



# 机器学习精要

(Machine Learning Essentials)

Mao, ChenXiao

2014/11/28

# Agenda

- 第一部分：什么是机器学习
- 第二部分：常用机器学习算法简介
- 第三部分：机器学习的高级主题
- 第四部分：机器学习的开源框架

# 第一部分：什么是机器学习

- 机器学习无处不在
- 机器学习的一个实际案例
- 机器学习方法分类
- 开发智能应用的流程
- 机器学习与相关领域的关系

# 机器学习无处不在



搜索引擎

精准广告



Frequently Bought Together



产品推荐

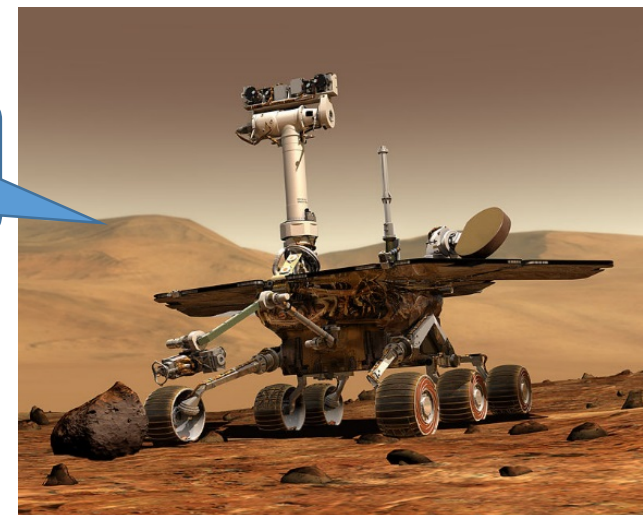
- ☒ **This item:** Machine Learning in Action by Peter Harrington Paperback \$28.64
- ☒ Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython by Wes McKinney Paperback \$25.24

Customers Who Bought This Item Also Bought

Page 1 of 25



机器人视觉



# 房屋估价问题

训练数据

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	\$250,000
2	800	Hipsterton	\$300,000
2	850	Normaltown	\$150,000
1	550	Normaltown	\$78,000
4	2000	Skid Row	\$150,000

列是特征

结果未知的测试数据

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Hipsterton	???

行是样本

房屋特征(几居室、面积、地段)

房屋估价程序

房屋价格\$

## 我们可以这么实现

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):  
    price = 0  
  
    price_per_sqft = 200 # In my area, the average house costs $200 per sqft  
    if neighborhood == "hipsterton": # but some areas cost a bit more  
        price_per_sqft = 400  
    elif neighborhood == "skid row": # and some areas cost less  
        price_per_sqft = 100  
  
    # start with a base price estimate based on how big the place is  
    price = price_per_sqft * sqft  
  
    # now adjust our estimate based on the number of bedrooms  
    if num_of_bedrooms == 0:  
        price = price - 20000 # Studio apartments are cheap  
    else:  
        # places with more bedrooms are usually more valuable  
        price = price + (num_of_bedrooms * 1000)  
  
    return price
```

把房屋销售人员的经验转换成代码

# 如果经验难以表达怎么办？定义带参数的模型

假设房价是各个特征的线性组合

特征的线性函数

房价 =  $w_0 + w_1 \times \text{居室数量} + w_2 \times \text{面积} + w_3 \times \text{地段}$

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):  
    price = 0  
  
    # a little pinch of this  
    price += num_of_bedrooms * .841231951398213  
  
    # and a big pinch of that  
    price += sqft * 1231.1231231  
  
    # maybe a handful of this  
    price += neighborhood * 2.3242341421  
  
    # and finally, just a little extra salt for good measure  
    price += 201.23432095  
  
    return price
```

犹如熬汤，各个成分的比例得当才能好吃



如果可以让计算机根据训练数据找到各个特征的合适权重，程序就可以写成这样

模型

$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3$ ，其中： $x_1$ =几居室； $x_2$ =面积； $x_3$ =地段

# 什么样的参数是好参数？参数选择的策略

权重的选择有无限多，该如何选择？首先定义什么样的权重是好的权重。

Bedrooms	Sq. feet	Neighborhood	Sale price	My Guess
3	2000	Normaltown	\$250,000	\$178,000
2	800	Hipsterton	\$300,000	\$371,000
2	850	Normaltown	\$150,000	\$148,000
1	550	Normaltown	\$78,000	\$101,000
4	2000	Skid Row	\$150,000	\$121,000

如果所有权重都为1.0

$$\text{Cost} = \frac{\sum_{i=1}^{500} (\text{MyGuess}(i) - \text{RealAnswer}(i))^2}{500 \cdot 2}$$

策略：最小二乘

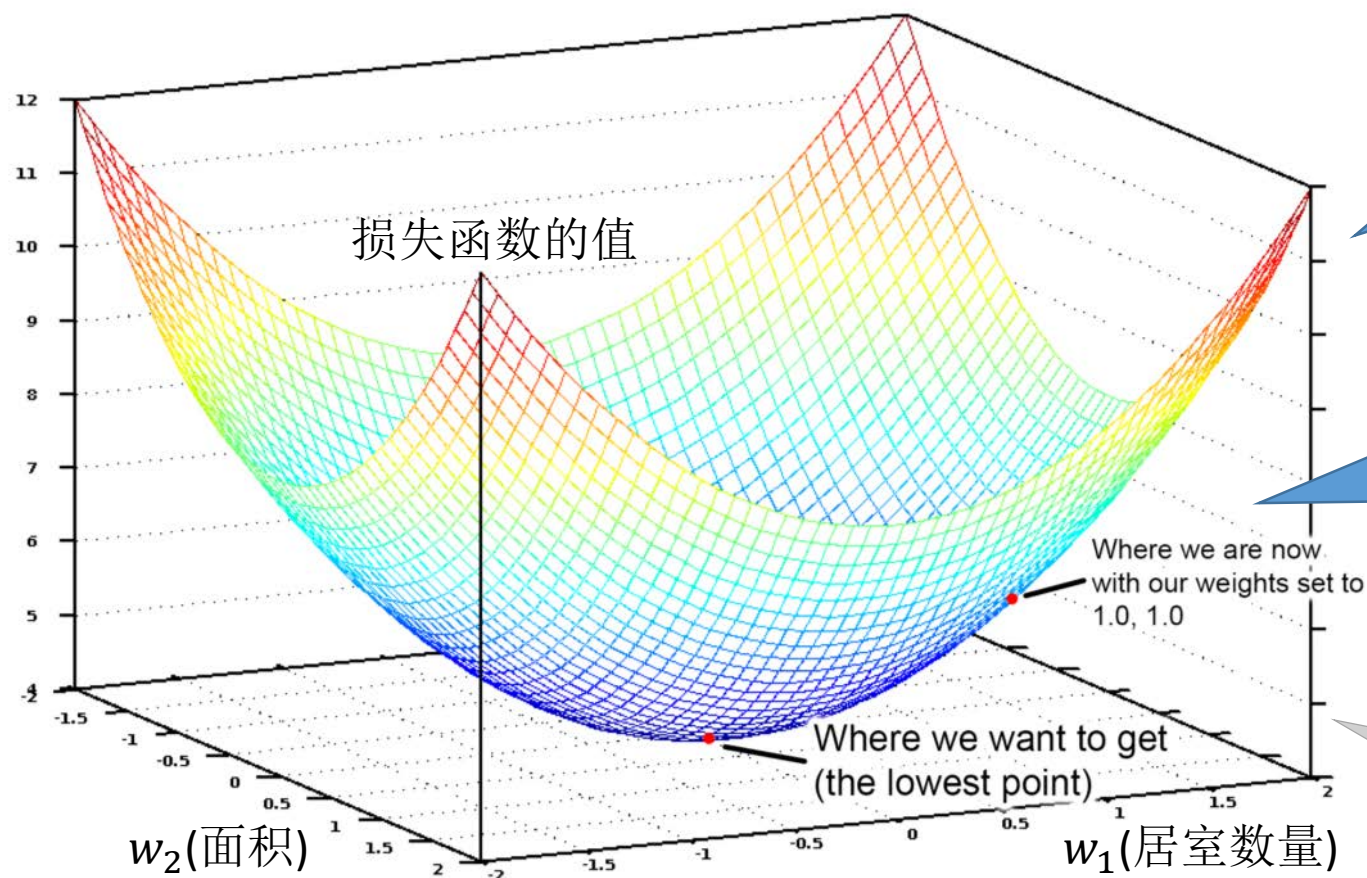
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \quad \text{其中, } \theta = (w_0, w_1, w_2, w_3)$$

损失函数(体现误差的大小)

利用训练数据来评估权重，使得误差尽量小的权重是好的



# 使用什么方法找到合适的参数呢？求解模型参数的算法



所有权重都初始化为1.0

似于爬山，沿着最陡峭的路线走是最快的

求解模型参数的算法：  
梯度下降法(Gradient Descent)

# 房屋估价问题是一个机器学习问题

机器学习，就是从数据出发，提取数据的特征，抽象出数据的模型，发现数据中的知识，又回到对数据的分析和预测中去。

训练数据的目标变量的值是已知的，所以属于监督学习(Supervised Learning)；

进一步，要预测的值是连续的，所以也是回归(Regression)问题

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	\$250,000
2	800	Hipsterton	\$300,000
2	850	Normaltown	\$150,000
1	550	Normaltown	\$78,000
4	2000	Skid Row	\$150,000

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Hipsterton	???

训练数据

模型的学习

梯度下降算法求解  
最小二乘问题

模型

多变量线性模型

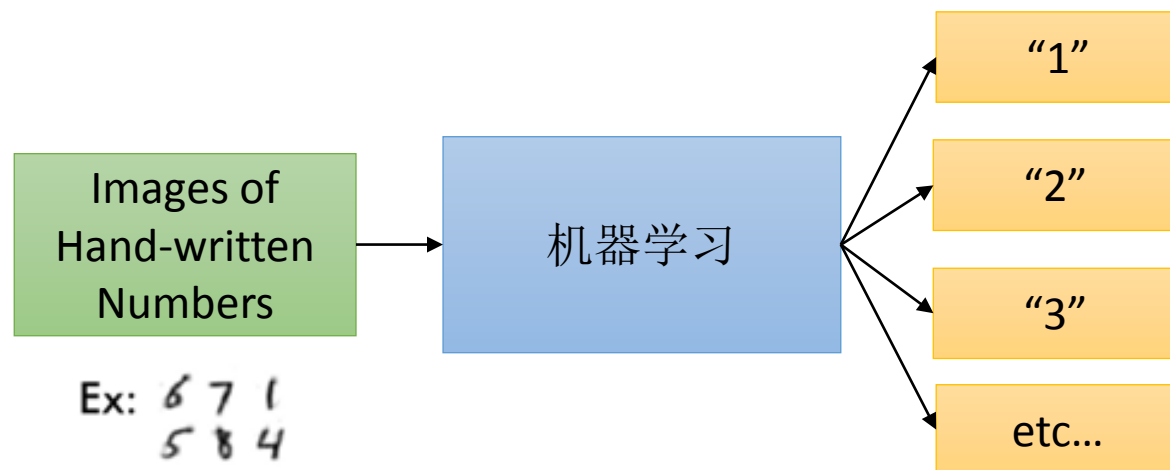
新的数据

模型的应用

将输入特征代入线性  
函数，求得输出即可

## 另一类监督学习问题：分类(Classification)问题

邮件自动化分拣：邮编手写识别



要预测的目标变量是离散值(有限个类别)

垃圾邮件过滤



如果不用机器学习，  
而是直接写程序，  
该怎么写呢？

# 非监督学习(Unsupervised Learning)

## 聚类

Bedrooms	Sq. feet	Neighborhood
3	2000	Normaltown
2	800	Hipsterton
2	850	Normaltown
1	550	Normaltown
4	2000	Skid Row

识别不同的分群，针对不同的客户采用不同的营销策略。比如说你发现：home buyers in the neighborhood near the local college really like small houses with lots of bedrooms, but home buyers in the suburbs prefer 3-bedroom houses with lots of square footage.

## 关联规则学习

发现...



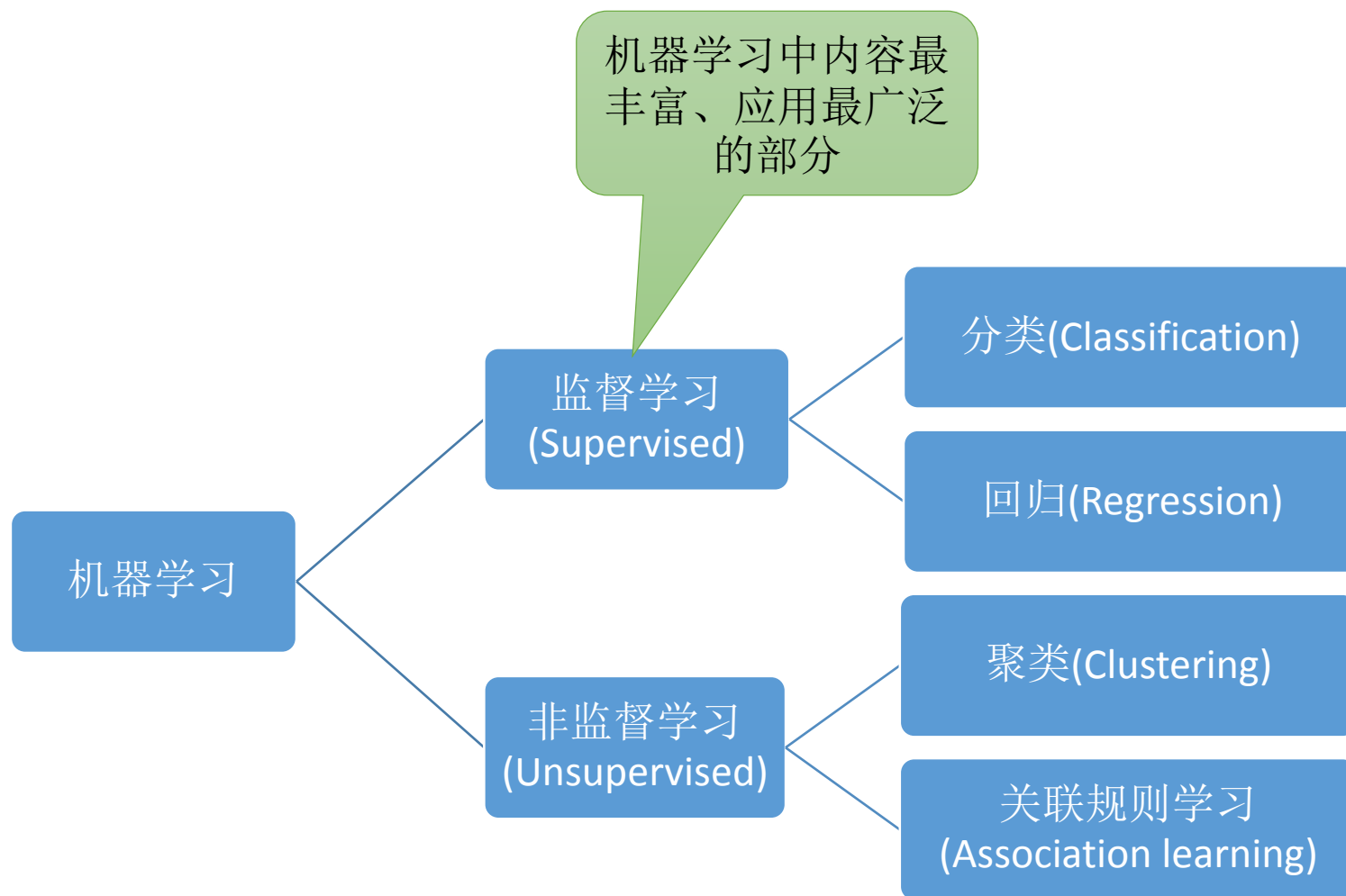
难道是...



事实是...

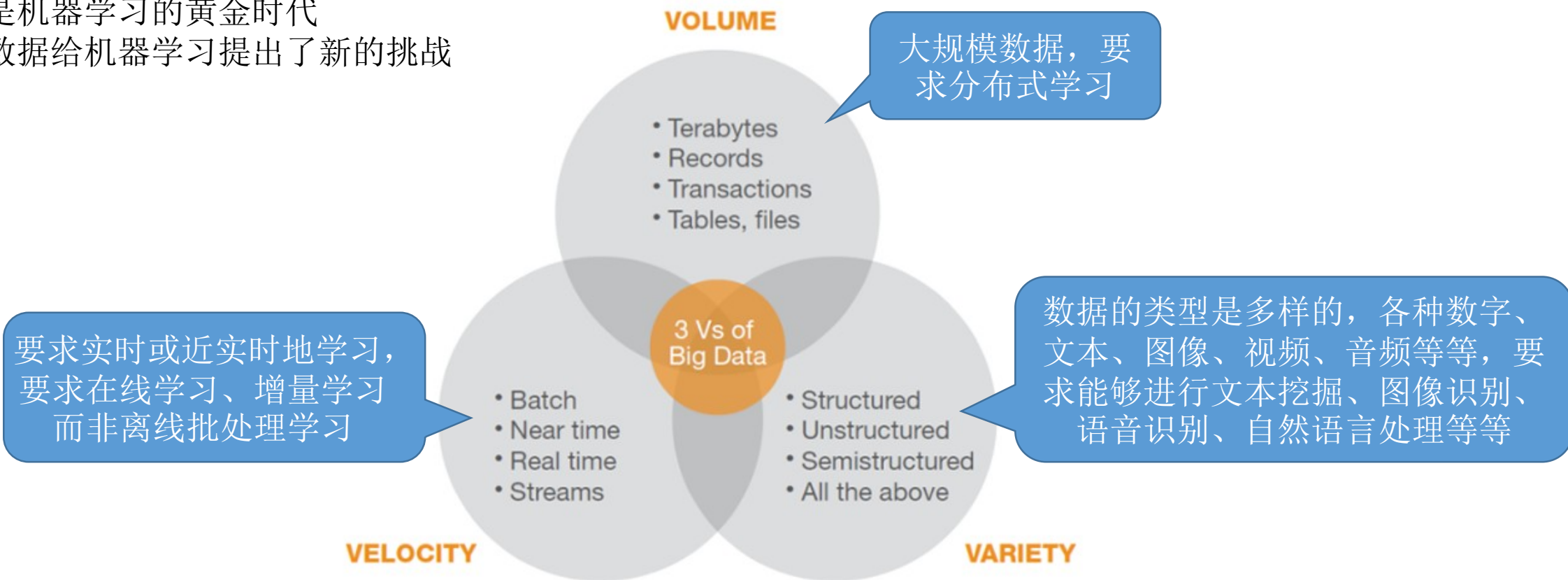
是新爸爸在新妈妈的指使下，前往超市给宝宝买尿布的时候，往往顺便给自己买点啤酒喝喝。于是沃尔玛因势利导，干脆在货物摆放时，将啤酒与尿布摆在了一起。

