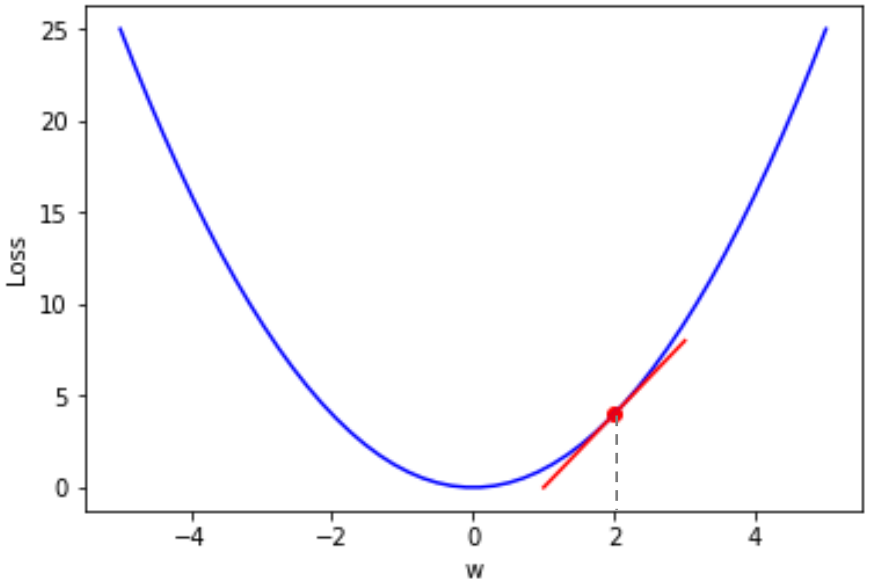


# 梯度下降法

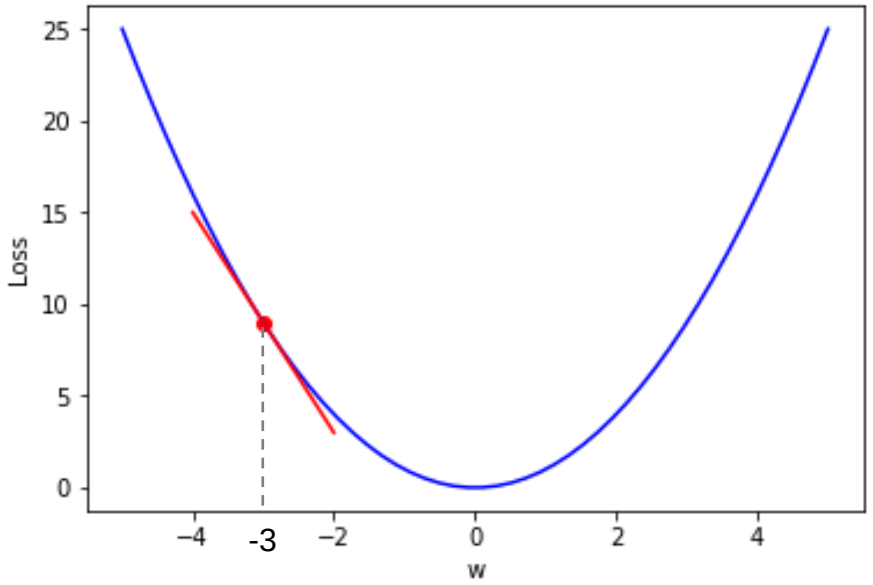
沿着**负梯度方向**去减小代价函数的值。

$$w \Leftarrow w - \eta \frac{\partial f}{\partial w}$$

假设：神经网络只有一个参数$w$




$w = 2$ 时，沿着**负**梯度方向，$w$ 的值变小。

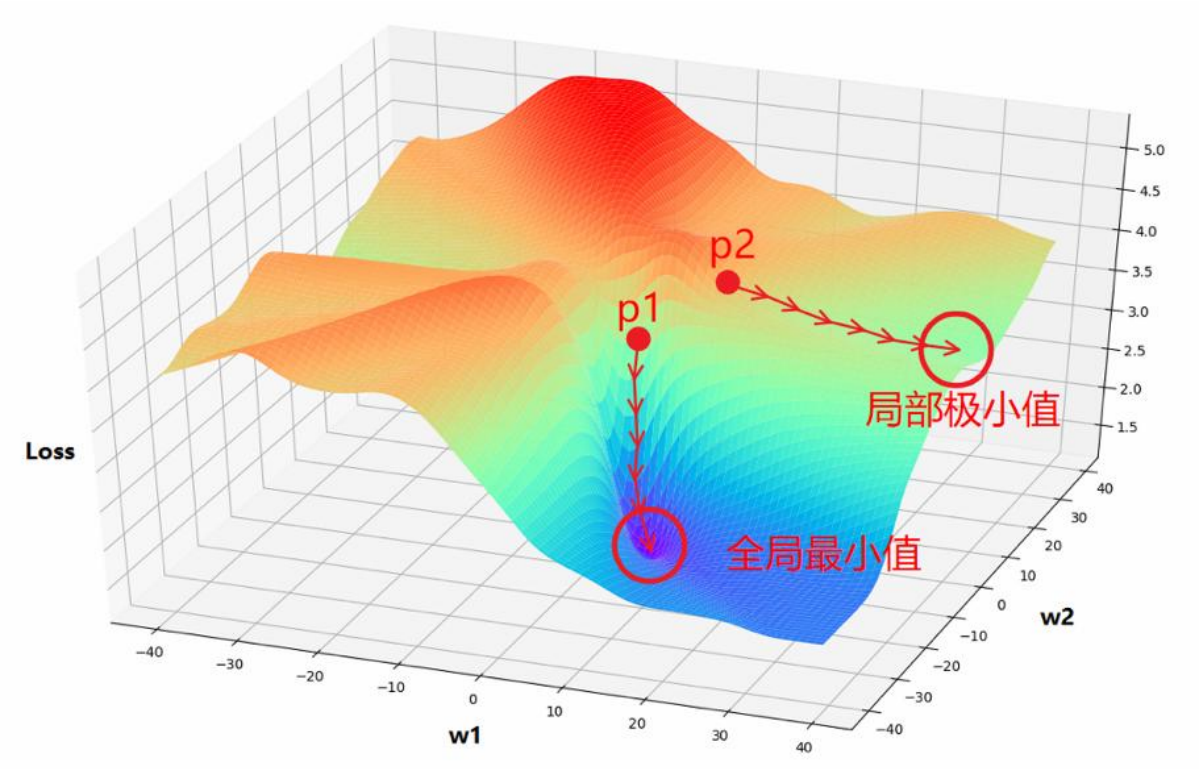


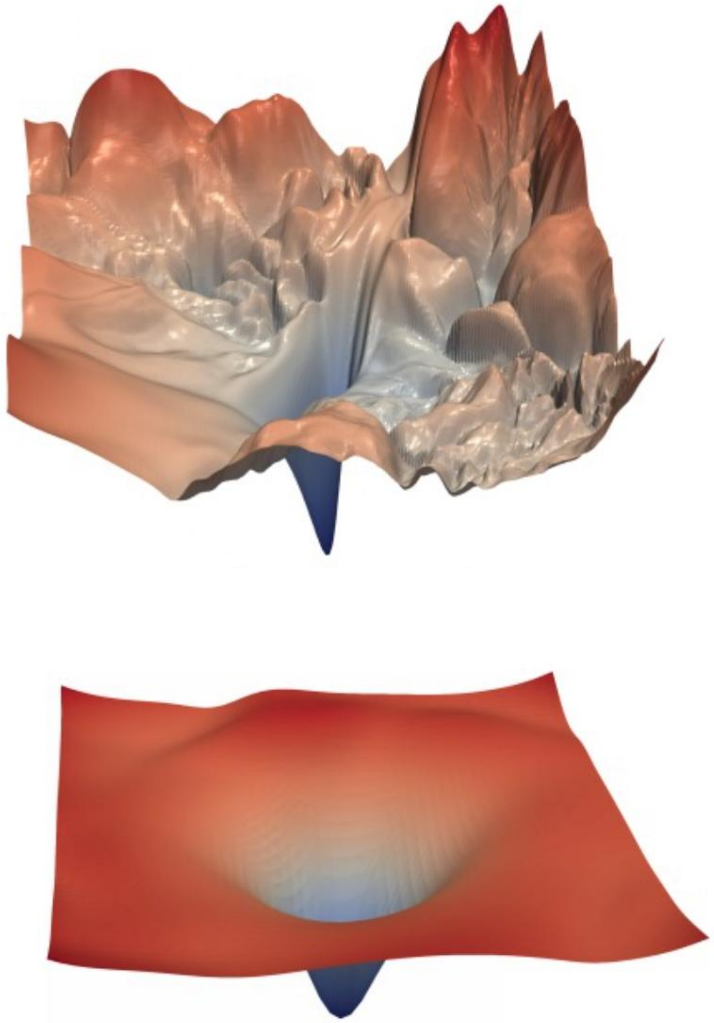

$w = -3$ 时，沿着**负**梯度方向，$w$ 的值变大。

# 梯度下降法：三维例子

**假设**：有两个参数 $w_1$ 和 $w_2$，

随机选取$w_1$和$w_2$初始值 p1 和 p2。



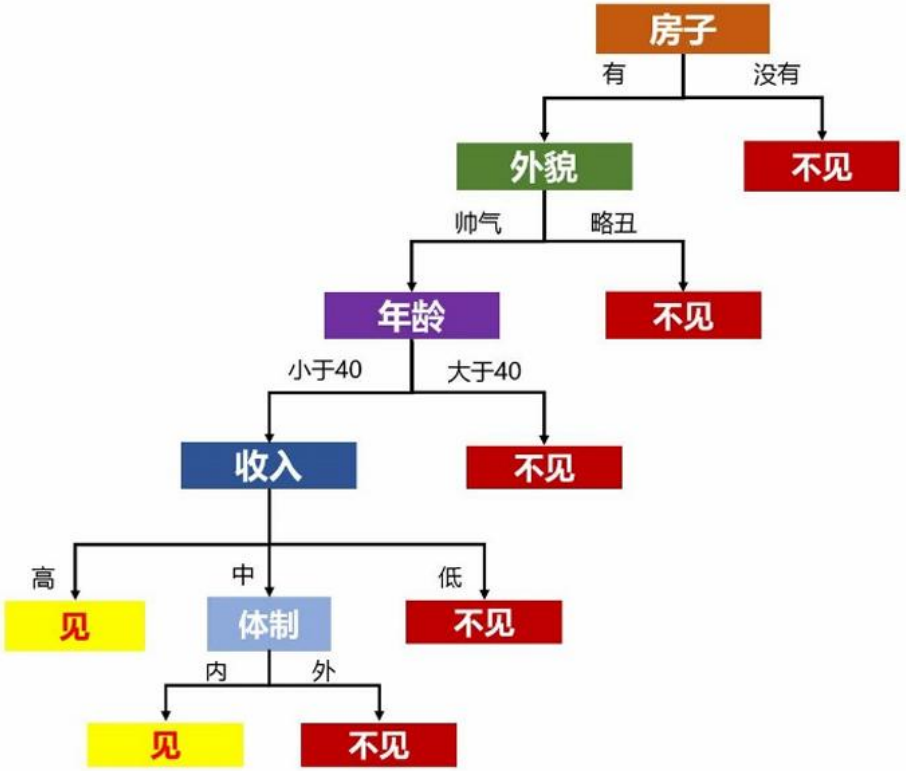
# 4

# 决策树

监督学习

# 决策树 Decision Tree

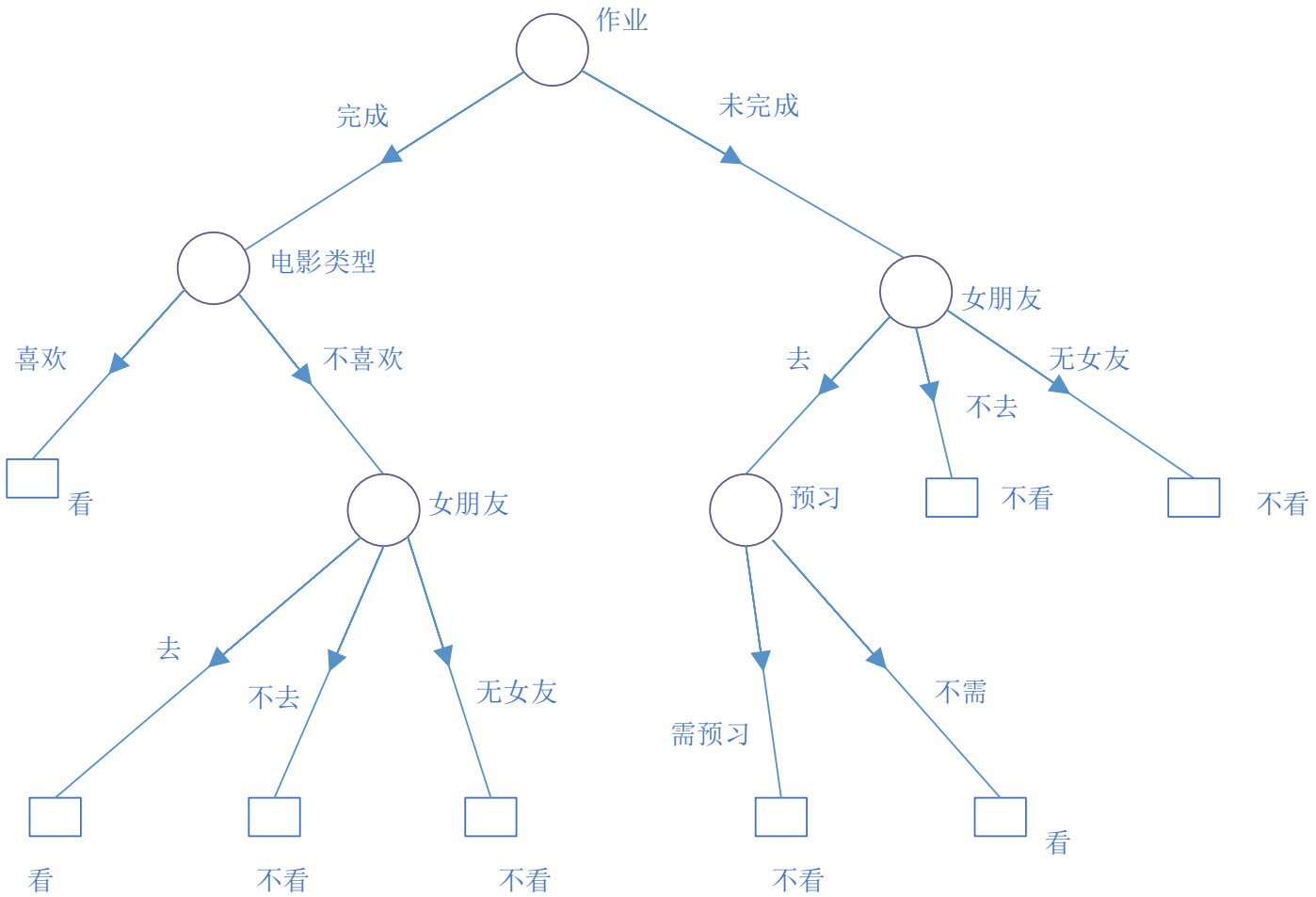
- 一种分层的决策结构，用于：**分类**、**回归**；
- **分治法**：难的问题 分解为 多个简单子问题；
- 推断速度快，可解释性强，应用广泛；
- 决策树**学习算法**：**ID3**、C4.5、CART...
- 可扩展成 随机森林、提升树 .....




相亲决策树

# 例3：看电影样本集 ==> 决策树

| | 女朋友 $A_1$ | 作业 $A_2$ | 预习 $A_3$ | 电影类型 $A_4$ | 决定 |
|---|---|---|---|---|---|
| 1 | 女友去 | 完成 | 需要 | 喜欢 | 看电影 |
| 2 | 女友去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 3 | 女友去 | 未完成 | 不需要 | 不喜欢 | 看电影 |
| 4 | 女友去 | 完成 | 需要 | 不喜欢 | 看电影 |
| 5 | 女友不去 | 完成 | 不需要 | 喜欢 | 看电影 |
| 6 | 女友不去 | 未完成 | 不需要 | 喜欢 | 不看 |
| 7 | 女友不去 | 完成 | 需要 | 喜欢 | 看电影 |
| 8 | 女友不去 | 完成 | 不需要 | 不喜欢 | 不看 |
| 9 | 女友不去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 10 | 无女友 | 完成 | 不需要 | 喜欢 | 看电影 |
| 11 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 12 | 无女友 | 未完成 | 需要 | 喜欢 | 不看 |
| 13 | 无女友 | 完成 | 不需要 | 不喜欢 | 不看 |
| 14 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 15 | 无女友 | 完成。 | 需要 | 喜欢 | 看电影 |



# ID3 算法

15个样本，4个特征、1个标签

| | 女朋友 $A_1$ | 作业 $A_2$ | 预习 $A_3$ | 电影类型 $A_4$ | 决定 |
|---|---|---|---|---|---|
| 1 | 女友去 | 完成 | 需要 | 喜欢 | 看电影 |
| 2 | 女友去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 3 | 女友去 | 未完成 | 不需要 | 不喜欢 | 看电影 |
| 4 | 女友去 | 完成 | 需要 | 不喜欢 | 看电影 |
| 5 | 女友不去 | 完成 | 不需要 | 喜欢 | 看电影 |
| 6 | 女友不去 | 未完成 | 不需要 | 喜欢 | 不看 |
| 7 | 女友不去 | 完成 | 需要 | 喜欢 | 看电影 |
| 8 | 女友不去 | 完成 | 不需要 | 不喜欢 | 不看 |
| 9 | 女友不去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 10 | 无女友 | 完成 | 不需要 | 喜欢 | 看电影 |
| 11 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 12 | 无女友 | 未完成 | 需要 | 喜欢 | 不看 |
| 13 | 无女友 | 完成 | 不需要 | 不喜欢 | 不看 |
| 14 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 15 | 无女友 | 完成。 | 需要 | 喜欢 | 看电影 |

样本集：$\boldsymbol{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$

$N$：样本数

$x_n$：特征向量

$y_n$：标签答案

不确定性
- 若所有样本答案都一样，不确定性=0
- 若两种答案数量相同，很难选择，不确定性最高

1979年，Quinlan用**熵**表示不确定性：**不确定性越大，熵越大**

$$H(\boldsymbol{D}) = -\sum_{k=1}^{K} \frac{c_k}{N} \log_2 \frac{c_k}{N}$$

概率

$K$：总的分类类型

$c_k$：第$k$类样本数目

