Product Jacobi-Theta Boltzmann machines with score matching

Andrea Pasquale, Daniel Krefl, Stefano Carrazza, Frank Nielsen

5th September 2022

UNIVERSITÀ
DEGLI STUDI
DI MILANO

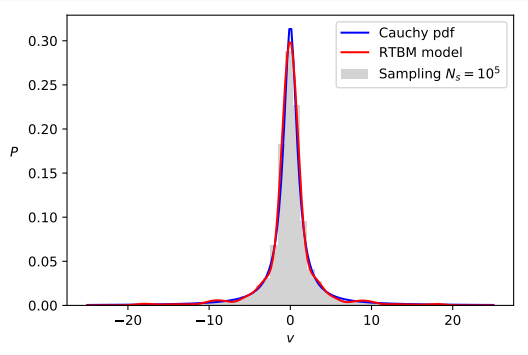
Introduction

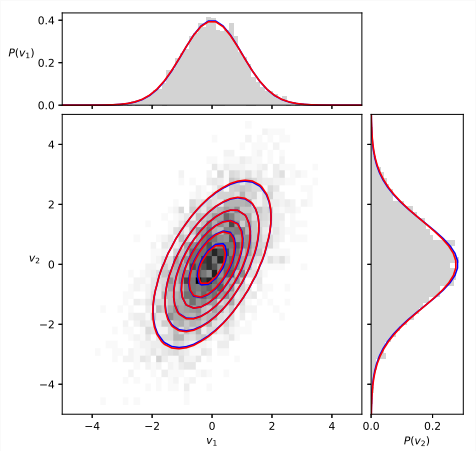
Introduction

We started this project aiming to build a model with:

- well suited for pdf estimation and pdf sampling
- built-in pdf normalization (close form expression)
- very flexible with a small number of parameters

We started from Boltzmann Machines

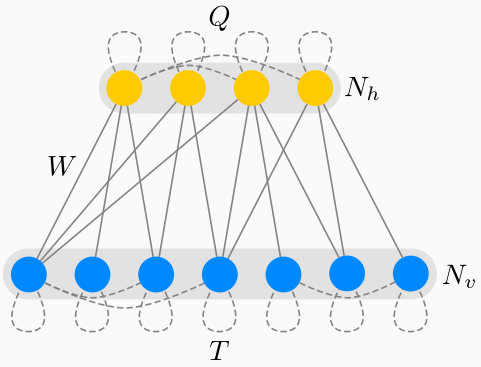




Theory

Boltzmann Machine

Graphical representation



Main Features:

- Visible sector with N_v nodes
- Hidden sector with N_h nodes
- Binary valued states **{0,1}** all the nodes
- Connection matrices Q, T and W between the nodes

This can be viewed as a statistical system with the following energy for a given state (v, h):

$$E(v,h) = \frac{1}{2}v^t T v + \frac{1}{2}h^t Q h + v^t W h + B_h h + B_v v$$

Boltzmann Machine

Following a statistical mechanics approach we can compute the canonical partition function as:

$$Z = \sum_{h,v} e^{-E(v,h)}$$

The probability of finding the system is given by the Boltzmann distribution:

$$P(v,h) = \frac{e^{-E(v,h)}}{Z}$$

The probability of finding the system in the state v can be computed by marginalizing h

$$P(v) = \sum_{h} \frac{e^{-E(v,h)}}{Z} = \frac{e^{-F(v)}}{Z}$$

where $F(v)$ is the the free energy of the system.

Boltzmann Machine (BM)

$P(v)$ is a parametric density probability which depends on the connection matrices Q, T and W and the biases B_v and B_h. Theoretically one could adjust these parameters to learn an unknown probability distribution of a given dataset.

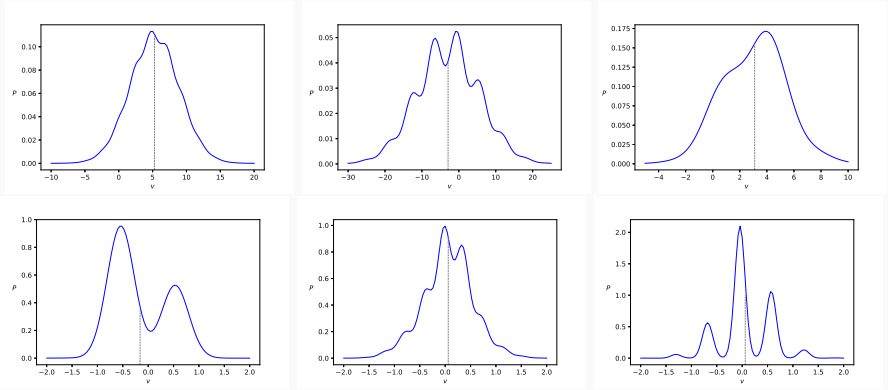
However, this approach is practically **not feasible**.

