Bayesian Models

Bayesian models are based on bayes rule. Bayesians always assume that our training data come from some distribution. And given prior probability of parameters $p(\theta)$, we hope to use bayesian models to compute its posterior probability $p(\theta|\mathbf{x})$ so as to help us estimate the true data distribution.

$$p(\theta|\mathbf{x}) = \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})}$$

Naive Bayes Classifier

Learning a Naive Bayes classifier is just a matter of counting how many times each attribute co-occurs with each class. There're two assumptions in naive bayes classifier:

- All **variables** in the dataset are independent.
- The distribution of each attribute somehow we know, but we don't know its parameters exactly.

Naive bayes classifier is a kind of lazy learning like KNN.

1. Model Setup

- What we know

$$\begin{aligned} p(Y=c|\mathbf{x}) &= \frac{p(\mathbf{x}|Y=c)p(Y=c)}{p(\mathbf{x})} \\ &= \frac{p(Y=c)}{p(\mathbf{x})}p(\mathbf{x}|Y=c) \\ &= \frac{p(Y=c)}{p(\mathbf{x})}\Pi_{i=1}^{d}p(x_i|Y=c) \end{aligned}$$

where x_i is the d^{th} feature of $\mathbf{x}$.

- What we want to know: $argmax_c\ p(Y=c|\mathbf{x})$

Since for all classes, $p(\mathbf{x})$ are same, we could know that

$$argmax_c\ p(Y=c|\mathbf{x}) = argmax_c\ p(Y=c)\Pi_{i=1}^{d}p(x_i|Y=c)$$

Usually, we would use just the frequency to estimate the prior probability. That means $p(Y=c) = \frac{D_c}{D}$ where D is the numbers of data and D_c is the numbers of data which are in c class.

As for the likelihood $p(x_i|Y=c)$, if the feature is qualitative, frequency is also used to estimate it.

$$p(x_i|Y=c) = \frac{D_{x_i,c}}{D_c}$$

But if the feature is quantitative, we would use the assumed probability distribution function to estimate it. For example, we assume that the feature is distributed as gaussian
$p(x_i|Y=c) \sim N(\mu_{c,i}, \sigma_{c,i}^2)$

$$p(x_i|Y=c) = \frac{1}{\sqrt{2\pi\sigma_{c,i}}} e^{-\frac{(x_i-\mu_{c,i})^2}{2\sigma_{c,i}^2}}$$

2. Smoothing Trick

In high dimension, chances are that we could have the curse of dimensionality issue. In this case, **for qualitative variables**, $D_{x_i,c}$ could be 0 which could make wrong decisions. In order to avoid this issue, we usually use smoothing trick, such as laplacian correction before estimation. The correction goes like

$$\hat{p}(Y=c) = \frac{D_c+1}{D+N}$$
$$p(x_i|Y=c) = \frac{D_{x_i,c}+1}{D_c+N_i}$$

Where N is the number of classes of dataset and N_i is the number of possible values of i^{th} variable.

