STA2700 - Graphical Models - Assignment 1

Luis Correia - Student No. 1006508566

October 10th 2020

Question 1

We consider binary (i.e., {0,1}-valued) sequences of length N, in which no two 1's are adjacent to each other. Let $\mathbf{x}$ stand for $(x_1, x_2, \ldots, x_N)$, where $x_i \in \mathcal{X} = \{0, 1\}$.

For $1 \leq i \leq N$ and for adjacent variables x_i, x_{i+1}, let the local factors be

$$f_i(x_i, x_{i+1}) = \begin{cases} 0 \text{ , if } x_i = x_{i+1} = 1 \\ 1 \text{ , otherwise} \end{cases} \tag{1}$$

The global function is then given by

$$f(\boldsymbol{x}) = \prod_{i=1}^{N-1} f_i(x_i, x_{i+1}) \tag{2}$$

and the normalization constant Z is

$$Z_N = \sum_{\boldsymbol{x} \in \mathcal{X}^N} f(\boldsymbol{x}) \tag{3}$$

Thus

$$p(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{Z_N}, \, \boldsymbol{x} \in \mathcal{X}^N \tag{4}$$

is a PMF on $\mathcal{X}^N$

Item(a)

For $N = 4$, draw the factor graph for factorization in (2)

{*Solution.*}

According with the representation learned in our lectures, we can represent the graph as follows:

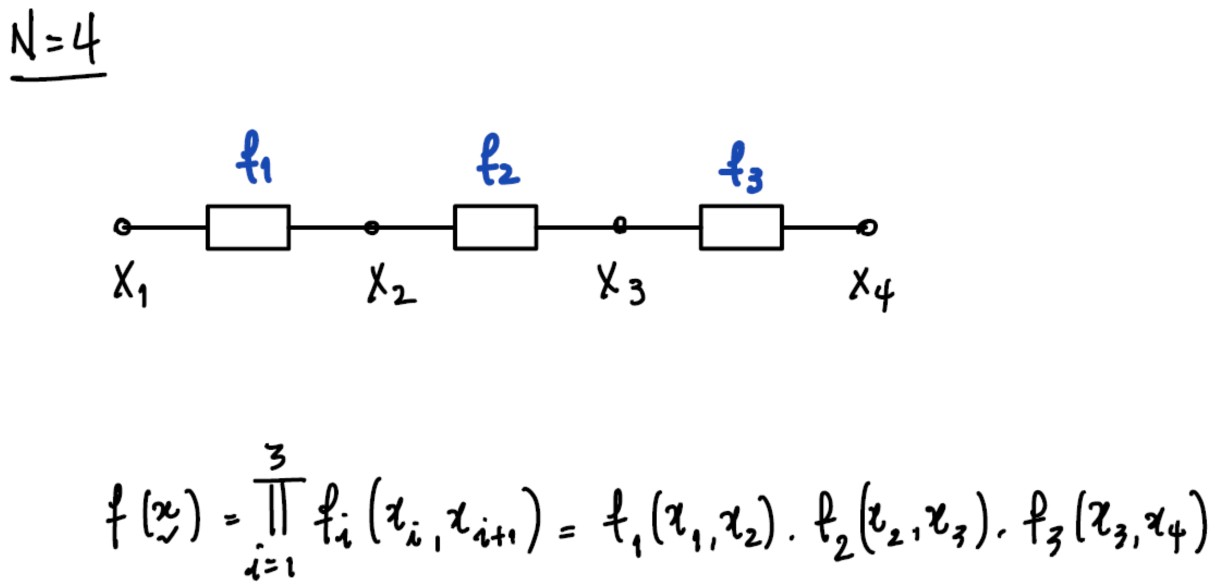

Figure 1: Graphic Representation for N=4

Item(b)

Argue that Z_N counts the number of sequences of length N in which no two 1's are adjancent to each other.

{*Solution.*}

As we will see in the sum-product algorithm, the configuration matrix for this problem, let say $\boldsymbol{M}$, is a $s \times s$ where s is the cardinality of the *configuration space* $\mathcal{X}^N$, obtained by substituting each row/column with the local-factor function calculated over all possible combination of points over $\mathcal{X}^N$.

In this case, it can be represented by :

$$\boldsymbol{M} = \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{5}$$

This matrix shows the valid sequences of messages considered by local factor functions.

Clearly, when $f(1,1) = 0$ it represents that when adjacent nodes have the same value and this value is equal to 1, we do not let into account this sequence to compute Z_N because the product of local functions will cancel whenever at least one sequence of consecutive 1's is found (i.e., it will be multiplied by *zero*).

