CV201, HW #2

Oren Freifeld and Meitar Ronen
The Department of Computer Science, Ben-Gurion University

Release Date – Partial: 29/11/2019. Full Version Release: 6/12/2019
Submission Deadline: 30/12/2019, 20:00

Abstract

Please make sure you have carefully read the general instructions in the course website at https://www.cs.bgu.ac.il/~cv201/HW_Instructions. These instructions address, among other things, what to do or not do with figures, Jupyter Notebooks, *etc.*, as well as what compression files are legit or not.

As for this assignment:

- § 1 focuses on the dependency structure in some MRF examples.
- § 2 centers on the Ising model.
- § 3 is dedicated to building an exact sampler for the Ising model, on an 8×8 2D regular lattice, using dynamic programming. At each of three temperatures, ten random samples are displayed and two empirical expectations are computed. The main Computer Exercise is Computer Exercise 7. Those before it are shorter, easier, and are used as building blocks or debugging tools for that Exercise. Computer Exercise 8 boils down to using the implementation from Computer Exercise 7. Finally, in Problem 14 you are asked to analyze the results from Computer Exercise 8.

Note well: results from this HW assignment will be reused in the next one.

When you draw graphs (which is required as part of the assignment), you are free to use any appropriate software tool to do that (*e.g.*, PowerPoint, LaTeX + TikZ, Inkscape, *etc.*). Our recommendation, however, for the most painless way to do it well, is that you use Or Dinari's fork of **daft**: https://github.com/dinarior/daft.

Version Log

- 1.06, 21/12/2019. In Computer Exercise 7: removed an extra "("; removed a redundant "If".
- 1.05, 8/12/2019. In § 3, replaced all instances of Z with Z_{Temp} to highlight the fact that the normalizer depends on the temperature and for consistency with the notation in § 2.

- 1.04, 6/12/2019. The issue referred to in Problem 13 is related to Python 2, but no longer appears in Python 3. So feel free to ignore this problem. Also, an additional and important remark has been added after Computer Exercise 7.
- 1.03, 6/12/2019. Release § 3. Added a comment to Problem 5.
- 1.02, 3/12/2019. Released § 2.
- 1.01, 1/12/2019. Problem 1. Switched to lowercase letters in the distributions. Also switched from N to n (for a different reason).
- 1.00, 29/11/2019. Initial release: only § 1.

Contents

1 Some General Questions about MRFs

Problem 1 *Let* $(X_i)_{i=1}^n$ *be the following* i.i.d.[1] *sequence of Bernoulli random variables:*

$$X_i = \begin{cases} 1 & \text{with probability } \theta \\ 0 & \text{with probability } 1-\theta \end{cases} \tag{1}$$

where $\theta \in [0,1]$ *is some known number. Define a new sequence of random variables, denoted by* $(Y_i)_{i=1}^n$*, via*

$$Y_i = \sum_{j=1}^{i} X_j \quad i \in \{1,2,\ldots,n\}\,. \tag{2}$$

Part (i) Find the joint pmf of X_1 *and* X_3*.*

Part (ii) Find the pmf of Y_1*.*

Part (iii) Find the pmf of Y_2*.*

Part (iv) Find the pmf of Y_3*.*

Part (v) Find $p(y_1|x_1)$*.*

[1] Reminder: *i.i.d.* stands for independent and identically distributed.

Part (vi) *Find* $p(y_2|x_1, x_2)$.

Part (vii) *Find* $p(y_2|y_1, y_2)$.

Part (viii) *Find* $p(y_3|x_1, x_2)$.

Part (ix) *Find* $p(y_3|x_1, x_2, y_1)$.

Part (x) *Find* $p(y_3|x_1, x_2, y_2)$.

Part (xi) *Find* $p(y_3|x_1, x_2, x_3, y_2)$.

Part (xii) *Find* $p(y_3|y_1, y_2, y_3)$.

Part (xiii) *Is* $(X_i)_{i=1}^n$ *a Markov Chain? Justify your answer.*

Part (xiv) *Is* $(Y_i)_{i=1}^n$ *a Markov Chain? Justify your answer.* ◊

In the problems below, you are asked to write down the set of cliques in a graph. The set of cliques is usually not unique since we can have situations where singletons can be absorbed into (say) a pairwise clique or where a pairwise clique can be absorbed into a triplet, and so on. Moreover, while each Gibbs distribution (*i.e.*, a product of clique functions) determines the graph G associated with it, in the other direction inspecting G might not reveal all possible information about the Gibbs distribution that created it.

Example 1 *Both*

$$p(x_1, x_2, x_3, x_4) = F_{12}(x_1, x_2)F_{23}(x_2, x_3)F_{13}(x_1, x_3)F_{24}(x_2, x_4) \tag{3}$$

and

$$p(x_1, x_3, x_3, x_4) = F_{123}(x_1, x_2, x_3)F_{24}(x_2, x_4) \tag{4}$$

imply the same graph G. Note that, by taking

$$F_{123}(x_1, x_2, x_3) = F_{12}(x_1, x_2)F_{23}(x_2, x_3)F_{13}(x_1, x_3)\,,$$

the first p is a particular case of the second one. Given that G, however, we can only say that the p has the second form, but we are unable to determine if it has the first form. ◊

The conclusion is that the mapping of (structures of) Gibbs distributions to their associated undirected Graphs is not injective. When you are asked to write down the set of cliques by reading it from p, pick the set that preserves the most information about the structure of p. In the case of the first p above, that would be $\mathcal{C} = \{\{1, 2\}, \{2, 3\}, \{1, 3\}, \{2, 4\}\}$ while for the second p it would be $\mathcal{C} = \{\{1, 2, 3\}, \{2, 4\}\}$.

