ML

决策树的计算

决策树（Decision Tree）是一种常用的机器学习方法，用于分类和回归任务。它的核心思想是基于特征值对数据集进行分割，形成树形结构以做出决策。构建决策树通常包括以下几个步骤：

1. **选择最佳分割特征**：根据某种准则（如信息增益、基尼不纯度）从当前数据集的特征中选择一个作为节点的分割特征。
2. **创建分支**：根据选定的特征分割数据集，为每个分割出的子集创建树的分支。
3. **递归构建子树**：对每个分支重复步骤1和2，直到满足某个停止条件（如达到最大深度、节点包含的样本数量小于最小样本数等）。
4. **剪枝**：为了防止过拟合，可能需要对树进行剪枝，删除一些分支。

下面是一些基本的公式：

- **信息增益**（用于ID3算法）：$\text{信息增益}(D, A) = \text{信息熵}(D) - \sum_{v \in A} \frac{|D^v|}{|D|} \text{信息熵}(D^v)$。其中，$D$是数据集，$A$是特征，$D^v$是特征$A$上值为$v$的子集。
- **基尼不纯度**（用于CART算法）：$\text{基尼不纯度}(D) = 1 - \sum_{k=1}^{K} p_k^2$。其中，$p_k$是数据集D中第$k$类样本所占的比例
- **信息熵**:$\text{信息熵}(D) = -\sum_{k=1}^{K} p_k \log_2 p_k$。

展示一个详细的决策树建立过程需要选择一个简单的数据集作为例子。我们可以使用一个虚构的、简单的数据集来说明这个过程。假设我们有以下数据集，用于预测一个人是否会去郊游（"Yes"或"No"）：

天气	温度	湿度	风速	郊游?
晴朗	热	高	弱	No
晴朗	热	高	强	No
阴天	热	高	弱	Yes
雨天	适中	高	弱	Yes
雨天	冷	正常	弱	Yes
雨天	冷	正常	强	No
阴天	冷	正常	强	Yes
晴朗	适中	高	弱	No
晴朗	冷	正常	弱	Yes
雨天	适中	正常	弱	Yes
晴朗	适中	正常	强	Yes
阴天	适中	高	强	Yes
阴天	热	正常	弱	Yes
雨天	适中	高	强	No

接下来，我将展示如何使用信息增益（Information Gain）构建决策树的过程。首先，我们需要计算整个数据集的信息熵。然后，对于每个特征，我们计算使用该特征进行分割后的信息增益，以决定哪个特征作为根节点。之后，我们递归地对每个子集重复这个过程，直到满足某个终止条件（例如，所有实例属于同一类别，或者达到预设的树的深度）。

首先，我将计算整个数据集的信息熵。

整个数据集的信息熵为约 0.94。

整个系统有14条数据，其中yes为9条，no为5条，所以信息熵等于$-(\frac{9}{14}\log_2^{\frac{9}{14}}+\frac{5}{14}\log_2^{\frac{5}{14}})=0.94$

这表示在没有任何信息的情况下，我们对数据集中的结果的不确定性程度。

接下来，我们需要为每个特征计算信息增益。信息增益是整个数据集的信息熵与给定特征分割后数据集的加权平均信息熵之差。对于每个特征，我们将数据集分割为几个子集，每个子集包含该特征的一个唯一值。然后，我们计算每个子集的信息熵，并求得加权平均值。

我们将首先计算“天气”特征的信息增益。在这个数据集中，“天气”特征有三个可能的值：晴朗（编码为 0），阴天（编码为 1），和雨天（编码为 2）。我们将根据这些值来分割数据集，并计算每个子集的信息熵。

对于每个特征，我们计算得到的信息增益如下：

- 天气特征的信息增益约为 0.247
- 温度特征的信息增益约为 0.029
- 湿度特征的信息增益约为 0.152
- 风速特征的信息增益约为 0.048

例如计算天气特征的信息增益：天气特征有三个值，将数据集划分为三部分。其中晴朗为5条，阴天为4条，雨天为5条。再在每个划分的子集里计算信息熵。根据公式
$0.94-\frac{5}{14}(-\frac{3}{5}\log_2^{\frac{3}{5}}-\frac{2}{5}\log_2^{\frac{2}{5}})-\frac{4}{14}*0-\frac{5}{14}(-\frac{3}{5}\log_2^{\frac{3}{5}}-\frac{2}{5}\log_2^{\frac{2}{5}})=0.247$

