Exact solution of statistical mechanical models via Dynamic Programming

Stefano Crotti - stefano.crotti@polito.it

February 14, 2024

Dynamic Programming is a set of techniques for solving some class of mathematical problems in a computationally efficient way. This is typically achieved through a recursive strategy: the solution to a problem is given in the form of an algorithm, and it is found by solving smaller sub-problems similar in aspect to the original one. Here we see two examples where these techniques can be exploited to compute observables of (finite-size) statistical mechanical models. There will be no fancy math involved, just a wise use of the good old summations, multiplications, max/min, and sometimes derivatives.

As a measure of the running time of an algorithm we will be using the big O notation.

Problem 1. Chain-factorized probability distributions

Given n discrete variables $x_1, x_2, \dots, x_n$ each taking values in $\mathcal{X} = \{1, 2, \dots, q\}$ and a function $f : \mathcal{X}^n \to \mathbb{R}$

$$f(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n-1} f_i(x_i, x_{i+1})$$

devise an algorithm running in polynomial time in n, q to compute:

1. $Z = \sum_{x_1, x_2, \dots, x_n} f(x_1, x_2, \dots, x_n)$
2. $p_i(\tilde{x}_i) = \sum_{x_1, x_2, \dots, x_n} \frac{1}{Z} f(x_1, x_2, \dots, x_n) \delta_{x_i, \tilde{x}_i}$

Bonus:

1. Can you compute all marginals $p_i(\tilde{x}_i)\ \forall i$ in time $\mathcal{O}(nq^2)$?
2. Find a polynomial strategy to draw samples from $p(x_1, x_2, \dots, x_n) = \frac{1}{Z} f(x_1, x_2, \dots, x_n)$ (Hint: any multivariate distribution can be decomposed as the product of conditional distributions via repeated application of Bayes' formula, then...)
3. Repeat all of the above for a system with periodic boundary conditions: $f(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} f_i(x_i, x_{i+1})$, with $x_{n+1} \equiv x_1$

Solution (1.) Observe that summing f over x_1(the same reasoning can be done, mirrored, starting from x_n) gives

$$\sum_{x_1} f(x_1, x_2, \ldots, x_n) = \sum_{x_1} f_1(x_1, x_2) \prod_{i=2}^{n-1} f_i(x_i, x_{i+1}) \tag{1}$$

By calling $Z_1(x_2) = \sum_{x_1} f_1(x_1, x_2)$ and $\tilde{f}_2(x_2, x_3) = Z_1(x_2) f_2(x_2, x_3)$, we get that, after summing over x_1, what is left is

$$Z = \sum_{x_2, x_3, \ldots, x_n} \tilde{f}_2(x_2, x_3) \prod_{i=3}^{n-1} f_i(x_i, x_{i+1}) \tag{2}$$

which of the same form as the original problem, but consisting of $n-1$ terms. Applying the same reasoning again, one defines $Z_2(x_3) = \sum_{x_2} \tilde{f}_2(x_2, x_3)$, and so on. The procedure can be iterated until all variables are summed over.

Explicitly, we define

$$Z_i(x_{i+1}) = \sum_{x_1, x_2, \ldots, x_i} \prod_{j=1}^{i} f_j(x_j, x_{j+1}) \tag{3}$$

which is computed recursively from Z_{i-1}as

$$Z_i(x_{i+1}) = \sum_{x_i} Z_{i-1}(x_i) f_i(x_i, x_{i+1}) \tag{4}$$

with initial condition $Z_0(x_1) = 1 \forall x_1$. Finally,

$$Z = \sum_{x_n} Z_{n-1}(x_n). \tag{5}$$

The computational cost is $\mathcal{O}(nq^2)$.

(2.) Marginals $p_i(\tilde{x}_i)$ are computed analogously, one just needs to incorporate the term $\delta_{x_i, \tilde{x}_i}$ when summing over x_i.