Problem 2 *Draw G, the graph associated with the distributions from the example above.* ◊

That said, *if both a clique and its sub-clique appear explicitly in p*, there is no point in keeping the sub-clique. For example, if

$$p(x_1, x_2) = F_{12}(x_1, x_2)F_1(x_1) \tag{5}$$

then it is enough to say that $\mathcal{C} = \{\{1, 2\}\}$ as there is no advantage in stating that $\mathcal{C} = \{\{1, 2\}, \{1\}\}$.

Problem 3 *Let* $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T$ *denote a (continuous) random vector with a probability density function (pdf) of the form*

$$\begin{aligned} p(\boldsymbol{x}) &= p(x_1, x_2, x_3, x_4, x_5, x_6) \\ &\propto F_{13}(x_1, x_3) F_{24}(x_2, x_4) F_{34}(x_3, x_4) F_{35}(x_3, x_5) F_{26}(x_2, x_6) F_{56}(x_5, x_6) \end{aligned} \tag{6}$$

such that $p(\boldsymbol{x}) > 0 \; \forall \boldsymbol{x} \in \mathbb{R}^6$. *It follows that* p *is a Markov Random Field (MRF) w.r.t. a certain graph,* G_{123456}.

Part (i) *Draw* G_{123456}.

Part (ii) *Let* $\mathcal{C}$ *denote the set of cliques in* G_{123456}. *Find* $\mathcal{C}$.

Part (iii) *Draw the following graphs:*
G_{23456}, *the graph of the MRF associated with*

$$p(x_2, x_3, x_4, x_5, x_6) = \int p(x_1, x_2, x_3, x_4, x_5, x_6) \, dx_1 \, ; \tag{7}$$

G_{3456}, *the graph of the MRF associated with*

$$p(x_3, x_4, x_5, x_6) = \int p(x_2, x_3, x_4, x_5, x_6) \, dx_2 \, ; \tag{8}$$

G_{456}, *the graph of the MRF associated with*

$$p(x_4, x_5, x_6) = \int p(x_3, x_4, x_5, x_6) \, dx_3 \, ; \tag{9}$$

G_{56}, *the graph of the MRF associated with*

$$p(x_5, x_6) = \int p(x_4, x_5, x_6) \, dx_4 \, ; \tag{10}$$

G_6, *the graph of the MRF associated with*

$$p(x_6) = \int p(x_5, x_6) \, dx_5 \, . \tag{11}$$

Part (iv) *Answer Yes or No to each of the following three questions:*

- *(a) Are* x_1 *and* x_6 *conditionally independent given* x_5*?*
- *(b) Are* x_1 *and* x_3 *conditionally independent given* x_2 *and* x_5*?*
- *(c) Are* x_1 *and* x_6 *conditionally independent given* x_4 *and* x_5*?*

Part (v) *Now suppose further that* $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T$ *is (a multi-variate) Gaussian with a* 6×6 *precision matrix* $\boldsymbol{Q}$. *Based on the form of* $p(x_1, x_2, x_3, x_4, x_5, x_6)$ *above, write the matrix* $\boldsymbol{Q}$ *as follows. For each entry of the matrix, write 0 if this entry is guaranteed to be zero; otherwise, leave the entry empty.* $\Diamond$

Problem 4 *Let* $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^T$ *denote a (continuous) random vector with a probability density function (pdf) of the form*

$$p(\boldsymbol{x}) = p(x_1, x_2, x_3, x_4, x_5) \propto F_{123}(x_1, x_2, x_3) F_{24}(x_2, x_4) F_{34}(x_3, x_4) F_{35}(x_3, x_5) \tag{12}$$

such that $p(\boldsymbol{x}) > 0 \; \forall \boldsymbol{x} \in \mathbb{R}^5$. *It follows that* p *is a Markov Random Field (MRF) w.r.t. a certain graph,* G_{12345}.

Part (i) *Draw* G_{12345}.

Part (ii) *Let* $\mathcal{C}$ *denote the set of cliques in* G_{12345}. *Find* $\mathcal{C}$.

Part (iii) *Draw the following four graphs:*
G_{2345}, *the graph of the MRF associated with*

$$p(x_2, x_3, x_4, x_5) = \int p(x_1, x_2, x_3, x_4, x_5) \, dx_1 \, ; \tag{13}$$

G_{345}, *the graph of the MRF associated with*

$$p(x_3, x_4, x_5) = \int p(x_2, x_3, x_4, x_5) \, dx_2 \, ; \tag{14}$$

G_{45}, *the graph of the MRF associated with*

$$p(x_4, x_5) = \int p(x_3, x_4, x_5) \, dx_3 \, ; \tag{15}$$

G_5, *the graph of the MRF associated with*

$$p(x_5) = \int p(x_4, x_5) \, dx_4 \, . \tag{16}$$

Part (iv) *Answer Yes or No to each of the following three questions: Are* x_1 *and* x_4 *conditionally independent given* x_2*? Are* x_1 *and* x_4 *conditionally independent given* x_2 *and* x_3*? Are* x_1 *and* x_5 *conditionally independent given* x_4*?*

Part (v) *Now suppose further that* $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^T$ *is (a multivariate) Gaussian with a* 5×5 *precision matrix* $\boldsymbol{Q}$. *Based on the form of* $p(x_1, x_2, x_3, x_4, x_5)$ *above, write the matrix* $\boldsymbol{Q}$ *as follows. For each entry of the matrix, write 0 if this entry is guaranteed to be zero; otherwise, leave the entry empty.* $\Diamond$