在这些特征中，"天气" 特征的信息增益最高。因此，我们将首先使用“天气”特征来分割数据集。这意味着在决策树的顶层，“天气”将作为决策节点。

接下来，我们将数据集分为三个子集：晴朗天气的子集，阴天的子集，和雨天的子集。然后，我们需要对每个子集重复这个过程，计算它们的信息增益，直到满足停止条件。

为了简化演示，我将仅展示对于“晴朗”这一分支的下一步计算。其他分支（“阴天”和“雨天”）将以类似的方式处理。

对于“晴朗”天气条件下的子集，我们计算得到的信息增益如下：

- 温度特征的信息增益约为 0.5710.571
- 湿度特征的信息增益约为 0.9710.971
- 风速特征的信息增益约为 0.0200.020

在这些特征中，"湿度" 特征的信息增益最高。因此，在“晴朗”分支下，我们将使用“湿度”作为下一个决策节点。

这个过程将继续进行，每次选择具有最高信息增益的特征进行分割，直到满足停止条件。这可能是所有实例属于同一类别，或者达到预设的树的深度等。

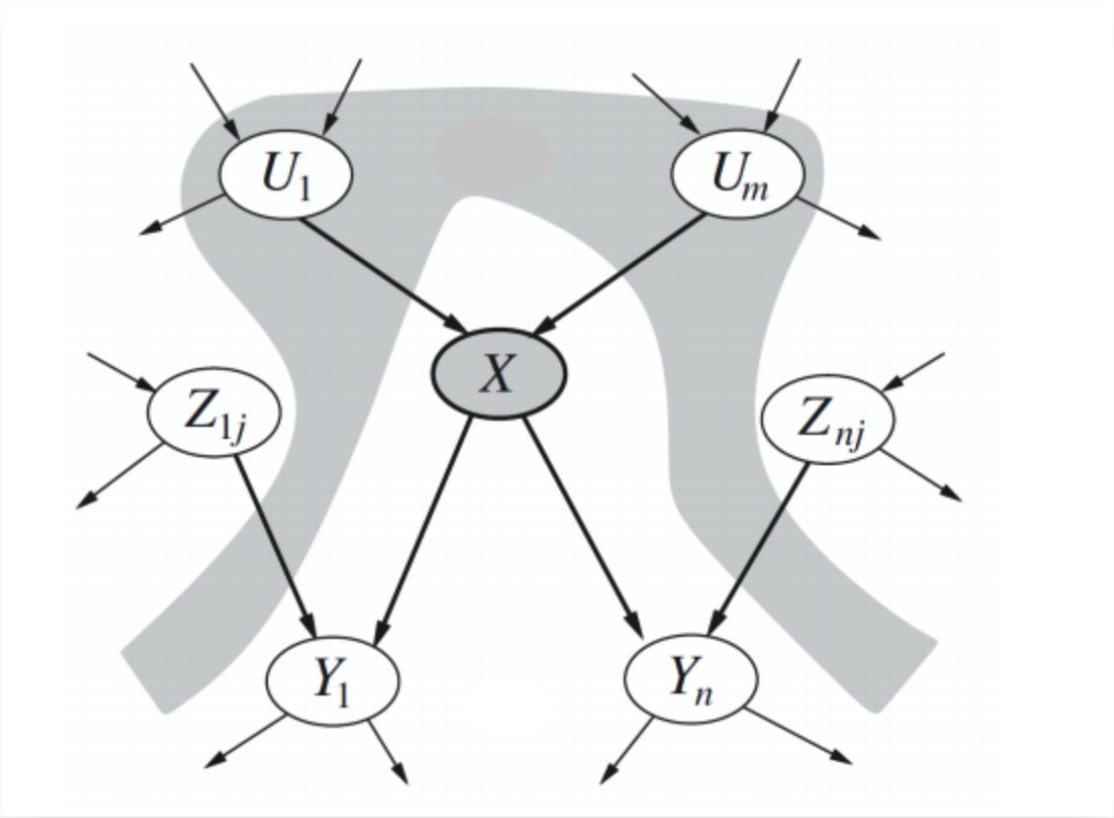
知道什么模型有监督无监督

在机器学习中，根据是否使用带有标签的数据进行训练，模型可以分为有监督学习模型和无监督学习模型：

1. **有监督学习（Supervised Learning）**：在这种类型的学习中，模型是用带有标签的数据进行训练的。这意味着每个训练样本都是带有正确答案（标签）的。模型的目的是学会根据输入数据预测标签。有监督学习的典型应用包括分类（如判断电子邮件是否为垃圾邮件）和回归（如预测房价）。
    - **举例**：线性回归（Linear Regression）、逻辑回归（Logistic Regression）、支持向量机（SVM）、决策树（Decision Trees）和神经网络（Neural Networks）。
2. **无监督学习（Unsupervised Learning）**：在这种类型的学习中，模型是用不带标签的数据进行训练的。因为没有标签，所以模型的目的是探索数据的结构，识别其中的模式。无监督学习通常用于聚类（如市场细分）和关联（如购物篮分析）。
    - **举例**：K-均值聚类（K-means Clustering）、层次聚类（Hierarchical Clustering）、主成分分析（PCA）和自编码器（Autoencoders）。

贝叶斯网络的画图和表征和条件独立

贝叶斯网络（bayesian network）是一种概率图形模型，它在图形模型中显式捕获已知的有向边的条件依赖性，它通过有向无环图(DAG)表示一组变量及其条件依赖关系。
贝叶斯网络非常适合用于获取已发生的事件并预测几种可能的已知原因中的任意一种。例如，贝叶斯网络可以表示疾病和症状之间的概率关系。给定症状，该网络可用于计算各种疾病存在的概率。



