# **Bayesuvius**,
a small visual dictionary of Bayesian Networks

Robert R. Tucci
www.ar-tiste.xyz

November 4, 2020

Figure 1: View of Mount Vesuvius from Pompeii

Figure 2: Mount Vesuvius and Bay of Naples

# Contents

# Chapter 14

# Expectation Maximization

This chapter is based on Refs.[39] and [62].

The Expectation Maximization (EM) algorithm is commonly used in Data Science to find the maximum over an **unknown parameter** $\theta$ of a likelihood function

$$P(\vec{x}|\theta) = \sum_{\vec{h}} P(\vec{x}, \vec{h}|\theta) \ , \tag{14.1}$$

where $\vec{x}$ denotes the **observed variables**, and $\vec{h}$ denotes the **latent variables**. Both $\theta$ and $\vec{h}$ are hidden (i.e., unobserved).[1]

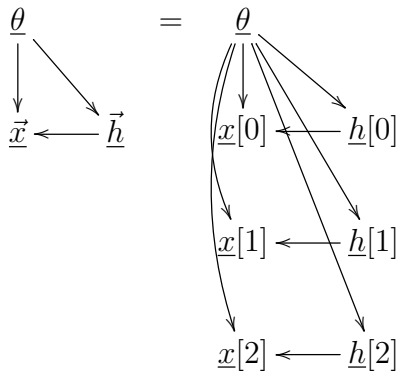


Figure 14.1: bnet for EM with $nsam = 3$.

The bnet for the EM algorithm is given by Fig.14.1 for $nsam = 3$. Later on in this chapter, we will give the node TPMs for this bnet for the special case in which $P(x[\sigma] \mid \theta)$ is a mixture (i.e., weighted sum) of Gaussians.

[1] The term "unknown parameter" is mainly of frequentist origin. For Bayesians, $\theta$ is a random variable with a delta function prior, whereas for frequentists, it is not a random variable at all, just an unknown parameter with no randomness.

Note that if we erase the $\underline{h}[\sigma]$ nodes from Fig.14.1, we get the bnet for naive Bayes, which is used for classification into the states of $\underline{\theta}$. However, there is one big difference. With naive Bayes, the leaf nodes have different TPMs. Here, we will assume they are i.i.d. Naive Bayes is used for classification: i.e., given the states of the leaf nodes, we infer the state of the root node. EM is used for clustering; i.e., given many i.i.d. samples, we fit their distribution by a weighted sum of prob distributions, usually Gaussians.

Let

$\mathcal{L}$ =likelihood function.

$nsam$ = number of samples.

$\vec{x} = (x[0], x[1], \ldots, x[nsam-1])$ $x[\sigma] \in S_{\underline{x}}$ for all $\sigma$.

$\vec{h} = (h[0], h[1], \ldots, h[nsam-1])$ $h[\sigma] \in S_{\underline{h}}$ for all $\sigma$.

We assume that the samples $(x[\sigma], h[\sigma])$ are i.i.d. for different $\sigma$ at fixed $\theta$. What this means is that there are probability distributions $P_{\underline{x}|\underline{h},\underline{\theta}}$ and $P_{\underline{h}|\underline{\theta}}$ such that

$$P(\vec{x}, \vec{h}|\theta) = \prod_{\sigma} \left[ P_{\underline{x}|\underline{h},\underline{\theta}}(x[\sigma] \mid h[\sigma], \theta) P_{\underline{h}|\underline{\theta}}(h[\sigma] \mid \theta) \right] \ . \tag{14.2}$$

Definition of likelihood functions:

$$\underbrace{P(\vec{x}|\theta)}_{\mathcal{L}(\theta;\vec{x})} \ = \ \sum_{\vec{h}} \underbrace{P(\vec{x}, \vec{h}|\theta)}_{\mathcal{L}(\theta;\vec{x},\vec{h})} \tag{14.3}$$

$\theta^*$ = maximum likelihood estimate of $\theta$ (no prior $P(\theta)$ assumed):

$$\theta^* = \underset{\theta}{\text{argmax}}\, \mathcal{L}(\theta; \vec{x}) \tag{14.4}$$

---

**The EM algorithm:**

1. **Expectation step:**[2]

$$Q(\theta|\theta^{(t)}) = E_{\vec{h}|\vec{x},\theta^{(t)}} \ln P(\vec{x}, \vec{h}|\theta) \tag{14.5}$$

2. **Maximization step:**

$$\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\, Q(\theta|\theta^{(t)}) \ . \tag{14.6}$$

Claim: $\lim_{t\to\infty} \theta^{(t)} = \theta^*$.

Fig.14.2 portrays the recursive nature of the EM algo as a dynamical, recurrent bnet. For Fig.14.2, the TPMs, printed in blue, for the $\underline{\theta}^{(t)}$ nodes for $t = 1, 2, \ldots$, are as follows:

$$P(\theta^{(t+1)}|\vec{x}, \theta^{(t)}) = \delta(\theta^{(t+1)}, \underset{\theta}{\text{argmax}}\, Q(\theta|\theta^{(t)})) \ . \tag{14.7}$$

---

[2] Note that that the right hand side of Eq.(14.5) is expressible in the form $\sum_\sigma \sum_{h[\sigma]} f(x[\sigma], h[\sigma])$.

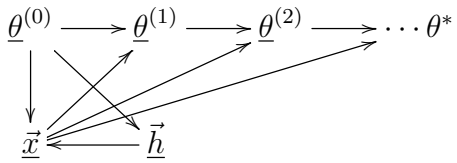


Figure 14.2: The EM algo generates a sequence of parameter estimates $(\theta^{(t)})_{t=0,1,2,...}$ that converges to the optimum (i.e., best-fit) parameter $\theta^*$.

# Motivation

$$
\begin{aligned}
Q(\theta|\theta^{(t)}) &= E_{\vec{h}|\vec{x},\theta^{(t)}} \ln P(\vec{x},\vec{h}|\theta) && (14.8)\\
&= E_{\vec{h}|\vec{x},\theta^{(t)}} [\ln P(\vec{h}|\vec{x},\theta) + \ln P(\vec{x}|\theta)] && (14.9)\\
&= -D_{KL}\left(P(\vec{h}|\vec{x},\theta^{(t)}) \parallel P(\vec{h}|\vec{x},\theta)\right) - H[P(\underline{\vec{h}}|\vec{x},\theta^{(t)})] + \ln P(\vec{x}|\theta) && (14.10)
\end{aligned}
$$

When $\theta^{(t)} = \theta$, this becomes

$$
Q(\theta|\theta) = -H[P(\underline{\vec{h}}|\vec{x},\theta)] + \ln P(\vec{x}|\theta) \; . \tag{14.11}
$$

Hence,

$$
\begin{aligned}
\partial_\theta Q(\theta|\theta) &= -\sum_{\vec{h}} \partial_\theta P(\underline{\vec{h}}|\vec{x},\theta) + \partial_\theta \ln P(\vec{x}|\theta) && (14.12)\\
&= \partial_\theta \ln P(\vec{x}|\theta) && (14.13)
\end{aligned}
$$

So if $\theta^{(t)} \to \theta$ and $Q(\theta|\theta)$ is max at $\theta = \theta^*$, then $\ln P(\vec{x}|\theta)$ is max at $\theta = \theta^*$ too.

For a more rigorous proof that $\lim_{t\to\infty} \theta^{(t)} = \theta^*$, see Wikipedia article Ref.[39] and references therein.