How can we change BMs to perform density estimation?

We can obtain a non-trivial model if we change the domain from binary values states to continuous/quantized values. If we consider $v \in \mathbb{R}^{N_v}$ and $h \in \mathbb{Z}^{N_h}$, under mild constraints on the connection matrices we can compute $P(v)$ in a closed form:

$$P(v) = \sqrt{\frac{\det T}{(2\pi)^{N_v}}} e^{-\frac{1}{2}v^t T v - B_v^t v - B_v^t T^{-1} B_v} \frac{\tilde{\theta}(B_h^t + v^t W | Q)}{\tilde{\theta}(B_h^t - B_v^t T^{-1} W | Q - W^t T^{-1} W)} \ .$$

which corresponds to a multi-variate gaussian moldulated by the functions $\tilde{\theta}$, which are known as **Riemann-Theta (RT)** functions.
We called this model the **Riemann-Theta Boltzmann Machine (RTBM)**.



A few examples of $P(v)$ for different parameters.

Properties of RTBMs

The RTBM can be used to perform density estimation by adjusting the values of the connection matrices and of the biases. In particular, we need to solve the following optimization problem:

$$\underset{Q,T,W,B_h,B_v}{\arg\min} \quad \mathcal{C}$$

where $\mathcal{C}$ is an arbitrary cost function.

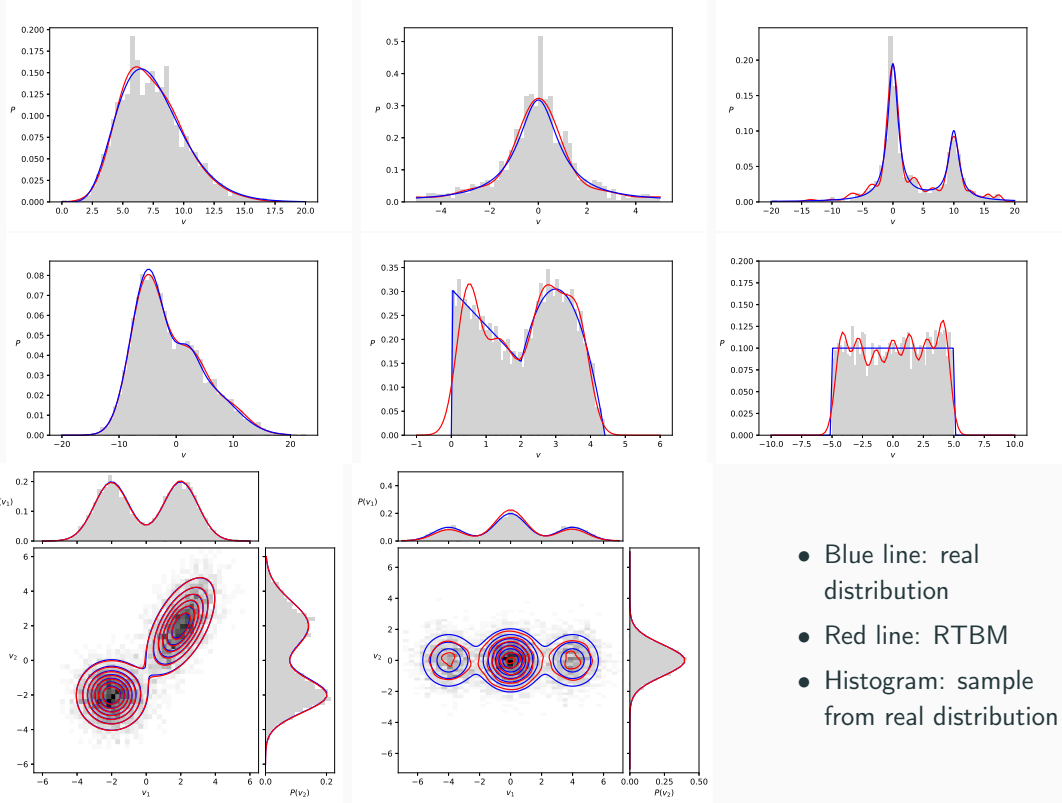
In the case of Maximum Likelihood Estimation (MLE) we get:

$$\underset{Q,T,W,B_h,B_v}{\arg\min} \; -\sum_{i=1}^{N} \log P(x_i) = \underset{Q,T,W,B_h,B_v}{\arg\max} \sum_{i=1}^{N} \log P(x_i)$$

Since we have an analytical expression for $P(v)$ we have the possibility to use both gradient or non-gradient based techniques.