选择特征$A$后的**条件熵**：

$$H(\boldsymbol{D}|A) = -\sum_{i\in[1,I]} \frac{N_i}{N} \sum_{k=1}^{K} \frac{c_{ik}}{N_i} \log_2 \frac{c_{ik}}{N_i}$$

$N_i$：子样本集$D_i$的样本数

$c_{ik}$：子样本集$D_i$第$k$类样本数

选择特征$A$后的**熵增益**：

$$G(\boldsymbol{D}, A) = H(\boldsymbol{D}) - H(\boldsymbol{D}|A)$$

选择**熵增益最大**的特征作为**根节点**

# 例3：看电影样本集 ==> 决策树 (2)

| | 女朋友 $A_1$ | 作业 $A_2$ | 预习 $A_3$ | 电影类型 $A_4$ | 决定 |
|---|---|---|---|---|---|
| 1 | 女友去 | 完成 | 需要 | 喜欢 | 看电影 |
| 2 | 女友去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 3 | 女友去 | 未完成 | 不需要 | 不喜欢 | 看电影 |
| 4 | 女友去 | 完成 | 需要 | 不喜欢 | 看电影 |
| 5 | 女友不去 | 完成 | 不需要 | 喜欢 | 看电影 |
| 6 | 女友不去 | 未完成 | 不需要 | 喜欢 | 不看 |
| 7 | 女友不去 | 完成 | 需要 | 喜欢 | 看电影 |
| 8 | 女友不去 | 完成 | 不需要 | 不喜欢 | 不看 |
| 9 | 女友不去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 10 | 无女友 | 完成 | 不需要 | 喜欢 | 看电影 |
| 11 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 12 | 无女友 | 未完成 | 需要 | 喜欢 | 不看 |
| 13 | 无女友 | 完成 | 不需要 | 不喜欢 | 不看 |
| 14 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 15 | 无女友 | 完成。 | 需要 | 喜欢 | 看电影 |

数据集D中“看”7项，“不看”8项。数据集的经验**熵**：

$$H(\boldsymbol{D}) = -\frac{7}{15}\log_2\frac{7}{15} - \frac{8}{15}\log_2\frac{8}{15} = \mathbf{0.9966}$$

特征$A_1$增益：

$$G(\boldsymbol{D}, A_1) = H(\boldsymbol{D}) - H(\boldsymbol{D}|A_1)$$

$$= 0.9966 - \frac{4}{15}\left(-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right)$$

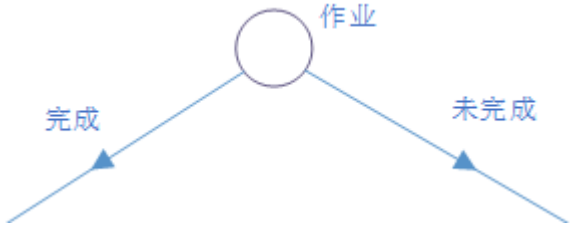
$$-\frac{5}{15}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right)$$

$$-\frac{6}{15}\left(-\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6}\right) = \mathbf{0.0866}$$

$G(\boldsymbol{D}, A_2) = \mathbf{0.2876}$ 最大：选择“作业”作为**根节点**

$G(\boldsymbol{D}, A_3) = \mathbf{0.027}$

$G(\boldsymbol{D}, A_4) = \mathbf{0.0366}$



# 例3：看电影样本集 ==> 决策树 (3)

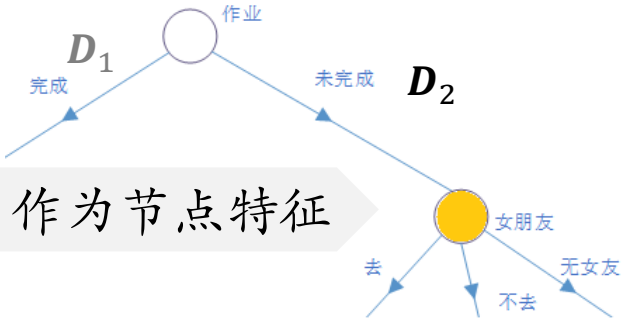
| | 女朋友 $A_1$ | 作业 $A_2$ | 预习 $A_3$ | 电影类型 $A_4$ | 决定 |
|---|---|---|---|---|---|
| 1 | 女友去 | 完成 | 需要 | 喜欢 | 看电影 |
| 2 | 女友去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 3 | 女友去 | 未完成 | 不需要 | 不喜欢 | 看电影 |
| 4 | 女友去 | 完成 | 需要 | 不喜欢 | 看电影 |
| 5 | 女友不去 | 完成 | 不需要 | 喜欢 | 看电影 |
| 6 | 女友不去 | 未完成 | 不需要 | 喜欢 | 不看 |
| 7 | 女友不去 | 完成 | 需要 | 喜欢 | 看电影 |
| 8 | 女友不去 | 完成 | 不需要 | 不喜欢 | 不看 |
| 9 | 女友不去 | 未完成 | 需要 | 不喜欢 | 不看 |
| 10 | 无女友 | 完成 | 不需要 | 喜欢 | 看电影 |
| 11 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 12 | 无女友 | 未完成 | 需要 | 喜欢 | 不看 |
| 13 | 无女友 | 完成 | 不需要 | 不喜欢 | 不看 |
| 14 | 无女友 | 未完成 | 不需要 | 喜欢 | 不看 |
| 15 | 无女友 | 完成 | 需要 | 喜欢 | 看电影 |