假设有一个包含五个变量的网络：A、B、C、D 和 E。这些变量之间的关系如下所示：

- A 影响 B 和 C
- B 影响 D
- C 影响 D 和 E

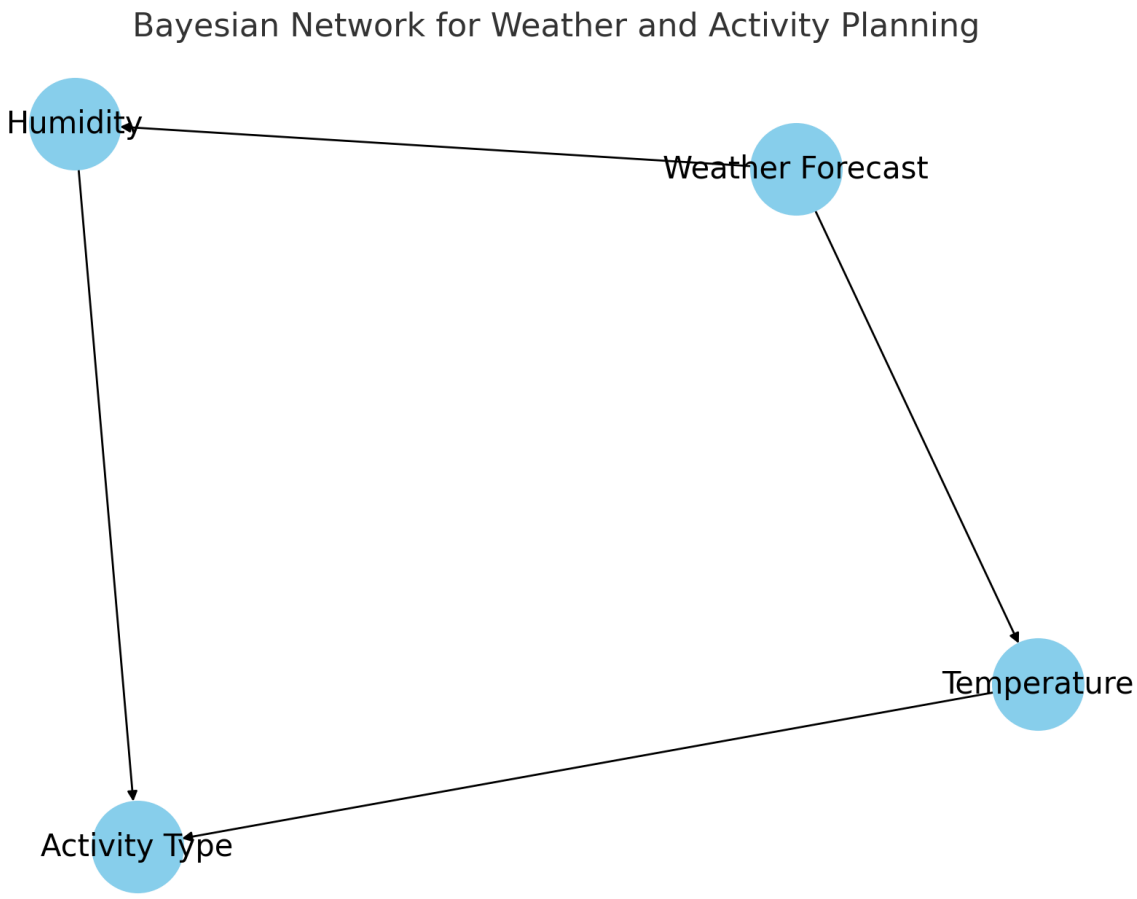
这个网络可以用以下结构表示：

- A → B → D
- A → C → D
- C → E

对应的条件概率表会包括以下内容：

- P(A)
- P(B | A)
- P(C | A)
- P(D | B, C)
- P(E | C)

示例



示例：天气与活动安排

假设你想决定周末是否进行户外活动。这个决定取决于几个因素：天气预报（晴朗或阴天）、温度（高或低）和湿度（高或低）。基于这些因素，你将决定进行户外活动还是室内活动。

贝叶斯网络结构

1. **节点**：天气预报、温度、湿度、活动类型。
2. **边**：天气预报影响温度和湿度；温度和湿度共同影响活动类型。

条件概率表

- **天气预报**：晴朗（70%），阴天（30%）。
- 温度 | 天气预报：
  - 如果晴朗：高（80%），低（20%）。
  - 如果阴天：高（30%），低（70%）。
- 湿度 | 天气预报：
  - 如果晴朗：高（30%），低（70%）。
  - 如果阴天：高（70%），低（30%）。
- 活动类型 | 温度，湿度：
  - 如果温度高且湿度低：户外（90%），室内（10%）。
  - 如果温度高且湿度高：户外（60%），室内（40%）。
  - 如果温度低且湿度低：户外（50%），室内（50%）。
  - 如果温度低且湿度高：户外（30%），室内（70%）。

设（T）代表温度（高温或低温），（H）代表湿度（高湿或低湿），（A）代表活动类型（户外或室内）。已知天气预报（W）为晴朗。我们要计算的是$P(A = \text{户外}|W = \text{晴朗})$，即在晴朗天气下进行户外活动的概率。

这个概率可以通过以下步骤计算：

1. 考虑所有可能的温度和湿度组合，我们有：

$$P(A = \text{户外}|W = \text{晴朗}) = \sum_{T,H} P(A = \text{户外}|T, H, W = \text{晴朗}) \cdot P(T, H|W = \text{晴朗})$$

2. 根据贝叶斯网络的结构，我们知道温度和湿度是条件独立的给定天气预报，因此：

$$P(T, H|W = \text{晴朗}) = P(T|W = \text{晴朗}) \cdot P(H|W = \text{晴朗})$$

3. 结合所有条件概率，我们得到：

$$P(A = \text{户外}|W = \text{晴朗}) = \sum_{T,H} P(A = \text{户外}|T, H) \cdot P(T|W = \text{晴朗}) \cdot P(H|W = \text{晴朗})$$

其中，$P(A = \text{户外}|T, H)$、$P(T|W = \text{晴朗})$和$P(H|W = \text{晴朗})$可以从条件概率表中得到。通过计算所有可能的（T）和（H）组合（高温低湿、高温高湿、低温低湿、低温高湿）的概率，然后将它们相加，我们就可以得到在晴朗天气下进行户外活动的总体概率。

```
P_Outdoor_Sunny = (P_Temp_High_Given_Sunny * P_Humidity_Low_Given_Sunny *
P_Outdoor_Given_HighTemp_LowHumidity +
                   P_Temp_High_Given_Sunny * P_Humidity_High_Given_Sunny *
P_Outdoor_Given_HighTemp_HighHumidity +
                   P_Temp_Low_Given_Sunny * P_Humidity_Low_Given_Sunny *
P_Outdoor_Given_LowTemp_LowHumidity +
                   P_Temp_Low_Given_Sunny * P_Humidity_High_Given_Sunny *
P_Outdoor_Given_LowTemp_HighHumidity)

P_Outdoor_Sunny
```