Examples

[Krefl et al., 2017]



- Blue line: real distribution
- Red line: RTBM
- Histogram: sample from real distribution

The probability for the visible sector can be expressed as:

$$P(v) = \sum_{[h]} P(v|h)P(h)\,,$$

where $P(v|h)$ is a multivariate gaussian. $P(v)$ can be easily sampled using the following algorithm:

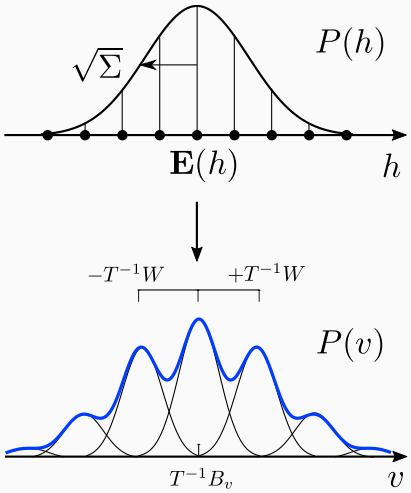
- sample $\mathbf{h} \sim P(h)$ using RT numerical evaluation $\theta = \theta_n + \epsilon(R)$ with ellipsoid radius R such that

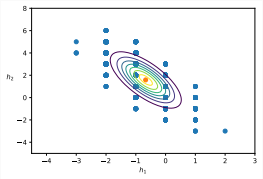
  $$p = \frac{\epsilon(R)}{\theta_n + \epsilon(R)} \ll 1$$

  is the probability that a point is sampled outside the ellipsoid of radius R, while

  $$\sum_{[h](R)} P(h) = \frac{\theta_n}{\theta_n + \epsilon(R)} \approx 1\,,$$

  is the sum over the lattice points inside the ellipsoid.
- sample $\mathbf{v} \sim P(v|\mathbf{h})$

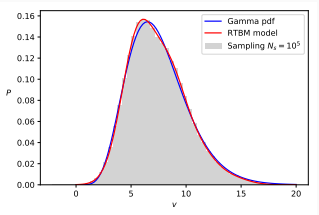


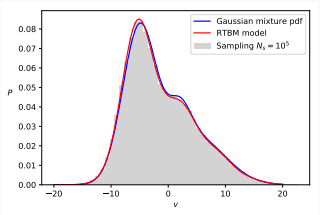


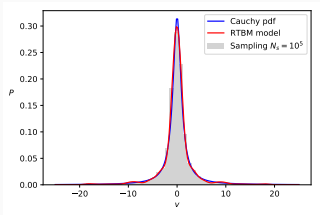
Sampling examples

[Carrazza and Krefl, 2018]

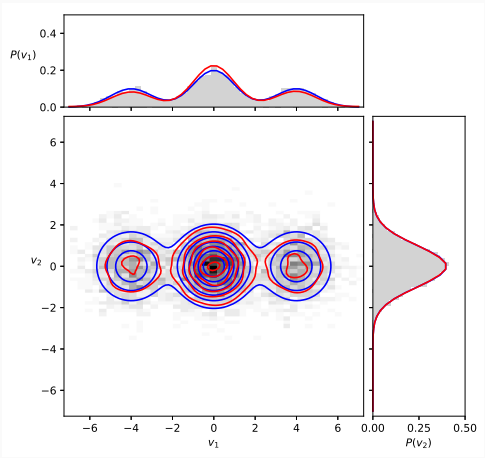
One dimensional case:







Multi-dimensional case



- Blue line: Real distribution
- Red line: RTBM
- Histogram: sample from RTBM

We observe that $P(v)$ stays in the same distribution under affine transformations, i.e. rotation and translation

$$\mathbf{w} = A\mathbf{v} + b\,, \quad \mathbf{w} \sim P_{A,b}(v)\,,$$

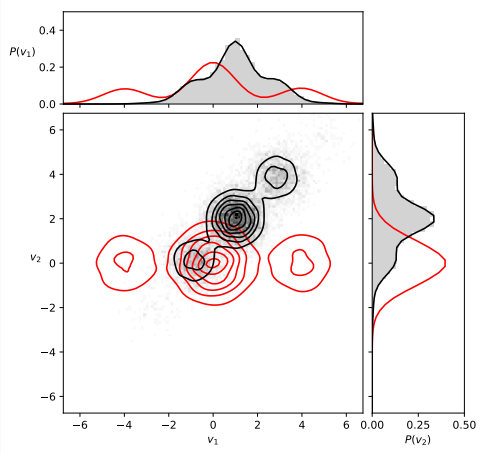
if the linear transformation A has a full column rank. The connection matrices and the biases of the transformed RTBM are given by:

$$T^{-1} \to AT^{-1}A^t\,, \quad B_v \to (A^+)^t B_v - Tb\,,$$
$$W \to (A^+)^t W\,, \quad B_h \to B_h - W^t b\,.$$

where A^+ is the left pseudo-inverse defined as

$$A^+ = (A^t A)^{-1} A^t\,.$$

Example: rotation of $\theta/4$ and scailing of $1/2$ $(N_v = 2, N_h = 2)$



Limitations

Limitations

Despite the promising results there is one major issue:

The learning process can become slow for ($N_h > 5$).

Therefore, it can become challenging to estimate the density of models which require a large hidden sector such as:

- Complicated low-dimensional models
- High dimensional models ($N_h \geq N_v$)

Why this happens?

$$\theta(z, \Omega) := \sum_{n \in \mathbb{Z}^N} e^{2\pi i \left(\frac{1}{2} n^t \Omega n + n^t z\right)} \ .$$

The computation of the RT and its derivatives is computationally challenging due to the infinite sum over an N-dimensional integer lattice $\mathbb{Z}^N$.

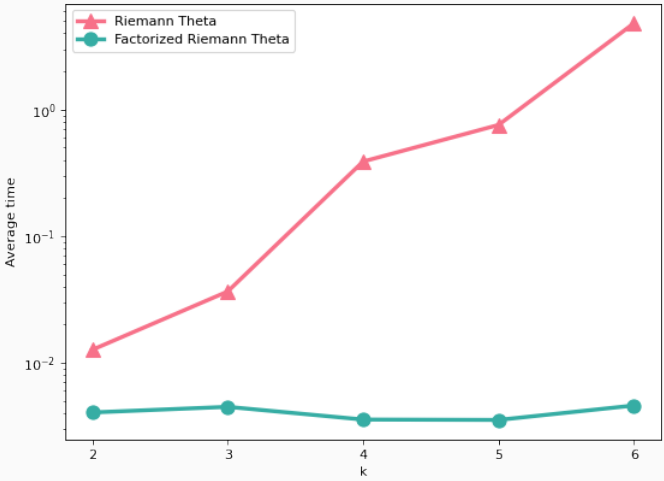
The computational times increase exponentially with N_h.

Improving the RTBM

The RT function has an interesting property that can become useful when dealing with large matrices. Lets consider the following RT function $\theta(z, \Omega)$, if we assume that Ω is diagonal the RT can be factorized as follows:

$$\theta(z, \Omega) = \prod_{i=1}^{n} \theta(z_i, \Omega_{ii}) \tag{1}$$

We have reduce the computation of a RT in n dimension to the evaluation of n one-dimensional RT functions!



Average time to compute the RT using Deconinck et al., 2002

Can we speed up the learning process just be evaluating $1d$-RT functions? Let's recall the expression of the probability distribution obtained by a RTBM:

$$P(v) = \sqrt{\frac{\det T}{(2\pi)^{N_v}}} e^{-\frac{1}{2}v^t T v - B_v^t - B_v^t T^{-1} B_v} \frac{\tilde{\theta}(B_h^t + v^t W|Q)}{\tilde{\theta}(B_h^t - B_v^t T^{-1} W|Q - W^t T^{-1} W)} .$$

We have to compute two RT function to evaluate $P(v)$

- $\tilde{\theta}(B_h^t + v^t W|Q)$ if Q diagonal $\longrightarrow$ *restricted RTBM*
- $\tilde{\theta}(B_h^t - B_v^t T^{-1} W|Q - W^t T^{-1} W)$ $Q - W^t T^{-1} W$ diagonal? *not feasible*

However, we can observe that the second term is a just for normalization.

