CS 561 Artificial Intelligence
Lecture # 19-21

Learning Probabilistic Models

Rashmi Dutta Baruah

Department of Computer Science & Engineering

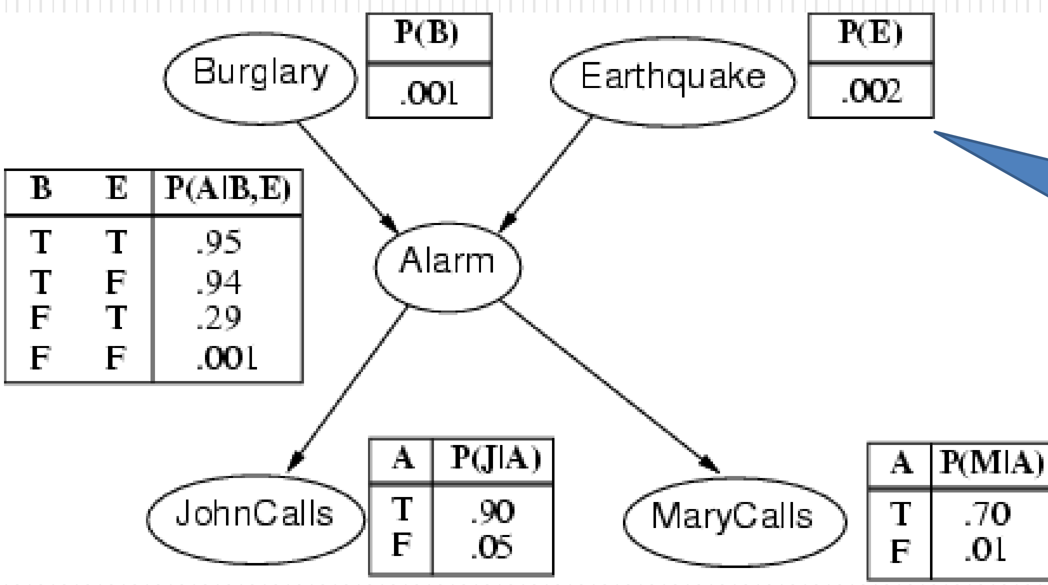
IIT Guwahati

Outline

- Learning Parameters of Bayesian Network
  - Maximum-likelihood approach
  - Bayesian approach
- Learning structure of Bayesian network
- Density estimation with non-parametric models
- Learning with Hidden variables
  - Expectation Maximization algorithm

Learning Parameters of Bayes Net

- Density estimation : the general task of learning a probability model, given data (defined later) that are assumed to be generated from that model.
- The structure of the Bayesian network is fixed and the task is to learn the conditional probabilities- parameter learning



How to get these probabilities?

Parameter Learning

- Learning with complete data
  - Maximum-likelihood parameter learning
  - Bayesian parameter learning
- Learning with hidden variables
  - Expectation Maximization Algorithm

Parameter learning: Candy example

- Candy Example
  - Two flavours: cherry and lime
  - Candy wrapped in the same opaque paper, regardless of flavour
- Data: evidence – instantiations of some or all of the random variables describing the domain
  - $D_i, i = 1{:}N$ represents data and D_i is a random variable with possible values $cherry$ and $lime$
- Hypothesis: probabilistic theories of how the domain works
  - H: possible values $h_1 \ldots h_5$
  - h_1: 100% $cherry$
  - h_2: 75% $cherry$ + 25% $lime$
  - ….
  - h_5: 100% $lime$

Parameter Learning: Candy Example

- We assume that the **observations are complete**; i.e., each data point contains values for every variable in the probability model being learned.

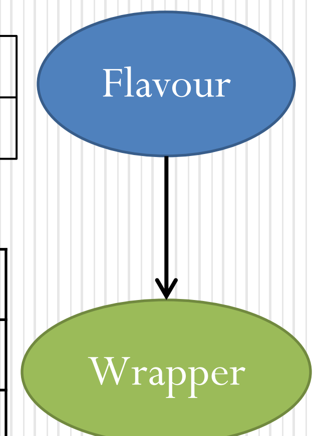
Bayesian networks for Candy example

$P(F = cherry)$
θ

$P(F = cherry)$
θ

$P(F = cherry)$
θ

F	$P(W = red \mid F)$
$cherry$	θ_1
$lime$	θ_2



Maximum-likelihood parameter learning – Discrete models

- Given the candy bag, what is the lime-cherry proportion?
  - It (hypothesis) could be anywhere between 0 and 1.
  - θ, proportion of cherry candies and h_θ is the hypothesis

- Suppose we unwrap N candies:
  - c are cherries and $l = N - c$ are limes
- Likelihood of this dataset is (assuming i.i.d.):

$$P(\mathbf{d}|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^c.(1-\theta)^l$$

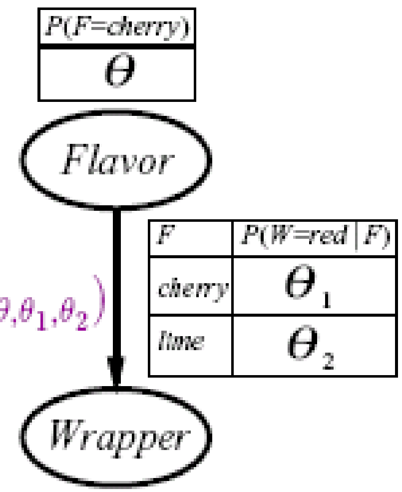
$P(F = cherry)$
θ

- Maximum-likelihood hypothesis is given by the value of θ that maximizes $P(\mathbf{d}|h_\theta)$ (easier with **log likelihood**)

Maximum-likelihood parameter learning – Discrete models

Multiple parameters

Red/green wrapper depends probabilistically on flavor:



Likelihood for, e.g., cherry candy in green wrapper:

$$
\begin{aligned}
&P(F = cherry, W = green | h_{\theta,\theta_1,\theta_2}) \\
&= P(F = cherry | h_{\theta,\theta_1,\theta_2}) P(W = green | F = cherry, h_{\theta,\theta_1,\theta_2}) \\
&= \theta \cdot (1 - \theta_1)
\end{aligned}
$$

N candies, r_c red-wrapped cherry candies, etc.:

$$P(\mathrm{d} | h_{\theta,\theta_1,\theta_2}) = \theta^c (1-\theta)^\ell \cdot \theta_1^{r_c} (1-\theta_1)^{g_c} \cdot \theta_2^{r_\ell} (1-\theta_2)^{g_\ell}$$

$$
\begin{aligned}
L = &[c \log \theta + \ell \log(1-\theta)] \\
&+ [r_c \log \theta_1 + g_c \log(1-\theta_1)] \\
&+ [r_\ell \log \theta_2 + g_\ell \log(1-\theta_2)]
\end{aligned}
$$

(from text book website)

Maximum-likelihood parameter learning – Discrete models

Multiple parameters contd.

Derivatives of L contain only the relevant parameter:

$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \qquad \Rightarrow \quad \theta = \frac{c}{c+\ell}$$

$$\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \qquad \Rightarrow \quad \theta_1 = \frac{r_c}{r_c+g_c}$$

$$\frac{\partial L}{\partial \theta_2} = \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1-\theta_2} = 0 \qquad \Rightarrow \quad \theta_2 = \frac{r_\ell}{r_\ell+g_\ell}$$

With complete data, parameters can be learned separately

(from text book website)

Maximum-likelihood parameter learning – Continuous models

- Learning the parameters of a Gaussian density function on a single variable:

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

Parameters for this model: μ (mean) and σ (standard deviation)

- for observed values: $x_1, \ldots, x_N$, the log likelihood can be given as

$$L = \sum_{j=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x_j-\mu)^2}{2\sigma^2}} = N\left(-log\sqrt{2\pi} - \log\sigma\right) - \sum_{j=1}^{N} \frac{\left(x_j - \mu\right)^2}{2\sigma^2}$$

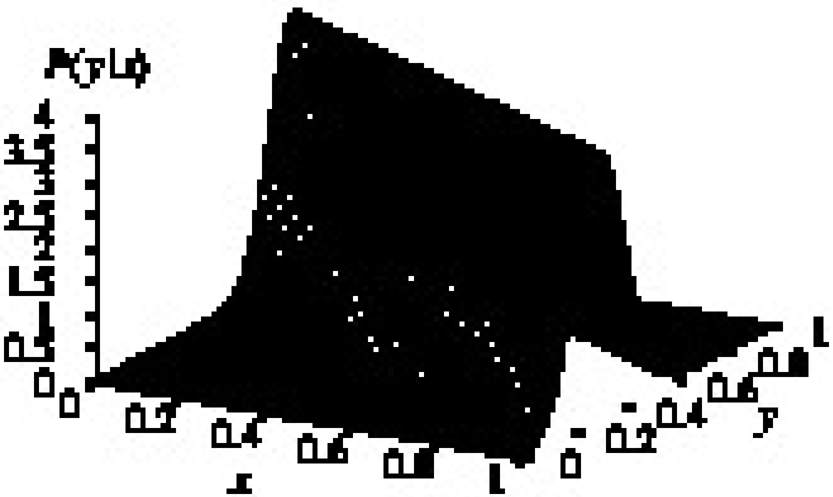
Maximum-likelihood parameter learning – Continuous models

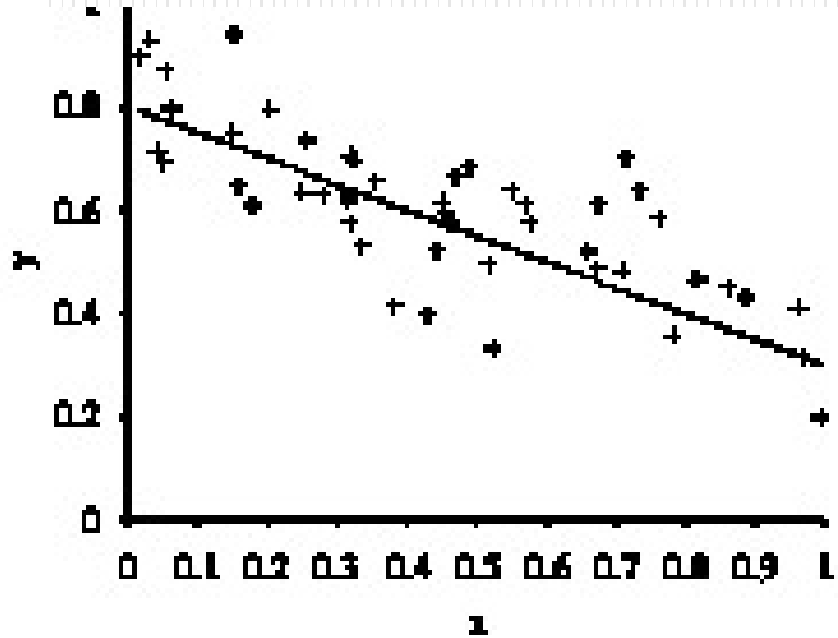
- Taking the derivative and setting to zero,

$$\frac{\partial L}{\partial \mu} = -\frac{1}{\sigma^2}\sum_{j=1}^{N}(x_j - \mu) = 0 \Rightarrow \mu = \frac{\sum_j x_j}{N} \quad \frac{\partial L}{\partial \sigma} = -\frac{N}{\sigma} + \frac{1}{\sigma^3}\sum_{j=1}^{N}(x_j - \mu)^2 = 0 \Rightarrow \sigma = \sqrt{\frac{\sum_j (x_j - \mu)^2}{N}}$$

- Maximum-likelihood value of the mean is the sample average and the maximum-likelihood value of the standard deviation is the square root of the sample variance.

Maximum-likelihood parameter learning – Continuous models





- Linear Gaussian Model with one continuous parent X and a continuous child Y.
- Y has Gaussian distribution, mean depends linearly on X and standard deviation is fixed.

Maximum-likelihood parameter learning – Continuous models

- To learn the conditional distribution $P(Y|X)$, maximize the conditional likelihood

$$P(y|x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(y-(\theta_1 x+\theta_2))^2}{2\sigma^2}}$$

Parameters for this model: θ_1 and θ_2 and σ

- For observations, $(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$
  - maximizing $P(y|x)$ w.r.t. θ_1 and θ_2 is same as minimizing the numerator $(y - (\theta_1 x + \theta_2))^2$ i.e. minimizing the sum of squared errors

$$E = \sum_{j=1}^{N} (y_j - (\theta_1 x_j + \theta_2))^2$$

minimizing the sum of squared errors gives the ML solution for a linear fit assuming Gaussian noise of fixed variance

Maximum-likelihood parameter learning

- ML parameter learning approach
  - Write down an expression for the likelihood of the data as a function of the parameter(s).
  - Write down the derivative of the log likelihood with respect to each parameter.
  - Find the parameter values such that the derivatives are zero.

- Maximum-likelihood learning has deficiencies with small datasets.

Bayesian Parameter Learning

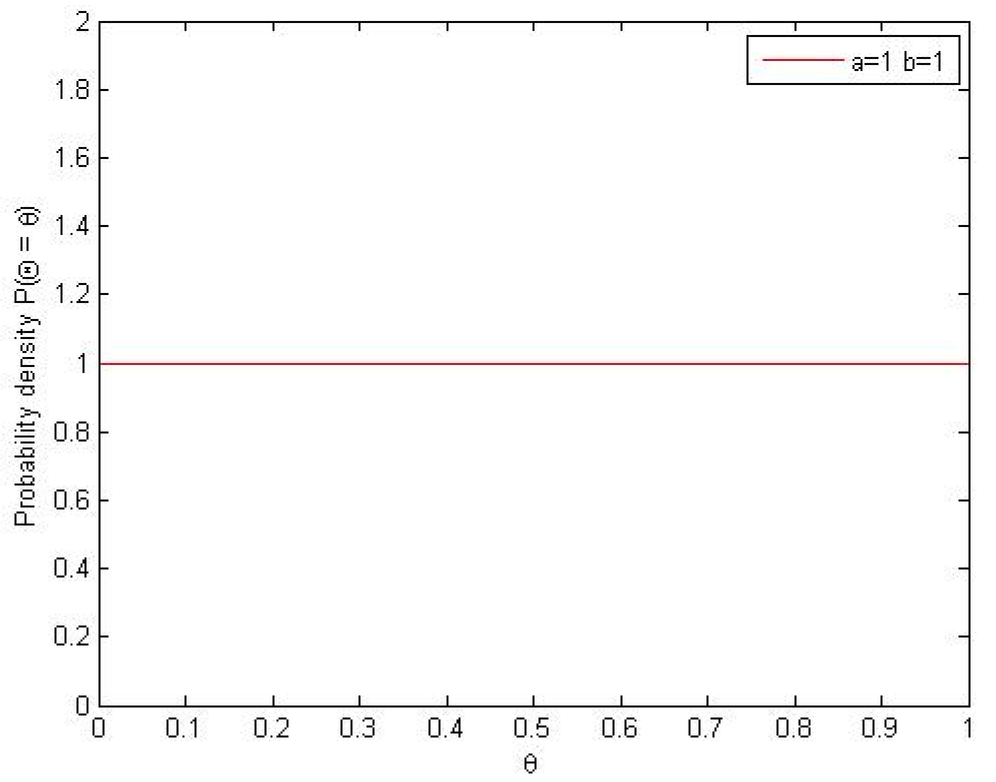
- Bayesian approach to parameter learning
  - define a prior probability distribution over the possible hypothesis (**hypothesis prior**)
  - as data arrives, update the posterior probability distribution

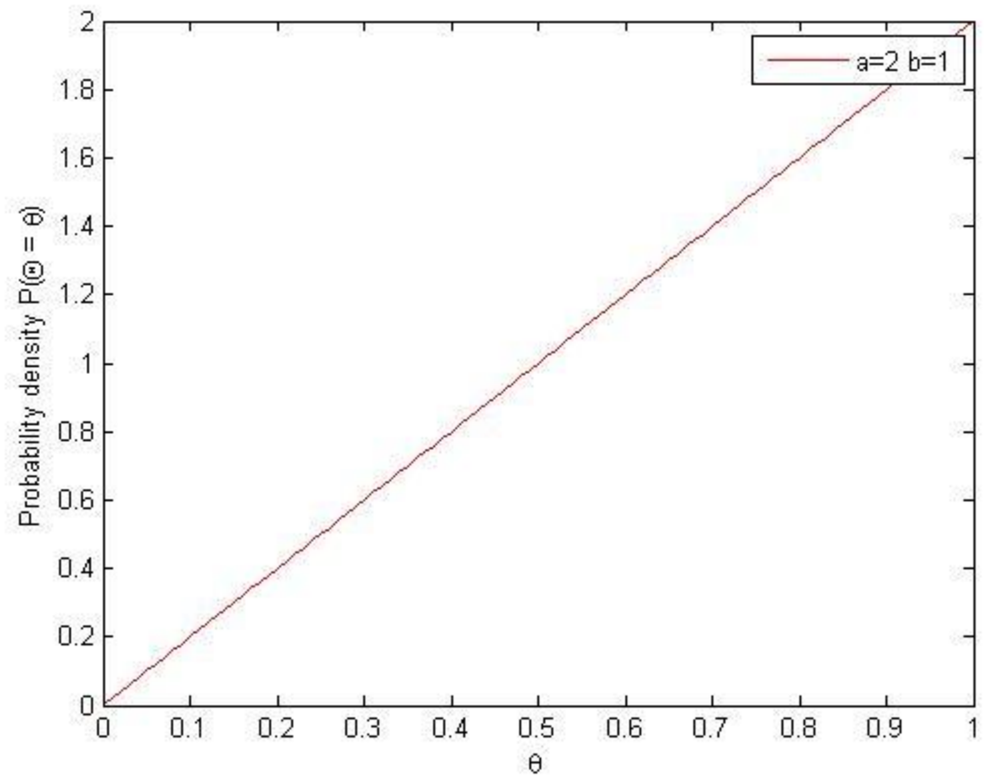
- Bayesian view: θ is the (unknown) value of a random variable Θ that defines the hypothesis space: the hypothesis prior is the prior distribution $\mathbf{P}(\Theta)$
- $P(\Theta = \theta)$ is the prior probability that the bag has a fraction θ of cherry candies
- θ can be any value between 0 and 1, then $\mathbf{P}(\Theta)$ must be a continuous distribution that is non-zero only between 0 and 1 and that integrates to 1.