# 机器学习方法的类别



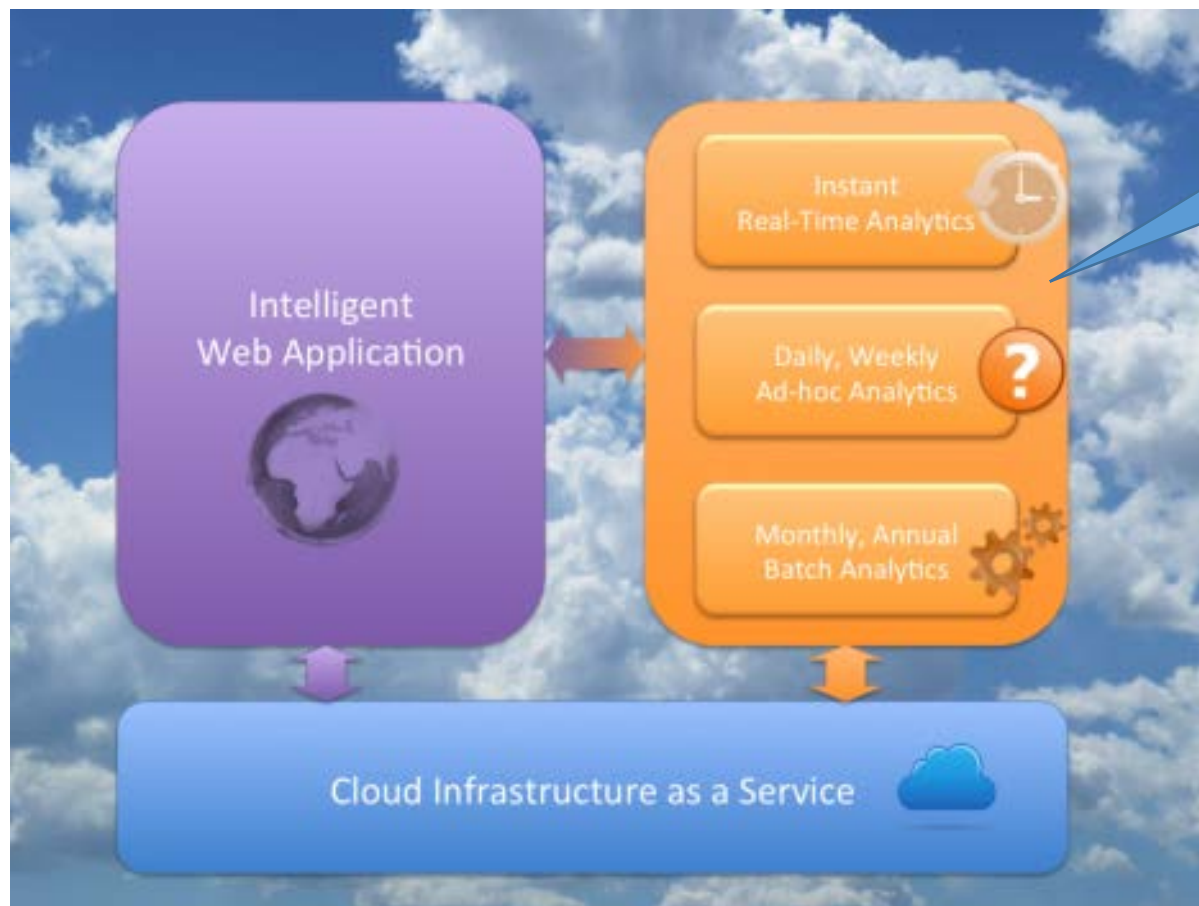
# 大数据(Big Data)和机器学习

- 机器学习是数据驱动的，大数据时代是机器学习的黄金时代
- 大数据给机器学习提出了新的挑战



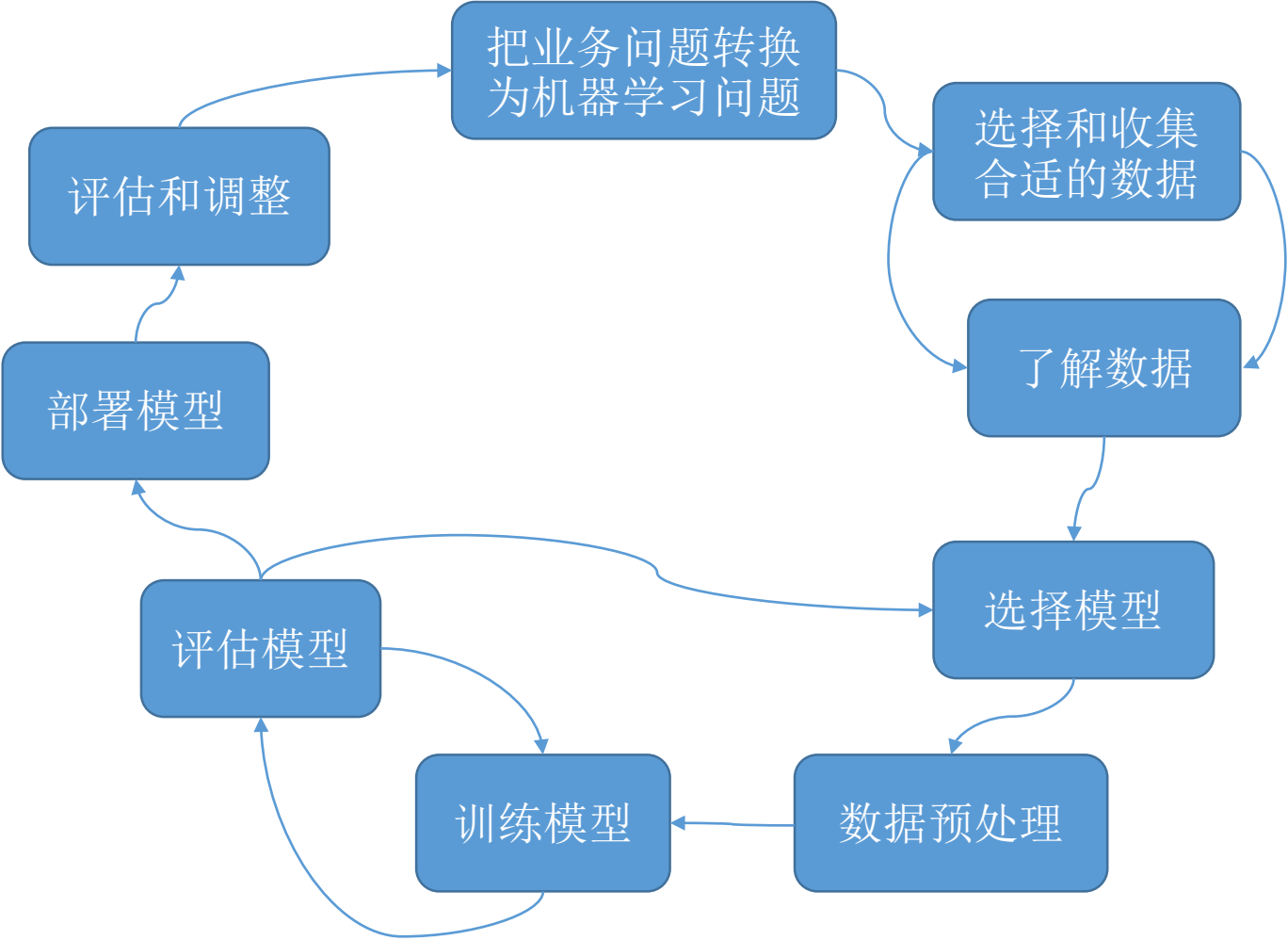


# 智能应用(Intelligent Application)时代来临



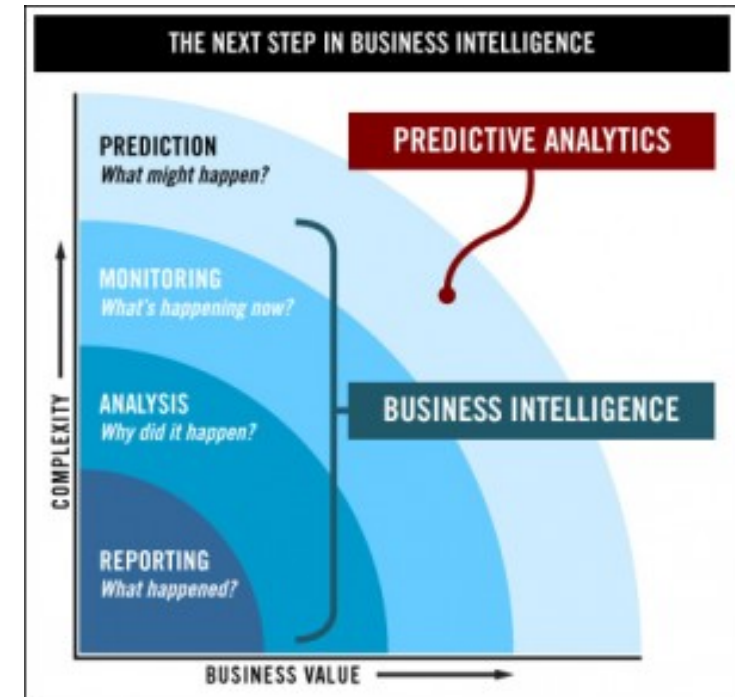
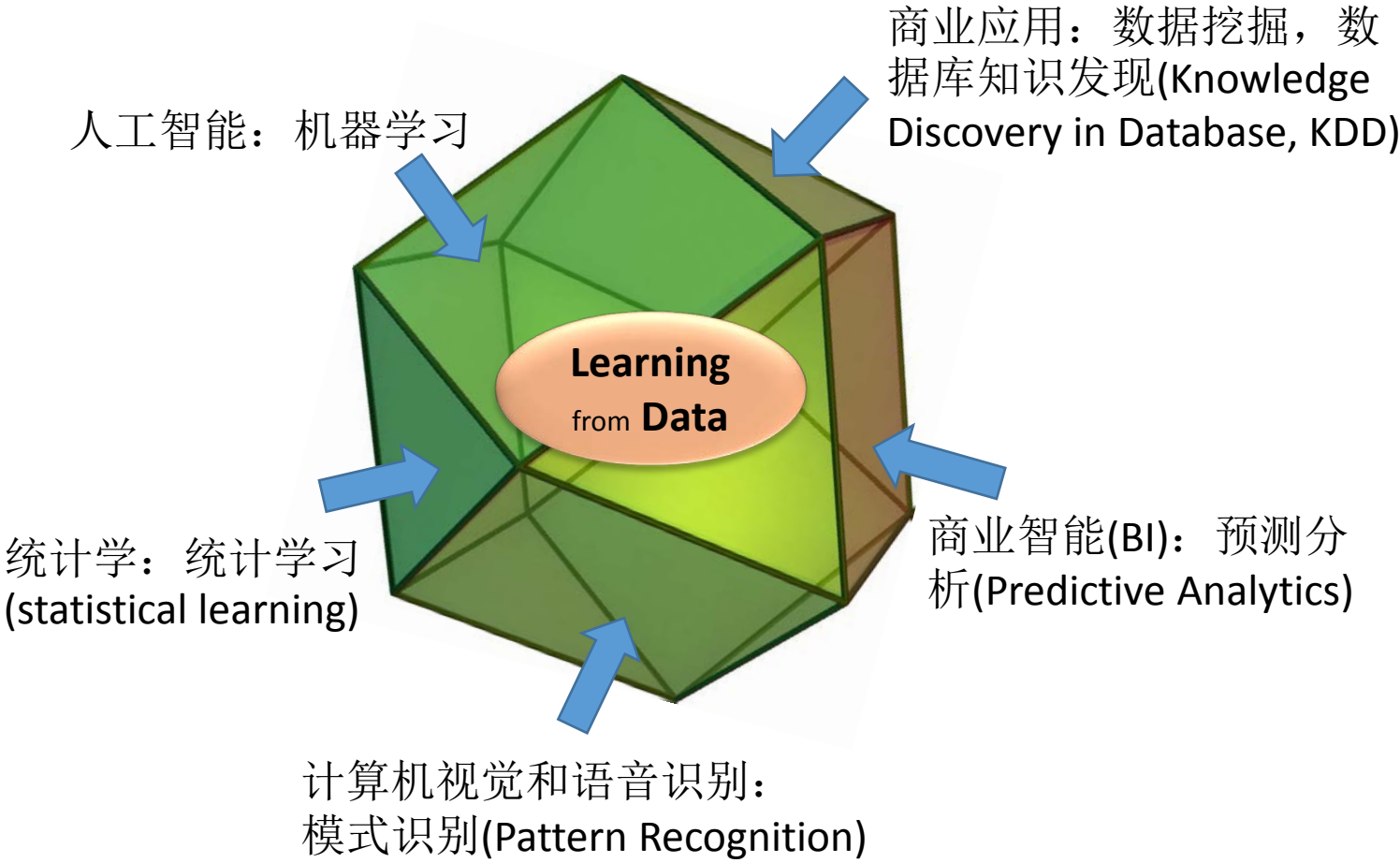
智能应用的核心技术是机器学习

# 利用机器学习构建智能应用的流程





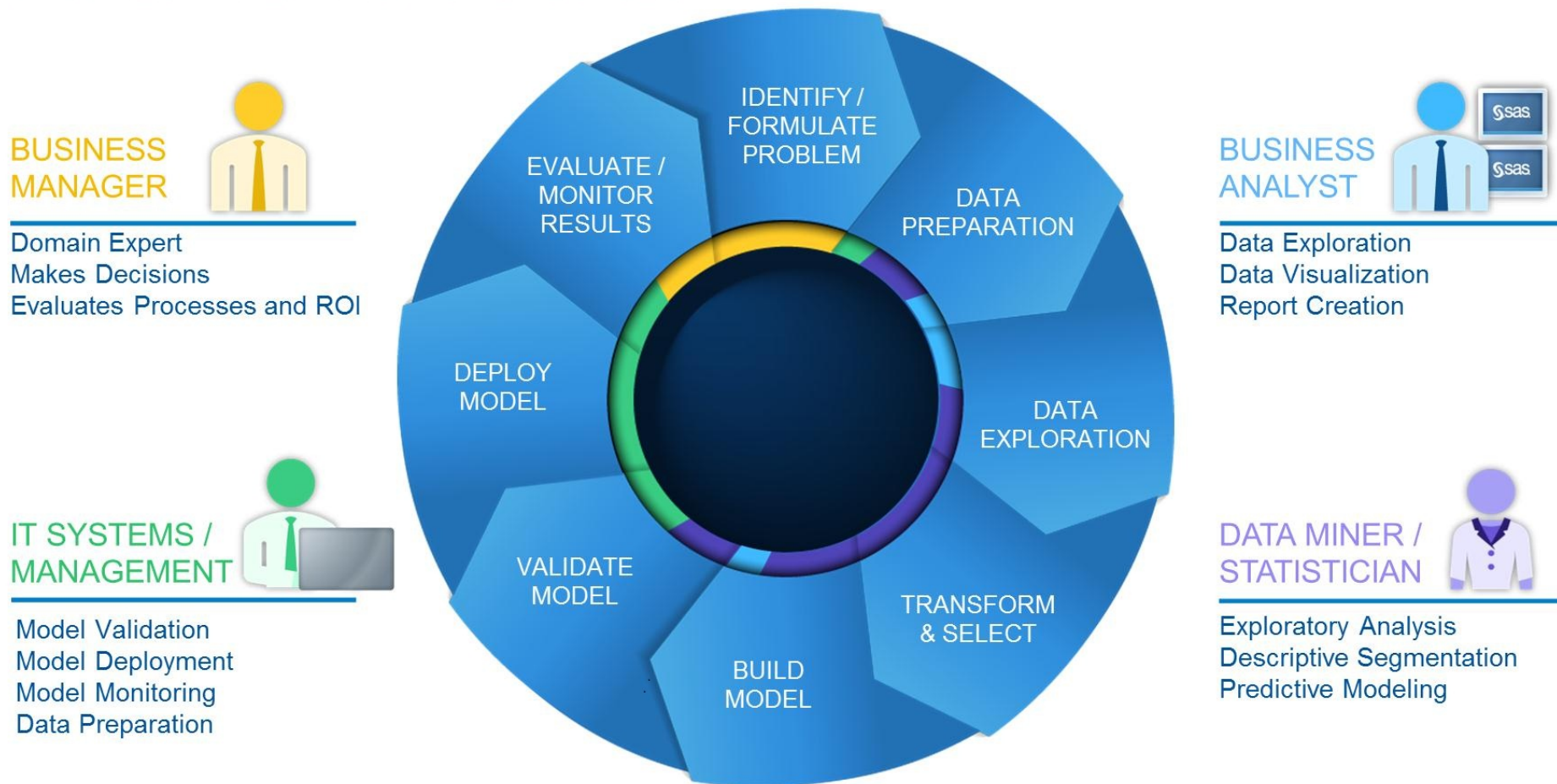
# 跟机器学习密切相关的几个领域



# 预测分析

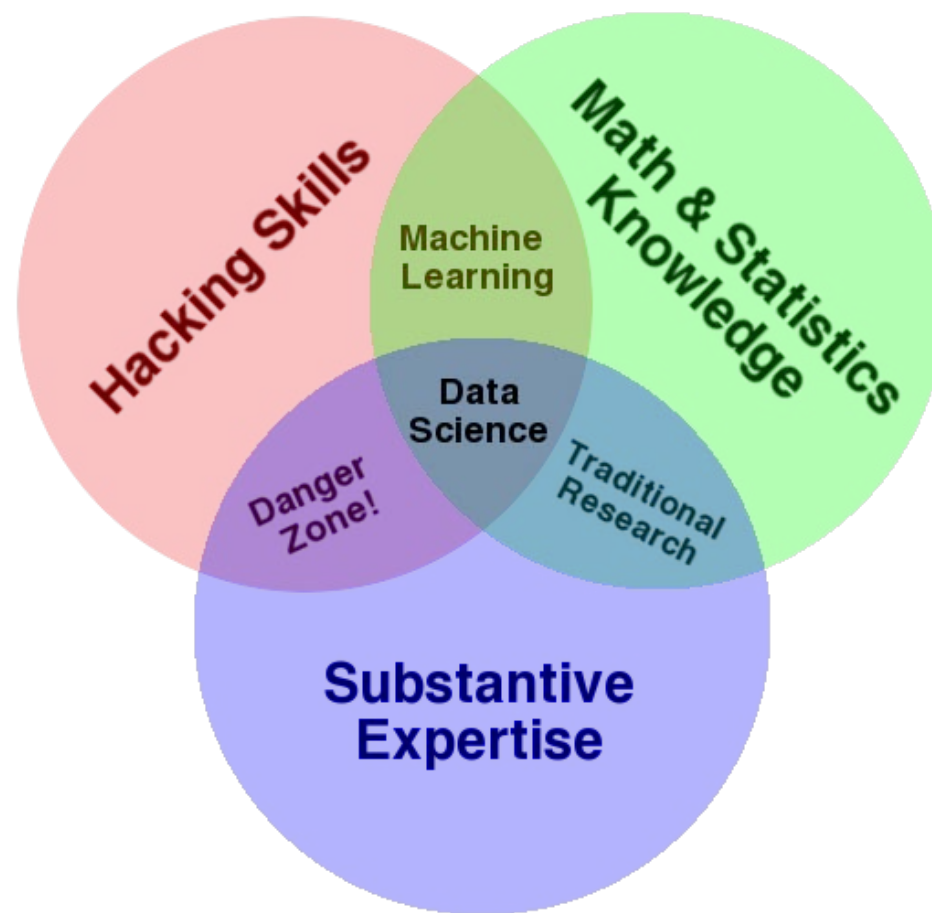
不是指预知未来的能力，而是从已知数据中找出某种规律并用来估计(estimation)未知数据的能力。

## THE PREDICTIVE ANALYTICS LIFECYCLE



# 数据科学(Data Science)

大数据分析催生了一门新的学科，这就是数据科学(data science)。数据科学的独门秘籍是什么呢？从大牛Drew Conway的总结可以看出，数据科学的核心技术之一是机器学习(machine learning)。



# 第二部分：常用机器学习算法简介

- 回归
  - 线性回归(Linear Regression)
- 分类
  - 决策树(Decision Tree)
  - 逻辑回归(Logistic Regression)
  - 支持向量机(SVM)
- 集成学习
  - 随机森林(Random Forest)
- 聚类
  - k均值聚类(k-means Clustering)
- 关联分析(Association Analysis)
  - Apriori
- 数据预处理
  - 主成分分析(PCA)
  - 独立性检验(Test for Independence)
- 特殊数据的分析
  - 时间序列分析(Time Series Analysis)

# 线性回归

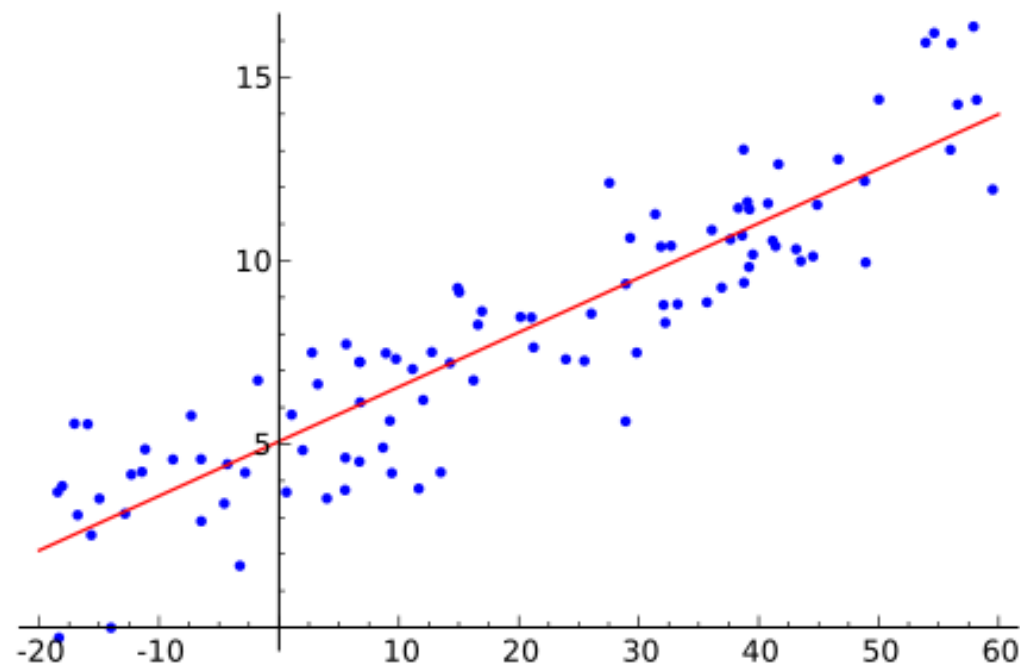
## 基本思想

可以看成是特征空间中，根据训练数据拟合一条直线。

通过训练数据集，学习一个线性模型。线性模型可以表示为：

$$y = w_0 + w_1x_1 + w_2x_2 + \cdots + w_kx_k$$

其中 $(x_1, x_2, \dots, x_k)$ 是输入特征， $y$ 是输出。输出是输入的线性组合。



## 如何实现

如果第 $i$ 个样本的输入特征用 $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(k)})$ 来表示，则在测试集上的总体误差可以表示为：

$$\sum_{i=1}^n (y_i - \sum_{j=0}^k w_j x_i^{(j)})^2$$

线性回归的目标是让该总体误差最小，这也就是通常所说的普通最小二乘(Ordinary Least Square)。

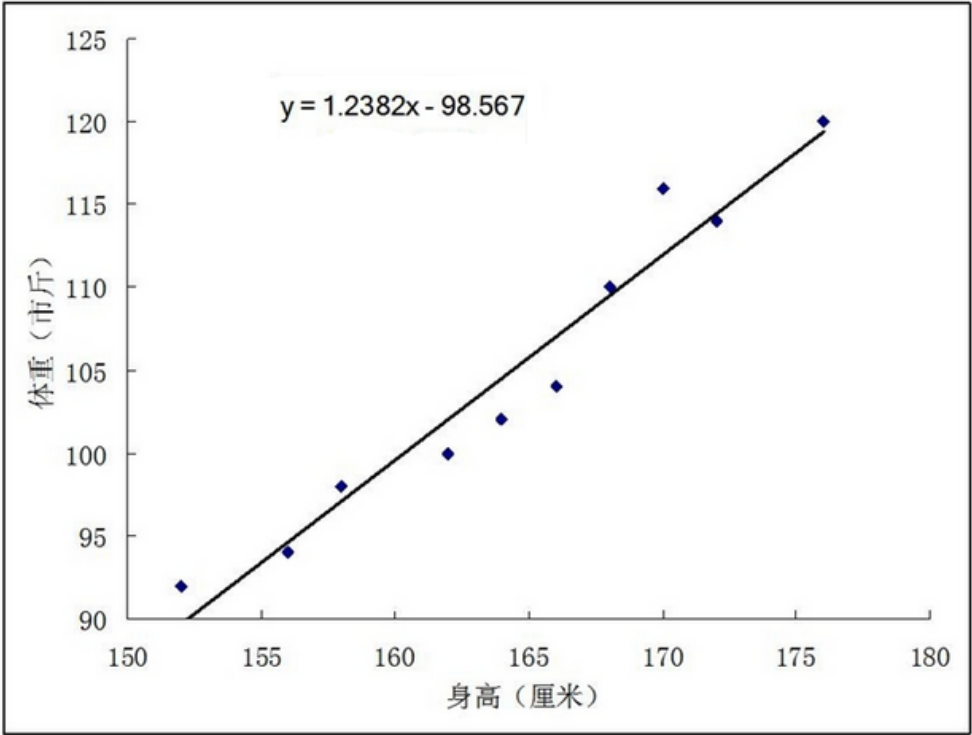
如何求解权重呢？可以对总体误差表达式按各个维度进行偏微分，并让其等于0，即可以求得各个维度的权重。

$$X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ 1 & x_{21} & \dots & x_{2k} \\ 1 & \dots & \ddots & \dots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix}$$

$$\hat{w} = (X^T X)^{-1} X^T y$$

# 二维数据示例

身高 (厘米)	体重 (市斤)
152	92
156	94
158	98
161	?
162	100
164	102
166	104
168	110
170	116
172	114
176	120
180	?