Idea: Is there a learning process in which we can avoid computing the partition function for all system?

A particular parameter learning methodology that can address this issue is *Score Matching*, which is based on the Fisher divergence.

$$D_F(p||q_\theta) = \int p(x) \left| \frac{\nabla_x \, p(x)}{p(x)} - \frac{\nabla_x \, q(x,\theta)}{q(x,\theta)} \right|^2 dx \ ,$$

which slightly different from the Kullback-Leiber divergence:

$$D_{KL}(p||q_\theta) = \int p(x) \log \frac{p(x)}{q(x,\theta)} dx \ ,$$

In the score matching there is no need to evaluate the partition function since all the terms are of the from $\frac{\nabla_x \, p(x)}{p(x)}$ which leads to the cancellation of the normalizing terms.

Therefore we can get rid of the the non diagonal term in $P(v)$ during the learning process.

We can simplify the espression for D_F under the assumption that our model $q(x, \theta)$ is sufficiently regular, which is the case of the RTBM.

$$D_F(p||q_\theta) = \int p(x) \left(\left| \nabla_x \log q(x,\theta) \right|^2 + 2\Delta_x \log q(x,\theta) \right) + \text{const}$$

$$\approx \boxed{\sum_{i=1}^{N} \left| \nabla_{v_i} \log q(v_i,\theta) \right|^2 + 2\Delta_{v_i} \log q(v_i,\theta)} + \text{const}\,.$$

Fisher Cost function

We will now show how the Fisher cost function can be evaluated only by computing 1-dimensional RT functions.

$$\partial_{v_i} \log P(v) = -(Tv)_i - (B_v)_i + (WD)_i \,,$$
$$\partial^2_{v_i} \log P(v) = -T_{ii} + (WHW^t)_{ii} + (WD)^2_i \,,$$

with D the normalized gradient and H the normalized hessian

$$(D)_i = \frac{\nabla_i \tilde{\theta}(B^t_h + v^t W|Q)}{\tilde{\theta}(B^t_h + v^t W|Q)} \,, \quad (H)_{ij} = \frac{\nabla_i \nabla_j \tilde{\theta}(B^t_h + v^t W|Q)}{\tilde{\theta}(B^t_h + v^t W|Q)} \,.$$

If Q is diagonal

$$\partial_{v_i} \log P(v) = -(Tv)_i - (B_v)_i + \sum_{j=1}^{N_h} \frac{\partial_{v_i} \tilde{\theta}((B^t_h + v^t W)_j|Q_{jj})}{\tilde{\theta}((B^t_h + v^t W)_j|Q_{jj})} W_{ji} \,,$$

$$\begin{aligned}\partial^2_{v_i} \log P(v) = &- T_{ii} + \sum_{j=1}^{N_h} \frac{\partial^2_{v_i} \tilde{\theta}((B^t_h + v^t W)_j|Q_{jj})}{\tilde{\theta}((B^t_h + v^t W)_j|Q_{jj})} W^2_{ji} \\ &- \sum_{j=1}^{N_h} \frac{(\partial_{v_i} \tilde{\theta}((B^t_h + v^t W)_j|Q_{jj}))^2}{\tilde{\theta}((B^t_h + v^t W)_j|Q_{jj})} W_{ji} \,.\end{aligned}$$

The cost function can be evaluated using only 1d RT functions!