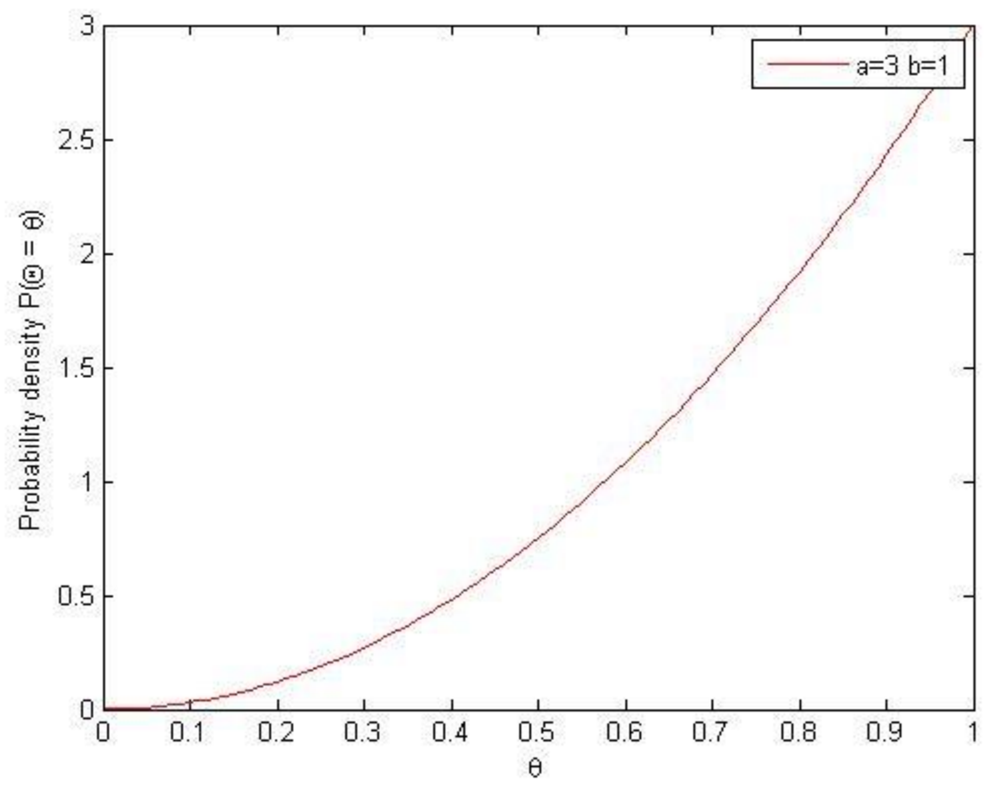
$P(F = cherry)$
θ

Bayesian Parameter Learning

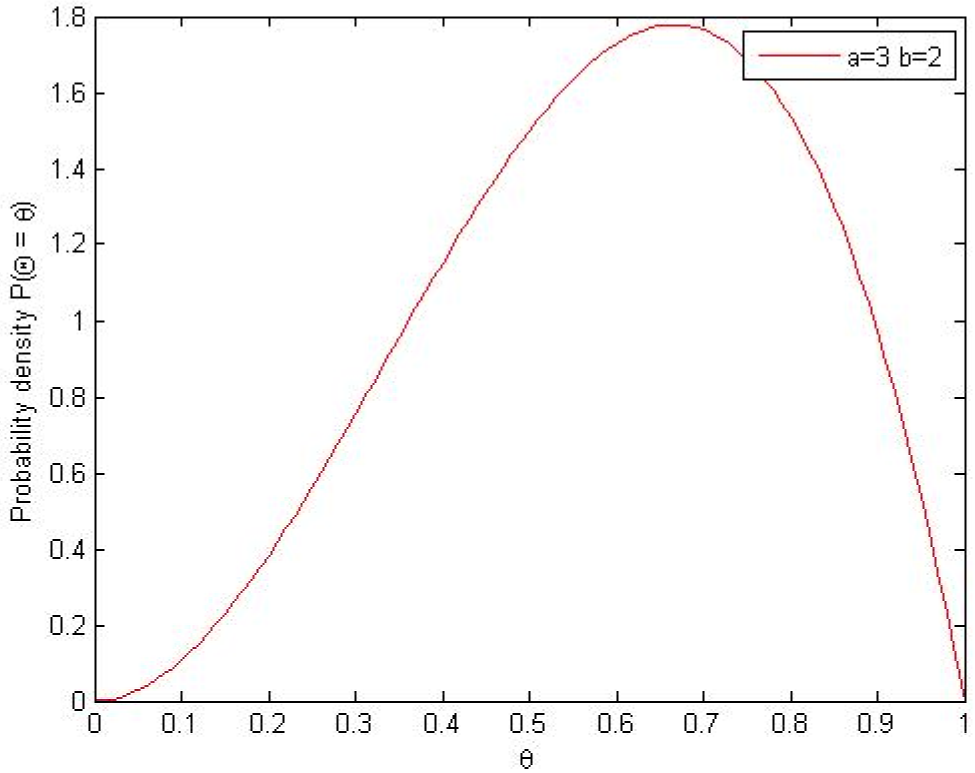
- $P(\theta)$ = beta distribution: candidate prior probability
  - $\text{beta}[a, b](\theta) = \alpha\theta^{a-1}(1-\theta)^{b-1}$ for θ in the range $[0,1]$
  - Mean of the distribution $\frac{a}{a+b}$
  - Larger values of a suggest a belief that Θ is closer to 1 than to 0
  - Larger values of $a + b$ make the distribution more peaked
  - beta distribution is closed under update
- Suppose we observe cherry candy


a=1 b=1
Probability density P(Θ = θ)
θ
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0.1
0.3
0.5
0.7
0.9

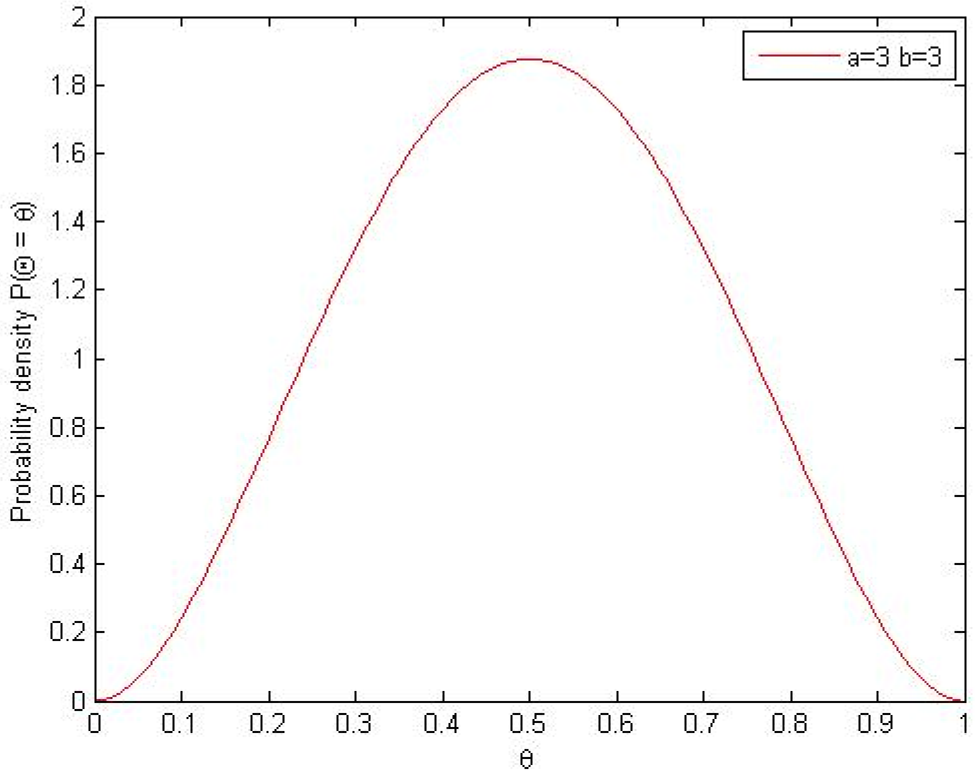

a=2 b=1
Probability density P(Θ = θ)
θ
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0.1
0.3
0.5
0.7
0.9



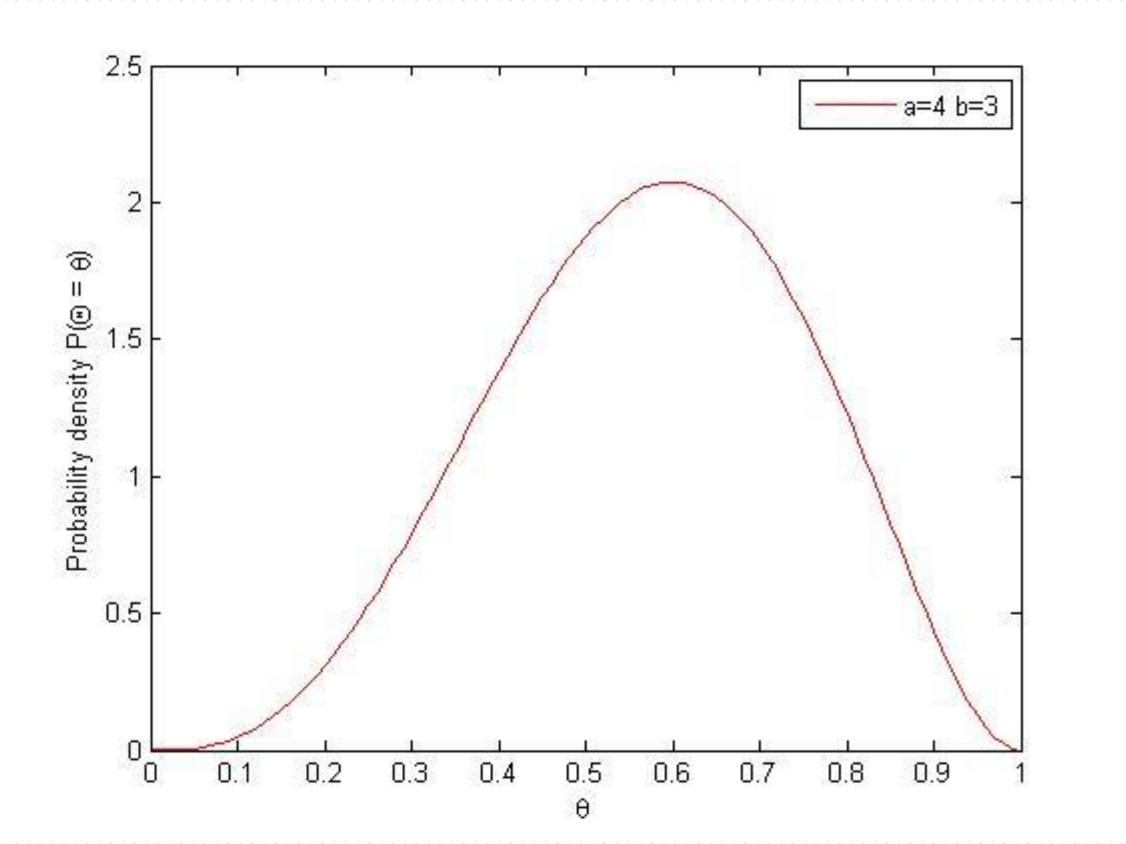
a=3 b=1
Probability density P(Θ = θ)
θ
0
0.5
1
1.5
2
2.5
3
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9



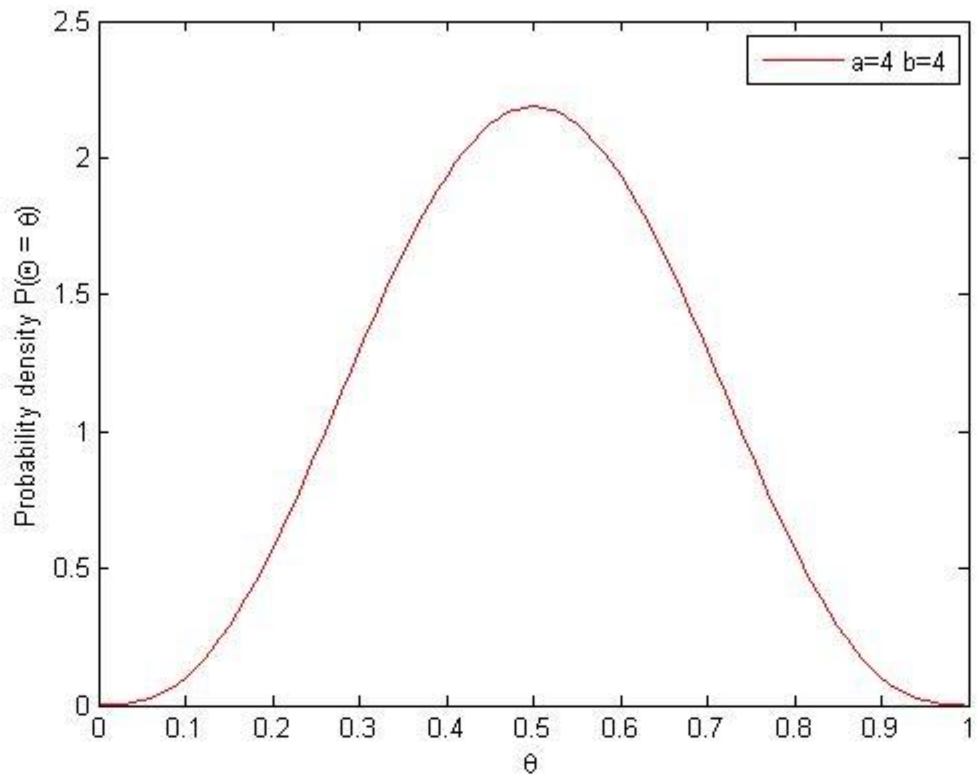
a=3 b=2
Probability density P(Θ = θ)
θ
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0.1
0.3
0.5
0.7
0.9



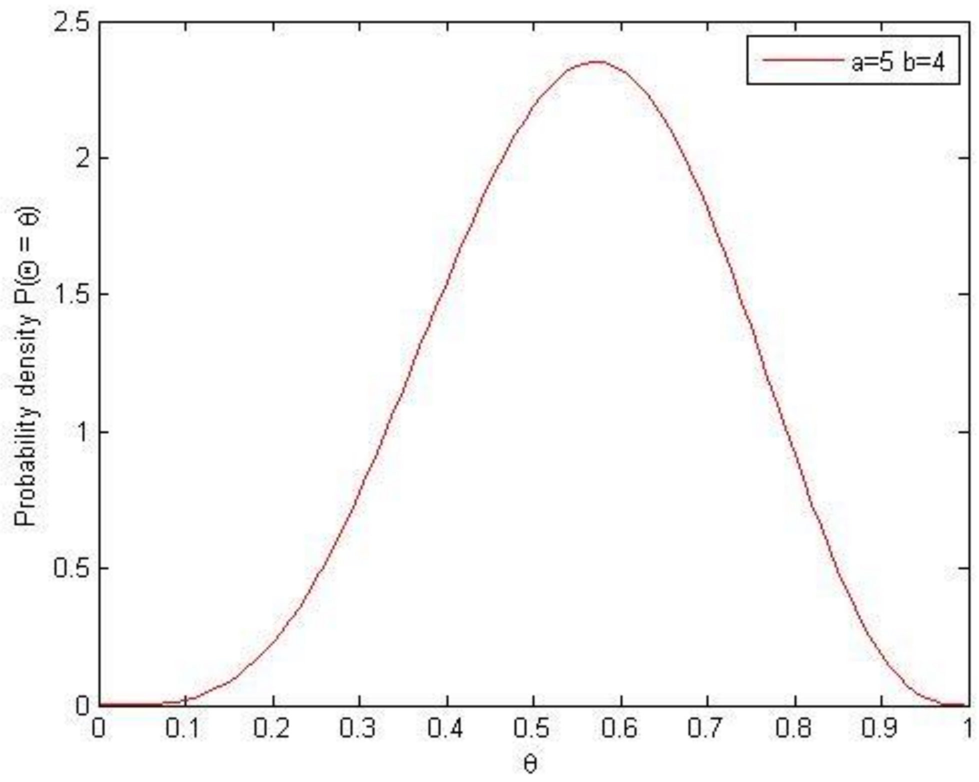
a=3 b=3
Probability density P(Θ = θ)
θ
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0.1
0.3
0.5
0.7
0.9