选择“**作业**”为根节点后，样本集分为两个子样本集：

子数据集：$D_1$“作业完成”，$\boldsymbol{D_2}$“**作业未完成**”

数据集$\boldsymbol{D_2}$中“**看**”1项，“**不看**”6项。$\boldsymbol{D_2}$的经验熵：

$$H(\boldsymbol{D_2}) = -\frac{1}{7}\log_2\frac{1}{7} - \frac{6}{7}\log_2\frac{6}{7} = \mathbf{0.592}$$




在第2层，选择“**女朋友**”作为节点特征

每个特征的增益：

$$G(\boldsymbol{D_2}, A_1) = \mathrm{H}(\boldsymbol{D_2}) - (\boldsymbol{D_2}|A_1)$$

$$= \mathbf{0.592} - \frac{2}{7}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{2}{7}(0) - \frac{3}{7}(0) = \mathbf{0.31}$$

**最大**

$$G(\boldsymbol{D_2}, A_3) = \mathrm{H}(\boldsymbol{D_2}) - (\boldsymbol{D_2}|A_3) = \mathbf{0.128}$$

$$G(\boldsymbol{D_2}, A_4) = \mathrm{H}(\boldsymbol{D_2}) - (\boldsymbol{D_2}|A_4) = \mathbf{0.2}$$

# ID3 算法

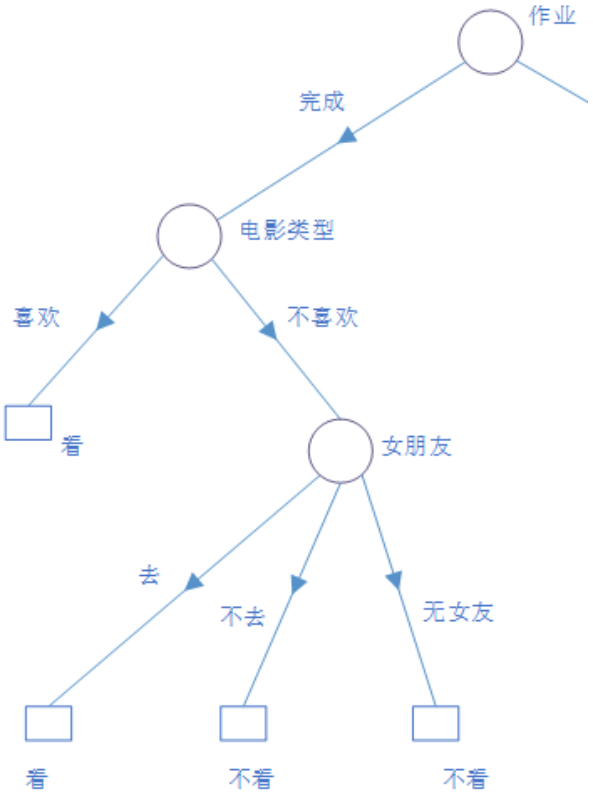
1. **输入与输出**
   - **输入**：训练数据集 $\boldsymbol{D}$，特征集合 $\boldsymbol{A}$，目标分类标签 $Y$
   - **输出**：决策树模型
2. **终止条件**
   - 当前节点所有样本属于同一类别（纯度最大）
   - 无剩余特征可用，或样本数量低于阈值（多数表决确定类别）
3. **递归构建树**

   1) 计算数据集 $\boldsymbol{D}$ 的 **信息熵** $H(\boldsymbol{D})$;

   2) 计算每个特征 $A_i$ 的 **信息增益** $G(\boldsymbol{D}, A_i)$ ;

   3) 选择 **信息增益最大** 的特征作为当前节点的分裂特征;

   4) 对每个子节点**递归**调用上述步骤，直到满足终止条件。



# 代码3：决策树ID3

02-决策树ID3.ipynb

```
# <1> 提取数据集
from sklearn import datasets

# 加载 yuān鸢尾花(iris)数据集
iris = datasets.load_iris()

# 花萼的长、宽、花瓣的长、宽
X = iris.data
# 分类 [0-setosa山鸢尾, 1-versicolor杂色鸢尾, 2-virginica]
y = iris.target

# <2> 数据集划分
from sklearn.model_selection import train_test_split

# 将数据集分为：训练集、测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size    = 0.3,  # 测试集占30%，隐含 训练集占70%
    random_state = 4,    # 保证每次运行程序时划分结果一致
    shuffle      = True, # 分割前对数据集进行洗牌
    stratify     = y   )  # 训练集和测试集中各类别样本的比例
```

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 150 | 4 | setosa | versicolor | virginica |
| 2 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| 3 | 4.9 | 3 | 1.4 | 0.2 | 0 |
| 4 | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| 5 | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| 6 | 5 | 3.6 | 1.4 | 0.2 | 0 |

```
# <3> 训练模型、预测
from sklearn.tree import DecisionTreeClassifier

# 创建决策树分类器,用信息熵作为划分标准。默认CART分类树
I_model = DecisionTreeClassifier(criterion='entropy')

I_model.fit(X_train, y_train)     # 在训练集上进行训练
y_pred = I_model.predict(X_test) # 训练后预测


# <4> 计算模型准确率
from sklearn.metrics import accuracy_score

# 准确率
acc = accuracy_score(y_pred, y_test)
print(f"准确率: {acc :.2f}")
```

运行结果：

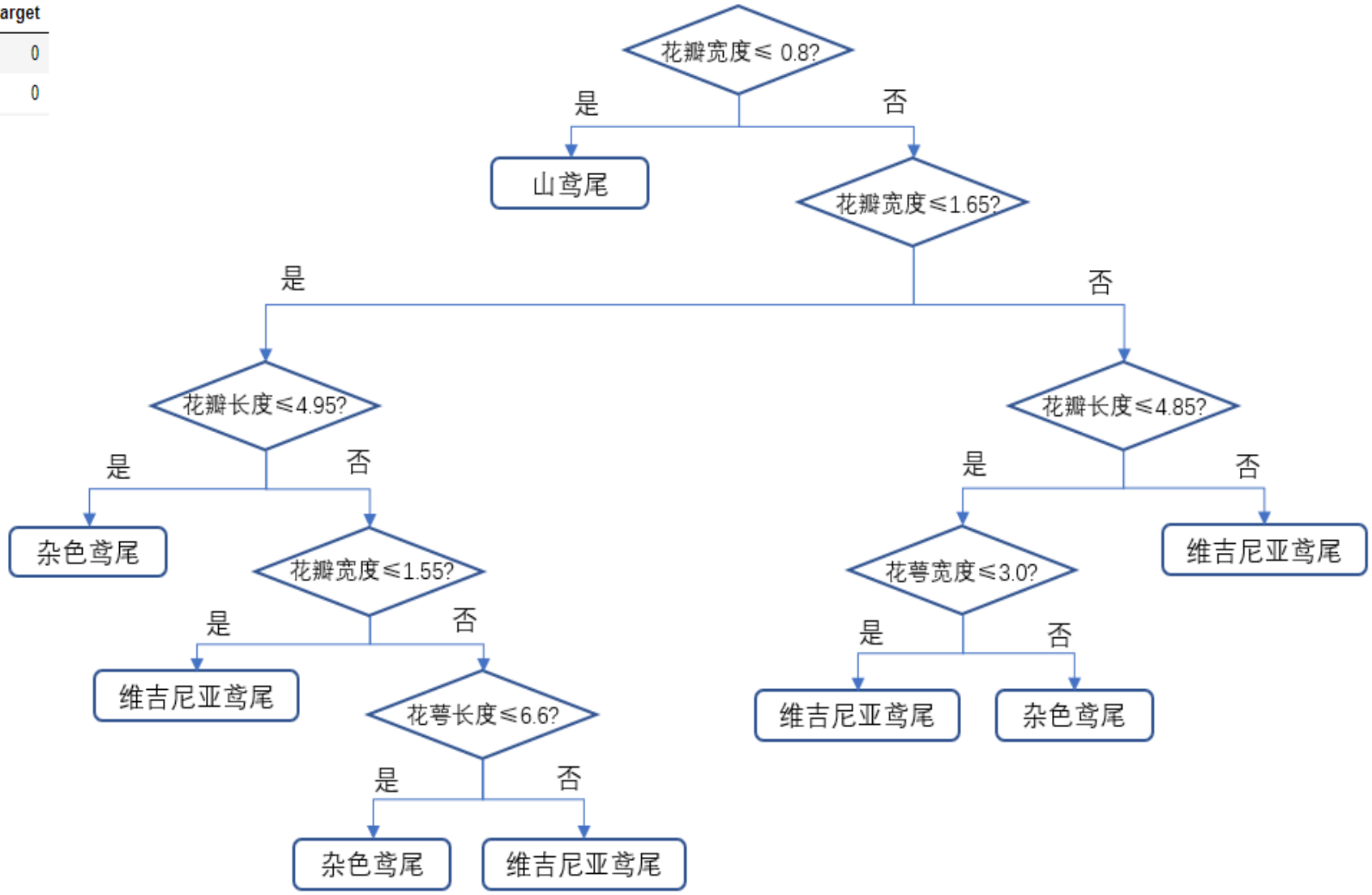
准确率: 0.98

```
# <5> 显示决策树
from sklearn import tree

# 生成决策树的文本描述
text_representation = tree.export_text(I_model)
print(text_representation)
```

# 代码3：ID3 绘制的决策树

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |

```
|--- feature_3 <= 0.80
|   |--- class: 0
|--- feature_3 >  0.80
|   |--- feature_3 <= 1.65
|   |   |--- feature_2 <= 4.95
|   |   |   |--- class: 1
|   |   |--- feature_2 >  4.95
|   |   |   |--- feature_3 <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- feature_3 >  1.55
|   |   |   |   |--- feature_0 <= 6.60
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  6.60
|   |   |   |   |   |--- class: 2
|   |--- feature_3 >  1.65
|   |   |--- feature_2 <= 4.85
|   |   |   |--- feature_1 <= 3.00
|   |   |   |   |--- class: 2
|   |   |   |--- feature_1 >  3.00
|   |   |   |   |--- class: 1
|   |   |--- feature_2 >  4.85
|   |   |   |--- class: 2
```



# 提 纲

① 传统机器学习

**监督学习**

② 线性回归（一元、多元）、Logistic回归

③ 二分类、多分类

④ 决策树

**⑤ 支持向量机 SVM**

**无监督学习**

⑥ K均值聚类 （聚类）

⑦ 主成分分析 （降维）

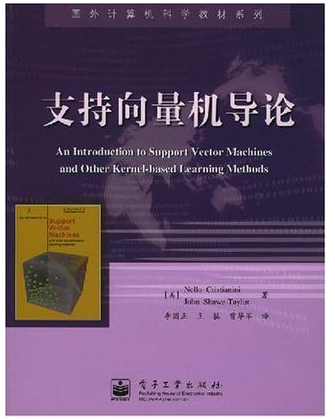
# 支持 向量 机 (SVM)