身高 (厘米)	体重 (市斤)	体重 预测值
161	?	$1.2382 \times 161 - 98.567 = 101$
180	?	$1.2382 \times 180 - 98.567 = 124$

# 多维数据示例

数据：CPU的特征和性能的对应关系表(weka/data/cpu.arff)

任务：我们能不能找到特征和性能之间的关系呢？

CPU Performance Data							
	Main Memory (Kb)				Channels		Performance
	Cycle Time (ns)	Min	Max	Cache (KB)	Min	Max	
	MYCT	MMIN	MMAx	CACH	CHMIN	CHMAX	PRP
1	125	256	6000	256	16	128	198
2	29	8000	32,000	32	8	32	269
3	29	8000	32,000	32	8	32	220
4	29	8000	32,000	32	8	32	172
5	29	8000	16,000	32	8	16	132
...							
207	125	2000	8000	0	2	14	52
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

假设线性表达式为：

$$\text{Performance} = w_5 * \text{MYCT} + w_4 * \text{MMIN} + w_3 * \text{MMAx} + w_2 * \text{CACH} + w_1 * \text{CHMAX} + w_0$$

通过OLS，求得模型如下：

$$\text{Performance} = 0.0491 * \text{MYCT} + 0.0152 * \text{MMIN} + 0.0056 * \text{MMAx} + 0.6298 * \text{CACH} + 1.4599 * \text{CHMAX} + (-56.075)$$

模型的应用：把实际的输入特征代入以上的线性表达式，求得相应的performance。



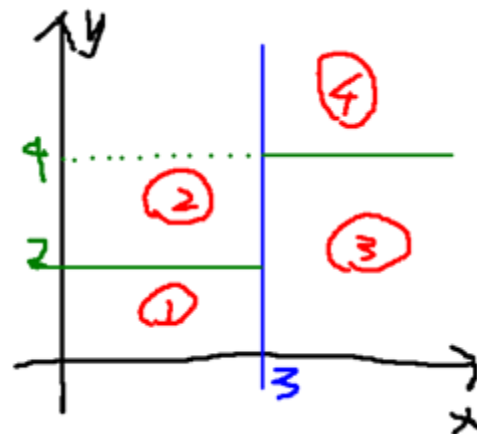
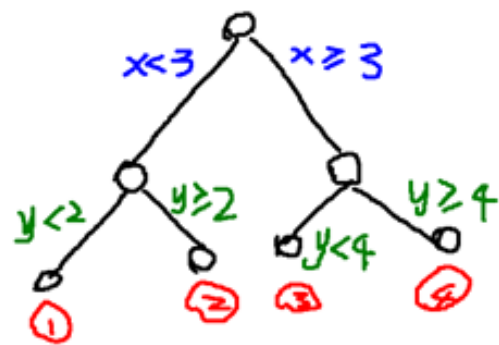
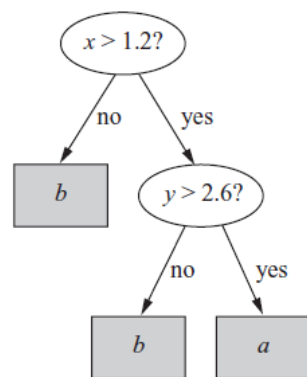
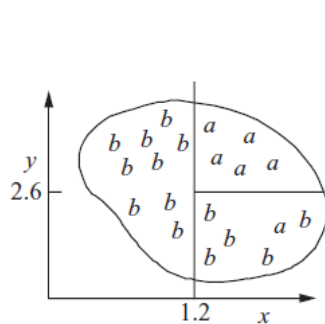
# 决策树(Decision Tree)

## 基本思想

分类决策树是以一种描述对实例进行分类的树形结构。决策树由结点(node)和有向边(directed edge)组成。结点有两种类型：内部结点(internal node)和叶结点(leaf node)。内部结点表示一个特征，叶结点表示一个类。

用决策树分类，从根结点开始，对实例的某一特征进行测试，根据测试结果，将实例分配到了其子结点；这是，每个子结点对应着该特征的一个取值。如此递归地对实例进行测试并分配，直至到达叶结点。最后将实例分到叶结点的类中。

决策树实际上是将空间用超平面进行划分的一种方法。



# 如何实现

决策树学习的目标是损失函数最小化。

构建决策树，通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，使得对各个子数据集有一个最好的分类的过程。

首先，构建根结点，将所有训练数据都放在根结点。选择一个最优特征，为每一个可能的特征值产生一个分支，使样本集分裂成多个子集。然后在每一个分支上递归地重复这个过程。如果一个节点上的所有实例拥有相同的类别，那么就停止该部分树的发展。

对于一个给定的、拥有不同类别的样本集，如果判断应该在哪个特征上进行分裂。结论是，为每一个特征计算信息增益 (information gain)，选择获得最多信息(information)的特征进行分裂。如果信息增益为零，也要停止分裂。

# 分裂特征的选择和信息增益

选择用于划分的特征是关键。划分数据集的大原则是：将无序的数据变得更加有序。信息论中，熵(entropy)是表示随机变量不确定性的度量。

随机变量X的熵定义为：

$$H(X) = -\sum_{i=1}^n p_i \log p_i \quad P(X = x_i) = p_i, \quad i = 1, 2, \dots, n$$

随机变量Y的条件熵定义为：

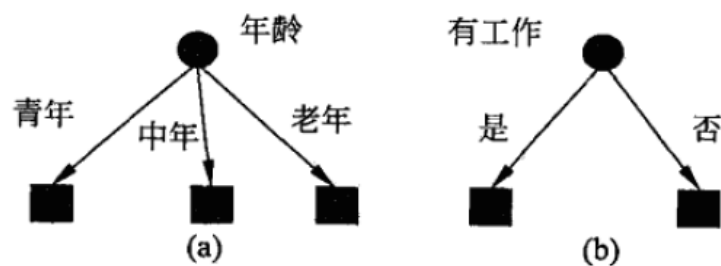
$$H(Y | X) = \sum_{i=1}^n p_i H(Y | X = x_i)$$

从上面的等式可以看出，数据集中实例所对应的类别越多，熵越高，越无序。

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。特征A对训练数据集D的信息增益 $g(D, A)$ ，定义为集合D的熵 $H(D)$ 与特征A给定条件下D的条件熵 $H(D | A)$ 之差。

$$g(D, A) = H(D) - H(D | A)$$

决策树学习应用信息增益准则选择特征。信息增益大的特征，具有更强的分类能力。具体方法是：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。



不同特征决定的不同决策树

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

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

$$g(D, A_2) = 0.324$$

$$g(D, A_3) = 0.420$$

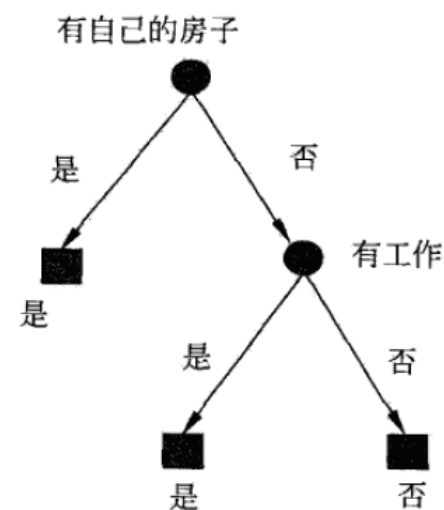
$$g(D, A_4) = 0.363$$

A1, A2, A3, A4 表示年龄、有工作、有自己的房子、和信贷情况4个特征。 $g(D, A_3)$  最大，所以选择A3(有自己的房子)进行划分

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

最终的决策树:



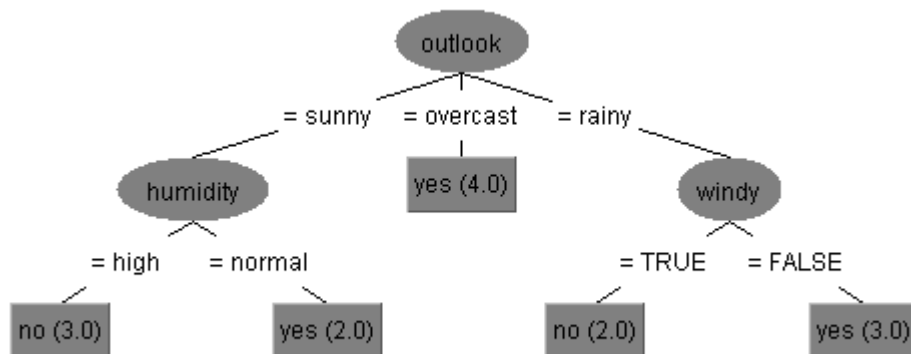
# 示例

数据：天气情况→是否进行活动  
(Weka/data/weather.nominal.arff)。

任务：根据天气特征，决定是不是可以进行某项活动？

Weather Data with Identification Codes					
ID code	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	false	no
b	sunny	hot	high	true	no
c	overcast	hot	high	false	yes
d	rainy	mild	high	false	yes
e	rainy	cool	normal	false	yes
f	rainy	cool	normal	true	no
g	overcast	cool	normal	true	yes
h	sunny	mild	high	false	no
i	sunny	cool	normal	false	yes
j	rainy	mild	normal	false	yes
k	sunny	mild	normal	true	yes
l	overcast	mild	high	true	yes
m	overcast	hot	normal	false	yes
n	rainy	mild	high	true	no

模型：决策树，C4.5是最经典的决策树实现算法，Weka的C48是C4.5的改进。



# 逻辑回归(Logistic Regression)

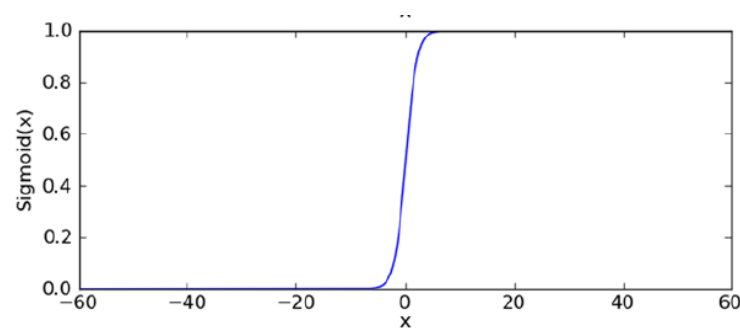
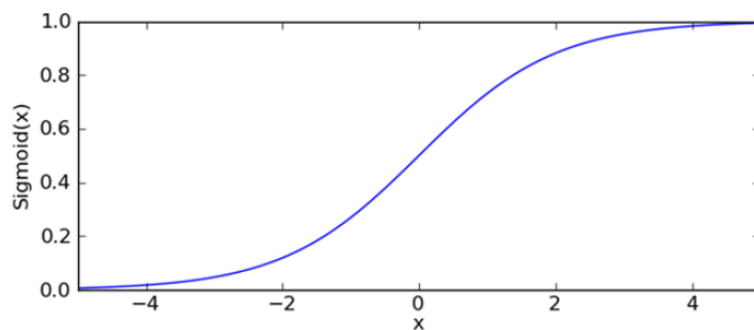
## 基本思想

逻辑回归(logistic regression), 适用于二类分类问题, 能够给出估计二分类的概率。虽然简单, 却非常有效。逻辑回归利用了sigmoid函数的特性。sigmoid类似于阶跃函数, 输入在负无穷到正无穷, 输出在0到1之间, 该范围和概率的取值范围是一致的。输入 $z$ 大于0时, 输出值大于0.5, 输入 $z$ 小于0时, 输出值小于0.5, 该值可以看做是输出为1的概率。这个特性可以用来进行二类分类, 同时也能够给出二类分类的概率。

使用逻辑回归的模型时, 先将输入属性 $(x_1, x_2, \dots, x_k)$ 进行一个线性组合 $z = w_0 + w_1x_1 + w_2x_2 + \dots + w_kx_k$ , 然后代入sigmoid函数中, 得到一个0到1之间的值, 这个值可以直接用作属于二类分类的概率。如果不需要考虑概率, 也可以得到分类值, 通常的做法是, 如果大于0.5则归为类别1, 如果小于0.5则归为类别0。针对具体的应用场景, 可以是任意的两个类别。

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$z = w_0 + w_1x_1 + w_2x_2 + \dots + w_kx_k$$

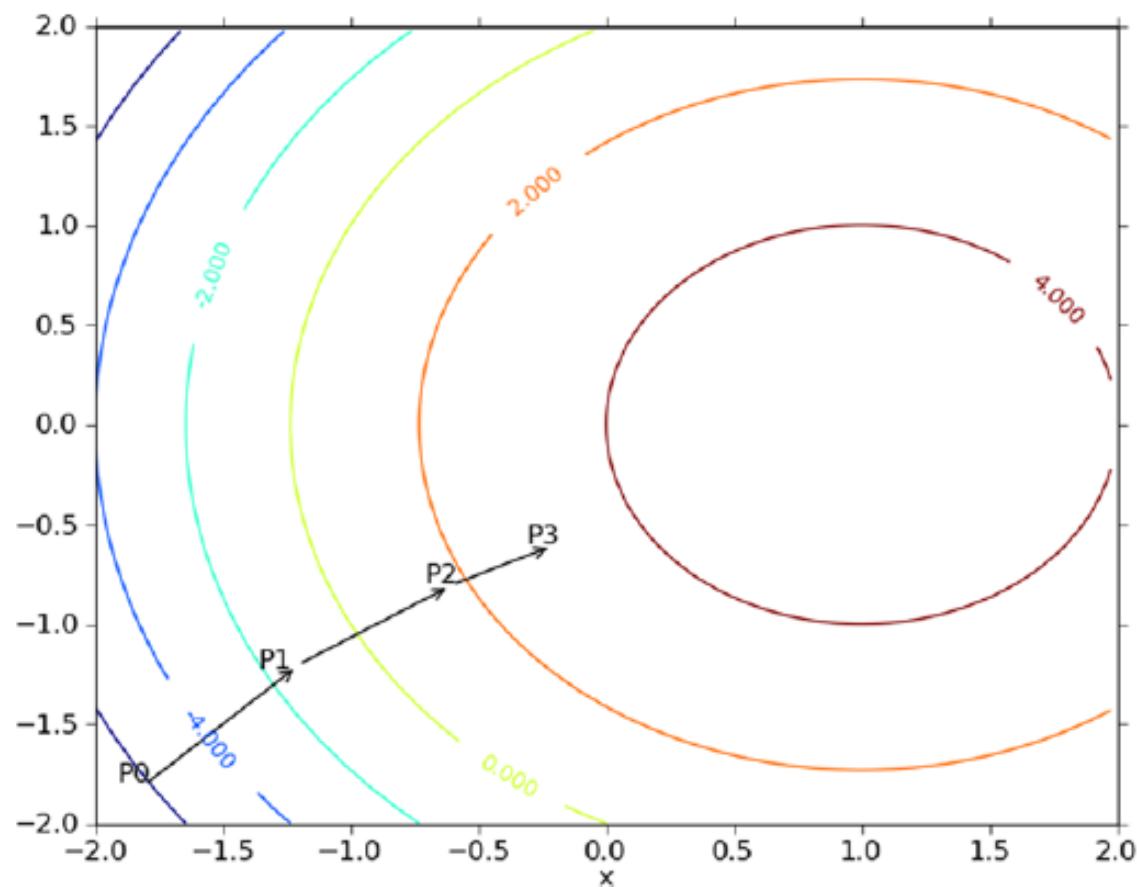


# 如何实现

逻辑回归模型的权重( $w_0, w_1, \dots, w_k$ )是在训练样本上训练得到的。最佳回归系数如何求得呢？目标是求得 $w_i$ 使得实际分类值和预测分类值之间的误差最小，这是一个优化问题。

梯度下降法(Gradient Descent)，是逻辑回归权重求解的常用方法。梯度下降法类似于爬山，沿着最陡峭的路线走是最快的。

为了提高梯度下降法的效率，还提出了随机梯度下降法(Stochastic Gradient Descent)。



# 示例

数据：检测指标→是否糖尿病(Weka/data/diabetes.arff)。

任务：根据检测指标，预测糖尿病的概率。

Relation: pima_diabetes									
No.	1: preg Numeric	2: plas Numeric	3: pres Numeric	4: skin Numeric	5: insu Numeric	6: mass Numeric	7: pedi Numeric	8: age Numeric	9: class Nominal
1	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0	tested_positive
2	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0	tested_negative
3	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0	tested_positive
4	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	tested_negative
5	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	tested_positive
6	5.0	116.0	74.0	0.0	0.0	25.6	0.201	30.0	tested_negative
7	3.0	78.0	50.0	32.0	88.0	31.0	0.248	26.0	tested_positive
8	10.0	115.0	0.0	0.0	0.0	35.3	0.134	29.0	tested_negative
9	2.0	197.0	70.0	45.0	543.0	30.5	0.158	53.0	tested_positive
10	8.0	125.0	96.0	0.0	0.0	0.0	0.232	54.0	tested_positive
11	4.0	110.0	92.0	0.0	0.0	37.6	0.191	30.0	tested_negative
12	10.0	168.0	74.0	0.0	0.0	38.0	0.537	34.0	tested_positive
13	10.0	139.0	80.0	0.0	0.0	27.1	1.441	57.0	tested_negative
14	1.0	189.0	60.0	23.0	846.0	30.1	0.398	59.0	tested_positive
15	5.0	166.0	72.0	19.0	175.0	25.8	0.587	51.0	tested_positive
16	7.0	100.0	0.0	0.0	0.0	30.0	0.484	32.0	tested_positive

训练得到权重如下：

Coefficients...	
Variable	Class tested_negative
=====	
preg	-0.1232
plas	-0.0352
pres	0.0133
skin	-0.0006
insu	0.0012
mass	-0.0897
pedi	-0.9452
age	-0.0149
Intercept	8.4047



# 讨论：为什么叫逻辑回归(Logistic Regression)?

要点：

- 和亚里士多德说的逻辑学(logic)没有关系
- 有时也翻译为“逻辑斯蒂回归”或“逻辑斯谛回归”
- 利用了logistic function: A logistic function or logistic curve is a common "S" shape (sigmoid curve)
- 为什么叫logistic function? Logarithmic(对数的)和logistic曾经是同义词
- 在广义线性模型(Generalized Linear Model)的角度来看，逻辑回归是一种对数(logarithmic)线性模型

The fact that the name was "given by Verhulst" does not explain why it was given. Usually people pick names for a reason, but maybe Verhulst was an exception. Ok, I looked at the French wikipedia, which states "Le nom de courbe logistique leur a été donné par Verhulst sans que l'on sache exactement pourquoi." - "**The name "logistic curve" was given to it by Verhulst, but no one knows exactly why**". The reference rasch.org gives the following commentary:

"**Verhulst writes "We will give the name logistic [logistique] to the curve" (1845 p.8).** Though he does not explain this choice, **there is a connection with the logarithmic basis of the function.** Logarithm was coined by John Napier (1550-1617) from Greek logos (ratio, proportion, reckoning) and arithmos (number). Logistic comes from the Greek logistikos (computational). **In the 1700's, logarithmic and logistic were syn** Since computation is needed to predict the supplies an army requires, logistics has come to be also used for the movement and supply of troops. So it appears the other meaning of "logistics" comes from the same logic as Verhulst terminology, but is independent (?). Verhulst paper is accessible; the definition is on page 8 (page 21 in the volume), and the picture is after the article (page 54 in the volume). "

参考资料：

[http://en.wikipedia.org/wiki/Logistic\\_function](http://en.wikipedia.org/wiki/Logistic_function)

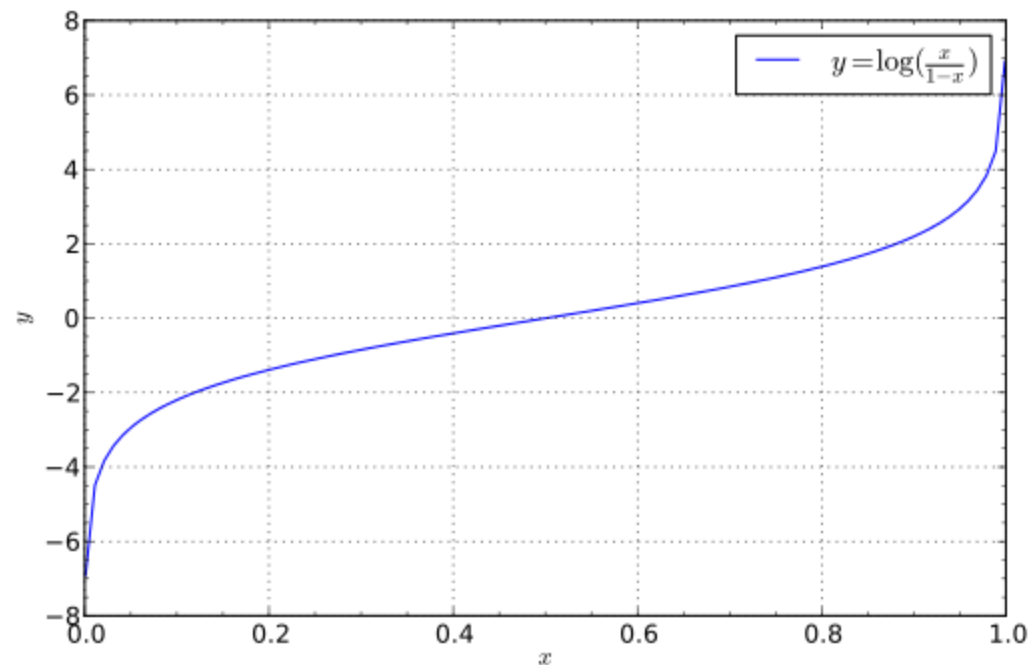
<http://math.stackexchange.com/questions/357918/why-is-logistic-equation-called-logistic>

# 广义线性模型(Generalized Linear Model)简介

广义线性模型是普通线性模型(ordinary linear model)的扩展。使得输入特征的线性组合和输出之间，可以通过一个函数(称为连接函数，link function)进行关联。

广义线性模型	连接函数名称	连接函数	分布
普通线性模型	恒等	$X\beta = \mu$	正态分布
逻辑回归模型	logit (即log-odds, 对数发生比)	$X\beta = \log\left(\frac{\mu}{1-\mu}\right)$	二项分布

