机器学习课程复习知识点总结

注意

根据课堂笔记整理，如有遗漏或补充请联系@CCP101，卷积神经网络应该会出一道简答题，根据汇报整理了一些问题。

本笔记部分内容来自于维基百科与CSDN等。关于深度学习部分，如未接触过该领域，推荐阅读鱼书与蜥蜴书（14章）相关章节。

注意： 本笔记内含$LaTeX$公式，如使用Typora请打开公式支持。

1.序论

1. **机器学习定义：**

机器学习是人工智能的一个分支。人工智能的研究历史有着一条从以“推理”为重点，到以“知识”为重点，再到以“学习”为重点的自然、清晰的脉络。显然，机器学习是实现人工智能的一个途径，即以机器学习为手段解决人工智能中的问题。机器学习在近30多年已发展为一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、计算复杂性理论等多门学科。机器学习理论主要是设计和分析一些让计算机可以自动“学习”的算法。机器学习算法是一类从数据中自动分析获得规律，并利用规律对未知数据进行预测的算法。因为学习算法中涉及了大量的统计学理论，机器学习与推断统计学联系尤为密切，也被称为统计学习理论。算法设计方面，机器学习理论关注可以实现的，行之有效的学习算法。很多推论问题属于无程序可循难度，所以部分的机器学习研究是开发容易处理的近似算法。
机器学习已广泛应用于数据挖掘、计算机视觉、自然语言处理、生物特征识别、搜索引擎、医学诊断、检测信用卡欺诈、证券市场分析、DNA序列测序、语音和手写识别、战略游戏和机器人等领域。

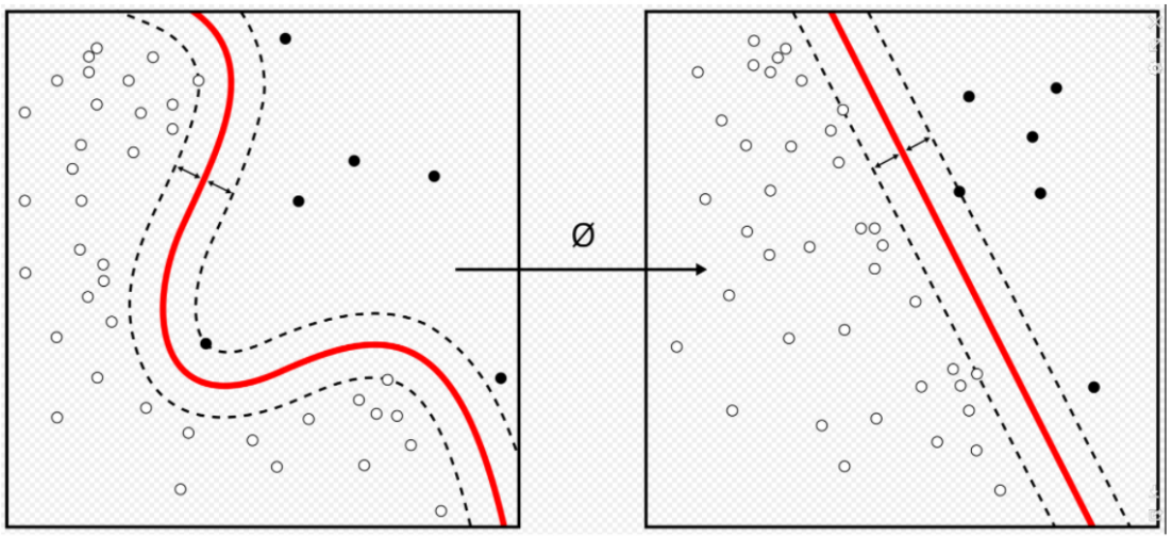
- 机器学习是一门人工智能的科学，该领域的主要研究对象是人工智能，特别是如何在经验学习中改善具体算法的性能。
- 机器学习是对能通过经验自动改进的计算机算法的研究。
- 机器学习是用数据或以往的经验，以此优化计算机程序的性能标准。

2. **深度学习定义：**

深度学习（英语：deep learning）是机器学习的分支，是一种以人工神经网络为架构，对资料进行表征学习的算法。
深度学习是机器学习中一种基于对数据进行表征学习的算法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如，人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。
表征学习的目标是寻求更好的表示方法并创建更好的模型来从大规模未标记数据中学习这些表示方法。表示方法来自神经科学，并松散地创建在类似神经系统中的信息处理和对通信模式的理解上，如神经编码，试图定义拉动神经元的反应之间的关系以及大脑中的神经元的电活动之间的关系。
至今已有数种深度学习框架，如深度神经网络、卷积神经网络和深度置信网络和循环神经网络已被应用在计算机视觉、语音识别、自然语言处理、音频识别与生物信息学等领域并获取了极好的效果。

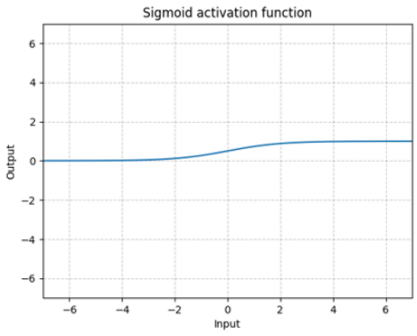
3. **二分法**:

最简单的分类问题，例如在一个二维空间内根据属性使用线性或非线性方法进行区分。实际上是使用单个点与决策边界做对比后得出分类结果。

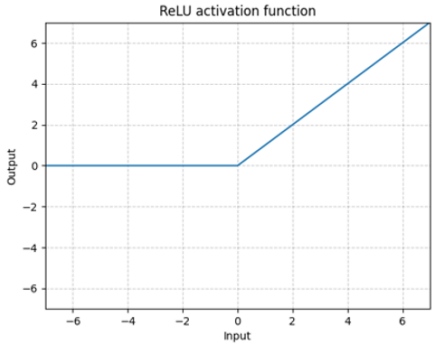


4. **激活函数**:

在计算网络中， 一个节点的激活函数定义了该节点在给定的输入或输入的集合下的输出。标准的计算机芯片电路可以看作是根据输入得到开（1）或关（0）输出的数字电路激活函数。这与神经网络中的线性感知机的行为类似。然而，只有非线性激活函数才允许这种网络仅使用少量节点来计算非平凡问题。主要的激活函数例如sigmoid函数与Relu函数。



$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + \exp(-x)}$$



$$\text{ReLU}(x) = (x)^+ = \max(0, x)$$

5. **softmax函数**:

又名归一化指数函数，Softmax函数实际上是有限项离散概率分布的梯度对数归一化。因此，Softmax函数在包括多项逻辑回归，多项线性判别分析，朴素贝叶斯分类器和人工神经网络等的多种基于概率的多分类问题方法中都有着广泛应用。

$$\text{Softmax}\left(x_i\right) = \frac{\exp\left(x_i\right)}{\sum_j \exp\left(x_j\right)}$$

6. **深度学习基本常识**

训练时各数据集的作用： 首先拆分为（训练集(train)+测试集(test)）+ (验证集(val))，保证验证集训练时模型不可见，保证了验证集的独立性。验证集主要用于评价训练完成后的模型独立结果，因为可能在训练集与测试集上发生过拟合。而训练集与测试集则为训练时可见的数据，首先在训练集上训练，其次在测试集上进行测试，根据测试结果反向传播更新结果。

维度灾难（爆炸）： 考虑这样一个例子，我们有一些图片，每张图片描绘的是小猫或者小狗。我们试图构建一个分类器来自动识别图片中是猫还是狗。要做到这一点，我们首先需要考虑猫、狗的量化特征，这样分类器算法才能利用这些特征对图片进行分类。例如我们可以通过毛皮颜色特征对猫狗进行识别，即通过图片的红色程度、绿色程度、蓝色程度不同，设计一个简单的线性分类器。为了得到更好的分类效果，我们可以增加更多特征，例如颜色、纹理分布和统计信息等。也许我们能得到上百个特征，但是

分类器的效果会变得更好吗？答案有些令人沮丧：并不能！事实上，特征数量超过一定值的时候，分类器的效果反而下降。这就是“维度灾难”。

剪枝：是机器学习与搜索算法当中通过移除决策树中分辨能力较弱的部分而减小决策树大小的方法。剪枝降低了模型的复杂度，因此能够降低过拟合风险，从而降低泛化误差。在决策树算法中，决策树过大会有过拟合的风险，从而在新样本上的泛化性能很差；决策树过小则无法从样本空间中获取重要的结构化信息。然而，由于很难判断新增一个额外的分裂结点能否显著降低误差，人们很难判断何时停止决策树的生长是恰当的。该问题被称为视界限制效应。一个通用的策略是让决策树一直生长，直到每个叶子结点都包含足够少量的样本，而后通过剪枝的方法，移除分辨能力较弱的结点。剪枝应当在减小决策树大小的同时，保证交叉验证下的精度不降低。

特征空间：从原始数据中提取特征是将原始数据映射到一个更高维的空间，特征空间中的特征是对原始数据更高维的抽象。特征工程主要需要解决的问题是对具体的问题构建出适合表示该问题的特征。

7. **分类/回归**

回归分析（英语：Regression Analysis）是一种统计学上分析数据的方法，目的在于了解两个或多个变量间是否相关、相关方向与强度，并建立数学模型以便观察特定变量来预测研究者感兴趣的变量。更具体的来说，回归分析可以帮助人们了解在只有一个自变量变化时因变量的变化量。一般来说，通过回归分析我们可以由给出的自变量估计因变量的条件期望。回归分析是建立因变量Y（或称依变量，反因变量）与自变量X（或称独变量，解释变量）之间关系的模型。简单线性回归使用一个自变量X，复回归使用超过一个自变量$X_1, X_2 \dots X_i$。

统计分类是机器学习非常重要的一个组成部分，它的目标是根据已知样本的某些特征，判断一个新的样本属于哪种已知的样本类。分类是监督学习的一个实例，根据已知训练集提供的样本，通过计算选择特征参数，创建判别函数以对样本进行的分类。与之相对的是无监督学习，例如聚类分析。

一般，对训练集

$$\left\{\left(\vec{x}_1, y_1\right), \left(\vec{x}_2, y_2\right), \dots, \left(\vec{x_m}, y_m\right)\right\}$$

进行学习，建立一个从输入空间 $\vec{x}$到输出空间Y的的映射，$f : \vec{x} \to y$。

8. **Bagging与Boosting**

- Bagging (bootstrap aggregating)即套袋法，其算法过程如下：

A）从原始样本集中抽取训练集.每轮从原始样本集中使用Bootstraping的方法抽取n个训练样本（在训练集中，有些样本可能被多次抽取到，而有些样本可能一次都没有被抽中）.共进行k轮抽取，得到k个训练集.（k个训练集相互独立）

B）每次使用一个训练集得到一个模型，k个训练集共得到k个模型.（注：根据具体问题采用不同的分类或回归方法，如决策树、神经网络等）

C）对分类问题：将上步得到的k个模型采用投票的方式得到分类结果；对回归问题，计算上述模型的均值作为最后的结果.

- Boosting是一族可将弱学习器提升为强学习器的算法，关于Boosting的两个核心问题：

1）在每一轮如何改变训练数据的权值或概率分布？

通过提高那些在前一轮被弱分类器分错样例的权值，减小前一轮分对样本的权值，而误分的样本在后续受到更多的关注.

2）通过什么方式来组合弱分类器？

通过加法模型将弱分类器进行线性组合，比如AdaBoost通过加权多数表决的方式，即增大错误率小的分类器的权值，同时减小错误率较大的分类器的权值.而提升树通过拟合残差的方式逐步减小残差，将每一步生成的模型叠加得到最终模型.

- Bagging，Boosting二者之间的区别

1）样本选择上：

Bagging：训练集是在原始集中有放回选取的，从原始集中选出的各轮训练集之间是独立的.

Boosting：每一轮的训练集不变，只是训练集中每个样例在分类器中的权重发生变化.而权值是根据上一轮的分类结果进行调整.

2）样例权重：

Bagging：使用均匀取样，每个样例的权重相等

Boosting：根据错误率不断调整样例的权值，错误率越大则权重越大.

3）预测函数：

Bagging：所有预测函数的权重相等.

Boosting：每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重.

4）并行计算：

Bagging：各个预测函数可以并行生成

Boosting：各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型的结果.

