1、概率图模型概述

概率图模型算法往往应用于NLP自然语言处理领域。

当然很多传统机器学习的算法也常用于 NLP 的任务。例如，用朴素贝叶斯进行文本分类、用 SVM 进行语义角色标注，虽然它们在某些 NLP 任务中都实现了很好的效果，但它们都相互独立， 没有形成体系。

随着近些年对智能推理和认知神经学的深入研究，人们对大脑和语言的内在机制了解得越来越多，也越来越能从更高层次上观察和认识自然语言，由此形成一套完整的算法体系。目前最流行的算法思想包含如下两大流派：

- 基于概率论和图论的概率图模型
- 基于人工神经网络的深度学习理论

2、贝叶斯

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

2.1、贝叶斯案例一

一个例子，现分别有 A、B 两个容器，在容器 A 里分别有 7 个红球和 3 个白球，在容器 B 里有 1 个红球和 9 个白球，现已知从这两个容器里任意抽出了一个球，且是红球，问这个红球是来自容器 A 的概率是多少?

2.2、贝叶斯案例二

例如：一座别墅在过去的 20 年里一共发生过 2 次被盗，别墅的主人有一条狗，狗平均每周晚上叫 3 次，在盗贼入侵时狗叫的概率被估计为 0.9，问题是：在狗叫的时候发生入侵的概率是多少?

答案

3、朴素贝叶斯

举个例子，大学的时候，某男生经常去007自习室上晚自习，发现他喜欢的那个女生也常去那个自习室，心中窃喜，于是每天买点好吃的在那个自习室**蹲点**等她来，可是人家女生不一定每天都来，眼看天气渐渐炎热，自习室又不开空调，如果那个女生没去自习室，该男生也就不去，每次男生鼓足勇气说：“嘿，你明天还来不？”,“啊，不知道，**看情况**”。

然后该男生每天就把她去自习室与否以及一些其他情况做一下记录，用Y表示该女生是否去自习室，即Y={去，不去}，X是跟去自习室有关联的一系列条件，比如当天上了哪门主课，蹲点统计了一段时间后，该男生打算今天不再蹲点，而是先预测一下她会不会去，现在已经知道了今天上了**常微分方程**这门主课，于是计算P(Y=去|常微分方程)与P(Y=不去|常微分方程)，看**哪个**概率大，如果P(Y=去|常微分方程) >P(Y=不去|常微分方程)，那这个男生不管多热都屁颠屁颠去自习室了，否则就不去自习室受罪了。

P(Y=去|常微分方程)的计算可以通过贝叶斯公式进行计算，公式如下：

$$\mathrm{P}(\mathrm{Y}=去|常微分方程) = \frac{\mathrm{P}(常微分方程|\mathrm{Y}=去)\mathrm{P}(\mathrm{Y}=去)}{\mathrm{P}(常微分方程)}$$

后来他发现还有一些其他条件可以挖，比如当天**星期几**、当天的**天气**，统计了一段时间后，该男子一计算，发现不好算了，因为总结历史的公式：

$$P(Y|X) = P(Y|X^{(1)}, X^{(2)}, \ldots, X^{(n)})$$

这里n = 3，x(1)表示主课，x(2)表示天气，x(3)表示星期几，Y仍然是{去，不去}，现在主课有8门，天气有晴、雨、阴三种，那么总共需要估计的参数有8×3×7×2=336个，每天只能收集到一条数据，那么等凑齐336条数据，黄花菜都凉了，男生大呼不妙！

于是做了一个**独立性假设**，假设这些影响她去自习室的因素是独立互不相关的！

有了这个独立假设后，需要估计的参数就变为，(8+3+7)×2 = 36个了，而且每天收集的一条数据，可以提供3个参数，这样该男生就预测越来越准了，天下武功唯快不破！迎娶白富美，全靠数学算！

