Sampling with Tensor-Trains

May 2022

Hongli Zhao, honglizhaobob@uchicago.edu

This short note accompanies the works [7, 5] and describes an important subroutine for obtaining samples from a high-dimensional probability distribution. In this note, we follow the same notation conventions in the works referenced. The sampling routine is based on the functional tensor-train (FTT) representation of the density, originally described in [3]. The coordinate samples are obtained individually by *conditional sampling* [4], first applied in [2], but only for discrete tensor-trains. Let d denote the number of dimensions, the following advantages are enjoyed by the method is this note:

(1) Continuity: sampling at arbitrary points in the space, not only grid points
(2) Scalable: runtime and storage complexity depends linearly on dimensions d.
(3) Valid probability density: marginals always are nonnegative and normalized; no additional approximation or correction is required.

Let $\mathbf{x} = (x_1, \ldots, x_d)$ denote relevant spatial coordinates, $p = p(x_1, \ldots, x_d)$ denote the target probability density (square integrable) and only known up to a normalization constant. Define $q = \sqrt{p}$. We suppose a functional tensor-train approximation is already constructed (e.g. with the TT-cross method [6]):

$$q \approx \sum_{\mathcal{I}} \left(\sum_{\alpha_1, \ldots, \alpha_{d-1}}^{r_1, \ldots, r_{d-1}} \mathcal{A}_1[1, i_1, \alpha_1] \cdots \mathcal{A}_d[\alpha_{d-1}, i_d, 1] \right) \phi_{\mathcal{I}}(x_1, \ldots, x_d) \tag{1}$$

where $\mathcal{I}$ is multi-index in $\mathbb{N}^d$, and $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is a multidimensional basis function (e.g. Chebyshev, Legendre). $\phi_{\mathcal{I}}(\mathbf{x}) = \phi_{i_1}(x_1) \cdots \phi_{i_d}(x_d)$.

$$= \sum_{\alpha_1, \ldots, \alpha_{d-1}} \left(\sum_{i_1=0}^{n_1} \mathcal{A}_1[1, i_1, \alpha_1] \phi_{i_1}(x_1) \right) \cdots \left(\sum_{i_d=0}^{n_d} \mathcal{A}_d[\alpha_{d-1}, i_d, 1] \phi_{i_d}(x_d) \right) = \mathcal{F}_1[:, x_1, :] \cdots \mathcal{F}_d[:, x_d, :]$$

Remark. Building a functional tensor-train To be more concrete, we take a detour and describe the process of computing a continuous interpolation given discrete sample points. Let $\{x^{(i)}\}_{i=1}^M$ be the grid points on interval $[a, b]$. We require:

$$I_n q(x^{(i)}) := \sum_{j=0}^{n} c_j \phi_j(x^{(i)}) = q(x^{(i)}), \forall i \in \{1, 2, \ldots, M\} \tag{2}$$

where I_N denotes the interpolation operator with order n. The above problem can either be solved with Galerkin projection using the fact that the basis functions ϕ are orthogonal, or regression. The regression will look like the following:

$$\begin{bmatrix} \phi_0(x^{(1)}) & \cdots & \phi_n(x^{(1)}) \\ \vdots & \ddots & \vdots \\ \phi_0(x^{(M)}) & \cdots & \phi_n(x^{(M)}) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} q(x^{(1)}) \\ \vdots \\ q(x^{(M)}) \end{bmatrix} \tag{3}$$

or more compactly:

$$\mathbf{\Phi c} = \mathbf{q}$$

The best fit coefficients is given by the pseudoinverse:

$$\mathbf{c}^* = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{q} \tag{4}$$

We make the remark that matrix $\mathbf{\Phi}$ is typically ill-conditioned when $n \gg M$. The above can be solved in each dimensions.