Bagging 是 Bootstrap Aggregating 的简称，意思就是再取样 (Bootstrap) 然后在每个样本上训练出来的模型取平均，所以是降低模型的 variance. Bagging 比如 Random Forest 这种先天并行的算法都有这个效果。Boosting 则是迭代算法，每一次迭代都根据上一次迭代的预测结果对样本进行加权，所以随着迭代不断进行，误差会越来越小，所以模型的 bias 会不断降低High variance 是model过于复杂overfit，记住太多细节noise，受outlier影响很大；high bias是underfit，model过于简单，cost function不够好。boosting是把许多弱的分类器组合成一个强的分类器。弱的分类器bias高，而强的分类器bias低，所以说boosting起到了降低bias的作用。variance不是boosting的主要考虑因素。bagging是对许多强（甚至过强）的分类器求平均。在这里，每个单独的分类器的bias都是低的，平均之后bias依然低；而每个单独的分类器都强到可能产生overfitting的程度，也就是variance高，求平均的操作起到的作用就是降低这个variance。

2.线性判别函数法

线性可分二分类判别函数式一般表达式： $g(\vec{x}) = \vec{w}^\top \vec{x} + w_0$ ， $\vec{w}$ 是权向量 $\vec{x}$ 为特征向量 w_0 为阈值权

1. **余弦相似度**

两个向量间的余弦值可以通过使用欧几里得点积公式求出：

$a \cdot b = \| a \| \, \|b\| \cos\theta$

给定两个属性向量， A和B，其余弦相似性θ由点积和向量长度给出，如下所示：

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$ ，这里的A_i和B_i别代表向量A和B的各分量。

给出的相似性范围从$(-1, 1)$，-1意味着两个向量指向的方向正好截然相反，1表示它们的指向是完全相同的，0通常表示它们之间是独立的，而在这之间的值则表示中间的相似性或相异性。

对于文本匹配，属性向量A 和B 通常是文档中的词频向量。余弦相似性，可以被看作是在比较过程中把文件长度正规化的方法。

2. **线性分类器**

线性分类器利用决策函数 $g(\vec{x}) = g_1(\vec{x}) - g_2(\vec{x})$，判别方法为：

$$\begin{cases} g(\vec{x}) > 0, & w_1 \\ g(\vec{x}) < 0, & w_2 \\ g(\vec{x}) = 0, & \text{can not judge} \end{cases}$$

3. **点互信息**

机器学习相关文献里面，经常会用到PMI（Pointwise Mutual Information）这个指标来衡量两个事物之间的相关性（比如两个词）。其原理很简单，公式如下：

$$PMI(x,y) = log_2 \frac{P(x,y)}{P(x) \cdot P(y)}$$

在概率论中，我们知道，如果x跟y不相关，则$p(x,y) = p(x)p(y)$。二者相关性越大，则$p(x,y)$就相比于$p(x)p(y)$越大。用后面的式子可能更好理解，在y出现的情况下x出现的条件概率$p(x|y)$除以x本身出现的概率$p(x)$，自然就表示x跟y的相关程度。

这里的log来自于信息论的理论，可以简单理解为，当对$p(x)$取log之后就将一个概率转换为了信息量（要再乘以-1将其变为正数），以2为底时可以简单理解为用多少个$bits$可以表示这个变量。

然而，在计算的过程中，为了避免出现 log_0 的情况，通常采用“正值点互信息”（positive pointwise mutual information, PPMI）来计算，公式如下

$$PPMI(x,y) = max(0, PMI(x,y))$$

4. **SVD奇异值分解**

奇异值分解（singular value decomposition）是线性代数中一种重要的矩阵分解，在信号处理、统计学等领域有重要应用。奇异值分解在某些方面与对称矩阵或厄米矩阵基于特征向量的对角化类似。然而这两种矩阵分解尽管有其相关性，但还是有明显的不同。对称阵特征向量分解的基础是谱分析，而奇异值分解则是谱分析理论在任意矩阵上的推广。深度学习领域目的主要是快速找特征值。

假设M是一个$m \cdot n$阶矩阵，其中的元素全部属于域K，也就是实数域或复数域。如此则存在一个分解使得

$$M = U\Sigma V^*,$$

其中U是$m \cdot m$阶酉矩阵；Σ是$m \cdot n$阶非负实数对角矩阵 而V^*，即V的共轭转置，是$n \cdot n$阶酉矩阵。这样的分解就称作M的奇异值分解。Σ对角线上的元素即为M的奇异值。常见的做法是将奇异值由大而小排列。如此Σ便能由M唯一确定了。（虽然U和V仍然不能确定。）

直观的解释：

- V的列（columns）组成一套对M的正交"输入"或"分析"的基向量。这些向量是M^*M的特征向量。
- U的列（columns）组成一套对M的正交"输出"的基向量。这些向量是MM^*的特征向量。
- Σ对角线上的元素是奇异值，可视为是在输入与输出间进行的标量的"膨胀控制"。这些是MM^*与M^*M的特征值的非负平方根，并与U和V的行向量相对应。

Input

singular value decomposition	$\begin{pmatrix} 1 & 0 & 2 \\ 0 & -1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

Result

$$M = U.\Sigma.V^{\dagger}$$

where

$$M = \begin{pmatrix} 1 & 0 & 2 \\ 0 & -1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$U = \begin{pmatrix} 0.883344 & 0.386937 & -0.264542 \\ 0.459765 & -0.605452 & 0.649649 \\ -0.0912058 & 0.695491 & 0.712723 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 2.45784 & 0 & 0 \\ 0 & 1.36768 & 0 \\ 0 & 0 & 0.297484 \end{pmatrix}$$

$$V = \begin{pmatrix} 0.359399 & 0.282915 & -0.889264 \\ -0.224169 & 0.951207 & 0.212024 \\ 0.905859 & 0.123144 & 0.405284 \end{pmatrix}$$

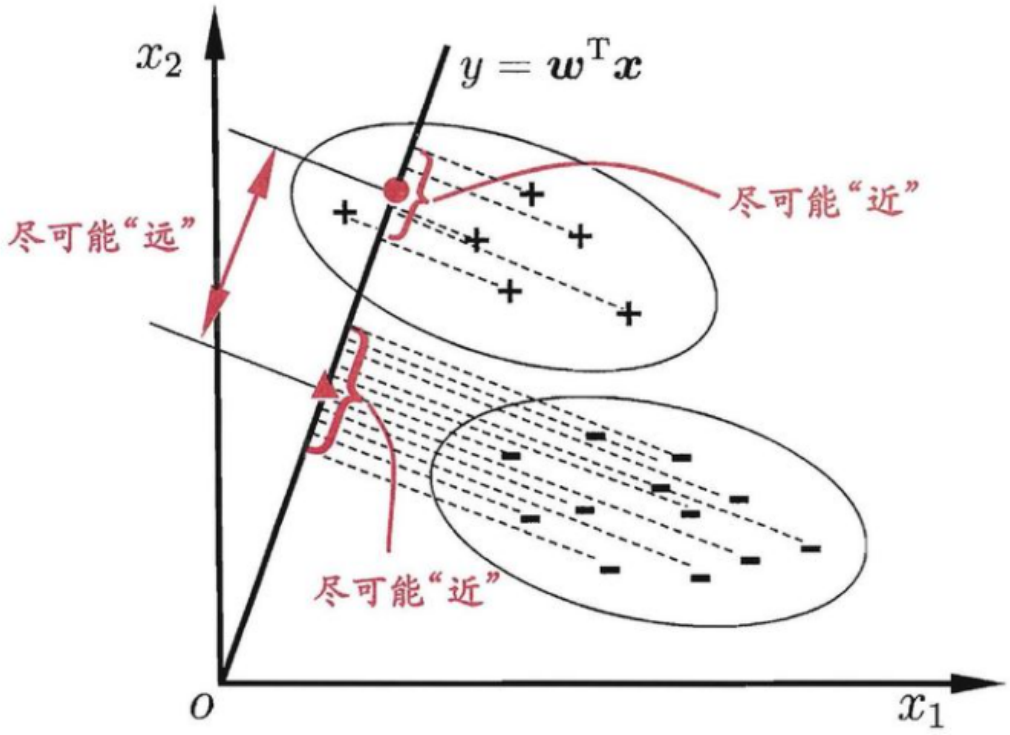
Exact forms

3.*Fisher、感知机

1. **Fisher投影降维**

从d维空间到一维空间的一般数学变换方法

Fisher判别分析的思想非常朴素：给定训练样例集，设法将样例投影到一条直线上，使得同类样例的投影点尽可能接近、不同类样例的投影点尽可能远离。在对新样本进行分类时，将其投影到同样的这条直线上，再根据新样本投影点的位置来确定它的类别。如下所示，给出了一个二维示意图。



2. **Fisher数据定义**

假设：集合X包含各d维样本：$\vec{x_1}, \vec{x_2}, \ldots, \vec{x_n}$，其中$N_1$个属于$W_1$类样本记为子集$x_1$，...，其中$N_n$个属于$W_n$类样本记为子集$x_n$

对于$\vec{x}_n$的分量做线性组合，可得$y_n = \vec{w}^T\vec{x_n}, n = 1, 2, 3, \ldots, N_i, i = 1, 2$

3. **Fisher算法**

- 在d维x空间各类样本均值向量 $\vec{m}_i$

$$\vec{M}_i = \frac{1}{N_i}\sum_{\vec{x}\in x_i}\vec{x}, i = 1, 2$$

- 样本类内离散度矩阵S_i和总类内离散度矩阵S_w

$$S_i = \sum_{\vec{x}\in x_i}(\vec{x} - \vec{m}_i)\cdot(\vec{x} - \vec{m}_i)^T, i = 1, 2$$

$$S_w = S_1 + S_2$$

- 样本类内离散度矩阵S_b

$$S_b = (\vec{m}_1 - \vec{m}_2)\cdot(\vec{m}_1 - \vec{m}_2)^T$$

- 希望投影后，在一维Y空间里各类样本尽可能地分开，也就是说希望两类样本均值之差越大越好，同时希望各类物品内部内部尽量密集，即类内离散度越小越好，因此可将Fisher准则函数定义为

$$J_{\mathrm{F}}(\boldsymbol{w}) = \frac{(\tilde{\boldsymbol{m}}_1 - \tilde{\boldsymbol{m}}_2)^2}{\tilde{\boldsymbol{s}}_1^2 + \tilde{\boldsymbol{s}}_2^2} = \frac{\vec{W}^T S_b \vec{W}}{\vec{W}^T S_w \vec{W}}$$

- 使得$J_{\mathrm{F}}(\boldsymbol{w})$取得最大值的$w^*$为

$$w^* = S_w^{-1}(m_1 - m_2)$$

- 将训练集内所有样本进行投影

$$Y = w^{*T}X$$

- 在一维Y空间中各类样本均值

$$\tilde{m_i} = \frac{1}{N_i}\sum_{y\in y_i} y, i = 1, 2$$

- 样本类内离散度$\tilde{s_i}^2$和总类内离散度$\tilde{S_w}$

$$\tilde{s_i}^2 = \sum_{\tilde{y}\in y_i}(y - \tilde{m_i})^2, i = 1, 2$$

$$\tilde{S_w} = \tilde{s_1}^2 + \tilde{s_2}^2$$

- 计算投影空间上的分割阈值y_0

$$y_0 = \frac{\tilde{m_1} + \tilde{m_2}}{2}$$

- 根据决策规则进行分类

$$\begin{cases} Y > y_0 & \Rightarrow X \in \omega_1 \\ Y < y_0 & \Rightarrow X \in \omega_2 \end{cases}$$

4. **感知准则函数**

感知器算法是一种可以直接得到线性判别函数的线性分类方法，它必须要求数据集是线性可分的。

假设有一个包含 n 个样本的样本集合 $X = (x_1, x_2, \ldots, x_n)$, 其中$x_i = (x_1; x_2; \ldots; x_d)$ 我们想要找到一个线性判别函数$g(x) = w^T x + w_0$, 为了讨论方便，我们将样本 x 增加了一维常数，得到增广样本向量$y = (1; x_1; x_2; \ldots; x_n)$ ，则n 个样本的集合表示为$(y_1; y_2; \ldots; y_n)$ ， 增广权矢量表示为 $\alpha = (w_0; w_1; \ldots; w_d)$ ，得到新的判别函数$g(y) = \alpha^T y$

对应的决策规则就变为：$g(y) > 0$ ，则决策为第一类；$g(y) < 0$ 决策为第二类.

