Bayesuvius,
a small visual dictionary of Bayesian Networks

Robert R. Tucci
www.ar-tiste.xyz

July 24, 2020

Figure 1: View of Mount Vesuvius from Pompeii

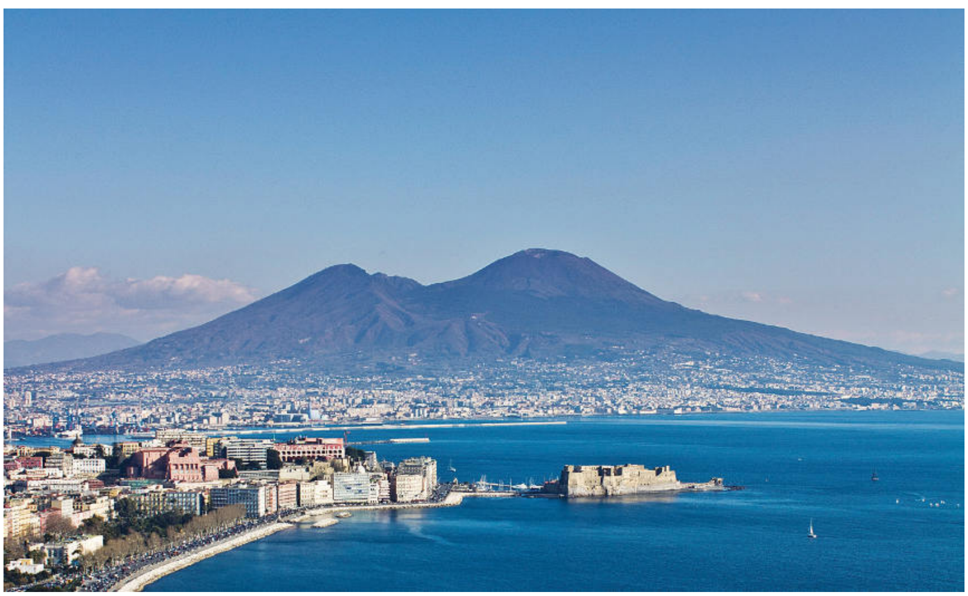

Figure 2: Mount Vesuvius and Bay of Naples

Contents

0.2 Notational Conventions

bnet=B net=Bayesian Network

Define $\mathbb{Z}, \mathbb{R}, \mathbb{C}$ to be the integers, real numbers and complex numbers, respectively.

For $a < b$, define $\mathbb{Z}_I$ to be the integers in the interval I, where $I = [a, b], [a, b), (a, b], (a, b)$ (i.e, I can be closed or open on either side).

$A_{>0} = \{k \in A : k > 0\}$ for $A = \mathbb{Z}, \mathbb{R}$.

Random Variables will be indicated by underlined letters and their values by non-underlined letters. Each node of a bnet will be labelled by a random variable. Thus, $\underline{x} = x$ means that node $\underline{x}$ is in state x.

$P_{\underline{x}}(x) = P(\underline{x} = x) = P(x)$ is the probability that random variable $\underline{x}$ equals $x \in S_{\underline{x}}$. $S_{\underline{x}}$ is the set of states (i.e., values) that $\underline{x}$ can assume and $n_{\underline{x}} = |S_{\underline{x}}|$ is the size (aka cardinality) of that set. Hence,

$$\sum_{x \in S_{\underline{x}}} P_{\underline{x}}(x) = 1 \tag{1}$$

$$P_{\underline{x},\underline{y}}(x, y) = P(\underline{x} = x, \underline{y} = y) = P(x, y) \tag{2}$$

$$P_{\underline{x}|\underline{y}}(x|y) = P(\underline{x} = x|\underline{y} = y) = P(x|y) = \frac{P(x, y)}{P(y)} \tag{3}$$

Kronecker delta function: For x, y in discrete set S,

$$\delta(x, y) = \begin{cases} 1 \text{ if } x = y \\ 0 \text{ if } x \neq y \end{cases} \tag{4}$$

Dirac delta function: For $x, y \in \mathbb{R}$,

$$\int_{-\infty}^{+\infty} dx\ \delta(x - y) f(x) = f(y) \tag{5}$$

Transition probability matrix of a node of a bnet can be either a discrete or a continuous probability distribution. To go from continuous to discrete, one replaces integrals over states of node by sums over new states, and Dirac delta functions by Kronecker delta functions. More precisely, consider a function $f : S \to \mathbb{R}$. Let $S_{\underline{x}} \subset S$ and $S \to S_{\underline{x}}$ upon discretization (binning). Then

$$\int_S dx\ P_{\underline{x}}(x) f(x) \to \frac{1}{n_{\underline{x}}} \sum_{x \in S_{\underline{x}}} f(x)\ . \tag{6}$$