Observation 1 A one-dimensional Ising model is described by a distribution over binary spins $\sigma_1, \sigma_2, \ldots, \sigma_n$, $\sigma_i \in \{1, -1\}$

$$p(\sigma_1, \sigma_2, \ldots, \sigma_n) = \frac{1}{Z} \prod_{i=1}^{n} f_i(\sigma_i, \sigma_{i+1})$$

with $f_i(\sigma_i, \sigma_{i+1}) = e^{J\sigma_i\sigma_{i+1} + h\frac{\sigma_i + \sigma_{i+1}}{2}}$ for some $J, h \in \mathbb{R}$. In case you have met the transfer matrix method for solving the one-dimensional Ising model, convince yourself that it is equivalent to the strategy we see here.

Observation 2 A (discrete-time, discrete-space) Markov process is a probability distribution describing the evolution of a random variable x through T time steps. It can always be written as

$$p(x_0, x_1, \ldots, x_T) = p(x_0)p(x_1|x_0)p(x_2|x_1)\cdots p(x_T|x_{T-1}) \tag{6}$$

$$= p(x_0)\prod_{t=0}^{T-1} p(x_{t+1}|x_t) \tag{7}$$

p is manifestly of the form described in problem 1 (if not a simpler sub-case, thanks to the normalization conditions $1 = \sum_{x_{t+1}} p(x_{t+1}|x_t)$). In particular, we can already conclude that the marginal distributions for a Markov process can be computed in time at most $\mathcal{O}(nq^2)$.

Problem 2. Uniform Ising model with external fields

Given n binary variables $\sigma_1, \sigma_2, \ldots, \sigma_n$ each taking values in $\mathcal{X} = \{1, -1\}$ and the probability distribution $p : \mathcal{X}^n \to \mathbb{R}$

$$p(\sigma_1, \sigma_2, \ldots, \sigma_n) = \frac{1}{Z}e^{-\beta H(\sigma_1, \sigma_2, \ldots, \sigma_n)}, \quad -H(\sigma_1, \sigma_2, \ldots, \sigma_n) = \frac{J}{n}\left(\sum_{i=1}^{n}\sigma_i\right)^2 + \sum_{i=1}^{n} h_i\sigma_i$$

with $J, \beta, h_1, h_2, \ldots, h_n \in \mathbb{R}$, devise an algorithm running in polynomial time in n to compute:

1. The partition function $Z = \sum_{\sigma_1, \sigma_2, \ldots, \sigma_n} e^{-\beta H(\sigma_1, \sigma_2, \ldots, \sigma_n)}$
2. Marginals $p_i(\tilde{\sigma}_i) = \sum_{\sigma_1, \sigma_2, \ldots, \sigma_n} p(\sigma_1, \sigma_2, \ldots, \sigma_n)\delta_{\sigma_i, \tilde{\sigma}_i}$
3. The average energy $\langle H \rangle = \sum_{\sigma_1, \sigma_2, \ldots, \sigma_n} p(\sigma_1, \sigma_2, \ldots, \sigma_n)H(\sigma_1, \sigma_2, \ldots, \sigma_n)$ (Hint: use linearity of expectation values)
4. Can you do 3., but faster than $\mathcal{O}(n^3)$? (Hint: $\langle H \rangle$ is related to the derivative of a quantity which you can compute in $\mathcal{O}(n^2)$)

Bonus

1. The entropy $k = -\langle \log p \rangle = -\sum_{\sigma_1, \sigma_2, \ldots, \sigma_n} p(\sigma_1, \sigma_2, \ldots, \sigma_n) \log p(\sigma_1, \sigma_2, \ldots, \sigma_n)$
2. A polynomial strategy do draw samples from p
3. Do the same for a Potts model: $\sigma_i \in \mathcal{X} = \{1, 2, \ldots, q\}$ with energy

$$-H(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} f(\sigma_i, \sigma_j) + \sum_{i=1}^{n} h_i(\sigma_i)$$

with $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $h_1, h_2, \ldots, h_n : \mathcal{X} \to \mathbb{R}$. The procedure must be polynomial in n, q. (Hint: observe that the first term only depends on how many variables are in each of the q states)

Observation 3 Consider the computation of factor-dependent partition functions in the context of the Belief Propagation algorithm (knowledge of the workings of the algorithm is not needed for this exercise, but if you are interested you can check out [MM09, Ch14]). For a ferromagnetic p-spin model the equation reads:

$$z_{factor} = \sum_{\sigma_1,\sigma_2,\dots,\sigma_p} e^{J\prod_{i=1}^{p}\sigma_i} \prod_{i=1}^{p} m_i(\sigma_i)$$

where the functions $\{m_i\}_{i=1:p}$ are probability distributions and are interpreted as "incoming messages" and $\sigma_i \in \{1,-1\}$. Naively , computing z_{factor} takes $\mathcal{O}(2^p)$ operations. Luckily, the expression above is of the same form as the one in problem 2, just with a product instead of a sum at the exponent. The calculation can be carried out in a similar way, in time $\mathcal{O}(p)$.

