Unsupervised scalable anomaly detection: application to medical imaging

Geoffroy Oudoumanessah, Michel Dojat, Florence Forbes

INRIA Grenoble
Grenoble Institute of Neurosciences

Research report

Unsupervised scalable anomaly detection: application to medical imaging

Geoffroy Oudoumanessah, Michel Dojat, Florence Forbes

October 21, 2022

Contents

List of Figures

1 Introduction

1.1 Parkinson disease

Parkinson's disease (PD) is a neurodegenerative disease (see subsection 6.1) involving painful and disabling symptoms for the patient. Today, although known and widespread, its diagnosis at the earliest stages is very complicated. On the one hand, patients often consult only when they start experiencing motor symptoms and those appear late in the disease resulting in a late diagnosis. On the other hand the lesions in the brain due to Parkinson's disease remain very slight in *de novo* (newly diagnosed) patients. Moreover, these lesions are often located in very fine regions of the brain such as the Substantia Nigra (see Figure 1). The challenge is therefore to determine new biomarkers that would allow the characterization of Parkinson's disease at its earliest stage.

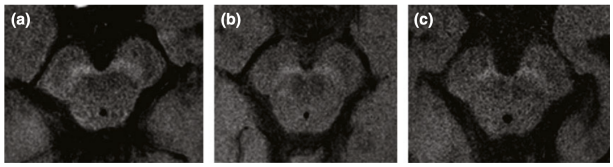


Figure 1: Neuromelanin-sensitive magnetic resonance images of the Substancia Nigra of a healthy control (a), a *de novo* PD patient (b) and a 2–5 year PD patient (c). [28]

In this study, we are interested in finding biomarkers in medical MRI images of *de novo* PD patients. This work follows that of a former PhD student Verónica Muñoz Ramírez [19] and aims at solving some of the issues encountered in this previous work. In particular, we propose and implement a solution to deal with the large quantity of data that prevented the use of some of the proposed methods. Indeed, it turns out that MRI images can be very heavy in terms of memory, which often implies problems of loading and exceeding the capacity of the RAM.

1.2 Unsupervised Anomaly Detection

For early PD diagnosis from MRI, we investigate the possibility to detect subtle brain lesions. The main difficulty is that such lesions are often not visible to the human eye and are very small in general. In contrast to brain tumors that can be quite large, de novo PD lesions are very unlikely to be detected via traditional segmentation or classification methods which require each class to be represented enough. Instead, they can be more reliably identified as abnormal tissues or in other words as brain anomalies.

To address the problem of detecting anomalies in MRI images of *de novo* PD patients, we will place ourselves in the context of unsupervised anomaly detection (UAD). Indeed, we consider a case where no ground truth or labeled images are available and the only information we have is whether a given image is coming from a PD patient or an healthy control.

UAD can be described in two main steps, the first one consists in the estimation of a reference model, computed from data without abnormalities. In our case, the reference model is computed from healthy subjects images. The second step is the detection step per se. In our case, the goal is to detect abnormal voxels (3D version

of a pixel), abnormal meaning far enough or not well explained by the reference model. Such a decision rule thus requires a notion of distance to the reference model developed in subsection 5.1.

1.3 Learning a reference model

In order to learn a reference model from healthy data, the choice was made to consider a mixture of multi-scale t-distributions [10] (see section 2). This model allows a richer variety of distribution shapes than standard Gaussian mixtures or more generally mixtures of elliptic distributions without requiring a too large number of components. In addition, the MRI modalities considered in this work often clearly exhibit non-Gaussian clusters as in Figure 20.

This type of mixtures has been already successfully used in previous work on lesions detection, see [2] and [19]. As often for mixtures, the algorithm used to learn the model was the EM algorithm [17]. Unfortunately, the standard EM implementation is greatly limited by the number of data points it can handle at once. As a consequence, in the previous works mentioned above only small (brain rat) images or a limited amount of larger human brain images were used. For human PD studies, this may limits the generality of the conclusions we can draw. The goal of this work is therefore to derive a version of the EM algorithm able to process the data online or sequentially in minibatches of smaller sizes than the entire available data set. More specifically, we will consider EM in the context of online learning using stochastic approximation [29] (see also [18] for a recent book covering related topics).

