前言

这门课程**目标**旨在帮助学生**掌握机器学习的基本概念、研究方法**，了解未来的发展趋势，为今后在该领域的深入研究打下基础。通过本课程的学习，使学生了解和掌握机器学习的基本概念、思想、方法，初步学习和掌握使用机器学习方法解决实际问题的能力，为今后进一步的学习奠定基础。

使用方法：建议配合人工智能现代方法 PPT 一起。

第一章：人工智能与机器学习引论

什么是人工智能:

wiki:**由机器展示的智能**。在计算机科学中，人工智能研究领域将自己定义为对于“智能体”的研究：**任何能够感知环境、采取行动最大化其在某个目标上成功机会**的机器。

人工智能找寻一种方法，将智能映射到机械硬件，并使一个结构进入该系统以形式化思维。

为什么需要人工智能:

应用：曲线拟合、图像分类、下棋;
人工智能是关于**数据、知识、智能**的科学。

什么是机器学习:

三大要素：T、P、E
对于**某种任务 T**、**性能度量 P**，一个及其程序被认为可以从**经验 E** 中学习指：利用经验 E，它在任务 T 上由性能度量 P 衡量的**性能提升**。
T：智能系统执行的、实现目标的工作 或 智能系统处理一个样本（对象中已量化特征）的工作。

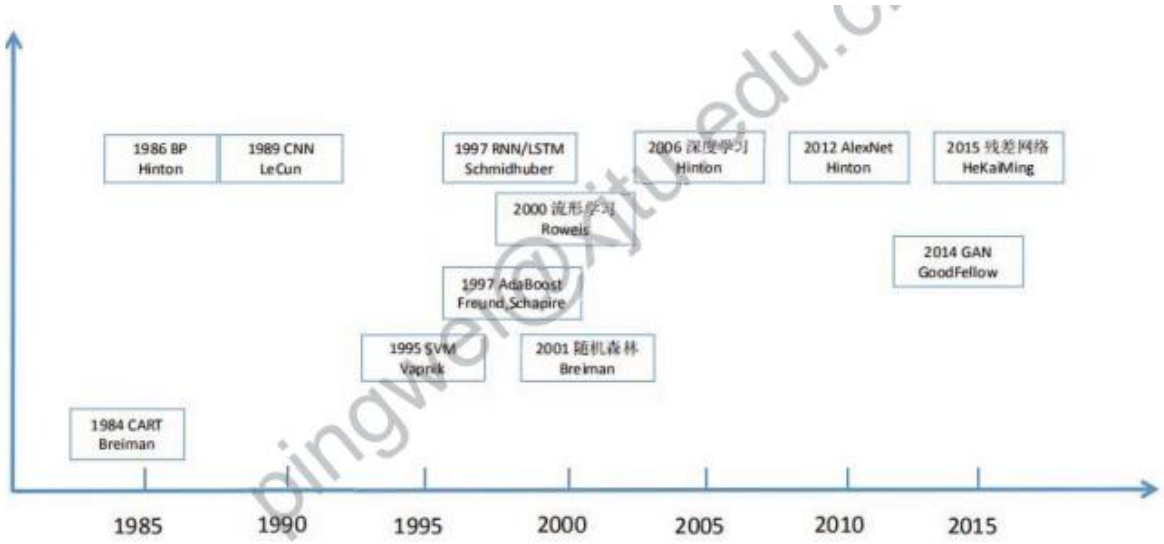
e.g.分类、输入缺失分类、回归、转录、翻译、结构输出（输出变量之间关系）、异常检测、合成和采样、缺失值填补、去噪、密度估计
P：性能度量用来描述机器学习算法能力，与任务相关，不一定能精确定义其性能度量。

e.g.分类正确率，概率估计，
E：经验是人们知识积累、经验就是数据集，数据点的集合

机器学习发展历史：

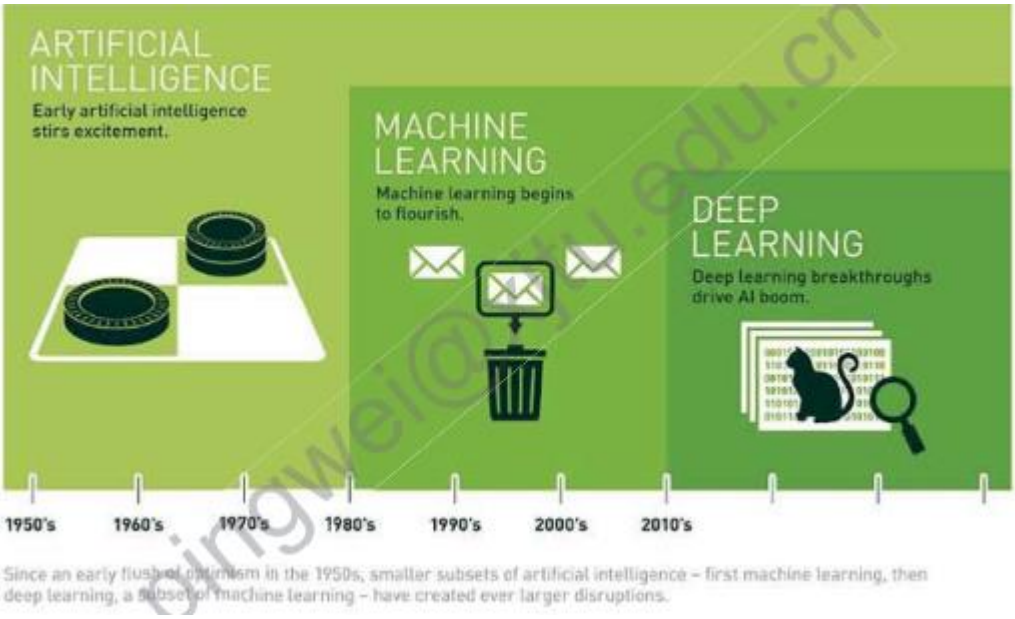
56 年达特茅斯会议-70 年推理期（自动定理证明）-90 年知识期（专家系统）-至今自动学

习期（归纳、强化、类比、示教学习）



机器学习典型应用：

图像理解：目标检测与识别、图像分割、语义描述。
视频理解：车辆识别、人体姿态估计、行为跟踪。
机器学习与机器人、机器学习与医疗、机器学习与网络、机器学习翻译、机器学习与金融。



人工智能(机器展示智能)>机器学习（数据获取知识）>深度学习（实现方式：神经网络）

第二章

矩阵论：

理解：这部分都是老生常谈的。

基本概念：

标量(点)：单独的数字

向量（线）：一组有序排列的数字，表示空间中一点

矩阵（面）：二维数组$A \in R^{m*n}$

张量（体）：一个数组中元素分布在若干维坐标的规则网络中$A \in R^{m*n*k*\dots}$

线性相关：非零线性组合后可以出现 0 向量。

线性组合：每个向量乘对应标量之后的和。$\sum_i c_i v^i$

矩阵的秩：线性无关的行/列的最大数量

生成子空间：原始向量线性组合后能抵达的点的集合。

范数：衡量向量的大小，性质：$\begin{cases} 1.\text{正定性：}\ f(x) = 0 \rightarrow x = 0 \\ 2.\text{三角不等式：}\ f(x+y) \leq f(x) + f(y) \\ 3.\text{正齐次性：}\ f(ax) = |a| f(x) \end{cases}$

L^p范数：$||x||_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$,其中$p \in R, p \geq 1$

常见范数：L^2范数、L^1范数、L^∞范数(最大范数)

概率论：

理解：这部分也是老生常谈的。

用概率原因：$\begin{cases} 1.\text{确定性是不确定性的特例} \\ 2.\text{人工智能要处理不确定量、随机量} \end{cases}$

概率：对一个或多个事件发生可能性大小的度量。

AI 领域：进行系统推理、分析系统行为。

随机变量：可以随机取不同值的变量。

分类：离散、连续

概率分布：描述随机变量或一簇随机变量在每个可能状态可能性大小。

离散->概率质量函数；连续->概率密度函数

概率质量函数（PMF）：将随机变量能取得的每个状态映射到随机变量取得状态的概率

$$p(\boldsymbol{x} = x): x \sim p(\boldsymbol{x})$$

可同时作用于多个随机变量，多个变量的概率分布被称为**联合概率分布**。

$\begin{cases} 1.\text{P 的定义域是 x 所有可能状态的集合} \\ 2.\forall x \in X, 0 \leq p(x) \leq 1 \\ 3.\sum_{x \in X} p(x) = 1 \end{cases}$

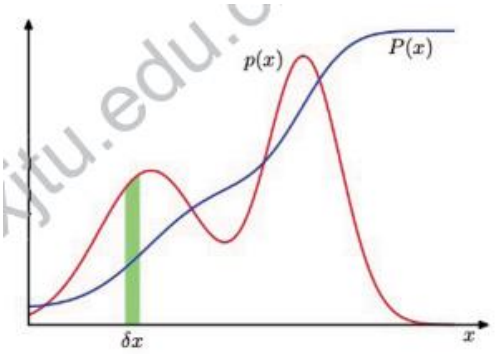
概率密度函数（PDF）：描述连续随机变量 x 的概率分布，并不是特定状态的概率，而是区域概率。

$$P(x \in [a, b]) = \int_a^b p(x) dx$$

要满足条件：$\begin{cases} 1.\text{P 的定义域是 x 所有可能状态的集合} \\ 2.\forall x \in X, 0 \le p(x), \\ 3.\int_{-\infty}^{\infty} p(x)dx = 1 \end{cases}$

累计分布函数（CDF）

$$P(x) = \int_{-\infty}^{x} p(x)dx$$



边缘概率：定义在一组变量的子集上的分布。

$$\forall x \in X, P(X = x) = \sum_{y} P(X = x, Y = y); p(x) = \int p(x, y)dy$$

条件概率：某事件在其他事件发生时出现的概率。

$$P(Y = y|X = x) = \frac{P(Y = y, X = x)}{P(X = x)}, P(X = x) > 0$$

两变量相互独立：

$$\forall x \in X, y \in Y, p(X = x, Y = y) = p(X = x)p(Y = y)$$

两变量条件独立：

$$\forall x \in X, y \in Y, z \in Z, p(X = x, Y = y|Z = z) = p(X = x|Z = z)p(Y = y|Z = z)$$

概率论基本原理：

对称原理：$p(x, y) = p(y, x)$

乘法原理：$p(x, y) = p(x|y)p(y)$

加法原理：离散变量、连续变量

期望：在概率分布$p(x)$下，函数$f(x)$的平均值被称为$f(x)$的期望。

$$E_x[f] = \sum_{x} p(x)f(x)$$

$$E_x[f] = \int p(x)f(x)dx$$

若 x,y 独立，$E[xy] = E[x]E[y]$

方差：度量了$f(x)$在均值$E_x[f]$附近变化性的大小

$$Var(f) = E[(f(x) - E[f(x)])^2] = E[f(x)^2] - E[f(x)]^2$$

协方差（相关性系数）：表示多大程度函数$f(x)$和函数$g(y)$共同变化

$$Cov(f(x),g(y)) = E[\ (f(x) - E[f(x)]) * (g(y) - E[g(y)])\ ]$$

如果 x、y 相互独立，他们协方差为 0；协方差为 0 两随机变量不相关。

独立一定不相关，不相关未必独立。

贝叶斯定理:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

$p(y)$:先验概率；$p(y|x)$：后验概率；$p(x|y)$：似然函数。

高斯分布:

$$N(x|\mu,\sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}}\exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$$

$E[x] = \int N(x|\mu,\sigma^2)xdx = \mu$

$E[x^2] = \int N(x|\mu,\sigma^2)x^2dx = \mu^2 + \sigma^2$

$var(x) = E[x^2] - E[x]^2 = \sigma^2$

多维:

$$N(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}^2) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\ \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T|\boldsymbol{\Sigma}|^{-1}(\boldsymbol{x}-\boldsymbol{\mu})|)$$

信息论：

理解：通过信息论概念（熵、交叉熵）导出误差的产生、求解方法。（主要记住 KL 散度）

研究：对一个信号包含多少信息的多少进行量化。

机器学习中被用来描述概率分布或量化概率分布之间的相似性。

基本想法：一个不太可能的事情发生要比一个非常可能的事情发生提供更多的信息。

1. **单调性**：概率越大的事件，信息量越小。
2. **非负性**：一个事件的信息大于等于 0。
3. **可加性：**多个独立事件总的信息量为各事件信息的和。

自信息（随机事件的信息量）：$h(x) = -\ln p(x)$【概率越小所含信息越大】

$$p(x,y) = p(x)p(y) \rightarrow h(x,y) = h(x) + h(y)$$

熵：描述系统的混乱程度；定义在一个概率分布之上，对于概率分布随机性进行度量，反映一组数据包含信息量的大小。

取值：对于信息量进行加权和。（**对随机变量取每个值的信息量的数学期望**）

信息熵是信息量的期望，就是平均而言发生一个事件我们得到信息量的大小:

离散型:

$$H(p) = E_x[-\ln p(x)] = -\sum_{i=1}^{n} p_i \ln p_i = \sum_{i=1}^{n} p_i \frac{1}{\ln p_i};\ p_i = p(x_i)$$

连续型:

$$H(p) = -\int_{-\infty}^{\infty} p(x) \ln p(x)\, dx$$

交叉熵：定义于两个概率分布之上，反映两个概率分布的差异程度。

交叉熵是用来衡量在给定的真实分布下，使用非真实分布所指定的策略消除系统的不确定性所需要付出的努力的大小。

机器学习时候要拟合分布，所以可以用**交叉熵构造损失函数**。

离散：

$$H(p,q) = E_p[-\ln q] = -\sum_x p(x) \ln q(x) = \sum_x p(x) \frac{1}{\ln q(x)}$$

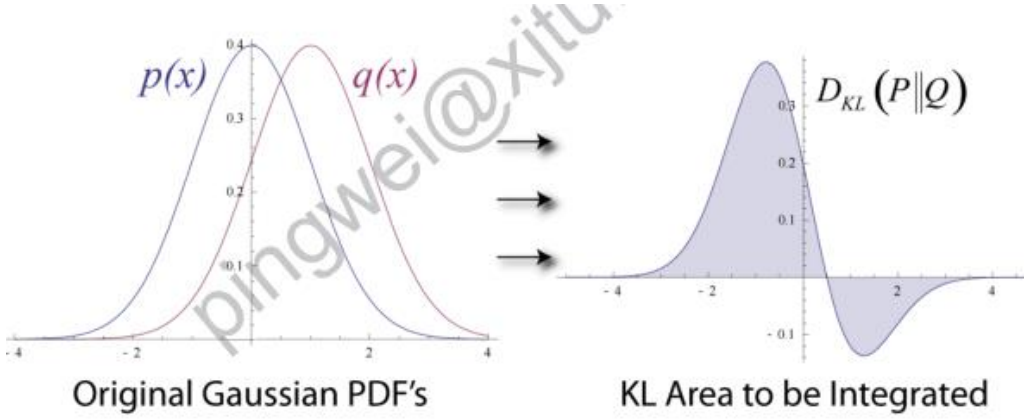
连续：

$$H(p,q) = E_p[-\ln q] = -\int_{-\infty}^{\infty} p(x) \ln q(x)\, dx$$

具有不对称性：$H(p,q) \neq H(q,p)$

kullback leibler(KL)散度/相对熵/信息增益：用于衡量两个概率分布之间的差距。

相对熵 = 某个策略的交叉熵（p->q 付出努力大小） - 信息熵(p 平均信息量)



离散：

$$D_{KL}(p||q) = \sum_x p(x) \ln \frac{p(x)}{q(x)} = \sum_x p(x) \frac{1}{\ln q(x)} - \sum_x p(x) \frac{1}{\ln p(x)};$$

连续：

$$D_{KL}(p||q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

其值越大说明两概率分布的差距越大；当两个分布完全相等时 KL 散度值为 0

具有不对称性：$D_{KL}(p||q) \neq D_{KL}(q||p)$

如果分布相同：$D_{KL}(p||q) = 0$

有非负性：$D_{KL}(p||q) > 0$; *当* $x > 0$ *时有* $\ln x \leq x - 1$

交叉熵是相对熵的一种特殊情况,即 p(x)分布是已知的,因而导致公式的后半部分为常数项。

Jensen-Shannon 散度：（对于两概率加和后做平均）

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||m) + \frac{1}{2}D_{KL}(q||m)$$

其中：$m(x) = \frac{1}{2}(p(x) + q(x))$

$m(x)$是这两个概率质量函数或概率密度函数的均值。

JS 散度描述两概率分布之间的差异，且具有对称性。

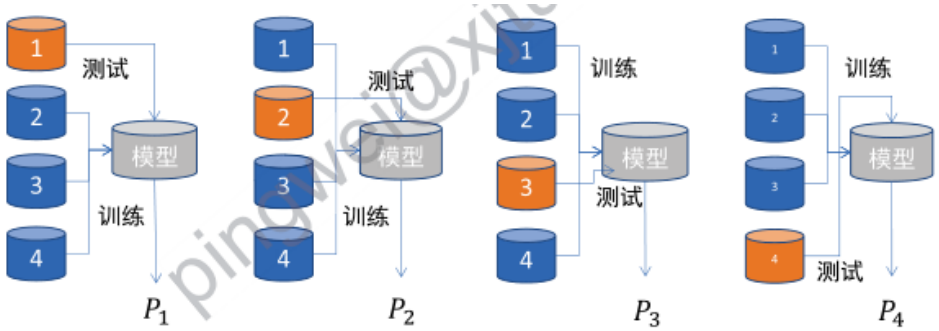
基本概念：

理解：数据、验证方式、学习方式、误差产生、误差求解（最大似然估计）；这部分了解就好。

数据集：
1. 测试集 – 测试误差
2. 验证集 – 验证误差
3. 训练集 – 训练误差

交叉验证技术（cross-validation）：

k 折(k-fold)交叉验证技术将样本均匀分成 k 份，轮流用其中 k-1 份做训练集，剩下一份做训练集，用 k 次统计的性能指标作为最后的性能指标。



$$P = \frac{P_1 + P_2 + P_3 + P_4}{4}$$

泛化能力（generalization）：一个算法在之前未观测数据上执行某种任务的能力。

最终目标：在测试集上有更小的误差。

独立同分布假设：理论上独立同分布训练集、测试集有相同误差；

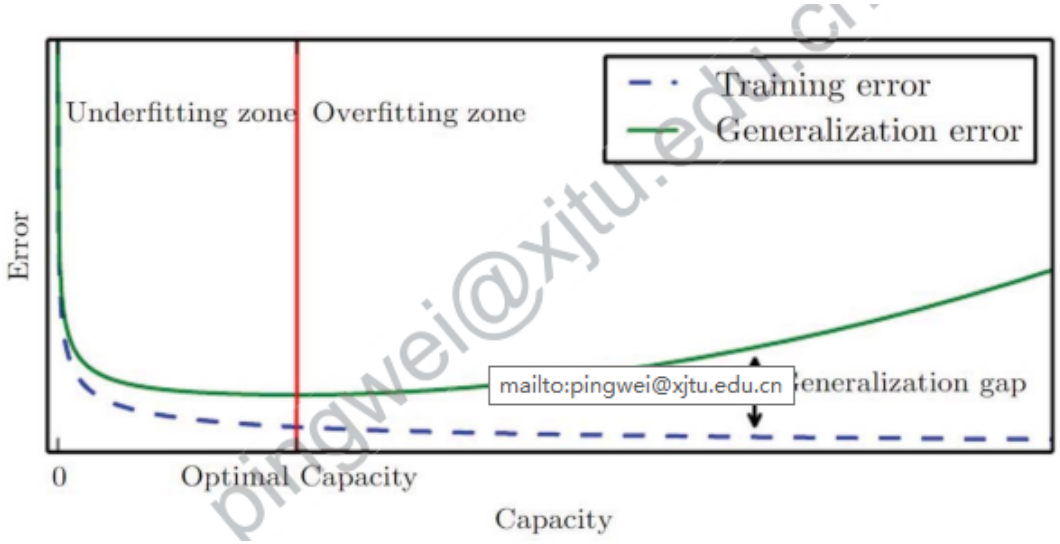
现实中**测试集误差>训练集误差**。

好坏判断：
1. 训练误差小。
2. 训练误差与测试误差之间差距小。

模型容量（Capacity）：拟合函数的能力

1. 容量低的模型很难拟合训练集
2. 容量高的模型记住不适用于测试集的训练集的性质。

过拟合、欠拟合（模型好坏）：



过拟合原因：
1. 模型过于复杂，拟合噪声了；需要进行剪枝
2. 训练样本少、缺乏代表性、增加样本数
3. 训练样本噪声干扰

欠拟合原因：模型简单，特征数太少无法确定正确的映射关系。

正则化(Regularization)：**防止过拟合**可以给损失函数加一个惩罚项，对复杂模型进行惩罚，让模型参数值尽可能小，使模型更简单，提高泛化性。

L^2正则化：L^2范数的平方：$J(\boldsymbol{\Theta}) = L(\boldsymbol{\Theta}; X, Y) + \frac{\lambda}{2}\boldsymbol{\Theta}^T\boldsymbol{\Theta}$