Result

```
0.736
```

朴素贝叶斯

朴素贝叶斯（Naive Bayes）分类器是一种基于贝叶斯定理的简单概率分类器。它假设特征之间相互独立。给定类别 y 和一组特征$(X_1, X_2, \ldots, X_n)$，朴素贝叶斯分类器通过以下公式来计算后验概率$P(y|X_1, X_2, \ldots, X_n)$：

$$P(y|X_1, X_2, \ldots, X_n) = \frac{P(y)P(X_1, X_2, \ldots, X_n|y)}{P(X_1, X_2, \ldots, X_n)}$$

根据朴素贝叶斯的假设，可以将 $P(X_1, X_2, \ldots, X_n|y)$ 分解为各个特征的条件概率乘积：

$$P(X_1, X_2, \ldots, X_n|y) = P(X_1|y)P(X_2|y)\ldots P(X_n|y)$$

因此，朴素贝叶斯分类器的公式可以简化为：

$$P(y|X_1, X_2, \ldots, X_n) = \frac{P(y)\prod_{i=1}^{n} P(X_i|y)}{P(X_1, X_2, \ldots, X_n)}$$

在实际应用中，通常只关心分母 $P(X_1, X_2, \ldots, X_n)$是常数，因为它对于所有类别是相同的，因此可以忽略不计。需要关注的是分子中的$P(y)\prod_{i=1}^{n} P(X_i|y)$ 的值，用来预测y的值。

【ML】贝叶斯网络(Bayesian Network)-CSDN博客

条件独立和d-separation（贝叶斯网络里的）

1. 条件独立：
   条件独立是指在给定某些条件的情况下，两个随机变量是独立的。在数学上，假设有三个随机变量 X, Y 和 Z，如果 X 和 Y 在给定 Z 的条件下是独立的，可以表示为：

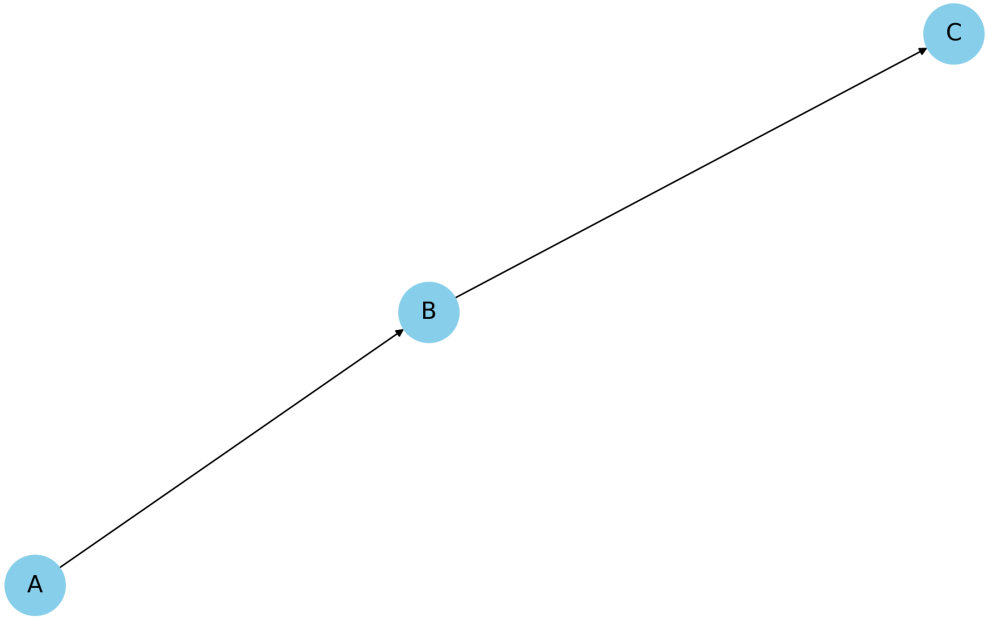
   $$P(X, Y \mid Z) = P(X \mid Z) \cdot P(Y \mid Z)$$