假设有一组样本$(y_1, y_2, \ldots, y_n)$，如果存在这样一个权矢量α，使得任何一个样本满足"属于第一类，$g(y) > 0$；属于第二类，$g(y) < 0$ "这样一个条件，那么我们就说这一组样本是线性可分的，即在样本特征空间中，至少会存在一个决策面可以将两类样本正确无误的分开，反之找不到一个决策面来区分的话，就说样本是线性不可分的。

5. **感知器算法**

感知器算法采用最直观的准则，即最小错分样本数。将错分样本到判别界面距离之和作为准则，称为感知器准则，表示如下：

$$J(\boldsymbol{a}) = \sum_{\boldsymbol{y}\in Y} -\boldsymbol{a}^T\boldsymbol{y}$$

为了求解感知器的准则函数，就是找到一个权矢量，使得惩罚函数最小化。我们使用机器学习中常用的梯度下降方法来迭代。惩罚函数对权矢量的梯度公式为：

$$\nabla J_p(\boldsymbol{a}) = \frac{\partial J_p(\boldsymbol{a})}{\partial \boldsymbol{a}} = \sum_{y\in Y}(-\boldsymbol{y})$$

利用梯度下降，我们使用如下公式， η_k为学习率

$$a_{k+1} = a_k - \eta_k \nabla J_p(a) = a_k + \eta_k \sum_{y\in Y} y$$

4.贝叶斯决策理论

推荐阅读材料

如何简单理解贝叶斯决策理论（Bayes Decision Theory）？

数学之美番外篇：平凡而又神奇的贝叶斯方法

怎么通俗易懂地解释贝叶斯网络和它的应用？

蒙提霍尔问题/三门问题（Monty Hall problem）

1. **贝叶斯基础**

有监督分类分为几何法与样本法

贝叶斯搭建模型的出发点不是为了对未知数据做预测，而是为了弄清楚并模拟数据产生的原理和机制。这里建议去了解一下蒙提霍尔问题, 虽然该问题的答案在逻辑上并不自相矛盾，但十分违反直觉，体现了条件概率的特点。

应用前提：

1.各类样本的概率分布是已知的

2.要决策的类别数量是一定的

即 设有C各类别，$w_i, i = 1, 2, , \ldots, c$,对应于$w_2$出现的先验概率$P(w_i)$及各类条件概率密度函数$P(\vec{x}|w_i)$是已知的。

在特征空间观察$\vec{x}, \vec{x} = [x_1, x_2, \ldots, x_n]^T$即特征空间的某一点$\vec{x}$ ∈哪一类?

2. **基于最小错误率的Bayes决策**

假设一个细胞有两种状态，正常w_1，不正常w_2，类别的状态是一个随机变量，已知正常状态的概率值$P(w_1)$,不正常概率$P(w_2)$,易得$P(w_1) + P(w_2) = 1$

$$g(x) = p(w_1) - p(w_2) > 0 \quad \vec{x} \in w_1$$
$$g(x) = p(w_1) - p(w_2) < 0 \quad \vec{x} \in w_2$$

在自然状态下观察到的类别条件概率分布

$P(\vec{x}|w_1)$是正常状态下特征观察$\vec{x}$的先验概率

$P(\vec{x}|w_2)$是异常状态下特征观察$\vec{x}$的先验概率

Bayes公式：

$$P\left(w_i \mid \vec{x}\right) = \frac{P\left(\vec{x} \mid w_i\right) P\left(w_i\right)}{\sum_{j=1}^{i} P\left(\vec{x} \mid w_j\right) P\left(w_j\right)} = \frac{P\left(x \mid w_i\right) P\left(w_i\right)}{P(x)}$$

判断：

$$P\left(w_i \mid \vec{x}\right) = max_{j=1,2} P\left(w_j \mid \vec{x}\right) \quad \vec{x} \in w_i$$

e.g.

$p(w_1) = 0.9, p(w_2) = 0.1, P(\vec{x}|w_1) = 0.2, P(\vec{x}|w_2) = 0.4$,试判断该细胞属于哪一类。

$$P(\vec{x}|w_1) \cdot p(w_1) = 0.2 \cdot 0.9 = 0.18$$
$$P(\vec{x}|w_2) \cdot p(w_2) = 0.1 \cdot 0.4 = 0.04 < 0.18$$
$$x \in w_1$$

3. **最小风险的Bayes决策**

最小错误率的贝叶斯决策可以找到正确率最高的分类结果，但实际问题中，不同类别分类错误的代价可能不同，例如将有毒蘑菇分类成无毒蘑菇的代价远远大于将无毒蘑菇分类成有毒蘑菇的代价。因此根据实际情况人为的引入分类错误风险，使得贝叶斯决策更加科学。

风险损失矩阵：λ_{ij}表示将实际为i的样本划分到j类别的风险系数，特别的当i等于j时，通常 $\lambda_{ij}=0$

将一个样本X将其划分到ω_1类别的风险的计算表达式为：

$$g_1(X)=R(\omega_1|X)=\lambda_{11}*P(\omega_1|X)+\lambda_{12}*P(\omega_2|X)$$

将一个样本X将其划分到ω_2类别的风险的计算表达式为：

$$g_2(X)=R(\omega_2|X)=\lambda_{21}*P(\omega_1|X)+\lambda_{22}*P(\omega_2|X)$$

判别依据：

$g_1(X)<g_2(X)$　则X判定为ω_1

$g_1(X)>g_2(X)$　则X判定为ω_2

特别地，当$\lambda_{12}=\lambda_{21}=1$也就是判别错误的风险一样大，那么转变为最小错误率的贝叶斯决策

同理决策面方程为对应判定为不同类别风险一样大时样本满足的方程：$g_1(X)=g_2(X)$

5.参数估计

推荐阅读材料：

详解最大似然估计（MLE）、最大后验概率估计（MAP），以及贝叶斯公式的理解

贝叶斯详解

估计理论是统计学和信号处理中的一个分支，主要是通过测量或经验数据来估计概率分布参数的数值。这些参数描述了实质情况或实际对象，它们能够回答估计函数提出的问题。例如，估计投票人总体中，给特定候选人投票的人的比例。这个比例是一个不可观测的参数，因为投票人总体很大；估计值建立在投票者的一个小的随机采样上。又如，雷达的目的是物体（飞机、船等）的定位。这种定位是通过分析收到的回声（回波）来实现的，定位提出的问题是"飞机在哪里？"为了回答这个问题，必须估计飞机到雷达之间的距离。如果雷达的绝对位置是已知的，那么飞机的绝对位置也是可以确定的。

估计理论的全部目的都是获取一个估计函数，最好是一个可以实现的估计函数。估计函数输入测量数据，输出相应参数的估计。我们通常希望估计函数能最优，一个最优的估计意味着所有的信息都被提取出来了；如果还有信息没有提取出来，那就意味着它不是最优的。

一般来说，求估计函数需要三步：

- 为了实现一个预测单个或者多个参数的所期望的估计器，首先需要确定系统的模型。这个模型需要将需要建模的过程以及不确定性和和噪声融合到一起，这个模型将描述参数应用领域的物理场景。
- 在确定模型之后，需要确定估计器的限制条件。这些限制条件可以通过如Cramér-Rao不等式这样的方法找到。
- 下一步，需要开发一个估计器或者应用一个已知的对于模型有效的估计器。这个估计器需要根据限制条件进行测试以确定它是否是最优估计器，如果是的话，它就是最好的估计器。
- 最后，在估计器上运行试验或者仿真以测试性能。

当实现一个估计器之后，实际的数据有可能证明推导出估计器的模型是不正确的，这样的话就需要重复上面的过程重新寻找估计器。不能实现的估计器需要抛弃，然后开始一个新的过程。总的来说，估计器根据实际测量的数据预测物理模型的参数。

1. **类条件概率如何获得**

先假定其具有某种确定的概率分布形式，再基于训练样本对概率分布的参数进行估计，具体地记关于类别w_i的类条件概率为$p(\vec{x} \mid w_i)$，假定$p(\vec{x} \mid w_i)$具有确定的形式且它的参数$\vec{\theta}_i$唯一确定，则任务完成。用训练集D估计$\vec{\theta}_i$可将$p(\vec{x} \mid w_i)$记为$p(\vec{x} \mid \vec{\theta}_i)$

2. **如何进行参数估计**

参数虽然未知，但客观存在固定值，通过优化似然函数等准则来确定估计值

参数是未观察到的随机变量，其本身也有分布，因此可假定参数服从一个先验分布，基于观测到的数据来计算参数的后验分布

3. **极大似然估计(MLE)**

极大似然估计，通俗理解来说，就是利用已知的样本结果信息，反推最具有可能（最大概率）导致这些样本结果出现的模型参数值！换句话说，极大似然估计提供了一种给定观察数据来评估模型参数的方法，即："模型已定，参数未知"。

令D_i表示训练集D中第i类样本的组合，假设这些样本独立并服从同一分布，则$\vec{\theta}_i$对于D_i的似然

$$P\left(D_i \mid \vec{\theta}_i\right)=\prod_{\vec{x} \in D_i}^{p} P(\vec{x} \mid \vec{\theta}_i)$$

$$LL(\vec{Q}_i)=\log P(D_i \mid \vec{Q}_i)=\sum_{\vec{x} \in D_i} \log P(\vec{x} \mid \vec{\theta}_i)$$

此时$\vec{\theta}_i$的极大似然估计值$\widehat{\vec{\theta}_i}$为

$$\widehat{\vec{\theta}_i}=\operatorname{argmax}_{\vec{\theta}} LL\left(\overrightarrow{\theta_i}\right)$$

连续情况：假设$P(\vec{x}|w_i)$~$N(\vec{H}_i, \vec{\sigma}_i^2)$

$$\vec{U}_i=\frac{1}{|D_i|} \sum_{\vec{x} \in D_i} \vec{x}$$

$$\vec{\sigma}_i=\frac{1}{|D_i|} \sum_{\vec{x} \in D_i}(\vec{x}-\widehat{\vec{H}_i}) \cdot(\vec{x}-\widehat{\vec{U}_i})^T$$

4. **朴素贝叶斯分类器**

朴素贝叶斯是一种构建分类器的简单方法。该分类器模型会给问题实例分配用特征值表示的类标签，类标签取自有限集合。它不是训练这种分类器的单一算法，而是一系列基于相同原理的算法：所有朴素贝叶斯分类器都假定样本每个特征与其他特征都不相关。举个例子，如果一种水果其具有红，圆，直径大概3英寸等特征，该水果可以被判定为是苹果。尽管这些特征相互依赖或者有些特征由其他特征决定，然而朴素贝叶斯分类器认为这些属性在判定该水果是否为苹果的概率分布上独立的。对于某些类型的概率模型，在监督式学习的样本集中能获取得非常好的分类效果。在许多实际应用中，朴素贝叶斯模型参数估计使用最大似然估计方法；换而言之，在不用到贝叶斯概率或者任何贝叶斯模型的情况下，朴素贝叶斯模型也能奏效。尽管是带着这些朴素思想和过于简单化的假设，但朴素贝叶斯分类器在很多复杂的现实情形中仍能够获取相当好的效果。2004年，一篇分析贝叶斯分类器问题的文章揭示了朴素贝叶斯分类器获取看上去不可思议的分类效果的若干理论上的原因。尽管如此，2006年有一篇文章详细比较了各种分类方法，发现更新的方法（如决策树和随机森林）的性能超过了贝叶斯分类器。朴素贝叶斯分类器的一个优势在于只需要根据少量的训练数据估计出必要的参数（变量的均值和方差）。由于变量独立假设，只需要估计各个变量的方法，而不需要确定整个协方差矩阵。

理论上，概率模型分类器是一个条件概率模型。$p(C|F_1, \ldots, F_n)$

独立的类别变量C有若干类别，条件依赖于若干特征变量$F_1F_1, F_2F_2, \ldots, F_n$。但问题在于如果特征数量$n$较大或者每个特征能取大量值时，基于概率模型列出概率表变得不现实。所以我们修改这个模型使之变得可行。 贝叶斯定理有以下式子：

$$p(C|F_1, \ldots, F_n) = \frac{p(C)\, p(F_1, \ldots, F_n|C)}{p(F_1, \ldots, F_n)}.$$

