Ecole Polytechnique Fédérale de Lausanne, IC department

Matrix estimation : Analytic evolution of gradient descent dynamics

Théo Damiani, Constantin de Bentzmann

17th June 2021

Abstract

In this report, you will find our semester project. For this project we were supervised by professor **Nicolas Macris** and doctoral student **Antoine Bodin**, both from LTHC laboratory.

Contents

1 Introduction

The project is an inference problem. Given a generated matrix we want to reconstruct its main components by minimizing a loss function by gradient descent. To avoid local minimum we add to the descent a perturbation in the form of a Brownian motion. In a first time we will implement a framework that will allow us in a second time to test different parameters' combinations on large generated data set over infinite time. This will allow us to confront theory and understand better how Brownian motion and stochastic differential equations (SDE) can help us on this particular problem.

2 Problem definition

2.1 Overview

Consider two vectors $u^*, v^* \sim U(S^{n-1})$ such that $(\|u^*\|^2 = n)$ and a rank one matrix Y such that: $Y_{i,j} = \sqrt{\frac{\lambda}{n}} u_i^* v_j^* + \xi_{i,j}$, where $\xi_{i,j} \sim N(0,1)$ and ξ is a symmetric matrix where all the upper-diagonal terms are independent to each other, that represents a Gaussian additive noise. We are interested in sampling the matrix Y according the Langevin dynamics, to reconstruct u^* and v^*. This is an inference problem. Our main tools for assessment and control will be the overlaps and the MSE. We define the overlaps as follows:

$q_u(t) = \frac{\langle u_t, u^* \rangle}{n}$ and $q_v(t) = \frac{\langle v_t, v^* \rangle}{n}$

The Langevin dynamics gives us:

$$du(t) = -\frac{1}{\lambda_1} \nabla_u \mathcal{H} dt - \frac{1}{\lambda_1} u(t) d\mu_u(t) + \sqrt{\frac{2}{\lambda_1 \beta_u}} dW_u(t)$$

With the following definitions:

- $\frac{1}{\lambda_u}$ is the learning rate of u,
- β_u corresponds to the inverse temperature of a bath,
- $\mathcal{H}(u,v) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} (Y_{ij} - \sqrt{\frac{\lambda}{N}} u_i v_j)^2$,
- $W_u(t)$ is a Brownian motion:

We called a Brownian motion with start in $x \in \mathcal{R}$, a real-valued stochastic process $\{B(t) : t \leq 0\}$ such that:

- $B(0) = x$,
- the process has independent increments: for all times $0 \leq t_1 \leq t_2 \leq ... \leq t_n$ we have that $B(t_n) - B(t_{n-1})$, $B(t_{n-1}) - B(t_{n-2})$, ... are independent random variables, each step is independent of the others,
- for all $t \geq 0$ and $h > 0$, the increments $B(t+h) - B(t)$ are normally distributed with expectation 0 and variance h,
- the function $t \to B(t)$ is continuous,

- So our $dW_u(t)$ represents a forward increment, a 'step' in a Brownian motion.
In the section **3.1 Discretization** we explain how we approximate it.

The first term $-\frac{1}{\lambda_1} \nabla_u \mathcal{H} dt$ is a classical gradient descent with the cost $\mathcal{H}$ that we want to minimize, then $-\frac{1}{\lambda_1} u(t) d\mu_u(t)$ imposes the constraint to stay on the sphere, it acts as a regularization term. The section **2.2 Equation Analysis** shows how we defined them. Finally $\sqrt{\frac{2}{\lambda_1 \beta_u}} dW_u(t)$ brings a notion of irregularity to the system like fluctuations.

Of course we have the same with $v(t)$: $dv(t) = \frac{1}{\lambda_2} \nabla_v \mathcal{H} dt - \frac{1}{\lambda_2} v(t) d\mu_v(t) + \sqrt{\frac{2}{\lambda_2 \beta_v}} dW_v(t)$

2.2 Equation Analysis

The idea here is to try to discretize the above equations such that we can use them in a gradient descent. We will assume and fix "Lagrange multipliers" as stochastic processes such that:

$$d\mu_u(t) = a_u(t)dt + b_u(t)\langle u(t), dW_u(t)\rangle$$
$$d\mu_v(t) = a_v(t)dt + b_v(t)\langle v(t), dW_v(t)\rangle$$

We know that:

$$d\big\|u(t)\big\|^2 = 2\langle u(t), du(t)\rangle + \langle du(t), du(t)\rangle$$

Indeed, $2\langle u(t), du(t)\rangle = -\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1}\langle u(t), u(t)\rangle d\mu_u(t) + 2\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle$
and $\langle du(t), du(t)\rangle = \frac{1}{\lambda_1^2}\langle u(t), u(t)\rangle d\mu_u(t)^2 + \frac{2}{\lambda_1\beta_u}\langle dW_u(t), dW_u(t)\rangle - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle d\mu_u(t)$
The others terms in $\langle du(t), du(t)\rangle$ such as $\frac{1}{\lambda_1}\langle \nabla_u \mathcal{H}, \nabla_u \mathcal{H}\rangle dt^2$ are too high in dt.

Thus we have:

$$\begin{aligned} d\big\|u(t)\big\|^2 = &-\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1}\langle u(t), u(t)\rangle d\mu_u(t) \\ &+2\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle + \frac{1}{\lambda_1^2}\langle u(t), u(t)\rangle d\mu_u(t)^2 \\ &+\frac{2}{\lambda_1\beta_u}\langle dW_u(t), dW_u(t)\rangle - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle d\mu_u(t) \end{aligned}$$

We use $\langle u(t), u(t)\rangle = \big\|u(t)\big\|^2$ and we replace $d\mu_u(t)$ with its definition:

$$\begin{aligned} d\big\|u(t)\big\|^2 = &-\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1}\big\|u(t)\big\|^2 (a_u(t)dt + b_u(t)\langle u(t), dW_u(t)\rangle) \\ &+ 2\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle + \frac{1}{\lambda_1^2}\big\|u(t)\big\|^2 (a_u(t)dt + b_u(t)\langle u(t), dW_u(t)\rangle)^2 \\ &+ \frac{2}{\lambda_1\beta_u}\langle dW_u(t), dW_u(t)\rangle - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle (a_u(t)dt + b_u(t)\langle u(t), dW_u(t)\rangle) \\ d\big\|u(t)\big\|^2 = &-\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1}\big\|u(t)\big\|^2 a_u(t)dt - \frac{2}{\lambda_1}\big\|u(t)\big\|^2 b_u(t)\langle u(t), dW_u(t)\rangle \\ &+ 2\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle + \frac{1}{\lambda_1^2}\big\|u(t)\big\|^2 (a_u(t)^2dt^2 + 2a_u(t)\cdot dt \cdot b_u(t)\langle u(t), dW_u(t)\rangle + b_u(t)^2\langle u(t), dW_u(t)\rangle^2) \\ &+ \frac{2}{\lambda_1\beta_u}\langle dW_u(t), dW_u(t)\rangle - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle a_u(t)dt - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}b_u(t)\langle u(t), dW_u(t)\rangle^2 \end{aligned}$$

We know that $dW_u(t)$ is defined such that $\approx \sqrt{dt}$ and it is a Gaussian random variable with zero mean and variance dt.
Besides, $\langle u(t), dW_u(t)\rangle^2 = \big\|u(t)\big\|^2 dt$ et $\langle dW_u(t), dW_u(t)\rangle = \sum \sqrt{dt}\sqrt{dt} = Ndt$
Thus, by removing the terms with a degree in dt too high, we obtain:

$$\begin{aligned} d\big\|u(t)\big\|^2 = &-\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1}\big\|u(t)\big\|^2 a_u(t)dt - \frac{2}{\lambda_1}\big\|u(t)\big\|^2 b_u(t)\langle u(t), dW_u(t)\rangle + 2\sqrt{\frac{2}{\lambda_1\beta_u}}\langle u(t), dW_u(t)\rangle \\ &+\frac{1}{\lambda_1^2}\big\|u(t)\big\|^2 b_u(t)^2\big\|u(t)\big\|^2 dt + \frac{2}{\lambda_1\beta_u}Ndt - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}b_u(t)\big\|u(t)\big\|^2 dt \end{aligned}$$