Bonus:

1. Can you also compute the "outgoing messages"

$$\mu_i(\sigma_i) \propto \sum_{\sigma_1,\sigma_2,\dots,\cancel{\sigma_i},\dots,\sigma_p} e^{J\prod_{j=1}^{k}\sigma_j} \prod_{j\neq i} m_j(\sigma_j)$$

   $\forall i \in \{1,2,\dots,p\}$ in time $\mathcal{O}(p)$? (Hint: use a similar reasoning as bonus point 1 in problem 1)

2. Can you come up with a similar strategy for the case of Low Density Parity Check decoding [MM09, Ch11]:

$$z_{factor} = \sum_{x_1,x_2,\dots,x_k} \mathbb{1}\left[\sum_{i=1}^{k} x_i = 0 \mod 2\right] \prod_{i=1}^{k} m_i(x_i)$$

   where $x_1,\dots,x_k \in \{0,1\}$, in time $\mathcal{O}(k)$? (Recall that $x \mod 2$ equals 0 if x is even, 1 if x is odd)

3. And for k-SAT [MM09, Ch10]:

$$z_{factor} = \sum_{\sigma_1,\sigma_2,\dots,\sigma_k} e^{\prod_{i=1}^{k}\frac{1-l_i\sigma_i}{2}} \prod_{i=1}^{k} m_i(\sigma_i)$$

   where $l_i \in \{1,-1\}$, $\sigma_i \in \{1,-1\}$, in time $\mathcal{O}(k)$?

Solution (1.) Re-write the summation by grouping together configurations for which the sum of the spins is equal:

$$Z = \sum_{\sigma_1,\sigma_2,\dots,\sigma_n} \sum_{s=-n}^{n} \mathbb{1}\left[\sum_{i=1}^{n}\sigma_i = s\right] e^{-\beta H(\sigma_1,\sigma_2,\dots,\sigma_n)} \tag{8}$$

$$= \sum_{s=-n}^{n} e^{\frac{\beta J}{n}s^2} \sum_{\substack{\sigma_1,\sigma_2,\dots,\sigma_n \\ \sum_{i=1}^{n}\sigma_i = s}} \prod_{i=1}^{n} e^{\beta h_i \sigma_i} \tag{9}$$

Now focus on the second summation. Let us call it

$$Z_n(s) = \sum_{\substack{\sigma_1,\sigma_2,\ldots,\sigma_n \\ \sum_{i=1}^n \sigma_i = s}} \prod_{i=1}^{n} e^{\beta h_i \sigma_i} \tag{10}$$

This can be computed recursively. One sums over σ_n first (the same reasoning can be done, mirrored, starting from σ_i)

$$Z_n(s) = \sum_{\sigma_n} e^{\beta h_n \sigma_n} \underbrace{\sum_{\substack{\sigma_1,\ldots,\sigma_{n-1} \\ \sum_{i=1}^{n-1} \sigma_i = s - \sigma_n}} \prod_{i=1}^{n-1} e^{\beta h_i \sigma_i}}_{=: Z_{n-1}(s-\sigma_1)} \tag{11}$$

$$= \sum_{\sigma_n} e^{\beta h_n \sigma_n} Z_{n-1}(s - \sigma_n) \tag{12}$$

The quantity Z_{n-1} is of the same form as Z_n. Similarly, each Z_i can be computed in terms of Z_{i-1}, and the procedure can be carried out recursively down to $i = 1$. Explicitly, we define

$$Z_i(s) = \sum_{\substack{\sigma_1,\sigma_2,\ldots,\sigma_i \\ \sum_{j=1}^{i} \sigma_j = s}} \prod_{j=1}^{i} e^{\beta h_j \sigma_j} \tag{13}$$

whose domain is $s \in \{-i, -i+1, \ldots, i-1, i\}$. It is computed from Z_{i-1}as

$$Z_i(s) = \sum_{\sigma_i} e^{\beta h_i \sigma_i} Z_{i-1}(s - \sigma_i) \tag{14}$$

with initial condition $Z_0(s) = \delta_{s,0}$.