$$\mathrm{P(Y|X)} = P(Y|X^{(1)}, X^{(2)}, \ldots, X^{(n)})$$

$$= \prod_{i=1}^{n} P(Y|X^{(i)})$$

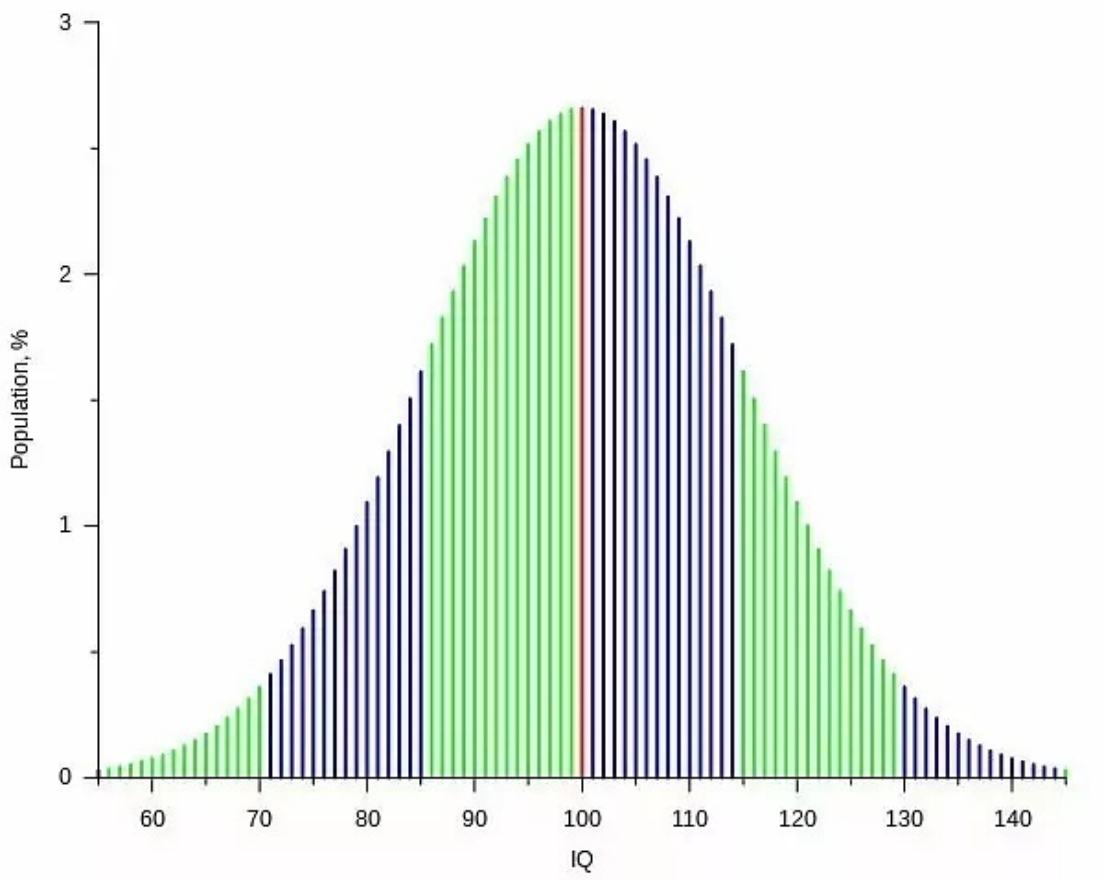
4、朴素贝叶斯实例讲解

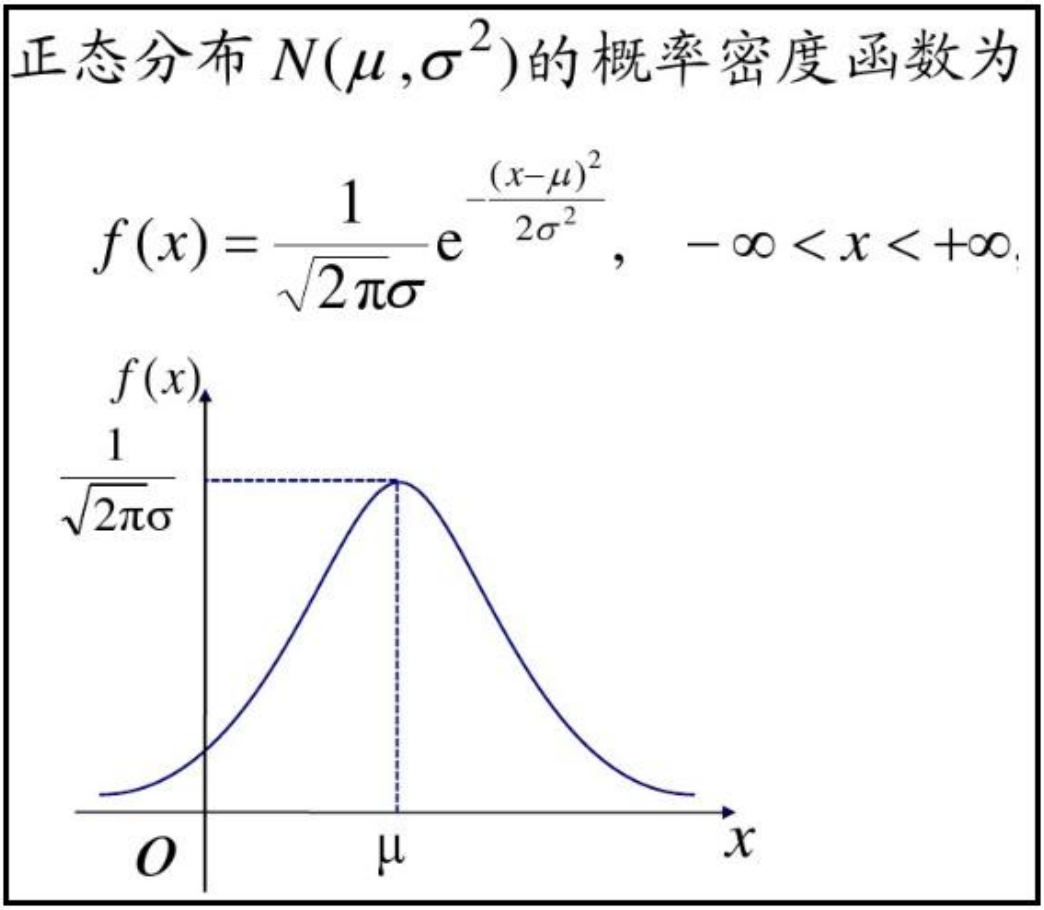
详情参考本人博客文章

5、朴素贝叶斯模型介绍

5.1、高斯分布朴素贝叶斯

高斯分布朴素贝叶斯------>正太分布





概率密度公式：

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

f(x)表示事件的**概率分布**

5.2、伯努利分布朴素贝叶斯

伯努利分布又叫做0-1分布，指一次随机试验，结果只有两种。也就是一个随机变量的取值只有0和1。

记为: 0-1分布 或B(1,p)，其中 p 表示一次伯努利实验中结果为正或为1的概率。

假设你要生孩子，生男孩子概率p，生女孩纸概率1-p

伯努利实验：例如，生一次孩子

伯努利分布：生一次孩子，生男孩子概率为p,生女孩纸概率1-p，这个就是伯努利分布

$$f(x|p)=\begin{cases}p^x q^{1-x}, & x=0,1;\\ 0, & x\neq 0;\end{cases}$$

伯努利实验就是做一次服从伯努利概率分布的事件，**它发生的可能性是p，不发生的可能性是1-p。**

由伯努利分布延伸到**二项分布**，**二项分布**是多次伯努利分布实验的概率分布。

以抛硬币举例，在抛硬币事件当中，每一次抛硬币的结果是独立的，并且每次抛硬币正面朝上的概率是恒定的，所以单次抛硬币符合伯努利分布。我们假设硬币**正面朝上的概率是p**，那么**反面朝上的概率是q=(1-p)**。我们重复抛n次硬币，其中有k项正面朝上的事件，就是**二项分布**：

$$P(X=k)=C_n^k p^k q^{n-k}$$

$$=\frac{n!}{(n-k)!k!}p^k q^{n-k}$$

5.3、多项式分布朴素贝叶斯