1.4 Common thread of the study

This study contains four parts:

1. The first one consists in presenting the multi-scale t-distribution.
2. The second part consists in explaining, implementing and testing on simulated data the online version of the EM algorithm for mixtures of multi-scale t-distributions.
3. The third one consists in designing an anomaly detection rule.
4. Finally, the last part is entirely dedicated to the medical application of UAD using mixtures of multi-scale t-distributions.

2 Mixture of heavy tailed distributions

In this short part, we introduce a generalization of the multivariate t-distribution which leads to a richer family of distributions referred to in [10] as the multi-scale t-distribution. We will then consider mixtures of those to build a reference model.

2.1 Multi-scale t-distribution ($\mathcal{MST}$)

To begin with, we recall the representation of the t-distribution as a infinite scaled mixture of Gaussians. It has the form:

$$p(\mathbf{y};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\theta}) = \int_0^\infty \mathcal{N}_M(\mathbf{y};\boldsymbol{\mu},\boldsymbol{\Sigma}/w) f_W(w,\boldsymbol{\theta}) dw, \tag{2.1}$$

with f_W the probability density function (p.d.f) of an one dimensional positive scaling variable W, and $\mathcal{N}_M(\boldsymbol{\mu},\boldsymbol{\Sigma}/w)$ the $M \in \mathbb{N}^*$ dimensional multivariate scaled Gaussian distribution, of mean $\boldsymbol{\mu} \in \mathbb{R}^M$, scaled covariance matrix $\boldsymbol{\Sigma}/w \in \mathbb{R}^{M\times M}$, and the scaling factor $w \in \mathbb{R}$.

When $\boldsymbol{\theta} = \nu/2 \in \mathbb{R}$ and $f_W = \mathcal{G}$ the p.d.f of the Gamma distribution, we recover the standard M-dimensional t-distribution of mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Sigma}$ and degree of freedom (dof) ν:

$$\begin{aligned} t_M(\mathbf{y};\boldsymbol{\mu},\boldsymbol{\Sigma},\nu) &= \int_0^\infty \mathcal{N}_M(\mathbf{y};\boldsymbol{\mu},\boldsymbol{\Sigma}/w)\mathcal{G}(w;\nu/2,\nu/2)dw \\ &= \frac{\Gamma((\nu+M)/2)}{|\boldsymbol{\Sigma}|^{1/2}\Gamma(\nu/2)(\pi\nu)^{M/2}} \left[1+\delta(\mathbf{y};\boldsymbol{\mu},\boldsymbol{\Sigma})/\nu\right]^{-(\nu+M)/2}, \end{aligned}$$

with Γ the gamma function, δ the Mahalanobis distance between $\mathbf{y}$ and $\boldsymbol{\mu}$ with covariance $\boldsymbol{\Sigma}$.

The generalization proposed in [10], is to insert more scaling factors by using the positive definiteness of the covariance matrix $\boldsymbol{\Sigma}$. Indeed, we have by the spectral theorem that $\boldsymbol{\Sigma} = \mathbf{DAD}^T$, with $\mathbf{D}$ orthogonal matrix of size M and A a diagonal matrix of size M [5]. Then we can naturally introduce multiple scaling factors as $\boldsymbol{\Delta}_\mathbf{w} = \text{diag}(w_1^{-1},\dots,w_M^{-1})$, $\mathbf{w} = [w_1 \dots w_M] \in \mathbb{R}^M$, such that the scaled covariance matrix becomes $\mathbf{D}\boldsymbol{\Delta}_\mathbf{w}\mathbf{AD}^T$. Then the proposed generalization is as follows:

$$p(\mathbf{y};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\theta}) = \int_0^\infty \mathcal{N}_M(\mathbf{y};\boldsymbol{\mu},\mathbf{D}\boldsymbol{\Delta}_\mathbf{w}\mathbf{AD}^T) f_\mathbf{W}(w_1,\dots,w_M;\boldsymbol{\theta}) dw_1 \dots dw_M. \tag{2.2}$$