Support Vector Machines

在深度学习之前，有着非常辉煌的历史，占据着相当重要的地位。

因其背后有深刻、成熟的数学理论，在小数据样本时表现优异。

由前苏联统计学家 Vapnik 在1960's 提出，1990's 完善。



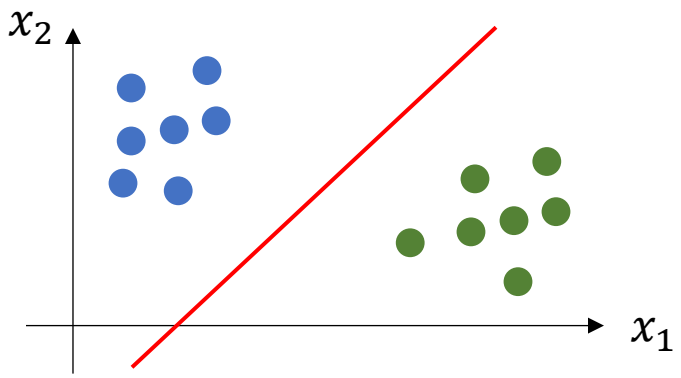

**优 点**

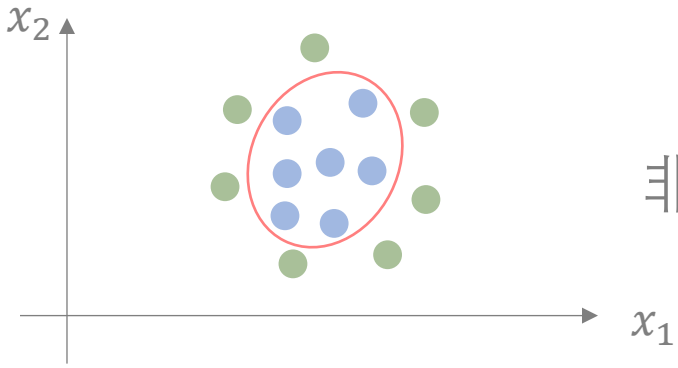
- 全局最优解（凸优化）
- 泛化能力强（最大化间隔）
- 省内存。依赖少量支持向量，而非全体样本。

**缺点**

- 大规模数据训练效率低。
- 核函数、参数敏感。
- 对缺失数据、噪声敏感。

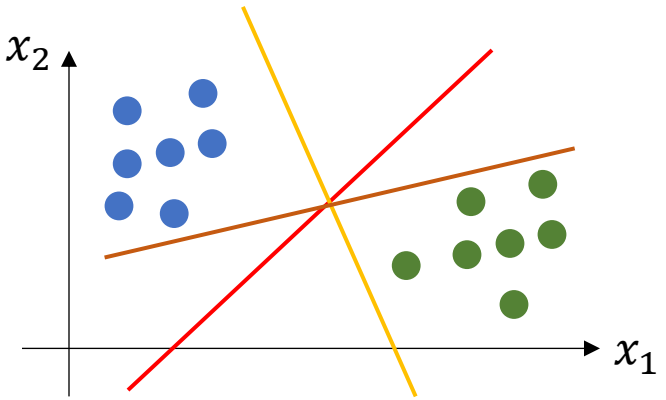
# 线性可分、线性不可分



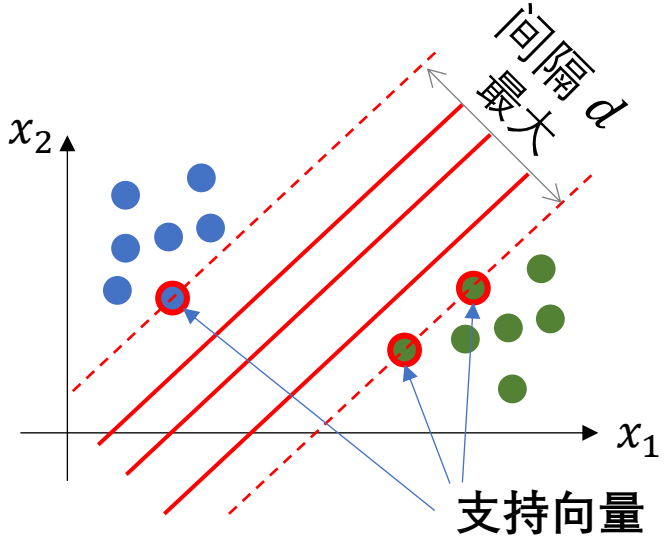




非线性可分

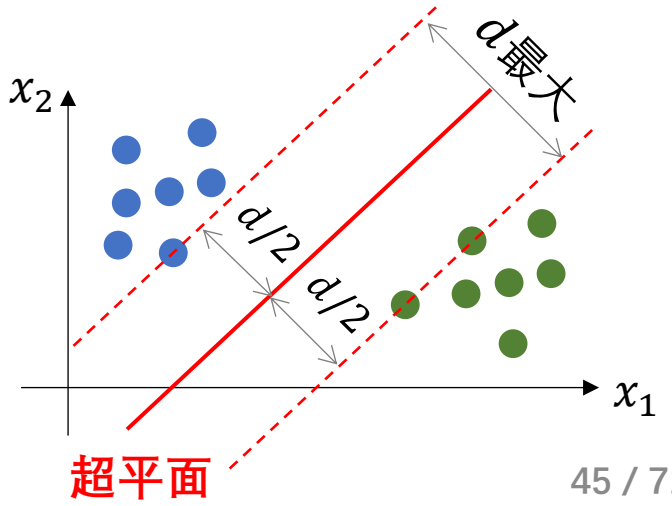
- 怎么分？有无穷种




- 如何判定**最好**？




- 如何判定**唯一**？



# 机器学习过程

1) 先确定一个**模型**，其中有待定参数。

2) 再用**训练数据**确定其中的**待定参数**。

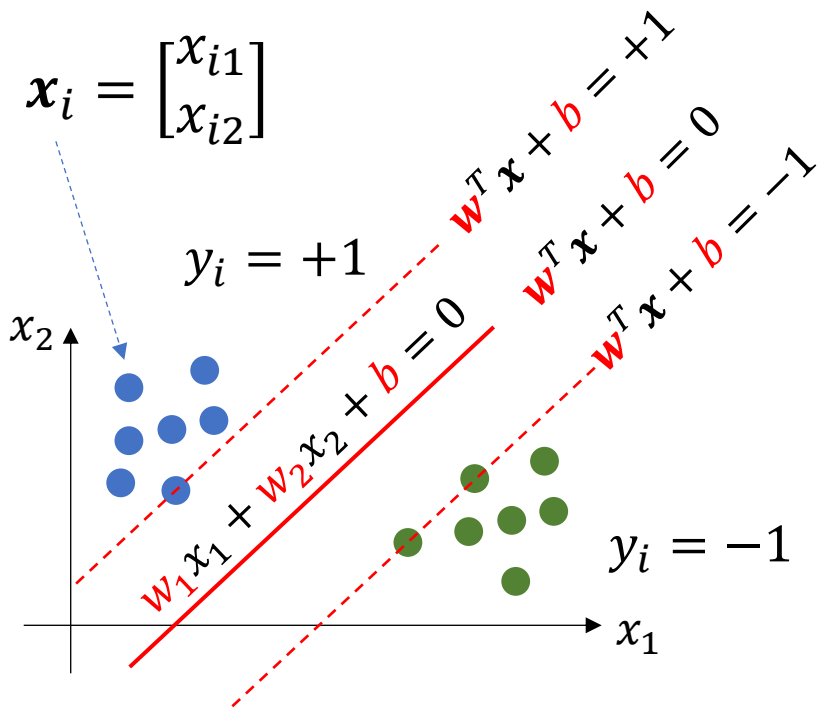
- 训练数据、标签：

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_i, y_i), \cdots, (\boldsymbol{x}_N, y_N)$$

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

- 线性模型：$\boldsymbol{w}^T\boldsymbol{x} + b = 0$

- 待求参数：$\boldsymbol{w}, b$



# 线性可分的数学描述

$\{(x_i, y_i)\}_{i=1\sim N}$

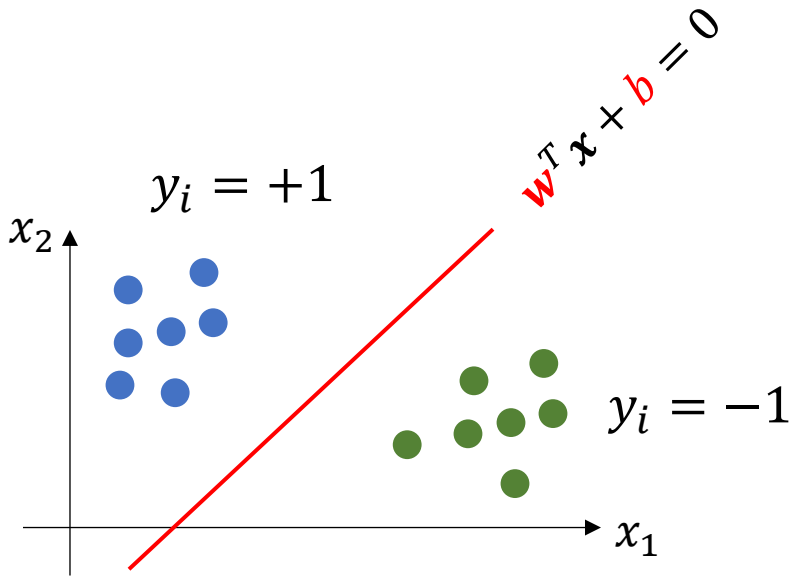
存在 $\boldsymbol{w}, b$，使：

对 任意 $i = 1\sim N$，有：

(1) 若 $y_i = +1$，则 $\boldsymbol{w}^T\boldsymbol{x}_i + b \geq 0$

(2) 若 $y_i = -1$，则 $\boldsymbol{w}^T\boldsymbol{x}_i + b < 0$

$$y_i\left[\boldsymbol{w}^T\boldsymbol{x}_i + b\right] \geq 0$$



# 凸优化

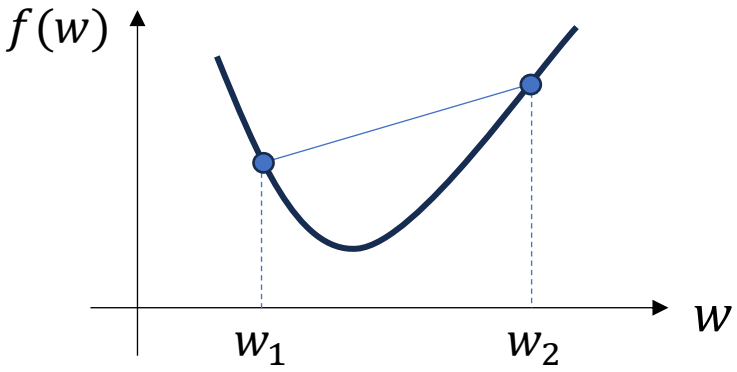
数学最优化的一个子领域，研究定义于凸集中的**凸函数**最小化的问题。

- **线性优化问题**

求解最大间隔划分超平面
对应于优化问题。

**最小化**：

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

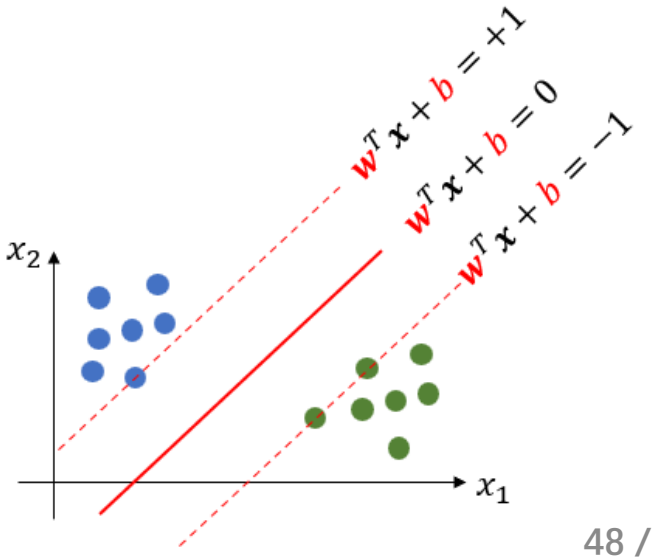



点 $\boldsymbol{x}\,(x_0, y_0)$ 到平面 $w_1 x + w_2 y + b = 0$ 的距离：

$$d = \frac{|w_1 x_0 + w_2 y_0 + b|}{\sqrt{w_1^2 + w_2^2}} = \frac{|\boldsymbol{w}^T \boldsymbol{x} + b|}{\|\boldsymbol{w}\|} = \frac{\text{常数}}{\|\boldsymbol{w}\|}$$

**限制条件**：

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 \qquad (i = 1 \sim N)$$




可用拉格朗日乘子法，得到拉格朗日函数求解...