有/无/弱监督学习：
1. 训练数据包含样本数据与标记或目标为有监督学习
2. 训练数据只包含样本数据而无标记或目标为无监督学习

有监督学习：样本带标签、模型实现特征向量到标签值的映射$y = f(\boldsymbol{x})$

监督信号：
1. 分类问题 – 标签值为整数编号
2. 回归问题 – 标签值为实数

有监督机器学习模型：
1. **生成模型**：对 $p(\mathbf{x}, y)$或 $p(\mathbf{x}|y)$进行建模
即对样本的特征向量服从某种概率分布建模
2. **判别模型**：对后验概率 $p(y|\mathbf{x})$进行建模
或直接预测 $y = f(x)$

类别	算法
生成模型	贝叶斯分类器
	高斯混合模型
	贝叶斯网络
	隐马尔科夫模型
	受限玻尔兹曼机
	生成对抗网络
	变分自动编码器
判别模型	决策树
	KNN算法
	人工神经网络
	支持向量机
	Logistic回归
	Softmax回归
	随机森林
	Boosting算法
	条件随机场

判别式模型举例：要确定一个羊是山羊还是绵羊，用判别模型的方法是从历史数据中学习到模型，然后通过提取这只羊的特征来预测出这只羊是山羊的概率，是绵羊的概率

生成式模型举例：利用生成模型是根据山羊的特征首先学习出一个山羊的模型，然后根据绵羊的特征学习出一个绵羊的模型，然后从这只羊中提取特征，放到山羊模型中看概率是多少，在放到绵羊模型中看概率是多少，哪个大就是哪个。

而判定式模型的世界是这个样子：

	y = 0	y = 1
x = 0	1	0
x = 1	1/2	1/2

$$\sum_y P(y|x) = 1$$

生成式模型的世界是这个样子：

	y = 0	y = 1
x = 0	1/2	0
x = 1	1/4	1/4

$$\Sigma P(x, y) = 1$$

无监督学习：样本没有标签值，没有训练过程，机器学习算法直接对样本进行处理，得到某种结果。

算法细分：
- 聚类问题 – 无监督分类，将一组样本划分成多个子集。
- 数据降维问题 – 将向量映射到更低维的空间。

弱监督学习：数据标注成本过高，很多任务难以获得全部真值标签，无监督学习困难。用较弱的监督信息来构建预测模型

分类：
1. **不完全监督**：训练集数据中只有**一部分**给了标签
2. **不确切监督**：训练数据只给出了**粗粒度**标签
3. **不精确监督**：给出标签**不总是正确**的

参数估计理解：参数估计首先假定出模型符合某种概率分布，再计算θ为何值时可以最大化整个概率，即通过改变θ来最大化概率。

最大似然估计（MLE）：模型学习中参数估计一般准则的最常用的一种。

频率学派：他们认为模型参数是个定值（世界是确定的），希望通过类似解方程组的方式从

数据中求得该未知数。（通常用于判别模型）

m 个独立同分布样本： $\boldsymbol{X} = (\boldsymbol{x}^1, \dots \boldsymbol{x}^m)$；分布：$p(\boldsymbol{x}, \boldsymbol{\theta})$,$\boldsymbol{\theta}$的最大似然估计：

$$\boldsymbol{\theta_{ML}} = argmax_\theta\, p(\boldsymbol{X}; \boldsymbol{\theta}) = argmax_\theta\, \Pi_{i=1}^m p\left(\boldsymbol{x^i}; \boldsymbol{\theta}\right) = argmax_\theta \sum_{i=1}^m \log p\left(\boldsymbol{x^i}; \boldsymbol{\theta}\right)$$

$$= argmax_\theta\, E_{x\sim\hat{p}_{data}} \log p(\boldsymbol{x}; \boldsymbol{\theta}) = argmin_\theta - E_{x\sim\hat{p}_{data}} \log p(\boldsymbol{x}; \boldsymbol{\theta})$$

上一步的加和，本质上是对于符合x分布的$x^{(i)}$的加和，即x概率密度越大的地方$x^{(i)}$出现次数越多，出现频率越高，对应着就是在求期望。

换一种说法，对于x进行随机采样得到的$x^{(i)}$符合x自己的分布，所以，最大化采样后所有$\boldsymbol{log}\ p(x^i; \theta)$值加和等价于最大化期望。

$D_{KL}(\hat{p}_{data}||p) = E_{x\sim\hat{p}_{data}}[\log \hat{p}_{data}(\boldsymbol{x}) - \log p(\boldsymbol{x})]$**最小化 KL 散度**

条件最大似然估计：

估计条件分布$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$

$$\boldsymbol{\theta_{ML}} = \arg\max_\theta p(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}) = \arg\max_\theta \sum_{i=1}^m \log p(\boldsymbol{y^i}|\boldsymbol{x^i}; \boldsymbol{\theta})$$

最大后验估计(Maximun A Posteriori)：根据经验数据获得对难以观察的量的估计。

贝叶斯学派：他们认为模型参数源自某种潜在分布（世界是不确定的），希望从数据中推知该分布。对于数据的观测方式不同或者假设不同，那么推知的该参数也会因此而存在差异。（通常用于生成模型）

$$\boldsymbol{\theta_{MAP}} = \arg\max_\theta p(\boldsymbol{\theta}|\boldsymbol{X}) = \arg\max_\theta p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = \arg\max_\theta(\log p(\boldsymbol{X}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}))$$

e.g.对于 N 个输入数据$x = (x_1, \dots, x_N)$其对应$t = (t_1, \dots, t_N)$,求$y(x, \boldsymbol{\omega}) = \sum_{j=0}^M \omega_j x^j$

首先确定数据分布

1.**假设数据分布符合高斯函数**：$p(t|x, \boldsymbol{\omega}, \sigma) = N(t|y(x, \boldsymbol{\omega}), \sigma^2)$

2.似然函数：$p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{\omega}, \sigma) = \Pi_{n=1}^N N(t_n|y(x_n, \boldsymbol{\omega}), \sigma^2)$

套用极大似然估计求解

3.条件最大似然估计：$(\boldsymbol{\omega}, \sigma)_{ML} = \arg\max_{\boldsymbol{\omega},\sigma} \sum_{n=1}^N \log p(t_n|x_n, \boldsymbol{\omega}, \sigma)$

$$\rightarrow \arg\max_{\omega,\sigma} \sum_{n=1}^N \log p(t_n|x_n, \boldsymbol{\omega}, \sigma) = \arg\max_{\omega,\sigma} \sum_{n=1}^N \log \sqrt{\frac{1}{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}\left(x_n - y(x_n, \boldsymbol{\omega})\right)^2)$$

$$= \arg\max_{\omega,\sigma} -\frac{N}{2}\log\sigma^2 - \frac{N}{2}\log 2\pi - \sum_{n=1}^N \frac{(y(x_n, \boldsymbol{\omega}) - t_n)^2}{2\sigma^2} = \arg\max_{\omega,\sigma}(y(x_n, \boldsymbol{\omega}) - t_n)^2$$

对于$\boldsymbol{\omega}$求导，得到最终结果；其等价于$M \times \boldsymbol{\omega} = \boldsymbol{Y} \rightarrow \boldsymbol{M^T} \times \boldsymbol{M} \times \boldsymbol{\omega} = \boldsymbol{M^T} \times \boldsymbol{Y}; \boldsymbol{M_{ij}} = \boldsymbol{x_i^j}$

第三章：概率图模型与马尔科夫随机场

一.概率图模型

理解：概率图模型之前讲过，和贝叶斯网络、马尔科夫随机场的上层概念。用图的方式表示联合概率。

1. 概率是对一个或多个事件发生可能性大小的度量
2. 概率图表达了多个变量的联合概率分布

1. 加法定理：$\mathrm{p(x)} = \sum_y p(x,y)$
2. 乘法定理：$\mathrm{p(x,y)} = \mathrm{p(y|x)p(x)}$

概率图基本概念：

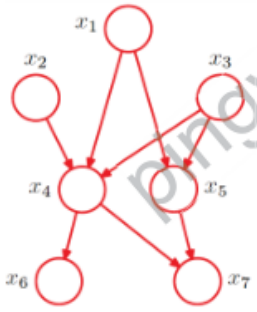
1. 概率图包含节点和边，**每个节点表示一个(组)随机变量，边表示变量之间的概率关系**
2. 概率图以可视化的方式表达了多个随机变量之间的依赖关系，每个概率图是图中**所有变量联合分布**
3. 概率图分为有向图和无向图。**无向图中边无方向，也称作马尔科夫随机场**。

有向图/贝叶斯网络：有向图中边有明确的方向。

有向图所有变量的联合分布：（每个节点以其所有父节点为条件概率分布乘积）

$$p(X) = \Pi_{k=1}^{K} p(x_k|par(x_k))$$

e.g.



$$p(x_1,\dots,x_7) =$$
$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)$$
$$\cdot\, p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$

马尔科夫随机场：无向图中的边没有方向，连接两个节点。

团：无向图中节点的子集，任意两点都有边相连。

极大团：一个团再增加图中任何一个其他节点都不成为团。

无向图的联合概率分布：（可以写作图的最大团块的势函数的乘积）

$$p(\boldsymbol{x}) = \frac{1}{Z}\Pi_C \psi_C(\boldsymbol{x}_C);\ C\text{: 团};\ \mathbf{x}_C\text{: 团中变量集合};\ \psi_C(\boldsymbol{x}_C)\text{: 势函数}$$

$$Z\text{: 分割函数，确保}\psi_C(\boldsymbol{x}_C) \geq 0 \text{ 时} p(\boldsymbol{x}) \geq 0;\rightarrow Z = \sum_x \Pi_C \psi_C(\boldsymbol{x}_C)$$

二.马尔科夫随机场

理解：介绍随机场的基本概念，包括其包含的位点空间S、相空间Λ、以及由Λ, S构成的随机场，以及每一个位点构成的近邻系统，以及马尔科夫随机场的（局部）特征。

随机场基本概念：

1. **位点空间**：有限元集合 $S=\{s_1,s_2,\dots,s_m\}$表示位点(sites)的集合
   e.g. 时间集合、空间坐标集合。
2. **相空间**：有限元素集合 $\Lambda=\{\lambda_1,\dots,\lambda_n\}$, 随机变量的取值空间
   e.g. 颜色、类别、几何属性
3. **随机场：**位置空间中每一个位点从相空间 Λ 中取值的随机变量集合 $\boldsymbol{X}=\{x(s)|s\in S,x(s)\in\Lambda\}$
4. **组合空间：** $\Lambda^S=\Lambda_1\times\dots\times\Lambda_m$表示所有位点 S 的取值空间

随机场的一般性质：

给定一个组合变量 x 和一个子集$\boldsymbol{A}\in\boldsymbol{S}$，我们将部分组$\boldsymbol{x}(\boldsymbol{A})$定义为：

$$\boldsymbol{x}(\boldsymbol{A})=\{x(s)|s\in A,x(s)\in\Lambda\}$$

如果我们用$\boldsymbol{S}-\boldsymbol{A}$表示 A 在 S 中的补集，则

$$\boldsymbol{x}=[\boldsymbol{x}(A),\boldsymbol{x}(S-A)]$$

对任意位置$\boldsymbol{s}\in\boldsymbol{S}$

$$\boldsymbol{x}=[\boldsymbol{x}(s),\boldsymbol{x}(S-s)]$$

马尔科夫随机场：是马尔科夫链的自然延伸，用来模拟局部结构或随机变量间的相互作用。

局部结构/相互作用：一般可用一个无向图$G(S,E)$来定义，S，E 分别为节点空间和边缘集。

近邻系统：一个点的邻域定义为与此位点距离小于一个阈值的位点的集合。

$$N_s=\{s'\in S|dist(s,s')\le a,s'\neq s\}$$

性质：

1. 一个位点不属于自己的邻域 $s\notin N_s$
2. 相邻关系是相互的，$s\in N_s\leftrightarrow s'\in N_s$

马尔科夫随机场定义：

$$\forall s\in S,P\left(x(s)\middle|\boldsymbol{x}(S-s)\right)=P(x(s)|\boldsymbol{x}(N_s))$$

位点 s 的马尔科夫局部特性：

$$P^{(s)}(\boldsymbol{x}):P\left(x(s)\middle|\boldsymbol{x}(S-s)\right)=P(x(s)|\boldsymbol{x}(N_s))$$

要素：

1. 位点空间: $S=\{s_1,s_2,\dots,s_m\}$
2. 相空间: $\Lambda=\{\lambda_1,\dots,\lambda_n\}$
3. 定义在 S 上的近邻系统: $N=\{N_s\}$
4. 局部特性: $\{P^{(s)}\}_{s\in S}$
5. 概率表达

三.吉布斯分布

模拟粒子之间的物理作用，引入团、势的概念来具体化局部组合以及相互作用。

要素：

1.团：任意单位点$\{s\}$都是一个团。元素数目大于 1 的子集$C\in S$，若其中任意两个不同位点都是近邻，则子集 C 是一个团。所有团的集合用 **C** 表示。

2.势：在Λ^S上的吉布斯势，是函数$V_C:\Lambda^S\to R$的集合$\{V_C\,|C\in C\}$,并且函数满足：

a.如果C不是团，则$V_c=0$

b.对所有$\boldsymbol{x},\boldsymbol{x}'\in\Lambda^S$，以及所有$C\in\boldsymbol{C}$,有$\boldsymbol{x}(C)=\boldsymbol{x}'(C)\to V_c(\boldsymbol{x})=V_c(\boldsymbol{x}')$

3.能量函数：$U:\Lambda^S\to R$定义为：

$$U(\boldsymbol{x}) = \sum_{C \in \boldsymbol{C}} V_C(\boldsymbol{x})$$

在$(\boldsymbol{S}, \boldsymbol{C})$上的吉布斯分布定义为:

$$p(\boldsymbol{x}) = \frac{1}{Z_T} e^{-\frac{1}{T}U(\boldsymbol{x})}; T > 0 \text{ 被称为温度,} \mathrm{Z_T}\text{是归一化常数，称为配分函数}$$

吉布斯分布是组合空间Λ^S上取值范围为$[0,1]$的概率预测量。它给与能量较低的组合$\boldsymbol{x} \in \Lambda^s$较高的概率。

定义近邻域和团是为了捕捉到目标系统中**元素间的局部作用关系**，定义吉布斯分布是为了对局部作用进行量化。

吉布斯与马尔科夫等价性：一个定义在(S, Λ, N)上的马尔科夫随机场的联合概率分布$p(\boldsymbol{x})$是一个吉布斯分布，即

$$p(\boldsymbol{x}) = \frac{1}{Z_T} e^{-\frac{1}{T}U(x)}$$

能量函数定义为:

$$U(\boldsymbol{x}) = \sum_{s \in C_1} V_1(x_s) + \sum_{\{s,s'\} \in C_2} V_2(x_s, x_{s'}) + \sum_{\{s,s',s''\} \in C_2} V_2(x_s, x_{s'}, x_{s''}) + \cdots$$

噪声图像　　还原图像

$$U(\mathbf{x}) = \sum_{i \in C_1} V_1(x_i) + \sum_{\{i,i'\} \in C_2} V_2(x_i, x_{i'})$$

$$= \sum_{i \in S} (x_i - y_i)^2 / (2\sigma^2) + \sum_{i \in S} \sum_{i' \in N_i} g(x_i - x_{i'})$$

$$g(\eta) = \eta^2$$

第四章：隐马尔可夫模型

这部分因为之前讲过好多次就没怎么写笔记

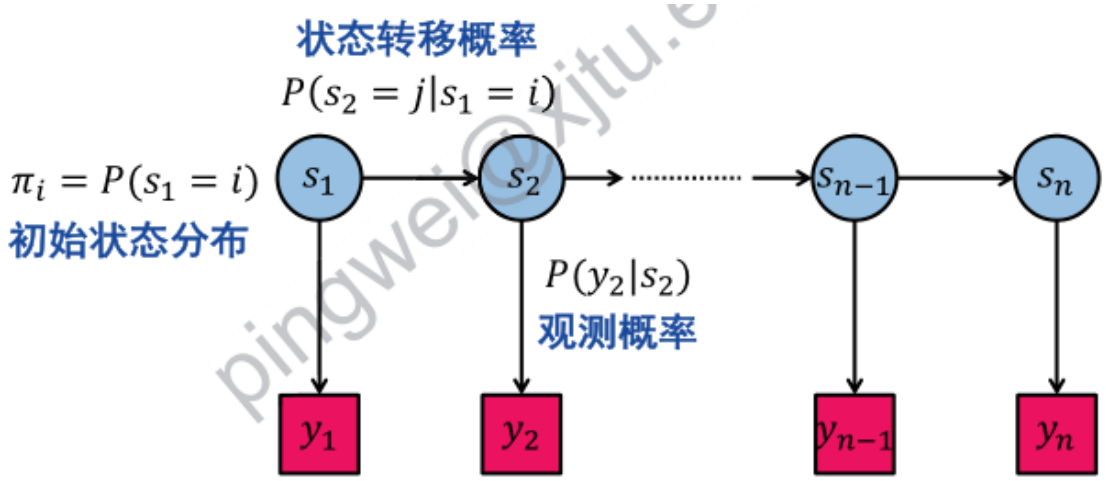
隐马尔科夫模型

隐马尔可夫模型的自由度是通过让马尔科夫链的状态生成可观测数据，并将状态序列隐藏起来使其无法观测到。

马尔科夫链每一个状态都要对应于一个可观测的物理事件。

HMM 组成:

1. **状态空间S**：HMM 状态空间通常是可数的。每个状态是隐藏的
2. **输出集Y**：元素$y_i \in Y$对应系统的一个物理输出。
3. **状态转移矩阵P** $= \{p_{ij}\}, p_{ij} = P(j|i), i, j \in S$。
4. **输出概率分布矩阵（传感器模型）**：$Q = \{q_{sy}\}$,其中$q_{sy} = P(y|s)$是状态s下输出 y 的概率。
5. **初始状态分布**：$\pi = \{\pi_i\}, i \in S$。



$$P(\boldsymbol{s},\boldsymbol{y}) = P(s_1, s_2, \dots, s_n, y_1, y_2, \dots, y_n) = P(s_1)\Pi_{i=1}^{n} P(y_i|s_i)\Pi_{i=1}^{n} P(s_i|s_{i-1})$$
$$= \pi_{s1} q_{s1y1} \Pi_{i=2}^{n} p_{si-1si} q_{siyi}$$

HMM 完整参数集：$\lambda = (P, Q, \pi)$

一.如何算出 P(y| λ)：前向算法

评估问题，目的在计算给定模型生成某观测输出序列的概率。(条件独立性)

$$P(y_1, y_2, \dots, y_t, s_t) = P(y_1, y_2, \dots, y_t|s_t)P(s_t)$$
$$= P(y_t|s_t)P(y_1, y_2, \dots, y_{t-1}|s_t)P(s_t) = P(y_t|s_t)P(y_1, y_2, \dots, y_{t-1}, s_t)$$
$$= p(y_t|s_t)\sum_{s_{t-1}} P(y_1, y_2, \dots, y_{t-1}, s_{t-1})P(s_t|s_{t-1})$$

初始化：$a_1(i) = \pi_i q_{iy_1}$

递归：$a_t(j) = \left[\sum_{i=1}^{|s|} a_{i-1}(i)p_{ij}\right] q_{jy_t}, 2 \le t \le n, 1 \le j \le |S|$

算法复杂度$O(n|S|^2)$

二.如何才能计算出最可能生成 y 序列的 s 状态序列

识别给定模型隐藏部分，找出最有可能生成观测输出序列的状态序列。

最大化$P(s|y)$等价于最大化$P(s_1, s_2, \dots, s_n, y_1, y_2, \dots, y_n)$

$$= P(s_1, \dots, s_t, y_1, \dots, y_t)P(s_{t+1}, \dots, s_n, y_{t+1}, .., y_n|s_t)$$

■ Viterbi 算法：$\gamma_t(i)=\max_{s_1,s_2,\mathrm{K},s_{t-1}} P(s_1,\mathrm{K},s_{t-1},s_t=i,y_1,\mathrm{K},y_t)$