Problem 5 *Let* $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ *denote a (continuous) random vector with a probability density function (pdf) of the form*

$$p(\boldsymbol{x}) = p(x_1, x_2, x_3) \propto F_{12}(x_1, x_2) F_{23}(x_2, x_3) \tag{17}$$

such that $p(\boldsymbol{x}) > 0 \; \forall \boldsymbol{x} \in \mathbb{R}^3$. *It follows that* p *is a Markov Random Field (MRF) w.r.t. a certain graph,* $G_{\boldsymbol{x}_{123}}$. *For* $i = 1, 2, 3$, *let* $y_i = x_i + n_i$, *where* $n_i \overset{iid}{\sim} \mathcal{N}(0, 1)$. *Moreover,* $(n_i)_{i=1}^3$ *are also assumed to be independent of* $(x_i)_{i=1}^3$.

Part (i) *Draw $G_{\boldsymbol{x}_{123}}$. Let $\mathcal{C}_{\boldsymbol{x}_{123}}$ denote the set of cliques in $G_{\boldsymbol{x}_{123}}$. Find $\mathcal{C}$.*

Part (ii) *Write $p(y_1, y_2, y_3|x_1, x_2, x_3)$ in terms of $\{G(x_i, y_i)\}_{i=1}^3$, where $G(x_i, y_i) \triangleq \mathcal{N}(y_i; x_i, 1)$.*

Part (iii) *Write $p(x_1, x_2, x_3, y_1, y_2, y_3)$.*

Part (iv) *$p(x_1, x_2, x_3, y_1, y_2, y_3)$ is an MRF w.r.t. a certain graph, $G_{\boldsymbol{x}_{123},\boldsymbol{y}_{123}}$. Draw $G_{\boldsymbol{x}_{123},\boldsymbol{y}_{123}}$.*

Part (v) *Draw $G_{\boldsymbol{y}_{123}}$, the graph of the MRF associated with*

$$p(y_1, y_2, y_3) = \int p(x_1, x_2, x_3, y_1, y_2, y_3)\, dx_1 dx_2 dx_3\,; \tag{18}$$

Part (vi) *Answer Yes or No to each of the following questions: Are x_1 and x_2 conditionally independent given x_3? Are x_1 and x_3 conditionally independent given y_1 and x_2? Are x_1 and y_3 conditionally independent given x_2?*

Part (vii) *Now suppose further that $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ is (a multi-variate) Gaussian. It follows that $\begin{bmatrix} x_1 & x_2 & x_3 & y_1 & y_2 & y_3 \end{bmatrix}^T$ is also (a multi-variate) Gaussian with a 6×6 precision matrix $\boldsymbol{Q}$. Based on the form of $p(x_1, x_2, x_3, y_1, y_2, y_3)$ above, write the matrix $\boldsymbol{Q}$ as follows. For each entry of the matrix, write 0 if this entry is guaranteed to be zero; otherwise, leave the entry empty.* ♢

Problem 6 *Let $S = \{1, \ldots, N\}$. Let $X = (X_s)_{s\in S}$ denote a collection of binary random variables such that $x_s \in \{-1, 1\}$ for every $s \in S$. Let $p(x)$ denote the pmf associated with X. We will think of X as latent variables. Let $y = (y_s)_{s\in S}$ denote the observations, where the following observation model is assumed:*

$$\mathbb{R} \ni y_s = x_s + n_s \quad s \in S \quad \mathbb{R} \ni n_s \overset{iid}{\sim} \mathcal{N}(0, \sigma^2).$$

Let $y = (y_s)_{s\in S}$ denote the collection of the observed values.

Part (i) *Write the likelihood function, $x \mapsto p(y|x)$.*

Part (ii) *Find the maximum-likelihood estimator,*

$$\widehat{x} = \arg\max_x p(y|x)\,. \tag{19}$$

To clarify, the maximization is done over all N binary tuples, $x = (x_s)_{s\in S}$ such that $x_s \in \{-1, 1\}$. Particularly, note that the $(x_s)_{s\in S}$ are discrete. *Thus, even though this is an optimization problem, you should turn off the part of your brain that wants to compute derivatives and set them to zero.*

Part (iii) *Explain how different choices of the prior, $p(x)$, affect, if at all, the maximum-likelihood estimator.* ♢

2 The Ising Model

The Ising model on an $n \times n$ lattice, at temperature Temp > 0, is

$$p(x) = \frac{1}{Z_{\text{Temp}}} \exp\left(\frac{1}{\text{Temp}} \sum_{s \sim t} x_s x_t\right) \tag{20}$$

with

$$Z_{\text{Temp}} = \sum_x \exp\left(\frac{1}{\text{Temp}} \sum_{s \sim t} x_s x_t\right), \tag{21}$$

$x_s \in \{-1, 1\}$ for every $s \in S$,

$$S = \{(i,j)\}_{1 \le i \le n, 1 \le j \le n} \tag{22}$$

is the $n \times n$ lattice, and $s \sim t$ means s and t form a pair of neighboring sites in S, as defined by a rectangular nearest-neighbor system (so that each interior site has four neighbors, to its North, South, East, and West).