In addition by considering independent scaling variables, the parameter $\boldsymbol{\theta} = [\theta_1 \dots \theta_M] \in \mathbb{R}^M$, Equation 2.2 becomes:

$$p(\mathbf{y};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\theta}) = \prod_{m=1}^M \int_0^\infty \mathcal{N}_1([\mathbf{D}^T(\mathbf{y}-\boldsymbol{\mu})]_m; 0, A_m w_m^{-1}) f_{W_m}(w_m;\theta_m) dw_m, \tag{2.3}$$

with the notation $A = \text{diag}(A_1,\dots,A_M)$ and $[\mathbf{x}]_m$ being the m^{th} element of the vector $\mathbf{x} \in \mathbb{R}^M$. Then taking $\boldsymbol{\nu} = [\nu_1 \dots \nu_M] \in \mathbb{R}^M$ and setting $f_{W_m}(w_m;\theta_m)$ to $\mathcal{G}(w_m;\nu_m/2,\nu_m/2)$ we finally get a generalization of the multivariate t-distribution, noted $\mathcal{MST}$ for multi-scale t-distribution:

$$\mathcal{MST}(\mathbf{y};\boldsymbol{\mu},\mathbf{A},\mathbf{D},\boldsymbol{\nu}) = \prod_{m=1}^{M} \frac{\Gamma((\nu_m+1)/2)}{\Gamma(\nu_m/2)\sqrt{A_m\nu_m\pi}} \left(1+\frac{[\mathbf{D}^T(\mathbf{y}-\boldsymbol{\mu})]_m^2]}{A_m\nu_m}\right)^{-((\nu_m+1)/2)}. \tag{2.4}$$

Like in the Gaussian case [5], we can give an interpretation of the following parameters:

- $\mathbf{A}$ corresponds to the spread or the volume of the distribution around the mean, the higher the coefficients of $\mathbf{A}$, the bigger is the spread.
- $\mathbf{D}$ gives the orientation of the distribution.
- $\boldsymbol{\nu}$ relates to the heaviness of the tail, the higher the coefficients of $\boldsymbol{\nu}$, the lighter the tail. An illustration is given in Figure 2.
- $\boldsymbol{\mu}$ represents the common mean.

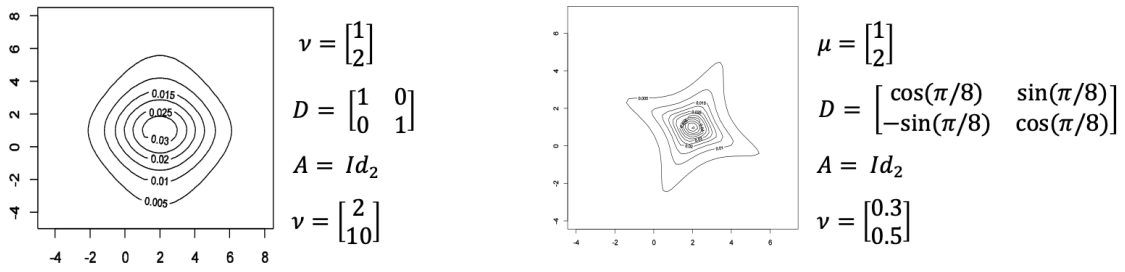


Figure 2: $\mathcal{MST}$ plot for different parameters.

2.2 Mixture of multi-scale t-distribution ($\mathcal{MMST}$)

In this report, the goal is to be able to make a clustering of a set of points using a finite mixture of the previously introduced multi-scale t-distributions. We can then simply define a finite mixture of such a distribution as:

$$\mathcal{MMST}(\mathbf{y};\boldsymbol{\pi},\overbrace{(\boldsymbol{\mu}_k,\mathbf{A}_k,\mathbf{D},\boldsymbol{\nu}_k)_{1\le k\le K}}^{:=\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\nu}}) = \sum_{k=1}^{K}\pi_k\mathcal{MST}(\mathbf{y};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k,\boldsymbol{\nu}_k), \tag{2.5}$$

with $\pi \in [0,1]^K$ such that $\sum_{k=1}^{K}\pi_k = 1$.

In contrast to Gaussian mixtures, this kind of mixtures will be really useful to model non elliptic clusters and to be more robust to extreme values. As illustrated in Figure 20 our data sets (see subsubsection 6.4.1) can exhibit non Gaussian clusters.