# Minorize-Maximize (MM) algorithms

A function $\mu(\theta|\theta^{(t)})$ **is said to minorize a target function** $\mathcal{L}(\theta)$ iff for all $\theta$ at fixed $\theta^{(t)}$, it satisfies the "$\mu \leq \mathcal{L}$ property"

$$
\mu(\theta|\theta^{(t)}) \leq \mathcal{L}(\theta) \; , \tag{14.14}
$$

and the "$\mu = \mathcal{L}$ property"

$$
\mu(\theta^{(t)}|\theta^{(t)}) = \mathcal{L}(\theta^{(t)}) \; . \tag{14.15}
$$

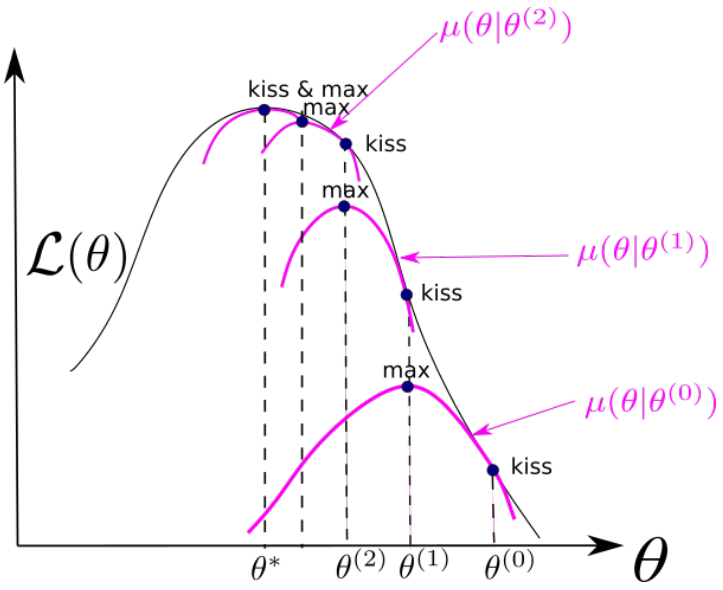


Figure 14.3: Function $\mu(\theta|\theta^{(t)})$ minorizes the function $\mathcal{L}(\theta)$. Note that $\mu(\theta|\theta^{(t)})$ is always below $\mathcal{L}(\theta)$. "max" indicates $\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\, \mu(\theta|\theta^{(t)})$. "kiss" indicates $\mu(\theta^{(t)}|\theta^{(t)}) = \mathcal{L}(\theta^{(t)})$.

We **recursively maximize a minorizing function** $\mu(\theta|\theta^{(t)})$ if we define a sequence $(\theta^{(t)})_{t=0,1,...}$ as follows:

$$\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\, \mu(\theta|\theta^{(t)}) \; . \tag{14.16}$$

The sequence $(\mathcal{L}(\theta^{(t)}))_{t=0,1,2,...}$ generated by recursively maximizing a minorizing function must be nondecreasing:

$$\mathcal{L}(\theta^{(t+1)}) \geq \mu(\theta^{(t+1)}|\theta^{(t)}) \geq \mu(\theta^{(t)}|\theta^{(t)}) = \mathcal{L}(\theta^{(t)}) \; . \tag{14.17}$$

A **minorize-maximize (MM) algorithm** is any algo that specifies a minorizing function $\mu(\theta|\theta^{(t)})$ for a particular target function $\mathcal{L}(\theta)$. One can also define a **majorize-minimize algo (also called MM)** by inverting the inequalities throughout.

The EM algo is an MM algo. Indeed, if we define

$$\mathcal{L}(\theta) = \ln P(\vec{x}|\theta) \tag{14.18}$$

and

$$\mu(\theta|\theta^{(t)}) = Q(\theta|\theta^{(t)}) + H(P(\vec{h}|\vec{x}, \theta^{(t)}) \; , \tag{14.19}$$

then Eq.(14.10) establishes the $\mu \leq \mathcal{L}$ and $\mu = \mathcal{L}$ properties required of a minorizing function.

How an MM algo works is portrayed in Fig.14.3.

# Examples

**Example (Gaussian mixture)**

$x[\sigma] \in \mathbb{R}^d = S_{\underline{x}}$. $S_{\underline{h}}$ discrete and not too large. $n_{\underline{h}} = |S_{\underline{h}}|$ is number of Gaussians that we are going to fit the samples with.

Let

$$\theta = [w_h, \mu_h, \Sigma_h]_{h \in S_{\underline{h}}} , \tag{14.20}$$

where $[w_h]_{h \in S_{\underline{h}}}$ is a probability distribution of weights, and where $\mu_h \in \mathbb{R}^d$ and $\Sigma_h \in \mathbb{R}^{d \times d}$ are the mean value vector and covariance matrix of a $d$-dimensional Gaussian distribution.

The TPMs, printed in blue, for the nodes of Fig.14.1, for the special case of a mixture of Gaussians, are as follows:

$$P(x[\sigma] \mid h[\sigma] \mid \theta) = \mathcal{N}_d(x[\sigma]; \mu_{h[\sigma]}, \Sigma_{h[\sigma]}) \tag{14.21}$$

$$P(h[\sigma] \mid \theta) = w_{h[\sigma]} \tag{14.22}$$

Note that

$$P(x[\sigma] \mid \theta) = \sum_h P(x[\sigma] \mid h[\sigma] = h, \theta) P(h[\sigma] = h \mid \theta) \tag{14.23}$$

$$= \sum_h w_h \mathcal{N}_d(x[\sigma]; \mu_h, \Sigma_h) \tag{14.24}$$

$$P(\vec{x}, \vec{h} | \theta) = \prod_\sigma \left[ w_{h[\sigma]} \mathcal{N}_d(x[\sigma]; \mu_{h[\sigma]}, \Sigma_{h[\sigma]}) \right] \tag{14.25}$$

$$= \prod_\sigma \prod_h \left[ w_h \mathcal{N}_d(x[\sigma]; \mu_h, \Sigma_h) \right]^{\mathbb{1}(h = h[\sigma])} \tag{14.26}$$

**Old Faithful:** See Wikipedia Ref.[39] for an animated gif of a classic example of using EM to fit samples with a Gaussian mixture. Unfortunately, could not include it here because pdflatex does not support animated gifs. The gif shows samples in a 2 dimensional space (eruption time, delay time) from the Old Faithful geyser. In that example, $d = 2$ and $n_{\underline{h}} = 2$. Two clusters of points in a plane are fitted by a mixture of 2 Gaussians.

**K-means clustering** is often presented as the main competitor to EM for doing **clustering (non-supervised learning)**. In K-means clustering, the sample points are split into $K$ mutually disjoint sets $S_0, S_1, \ldots, S_{K-1}$. The algorithm is easy to describe:

1. Initialize by choosing at random $K$ data points $(\mu_k)_{k=0}^{K-1}$ called means or centroids and placing $\mu_k$ in $S_k$ for all $k$.

2. **STEP 1:** For each data point, add it to the $S_k$ whose centroid $\mu_k$ is closest to it.

3. **STEP 2:** Recalculate the centroids. Set $\mu_k$ equal to the mean value of set $S_k$.

4. Repeat steps 1 and 2 until the centroids stop changing by much.

Step 1 is analogous to the expectation step in EM, and Step 2 to the maximization step in EM ($\theta$ estimation versus $\mu_k$ estimation). We won't say anything further about K-means clustering because it isn't related to bnets in any way, and this is a book about bnets. For more info about K-means clustering, see Ref.[47].

---

**Example (Blood Genotypes and Phenotypes):**

Notation: $\vec{\underline{a}} = (\underline{a}[\sigma])_{\sigma=0,1,\ldots,nsam-1}$, where $nsam$ is the number of samples. Will sometimes denote $a[\sigma]$ by $a^{[\sigma]}$.

Suppose $\vec{\underline{x}} = (\vec{\underline{x}}_0)$ (i.e., just one component)

$\vec{\underline{h}} = (\vec{\underline{h}}_0)$ (i.e., just one component)

$\underline{h}[\sigma] \in S_{\underline{h}} = \{AA, AO, BB, BO, OO, AB\}$ (the 6 blood genotypes)

$\underline{x}[\sigma] \in S_{\underline{x}} = \{A, B, O, AB\}$ (the 4 blood phenotypes)

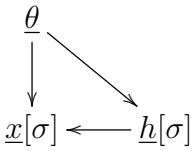


Figure 14.4: bnet for blood phenotypes $x[\sigma]$ and genotypes $h[\sigma]$.