多项分布是在二项分布的基础上进一步的拓展。

以掷色子为例，在掷色子实验中可能出现的结局是：1,2,3,4,5,6（6标记为k，便于书写公式），分别记为变量 X_1、X_2、 ...、X_k，它们的概率分布分别是p_1，p_2， ...，p_k。那么在n次实验的结果中，1出现 n_1 次、2出现 n_2 次、...、6出现 n_k 次，这种事件的出现概率P有下面公式：

$$P(X_1=n_1,\ldots,\ X_k=n_k)=\frac{n!}{n_1!\cdots n_k!}p_1^{n_1}\cdots p_k^{n_k},\sum_{i=1}^{k}n_i=n$$

另一种写法：

$$P(X_1=n_1,\ldots,\ X_k=n_k)=n!\prod_{i=1}^{k}\frac{p_i^{n_i}}{n_i!},\sum_{i=1}^{k}n_i=n$$

6、朴素贝叶斯模型使用

使用正太分布数据，鸢尾花作为示例（鸢尾花是自然界的植物，其自身特征数据是正态分布的~）

6.1、数据加载

```
import numpy as np
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB,BernoulliNB,MultinomialNB
from sklearn.model_selection import train_test_split
X,y = datasets.load_iris(return_X_y=True)
```

6.2、高斯分布朴素贝叶斯表现

```
score = 0
model = GaussianNB()
for i in range(100):
    X_train,X_test,y_train,y_test = train_test_split(X,y)
    model.fit(X_train,y_train)
    score += model.score(X_test,y_test)/100
print('高斯朴素贝叶斯模型平均预测准确率：',score)
'''
高斯朴素贝叶斯模型平均预测准确率： 0.9557894736842099
'''
```