We can rewrite as:

$$\begin{aligned} d\big\|u(t)\big\|^2 = &-\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1}Na_u(t)dt + \frac{1}{\lambda_1^2}Nb_u(t)^2Ndt + \frac{2}{\lambda_1\beta_u}Ndt \\ &+\langle u(t), dW_u(t)\rangle(-\frac{2}{\lambda_1}Nb_u(t) + 2\sqrt{\frac{2}{\lambda_1\beta_u}}) - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}b_u(t)Ndt \end{aligned}$$

If we let $b_u(t) = \frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}}$, we have:

$$d\big\|u(t)\big\|^2 = -\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1} N a_u(t) dt + \frac{1}{\lambda_1^2} N (\frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}})^2 N dt$$
$$+ \frac{2}{\lambda_1 \beta_u} N dt + \langle u(t), dW_u(t)\rangle (-\frac{2}{\lambda_1} N \frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}} + 2\sqrt{\frac{2}{\lambda_1\beta_u}}) - \frac{2}{\lambda_1}\sqrt{\frac{2}{\lambda_1\beta_u}}\frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}} N dt$$
$$d\big\|u(t)\big\|^2 = -\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1} N a_u(t) dt + \frac{1}{\lambda_1^2} N \frac{1}{N^2}\frac{2\lambda_1}{\beta_u} N dt$$
$$+ \frac{2}{\lambda_1 \beta_u} N dt + \langle u(t), dW_u(t)\rangle (-2\sqrt{\frac{2}{\beta_u}}\frac{\sqrt{\lambda_1}}{\lambda_1} + 2\sqrt{\frac{2}{\beta_u}}\frac{1}{\sqrt{\lambda_1}}) - \frac{2}{\lambda_1}\sqrt{\frac{2\cdot 2\cdot \lambda_1}{\lambda_1 \beta_u^2}} dt$$

$$d\big\|u(t)\big\|^2 = -\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1} N a_u(t) dt + \frac{1}{\lambda_1}\frac{2}{\beta_u} dt$$
$$+ \frac{2}{\lambda_1 \beta_u} N dt + \langle u(t), dW_u(t)\rangle (-2\sqrt{\frac{2}{\beta_u}}\frac{\sqrt{\lambda_1}}{\lambda_1} + 2\sqrt{\frac{2}{\beta_u}}\frac{\sqrt{\lambda_1}}{\lambda_1}) - \frac{2}{\lambda_1}\sqrt{\frac{4}{\beta_u^2}} dt$$
$$d\big\|u(t)\big\|^2 = -\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1} N a_u(t) dt + \frac{2}{\lambda_1 \beta_u} dt + \frac{2}{\lambda_1 \beta_u} N dt - \frac{4}{\lambda_1 \beta_u} dt$$
$$d\big\|u(t)\big\|^2 = -\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1} N a_u(t) dt - \frac{2}{\lambda_1 \beta_u} dt + \frac{2}{\lambda_1 \beta_u} N dt$$

We want our vector u on the sphere of radius $\sqrt{N}$:
$\big\|u(t)\big\|^2 = N$ and $d\big\|u(t)\big\|^2 = 0$ for each instance

Therefore

$$0 = -\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt - \frac{2}{\lambda_1} N a_u(t) dt - \frac{2}{\lambda_1 \beta_u} dt + \frac{2}{\lambda_1 \beta_u} N dt$$

And we can find $a_u(t)$:

$$a_u(t) = -\frac{\lambda_1}{2N dt}(\frac{2}{\lambda_1}\langle u(t), \nabla_u \mathcal{H}\rangle dt + \frac{2}{\lambda_1 \beta_u} dt - \frac{2}{\lambda_1 \beta_u} N dt) = -\frac{1}{N}\langle u(t), \nabla_u \mathcal{H}\rangle - \frac{1}{\beta_u N} + \frac{1}{\beta_u}$$

Now we have:

$$b_u(t) = \frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}}$$
$$a_u(t) = -\frac{1}{N}\langle u(t), \nabla_u \mathcal{H}\rangle - \frac{1}{\beta_u N} + \frac{1}{\beta_u}$$

By the definition of $d\mu_u(t)$:

$$d\mu_u(t) = a_u(t) dt + b_u(t)\langle u(t), dW_u(t)\rangle$$
$$d\mu_u(t) = -\frac{1}{N}\langle u(t), \nabla_u \mathcal{H}\rangle dt + (\frac{1}{\beta_u} - \frac{1}{\beta_u N}) dt + \frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}}\langle u(t), dW_u(t)\rangle$$

With our starting stochastic differential equation, we find:

$$du(t) = -\frac{1}{\lambda_1}\nabla_u \mathcal{H} dt - \frac{1}{\lambda_1} u(t) d\mu_u(t) + \sqrt{\frac{2}{\lambda_1 \beta_u}} dW_u(t)$$
$$du(t) = -\frac{1}{\lambda_1}\nabla_u \mathcal{H} dt - \frac{1}{\lambda_1} u(t)(-\frac{1}{N}\langle u(t), \nabla_u \mathcal{H}\rangle dt + (\frac{1}{\beta_u} - \frac{1}{\beta_u N}) dt + \frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}}\langle u(t), dW_u(t)\rangle) + \sqrt{\frac{2}{\lambda_1 \beta_u}} dW_u(t)$$
$$du(t) = -\frac{1}{\lambda_1}\nabla_u \mathcal{H} dt + \frac{1}{\lambda_1}\frac{1}{N} u(t)\langle u(t), \nabla_u \mathcal{H}\rangle dt - (\frac{1}{\beta_u \lambda_1} - \frac{1}{\beta_u \lambda_1 N}) u(t) dt$$
$$- \frac{1}{\lambda_1}\frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}} u(t)\langle u(t), dW_u(t)\rangle) + \sqrt{\frac{2}{\lambda_1 \beta_u}} dW_u(t)$$

Using $\frac{1}{\lambda_1}\frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}} = \frac{1}{\sqrt{\lambda_1^2}}\frac{1}{N}\sqrt{\frac{2\lambda_1}{\beta_u}} = \frac{1}{N}\sqrt{\frac{2}{\lambda_1\beta_u}}$ and rearranging terms, we finally got:

$$du(t) = -\frac{1}{\lambda_1}(\mathbb{1} - \frac{u(t)u(t)^\top}{N})\nabla_u\mathcal{H}dt + \sqrt{\frac{2}{\lambda_1\beta_u}}(\mathbb{1} - \frac{u(t)u(t)^\top}{N})dW_u(t) - (\frac{N}{\beta_u\lambda_1 N} - \frac{1}{\beta_u\lambda_1 N})u(t)dt$$

$$du(t) = -\frac{1}{\lambda_1}(\mathbb{1} - \frac{u(t)u(t)^\top}{N})\nabla_u\mathcal{H}dt + \sqrt{\frac{2}{\lambda_1\beta_u}}(\mathbb{1} - \frac{u(t)u(t)^\top}{N})dW_u(t) - \frac{(N-1)}{N}\frac{1}{\beta_u\lambda_1}u(t)dt$$

3 Model Implementations

Our framework architecture is as follows:
Basic functions and definitions are implemented in the *util.py* file. You will find how we generate u^* and v^*, how we compute the overlaps, how we generate the matrix Y and compute the projector that helps us fix the vectors on the hypershpere during the descent. Finally we defined two versions of the gradients for u(t) and v(t).
Then, in the *gradient_descent.py* file we defined three gradient descents for each type of gradient, one using projections, one without projections and one without projections but with normalization at each step.
We ran the simulations on two different jupyter notebooks *simulation2D.ipynb* which displays MSEs or overlaps in functions of β_u and β_v for a certain learning rate ratio ($\frac{\lambda_u}{\lambda_v}$). The second one, *simulation3D.ipynb* compares two different simulations (different learning speed ratios) on the same graph.

