# - Learning factors for stock market returns prediction -

Raphaël Maillet
(Dated: February 12, 2025)

This document focuses on predicting short-term daily average stock market values over a three-year period. It presents a comparative analysis of two primary data-fitting approaches. The first approach uses various gradient descent techniques on the Stiefel manifold, leveraging its geometric properties for parameter estimation. The second approach relies on a more traditional method using classical regression to estimate both the parameters $\beta$ and $A$. The study aims to evaluate the effectiveness and precision of these methods in capturing the underlying patterns of stock market behavior during the specified timeframe.

## I. INTRODUCTION

### A. Description of the challenge

What we want to do is to learn factors that help to predict future returns. We observe returns for $N = 250$ companies over three years, which makes $M = 754$ days. At each time $t \in \{1, \dots, M\}$, we will use the following model:

$$R_{t+1} = \sum_{\ell=1}^{F} \beta_\ell F_{t,\ell} + \varepsilon_t, \tag{1}$$

where $F = 10$, $\beta \in \mathbb{R}^F$. We want to learn the factors $F_t := [F_{t,1}, \dots, F_{t,F}]$. However, we only consider the following linear family of factors:

$$F_{t,\ell} = \sum_{m=1}^{D} A_{m,\ell} R_{t+1-m},$$

where $D = 250$, the maximum depth. This can be written

$$F_{t,\ell} = A_\ell R_t^{t+1-D},$$

where $R_t^{t+1-D} = [R_{t+1-D}, \dots, R_t]$, ensuring that we only use the information available at time $t$. Moreover, there is a non-linear constraint, ensuring that the matrix $A$ does not contain redundant information. More precisely, we impose that

$$A_\ell \cdot A_k = \delta_{\ell k},$$

for any $\ell, k$.

The goal here is not only to predict the returns but to find a model, $i.e$ to be able to predict good factors for the estimation of $\beta$ and the predictions of returns for other stocks.

### B. Organisation of the Note

This note is structured as follows. We begin by describing the problem and formulating it as a constrained optimization problem on the Stiefel manifold. Specifically, we aim to maximize a cost function subject to the constraint that the matrix $A$ lies in the Stiefel manifold, which imposes orthonormality on its columns.

Next, we present a detailed study of four optimization methods for solving this problem. Two of these methods are based on classical optimization theory. The remaining two methods leverage the intrinsic geometry of the Stiefel manifold. Following this, we introduce an alternative perspective by reformulating the problem as a regression model. Finally, we compare the different methods in terms of computational efficiency and stability. Based on this analysis, we discuss the trade-offs involved and justify the selection of the final model.

This report is accompanied by a Jupyter notebook containing the implementation of all the functions used to solve this challenge. Some functions are still present in the document but are not used, as I chose to structure the report in a way that ensures each method is applied to the same training and test dataset (see Section-section III for more details).

## II. THE OPTIMISATION PROBLEM

We can reformulate the problem in two steps. First, we aim to compute

$$F_t = \varphi_t(A) := A R_t^{t+1-D},$$

which allows us to rewrite model (1) as:

$$R_{t+1} = \beta \cdot \varphi_t(A) + \varepsilon_t.$$

Next, we define:

$$\widetilde{S}_t(A, \beta) = \beta \cdot \varphi_t(A),$$

and rewrite the loss function as:

$$\mathcal{L}(A, \beta) = \frac{1}{M - D} \sum_{t=D}^{M} \frac{\langle \widetilde{S}_t(A, \beta), \widetilde{R}_t \rangle}{|\widetilde{S}_t(A, \beta)||\widetilde{R}_t|}, \tag{2}$$

where $\widetilde{R}_t$ represents the actual returns. This leads to an optimization problem on the Stiefel manifold:

$$\max_{(A,\beta) \in \mathcal{V}_{D \times F} \times \mathbb{R}^F} \mathcal{L}(A, \beta),$$

where

$$\mathcal{V}_{D\times F} := \{A \in \mathcal{M}_{D\times F}, \text{s.t } A^*A = \mathrm{I}_F\}\,.$$

## III. GRADIENT DESCENT ON THE STIEFEL MANIFOLD

In this section, we delve into four distinct approaches to performing gradient descent on the Stiefel manifold denoted as $\mathcal{V}_{D\times F}$. This geometric structure requires specialized optimization techniques to ensure that updates remain within the manifold constraint.