用朴素的语言可以表达为：

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

实际中，我们只关心分式中的分子部分，因为分母不依赖于C而且特征F_i的值是给定的，于是分母可以认为是一个常数。这样分子就等价于联合分布模型。

$$p(C, F_1, \ldots, F_n)$$

现在"朴素"的条件独立假设开始发挥作用:假设每个特征F_i对于其他特征F_j,$j \neq i$是条件独立的。这就意味着

$$p(F_i|C, F_j) = p(F_i|C)$$

对于$i \neq j$，这意味着上述假设下，类变量C的条件分布可以表达为：

$$p(C|F_1, \ldots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C)$$

其中Z(证据因子)是一个只依赖与$F_1, \ldots, F_n$等的缩放因子，当特征变量的值已知时是一个常数。 由于分解成所谓的类先验概率$p(C)$和独立概率分布$p(F_i|C)$，上述概率模型的可掌控性得到很大的提高。如果这是一个k分类问题，且每个$p(F_i|C = c)$可以表达为r个参数，于是相应的朴素贝叶斯模型有$(k-1) + nrk$个参数。实际应用中，通常取$k = 2$（二分类问题），$r = 1$(伯努利分布作为特征），因此模型的参数个数为$2n + 1$，其中n是二值分类特征的个数。

朴素贝叶斯的主要优点：

朴素贝叶斯模型发源于古典数学理论，有稳定的分类效率。对小规模的数据表现很好，能够处理多分类任务，适合增量式训练，尤其是数据量超出内存时，我们可以一批批的去增量训练。对缺失数据不太敏感，算法也比较简单，常用于文本分类。

朴素贝叶斯的主要缺点：

理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型给定输出类别的情况下，假设属性之间相互独立，这个假设在实际应用中往往是不成立的，在属性个数比较多或者属性之间相关性较大时，分类效果不好。而在属性相关性较小时，朴素贝叶斯性能最为良好。对于这一点，有半朴素贝叶斯之类的算法通过考虑部分关联性适度改进。需要知道先验概率，且先验概率很多时候取决于假设，假设的模型可以有很多种，因此在某些时候会由于假设的先验模型的原因导致预测效果不佳。由于我们是通过先验和数据来决定后验的概率从而决定分类，所以分类决策存在一定的错误率。对输入数据的表达形式很敏感。

以下为上课公式，以上为维基百科公式。

属性条件独立性假设

$$P(w_i \mid \vec{x}) = \frac{P(w_i) \cdot P(\vec{x} \mid w_i)}{P(\vec{x})} = \frac{P(w_i)}{P(\vec{x})} \prod_{j=1}^{d} P(x_j \mid w_i)$$

贝叶斯决策规则

$$P(w_i \mid \vec{x}) = \text{argmax}_{w_i \in w_j} P(w_i) \cdot \prod_{j=1}^{d} P(x_j \mid w_i)$$

5. **半朴素贝叶斯分类器**

为了降低贝叶斯公式中估计后验概率$p(c|x)$的困难，朴素贝叶斯分类器采用了特征条件独立性假设，但在现实任务中这个假设往往很难成立。于是，人们尝试对特征条件独立性假设进行一定程度的放松，由此产生了一类称为半朴素贝叶斯分类的学习方法。

半朴素贝叶斯分类器的基本思想：适当考虑一部分特征间的相互依赖信息，从而既不需进行完全联合概率计算，又不至于彻底忽略了比较强的特征依赖关系。独依赖估计（简称ODE）是半朴素贝叶斯分类器最常用的一种策略。“独依赖”就是假设每个特征在类别之外最多仅依赖于一个其他特征，即：

$$P(x|c_i) = \prod_{j=1}^{d} P(x_j|c_i, \text{pa}_j).$$

其中，pa_j为属性x_j所依赖的属性，称为x_j的父属性。此时，对每个特征x_j若其父特征pa_j已知，则可以使用以下公式：

$$P(x_j|c_i, \text{pa}_j) = \frac{P(x_j, c_i, \text{pa}_j)}{P(c_i, \text{pa}_j)}.$$

于是，问题的关键变成了如何确定每个属性的父属性。不同的做法产生了不同的独依赖分类器。

- SPODE（Super-Parent ODE）假设所有的属性都依赖于同一个属性，称为超父。
- TAN（Tree Augmented naive Bayes）则在最大带权生成树算法的基础上发展的一种方法。
- AODE（Averaged ODE）是一种集成学习的方法，尝试将每个属性作为超父来构建SPODE，与随机森林的方法有所相似。

6. **贝叶斯网络**

贝叶斯网络（Bayesian network），又称信念网络（belief network）或是有向无环图模型（directed acyclic graphical model），是一种概率图型模型，借由有向无环图（directed acyclic graphs, or DAGs）中得知一组随机变量$\{X_1, X_2, \ldots, X_n\}$及其$n$组条件概率分布（conditional probability distributions, or CPDs）的性质。举例而言，贝叶斯网络可用来表示疾病和其相关症状间的概率关系；倘若已知某种症状下，贝叶斯网络就可用来计算各种可能罹患疾病之发生概率。

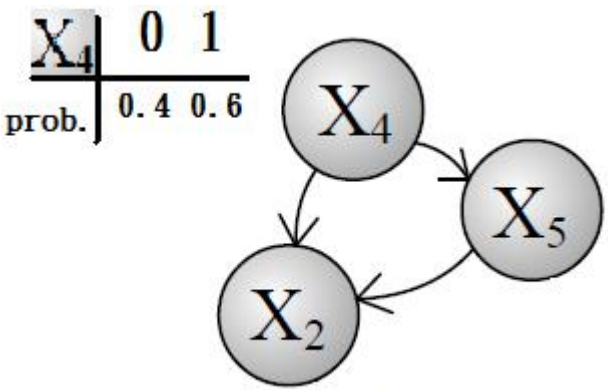
贝叶斯网是一种系统地描述随机变量之间关系的语言。构造贝叶斯网的主要目的是进行概率推理，即计算一些事件发生的概率。贝叶斯网是概率论与图论结合的产物，一方面用图论的语言直观揭示问题的结构，另一方面按概率论的原则对问题结构加以利用。许多经典多元概率模型都是贝叶斯网的特例：隐马尔科夫模型、卡
尔曼滤波器等。

一般而言，贝叶斯网络的有向无环图中的节点表示随机变量，它们可以是可观察到的变量，抑或是隐变量、未知参数等。连接两个节点的箭头代表此两个随机变量是具有因果关系或是非条件独立的；而两个节点间若没有箭头相互连接一起的情况就称其随机变量彼此间为条件独立。若两个节点间以一个单箭头连接在一起，表示其中一个节点是“因（parents）”，另一个是“果（descendants or children）”，两节点就会产生一个条件概率值。比方说，我们以X_i表示第i个节点，而X_i的“因”以P_i表示，X_i的“果”以C_i表示；图一就是一种典型的贝叶斯网络结构图，依照先前的定义，我们就可以轻易的从图一可以得知：

$$P_2 = \{X_4, X_5\}, C_2 = \emptyset, P_4 = \emptyset, C_4 = \{X_2, X_5\}, P_5 = \{X_4\}以及C_5 = \{X_2\}$$

大部分的情况下，贝叶斯网络适用在节点的性质是属于离散型的情况下，且依照$P(X_i|P_i)$此条件概率写出条件概率表（conditional probability table, or CPT），此条件概率表的每一行（row）列出所有可能发生的P_i，每一列（column）列出所有可能发生的X_i，且任一行的概率总和必为1。写出条件概率表后就很容易将事情给条理化，且轻易地得知此贝叶斯网络结构图中各节点间之因果关系；但是条件概

率表也有其缺点：若是节点X_i是由很多的“因”所造成的“果”，如此条件概率表就会变得在计算上既复杂又使用不便。下图为贝叶斯网络中某部分结构图之条件概率表。



X_4	0	1
0	0.32	0.68
1	0.54	0.46

X_5	X_4	X_2 0	1
0	0	1.0	0.0
0	1	0.2	0.8
1	0	0.91	0.09
1	1	0.02	0.98

7. **EM**

最大期望算法（Expectation-maximization algorithm，又译期望最大化算法）在统计中被用于寻找，依赖于不可观察的隐性变量的概率模型中，参数的最大似然估计。

在统计计算中，最大期望（EM）算法是在概率模型中寻找参数最大似然估计或者最大后验估计的算法，其中概率模型依赖于无法观测的隐变量。最大期望算法经常用在机器学习和计算机视觉的数据聚类（Data Clustering）领域。最大期望算法经过两个步骤交替进行计算，第一步是计算期望（E），利用对隐藏变量的现有估计值，计算其最大似然估计值；第二步是最大化（M），最大化在E步上求得的最大似然值来计算参数的值。M步上找到的参数估计值被用于下一个E步计算中，这个过程不断交替进行。

EM 算法的核心思想非常简单，分为两步：Expection-Step 和 Maximization-Step。E-Step 主要通过观察数据和现有模型来估计参数，然后用这个估计的参数值来计算似然函数的期望值；而 M-Step 是寻找似然函数最大化时对应的参数。由于算法会保证在每次迭代之后似然函数都会增加，所以函数最终会收敛。

EM算法可以保证收敛到一个稳定点，即EM算法是一定收敛的。

具体算法实践：

【机器学习】EM——期望最大

EM算法详解

6.近邻法

1. **最近邻法**

假设有c个类别，$w_1, w_2, \ldots, w_c$每类有表明类比的样本N_i个，$i = 1, 2, \ldots, c$，令w_i类的判别函数为

$$g_i(\vec{x}) = min_k||\vec{x} - \vec{x_i^k}||, k = 1, 2, \ldots, N_i$$

x_i^k中i表示w_i类，k表示w_i类N_i^k个样本中的第k个，决策规则为

$$g_j(\vec{x}) = min_i\ g_i(\vec{x}), i = 1, 2, \ldots, c$$
$$then \quad \vec{x} \in w_j$$

2. **K近邻法**

KNN是通过测量不同特征值之间的距离进行分类。它的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别，其中K通常是不大于20的整数。KNN算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

在KNN中，通过计算对象间距离来作为各个对象之间的非相似性指标，避免了对象之间的匹配问题，在这里距离一般使用欧氏距离或曼哈顿距离。

$$\text{曼哈顿距离: } d(x,y) = \sum_{k=1}^{n} |x_k - y_k| \quad q = 1$$

$$\text{欧式距离: } d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \quad q = 2$$

$$\text{切比雪夫距离: } d(x,y) = \max_{1 \le k \le p} |x_k - y_k| \quad q = \infty$$

$$\text{明氏距离: } d(x,y) = (\sum_{k=1}^{n} |x_k - y_k|)^{\frac{1}{q}}$$

在n个已知样本中，找出$\vec{x}$的k个近邻，设在N个样本中来自w_1类的样本有N_1个，...，来自w_c类的样本有N_c个，若$k_1, k_2, \ldots, k_c$分别是k个近邻中属于$w_1, w_2, \ldots w_c$类的样本数，则定义

$$g_i(\vec{x}) = k_i, i = 1, 2, \ldots, c$$

错误率：

$$p^* \le p \le 2p^*$$

3. **解决方法**

- 近邻法的快速算法

基本思想：把已知样本集分级划分成多个子集，形成一个树状结构，每个节点是一个子集，每个子集只用少量的样本来代表，通过把新样本按顺序与各节点进行比较来排除不可能包含最近邻的子集，只在最后的节点上才需要与每个样本进行比较。

具体做法：

分级：不相交的子集

第一阶段：样本集X的分级分解。

第二阶段：搜索

检验：$\vec{x}$的最近邻是否在X_p

- 剪辑近邻法

样本->设计集、测试集

改进点：在很多情况下，两类数据的分布可能会有一定的重叠，这时样本就不会完全可分，并且分错的样本还将会误导决策，使分类效果不佳，并且可能会使分类面的形状过于复杂。

基本思想：设法将交界区（阴影部分）的已知样本去掉，决策时就不会受到这些样本的影响，使近邻法的决策面更加接近最优分类面。

1. 两分剪辑近邻法

- 剪辑：利用已知样本集X^n中的样本进行预分类，并剪辑掉被错分的样本，留下的样本构成剪辑样本X^{NE}
- 利用剪辑样本集x^{NE}和近邻规则对$\vec{x}$进行分类

2. 重复剪辑近邻法

将样本集K^N 随机划分为$S \geq 3$ 个子集，$K_1, \ldots, K_S$

- 用最近邻法，以 $K_j(j=(i+1) mod\ S)$为参考集， 对 K_i 中的样本进行分类，其中i=$1, \ldots, S$
- 去掉步骤2中被错分类的样本;
- 用所有留下的全部样本的构成新的样本集K^{NT}。
- 如经k次迭代后，再没有样本被剪辑掉，则停止，用最后的K^{NE}作为剪辑样本集；否则转步骤1。

7.*聚类

1. **聚类**

聚类分析（英语：Cluster analysis）亦称为集群分析，是对于统计数据分析的一门技术，在许多领域受到广泛应用，包括机器学习，数据挖掘，模式识别，图像分析以及生物信息。聚类是把相似的对象通过静态分类的方法分成不同的组别或者更多的子集（subset），这样让在同一个子集中的成员对象都有相似的一些属性，常见的包括在坐标系中更加短的空间距离等。