Applications

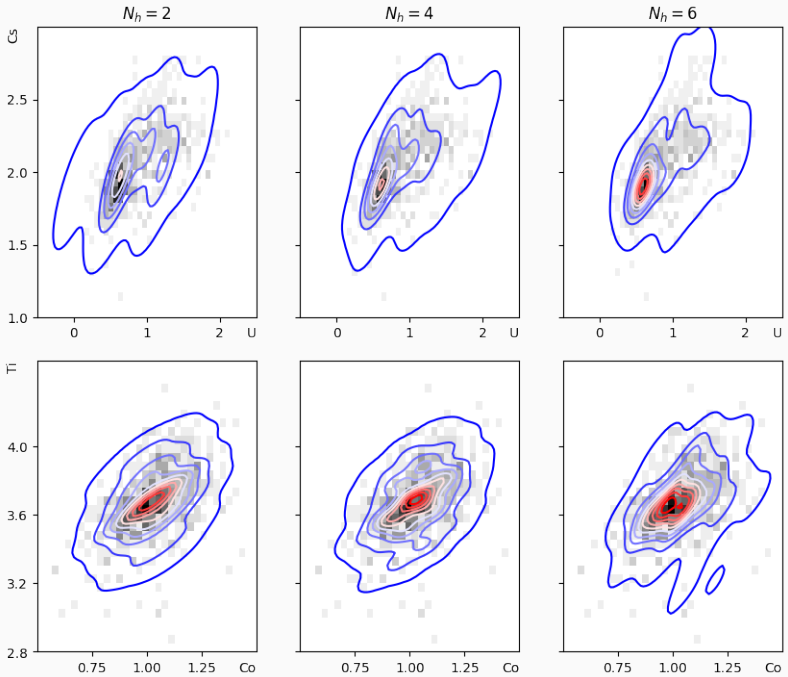
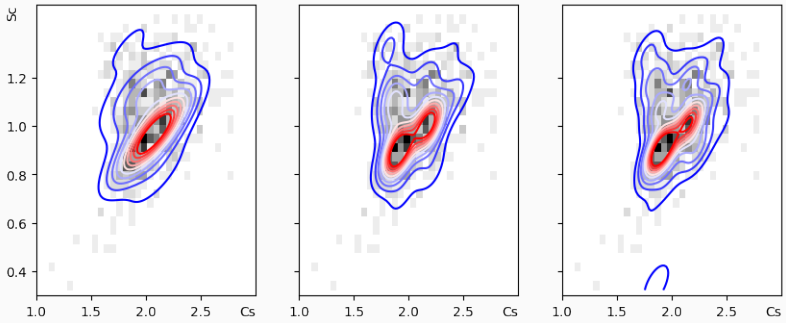


Figure 1: rRTBMs modelling the concentrations of Uranium and Cesium (first row), Cobalt and Titanium (second row) and, Cesium and Scandium (third row) for $N_h = 2, 4, 6$ (left,center,right). The rRTBM contours and histograms of the original data are shown.



Dataset	$\mathrm{KS}_{N_h=2}$	$\mathrm{KS}_{N_h=4}$	$\mathrm{KS}_{N_h=6}$	KS_{NF}	KS_{kde}
U Cs	0.114 ± 0.003	0.114± 0.003	**0.103** ± 0.002	0.124 ± 0.002	0.109 ± 0.001
Co Ti	0.099 ± 0.003	0.123 ± 0.002	**0.081** ± 0.002	0.089 ± 0.003	0.135 ± 0.004
Cs Sc	0.094 ± 0.002	**0.085** ± 0.001	0.201 ± 0.002	0.102 ± 0.002	0.094 ± 0.002

Table 3: KS distance between the rRTBM, normalizing flow (NF) and kernel density estimation (kde) models and the original data. For each model we averaged the KS distances for 10 independent samples of 5000 data points, and reported the mean and standard deviation. The lowest mean distance obtained for each dataset is printed in bold.

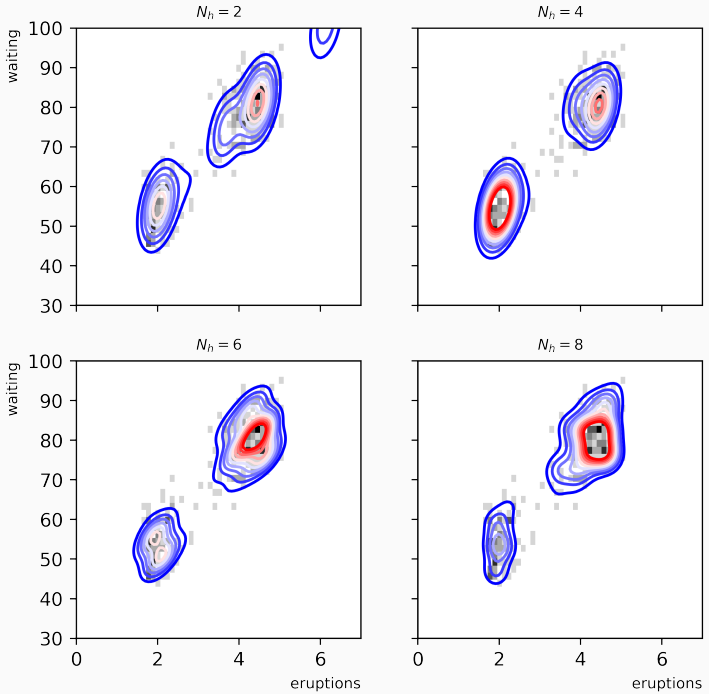


Figure 2: rRTBMs trained to model the waiting time between eruptions and the duration of the eruption for the Old Faithful geyser for $N_h = 2$ (top left), $N_h = 4$ (top right), $N_h = 6$ (bottom left) and $N_h = 8$ (bottom right). The curves correspond to the rRTBM model and the gray histogram is obtained from the original data with 30 bins.