The four methods we consider are the QR decomposition/Projection method, the Lagrangian approach, the Cayley transform, and the matrix exponential method. Each of these techniques maintains the orthonormality of the columns of $A$ through different strategies:

- The **Projection method** uses orthogonalization via QR factorization after each gradient step to project the updated matrix back onto the manifold.
- The **Lagrangian approach** introduces a penalty term or constraint directly into the optimization process to enforce orthonormality during updates.
- The **Cayley transform** leverages a parameterization of the Stiefel manifold.
- The **Matrix exponential method** applies the exponential map to generate updates that remain on the manifold by ensuring orthogonality is preserved, this is inspired by the study of geodesics on the manifold.

Before delving into the details of each method, let us describe the general setup of the training process. The training loop is implemented in PyTorch to leverage automatic differentiation, which greatly simplifies gradient computations. For each training iteration, the dataset was divided into a training set and a test set. To align with the evaluation task, we separated the data by randomly selecting $k = 5$ stocks (10% of the dataset) to form the test set, leaving the remaining stocks in the training set. This approach mirrors the evaluation on the public dataset, which covers the same time period but uses different stocks for testing.

Additionally, for time series data, another natural strategy is to remove specific time periods at random. This variation is discussed further in Section IV and may provide alternative insights into the robustness of the models. To prevent overfitting, we incorporated $L1$ regularization into some models and performed cross-validation. The cross-validation process, conducted by repeatedly splitting stocks into training and validation sets, helped us select the best-performing model. For optimization efficiency and convergence, we used the Adam optimizer, fine-tuning the learning rate for better results.

### A. The Projection method

The approach is conceptually straightforward: we aim to maximize the function $\mathcal{L}$ defined in Equation (2) by performing gradient descent with respect to the parameters $A$ and $\beta$. To ensure that the updated matrix $A$ remains on the Stiefel manifold $\mathcal{V}_{D\times F}$, we project $A$ onto the manifold after each gradient descent step using the QR decomposition. Specifically, we write:

$$A = QR, \tag{3}$$

where $Q \in \mathbb{R}^{D\times F}$ satisfies $Q^*Q = \mathrm{Id}$ (i.e., the columns of $Q$ are orthonormal) and $R \in \mathbb{R}^{F\times F}$ is an upper triangular matrix. The projection of $A$ onto the Stiefel manifold $\mathcal{V}_{D\times F}$ is then simply the matrix $Q$.

This method is both intuitive and easy to implement, as it directly enforces the orthonormality constraint through the QR decomposition. However, a key limitation lies in its inefficiency: the approach does not exploit any specific structure or sparsity in the matrix $A$, resulting in the need to estimate a large number of parameters. Moreover, the convergence to the maximum is not monotonic at all because of the projection steps which does not ensure that we are still going into the direction of the gradient.

### B. The Lagragian method

This idea is highly intuitive and derives from classical optimization theory. The main concept is to modify the cost function $\mathcal{L}$ by introducing a penalization term that discourages deviations of the matrix $A$ from the desired manifold. Specifically, we define the augmented cost function as:

$$\mathcal{H}(A, \beta, \lambda) = \mathcal{L}(A, \beta) - \lambda\|A^*A - \mathrm{Id}\|^2, \tag{4}$$

where $\|\cdot\|$ denotes the Frobenius norm on the space of matrices. The penalization term $-\lambda\|A^*A - \mathrm{Id}\|^2$ ensures that the orthonormality of the columns of $A$ is preserved as closely as possible by penalizing deviations from the identity matrix.

In this framework, we once again perform gradient descent, but with an additional parameter $\lambda \in \mathbb{R}$ that needs to be estimated. The parameter $\lambda$ controls the strength of the penalization, balancing the original objective and the enforcement of orthonormality. This method is appealing and performs well when dealing with smooth constraints. However, despite its advantages, it requires particular caution in its application. Compared to other methods, this approach is more prone to challenges arising from the optimization landscape. Specifically, when using gradient descent on the function $\mathcal{H}$, there is a significant risk that the algorithm may converge to saddle points of $\mathcal{H}$, which are not maxima of the original function $\mathcal{L}$ under the given constraints.