6.3、伯努利分布朴素贝叶斯表现

```
score = 0
model = BernoulliNB()
for i in range(100):
    X_train,X_test,y_train,y_test = train_test_split(X,y)
    model.fit(X_train,y_train)
    score += model.score(X_test,y_test)/100
print('伯努利朴素贝叶斯模型平均预测准确率：',score)
'''
伯努利朴素贝叶斯模型平均预测准确率： 0.26105263157894737
'''
```

6.4、多项式分布朴素贝叶斯表现

```
score = 0
model = MultinomialNB()
for i in range(100):
    X_train,X_test,y_train,y_test = train_test_split(X,y)
    model.fit(X_train,y_train)
    score += model.score(X_test,y_test)/100
print('多项式朴素贝叶斯模型平均预测准确率：',score)
'''
多项式朴素贝叶斯模型平均预测准确率： 0.8255263157894736
'''
```

7、文本分类

文本分类的结构化方法就是 one-hot 表达模型。它是最直观，也是目前为止最常用的词表示方法，虽然越来越多的实践已经证明，这种模型存在局限性，但它仍在文本分类中得到广泛应用。

假设把语料库中的所有词都收集为一个词典 D，词典容纳了语料库中所有句子的词汇。

One-hot 方法就是把每个词表示为一个长长的向量。这个向量的维度是词典大小，其中绝大多数元素为 0，只有一个维度的值为 1。这个维度就代表了当前的词。

7.1、英文one-hot编码

文本一：My dog ate my homework；

文本二：My cat ate the fish；

文本三：Precious things are very few in the world,that is the reason there is only one you!

```
import jieba
import numpy as np
data = ['My dog ate my homework.','My cat ate the fish.',
        'Precious things are very few in the world,that is the reason there is only one
you!']

result = []
for s in data:
    result.extend([i for i in jieba.lcut(s) if i not in [' ',',','.','!']])
result = np.array(result)

result = np.unique(result)
print(result)
for s in data:
    word_embedding = [(i == result).astype(np.int8) for i in jieba.lcut(s) if i not in ['
',',','.','!']]
    print(np.array(word_embedding))
```

```
'''
['My' 'Precious' 'are' 'ate' 'cat' 'dog' 'few' 'fish' 'homework' 'in' 'is'
 'my' 'one' 'only' 'reason' 'that' 'the' 'there' 'things' 'very' 'world'
 'you']
[[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]]
'''
```

7.2、中文one-hot编码

s1 = '喜欢上一个人'

s2 = '尼姑亲吻了和尚的嘴唇'

s3 = '老师你教的都是没用的东西'

```
import jieba
import numpy as np
data = ['喜欢上一个人','尼姑亲吻了和尚的嘴唇','老师你教的都是没用的东西']

result = []
for s in data:
    result.extend([i for i in jieba.lcut(s)])
result = np.array(result)

result = np.unique(result)
print(result)
for s in data:
    word_embedding = [(i == result).astype(np.int8) for i in jieba.lcut(s) if i not in ['
','，','。','！']]
    print(np.array(word_embedding))

'''
['一个' '上' '东西' '了' '亲吻' '人' '你' '和尚' '喜欢' '嘴唇' '尼姑' '教' '是' '没用' '的' '老师'
 '都']
[[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]]
'''
```

7.3、TF-IDF

7.3.1、词频-逆向文件频率介绍

TF-IDF（term frequency–inverse document frequency，词频-逆向文件频率）是一种用于信息检索（information retrieval）与文本挖掘（text mining）的常用**加权技术。**

TF-IDF是一种统计方法，用以评估某**字词**对于一个文件集或一个语料库中的其中一份文件的**重要程度。**字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

TF-IDF的主要思想是：如果某个单词在一篇文章中出现的**频率TF高**，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

7.3.2、词频TF计算

词频（TF）表示词条（关键字）在文本中出现的频率。

$$\mathrm{TF} = \frac{\text{词条v出现的次数}}{\text{总词条数目}}$$

7.3.3、逆向文件频率IDF计算

逆向文件频率 (IDF)：某一特定词语的IDF，可以由**总文件数目除以包含该词语的文件的数目，再将得到的商取对数得到。**

如果包含词条t的文档越少, IDF越大，则说明词条具有很好的类别区分能力。

$$\mathrm{IDF} = \log \frac{|\mathrm{D}|}{|\{\mathrm{j}:\mathrm{t_i} \in \mathrm{d_j}\}|}$$

7.3.4、TF-IDF计算

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。

$$\mathrm{TF-IDF} = \mathrm{TF} \times \mathrm{IDF}$$

7.3.5、TF-IDF算例演示

有很多不同的数学公式可以用来计算TF-IDF。词频 (TF) 是一词语出现的次数除以该文件的总词语数。假如一篇文件的总词语数是100个，而词语“**Python**”出现了5次，那么“**Python**”一词在该文件中的词频就是3/100=0.05。一个计算文件频率 (IDF) 的方法是文件集里包含的文件总数除以测定有多少份文件出现过“**Python**”一词。所以，如果“**Python**”一词在1000份文件出现过，而文件总数是10000000份的话，其逆向文件频率就是 lg(10000000 / 1000)=4。最后的TF-IDF的分数为0.05 * 4=0.2。