Remark 1 *Note that the normalizing constant, Z_{Temp}, depends on* Temp. ♢

Remark 2 *To clarify, in the summation above each unordered pair of sites appears only once (*i.e.*, no double counting). For example, in a 2×2 lattice, where the sites are given by $S = \{(1,1),(2,1),(2,1),(2,2)\}$, this would mean 4 (and not 8) summands. For example, if* Temp $= 1$*, we get:*

$$p(x) = \frac{1}{Z_{\text{Temp}}} \exp\left(x_{(1,1)}x_{(1,2)} + x_{(1,1)}x_{(2,1)} + x_{(2,1)}x_{(2,2)} + x_{(1,2)}x_{(2,2)}\right) \tag{23}$$

and

$$\begin{aligned} Z_{\text{Temp}} &= \sum_x \exp\left(x_{(1,1)}x_{(1,2)} + x_{(1,1)}x_{(2,1)} + x_{(2,1)}x_{(2,2)} + x_{(1,2)}x_{(2,2)}\right) \\ &= \sum_{x_{(1,1)} \in \mathcal{R}} \sum_{x_{(1,2)} \in \mathcal{R}} \sum_{x_{(2,1)} \in \mathcal{R}} \sum_{x_{(2,2)} \in \mathcal{R}} \exp\left(x_{(1,1)}x_{(1,2)} + x_{(1,1)}x_{(2,1)} + x_{(2,1)}x_{(2,2)} + x_{(1,2)}x_{(2,2)}\right). \end{aligned} \tag{24}$$

where $\mathcal{R} = \{-1, 1\}$. ♢

Remark 3 *In Python, when evaluating* 1/Temp*, the variable* Temp *must be represented as float, not an integer (*e.g.*, 1/2 is zero in Python).* ***Update:*** *This used be an issue in Python 2 but is no longer the case in Python 3.* ♢

Problem 7 *Find $E(x) = \sum_x xp(x)$ where p is given by the Ising model. Does your answer depend on* Temp*?* ♢

Problem 8 *Consider two configurations in 2×2 lattice.*

$$x = \begin{bmatrix} -1 & -1 \\ +1 & +1 \end{bmatrix} \quad \textit{and} \quad x = \begin{bmatrix} -1 & -1 \\ -1 & +1 \end{bmatrix} \tag{25}$$

Part (i) Which configuration is more likely according to the Ising model if Temp $= 1$*?*

Part (ii) Which configuration is more likely according to the Ising model if $\text{Temp} = 1000$*?* ♢

Problem 9 *In the case of a 3×3 lattice, with* $\beta = 42$*, which configuration is more likely according to the Ising model,*

$$x = \begin{bmatrix} -1 & -1 & -1 \\ +1 & +1 & +1 \\ +1 & -1 & +1 \end{bmatrix} \quad or \quad x = \begin{bmatrix} -1 & +1 & -1 \\ -1 & +1 & -1 \\ -1 & +1 & -1 \end{bmatrix} \; ? \tag{26}$$

♢

Problem 10 *Explain the general effect* Temp *has on the model.* ♢

Problem 11 *It is not hard to see that, according to the Ising model,*

$$p(x_s = 1 \, \forall s \in S) = p(x_s = -1 \, \forall s \in S) \, . \tag{27}$$

Consider a modified version of the model:

$$p_{modified}(x) = \frac{1}{Z_{\text{Temp}}} \exp\left(\left[\frac{1}{\text{Temp}} \sum_{s \sim t} x_s x_t \right] + \sum_s \phi(x_s) \right) \tag{28}$$

where

$$Z_{\text{Temp}} = \sum_x \exp\left(\left[\frac{1}{\text{Temp}} \sum_{s \sim t} x_s x_t \right] + \sum_s \phi(x_s) \right) \tag{29}$$

and $\sum_s$ *means summing over all the sites* $s \in S$ *(while* $\sum_x$ *means summing over the possible values of* x*). Suggest a form for the unary term* ϕ *such that we will have* $p_{mofidied}(x_s = 1 \, \forall s \in S) > p_{modified}(x_s = -1 \, \forall s \in S)$*.* ♢

Problem 12 *In the case of an 8×8 lattice, consider the case where* $\text{Temp} \to \infty$*. What will be the resulting distribution over the* 2^{64} *possible configurations?* ♢

3 Dynamic Programming in the Ising Model

3.1 The Clique Functions in the Ising Model

Computer Exercise 1 *Implement a function,* G*, whose input is*

1. *a 1D numpy array, denoted here by* r *and in your code by* `row_s`*;*
2. *a scalar* Temp*,*

and whose output is the scalar

$$\exp\left(\frac{1}{\text{Temp}} \sum_{i=1}^{\text{length}(r)-1} r_i r_{i+1} \right) \, . \tag{30}$$

It should be a single line of code, and you must not use a loop (nor a list/dict comprehension). ♢

Computer Exercise 2 *Implement a function, F, whose input is*

1. *two 1D numpy arrays of the same length, denoted here by r and $\tilde{r}$ and in your code by* `row_s` *and* `row_t`*;*
2. *a scalar* Temp*,*

and whose output is the scalar

$$\exp\left(\frac{1}{\text{Temp}}\sum_{i=1}^{\text{length}(r)} r_i\tilde{r}_i\right). \tag{31}$$

Again, it should be a single line of code, you must not use a loop (nor a list/dict comprehension). ♢

3.2 Brute Force on Small Lattices

Computer Exercise 3 *In the case of a 2×2 lattice,* i.e.

$$S = \{(i,j)\}_{1\le i\le 2, 1\le j\le 2}\,, \tag{32}$$

compute Z_{Temp} (for three different values of Temp *where* Temp $\in\{1, 1.5, 2\}$*) using brute force (use 4 nested For loops, one for each of the x_s's, the looping is done over the values that x_s can take: $\{-1,1\}$). To help you debug: For* Temp $= 1$*, your result should be $Z_{\text{Temp}} = 121.23\ldots$.* ♢