This issue highlights a critical limitation: the added penalization term does not inherently guarantee that the optimization process will lead to the correct solutions that satisfy both the manifold constraint and the original optimization objective. There is a real issue with stability with this method. Moreover, we need to be careful at the end of the training and check that the retain model indeed satisfies the constraint.

### C. The Cayley transform

The following method is less intuitive and leverages the geometry of the Stiefel manifold. As established in the literature on optimization over the Stiefel manifold, the Cayley transform is a tool that computes a parametric curve on the manifold using a skew-symmetric matrix; see, for instance, [2, 4]. In this section, we do not directly use optimization methods that follow these parametric curves. However, the Cayley transform provides a parametrization of the Stiefel manifold. Specifically, for any $A \in \mathcal{V}_{D\times F}$, we can express it as:

$$A = \left[\mathrm{Id} - \frac{\eta}{2}\Delta\right]^{-1}\left[\mathrm{Id} + \frac{\eta}{2}\Delta\right] X,$$

where $X \in \mathcal{V}_{D\times F}$, $\Delta$ is a skew-symmetric matrix, and $\eta > 0$ is a step-size parameter.

The key idea we propose here is to use this parametrization to perform gradient descent directly on the space of skew-symmetric matrices.

This approach is particularly interesting because it significantly reduces the dimensionality of the optimization problem. Instead of optimizing over $D^2$ parameters for $A$, we reduce the number of parameters to $D(D-1)/2 + 1$, corresponding to the degrees of freedom of skew-symmetric matrices plus a single step-size parameter. This reduction makes the method computationally efficient while still respecting the geometric constraints of the Stiefel manifold, avoiding the projection step. Even if the computation of the inverse matrix might be long (which is not the case here), we can think of methods to overcome this difficulty that were not implemented here (see [2])

### D. An exponential mapping

One effective way to study the geometry of a manifold is by understanding its geodesics. To compute geodesics, one might initially consider defining a simple metric on the tangent space, such as:

$$\langle \Delta_1, \Delta_2 \rangle = \mathrm{Tr}(\Delta_1^* \Delta_2),$$

for any $\Delta_1, \Delta_2 \in \mathcal{T}_A\mathcal{V}_{D\times F}$, where $\mathcal{T}_A\mathcal{V}_{D\times F}$ stands for the tangent space located at $A$. However, this definition does not account for the specific geometry at a given point $A$. To address this, we follow the approach outlined in [3, Section 2.4.1] and define the *canonical inner product* on the Stiefel manifold:

$$g_A(\Delta_1, \Delta_2) = \mathrm{Tr}\left[\Delta_1^*\left(\mathrm{Id} - \frac{1}{2}AA^*\right)\Delta_2\right].$$

This canonical inner product respects the geometry of the Stiefel manifold and provides a solid foundation for computing geodesics. Using this framework, one can prove that the geodesic paths on the Stiefel manifold are given by:

$$Y(t) = \exp\left(t\begin{bmatrix}\Omega & -K^* \\ K & 0_{D-F}\end{bmatrix}\right)\begin{bmatrix}I_F \\ 0\end{bmatrix}.$$

As in the previous section, this formulation provides clear charts for parametrizing the Stiefel manifold. Specifically, we consider $\Omega$ and $K$ as the parameters to optimize. The matrix $A$ is then constructed as:

$$A = \exp\left(\begin{bmatrix}\Omega & -K^* \\ K & 0_{D-F}\end{bmatrix}\right)\begin{bmatrix}I_F \\ 0\end{bmatrix}.$$

This method is similar to the previous one but modifies the number of parameters to be estimated, focusing solely on $\Omega$ which is a skew-symmetric matrix and $K$.

A variant which is a little bit more simple to implement has also been tested by considering

$$A = \exp(O)\begin{bmatrix}I_F \\ 0\end{bmatrix},$$

where $O$ is a skew-symmetric matrix.

This method heavily leverages the geometry of the Stiefel manifold, making it particularly interesting. However, its training time is longer than other methods due to the need to compute the matrix exponential at each gradient descent step.