For the bnet of Fig.14.4, the TPMs, printed in blue, are:

$$P(h^{[\sigma]}|\theta) = \begin{array}{c|c} & \\ \hline AA & p_A^2 \\ AO & 2p_Ap_O \\ BB & p_B^2 \\ BO & 2p_Bp_O \\ OO & p_O^2 \\ AB & 2p_Ap_B \end{array} , \tag{14.27}$$

where $p_A + p_B + p_O = 1$.

$$P(x^{[\sigma]} \mid h^{[\sigma]}, \theta) = \begin{array}{c|cccccc} & AA & AO & BB & BO & OO & AB \\ \hline A & 1 & 1 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 1 & 0 & 0 \\ O & 0 & 0 & 0 & 0 & 1 & 0 \\ AB & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \tag{14.28}$$

$$\theta = (p_A, p_B) \tag{14.29}$$

Multiplying the TPMs in Eqs.(14.27 and (14.28), we get

$$P(x^{[\sigma]} \mid \theta) = \begin{array}{c|l} & \\ \hline A & p_A^2 + 2p_A p_O (= \pi_A) \\ B & p_B^2 + 2p_B p_O (= \pi_B) \\ O & p_O^2 (= \pi_O) \\ AB & 2p_A p_B (= \pi_{AB}) \end{array} \tag{14.30}$$

Note that

$$\begin{aligned} P(\vec{x}|\theta) &= \prod_\sigma P(x^{[\sigma]}|\theta) && (14.31) \\ &= (\pi_A)^{N_A}(\pi_B)^{N_B}(\pi_O)^{N_O}(\pi_{AB})^{N_{AB}} , && (14.32) \end{aligned}$$

where $N_x$ for $x \in S_{\underline{x}} = \{A, B, O, AB\}$ are the counts from the data. We can get estimates for the parameters $p_A$ and $p_B$ right here without doing EM. Just note that

$$\hat{\pi}_x = \frac{N_x}{N_+} \tag{14.33}$$

for $x \in S_{\underline{x}}$, where $N_+ = \sum_x N_x$. Eqs.(14.33) give 4 quadratic equations that can be solved for the parameters $p_A, p_B$ in terms of the observed counts $N_x$ for $x \in S_{\underline{x}}$.

If, instead, you want to find the optimum parameters $p_A, p_B$ using EM, note that

$$\begin{aligned} Q(\theta|\theta^{(t)}) &= \sum_{\vec{h}} P(\vec{h}|\theta^{(t)}) \ln P(\vec{x}, \vec{h}|\theta) && (14.34) \\ &= \sum_{\vec{h}} \left[ \prod_\sigma P(h^{[\sigma]}|\theta^{(t)}) \right] \ln \left[ \prod_\sigma P(x^{[\sigma]}, h^{[\sigma]}|\theta) \right] && (14.35) \\ &= \sum_\sigma \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) \ln P(x^{[\sigma]}, h^{[\sigma]}|\theta) && (14.36) \\ &= \sum_\sigma \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) [\ln P(x^{[\sigma]}|h^{[\sigma]}, \theta) + \ln P(h^{[\sigma]}|\theta)] && (14.37) \\ &= nsam \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) \ln P(h^{[\sigma]}|\theta) . && (14.38) \end{aligned}$$

---

**Example (Missing Data/Imputation):**

The previous example on blood genotypes and phenotypes assumed no missing data in compiling the counts $N_x$. But what if there is missing data? Can one still apply the EM algo in that case? Yes! See Chapter 28.

# Chapter 25

# Markov Chain Monte Carlo (MCMC)

Monte Carlo methods are methods for using random number generation to sample probability distributions. The subject of Monte Carlo methods has many branches, as you can see from its Wikipedia category list, Ref.[53]. MCMC (Markov Chain Monte Carlo) is just one of those branches, albeit a major one. Metropolis-Hastings (MH) sampling is a very important MCMC method. Gibbs sampling is a special case of MH sampling. This chapter covers both, MH and Gibbs sampling. It also covers a few other types of sampling.

Throughout this chapter, we use $P_{\underline{x}} : S_{\underline{x}} \to [0, 1]$ to denote the target probability distribution that we wish to obtain samples from.

## Inverse Cumulative Sampling

For more info about this topic and some original references, see Ref.[45].

This is one of the simplest methods for obtaining samples from a probability distribution $P_{\underline{x}}$, but it requires knowledge of the inverse cumulative distribution of $P_{\underline{x}}$, which is often not available.

The **cumulative distribution** function is defined by:

$$CUM_{\underline{x}}(x) = P(\underline{x} < x) = \int_{x' < x} dx' \; P_{\underline{x}}(x') \; . \tag{25.1}$$

Note that

$$P_{\underline{x}}(x) = \frac{d}{dx} CUM_{\underline{x}}(x) \; . \tag{25.2}$$

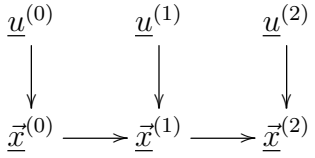


Figure 25.1: bnet for Inverse Cumulative Sampling

For $t = 0, 1, \ldots, T-1$, let

$\underline{u}^{(t)} \in [0,1]$= random variable, uniformly distributed over $[0,1]$.

$\vec{\underline{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\ldots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all $\sigma$. Vector of samples collected up to time $t$.

The TPMs, printed in blue, for the nodes of bnet Fig.25.1, are:

$$P(u^{(t)}) = 1 \tag{25.3}$$

$$P(\vec{x}^{(t)} | \vec{x}^{(t-1)}, u^{(t)}) = \delta(\quad \vec{x}^{(t)}, [\vec{x}^{(t-1)}, CUM_{\underline{x}}^{-1}(u^{(t)})]\quad) \tag{25.4}$$

**Motivation**

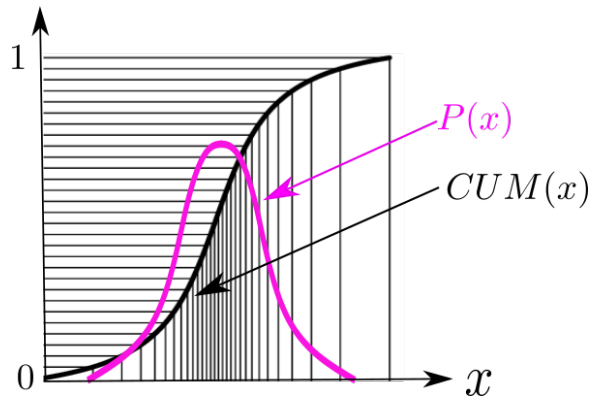


Figure 25.2: Motivation for Inverse Cumulative Sampling.

See Fig.25.2.

Note that if $\underline{u}$ is uniformly distributed over the interval $[0,1]$ and $a \in [0,1]$, then

$$P(\underline{u} < a) = a\ . \tag{25.5}$$

Thus

$$\begin{aligned} P(CUM_{\underline{x}}^{-1}(\underline{u}) < x) &= P(\underline{u} < CUM_{\underline{x}}(x)) & (25.6)\\ &= CUM_{\underline{x}}(x)\ . & (25.7) \end{aligned}$$

Therefore,

$$dP(CUM_{\underline{x}}^{-1}(\underline{u}) < x) = P_{\underline{x}}(x)dx\ . \tag{25.8}$$

# Rejection Sampling

For more info about this topic and some original references, see Ref.[59].

This method samples from a "candidates" probability distribution $P_{\underline{c}} : S_{\underline{x}} \to [0,1]$, in cases where sampling directly from the target probability distribution $P_{\underline{x}} : S_{\underline{x}} \to [0,1]$ is not possible.

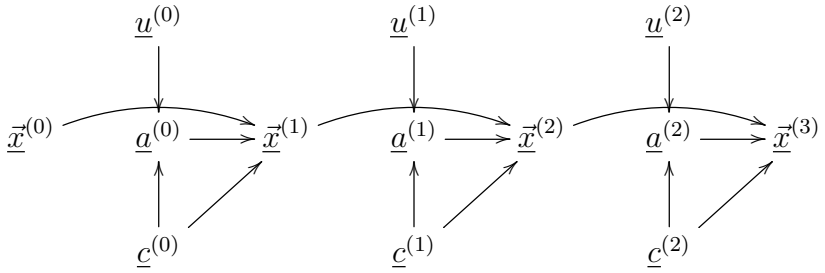


Figure 25.3: bnet for Rejection Sampling

For $t = 0, 1, \ldots, T-1$, let

$\underline{u}^{(t)} \in [0,1]$= random variable, uniformly distributed over $[0,1]$.