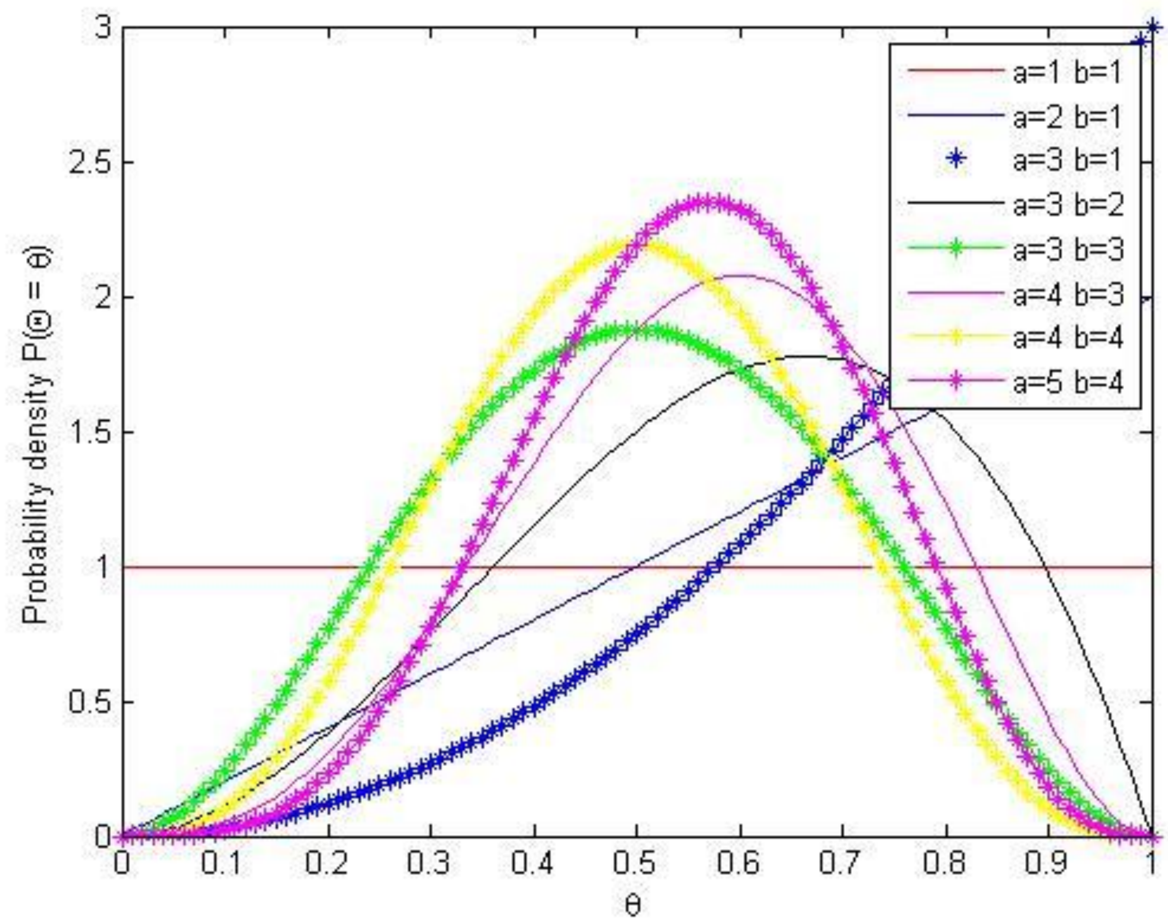


a=4 b=3
Probability density P(Θ = θ)
θ
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.5
1.5
2
2.5



a=4 b=4
Probability density P(Θ = θ)
θ
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.5
1.5
2
2.5


a=5 b=4
Probability density P(Θ = θ)
θ
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.5
1.5
2
2.5


Probability density P(Θ = θ)
θ
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.5
1.5
2
2.5
3
a=1 b=1
a=2 b=1
a=3 b=1
a=3 b=2
a=3 b=3
a=4 b=3
a=4 b=4
a=5 b=4

Density estimation using Non-parametric models

- Density estimation (repeated from slide # 3)
  - general task of learning a probability model, given data that are assumed to be generated from that model

Background

- Continuous random variable X can take an infinite number of possible values, the probability density function describes the relative likelihood that the random variable takes on a given value.

$$P(a \leq X \leq b) = \int_a^b p(x)$$

We are interested in construction of estimate of the density function from the observed data

- For a Discrete random variable, X the probability distribution is given by a list of probabilities associated with each possible outcome, i.e. $P(x_i) = P(X = x_i)$, which satisfies the conditions:

$$\sum_{i=1}^{n} P(x_i) = 1 \text{ and } 0 \leq P(x_i) \leq 1$$

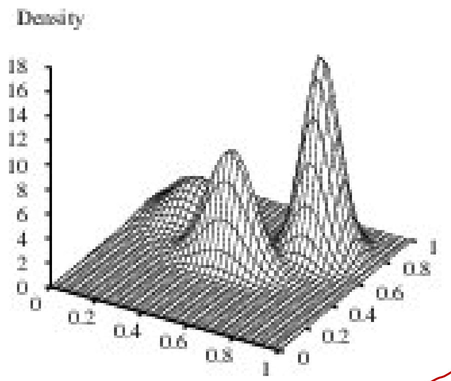
Parametric approach vs. Non-parametric approach

- Parametric approach
  - assumes that the data are drawn from one of a known parametric family of distributions, for example the normal distribution with mean μ and variance σ^2
  - density p underlying the data could then be estimated by finding estimates of μ and σ^2 from the data and substituting these estimates into the formula for the normal density.

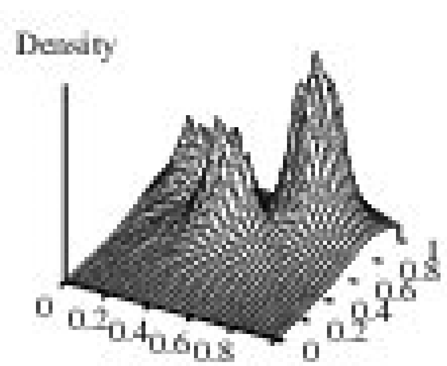
- Non-parametric approach
  - No assumptions about the distribution of data is made
  - Data will speak for themselves in determining the estimate of density p

Non-parametric density estimation

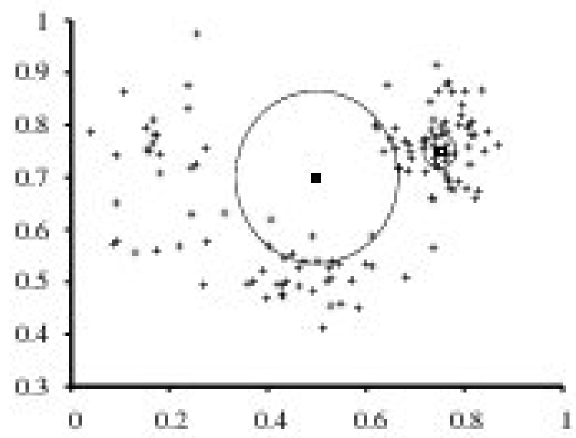
Actual distribution



Estimated distribution



Observed data



Non-parametric density estimation

- Histogram – simplest form of non-parametric density estimation
- Kernel density estimation
- k-nearest neighbour approach

Non-parametric density estimation

- Histogram
  - sample space is divided into a number of bins and density is approximated at the centre of each bin by the fraction of points in the observed data that fall into corresponding bin.
  - Given an *origin* x_0 and a *bin width* h, the *bins* of the histogram are intervals $[x_0 + mh, x_0 + (m+1)h]$ for positive and negative integer m.
  - Histogram is then defined by
    - $\hat{p}\,(x) = \frac{1}{nh}\,(\#\, X_i \textit{ in same bin as } x)$
    - Generally, $\hat{p}\,(x) = \frac{1}{n}\frac{(\#\, X_i \textit{ in same bin as } x)}{(\textit{width of bin containing } x)}$

Histogram

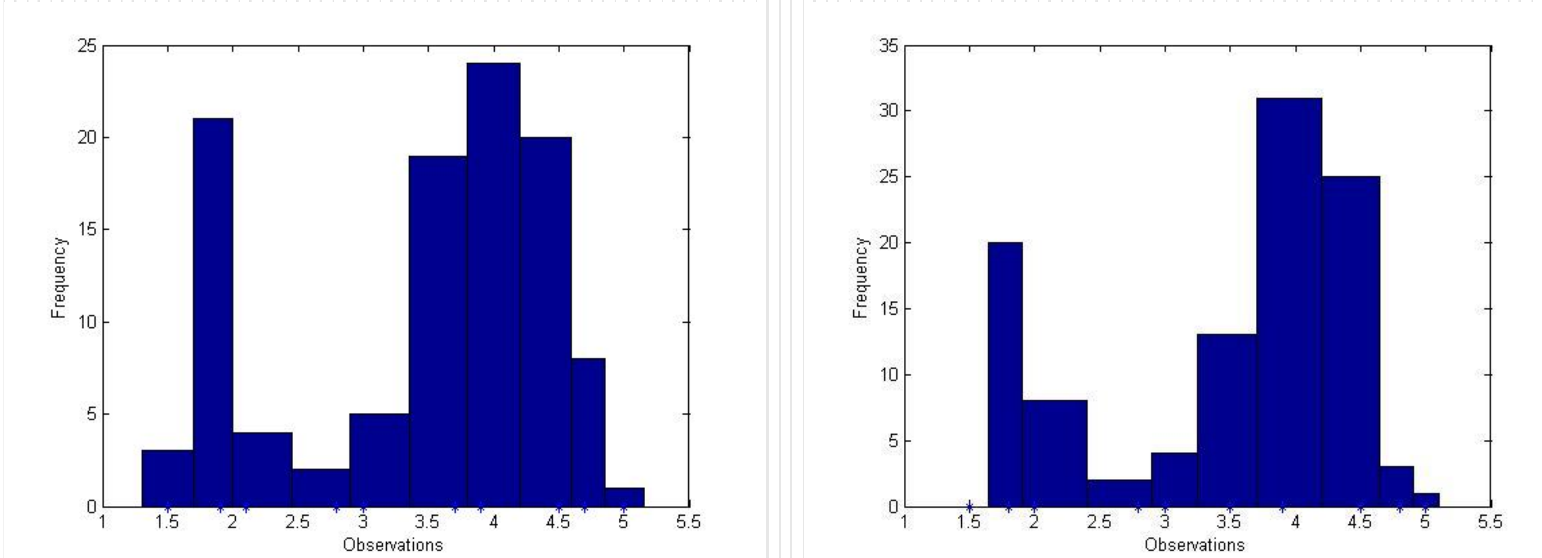


Figure 1. Histograms for the same data with different origins

- histogram requires bin width and origin to be defined
- final shape of the density estimate depends on the origin
- histogram unsuitable for most practical applications, useful for quick visualization of results in one or two dimensions

Kernel density estimate

- Basic idea: take the weighted local density estimates at each observation x_i and then aggregate them to yield an overall density.
- Consider a vector $\mathbf{x}$, the probability P that a vector falls in a region $\mathcal{R}$ can be given as

$$P = \int_{\mathcal{R}} p(\mathbf{x}) dx$$

- Assuming that $\mathcal{R}$ is very small such that $p(\mathbf{x})$ does not vary much within it, we can write

$$P = \int_{\mathcal{R}} p(\mathbf{x}) dx \approx p(\mathbf{x}) \int_{\mathcal{R}} dx = p(x)V$$

where V is the "volume" of region $\mathcal{R}$

Kernel density estimate

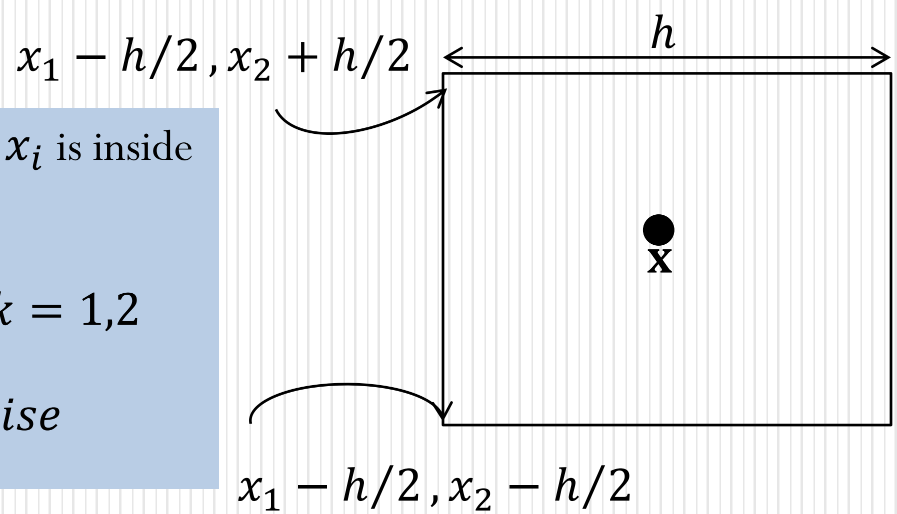
- Suppose that n samples $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ are independently drawn according to the probability density function $p(\mathbf{x})$, and there are k out of n samples falling within the region $\mathcal{R}$, we have

$$P = k/n$$

- Thus, the estimate of $p(\mathbf{x})$ can be given as

$$\hat{p}(\mathbf{x}) = \frac{k}{n}/V$$

- Consider that $\mathcal{R}$ is a square with length of edge $= h$



Let us define a function that indicates whether x_i is inside the square or not.

$$\emptyset\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right) = \begin{cases} 1 & \frac{|x_{ik} - x_k|}{h} \leq 1/2, k = 1,2 \\ 0 & otherwise \end{cases}$$

Kernel density estimate (KDE)

- # samples within the region $\mathcal{R}$ out of n samples is given by

$$k = \sum_{i=1}^{n} \emptyset\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)$$

- The density estimate for 2-D is given by

$$\hat{p}(\mathbf{x}) = \frac{k/n}{V} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^2}\emptyset\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)$$

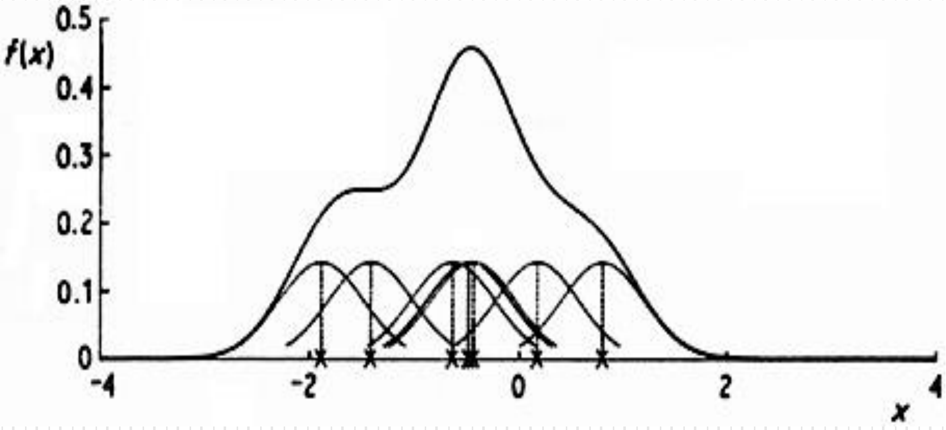
Kernel (or window) function)

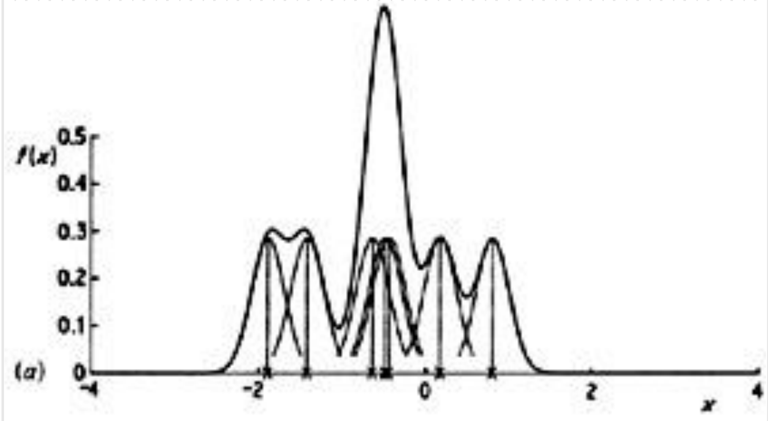
where h is the window width, also called the smoothing parameter or bandwidth.