2. **k-means(k均值)算法**

k-均值算法（英文：k-means clustering）源于信号处理中的一种向量量化方法，现在则更多地作为一种聚类分析方法流行于数据挖掘领域。k-均值聚类的目的是：把n个点（可以是样本的一次观察或一个实例）划分到k个聚类中，使得每个点都属于离他最近的均值（此即聚类中心）对应的聚类，以之作为聚类的标准。这个问题将归结为一个把数据空间划分为Voronoi cells的问题。

这个问题在计算上是NP困难的，不过存在高效的启发式算法。一般情况下，都使用效率比较高的启发式算法，它们能够快速收敛于一个局部最优解。这些算法通常类似于通过迭代优化方法处理高斯混合分布的最大期望算法（EM算法）。而且，它们都使用聚类中心来为数据建模；然而k-均值聚类倾向于在可比较的空间范围内寻找聚类，期望-最大化技术却允许聚类有不同的形状。

注意：k-均值聚类与k-近邻之间没有任何关系（后者是另一流行的机器学习技术）。

K-均值是一个迭代算法，假设我们想要将数据聚类成 n 个组，其方法为:

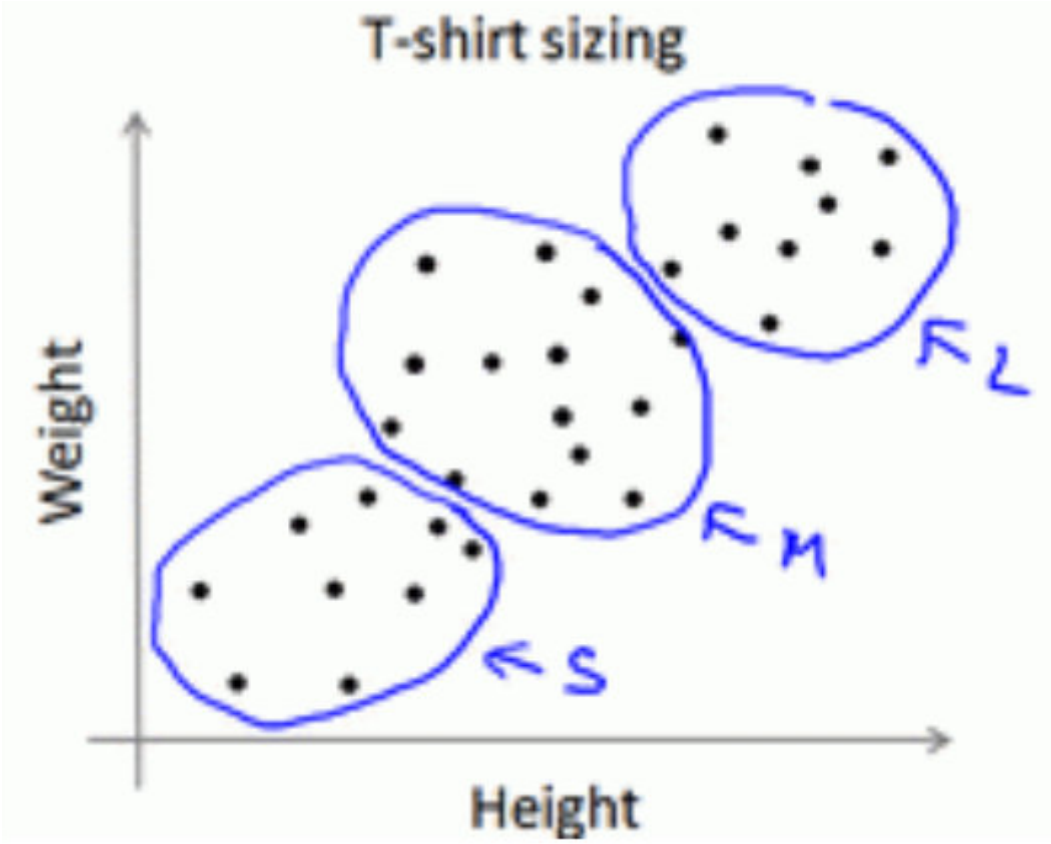
- 首先选择K个随机的点，称为聚类中心（cluster centroids）；
- 对于数据集中的每一个数据，按照距离K个中心点的距离，将其与距离最近的中心点关联起来，与同一个中心点关联的所有点聚成一类。
- 计算每一个组的平均值，将该组所关联的中心点移动到平均值的位置。
- 重复步骤，直至中心点不再变化。

用$u^1, u^2, \ldots, u^k$来表示聚类中心，用$c(1), c(2), \ldots, c(m)$来存储与第i个实例数据最近的聚类中心的索引，K-均值算法的伪代码如下：

```
Repeat {
    for i = 1 to m
    c(i) := index (form 1 to K) of cluster centroid closest to x(i)
    for k = 1 to K
    μk := average (mean) of points assigned to cluster k
}
```

算法分为两个步骤，第一个 for 循环是赋值步骤，即：对于每一个样例i，计算其应该属于的类。第二个 for 循环是聚类中心的移动，即：对于每一个类K，重新计算该类的质心。

K-均值算法也可以很便利地用于将数据分为许多不同组，即使在没有非常明显区分的组群的情况下也可以。下图所示的数据集包含身高和体重两项特征构成的，利用 K-均值算法将数据分为三类，用于帮助确定将要生产的T-恤衫的三种尺寸。



K-均值最小化问题，是要最小化所有的数据点与其所关联的聚类中心点之间的距离之和，因此 K-均值的代价函数（又称畸变函数 Distortion function）为：

$$J\left(c^{(1)},c^{(2)},\ldots,c^{(m)},u_1,\ldots,u_k\right)=\frac{1}{m}\sum_{i=1}^{m}\left\|X^{(1)}-u_{c^{(i)}}\right\|^2$$

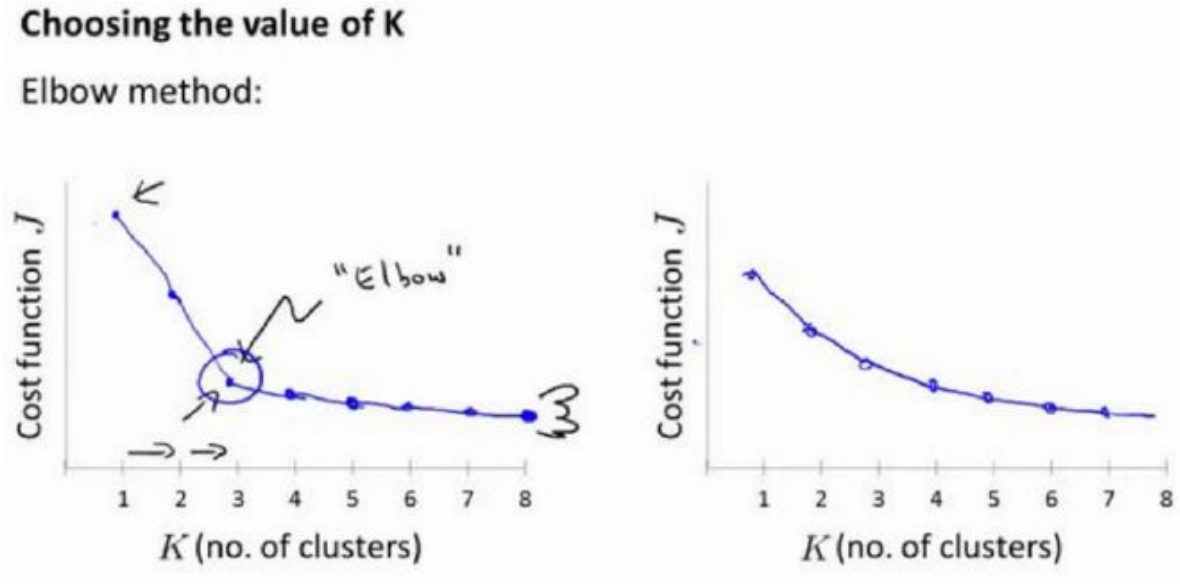
其中$u_{c^{(i)}}$代表与$x^{(i)}$最近的聚类中心点，优化目标便是找出使得代价函数最小的 $c^{(1)},c^{(2)},\ldots,c^{(m)}$和 $u_1,\ldots,u_k$。

在运行 K-均值算法的之前，我们首先要随机初始化所有的聚类中心点，下面介绍怎样做：

1. 我们应该选择$K<m$，即聚类中心点的个数要小于所有训练集实例的数量。
2. 随机选择K个训练实例，然后令K个聚类中心分别与这K个训练实例相等K-均值的一个问题在于，它有可能会停留在一个局部最小值处，而这取决于初始化的情况。

为了解决这个问题，我们通常需要多次运行 K-均值算法，每一次都重新进行随机初始化，最后再比较多次运行 K-均值的结果，选择代价函数最小的结果。这种方法在K较小的时候（2--10）还是可行的，**但是如果K较大，这么做也可能不会有明显地改善。**

没有所谓最好的选择聚类数的方法，通常是需要根据不同的问题，人工进行选择的。选择的时候思考我们运用 K-均值算法聚类的动机是什么。有一个可能会谈及的方法叫作**“肘部法则”。**关 于“肘部法则”，我们所需要做的是改变K值，也就是聚类类别数目的总数。我们用一个聚类来运行K均值聚类方法。这就意味着，所有的数据都会分到一个聚类里，然后计算成本函数或者计算畸变函数J。K代表聚类数字。



我们可能会得到一条类似于这样的曲线。像一个人的肘部。这就是“肘部法则”所做的，让我们来看这样一个图，看起来就好像有一个很清楚的肘在那儿。你会发现这种模式，它的畸变值会迅速下降，从 1 到 2，从 2 到 3 之后，你会在 3 的时候达到一个肘点。在此之后，畸变值就下降的非常慢，看起来就像使用 3 个聚类来进行聚类是正确的，**这是因为那个点是曲线的肘点，畸变值下降得很快，K = 3之后就下降得很慢，那么我们就选K = 3。**当你应用“肘部法则”的时候，如果你得到了一个像上面这样的图，那么这将是一种用来选择聚类个数的合理方法。

3. **KNN与K-means区别**

KNN	K-Means
1.KNN是分类算法 2.属于监督学习 3.训练数据集是带label的数据	1.K-Means是聚类算法 2.属于非监督学习 3.训练数据集是无label的数据，是杂乱无章的，经过聚类后变得有序，先无序，后有序。
没有明显的前期训练过程，属于memory based learning	有明显的前期训练过程
K的含义：一个样本x，对它进行分类，就从训练数据集中，在x附近找离它最近的K个数据点，这K个数据点，类别c占的个数最多，就把x的label设为c。	K的含义：K是人工固定好的数字，假设数据集合可以分为K个簇，那么就利用训练数据来训练出这K个分类。

相似点

都包含这样的过程，给定一个点，在数据集中找离它最近的点。即二者都用到了NN(Nears Neighbor)算法思想。

4. **K-Means优缺点及改进**

K-Means的主要优点有：

- 原理比较简单，实现也是很容易，收敛速度快。
- 聚类效果较优。
- 算法的可解释度比较强。
- 主要需要调参的参数仅仅是簇数k。

K-Means的主要缺点有：

- K值的选取不好把握(改进：可以通过在一开始给定一个适合的数值给k，通过一次K-means算法得到一次聚类中心。对于得到的聚类中心，根据得到的k个聚类的距离情况，合并距离最近的类，因此聚类中心数减小，当将其用于下次聚类时，相应的聚类数目也减小了，最终得到合适数目的聚类数。可以通过一个评判值E来确定聚类数得到一个合适的位置停下来，而不继续合并聚类中心。重复上述循环，直至评判函数收敛为止，最终得到较优聚类数的聚类结果)。
- 对于不是凸的数据集比较难收敛(改进：基于密度的聚类算法更加适合，比如DESCAN算法)
- 如果各隐含类别的数据不平衡，比如各隐含类别的数据量严重失衡，或者各隐含类别的方差不同，则聚类效果不佳。
- 采用迭代方法，得到的结果只是局部最优。
- 对噪音和异常点比较的敏感(改进1：离群点检测的LOF算法，通过去除离群点后再聚类，可以减少离群点和孤立点对于聚类效果的影响；改进2：改成求点的中位数，这种聚类方式即K-Mediods聚类（K中值))。
- 初始聚类中心的选择(改进1：k-means++;改进2：二分K-means)