3.1 Discretization

- We used the following $\mathcal{H}$ for the results we present:

$$\mathcal{H}(u,v) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}(Y_{ij} - \sqrt{\frac{\lambda}{N}}u_i v_j)^2$$

$$\nabla_u\mathcal{H}(u,v) = \begin{pmatrix} \frac{\partial\mathcal{H}(u_1,v)}{\partial u_1} \\ \vdots \\ \frac{\partial\mathcal{H}(u_N,v)}{\partial u_N} \end{pmatrix} \qquad \nabla_v\mathcal{H}(u,v) = \begin{pmatrix} \frac{\partial\mathcal{H}(u,v_1)}{\partial v_1} \\ \vdots \\ \frac{\partial\mathcal{H}(u,v_M)}{\partial v_M} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2}\sum_{j=1}^{M}(-2\sqrt{\frac{\lambda}{N}}v_j)(Y_{1j} - \sqrt{\frac{\lambda}{N}}u_1 v_j) \\ \vdots \\ \frac{1}{2}\sum_{j=1}^{M}(-2\sqrt{\frac{\lambda}{N}}v_j)(Y_{Nj} - \sqrt{\frac{\lambda}{N}}u_N v_j) \end{pmatrix} \qquad = \begin{pmatrix} \frac{1}{2}\sum_{i=1}^{N}(-2\sqrt{\frac{\lambda}{N}}u_i)(Y_{i1} - \sqrt{\frac{\lambda}{N}}u_i v_1) \\ \vdots \\ \frac{1}{2}\sum_{i=1}^{N}(-2\sqrt{\frac{\lambda}{N}}u_i)(Y_{iM} - \sqrt{\frac{\lambda}{N}}u_i v_M) \end{pmatrix}$$

$$= -\sqrt{\frac{\lambda}{N}}\begin{pmatrix} \sum_{j=1}^{M} v_j(Y_{1j} - \sqrt{\frac{\lambda}{N}}u_1 v_j) \\ \vdots \\ \sum_{j=1}^{M} v_j(Y_{Nj} - \sqrt{\frac{\lambda}{N}}u_N v_j) \end{pmatrix} \qquad = -\sqrt{\frac{\lambda}{N}}\begin{pmatrix} \sum_{i=1}^{N} u_i(Y_{i1} - \sqrt{\frac{\lambda}{N}}u_i v_1) \\ \vdots \\ \sum_{i=1}^{N} u_i(Y_{iM} - \sqrt{\frac{\lambda}{N}}u_i v_M) \end{pmatrix}$$

Our discretized gradient descent computed in the previous section :

$du(t) = -\frac{1}{\lambda_1}(\mathbb{1} - \frac{u(t)u(t)^\top}{N})\nabla_u\mathcal{H}dt + \sqrt{\frac{2}{\lambda_1\beta_1}}(\mathbb{1} - \frac{u(t)u(t)^\top}{N})dW_u(t) - \frac{N-1}{N\lambda_1\beta_1}u(t)dt$

With the following assumptions:
- $du(t) = u_{n+1} - u_n$

- $dt = h$, h est le pas de temps > 0, $t_{n+1} = t_n + h$
- $u(t_n) = u_n$
- $dW_u(t) \approx \mathcal{N}(0, h) = w$

Thus our algorithm computes at each step :
$u_{n+1} = u_n - \frac{1}{\lambda_1}(\mathbb{1} - \frac{u_n u_n^\top}{N})\nabla_u \mathcal{H}(u_n)h + \sqrt{\frac{2}{\lambda_1 \beta_1}}(\mathbb{1} - \frac{u_n u_n^\top}{N})w - \frac{N-1}{N\lambda_1\beta_1}u_n h$

followed by normalization.

3.2 Simulation choices : Loss and errors

- With $MSE = uv^T - u^* v^{*T}$:

$$\begin{aligned}\frac{\|MSE\|_F^2}{n_u n_v} &= \frac{1}{n_u n_v}\left\|uv^T - u^* v^{*T}\right\|_F^2 \\ &= \frac{1}{n_u n_v} Tr[(uv^T - u^* v^{*T})(vu^T - v^* u^{*T})] \\ &= \frac{1}{n_u n_v}(\|v\|^2\|u\|^2 + \|v*\|^2\|u*\|^2 - 2(u^T u^*)(v^T v^*)) \\ &= 2(1 - q_u q_v)\end{aligned}$$

This equality shows that this particular MSE and the overlaps shares a very close bound, which is why we used this MSE in some cases, to print the evolution of our descent. **This was never used as the loss**.

- With $MSE = Y - \sqrt{\frac{\lambda}{n}}uv^T$:

$$\begin{aligned}\frac{\|MSE\|_F^2}{n_u n_v} &= \frac{1}{n_u n_v}\left\|\sqrt{\frac{\lambda}{n}}(u^* v^{*T} - uv^T) + \xi\right\|_F^2 \\ &= \frac{\lambda}{n n_u n_v}\left\|uv^T - u^* v^{*T}\right\|_F^2 + 2\sqrt{\frac{\lambda}{n}}\frac{1}{n_u n_v} Tr[(uv^T - u^* v^{*T})\xi^T] + \frac{1}{n_u n_v}\|\xi\|_F^2\end{aligned}$$

Here, we look at the MSE that we minimize in the gradient descent. If we decompose it: first term tends towards 0. Second term typically tends to 0 when n tends to $+\infty$ when u and v are initialized at the beginning. Third term will typically tend towards 1 when n goes towards $+\infty$
To conclude : the third term is not of the same order as the first two and therefore the greater n will be, the more this mass risks being very concentrated in 1. Therefore is is not interesting for plots. We thus did not use this one for the results we present.

Important note : this issue disappears when we calculate its gradient with respect to u and v because the third term disappears since it is constant with respect to u and v, thus in the end the gradient is only calculated from the first two terms, and the second is a sort of correlation to the noise that we end up having because of the fact that we only have access to Y but not to u^* and v^* directly.
This is why as we stated in the previous section **we did use a variation of this one as our loss in the gradient descent**.

For the following section, we present results in function of three different error evaluation methods that we call MSE1, MSE2 and MSE3 defined as such:
$MSE1 = u - u^* + v - v^*$
$MSE2 = u^* v^{*T} - uv^T$
$MSE3 = Y - \sqrt{\frac{\lambda}{n}}uv^T$

However we did not use the MSE3 very much. As you can see in the following graph, it is not very meaningful (We can see by descending order of magnitude, MSE1, MSE2, MSE3). The problem is probably due to the fact that we compare Y to a perfect matrix $\sqrt{\frac{\lambda}{n}}uv^T$ without taking the noise into account. This results in the difference being distorted and probably mainly noise.

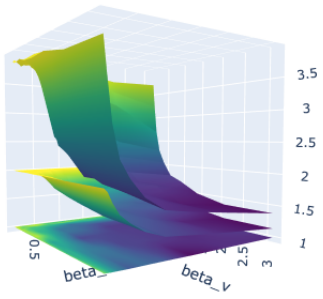

Figure 1: the 3 MSE for n/m = $\lambda_u/\lambda_v = 1$

4 Results

In this problem we have 8 degrees of freedom: $N, M, \lambda, \lambda_1, \lambda_2, \beta_u, \beta_v, dt$.
So to discover our implementation, we decided to run our script with the most simple configuration. We nullify the Brownian noise from the gradient descent, by making β_u and β_v tends to the infinity (in python we did with float("inf")). λ_1 and λ_2 are set to 1. The signal-to-noise ratio representing by λ is equal to 2, so that there is some noise in the generation of Y and the result doesn't converge too fast. N and M are set arbitrary to 1000.