# 原问题 → 对偶问题

**最小化**：$f(\boldsymbol{w})$

**限制条件**：$\begin{cases} g_i(\boldsymbol{w}) \le 0 \quad (i = 1 \sim K) \\ h_i(\boldsymbol{w}) = 0 \quad (i = 1 \sim M) \end{cases}$

**最大化**：$\theta(\boldsymbol{\alpha}, \boldsymbol{\beta}) = min[f(\boldsymbol{w}) + \boldsymbol{\alpha}^{\mathrm{T}} g(\boldsymbol{w}) + \boldsymbol{\beta}^{\mathrm{T}} h(\boldsymbol{w})]$

对所有$\boldsymbol{w}$求最小值

**限制条件**：$\alpha_i \ge 0 \quad (i = 1 \sim K)$

有时对偶问题更容易求解

## 强对偶定理

若$f(\boldsymbol{w})$为凸函数，且$g(\boldsymbol{w}) = A\boldsymbol{w} + b$，$h(\boldsymbol{w}) = C\boldsymbol{w} + d$，则：

此优化问题的原问题、对偶问题间距为0，即 $f(\boldsymbol{w}^*) = \theta(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$

原问题的解　　对偶原问题的解

## KKT条件

对 $\forall\ i = 1 \sim K$：　或者 $\alpha_i^* = 0$

或者 $g_i^*(\boldsymbol{w}^*) = 0$

# 原问题　　示例　　对偶问题

**最小化**：　$f(x) = x^2$

**限制条件**：　$x - 1 \geq 0$

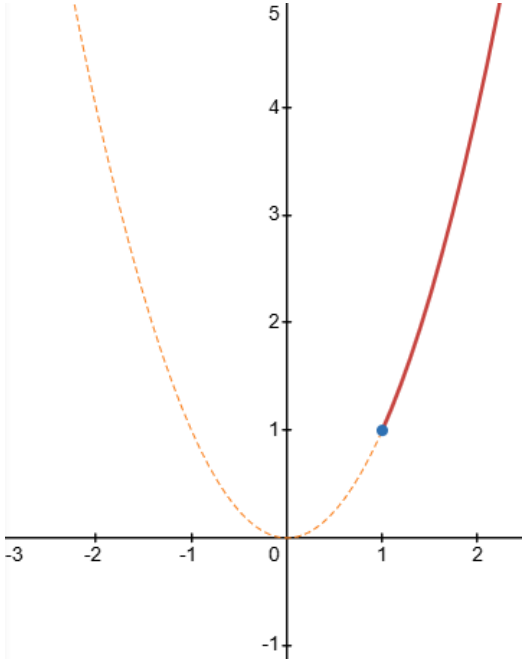
$x = 1$ 时，　$min\, f(x) = 1$

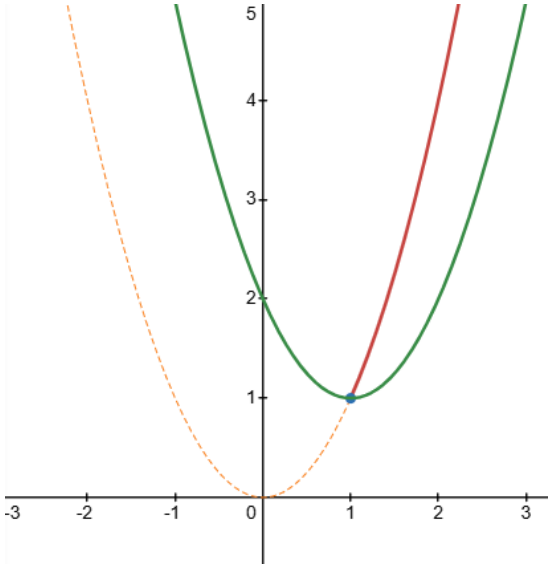
**最大化**：　$\theta(\alpha) = min[x^2 - \alpha(x-1)] = min\, L(x, \alpha)$

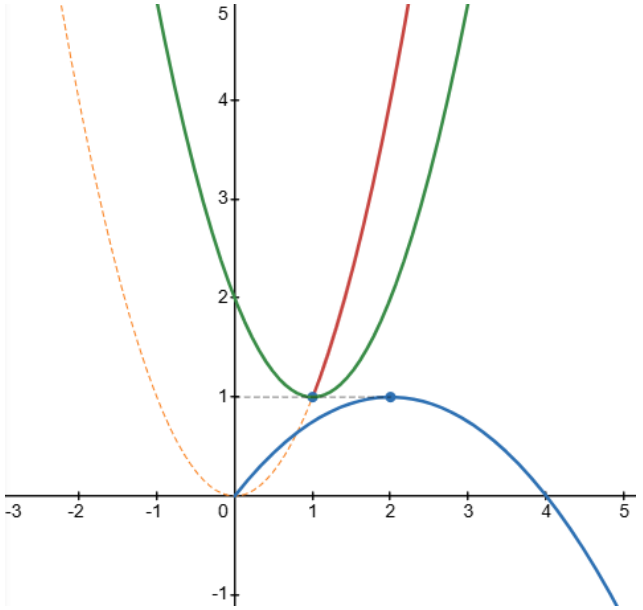
**限制条件**：　$\alpha \geq 0$

求 $min\, L(x, \alpha)$：　$\frac{\partial L}{\partial x} = 0$　➡　$x = \frac{\alpha}{2}$　　$\theta(\alpha) = -\frac{\alpha^2}{4} + \alpha$

求 $max\, \theta(\alpha)$：　$\frac{\partial \theta}{\partial \alpha} = 0$　➡　$\alpha = 2$　$(x = 1)$　　$\theta(\alpha) = 1$



# 非线性模型 ⟶ 线性模型

牛顿万有引力定律: $F = G\dfrac{m_1 m_2}{r^2}$ （非线性）

坐标变换: $(m_1, m_2, r) \longmapsto (x, y, z) = (\ln m_1, \ln m_2, \ln r)$

$$g(x, y, z) = \ln F = \ln C + \ln m_1 + \ln m_2 - 2\ln r$$

$$g(x, y, z) = c + x + y - 2z \quad （线性）$$

# 非线性模型

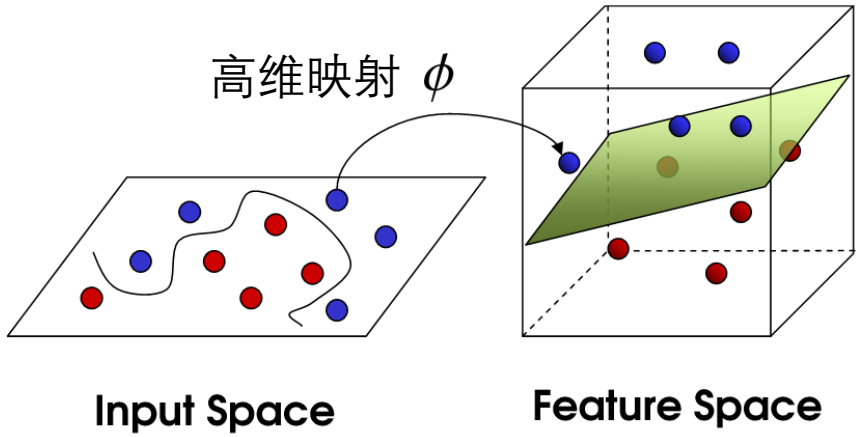
Kernel function

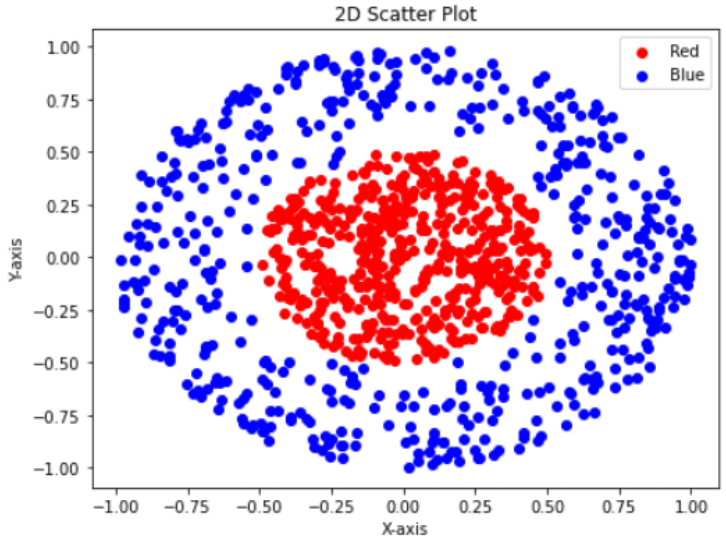
**核函数** $K$：通过非线性变换 $\phi$，

将低维输入空间$\boldsymbol{x}$映射到高维特征空间$\phi(\boldsymbol{x})$。

$$\boldsymbol{x} \xrightarrow{\phi} \phi(\boldsymbol{x})$$

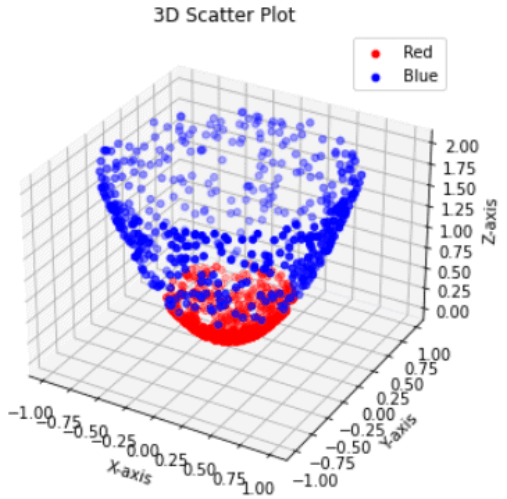
低维　　高维

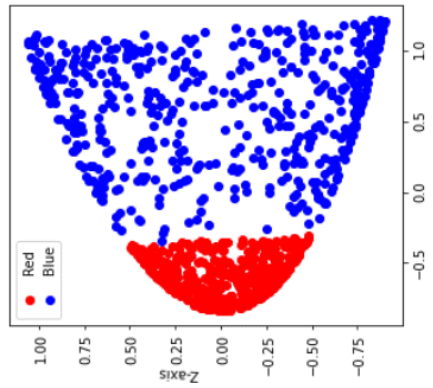



$$\phi(v) = \phi\big((x, y)\big)$$

$$= (x, y, 2x^2 + 2y^2)$$



# Mercer 定理

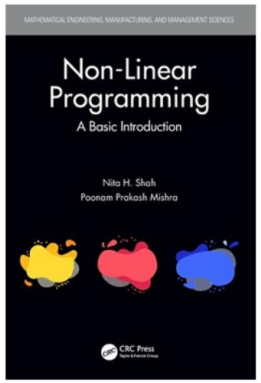
$\boldsymbol{K}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ 能写成 $\boldsymbol{\phi}(\boldsymbol{x}_1)^T \cdot \boldsymbol{\phi}(\boldsymbol{x}_2)$ 的充要条件：

①（交换性） $\boldsymbol{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{K}(\boldsymbol{x}_2, \boldsymbol{x}_1)$

②（半正定性） $\forall\ c_i, \boldsymbol{x}_i\ \ (i = 1 \sim N)$，有：$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

**高斯核**：$\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}}$

(RBF核)