所以是一种对数线性模型



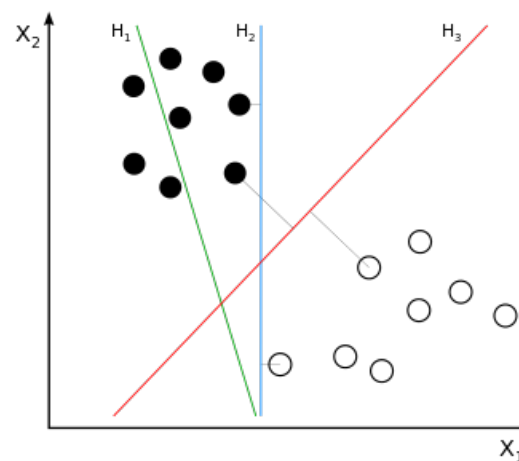
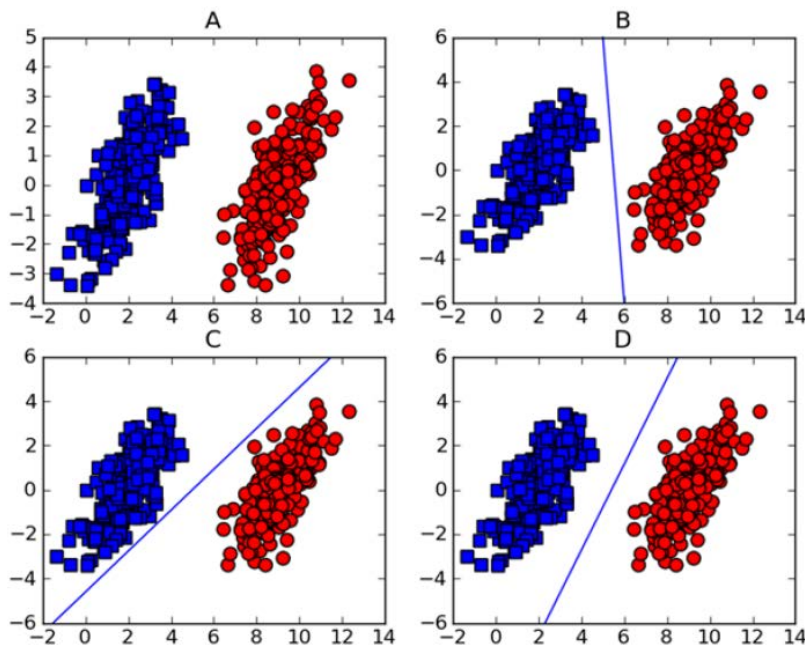
logit函数，其反函数为logistic函数

# 支持向量机(Support Vector Machine)

## 基本思想

对二维平面来说，寻找一条直线，把二维平面上的两个类别分开，支持向量就是离分隔直线最近的那些点，支持向量机的目标就是使得支持向量离分隔直线尽量远。

如果推广到多维空间，支持向量就是离分隔超平面(separating hyperplane)最近的那些点，支持向量机要试着最大化支持向量到分隔面的间隔(maximum margin)。



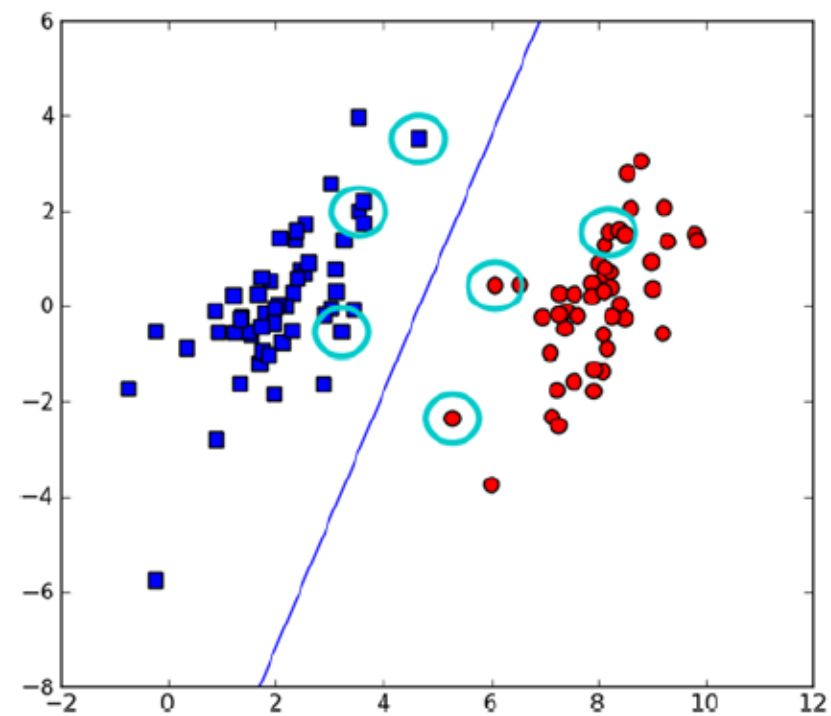
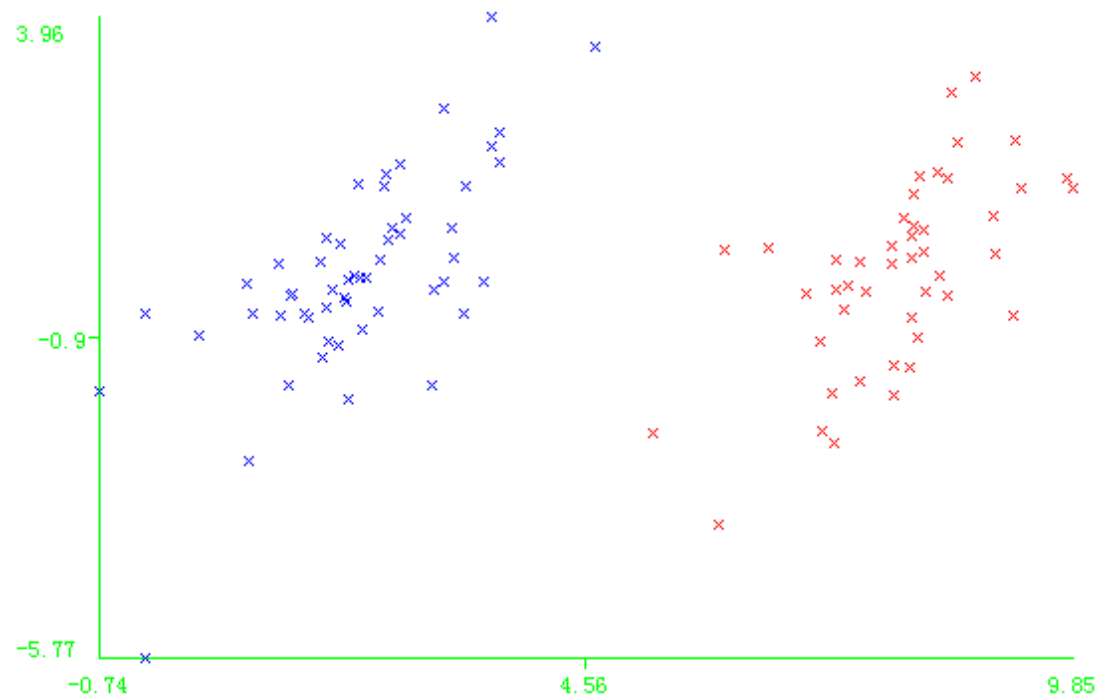
$H_1$ 没正确分类  
 $H_2$ 正确分类了，但是间隔很小  
 $H_3$ 不但正确分类了，间隔也最大

# 如何实现

支持向量机的学习问题，可以形式化为求解凸二次规划问题。解决凸二次规划问题的常用方法是，将它作为原始最优化问题，应用拉格朗日对偶性(Lagrange Duality)，将原始问题转化为对偶问题，通过求解对偶问题(dual problem)，得到原始问题(primal problem)的最优解。由于涉及复杂的数学公式，这里不详细讨论。

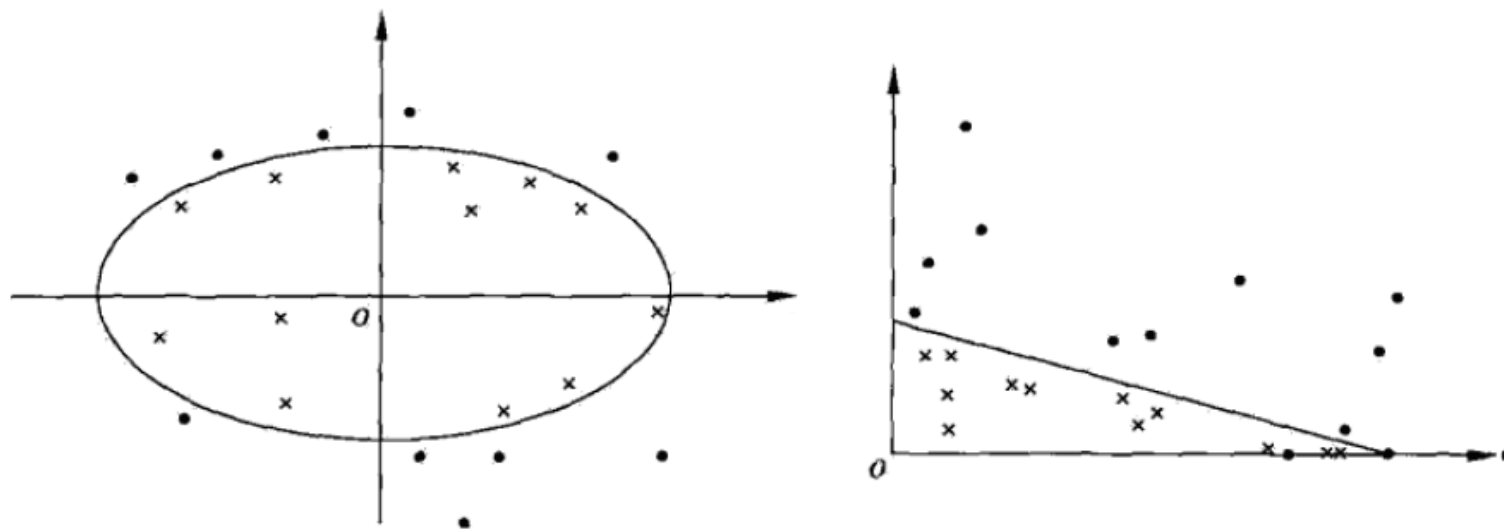
一种高效的实现是序列最小优化算法(Sequential Minimal Optimization, SMO)。

# 线性分类问题示例



# 非线性分类问题与核技巧(Kernel Trick)

SVM本质上是线性分类器，对于不可线性分类的数据，需要利用核函数(Kernel Function)把非线性分类问题转换成线性分类问题，然后再使用支持向量机进行分类。



$$(x, y) \Rightarrow (x^2, y^2)$$

$$w_1x^2 + w_2y^2 + b = 0 \Rightarrow w_1x + w_2x + b = 0$$

实际上是一个多项式核:  $x \rightarrow x^2$

# 常见核函数

常用的核函数有：线性核函数，多项式核函数，径向基核函数，Sigmoid核函数和复合核函数。

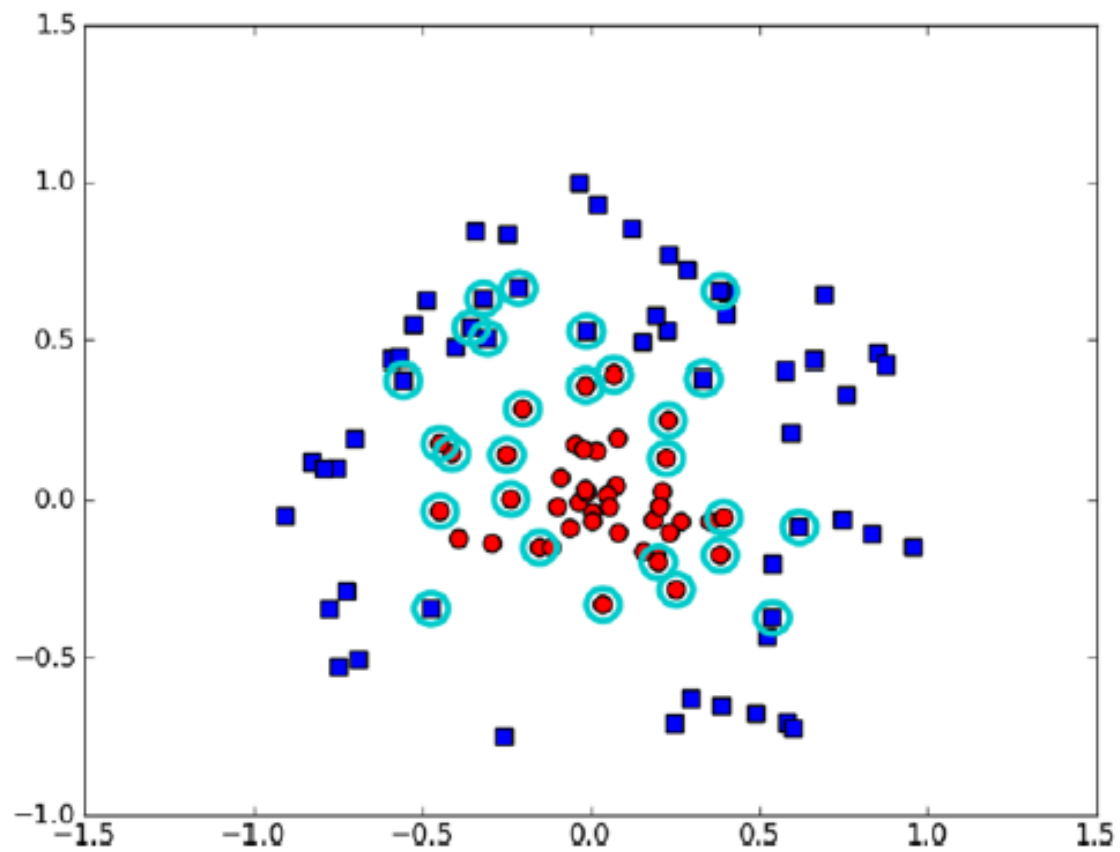
径向基函数(Radial Basis Function Kernel):

$$k(x) = e^{-x^2/(2\sigma^2)}$$

# 核函数(RBF)示例

参数 $\sigma$ 的选择

参数 $\sigma$	错误率(%)	支持向量数目
0.01	51	90
0.1	51	90
1	15	82
5	11	32
10	7	28
25	8	30
50	6	41
100	7	58
150	8	76
200	7	82
250	8	83
500	13	89





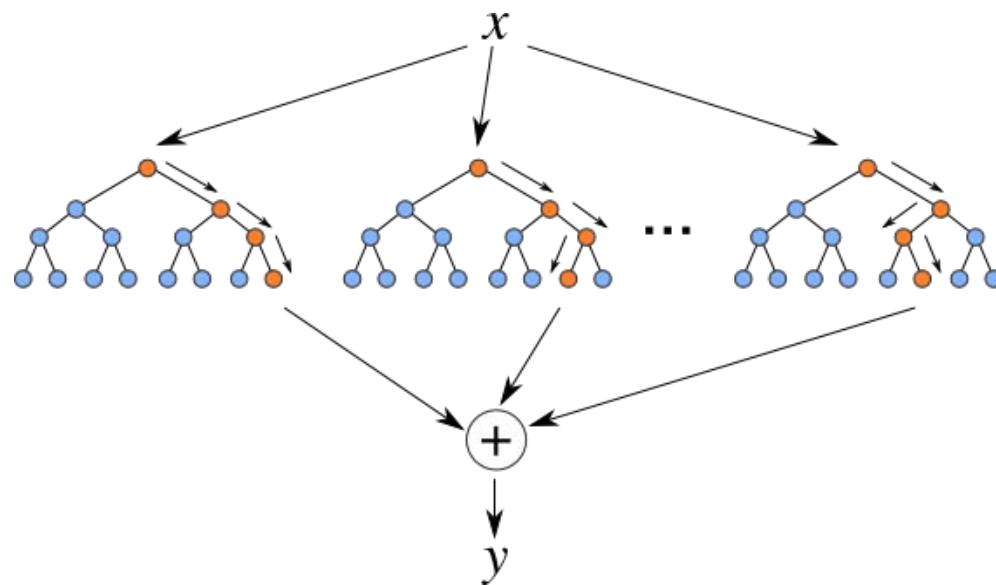
# 随机森林(Random Decision Forest)

## 基本思想

随机森林是一个包含多个决策树的分类器，并且其输出的类别是由个别树输出的类别的众数而定。具体而言，森林里面有很多的决策树组成，每一棵决策树之间是没有关联的。在得到森林之后，当来了一个新的输入样本时，就让森林中的每一棵决策树分别进行判断，看看这个样本应该属于哪一类，然后看看哪一类被选择最多，就预测这个样本为那一类。因为在构建森林的时候，有几个地方使用了随机化的方式，所以叫做随机森林。

随机森林的优点是：

- 效果往往好于单独的决策树
- 训练速度快，在很多数据集上表现良好
- 能够处理很高维度，并且不用做特征选择
- 实现比较简单，容易并行化，非常适合用MapReduce实现



# 如何实现

在建立每一棵决策树的过程中，有两点需要注意：采样与完全分裂。

首先是两个随机采样的过程，random forest对输入的数据要进行行、列的采样。

- 对于行采样，采用有放回的方式，也就是在采样得到的样本集合中，可能有重复的样本。假设输入样本为 $N$ 个，那么采样的样本也为 $N$ 个(即bootstrap取样)。这样使得在训练的时候，每一棵树的输入样本都不是全部的样本，使得相对不容易出现over-fitting。
- 然后进行列采样，从 $M$ 个feature中，选择 $m$ 个( $m \ll M$ )。决策森林中每棵树的每个节点在选择该点的分裂特征时并不是从所有的输入特征里选择一个最好的，而是从所有的 $M$ 个输入特征里随机的选择 $m$ 个特征，然后从这 $m$ 个特征里选择一个最好的（这样比较适合那种输入特征数量特别多的应用场景，在输入特征数量不多的情况下，我们可以取 $m=M$ ）。

之后就是对采样之后的数据使用完全分裂的方式建立出决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有样本的都是指向的同一个分类。

一般很多的决策树算法都一个重要的步骤 - 剪枝，由于之前的两个随机采样的过程保证了随机性，所以就算不剪枝，也不会出现over-fitting。

# 示例

数据：玻璃的成分与类别(Weka/data/glass.arff)  
任务：玻璃鉴别。

Relation: Glass										
No.	1: RI Numeric	2: Na Numeric	3: Mg Numeric	4: Al Numeric	5: Si Numeric	6: K Numeric	7: Ca Numeric	8: Ba Numeric	9: Fe Numeric	10: Type Nominal
1	1.51793	12.79	3.5	1.12	73.03	0.64	8.77	0.0	0.0	build wind float
2	1.51643	12.16	3.52	1.35	72.89	0.57	8.53	0.0	0.0	vehic wind float
3	1.51793	13.21	3.48	1.41	72.64	0.59	8.43	0.0	0.0	build wind float
4	1.51299	14.4	1.74	1.54	74.55	0.0	7.59	0.0	0.0	tableware
5	1.53393	12.3	0.0	1.0	70.16	0.12	16.19	0.0	0.24	build wind non-float
6	1.51655	12.75	2.85	1.44	73.27	0.57	8.79	0.11	0.22	build wind non-float
7	1.51779	13.64	3.65	0.65	73.0	0.06	8.93	0.0	0.0	vehic wind float
8	1.51837	13.14	2.84	1.28	72.85	0.55	9.07	0.0	0.0	build wind float
9	1.51545	14.14	0.0	2.68	73.39	0.08	9.07	0.61	0.05	headlamps
10	1.51789	13.19	3.9	1.3	72.33	0.55	8.44	0.0	0.28	build wind non-float
11	1.51625	13.36	3.58	1.49	72.72	0.45	8.21	0.0	0.0	build wind non-float
12	1.51743	12.2	3.25	1.16	73.55	0.62	8.9	0.0	0.24	build wind non-float
13	1.52223	13.21	3.77	0.79	71.99	0.13	10.02	0.0	0.0	build wind float
14	1.52121	14.03	3.76	0.58	71.79	0.11	9.65	0.0	0.0	vehic wind float
15	1.51665	13.14	3.45	1.76	72.48	0.6	8.38	0.0	0.17	vehic wind float
16	1.51707	13.48	3.48	1.71	72.52	0.62	7.99	0.0	0.0	build wind non-float
17	1.51719	14.75	0.0	2.0	73.02	0.0	8.53	1.59	0.08	headlamps
18	1.51629	12.71	3.33	1.49	73.28	0.67	8.24	0.0	0.0	build wind non-float
19	1.51994	13.27	0.0	1.76	73.03	0.47	11.32	0.0	0.0	containers
20	1.51811	12.96	2.96	1.43	72.92	0.6	8.79	0.14	0.0	build wind non-float
21	1.52152	13.05	3.65	0.87	72.22	0.19	9.85	0.0	0.17	build wind float
22	1.52475	11.45	0.0	1.88	72.19	0.81	13.24	0.0	0.34	build wind non-float
23	1.51841	12.93	3.74	1.11	72.28	0.64	8.96	0.0	0.22	build wind non-float
24	1.51754	13.39	3.66	1.19	72.79	0.57	8.27	0.0	0.11	build wind float

算法, #trees	正确率(%)	耗时(s)
决策树(J48)	66.8	0.01
随机森林,10	74.3	0.01
随机森林,20	76.6	0.02
随机森林,50	79.4	0.06
随机森林,100	79.9	0.09
随机森林,200	80.8	0.12
随机森林,300	80.4	0.18
随机森林,400	80.4	0.24
随机森林,500	80.4	0.3
随机森林,1000	80.8	0.64

## 集成方法(Ensemble Methods)

为什么随机森林的表现会好于单个的决策树呢？实际上按这种算法得到的随机森林中的每一棵都是比较弱的，但是大家组合起来表现比较强。每一棵决策树是一个精通于某一个窄领域的专家（后面会提到我们从所有 $M$ 个特征中选择较少的 $m$ 个特征让每一棵决策树进行学习），这样在随机森林中就有了很多个精通不同领域的专家，对一个新的问题（新的输入数据），可以用不同的角度去看待它，最终由各个专家，投票得到结果。这好比是三个臭皮匠顶个诸葛亮的效果。