Computer Exercise 4 *In the case of a 3×3 lattice,* i.e.

$$S = \{(i,j)\}_{1\le i\le 3, 1\le j\le 3}\,, \tag{33}$$

compute Z_{Temp} (for three different values of Temp *where* Temp $\in\{1, 1.5, 2\}$*) using brute force (use 9 nested For loops, one for each of the x_s's, the looping is done over the values that x_s can take: $\{-1,1\}$). To help you debug: For* Temp $= 1$*, your result should be $Z_{\text{Temp}} = 10^5\cdot 3.65\ldots$.* ♢

Obviously, 9 For loops is pretty ugly. We will see below we can use fewer loops, even if we remain in a brute-force approach.

The most computationally-efficient approach, to achieve the main goal of this HW assignment (exact sampling from the Ising model), is to use a raster or boustrophedonic[1] site-visitation schedule and then compute conditional probabilities, site-by-site, in a backward ordering. The number of computations is $O(64\cdot 2^9) = O(2^{15})$. **However, this is NOT what you are asked to do. Instead, you are requested to do something much simpler (and sub-optimal):** the programming burden becomes substantially easier if the 8×8 lattice is represented as a Markov chain with 8 sites (where each of them encodes an entire row from the original 8×8 lattice) and 8 corresponding variables, $y_1,\ldots,y_8$. Although computational complexity becomes $O(8\cdot 2^{16}) = O(2^{19})$, the programming complexity is vastly reduced – so we will go with that. Each site variable, y_s (where $s = 1,\ldots,8$) has 2^8 states, $y_s\in\{0,1,\ldots,255\}$, corresponding to the 2^8 possible configurations of row s in the Ising lattice. The

[1] Namely, from right to left and from left to right in alternate lines. The word came from Greek and refers to how an ox turns when plowing a field.

correspondence is thus $y_s \leftrightarrow (x_{(s,1)}, \ldots, x_{(s,8)})$. A convenient mapping between y_s and $(x_{(s,i)})_{i=1}^8$ is to use the 8-bit binary representation of y_s, together with the conversion of 0's to -1's.

Example 2 *Suppose $y_s = 136$. The 8-bit binary representation of 136 is '10001000'. Therefore, the corresponding row (*i.e.*, $(x_{(s,i)})_{i=1}^8$) is [1 -1 -1 -1 1 1 -1 -1 -1].* ♢

The following helper function computes the $y_s \to (x_{(s,i)})_{i=1}^8$ mapping.

```
import numpy as np
def y2row(y,width=8):
    """
    y: an integer in (0,...,(2**width)-1)
    """
    if not 0<=y<=(2**width)-1:
        raise ValueError(y)
    my_str=np.binary_repr(y,width=width)
    # my_list = map(int,my_str) # Python 2
    my_list = list(map(int,my_str)) # Python 3
    my_array = np.asarray(my_list)
    my_array[my_array==0]=-1
    row=my_array
    return row
```

Remark 4 *Note that in this HW assignment, the y's merely stand for groups of x's, used as a mechanism for simplifying the programming. These y's should not be confused with "observations". In this assignment we use only the prior, $p(x)$, and there are no noisy observations;* i.e.*, this HW assignment does not involve a posterior distribution. However, if we let z denote hypothetical observations, the sampling mechanism from this HW assignment can be easily adjusted to sampling from the posterior, $p(x|z)$, in the case that each data point is connected in the graph to a single site in the original graph of $p(x)$;* i.e.*, if the likelihood, $p(z|x)$, factorizes as $p(z|x) = \prod_{s\in S} p(z_s|x_s)$. As we saw in class, the only thing that will have to change is the functional form of the clique functions. But again, to clarify, this is not something you are asked to do in this HW assignment.* ♢

Computer Exercise 5 *In the case of a 2×2 lattice, converted to a chain of length 2, compute Z_{Temp} (for three different values of* Temp *where* Temp $\in \{1, 1.5, 2\}$*) using brute force (use 2 nested For loops, one for each of the y_s's, the looping is done over the values that y_s can take: $\{0, 1, 2, 3\}$). In effect,*

$$p(y) = p(y_1, y_2) = \frac{1}{Z_{\text{Temp}}} G(y_1)G(y_2)F(y_1, y_2) \tag{34}$$

and

$$Z_{\text{Temp}} = \sum_y G(y_1)G(y_2)F(y_1, y_2) = \sum_{y_1}\sum_{y_2} G(y_1)G(y_2)F(y_1, y_2) \tag{35}$$

where the singleton function

$$G(y_s) = \exp\left(\frac{1}{\text{Temp}} x_{(s,1)}x_{(s,2)}\right), \qquad s = 1, 2 \tag{36}$$

representing, for each of the two rows, s, the intra-row pair clique from the original graph, and a pair clique function

$$F(y_1, y_2) = \exp\left(\frac{1}{\text{Temp}} \sum_{i=1}^{2} x_{(1,i)} x_{(2,i)}\right) \tag{37}$$

representing the product of the two inter-row pair cliques from the original graph.