1. 初始化：$\gamma_1(i)=\pi_i q_{iy_1},\quad A(1,i)=0,\quad 1\le i\le|S|$
2. 递归：$\gamma_t(j)=\left(\max_i \gamma_{t-1}(i)p_{ij}\right)q_{jy_t}$
3. 终止：$A(t,j)=\arg\max_i\left[\gamma_{t-1}(i)p_{ij}\right],\ 2\le t\le n,1\le j\le|S|$
4. 回溯路径（Path backtracking）：

$$\max_{\mathbf{s}} P(\mathbf{s},\mathbf{y})=\max_i \gamma_n(i),\ \mathbf{s}^*=\arg\max_i \gamma_n(i)$$

$$s_t^*=A(t+1,s_{t+1}^*),\quad t=n-1,\ n-2,\mathrm{L}\ ,1$$

三.如何找到最优模型参数集λ，使得 P(y| λ)最大

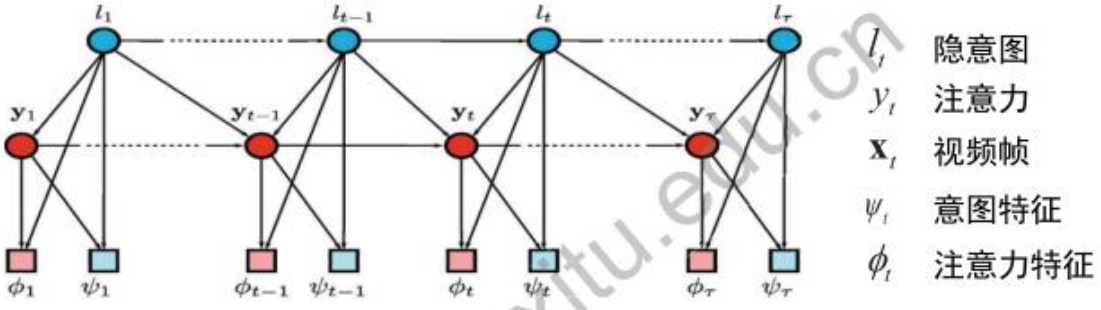
EM 算法-Expection Maximization

标准的 EM 算法常常开始于初始参数值θ',通过交替重复以下两个步骤直到$P_{\theta'}(\mathbf{Y})$的值停止攀升。

1.求期望值（Expection）：求得随机变量$\log P_\theta(X,Y)$关于旧分布$P_{\theta'}(X|Y)$的期望：

$$\sum_x P_{\theta'}(x|y)\log P_\theta(X,Y)$$

2.最大化：将该期望值看做是参数值θ的函数最大化这一期望。



$$p(\mathbf{X},\mathbf{l},\mathbf{Y})=$$

$$\underbrace{\mathrm{p}(\mathrm{l}_1)\,\mathrm{p}(\mathbf{y}_1\,|\,\mathrm{l}_1)}_{\text{初始分布}}\underbrace{\prod_{t=1}^{\tau}p(\varphi(\mathbf{x}_t)\,|\,\mathrm{l}_t,\mathbf{y}_t)}_{\text{意图匹配}}\underbrace{\prod_{t=2}^{\tau}p(\mathrm{l}_t\,|\,\mathrm{l}_{t-1})}_{\text{意图转移}}\underbrace{\prod_{t=1}^{\tau}p(\phi(\mathbf{x}_t)\,|\,\mathbf{y}_t,\mathrm{l}_t)}_{\text{注意力回归}}\underbrace{\prod_{t=2}^{\tau}p(\mathbf{y}_t\,|\,\mathbf{y}_{t-1},\mathrm{l}_t,\mathrm{l}_{t-1})}_{\text{注意力转移}}$$

$\mathbf{X}=(\mathbf{x}_1,\ldots,\mathbf{x}_\tau)$ 输入视频数据

$\mathbf{Y}=(\mathbf{y}_1,\ldots,\mathbf{y}_\tau)$ 注意力序列

$\mathbf{l}=(l_1,\ldots,l_\tau)$ 隐意图序列

第五章：深度前馈网络

理解：介绍网络构成，激活函数，代价函数，输出单元，架构

人工神经网络

1. **结构**：网络中的变量和它们的拓扑关系
2. **激励函数**：神经元如何根据其他神经元的活动改变自己的激励值
3. **学习规则：**网络中权重如何随着时间推进而调整

前馈神经网络：

1. 机器学习任务可以表达为一个广义映射 $y = f(x;\theta)$
2. $y = f(x;\theta)$可以是线性模型也可以是非线性模型
3. ϕ 是一个非线性函数，非线性模型可看做线性模型作用于输入 x 的非线性变化。
$\phi(x)$称为输入 x 的特征。

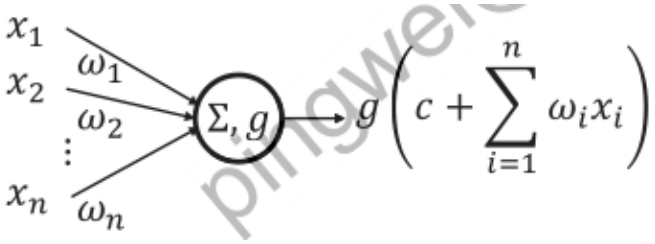
构建ϕ的三种方式：
1. 通用映射，将 x 变换至无限维空间 $\phi(x)$
2. 手动设计 ϕ, 以人的经验选择特征，如边缘、HOG、SIFT
3. 自主学习 ϕ, 从数据中挖掘和学习实现某一任务最佳 $\phi(x)$
深度学习、神经网络属于此类。

前馈神经网络：也叫多层感知机，是一种在模型输入与模型本身没有反馈连接的神经网络。

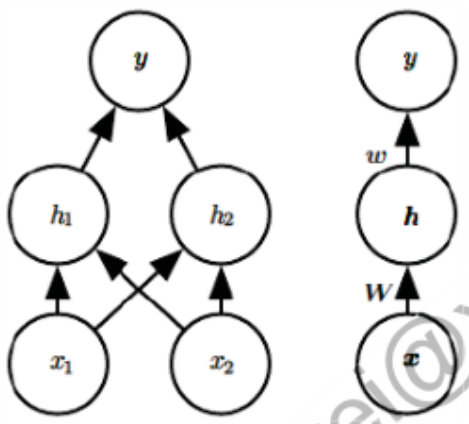
定义了一个函数$y = f(x;\theta)$来近似某个目标函数f^*;它学习参数θ的值，使其尽可能接近f^*。

前馈神经网络用多个不同函数复合在一起表示$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$

普遍近似原理：一个前馈神经网络如果具有线性输入层和至少一层具有任何一种"挤压"性质的激活函数的隐藏层，只要给予网络足够数量的隐藏单元，它可以**以任意精度来近似任何从一个有限维空间到另一个有限维空间的 Borel 可测函数**。



单个神经元计算功能：
1. 求和 Σ
2. 激励 g



$$\boldsymbol{h} = f^{(1)}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c})$$

$$\boldsymbol{h} = g(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c})$$

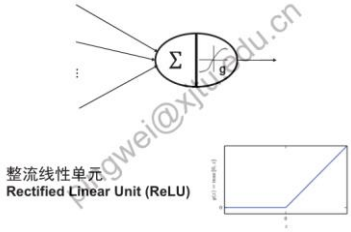
$$h_i = g(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}_{:,i} + c_i)$$

$$y = f^{(2)}(\boldsymbol{h}; \omega, b)$$

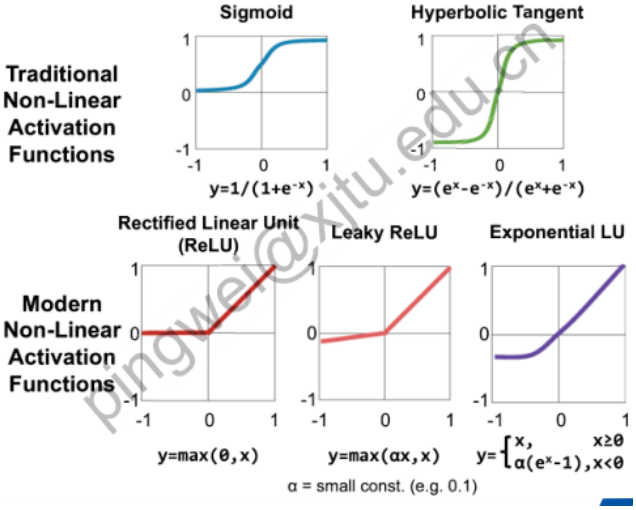
$\boldsymbol{h}$：隐层变量　　$\boldsymbol{W}$：权重矩阵　　$\boldsymbol{c}$：偏移　　$g(\cdot)$：非线性激励函数

$$\boldsymbol{h}^{(1)} = \boldsymbol{g}^{(1)}(\boldsymbol{W}^{(1)\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}^{(1)}) \qquad \boldsymbol{h}^{(2)} = \boldsymbol{g}^{(2)}(\boldsymbol{W}^{(2)\mathrm{T}}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)})$$

激活函数：



常见的激活函数:



代价函数：

代价函数：定义了神经网络所要达到的任务目标，**反应人们使用神经网络的目的意图**。
大多数应用模型可以抽象为一个条件分布$p(y|x;\theta)$,**由最大似然估计得到**。

$$\boldsymbol{J}(\boldsymbol{\theta}) = -E_{x,y\sim\hat{p}_{data}} \log p_{model}(\boldsymbol{y}|\boldsymbol{x})$$

2 范数代价函数：

$$f^* = \arg\min_f E_{\boldsymbol{x},\boldsymbol{y}\sim p_{data}} ||\boldsymbol{y} - f(\boldsymbol{x})||^2$$

$$f^* = E_{\boldsymbol{y}\sim P_{data}(\boldsymbol{y}|x)}\boldsymbol{y}$$

1 范数代价函数-平均绝对误差：

$$f^* = \arg\min_f E_{\boldsymbol{x},\boldsymbol{y}\sim p_{data}} ||\boldsymbol{y} - f(\boldsymbol{x})||^1$$

输出单元：

除了输出层的神经网络是一个函数$\boldsymbol{h} = F(\boldsymbol{x};\boldsymbol{\theta}), \boldsymbol{h}$称作特征，输出单元处理从特征到最终结果输出。

线性单元——回归问题：

$$\hat{\boldsymbol{y}} = \boldsymbol{W}^T\boldsymbol{h} + b$$

$$p(\boldsymbol{y}|\boldsymbol{x}) = N(\boldsymbol{y}:\hat{\boldsymbol{y}}, \boldsymbol{I})$$

Sigmoid 单元——二分类问题：

$$p(\boldsymbol{y} = 1|\boldsymbol{x})$$

$$\hat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{w}^T\boldsymbol{h} + \boldsymbol{b})$$

Softmax 单元——多分类问题：

$$\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{h} + \boldsymbol{b}$$

$$softmax(\boldsymbol{z})_i = \frac{\exp(\boldsymbol{z}_i)}{\sum \exp(\boldsymbol{z}_i)}$$

架构设计：
1. 神经网络层数
2. 神经网络连接
3. 神经网络每层包含单元

普遍近似原理
1. 单元数需要非常多才能精确拟合
2. 参数过多无法正确优化和学习
3. 没有足够数据支持学习——过拟合、泛化能力差

第六章：卷积神经网络

简介：

理解：卷积神经网络引入卷及操作，可以对于局部数据提取特征（局部感知），并且可以对同一数据进行多种卷积得到不同特征（参数共享），最后对于得到的特征在不改变特征情况下进行降维（池化）

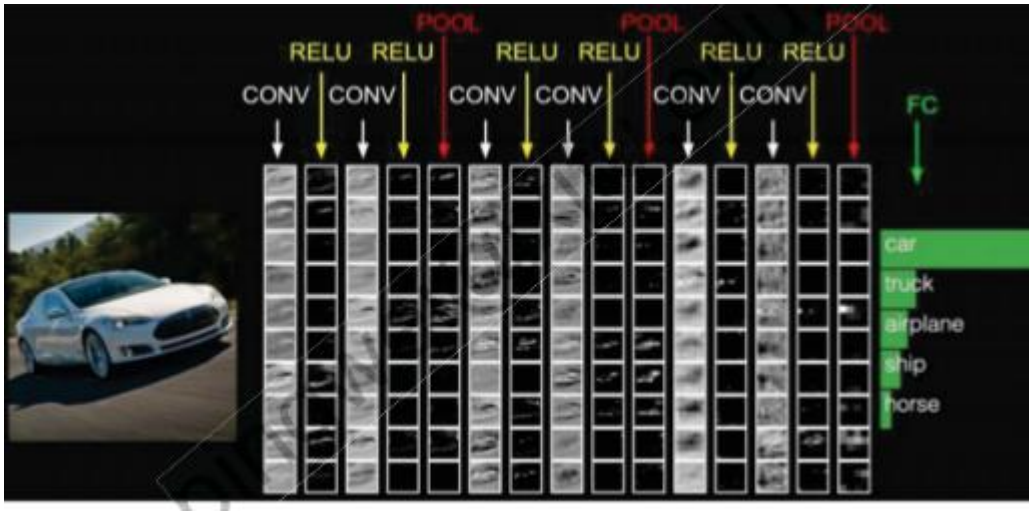
‘端到端’：

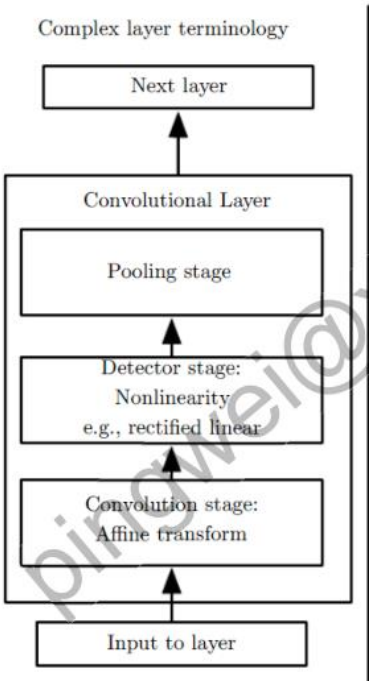
卷积神经网络通过卷积操作、池化操作、非线性激活函数映射等一系列操作的层层堆叠，将高层语义信息逐层由原始数据输入层中提取出来，逐层抽象，知道完成目标任务。

定义：卷积神经网络是一种前馈神经网络，由一个或多个卷积层和顶端的全连通层组成，同时也包括关联权重和池化层，对于大型图片有出色表现。

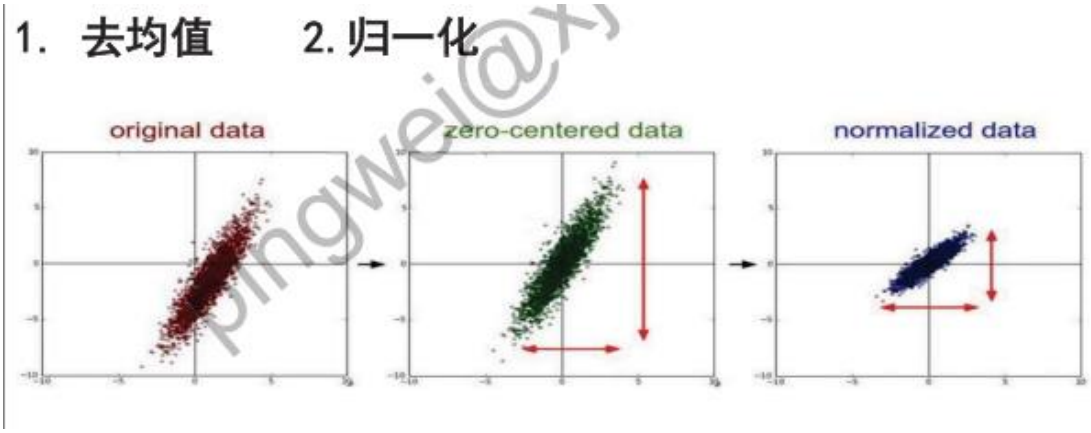
特点：
1. **局部感知**
2. **参数共享**
3. **池化**

结构：图片->输入层->卷积层（Conv+Relu）->池化层->全连接层->输出层。



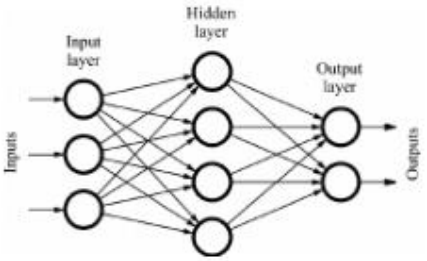


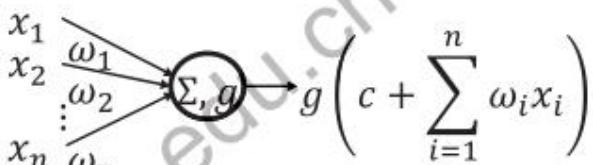
输入层操作：去均值、归一化；



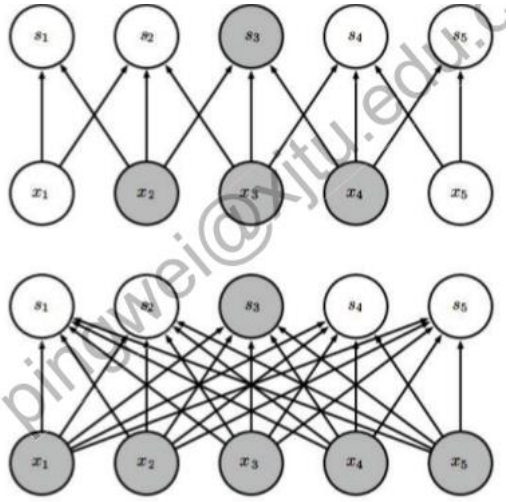
卷积单元：局部感知域、参数共享。

局部感知域表示：激励和求和





参数共享:



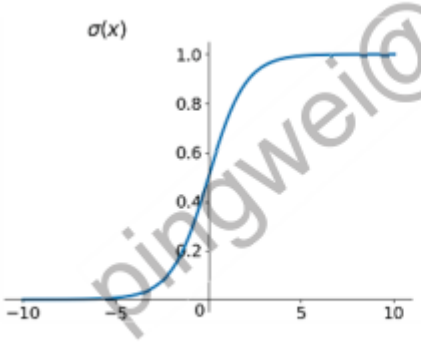
表示每一个节点都参与$s_1, \ldots, s_5$的参数计算中

卷积单元：激活函数

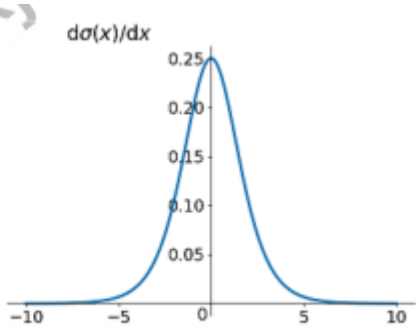
激活函数层又称非线性映射层，**激活函数的引入为了增加整个网络的表达能力**（非线性）

Sigmoid 型函数:

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



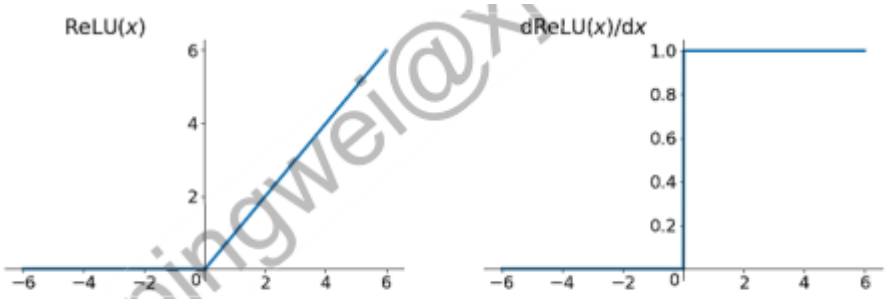
Sigmoid型函数



Sigmoid型函数梯度

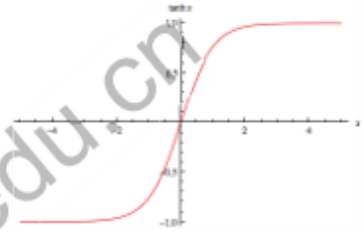
ReLU 函数:

$$Relu(x) = \max(0, x) = \begin{cases} x \; if \; x \geq 0 \\ 0 \; if \; x < 0 \end{cases}$$