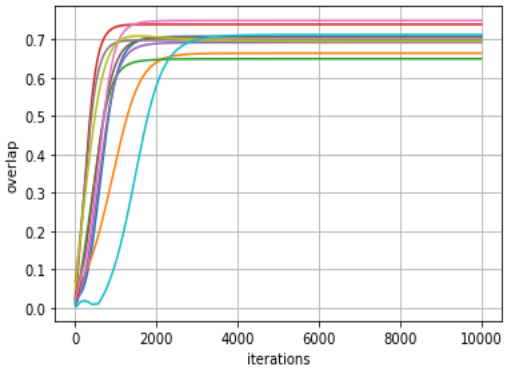

Figure 2: Overlap in function of the time.

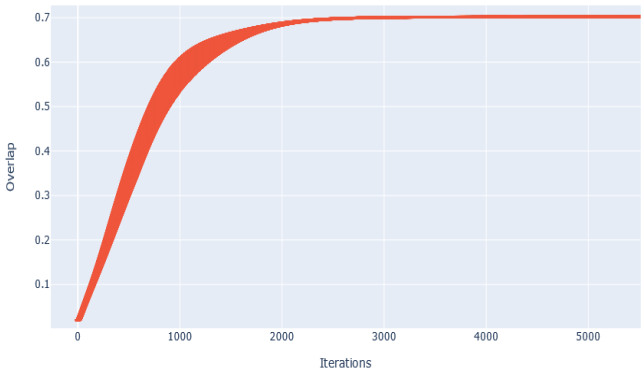

Figure 3: Average overlap over the 10 runs of the Figure 1.

For these 10 runs, we keep the same configuration but for each we generate a new Y and new starting vectors u and v. The algorithm is able to recover a good approximation of the original u^* and v^*: $q_u(t) \approx q_v(t) \approx 0.7$. Each runs seems to tends to the same asymptote. The variance (which is represented by the height of the curve in the Figure 2) is almost at 0.1 during the gradient descent, but after 2000 iterations, the curves stabilize and the variance decreases and becomes very low.

4.1 Influence of λ

Then we decided to keep the same configuration but we varied λ. We clearly see that when λ is big, it is very easy to get an overlap closed to 1. Indeed, the larger λ is and the less influential the noise is, in Y. We recall that $Y_{i,j} = \sqrt{\frac{\lambda}{n}} u_i^* v_j^* + \xi_{i,j}$, so when λ is small, $u_i^* v_j^*$ has a smaller "weight" than $\xi_{i,j}$ and so more noises are added to the generation of Y. Therefore the interesting values of λ are between 0 and 3:

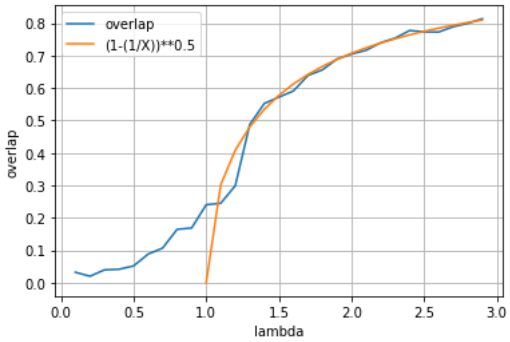

Figure 4: Average overlap in function of lambda

We can see that for $\lambda < 1$, it is very hard to reconstruct u^* and v^*. But with $\lambda \geq 1$, the overlap is following the function $f(y) = \sqrt{1 - \frac{1}{x}}$. So for the rest of our report, we made the decision to fix $\lambda = 2$. It is an arbitrary decision but it is good balance between having noise but still being able to recover u^* and v^*.

4.2 Influence of β

Then we wanted to look at the influence of β_u and β_v. So first we let $\beta_u = \beta_v$ and we varied it between the same range as we did with lambda. In the following graphics $\lambda = 2$.

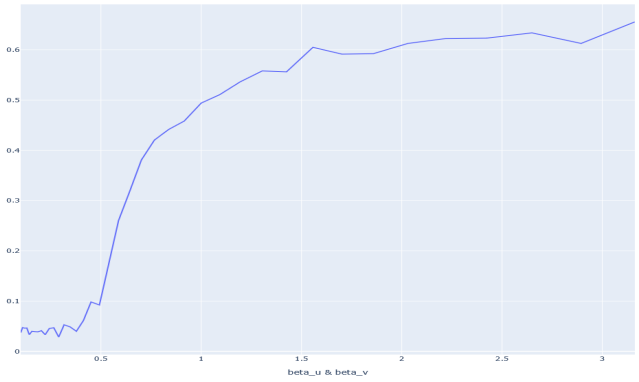

Figure 5: Overlap in function of $\beta_u = \beta_v$.

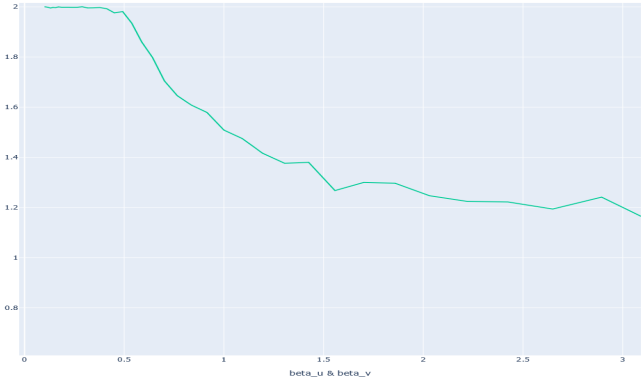

Figure 6: MSE in function of $\beta_u = \beta_v$.

We can observe the strong correlation between the overlap and the mean square error (here we used $MSE = \frac{1}{nm}\left\|u^*v^{*T} - uv^T\right\|_F^2$). For example for $\beta_u, \beta_v = 1$, we have an overlap ≈ 0.5 and for the MSE we find $\approx 1.5 = 2(1 - q_u \cdot q_v) = 2(1 - 0.5 \cdot 0.5)$ as we explained in the previous part (3.2.1).

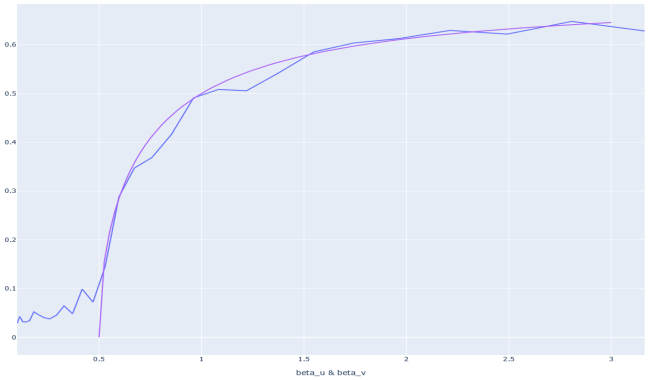

Figure 7: Overlap in function of $\beta_u = \beta_v$ with $\lambda = 2$.

A second interesting remark is the phase transition with a fixed λ that we can observe at $\beta = 0.5$. The Figure 6 is the same as the Figure 4, we just plot a new function in purple that follows the developpement of the overlap in blue. This purple function is defined by $0.7\sqrt{1 - \frac{0.5}{\beta}}$.