In your implementation, use the provided helper function ***y2row*** *(in the second argument, pass width=2 since here there are only two x_s's in each row) and your implemented G and F from the previous exercises.*

To help you debug: For $\text{Temp} = 1$*, your result should be* $Z_{\text{Temp}} = 121.23\ldots$ *(which is of course the same result from before).* ♢

Computer Exercise 6 *In the case of a* 3×3 *lattice, converted to a chain of length 3, compute* Z_{Temp} *(for three different values of* Temp *where* $\text{Temp} \in \{1, 1.5, 2\}$*) using brute force (use 3 nested For loops, one for each of the y_s's, the looping is done over the values that y_s can take: $\{0, 1, \ldots, 7\}$). In effect,*

$$p(y) = p(y_1, y_2, y_3) = \frac{1}{Z_{\text{Temp}}} G(y_1)G(y_2)G(y_3)F(y_1, y_2)F(y_2, y_3) \tag{38}$$

and

$$\begin{aligned} Z_{\text{Temp}} &= \sum_{y} G(y_1)G(y_2)G(y_3)F(y_1, y_2)F(y_2, y_3) \\ &= \sum_{y_1}\sum_{y_2}\sum_{y_3} G(y_1)G(y_2)G(y_3)F(y_1, y_2)F(y_2, y_3) \end{aligned} \tag{39}$$

where the singleton function

$$G(y_s) = \exp\left(\frac{1}{\text{Temp}} \sum_{i=1}^{2} x_{(s,i)} x_{(s,i+1)}\right), \qquad s = 1, 2, 3 \tag{40}$$

representing, for each of the three rows, s, the product of the two intra-row pair cliques from the original graph, and pair clique functions

$$F(y_s, y_{s+1}) = \exp\left(\frac{1}{\text{Temp}} \sum_{i=1}^{3} x_{(s,i)} x_{(s+1,i)}\right) \qquad s = 1, 2 \tag{41}$$

representing, for each of the two pairs of rows, $(s, s+1)$, the product of the three inter-row pair cliques from the original graph.

In your implementation, use the provided helper function ***y2row*** *(in the second argument, pass width=3 since here there are three x_s's in each row) and your implemented G and F from the previous exercises.*

To help you debug: For $\text{Temp} = 1$*, your result should be* $Z_{\text{Temp}} = 10^5 \cdot 3.65\ldots$ *(which is of course the same result from before).* ♢

3.3 Dynamic Programming on an 8×8 Lattice

The following Computer Exercise is the main task in this HW assignment.

Computer Exercise 7 *Using dynamic programming and an* 8×8 *lattice, build an exact sampler for each of the three temperatures,* $\text{Temp} \in \{1, 1.5, 2\}$. ***The details for how to do it appear at the notes at the end of this document.*** *Use the samplers to generate ten independent samples (to clarify, each such sample is an entire* 8×8 *image) at each of the three temperatures, and display these as thirty images all in a single figure (three rows, one for each temperature, and ten columns of* 8×8 *black-and-white images; the subplot command should be useful here). When displaying your images using plt.imshow (in Python), you should use the Interpolation="None" option (*i.e.*, passing the string "None", not to be confused with Interpolation=None, which passes the Python built-in None).*
If your result for $\text{Temp} = 2$ *looks completely random and does not make sense, perhaps you should first solve Problem 13.* ◊

Remark 5 *In the problem above, when T=1, some of you may be puzzled to see that most of the samples (where each sample is an entire 8-by-8 image) are "all blue" (or "all black", depending on the colormap you are using) and you will be wondering how come this is not symmetric. In effect, you will feel that it makes sense that most of the images are unicolor, but you would have expected that about half of them will be "all blue" ("-1") and about half of them will be "all red" ("+1"). Well, this is not how imshow works by default. Here is what is going on: if you hover with the mouse over the pixels in these unicolor images, you will see (at the bottom of the figure) that, in fact, some of them are all "-1" and some of them are all "+1", even though both cases are "all blue". This is because by default, imshow(img) scales the display such that the blue stands for img.min(), and red stands for img.max(). If var(img)=0 (*i.e.*, all pixels share the same value), then img.min()=img.max() so only a single color (blue) is used. This is regardless whether all the pixels are -1 or all the pixels are +1. One way to overcome this visualization issue is using the vmin and vmax optional parameters: imshow(...,vmin=-1, vmax=+1). Read imshow's documentation about it. That said, later in HW3, when you show images corrupted by real-valued noise, you should avoid this [-1,1] range because it will trim the values which may be outside that range.* ◊

Problem 13 ***Update: the issue referred to at this problem pertains to the fact that, in Python 2,*** $1/2 = 0$. ***This is no longer the case in Python 3. Thus, you can ignore this problem and leave it unanswered.*** *A reckless student, not you of course, encountered a weired problem when working on Computer Exercise 7. While the resulting samples for* $\text{Temp} = 1$ *and* $\text{Temp} = 1.5$ *look as one might expect, for* $\text{Temp} = 2$ *the sample that student obtained looked completely random with no structure whatsoever. Explain what, probably, the programming bug the student had was, and from which distribution over binary images that student really sampled from (instead of the Ising model).* ◊

Computer Exercise 8 *Using the three samplers you implemented above, at each of the three temperatures, draw 10,000 samples,* $x(1), \ldots, x(10000)$ *(each*

such sample is an 8×8 *binary image) and compute two empirical expectations:*

$$\widehat{E}_{\text{Temp}}(X_{(1,1)}X_{(2,2)}) \triangleq \frac{1}{10000}\sum_{n=1}^{10000} x_{(1,1)}(n)x_{(2,2)}(n) \tag{42}$$

$$\widehat{E}_{\text{Temp}}(X_{(1,1)}X_{(8,8)}) \triangleq \frac{1}{10000}\sum_{n=1}^{10000} x_{(1,1)}(n)x_{(8,8)}(n) \tag{43}$$

where $\text{Temp} = 1, 1.5,$ *and* 2 *and where* $x_{(i,j)}(n)$ *is the value at the* (i,j)*-th lattice site of sample* n*.* ◊