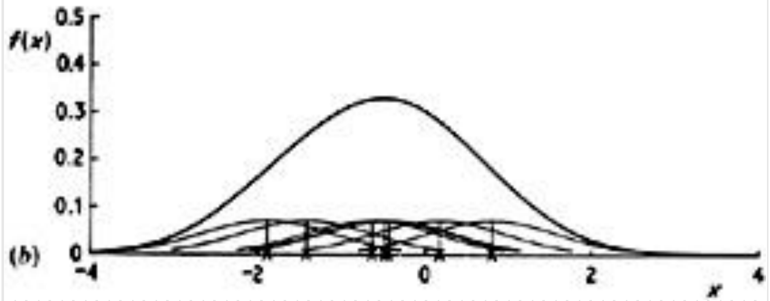
- For d-dimension

$$\hat{p}(\mathbf{x}) = \frac{k/n}{V} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\emptyset\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)$$

Kernel density estimate







1-D Gaussian function is used here for 1-D with various values of bandwidth parameter.

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma}}exp\left(-\frac{(x_i-x)^2}{2\sigma^2}\right)$$

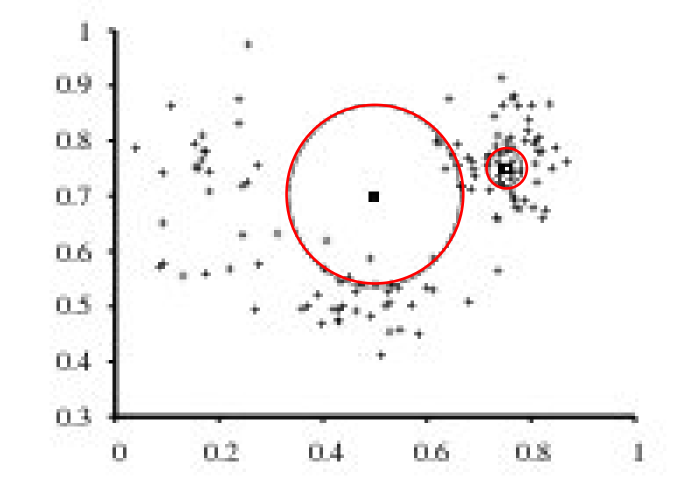
k-nearest neighbour

- Recall the generic expression for density estimation

$$\hat{p}(\mathbf{x}) = \frac{k}{n}/V$$

- In KDE, V is fixed and that determines k, the number of samples inside V.
- In k-nearest neighbour approach, k is fixed and we find the V that contains k points inside.

10 nearest neighbours of square points



Learning structure of the Bayesian Networks

Bayesian Belief Network Training

Algorithm 3 PC algorithm for skeleton learning.

Start with a complete undirected graph G on the set $\mathcal{V}$ of all vertices.
$i = 0$
repeat
    for $x \in \mathcal{V}$ **do**
        for $y \in Adj\{x\}$ **do**
            Determine if there a subset $\mathcal{S}$ of size i of the neighbours of x (not including y) for which $x \perp\!\!\!\perp y | \mathcal{S}$. If this set exists remove the $x - y$ link from the graph G and set $\mathcal{S}_{xy} = \mathcal{S}$.
        end for
    end for
    $i = i + 1$.
until all nodes have $\leq i$ neighbours.

From the Book: Bayesian Reasoning and Machine Learning, David Barber (Chapter 9)

Learning structure of the Bayesian Networks

Bayesian Belief Network Training

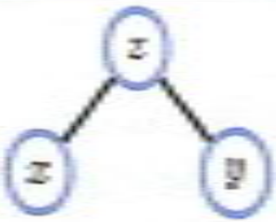
Algorithm 4 Skeleton orientation algorithm (returns a DAG).

1: **Unmarried Collider:** Examine all undirected links $x - z - y$. If $z \notin S_{xy}$ set $x \rightarrow z \leftarrow y$.
2: **repeat**
3: $\quad x \rightarrow z - y \Rightarrow x \rightarrow z \rightarrow y$
4: $\quad$ For $x - y$, if there is a directed path from x to y orient $x \rightarrow y$
5: $\quad$ If for $x - z - y$ there is a w such that $x \rightarrow w$, $y \rightarrow w$, $z - w$ then orient $z \rightarrow w$
6: **until** No more edges can be oriented.
7: The remaining edges can be arbitrarily oriented provided that the graph remains a DAG and no additional colliders are introduced.

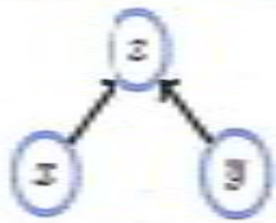
From the Book: Bayesian Reasoning and Machine Learning, David Barber (Chapter 9)

Learning structure of the Bayesian Networks

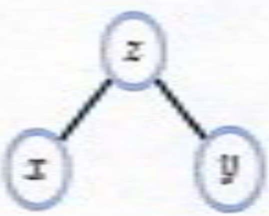
Example 41 (Skeleton orienting).



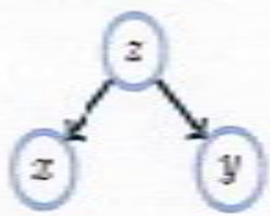
$x \perp\!\!\!\perp y \mid \emptyset \Rightarrow$



If x is (unconditionally) independent of y, it must be that z is a collider since otherwise marginalising over z would introduce a dependence between x and y.



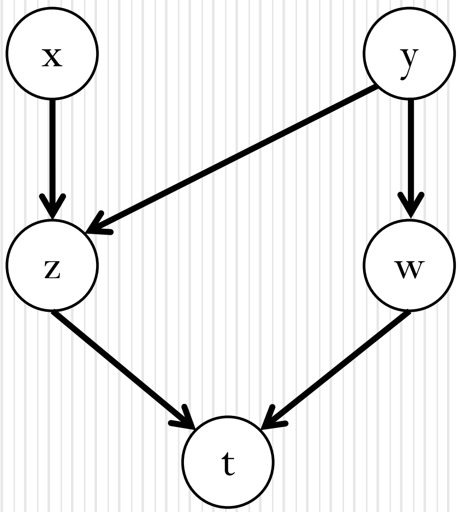
$x \perp\!\!\!\perp y \mid z \Rightarrow$

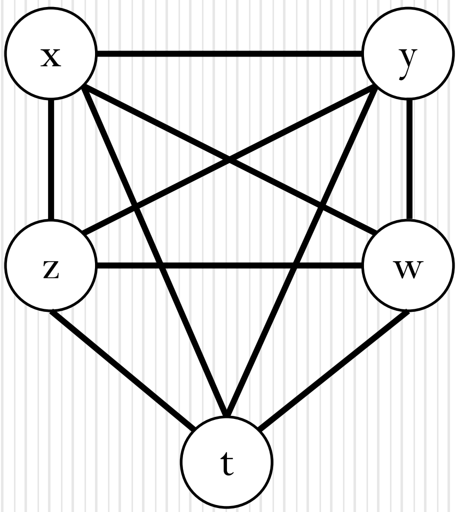


If x is independent of y conditioned on z, z must not be a collider. Any other orientation is appropriate.

From the Book: Bayesian Reasoning and Machine Learning, David Barber (Chapter 9)

Learning structure of the Bayesian Networks

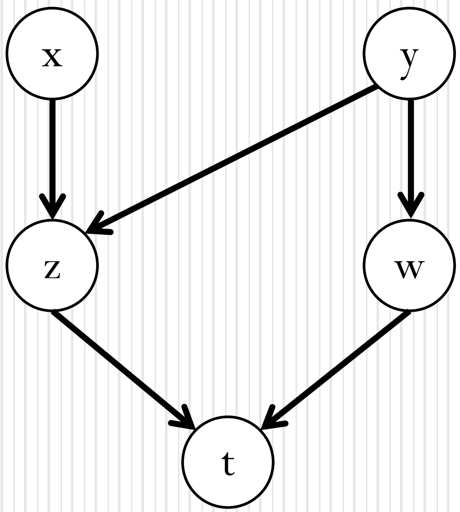


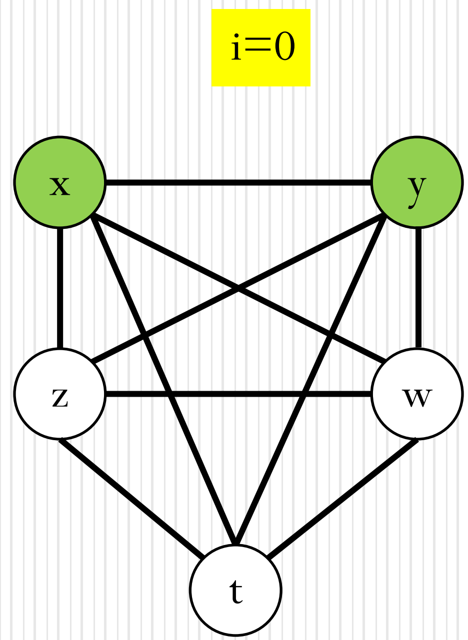


Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

We start with a structure (skeleton) that is fully connected.

Learning structure of the Bayesian Networks





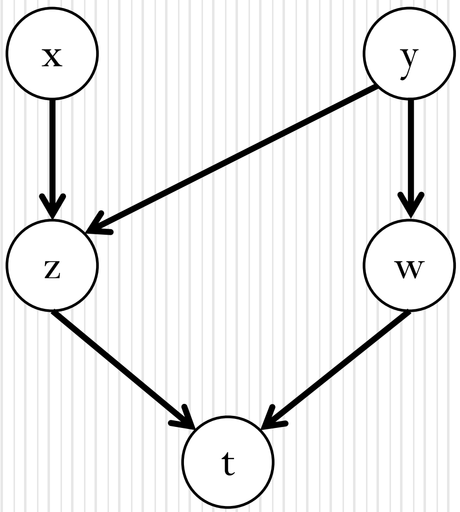
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

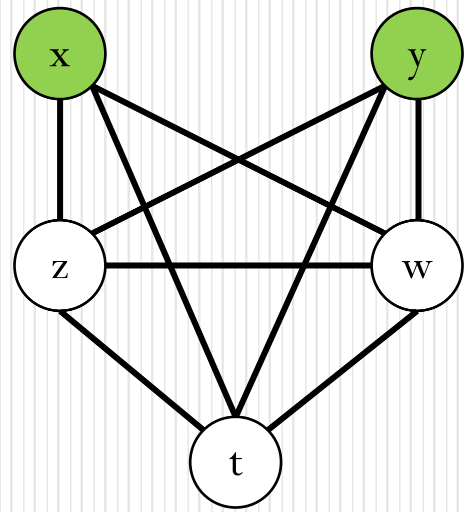
i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks

i=0

$S_{xy} = \{\quad\},$



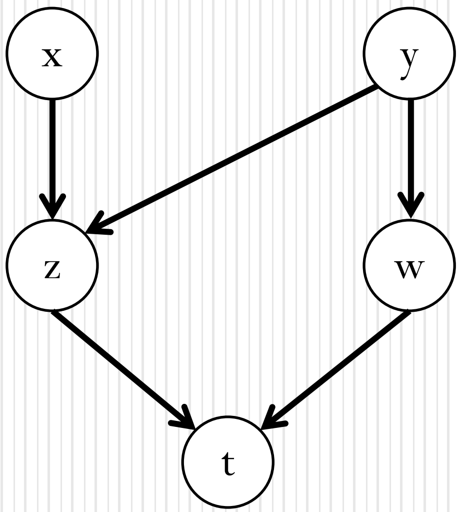


Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

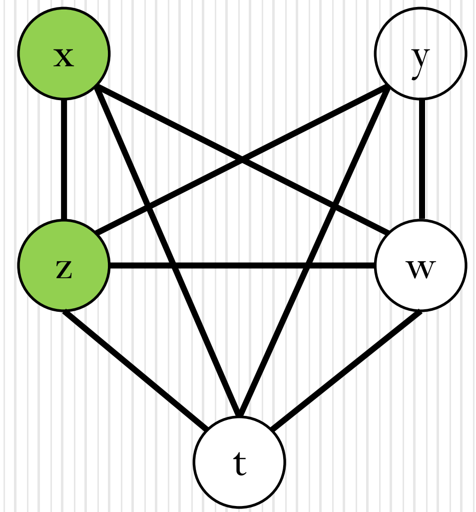
i=0, size of subset S = 0 for which $x \perp y|S$

i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks



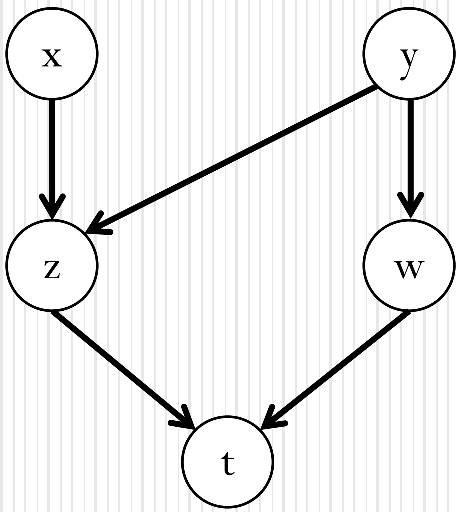
i=0



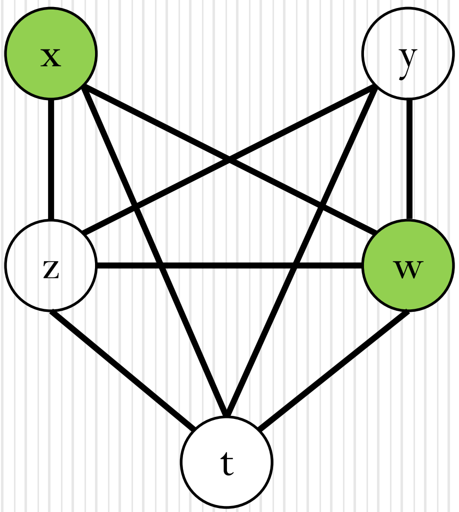
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

i=0, size of subset S = 0 for which $x \perp y | S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks



i=0



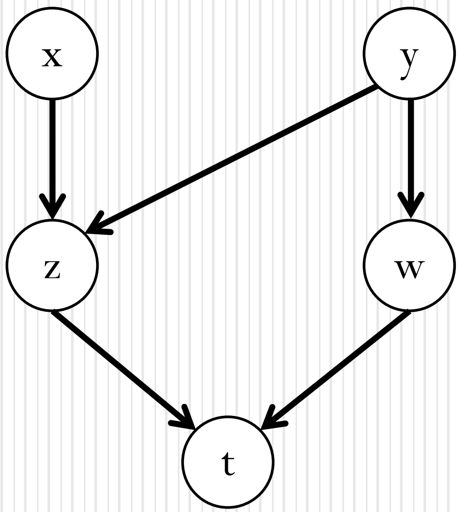
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