# k均值聚类(k-means Clustering)

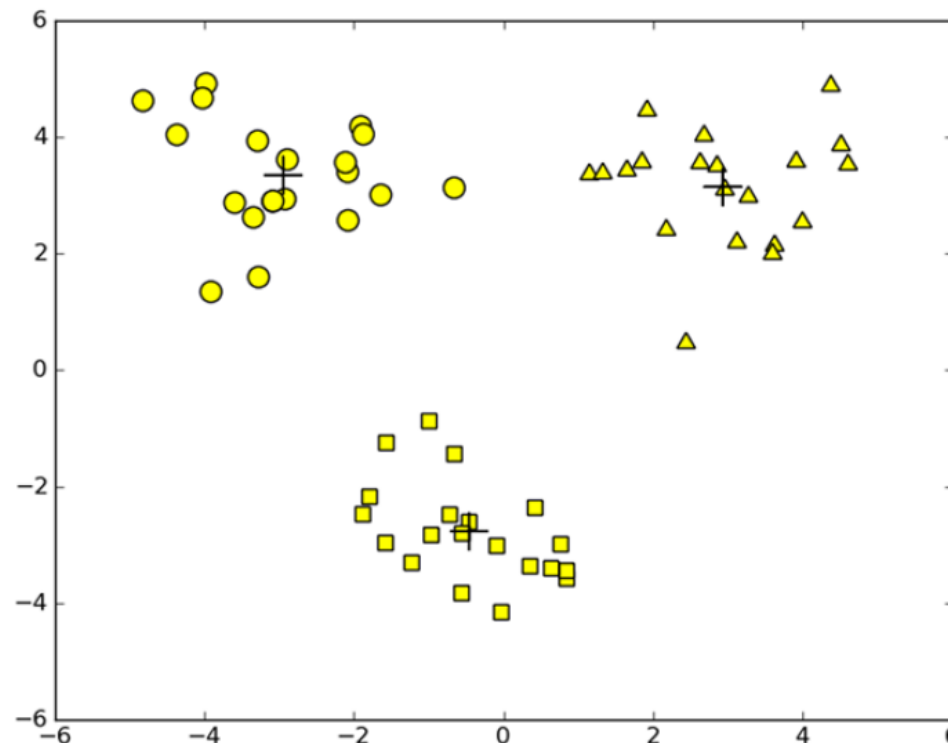
## 基本思想

k-均值是发现给定数据集的k个簇(cluster)的算法。簇的个数k是用户给定的，每一个簇通过其质心(centroid)，即簇中所有点的中心来描述。由于k均值算法实现简单且性能良好，k均值是最常用的聚类算法。

## 如何实现

基本步骤：

- (1) 从  $n$  个样本中任意选择  $k$  个作为初始聚类中心；
- (2) 对每个簇，计算其所有样本的均值得到簇的质心；
- (3) 对每个样本，计算它跟所有簇的质心的距离，并重新将其划分到最近的簇；
- (4) 如果簇不再变化，则停止。否则，循环 (2) 到 (3)。



# 基本K均值聚类算法的改进

## 二分k-均值(bisecting k-means)

为了克服k-均值收敛于局部最优的问题，可以使用。该算法首先将所有点作为一个簇，然后将该簇一分为二。之后选择其中一个簇继续进行划分，选择哪一个簇进行划分，取决于对其划分是否可以最大程度降低SSE，或者选择SSE最大的簇进行划分，直到簇数目达到用于指定的数目为止。

## k-means++ (Weka, R, Oryx都实现了该算法)

k-means算法的先决条件是：必须选择最终结果需要聚为几类，就是k的大小；初始化聚类中心点，也就是seeds。我们可以在输入的数据集中随机的选择k个点作为seeds，但是随机选择初始seeds可能会造成聚类的结果和数据的实际分布相差很大。既然选择初始的seeds这么重要，那有没有办法可以帮助选择初始的seeds？k-means++就是选择初始seeds的一种算法。其基本思想是，初始的聚类中心之间的相互距离要尽可能的远，在选择好k个初始聚类中心后，利用这k个初始的聚类中心来运行标准的k-means算法。

选择初始聚类中心的步骤描述如下：

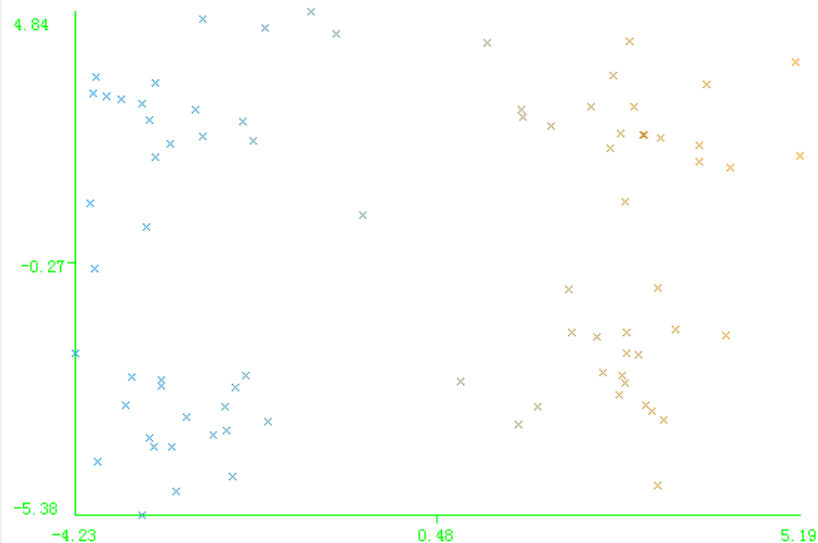
- (1) 从输入的数据点集合中随机选择一个点作为第一个聚类中心
- (2) 对于数据集中的每一个点 $x$ ，计算它与最近聚类中心(指已选择的聚类中心)的距离 $D(x)$
- (3) 选择一个新的数据点作为新的聚类中心，选择的原理是： $D(x)$ 较大的点，被选取作为聚类中心的概率较大
- (4) 重复2和3直到k个聚类中心被选出来

## 重叠聚类和层次聚类

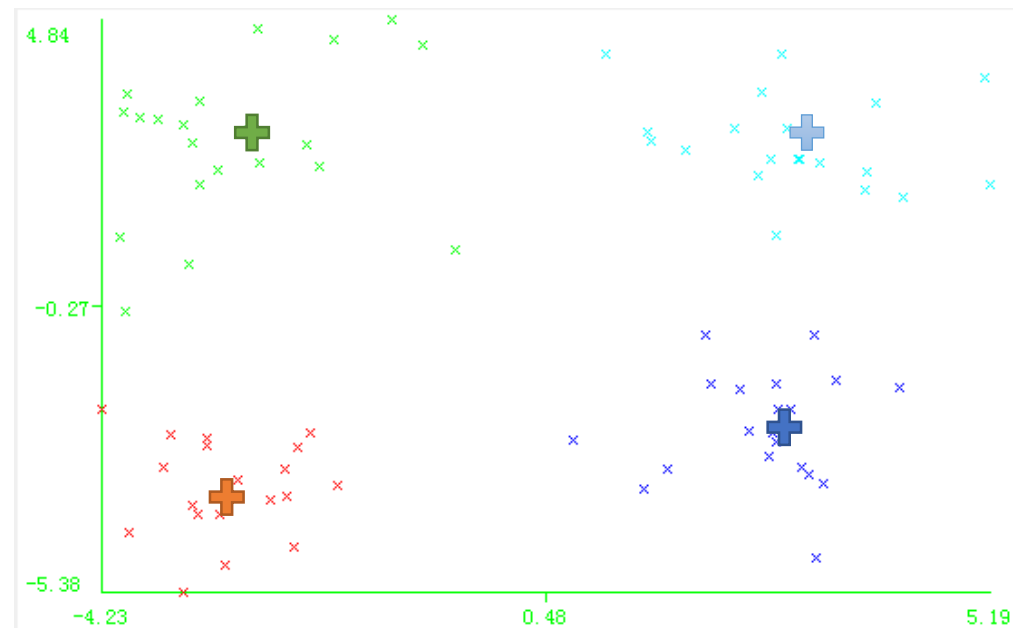
# 示例 - 二维数据

训练数据(Weka/data/kmeans.arff)

No.	1: x Numeric	2: y Numeric
1	1.658985	4.285136
2	-3.453687	3.424321
3	4.838138	-1.151539
4	-5.379713	-3.362104
5	0.972564	2.924086
6	-3.567919	1.531611
7	0.450614	-3.302219
8	-3.487105	-1.724432
9	2.668759	1.594842
10	-3.156485	3.191137
11	3.165506	-3.999838
12	-2.786837	-3.099354
13	4.208187	2.984927
14	-2.123337	2.943366
15	0.704199	-0.479481
16	-0.39237	-3.963704
17	2.831667	1.574018
18	-0.790153	3.343144
19	2.943496	-3.357075
20	-3.195883	-2.283926



聚类结果



Final cluster centroids:

Attribute	Full Data (80)	Cluster#			
		0 (20)	1 (19)	2 (21)	3 (20)
x	-0.1036	-2.4615	-3.5397	2.6508	2.6285
y	0.0543	2.7874	-2.8938	-2.7902	3.1087

# 示例 - 多维数据

训练数据(Weka/data/bank.arff)  
银行客户信息，进行客户分群

聚类结果

Relation: bank									
No.	1: age Numeric	2: sex Nominal	3: region Nominal	4: income Numeric	5: married Nominal	6: children Nominal	7: car Nominal	8: mortgage Nominal	9: pep Nominal
1	48.0	FEMALE	INNER_CITY	17546.0	NO	YES	NO	NO	YES
2	40.0	MALE	TOWN	30085.1	YES	YES	YES	YES	NO
3	51.0	FEMALE	INNER_CITY	16575.4	YES	NO	YES	NO	NO
4	23.0	FEMALE	TOWN	20375.4	YES	YES	NO	NO	NO
5	57.0	FEMALE	RURAL	50576.3	YES	NO	NO	NO	NO
6	57.0	FEMALE	TOWN	37869.6	YES	YES	NO	NO	YES
7	22.0	MALE	RURAL	8877.07	NO	NO	NO	NO	YES
8	58.0	MALE	TOWN	24946.6	YES	NO	YES	NO	NO
9	37.0	FEMALE	SUBURBAN	25304.3	YES	YES	YES	NO	NO
10	54.0	MALE	TOWN	24212.1	YES	YES	YES	NO	NO
11	66.0	FEMALE	TOWN	59803.9	YES	NO	NO	NO	NO
12	52.0	FEMALE	INNER_CITY	26658.8	NO	NO	YES	YES	NO
13	44.0	FEMALE	TOWN	15735.8	YES	YES	NO	YES	YES
14	66.0	FEMALE	TOWN	55204.7	YES	YES	YES	YES	YES
15	36.0	MALE	RURAL	19474.6	YES	NO	NO	YES	NO
16	38.0	FEMALE	INNER_CITY	22342.1	YES	NO	YES	YES	NO
17	37.0	FEMALE	TOWN	17729.8	YES	YES	NO	YES	NO
18	46.0	FEMALE	SUBURBAN	41016.0	YES	NO	NO	YES	NO
19	62.0	FEMALE	INNER_CITY	26909.2	YES	NO	NO	NO	YES
20	31.0	MALE	TOWN	22522.8	YES	NO	YES	NO	NO

Final cluster centroids:

Attribute	Full Data	Cluster#					
		0	1	2	3	4	5
	(300)	(74)	(56)	(31)	(47)	(43)	(49)
=====							
age	42.57	37.2838	45.5714	35.9032	40.7021	45.3023	50.7347
sex	MALE	FEMALE	FEMALE	MALE	MALE	FEMALE	FEMALE
region	INNER_CITY	INNER_CITY	INNER_CITY	INNER_CITY	INNER_CITY	TOWN	TOWN
income	27655.4981	24043.165	29971.3366	22668.251	26668.126	29185.8163	33223.522
married	YES	YES	NO	NO	YES	YES	YES
children	YES	NO	YES	NO	YES	NO	YES
car	NO	YES	NO	YES	YES	NO	YES
mortgage	NO	NO	NO	YES	YES	YES	NO
pep	NO	NO	YES	NO	YES	NO	YES



# 关联分析和Apriori

## 基本思想

- 关联分析(association analysis)。关联分析是指从大规模数据集中寻找项(item)之间的关系。这些关系有两种形式:频繁项集(frequent itemsets)和关联规则(association rules)
- 频繁项集。项集(itemset)的支持度(support)，也就是覆盖量(coverage)，是项集在所有数据集中出现的比例。设置项集的最低支持度阈值，支持度大于阈值的项集称为频繁项集。
- 关联规则是指项集之间的一种依赖关系，形式是，作为前件(antecedent)的项集 ==> 作为后件(consequent)的项集，其含义是如果前件成立则后件成立。
- 关联规则的置信度(confidence):  $\text{confidence}(P \Rightarrow H) = \text{support}(P \mid H) / \text{support}(P)$

交易号	商品			
0	豆奶	生菜		
1	生菜	尿布	葡萄酒	甜菜
2	豆奶	尿布	葡萄酒	橙汁
3	生菜	豆奶	尿布	葡萄酒
4	生菜	豆奶	尿布	橙汁

{尿布, 葡萄酒}的支持度为3/5

{尿布}的支持度为4/5

尿布==>葡萄酒的置信度为3/4=0.75

# 如何实现

首先找到频繁项集，然后再从频繁项集中获得关联规则。

物品组合空间巨大，蛮力(brute-force)搜索效率太低。为了提高效率，常用算法有：

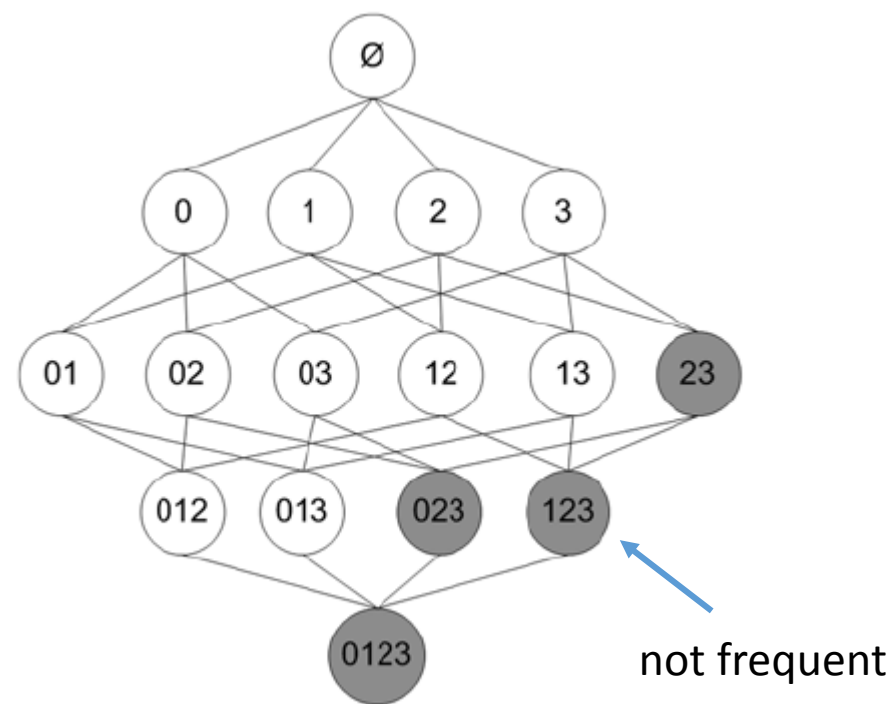
- Apriori：使用Apriori原理，降低搜索空间的大小。
- FP-Growth：使用一个紧凑的数据结构(FP树)，进一步提高算法的效率。

# Apriori算法

**Apriori原理:** 频繁项集的所有子集都是频繁的; 反之, 对一个非频繁项集而言, 其超集也一定是非频繁的。  
不管是发现频繁项集还是关联规则, 算法的核心操作都是join和prune。

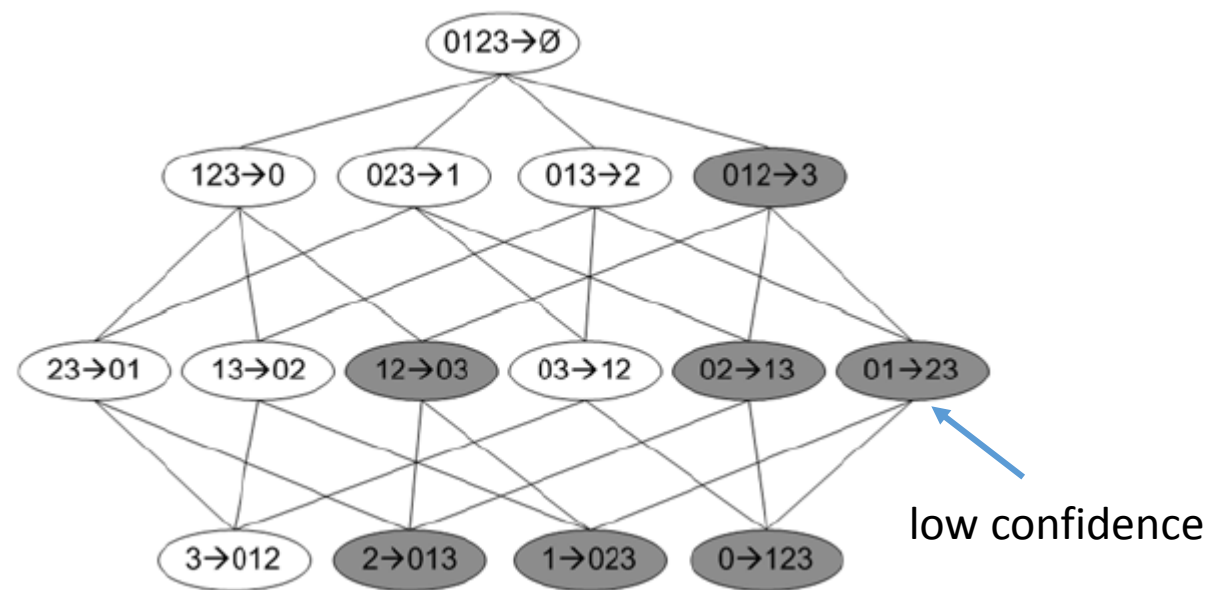
使用Apriori发现频繁项集算法描述:

1. 生成所有单项项集C1
2. 去掉不满足最小支持度的项集得到L1
3. 剩下的项集, 假设每个项集中元素个数为k, 则使用组合方法, 生成包含项个数为k+1的项集集合
4. 重复2和3, 直至3无法组合更大的项集为止



从频繁项集挖掘关联规则算法描述：

1. 为一个频繁项集创建一个初始规则列表，其中规则后件只包含一个项
2. 对这些规则进行筛选，去掉所有不满足最小置信度的规则
3. 假设剩余规则后件包含 $k$ 个项，则合并剩余规则来创建新的规则列表，其中规则后件包含 $k+1$ 个元素
4. 重复2和3，直至无法组合更大的后件为止



# 示例

数据: 超市数据(Weka/data/supermarket.arff)

任务: 超市购物篮分析

运行结果:

```
1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723    <conf: (0.92)> lift: (1.27) lev: (0.03) [155] conv: (3.35)
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696    <conf: (0.92)> lift: (1.27) lev: (0.03) [149] conv: (3.28)
3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705    <conf: (0.92)> lift: (1.27) lev: (0.03) [150] conv: (3.27)
4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf: (0.92)> lift: (1.27) lev: (0.03) [159] conv: (3.26)
5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779    <conf: (0.91)> lift: (1.27) lev: (0.04) [164] conv: (3.15)
6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf: (0.91)> lift: (1.26) lev: (0.03) [151] conv: (3.06)
7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf: (0.91)> lift: (1.26) lev: (0.03) [145] conv: (3.01)
8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866    <conf: (0.91)> lift: (1.26) lev: (0.04) [179] conv: (3)
9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757    <conf: (0.91)> lift: (1.26) lev: (0.03) [156] conv: (3)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877    <conf: (0.91)> lift: (1.26) lev: (0.04) [179] conv: (2.92)
```

前件

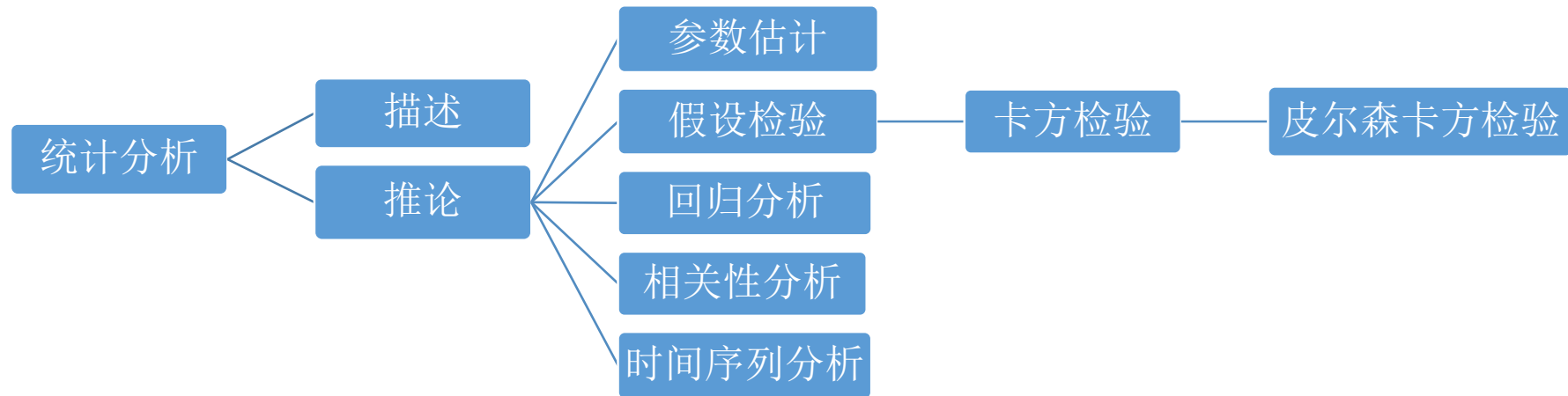
后件

支持度

置信度

# 独立性检验(Test for Independence)

## 统计分析(Statistical Analysis)方法的家谱



- 描述(description): 总结叙述收集来的数据, 如均值、方差、中位数等等。
- 推论(inference): 将样本数据模型化, 计算它的概率并且做出对于母群体的推论
- 参数估计: 是指用样本指标(称为统计量)估计总体指标(称为参数)
- 假设检验: 根据样本观测值来判断一个有关总体的假设是否成立的问题。基本思想可以用小概率原理来解释。