Then we decided to test with different values of λ:

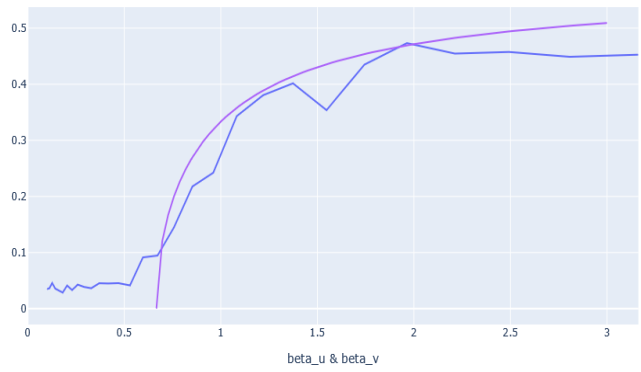

Figure 8: Overlap in function of $\beta_u = \beta_v$ with $\lambda = 1.5$.

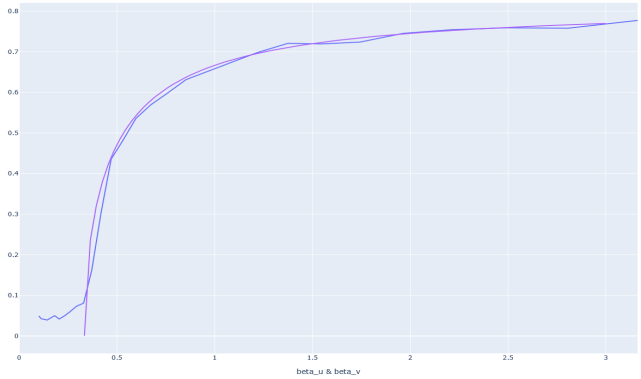

Figure 9: Overlap in function of $\beta_u = \beta_v$ with $\lambda = 3$.

In the Figure 7, $\lambda = 1.5$ and we can see that the transition is at $0.6666 = \frac{1}{1.5} = \frac{1}{\lambda}$. For the Figure 8 with $\lambda = 3$, the transition begins at $0.3333 = \frac{1}{3} = \frac{1}{\lambda}$.
So in these graphics, the purple curve is defined as $\sqrt{(1-\frac{1}{\lambda})}\sqrt{(1-\frac{1}{\lambda\beta})}$.
And we can see that the phase transition changes and is more likely represented by $\frac{1}{\lambda}$. So it is maybe possible that the evolution of the overlap in function of β_u and β_v (in the case that $\beta_u = \beta_v$) is following the function $\sqrt{(1-\frac{1}{\lambda})(1-\frac{1}{\lambda\beta})}$ with $\beta = \beta_u = \beta_v$.

4.3 Differences between β_u and β_v

As the problem has already been studied when β_u and β_v are equal, we wanted to see what happen with different value of β. So we implemented some contour plots of the overlap and the MSE while β_u and β_v are changing. We tested with $n = m = 500$, $\lambda = 2$ and $\lambda_1 = \lambda_2 = 1$:

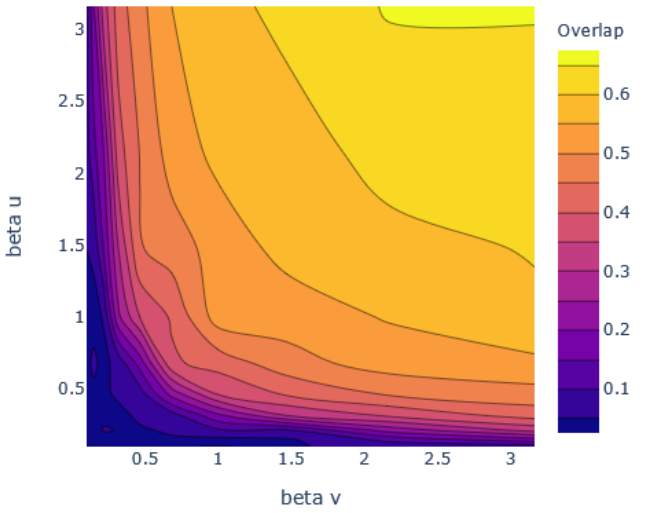

Figure 10: Contour plot of the overlap.

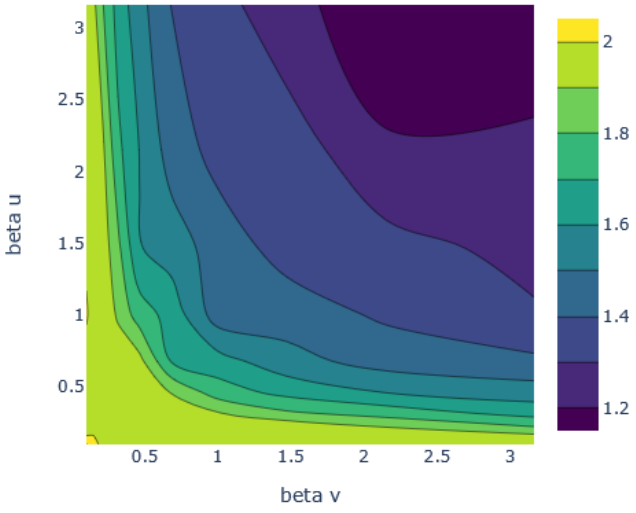

Figure 11: Contour plot of the 2nd MSE.

We can see that the graphs are symmetric along the diagonal (y=x with the x-axis horizontal and y-axis vertical). We could say that β_u and β_v have the same influence in the result, which seems logic as we are learning u and v at the same rate and they are of the same size. if we stay on this diagonal we can also recognize the Figure 4, with a phase transition at 0.5 where in the Figure 10, the contour lines are close to each other. To better understand and visualize these graphs we also provide a surface plot:

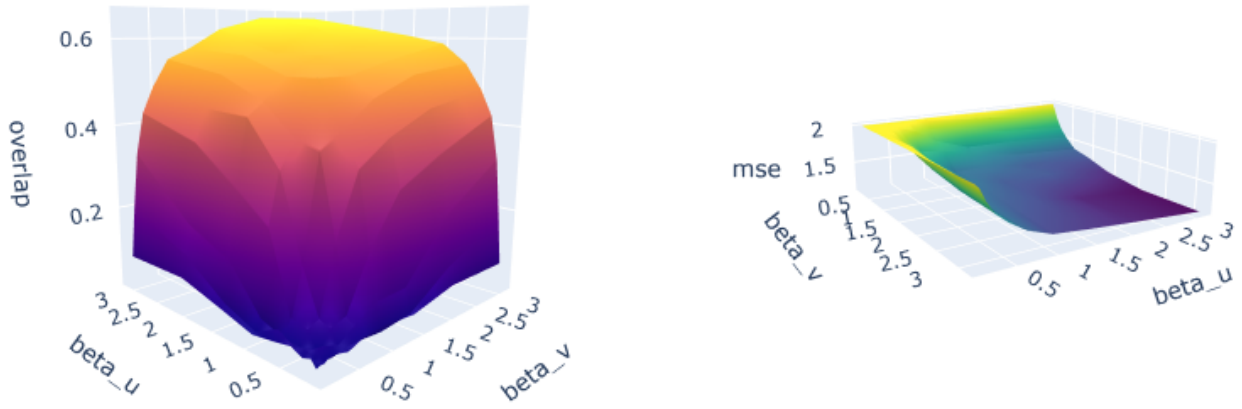

Figure 12: 3D surface plot.

These figures are in fact screenshots of a dynamic plot. You can open the figure11.html to test it yourself.