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import jieba
import numpy as np

data = np.array(['政治 历史 地理 语文 化学',
                 'Python 计算机 语文 英语 数学'])
# 词频统计
vectorizer = CountVectorizer()
# tf-idf权值计算
tf_idf_transformer = TfidfTransformer()
# 对样本进行转换
tf_idf = tf_idf_transformer.fit_transform(vectorizer.fit_transform(data))
# 数据提取
tf_idf_weight = tf_idf.toarray()
vocabulary = vectorizer.vocabulary_
vocabulary = sorted( vocabulary.items(),key = lambda x:x[1],reverse = False)
display(vocabulary)
display(tf_idf_weight)
'''
[('python', 0),
 ('化学', 1),
 ('历史', 2),
 ('地理', 3),
 ('政治', 4),
 ('数学', 5),
 ('英语', 6),
 ('计算机', 7),
 ('语文', 8)]
```

```
array([[0.        , 0.47107781, 0.47107781, 0.47107781, 0.47107781,
        0.        , 0.        , 0.        , 0.33517574],
       [0.47107781, 0.        , 0.        , 0.        , 0.        ,
        0.47107781, 0.47107781, 0.47107781, 0.33517574]])
'''
```

8、垃圾短信分类项目实战

8.1、数据加载与介绍

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB,BernoulliNB,MultinomialNB
from sklearn.model_selection import train_test_split
from scipy import sparse # 稀松矩阵
# feature_selection 特征选择!
# feature_extraction 特征提取，萃取
# 土壤中，炼铁，这个过程类比 萃取
# 统计词频！通过词频，判断类别，判断短信是否是垃圾短信
# free、phone、获奖、优惠
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
# 短信数据
sms = pd.read_csv('./自动识别垃圾短信.csv',sep = '\t',header= None)
sms.rename({0:'label',1:'message'},axis = 1,inplace = True)
display(sms.shape,sms.head()) #文本数据无法直接建模
```

8.2、文本数据处理

```
cv = CountVectorizer()# stop-words 停用词，英文中的标点符号，对分类作用不大
# 主要目的，节省内存空间
# 量化
X = cv.fit_transform(sms['message'])
display(X)
print(X)
```

稀松矩阵介绍

```
# scipy中提供了方法
# 稀松矩阵、稠密矩阵 对比
a = np.random.randint(0,10,size = (1000,5))
a[a >=3] = 0
np.savez('稠密矩阵.npz',a) # 20.3kb
s = sparse.csc_matrix(a)
sparse.save_npz('稀松矩阵.npz',s) # 2.84kb
display(a,s)
print(s)
```

TF-IDF转换

```
# 此时X是稀松矩阵
# 稀松矩阵也可以进行拆分
tf_idf = TfidfTransformer()
X = tf_idf.fit_transform(X)
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 1187)
X_train
```

8.3、数据建模评估

```
%%time
gNB = GaussianNB() # 高斯分布，是正太分布，数据属性必须是稠密矩阵
gNB.fit(X_train.toarray(),y_train)
print('高斯分布',gNB.score(X_test.toarray(),y_test))
'''
高斯分布 0.8869955156950673
Wall time: 717 ms
'''
```

```
%%time
# 稀松矩阵，计算优势，内存中占有量很小
bNB = BernoulliNB() # 二项分布，数据可以稀松的，准确率提升了很多，计算时间大大缩短
bNB.fit(X_train,y_train)
print('伯努利：',bNB.score(X_test,y_test))
'''
伯努利： 0.9730941704035875
Wall time: 13 ms
'''
```

```
%%time
mNB = MultinomialNB()
mNB.fit(X_train,y_train)
print('多项式分布：',mNB.score(X_test,y_test))
'''
多项式分布： 0.95695067264574
Wall time: 15 ms
'''
```

8.5、构建新短信预测

```
X_test = ['Your free ringtone is waiting to be collected. Simply text the password "MIX" to 85069 to verify.I see the letter B on my car Please call now 08000930705 for delivery tomorrow',
          'Precious things are very few in the world,that is the reason there is only one you',
          "GENT! We are trying to contact you. Last weekends draw shows that you won a £1000 prize GUARANTEED. U don't know how stubborn I am. Congrats! 1 year special cinema pass for 2 is yours.",
          'Congrats! 1 year special cinema pass for 2 is yours. call 09061209465 now! C Suprman V, Matrix3, StarWars3, etc all 4 FREE! bx420-ip4-5we. 150pm. Dont miss out!']

X_test = tf_idf.transform(cv.transform(X_test))
bNB.predict(X_test)
'''
array(['spam', 'ham', 'spam', 'spam'], dtype='<U4')
'''
```