Ref: https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf

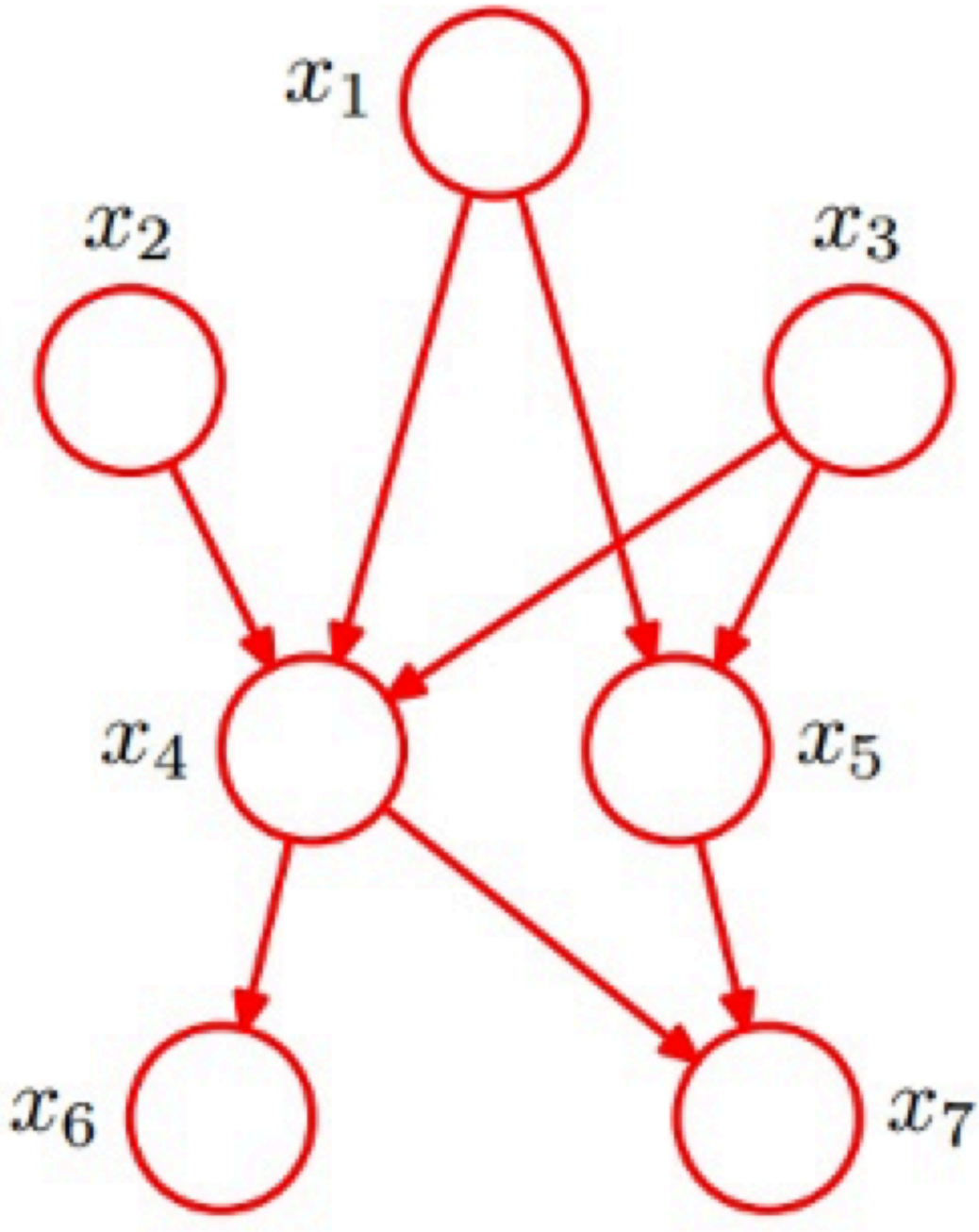
Bayesian Network

Naive bayesian models of course are naive because it's ideal to make variables to be independent to each other. In real-world data, features are usually somehow correlated with others. Bayesian networks are models that could represent dependency and conditional independency of variables in our dataset. A bayesian network falls under the category of probabilistic graphical modelling technique. Bayesian networks are used to model uncertainties by using DAG (directed acyclic graphs).

A bayesian network could be modelled as the joint distribution of variables:

$$P(x_1, x_2, \ldots, x_k) = \Pi_{i \in I} P(x_i|x_{pa(i)})$$

where $x_{pa(i)}$ are the set of parent nodes of x_i. Based on the form of joint distribution, we could know the structure of bayesian network.