- 核函数避免了“维数灾难”，大大减小了计算量。
- 无需知道非线性变换函数$\phi$的形式和参数。

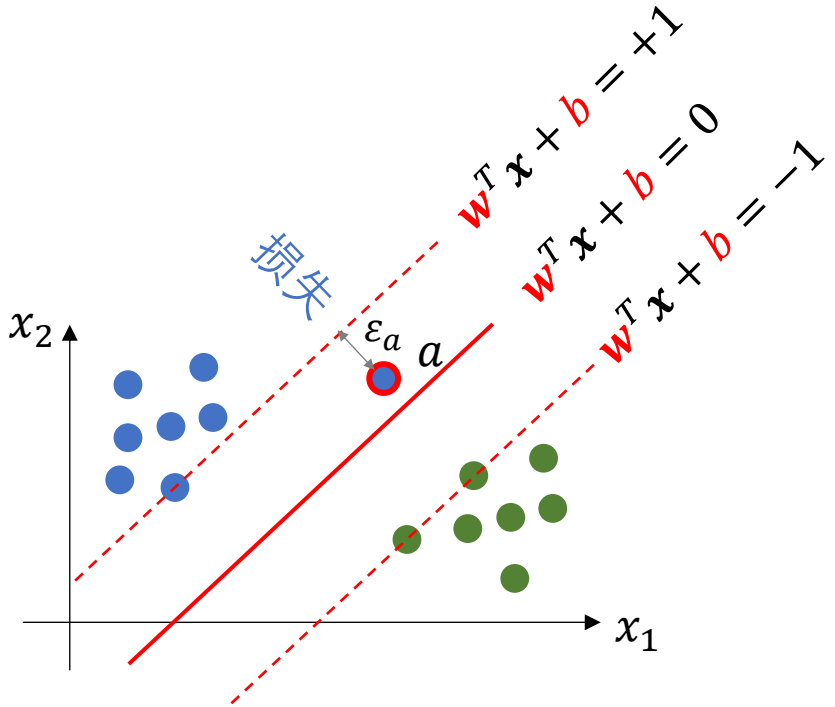
# 软间隔 Soft Margin

**硬间隔**限制条件: $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$

点$a$违背限制条件: $y_a(\boldsymbol{w}^T\boldsymbol{x}_a + b) < 1$

点 $a$ 误差: $\varepsilon_a = 1 - y_a(\boldsymbol{w}^T\boldsymbol{x}_a + b)$

$\varepsilon_a = 1 - A$

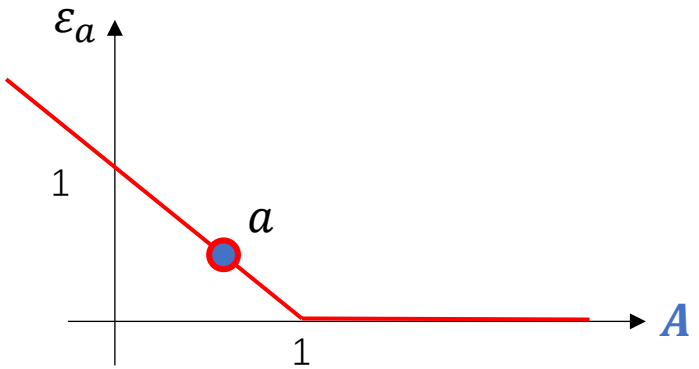



**铰链损失函数**: $\varepsilon_a = max\left(0,\ 1 - y_a(\boldsymbol{w}^T\boldsymbol{x}_a + b)\right)$

Hinge Loss

等价于 ⇨

$$\begin{cases} \varepsilon_a \geq 0 \\ y_a\left[\boldsymbol{w}^T\boldsymbol{x}_a + b\right] + \varepsilon_a \geq 1 \end{cases}$$



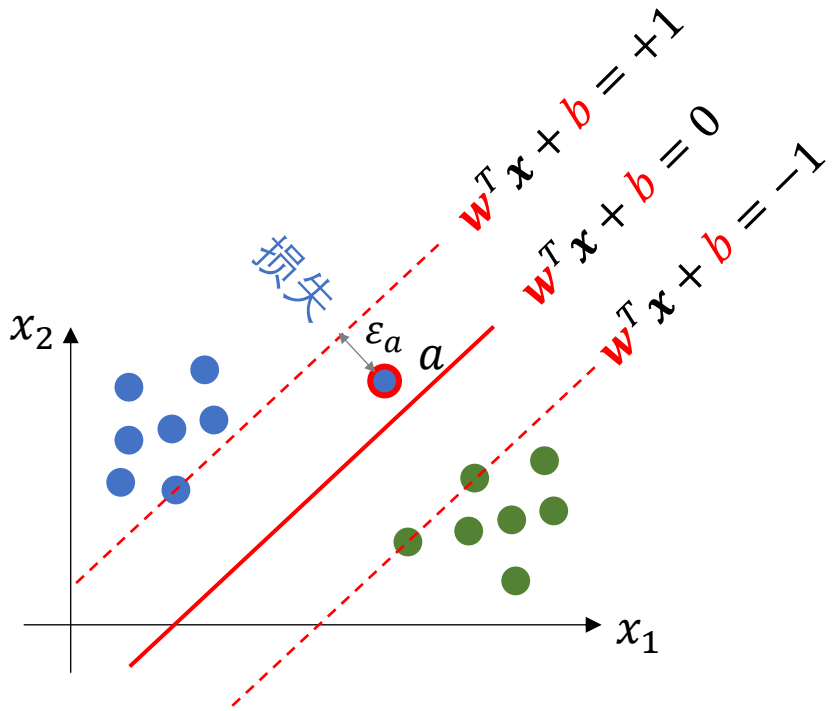
# 非线性优化

**最小化**: $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\varepsilon_i$

正则项

松弛变量

事先设定的惩罚参数，$\varepsilon_i$的容忍度，当 $C=\infty$ 时，$\varepsilon_i=0$，即硬间隔




**限制条件**: $\varepsilon_i \geq 0$

非线性映射

$$y_i\left[\boldsymbol{w}^T\phi(\boldsymbol{x}_i)+b\right] \geq 1-\varepsilon_i \quad (i=1\sim N)$$

可能是无限维的，$\boldsymbol{w}$ 也必须是相同维度的，如何求解?

只要用一个核函数，无需$\phi(\boldsymbol{x}_i)$的显示表达式，就能求解。

$$\boldsymbol{K}(\boldsymbol{x}_1,\boldsymbol{x}_2)=\phi(\boldsymbol{x}_1)^T\cdot\phi(\boldsymbol{x}_2)$$

两个无限向量的内积

# SVM 算法：**训练过程**

输入：$\{(x_i, y_i)\}_{i=1\sim N}$

转化为**对偶问题**求解

**最大化**：

$$\theta(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

用核函数

$$\theta(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

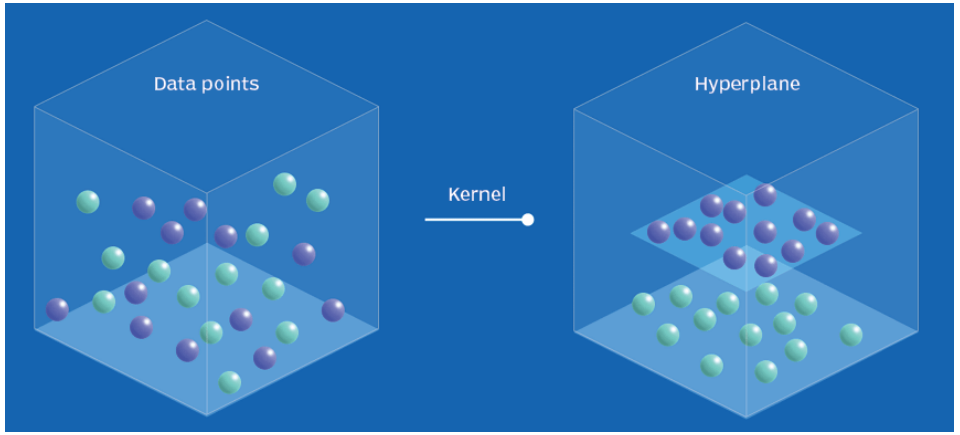
拉格朗日乘子

可采用**SMO**算法求解

**S**equential **M**inimal **O**ptimization

**限制条件**：

$$\begin{cases} 0 \le \alpha_i \le C \qquad (i = 1\sim N) \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases}$$



# SVM 算法：测试过程

测试样本 $\boldsymbol{x}$，

$$\begin{cases} \text{若 } \boldsymbol{w}^T\phi(\boldsymbol{x}) + b \geq 0\text{，则 } y_i = +1 \\ \text{若 } \boldsymbol{w}^T\phi(\boldsymbol{x}) + b < 0\text{，则 } y_i = -1 \end{cases}$$

$$\boldsymbol{w}^T\phi(\boldsymbol{x}) = \sum_{i=1}^{N}[\alpha_i y_i \phi(x_i)]^T\phi(\boldsymbol{x})$$

$$= \sum_{i=1}^{N}\alpha_i y_i\, \phi(x_i)^T\phi(\boldsymbol{x}) = \sum_{i=1}^{N}\alpha_i y_i\, K(x_i, \boldsymbol{x})$$

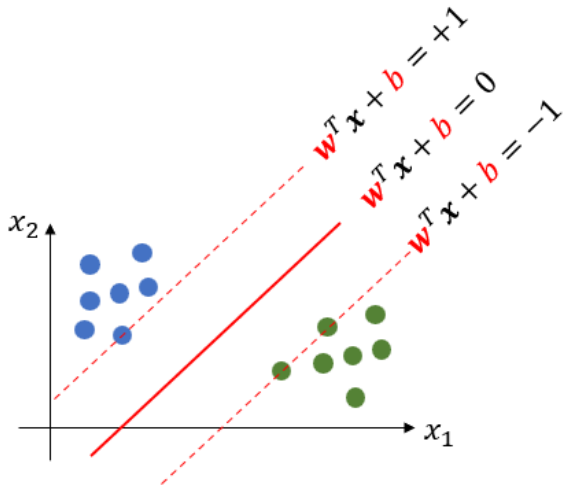
由KKT条件可以求出：

$$b = \frac{1 - y_i\sum_{j=1}^{N}\alpha_j y_j\, K(x_i, x_j)}{y_i}$$

$$\Rightarrow \begin{cases} \text{若 } \sum_{i=1}^{N}\alpha_i y_i\, K(x_i, \boldsymbol{x}) + \frac{1-y_i\sum_{j=1}^{N}\alpha_j y_j\, K(x_i,x_j)}{y_i} \geq 0\text{，则 } y_i = +1 \\ \text{若 } \sum_{i=1}^{N}\alpha_i y_i\, K(x_i, \boldsymbol{x}) + \frac{1-y_i\sum_{j=1}^{N}\alpha_j y_j\, K(x_i,x_j)}{y_i} < 0\text{，则 } y_i = -1 \end{cases}$$

# SVM 小结

- 一种监督学习模型，主要用于**分类**问题
- 目标：找到一个**超平面**来分隔不同类别的数据点
- 确定超平面：**最大化间隔**
- **优化问题**：最小化 $||w||$
- **凸二次规划**：格朗日乘数法
- 有噪声数据：**软间隔**。
- 数据线性不可分时：用**核技巧**，把数据映射到高维空间（线性可分）
- **核函数**：如，高斯核（RBF）...
- 当原问题不易求解时，用其**对偶问题**求解。




$$\min_{w,b,\varepsilon} \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} \varepsilon_i$$

$$\text{s.t.} \quad y_i[\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b] \geq 1 - \varepsilon_i, \quad \varepsilon_i \geq 0.$$

# 代码4：用SVM分类乳腺癌数据

02-SVM.ipynb

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# 支持向量机分类器，SVC：Support Vector Classifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# 1.加载数据集（乳腺癌二分类数据集）
data = datasets.load_breast_cancer()
X = data.data     # 特征
y = data.target   # 标签（0-恶性，1-良性）

# 2.划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X,
                    y, test_size=0.2, random_state=42)

# 3.数据标准化（SVM对特征尺度敏感，必须标准化!否则准确率 0.94）
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test =  scaler.fit_transform(X_test)
```

运行结果：

准确率：**0.98**

```
第一个样本(30个指标):
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]]
```

```
# 4. 初始化SVM模型
svm_model = SVC(kernel='rbf',    # 高斯核函数（RBF核）
                C=1.0,           # 正则化参数（默认值）
                gamma='scale',   # RBF核的带宽参数（默认值）
                random_state=42)