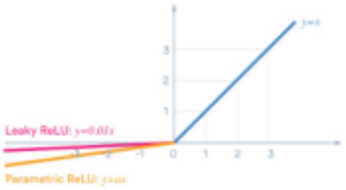
Relu型函数　　Relu型函数梯度

其它激活函数：



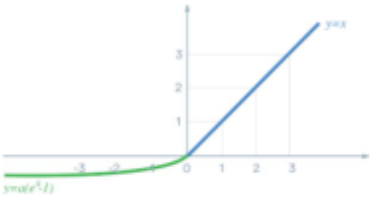
tanh(x)型函数

$$\tanh(x) = 2\sigma(2x) - 1$$



Leaky Relu型函数

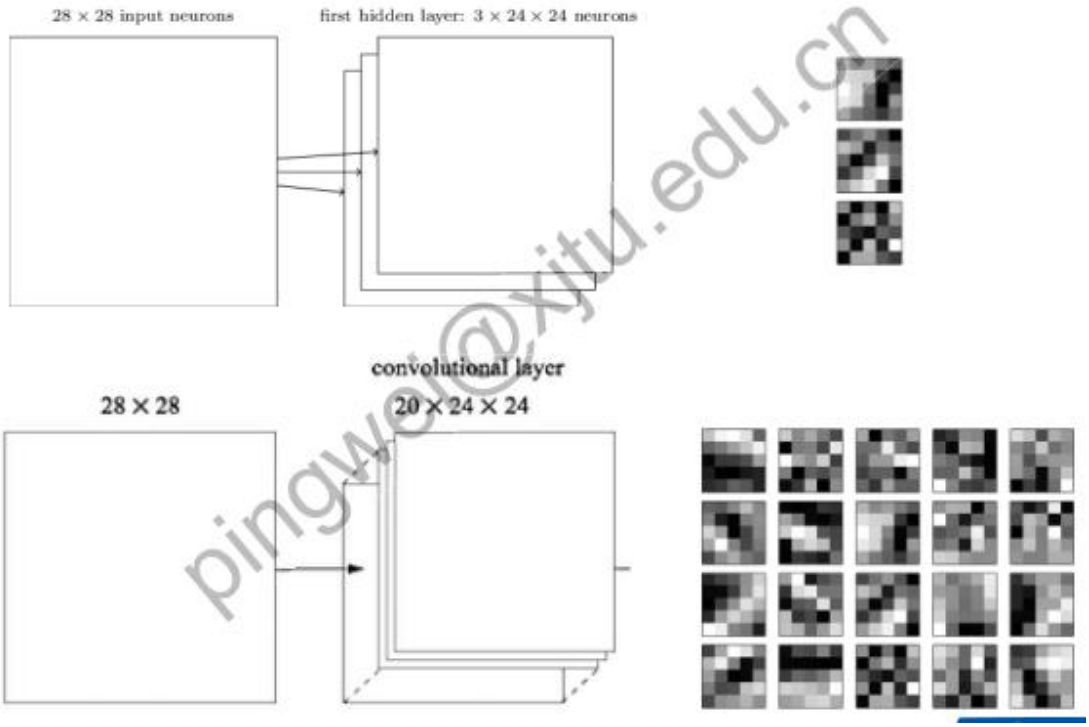
$$\text{Leaky Relu} = \begin{cases} x & if \; x \geq 0 \\ \alpha x & if \; x < 0 \end{cases}$$



Exponential Linear Unit (ELU) 函数

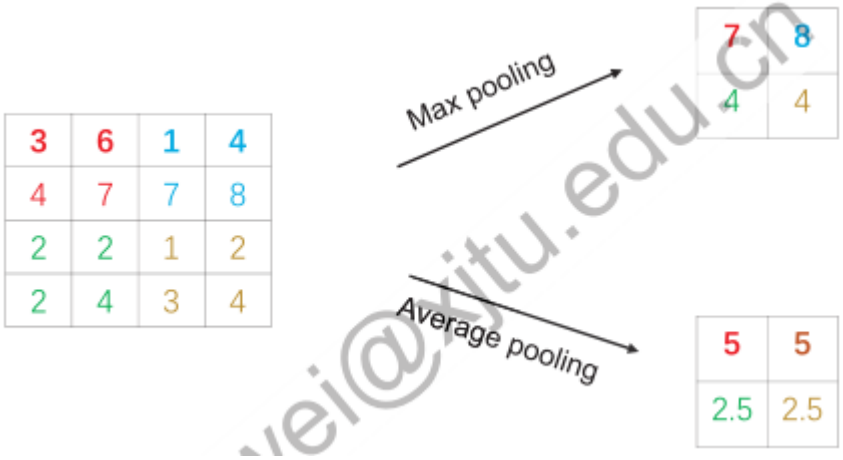
$$ELU(x) = \begin{cases} x & if \; x \geq 0 \\ \lambda(\exp(x) - 1) & if \; x < 0 \end{cases}$$

卷积单元：多通道卷积



卷积单元：池化

定义：池化层夹在连续的卷积层中间，用于压缩数据和参数的量，减小过拟合；
常用池化函数：**最大值池化、平均池化**：



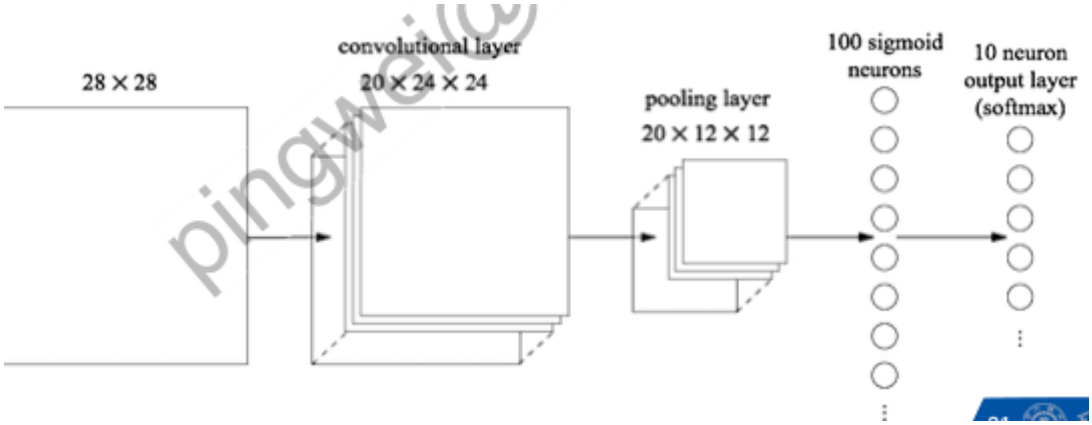
作用：
1. **特征不变性**：池化操作使模型更关注是否存在某些特征，而不是特征具体位置。
2. **特征降维性**：从而减少参数数量，避免过拟合

卷积单元的作用：
1. **局部感知**：相邻细胞具有相似和重叠的感受域
2. **权值共享**：权重不因位置不同而不同。
3. **池化降维：**保持信息，降低参数维度，减少过拟合。

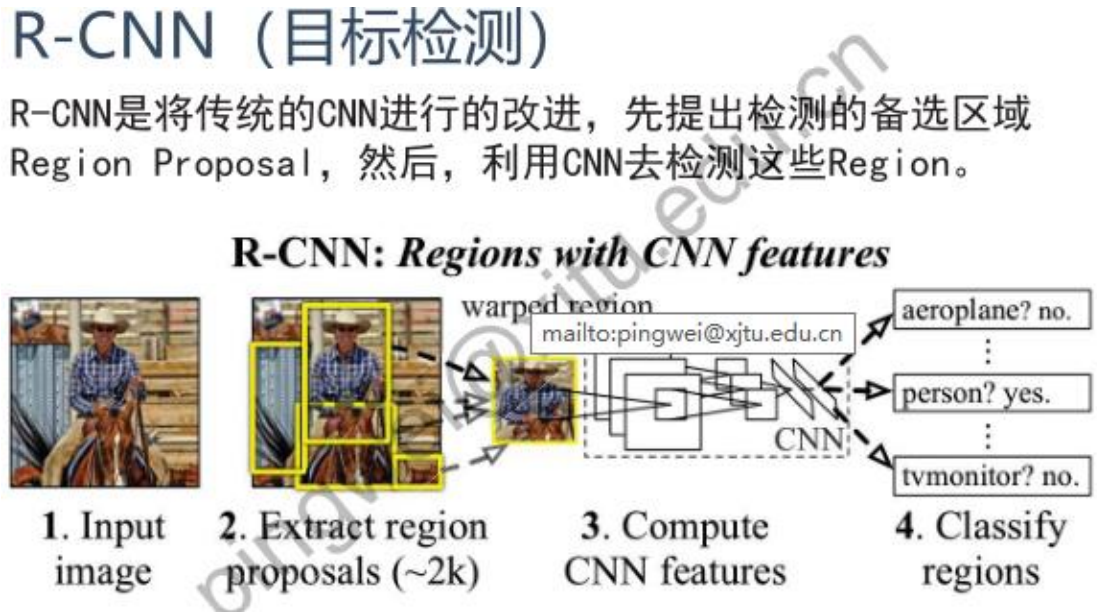
全连接层：

在整个卷积神经网络中起到“分类器”的作用。卷积层、池化层、激活函数层等操作提取相关特征，全连接层将学到的特征表示映射到样本的标记空间，实现目标输出。（输出层中使用 softmax 激活函数）

$$z = W^T h + b; K: 类别个数; z: K \times 1 \text{ 向量}$$

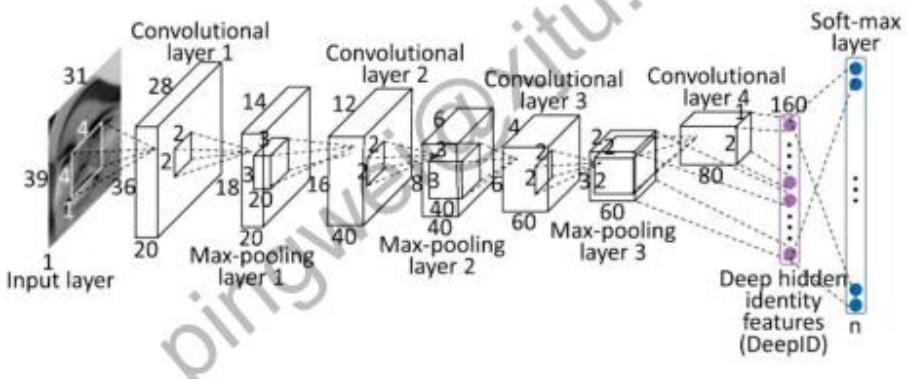


具体例子：

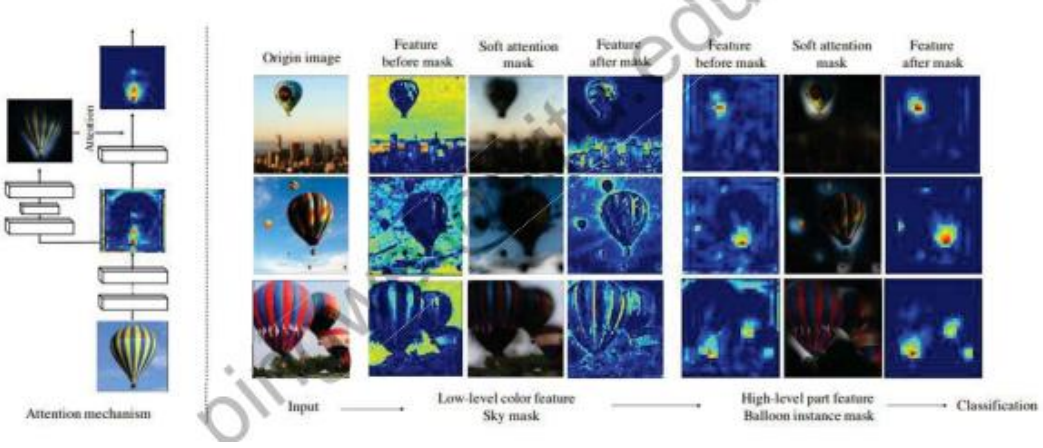


DeepID（人脸验证）

利用CNN提取特征（考虑局部的特征，又考虑全局的特征），推断两张图片是不是同一个人。



Attention Model



第七章：循环神经网络

理解：通过一个隐层来确定信息是否存储，考试的时候会考 LSTM 的内部结构。

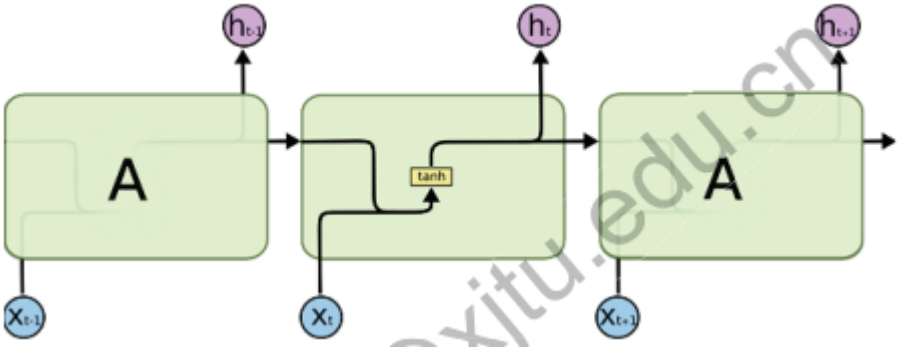
前馈神经网络：模型输出和模型本身没有反馈连接的神经网络。

容易处理网格数据，很难处理序列数据、没有长期记忆能力

循环神经网络：是一种具有从后续层到前面层反馈连接或者同层之间神经元连接的神经网络，常用于处理顺序数据。

网络中具有环结构

结构：输入->隐层->输出



RNN 每个时间点的网络拓扑结构相同，在任意 t 时间下，包含输入层、隐层、输出层。RNN 的隐层的输出一分为二，一份传给输出层，一份与下一时刻外界输入作为隐层输入。

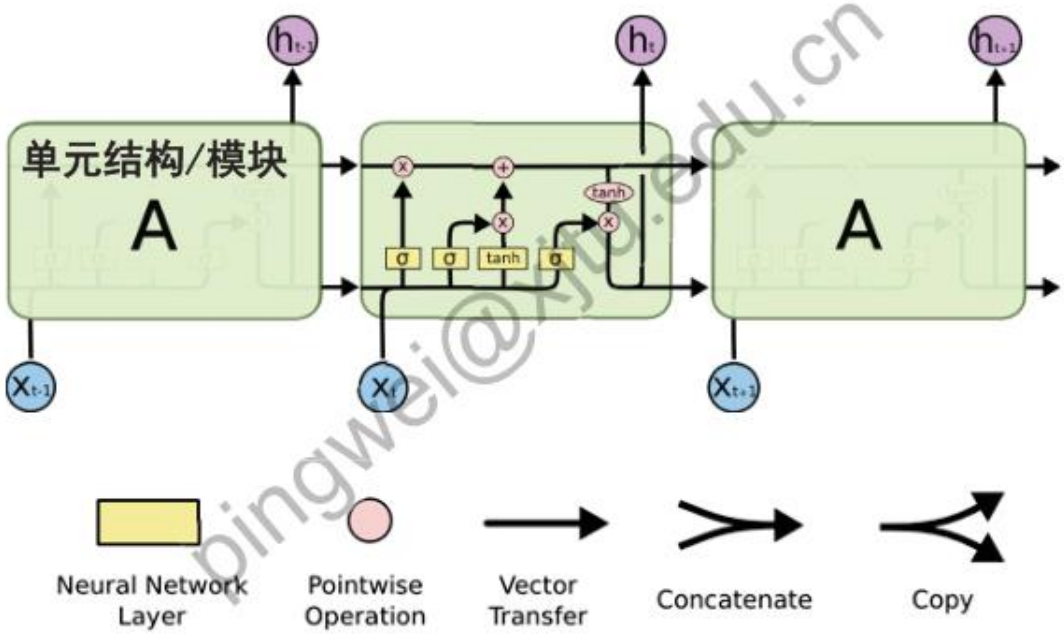
激活函数一般为 tanh 函数

隐层输出不仅进入输入端，还进入了下一时间步骤的隐层，所以它能持续保留信息，能够根据之前状态推出后面状态。

缺点：
1. 梯度消失
2. 难以刻画“长期依赖关系”

长短记忆网络

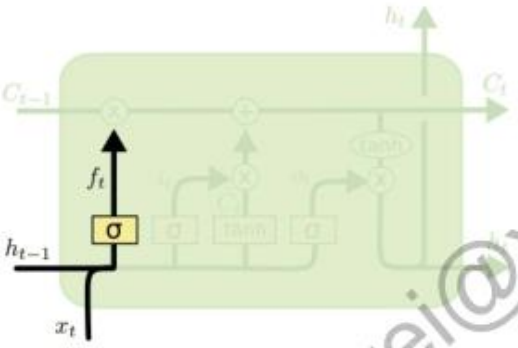
定义：是 RNN 的一种，适合处理和预测时间序列中间隔和延迟相对较长的重要事件。



LSTM 核心：单元状态，及图上方流动的水平线。单元状态类似于传送带，直接在整个链上运行，只有少量的线性交互，保证信息流动稳定性。

通过精心设计的称作为“门”的结构来去除或增加信息到单元状态的能力。门是一种让信息选择式通过的方法。LSTM 拥有三个门来保护和控制状态：**遗忘门、输入门、输出门**。

■ 遗忘门

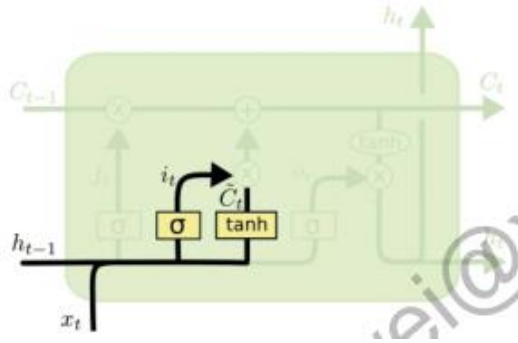


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

理解：根据输出状态、输入，确定前一层单元状态要遗忘的东西(σ函数表示允许通过信息的多少)

LSTM 中的第一步是决定从单元状态中**丢弃什么信息**，而这个决定通过一个叫做遗忘门完成，该门会读取h_{t-1}, x_t，经过σ函数处理，输出一个 0,1 之间的数字。表示信息完全舍弃~完全保留。

■ 输入门

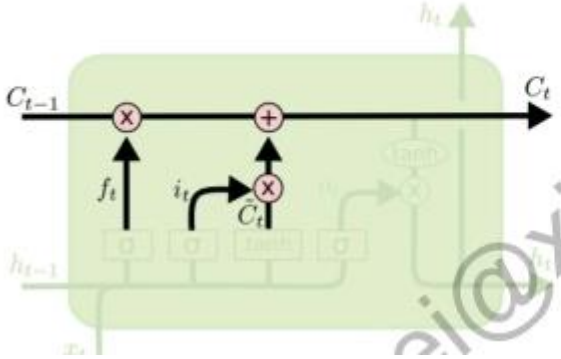


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

理解：决定什么值要新增到单元状态中，$\check{C}_t$生成这层的记忆单元、i_t类似之前遗忘层，表示这一层的记忆单元$\check{C}_t$对单元状态更新的程度。

该步确定**什么样的新信息被存放在单元状态中**。包含两部分，σ称为“输入门层”，决定什么值更新；同时，一个$tanh$层创建一个新的**候选值向量**$\check{C}_t$，被加入状态中。