4.4 Differences between N and M

Then, we tried with different N and M. Here we have $N = 500, M = 1000, \lambda = 2$ and $\lambda_1 = \lambda_2 = 1$:

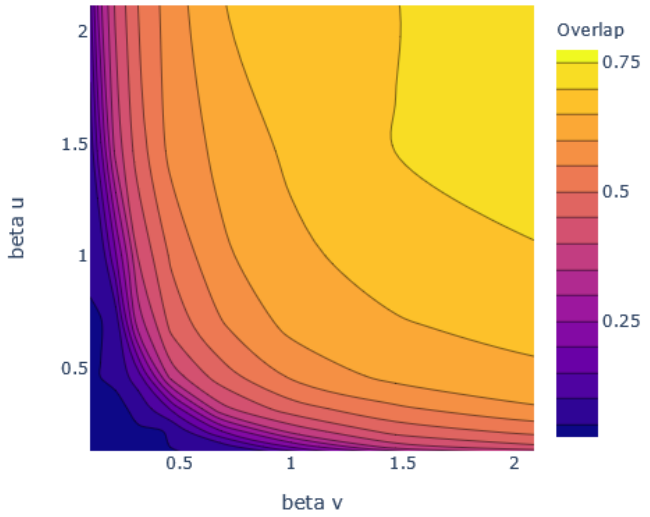

Figure 13: Contour plot of the overlap.

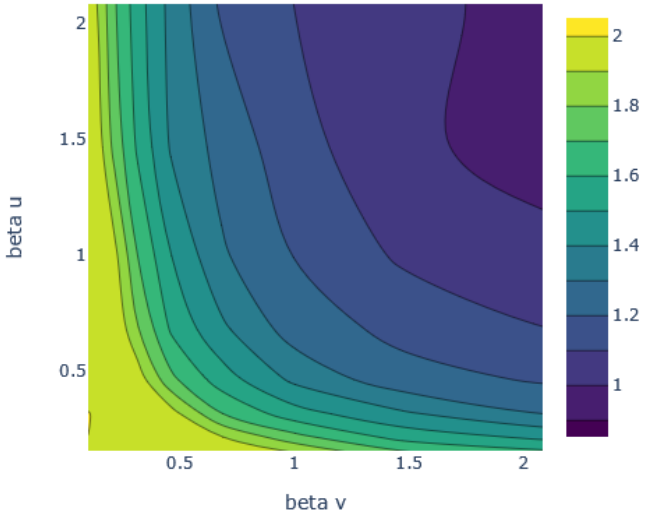

Figure 14: Contour plot of the 2nd MSE.

We can clearly see that the graphs are now asymmetric. We zoom in to the range 0-2 for β so it is easier to see the contour lines. But we are wondering if there is an advantage to take a different λ_1 and λ_2. With a different N, M, β_u and β_v, do we have a better result by learning one vector (u or v) faster than the other one?

4.5 Differences between λ_1 and λ_2

In this section, we will analyze some results we got with the following configuration:
$N = 200$, $M = 600$, $\lambda = 2$.
We took this specific N and M because we needed to have a significant difference between these sizes.But we couldn't take N or M too big because this slows down our algorithm and this configuration takes already 4 hours to run to completion.
First we tried to learn the vector v of size $M = 600$ slower than u of size $N = 200$:
We plotted along x and y axis the parameters β_u and β_v and the axis vertical z represents the overlap in the Figure 14 and the MSE in the Figure 15.

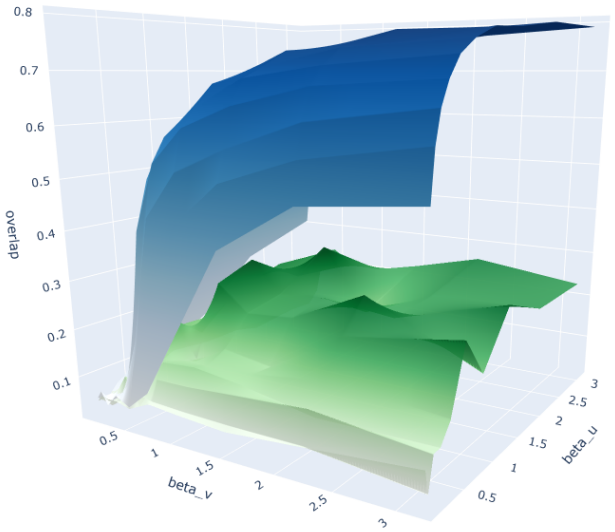

Figure 15: Surface plot of the overlap

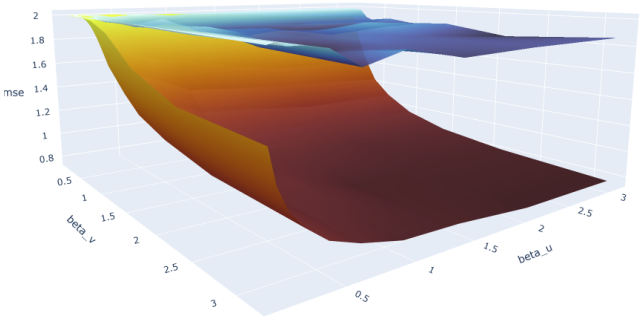

Figure 16: Surface plot of the 2nd MSE.

In the Figure 14, the surface in blue represent the overlap with $\lambda_1 = \lambda_2 = 1$ and the one in green is with $\lambda_1 = 1$ and $\lambda_2 = 100$. Obviously with $\lambda_2 = 100$ you are learning much slower so it is impossible to recover a good result and so the surface in green is always below the one in blue. We can observe the same thing in the Figure 15. The surface in blue is for $\lambda_1 = 1$ and $\lambda_2 = 100$ and in red/yellow with $\lambda_1 = \lambda_2 = 1$. Maybe we can get a better result if we do more and more iteration in the gradient descent but it was impossible for us to test it as it takes too much time run.

So we decided to give a small λ_2 so we would learning much faster the vector M (the vector with the highest dimension). The Figure 16 represent the same configuration as above in blue ($N = 200$, $M = 600$, $\lambda = 2$, $\lambda_1 = \lambda_2 = 1$), but in green we have: $N = 200$, $M = 600$, $\lambda = 2$, $\lambda_1 = 1$, $\lambda_2 = 0.01$.

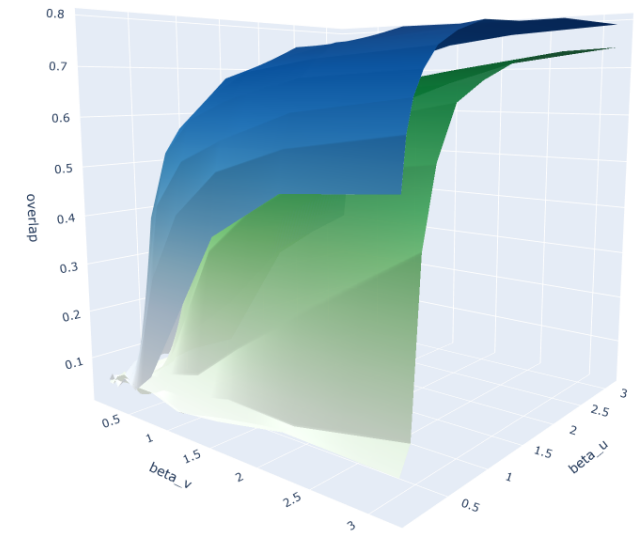

Figure 17: Surface plot of the overlap with $\lambda_1 = 1$, $\lambda_2 = 0.01$.

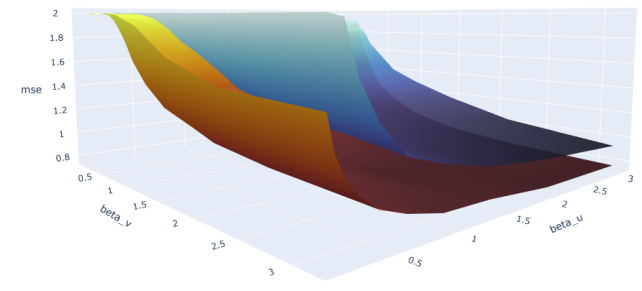

Figure 18: Surface plot of the 2nd MSE with $\lambda_1 = 1$, $\lambda_2 = 0.01$.