$\underline{a}^{(t)} \in \{0,1\}$= accept candidate? (no=0, yes=1)

$\underline{c}^{(t)} \in S_{\underline{x}}$= sample that is a candidate for being accepted

$\vec{\underline{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\ldots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all $\sigma$. Vector of samples collected up to time $t$.

This algorithm requires a priori definition of a candidate probability distribution $P_{\underline{c}} : S_{\underline{x}} \to \mathbb{R}$ such that

$$P_{\underline{x}}(x) < \beta P_{\underline{c}}(x) \tag{25.9}$$

for all $x \in S_{\underline{x}}$, for some $\beta \in \mathbb{R}$.

The TPMs, printed in blue, for the nodes of bnet Fig.25.3, are:

$$P(u^{(t)} = u) = 1 \tag{25.10}$$

$$P(\underline{c}^{(t)} = c) = P_{\underline{c}}(c) \tag{25.11}$$

$$P(\underline{a}^{(t)} = a | \underline{c}^{(t)} = c, \underline{u}^{(t)} = u) = \begin{cases} \delta(a,0) & \text{if } u\beta P_{\underline{c}}(c) \geq P_{\underline{x}}(c) \\ \delta(a,1) & \text{if } u\beta P_{\underline{c}}(c) < P_{\underline{x}}(c) \end{cases} \tag{25.12}$$

$$P(\vec{x}^{(t)} | \vec{x}^{(t-1)}, \underline{a}^{(t)} = a, \underline{c}^{(t)} = c) = \begin{cases} \delta(\vec{x}^{(t)}, \vec{x}^{(t-1)}) & \text{if } a = 0 \\ \delta(\vec{x}^{(t)}, [\vec{x}^{(t-1)}, c]) & \text{if } a = 1 \end{cases} \tag{25.13}$$

This last equation is only defined for $t > 0$. For $t = 0$, the left hand side reduces to $P(\vec{x}^{(0)})$ which must be specified a priori.

**Motivation**

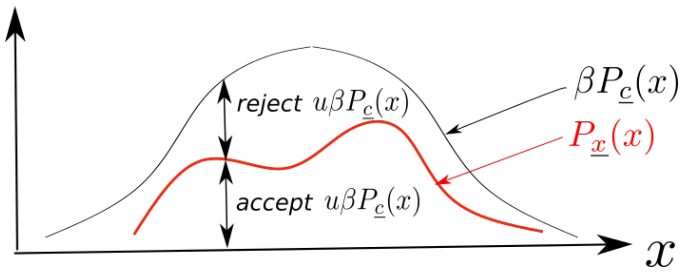


Figure 25.4: Motivation for Rejection Sampling.

See Fig.25.4.

# Metropolis-Hastings Sampling

For more info about this topic and some original references, see Refs.[1] and [51].

An advantage of this method is that it can sample unnormalized probability distributions $(constant)P_{\underline{x}}$ because it only uses ratios of $P_{\underline{x}}$ at two different points. Another advantage of this method is that it scales much better than other sampling methods as the number of dimensions of the sampled variable $\underline{x}$ increases.

This method produces samples that take a finite amount of time to reach steady state. The samples are also theoretically correlated instead of being i.i.d. as one desires. To mitigate for the steady state problem, one discards an initial set of samples (the "burn-in" period). To mitigate for the correlation problem, one calculates the autocorrelation between the samples and keeps only samples separated by a time interval after which the samples cease to be autocorrelated to a good approximation.

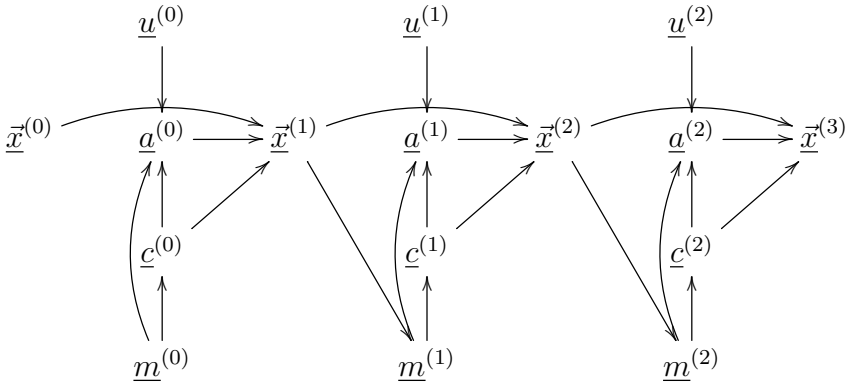


Figure 25.5: bnet for Metropolis-Hastings Sampling

For $t = 0, 1, \ldots, T-1$, let
$\underline{u}^{(t)} \in [0,1]$= random variable, uniformly distributed over $[0,1]$.
$\underline{a}^{(t)} \in \{0,1\}$= accept candidate? (no=0, yes=1)
$\underline{c}^{(t)} \in S_{\underline{x}}$= sample that is a candidate for being accepted
$\underline{m}^{(t)} \in S_{\underline{x}}$= memory of last accepted sample
$\underline{\vec{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\ldots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all $\sigma$. Vector of samples collected up to time $t$.

A **proposal TPM** $P_{\underline{c}|\underline{x}} : S_{\underline{x}}^2 \to [0,1]$ must be specified a priori for this algorithm.
The TPMs, printed in blue, for the nodes of bnet Fig.25.5, are:

$$P(u^{(t)} = u) = 1 \tag{25.14}$$

$$P(\underline{c}^{(t)} = c | \underline{m}^{(t)} = m) = P_{\underline{c}|\underline{x}}(c|m) \tag{25.15}$$

$$P(\underline{a}^{(t)} = a | \underline{c}^{(t)} = c, \underline{u}^{(t)} = u, \underline{m}^{(t)} = m) = \begin{cases} \delta(a,0) & \text{if } u \geq \alpha(c|m) \\ \delta(a,1) & \text{if } u < \alpha(c|m) \end{cases} \tag{25.16}$$

where the **acceptance probability** $\alpha$ is defined as

$$\alpha(c|m) = \min\left(1, \frac{P_{\underline{c}|\underline{x}}(m|c) P_{\underline{x}}(c)}{P_{\underline{c}|\underline{x}}(c|m) P_{\underline{x}}(m)}\right) . \tag{25.17}$$

Note that if the proposal distribution is symmetric, then

$$\alpha(c|m) = \min\left(1, \frac{P_{\underline{x}}(c)}{P_{\underline{x}}(m)}\right) . \tag{25.18}$$

$$P(\vec{x}^{(t)} | \vec{x}^{(t-1)}, \underline{a}^{(t)} = a, \underline{c}^{(t)} = c) = \begin{cases} \delta(\vec{x}^{(t)}, \vec{x}^{(t-1)}) & \text{if } a = 0 \\ \delta(\vec{x}^{(t)}, [\vec{x}^{(t-1)}, c]) & \text{if } a = 1 \end{cases} \tag{25.19}$$

This last equation is only defined for $t > 0$. For $t = 0$, the left hand side reduces to $P(\vec{x}^{(0)})$ which must be specified a priori.

$$P(\underline{m}^{(t)} = m | \vec{x}^{(t)}) = \delta(m, \text{last component of } \vec{x}^{(t)}) . \tag{25.20}$$

This last equation is only defined for $t > 0$. For $t = 0$, the left hand side reduces to $P(\underline{m}^{(0)} = m)$ which must be specified a priori.

**Motivation**

See Fig.25.6.