   或等价地表示为：

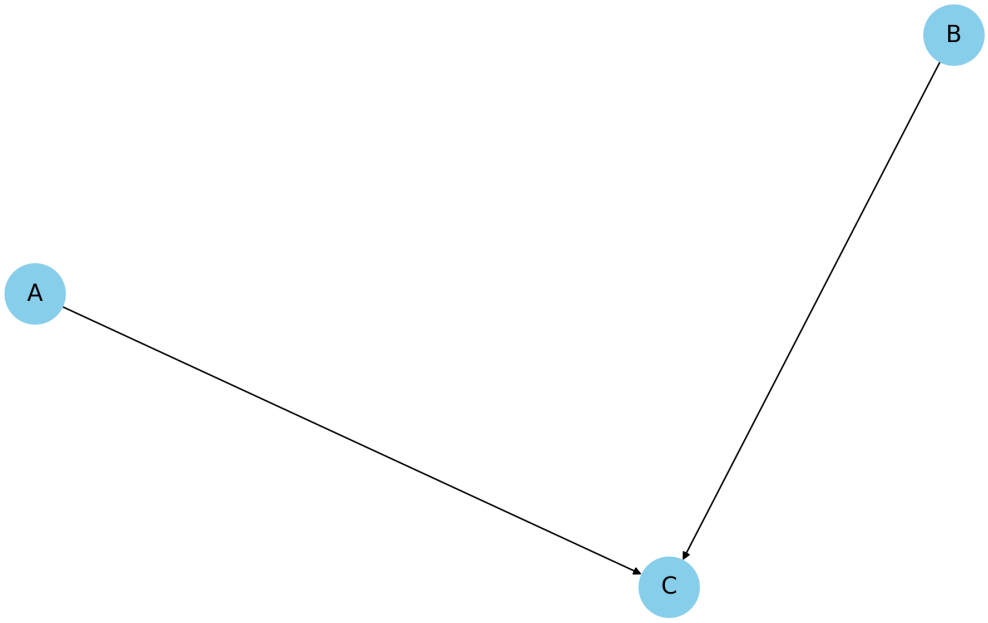
   $$P(X \mid Y, Z) = P(X \mid Z)$$

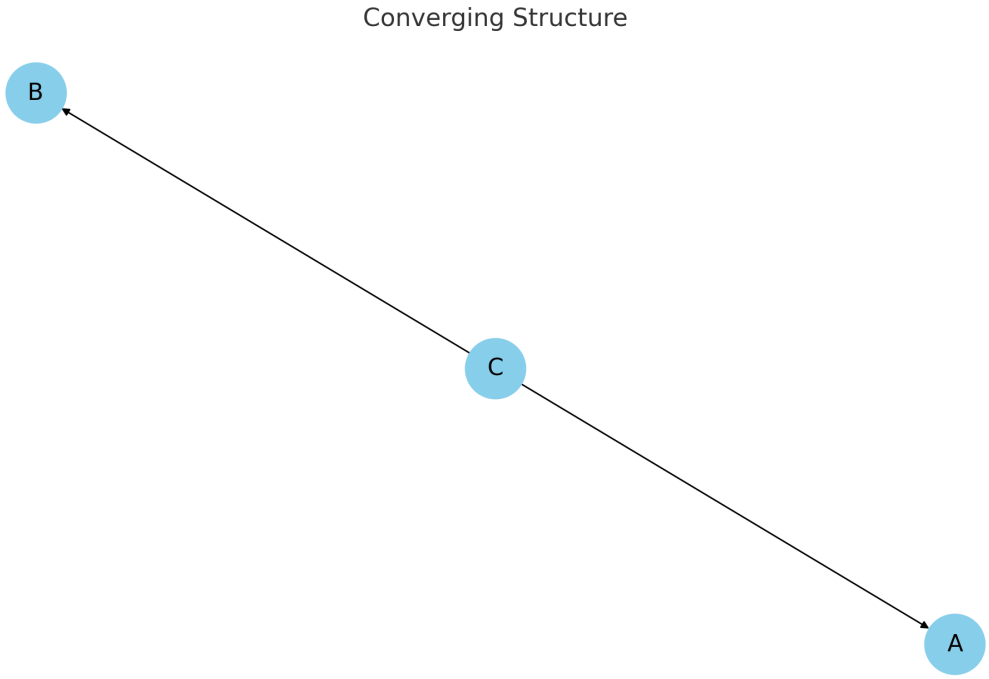
2. D-分离（d-separation）是贝叶斯网络中用来确定变量间条件独立性的关键概念。它基于网络结构来判断是否存在一种条件独立性。D-分离主要考虑三种基本结构：链式（serial）、分叉（diverging）和合并（converging）结构。

Serial Structure



Diverging Structure





1. 链式结构（Serial Connection）

在链式结构中，两个节点通过一个中间节点连接，形成一个链条。例如，A→B→C。

- **未观测中间节点**：当中间节点 B没有被观测时，A和C 是条件独立的。
- **观测中间节点**：如果 B 被观测，路径被激活，使得 A和 C条件相关。

2. 分叉结构（Diverging Connection）

分叉结构是指一个节点向两个不同的节点发散。例如，A→*B*←*C*。

- **未观测中间节点**：当 B 没有被观测时，A 和C 是条件独立的。
- **观测中间节点**：如果 B 被观测，路径被激活，A 和 C 条件相关。

3. 合并结构（Converging Connection）

合并结构是指两个节点向同一个节点合并。例如，A←*B*→*C*。

- **未观测中间节点**：如果 B没有被观测，且没有后续的子孙节点被观测，A和 C是条件独立的。
- **观测中间节点或其后代**：如果 B 或其任何后代被观测，A 和 C 变为条件相关。

特别注意：

- **后代节点**：在合并结构中，中间节点的后代也会影响条件独立性。如果中间节点的任何后代被观测，它会打开路径，使得两边的节点条件相关。
- **多路径**：在复杂的网络中，可能存在多条路径连接同一对节点。所有路径都必须被阻断，才能判断这两个节点是条件独立的。

mle和map

1. 最大似然估计（MLE）：
   定义: 最大似然估计是一种参数估计方法，它选择能够使给定观测数据出现概率（似然函数）最大的参数值。
   公式:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} L(\theta; X)$$

其中，$L(\theta; X)$ 是似然函数，θ是模型参数，X 是观测数据。

2. 最大后验概率估计（MAP）：
   定义: 最大后验概率估计结合了先验概率和似然函数，选择能够使参数的后验概率最大化的参数值。
   公式:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \left(P(\theta|X)\right) = \arg\max_{\theta} \left(P(X|\theta)P(\theta)\right)$$