Problem 14 *Using the results of the Computer Exercise above, explain the relative values of* $\widehat{E}$*, in terms of the spatial distance of the lattice sites and in terms of the temperature.* ◊

3.4 Detailed Notes and How to Debug

1. You should do the sampling in terms of the y_s's, not the x_s's, since it is considerably simpler.

2. In this representation, where we use an 8-length Markov chain (with each of its variables taking values in $\{0, 1, \ldots, 255\}$) to represent a binary MRF defined over an 8×8 lattice, there are two kinds of clique functions:

   (a) Singletons of the form

   $$G(y_s) = \exp\left(\frac{1}{\text{Temp}}\sum_{i=1}^{7} x_{(s,i)}x_{(s,i+1)}\right), \qquad s = 1, 2, \ldots, 8 \tag{44}$$

   representing, for each of the eight rows, s, the product of the seven intra-row pair cliques from the original graph;

   (b) pair cliques of the form

   $$F(y_s, y_{s+1}) = \exp\left(\frac{1}{\text{Temp}}\sum_{i=1}^{8} x_{(s,i)}x_{(s+1,i)}\right) \qquad s = 1, 2, \ldots, 7 \tag{45}$$

   representing, for each of the seven pairs of rows, $(s, s+1)$, the product of the eight inter-row pair cliques from the original graph.

   Thus,

   $$p(y_1, \ldots, y_8) \propto \left(\prod_{s=1}^{8} G(y_s)\right)\left(\prod_{s=1}^{7} F(y_s, y_{s+1})\right). \tag{46}$$

   By now, you should already have these two types of functions implemented.

The dynamic programming iterations, adopting the site-visitation schedule $1, 2, \ldots, 8$, is then

$$T_1(y_2) = \sum_{y_1=0}^{255} G(y_1)F(y_1, y_2) \qquad y_2 \in \{0, 1, \ldots, 255\} \tag{47}$$

$$T_k(y_{k+1}) = \sum_{y_k=0}^{255} T_{k-1}(y_k)G(y_k)F(y_k, y_{k+1}) \qquad 2 \le k \le 7 \quad y_{k+1} \in \{0, 1, \ldots, 255\} \tag{48}$$

$$\mathbb{R} \ni T_8 = \sum_{y_8=0}^{255} T_7(y_8)G(y_8) \quad (= Z_{\text{Temp}}) \tag{49}$$

Working backwards:

$$p_8(y_8) = \frac{T_7(y_8)G(y_8)}{Z_{\text{Temp}}} \qquad y_8 \in \{0, 1, \ldots, 255\} \tag{50}$$

$$p_{k|k+1}(y_k|y_{k+1}) = \frac{T_{k-1}(y_k)G(y_k)F(y_k, y_{k+1})}{T_k(y_{k+1})} \qquad k = 7, 6, \ldots, 2 \quad y_k, y_{k+1} \in \{0, 1, \ldots, 255\} \tag{51}$$

$$p_{1|2}(y_1|y_2) = \frac{G(y_1)F(y_1, y_2)}{T_1(y_2)} \tag{52}$$

3. On a computer:
   (a) p_8 is a 1D array of length $2^8 = 256$;
   (b) for each $k \in \{1 \ldots, 7\}$, $p_{k|k+1}$ is a $2^8 \times 2^8 = 256 \times 256$ array (so you should have 7 such 2D arrays: $p_{7|8}$; $p_{6|7}$; $p_{5|6}$; $p_{4|5}$; $p_{3|4}$; $p_{2|3}$; $p_{1|2}$).

4. Sampling is fast; computing the conditional probabilities is slow. So you might want to save the conditional probabilities (each of which is a 256×256 array as mentioned above) after they are computed (or, at least save the "T functions").

5. Debug your dynamic-programming implementation on 2×2 and 3×3 lattices by computing Z_{Temp}, the normalizing constant, and the associated T functions. Here are some results for small lattices (with Temp = 1):

```
2x2 lattice results:
T1: [ 21.18917525   8.20463255   8.20463255  21.18917525]
T2 = Z: 121.232931344

3x3 lattice results:
T1: [ 155.37102759   46.44297052   31.70116107   46.44297052   46.44297052
   31.70116107   46.44297052  155.37102759]
T2: [ 23416.16435187   4634.76802124   3916.10003703   4634.76802124
   4634.76802124   3916.10003703   4634.76802124  23416.16435187]
T3 = Z: 365645.749136
```

In other words, before you start the sampling (which is done in the "backward pass", in effect, in ordering 8,7,6,...,1), you should debug on a small lattice (2×2 or 3×3), making sure you get the right results for the "forward pass". In effect, computing the T's, the last of which is $Z_{\rm Temp}$.