2.3 Sampling from a $\mathcal{MST}$ distribution

Simulating data that follows a $\mathcal{MST}$ distribution is relatively easy [10]. Indeed, we can see the construction of elements following Equation 2.4, in a generative way, by first constructing a Gaussian vector $\mathbf{X} = [X_1 \dots X_M]^T$ with $X \sim \mathcal{N}_M(\mathbf{0}_M, \mathbf{I}_M)$ and generating a vector of $W = [W_1 \dots W_M]^T$ with $W_i \sim f_{W_i}$, to finally get:

$$\mathbf{Y} = \boldsymbol{\mu} + \mathbf{D}\mathbf{A}^{1/2} \left[\frac{X_1}{\sqrt{W_1}} \dots \frac{X_M}{\sqrt{W_M}} \right]^T . \tag{2.6}$$

In particular, simulations will be used in section 4 to assess the quality of our estimation algorithm.

3 Online estimation of the reference model as a mixture

To build a reference model, from a reference data set we consider the previous family of distribution from which we need to learn the parameters. For mixtures models, estimation is usually made with the EM algorithm. [8] [17] This model has already been used in [19] in a similar context, but, in contrast to [19], we are going to use an online version of the EM algorithm for mixture of $\mathcal{MST}$ distributions. This is motivated by the fact that in [19], only two controls subjects (meaning healthy subjects) have been studied, this limitation was due to the number of data available but also by the amount of data contained in just two subjects that results in exploding the time complexity and RAM issues for the standard EM algorithm. Having an online version of the EM algorithm for $\mathcal{MST}$ distributions will enable us to process more data by passing the data point-by-point avoiding memory problems.

3.1 General online EM algorithm

The idea of online estimation algorithms is to be able to estimate the parameters of a model as the data comes, meaning that we don't need to pass all the data in one shot. An online version of the EM algorithm have been introduced in [4] for distributions that follow some restrictions.

Let's first define the observations, suppose that we have a sequence of n iid (independent and identically distributed) random variables, where $\mathbf{Y_i} \in \mathbb{Y} \subset \mathbb{R}^M$ is the visible component of $\mathbf{X}_i^T = (\mathbf{Y}_i^T, \mathbf{Z}_i^T)$ for $i = 1 \dots n$, and where $\mathbf{Z}_i \in \mathbb{Z} \subset \mathbb{R}^l$ is a latent variable. In addition, we suppose that $\mathbf{Y}$ is a random variable following the distribution we aim to estimate, with $\boldsymbol{\theta} \in \mathbb{T} \subset \mathbb{R}^p$ the parameter. In the context of online learning, the $\mathbf{Y}_i$ is one observation, we note that in the standard EM algorithm the considered pair is $\mathbf{X}^T = (\mathbf{Y}^T, \mathbf{Z}^T)$, with $\mathbf{Y} = (\mathbf{Y}_1, \dots, \mathbf{Y}_n)$ and $\mathbf{Z} = (\mathbf{Z}_1, \dots, \mathbf{Z}_n)$ containing all the observations.

In [4], some restrictions need to be verified before applying the proposed algorithm. The first condition is that the complete data likelihood (Meaning the one related to the pair $\mathbf{X}$) needs to be of the exponential family form:

$$f(\mathbf{x}, \boldsymbol{\theta}) = h(\mathbf{x}) \exp\left(s(\mathbf{x})^T \phi(\boldsymbol{\theta}) - \psi(\boldsymbol{\theta}) \right), \tag{3.1}$$

with $h : \mathbb{R}^{M+l} \mapsto [0, +\infty[$, $\psi : \mathbb{R}^p \mapsto \mathbb{R}$, $s : \mathbb{R}^{M+l} \mapsto \mathbb{R}^q$, and $\phi : \mathbb{R}^p \mapsto \mathbb{R}^q$, for $q \in \mathbb{N}$.

In addition the function $\bar{s}(\mathbf{y}; \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}[s(\mathbf{X})|\mathbf{Y} = \mathbf{y}]$ is well-defined for all $\mathbf{y} \in \mathbb{Y}$ and $\boldsymbol{\theta} \in \mathbb{T}$. Finally, there exists a convex subset $\mathbb{S} \subset \mathbb{R}^q$ such that $\forall \gamma \in [0, 1]$, $\mathbf{s} \in \mathbb{S}$, $\mathbf{y} \in \mathbb{Y}$, and $\boldsymbol{\theta} \in \mathbb{T}$

$$(1 - \gamma)s + \gamma\bar{s}(\mathbf{y}; \boldsymbol{\theta}) \in \mathbb{S}, \tag{3.2}$$

and such that $\forall \mathbf{s} \in \mathbb{S}$,

$$Q(\mathbf{s}; \boldsymbol{\theta}) = \mathbf{s}^T \phi(\boldsymbol{\theta}) - \psi(\boldsymbol{\theta}), \tag{3.3}$$

has a unique maximizer on $\mathbb{T}$ denoted by $\bar{\boldsymbol{\theta}}(\mathbf{s})$.