其中，$P(\theta|X)$ 是后验概率，$P(X|\theta)$是似然函数，$P(\theta)$是关于参数的先验概率分布。

MLE 示例：抛硬币

1. 数据: 抛 N 次硬币，得到正面 k 次。
2. 似然函数: $L(p; k, N) = p^k(1-p)^{N-k}$。
3. 最大似然估计: 找到使似然函数最大的 p 值。
4. 解决方法: 对似然函数关于 p 求导，然后令导数为 0，解得 $\hat{p}_{MLE} = \frac{k}{N}$。

$$\hat{p}_{MLE} = \frac{k}{N}$$

MAP 示例：抛硬币

1. 数据: 抛 N 次硬币，得到正面 k 次。
2. 似然函数: $L(p; k, N) = p^k(1-p)^{N-k}$。
3. 先验分布: 选择贝塔分布，$ P(p) = \text{Beta}(p; \alpha, \beta)$。
4. 后验概率:$P(p | k, N) \propto L(p; k, N) P(p) $。
5. 最大后验估计: 找到使后验概率最大的 p 值。
6. 解决方法: 后验分布将是贝塔分布，计算得 $hatp_{MAP} = \frac{k+\alpha}{N+\alpha+\beta}$。

$$\hat{p}_{MAP} = \frac{k+\alpha}{N+\alpha+\beta}$$

示例场景

假设我们正在估计一个硬币正面朝上的概率 p。我们抛了这枚硬币10次，观察到了7次正面和3次反面。

先验分布

我们选择一个先验分布，这里使用Beta分布，Beta分布是二项分布的共轭先验。设定Beta分布的参数为 α 和 β。为了简化示例，我们假设这两个参数都是2。,先验分布为
$P(p) = \text{Beta}(p; 1, 1) = p^{\alpha-1}(1-p)^{\beta-1} = p(1-p)$ 。

似然函数

似然函数表示给定参数 p 下观测到当前数据的概率。在我们的例子中，似然函数是一个二项分布。如果抛了10次硬币，观察到7次正面，那么似然函数为：$L(p; 7, 10) = p^7(1-p)^3$ 。

后验分布

在贝叶斯框架下，后验分布是先验分布和似然函数的乘积，比例于二者的乘积。由于我们的先验是Beta分布，似然是二项分布，后验分布同样是Beta分布：
$P(p|k,N) \propto p^7(1-p)^3\mathrm{Beta}(p;1,1) = p^8(1-p)^4$。对乘积进行归一化使其的概率之和为1。需要对其乘一个常数系数，因为常数不影响求导，所以可以直接用乘积进行求导。注意这里的后验分布是正比于乘积而不是等于。实际上是除以这个乘积在定义域上的积分值(概率之和)。

MAP估计

对后验分布进行求导得出p为$\frac{2}{3}$ 。可以看出MLE和MAP的区别是MLE多了一个先验分布，先验分布会影响最后的参数最大似然估计。

pca的作用和运用过程，应用范围

主成分分析（PCA）是一种统计方法，用于简化数据集的复杂性，同时保留其主要特征。PCA 的主要作用和应用过程可以分为以下几个步骤：

1. **标准化数据**：首先，数据集中的每个特征（或变量）都需要标准化，以确保它们具有相同的比例。这通常涉及减去均值并除以标准差。
2. **计算协方差矩阵**：然后，计算标准化数据的协方差矩阵。协方差矩阵捕获了不同变量间的相关性。
3. **计算特征值和特征向量**：对协方差矩阵进行特征分解以获得其特征值和相应的特征向量。特征向量表示数据的主要方向，而特征值表示这些方向的重要性。
4. **选择主要成分**：根据特征值的大小选择前几个最重要的特征向量。这些特征向量代表了数据的“主要成分”。
5. **变换到新的子空间**：使用选定的主要成分将原始数据转换到新的子空间。这通常涉及将原始数据矩阵与选定的特征向量矩阵相乘。

PCA 的应用范围非常广泛，包括：

- **数据降维**：在机器学习和数据挖掘中，用于减少数据集的维数，同时保留最重要的信息。
- **噪声过滤**：消除数据中的噪声并提高数据质量。
- **特征提取**：在模式识别和信号处理中用于提取重要特征。
- **数据可视化**：将多维数据降维到二维或三维，以便于可视化和分析。

1. 标准化数据：$X_{std} = \frac{X-\mu}{\sigma}$

2. 计算协方差矩阵：$\mathrm{Cov}(X_{std}) = \frac{1}{n-1}X_{std}^T X_{std}$

3. 特征分解：$\mathrm{Cov}(X_{std}) = V\Lambda V^T$, 其中，$V$ 是特征向量，Λ 是特征值对角矩阵。

4. 选择主要成分：基于特征值选择前 k 个特征向量

5. 数据转换：$X_{pca} = X_{std}V_k$

vc维

VC维（Vapnik-Chervonenkis 维）是一种衡量给定函数集（假设集）学习能力的指标，用于计算机学习理论中。它是由Vladimir Vapnik和Alexey Chervonenkis提出的。

给定一个假设集 H 和一个集合 S 包含 n 个点，如果存在 2^n 种不同的方式通过假设集 H 将 S 中的点划分为两类。即对于 S 的每一种二分法，都存在一个假设在 H 中可以实现这种二分，那么我们说 H 可以打散 S 。

VC维定义为假设集 H 可以打散的最大集合的大小。形式上，如果存在大小为 d 的最大集合被 H 打散，但任何大小为 d+1 的集合都不能被 H 完全打散，那么假设集 H 的VC维是 d 。

用公式表示，如果 H 的VC维是 d ，则有：

- 对于任何大小为 d 的集合 S , H 可以打散 S
- 对于任何大小大于 d 的集合 S', H 不能打散 S'