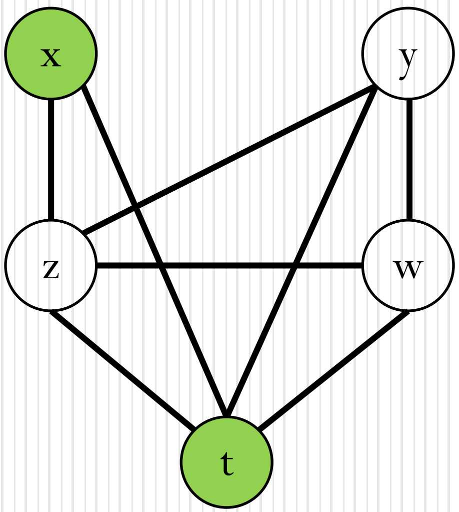
i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks

i=0

$S_{xw} = \{\quad\}$

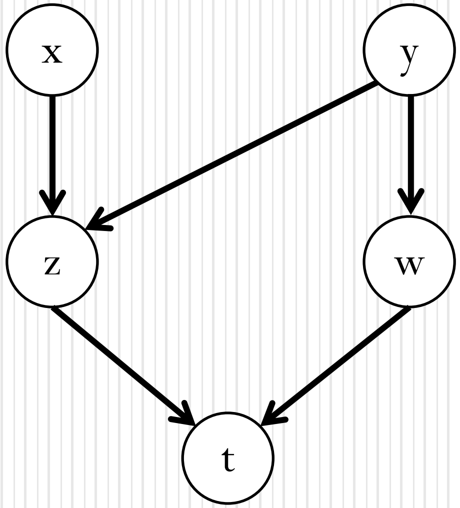




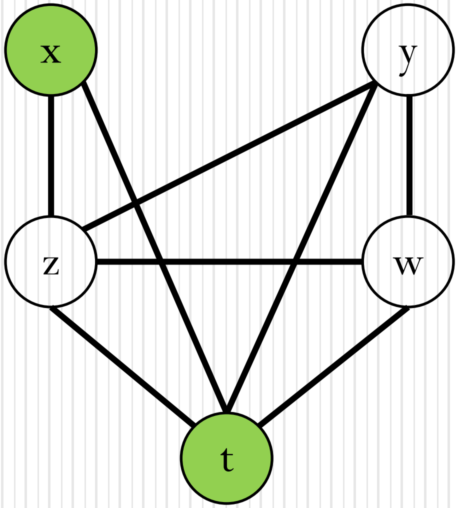
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks



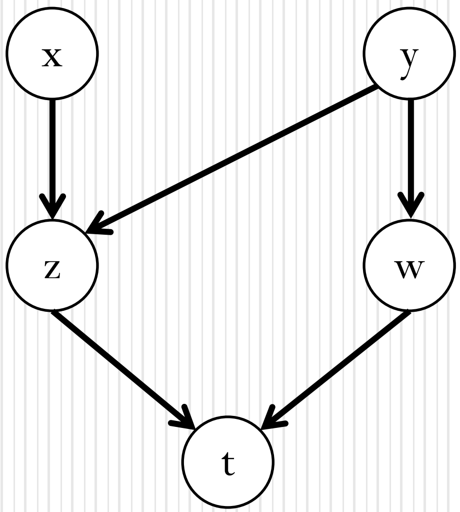
i=0

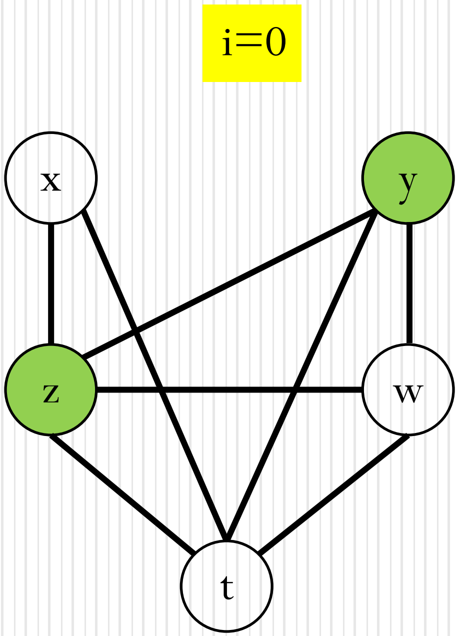


Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks

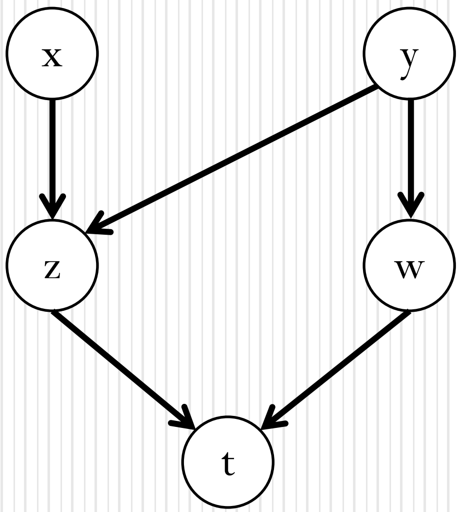


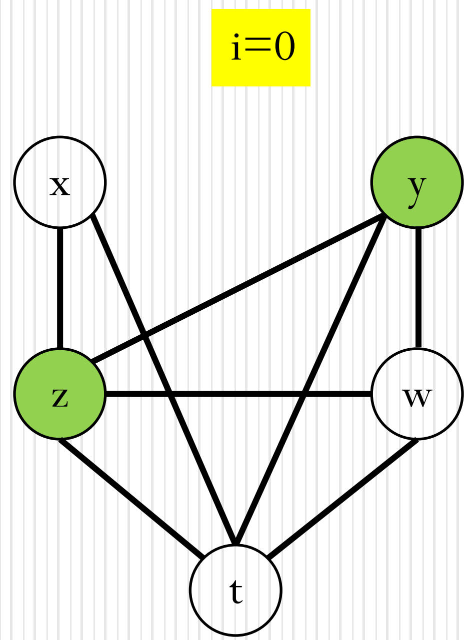


Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

i=0, size of subset S = 0 for which $x \perp y | S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks

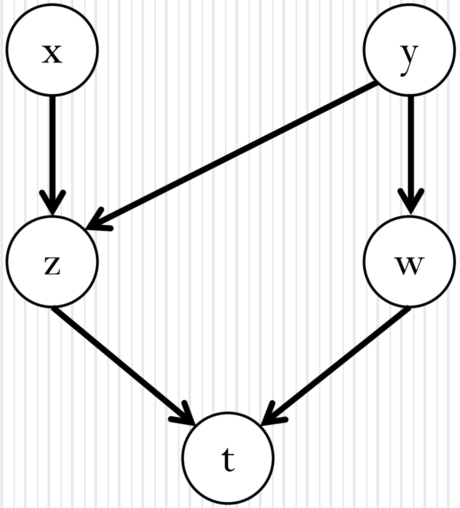




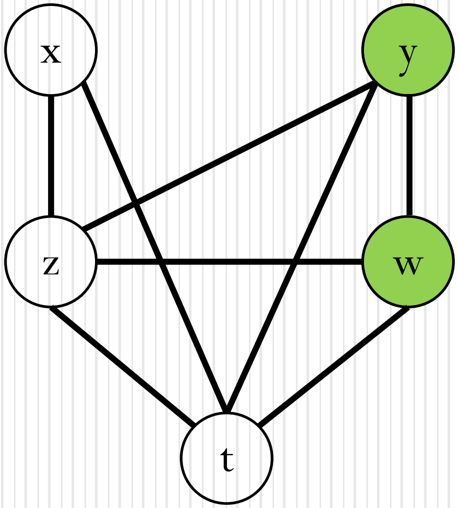
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks



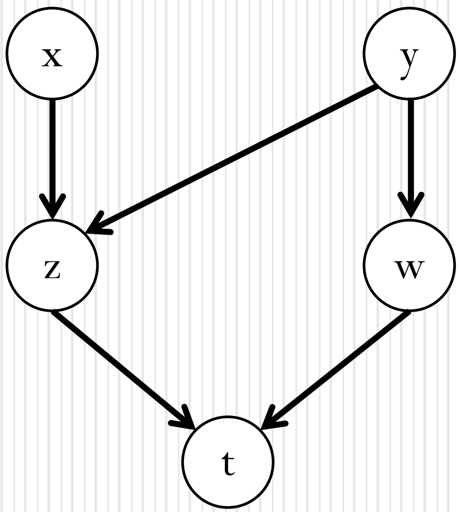
i=0



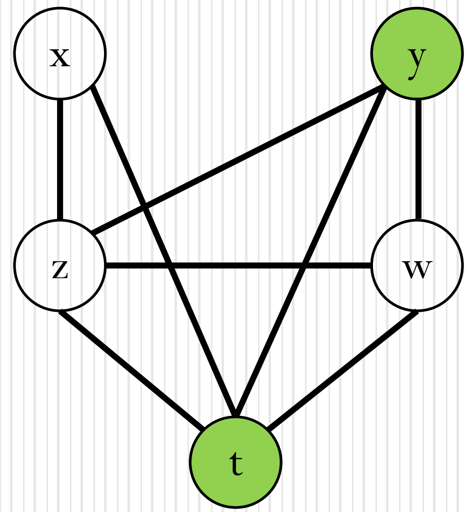
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks



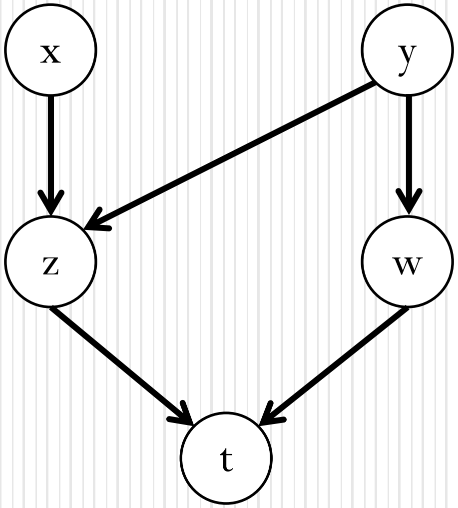
i=0



Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

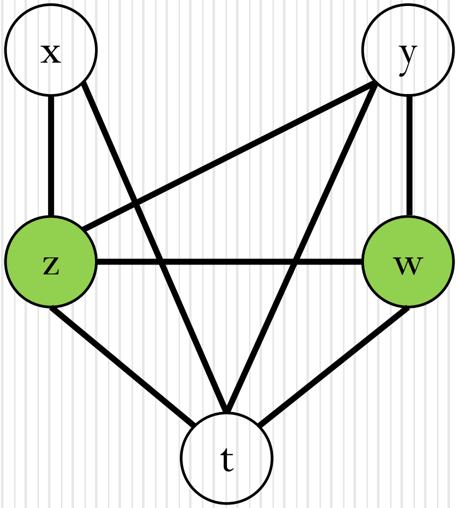
i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks



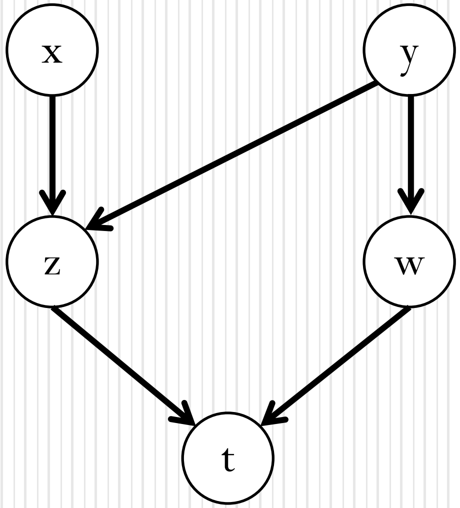
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

i=0



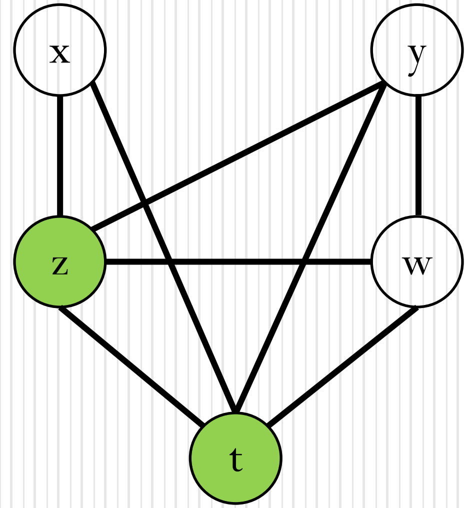
i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

Learning structure of the Bayesian Networks



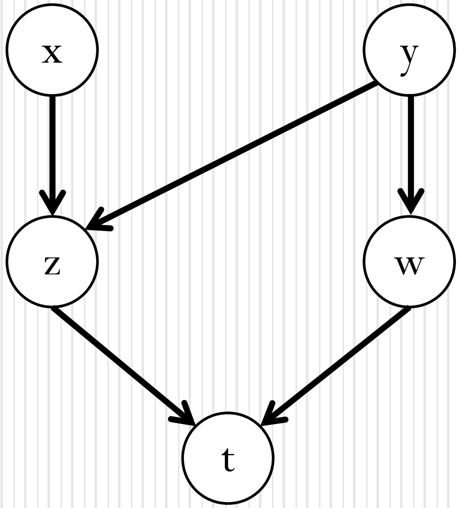
Bayesian Network from which data is assumed to be generated and against which conditional independence tests will be performed.

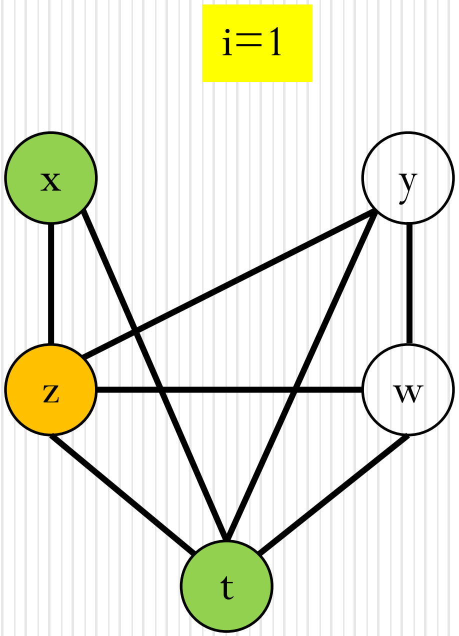
i=0



i=0, size of subset S = 0 for which $x \perp y|S$
i.e. we test all pairs $x - y$ for independence and for such pairs the link is removed

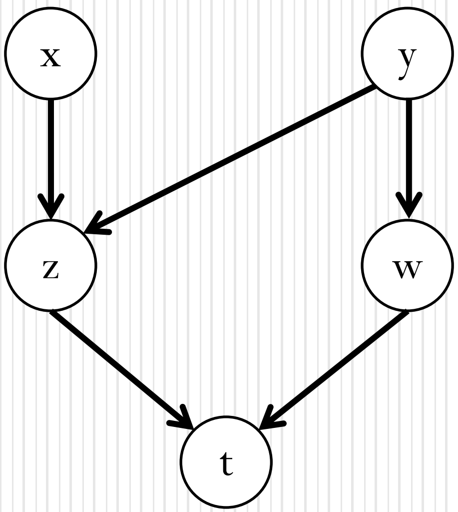
Learning structure of the Bayesian Networks

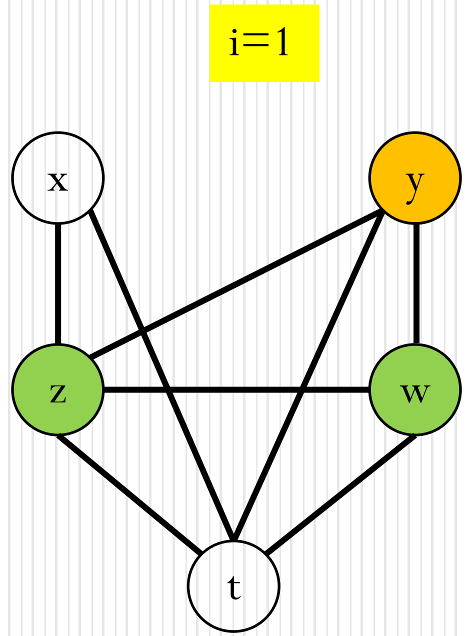




i=1, size of subset S = 1 for which $x \perp y|S \;\; or \; x \perp y|z, \;\; z \in S_{xy}$
we test all pairs $x - y$ for independence, conditioned on single neighbour variable z and for such pairs the link is removed.

Learning structure of the Bayesian Networks

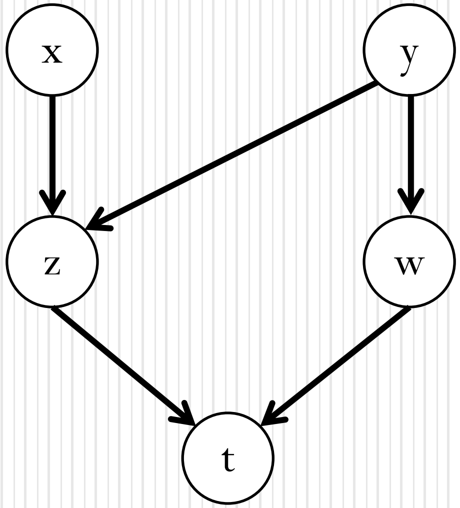


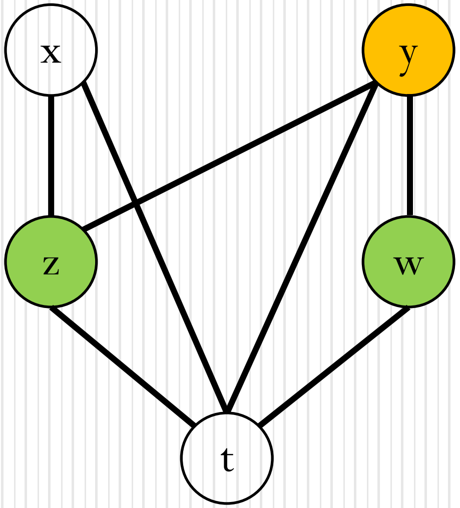


i=0, size of subset S = 1 for which $x \perp y|S \;\; or \; x \perp y|z, \;\; z \in S_{xy}$
we test all pairs $x - y$ for independence, conditioned on single neighbour variable z and for such pairs the link is removed.