Continuing our sampling derivation, the last expression in (1) gives a more direct interpretation as a tensor-train where each core is a square-integrable function in the variable x_k. We begin by deriving the first marginal using the TT representation of q.

$$p_1(x_1) = \int p(x_1, \ldots, x_d) dx_2 \cdots dx_d = \int q^2(x_1, \ldots, x_d) dx_2 \cdots dx_d \tag{5}$$

$$= \sum_{\substack{i_1,\ldots,i_d \\ j_1,\ldots,j_d}} \mathcal{C}[i_1, \mathcal{I}_{>1}] \mathcal{C}[j_1, \mathcal{J}_{>1}] \phi_{i_1}(x_1) \phi_{j_1}(x_1) \delta_{i_1,j_1} \cdots \delta_{i_d,j_d} = \sum_{i_1,j_1} \mathcal{B}[i_1, j_1] \phi_{i_1}(x_1) \phi_{j_1}(x_1)$$

where $\mathcal{C}$ is formed by contracting the *TT cores* $\mathcal{A}$'s at appropriate indices. And $\mathcal{B}[i_1, j_1] := \sum_{i_2,j_2,\ldots,i_d,j_d} \mathcal{C}[i_1, \ldots, i_d] \mathcal{C}[j_1, \ldots, j_d]$. $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise. One can imagine a ladder-like structure, and for each marginal k, we are contracting all the rungs other than position k.

We notice that the marginal is continuous and is non-negative by definition of the matrix $\mathcal{B}$. To sample from the marginal numerically, one may then specify a quadrature with any desired level of refinement, and apply inverse transform sampling:

$$u_1 \sim \mathcal{U}(0, 1), \text{ solve } F(x_1) = \int_{-\infty}^{x_1} p_1(y) dy = u_1 \text{ for } x_1 \tag{6}$$

Since we know the basis functions analytically, we may furtuer simplify:

$$u_1 = \int_{-\infty}^{x_1} p_1(y) dy = \int_{-\infty}^{x_1} \sum_{i_1,j_1} \mathcal{B}[i_1, j_1] \phi_{i_1}(y) \phi_{j_1}(y) dy = \sum_{i_1,j_1} \mathcal{B}[i_1, j_1] \Phi_{i_1 j_1}(x_1) \tag{7}$$

where:

$$\Phi_{i_1 j_1}(x) = \int_{-\infty}^{x} \phi_{i_1}(y) \phi_{j_1}(y) dy \tag{8}$$

is the antiderivative of the product, which is known analytically because it is a polynomial (if we use a certain class of polynomial bases). This can be evaluated at very little cost. Sampling (6) can then be considered as solving a root-finding problem and can be done using any standard algorithms.

The computation of other marginals/conditionals follow a similar, sequential procedure. The former already-sampled indices must be fixed by evaluating:

$$\phi^*_{\mathcal{I}<k} := \phi_{i_1}(x_1^*)\cdots\phi_{i_{k-1}}(x^*_{k-1}) \tag{9}$$

and contracted with corresponding indices of $\mathcal{C}$. And the latter is again simplified due to orthogonality of basis functions. At sampling coordinate k, the marginal distribution in x_k has the following form:

$$p_k(x_k|\mathbf{x}^*_{<k}) \propto \sum_{\substack{i_1,\dots,i_d\\ j_1,\dots,j_d}} \mathcal{C}[\mathcal{I}_{<k}, i_k, \mathcal{I}_{>k}]\mathcal{C}[\mathcal{J}_{<k}, j_k, \mathcal{J}_{>k}]\phi_{\mathcal{I}^*_{<k}}\phi^*_{\mathcal{J}>k}\phi_{i_k}(x_k)\phi_{j_k}(x_k)\delta_{\mathcal{I}>k,\mathcal{J}_{>k}} \tag{10}$$