# 5. 训练模型
svm_model.fit(X_train, y_train)

# 6. 预测测试集
y_pred = svm_model.predict(X_test)

# 7. 评估模型性能
print("测试集准确率:", accuracy_score(y_test, y_pred))
```

# 提 纲

① 传统机器学习

**监督学习**

② 线性回归（一元、多元）、Logistic回归

③ 二分类、多分类

④ 决策树

⑤ 支持向量机 SVM

**无监督学习**

**⑥ K均值聚类** （聚类）

⑦ 主成分分析 （降维）

# $K$均值聚类 K-means Clustering

- **无监督学习**：对没有标记信息的训练样本进行学习。

  揭示数据内在性质、规律，主要用于**聚类**、**降维**。

- **$K$均值聚类**：把$N$个点划分到 $K$ (事先指定) 个聚类中。硬聚类

$N$个样本集合：$X = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\}$

样本间距离：$d = \left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|^2$

样本与其**子类中心**的距离总和 = 损失函数：

$$\mathcal{L} = \sum_{l=1}^{K} \sum_{C(i)=l} \|\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(l)}\|^2$$

# $K$均值聚类算法

1) 给定$K$个子类的**中心点**，

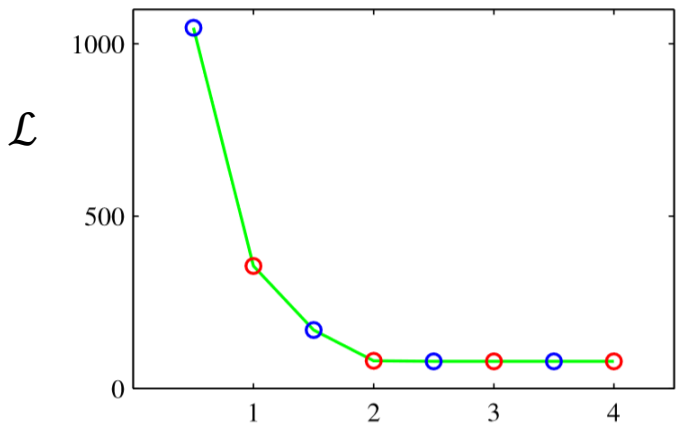
   求出以此为中心的**聚类**结果；

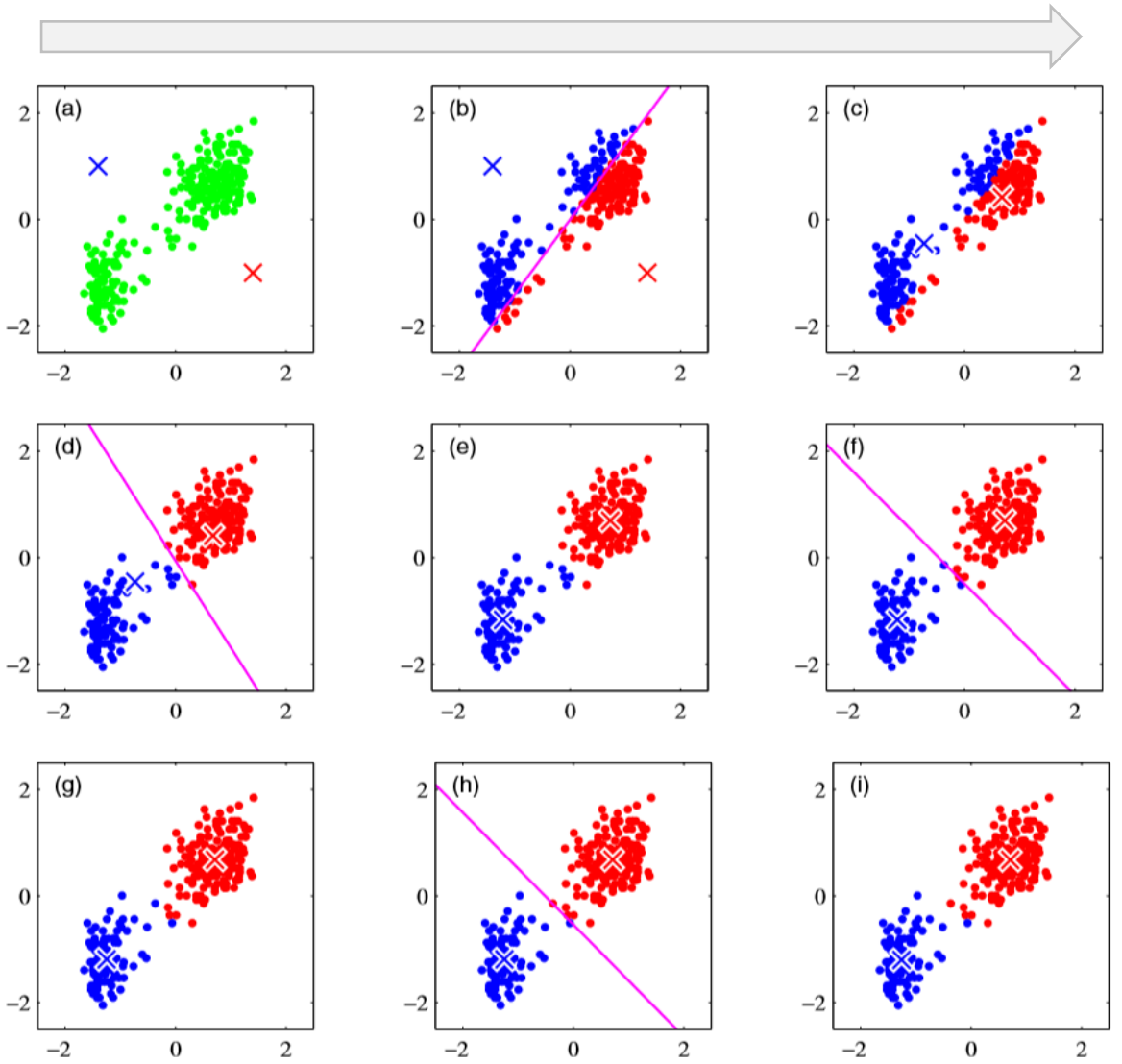
2) 根据上面聚类结果，

   求出每个子类**新的中心点**。

3) 重复上面 1）、2）

   直到收敛。



# 例5： 用K均值聚类算法 聚类到2个类中

给定含有5个样本的集合：$X=\begin{bmatrix}0&0&1&5&5\\2&0&0&0&2\end{bmatrix}$

1) 选择2个子类的中心.

类$C_1^{(0)}$的中心：$\left(\bar{\boldsymbol{x}}^{(1)}\right)^{(0)}=\boldsymbol{x}^{(1)}=(0,2)^T$

类$C_2^{(0)}$的中心：$\left(\bar{\boldsymbol{x}}^{(2)}\right)^{(0)}=\boldsymbol{x}^{(2)}=(0,0)^T$

求出聚类结果：

$d\left(\boldsymbol{x}^{(3)},\left(\bar{\boldsymbol{x}}^{(1)}\right)^{(0)}\right)=5$　　$d\left(\boldsymbol{x}^{(3)},\left(\bar{\boldsymbol{x}}^{(2)}\right)^{(0)}\right)=1$　　$\boldsymbol{x}^{(3)}$分到类$C_2^{(0)}$

$d\left(\boldsymbol{x}^{(4)},\left(\bar{\boldsymbol{x}}^{(1)}\right)^{(0)}\right)=29$　　$d\left(\boldsymbol{x}^{(4)},\left(\bar{\boldsymbol{x}}^{(2)}\right)^{(0)}\right)=25$　　$\boldsymbol{x}^{(4)}$分到类$C_2^{(0)}$

$d\left(\boldsymbol{x}^{(5)},\left(\bar{\boldsymbol{x}}^{(1)}\right)^{(0)}\right)=25$　　$d\left(\boldsymbol{x}^{(5)},\left(\bar{\boldsymbol{x}}^{(2)}\right)^{(0)}\right)=29$　　$\boldsymbol{x}^{(5)}$分到类$C_1^{(0)}$

类$C_1^{(1)}=\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(5)}\}$　　类$C_2^{(1)}=\{\boldsymbol{x}^{(2)},\boldsymbol{x}^{(3)},\boldsymbol{x}^{(4)}\}$

2) 求2个子类的**新**中心.

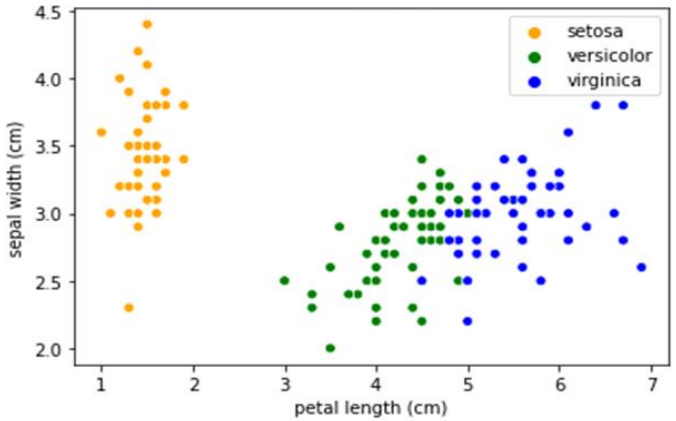
类$C_1^{(1)}$的中心：$\left(\bar{\boldsymbol{x}}^{(1)}\right)^{(1)}=(2.5,2)^T$　　类$C_2^{(1)}$的中心：$\left(\bar{\boldsymbol{x}}^{(2)}\right)^{(1)}=(2,0)^T$

求出聚类结果：　　类$C_1^{(2)}=\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(5)}\}$　　类$C_2^{(2)}=\{\boldsymbol{x}^{(2)},\boldsymbol{x}^{(3)},\boldsymbol{x}^{(4)}\}$　　聚类停止。

# 代码5：K均值聚类

02-K均值聚类.ipynb

```python
# <1> 提取数据集
from sklearn import datasets

iris = datasets.load_iris()

X = iris.data
# 分类 [0-setosa山鸢尾, 1-versicolor杂色鸢尾, 2-virginica维吉尼亚鸢尾]
y = iris.target

# <2> 数据集划分
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size    = 0.3,  # 测试集占30%，隐含 训练集占70%
    random_state = 4,    # 保证每次运行程序时划分结果一致
    shuffle      = True, # 分割前对数据集进行洗牌
    stratify     = y   ) # 训练集和测试集中各类别样本的比例
```