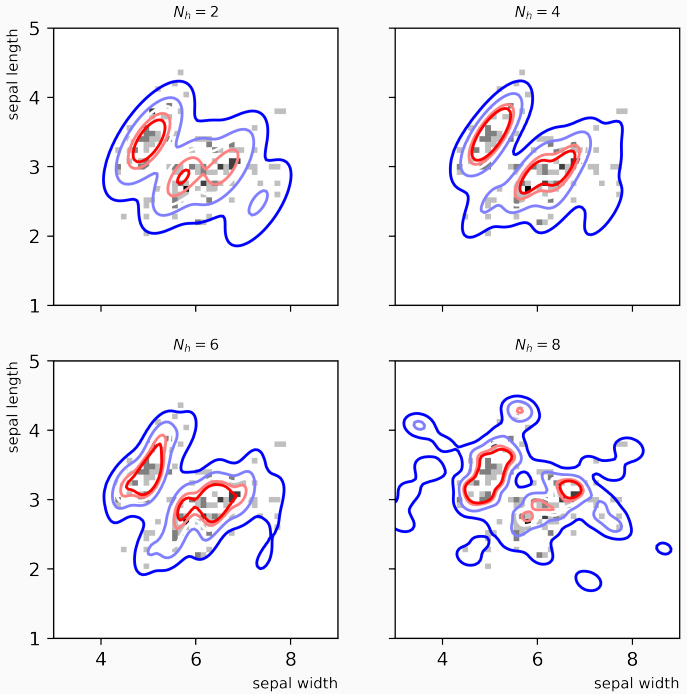


Figure 3: rRTBMs trained to model the joint distribution of sepal width and sepal length from the Iris dataset for $N_h = 2$ (top left), $N_h = 4$ (top right), $N_h = 6$ (bottom left) and $N_h = 8$ (bottom right). The curves correspond to the rRTBM model and the gray histogram is obtained from the original data with 30 bins.

Dataset	$\mathrm{KS}_{N_h=2}$	$\mathrm{KS}_{N_h=4}$	$\mathrm{KS}_{N_h=6}$	$\mathrm{KS}_{N_h=8}$	KS_{kde}	KS_{NF}
Iris	0.205 ± 0.003	0.202 ± 0.003	0.399 ± 0.003	0.333 ± 0.003	$\mathbf{0.173} \pm 0.003$	0.215 ± 0.002
Old Faithful	0.215 ± 0.003	0.196 ± 0.003	0.299 ± 0.003	0.253 ± 0.003	$\mathbf{0.151} \pm 0.001$	0.167 ± 0.001

Table 4: KS distance between the rRTBM, kernel density estimation (kde) and normalizing flow (NF) models and the original data. Reported values are averaged over independent runs, as in table 3.

The RTBM in this case does not reach a better perfomance but it is still competitive.

Conclusion

Outlook

In summary

- The (r)RTBM is a valid model to perform density estimation
- Using score matching we are able to train efficiently using large values of N_h
- Open source code soon available here: `https://github.com/RiemannAI/theta`

For the future

- Speed up the computation of the RT by moving to a GPU implementation
- Possibility to use this mechanism to perform MC multi-dimensional integration for physics related problem

Thanks for listening!

Appendix

The computation of the RT and its derivatives is computationally challenging due to the infinite sum over an N-dimensional integer lattice $\mathbb{Z}^N$

$$\theta(z,\Omega) := \sum_{n\in\mathbb{Z}^N} e^{2\pi i\left(\frac{1}{2}n^t\Omega n + n^t z\right)} \ .$$

For the multi-dimensional case we can obtain a numerical approximation by summing over an finite subset of lattice points.