9、新闻类别划分

9.1、加载数据（联网国外下载）

```
from sklearn.naive_bayes import GaussianNB,BernoulliNB,MultinomialNB
from sklearn import datasets
# countVectorizer词频
# tf-idf term frequency（词频） inverse document frequency（你文本词频） + 权重
from sklearn.feature_extraction.text import
TfidfVectorizer,CountVectorizer,ENGLISH_STOP_WORDS
from sklearn.model_selection import train_test_split
# 会存储到本地文件
news = datasets.fetch_20newsgroups(subset='all',remove=('headers', 'footers', 'quotes'))
```

9.2、文本数据转换

```
'''Convert a collection of raw documents to a matrix of TF-IDF features.
Equivalent to :class:`CountVectorizer` followed by
:class:`TfidfTransformer`.'''
tf_idf = TfidfVectorizer(stop_words=ENGLISH_STOP_WORDS)
X_tf_idf = tf_idf.fit_transform(X)
X_tf_idf
X_train,X_test,y_train,y_test = train_test_split(X_tf_idf,y,test_size = 0.2)
```

9.3、数据建模

```
%%time
bNB = BernoulliNB()
bNB.fit(X_train,y_train)
bNB.score(X_test,y_test)
'''
Wall time: 18.9 ms
0.7553191489361702
'''
```

```
%%time
mNB = MultinomialNB()
mNB.fit(X_train,y_train)
mNB.score(X_test,y_test)
'''
Wall time: 11 ms
0.8337765957446809
'''
```

```
%%time
gNB = GaussianNB()
gNB.fit(X_train.toarray(),y_train)
gNB.score(X_test.toarray(),y_test)
'''
Wall time: 2.66 s
0.8125
'''
```