```python
# <3> 特征标准化
from sklearn.preprocessing import StandardScaler

# 标准化数据，使得数据符合标准正态分布
ss = StandardScaler()

# 使用ss对象对训练集和测试集进行标准化处理
X_train_std = ss.fit_transform(X_train)
X_test_std  = ss.fit_transform(X_test)


# <4> 训练模型、预测
from sklearn.cluster import KMeans

K_model = KMeans(n_clusters = 3)     # 分3类

K_model.fit(X_train_std, y_train)    # 在训练集上进行训练

y_pred = K_model.predict(X_test_std) # 训练后预测


# <5> 计算模型准确率
from sklearn.metrics import accuracy_score

# 准确率
acc = accuracy_score(y_pred, y_test)
print(f"准确率: {acc :.2f}")
```

运行结果：

准确率： **0.40**

# 6

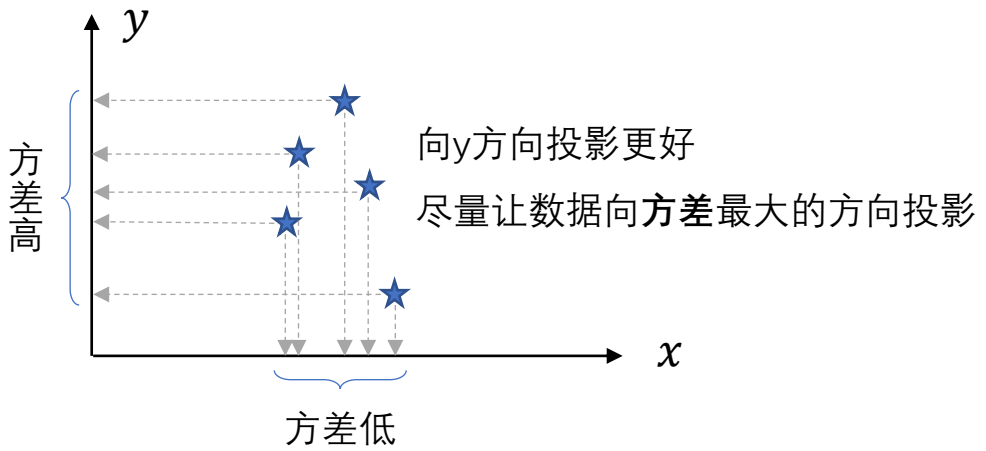
# 主成分分析

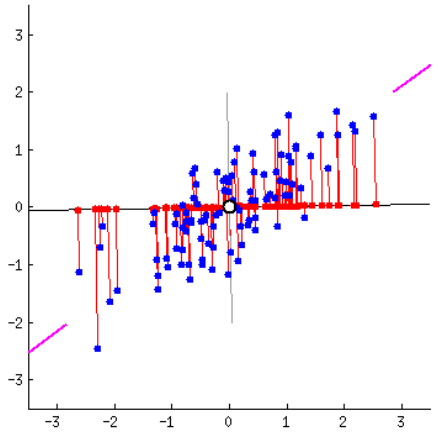
无监督学习

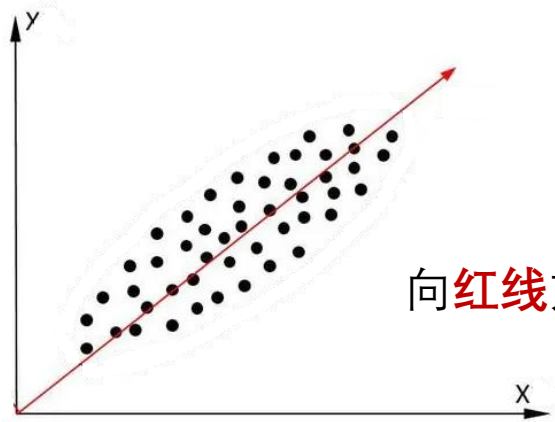
# 主成分分析 Principal Component Analysis

- 一种**特征降维**方法，在消除数据噪声、冗余有广泛的应用。
- 通过分析找到数据特征的主要成分，来代替原始数据。
- 作用：**简化数据、加深对数据的理解**。
- 要求：**降维后要保持原数据的原有结构**。
  - 文本数据，要保持单词之间的相似性
  - 图像数据，要保持视觉对象区域空间分布

保留数据中最重要的信息（方差最大的方向）




向**红线**方向投影更好

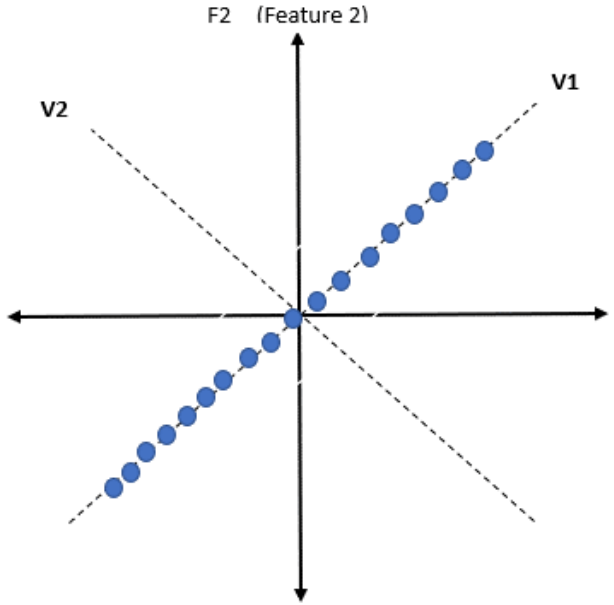
# 方 差 variance

描述数据的**离散程度**：**数据分散、方差大；**数据集中，方差小。

$$var(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

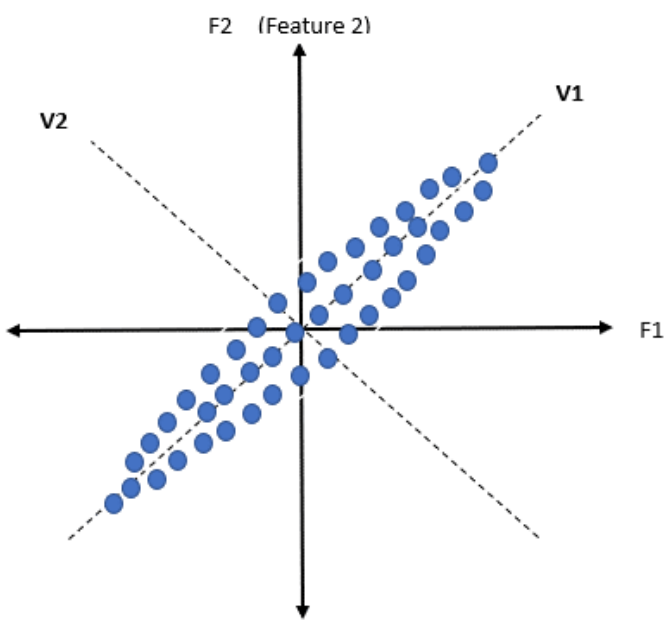
$X = \{x_i\}, (i = 1, 2, \dots n)$ $n$个数据

样本均值：$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$



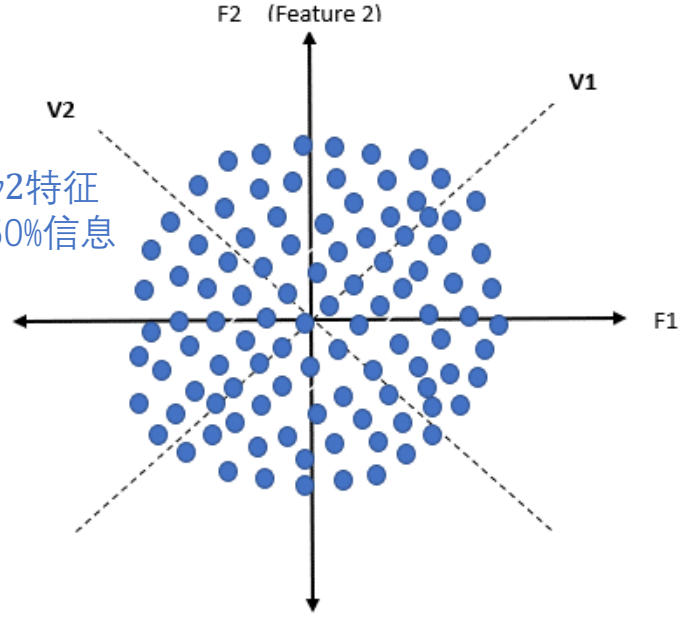

设：特征值 $\lambda_1 = 3$，$\lambda_2 = 0$

则，$v1$方向方差 $= \frac{\lambda_1}{\lambda_1+\lambda_2} = 1 = 100\%$




设：特征值 $\lambda_1 = 3$，$\lambda_2 = 1$

则，$v1$方向方差 $= \frac{\lambda_1}{\lambda_1+\lambda_2} = 0.75$




若删除$v2$特征
则损失50%信息

设：特征值 $\lambda_1 = 3$，$\lambda_2 = 3$

则，$v1$方向方差 $= \frac{\lambda_1}{\lambda_1+\lambda_2} = 0.5$

# 协方差

- **方差** variance：描述离散程度。

$$var(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

$X = \{x_i\}, (i = 1, 2, \dots n)$ $n$个数据

样本均值：$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

- **协方差**：衡量两个变量之间的相关度。

$$cov(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$(X, Y) = \{(x_i, y_i)\}, (i = 1, 2, \dots n)$

$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$　　$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

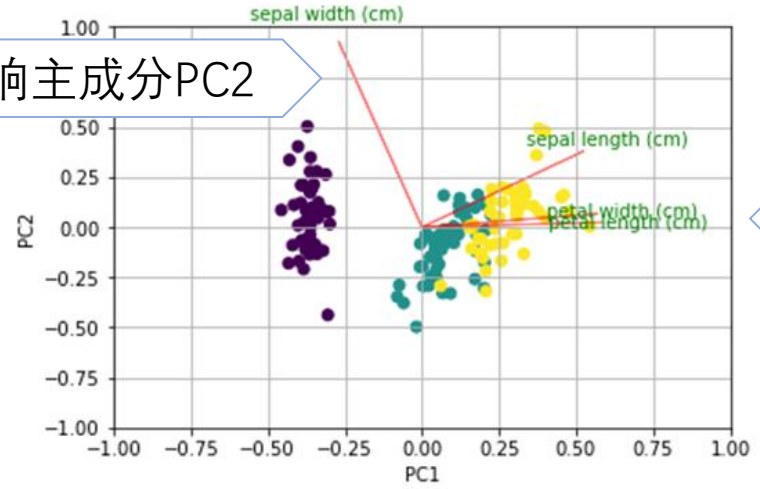
- 当$cov(X, Y) > 0$，$X, Y$ 正相关；
- 当$cov(X, Y) < 0$，$X, Y$ 负相关；
- 当$cov(X, Y) = \mathbf{0}$，$X, Y$ **不相关**。

# 主成分分析 **步骤**

1. **中心化数据**：将数据平移至原点（**均值为0**）。
2. **协方差矩阵**：计算数据各维度间的**相关性**，若两个变量协方差=0，则二者**正交**。
3. **特征值分解**：找到协方差矩阵的特征向量（主成分方向）和对应的特征值（方差大小）。
4. **选择主成分**：按特征值从大到小排序，选取前 $k$ 个**主成分**（贡献最大的方向）。
5. **投 影 数 据**：将原始数据投影到选定的**主成分**上，得到**低维**表示。

- 每个**主成分方向**是原始数据在该方向上投影**方差最大的方向**，保留最多的信息量。
- 所有**主成分之间彼此正交**（协方差为0），消除冗余信息，各维度独立表征数据特征。

主要影响主成分PC2




4维变2维

主要影响主成分PC1

# 代码6：主成分分析PCA

02-主成分分析PCA.ipynb

```python
# <1> 提取数据集
from sklearn import datasets

iris = datasets.load_iris()

# 花萼的长、宽、花瓣的长、宽
X = iris.data
# 分类 [0-setosa山鸢尾, 1-versicolor杂色鸢尾, 2-virginica]
y = iris.target

# <2> 数据集划分
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size    = 0.3,  # 测试集占30%，隐含 训练集占70%
    random_state = 4,    # 保证每次运行程序时划分结果一致
    shuffle      = True, # 分割前对数据集进行洗牌
    stratify     = y   ) # 训练集和测试集中各类别样本的比例
```

```
降维前:
[[-0.88403283  0.6819368  -1.17409495 -0.93763658]
 [-0.77069529 -0.8272676   0.07172927  0.22365643]
 [ 0.13600505 -0.0726654   0.58138464  0.73978666]]
降维后:
[[-1.85972726  0.16025554]
 [ 0.01312903 -1.04442524]
 [ 0.84327606  0.06270081]]
```

```python
# <3> 特征标准化
from sklearn.preprocessing import StandardScaler

# 标准化数据，使得数据符合标准正态分布
ss = StandardScaler()

# 使用ss对象对训练集和测试集进行标准化处理
X_train_std = ss.fit_transform(X_train)
X_test_std  = ss.fit_transform(X_test)


# <4> 训练模型、预测
from sklearn.decomposition import PCA

# 创建PCA模型实例，4维降到2维
P_model = PCA(n_components=2)

P_model.fit(X_train_std)    # 在训练集上进行训练


# <5> 显示降维前后数据
X_test_PCA = P_model.transform(X_test_std)
print("降维前: ", X_test_std[: 3], sep="\n")
print("降维后: ", X_test_PCA[: 3], sep="\n")
```

# 小结

下一节

① 传统机器学习

**监督学习**

② 线性回归、Logistic回归

③ 二分类、多分类

④ 决策树

⑤ 支持向量机 SVM

**无监督学习**

⑥ K均值聚类 （聚类）

⑦ 主成分分析 （降维）

深度学习

前馈神经网络

① 数据准备

② 定义模型

③ 训练模型

④ 评估模型