Consider a time homogeneous (its TPM is the same for all times) Markov chain with TPM $P(x'|x) = [T]_{x',x}$. Its **stationary distribution**, if it exists, is defined as

$$\pi = \lim_{n \to \infty} T^n \pi_0 . \tag{25.21}$$

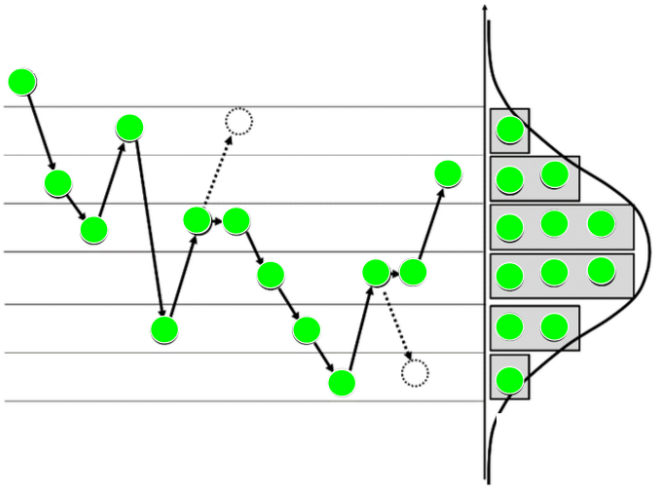

Figure 25.6: Motivation for Metropolis-Hastings Sampling.

Suppose the prob distribution $P_{\underline{x}}(x)$ that we wish to sample from satisfies

$$P_{\underline{x}}(x) = \pi(x) \ . \tag{25.22}$$

**Reversibility (detailed balance):** For all $x, x' \in S_{\underline{x}}$,

$$P(x'|x)\pi(x) = P(x|x')\pi(x') \ . \tag{25.23}$$

Detailed balance is a sufficient (although not necessary) condition for a unique stationary prob distribution $\pi$ to exist.[1]

Let

$$P(x'|x) = P(\underline{a} = 1|x', x)P_{\underline{c}|\underline{x}}(x'|x) + \delta(x, x')P(\underline{a} = 0|x) \ , \tag{25.24}$$

where

$$P(\underline{a} = 0|x) = \sum_{x'} P(\underline{a} = 0|x', x)P_{\underline{c}|\underline{x}}(x'|x) \ . \tag{25.25}$$

**Claim 18** *If*

$$P(\underline{a} = 1|x', x) = \alpha(x'|x) \ , \tag{25.26}$$

*then detailed balance is satisfied.*

**proof:** Assume $x \neq x'$.

[1] As explained lucidly in Ref.[1], besides detailed balance, 2 other properties must also be satisfied by the Markov chain, irreducibiity and aperiodicity. However, because of how it is constructed, the Metropolis-Hastings algorithm automatically produces a Markov chain that has those 2 properties.

$$
\begin{aligned}
P(x'|x)P(x) &= P(\underline{a}=1|x',x)P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x) && (25.27)\\
&= \min\left(1, \frac{P_{\underline{c}|\underline{x}}(x|x')P_{\underline{x}}(x')}{P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x)}\right) P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x) && (25.28)\\
&= \min\left(P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x), P_{\underline{c}|\underline{x}}(x|x')P_{\underline{x}}(x')\right) && (25.29)\\
&= P(x|x')P(x') && (25.30)
\end{aligned}
$$

**QED**

# Gibbs Sampling

For more info about this topic and some original references, see Ref.[42].

Gibbs sampling is a special case of Metropolis-Hastings sampling. Gibbs sampling is ideally suited for application to a bnet, because it is stated in terms of the conditional prob distributions of $N$ random variables, and conditional prob distributions are part of the definition of a bnet.

Consider a bnet with nodes $\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{N-1}$

Identify the random variable $\underline{x} = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{N-1})$ with the random variable $\underline{x}$ used in Metropolis-Hastings sampling. For Gibbs sampling, we use the following proposal distribution:

$$
P_{\underline{c}|\underline{x}}(c|m) = \prod_{j=0}^{N-1} P(c_j \mid [m_i]_{i\neq j}) \; . \tag{25.31}
$$

Eq.(25.31) can be simplified using Markov Blankets (see Chapter 24) to the following:

$$
P_{\underline{c}|\underline{x}}(c|m) = \prod_{j=0}^{N-1} P(c_j \mid [m_i : \forall i \ni \underline{x}_i \in MB(\underline{x}_j)]) \; , \tag{25.32}
$$

where, for any node $\underline{a}$, we denote its Markov blanket by $MB(\underline{a})$.

An alternative proposal distribution that leads to much faster convergence is as follows. The idea is to make the components $c_j^{(t)}$ of candidate sample $c^{(t)}$ depend on the previous components $(c_i^{(t)})_{i<j}$. See the bnet Fig.25.7. The TPM for the nodes of that bnet are

$$
P(\underline{c}_j^{(t)} = c_j \mid (\underline{c}_i^{(t)})_{i<j} = (c_i)_{i<j}, \underline{m}^{(t-1)} = m) = P(c_j|(c_i)_{i<j}, (m_i)_{i>j}) \tag{25.33}
$$

for $j = 0, 1, \ldots, N-1$. This implies

$$
P_{\underline{c}|\underline{x}}(\underline{c}^{(t)} = c|\underline{m}^{(t-1)} = m) = \prod_{j=0}^{N-1} P(c_j|(c_i)_{i<j}, (m_i)_{i>j}) \; . \tag{25.34}
$$

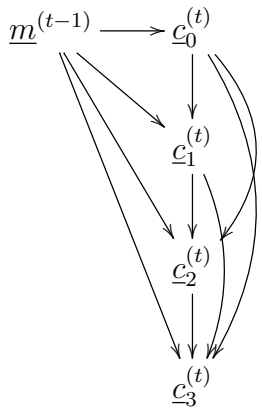


Figure 25.7: In Gibbs sampling, the proposal distribution $P_{\underline{c}|\underline{x}}$ can be defined by making the components $c_j^{(t)}$ of candidate sample $c^{(t)}$ depend on the previous components $(c_i^{(t)})_{i<j}$.

As before, we can condition only on the Markov blanket of each node $\underline{x}_j$.

$$P_{\underline{c}|\underline{x}}(\underline{c}^{(t)} = c|\underline{m}^{(t-1)} = m) = \prod_{j=0}^{N-1} P(c_j|(c_i)_{i<j}, (m_i)_{i>j}, \text{ use only } c_i \text{ and } m_i \ni \underline{x}_i \in MB(\underline{x}_j)) \ . \tag{25.35}$$

# Importance Sampling

For more info about this topic and some original references, see Ref.[44].

Suppose random variables $\underline{x}[\sigma] \in S_{\underline{x}}$ for $\sigma = 0, 1, \ldots, nsam - 1$ are i.i.d. with probability distribution $P_{\underline{x}}$. Then

$$E_{\underline{x}}[f(x)] \approx \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} f(x[\sigma]) \tag{25.36}$$

for any $f : S_{\underline{x}} \to \mathbb{R}$. Sometimes, instead of using i.i.d. samples $\underline{x}[\sigma] \in S_{\underline{x}}$ where $\underline{x}[\sigma] \sim P_{\underline{x}}$, we wish to use i.i.d. samples $\underline{y}[\sigma] \in S_{\underline{x}}$ where $\underline{y}[\sigma] \sim P_{\underline{y}}$.

$$\begin{aligned} E_{\underline{x}}[f(\underline{x})] &= \sum_x P_{\underline{x}}(x) f(x) && (25.37) \\ &= \sum_x P_{\underline{y}}(x)) \frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} f(x) && (25.38) \\ &= E_{\underline{y}}[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)] && (25.39) \end{aligned}$$

Sampling from $P_{\underline{y}}(y)$ instead of $P_{\underline{x}}(x)$ might reduce (or increase) variance for a particular $f : S_{\underline{x}} \to \mathbb{R}$.

$$Var_{\underline{x}}[f(x)] = E_{\underline{x}}[(f(x))^2] - (E_{\underline{x}}[f(x)])^2 \tag{25.40}$$

$$\begin{aligned} Var_{\underline{y}}[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)] &= E_{\underline{y}}[(\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y))^2] - (E_{\underline{y}}[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)])^2 & (25.41) \\ &= E_{\underline{x}}[\frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} (f(x))^2] - (E_{\underline{x}}[f(x)])^2 & (25.42) \end{aligned}$$