VC维是理解机器学习模型的复杂性和泛化能力的一个重要概念。一个模型的VC维越高，意味着它在训练数据上的拟合能力越强，但同时可能也意味着它对新数据的泛化能力较差（过拟合的风险）。

举例，二维平面上的一条直线的vc维为3。因为对于一个点和两个点的情况，都可以用一条直线划分，三个点则不能。

em算法

EM算法（Expectation-Maximization algorithm）是一种迭代优化策略，用于估计概率模型中的参数，特别是在存在不完全数据或隐藏变量时。这种算法包括两个步骤：

1. **期望步骤（E步）**：给定当前参数估计，计算隐藏变量的期望值或分布。
2. **最大化步骤（M步）**：根据在E步中计算的隐藏变量的期望值，更新参数以最大化模型的似然函数。

EM算法通常用于混合模型的参数估计、自然语言处理、计算机视觉等领域。它的关键优点在于能够高效地处理缺失数据或隐藏变量的情况。然而，这种算法可能会陷入局部最优解，并且对初始参数敏感。

数据集

假设我们有以下观测数据集：$x_1, x_2, \ldots, x_n$，其中每个x_i是一个实数值。

初始化

1. **选择初始参数**：对于两个高斯分布，我们随机初始化它们的均值μ_1, μ_2，标准差σ_1, σ_2，以及混合系数π（表示第一个高斯分布的权重，第二个分布的权重则为$1 - \pi$）。

EM算法迭代

进行迭代直到收敛（即参数变化非常小或达到预定的迭代次数）。

E步骤

对于每个数据点x_i，计算它来自两个高斯分布的“责任”（responsibility）：

- $\gamma_{i1} = \frac{\pi \cdot \mathcal{N}(x_i|\mu_1,\sigma_1)}{\pi \cdot \mathcal{N}(x_i|\mu_1,\sigma_1) + (1-\pi) \cdot \mathcal{N}(x_i|\mu_2,\sigma_2)}$
- $\gamma_{i2} = 1 - \gamma_{i1}$

这里，$\mathcal{N}(x|\mu,\sigma)$是高斯分布的概率密度函数。

M步骤

更新参数：

1. **更新均值**：
   - $\mu_1^{new} = \frac{\sum_{i=1}^{n} \gamma_{i1} x_i}{\sum_{i=1}^{n} \gamma_{i1}}$
   - $\mu_2^{new} = \frac{\sum_{i=1}^{n} \gamma_{i2} x_i}{\sum_{i=1}^{n} \gamma_{i2}}$
2. **更新标准差**：
   - $\sigma_1^{new} = \sqrt{\frac{\sum_{i=1}^{n} \gamma_{i1} (x_i - \mu_1^{new})^2}{\sum_{i=1}^{n} \gamma_{i1}}}$
   - $\sigma_2^{new} = \sqrt{\frac{\sum_{i=1}^{n} \gamma_{i2} (x_i - \mu_2^{new})^2}{\sum_{i=1}^{n} \gamma_{i2}}}$
3. **更新混合系数**：
   - $\pi^{new} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{i1}$

重复E步骤和M步骤

在每次迭代中重复E步骤和M步骤，直到参数的变化很小或达到了预先设定的迭代次数。

结果

最终，我们将得到两个高斯分布的参数（均值、标准差）和它们在混合模型中的权重，这些参数可以最大化给定数据的似然函数。

bias variance

在统计学和机器学习领域，"偏差-方差权衡"（Bias-Variance Tradeoff）是一个重要概念。用中文来解释：

- **偏差（Bias）**：指的是模型在训练数据上的预测结果与真实结果之间的差异。高偏差通常意味着模型过于简单，无法捕捉到数据的所有特征和复杂性，这种现象被称为“欠拟合”。
- **方差（Variance）**：指的是模型在不同的训练数据集上的性能波动。高方差通常意味着模型过于复杂，过度拟合了训练数据的随机噪声，而不是底层数据的真实分布。

偏差-方差权衡是指在减少一个（比如偏差）的同时，往往会增加另一个（比如方差），反之亦然。理想的模型应该在偏差和方差之间找到平衡，以达到最佳的泛化性能。

kmeans

K-means 算法步骤

1. **初始化**：选择 K（簇的数量）个点作为初始中心点。在我们的例子中，K=2，我们随机选择两个点作为初始中心。比如说，我们选择 (1, 2) 和 (10, 4)。
2. **分配**：将每个数据点分配到最近的中心点所代表的簇。这里，我们计算每个数据点到两个中心点的距离，并将它们分配到最近的簇。
3. **更新**：对于每个簇，重新计算中心点，这是通过取簇中所有点的平均值来完成的。
4. **重复**：重复步骤 2 和 3，直到中心点不再显著变化或达到预定的迭代次数。

逻辑回归

假设我们有一个数据集，包括学生的两个特征：学习时间（小时）和上课出勤率（%），以及他们是否通过了考试（通过=1，未通过=0）。我们的目标是建立一个逻辑回归模型来预测学生是否会通过考试。

示例数据集

学生	学习时间（小时）	出勤率（%）	是否通过考试（1=通过，0=未通过）
A	10	90	1
B	8	75	1
C	3	50	0
D	2	60	0
E	5	70	1