This implies that Z_N counts the number of sequences of length N which no two 1's are adjacent to each other.

Item(c)

Apply the sum-product algorithm to compute the *exact value* of Z_N for the following values of N

$$N = 3, 5, 10, 20, 50, 10^2, 10^4, 10^6$$

Plot $ln(Z_N)/N$ as a function of N. (Use logscale on N-axis)

{*Solution.*}

We will define two functions in R to calculate Z_N:

- `Function f(a, b)` - Factor Function provided by the problem. This function receives a pair of vertices and calculates the value of local factor;
- `Function ZN(N, normalize = TRUE)` - Function which returns the value of Z_N which implements the sum-product algorithm using the configuration matrix $\boldsymbol{M}$. The boolean flag `normalize` indicates the use of step-normalization in order to avoid floating-point overflow when multiplying local functions.

Using $N = 3, 5, 10, 20, 50, 10^2, 10^4, 10^6$, the graph of $ln(Z_N)/N$ is shown below - the blue dotted line represents the asymptotic result.

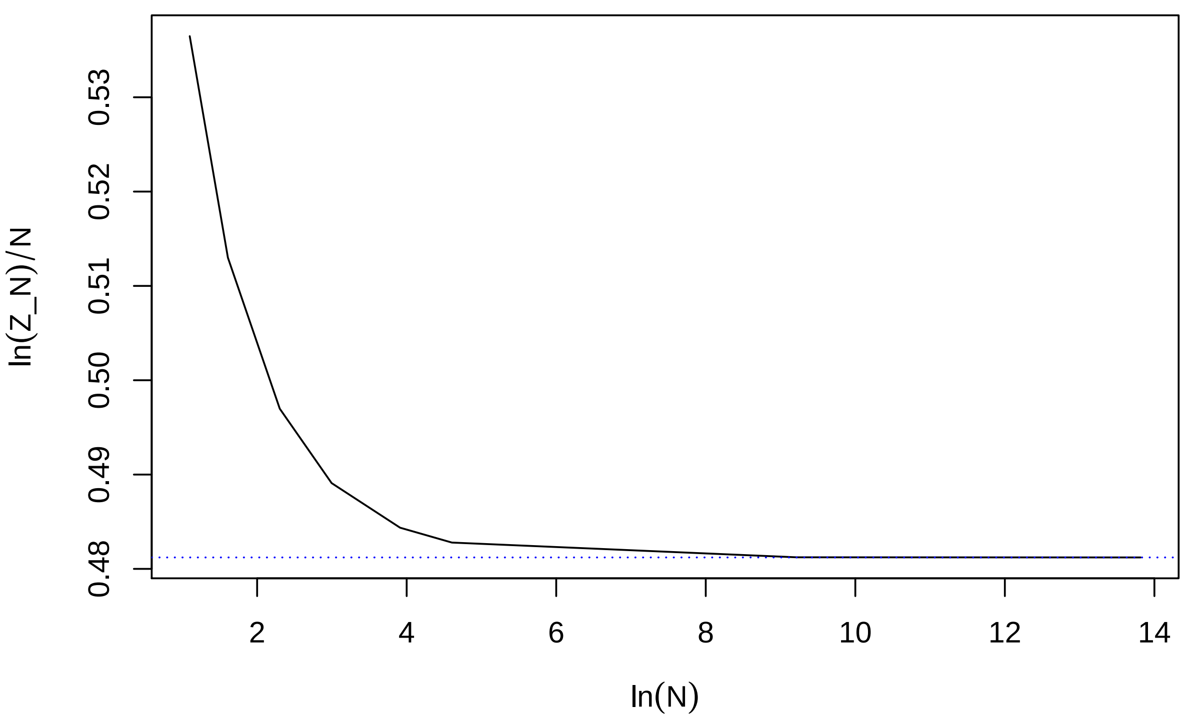

Figure 2: Plot of ln(Z_N)/N (I)

Item(d)

We can also solve this problem analytically. Justify that

$$Z_N = Z_{N-1} + Z_{N-2} \tag{6}$$

with $Z_1 = 2$ and $Z_2 = 3$

{*Solution.*}

According with the sum-product algorithm developed in item(c), we saw that, in order to calculate Z_N, it is sufficient to multiply the power-matrix of the Configuration Matrix $\boldsymbol{M}$ up to a power N by the unit vector $[1, 1]^T$. The sum of coordinates of the resulting vector is Z_N.