# Chapter 28

# Missing Data, Imputation

This chapter assumes that the reader has read some parts of Chapter 14 on the Expectation Maximization (EM) algo and Chapter 25 on Markov Chain Monte Carlo (MCMC).

| | $h_0$ | $x_0$ | $x_1$ | $x_2$ |
|---|---|---|---|---|
| 1 | NA | 0 | 1 | 1 |
| 2 | NA | 0 | 0 | 0 |
| 3 | NA | 1 | 1 | 0 |
| 4 | NA | NA | 1 | NA |
| 5 | NA | 0 | NA | 1 |
| 6 | NA | 0 | 0 | 1 |

| | $h_0$ | $x_0$ | $x_1$ | $x_2$ | $m$ |
|---|---|---|---|---|---|
| 1 | NA | 0 | 1 | 1 | (0,0,0) |
| 2 | NA | 0 | 0 | 0 | (0,0,0) |
| 3 | NA | 1 | 1 | 0 | (0,0,0) |
| 4 | NA | 0<br>0<br>1<br>1 | 1 | 0<br>1<br>0<br>1 | (1,0,1) |
| 5 | NA | 0 | 0<br>1 | 1 | (0,1,0) |
| 6 | NA | 0 | 0 | 1 | (0,0,0) |

Table 28.1: **Left Table:** Dataset with $nsam = 6$ and some missing entries, for 4 binary variables $h_0, x_0, x_1, x_2$. NA=not available. The $h_0$ column is completely missing because $h_0$ is an unobserved latent variable. **Right Table:** All possibilities for $x_i = NA$ cells of left table have been enumerated. A new column labeled $m$ has been added. $m_i = \mathbb{1}(x_i \text{ is missing})$ for $i = 0, 1, 2$.

Suppose you have compiled a dataset from a study. It consists of $nsam$ number of samples (sample= row), and $nx$ columns (each column is a different feature, or observation). Suppose that some of the cells in this matrix are empty. Throwing away all the incomplete rows is okay if the number of incomplete rows is much smaller than $nsam$. If not, throwing them away would throw away a substantial amount of information from all the filled cells in those incomplete rows, plus it might bias your dataset. This chapter deals with how to fill those empty cells with plausible fake data. A fancy name for this process is **imputation**. There is no unique way of fabricating fake data, but some fakes are better than others by some metrics. This chapter will consider two popular ways (EM and MCMC) of filling those empty cells with their "most likely" values based on the cells of the dataset that aren't missing, and perhaps also based on some model (DAG) that is expected to describe well the dataset.

Notation: $\vec{\underline{a}} = (\underline{a}[\sigma])_{\sigma=0,1,\ldots,nsam-1}$, where $nsam$ is the number of samples. Will sometimes denote $a[\sigma]$ by $a^{[\sigma]}$.

For concreteness, we will apply the concepts of this chapter to the dataset with missing data given by Table 28.1.

# Imputation via EM

We begin by augmenting Fig.14.1 (the first figure in Chapter 14). We augment it to Figs.28.1 and 28.2 by adding a new node $\vec{\underline{m}}$ called the **missingness variable**. There are 3 popular ways of connecting node $\vec{\underline{m}}$ to the other nodes in the graph. For doing imputation via EM, we connect node $\vec{\underline{m}}$ as shown in the middle bnet (called MAR) of Fig.28.1. Recall that node $\underline{\theta}$ represents the **unknown parameters**, node $\vec{\underline{x}}$ represents the **observed variables**, and node $\vec{\underline{h}}$ represents the **latent variables**. Both $\underline{\theta}$ and $\vec{\underline{h}}$ are hidden (i.e., unobserved).

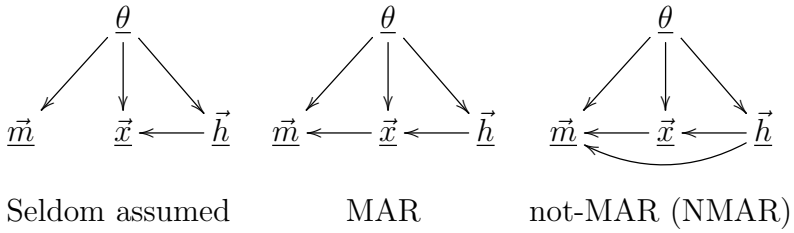


Figure 28.1: The left bnet is seldom assumed. The middle bnet is referred to as the MAR (missing at random) assumption. The right bnet is referred to as the not-MAR (NMAR) assumption.

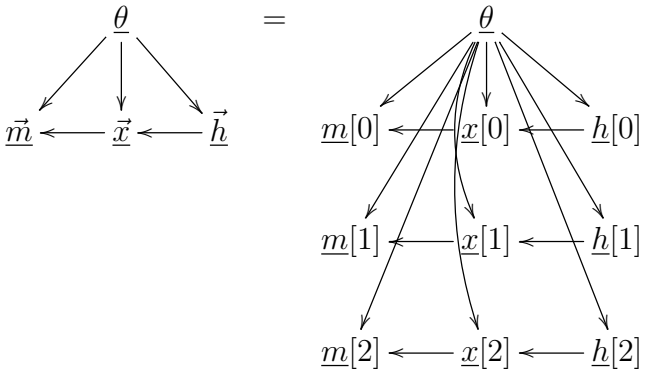


Figure 28.2: MAR bnet with $nsam = 3$.

From Fig.28.1, we have

$$P(\vec{m}|\vec{x},\vec{h},\theta) = \begin{cases} P(\vec{m}|\theta) & \text{Seldom assumed. Called missing-CAR (MCAR)} \\ P(\vec{m}|\vec{x},\theta) & \text{MAR} \\ P(\vec{m}|\vec{x},\vec{h},\theta) & \text{not-MAR (NMAR)} \end{cases} . \quad (28.1)$$

For the example of Table 28.1, we have variables $\underline{\vec{m}}, \underline{\vec{x}}$ and $\underline{\vec{h}}$ whose values range over the following sets:

$\underline{\vec{x}} = (\underline{\vec{x}}_0, \underline{\vec{x}}_1, \underline{\vec{x}}_2)$
$\underline{\vec{h}} = (\underline{\vec{h}}_0)$
$\underline{h}_0[\sigma] \in \{0, 1\}$,
$\underline{x}_i[\sigma] \in \{0, 1\}$ for $i = 0, 1, 2$,
$\underline{m}_i[\sigma] \in \{0, 1\}$ for $i = 0, 1, 2$.

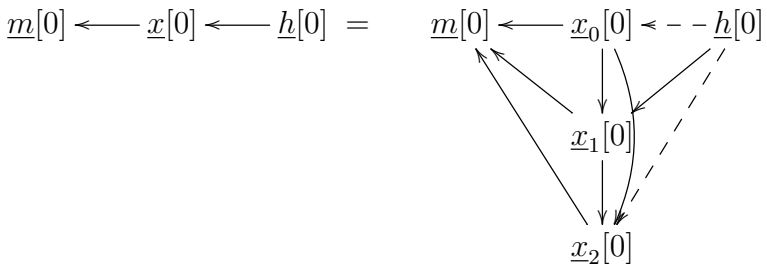


Figure 28.3: Our example for imputation via EM assumes this bnet between nodes $\underline{m}[\sigma], \underline{x}[\sigma], \underline{h}[\sigma]$.