优化k-means的建议：

- 减少聚类的数目K。因为，每个样本都要跟类中心计算距离。
- 减少样本的特征维度。比如说，通过PCA等进行降维。
- 考察其他的聚类算法，通过选取toy数据，去测试不同聚类算法的性能。
- hadoop集群，K-means算法是很容易进行并行计算的。
- 算法可能找到局部最优的聚类，而不是全局最优的聚类。使用改进的二分k-means算法。

二分k-means算法：首先将整个数据集看成一个簇，然后进行一次k-means（k=2）算法将该簇一分为二，并计算每个簇的误差平方和，选择平方和最大的簇迭代上述过程再次一分为二，直至簇数达到用户指定的k为止，此时可以达到的全局最优。

8.决策树

推荐阅读：

【机器学习】决策树（上）——ID3、C4.5、CART（非常详细）

机器学习实战（三）——决策树

1. **ID3**

ID3 算法是建立在奥卡姆剃刀（用较少的东西，同样可以做好事情）的基础上：越是小型的决策树越优于大的决策树。

从信息论的知识中我们知道：信息熵越大，从而样本纯度越低，。ID3 算法的核心思想就是以信息增益来度量特征选择，选择信息增益最大的特征进行分裂。算法采用自顶向下的贪婪搜索遍历可能的决策树空间（C4.5 也是贪婪搜索）。 其大致步骤为：

- 初始化特征集合和数据集合；
- 计算数据集合信息熵和所有特征的条件熵，选择信息增益最大的特征作为当前决策节点；
- 更新数据集合和特征集合（删除上一步使用的特征，并按照特征值来划分不同分支的数据集合）；
- 重复 2，3 两步，若子集值包含单一特征，则为分支叶子节点。

ID3 使用的分类标准是信息增益，它表示得知特征 A 的信息而使得样本集合不确定性减少的程度。

数据集的信息熵：

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

其中D_i表示集合 D 中属于第 k 类样本的样本子集。针对某个特征A，对于数据集 D 的条件熵 $H(D|A)$为：

$$
\begin{aligned}
H(D|A) &= \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) \\
&= -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \left(\sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} log_2 \frac{|D_{ik}|}{|D_i|} \right)
\end{aligned}
$$

其中 D_i 表示 D 中特征 A 取第i个值的样本子集， D_{ik}表示D_i中属于第 k 类的样本子集。信息增益 = 信息熵 - 条件熵：

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

信息增益越大表示使用特征A 来划分所获得的“纯度提升越大”。

ID3的缺点为

- ID3 没有剪枝策略，容易过拟合；
- 信息增益准则对可取值数目较多的特征有所偏好，类似“编号”的特征其信息增益接近于 1；
- 只能用于处理离散分布的特征；
- 没有考虑缺失值。

2. **C4.5**

C4.5 算法最大的特点是克服了 ID3 对特征数目的偏重这一缺点，引入信息增益率来作为分类标准。

C4.5 相对于 ID3 的缺点对应有以下改进方式：

- 引入悲观剪枝策略进行后剪枝；
- 引入信息增益率作为划分标准；
- 将连续特征离散化，假设 n 个样本的连续特征 A 有 m 个取值，C4.5 将其排序并取相邻两样本值的平均数共 m-1 个划分点，分别计算以该划分点作为二元分类点时的信息增益，并选择信息增益最大的点作为该连续特征的二元离散分类点；
- 对于缺失值的处理可以分为两个子问题：
- 问题一：在特征值缺失的情况下进行划分特征的选择？（即如何计算特征的信息增益率）
- 问题二：选定该划分特征，对于缺失该特征值的样本如何处理？（即到底把这个样本划分到哪个结点里）
- 针对问题一，C4.5 的做法是：对于具有缺失值特征，用没有缺失的样本子集所占比重来折算；
- 针对问题二，C4.5 的做法是：将样本同时划分到所有子节点，不过要调整样本的权重值，其实也就是以不同概率划分到不同节点中。

利用信息增益率可以克服信息增益的缺点，其公式为

$$
\begin{aligned}
Gain_{ratio}(D, A) &= \frac{Gain(D, A)}{H_A(D)} \\
H_A(D) &= -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|}
\end{aligned}
$$

$H_A(D)$称为特征A 的固有值。

这里需要注意，信息增益率对可取值较少的特征有所偏好（分母越小，整体越大），因此 C4.5 并不是直接用增益率最大的特征进行划分，而是使用一个启发式方法：先从候选划分特征中找到信息增益高于平均值的特征，再从中选择增益率最高的。

C4.5 采用的悲观剪枝方法，用递归的方式从低往上针对每一个非叶子节点，评估用一个最佳叶子节点去代替这课子树是否有益。如果剪枝后与剪枝前相比其错误率是保持或者下降，则这棵子树就可以被替换掉。C4.5 通过训练数据集上的错误分类数量来估算未知样本上的错误率。后剪枝决策树的欠拟合风险很小，泛化性能往往优于预剪枝决策树。但同时其训练时间会大的多。

C4.5的缺点为：

- 剪枝策略可以再优化；
- C4.5 用的是多叉树，用二叉树效率更高；
- C4.5 只能用于分类；
- C4.5 使用的熵模型拥有大量耗时的对数运算，连续值还有排序运算；
- C4.5 在构造树的过程中，对数值属性值需要按照其大小进行排序，从中选择一个分割点，所以只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时，程序无法运行。

3. **CART**

ID3 和 C4.5 虽然在对训练样本集的学习中可以尽可能多地挖掘信息，但是其生成的决策树分支、规模都比较大，CART 算法的二分法可以简化决策树的规模，提高生成决策树的效率。

CART 包含的基本过程有分裂，剪枝和树选择。

- **分裂：**分裂过程是一个二叉递归划分过程，其输入和预测特征既可以是连续型的也可以是离散型的，CART 没有停止准则，会一直生长下去；
- **剪枝：**采用**代价复杂度剪枝**，从最大树开始，每次选择训练数据熵对整体性能贡献最小的那个分裂节点作为下一个剪枝对象，直到只剩下根节点。CART 会产生一系列嵌套的剪枝树，需要从中选出一颗最优的决策树；
- **树选择：**用单独的测试集评估每棵剪枝树的预测性能（也可以用交叉验证）。

CART 在 C4.5 的基础上进行了很多提升。

- C4.5 为多叉树，运算速度慢，CART 为二叉树，运算速度快；
- C4.5 只能分类，CART 既可以分类也可以回归；
- CART 使用 Gini 系数作为变量的不纯度量，减少了大量的对数运算；
- CART 采用代理测试来估计缺失值，而 C4.5 以不同概率划分到不同节点中；
- CART 采用"基于代价复杂度剪枝"方法进行剪枝，而 C4.5 采用悲观剪枝方法。

熵模型拥有大量耗时的对数运算，基尼指数在简化模型的同时还保留了熵模型的优点。基尼指数代表了模型的不纯度，基尼系数越小，不纯度越低，特征越好。这和信息增益（率）正好相反。

$$
\begin{aligned}
Gini(D) &= \sum_{k=1}^{K} \frac{|C_k|}{|D|}(1 - \frac{|C_k|}{|D|}) \\
&= 1 - \sum_{k=1}^{K} (\frac{|C_k|}{|D|})^2 \\
Gini(D|A) &= \sum_{i=1}^{n} \frac{|D_i|}{|D|} Gini(D_i)
\end{aligned}
$$

其中 k代表类别。基尼指数反映了从**数据集中随机抽取两个样本，其类别标记不一致的概率。**因此基尼指数越小，则数据集纯度越高。基尼指数偏向于特征值较多的特征，类似信息增益。基尼指数可以用来度量任何不均匀分布，是介于 0~1 之间的数，0 是完全相等，1 是完全不相等，

此外，当 CART 为二分类，其表达式为：

$$
Gini(D|A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)
$$

可以看到在平方运算和二分类的情况下，其运算更加简单。当然其性能也与熵模型非常接近。

CART 的一大优势在于：无论训练数据集有多失衡，它都可以将其子冻消除不需要建模人员采取其他操作。CART 使用了一种先验机制，其作用相当于对类别进行加权。这种先验机制嵌入于 CART 算法判断分裂优劣的运算里，在 CART 默认的分类模式中，总是要计算每个节点关于根节点的类别频率的比值，这就相当于对数据自动重加权，对类别进行均衡。

4. **总结**

最后通过总结的方式对比下 ID3、C4.5 和 CART 三者之间的差异。

- **划分标准的差异：**ID3 使用信息增益偏向特征值多的特征，C4.5 使用信息增益率克服信息增益的缺点，偏向于特征值小的特征，CART 使用基尼指数克服 C4.5 需要求 log 的巨大计算量，偏向于特征值较多的特征。
- **使用场景的差异：**ID3 和 C4.5 都只能用于分类问题，CART 可以用于分类和回归问题；ID3 和 C4.5 是多叉树，速度较慢，CART 是二叉树，计算速度很快；
- **样本数据的差异：**ID3 只能处理离散数据且缺失值敏感，C4.5 和 CART 可以处理连续性数据且有多种方式处理缺失值；从样本量考虑的话，小样本建议 C4.5、大样本建议 CART。C4.5 处理过程中需对数据集进行多次扫描排序，处理成本耗时较高，而 CART 本身是一种大样本的统计方法，小样本处理下泛化误差较大；
- **样本特征的差异：**ID3 和 C4.5 层级之间只使用一次特征，CART 可多次重复使用特征；
- **剪枝策略的差异：**ID3 没有剪枝策略，C4.5 是通过悲观剪枝策略来修正树的准确性，而 CART 是通过代价复杂度剪枝。

9.概率图

阅读材料：

概率图模型

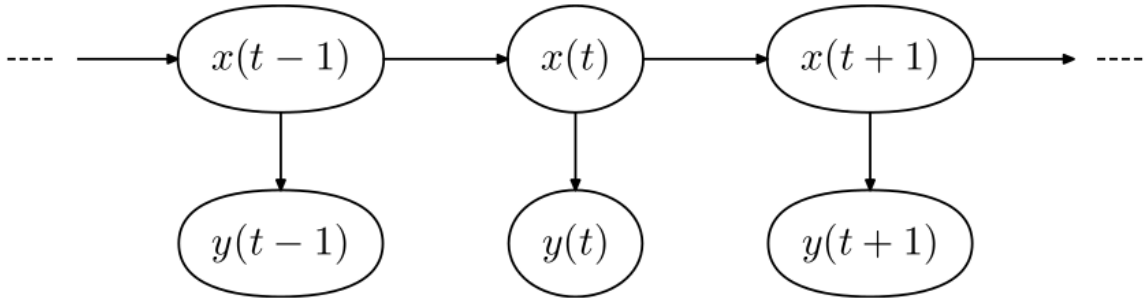
在概率论、统计学及机器学习中，概率图模型（Graphical Model）是用图论方法以表现数个独立随机变量之关联的一种建模法。一个p个节点的图中，节点i对应一个随机变量，记为X_i。概率图模型被广泛地应用于贝叶斯统计与机器学习中。

对于这个世界的任何物质，大到整个宇宙小到分子原子，其出现和演化的规律均存在一定的不确定性，而从另一个角度进行观察我们则可通过概率来描述。同样对于对于深度学习，我们可以理解为在给定样本下，我们通过神经网络学习出其内在的深层抽象特诊表示，进而对未知的样本进行估计和推断。其中样本的分布规律我们可以通过概率分布进行刻画，而推断的结果的不确定性我们也可以利用概率来表示。因此概率模型（probabilistic model） 为机器学习打开了一扇新的大门，其将学习任务转变为计算变量的概率分布。然而在实际情况中却是各个变量间存在显示或隐示的相互依赖，若我们直接基于训练数据去求解变量的联合概率分布这无论在时间复杂度还是空间复杂度上均是不可信的或不划算的。那么，我们如何有效（高效）紧凑且简洁的表示和计算它们的联合分布概率或条件概率呢，这里便引入了图的概念。通过充分利用变量间的依赖关系、或条件独立假设，我们可以大大简化参数求解的计算开销（若每个变量可能的取值数目为 k，变量数目为 n，而m表示单个条件概率分布最大的变量数目且$m << n$，通过图表示我们可以将联合概率求解的复杂度由$O(k^n)$减少为$O(k^m)$）。

隐马尔可夫模型（Hidden Markov Model；缩写：HMM）或称作隐性马尔可夫模型，是统计模型，它用来描述一个含有隐含未知参数的马尔可夫过程。其难点是从可观察的参数中确定该过程的隐含参数。然后利用这些参数来作进一步的分析，例如模式识别。

在正常的马尔可夫模型中，状态对于观察者来说是直接可见的。这样状态的转换概率便是全部的参数。而在隐马尔可夫模型中，状态并不是直接可见的，但受状态影响的某些变量则是可见的。每一个状态在可能输出的符号上都有一概率分布。因此输出符号的序列能够透露出状态序列的一些信息。

下边的图示强调了HMM的状态变迁。有时，明确的表示出模型的演化也是有用的，我们用$x(t1)$ 与 $x(t2)$ 来表达不同时刻 $t1$ 和 $t2$ 的状态。图中箭头方向则表示不同信息间的关系性，因此可以得知$x(t)$和$x(t-1)$有关，而$x(t-1)$又和$x(t-2)$有关。而每个$y(t)$只和$x(t)$有关，其中$x(t)$我们称为隐藏变量（hidden variable），是观察者无法得知的变量。隐性马尔可夫模型常被用来解决有未知条件的数学问题。假设隐藏状态的值对应到的空间有N个元素，也就是说在时间t时，隐藏状态会有N种可能。同样的，$t+1$也会有N种可能的值，所以从t到$t+1$间的关系会有N^2种可能。除了$x(t)$间的关系外，每组$x(t), y(t)$间也有对应的关系。若观察到的$y(t)$有M种可能的值，则从$x(t)$到$y(t)$的输出模型复杂度为$O(NM)$。如果$y(t)$是一个M维的向量，则从$x(t)$到$y(t)$的输出模型复杂度为$O(NM^2)$。在这个图中，每一个时间块$(x(t), y(t))$都可以向前或向后延伸。通常，时间的起点被设置为$t=0$ 或 $t=1$.