# 参数估计和假设检验示例

某一群人的身高构成一个总体，通常认为身高是服从正态分布的，但不知道这个总体的均值，随机抽部分人，测得身高的值，用这些数据来估计这群人的平均身高，这就是一种统计推断形式，即参数估计。若感兴趣的问题是“平均身高是否超过1.7（米）”，就需要通过样本检验此命题是否成立，这也是一种推断形式，即假设检验。由于统计推断是由部分（样本）推断整体（总体），因此根据样本对总体所作的推断，不可能是完全精确和可靠的，其结论要以概率的形式表达。

## 卡方检验和皮尔森卡方检验

卡方检验(chi square test): 因为使用卡方分布(chi-square distribution), 故得名。它根据次数数据判断两类因子是否相互独立。

皮尔森卡方检验: 最有名的卡方检验。当提及卡方检验而没有特别指明类型时，通常即指皮尔森卡方检验。

# 示例 – 用卡方检验测试独立性

<http://www.r-tutor.com/elementary-statistics/goodness-fit/chi-squared-test-independence>

```
> library(MASS)    # load the MASS package
> tbl = table(survey$Smoke, survey$Exer)
> tbl              # the contingency table
```

	Freq	None	Some
Heavy	7	1	3
Never	87	18	84
Occas	12	3	4
Regul	9	1	7

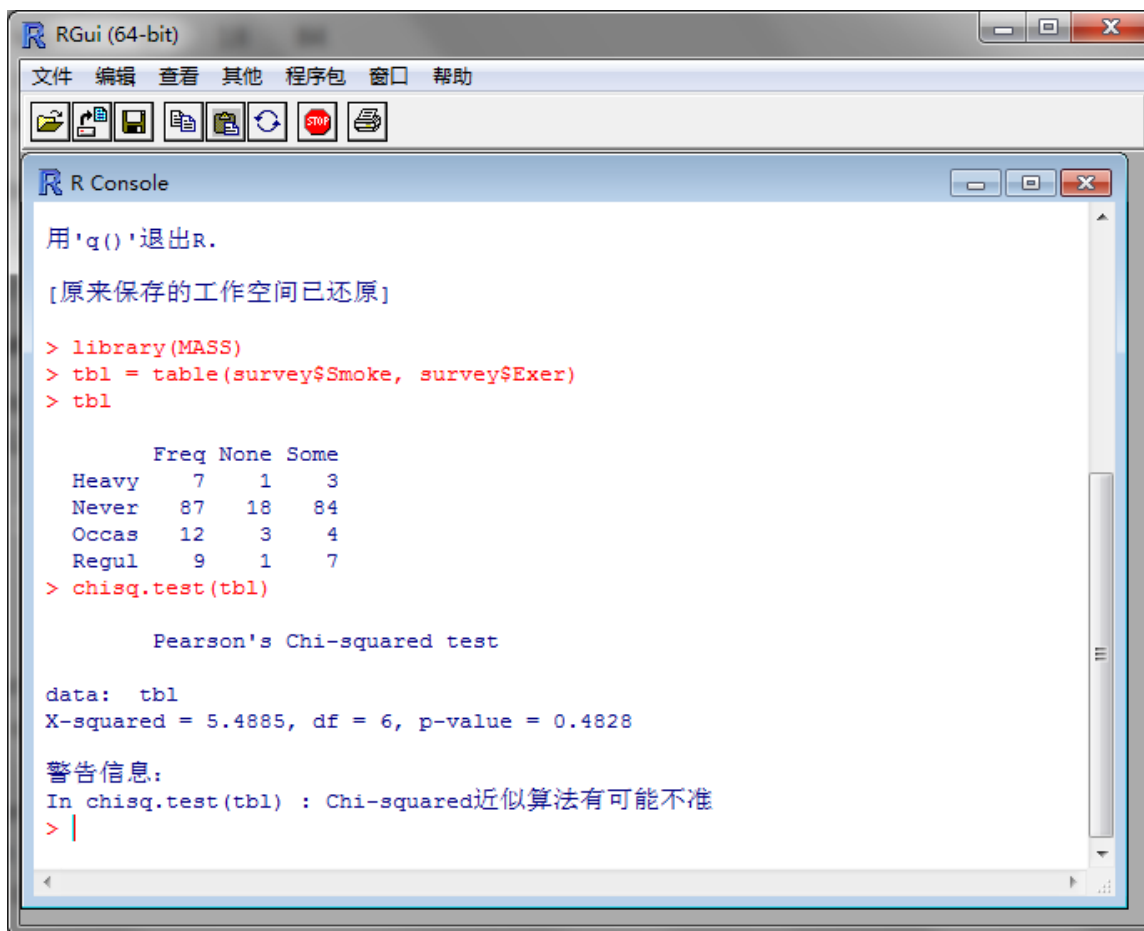
```
> chisq.test(tbl)
```

Pearson's Chi-squared test

```
data: table(survey$Smoke, survey$Exer)
X-squared = 5.4885, df = 6, p-value = 0.4828
```

Warning message:

```
In chisq.test(table(survey$Smoke, survey$Exer)) :
  Chi-squared approximation may be incorrect
```



```
RGui (64-bit)
文件 编辑 查看 其他 程序包 窗口 帮助

R Console

用'q()'退出R.
[原来保存的工作空间已还原]

> library(MASS)
> tbl = table(survey$Smoke, survey$Exer)
> tbl

      Freq None Some
Heavy    7    1    3
Never   87   18   84
Occas   12    3    4
Regul    9    1    7
> chisq.test(tbl)

      Pearson's Chi-squared test

data:  tbl
X-squared = 5.4885, df = 6, p-value = 0.4828

警告信息:
In chisq.test(tbl) : Chi-squared近似算法有可能不准
> |
```



# 时间序列分析(Time Series Analysis)

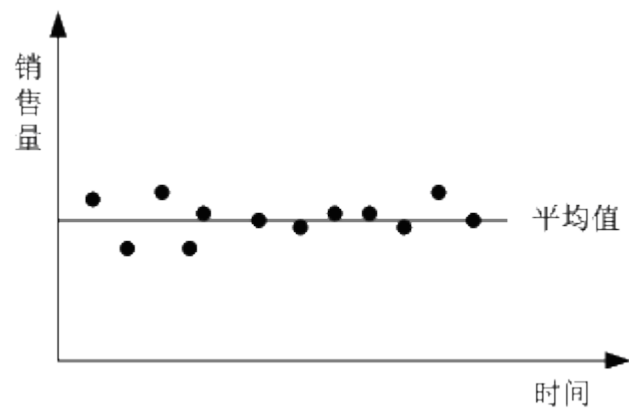
## 基本思想

时间序列是指把历史统计资料按时间顺序排列起来得到的一组数据序列。例如，按月份排列的某种商品的销售量；工农业总产值按年度顺序排列起来的数据序列等等，都是时间序列。

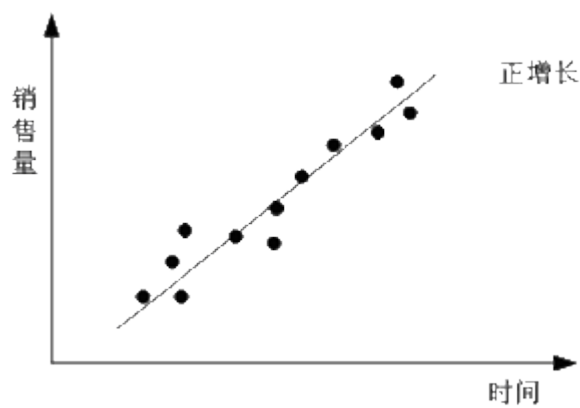
时间序列预测法是将预测目标的历史数据按时间的顺序排列成为时间序列，然后分析它随时间变化的发展趋势，外推预测目标的未来值。也就是说，时间序列预测法将影响预测目标的一切因素都由“时间”综合起来加以描述。因此，时间序列预测法主要用于分析影响事物的主要因素比较困难或相关变量资料难以得到的情况，预测时先要进行时间序列的模式分析。

时间序列预测法通常又分为移动平均法、指数平滑法、趋势外推法、季节分析法和生命周期法等多种方法。

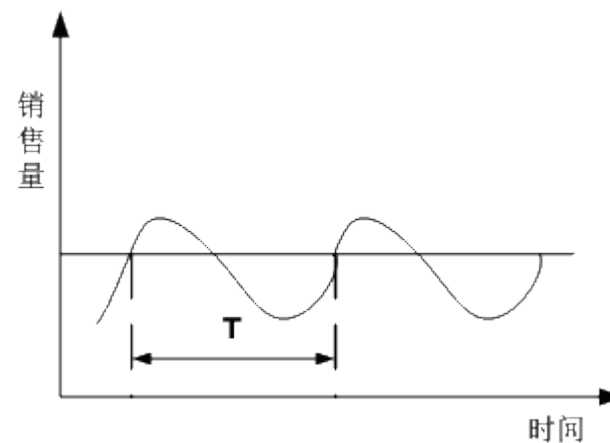
不同的时间序列预测方法只适用于一定的数据时间序列模式。因此，对时间序列模式的理解是学习时间序列预测方法的基础。时间序列的模式，是指历史时间序列所反映的某种可以识别的事物变动趋势形态。时间序列的基本模式，可以归纳为水平型、趋势型、周期变动型和随机型等四种类型。



水平型



趋势型



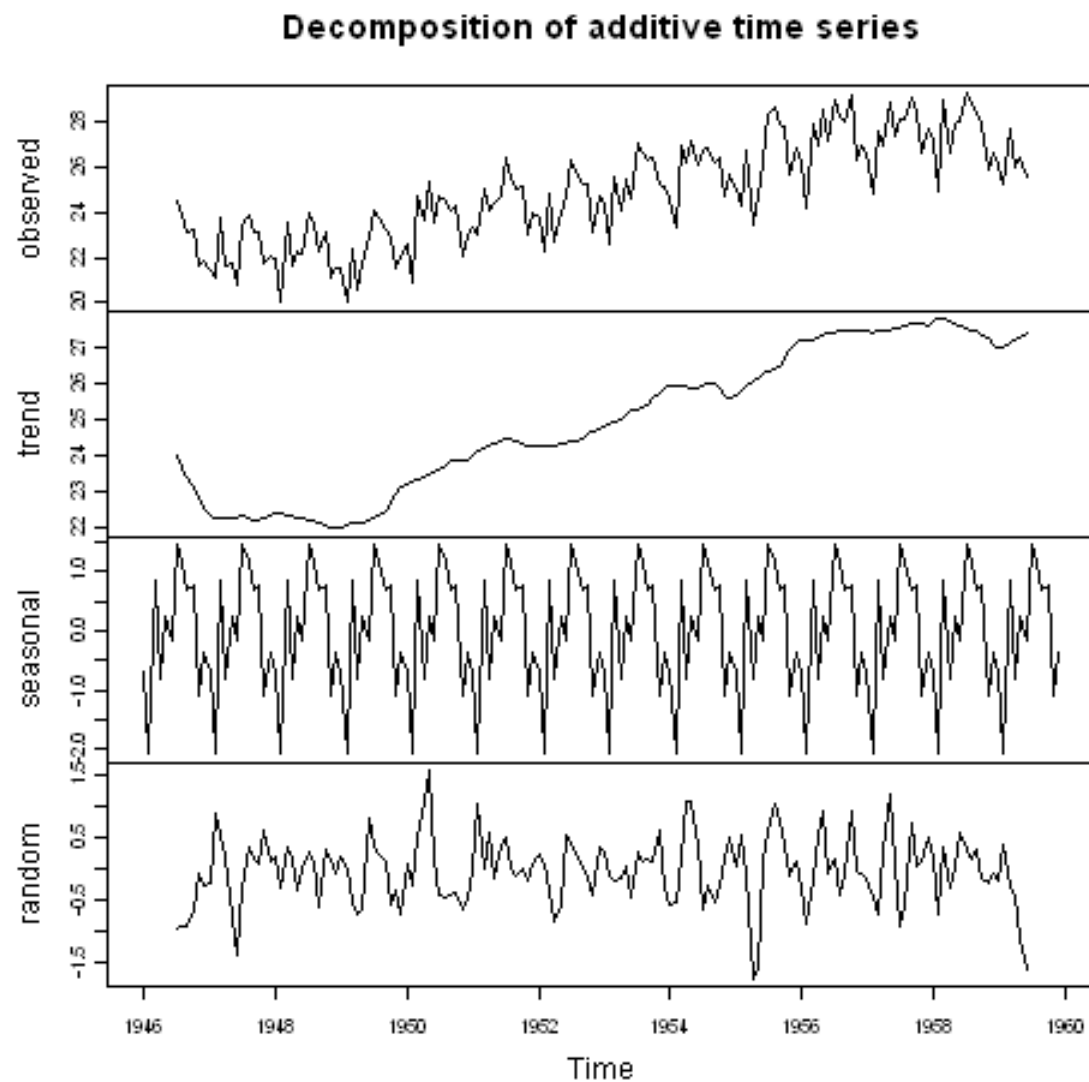
季节型

# 如何实现

常用的时间序列分析方法包括：

- 传统方法是ARMA、ARIMA、ARIMAX等模型
- 指数平滑法，最常见是Holt-Winters模型
- 时间序列分析的最新方法倾向于使用SVM方法

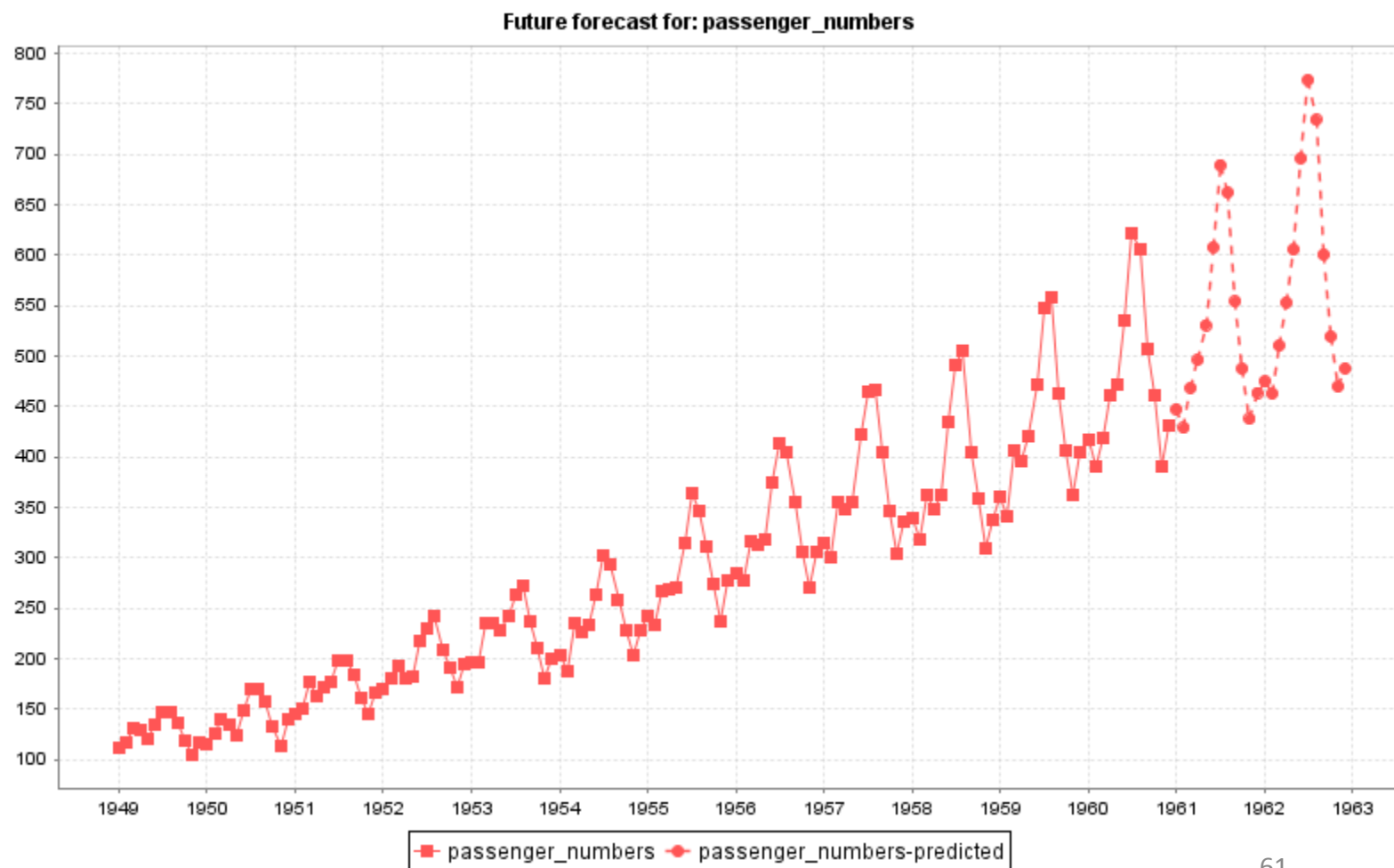
## 示例 – 用R分析时间序列



# 示例 -用Weka分析和预测时间序列

预测航班乘客人数(/wekafiles/packages/timeseriesForecasting/sample-data/airline.arff)

No.	1: passenger_numbers Numeric	2: Date Date
1	112.0	1949-01-01
2	118.0	1949-02-01
3	132.0	1949-03-01
4	129.0	1949-04-01
5	121.0	1949-05-01
6	135.0	1949-06-01
7	148.0	1949-07-01
8	148.0	1949-08-01
9	136.0	1949-09-01
10	119.0	1949-10-01
11	104.0	1949-11-01
12	118.0	1949-12-01
13	115.0	1950-01-01
14	126.0	1950-02-01
15	141.0	1950-03-01
16	135.0	1950-04-01
17	125.0	1950-05-01
18	149.0	1950-06-01
19	170.0	1950-07-01
20	170.0	1950-08-01
21	158.0	1950-09-01
22	133.0	1950-10-01
23	114.0	1950-11-01
24	140.0	1950-12-01



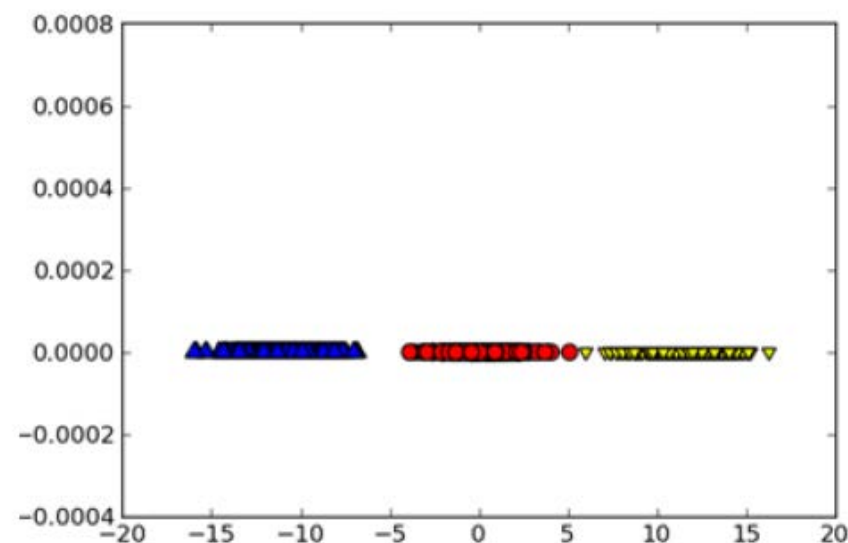
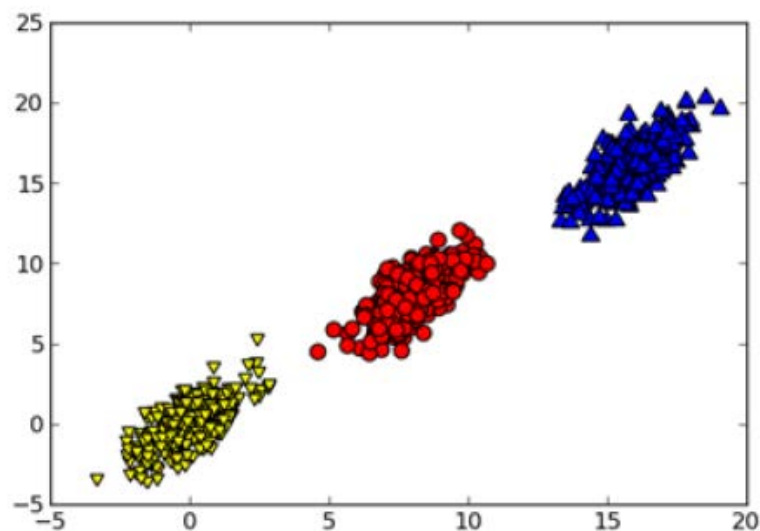
- Weka的时间序列框架，使用机器学习(machine learning)/数据挖掘(data mining)方法对时间序列进行建模，基本思路是将数据转换成标准的学习算法能够处理的形式。
- 将时间上的依赖关系转换为额外的输入特征，这些特征有时被称为滞后变量(lagged variables)。另外，还增加了其他一些特征，用来对时间序列的趋势(trends)和周期性(seasonality)进行建模。数据经过转换后，Weka的任何回归模型都可以用来学习模型。
- 很显然可以使用(multiple linear regression)方法，实际上能够预测连续目标值的方法都可以，包括强大的非线性算法，比如支持向量回归(support vector machines for regression)和模型树(model trees,叶节点为线性回归函数的决策树)。
- 用这种方法进行时间序列的分析和预测，往往比经典的统计技术如ARMA或ARIMA等更加强大和灵活。