Here are some examples:

$$\begin{bmatrix} Z_5 \\ Z_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^5}_{\boldsymbol{M^5}} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 & 5 \\ 5 & 3 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 13 \\ 8 \end{bmatrix}$$

$$\begin{bmatrix} Z_{10} \\ Z_9 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{10}}_{\boldsymbol{M^{10}}} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 89 & 55 \\ 55 & 34 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 144 \\ 89 \end{bmatrix}$$

We can extrapolate this formula in a generic way, as follows:

$$\begin{bmatrix} Z_N \\ Z_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^N}_{\boldsymbol{M^N}} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{7}$$

We can now rewrite (7) as follows:

$$\begin{aligned}
\begin{bmatrix} Z_N \\ Z_{N-1} \end{bmatrix} &= \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^N}_{\boldsymbol{M^N}} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \boldsymbol{M}^N \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}}_{\boldsymbol{M}} \times \underbrace{\boldsymbol{M}^{N-1} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\begin{bmatrix} Z_{N-1} \\ Z_{N-2} \end{bmatrix}} \\
&= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} Z_{N-1} \\ Z_{N-2} \end{bmatrix} \\
&= \begin{bmatrix} Z_{N-1} + Z_{N-2} \\ Z_{N-1} \end{bmatrix}
\end{aligned}$$

$$\implies \begin{bmatrix} Z_N \\ Z_{N-1} \end{bmatrix} = \begin{bmatrix} Z_{N-1} + Z_{N-2} \\ Z_{N-1} \end{bmatrix}$$

... and finally

$$\implies Z_N = Z_{N-1} + Z_{N-2} \tag{8}$$

It is interesting note that, when increasing N by 1, the sequence of Z_N's will follow a *Fibonacci* sequence.

Item(e)

Use the difference equation in (6) to prove that

$$\lim_{N\to\infty}\left(\frac{ln(Z_N)}{N}\right) = ln\left(\frac{1+\sqrt(5)}{2}\right) \tag{9}$$

Compare the above asymptotic result with your numerical experiment in part c).

{*Solution.*}

From the sum-product algorithm implemented in item (c) and (7), we can derive the formula of Z_N in a matricial form as follows:

$$Z_N = \begin{bmatrix}1 & 1\end{bmatrix} \times \begin{bmatrix}Z_{N-1}\\ Z_{N-2}\end{bmatrix} = \begin{bmatrix}1 & 1\end{bmatrix} \times \left(\begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}^{N-1} \times \begin{bmatrix}1\\ 1\end{bmatrix}\right) \tag{10}$$

Using the *Spectral Decomposition* over the configuration matrix $\boldsymbol{M}^{N-1}$, we have the following:

$$\boldsymbol{M}^{N-1} = \boldsymbol{P} \times \boldsymbol{\Lambda}^{N-1} \times \boldsymbol{P}^T \tag{11}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix with eigenvalues $\lambda_1 = \dfrac{1+\sqrt{5}}{2}$ and $\lambda_2 = \dfrac{1-\sqrt{5}}{2}$ and $\boldsymbol{P}$ is an orthogonal matrix of eigenvectors of $\boldsymbol{M}$, such that:

$$\boldsymbol{P} = \begin{bmatrix}\lambda_1 & -\lambda_2^{-1}\\ 1 & 1\end{bmatrix}$$

Substituting (11) in (10) we will obtain a representation of Z_N in terms of eigenvalues and eigenvectors in function of N. For simplicity of notation, we will maintain the eigenvalues represented by their symbols λ_1 and λ_2.

$$\begin{aligned}
Z_N &= \begin{bmatrix}1 & 1\end{bmatrix} \times \left(\begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}^{N-1} \times \begin{bmatrix}1\\ 1\end{bmatrix}\right)\\
&= \begin{bmatrix}1 & 1\end{bmatrix} \times \left[\left(\underbrace{\begin{bmatrix}\lambda_1 & -\lambda_2^{-1}\\ 1 & 1\end{bmatrix} \times \begin{bmatrix}\lambda_1^{N-1} & 0\\ 0 & \lambda_2^{N-1}\end{bmatrix} \times \begin{bmatrix}\lambda_1 & 1\\ -\lambda_2^{-1} & 1\end{bmatrix}}_{\boldsymbol{M}^{N-1}}\right) \times \begin{bmatrix}1\\ 1\end{bmatrix}\right]
\end{aligned}$$

After performing all matrix products we obtain the following polynomial equation on N, and constants c_1 and c_2.