## IV. THE REGRESSION APPROACH

Another way to approach the problem, which is quite different, is to formulate it as a simple regression problem in a single step. Specifically, one can rewrite the initial problem as:

$$R_{t+1} = \vartheta R_t^{t+1-D} + \varepsilon_t,$$

where $\vartheta \in \mathbb{R}^D$ is a vector of parameters. This allows for the following interpretation: $\vartheta = A\beta$.

A simple way to construct $A$ and $\beta$ that satisfy this equality is to generate a matrix $A$ uniformly on the Stiefel manifold $\mathcal{V}_{D\times F}$. This can be achieved, for example, by generating normally distributed coefficients and then applying the Gram-Schmidt algorithm. Once $A$ is generated, $\beta$ can be computed as $\beta = A^*\vartheta$.

At this point, the problem is reduced to an unconstrained optimization problem, which can be solved using classical regression techniques. This approach is more brute-force, but it yields more results and significantly reduces the number of parameters to estimate. However, it is not particularly natural within the context of the challenge, as it does not allow for an interpretable decomposition of the factors.

I did not explore this direction extensively, as I found a relevant paper [1] that uses this strategy to address the challenge. Although their method for reconstructing the matrix $A$ and vector $\beta$ after regression differs slightly, the overall approach is similar. The authors study various dimension-reduction techniques and cross-validation strategies. They conclude that the best-performing model is classical regression together with L1 regularization. I implemented this model for completeness but did not reimplement the other techniques discussed in the paper.

## V. RESULTS

The five implemented methods successfully maximize our function of interest and generate a model (i.e., $A \in \mathcal{V}_{D\times F}$ and $\beta \in \mathbb{R}^F$). However, as highlighted in previous sections, some methods face greater difficulties in convergence (e.g., the projection method) or exhibit lower stability (e.g., the Lagrangian method).

In my opinion, the best approach to reducing the number of parameters while ensuring both convergence and stability is to leverage the geometry of the underlying manifold. By doing so, we can identify an optimal path on the manifold of interest, significantly reducing the number of parameters to estimate. The exponential method achieves this perfectly, and it was the approach I used to obtain my best score on the public leaderboard: **0.0854**. The only drawback of this method is that it requires significantly more training time due to the computation of the matrix exponential at each training iteration.

Similarly, the Cayley transform method also exploits the geometry of the Stiefel manifold and achieves competitive results on the public leaderboard with a score of **0.0838**. Although slightly less performant, it is computationally faster since it does not require the computation of a matrix exponential. While matrix inversion can sometimes be computationally expensive, this is not a major issue in our case. Moreover, efficient numerical techniques can further accelerate matrix inversion or circumvent the problem (see, for example, iterative Cayley transform [2, Section 3.2.1]).

## VI. CONCLUSION

In this note, we presented several models designed to address the QRT data challenge. Our primary focus was on understanding the problem and proposing modeling solutions. While we performed some parameter tuning, it was not fully optimized due to the daily submission limit.

As highlighted, preventing overfitting is crucial given the two-step approach. Although we took steps to reduce overfitting by incorporating regularization and selecting the best-performing model on the test set at each step—a strategy akin to early stopping—there is room for improvement. For instance, we could further refine the regularization parameters or explore additional techniques, such as dropout.

To enhance generalization to unseen data, we implemented a cross-validation method tailored to this challenge. Specifically, we adapted the validation set design by randomly selecting stock indexes, reflecting the test design. However, since the ultimate goal of the challenge is to predict future returns, it may be beneficial to explore cross-validation techniques that account for temporal structures, similar to those proposed by [1].


[1] M'ghari, Mouad and Oneci, Codrin and Chen, Curtis, *Learning factors for stock market returns prediction*, 2022.

[2] Li, Jun, Fuxin Li, and Sinisa Todorovic, *Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform*, International Conference on Learning Representations, 2020.

[3] Edelman, Alan, Tomás A. Arias, and Steven T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM journal on Matrix Analysis and Applications, 1998.

[4] Nishimori, Yasunori, and Shotaro Akaho, *Learning algorithms utilizing quasi-geodesic flows on the Stiefel manifold*, Neurocomputing, 2005.