# 主成分分析(PCA)

## 基本思想

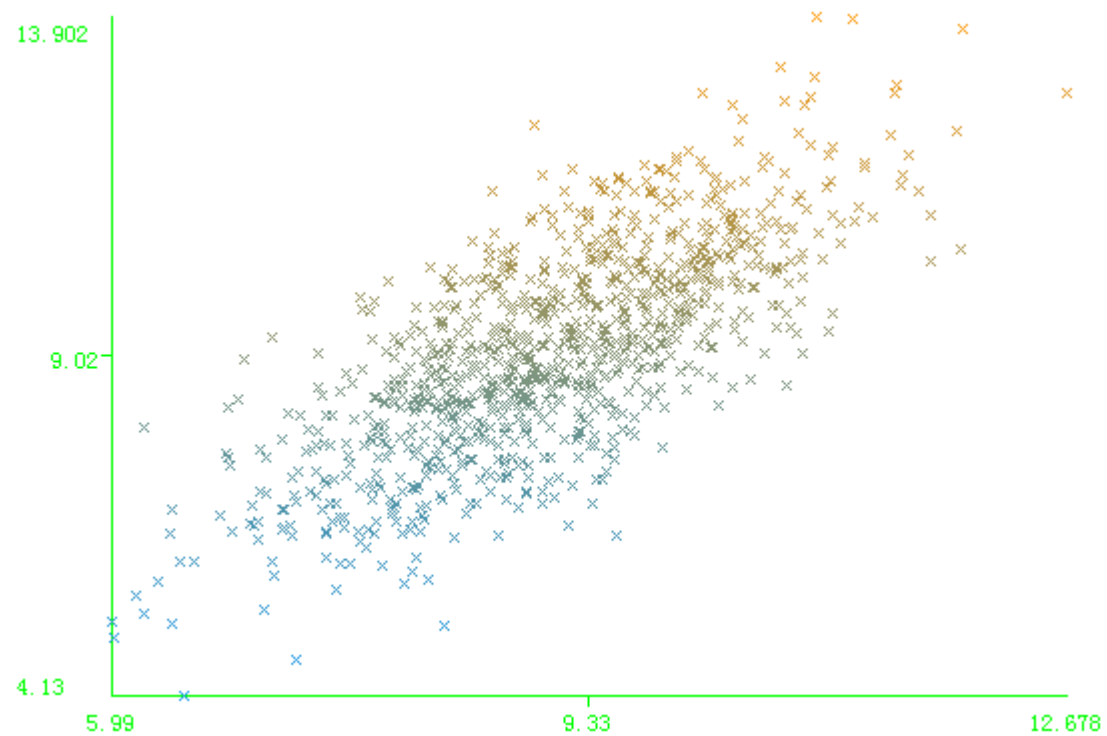
PCA (Principal Component Analysis) 可以从数据中识别其主要特征，它是通过沿着数据最大方差方向旋转坐标来实现的。

如图所示，原来的输入特征有2个，通过逆时针旋转坐标轴，可以让一个维度的特征变成常量，既然是常量，训练模型的时候，这个维度的数据就可以忽略，这样，经过PCA后的特征就减少为1个。



# 示例 - 二维数据

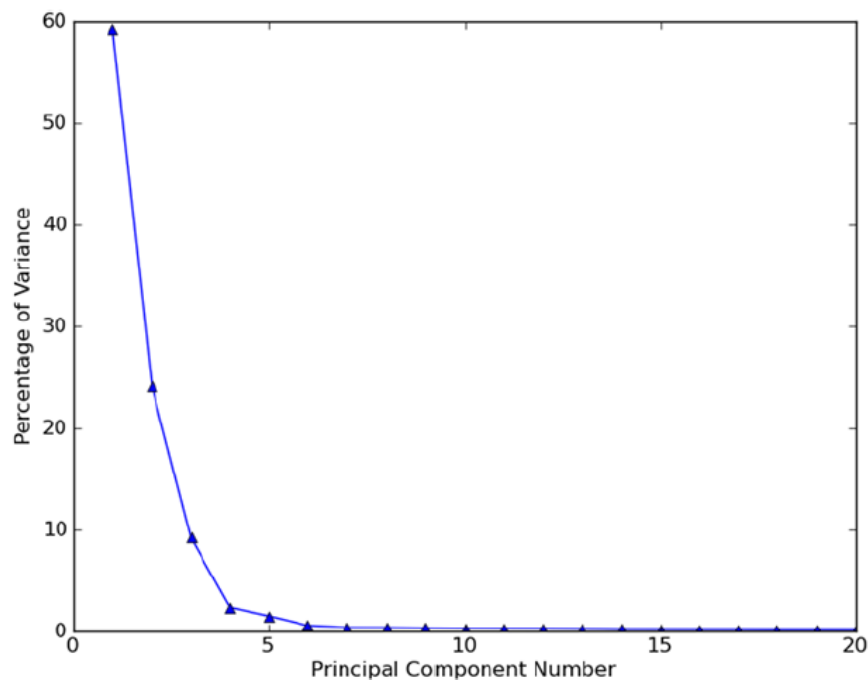
训练数据(Weka/data/pca-2D.arff)。





# 示例 - 多维数据

数据集(<http://archive.ics.uci.edu/ml/machine-learning-databases/secom/>)



% variance for the first 7 principal components of the semiconductor data

Principal component	% Variance	% Cumulative
1	59.2	59.2
2	24.1	83.4
3	9.2	92.5
4	2.3	94.8
5	1.5	96.3
6	0.5	96.8
7	0.3	97.1
20	0.08	99.3

方差较大的前20个特征，包含了大部分的信息。仅保留前6个特征，包含的方差占有所有特征的方差总和的96.8%。将590个特征减少到6个特征，相当于压缩了将近100倍。

# 第三部分：机器学习的高级主题

- 机器学习三要素(模型,策略,算法)
- 模型选择
- 集成学习
- 预处理
- 数据挖掘十大算法
- 机器学习的最新进展：深度学习

# 机器学习三要素：模型、策略、算法

- 以监督学习为例，从给定的、有限的、用于学习的训练数据(training data)集合出发，假设数据是独立同分布产生的；并且假设要学习的模型属于某个函数的集合，称为假设空间(hypothesis space)；
- 应用某个评价准则(evaluation criteria)，从假设空间中选取一个最优的模型，使它对已知训练数据及未知测试数据(test data)在给定的评价准则下有最优的预测；
- 最优模型的选取由算法实现。
- 模型的假设空间、模型选择的准则、模型学习的算法，称为机器学习方法的三要素，简称为模型(model)、策略(strategy)、算法(algorithm)。机器学习方法之间的不同，主要来自其模型、策略、算法的不同。确定了模型、策略和算法，机器学习方法也就确定了，这也就是将其称为机器学习三要素的原因。

要素	例子	说明
模型	线性模型	输入空间(input space)为实数 输出空间(output space)为实数 a和b是权重(weight)，也称为参数(parameter)
策略	OLS	Ordinary Least Square
算法	梯度下降法	微积分、线性代数

# 模型

输入和输出：

- 输入空间(input space)
- 输出空间(output space)
- 每个输入是一个实例(instance)，通常由特征向量(feature vector)表示，所有特征向量存在的空间称为特征空间(feature space)
- 所有输入实例可以表示为一个矩阵(matrix)

模型本质上是决策函数或者条件概率模型，但是可以多种方式表示，比如：

- 条件概率分布 $P(Y|X)$
- 决策函数(decision function)
- 分类规则
- 关联规则
- 决策树
- 聚类

# 策略

损失函数(loss function)

平方损失函数 (quadratic loss function)

$$L(Y, f(X)) = (Y - f(X))^2$$

绝对损失函数 (absolute loss function)

$$L(Y, f(X)) = |Y - f(X)|$$

风险函数(risk function):  
损失函数的期望值

经验风险 :  $R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i))$

经验风险最小化(Empirical  
Risk Minimization)策略

经验风险最小化 :  $\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i))$

结构风险最小化(Structural  
Risk Minimization)策略 = 正则化(Regularization)

结构风险最小化 :  $R_{\text{sm}}(f) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i)) + \lambda J(f)$

# 算法

输入和输出：

- 输入：训练数据和测试数据
- 输出：模型

基于训练数据集，根据学习策略，从假设空间中选择最优模型，这样，机器学习问题归结为最优化问题。所以，很多机器学习问题可以借助最优化算法来求解。

算法用到的数学：

- 线性代数：线性回归、矩阵分解、主成分分析
- 数理统计和概率论：朴素贝叶斯使用极大似然估计、贝叶斯估计
- 微积分：逻辑回归中使用梯度下降
- 信息论：决策树中基于熵的分裂特征选择
- 最优化、规划问题：**SVM**使用的拉格朗日对偶法

# 模型选择

## 正则化和交叉验证

正则化和交叉验证，是两种常用的模型选择方法。

正则化(regularization): 对模型的复杂度进行惩罚。这符合奥卡姆剃刀原理(Occam's Razor)。在所有可能选择的模型中，能够很好地解释已知数据并且十分简单才是最好的模型。

交叉验证(cross-validation): 把给定的数据进行切分，将切分的数据集组合为训练集和测试集，在此基础上反复地进行训练、测试以及模型选择。

- 简单交叉验证
- S折交叉验证，如10-fold cross-validation
- 留一交叉验证

# 训练误差(training error)和测试误差(test error)

监督学习从训练数据(training data)集合中学习模型，对测试数据(test data)进行预测。

训练误差(training error): 关于训练数据集的平均损失

$$R_{\text{emp}}(\hat{f}) = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{f}(x_i))$$

测试误差(test error): 关于测试数据集的平均损失

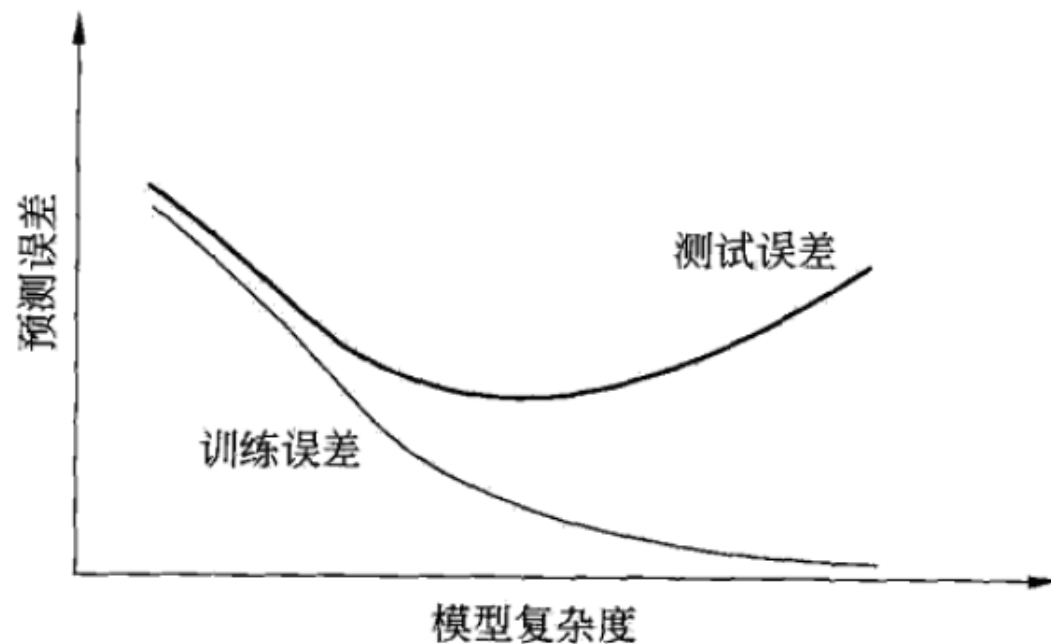
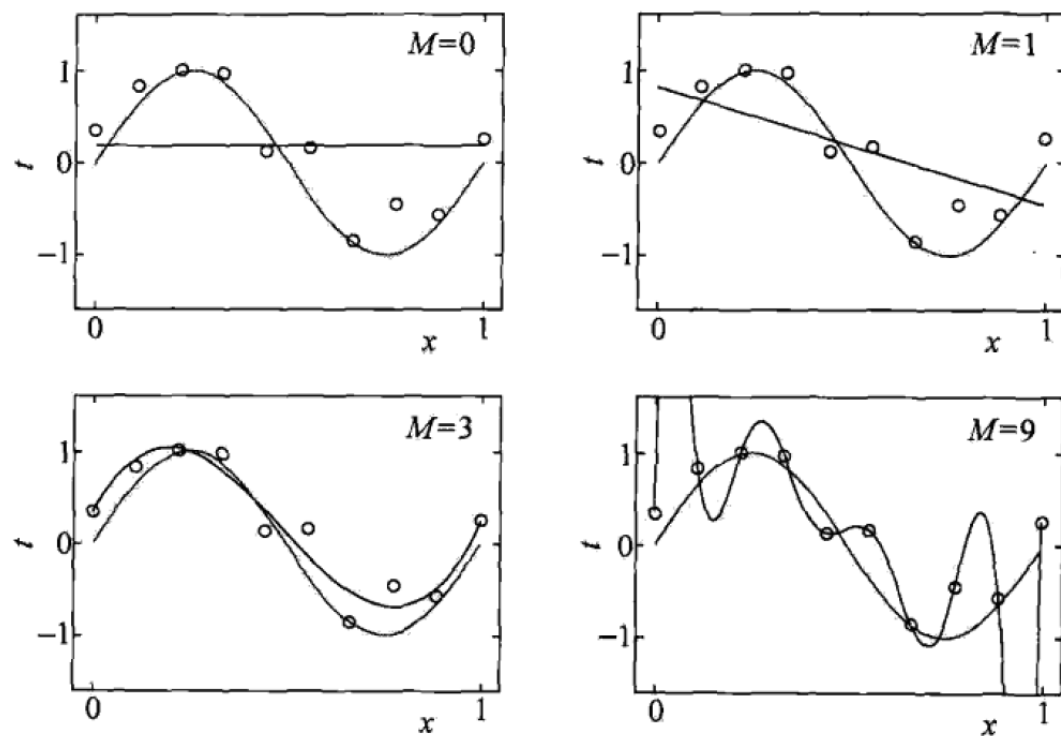
$$e_{\text{test}} = \frac{1}{N'} \sum_{i=1}^{N'} L(y_i, \hat{f}(x_i))$$



# 过拟合(Overfit)

过拟合(overfit)的例子：M次多项式函数拟合

$$f_M(x, w) = w_0 + w_1x + w_2x^2 + \cdots + w_Mx^M = \sum_{j=0}^M w_jx^j$$



可以说，模型选择(model selection)旨在避免过拟合并提高模型的预测能力。

# 集成学习(Ensemble)

- 也叫元算法，使用弱分类器学习强分类器。包括：
  - 装袋(bagging)
  - 提升(boosting)
  - 堆栈(stack)
- 随机森林，正是bagging方法的一种。
- Adaboost是一种常用的boosting算法。

# Bagging

Bagging是Bootstrap aggregating的缩写，跟bag并没有关系。bagging方法是一个统计重采样(resampling)的技术，它的基础是bootstrapping，本质上是一种可重复的随机采样技术(random sampling with replacement)。

基本思想是：利用bootstrapping方法重采样来生成多个版本的预测分类器，然后把这些分类器进行组合。通常情况下组合的分类器给出的结果比单一分类器的好，因为综合了各个分类器的特点。之所以用可重复的随机采样技术bootstrapping，是因为进行重复的随机采样所获得的样本可以得到没有或者含有较少的噪声数据。

# 预处理

- 把机器学习技术应用于实际的数据挖掘问题时，需要进行必要的转换，将输入数据设计成一种适合所选学习方案的形式
- 预处理技术
  - 特征选择(feature selection)
  - 属性离散化(discretizing numeric features)
  - 抽样(sampling)
  - 数据投影(projection)
  - 数据清洗(cleansing)
  - 多分类转二分类问题(transforming multiple classes to binary ones)
  - 降维(Dimensionality Reduction)
  - 归一化(Normalization)和标准化(Standardization)

## 特征选择(Feature Selection)

- 两种方法
  - 过滤(filter)
  - 包装(wrapper)
- 机器学习算法可用于属性选择
  - 在整个数据集上应用决策树算法，然后选择那些在决策树中真正用到的属性。
  - 建立一个线性模型，根据系数的大小来进行属性排序。系数缩减技术属于这一类：如岭回归(Ridge Regression)、LASSO(Least Absolute Shrinkage and Selection Operator)

## 特征离散化(Discretizing Numeric Features)

- 无监督离散化
  - 等间距装箱
  - 等频装箱
  - 均衡k区间离散化
- 基于熵的离散化

## 数据投影(Projection)

- 主成分分析(PCA)
- 随机投影
- 偏最小二乘回归(partial least square)
- 文本的投影
  - 词频(TF)
  - 词频-反文档词频(TF-IDF)
- 时间序列的投影
  - 滞后变量(lag variable)
  - 外部变量(external variable)

## 数据清洗(Cleansing)

- 去除噪声
- 处理离群点
- 处理缺失值

## 抽样(Sampling)

- 抽样的目的
  - 降低数据量
  - 将数据划分为训练集和测试集
- 常用抽样方法
  - 有放回抽样
  - 无放回抽样
  - 蓄水池抽样

## 多分类转二分类

- 一对多(one vs rest)
- 成对分类(pairwise classification)

## 降维

- 特征选择(feature selection)
- 特征投影(feature projection)

## 归一化和标准化

- Normalization
- Standardization



# 数据挖掘十大算法

排名	算法	类别	备注
1	C4.5	分类	决策树
2	K均值	聚类	
3	SVM	统计学习	支持向量机
4	Apriori	关联分析	
5	EM	统计学习	Expectation-Maximization(期望最大化)
6	PageRank	链接挖掘	Google搜索引擎的核心算法
7	Adaboost	集成学习	提升法
8	kNN	分类	K近邻
9	Naive Bayes	分类	朴素贝叶斯
10	CART	分类	分类回归树

# 机器学习的最新进展：深度学习(Deep Learning)

神经网络  
一派

神经网络与支持向量机  
一直处于“竞争”关系。

支持向量  
机一派

[1986]

1986年，Rummelhart与McClelland发明了神经网络的学习算法Back Propagation。

[1992]

后来，Vapnik等人于1992年提出了支持向量机。神经网络是多层（通常是三层）的非线性模型，支持向量机利用核技巧把非线性问题转换成线性问题。Scholkopf是Vapnik的大弟子，支持向量机与核方法研究的领军人物。据Scholkopf说，Vapnik当初发明支持向量机就是想“干掉”神经网络（He wanted to kill Neural Network）。支持向量机确实很有效，一段时间支持向量机一派占了上风。

[2006]

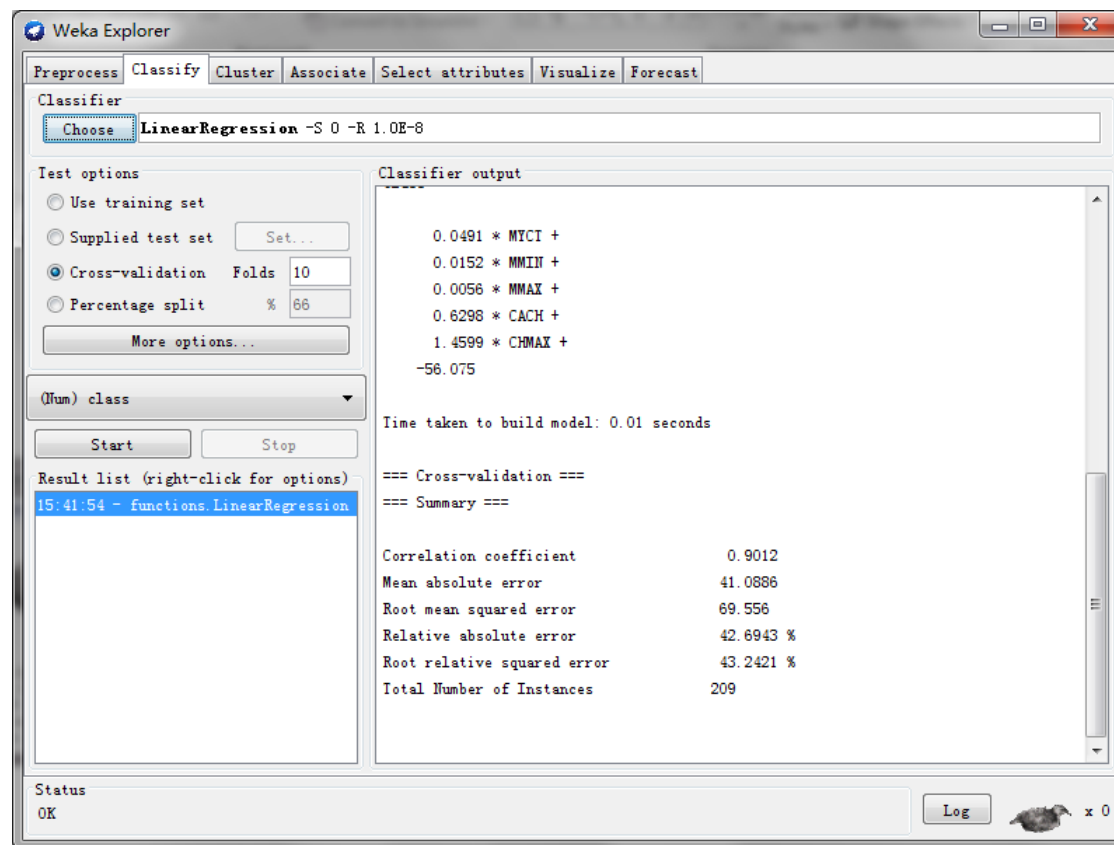
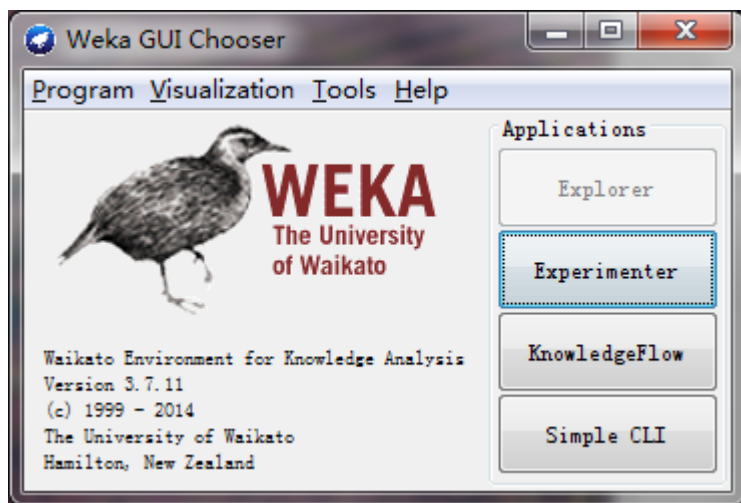
近年来，神经网络一派的大师Hinton又提出了神经网络的Deep Learning算法（2006年），使神经网络的能力大大提高，可与支持向量机一比。Deep Learning假设神经网络是多层的，首先用Boltzman Machine(非监督学习)学习网络的结构，然后再通过Back Propagation(监督学习)学习网络的权值。关于Deep Learning的命名，Hinton曾开玩笑地说：I want to call SVM shallow learning. (注：shallow 有肤浅的意思)。其实Deep Learning本身的意思是深度学习，因为它假设神经网络有多层。Deep Learning是值得关注的统计学习新方法。