Learning structure of the Bayesian Networks

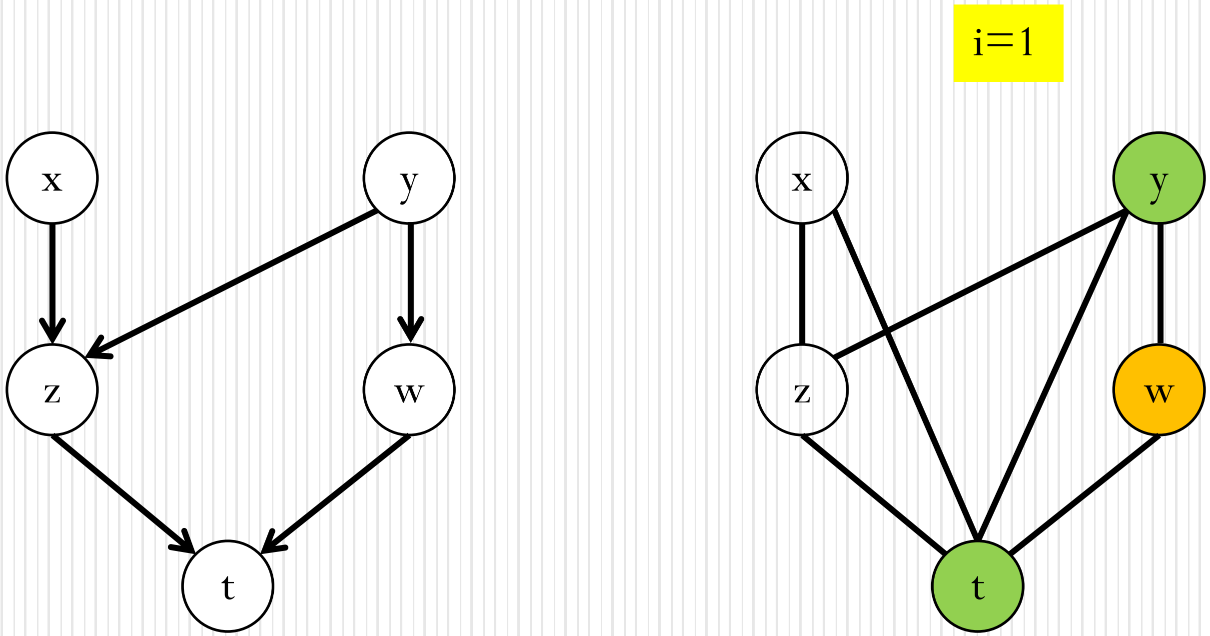
i=1 $S_{zw} = \{y\}$





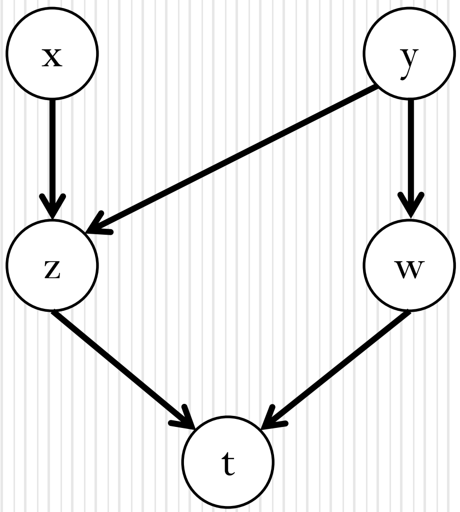
i=0, size of subset S = 1 for which $x \perp y|S \;\; or \; x \perp y|z, \;\; z \in S_{xy}$
we test all pairs $x - y$ for independence, conditioned on single neighbour variable z and for such pairs the link is removed.

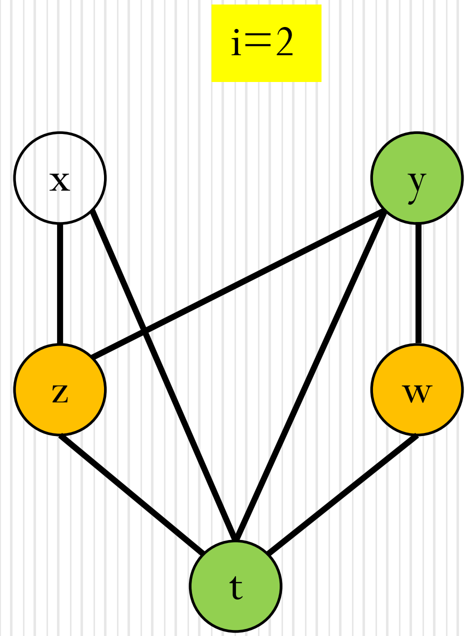
Learning structure of the Bayesian Networks



i=0, size of subset S = 1 for which $x \perp y|S \quad or\ x \perp y|z,\ \ z \in S_{xy}$
we test all pairs $x - y$ for independence, conditioned on single neighbour variable z and for such pairs the link is removed.

Learning structure of the Bayesian Networks



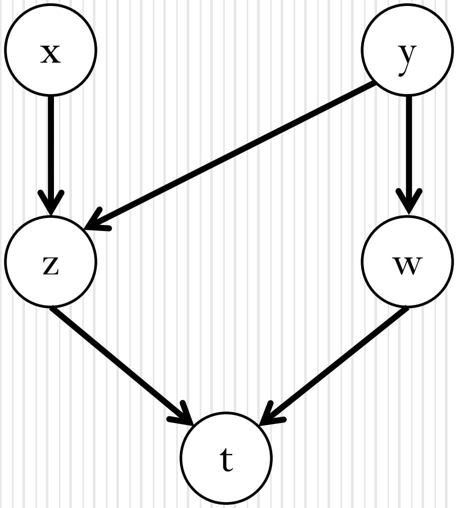


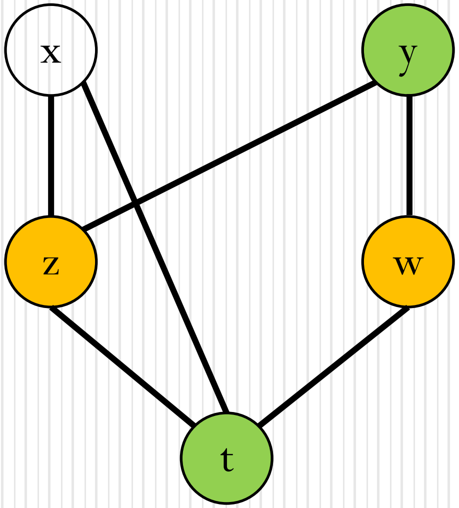
i=2, size of subset S = 2 for which $x \perp y|\{a, b\}$, $S_{xy} = \{a, b\}$
we test all pairs $x - y$ for independence, conditioned on two neighbours, for such pairs the link is removed.

Learning structure of the Bayesian Networks

i=2

$S_{yt} = \{z, w\}$

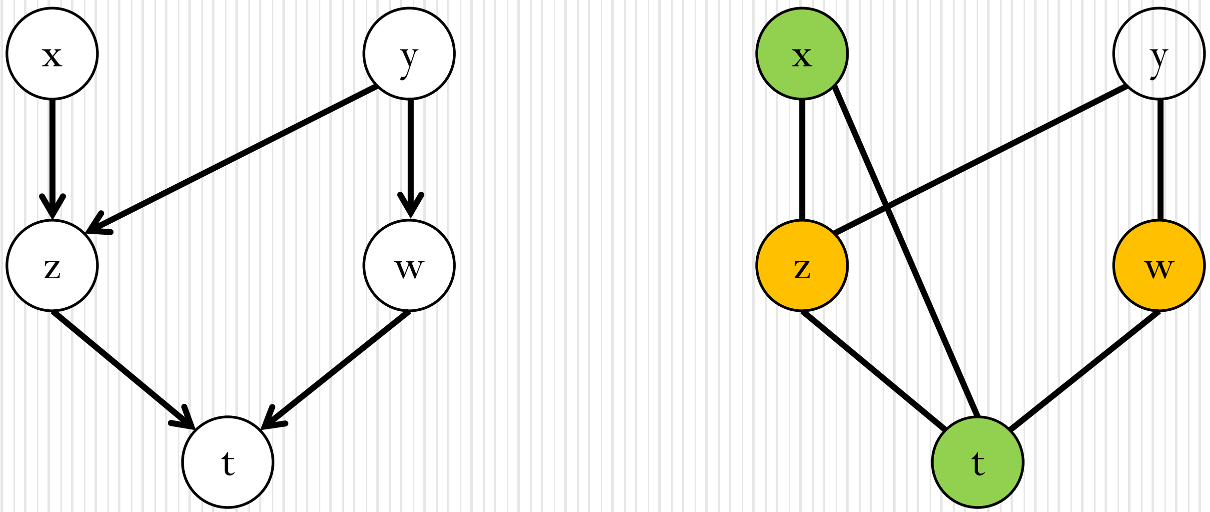




i=2, size of subset S = 2 for which $x \perp y|\{a, b\}$, $S_{xy} = \{a, b\}$
we test all pairs $x - y$ for independence, conditioned on two neighbours,
for such pairs the link is removed.

Learning structure of the Bayesian Networks

i=2

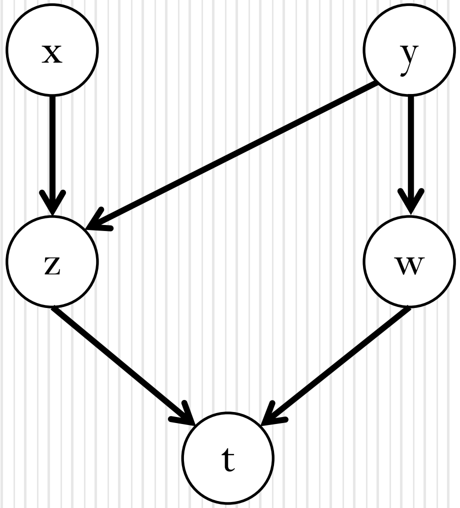


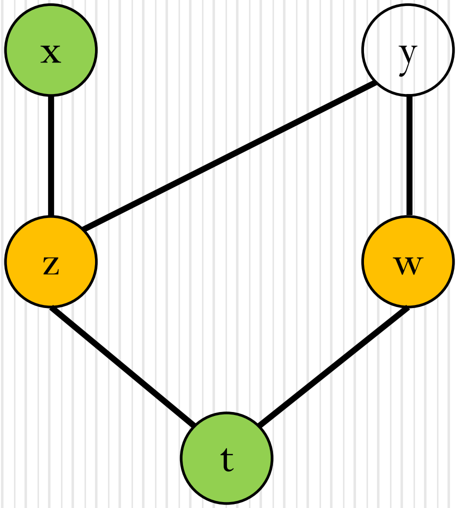
i=2, size of subset S = 2 for which $x \perp y|\{a, b\}$, $S_{xy} = \{a, b\}$
we test all pairs $x - y$ for independence, conditioned on two neighbours, for such pairs the link is removed.

Learning structure of the Bayesian Networks

i=2

$S_{xt} = \{z, w\}$

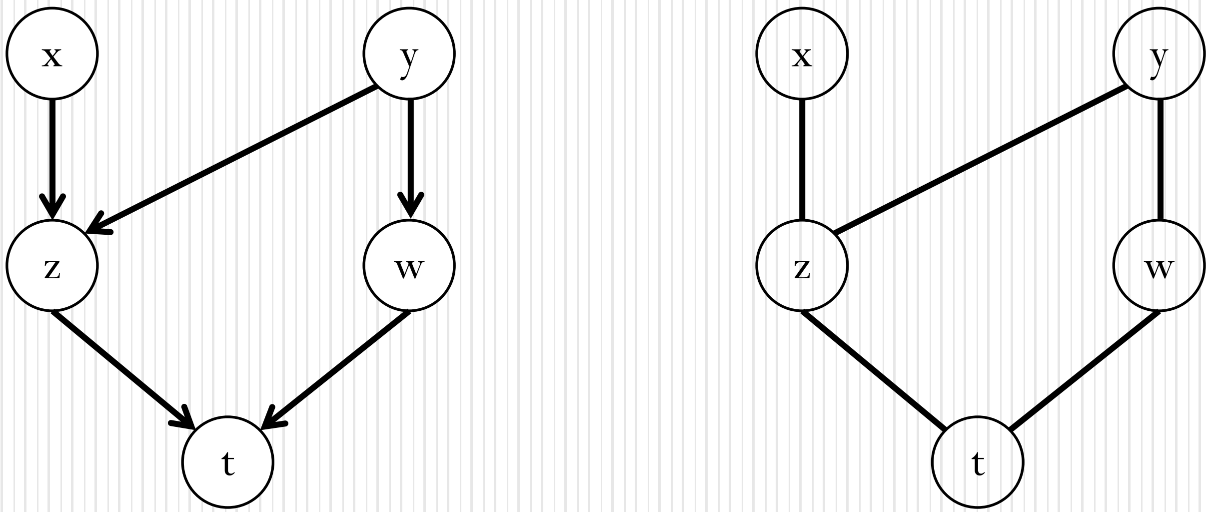




i=2, size of subset S = 2 for which $x \perp y | \{a, b\}$, $S_{xy} = \{a, b\}$
we test all pairs $x - y$ for independence, conditioned on two neighbours, for such pairs the link is removed.

Learning structure of the Bayesian Networks

i=3

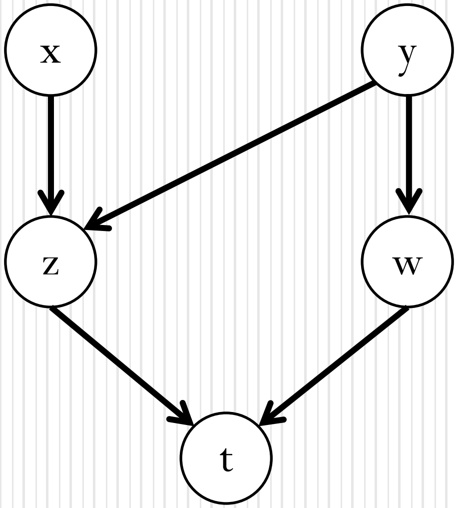


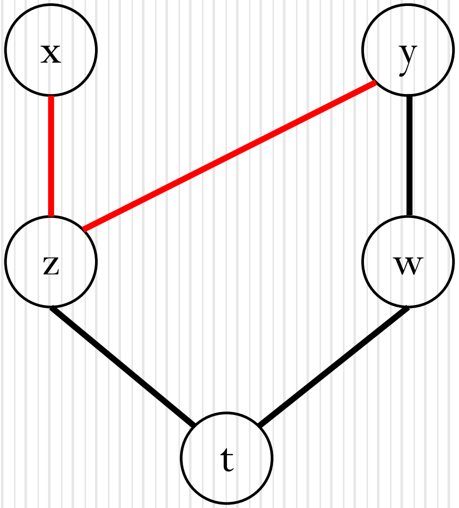
The algorithm terminates after this round as there are no nodes with 3 or more neighbours

$S_{xy} = \{\ \ \}, S_{xw} = \{\ \ \}, S_{zw} = \{y\}, S_{xt} = \{z, w\}, S_{yt} = \{z, w\}$