We recall the goal of the EM algorithm that is to find a maximum likelihood estimator (MLE) of $\boldsymbol{\theta}$, the basic idea here is then to maximize f with respect to $\boldsymbol{\theta}$ that can be done by maximizing Q only. Now the basic idea is to replace the expectation step, in this case, the estimation of $\bar{\mathbf{s}}$, by a stochastic approximation step [18]. And, by introducing a sequence of learning rates $(\gamma_i)_{i\in\mathbb{N}} \in [0, 1]^{\mathbb{N}}$, $\boldsymbol{\theta}_0 \in \mathbb{T}$, the sequences:

$$\mathbf{s}^i = \gamma_i \bar{\mathbf{s}}(\mathbf{Y}_i; \theta^{i-1}) + (1 - \gamma_i)\mathbf{s}^{i-1}, \tag{3.4}$$

and

$$\boldsymbol{\theta}^i = \bar{\boldsymbol{\theta}}(\mathbf{s}^i) \tag{3.5}$$

Cappé and Moulines in [4] proved that $(\boldsymbol{\theta}^i)_{i\in\mathbb{N}}$ converges to a stationary point of the likelihood.

Now that we have seen the general algorithm for the online EM algorithm, we still need to apply and verify the feasibility of this algorithm for a mixture of $\mathcal{MST}$ distributions (MMST). Namely, we need:

- To find, if it exists, an exponential form for the complete-data likelihood for the $\mathcal{MMST}$ model.
- Find a tractable form of $\bar{\mathbf{s}}$ for the E-step.
- Find a way to maximize Q for the M-step.

3.2 The exponential form of the $\mathcal{MST}$ distribution

It is easy to check that a mixture of exponential form distributions has an exponential form [21], therefore in our case, it is enough to exhibit an exponential form for a simple MST.

We recall the complete likelihood of a $\mathcal{MST}$ distribution Equation 2.3:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}_M(\mathbf{y}; \boldsymbol{\mu}, \mathbf{D}\boldsymbol{\Delta}_{\mathbf{W}}\mathbf{A}\mathbf{D}^{\mathbf{T}}) \prod_{m=1}^{M} \mathcal{G}\left(w_m; \frac{\nu_m}{2}, \frac{\nu_m}{2}\right). \tag{3.6}$$

We note that the Gamma distribution belongs to the exponential family, and it can be written as,

$$\mathcal{G}\left(w_m; \frac{\nu_m}{2}, \frac{\nu_m}{2}\right) = h_{W_m}(w_m) \exp(s_{W_m}(w_m)\phi_{W_m}(\nu_m) - \psi_{W_m}(\nu_m)), \tag{3.7}$$

with $h_{W_m}(w_m) = \frac{1}{w_m}$, $s_{W_m} = \log(w_m) - w_m$, $\phi_{W_m}(\nu_m) = \frac{\nu_m}{2}$, and $\psi_{W_m}(\nu_m) = \log\left(\Gamma(\frac{\nu_m}{2})\right) - \frac{\nu_m}{2}\log\left(\frac{\nu_m}{2}\right)$. Therefore, the product Equation 3.6 of Gamma laws can be easily written with:

- $h_{\mathbf{W}}(\mathbf{w}) = h_{W_1}(w_1) \times \cdots \times h_{W_M}(w_M)$
- $s_{\mathbf{W}}(\mathbf{w}) = (s_{W_1}(w_1) \dots s_{W_M}(w_M))^T$
- $\phi_{\mathbf{W}}(\boldsymbol{\nu}) = (\phi_{W_1}(\nu_1) \dots \phi_{W_M}(\nu_M))^T$
- $\psi_{\mathbf{W}}(\boldsymbol{\nu}) = \sum_{m=1}^{M} \psi_{W_m}(\nu_m)$.