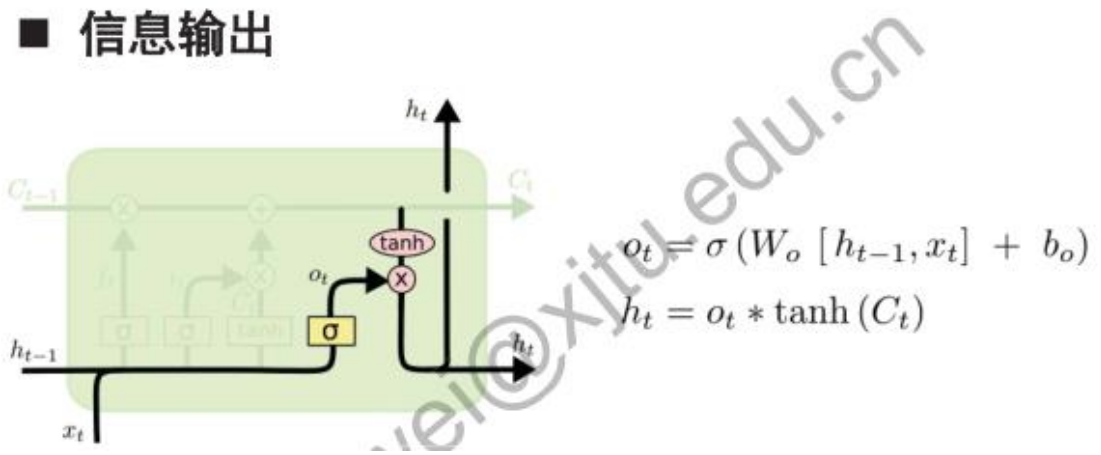
■ 状态更新



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

理解：当前单元状态=上一层单元状态*更新系数+本层记忆单元*更新系数

现在对旧单元进行更新，C_{t-1}更新为C_t。



理解：本层的单元状态经过 tanh 变成待输出量、o_t类似之前遗忘层，表示输出量与待输出量的相关程度。

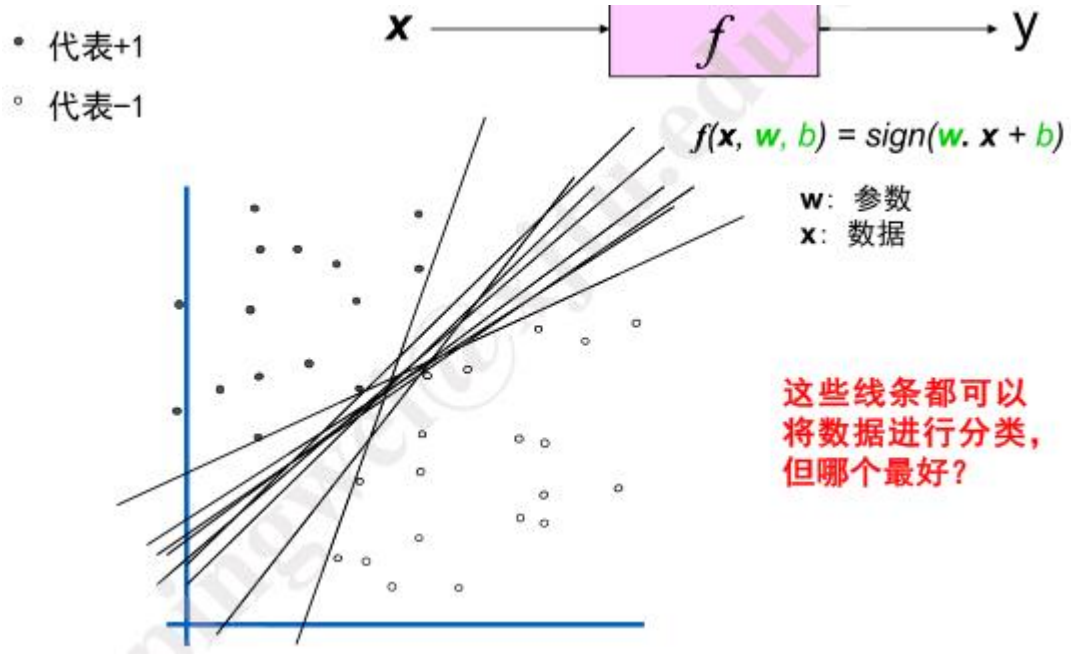
将o_t与$\tanh(C_t)$的乘积作为输出信息h_t。

第八章：支持向量机与核方法

理解：SVM 的基本思想是找到一条最大切分的超平面，我们给定平面方程与约束条件，得到拉格朗日方程，求其对偶问题的解,就是我们要找的系数 w、b。核函数是一种映射，但是在求解 SVM 的时候，我们只用知道类似$x_n^T x_m, e.g. k(x_n, x_m)$来定义两点之间距离就好。

1.支持向量机(SVM)：一类按监督学习方式对数据进行分类的广义线性分类器。

2.SVM 的决策边界是对学习样本求解的最大边距超平面。



分类器间隔：$\begin{cases} 1. \text{边界在遇到一个数据点前能增加的距离。} \\ 2. \text{边界最近的数据点与此边界之间的距离。} \end{cases}$

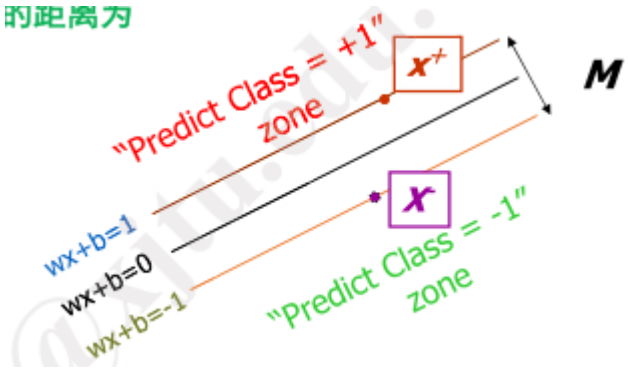
最大间隔：线性分类器为具有最大间隔的线性分类器。（支撑向量机）

该间隔的位置由数据点的子集确定，这些数据点称为支撑向量。

一个点 z 到平面 wx+b=0 的距离为： $\frac{w^T z+b}{|w|}$

目标：1.实现对于所有样本的正确分类：$w^T x_i + b \geq 1 \; if \; y_i = +1;$

$w^T x_i + b \leq 1 \; if \; y_i = -1 \rightarrow y_i(w^T x_i + b) \geq 1$



最大化间隔：$M = \frac{(x^+ - x^-)w}{|w|} = \frac{2}{|w|}$等价于最小化$\frac{1}{2}w^T w \; \because \begin{cases} 1.\, \mathrm{wx}^+ + b = +1 \\ 2.\, \mathrm{wx}^- + b = -1 \\ 3.\, \mathrm{w(x^+ - x^-)} = 2 \end{cases}$

最大化间隔分类器可表达为以下形式：$\min \phi(w) = \frac{1}{2}w^T w; s.t. \forall i, y_i(w^T x_i + b) \geq 1$

拉格朗日乘数法： $\boldsymbol{L(x, \lambda) = f(x) - \lambda g(x)}$

多条件拉格朗日方法： $\max f(x_1, \dots, x_n) \; s.t. \; g_i(x_1, \dots, x_n) = 0 \; i = 1, \dots, M$

拉格朗日方程：$L(x_1, \dots, x_n, \lambda_1, \dots, \lambda_M) = f(x_1, \dots, x_n) - \sum_{k=1}^{M} \lambda_k g_k(x_1, \dots, x_n)$

KKT 条件：最优化$f(x)$；使得$g_i(x) \leq 0, h_j(x) = 0$

$L(x, \mu, \lambda) = f(x) + \mu^T g(x) + \lambda^T h(x)$

其中：$g(x) = \big(g_1(x), \dots, g_m(x)\big)^T, h(x) = \big(h_1(x), \dots, h_l(x)\big)^T$

最大化 f(x)：$\nabla f(x^*) - \sum_{i=1}^{m} \mu_i \nabla \mathrm{g_i}(x^*) - \sum_{j=1}^{l} \lambda_j \nabla \mathrm{h_j}(x^*) = 0$

最小化 f(x)：$\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla \mathrm{g_i}(x^*) + \sum_{j=1}^{l} \lambda_j \nabla \mathrm{h_j}(x^*) = 0$

原始可行性：$g_i(x^*) \leq 0, for \; i = 1, \dots, m; h_j(x^*) = 0, for \; j = 1, \dots, l$

对偶可行性：$\mu_i \geq 0 \; for \; i = 1, .., m$

互补松弛性:$\sum_{i=1}^{m} \mu_i g_i(x^*) = 0$

求解 svm:

1. $L(\boldsymbol{w}, b, a) = \frac{1}{2}w^T w - \sum_{n=1}^{N} a_n \{y_n(w^T x_n + b) - 1\}$

2. $\arg\min_{w,b} \frac{1}{2}w^T w \; s.t. \forall i = 1, \dots, N, y_i(w^T x_i + b) \geq 1$

求导后：$w_n = \sum_{n=1}^{N} a_n y_n x_n \; ; 0 = \sum_{n=1}^{N} a_n y_n$

3. $L^*(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m y_n y_m k(x_n, x_m)$;

$k(x_n, x_m) = x_n^T x_m; s.t. \sum_{n=1}^{N} a_n y_n = 0; a_n \geq 0 \; n = 1, \dots, N$

4.利用 SMO 算法，迭代求解最优解a_i →再求出w

解决线性不可分问题：$\begin{cases}1.\text{使用松弛变量和惩罚因子，允许错分样本的存在}\\2.\text{使用核函数，将支持向量机变为非线性模型}\end{cases}$

1.引入松弛变量

$\arg\min_w \frac{1}{2}w^Tw + C\sum_{i=1}^{N}\gamma_i$ (γ_i是松弛变量，非零表示样本违背约束条件)

$s.t.\forall i = 1,\dots,N; y_i(\boldsymbol{w^T x_i} + b) \geq 1 - \gamma_i; \gamma_i \geq 0$

2.使用核方法：

对偶问题：$L^*(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m y_n y_m x_n^T x_m$

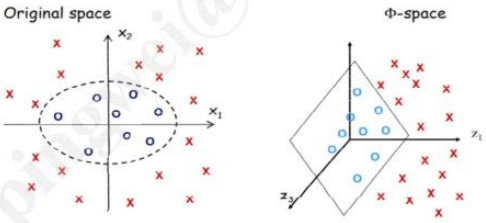
若不存在一个超平面，则将特征空间向更高维映射。

核函数定义$k(x,x') = \phi(x)^T\phi(x')$

核函数是对称函数,$k(x,x') = k(x',x)$

e.g.

- $\boldsymbol{x}$和$\boldsymbol{z}$是二维向量，$\boldsymbol{x} = (x_1,x_2)^{\mathrm{T}}$，$\boldsymbol{z} = (z_1,z_2)^{\mathrm{T}}$
- $k(\boldsymbol{x},\boldsymbol{z}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{z})^2$是一个核函数，特征空间映射为
  $\phi: R^2 \to R^3 \qquad \phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$
- $k(\boldsymbol{x},\boldsymbol{z}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{z})^2 = (x_1z_1 + x_2z_2)^2 =$
  $(x_1^2, x_2^2, \sqrt{2}x_1x_2)\cdot(z_1^2, z_2^2, \sqrt{2}z_1z_2) = \phi(\boldsymbol{x})\cdot\phi(\boldsymbol{z})$



常见的核函数：$k(x,x') = (x\times x')^d$或$k(x,x') = (x\times x'+c)^d$

高斯核：$k(x,x') = \exp[-\frac{||x-x'||^2}{2\sigma^2}]$

sigmoid 核：$k(x,x') = \tanh(ax^Tx' + b)$

构建核：确保选择函数对应于某个特征空间中的内积。

基于已有的核函数创造新的核函数：

- $k(x,x')=ck_1(x,x')$
- $k(x,x')=f(x)k_1(x,x')f(x')$
- $k(x,x')=q(k_1(x,x'))$
- $k(x,x')=\exp(k_1(x,x'))$
- $k(x,x')=k_1(x,x')+k_2(x,x')$
- $k(x,x')=k_1(x,x')k_2(x,x')$
- $k(x,x')=k_3(\phi(x),\phi(x'))$
- $k(x,x')=x^TAx'$
- $k(x,x')=k_a(x_a,x_a')+k_b(x_b,x_b')$
- $k(x,x')=k_a(x_a,x_a')k_b(x_b,x_b')$

其中，c > 0是一个常数，f(·)是任意函数，q(·)是一个非负系数多项式，$\phi(x)$是一个从x到R^M的函数，$k_3(\cdot,\cdot)$是一个在 上的有效的核函数，A是对称的正半正定矩阵，x_a和x_b是$x=(x_a,x_b)$的变量，k_a和k_b是各自空间上的有效核函数

核方法：核的概念被表述为特征空间中的内积，这允许我们通过使用核替换来构建许多著名的算法的扩展，PCA、最近邻分类器.

基本思想：如果一个算法中输入向量 x 以内积的形式出现，则可以将此输入向量的内积替换为其他核(kernel),即将x^Tx'替换为$k(x,x')$.

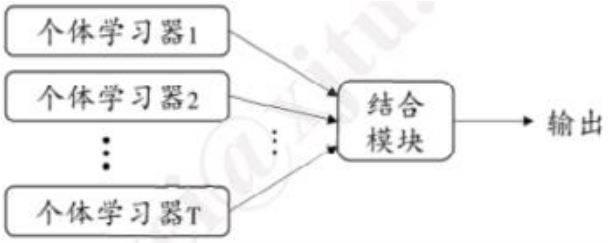
核方法提供强大的模块化，不需要改变底层学习算法来适应核函数的特定选择；此外，我们可以替换不同的算法，保持相同的核。

第九章：集成学习与随机森林

理解：迭代构建多个学习器，通过构建学习器分类情况来更新样本权重;

1.个体与集成

集成学习：通过构建并结合多个学习器来完成学习任务，提升性能；也称为多分类器系统、基于委员会的学习。



“**同质**”：集成中质包含同种类型的个体学习器，其中的个体学习器称为“基学习器”

“**异质**”：集成中质包含不同种类型的个体学习器，其中的个体学习器称为“组件学习器”

个体与集成：集成个体应“好而不同”

理论上可以证明，随着集成中个体分类器数目 T 的增大，集成的错误率将指数级下降，最终趋于 0。

假设：基学习器的误差相互独立。但个体学习器是为解决同一个问题训练出来的，不可能独立。因此，个体学习器的“准确性”和“多样性”存在冲突。如何产生并结合“好而不同”的个体学习器是集成学习的核心。

分类：$\begin{cases} 个体学习器之间存在强依赖关系、必须串行生成的序列化方法，如：boosting \\ 个体学习器之间不存在强依赖关系、可同时生成的序列化方法，如：bagging \end{cases}$

Boosting

理解：同一种学习器对样本进行学习，主要用于二分类任务中，生成一个学习器后，得到其性能评分，通过评分对于样本分布（样本重要性）进行改变，最后得到一个强学习器。

定义：一族可将弱学习器提升为强学习器的算法。

思想：对于一个复杂任务，将多个专家判断进行适当的综合所得的判断，要比其中任何一个专家单独的判断好。

概率近似正确（PAC）学习框架中，一个概念（类），如果存在一个多项式的学习算法能学习它，并且正确率很好，那么就称这个概念是强可学习的；如果一个概念，多项式学习后正确率仅比随机猜测略好，那么就称这个概念是弱可学习的。

PAC 学习框架中，一个概念是强可学习的充要条件是这个概念是弱可学习的。

工作机制：先从初始训练集训练出一个基学习器，再根据学习器表现对训练样本分布进行调整，使得**先前基学习器做错的训练样本在后续受更多关注**，然后基于调整后样本分布来训练下一个基学习器；如此重复进行，使得学习器数目达到先前的值 T，最终加权结合。

1. 个体学习器存在强依赖关系，串行生成
2. 每次调整训练数据集的样本分布

Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:
1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim D_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

代表算法：AdaBoost 算法：

理解：迭代训练分类器，更新不同分类器权值与数据集分布，构成最终分类器。首先计算生成分类器的误差率（$e_m = \sum_{G_m(x_i) \neq y_i} w_{mi}$），根据误差率算出这个分类器权重$a_m = \frac{1}{2}\log(\frac{1}{e_m} - 1)$;再根据此次训练情况来更新分布$w_{m+1,i} = \frac{w_{m,i}}{z_m}\exp(a_m y_i G_m(x_i))$,分错了分类器权重越高其占比重越大,最后线性组合后通过 sign 函数得到最终分类器。

二分类问题中，$y_i \in \{-1, +1\}, f(x)$为真实函数，$h_t(x)$为第 t 个分类器。

$$h_1(x) \in \{-1,+1\}$$
$$h_2(x) \in \{-1,+1\}$$
$$\vdots$$
$$h_T(x) \in \{-1,+1\}$$

$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Weak classifiers　　strong classifier

二分类问题，$y_i \in \{-1,+1\}$，$f(x)$为真实函数，$h_t(x)$为第t个分类器

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程：

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m.$　　初始化样本权值分布

2: **for** $t = 1, 2, \ldots, T$ **do**

3: $h_t = \mathfrak{L}(D, \mathcal{D}_t);$　　基于分布D_t从数据集D中训练出分类器$h_t(x)$

4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}));$　　估计h_t的误差

5: **if** $\epsilon_t > 0.5$ **then break**

6: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right);$　　确定h_t的权重，误分类误差率越小的基本分类器在最终分类器的作用越大

7: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$　　更新样本分布D_t，Z_t为规范化因子，确保D_{t+1}为一个分布，误分类样本的权值得以扩大，被正确分类的样本的权值得以缩小

8: **end for**

输出： $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$　　输出为基学习器的线性加权组合

AdaBoost算法：

输入：训练数据集　$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$　，其中　$x_i \in \mathcal{X} \subseteq R^n, y_i \in \mathcal{Y} = \{+1, -1\}, i = 1, 2, \cdots, N$　；弱学习算法

输出：分类器 $G(x)$

1. 初始化训练数据的权值分布

$$D_1 = (w_{11}, w_{12}, \cdots, w_{1N}), \quad w_{1i} = \frac{1}{N}, \quad i = 1, 2, \cdots, N$$

2. 对　$m = 1, 2, \cdots, M$

2.1 使用具有权值分布 D_m 的训练数据集学习，得到基本分类器

$$G_m(x): \mathcal{X} \to \{-1, +1\}$$

2.2 计算 $G_m(x)$ 在训练数据集上的分类误差率

$$e_m = \sum_{i=1}^{N} P(G_m(x_i) \neq y_i)$$
$$= \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i) = \sum_{G_m(x_i) \neq y_i} w_{mi}$$

2.3 计算 $G_m(x)$ 的系数

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}$$

2.4 更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1}, \cdots, w_{m+1,i}, \cdots, w_{m+1,N})$$
$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)),$$
$$= \begin{cases} \frac{w_{mi}}{Z_m} \exp(-\alpha_m), G_m(x_i) = y_i \\ \frac{w_{mi}}{Z_m} \exp(\alpha_m), G_m(x_i) \neq y_i \end{cases} \quad i = 1, 2, \cdots,$$

其中，Z_m 是规范化因子

$$Z_m = \sum_{i=1}^{N} w_{mi} \exp(-\alpha_m y_i, G_m(x_i))$$

3. 构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

得到最终分类器

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

注意：

A.Boosting 算法要求学习器能对特定数据分布进行学习：

1.**重赋权法**：在训练的每一轮，根据样本分布为每个训练样本重新赋一个权重。

2.**重采样法**（无法接受带权样本的基学习算法）：在每一轮，根据样本分布对训练集重新采样，采用重采样的样本集对基学习器进行训练。

B.每一轮都要检验当前基分类器是否比随机猜测好，不满足时，抛弃当前学习器，学习终止。初始设置的轮数未达到，最终集成性能不佳（过早停止），此时采用重采样，可以重启，避免训练过程过早停止，学习持续到预设的 T 轮结束。

举例：

- **例1：给定如表8.1所示的训练数据，假设弱分类器由$x < v$或$x > v$产生，其阈值v使得该分类器在数据集上分类误差最低。试用AdaBoost算法学习学习一个强分类器。**

表 8.1 训练数据表

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

- **解：初始化数据权值分布：** $D_1 = (w_{11}, w_{12}, \cdots, w_{110})$

  对m=1 $w_{1i} = 0.1, \quad i = 1,2,\cdots,10$

(a) 在权值分布为D_1的训练器上，阈值v取2.5时分类误差率最低，故基本分类器为

$$G_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x > 2.5 \end{cases}$$

(b) $G_1(x)$**在训练数据集上的误差率**$e_1 = P(G_1(x_i) \neq y_i) = 0.$

(c) 计算$G_1(x)$**的系数：** $\alpha_1 = \frac{1}{2} log \frac{1-e_1}{e_1} = 0.4236$

(d) 更新训练数据的权值分布：

$$D_2 = (w_{21}, \cdots, w_{2i}, \cdots, w_{210})$$

$$w_{2i} = \frac{w_{1i}}{Z_1} \exp(-\alpha_1 y_i G_1(x_i)), \quad i = 1,2,\cdots,10$$

$$D_2 = (0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.1666, 0.1666, 0.1666, 0.0715)$$

$$f_1(x) = 0.4236 G_1(x)$$

分类器在$sign[f_1(x)]$**在训练数据集上有3个误分类点。**

对m=2：

(a) 在权值分布为D_2的训练器上，阈值v取8.5时分类误差率最低，故基本分类器为

$$G_2(x)=\begin{cases}1, & x<8.5\\ -1, & x>8.5\end{cases}$$

(b) $G_2(x)$在训练数据集上的误差率$e_2=0.2145$

(c) 计算$G_2(x)$的系数：$\alpha_2=0.6496$

(d) 更新训练数据的权值分布：

$$D_3=(0.0455,0.0455,0.0455,0.1667,0.1667,0.1667,0.1060,0.1060,0.1060,0.0455)$$

$$f_2(x)=0.4236G_1(x)+0.6496G_2(x)$$

分类器在$sign[f_2(x)]$在训练数据集上有3个误分类点。

对m=3：

(a) 在权值分布为D_3的训练器上，阈值v取5.5时分类误差率最低，故基本分类器为

$$G_3(x)=\begin{cases}1, & x>5.5\\ -1, & x<5.5\end{cases}$$

(b) $G_3(x)$在训练数据集上的误差率$e_3=0.1820$

(c) 计算$G_2(x)$的系数：$\alpha_2=0.7514$

(d) 更新训练数据的权值分布：

$$D_4=(0.125,0.125,0.125,0.102,0.102,0.102,0.065,0.065,0.065,0.125)$$

$$f_3(x)=0.4236G_1(x)+0.6496G_2(x)+0.7514G_3(x)$$

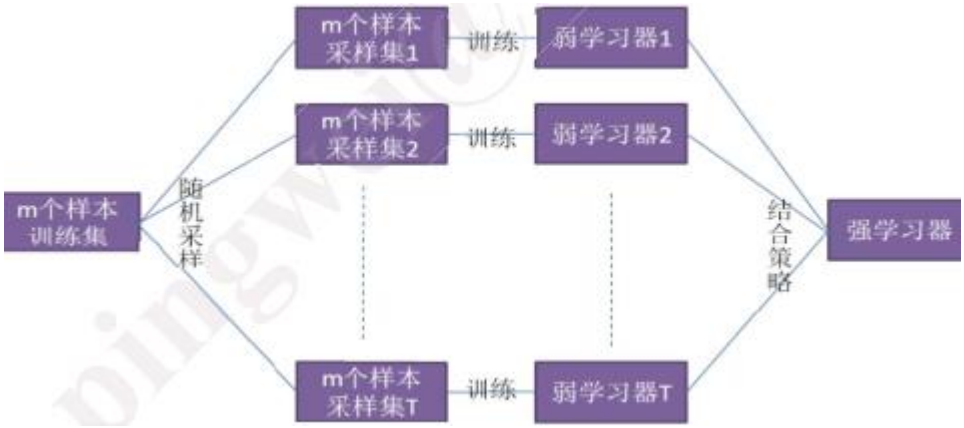
分类器在$sign[f_3(x)]$在训练数据集上有0个误分类点，得到最终分类器：$G(x)=\text{sign}[f_3(x)]=\text{sign}[0.4236G_1(x)+0.6496G_2(x)+0.7514G_3(x)]$