HMM有三个典型(canonical)问题:

预测(filter)：已知模型参数和某一特定输出序列，求最后时刻各个隐含状态的概率分布，即求$P(x(t) \mid y(1), \ldots, y(t))$. 通常使用前向算法解决.
平滑(smoothing)：已知模型参数和某一特定输出序列，求中间时刻各个隐含状态的概率分布，即求$P(x(k) \mid y(1), \ldots, y(t)), k < t$. 通常使用前向-后向算法解决.
解码(most likely explanation): 已知模型参数，寻找最可能的能产生某一特定输出序列的隐含状态的序列. 即求$P([x(1) \ldots x(t)] | [y(1) \ldots, y(t)])$,通常使用Viterbi算法解决.

e.g.

假设你有一个住得很远的朋友，他每天跟你打电话告诉你他那天做了什么。你的朋友仅仅对三种活动感兴趣：公园散步，购物以及清理房间。他选择做什么事情只凭天气。你对于他所住的地方的天气情况并不了解，但是你知道总的趋势。在他告诉你每天所做的事情基础上，你想要猜测他所在地的天气情况。

你认为天气的运行就像一个马尔可夫链。其有两个状态“雨”和“晴”，但是你无法直接观察它们，也就是说，它们对于你是隐藏的。每天，你的朋友有一定的概率进行下列活动：“散步”、“购物”、“清理”。因为你朋友告诉你他的活动，所以这些活动就是你的观察数据。这整个系统就是一个隐马尔可夫模型（HMM）。

你知道这个地区的总的天气趋势，并且平时知道你朋友会做的事情。也就是说这个隐马尔可夫模型的参数是已知的。你可以用程序语言（Python）写下来：

```
states = ('Rainy', 'Sunny')

 observations = ('walk', 'shop', 'clean')

 start_probability = {'Rainy': 0.6, 'Sunny': 0.4}

 transition_probability = {
    'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
    'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6},
    }

 emission_probability = {
    'Rainy' : {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},
    'Sunny' : {'walk': 0.6, 'shop': 0.3, 'clean': 0.1},
    }
```

在这些代码中,start_probability代表了你对于你朋友第一次给你打电话时的天气情况的不确定性（你知道的只是那个地方平均起来下雨多些）。在这里，这个特定的概率分布并非平衡的，平衡概率应该接近（在给定变迁概率的情况下）{'Rainy': 0.571, 'Sunny': 0.429}。 transition_probability 表示基于马尔可夫链模型的天气变迁，在这个例子中，如果今天下雨，那么明天天晴的概率只有30%。代码 emission_probability 表示了你朋友每天做某件事的概率。如果下雨，有50% 的概率他在清理房间；如果天晴，则有60%的概率他在外头散步。

10.非监督学习算法

阅读材料：

[清华大学对应章节PPT]

无监督学习是一类用于在数据中寻找模式的机器学习技术。无监督学习算法使用的输入数据都是没有标注过的，这意味着数据只给出了输入变量（自变量 X）而没有给出相应的输出变量（因变量）。在无监督学习中，算法本身将发掘数据中有趣的结构。人工智能研究的领军人物 Yann LeCun，解释道：无监督学习能够自己进行学习，而不需要被显式地告知他们所做的一切是否正确。

无监督学习主要是针对（有）监督学习和强化学习而言的，可以通过对输入的解释将强化学习、监督学习和无监督学习区分开来。我们可以通过描述它们学习的「东西」来说明它们的不同之处。

- 无监督学习：那东西就是这个样子的。（无监督学习算法学到了没有名字的事物之间的相似性，通过进一步的扩展，它们可以通过识别不寻常或者不相似的实例来发现相反或者执行异常检测）
- 监督学习：那个东西是一块「双层吉士汉堡」。（标签，联系名字和面孔……）这些监督学习算法学到了数据实体实例和它们的标签之间的关联；也就是说，监督学习算法需要有一个有标签的数据集。那些标签被用来「监督」和矫正算法，因为算法在预测标签的时候可能会做出错误的猜测。
- 强化学习：吃了这个东西，因为它味道蛮不错，而且可以让你活得更久。（基于短期和回报和长期回报的奖励，就相当于你摄入的卡路里或者你生存的时间一样。）强化学习可以被看做是在一个具有稀疏反馈的环境中的监督学习。

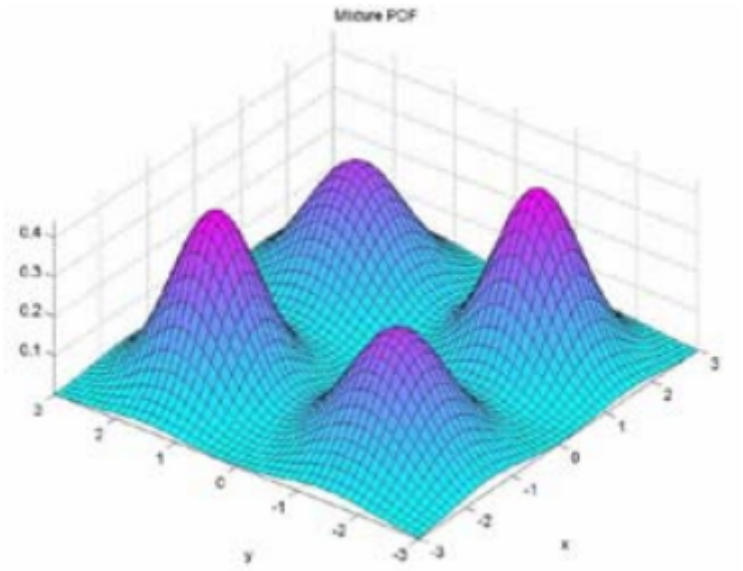
在监督学习中，系统试图从之前给出的示例中学习。（而在无监督学习中，系统试图从给定的示例中直接找到模式。）因此，如果数据集被标注过了，这就是一个监督学习问题；而如果数据没有被标注过，这就是一个无监督学习问题。

主要分为：

- 基于概率密度函数估计的直接聚类方法
- 基于样间相似性度量的间接聚类方法

1. 单峰子集的分离方法

基本思想：把特征空间分为若干个区域，在每个区域上混合概率密度函数是单峰的，每个单峰区域对应一个类别。如下图所示，$x=0$ 和$y=0$ 这两个超平面可以把(x,y)平面分成四个单峰区域。



2. 距离

各种距离见本笔记5.2章节

做标准化处理

$$令 \quad x_{ij}=\frac{x_i-\bar{x_j}}{\sqrt{s_{ji}}}$$

$$样本均值 \quad \bar{x_j}=\frac{1}{n}\Sigma_{i=1}^{n}x_{ij}$$

$$方差 \quad s_{jj}=\frac{1}{n-1}\Sigma_{i=1}^{n}(x_{ij}-\bar{x_j})^2$$

3. 相似系数

$$C_{ij}=\frac{\sum_{k=1}^{n}x_{ki}x_{kj}}{\sqrt{[(\sum_{k=1}^{n}x_{ki}^2)(\sum_{k=1}^{n}x_{kj}^2)]}}$$

$$C_{ij}=\frac{\sum_{k=1}^{n}\left(x_{ki}-\vec{x}_i\right)\left(x_{kj}-\vec{x}_j\right)}{\sqrt{[\sum_{k=1}^{n}(x_{ki}-\vec{x}_i)^2][\sum_{k=1}^{n}(x_{kj}-\vec{x}_j)^2]}}$$

4. 层次聚类法

层次聚类(Hierarchical Clustering)是聚类算法的一种，通过计算不同类别数据点间的相似度来创建一棵有层次的嵌套聚类树。在聚类树中，不同类别的原始数据点是树的最低层，树的顶层是一个聚类的根节点。

层次聚类的合并算法通过计算两类数据点间的相似性，对所有数据点中最为相似的两个数据点进行组合，并反复迭代这一过程。简单的说层次聚类的合并算法是通过计算每一个类别的数据点与所有数据点之间的距离来确定它们之间的相似性，距离越小，相似度越高。并将距离最近的两个数据点或类别进行组合，生成聚类树。

d_{ij}表示d_i与d_j间样本距离,$w_1,\dots,w_n$表示类,D_{kl}表示w_k与w_l的距离，初始$D_{kl}=d_{kl}$，距离矩阵$D(O)=d(ij)$

5. 最短距离法

$$D_{kl}=\min_{i=w_k,k=w_l}d_{ij}$$

聚类步骤：

- 计算D(O)
- 选择D(O)中的最小元素，记为D_{kl}，则将w_k与w_l合并成一个新类，记为w_m，即w_m = {w_k,w_l}
- 计算新类与任意类w之间的距离，递推公式为

$$\begin{aligned}D_{mj}=min_{i=w_m,j=w_j}d_{ij}&=min\{min_{i=w_m,j=w_j}d_{ij},min_{i=w_l,j=w_j}d_{ij}\}\\&=min\{D_{kj},D_{ij}\}\ 得到D(1)\end{aligned}$$

- 对$D(1)$重复上述过程，得到$D(2)$，以此类推，直到聚类完成。

e.g.