Both sides of last equation are 1 when $f(x) = 1$. Furthermore, if $y \in S_{\underline{x}}$, then

$$\int_S dx\ \delta(x - y) f(x) = f(y) \to \sum_{x \in S_{\underline{x}}} \delta(x, y) f(x) = f(y)\ . \tag{7}$$

Indicator function (aka Truth function):

$$\mathbb{1}(\mathcal{S}) = \begin{cases} 1 \text{ if } \mathcal{S} \text{ is true} \\ 0 \text{ if } \mathcal{S} \text{ is false} \end{cases} \tag{8}$$

For example, $\delta(x,y) = \mathbb{1}(x=y)$.

$$\vec{x} = (x[0], x[1], x[2] \ldots, x[nsam(\vec{x})-1]) = x[:] \tag{9}$$

$nsam(\vec{x})$ is the number of samples of $\vec{x}$. $\underline{x}[i]$ are i.d.d. (independent identically distributed) samples with

$$x[i] \sim P_{\underline{x}} \;\; \text{(i.e. } P_{\underline{x[i]}} = P_{\underline{x}}) \tag{10}$$

$$P(\underline{x} = x) = \frac{1}{nsam(\vec{x})} \sum_i \mathbb{1}(x[i] = x) \tag{11}$$

If we use two sampled variables, say $\vec{x}$ and $\vec{y}$, in a given bnet, their number of samples $nsam(\vec{x})$ and $nsam(\vec{y})$ need not be equal.

$$P(\vec{x}) = \prod_i P(x[i]) \tag{12}$$

$$\sum_{\vec{x}} = \prod_i \sum_{x[i]} \tag{13}$$

$$\partial_{\vec{x}} = [\partial_{x[0]}, \partial_{x[1]}, \partial_{x[2]}, \ldots, \partial_{x[nsam(\vec{x})-1]}] \tag{14}$$

$$\begin{aligned} P(\vec{x}) &\approx [\prod_x P(x)^{P(x)}]^{nsam(\vec{x})} && (15) \\ &= e^{nsam(\vec{x})\sum_x P(x)\log P(x)} && (16) \\ &= e^{-nsam(\vec{x})H(P_{\underline{x}})} && (17) \end{aligned}$$

$$f^{[1,\partial_x,\partial_y]}(x,y) = [f, \partial_x f, \partial_y f] \tag{18}$$

$$f^+ = f^{[1,\partial_x,\partial_y]} \tag{19}$$

For probabilty distributions $p(x), q(x)$ of $x \in S_{\underline{x}}$

- Entropy:

$$H(p) = -\sum_x p(x) \log p(x) \geq 0 \tag{20}$$

- Kullback-Liebler divergence:

$$D_{KL}(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0 \tag{21}$$

- Cross entropy:

$$\begin{aligned} CE(p \to q) &= -\sum_x p(x) \log q(x) && (22) \\ &= H(p) + D_{KL}(p \parallel q) && (23) \end{aligned}$$

Normal Distribution: $x, \mu, \sigma \in \mathbb{R}$, $\sigma > 0$

$$\mathcal{N}(\mu, \sigma^2)(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{24}$$

Uniform Distribution: $a < b$, $x \in [a, b]$

$$\mathcal{U}(a, b)(x) = \frac{1}{b-a} \tag{25}$$

Expected Value

Given a random variable $\underline{x}$ with states $S_{\underline{x}}$ and a function $f : S_{\underline{x}} \to \mathbb{R}$, define

$$E_{\underline{x}}[f(\underline{x})] = E_{x\sim P(x)}[f(x)] = \sum_x P(x) f(x) \tag{26}$$

Conditional Expected Value

Given a random variable $\underline{x}$ with states $S_{\underline{x}}$, a random variable $\underline{y}$ with states $S_{\underline{y}}$, and a function $f : S_{\underline{x}} \times S_{\underline{y}} \to \mathbb{R}$, define

$$E_{\underline{x}|\underline{y}}[f(\underline{x}, \underline{y})] = \sum_x P(x|\underline{y}) f(x, \underline{y}) \ , \tag{27}$$

$$E_{\underline{x}|\underline{y}=y}[f(\underline{x}, y)] = E_{\underline{x}|y}[f(\underline{x}, y)] = \sum_x P(x|y) f(x, y) \ . \tag{28}$$