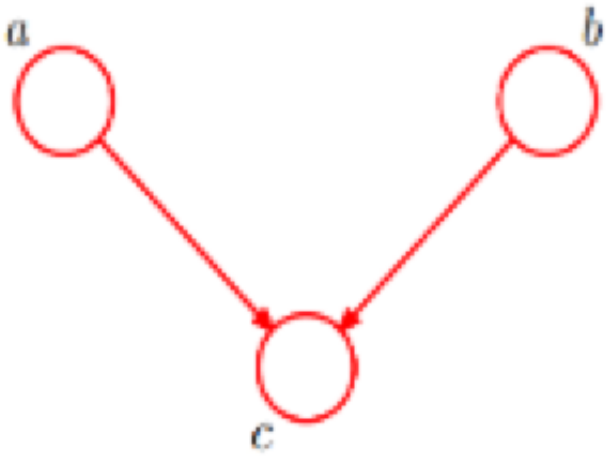
For example, based on the structure above, we could write down the joint distribution of variables x_1 to x_7:

$$P(x_1, x_2, \ldots, x_7) = p(x_7|x_4, x_5)p(x_6|x_4)p(x_5|x_1, x_3)p(x_4|x_2, x_3)p(x_3)p(x_2)p(x_1)$$

1. Assumption: Conditional Independence

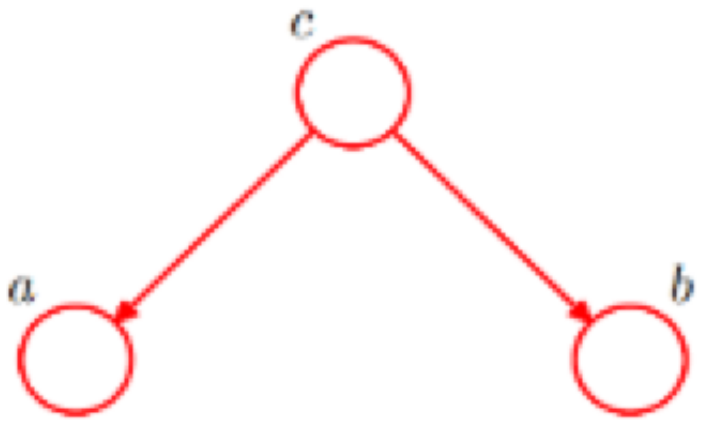
Conditional independence is a very import concept for bayesian models. The bayes net assumption says: “Each variable is conditionally independent of its non-descendants, given its parents.” There are three types of structure of conditional independence.

- Head to Head



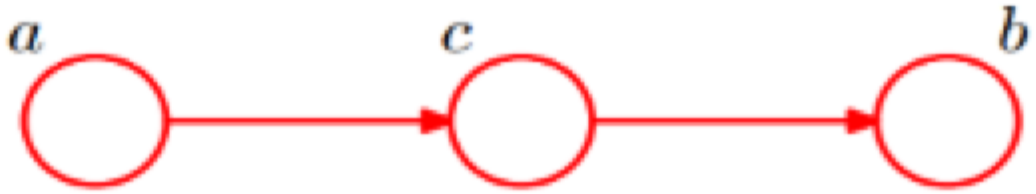
Since we have $p(a,b,c) = p(c|a,b)p(a)p(b)$ and then **only if we don't know the value of** c, $p(a,b) = p(a)p(b)\int_c p(c|a,b) = p(a)p(b)$. In this case, a and b are called head-to-head conditional independent to each other.

- Tail to Tail



Since we have $p(a,b,c) = p(a|c)p(b|c)p(c)$ and then **given the value of** c, $p(a,b|c) = \frac{p(a,b,c)}{p(c)} = p(a|c)p(b|c)$. In this case, a and b are called tail-to-tail conditional independent to each other $a \perp b|c$

- Head to Tail



Since we have $p(a,b,c) = p(b|c)p(c|a)p(a)$ and then **given the value of** c, $p(a,b|c) = \frac{p(a,b,c)}{p(c)} = p(a|c)p(b|c)$. In this case, a and b are called head-to-tail conditional independent to each other $a \perp b|c$

In practise, we would use **d-separation** to analyze conditional independence of variables in bayesian networks. The main idea of d-separation is to turn DAG into moral graph in two steps:

- Step 1: Find all head-to-head structures in the network and then connect each pair of parent

nodes in head-to-head structure via one edge.
- Step 2: Change all directed edges into undirected ones.