To be more concrete:

- For a 2×2 lattice:
  Each y_s (for $s \in \{1, 2\}$), *i.e.*, a row, is a 1D array of length $2^2 = 4$, whose entries take integral values between 0 and $3 = 2^2 - 1$, inclusive. T_1 is a function of y_2. Since y_2 has 4 different values, it follows that T_1 should be represented as a 1D array of length 4. Particularly, since $T_1(y_2) = \sum_{y_1=0}^{3} G(y_1)F(y_1, y_2)$, it follows that the 4 entries of the T_1 array are:

$$T_1(y_2 = 0) = \sum_{y_1=0}^{3} G(y_1)F(y_1, 0) \tag{53}$$

$$T_1(y_2 = 1) = \sum_{y_1=0}^{3} G(y_1)F(y_1, 1) \tag{54}$$

$$T_1(y_2 = 2) = \sum_{y_1=0}^{3} G(y_1)F(y_1, 2) \tag{55}$$

$$T_1(y_2 = 3) = \sum_{y_1=0}^{3} G(y_1)F(y_1, 3)\,. \tag{56}$$

  T_2 is a scalar because it is the last T (since we have only two rows). In fact, $T_2 = Z_{\rm Temp}$, the normalizing constant:

$$T_2 = Z_{\rm Temp} = \sum_{y_2=0}^{3} T_1(y_2)G(y_2)\,. \tag{57}$$

  Now you can compare it with the brute-force result.

- For a 3x3 lattice: Each y_s (for $s \in \{1, 2, 3\}$), *i.e.*, a row, is a 1D array of length $2^3 = 8$, whose entries take integral values between 0 and $7 = 2^3 - 1$, inclusive. T_1 is a function of y_2. Since y_2 has $2^3 = 8$ different values, it follows that T_1 should be represented as a 1D array of length 8. Particularly, since $T_1(y_2) = \sum_{y_1=0}^{7} G(y_1)F(y_1, y_2)$, it follows that the 8 entries of the T_1 array are:

$$T_1(y_2 = 0) = \sum_{y_1=0}^{7} G(y_1)F(y_1, 0) \tag{58}$$

$$T_1(y_2 = 1) = \sum_{y_1=0}^{7} G(y_1)F(y_1, 1) \tag{59}$$

$$\vdots \tag{60}$$

$$T_1(y_2 = 7) = \sum_{y_1=0}^{7} G(y_1)F(y_1, 7). \tag{61}$$

Now, T_2 is a function of y_3. Since y_3 has $2^3 = 8$ different values, it follows that T_2 should be represented as a 1D array of length 8. Particularly, since $T_2(y_3) = \sum_{y_2=0}^{7} T_1(y_2)G(y_2)F(y_2, y_3)$, it follows that the 8 entries of the T_2 array are:

$$T_2(y_3 = 0) = \sum_{y_2=0}^{7} T_1(y_2)G(y_2)F(y_2, 0) \tag{62}$$

$$T_2(y_3 = 1) = \sum_{y_2=0}^{7} T_1(y_2)G(y_2)F(y_2, 1) \tag{63}$$

$$\vdots \tag{64}$$

$$T_2(y_3 = 7) = \sum_{y_2=0}^{7} T_1(y_2)G(y_2)F(y_2, 7)\,. \tag{65}$$

T_3, since it is the last T (since we have only 3 rows) is a scalar. In fact, $T_3 = Z_{\text{Temp}}$, the normalizing constant.

$$T_3 = Z_{\text{Temp}} = \sum_{y_3=0}^{7} T_2(y_3)G(y_3)\,. \tag{66}$$

Now you can compare it with the brute-force result.

- For an 8×8 lattice: Each y_s (for $s \in \{1, 2, 3, 4, 5, 6, 7, 8\}$), *i.e.*, a row, is a 1D array of length 8, whose entries take integral values between 0 and $255 = 2^8 - 1$, inclusive. So $T_1, T_2, ..., T_7$ are arrays of length $2^8 = 256$, while $T_8 = Z_{\text{Temp}}$ since it is the last one.

  Once you are done with this forward pass, you use these T's to compute the p's.

  p_8 is function of y_8 (in fact it is the pmf over the 256 states y_8 can take). So p_8 should be represented as a 1D array of length $2^8 = 256$.

  $p_{7|8}(y_7, y_8)$ is the probability mass function of y_7 given y_8. So for every value of y_8 – and we have 256 of these – we should have 256 values to cover the probability of all possible 256 states y_7 may assume. So we represent $p_{7|8}$ using a 256 by 256 array.

  Similarly for $p_{6|7}(y_6, y_7)$, $p_{5|6}(y_5, y_6)$, $\ldots$, $p_{1|2}(y_1, y_2)$.

  Finally, to produce the entire sample, we first sample y_8 from p_8 (one way to sample from distribution of a discrete RV is using np.random.choice). Then, we pick the row (or column, depends how you built your $p_{7|8}$) that corresponds to that y_8 we drew, and use it to sample $y_{7|8}$. And so on till we end with sampling $y_{1|2}$.

Remark 6 *As usual, when writing p, we could have also absorbed singletons*

into the pairwise functions, e.g.*:*

$$
\begin{aligned}
p(y_1,\ldots,y_8) \propto & \underbrace{G(y_1)F(y_1,y_2)}_{H_{1,2}(y_1,y_2)}\underbrace{G(y_2)F(y_2,y_3)}_{H_{2,3}(y_2,y_3)}\underbrace{G(y_3)F(y_3,y_4)}_{H_{3,4}(y_3,y_4)} \\
& \times \underbrace{G(y_4)F(y_4,y_5)}_{H_{4,5}(y_4,y_5)}\underbrace{G(y_5)F(y_5,y_6)}_{H_{5,6}(y_5,y_6)}\underbrace{G(y_6)F(y_6,y_7)}_{H_{6,7}(y_6,y_7)} \\
& \times \underbrace{G(y_7)G(y_8)F(y_7,y_8)}_{H_{7,8}(y_7,y_8)} = \prod_{s=1}^{7} H_{s,s+1}(y_s,y_{s+1})\,. \qquad (67)
\end{aligned}
$$

$\Diamond$