Note that

$$\begin{aligned}
E_{\underline{y}}[E_{\underline{x}|\underline{y}}[f(\underline{x}, \underline{y})]] &= \sum_{x,y} P(x|y)P(y)f(x, y) && (29)\\
&= \sum_{x,y} P(x, y) f(x, y) && (30)\\
&= E_{\underline{x},\underline{y}}[f(\underline{x}, \underline{y})] \ . && (31)
\end{aligned}$$

Sigmoid function: For $x \in \mathbb{R}$,

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}} \tag{32}$$

$\mathcal{N}(!a)$ will denote a normalization constant that does not depend on a. For example, $P(x) = \mathcal{N}(!x)e^{-x}$ where $\int_0^\infty dx \ P(x) = 1$.

Chapter 20

Turbo Codes

This chapter is based on Ref.[12].

In this chapter, vectors with n components will be indicated by an n superscript. For example, $a^n = (a_0, a_1, \ldots, a_{n-1})$.

Consider an n-letter message $u^n = (u_0, u_1, \ldots, u_{n-1})$, where for all i, $u_i \in \mathcal{A}$ is an element of an alphabet $\mathcal{A}$, and where for all i, the $\underline{u}_i$ are i.i.d.. Suppose u^n is encoded deterministically in two different ways, $e_1(u^n)$ and $e_2(u^n)$. After passing through the same memoryless channel, the variables u^n, e_1, e_2 become $\tilde{u}^n, \tilde{e}_1, \tilde{e}_2$, respectively. The letter u stands for unencoded, and e for encoded. Quantities with a tilde $\tilde{u}^n, \tilde{e}_1, \tilde{e}_2$ occur after channel passage and are visible (measurable). Quantities without a tilde u^n, e_1, e_2 are hidden (unmeasurable).

The situation just described can be represented by the bnet Fig.20.1, or by its abridged version Fig.20.2. But note that the abridged version does not show explicitly the memorylessness of the channel (i.e., that the u_i for $i = 0, 1, \ldots, n-1$ pass independently through the channel).

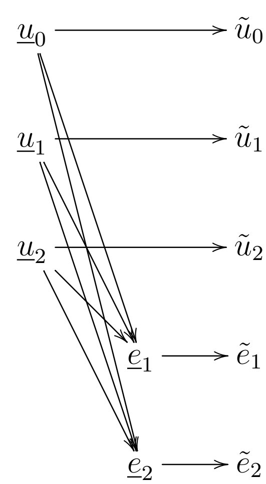

Figure 20.1: Turbo coding B net representing a message being encoded two different ways and then the original message and the 2 encodings pass through a memoryless channel.

$$
\begin{tikzcd}
\underline{u}^n \arrow[rr] \arrow[dr] \arrow[ddr] & & \tilde{u}^n \\
& \underline{e}_1 \arrow[r] & \tilde{e}_1 \\
& \underline{e}_2 \arrow[r] & \tilde{e}_2
\end{tikzcd}
$$

Figure 20.2: Abridged version of Fig.20.1.

Define

$$x = (u^n, e_1, e_2) \tag{20.1}$$

and

$$\tilde{x} = (\tilde{u}^n, \tilde{e}_1, \tilde{e}_2) \ . \tag{20.2}$$

Fig.20.1 implies that

$$P(x, \tilde{x}) = P(\tilde{u}^n|u^n) \left[\prod_{r=1,2} P(\tilde{e}_r|e_r) P(e_r|u^n) \right] P(u^n) \ . \tag{20.3}$$

Because the channel is memoryless,

$$P(\tilde{u}^n|u^n) = \prod_i P(\tilde{u}_i|u_i) \ . \tag{20.4}$$

Because the encoding is deterministic, we must have for $r = 1, 2$

$$P(e_r|u^n) = \delta(e_r, e_r(u^n)) \ . \tag{20.5}$$

Define the belief functions

$$BEL_i = BEL_i(\underline{u}_i = a) = P(\underline{u}_i = a|\tilde{x}) \ . \tag{20.6}$$