If x and y could be blocked by z which means that if we remove z from moral graph, x and y would belong to two different branches, we could say that $x \perp y|z$.

Ref: http://web.mit.edu/jmn/www/6.034/d-separation.pdf

2. Learning

There're two main learning tasks in bayesian networks: structure and conditional probability table.

Structure Learning

Ref: https://ermongroup.github.io/cs228-notes/learning/structure/

There are two major approaches for the structure learning: score-based approach and constraint-based approach.

- Score-based approach

  The score-based approach first defines a criterion to evaluate how well the Bayesian network fits the data, then searches over the space of DAGs for a structure with maximal score.

  - Score metrics

    The score metrics for a structure G and data D can be generally defined as:

    $$S(G|D) = LL(G|D) - f(\theta)|G|$$

    where G is the network, D is the dataset, $LL(G|D)$ is the log-likelihood of the data under the graph structure G. The parameters in Bayesian network G are estimated based on MLE and the log-likelihood score is calculated based on the estimated parameters. If we consider only the log-likelihood in the score function, we will end up with an overfitting structure (namely, a complete graph.) That is why we have the second term in the scoring function. $|G|$is the number of parameters in the graph G and $f(\theta)$ is the bytes used to represent each parameter. **With this extra term, we will penalize the over-complicated graph structure and avoid overfitting**.

    - $f(\theta) = 1$, $S(G|D) = LL(G|D) - |G|$ is called AIC score.
    - $f(\theta) = \frac{1}{2}log(m)$, $S(G|D) = LL(G|D) - \frac{log(m)}{2}|G|$ is called BIC score.
    - $f(\theta) = 0$, $S(G|D) = LL(G|D)$ is just the log likelihood.
  - Search algorithms

    The most common choice for search algorithms are local search and greedy search.

    - Local search: It starts with an empty graph or a complete graph. At each step, it attempts to change the graph structure by a single operation of adding an edge, removing an edge or reversing an edge. (Of course, the operation should preserve the acyclic property.) If the score increases, then it adopts the attempt and does the change, otherwise it makes another attempt.
    - Greedy search: we first assume a topological order of the graph. For each variable, we restrict its parent set to the variables with a higher order. While searching for parent set for each variable, it takes a greedy approach by adding the parent that

increases the score most until no improvement can be made.

- Constraint-based approach

  The constraint-based case employs the independence test to identify a set of edge constraints for the graph and then finds the best DAG that satisfies the constraints. For example, we could distinguish tail to tail structure and head to tail structure by doing an independence test for the two variables on the sides conditional on the variable in the middle. This approach works well with some other prior (expert) knowledge of structure but requires lots of data samples to guarantee testing power. So it is less reliable when the number of sample is small.

Parameter Learning

Given structure, we could estimate parameters via common methods, such as MLE or MAP.

3. Inference

Knowing structure and parameters, we could make inference for features that we want to know by observing some other features (a.k.a posterior probability). In order to do so, we could use variational estimation, Gibbs sampling (example: http://vision.psych.umn.edu/users/schrater/schrater_lab/courses/AI2/gibbs.pdf) or MCMC sampling to compute the posteriror probability.

One thing to note, Gibbs sampling or MCMC sampling (*will cover these in another session*) would generate samples directly and then we could use these samples to estimate the posterior distribution. Given posterior distribution, we could make inference as we want.

Good to see: http://www.cs.cmu.edu/~awm/15781/slides/bayesinf05a.pdf and *https://www.youtube.com/watch?v=TuGDMj43ehw* and *https://zhuanlan.zhihu.com/p/30139208*