# 第三部分：机器学习开源框架

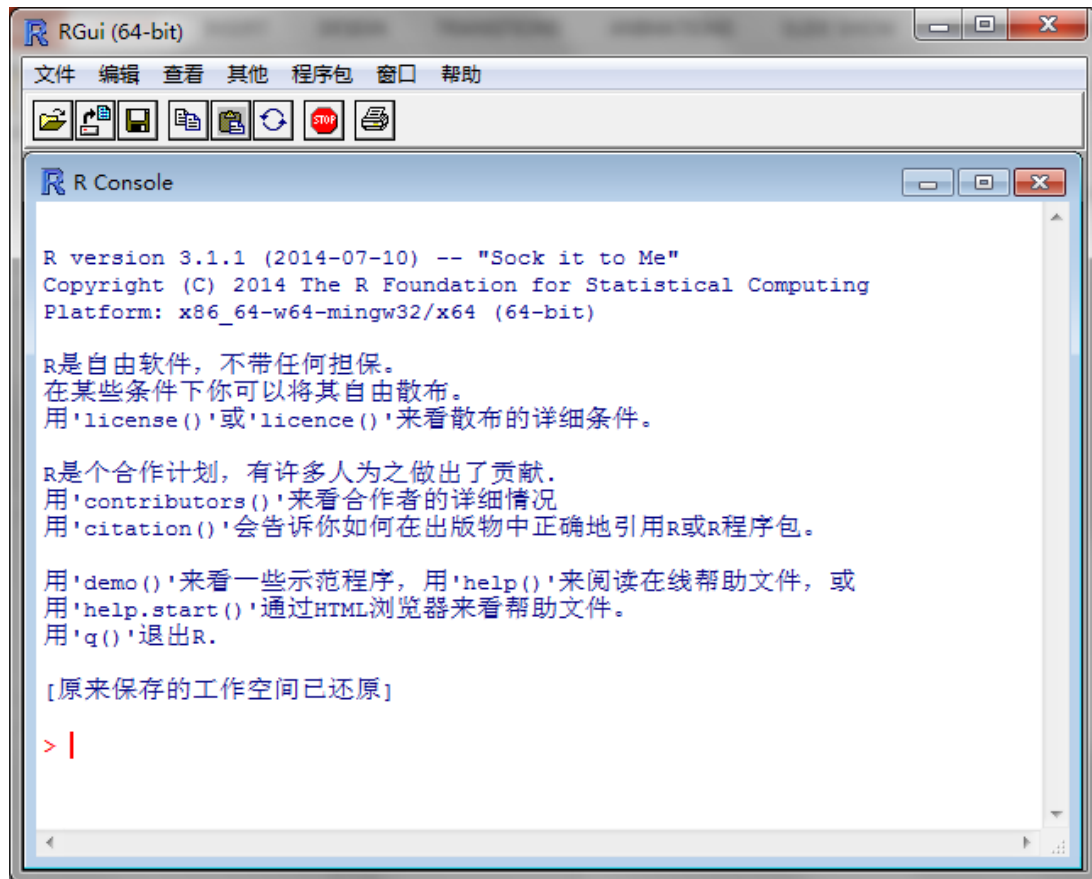
- 探索、原型阶段的工具
  - Weka
  - R
  - Scikit-learn
- 大数据挖掘工具：
  - 基于Hadoop的框架(Mahout 0.9/Oryx 1.0)
  - 基于Spark的的框架(Spark MLlib/Mahout 1.0/Oryx 2.0)
  - 基于Storm的框架

# 探索、原型阶段的工具

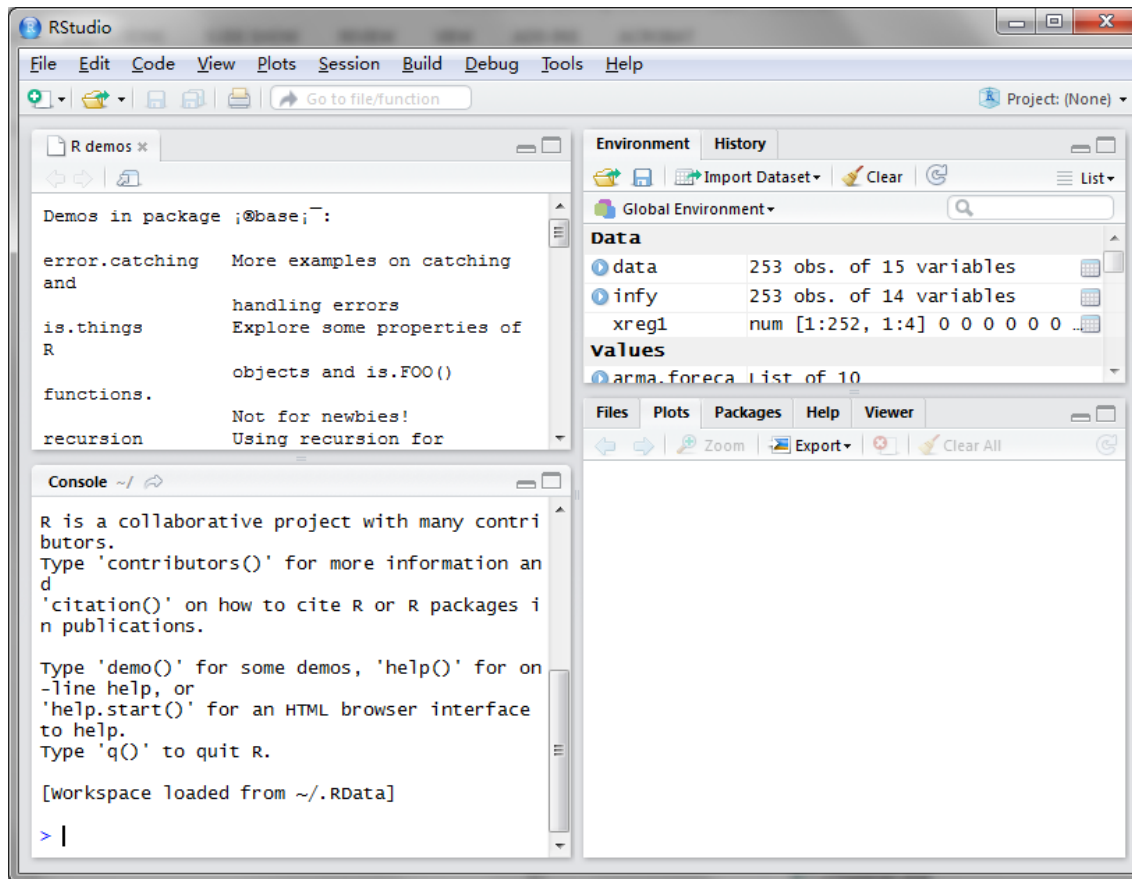
Java写的著名的数据挖掘工具



# R语言和统计分析平台

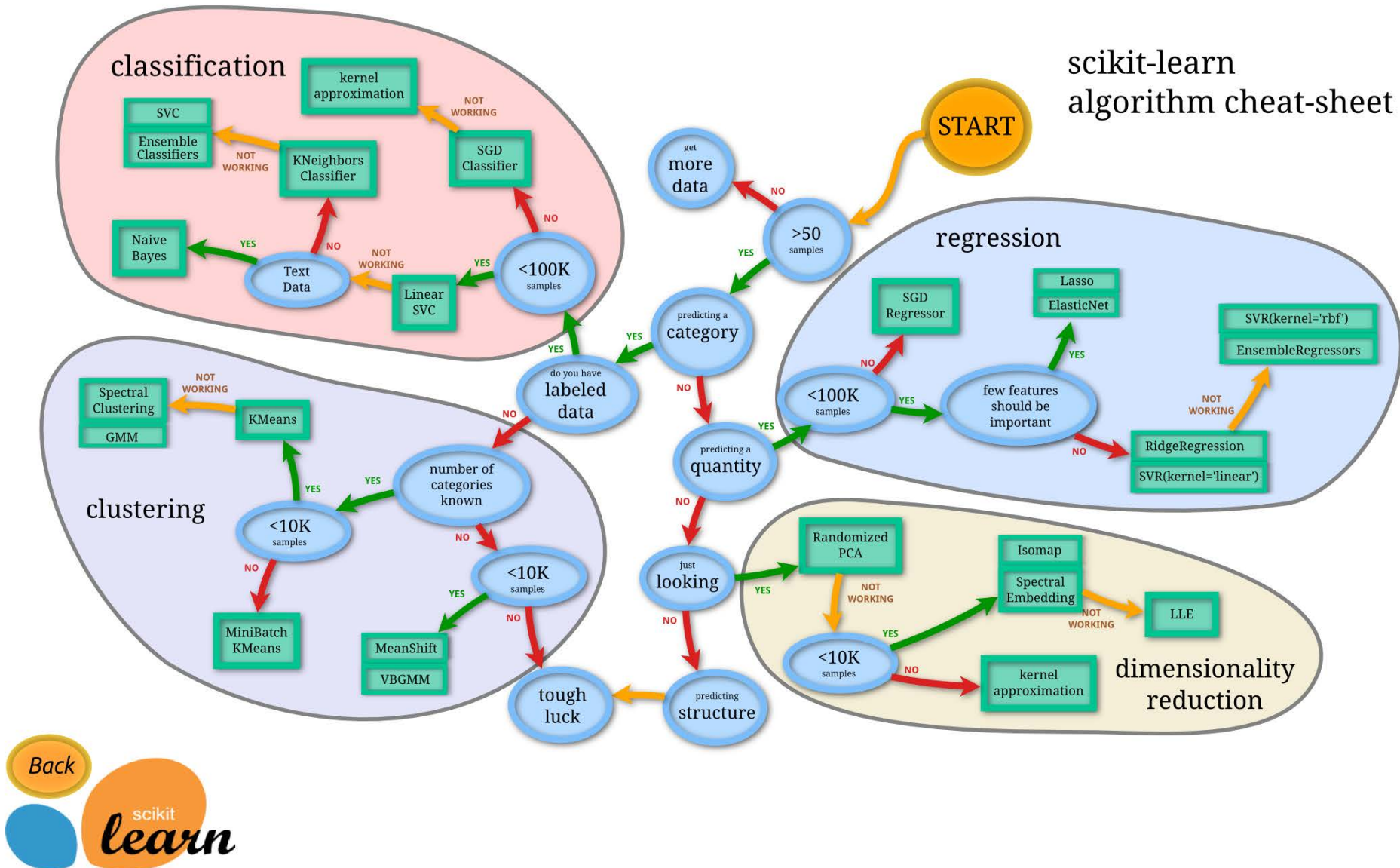


R Console



RStudio

# scikit-learn: Machine Learning in Python



# 大数据挖掘工具



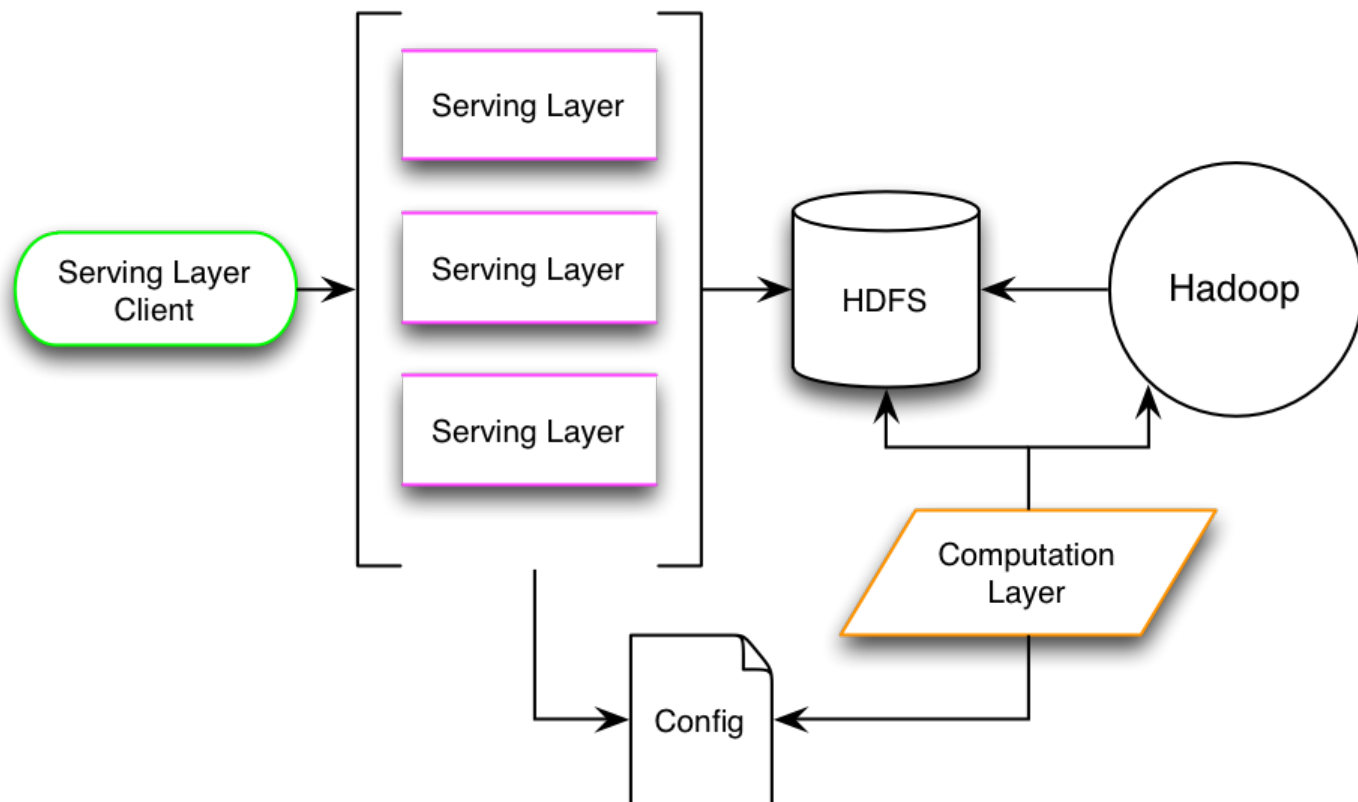
- 算法
  - 推荐(recommendation)
  - 分类(classification)
  - 聚类(clustering)
- 实现
  - 0.9为止基于MapReduce
  - 从1.0开始转向Spark,H2O,Flink

算法类别	算法名称
Recommendation	User-Based/Item-Based Collaborative Filtering
	Matrix Factorization with ALS (on Implicit Feedback)
	Weighted Matrix Factorization, SVD++
Classification	Logistic Regression - trained via SGD
	Naive Bayes / Complementary Naive Bayes
	Random Forest
	Hidden Markov Models
Clustering	Multilayer Perceptron
	k-Means Clustering
	Fuzzy k-Means
	Streaming k-Means
Dimensionality Reduction	Singular Value Decomposition
	Lanczos Algorithm
	PCA (via Stochastic SVD)
Topic Models	Latent Dirichlet Allocation (LDA)



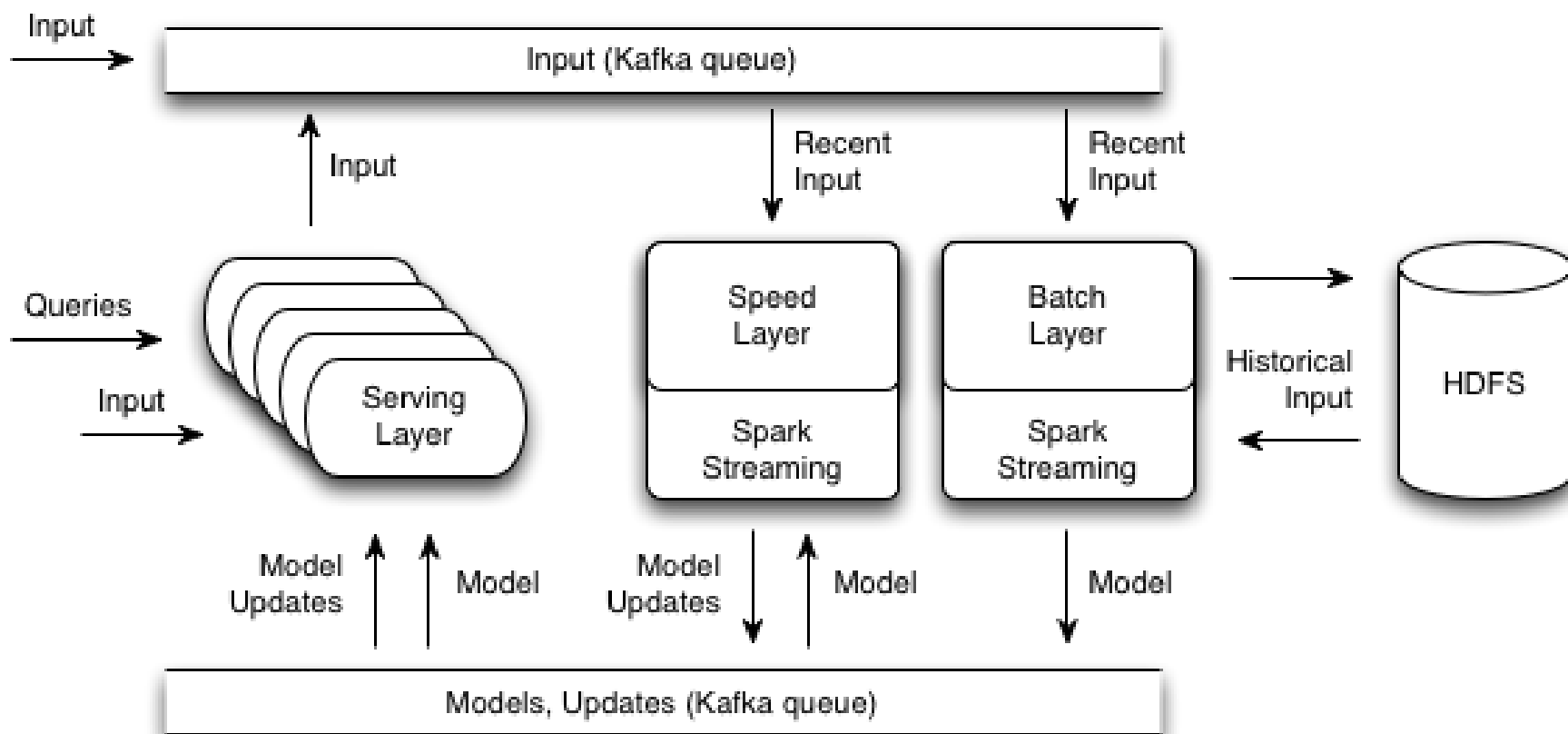
Oryx

算法类别	算法名称
Recommendation/ Collaborative filtering/ Matrix factorization	Alternating Least Squares (ALS)
Classification and Regression	Random Decision Forests (RDF)
Clustering	scalable k-means++



Oryx 1.x Architecture





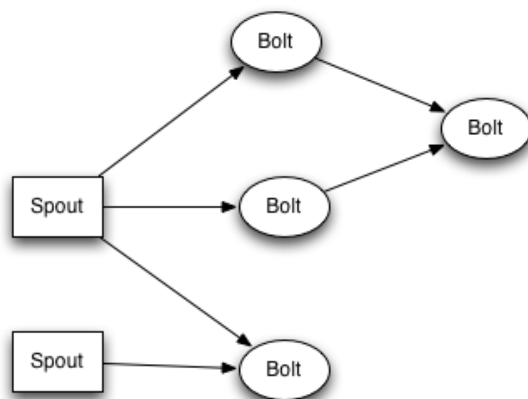
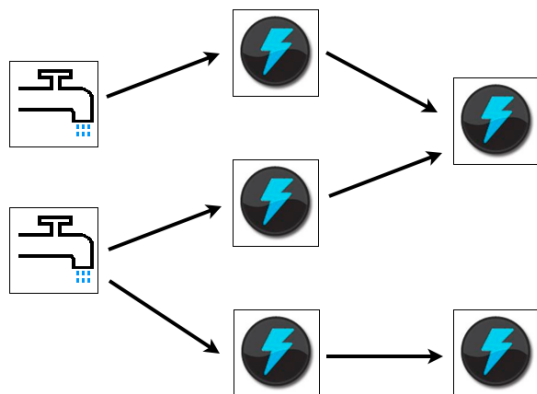
Oryx 2.x Architecture



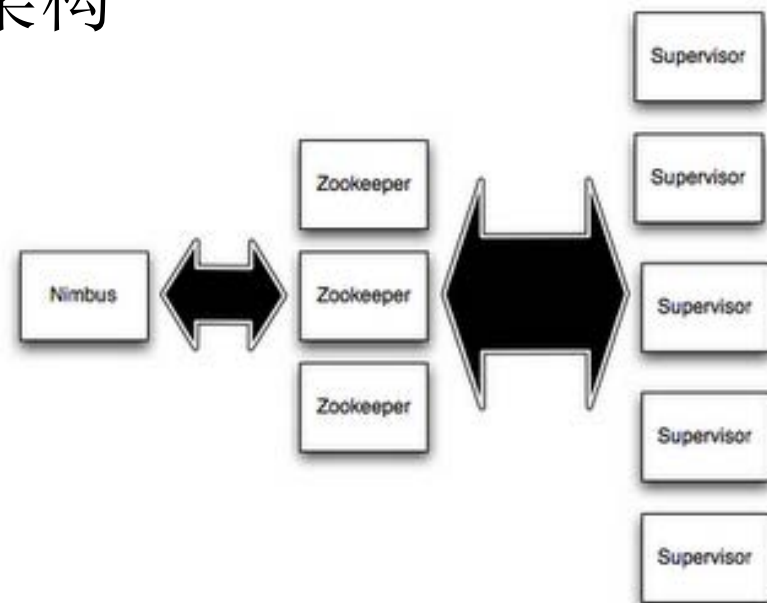
- MLlib 是Spark对常用的机器学习算法的实现库，同时包括相关的测试和数据生成器。
- MLlib 目前支持四种常见的机器学习问题：二元分类，回归，聚类以及协同过滤，同时也包括一个底层的梯度下降优化基础算法。



## 编程模型



## 架构



## 基于Storm的机器学习

- Trident-ML
- SAMOA (Scalable Advanced Massive Online Analysis)

# 参考资料

- Machine Learning is Fun! The world's easiest introduction to Machine Learning <https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471>
- Andrew Ng's free Machine Learning class on Coursera. <https://www.coursera.org/course/ml>
- 数据挖掘：实用机器学习工具和技术(第3版). Data Mining: Practical Machine Learning Tools and Techniques.
- 机器学习实战. Machine Learning in Action.
- 统计学习方法. 李航. 2012.
- Mahout实战. Mahout in Action.
- 大数据：互联网大规模数据挖掘与分布式处理. Mining of Massive Datasets.