The best estimate of u_j given all visible evidence $\tilde{x}$ is

$$\hat{u}_i = \text{argmax}_{u_i} BEL_i(u_i) \ . \tag{20.7}$$

Define the probability functions

$$\pi_i = \pi_i(u_i) = P(u_i) \ , \tag{20.8}$$

and the likelihood functions

$$\lambda_i = \lambda_i(u_i) = P(\tilde{u}_i|u_i) \; . \tag{20.9}$$

For $r = 1, 2$, define the Kernel functions

$$K_r = K_r(u^n) = P(\tilde{e}_r|e_r = e_r(u^n)) \; . \tag{20.10}$$

In this book, $\mathcal{N}(!a)$ denotes a normalization constant that does not depend on a. Define

$$\mathcal{N}_i = \mathcal{N}(!u_i) \; . \tag{20.11}$$

Claim 1

$$BEL_i = \mathcal{N}_i \lambda_i \pi_i \mathcal{T}_i^{K_1 K_2}[\prod_{j \neq i} \lambda_j \pi_j] \; , \tag{20.12}$$

where $\mathcal{T}_i^{K_1 K_2}(\cdot)$ is an operator (transform) acting on functions of u^n:

$$\mathcal{T}^K(\cdot) = \sum_{u^n} \delta(u_i = a) K(u^n)(\cdot) \; . \tag{20.13}$$

proof:

$$\begin{aligned}
& P(\underline{u}_i = a|\tilde{x}) = \\
&= \sum_x \delta(u_i, a) P(x|\tilde{x}) && (20.14)\\
&= \sum_x \delta(u_i, a) \frac{P(\tilde{x}|x)P(x)}{P(\tilde{x})} && (20.15)\\
&= \mathcal{N}(!a) \sum_x \delta(u_i, a) P(\tilde{x}|x) P(x) && (20.16)\\
&= \mathcal{N}(!a) \sum_x \delta(u_i, a) P(u^n) \left[\prod_{r=1,2} P(\tilde{e}_r|e_r)\delta(e_r, e_r(u^n))\right] \prod_j P(\tilde{u}_j|u_j) && (20.17)\\
&= \mathcal{N}(!a) \lambda_i(a) \pi_i(a) R \; , && (20.18)
\end{aligned}$$

where

$$\begin{aligned}
R &= \sum_{u^n} \delta(u_i, a) \left[\prod_{r=1,2} P(\tilde{e}_r|e_r(u^n))\right] \prod_{j \neq i} P(\tilde{u}_j|u_j) P(u_j) && (20.19)\\
&= \sum_{u^n} \delta(u_i, a) \left[\prod_{r=1,2} K_r(u^n)\right] \prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j) && (20.20)\\
&= \mathcal{T}_i^{K_1 K_2}[\prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j)] \; . && (20.21)
\end{aligned}$$

Hence

$$BEL_i(a) = \mathcal{N}(!a)\lambda_i(a)\pi_i(a)\mathcal{T}_i^{K_1 K_2}[\prod_{j\neq i}\lambda_j(u_j)\pi_j(u_j)] \; . \tag{20.22}$$

QED

Decoding Algorithm

The Turbo algorithm for decoding the encode message is as follows. For $m = 0$, let

$$\pi_j^{(0)}(u_j) = \frac{1}{n_{\underline{u}_j}} \; . \tag{20.23}$$

Then for $m = 1, 2, \ldots$, let

$$\pi_i^{(m)} = \mathcal{N}_i \mathcal{T}_i^{K_{m\%2}}[\prod_{j\neq i}\lambda_j \pi_j^{(m-1)}] \; , \tag{20.24}$$

where $m\%2 = 1$ if m is odd and $m\%2 = 2$ if m is even. Furthermore, for $m > 0$, let

$$\begin{aligned} BEL_i^{(m)} &= \mathcal{N}_i\lambda_i\pi_i^{(m-1)}\pi_i^{(m)} & (20.25)\\ &= \mathcal{N}_i\lambda_i\pi_i^{(m-1)}\mathcal{T}_i^{K_{m\%2}}[\prod_{j\neq i}\lambda_j\pi_j^{(m-1)}] \; . & (20.26) \end{aligned}$$

As $m \to \infty$, $BEL_i^{(m)}$ given by Eq.(20.26) is expected to converge to the the exact BEL_i given by Eq.(20.12).

Turbo decoding can be represented by the bnets Figs.20.3 and 20.4.

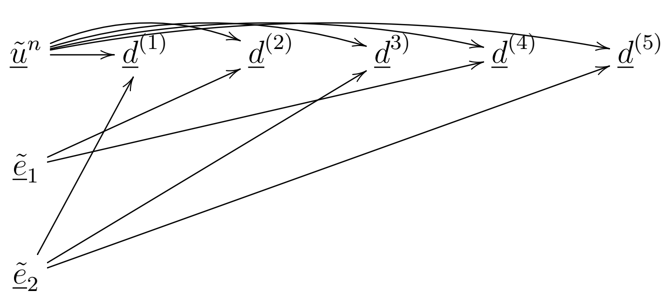

Figure 20.3: B net describing Turbo code generation of $BEL_i^{(m)}$ for $m = 1, 2, \ldots$.