$$Z_N = \lambda_1^N \underbrace{\frac{(\lambda_1+1)^2}{\lambda_1}}_{c_1} + \lambda_2^N \underbrace{\frac{(\lambda_2-1)^2}{\lambda_2^3}}_{c_2} \tag{12}$$

From our hypothesis and using (12), we have:

$$
\begin{aligned}
\frac{\ln Z_N}{N} &= \ln\left(Z_N^{1/N}\right) \\
&= \ln\left[\left(\lambda_1^N \times c_1 + \lambda_2^N \times c_2\right)^{1/N}\right] \\
&= \ln\left[\lambda_1^N\left(1 \times c_1 + \frac{\lambda_2^N}{\lambda_1^N} \times c_2\right)\right]^{1/N} \\
&= \ln\left[\lambda_1\left(1 \times c_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^N \times c_2\right)^{1/N}\right] \\
&= \ln\lambda_1 + \ln\left(1 \times c_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^N \times c_2\right)^{1/N}
\end{aligned}
$$

Applying the limit for $N \to \infty$ in both sides we have:

$$
\begin{aligned}
\lim_{N\to\infty}\frac{\ln Z_N}{N} &= \lim_{N\to\infty}\left(\ln\lambda_1 + \ln\left(1 \times c_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^N \times c_2\right)^{1/N}\right) \\
&= \ln\lambda_1 + \lim_{N\to\infty}\ln\left(1 \times c_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^N \times c_2\right)^{1/N}
\end{aligned}
$$

Note that $\dfrac{\lambda_2}{\lambda_1} < 1$ then the limit on right hand of this equation converges to 0, which implies:

$$
\lim_{N\to\infty}\frac{\ln Z_N}{N} = \ln\lambda_1 = \ln\frac{1+\sqrt{5}}{2} \tag{13}
$$

Comparing with the results on item(c) we have a perfect convergence to the asymptotic result, which is $\ln\dfrac{1+\sqrt{5}}{2} \approx 0.481212$.

Item(f)

Now suppose in order to generate samples uniformly and independently according to $p(\boldsymbol{x})$, we first generate samples uniformly and independently in $\{0,1\}^N$, and then reject the samples that have two 1's next to each other. Is this an efficient method to draw samples according to $p(\boldsymbol{x})$ for large N? why?

{*Solution.*}

The probability associated with this procedure is tied to the probability $p(\boldsymbol{x})$ which is given by:

$$p(\boldsymbol{x}) = \frac{Z_N}{2^N} \tag{14}$$

As we could see in the previous item, the *complexity* of Z_N in terms of order of magnitude for this particular case is proportional to $O(Z_N) \propto O(\lambda_1^N) < O(2^N)$.

For this reason, the probability of encounter sequences with no two 1's next to each other drops dramatically when N becomes large, implying this as **not efficient method to draw samples**.

We can see numerically this behaviour in the graph below, plotting $p(\boldsymbol{x})$ vs. $ln(N)$.