For concreteness, we will assume that the Markov chain $\underline{m}[\sigma] \leftarrow \underline{x}[\sigma] \leftarrow \underline{h}[\sigma]$ has a finer grained DAG structure given by Fig.28.3. where we will omit the dashed arrows. If one doesn't want to assume that the data can be fitted well by the bnet of Fig.28.3 without the dashed arrows, one can include those arrows too, at the expense of more unknown parameters (i.e., degrees of freedom) to be lumped into $\theta$. We will parmaterize the TPMs corresponding to Fig.28.3 using a Categorical Distribution for each column of the TPMs. We will thus assume that the bnet of Fig.28.3 has the following TPMs, printed in blue.

$$P(h_0^{[\sigma]}|\theta) = \begin{array}{c|c} & \\ \hline & 1-\theta_0 \\ 1 & \theta_0 \end{array} \tag{28.2}$$

$$P(x_0^{[\sigma]}|\theta) = \begin{array}{c|c} & \\ \hline 0 & 1-\theta_1 \\ 1 & \theta_1 \end{array} \tag{28.3}$$

$$P(x_1^{[\sigma]} \mid x_0^{[\sigma]}, h^{[\sigma]}, \theta) = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 0 & 1-\theta_2 & 1-\theta_3 & 1-\theta_4 & 1-\theta_5 \\ 1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{array} \tag{28.4}$$

$$P(x_2^{[\sigma]} \mid x_1^{[\sigma]}, x_0^{[\sigma]}, \theta) = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 0 & 1-\theta_6 & 1-\theta_7 & 1-\theta_8 & 1-\theta_9 \\ 1 & \theta_6 & \theta_7 & \theta_8 & \theta_9 \end{array} \tag{28.5}$$

$$P(m^{[\sigma]}|x^{[\sigma]},\theta) = \frac{1}{nsam} P((x_i)_{\forall i \ni m_i=1} \mid (x_i)_{\forall i \ni m_i=0}, \theta) \tag{28.6}$$

Eq.(28.6) can be illustrated as follows. In Table 28.2, we added a $P(m)$ column to Table 28.1.

| | $h_0$ | $x_0$ | $x_1$ | $x_2$ | $m$ | $P(m)$ |
|---|---|---|---|---|---|---|
| 1 | NA | 0 | 1 | 1 | (0,0,0) | $\frac{1}{nsam}$ |
| 2 | NA | 0 | 0 | 0 | (0,0,0) | $\frac{1}{nsam}$ |
| 3 | NA | 1 | 1 | 0 | (0,0,0) | $\frac{1}{nsam}$ |
| 4 | NA | 0<br>0<br>1<br>1 | 1 | 0<br>1<br>0<br>1 | (1,0,1) | $\frac{1}{nsam}P(x_0=0,x_2=0 \mid x_1=1,\theta)$<br>$\frac{1}{nsam}P(x_0=0,x_2=1 \mid x_1=1,\theta)$<br>$\frac{1}{nsam}P(x_0=1,x_2=0 \mid x_1=1,\theta)$<br>$\frac{1}{nsam}P(x_0=1,x_2=1 \mid x_1=1,\theta)$ |
| 5 | NA | 0 | 0<br>1 | 1 | (0,1,0) | $\frac{1}{nsam}P(x_1=0 \mid x_0=0,x_2=1,\theta)$<br>$\frac{1}{nsam}P(x_1=1 \mid x_0=0,x_2=1,\theta)$ |
| 6 | NA | 0 | 0 | 1 | (0,0,0) | $\frac{1}{nsam}$ |

Table 28.2: $P(m)$ column added to Table 28.1. Note that $\sum_m P(m) = 1$.

$$\theta = (\theta_i)_{i=0,1,\ldots,9} \tag{28.7}$$

$$P(m^{[\sigma]},x^{[\sigma]},h^{[\sigma]}|\theta) = P(m^{[\sigma]}|x^{[\sigma]},\theta)P(x^{[\sigma]}|h^{[\sigma]},\theta)P(h^{[\sigma]}|\theta) \tag{28.8}$$

$$P(x^{[\sigma]}|h^{[\sigma]},\theta) = P(x_2^{[\sigma]}|x_1^{[\sigma]},x_0^{[\sigma]},\theta)P(x_1^{[\sigma]}|x_0^{[\sigma]},h^{[\sigma]},\theta)P(x_0^{[\sigma]}|\theta) \tag{28.9}$$

$$P(x_1^{[\sigma]}|x_0^{[\sigma]},\theta) = \sum_h P(x_1^{[\sigma]}|x_0^{[\sigma]},h^{[\sigma]},\theta)P(h^{[\sigma]}|\theta) \tag{28.10}$$

$$P(x^{[\sigma]}|\theta) = P(x_2^{[\sigma]}|x_1^{[\sigma]},x_0^{[\sigma]},\theta)P(x_1^{[\sigma]}|x_0^{[\sigma]},\theta)P(x_0^{[\sigma]}|\theta) \tag{28.11}$$

$$
\begin{aligned}
Q(\theta|\theta^{(t)}) &= \sum_{\vec{m},\vec{h}} P(\vec{m},\vec{h} \mid \vec{x},\theta^{(t)}) \ln P(\vec{m},\vec{x},\vec{h}|\theta) && (28.12)\\
&= \sum_{\vec{m},\vec{h}} \left[\prod_{\sigma} P(m^{[\sigma]},h^{[\sigma]} \mid x^{[\sigma]},\theta^{(t)})\right] \ln \left[\prod_{\sigma} P(m^{[\sigma]},x^{[\sigma]},h^{[\sigma]}|\theta)\right] && (28.13)\\
&= \sum_{\sigma} \sum_{m^{[\sigma]},h^{[\sigma]}} P(m^{[\sigma]},h^{[\sigma]} \mid x^{[\sigma]},\theta^{(t)}) \ln P(m^{[\sigma]},x^{[\sigma]},h^{[\sigma]}|\theta) && (28.14)\\
&= \sum_{\sigma} \sum_{m^{[\sigma]},h^{[\sigma]}} \frac{P(m^{[\sigma]},h^{[\sigma]},x^{[\sigma]} \mid \theta^{(t)})}{P(x^{[\sigma]} \mid \theta^{(t)})} \ln P(m^{[\sigma]},x^{[\sigma]},h^{[\sigma]}|\theta) && (28.15)
\end{aligned}
$$

Once you find optimal parameters $\theta^*$ by recursing this $Q(\theta|\theta^{(t)})$, you can evaluate numerically the $P(m)$ column of Table 28.2. In Table 28.2, out of the 4 sub-rows for row 4, choose the one with the highest probability. Similarly, out of the 2 sub-rows for row 5, choose the one with the highest probability.

# Imputation via MCMC

A simple and popular way to do inputation via MCMC is described in Ref.[26]. It goes as follows.

Let

$$\underline{H}[\sigma] = (\underline{h}[\sigma], \underline{m}[\sigma]) \tag{28.16}$$

for $\sigma = 0, 1, \ldots, nsam-1$. Initialize $\theta^{(0)}$ to a random value within the allowed ranges. Do the following 2 steps, for $t = 0, 1, \ldots, T-1$, where $T$ is large enough that $\theta^{(t)}$ has reached a steady value that is independent of $\theta^{(0)}$. To do the sampling, use a standard sampling technique such as Gibbs sampling.

- **STEP 1:** For $\sigma = 0, 1, \ldots, nsam-1$, find a sample

$$(H^{[\sigma]})^{(t+1)} \sim P(H^{[\sigma]}|x^{[\sigma]},\theta^{(t)}) \; . \tag{28.17a}$$

- **STEP 2:** Find a sample

$$\theta^{(t+1)} \sim P^{(t+1)}(\theta) \tag{28.17b}$$

where

$$P^{(t+1)}(\theta) = \mathcal{N}(!\theta) \prod_{\sigma} P(x^{[\sigma]},(H^{[\sigma]})^{(t+1)}|\theta) \; . \tag{28.17c}$$

Fig.28.4 illustrates this two step process using a bnet.

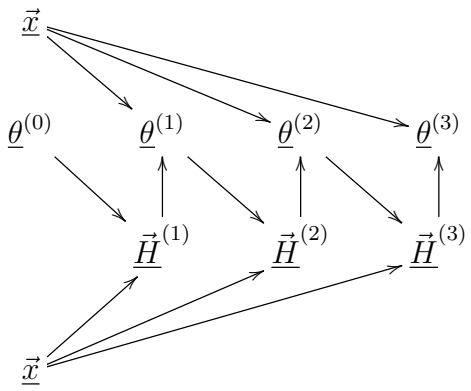


Figure 28.4: bnet illustrating Eqs.(28.17) for doing imputation via MCMC. The **same** node $\underline{\vec{x}}$ appears twice to make the graph clearer.