Learning structure of the Bayesian Networks

Determine the orientation



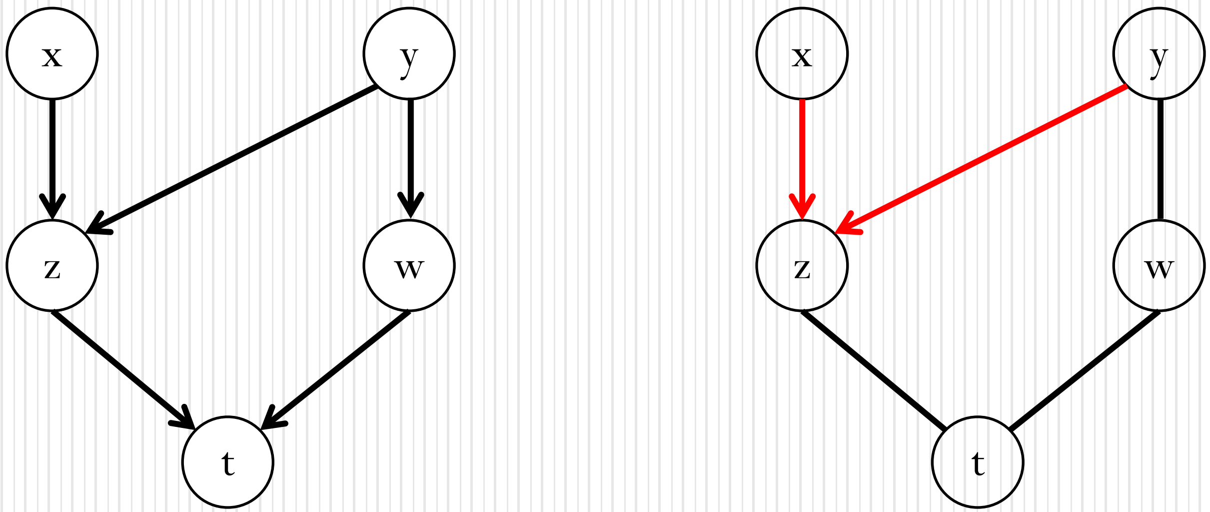


$S_{xy} = \{\quad\}, S_{xw} = \{\quad\}, S_{zw} = \{y\}, S_{xt} = \{z, w\}, S_{yt} = \{z, w\}$

Learning structure of the Bayesian Networks

Determine the orientation

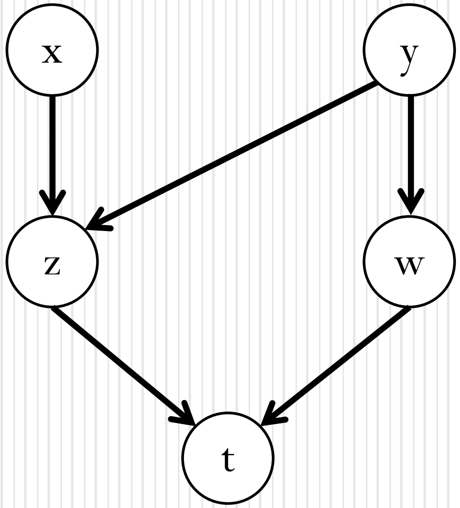
$z \notin S_{xy}$



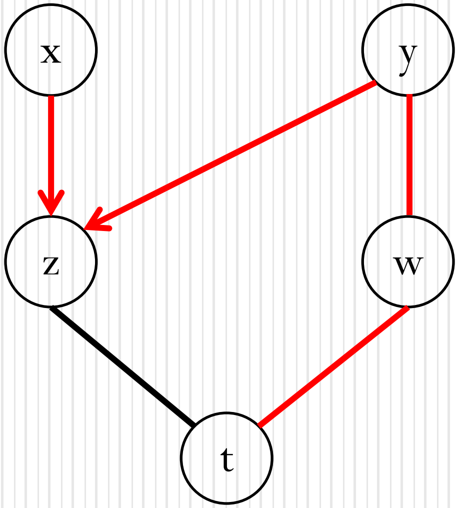
$S_{xy} = \{\ \}, S_{\mathrm{xw}} = \{\ \}, S_{\mathrm{zw}} = \{\mathrm{y}\}, S_{\mathrm{xt}} = \{\mathrm{z}, \mathrm{w}\}, S_{\mathrm{yt}} = \{\mathrm{z}, \mathrm{w}\}$

Learning structure of the Bayesian Networks

Determine the orientation



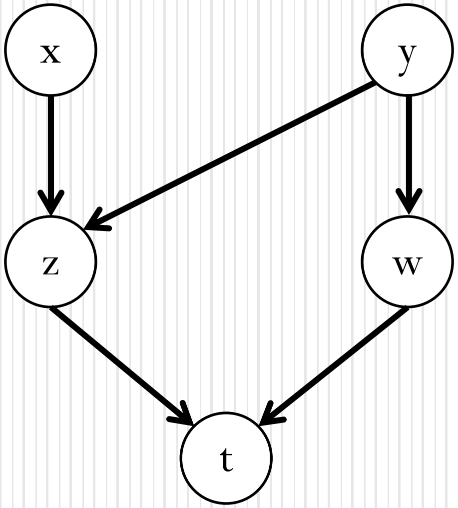
$w \in S_{yt}$



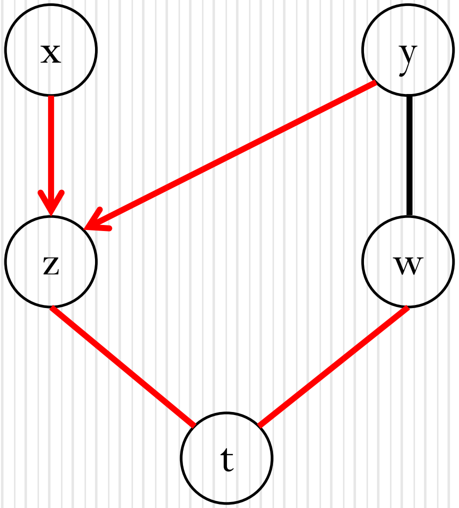
$S_{xy} = \{\ \}, S_{\mathrm{xw}} = \{\ \}, S_{\mathrm{zw}} = \{\mathrm{y}\}, S_{\mathrm{xt}} = \{\mathrm{z}, \mathrm{w}\}, S_{\mathrm{yt}} = \{\mathrm{z}, \mathrm{w}\}$

Learning structure of the Bayesian Networks

Determine the orientation



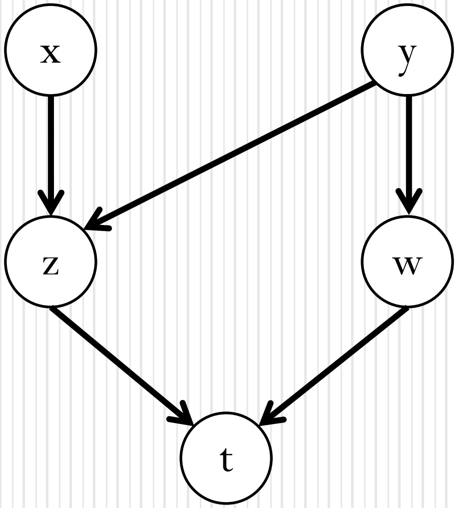
$t \notin S_{zw}$



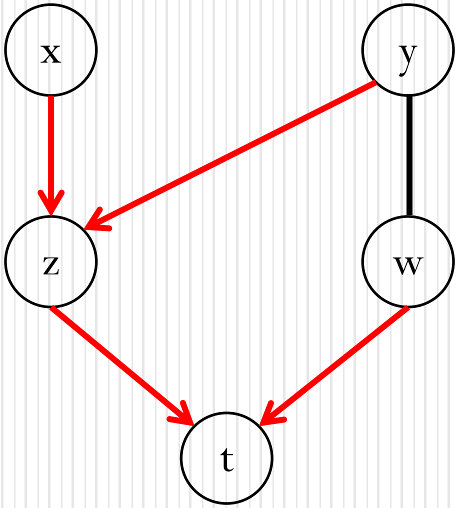
$S_{xy} = \{\ \}, S_{\mathrm{xw}} = \{\ \}, S_{\mathrm{zw}} = \{\mathrm{y}\}, S_{\mathrm{xt}} = \{\mathrm{z}, \mathrm{w}\}, S_{\mathrm{yt}} = \{\mathrm{z}, \mathrm{w}\}$

Learning structure of the Bayesian Networks

Determine the orientation



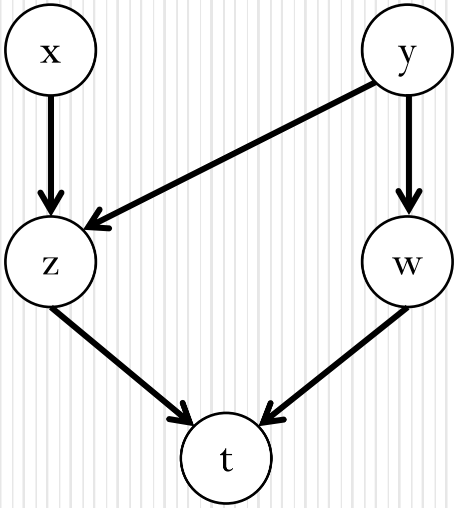
$t \notin S_{zw}$



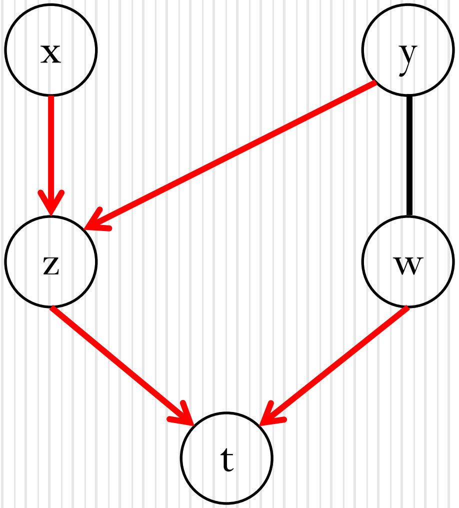
$S_{xy} = \{\ \}, S_{\mathrm{xw}} = \{\ \}, S_{\mathrm{zw}} = \{\mathrm{y}\}, S_{\mathrm{xt}} = \{\mathrm{z}, \mathrm{w}\}, S_{\mathrm{yt}} = \{\mathrm{z}, \mathrm{w}\}$

Learning structure of the Bayesian Networks

Determine the orientation

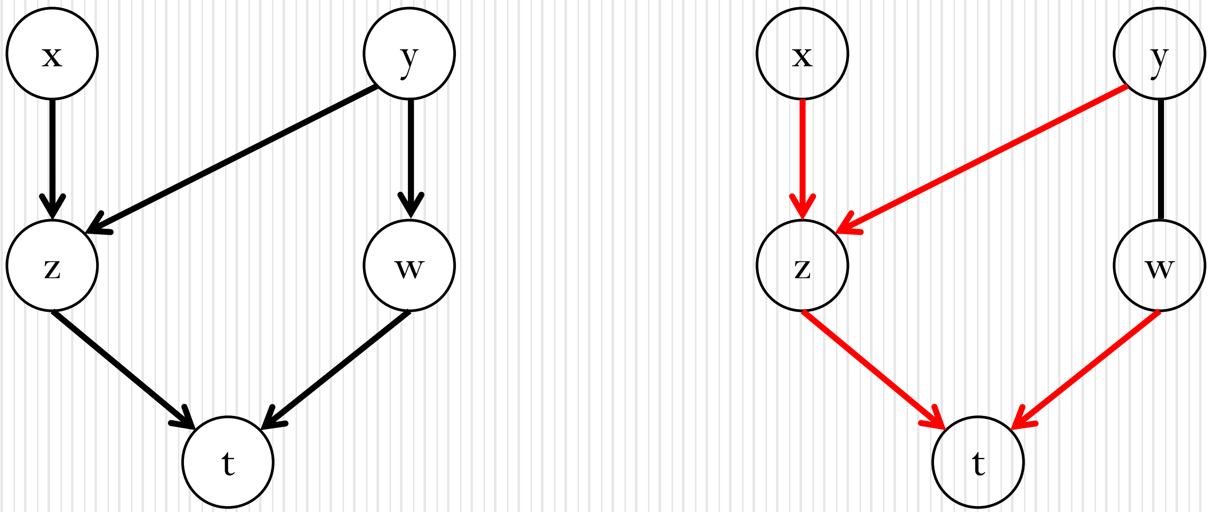


$t \notin S_{zw}$



$S_{xy} = \{\ \ \}, S_{xw} = \{\ \ \}, S_{zw} = \{y\}, S_{xt} = \{z, w\}, S_{yt} = \{z, w\}$

Learning structure of the Bayesian Networks

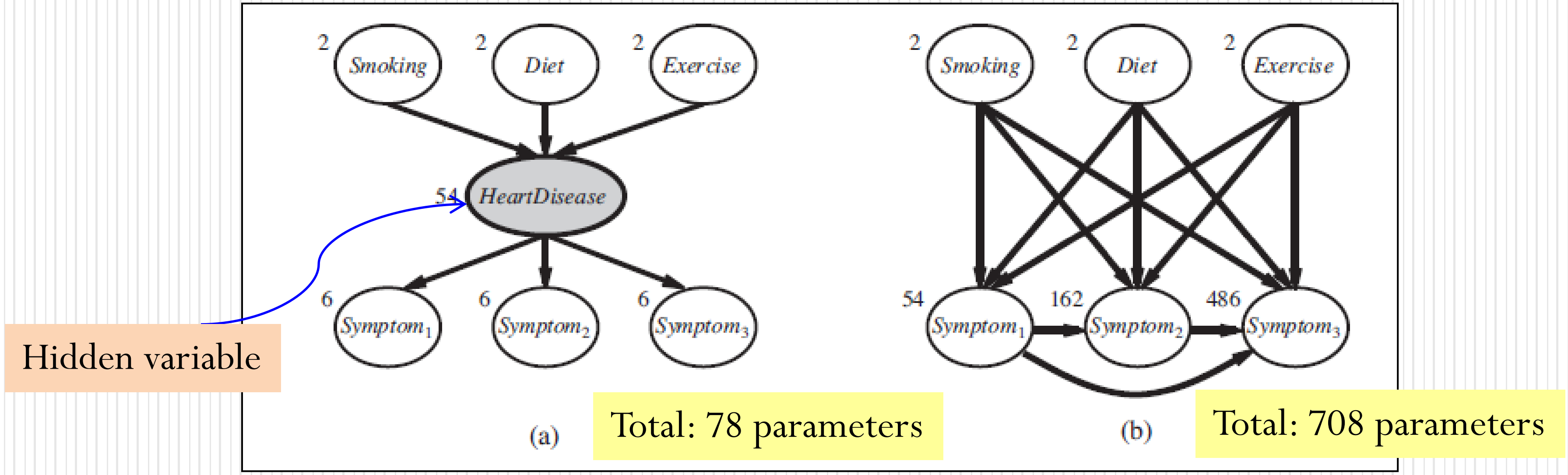


Partial DAG : the remaining edges can be arbitrarily oriented provided that the graph remains a DAG and no additional convergent edges are introduced.

$$\boldsymbol{S_{xy}} = \{\quad\}, \mathbf{S_{xw}} = \{\quad\}, \mathbf{S_{zw}} = \{\mathbf{y}\}, \mathbf{S_{xt}} = \{\mathbf{z}, \mathbf{w}\}, \mathbf{S_{yt}} = \{\mathbf{z}, \mathbf{w}\}$$

Learning Bayes Net with Hidden variables

- Hidden variables (or latent variables) – not observable in the data that are available for learning.
- Latent variables dramatically reduce the number of parameters in a Bayesian network and, in turn, reduces the amount of data needed to learn the parameters.

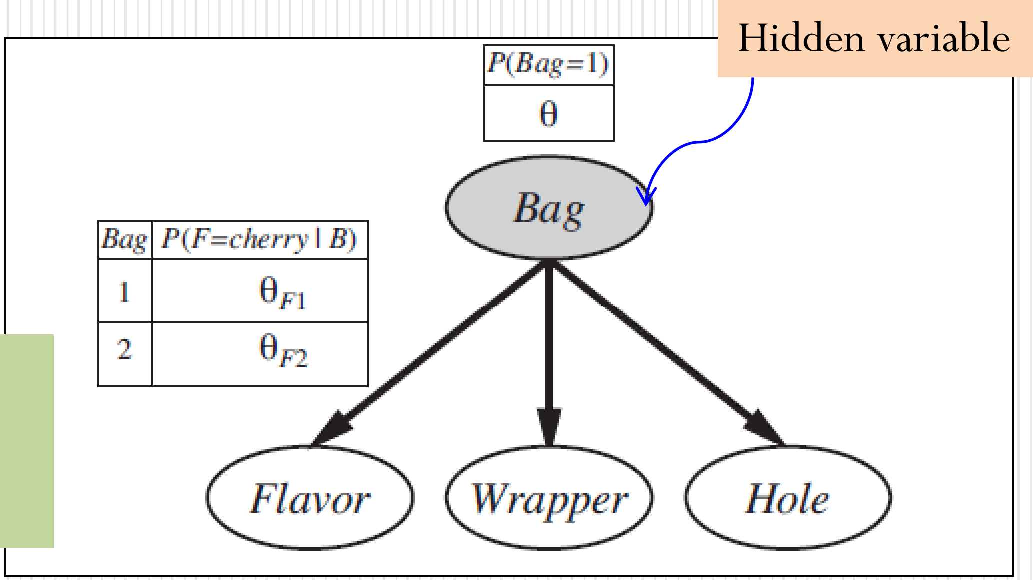


Each variable has three possible values and is labeled with the no. of independent parameters in its conditional distribution.