Bagging 与随机森林

理解：可以使用不同分类器，从样本中抽样 m，分别训练出学习器，最后通过最多投票（分类问题）、简单平均方法（回归问题）得到结果

个体学习器不存在强依赖关系、并行化生成。

自助采样法：给定 m 个样本的数据集，先随机取出一个样本放入采样集中，再把该样本放回初始数据集，使得下次采样时样本仍有可能被选中。经过 m 次随机后，得到 m 个样本集。



流程：采样出 T 个含 m 个训练集大的采样集，然后基于每个采样集训练出一个基学习器，然后再将这些基学习器相结合。

在对预测输出进行结合时，对分类任务使用简单投票法，对回归任务使用简单平均法。

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程：

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$
3: **end for**

输出： $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

算法：1.输入为样本集$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$弱学习算法，弱学习分类迭代次数 T。

输出为最终的强分类器$f(x)$

对于 t=1,2,⋯,T: a.对训练集进行第 t 次随机采样，共采样 m 次，得到包含 m 个样本的采样集D_t b.用采样集D_t训练第 t 个弱学习器$G_t(x)$

如果是分类算法预测，则 T 个弱学习器投出最多票数的类别或者类别之一为最终类别。如果是回归算法，T 个弱学习器得到的回归结果进行算数平均得到的值为最终模型输出。

特点：1.时间复杂度低：假定基学习器的计算复杂度为$O(m)$，采样与投票/平均过程的复杂度为$O(s)$,Bagging 算法的时间复杂度大致为$T(O(m) + O(s))$

由于$O(s)$很小，且 T 为不大的常数，所以训练一个 bagging 集成与直接使用基学习器的复杂度同阶(高效)

与标准的 AdaBoost 只适用于二分类任务不同，Bagging 能不经修改的用于多分类任务、回归任务。

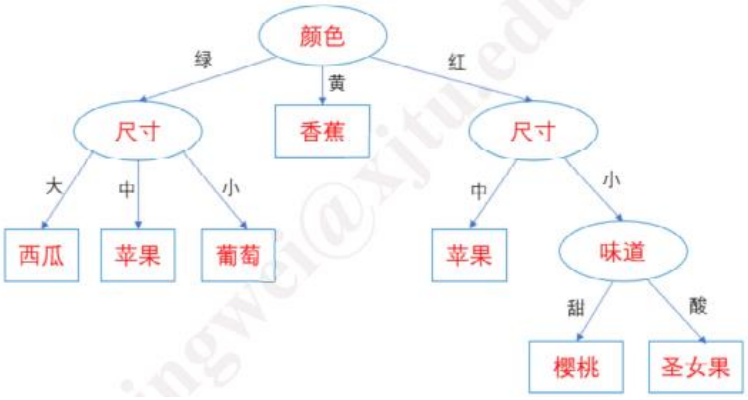
可使用包外估计(由于 Bootstrap sampling 只使用 2/3 左右数据)，剩下样本留作验证。

决策树（Decision Tree）

理解：决策树是找到最优特征，将训练集分为不同子集；迭代找到属性结构。

决策树是一种基本的分类与回归方法，当决策树用于分类时叫分类树，用于回归时叫回归树。
组成：结点、有向边。结点：内部结点（表示一个特征或属性）、叶节点（表示一个类）、根节点（开始测试的特征或属性）

每个节点包含的样本集合根据**属性测试**的结果被划分到子节点中，根节点包含样本全集，从根节点到每个叶节点的路径对应了一个判定测试序列。



e.g.

分类树

分类树：一种描述对实例进行分类的树形结构。在使用分类树进行分类时，从根节点开始，对实例的某一特征进行测试，根据测试结果，将实例分配到其子节点。这时，每一个子节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，直至达到叶结点。最后将实例分到叶节点的类中。
决策树学习：本质上是从训练数据集中归纳出一组分类规则。与训练数据集不相矛盾的决策树（即能对训练数据进行正确分类的决策树）可能有多个，也可能一个都没有。我们需要训练出一个与训练集数据矛盾较小的决策树，同时具有很好的泛化能力。
另一个角度看：决策树学习是由训练数据集估计**条件概率模型**。基于特征空间划分的类的条件概率模型有无穷多个，我们选择的条件概率模型应该不仅对训练数据集有很好的拟合，而且对未知数据有很好的预测。
分类树学习目标：根据给定的训练数据集构建一个决策树模型，使它能够对实例进行正确的分类。

决策树学习用损失函数表示这一目标，其损失函数通常是正则化的极大似然函数，决策树学习的策略是以损失函数为目标函数的最小化。当损失函数确定以后，学习问题就变为在损失函数意义下选择最优策略树的问题。因为**从所有可能的决策树中选取最优决策树是 NP 完全问题**，所以现实中学习算法通常采用启发式算法，近似求解这一最优化问题。这样得到的决策树是次最优的。

给定训练数据集$D=\{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\}$
$x_i=\{x_i^1,x_i^2,\dots x_i^n\}^T$为输入数据，n 为特征维度
$y_i\in\{1,\dots,Q\}$为目标类别

决策树分类算法：
输入：
训练集：D=$(x_1,y_1),(x_2,y_2),\dots,(x_n,y_n)$
属性集：$A=a_1,\dots,a_n$
过程：
函数$TreeGenerate(D,A)$

算法：

1.生成节点根 node

给定结点的所有样本属于同一类

2.if D 中样本全都属于一类别$C_k(\in Y)$ then:

将 node 标记为C_k类叶节点

返回

end if

#无剩余属性可以继续划分

3.if $A=\phi$ OR D 中样本在 A 上取值相同 then

将 node 标记为叶节点，其类别标记为 D 中样本最多的类

return

end if

找到新的最优划分

4.**从 A 中选择最优划分属性a_*：令D_v表示 D 中在a_*上取值为a_*^v的样本子集**

5.for a_*的每一个值a_*^v do:

if D_v为空 then

为 node 生成一个分支

return

else

以$TreeGenerate(D_v, A-\{a_*\})$为分支节点

end if

end for

A.构建根节点，将所有训练数据放在根节点，选择一个最优特征，按照这个特征将训练集分割成子集，使得各个子集在当前条件下最好的分类。

B.如果这些子集已经能够被基本正确分类，那么构建叶节点，并将这些子集分到所对应的叶节点中去。

C.如果还有子集不能被基本正确分类，那么就对这些子集选择新的最优特征，继续对其尽心分割，构建相应的节点

D.如此递归地进行下去，直至所有训练数据子集被基本正确分类，或者没有合适的特征为止。

选择最优划分属性：

从 A 中选择最优划分属性a^* –特征选择

如果利用一个特征进行的分类结果与随机分类的结果没有太大差别，则称这个特征无分类能力。

随着划分过程不断进行，我们希望决策树的分支节点所包含的样本尽可能属于同一类别，即节点的“纯度”越来越高。

信息增益方法：

信息熵：设当前样本集合D中第 k 类样本所占的比例为$p_k(k=1,2,\dots,Q)$，则 **D 的信息熵定义为**：$Ent(D)=-\sum_{k=1}^{Q}p_k\log_2 p_k$；**只有两个变量的话，比例越接近于中间（0.5）越好。**

设离散散度属性 a 有 V 个可能的取值$\{a^1,a^2,..,a^V\}$,若使用 a 来对样本进行划分，则会产生 V 个分支节点，其中第 v 个分支节点包含了 D 中所有在属性 a 上取值为a^v的样本，记为D^v。用属性 a 对样本集 D 进行划分所得到的信息增益为：

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v)$$

则**最优属性为**$\boldsymbol{a^* = \arg\max_{a \in A} Gain(D,a)}$

随机森林

随机森林（random forest）：是 bagging 的一个扩展变种。RF 在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。

具体来说：传统的决策树在决策划分属性时：在当前节点的属性集合（假定有 n 个属性）中选择一个最优属性；

在 RF 中：对基决策树的每个结点，先从该节点的属性集合中随机选择一个包含 K 个属性的子集，然后再从这个子集中选择一个优属性用于划分。

参数 k 控制了随机性的引入程度：若令 K=n，则基决策树的构建与传统决策数相同；若 K=1，则是随机选择一个属性用于划分；一般情况下，推荐值$K = \log_2 n$

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Feature subset size K.

Process:

```
1.  N ← create a tree node based on D;
2.  if all instances in the same class then return N
3.  F ← the set of features that can be split further;
4.  if F is empty then return N
5.  F̃ ← select K features from F randomly;
6.  N.f ← the feature which has the best split point in F̃;
7.  N.p ← the best split point on N.f;
8.  D_l ← subset of D with values on N.f smaller than N.p;
9.  D_r ← subset of D with values on N.f no smaller than N.p;
10. N_l ← call the process with parameters (D_l, K);
11. N_r ← call the process with parameters (D_r, K);
12. return N
```

Output: A random decision tree

算法：

输入为样本集$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$弱学习器算法，弱分类器迭代次数 T。

输出位最终的强分类器$f(x)$

对于$t = 1,2, \ldots, T$

a.**对训练集进行 t 次随机采样**，共采样 m 次，得到包含 m 个样本的采样集D_t

b.用采样集D_t训练第 t 个决策树模型$G_t(x)$。在训练决策树模型的节点的时候，**在节点上所有的样本特征中选择 K 个样本特征**，在这些随机选择的部分样本特征中选择一个最优的特征来做决策树的左右子树划分。

如果分类算法预测，则 T 个弱学习器投出最多票数的类别或类别之一作为最终类别。

特点：

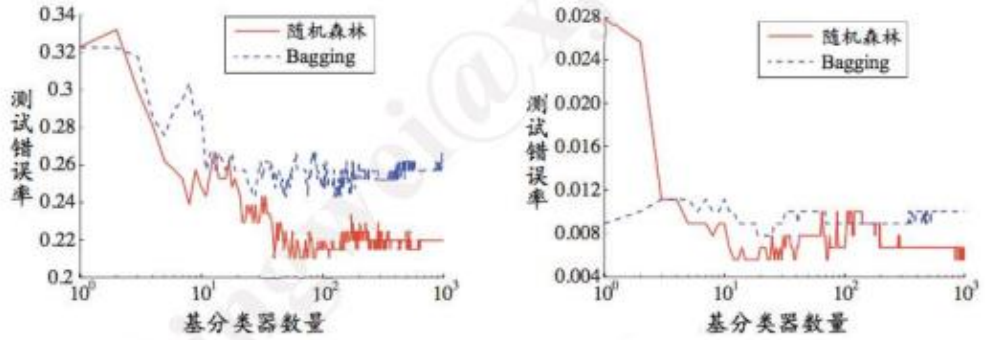
1.随机森林简单、容易实现、计算开销小，在许多现实任务中展现出强大的性能-“代表集成学习技术水平的方法”

2.RF 只对 Bagging 做了小改动：Bagging 中基学习器的多样性仅通过**样本扰动**（通过对初始训练集采样）而来；RF 中基学习器的多样性不仅来**自样本扰动**，还来自**属性扰动**。这使得最终集成的泛华性能可通过个体学习器之间差异度的增加而进一步提升。

3.随机森林的训练效率一般优于 Bagging，因为在个体决策树的构建过程中，Bagging 使用的是“确定型”决策树，**在选择划分属性时要对节点的所有属性进行考察**，而 RF 使用的**“随机型”决策树只需考虑一个属性子集**。

性能对比：

随机森林的收敛性与Bagging相似，如图所示，随机森林的起始性能往往相对较差，特别是在集成中只包含一个基学习器时，因为通过引入属性扰动，随机森林中个体学习器的性能往往有所降低，然而随着个体学习器数目的增加，随机森林通常会收敛到更低的泛化误差。



在两个UCI数据集上，集成规模对随机森林与Bagging的影响

第十章：无监督学习与聚类

无监督学习

在无监督学习中，训练样本的标记信息是未知的，目标是通过对无标记训练样本的学习来揭示数据的内在性质及其规律，为进一步的数据分析提供基础。

聚类（Clustering）

聚类：是无监督学习中应用最广、研究最多的一个内容，其目的是能够自动将未标记的数据根据自身的特点划分为若干个通常是不相交的子集（“**簇（cluster）**”）。

通过聚类算法，不仅可以自动组织数据，还能挖掘一些数据的隐藏结构和属性，如：“浅色

瓜”,…
聚类算法也可以作为其他数据处理的基础，如数据降维、可视化。
应用：通过话题聚类网页；根据表达式聚类蛋白质序列；根据消费记录对客户进行分类。

定义：

假定样本集$D=\{x_1,x_2,\dots,x_m\}$包含 m 个无标记样本，每个样本$x_i=(x_{i1},x_{i2},\dots x_{in})$是一个 n 维特征向量，聚类算法将样本集 D 划分为 k 个不想交的簇$\{C_l|l=1,2,\dots,k\}$

其中$C_{l'}\cap_{l'\neq l}C_l\neq\phi$且$D=\cup_l C_l$

$\lambda_j\in\{1,2,..,k]$表示样本x_j的簇标记，即$x_j\in C_{\lambda_j}$

聚类结果可以用 m 个元素的簇标记向量表示：$\lambda=(\lambda_1,\lambda_2,..,\lambda_m)$

性能度量：

聚类的性能度量也称作为聚类的“有效性指标”（validity index）

对于聚类结果，需要通过某种性能度量来评估其好坏。

“物以类聚”，同一簇样本尽可能的相似，不同簇的样本尽可能不同。

性能度量分为两类：一类是将聚类结果直接与某个参考模型进行比较（外部指标）；另一类是直接考察聚类结果而不用任何参考模型（内部指标）

外部指标：对比聚类结果和参考模型的簇划分、聚类结果的簇标记响亮和参考模型的簇标记向量。

内部指标：基于样本间的距离和簇中心点间的距离。

距离计算：

性质：

对函数 dist(‘,’),若它是一个“距离度量”，则需要满足一些基本性质:

非负性：$dist(x_i,x_j)=0$;同一性：$dist(x_i,x_j)=0\leftrightarrow x_i=x_j$;

对称性：$dist(x_i,x_j)=dist(x_j,x_i)$;直递性：$dist(x_i,x_j)\leq dist(x_i,x_k)+dist(x_k,x_j)$

具体计算：

闵可夫斯基距离：$dist_{mk}(x_i,x_j)=\left(\sum_{u=1}^{n}|x_{iu}-x_{ju}|^p\right)^{\frac{1}{p}}$

$p=2$时，欧氏距离：$dist_{mk}(x_i,x_j)=\sqrt{\sum_{u=1}^{n}|x_{iu}-x_{ju}|^2}$

$p=1$时，曼哈顿距离：$dist_{mk}(x_i,x_j)=\sum_{u=1}^{n}|x_{iu}-x_{ju}|$

K 均值聚类

理解：这部分内容还算是老生常谈

给定样本集$D=\{x_1,x_2,\dots,x_m\}$，“k 均值”(k-means)算法针对聚类所得划分$C=\{C_1,C_2,..,C_k\}$最小化平方误差：

$$E=\sum_{i=1}^{k}\sum_{x\in C_i}||x-\mu_i||_2^2;\text{其中}\mu_i=\frac{1}{|C_i|}\sum_{x\in C_i}x\text{是簇}C_i\text{的均值向量}$$

上式刻画了粗内样本围绕簇均值向量的紧密程度，E 值越小，则簇内样本相似度越高。

输入： 样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$;
聚类簇数 k.

过程：

1: 从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\ldots,\boldsymbol{\mu}_k\}$
2: **repeat**
3: 令 $C_i=\varnothing\ (1\leqslant i\leqslant k)$
4: **for** $j=1,2,\ldots,m$ **do**
5: 计算样本 $\boldsymbol{x}_j$ 与各均值向量 $\boldsymbol{\mu}_i\ (1\leqslant i\leqslant k)$ 的距离: $d_{ji}=\|\boldsymbol{x}_j-\boldsymbol{\mu}_i\|_2$;
6: 根据距离最近的均值向量确定 $\boldsymbol{x}_j$ 的簇标记: $\lambda_j=\arg\min_{i\in\{1,2,\ldots,k\}} d_{ji}$;
7: 将样本 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j}=C_{\lambda_j}\bigcup\{\boldsymbol{x}_j\}$;
8: **end for**
9: **for** $i=1,2,\ldots,k$ **do**
10: 计算新均值向量: $\boldsymbol{\mu}_i'=\frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$;
11: **if** $\boldsymbol{\mu}_i'\neq\boldsymbol{\mu}_i$ **then**
12: 将当前均值向量 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}_i'$
13: **else**
14: 保持当前均值向量不变
15: **end if**
16: **end for**
17: **until** 当前均值向量均未更新

输出： 簇划分 $C=\{C_1,C_2,\ldots,C_k\}$

假定聚类簇数 k=3，初始随机选取 3 个样本作为初始均值向量，对每个向量分别计算到 k 个均值向量的距离，根据最短距离确定簇标记。然后不停重复算法，直到第 n 轮迭代的簇划分结果与第 n-1 轮相同，算法停止。

学习向量量化:

理解：这是类似监督学习的特殊 K-means（给定标签、各原型向量预设的类别标记 $\{t_1,t_2,\ldots,t_q\}$）所以，我们要分多少类 A，多少类 B，多少类 C 都已经确定了。根据区域内符类匹配的点（+），类不匹配（-）的点，来更新类原型向量p_i。

学习向量量化算法（LVQ）：假设数据样本带有类别标记，学习过程利用样本标记作为监督信息辅助聚类。

给定样本集$D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$，每个样本$x_j$的是由 n 个属性描述的特征向量$(x_{j1},x_{j2},\ldots,x_{jn})$,$y_i\in Y$是样本$x_j$的类别标记。LVQ 的目标是学得一组 n 维原型向量$\{p_1,\ldots,p_q\}$,每个原型向量代表一个聚类簇，簇标记$t_i\in Y$。

输入: 样本集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$;
原型向量个数 q, 各原型向量预设的类别标记 $\{t_1,t_2,\ldots,t_q\}$;
学习率 $\eta\in(0,1)$.

过程:

1: 初始化一组原型向量 $\{\boldsymbol{p}_1,\boldsymbol{p}_2,\ldots,\boldsymbol{p}_q\}$
2: **repeat**
3: 从样本集 D 随机选取样本 $(\boldsymbol{x}_j,y_j)$;
4: 计算样本 $\boldsymbol{x}_j$ 与 $\boldsymbol{p}_i$ $(1\leqslant i\leqslant q)$ 的距离: $d_{ji}=\|\boldsymbol{x}_j-\boldsymbol{p}_i\|_2$;
5: 找出与 $\boldsymbol{x}_j$ 距离最近的原型向量 p_{i^*}, $i^*=\arg\min_{i\in\{1,2,\ldots,q\}} d_{ji}$;
6: **if** $y_j=t_{i^*}$ **then**
7: $\boldsymbol{p}'=\boldsymbol{p}_{i^*}+\eta\cdot(\boldsymbol{x}_j-\boldsymbol{p}_{i^*})$
8: **else**
9: $\boldsymbol{p}'=\boldsymbol{p}_{i^*}-\eta\cdot(\boldsymbol{x}_j-\boldsymbol{p}_{i^*})$
10: **end if**
11: 将原型向量 $\boldsymbol{p}_{i^*}$ 更新为 $\boldsymbol{p}'$
12: **until** 满足停止条件

输出: 原型向量 $\{\boldsymbol{p}_1,\boldsymbol{p}_2,\ldots,\boldsymbol{p}_q\}$

输入: 样本集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$;
原型向量个数 q, 各原型向量预设的类别标记 $\{t_1,t_2,\ldots,t_q\}$;
学习率 $\eta\in(0,1)$.