As indices other than i_k, j_k are fixed, it is appropriate to put the above expression in a similar format to (5):

$$p_k(x_k|\mathbf{x}^*_{<k}) \propto \sum_{i_k,j_k} \mathcal{B}_k[i_k, j_k]\phi_{i_k}\phi_{j_k} \tag{11}$$

where:

$$\mathcal{B}_k = \sum_{\substack{I_{<k},\mathcal{I}_{>k}\\ \mathcal{J}_{<k},\mathcal{J}_{>k}}} \mathcal{C}[\mathcal{I}_{<k}, i_k, \mathcal{I}_{>k}]\mathcal{C}[\mathcal{J}_{<k}, j_k, \mathcal{J}_{>k}]\delta_{\mathcal{I}_{<k},\mathcal{J}_{>k}} \tag{12}$$

In words, computing the matrix $\mathcal{B}_k$ now only amounts to computing the products between the first $(k-1)$ cores instead of all d cores. Moreover, in the sequential procedure, the "history" (of computed matrices) can be stored, updated and queried for subsequent computations.

We summarize the full sampling process in the following section.

1. Main Routine

The preparation phase refers to putting the tensor-train in, for instance, "right-left orthogonal" form by sequential QR decomposition. This step is not required, but would greatly save computational overhead during sampling during contracting of the "rungs", where we effectively obtain identity matrices. More details can be found in [1]. It is also possible to consider "middle out" QR forms, hierarchical, or other patterned QR forms when d is considerably high. However, we leave that exploration to future work.

In this section, we derive the sampling procedure. We sequentially keep track of the fixed coordinates $\mathbf{x}^*_{<k} = (x_1^*, \dots, x^*_{k-1})$. At each step, we need to have a representation of the marginal $p(x_1, \dots, x_{k-1}, x_k)$, where we fix the first $(k-1)$ variables and marginalize out the trailing variables $x_{k+1}, \dots, x_d$. By expression the marginal distribution as a Hadamard product of tensors, we obtain:

$$p(\mathbf{x}^*_{<k}, x_k) = \int q^2(\mathbf{x}^*_{<k}, x_k, \mathbf{x}_{>k})d\mathbf{x}_{>k} = \sum_{i_k,j_k} \mathcal{B}_k[i_k, j_k]\phi_{i_k}(x_k)\phi_{j_k}(x_k) \tag{13}$$

with:

$$\begin{aligned}\mathcal{B}_k[i_k, j_k] &= \sum_{\substack{I_{<k}, \mathcal{I}_{>k} \\ \mathcal{J}_{<k}, \mathcal{J}_{>k}}} \mathcal{C}[\mathcal{I}_{<k}, i_k, \mathcal{I}_{>k}]\mathcal{C}[\mathcal{J}_{<k}, j_k, \mathcal{J}_{>k}]\phi_{\mathcal{I}^*_{<k}}\phi^*_{\mathcal{J}_{<k}} \int \phi_{\mathcal{I}>k}\phi_{\mathcal{J}_{>k}} d\mathbf{x}_{>k} \\ &= \sum_{\substack{I_{<k}, \mathcal{I}_{>k} \\ \mathcal{J}_{<k}, \mathcal{J}_{>k}}} \mathcal{C}[\mathcal{I}_{<k}, i_k, \mathcal{I}_{>k}]\mathcal{C}[\mathcal{J}_{<k}, j_k, \mathcal{J}_{>k}]\phi_{\mathcal{I}^*_{<k}}\phi^*_{\mathcal{J}_{<k}}\delta_{\mathcal{I}_{>k}, \mathcal{J}_{>k}}\end{aligned}$$

note that the dirac delta arises from orthogonality of basis functions when marginalizing the trailing variables. As mentioned, we may by-pass explicitly contracting cores involving the trailing variables. Substituting in the orthogonalized cores $\mathcal{Q}, \mathcal{R}$, we have:

$$\left(\sum_{i_1, i_1'} \mathcal{R}_1[1, i_1, :]\mathcal{R}_1[1, i_1', :]\right) \cdot \left(\sum_{i_1, i_1'} \mathcal{Q}_1[1, i_1, :]\mathcal{Q}_1[1, i_1', :]\right) \tag{14}$$

for indices $\mathcal{I}_{<k}, \mathcal{J}_{<k}$. We make clear the contraction operations needed for each coordiante and derive a sequential procedure. Let us define:

$$\begin{cases} \tilde{R}_1[\alpha_1; \alpha_1'] = \sum_{i_1, i_1'} \mathcal{R}_1[1, i_1, \alpha_1]\mathcal{R}_1[1, i_1', \alpha_1']\phi^*_{i_1}\phi^*_{i_1'} \\ \tilde{Q}_s[\alpha_{s-1}, \alpha_s; \alpha_{s-1}', \alpha_s'] = \sum_{i_s, i_s'} \mathcal{Q}_s[\alpha_{s-1}, i_s, \alpha_s]\mathcal{Q}_s[\alpha_{s-1}', i_s', \alpha_s']\phi^*_{i_s}\phi^*_{i_s'} \\ (s = 2, \ldots, k-1) \end{cases}$$

As for indices $\mathcal{I}_{>k}, \mathcal{J}_{>k}$, we make the dirac delta arising from orthogonality of the basis functions more explicit:

$$\begin{cases} \tilde{Q}_s[\alpha_{s-1}, \alpha_s; \alpha_{s-1}', \alpha_s] = \sum_{i_s, i_s'} \mathcal{Q}_s[\alpha_{s-1}, i_s, \alpha_s]\mathcal{Q}_s[\alpha_{s-1}', i_s', \alpha_s']\delta_{i_s, i_s'} \\ (s = k+1, \ldots, d) \end{cases} \tag{15}$$

where we note that:

$$\sum_{\{\alpha_s = \alpha_s'\}} \tilde{\mathcal{Q}}_s[\alpha_{s-1}, \alpha_s; \alpha_{s-1}', \alpha_s'] = \langle \mathcal{Q}_s[\alpha_{s-1}, :, :], \mathcal{Q}_s[\alpha_{s-1}', :, :]\rangle_{i_s} = I_{r_s} \tag{16}$$

By the above definitions, we obtain from left to right via contracing $\alpha_1, \ldots, \alpha_{s-1}$, and $\alpha_{s+1}, \ldots, \alpha_d$, which is updated sequentially, one can also see positive-semidefiniteness from the below expressions:

$$\mathcal{B}_1[i_1, i_1'] = \sum_{\alpha_1, \alpha_1'} \mathcal{R}_1[1, i_1, \alpha_1]\mathcal{R}_1[1, i_1', \alpha_1'] \tag{17}$$

and:

$$\mathcal{B}_k[i_k, i'_k] = \sum_{\substack{1,\ldots,\alpha_{k-1},\alpha_k \\ \alpha'_1,\ldots,\alpha'_{k-1},\alpha'_k}} \underbrace{\tilde{R}_1\tilde{Q}_2\cdots\tilde{Q}_{k-1}}_{=\mathcal{B}_{k-1}} \mathcal{Q}_k[\alpha_{k-1}, i_k, \alpha_k]\mathcal{Q}_k[\alpha'_{k-1}, i'_k, \alpha'_k] \tag{18}$$

for $k > 1$. This means the computation of $\mathcal{B}_k$ only involves cores up to $(k-1)$, instead of all d cores.

Finally, by direct integration, we have that:

$$\sum_{i_k, i'_k} \int \phi_{i_k}\phi_{i'_k} dx_k = \sum_{i_k, i'_k} \mathcal{B}_k \delta_{i_k, i'_k} = \mathrm{Tr}[\mathcal{B}_k] \tag{19}$$

such that each conditional density can be normalized:

$$p_k = \frac{1}{\mathrm{Tr}[\mathcal{B}_k]} \sum_{i_k, i'_k} \mathcal{B}_k \phi_{i_k}\phi_{i'_k} \tag{20}$$

The next section discusses computational complexity involved.