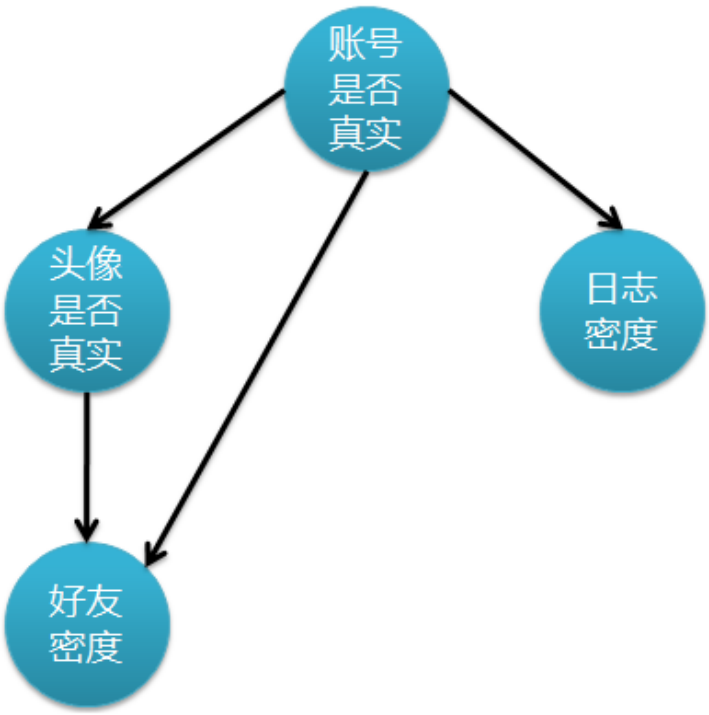
10、贝叶斯网络

10.1、朴素贝叶斯与贝叶斯网络

朴素贝叶斯可以看做是贝叶斯网络的特殊情况：即该网络中无边，各个节点都是独立的（前提是独立性假设）。

那么，当朴素贝叶斯中的假设：独立同分布不成立时，应该如何解决呢？可以使用贝叶斯网络。

贝叶斯网络借助**有向无环图**来刻画属性之间的依赖关系，并使用条件概率表来描述属性的联合概率分布。

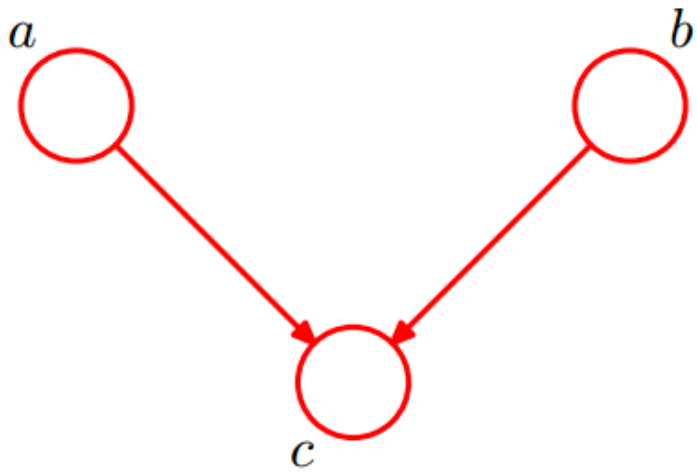


10.2、贝叶斯网络定义

贝叶斯网络(Bayesian network)，又称信念网络(Belief Network)，或有向无环图模型(directed acyclic graphical model)，是一种概率图模型，于1985年由Judea Pearl首先提出。它是一种模拟人类推理过程中因果关系的不确定性处理模型，其网络拓朴结构是一个有向无环图(DAG)。

10.3、贝叶斯网络三种结构

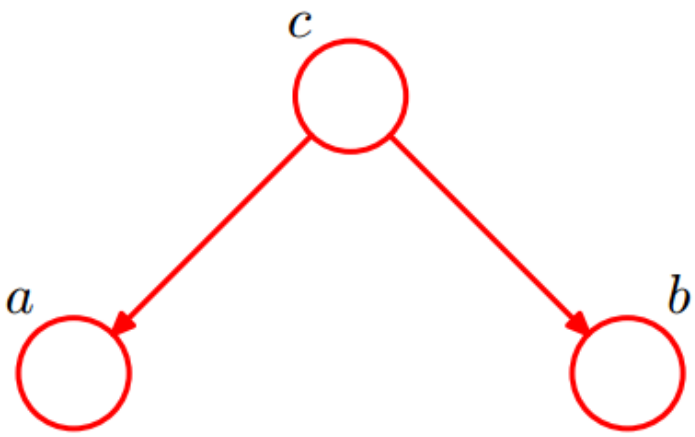
形式一：head-to-head



上图概率公式如下：P(a,b,c) = P(a) * P(b) * P(c|a,b)。

在 c 未知的条件下，a、b被阻断(blocked)，是独立的，称之为head-to-head条件独立。也就是a和b符合独立性假设。

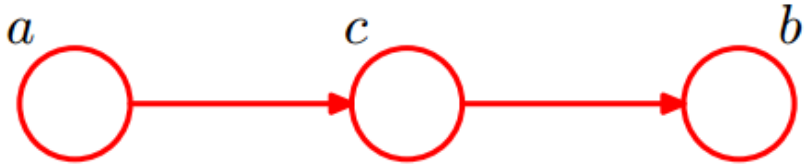
形式二：tail-to-tail



1. 在 c 未知的时候，有：P(a,b,c)=P(c) * P(a|c) * P(b|c)，此时，没法得出 P(a,b) = P(a) * P(b)，即 c 未知时，a、b 不独立。

2. 在 c 已知的时候，有：P(a,b|c)=P(a,b,c) / P(c)，然后将 P(a,b,c)=P(c) * P(a|c) * P(b|c)带入式子中，得到：P(a,b|c)=P(a,b,c) / P(c) = P(c) * P(a|c) * P(b|c) / P(c) = P(a|c) * P(b|c)，即 c 已知时，a、b 独立。

形式三：head-to-tail



1. c 未知时，有：P(a,b,c)=P(a) * P(c|a) * P(b|c)，但无法推出 P(a,b) = P(a) * P(b)，即 c 未知时，a、b 不独立。

2. c 已知时，有：P(a,b|c)=P(a,b,c) / P(c)，且根据 P(a,c) = P(a) * P(c|a) = P(c) * P(a|c)，可化简得到：

   P(a,b|c) =P(a,b,c) / P(c)

   = P(a) * P(c|a) * P(b|c) / P(c)

   =P(a,c) * P(b|c) / P(c)

   =P(a|c) * P(b|c)