In the Figure 16, we can see that the surface in green is below the one in blue. And in the Figure 17, we can observe the same thing with surface in red/orange ($\lambda_1 = \lambda_2 = 1$). So we doesn't have a better result by learning at this rate.
The Figure 18 and 19 are different screenshots of the Figure 16. In the following figures we can observe the asymmetry we discussed in the previous section. In the Figure 18, if we look at the left side, where β_u is small and along the axis of β_v, we can see that the green surface has an offset with the blue surface. As β_u is small it is nearly impossible to make progress with u so we are more dependent of v. But as we are learning v much faster than u, λ_2 is small, the noise in the gradient descent from the Brownian motion has more influence

$(\sqrt{\frac{2}{\lambda_1 \beta_u}}(\mathbb{1} - \frac{u(t)u(t)^\top}{N})dW_u(t))$. So it is harder to recover a good overlap and maybe it is the reason of this offset. Maybe the step is also too big and the vector v don't converge in the wanted minima.
In the Figure 19, if we look at the right side of the graph when β_v is small (the inverse as previous), the surface in green follows the one in blue. If β_v is small, we have a bigger noise for v in the gradient descent and so we are more dependent of u but as $\lambda_1 = 1$ we can obtain some result.

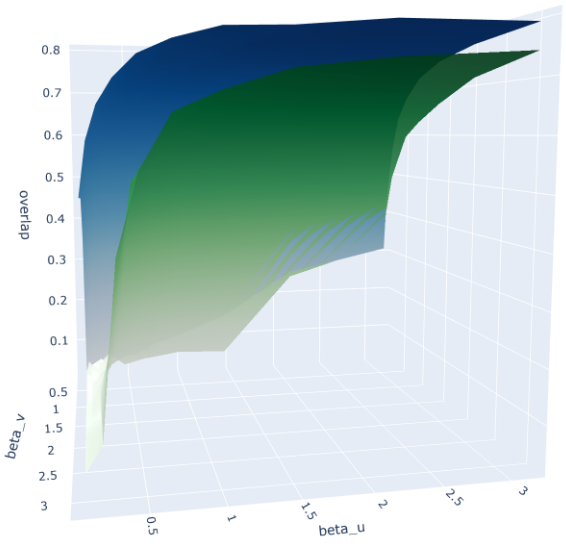

Figure 19: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.01$.

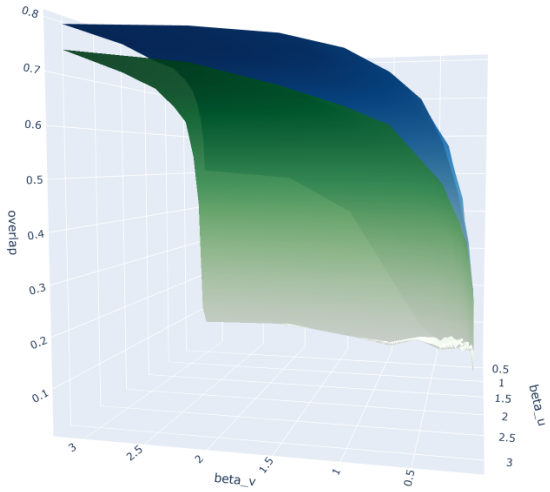

Figure 20: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.01$.

As taking a smaller λ_2 doesn't slow down our algorithm we decided to try with other values:

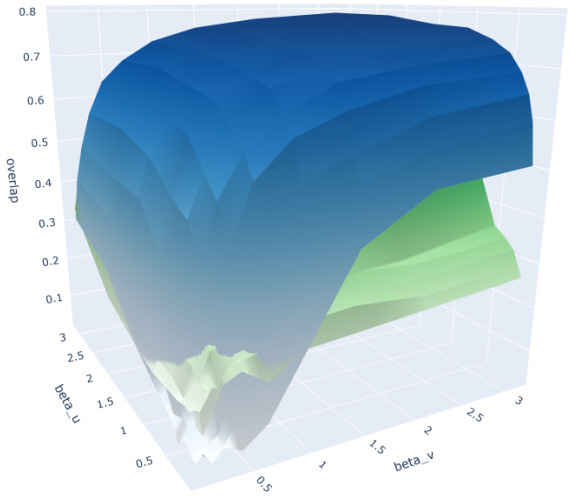

Figure 21: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.001$.

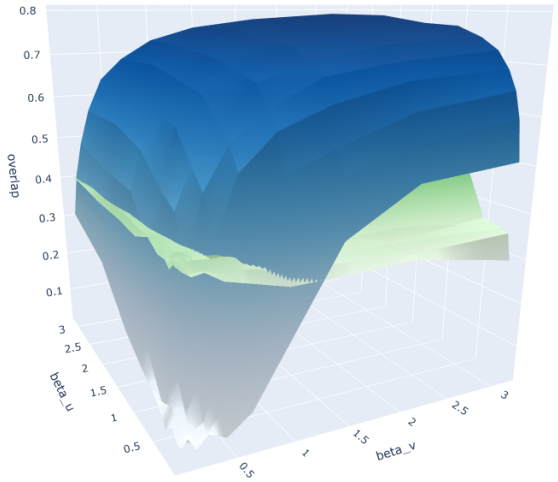

Figure 22: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.0001$.

We can see that when we are decreasing the value of λ_2 the surface in green (with this new λ_2) is above the surface in blue (which has $\lambda_2 = 1$) for small β_u and β_v. Therefore it seems to be better to learn a vector faster than the other one when β_u and β_v are very small.

To better understand this result we plot in the Figure 22, the overlap of u and v separately:

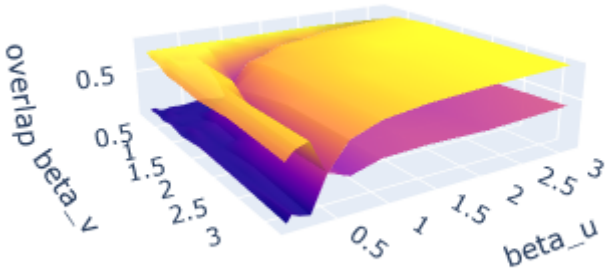

Figure 23: Overlap of u and v with $\lambda_1 = 1$, $\lambda_2 = 0.0001$.

We can observe that the overlap of v is high when β_u or $\beta_v < 0.5$, and decrease when β_u or β_v are bigger. The overlap of u is the opposite and they are both crossing each other when β_u or $\beta_v = 0.5$. The overlap of u has a natural evolution. The bigger β_u and β_v are, the overlap of u increase but the overlap of v is reaching is highest point when β_u or β_v are small.
In the Figure 23, we are showing the overlap of u in blue and v in red when β_u or β_v are small ($\approx < 0.5$). We can see that the overlap of v is increasing nicely until it reaches its limits and it stabilizes but the overlap of u stays always very noisy and is not able to make any progress.
In the Figure 24, we are again showing the overlap of u in blue and v in red but after they both crossed each other in the Figure 22 so when β_u and β_v are approximately > 0.5.