计算步骤

1. **定义模型**：
   $P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2)}}$
   其中，X_1 是学习时间，X_2 是出勤率，$\beta_0, \beta_1, \beta_2$ 是模型参数。
2. **初始化参数**：
   假设初始参数 $\beta_0, \beta_1, \beta_2$ 都是0。
3. **计算预测概率**：
   对每个学生，将他们的学习时间和出勤率代入模型，计算预测概率。例如，对学生A:
   $P(Y=1) = \frac{1}{1+e^{-(0+0\times 10+0\times 90)}} = 0.5$
   初始时，所有预测概率都是0.5。
4. **计算损失函数**：
   使用对数似然损失函数来计算损失。对于单个样本：
   $L(\beta) = -[y\log(P) + (1-y)\log(1-P)]$
   对于整个数据集，计算所有样本损失的平均值。
5. **梯度下降**：
   使用梯度下降算法来更新参数 $\beta_0, \beta_1, \beta_2$。对每个参数，计算损失函数关于该参数的偏导数，然后更新参数：
   $\beta_j := \beta_j - \alpha \frac{\partial L}{\partial \beta_j}$
   其中，α 是学习率。
6. **重复迭代**：
   重复步骤3到5，直到模型收敛（即参数变化非常小或达到预设的迭代次数）。
7. **模型评估**：
   使用测试数据来评估模型的性能。通常通过计算准确率、召回率、F1分数等指标。
8. **预测新数据**：
   用训练好的模型来预测新数据的结果。

线性回归

$$\beta = \frac{n(\sum XY) - (\sum X)(\sum Y)}{n(\sum X^2) - (\sum X)^2}$$

$$\alpha = \frac{\sum Y - \beta(\sum X)}{n}$$

svm

支持向量机（SVM，Support Vector Machines）是一种广泛使用的监督学习算法，主要用于分类问题。SVM的基本思想是寻找一个最优的分割超平面，使得不同类别的样本之间的间隔最大化。

在数学上，SVM的目标是求解以下优化问题：

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \quad \forall i = 1, 2, \ldots, N \end{aligned}$$

其中：

- w是超平面的法向量。
- b 是超平面的截距。
- (x_i, y_i)是训练样本，其中 x_i是特征向量，y_i 是类别标签（通常取值为 −1或 1）。
- N 是训练样本的数量。

梯度下降

梯度下降（Gradient Descent）是一种用于寻找函数局部最小值的一阶迭代优化算法。在机器学习和深度学习中，梯度下降被广泛用于优化损失函数，即调整模型参数以最小化损失函数的值。

梯度下降的基本思想是：首先选择一个初始点作为起始点，然后在每一步迭代中，沿着函数梯度的反方向（即下降最快的方向）更新点的位置，直到达到一个局部最小值。

梯度下降的更新公式为：

$\theta_{\text{new}} = \theta_{\text{old}} - \alpha \nabla_\theta J(\theta)$

其中：

- θ是模型参数。
- $J(\theta)$是损失函数。
- $\nabla_\theta J(\theta)$是损失函数相对于参数 θ的梯度。
- α 是学习率，控制了在梯度方向上的步长大小。

马尔可夫毯

马尔可夫毯是概率论和统计学中的一个概念，用于描述随机变量之间的局部依赖关系。在图模型（如贝叶斯网络或马尔可夫网络）中，一个变量的马尔可夫毯包括所有直接影响该变量的其他变量，也就是说，这些变量直接与该变量相连。

具体来说，对于图模型中的一个节点（代表一个随机变量）来说，其马尔可夫毯包括：

1. **父节点**：在有向图（如贝叶斯网络）中，指向该节点的其他节点。
2. **子节点**：在有向图中，该节点指向的其他节点。
3. **共同子节点的其他父节点**：与该节点共享同一个子节点的其他节点。

马尔可夫毯的概念表明，给定其马尔可夫毯中的变量，该节点与图中其他节点条件独立。这是概率图模型中进行推断和学习的一个关键概念。

实例：使用贝叶斯网络进行医疗诊断

假设我们有一个简化的贝叶斯网络，它包含以下节点：

- 疾病（例如心脏病）
- 症状（如胸痛、高血压）
- 生活习惯（如抽烟、运动）

构建网络

1. **疾病节点**：心脏病的概率可能受生活习惯的影响。
2. **症状节点**：胸痛和高血压可能直接受心脏病的影响。
3. **生活习惯节点**：抽烟和运动可能影响心脏病的风险。

马尔可夫毯的应用

1. 对于“心脏病”这个节点，它的马尔可夫毯包括“生活习惯”（父节点）和“胸痛”、“高血压”（子节点）。
2. 对于“胸痛”这个节点，它的马尔可夫毯只包括“心脏病”（父节点）。

诊断过程

在实际应用中，医生可能观察到特定的症状（如胸痛），然后使用这个网络来评估患者患有特定疾病（如心脏病）的概率。通过考虑与心脏病节点直接相连的变量（即它的马尔可夫毯），可以有效地计算出心脏病的条件概率，而忽略网络中与心脏病条件独立的其他变量。

复杂度（参数量）

1. **复杂度 (Complexity)**: 这通常指的是一个算法执行所需的时间或空间资源的量度。在计算机科学中，复杂度经常用大O符号（Big O notation）来表示，例如 O(n), O(n^2), O(log n) 等。这里的 'n' 代表输入数据的大小。
    - 时间复杂度（Time Complexity）: 衡量一个算法执行所需时间的长短。
    - 空间复杂度（Space Complexity）: 衡量一个算法执行所需空间的大小。
2. **参数量 (Parameter Count)**: 这通常指的是在机器学习模型或数学函数中使用的参数的数量。例如，一个神经网络的参数量可以是指网络中所有权重（weights）和偏差（biases）的总和。
    - 在深度学习模型中，参数量越大，模型通常越复杂，能够学习更多的特征，但同时也需要更多的数据来训练，且更容易过拟合。