The node transition matrices, printed in blue, for Fig.20.3, are given by:

$$P(d^{(m)} = a \mid \tilde{u}^n, \tilde{e}_{m\%2}) = BEL_i^{(m)}(a) \; . \tag{20.27}$$

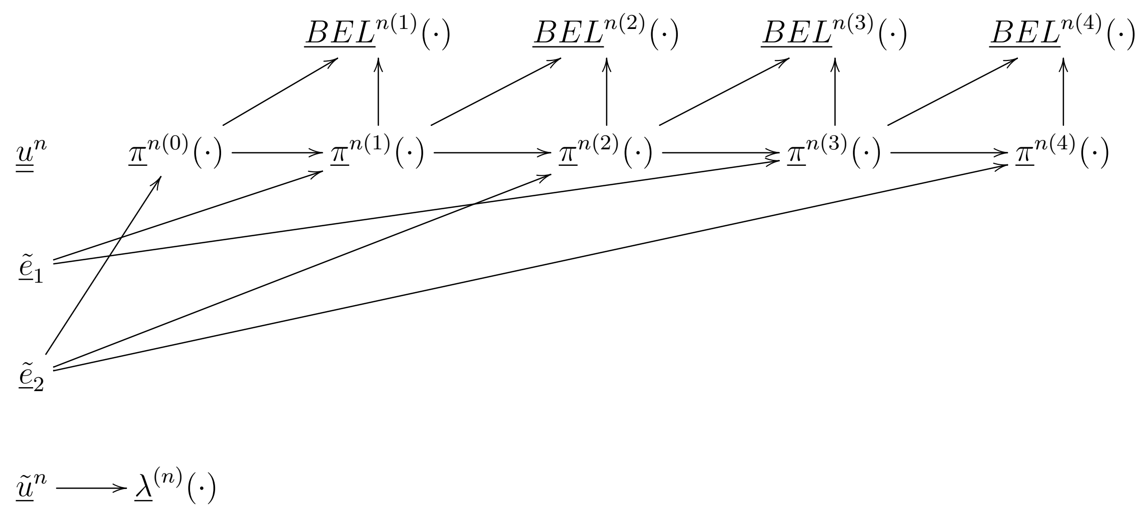

Figure 20.4: B net describing Turbo code generation of $BEL_i^{(m)}$ and $\pi^{n(m)}(\cdot)$ for $m = 0, 1, 2 \ldots$. The following arrows were not drawn so a not to unduly clutter the diagram: Arrows pointing from node $\underline{\lambda}^n(\cdot)$ to nodes $\underline{\pi}^{n(m)}(\cdot)$ and $\underline{BEL}^{n(m)}(\cdot)$ for $m = 0, 1, 2, \ldots$.

The node transition matrices, printed in blue, for Fig.20.4, are given by:

$$P(\lambda'(\cdot)|\tilde{u}^n) = \delta(\lambda'(\cdot), \lambda(\cdot)) \tag{20.28}$$

$$P(\pi^{n(m)}(\cdot)|\lambda^n(\cdot), \pi^{n(m-1)}(\cdot)) = \prod_i \prod_{u_i} \delta(\pi^{(m)}(u_i), \mathcal{N}_i \mathcal{T}_i^{K_{m\%2}}[\prod_{j \neq i} \lambda_j \pi_j^{(m-1)}]) \tag{20.29}$$

$$P(BEl^{n(m)}(\cdot)|\lambda^n(\cdot), \pi^{n(m)}(\cdot)\pi^{n(m-1)}(\cdot)) = \prod_i \prod_{u_i} \delta(BEL_i(u_i), \mathcal{N}_i \lambda_i(u_i) \pi_i^{(m-1)}(u_i) \pi_i^{(m)})(u_i) \tag{20.30}$$

Message Passing Interpretation of Decoding Algorithm

Ref.[12] shows that the Turbo code decoding algo can be interpreted as an application of Message Passing. We explain Message Passing in a separate chapter, Chapter 11.

Bibliography

[1] 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/.

[2] 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.

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

[4] 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.

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

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

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

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

[9] 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".

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

[11] Robert R. Tucci. Simpson's paradox, the bane of clinical trials. blog post in blog Quantum Bayesian Networks https://qbnets.wordpress.com/2020/07/09/simpsons-paradox-the-bane-of-clinical-trials/.

[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.