Learning Bayes Net with Hidden variables

- Example: Consider a situation where two bags of candies are mixed together.
- Candies are describe by three features:
  - *Flavour* (lime and cherry), *Wrapper* (red and green), *Hole* (candy with hole or without hole)
- Proportions of different flavours, wrappers, presence of holes depend on the bag, which is not observed.

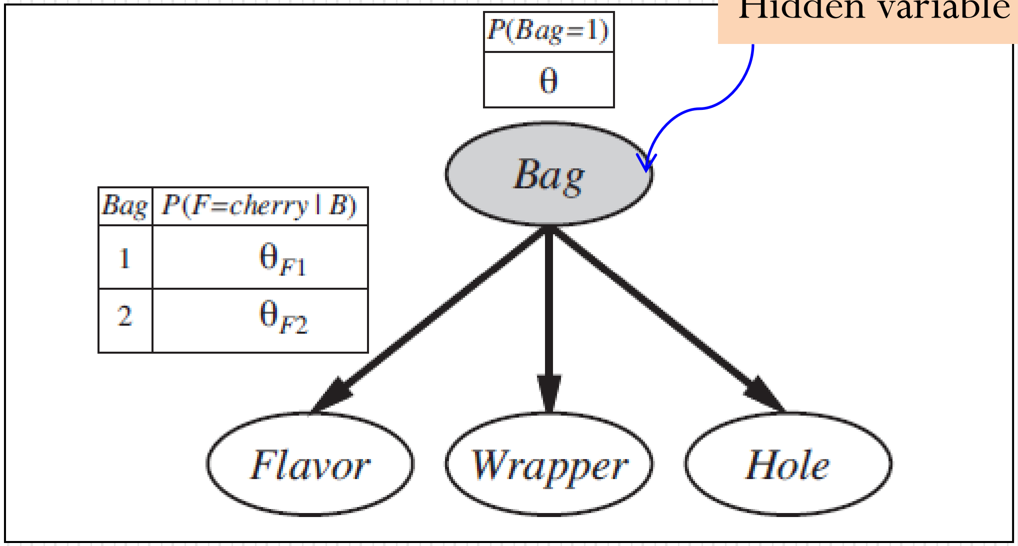
We want to predict for each candy, which was its original bag, from its features.



Learning Bayes Net with Hidden variables

- Parameters: θ is the prior probability that a candy comes from Bag 1 θ_{F1} and θ_{F2} are the probabilities that the flavor is cherry, given that the candy comes from Bag 1 or Bag 2 respectively; θ_{W1} and θ_{W2} give the probabilities that the wrapper is red; and θ_{H1} and θ_{H2} give the probabilities that the candy has a hole.
- the overall model can be considered as a mixture model

Hidden variable



- We will use Expectation –Maximization (EM) algorithm to learn the parameters of the model where Bag is hidden variable.

Learning Bayes Net with Hidden variables

- Expectation Maximization- applied to unsupervised clustering
- Clustering : revealing multiple categories in collection of objects where category labels are not given.

- In clustering, the data is assumed to be generated from a mixture distribution P with k components, each of which is a distribution in its own right.
- Let the random variable C denote the component, with values $1, \ldots, k$; then the mixture distribution is given by

$$\boxed{P(\mathbf{x}) = \sum_{i=1}^{k} P(C=i)\, P(\mathbf{x} \mid C=i)\ ,}$$

- For continuous data, the component distributions can be mixture of Gaussians that gives the mixture of Gaussians distributions.

Learning Bayes Net with Hidden variables

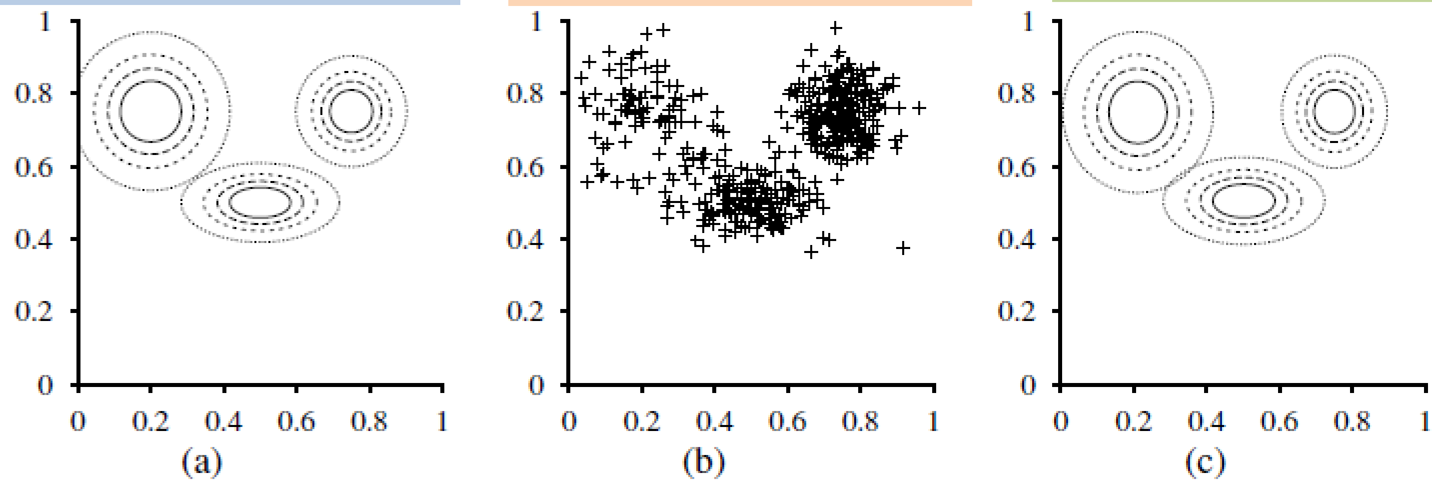
- Parameters of mixture of Gaussians:
  - $w_i = P(C = i)$, weight of each component
  - $\boldsymbol{\mu}_i$, mean of each component
  - $\boldsymbol{\Sigma}_i$, covariance of each component

unsupervised clustering problem, then, is to recover a mixture model like the one

Original distribution with 3 components

Raw data sampled from model in (a)

Model reconstructed by EM from data in (b)

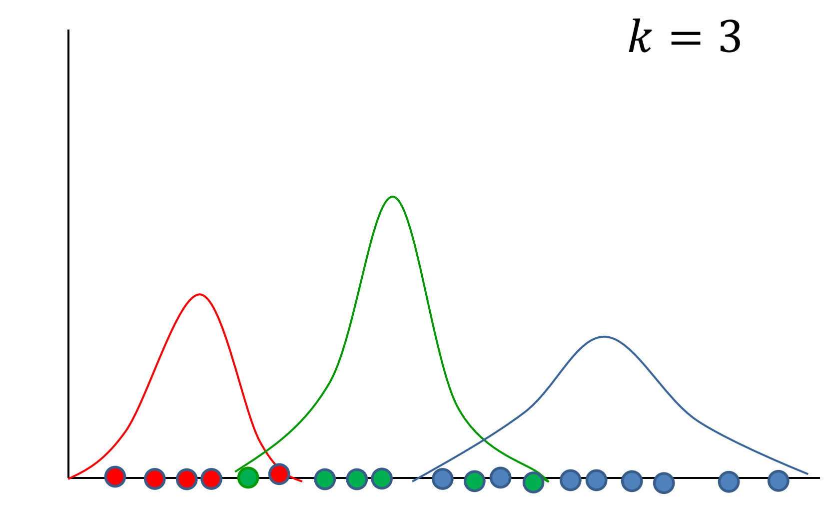


Learning Bayes Net with Hidden variables

- If it is known that which data sample came from which component distribution then we can estimate the parameters (i.e. the μ and σ of the Gaussians).
- If we know the parameters of the components (Gaussians) then we can probabilistically assign each data point to a component.

We know neither the assignments nor the parameters –Problem!

EM algorithm assumes that the parameters are known, it then infers the probability that each data point belongs to each component. After that, it again refits the component to entire data and this is iterated over until convergence.



Learning Bayes Net with Hidden variables

- EM algorithm
  - Initialize the model parameters (randomly) and iterate the following steps

1. **E-step:** Compute the probabilities $p_{ij} = P(C = i \mid \mathbf{x}_j)$, the probability that datum $\mathbf{x}_j$ was generated by component i. By Bayes' rule, we have $p_{ij} = \alpha P(\mathbf{x}_j \mid C = i) P(C = i)$. The term $P(\mathbf{x}_j \mid C = i)$ is just the probability at $\mathbf{x}_j$ of the ith Gaussian, and the term $P(C = i)$ is just the weight parameter for the ith Gaussian. Define $n_i = \sum_j p_{ij}$, the effective number of data points currently assigned to component i.

2. **M-step:** Compute the new mean, covariance, and component weights using the following steps in sequence:

$$\begin{aligned}
\boldsymbol{\mu}_i &\leftarrow \sum_j p_{ij} \mathbf{x}_j / n_i \\
\boldsymbol{\Sigma}_i &\leftarrow \sum_j p_{ij} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^\top / n_i \\
w_i &\leftarrow n_i / N
\end{aligned}$$

From Russell & Norvig Book

Learning Bayes Net with Hidden variables

- EM algorithm

where N is the total number of data points. The E-step, or *expectation* step, can be viewed as computing the expected values p_{ij} of the hidden **indicator variables** Z_{ij}, where Z_{ij} is 1 if datum $\mathbf{x}_j$ was generated by the ith component and 0 otherwise. The M-step, or *maximization* step, finds the new values of the parameters that maximize the log likelihood of the data, given the expected values of the hidden indicator variables.

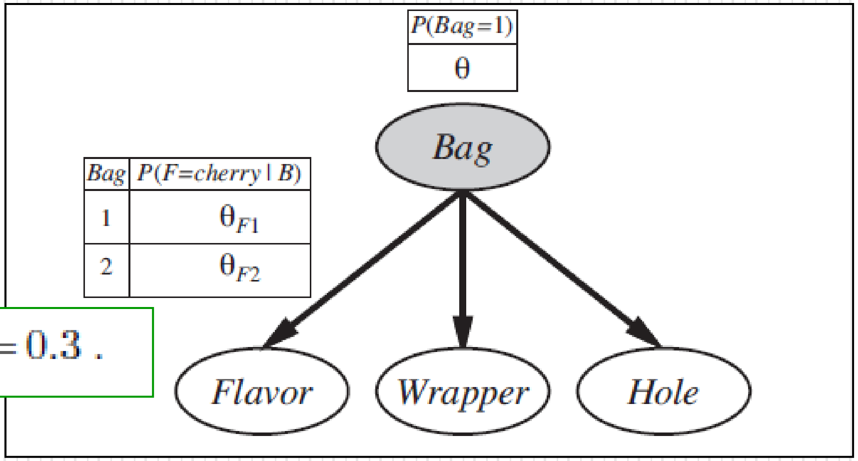
- EM increases the log-likelihood of data at every iteration (this can be proved in general).

Learning Bayes Net with Hidden variables

Coming back to initial problem !

- Parameters: θ is the prior probability that a candy comes from Bag 1; θ_{F1} and θ_{F2} are the probabilities that the flavor is cherry, given that the candy comes from Bag 1 or Bag 2 respectively; θ_{W1} and θ_{W2} give the probabilities that the wrapper is red; and θ_{H1} and θ_{H2} give the probabilities that the candy has a hole.

- 1000 samples are generated from the following (true) model:



$$\theta = 0.5, \quad \theta_{F1} = \theta_{W1} = \theta_{H1} = 0.8, \quad \theta_{F2} = \theta_{W2} = \theta_{H2} = 0.3 .$$

Learning Bayes Net with Hidden variables

	$W = red$		$W = green$	
	$H=1$	$H=0$	$H=1$	$H=0$
$F = cherry$	273	93	104	90
$F = lime$	79	100	94	167

- For EM algorithm initialize the parameters (arbitrarily):

$$\theta^{(0)} = 0.6, \quad \theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6, \quad \theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4 \;.$$

- <u>Learning of Parameter θ</u>
  - E-step:

    find the expected count of number of candies in Bag1 : Sum of the probabilities that each of the N data points comes from bag 1

$$\hat{N}(Bag = 1) = \sum_{j=1}^{N} P(Bag = 1 | flavor_j, wrapper_j, hole_j)$$

Learning Bayes Net with Hidden variables

$$\hat{N}(Bag=1) = \sum_{j=1}^{N} P(Bag=1|flavor_j, wrapper_j, whole_j)$$

$$= \sum_{j=1}^{N} \frac{P(flavor_j, wrapper_j, hole_j|Bag=1)P(Bag=1)}{P(flavor_j, wrapper_j, hole_j)}$$

$$= \sum_{j=1}^{N} \frac{P(flavor_j|Bag=1)P(wrapper_j|Bag=1)P(hole_j|Bag=1)P(Bag=1)}{\sum_i P(flavor_j|Bag=i)P(wrapper_j|Bag=i)P(hole_j|Bag=i)P(Bag=i)}$$

	$W=red$		$W=green$	
	$H=1$	$H=0$	$H=1$	$H=0$
$F=cherry$	$\boxed{273}$	93	104	90
$F=lime$	79	100	94	167

This summation can be broken down for 8 candy groups

For example, sum over 273 cheery candies with red wrapper and holes is:

$$= 273 \frac{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)}}{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)} + \theta_{F2}^{(0)}\theta_{W2}^{(0)}\theta_{H2}^{(0)}(1-\theta^{(0)})} =$$

$$273 \frac{0.6^4}{0.6^4 + 0.4^4} = 273 \frac{0.1296}{0.1552} = 227.97$$

Learning Bayes Net with Hidden variables

- summing over other 7 groups in the table

$$\hat{N}(Bag = 1) = 612.4$$

- M-step
  - Compute new θ using the expected count of candies that came from Bag 1.

$$\boxed{\theta(1) = \frac{\hat{N}(Bag = 1)}{N} = 0.6124}$$

- Similarly, we can learn other 5 parameters.
- Learning parameter θ_{F1}

E-step: compute the expected count of cherry candies from Bag 1

$$\hat{N}(Bag = 1 \wedge Flavor = cherry) = \sum_{j:Flavor_j = cherry} P(Bag = 1 \mid Flavor_j = cherry, wrapper_j, hole_j)$$

Learning Bayes Net with Hidden variables

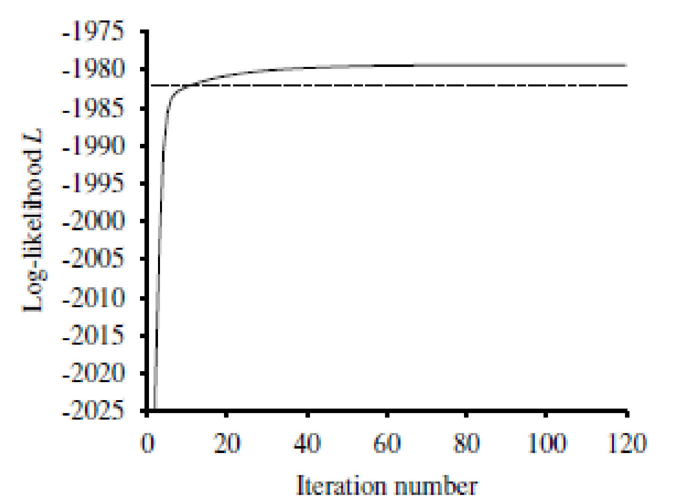
M-step: refine θ_{F1} by computing the corresponding expected frequencies

$$\theta_{F1}^{(1)} = \frac{\hat{N}(Bag = 1 \wedge Flavor = cherry)}{\hat{N}(Bag = 1)}$$

After a complete cycle through all the parameters, we get

$$\theta^{(1)} = 0.6124;$$
$$\theta_{F1}^{(1)} = 0.6684; \quad \theta_{W1}^{(1)} = 0.6483; \quad \theta_{H1}^{(1)} = 0.658;$$
$$\theta_{F2}^{(1)} = 0.3887; \quad \theta_{W2}^{(1)} = 0.3817; \quad \theta_{H2}^{(1)} = 0.3827;$$

- EM increases the log likelihood of the data at every iteration



What did we discussed in L19-21?

- How to learn parameters (CPTs) of a Bayesian Network when complete observations are given?
  - Maximum-likelihood approach
  - Bayesian approach

- How to learn the structure of Bayesian network?

- Some non-parametric approaches of Density estimation
  - Histograms
  - KDE
  - k-NN

- How to learn the parameters of Bayesian Network with Hidden variables?
  - Expectation Maximization algorithm