The computational cost for $Z_i(s)$ is constant (independent of n) for each of the $2i+1$ possible values of s. Summed up for all i gives a total $\sum_{i=1}^{n}(2i+1) = n + n(n+1) = \mathcal{O}(n^2)$.

(2.) Marginals can be computed analogously, one just needs to incorporate the term $\delta_{\sigma_i,\tilde{\sigma}_i}$ when computing Z_i. The cost is the same as the one for Z, repeated once per each variable, hence $\mathcal{O}(n^3)$.

(3.) By linearity, the average energy is

$$-\langle H \rangle = \frac{J}{n} \left\langle \left(\sum_{i=1}^{n} \sigma_i \right)^2 \right\rangle + \sum_{i=1}^{n} h_i \langle \sigma_i \rangle \tag{15}$$

where the second term is obtained easily from the marginals. The first term is given by

$$\left\langle \left(\sum_{i=1}^{n} \sigma_i \right)^2 \right\rangle = \sum_{\sigma_1, \sigma_2, \ldots, \sigma_n} \left(\sum_{i=1}^{n} \sigma_i \right)^2 p(\sigma_1, \sigma_2, \ldots, \sigma_n) \tag{16}$$

$$= \frac{1}{Z} \sum_{s=-n}^{n} s^2 e^{\frac{\beta J}{n} s^2} \sum_{\substack{\sigma_1, \sigma_2, \ldots, \sigma_n \\ \sum_{i=1}^{n} \sigma_i = s}} \prod_{i=1}^{n} e^{\beta h_i \sigma_i} \tag{17}$$

$$= \frac{1}{Z} \sum_{s=-n}^{n} s^2 e^{\frac{\beta J}{n} s^2} Z_n(s) \tag{18}$$

for a total $\mathcal{O}(n^3)$ cost. Notice that the bottleneck is given by the computation of the marginals.

(4.) Use the identity

$$\langle H \rangle = -\frac{1}{Z} \frac{\partial Z}{\partial \beta} \tag{19}$$

Substituting the efficient computation for Z gives

$$\frac{\partial Z}{\partial \beta} = \frac{\partial}{\partial \beta} \left[\sum_{s} e^{\frac{\beta J}{n} s^2} Z_n(s) \right] \tag{20}$$

$$= \sum_{s} \left[\frac{J}{n} s^2 Z_n(s) + \frac{\partial}{\partial \beta} Z_n(s) \right] e^{\frac{\beta J}{n} s^2} \tag{21}$$

Now computing $\frac{\partial}{\partial \beta} Z_n(s)$ simply amounts to differentiating through the recursion for $Z_n(s)$. At each step $\frac{\partial}{\partial \beta} Z_i$ is computed in terms of $Z_{i-1}$and $\frac{\partial}{\partial \beta} Z_{i-1}$:

$$\frac{\partial}{\partial \beta} Z_i(s) = \frac{\partial}{\partial \beta} \left[\sum_{\sigma_i} e^{\beta h_i \sigma_i} Z_{i-1}(s - \sigma_i) \right] \tag{22}$$

$$= \sum_{\sigma_i} \left[h_i \sigma_i Z_{i-1}(s - \sigma_i) + \frac{\partial}{\partial \beta} Z_{i-1}(s - \sigma_i) \right] e^{\beta h_i \sigma_i} \tag{23}$$

with initial condition

$$\frac{\partial}{\partial \beta} Z_0(s) = 0 \tag{24}$$

Remarkably, the (asymptotic) computational cost is the same as the one for computing Z.

Acknowledgement These exercises are heavily inspired by Alfredo Braunstein's notes for the course "Algorithms for optimization, inference, and learning".

References

[MM09] Marc Mézard and Andrea Montanari. *Information, Physics, and Computation.* Oxford University Press, USA, 2009.