过程:

1: 初始化一组原型向量 $\{\boldsymbol{p}_1,\boldsymbol{p}_2,\ldots,\boldsymbol{p}_q\}$
2: **repeat**
3: 从样本集 D 随机选取样本 $(\boldsymbol{x}_j,y_j)$;
4: 计算样本 $\boldsymbol{x}_j$ 与 $\boldsymbol{p}_i$ $(1\leqslant i\leqslant q)$ 的距离: $d_{ji}=\|\boldsymbol{x}_j-\boldsymbol{p}_i\|_2$;
5: 找出与 $\boldsymbol{x}_j$ 距离最近的原型向量 p_{i^*}, $i^*=\arg\min_{i\in\{1,2,\ldots,q\}} d_{ji}$;
6: **if** $y_j=t_{i^*}$ **then**
7: $\boldsymbol{p}'=\boldsymbol{p}_{i^*}+\eta\cdot(\boldsymbol{x}_j-\boldsymbol{p}_{i^*})$
8: **else**
9: $\boldsymbol{p}'=\boldsymbol{p}_{i^*}-\eta\cdot(\boldsymbol{x}_j-\boldsymbol{p}_{i^*})$
10: **end if**
11: 将原型向量 $\boldsymbol{p}_{i^*}$ 更新为 $\boldsymbol{p}'$
12: **until** 满足停止条件

输出: 原型向量 $\{\boldsymbol{p}_1,\boldsymbol{p}_2,\ldots,\boldsymbol{p}_q\}$

如果原型向量p_{i*}与x_j的类别标记相同，则令p_{i*}向x_j靠拢

如果类别标记不同，则p_{i*}远离x_j

举例:

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

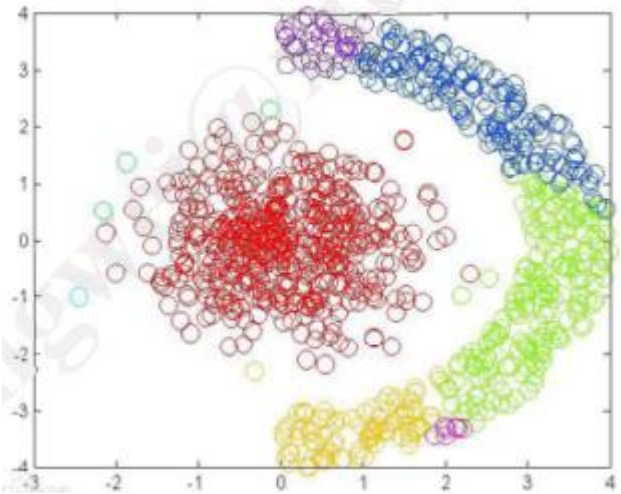
设初始化为样本$\boldsymbol{x}_5, \boldsymbol{x}_{12}, \boldsymbol{x}_{18}, \boldsymbol{x}_{23}, \boldsymbol{x}_{29}$，假定随机选择样本为$\boldsymbol{x}_1$，该样本与当前原型向量$\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4, \boldsymbol{p}_5$的距离分别为0.283, 0.506, 0.434, 0.260, 0.032。更新$\boldsymbol{p}_5$:

$$\begin{aligned}\boldsymbol{p}' &= \boldsymbol{p}_5 + \boldsymbol{\eta}(\boldsymbol{x}_1 - \boldsymbol{p}_5) \\ &= (0.725\ 0.445) + 0.1\{(0.697\ 0.460) - (0.725\ 0.445)\} \\ &= (0.722\ \ 0.447)\end{aligned}$$

密度聚类（DBSCAN）:

理解：无监督学习算法，首先根据$(\epsilon, MinPts)$确定核心对象，在引出直达、可达、相连的概念。使用贪心算法，对于一个核心对象找最大簇，在对于剩下的核心对象迭代寻找。

密度聚类亦称“基于密度的聚类”，此类算法假设聚类结构能通过样本分布的紧密程度确定。通常情况下，密度聚类算法从样本密度的角度来考察样本之间的可连接性，并基于可连接样本不断扩展聚类簇以最终获得聚类结果。



DBSCAN：一种著名的密度聚类算法，它基于一组“邻域”参数$(\epsilon, MinPts)$来刻画样本分布的紧密程度。给定数据集$D = \{x_1, x_2, \ldots, x_m\}$,定义下面概念:

ϵ −邻域：对$x_j \in D$，其ϵ − 邻域包含样本集 D 中与x_j的距离不大于ϵ的样本，$N_\epsilon(x_j) = \{x_j \in D | dist(x_i, x_j) \leq \epsilon\}$

核心对象：若x_j的ϵ −邻域至少包含 MinPts 个样本，则x_j是一个核心对象。

密度直达：若x_j位于x_i的ϵ −邻域中，且x_i是核心对象，则称x_j由x_i密度直达。

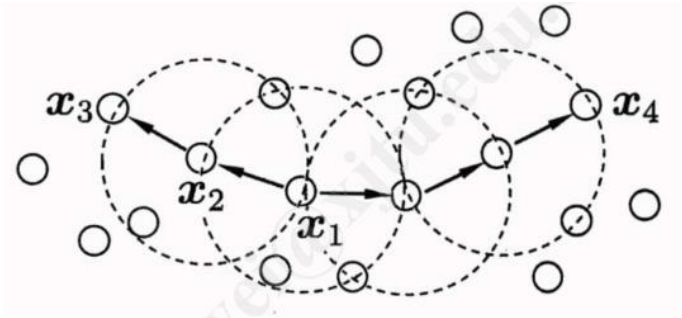
密度可达：对x_i, x_j,若存在样本序列$p_1, p_2, \ldots, p_n$，其中$p_1 = x_i, p_n = x_j$且p_{i+1}由p_i密度直

达，则称x_j由x_i密度可达。

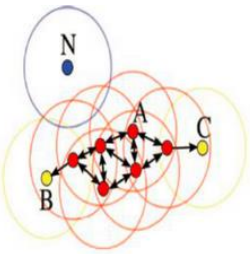
密度相连：对x_i与x_j,若存在x_k使得x_i与x_j均由x_k密度可达，则称x_i与x_j密度相连。

核心对象->密度直达（两个核心）->密度可达（序列）->密度相连（两个点（之间存在序列））

所以只要密度可达，必然密度相连



$MinPts = 3$，虚线为ϵ**-邻域，** x_1**是核心对象，** x_2由x_1**密度直达，** x_3由x_1密度可达，x_4由x_1密度可达，x_3由x_4密度相连



看上图，顺便回顾知识点：

A->B: 密度可达。

B->A: 密度相连。(B->A走不过去)

A->A附近的点：直接密度可达。

DBSCAN 将簇定义为：由密度可达关系导出的最大的密度相连样本集合。形式化地说，给定邻域参数$(\epsilon, MinPts)$,簇$C \in D$是满足以下性质的非空样本子集：

连接性：$x_i \in C, x_j \in C \rightarrow x_i$与$x_j$密度相连

最大性：$x_i \in C, x_j$与x_i密度可达 $\rightarrow x_j \in C$

DBSCAN 算法：先任选数据集中一个核心对象为“种子”，再由此发出确定相应的聚类簇。首先根据给定的邻域参数$(\epsilon, MinPits)$找出所有的核心对象，再从任一核心对象出发，找出其密度可达的样本生成聚类簇，直到所有核心对象均被访问过为止。

输入：样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$；
邻域参数 $(\epsilon, MinPts)$.

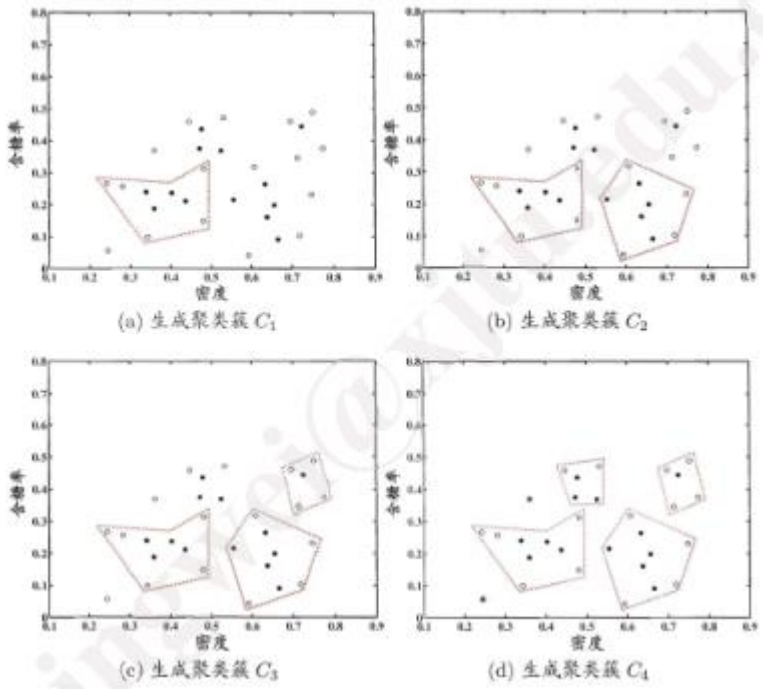
过程：

```
1: 初始化核心对象集合: Ω = ∅
2: for j = 1, 2, ..., m do
3:     确定样本 x_j 的 ε-邻域 N_ε(x_j);
4:     if |N_ε(x_j)| ≥ MinPts then
5:         将样本 x_j 加入核心对象集合: Ω = Ω ∪ {x_j}
6:     end if
7: end for
8: 初始化聚类簇数: k = 0
9: 初始化未访问样本集合: Γ = D
10: while Ω ≠ ∅ do
11:     记录当前未访问样本集合: Γ_old = Γ;
12:     随机选取一个核心对象 o ∈ Ω, 初始化队列 Q =< o >;
13:     Γ = Γ \ {o};
14:     while Q ≠ ∅ do
15:         取出队列 Q 中的首个样本 q;
16:         if |N_ε(q)| ≥ MinPts then
17:             令 Δ = N_ε(q) ∩ Γ;
18:             将 Δ 中的样本加入队列 Q;
19:             Γ = Γ \ Δ;
20:         end if
21:     end while
22:     k = k + 1, 生成聚类簇 C_k = Γ_old \ Γ;
23:     Ω = Ω \ C_k
24: end while
```

输出：簇划分 $\mathcal{C}=\{C_1,C_2,\ldots,C_k\}$

1-7 行：根据给定的邻域参数 $(\epsilon, MinPts)$ 找出所有的核心对象

10-24行：再从任一核心对象出发，找出由其密度可达的样本生成聚类簇，直到所有核心对象均被访问过为止



(a) 生成聚类簇 C_1　(b) 生成聚类簇 C_2

(c) 生成聚类簇 C_3　(d) 生成聚类簇 C_4

层次聚类：

理解：这里有层次二字，指的是再一次操作后就有两个点/类会被分到一起，最后形成树状层次结构。

层次聚类：试图在不同层次对数据进行划分，从而形成树形的聚类结构。数据集的划分可采用“自底向上”的聚合策略，也可采用“自顶向下”的分拆策略。

AGNES 算法：是一种采用自底向上聚合策略的层次聚类算法。

AGNES 算法：将数据集中每个样本看做一个初始聚类簇，然后在算法运行的每一步中找出距离最近的两个聚类簇进行合并，该过程不断重复，直到达到预设的聚类簇个数。

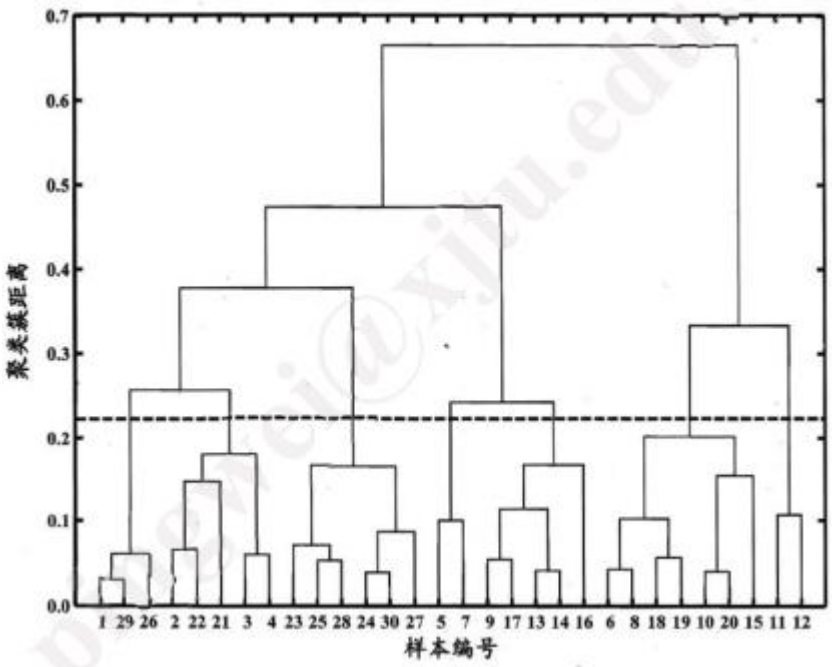
如何计算聚类簇之间的距离

AGNES:给定聚类簇C_i, C_j,可以通过下面式子计算距离:

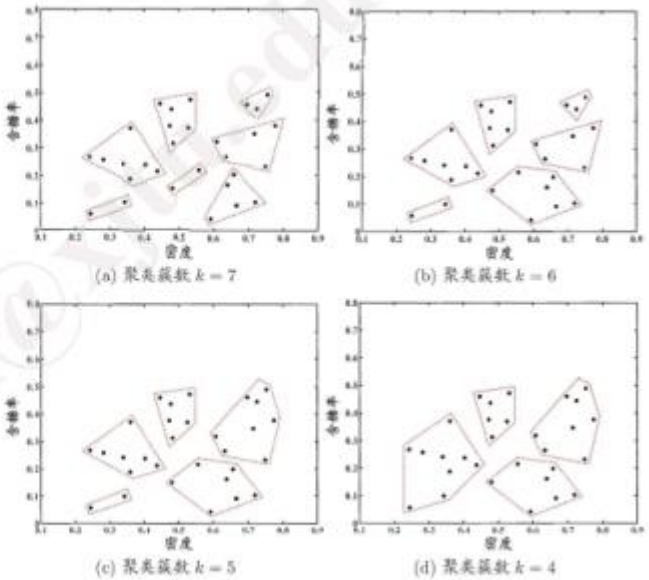
$$最小距离：\ d_{min}(C_i, C_j) = \min_{x \in C_i, z \in C_j} dist(x, z)$$

$$最大距离：\ d_{min}(C_i, C_j) = \max_{x \in C_i, z \in C_j} dist(x, z)$$

$$平均距离：\ d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{z \in C_j} dist(x, z)$$



输入：样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$；
聚类簇距离度量函数 d；
聚类簇数 k.
过程：
1: **for** $j=1,2,\ldots,m$ **do**
2: $C_j=\{\boldsymbol{x}_j\}$
3: **end for**
4: **for** $i=1,2,\ldots,m$ **do**
5: **for** $j=1,2,\ldots,m$ **do**
6: $M(i,j)=d(C_i,C_j)$;
7: $M(j,i)=M(i,j)$
8: **end for**
9: **end for**
10: 设置当前聚类簇个数: $q=m$
11: **while** $q>k$ **do**
12: 找出距离最近的两个聚类簇 C_{i^*} 和 C_{j^*};
13: 合并 C_{i^*} 和 C_{j^*}: $C_{i^*}=C_{i^*}\bigcup C_{j^*}$;
14: **for** $j=j^*+1,j^*+2,\ldots,q$ **do**
15: 将聚类簇 C_j 重编号为 C_{j-1}
16: **end for**
17: 删除距离矩阵 M 的第 j^* 行与第 j^* 列;
18: **for** $j=1,2,\ldots,q-1$ **do**
19: $M(i^*,j)=d(C_{i^*},C_j)$;
20: $M(j,i^*)=M(i^*,j)$
21: **end for**
22: $q=q-1$
23: **end while**
输出：簇划分 $\mathcal{C}=\{C_1,C_2,\ldots,C_k\}$



(a) 聚类簇数 $k=7$ (b) 聚类簇数 $k=6$
(c) 聚类簇数 $k=5$ (d) 聚类簇数 $k=4$

第十一章：采样方法

采样：

理解：通过采样方法可以更容易的模拟复杂结构或一些函数在特定分布下的期望。就是用采样出来的结果来模拟/替代原结果。

采样定义：从一个分布中生成一批服从该分布的样本。

采样的本质上是对随机现象的模拟，根据给定的概率分布，来模拟产生一个对应的随机事件。采样可以让人们对随机事件及其产生过程有更直观的认识。

采样的作用：采样得到的样本集也可以看做是一种非参数模型，即用较少的样本点（经验分布）来近似总体分布，并刻画总体分布中的不确定性。从这个角度来讲采样其实也是一种信息的降维，起到简化问题的作用。

在机器学习中，可能会遇到样本量过大或模型结构复杂导致的求解难度大、没有显示解析解等问题，这种情况下，可以利用采样方法进行模拟，从而对这些复杂模型进行近似求解或推理。一般会转化为**某些函数在特定分布下的积分或期望**，或者是求某些随机变量或参数**在给定数据下的后验分布**。

蒙特卡罗方法：

理解：样本有两个参量，一个是样本本身分布$p(z)$、样本对应的函数值$f(z)$。通过采样计算期望的方法，如果抽取的样本非常多，那么样本的概率乘对应函数值的积分就是期望。

对于大多数实际应用中的概率模型来说，无法精确计算其和或积分，可以采用基于数值采样的近似推断方法，也被称为蒙特卡罗方法。

解决基本问题：寻找某个定义在概率分布$p(z)$上的函数$f(z)$的期望，即计算

$$E[f] = \int f(z)p(z)dz$$

$$E[f] = \frac{1}{L}\sum_{l=1}^{L} f(z^{(l)})$$

对于这个问题，蒙特卡罗方法是从概率$p(z)$中独立抽取 l 个样本$z^{(1)}, z^{(2)}, \ldots, z^{(l)}$，这样期望即可通过有限和的方式计算，以此得到一个经验平均值。

常见的采样方法：

均匀分布采样、逆变换采样、拒绝采样、重要采样、马尔科夫链蒙特卡罗方法、Metrooplis-Hasting 方法、吉布斯采样。

均匀分布采样:

理解：样本点对应的分布是均匀分布的。

均匀分布指整个样本空间中的每一个样本点对应的概率（密度）都是相等的；根据样本空间是否连续，又分为离散均匀分布和连续均匀分布。

均匀分布可以算是最简单的概率分布，几乎是所有采样算法都要用到的基本操作。

用线性同余法生成区间$[0, m-1]$上的伪随机数序列:

$x_{t+1} = (a\,x_t + c)\ mod\ m; m > 0$、系数 $0 < \mathrm{a} < \mathrm{m}$、增量 $0 \le \mathrm{c} < \mathrm{m}; 0 \le \mathrm{x}_0 < m$

一般计算机的程序都是确定的，无法产生真正意义上的完全均匀分布的随机数。

逆变换采样

理解：对于累计分布函数采样;因为累计分布函数是从 0~1 的$\boldsymbol{U}$，且$P = \phi(z) = \int_{\infty}^{z} p(z)dz\,; P \in [0,1] \to z \to \phi^{-1}(\boldsymbol{U})$。即通过对于密度函数分布→累计分布函数逆变换得到→不同累计概率分布时对应的值。

逆变换采样：又称为逆概率积分变换、逆变换法，是伪随机数采样的一种基本方法，从基本的均匀分布产生简单分布。

回顾—累计分布函数，均匀分布

随机变量x位于区间$[a,b]$的概率

$$P(x \in [a,b]) = \int_a^b p(x)dx$$

随机变量x的累积分布函数为

$$P(x \le X) = P(X) = \int_{-\infty}^{X} p(x)dx$$

- 区间[a, b]均匀分布的概率密度

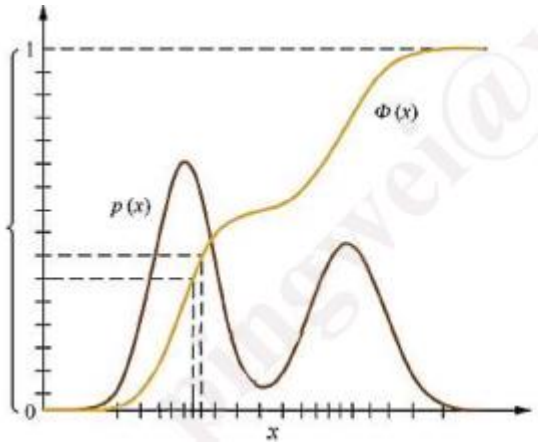
$$p(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & \text{elsewhere} \end{cases}$$