# Multiple Imputations

**Multiple imputations** means calculating $\theta^*$ (i.e., the optimum $\theta$) and the concomitant dataset $\vec{x}^*, \vec{H}^*$ , via any method (such as EM or MCMC), a large number of times, starting from different, randomly chosen $\theta^{(0)}$ initial parameters. Then calculating the average and the variance of $\theta^*, \vec{x}^*, \vec{H}^*$ and functions thereof.

# Bibliography


[1] Dan Bendel. Metropolis-Hastings: A comprehensive overview and proof. https://similarweb.engineering/mcmc/.

[2] Alexandra M Carvalho. Scoring functions for learning Bayesian networks. http://www.lx.it.pt/~asmc/pub/talks/09-TA/ta_pres.pdf.

[3] Charles Fox, Neil Girdhar, and Kevin Gurney. A causal Bayesian network view of reinforcement learning. https://www.aaai.org/Papers/FLAIRS/2008/FLAIRS08-030.pdf".

[4] Bruno Gonçalves. Model testing and causal search. blog post https://medium.com/data-for-science/causal-inference-part-vii-model-testing-and-causal-search-536b796f

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, David Warde-Farley Bing Xu, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. https://arxiv.org/abs/1406.2661.

[6] Christina Heinze-Deml. Causality, spring semester 2019 at ETH Zurich. https://stat.ethz.ch/lectures/ss19/causality.php#course_materials.

[7] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International journal of approximate reasoning*, 15(3):225–263, 1996. http://www.ar-tiste.com/Huang-Darwiche1996.pdf.

[8] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988. http://www.eecis.udel.edu/~shatkay/Course/papers/Lauritzen1988.pdf.

[9] Sergey Levine. Course CS 285 at UC Berkeley, Deep reinforcement learning. http://rail.eecs.berkeley.edu/deeprlcourse/.

[10] Dimitris Margaritis. Learning Bayesian network model structure from data (thesis, 2003, Carnegie Mellon Univ). https://apps.dtic.mil/sti/citations/ADA461103.

[11] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. In *BMC bioinformatics*, volume 7, page S7. Springer, 2006. https://link.springer.com/article/10.1186/1471-2105-7-S1-S7.

[12] Robert J. McEliece, David J. C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearls belief propagation algorithm. http://authors.library.caltech.edu/6938/1/MCEieeejstc98.pdf.

[13] Richard E Neapolitan. *Learning Bayesian networks*. Pearson Prentice Hall, 2004.

[14] Andrew Ng. Lecture at deeplearning.ai on recurrent neural networks. http://www.ar-tiste.com/ng-lec-rnn.pdf.

[15] Gregory Nuel. Tutorial on exact belief propagation in Bayesian networks: from messages to algorithms. https://arxiv.org/abs/1201.4724.

[16] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. https://www.aaai.org/Papers/AAAI/1982/AAAI82-032.pdf, 1982.

[17] Judea Pearl. *Probabilistic Inference in Intelligent Systems*. Morgan Kaufmann, 1988.

[18] Judea Pearl. *Causality: Models, Reasoning, and Inference, Second Edition*. Cambridge University Press, 2013.

[19] Judea Pearl. Causal and counterfactual inference. *The Handbook of Rationality*, pages 1–41, 2019. https://ftp.cs.ucla.edu/pub/stat_ser/r485.pdf.

[20] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[21] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[22] ReliaSoft. System analysis reference. http://reliawiki.org/index.php/System_Analysis_Reference.

[23] Marco Scutari. bnlearn. https://www.bnlearn.com/.

[24] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019. https://arxiv.org/abs/1805.11908.

[25] Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[26] Masayoshi Takahashi. Statistical inference in missing data by MCMC and non-MCMC multiple imputation algorithms: Assessing the effects of between-imputation iterations. *Data Science Journal*, 16, 2017. https://datascience.codata.org/articles/10.5334/dsj-2017-037/.

[27] theinvestorsbook.com. Pert analysis. https://theinvestorsbook.com/pert-analysis.html.

[28] Robert R. Tucci. Bell's inequalities for Bayesian statisticians. blog post in blog Quantum Bayesian Networks, https://qbnets.wordpress.com/2008/09/19/bells-inequaties-for-bayesian-statistician/.

[29] Robert R. Tucci. Quantum Fog. https://github.com/artiste-qb-net/quantum-fog.

[30] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook nureg-0492. https://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/.

[31] Wikipedia. Belief propagation. https://en.wikipedia.org/wiki/Belief_propagation.

[32] Wikipedia. Beta function. https://en.wikipedia.org/wiki/Beta_function.

[33] Wikipedia. Binary decision diagram. https://en.wikipedia.org/wiki/Binary_decision_diagram.

[34] Wikipedia. Boolean algebra. https://en.wikipedia.org/wiki/Boolean_algebra.

[35] Wikipedia. Categorical distribution. https://en.wikipedia.org/wiki/Categorical_distribution.

[36] Wikipedia. Chow-Liu tree. https://en.wikipedia.org/wiki/Chow%E2%80%93Liu_tree.

[37] Wikipedia. Data processing inequality. https://en.wikipedia.org/wiki/Data_processing_inequality.

[38] Wikipedia. Dirichlet distribution. https://en.wikipedia.org/wiki/Dirichlet_distribution.

[39] Wikipedia. Expectation maximization. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.

[40] Wikipedia. Gamma function. https://en.wikipedia.org/wiki/Gamma_function.

[41] Wikipedia. Gated recurrent unit. https://en.wikipedia.org/wiki/Gated_recurrent_unit.

[42] Wikipedia. Gibbs sampling. https://en.wikipedia.org/wiki/Gibbs_sampling.

[43] Wikipedia. Hidden Markov model. https://en.wikipedia.org/wiki/Hidden_Markov_model.

[44] Wikipedia. Importance sampling. https://en.wikipedia.org/wiki/Importance_sampling.

[45] Wikipedia. Inverse transform sampling. https://en.wikipedia.org/wiki/Inverse_transform_sampling.

[46] Wikipedia. Junction tree algorithm. https://en.wikipedia.org/wiki/Junction_tree_algorithm.

[47] Wikipedia. k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering.

[48] Wikipedia. Kalman filter. https://en.wikipedia.org/wiki/Kalman_filter.

[49] Wikipedia. Long short term memory. https://en.wikipedia.org/wiki/Long_short-term_memory.

[50] Wikipedia. Markov blanket. https://en.wikipedia.org/wiki/Markov_blanket.

[51] Wikipedia. Metropolis-Hastings method. https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm.

[52] Wikipedia. Minimum spanning tree. https://en.wikipedia.org/wiki/Minimum_spanning_tree.

[53] Wikipedia. Monte Carlo methods. https://en.wikipedia.org/wiki/Category:Monte_Carlo_methods.

[54] Wikipedia. Multinomial distribution. https://en.wikipedia.org/wiki/Multinomial_distribution.

[55] Wikipedia. Multinomial theorem. https://en.wikipedia.org/wiki/Multinomial_theorem.

[56] Wikipedia. Multivariate normal distribution. https://en.wikipedia.org/wiki/Multivariate_normal_distribution.

[57] Wikipedia. Non-negative matrix factorization. https://en.wikipedia.org/wiki/Non-negative_matrix_factorization.

[58] Wikipedia. Program evaluation and review technique. https://en.wikipedia.org/wiki/Program_evaluation_and_review_technique.

[59] Wikipedia. Rejection sampling. https://en.wikipedia.org/wiki/Rejection_sampling.

[60] Wikipedia. Simpson's paradox. https://en.wikipedia.org/wiki/Simpson's_paradox.

[61] Wikipedia. Variational Bayesian methods. https://en.wikipedia.org/wiki/Variational_Bayesian_methods.

[62] Hao Wu and Zhaohui Steve Qin. course notes, BIOS731: Advanced statistical computing, 2016 Emory Univ. http://web1.sph.emory.edu/users/hwu30/teaching/statcomp/statcomp.html.