For the 1d case there exist more efficient methods. A possibility is to truncate the series:

$$\theta(z,\Omega) \approx S_B(z,\Omega) = 1 + \sum_{0<n<B} q^{n^2}(e^{2\pi i n z} + e^{-2\pi i n z}) =: 1 + \sum_{0<n<B} v_n \, .$$

It can be shown (see Labrande, 2015) that v_n can be computed recursively, giving us a fast algorithm to evaluate the RT in dim 1.

$$v_{n+1} = q^{2n} v_1 v_n - q^{4n} v_{n-1} \, . \tag{2}$$

where $q = e^{i\pi\Omega}$.

What about the gradients?

[Pasquale et al., 2022 in preparation]

To speed up the learning process for the RTBM we would like to have a similar algorithm for the derivatives of the RT function. In our work he prove that this is possible:

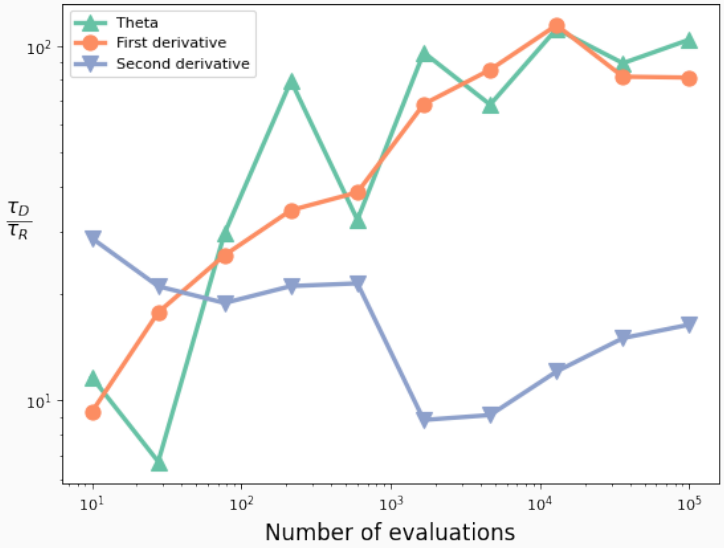
$$\frac{d}{dz}\theta(z,\Omega) \approx U_B(z,\Omega) = \sum_{1<n<B} -4\pi n q^{n^2} \sin(2\pi n z) =: \sum_{1<n<B} w_n \ ,$$

$$\frac{d^2}{dz^2}\theta(z,\Omega) \approx V_B(z,\Omega) = \sum_{1<n<B} -8\pi^2 n^2 q^{n^2} \cos(2\pi n z) =: \sum_{1<n<B} \xi_n \ .$$

After a few mathematical passages it can be shown that there exist a recurrence to compute both w_n and ξ_n.

$$w_{n+1} = (n+1)\left[\frac{2\cos(2\pi z)}{n} q^{2n+1} w_n - \frac{q^{4n}}{n-1} w_{n-1}\right],$$

$$\xi_{n+1} = (n+1)^2\left[\frac{2\cos(2\pi z)}{n^2} q^{2n+1} \xi_n - \frac{1}{(n-1)^2} q^{4n} \xi_{n-1}\right].$$


Theta
First derivative
Second derivative
10^2
10^1
$\frac{\tau_D}{\tau_R}$
10^1
10^2
10^3
10^4
10^5
Number of evaluations

References

- Carrazza, Stefano and Daniel Krefl (2018). "Sampling the Riemann-Theta Boltzmann Machine". In: *Comput. Phys. Commun.* 256, p. 107464. doi: 10.1016/j.cpc.2020.107464. arXiv: 1804.07768 [stat.ML].
- Deconinck, Bernard et al. (2002). *Computing Riemann Theta Functions*. arXiv: nlin/0206009 [nlin.SI].
- Krefl, Daniel et al. (May 2017). "Riemann-Theta Boltzmann machine". In: *Neurocomputing* 388, pp. 334–345. issn: 0925-2312. doi: 10.1016/j.neucom.2020.01.011. url: http://dx.doi.org/10.1016/j.neucom.2020.01.011.
- Labrande, Hugo (Nov. 2015). "Computing Jacobi's θ in quasi-linear time". In: *Mathematics of Computation* 87. doi: 10.1090/mcom/3245.
- Lyu, Siwei (2012). *Interpretation and Generalization of Score Matching*. arXiv: 1205.2629 [cs.LG].
- Pasquale et al., In preparation (2022). *Product Jacobi-Theta Boltzmann machines with score matching*.