- 均匀分布的累积分布函数

$$\mathrm{P}(X) = \begin{cases} 0 & X < a \\ \frac{X-a}{b-a} & a \le X < b \\ 1 & X \ge b \end{cases}$$

待采样目标分布为$p(x)$，它的累计分布函数为$z = \phi(x) = \int_{-\infty}^{x} p(u)du$，则逆变换采样法步骤为：1.从$U(0,1)$产生一个随机数$z_i$;

2.计算逆函数$x_i = \phi^{-1}(z_i)$;循环上述步骤，产生更多样本。



- 例子：指数分布密度函数

$$p(x) = \lambda \exp(-\lambda x)\ (0 \le x < \infty)$$

- 累积分布函数

$$z = \Phi(x) = 1 - \exp(-\lambda x)$$

- 逆变换

$$x = -\lambda^{-1}\ln(1-z)$$

定理：设 y 是一个连续随机变量，概率密度函数为$p(y)$,累计分布函数为$h(y) = P(y)$，则$z = h(y)$是定义在区间$0 \le z \le 1$上的均匀分布，即$p(z) = 1(0 \le z \le 1)$即 U[0,1]与 P(y)等价。

证明：$h(y)$是累计分布函数，则$0 \le z = h(y) \le 1$，且$h(y)$是单调递增函数;

z 的累计分布函数：$P(z \le Z) = P(h(y) \le Z) = P\left(y \le h^{-1}(Z)\right) = h\left(h^{-1}(Z)\right) = Z$

（累计概率分布 z 的值 小于等于 Z（const var） 的概率是 Z）

$$P(z) = \frac{dF(z)}{dZ} = 1\ (0 \le z \le 1)$$

拒绝采样:

理解：这里累计分布函数逆变换可能比较难于求得，我们首先找到一个简单的概率分布，覆盖原来的概率分布，再在简单的概率分布中取样，得到要求的概率分布。

首先在简单概率分布中抽取z_0，（可通过逆变换采样）再在$[0, kq(z_0)]$上进行抽样。

拒绝采样，又称为接受-拒绝采样，基本思想是用一“更大的概率分布”或“**更简单的概率分布”$q(z)$覆盖原本的概率分布，这个更简单的概率分布比较方便采样（如正态分布）**

$p(z)=\frac{1}{Z_p}p'(z)$为**采样分布**，$p'(z)$为已知分布，Z_p归一化因子

引入较简单分布$q(z)$，称为**提议分布**，从中可以较容易采样。

引入常数 k，对任意 z 满足$kq(z)\geq p'(z)$，$kq(z)$称为**比较函数**

步骤：1.从$q(z)$采样生成一个样本z_0

2.生成区间$[0,kq(z)]$上的均匀分布的一个样本u_0

3.如果$u_0>p'(z_0)$，则该样本被拒绝；否则z_0被接受

4.重复以上过程得到$[z_0,z_1,\dots,z_n]$即是对$p(z)$的一个近似

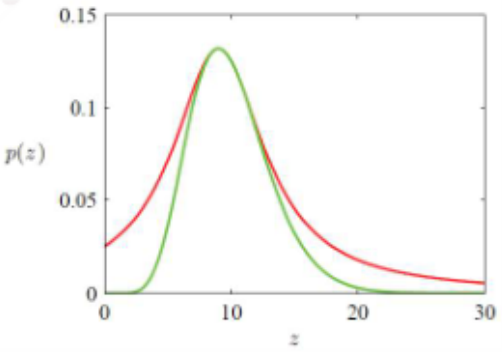
在上述拒绝采样方法中，z 的原始值从概率分布$q(z)$中生成，这些样本之后被接受的概率为$\frac{p'(z)}{kq(z)}$,因此，样本被接受的概率为：$p(accept)=\int\left\{\frac{p'(z)}{kq(z)}\right\}q(z)dz=\frac{1}{k}\int p'(z)dz$

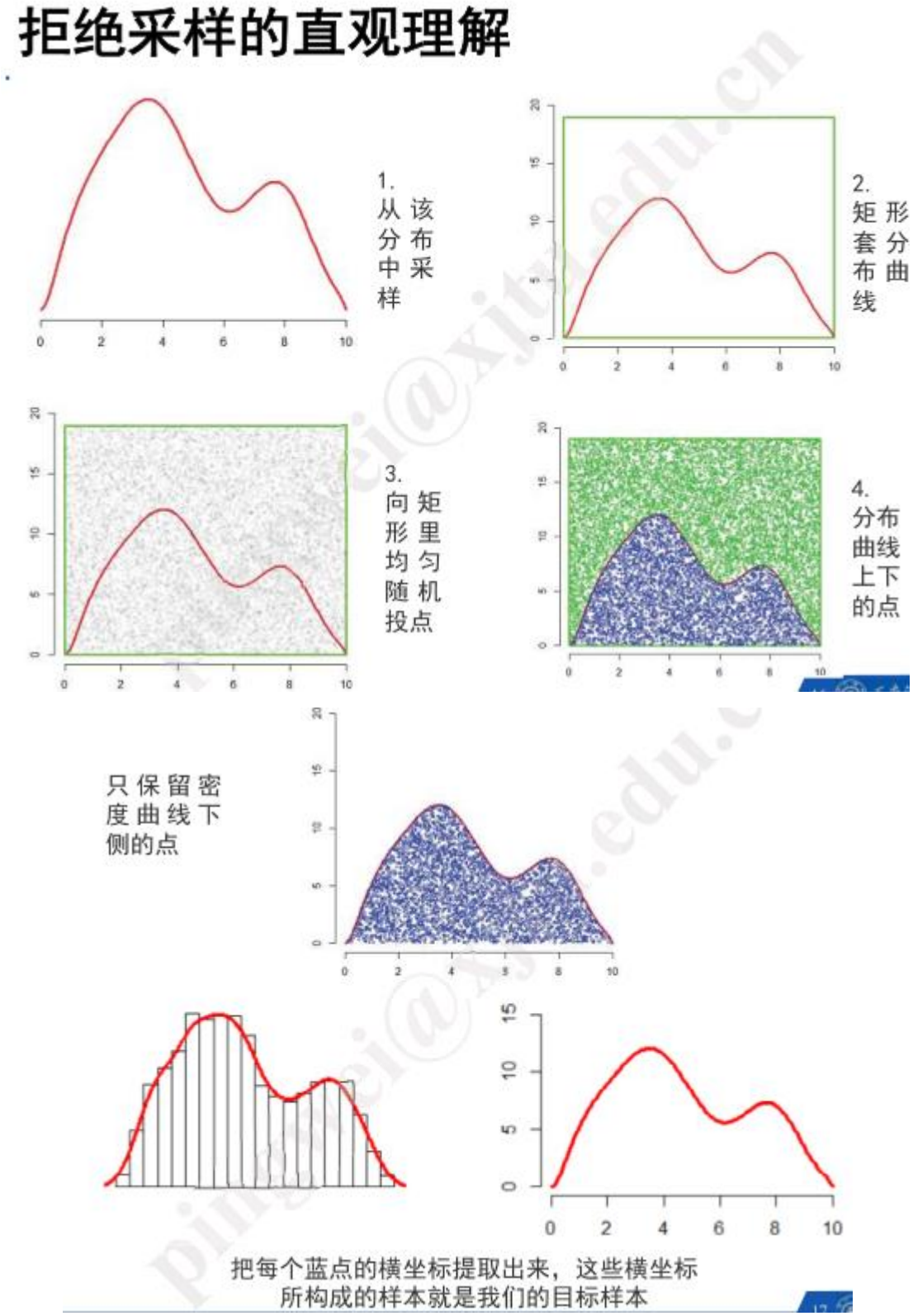
原则上 k 可以取得很大，从而满足总能全覆盖，但是不难发现，k 取得越大，拒绝概率也更高；因此，选取的 k 还要尽可能的小，并且使得$kq(z)$恰好能覆盖$p'(z)$

即：k 尽可能小，采样的有效率越高。

举例：

- Gamma分布的形式为：$\mathrm{Gam}(z|a,b)=\frac{b^a z^{a-1}\exp(-bz)}{\Gamma(a)}$
- 对于$a>1$的情形，它的形状是钟形曲线。使用柯西分布作为提议分布非常合适，柯西分布也是钟形曲线，并且易于采样
- 为使Gamma分布曲线能被完全覆盖，使用$z=b\tan y+c$进行变换，得到$q(z)=\frac{k}{1+\frac{(z-c)^2}{b^2}}$
- 绿色曲线表示Gamma分布的图像，红色曲线表示放缩后的柯西提议分布。从Gamma分布中抽取的样本可以通过从柯西分布中采样然后使用拒绝采样准则的方法得到





重要采样(Importance Sampling)

理解：为了求解期望，将原来难于求解的$p(z)$变成$\frac{p(z)}{q(z)} * q(z)$,其中$q(z)$为容易采样的函数，而

且，我们希望落在$p(z)$较大的区域或者$f(z)p(z)$较大的区域。重要性采样的关键就在这里，把对 f(x)不好求的期望，变成了一个在另一个分布下相对好求的期望。

重要采样方法提供了一种直接近似期望的框架，它本身不是一种从概率分布$p(z)$中采样的方法。

重要采样基于以下假设：直接从$p(z)$中采样非常困难或无法完成，但对于任一给定的 z 值，可以很容易地计算$p(z)$值。

仿照拒绝采样的思路，可以利用提议分布；与拒绝采样不同的是：提一分部要使用最优的采样函数，不用全部覆盖原分布函数；并且所有生成的样本都会被保留。

如果有服从$p(z)$的 L 个独立样本$z^{(l)}(1,..,,L)$，可近似计算

$$avg\ f = \frac{1}{L}\sum_{l=1}^{L} f(z^{(l)})或E(f) = \sum_{l=1}^{L} p(z^{(l)})f(z^{(l)})$$

以上方法明显的缺陷是当 z 的维度较高时，需要采样的点数会很多；其次，在高维空间的均匀采样效率低。

我们希望选择落在$p(z)$较大的区域内的点或者$f(z)p(z)$较大

我们可以利用从提议分布$q(z)$采样的样本$\{z^{(l)}\}$表达期望

$$E(f) = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz \approx \frac{1}{L}\sum_{l=1}^{L}\frac{p(z^{(l)})}{q(z^{\{l\}})}f(z^{(l)})$$

理解：这里为什么是$\frac{1}{L}$,因为$Q(z)$在$U[0,1]$采样后的分布，天然符合 q(z),即取到每个点的概率都相同，所以这里是$\frac{1}{L}$。

$r_l = \frac{p(z^{(l)})}{q(z^{(l)})}$称为重要性权重，修正由于从错误概率分布中采样引出的偏差。

引入$p(z) = \frac{1}{Z_p}\check{p}(z), q(z) = \frac{1}{Z_q}\check{q}(z); \check{r} = \frac{\check{p}(z)}{\check{q}(z)}$

$$E(f) = \int f(z)p(z)dz = \frac{z_q}{z_p'}\int f(z)\frac{\check{p}^{(z)}}{\check{q}^{(z)}}q(z)dz \approx \frac{Z_q}{Z_p}\frac{1}{L}\sum_{l=1}^{L}\check{r}f(z^{(;)})$$

$$\frac{Z_p}{Z_q} = \frac{1}{Z_q}\int \check{p}(z)dz = \int \frac{\check{p}^{(z)}}{\check{q}^{(z)}}q(z)dz \approx \frac{1}{L}\sum_{l=1}^{L}\check{r}_l$$

因此：

$$E(f) \approx \sum_{l=1}^{L}\omega_l f(\boldsymbol{z}^{(l)});\ \omega_l = \frac{\check{r}_l}{\sum_m \check{r}_m} = \frac{\check{p}(z^{(l)})/q(z^{(l)})}{\sum_m \check{p}(z^{(m)})/q(z^{(m)})}$$

马尔科夫蒙特卡罗方法（MCMC）

拒绝采样和重要性都使用了另一个容易采样的分布$q(z)$进行计算。在某些学习算法中，概率分布$p(z)$往往会被表示成无向模型。为了从分布$p(z)$中近似采样，会利用马尔科夫链进行蒙特卡罗估计。这种方法被称为马尔科夫蒙特卡罗估计方法。

待采样分布表示为$p(z) = \frac{1}{Z_p}\check{p}(z)$,$\check{p}(z)$容易计算

MCMC 方法假设一个提议分布，此分部依赖于现有状态$q(z|z^{(\tau)})$,τ为当前时刻索引；提议分布尽可能简单，可以从中直接采样。

在计算的每一个循环，从提议分布中产生样本，然后根据一定的概率接受此样本。

Metropolis/ Metropolis-Hastings 算法

细致平稳性（detail balance condition）：当非周期性的马尔可夫链的概率转移矩阵和每一个状态的概率满足：

$$\pi(i)p(j|i) = \pi(j)p(i|j)$$

最终得到的状态 π是该马尔可夫链的平稳分布。

证明:

$$\sum_{i=1}^{\infty}\pi(i)p(j|i) = \sum_{i=1}^{\infty}\pi(j)p(i|j) = \pi(j)\sum_{i=1}^{\infty}p(i|j) = \pi(j)$$

理解：这个算法主要是**构造细致平稳性地马尔可夫链**（即提议分布是对称的）在 i 到 j 的转移过程中$\pi(i)q(j|i)\alpha(j|i)$;其中$\alpha(j|a)$表示接受概率。我们假设马尔科夫链的转移方式是以提议分布q来进行的。

$$\check{p}(z^{(\tau)})q(z^{(*)}|z^{(\tau)})\check{p}(z^{(*)})q(z^{(\tau)}|z^{(*)}) = \check{p}(z^{(*)})q(z^{(\tau)}|z^{(*)})\check{p}(z^{(\tau)})q(z^{(*)}|z^{(\tau)})$$

$$\rightarrow \check{p}(z^{(\tau)})q(z^{(*)}|z^{(\tau)}) * \frac{\check{p}(z^{(*)})q(z^{(\tau)}|z^{(*)})}{\check{p}(z^{(\tau)})q(z^{(*)}|z^{(\tau)})} = \check{p}(z^{(*)})q(z^{(\tau)}|z^{(*)})$$

即其会依据$\frac{\check{p}(z^{(*)})q(z^{(\tau)}|z^{(*)})}{\check{p}(z^{(\tau)})q(z^{(\tau)})}$的概率进行状态转移

在基本 Metropolis 算法中，假设提议分布是对称的，即$q(z_A|z_B) = q(z_B|z_A)$

候选样本的接受概率定义为$A(z^*, z^{(\tau)}) = \min\{1, \frac{\check{p}(z)}{\check{p}(z^{(\tau)})}\}$

在 Metropolis-Hastings 算法中，假设提议分布不是对称的，即$q(z_A|z_B) \neq q(z_B|z_A)$

候选样本的接受概率定义为$A(z^*, z^{(\tau)}) = \min\{1, \frac{\check{p}(z^*)q(z^{(\tau)}|z^*)}{\check{p}(z^{(\tau)})q(z^*|z^{(\tau)})}\}$

流程：

初始化：最大迭代次数 T, 需要样本数目 n，初始化样本$z^{(0)} \sim q(z)$, 循环迭代以下过程：

1. 从条件概率分布$q(z|z^{(\tau)})$中采样得到样本值z^*

2.从均匀分布中采样$u \sim (0,1)$

3.如果$A(z^*, z^{(\tau)}) > u$,则接受z^*，即$z^{(\tau+1)} = z^*$；否则z^*被舍弃，且$z^{(\tau+1)} = z^{(\tau)}$（实现时只保留一个重复样本，记录重复次数）

4.如果未达到最大迭代次数，循环上述过程；否则停止。

输出：根据需要截取尾部 n 个样本

Metropolis-Hastings算法

但是Metropolis算法构造出的接受概率可能会很小，这样造成算法要经过很多的迭代才能到达平稳分布。为了满足细致平稳，不等式的两边都乘以了一个小的接受概率，那我们可以把其中一个接受概率乘以一个数变为1，另外一边的接受概率也乘上相同的倍数，具体的做法如下：

$$\pi(i)q(j|i)\alpha(j|i) = \pi(j)q(i|j)\alpha(i|j)$$
$$\pi(i)q(j|i)\pi(j)q(i|j) = \pi(j)q(i|j)\pi(i)q(j|i)$$
$$\pi(i)q(j|i)\frac{\pi(j)q(i|j)}{\pi(i)q(j|i)} = \pi(j)q(i|j)$$
$$or$$
$$\pi(i)q(j|i) = \pi(j)q(i|j)\frac{\pi(i)q(j|i)}{\pi(j)q(i|j)}$$

整合上面等式可以得到：

$$\alpha(j|i) = min\{\frac{\pi(j)q(i|j)}{\pi(i)q(j|i)}, 1\}$$

算法流程：

Algorithm 24.2: Metropolis Hastings algorithm

Initialize $x^{(0)}$;
for $s = 0, 1, 2, \ldots$ **do**
  Define $x = x^s$;
  Sample $x' \sim q(x'|x)$;
  Compute acceptance probability

$$\alpha = \frac{\tilde{p}(x')q(x|x')}{\tilde{p}(x)q(x'|x)}$$

  Compute $r = \min(1, \alpha)$;
  Sample $u \sim U(0, 1)$;
  Set new sample to

$$x^{s+1} = \begin{cases} x' & \text{if } u < r \\ x^s & \text{if } u \geq r \end{cases}$$

Gibbs 采样

理解：可以看成多维下的马尔科夫链蒙特卡罗方法。转移函数满足蒙特卡罗算法要求。

Gibbs 采样是一种广泛应用的马尔科夫链蒙特卡罗算法，也可以被看做 Metropolis-hasting 算法的一个具体形式。

待采样分布为：$p(z) = p(z_1, z_2, .., z_M)$

吉布斯采样的每一步将一个变量的值替换为以剩余变量的值为条件，从这个概率分布中采样那个变量的值。即将z_i替换为从概率分布$p(z_i|z_{\neq i})$中抽取的值。$z_{\neq i}$表示$z_1, .., z_M$去掉z_i。

$$\pi\left(x_1^{(1)}, x_2^{(1)}\right)\pi\left(x_2^{(2)}\middle|x_1^{(1)}\right) = \pi\left(x_1^{(1)}\right)\pi\left(x_2^{(1)}\middle|x_1^{(1)}\right)\pi\left(x_2^{(2)}\middle|x_1^{(1)}\right) = \pi\left(x_1^{(1)}, x_2^{(2)}\right)\pi\left(x_2^{(1)}\middle|x_1^{1}\right)$$

基于上面的发现，我们可以这样构造分布$\pi(x_1, x_2)$的马尔可夫链对应的状态转移矩阵P:

$$P(A \to B) = \pi(x_2^{(B)}|x_1^{(1)}) \ \ if \ x_1^{(A)} = x_1^{(B)} = x_1^{(1)}$$

$$P(A \to C) = \pi(x_1^{(C)}|x_2^{(1)}) \ \ if \ x_2^{(A)} = x_2^{(C)} = x_2^{(1)}$$

$$P(A \to D) = 0 \ \ else$$

有了上面这个状态转移矩阵，我们很容易验证平面上的任意两点E, F，满足细致平稳条件:

$$\pi(E)P(E \to F) = \pi(F)P(F \to E)$$

例如，假设有一个在三个变量上的概率分布$p(z_1, z_2, z_3)$

首先将$z_1^{(\tau)}$替换为新值$z_1^{(\tau+1)}$，这个新值是从条件概率分布布$p(z_1|z_2^{(\tau)}, z_3^{(\tau)})$中采样得到的，

接下来，将$z_2^{(\tau)}$替换为$z_2^{(\tau+1)}$,这个新值是从条件概率分布$p(z_2|z_1^{(\tau+1)}, z_3^{(\tau)})$中采样得到的，即$z_1$的新值可以再接下来的采样步骤中直接使用。

然后再用$z_3^{(\tau+1)}$更新$z_3^{(\tau)}$，其中$z_3^{(\tau+1)}$是从$p\left(z_3 \middle| z_1^{(\tau+1)}, z_2^{(\tau+1)}\right)$中进行采样的。

以此类推，在这三个变量之间进行循环。

Gibbs Sampling

1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1|z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2|z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j|z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M|z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.