设有5个样本，特征1，2，6，8，11，试用最短距离法进行聚类

- $step1$ 1,2,6,8,11

$D(0)$	w_1	w_2	w_3	w_4	w_5
w_1	0				
w_2	2-1=1	0			
w_3	6-1=5	4	0		
w_4	8-1=7	6	2	0	
w_5	11-1=10	9	5	3	0

- $step2$ $D_{12}=1$最小，选取w_1,w_2合并为w_6，计算$D(1)$ 2,6,8,11

$D(1)$	w_6	w_3	w_4	w_5
w_6	0			
w_3	6-2=4	0		
w_4	8-2=6	2	0	
w_5	11-2=9	5	3	0

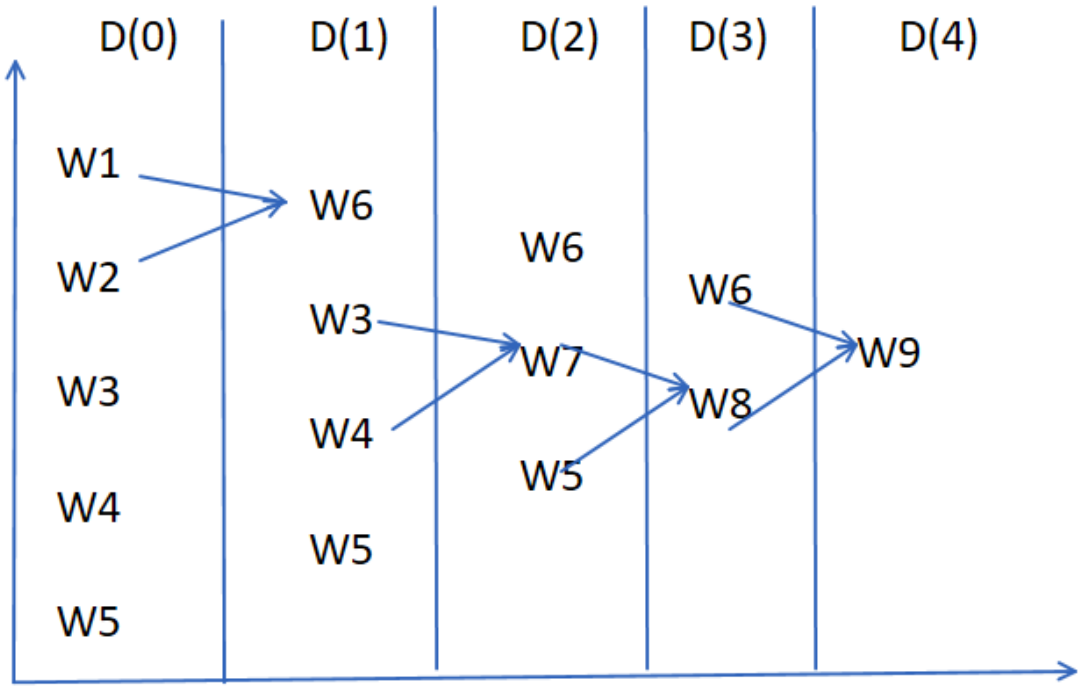
$step3$ $D_{34}=2$最小，选取w_3,w_4合并为w_7，计算$D(2)$ 2,8,11

$D(2)$	w_6	w_7	w_5
w_6	0		
w_7	8-2=6	0	
w_5	11-2=9	11-8=3	0

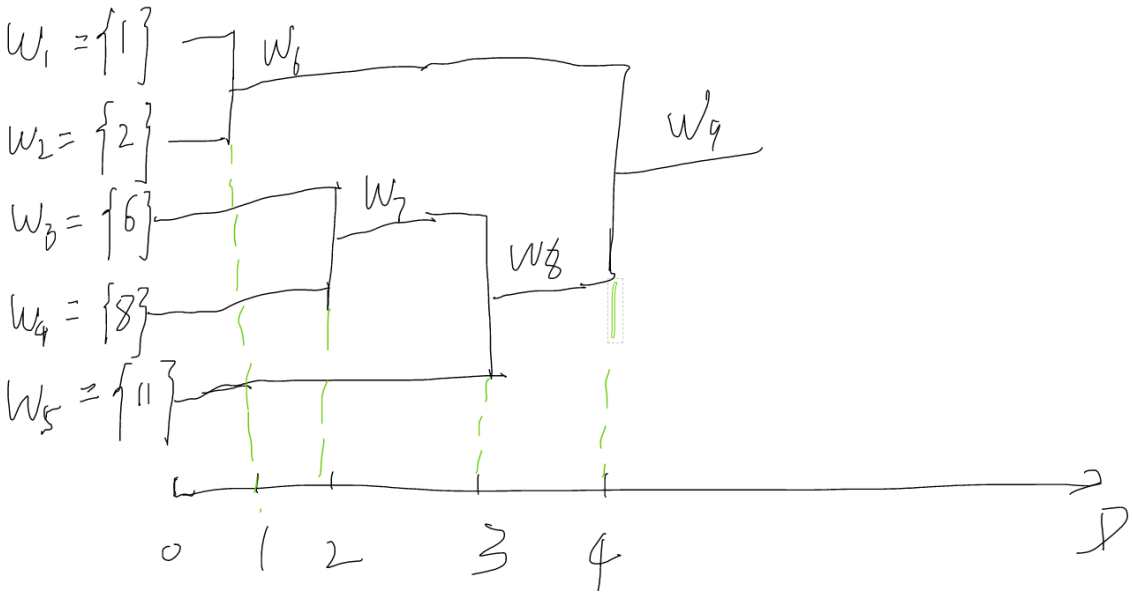
$step4$ $D_{57}=3$最小，选取w_5,w_7合并为w_8，计算$D(3)$ 2,8

$step5$ 合并w_6,w_8为w_9

计算过程



画出谱系图



6. **动态聚类法**

该算法的结果受取定的类数，聚类的初始中心为止影响，在实际中需测探不同的c值以及选择不同的聚类中心初始值以得到较好的结果。如果模式分布呈现类内团聚状，该算法能得到很好的聚类结果。

条件与约定:设待分类的模式的特征矢量为$x_1, x_2, \ldots, x_N$,选定类的数目为c。

算法思想:该方法取定 c个类别和选取 c个初始聚类中心，按最小距离原则将各模式分配到c类中的某一类，之后不断地计算类心和调整各模式的类别，最终使各模式到其判属类别中心的距离平方之和最小。

算法步骤

- 任选c个模式特征矢量作为初始聚类中心
- 将待分类的模式特征矢量集x_i中的模式按最小距离原则划分到c类中
- 重新计算各类类心

$$z_j^{(k)}=\frac{1}{n_j^{(k+1)}}\sum_{x_i\in\omega_j^{(k+1)}}x_i,(j=1,2,\ldots,c)$$

- 如果$z_j^{(k+1)}=z_j^{(k)}$则结束；否则返回第二步。

11.机器学习

1. **卷积神经网络**

卷积计算公式

$$\operatorname{out}\left(N_i,C_{\text{out}_j}\right)=\operatorname{bias}\left(C_{\text{out}_j}\right)+\sum_{k=0}^{C_{\text{in}}-1}\operatorname{weight}\left(C_{\text{out}_j},k\right)\star\operatorname{input}\left(N_i,k\right)$$

池化计算公式

$$\begin{aligned}\operatorname{out}\left(N_i,C_j,h,w\right)=&\max_{m=0,\ldots,kH-1}\max_{n=0,\ldots,kW-1}\\&\operatorname{input}\left(N_i,C_j,\operatorname{stride}\left[0\right]\times h+m,\operatorname{stride}\left[1\right]\times w+n\right)\end{aligned}$$

2. **课堂问答**

Q1：简单介绍一下神经网络的实际作用

A1：卷积神经网络CNN起源于对大脑的视觉皮层的研究，从20世纪80年代开始被用于图像识别。1958年对于动物的一系列视觉皮层研究发现，视觉皮层的神经元有一个小的局部接受野，这就意味着它们只对视觉的局部区域内的视觉刺激做出反应。此外，他们指出一些神经元作用于图片的水平方向，而另外一些神经元作用于其他方向。他们也注意到有些神经元有更大的接受野，他们作用于由多个低阶模式组成的复杂模式。这个发现可以推测出，高阶神经元基于相邻的低阶神经元的输出。这种强大的组织结构可以检测到视觉区域内的所有复杂模式。这些关于视觉皮层的研究影响了感知机。并逐步演变成我们所说的卷积神经网络。

Q2：为什么不只使用全连接网络?

A2：尽管深层的全连接网络对于例如MNIST数据集（28*28pixel）此类的较小图像表现良好，但是由于需要大量的参数，因此对于较大的图像无能为力。例如，一个100*100像素的图片有10000个像素。如果第一层只有1000个神经元（已经严重限制了传输到下一层的信息量），则意味着总共有10000000个连接，而这只是第一层。通过在全连接网络之前加入卷积神经网络，有效的解决了这个问题。

Q3：Dropout的具体作用是什么

A3：Dropout的具体作用是实现训练过程中对指定概率的神经元执行遮盖操作，例如指定25%的概率，那么每次只有75%的神经元参与训练。该操作一定程度防止了过拟合现象的发生，同时也防止了某些神经元逐步成为永远得不到更新的（死的）神经元。

3. **强化学习**

强化学习（英语：Reinforcement learning，简称RL）是机器学习中的一个领域，强调如何基于环境而行动，以取得最大化的预期利益。强化学习是除了监督学习和非监督学习之外的第三种基本的机器学习方法。与监督学习不同的是，强化学习不需要带标签的输入输出对，同时也无需对非最优解的精确地纠正。其关注点在于寻找探索（对未知领域的）和利用（对已有知识的）的平衡，强化学习中的“探索-利用”的交换，在多臂老虎机问题和有限MDP中研究得最多。

其灵感来源于心理学中的行为主义理论，即有机体如何在环境给予的奖励或惩罚的刺激下，逐步形成对刺激的预期，产生能获得最大利益的习惯性行为。这个方法具有普适性，因此在其他许多领域都有研究，例如博弈论、控制论、运筹学、信息论、仿真优化、多智能体系统、群体智能、统计学以及遗传算法。在运筹学和控制理论研究的语境下，强化学习被称作"近似动态规划"（approximate dynamic programming，ADP）。在最优控制理论中也有研究这个问题，虽然大部分的研究是关于最优解的存在和特性，并非是学习或者近似方面。在经济学和博弈论中，强化学习被用来解释在有限理性的条件下如何出现平衡。

在机器学习问题中，环境通常被抽象为马尔可夫决策过程（Markov decision processes，MDP），因为很多强化学习算法在这种假设下才能使用动态规划的方法。传统的动态规划方法和强化学习算法的主要区别是，后者不需要关于MDP的知识，而且针对无法找到确切方法的大规模MDP。

4. **集成学习**

在统计学和机器学习中，集成学习方法使用多种学习算法来获得比单独使用任何单独的学习算法更好的预测性能。不像统计力学中的系综通常是无限的，机器学习集合仅由一组具体的有限的可替代模型组成，但通常允许在这些可替代方案中存在更灵活的结构。

监督学习算法通常被描述为执行搜索假设空间的任务以找到合适的假设，该假设将对特定问题做出良好预测。即使假设空间包含非常适合特定问题的假设，也可能很难找到一个很好的假设。集成学习结合多个假设，形成一个（希望）更好的假设。术语集成通常保留用于使用相同基础学习器生成多个假设的方法。多分类器系统的更广泛术语还包括由非相同基础学习器得到的假设的结合。这种方法和现象也被另一个术语"群智"所描述，该术语来自多个DREAM生物医学数据科学挑战。

评估集成学习的预测通常需要比评估单个模型的预测更多的计算，因此集成可以被认为是通过执行大量额外计算来补偿差的学习算法的方式。诸如决策树之类的快速算法通常用于集合方法（如随机森林），尽管较慢的算法也可以从集成方法中受益。通过类比，集成技术也已用于无监督学习场景中，如共识聚类或异常检测。

集成学习本身是一种监督学习算法，因为它可以被训练然后用于进行预测。因此，训练后的集成模型代表了一个假设，但这个假设不一定被包含在构建它的模型的假设空间内。因此，可以证明集成学习在它们可以表示的功能方面具有更大的灵活性。理论上，这种灵活性使他们能够比单一模型更多地过拟合训练数据，但在实践中，一些集成算法（如Bagging算法）倾向于减少对训练数据过拟合相关的问题。

根据经验，当模型之间存在显著差异时，集成往往会产生更好的结果。因此，许多集成方法试图促进它们组合的模型之间的多样性。[6][7]尽管可能不是直观的，更随机的算法（如随机决策树）可用于产生比非常有意识的算法（如熵减少决策树）更强大的集成模型。然而，使用各种强大的学习算法已被证明是比使用试图愚弄模型以促进多样性的技术更有效。

5. **半监督学习**

半监督学习属于无监督学习（没有任何标记的训练数据）和监督学习（完全标记的训练数据）之间。许多机器学习研究人员发现，将未标记数据与少量标记数据结合使用可以显着提高学习准确性。对于学习问题的标记数据的获取通常需要熟练的人类代理（例如转录音频片段）或物理实验（例如，确定蛋白质的3D结构或确定在特定位置处是否存在油）。因此与标签处理相关的成本可能使得完全标注的训练集不可行，而获取未标记的数据相对便宜。在这种情况下，半监督学习可能具有很大的实用价值。半监督学习对机器学习也是理论上的兴趣，也是人类学习的典范。

6. **无监督学习**

无监督学习（英语：unsupervised learning）是机器学习的一种方法，没有给定事先标记过的训练示例，自动对输入的资料进行分类或分群。无监督学习的主要运用包含：聚类分析（cluster analysis）、关系规则（association rule）、维度缩减（dimensionality reduce）。它是监督式学习和强化学习等策略之外的一种选择。

一个常见的无监督学习是数据聚类。在人工神經网络中，生成对抗网络（GAN）、自组织映射（SOM）和适应性共振理论（ART）则是最常用的非监督式学习。ART模型允许集群的个数可随着问题的大小而变动，并让用户控制成员和同一个集群之间的相似度分数，其方式为透过一个由用户自定而被称为警觉参数的常量。ART也用于模式识别，如自动目标识别和数字信号处理。第一个版本为"ART1"，是由卡本特和葛罗斯柏格所发展的。