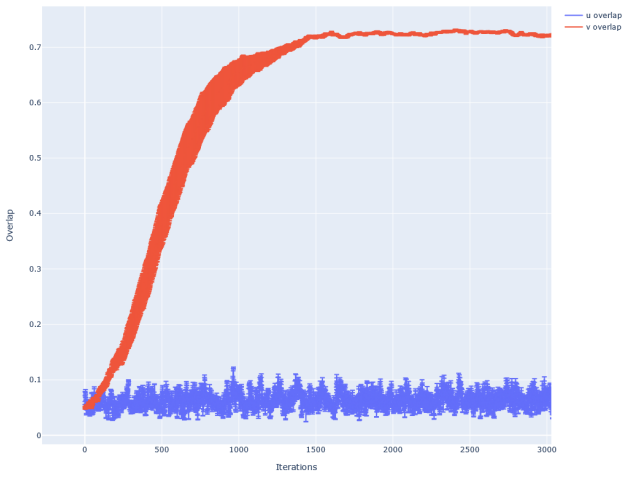

Figure 24: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.0001$.

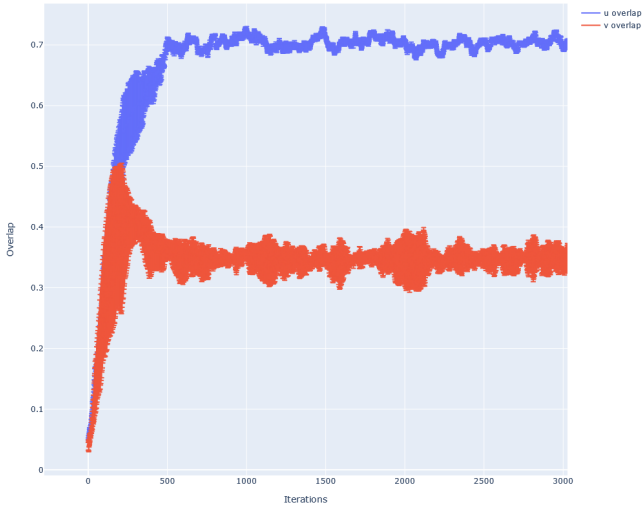

Figure 25: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.0001$.

We can observe that both overlap are increasing this time but after some time, the overlap of v is decreasing a little and stabilize.
We got this same result every time β_u and β_v are > 0.5. Here some others examples:

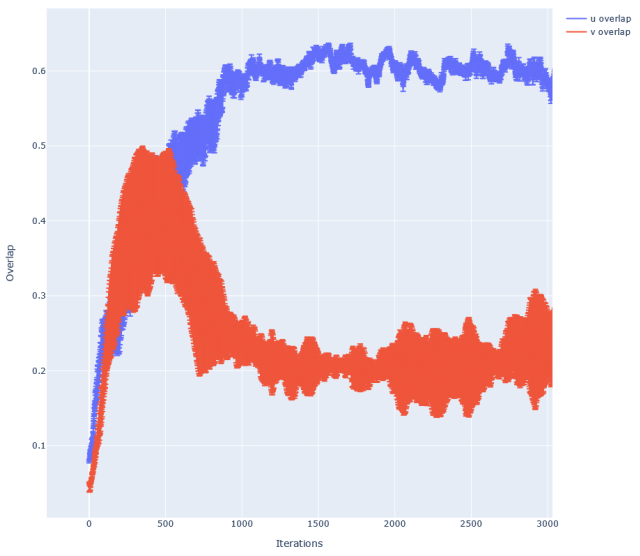

Figure 26: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.0001$.

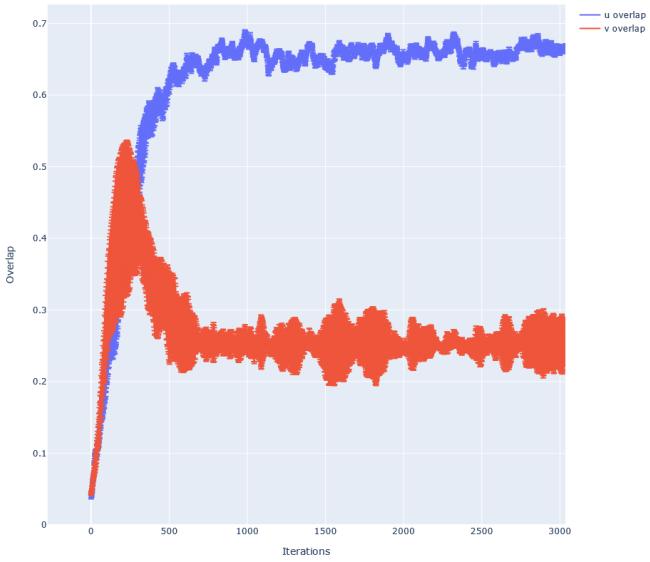

Figure 27: Overlap with $\lambda_1 = 1$, $\lambda_2 = 0.0001$.

Every time the overlap of u is reaching 0.5, we can see that the overlap of v starting decreasing before its stabilization. We could maybe say that when the βs are small the overlap of v is the only one making progress because its associated λ_2 is very small but when the βs are enough high so the overlap of u can make progress, the overlap of v is decreasing.

This phenomenon is not observable with the second or third MSE, as they are depending on $uv^\top$ and not a mean between the MSE of u and the MSE of v. For the overlap when we are showing only on overlap for a specific run we average the overlap of u and v. So if v is a good approximation but not u, the mean of the overlap of u and the overlap of v will be by half great but with $uv^\top$ this might not been the case. In the Figure 27, we are plotting the second MSE and indeed we can see that the surface in red/orange representing $\lambda_2 = 1$ is always below (perform better) than the other one in blue which is with $\lambda_2 = 0.0001$.

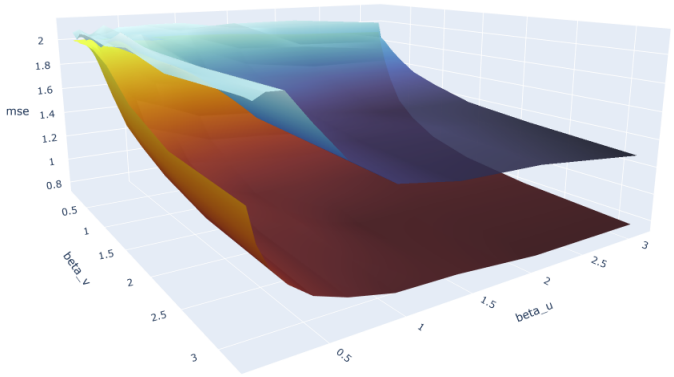

Figure 28: MSE with $\lambda_2 = 1$ (Red), $\lambda_2 = 0.0001$. (Blue)

5 Conclusion

In the previous section we tried to show the influence of the different parameters on the learning process. We notably tried different ratios of learning rates (λ_u, λ_v) combined with corresponding size ratios (N,M). However we went no further than a ratio of 10, which was already quite long to run. This ratio did not show any improvement in terms of learning, the MSE was symmetrical as well. It would be interesting to try combinations of ratios such as 100 or more.
The next step for this project would probably be to work on a more powerful machine. We had plenty of work to do on our own computers, but to go further and launch new simulations, with greater ratios, it would be more efficient on the cluster. Testing our implementation with greater ratios could help us better modelize the influence of learning rates (λs) and temperatures (βs) on the optimization. Each one of our simulations took between 2 and 5 hours. Another possible improvement would be to use the GPUs to make the calculations faster. Indeed, to keep control on our gradients we decided to hard code it and not use autograd. Now that we have tested the architecture and confronted to different checks, we can safely use this functionality.

This research project was a very interesting experience in the research world. We designed a framework that we used to test assumptions and search for different results through trial and error, under the regular supervision of our laboratory. We are very grateful to our professor Nicolas Macris for this opportunity and to doctoral student Antoine Bodin for his help.

References

[1] "Brownian Motion" Mars 2020
Peter Mörters and Yuval Peres

[2] "Rank-one matrix estimation: analytic time evolution of gradient descent dynamics"
author names withheld

[3] "Statistical limits of high-dimensional inference problems" 5 Mars 2021
Clément Dominique Luneau

[4] "Fundamental limits of inference : a statistical physics approach", 2019
Léo Miolane