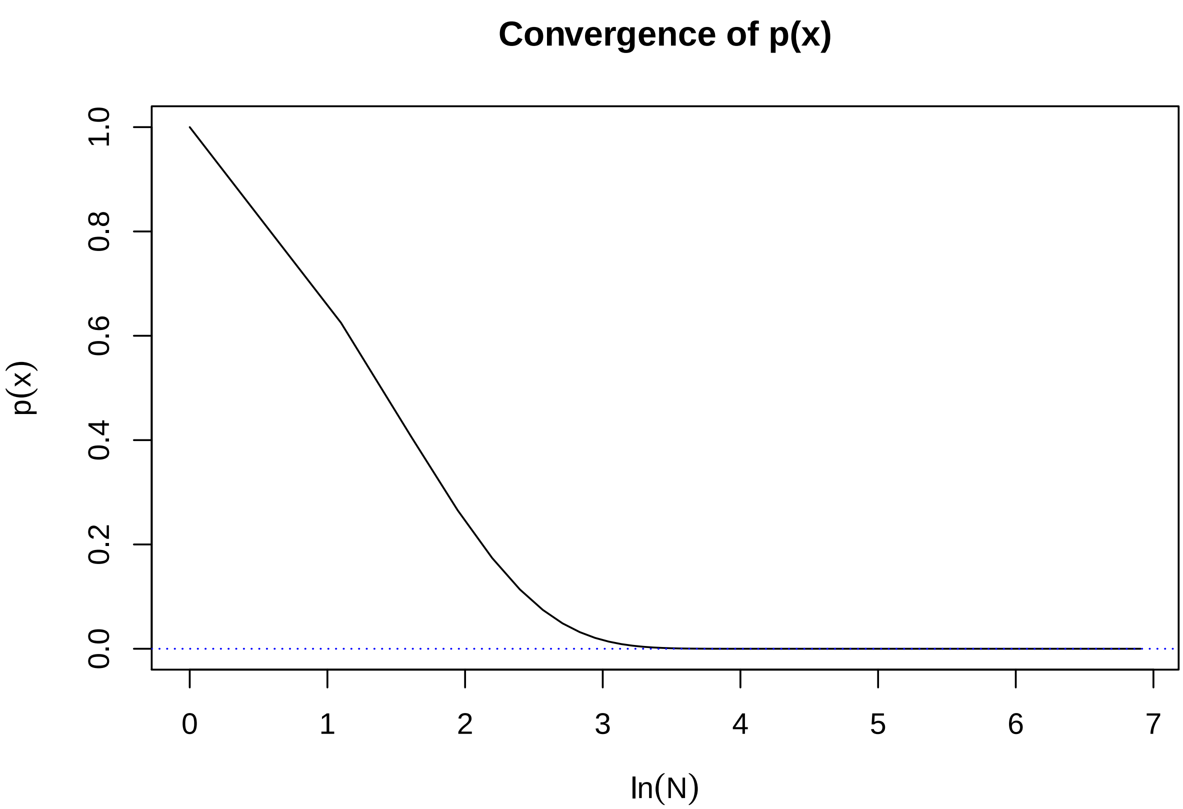

Figure 3: Convergence of P(x)

Appendix - Source Code

```
# Factor Funcion
f <- function (xA, xB) {
  return(ifelse((xA==1 && xB==1),0,1))
}

# Global Normalization Constant ZN - using Messaging passing
ZN <- function (N, normalized = TRUE) {
  # Passing Nodes/Factors
  X <- array(rep(0,2*N), dim = c(2, N))
  F <- array(rep(0,2*(N-1)), dim = c(2, N-1))
  Z <- array(rep(0,N), dim = N)
  Z_ln <- array(rep(0,N), dim = N)

  # Configuration Matrix for present problem
  M <- rbind(c(f(0,0), f(1,0)),
             c(f(0,1), f(1,1)))
  if (N<=0)
    return(NA)
  else
    if(N==1)
      return(2)

  # Initial Normalized Value (default) - returns ln(ZN)
  if (normalized){
    X[,1] <- c(1/2,1/2)
    Z[1] <- 2
    Z_ln[1] <- log(Z[1])
    for (k in 1:(N-1)) {
      F[,k] <- X[,k]              # Message Passing from X to F
      X[,(k+1)] <- M%*%F[,k]      # Calculate the new message for next step
      Z[k+1] <- sum(X[,(k+1)])    # Calculate current Z (sum of X's)
      Z_ln[k+1] <- log(Z[k+1])
      X[,(k+1)] <- X[,(k+1)]/Z[k+1]
    }
    return(sum(Z_ln))
  }
  else {
    if (N>10^3)
      cat("\n Error: Floating Point Overflow - N is too large (try some N <= 1000)\n")
    else {
      X[,1]<- c(1,1)
      Z[1] <- 2
      for (k in 1:(N-1)) {
        F[,k] <- X[,k]
        X[,(k+1)] <- M%*%F[,k]
        Z[k+1] <- sum(X[,(k+1)])    # Calculate current Z (sum of X's)
      }
      return(Z[N])
    }
  }
}
```

```r
# Numbers To be tested
NN <- c(3,5,10,20,50,10^2, 10^4,10^6)

# Calculating the values of Z
Z <- vector()
for (i in 1:length(NN))
  Z <- append(Z,ZN(NN[i])/NN[i])  # Using the 'normalized' version (i.e., default)

# plotting the graph
plot(log(NN), Z, # main = "Convergence of ZN",
     ylab = expression(ln(Z_N)/N),
     xlab = expression(ln(N)),
     type = "l")
abline(h=log((1+sqrt(5))/2), col="blue", lty=3)

# How is the behaviour of p(x)=ZN/s^N
p <- vector()

NN <- seq(from=1, to=1000, by=2)

for (i in 1:length(NN))
  p <- append(p,ZN(NN[i], normalized = FALSE)/2^NN[i])

plot(log(NN),p,main = "Convergence of p(x)",
     ylab = expression(p(x)),
     xlab = expression(ln(N)),
     type = "l")
abline(h=0, col="blue", lty=3)
```