2. Complexity of Sampling

Let $r = \max_{1\le k\le d} r_k$, $n = \max_{1\le k\le d} n_k$, and sample size be N. In the preparatoion phase, a reduced QR decomposition on $O(d)$ unfolding matrices, each of size $nr \times r$, costing at most $O(nr^3)$. The preparation of orthogonalizing each core costs a total of $O(dnr^3)$ to complete. During the sampling phase, we never expkicitly form the tensor Hadamard product, and only keep track of a "square root marginal matrix" $P^{(k)}$ such that $\mathcal{B}_k = P^{(k)}(P^{(k)})^T$ and fixed basis vectors $\boldsymbol{\phi}_1^*, \cdots, \boldsymbol{\phi}_{k-1}^*$. We modify the matrix $P^{(k+1)} \leftarrow \texttt{update}(P^{(k)})$ from left to right as we proceed with sampling the coordinates. Forming basis vectors require $O(dn)$ time in total (assuming polynomial evaluation is $O(1)$), a matrix-vector multiply to fix the last coordinate costs $O(n^2 r)$, updating the matrix $P^{(k)}$ by a tensor-vector cotraction with core $\mathcal{Q}_k$ costs $O(nr^2)$. Finally, forming the polynomial in (8) can be done in one matrix-vector multiply as $\mathbf{v}_k(x_k) = (P^{(k)})\boldsymbol{\phi}_k(x_k)$, and returning $\mathbf{v}(x_k)\mathbf{v}_k^T(x_k)$, which in total costs $O(n^2 r + r^2)$. We assume solving the polynomial equation costs $O(1)$ time per coordinate, or $O(d)$ in total. The full runtime complexity is thus:

$$O(dnr^3) + O(Nd \cdot (n + 2n^2 r + r^2 + nr^2)) \sim O(dnr^3) + O(Ndn^2 r^2) \tag{21}$$

which is linear in dimensions d and sample size N, and quadratic in rank r. Thus, having a low rank structure for the problem is crucial for efficiency.

References

[1] Olivier Coulaud, Luc Giraud, and Martina Iannacito. *On some orthogonalization schemes in Tensor Train format.* 2022. arXiv: `2211.08770 [cs.DC]`.

[2] Sergey Dolgov et al. *Approximation and sampling of multivariate probability distributions in the tensor train decomposition.* 2019. arXiv: `1810.01212 [math.NA]`.

[3] Alex Gorodetsky, Sertac Karaman, and Youssef Marzouk. "A continuous analogue of the tensor-train decomposition". In: *Computer Methods in Applied Mechanics and Engineering* 347 (2019), pp. 59–84. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2018.12.015. URL: https://www.sciencedirect.com/science/article/pii/S0045782518306133.

[4] Wolfgang Hörmann, Josef Leydold, and Gerhard Derflinger. "Transformed Density Rejection (TDR)". In: *Automatic Nonuniform Random Variate Generation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 55–111. ISBN: 978-3-662-05946-3. DOI: 10.1007/978-3-662-05946-3_4. URL: https://doi.org/10.1007/978-3-662-05946-3_4.

[5] Yuehaw Khoo, Michael Lindsey, and Hongli Zhao. *Tensorizing flows: a tool for variational inference*. 2023. arXiv: 2305.02460 [cs.LG].

[6] Ivan Oseledets and Eugene Tyrtyshnikov. "TT-cross approximation for multidimensional arrays". In: *Linear Algebra and its Applications* 432.1 (2010), pp. 70–88. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2009.07.024. URL: https://www.sciencedirect.com/science/article/pii/S0024379509003747.

[7] Yinuo Ren et al. *High-dimensional density estimation with tensorizing flow*. 2022. arXiv: 2212.00759 [cs.LG].