   所以，在 c 给定的条件下，a，b 被阻断(blocked)，是独立的，称之为 head-to-tail 条件独立。

   拓展一下，head-to-tail 其实就是一个链式网络，如下图所示：

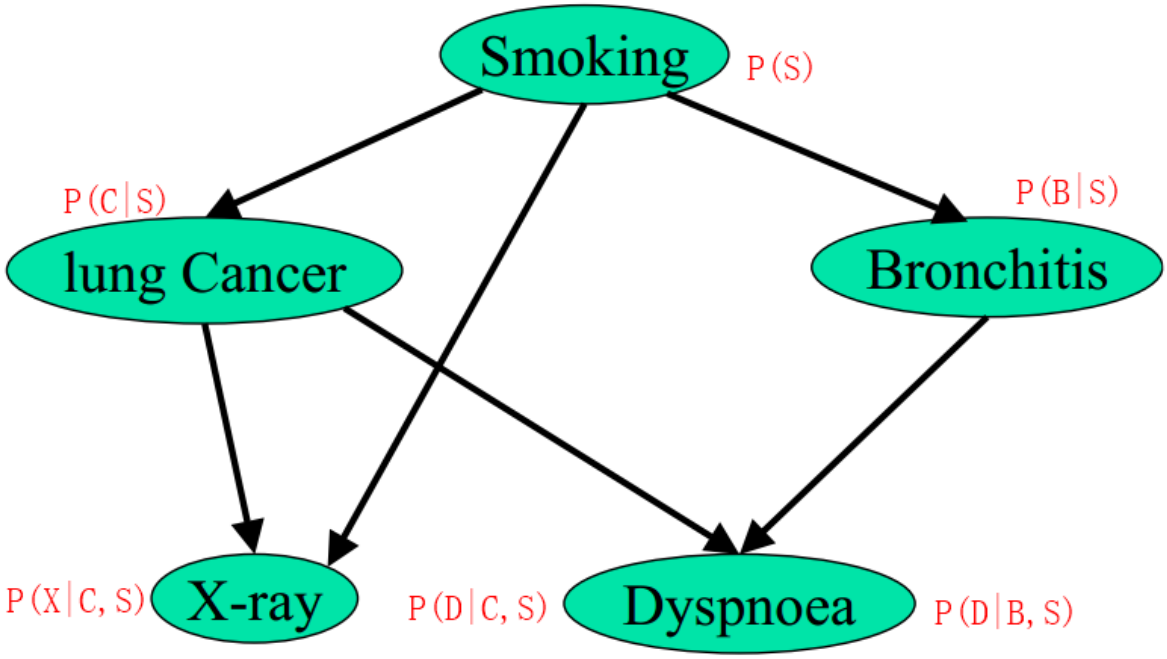
   根据之前对**head-to-tail**的讲解，我们已经知道，在X_i给定的条件下，X_{i+1}的分布和$X_1, X_2 \dots X_{i-1}$条件独立。意味着啥呢？意味着：X_{i+1}的分布状态只和X_i有关，和其他变量条件独立。通俗点说，当前状态只跟上一状态有关，跟之前的状态无关。这种顺次演变的随机过程，就叫做马尔科夫链（Markov chain）。且有：

$$P(X_{n+1}|X_0, X_1, \cdots, X_n) = P(X_{n+1}|X_n)$$

朴素贝叶斯可以看做是贝叶斯网络的特殊情况：即该网络中无边，各个节点都是独立的。 朴素贝叶斯朴素在哪里呢？ 一个特征出现的概率与其他特征（条件）独立！

10.4、贝叶斯网络实例

有如下贝叶斯网络：



其中，各个单词、表达式表示的含义如下：

- smoking 表示吸烟，其概率用 P(S) 表示，lung Cancer 表示肺癌，一个人在吸烟的情况下得肺癌的概率用 P(C|S) 表示，X-ray 表示需要照医学上的 X 光，肺癌可能会导致需要照 X 光，吸烟也有可能会导致需要照 X 光（所以 smoking 也是 X-ray 的一个因素），所以，因吸烟且得肺癌而需要照X光的概率用 P(X|C,S) 表示。
- Bronchitis 表示支气管炎，一个人在吸烟的情况下得支气管炎的概率用 P(B|S)，Dyspnoea 表示呼吸困难，支气管炎可能会导致呼吸困难，肺癌也有可能会导致呼吸困难（所以 lung Cancer 也是Dyspnoea的一个因素），因吸烟且得了支气管炎导致呼吸困难的概率用P(D|S,B)表示。

lung Cancer 简记为 C，Bronchitis 简记为 B，Dyspnoea 简记为 D，且 C = 0 表示 lung Cancer 不发生的概率，C = 1表示 lung Cancer 发生的概率，其他含义类似。

10.5、概率图模型

概率图模型是一类用图形模式表达基于概率相关关系的模型的总称。概率图模型结合概率论与图论的知识，利用图来表示与模型有关的变量的联合概率分布。近10年它已成为不确定性推理的研究热点，在人工智能、机器学习和计算机视觉等领域有广阔的应用前景。

根据是否是有向图，可以分为有向图模型和无向图模型。
有向图模型（又称为贝叶斯网络），例如：隐马尔科夫模型（Hidden Markov Model，HMM）
无向图模型（又称为马尔科夫网络），例如：条件随机场（Conditional Random Fields，CRF）

后面课程中会，进行介绍说明。