We also notice that

$$\mathcal{N}_M(\mathbf{y}; \boldsymbol{\mu}, \mathbf{D}\boldsymbol{\Delta}_{\mathbf{W}}\mathbf{A}\mathbf{D}^{\mathbf{T}}) = \prod_{m=1}^{M} \mathcal{N}_1([\mathbf{D}^T(\mathbf{y} - \boldsymbol{\mu})]_m; 0, A_m w_m^{-1})$$

, with $\mathcal{N}_1([\mathbf{D}^T(\mathbf{y} - \boldsymbol{\mu})]_m; 0, A_m w_m^{-1}) = \frac{1}{\sqrt{2\pi A_m w_m^{-1}}} \exp\left(-\frac{[\mathbf{D}^T(\mathbf{y}-\boldsymbol{\mu})]_m^2}{2A_m w_m^{-1}}\right)$. By using the notation $[\mathbf{D}^T(\mathbf{y}-\boldsymbol{\mu})]_m^2 = \text{vec}(\mathbf{d}_m\mathbf{d}_m^T)^T\text{vec}(\mathbf{y}\mathbf{y}^T) + \text{vec}(\mathbf{d}_m\mathbf{d}_m^T)^T\text{vec}(\boldsymbol{\mu}\boldsymbol{\mu}^T) - 2\boldsymbol{\mu}^T\mathbf{d}_m\mathbf{d}_m^T\mathbf{y}$, with $\mathbf{D} = (\mathbf{d}_1 \dots \mathbf{d}_M)^T$, and $\text{vec}(A) = [a_{1,1}, \dots, a_{m,1}, a_{1,2}, \dots, a_{m,2}, \dots, a_{1,n}, \dots, a_{m,n}]^T$. We can finally find after some calculations, the exponential form of the $\mathcal{MST}$ distribution as

$$f(\mathbf{x}, \overbrace{\boldsymbol{\mu}, \mathbf{A}, \mathbf{D}, \boldsymbol{\nu}}^{:=\boldsymbol{\theta}}) = h(\mathbf{y}, \mathbf{w}) \exp\left(s(\mathbf{y}, \mathbf{w})^T \phi(\boldsymbol{\mu}, \mathbf{D}, \mathbf{A}, \boldsymbol{\nu}) - \psi(\boldsymbol{\mu}, \mathbf{D}, \mathbf{A}, \boldsymbol{\nu})\right), \tag{3.8}$$

with:

$$h(\mathbf{y}, \mathbf{w}) = \frac{(w_1 \dots w_M)^{1/2}}{\sqrt{2\pi}^M} h_{\mathbf{W}}(\mathbf{w}) \tag{3.9}$$

$$s(\mathbf{y}, \mathbf{w}) = \begin{bmatrix} w_1\mathbf{y} \\ w_1\text{vec}(\mathbf{y}\mathbf{y}^T) \\ w_1 \\ \log w_1 \\ \vdots \\ w_M\mathbf{y} \\ w_M\text{vec}(\mathbf{y}\mathbf{y}^T) \\ w_M \\ \log w_M \end{bmatrix} \tag{3.10}$$

$$\phi(\boldsymbol{\mu}, \mathbf{D}, \mathbf{A}, \boldsymbol{\nu}) = \begin{bmatrix} \frac{\mathbf{d}_1\mathbf{d}_1^T\boldsymbol{\mu}}{A_1} \\ -\frac{\text{vec}(\mathbf{d}_1\mathbf{d}_1^T)}{2A_1} \\ -\frac{\text{vec}(\mathbf{d}_1\mathbf{d}_1^T)\text{vec}(\boldsymbol{\mu}\boldsymbol{\mu}^T)}{2A_1} - \frac{\nu_1}{2} \\ \frac{1+\nu_1}{2} \\ \vdots \\ \frac{\mathbf{d}_M\mathbf{d}_M^T\boldsymbol{\mu}}{A_M} \\ -\frac{\text{vec}(\mathbf{d}_M\mathbf{d}_M^T)}{2A_M} \\ -\frac{\text{vec}(\mathbf{d}_M\mathbf{d}_M^T)\text{vec}(\boldsymbol{\mu}\boldsymbol{\mu}^T)}{2A_M} - \frac{\nu_M}{2} \\ \frac{1+\nu_M}{2} \end{bmatrix} \tag{3.11}$$

$$\psi(\boldsymbol{\theta}) = \sum_{m=1}^{M} \left(\frac{\log A_m}{2} + \log\Gamma\left(\frac{\nu_m}{2}\right) - \frac{\nu_m}{2}\log\left(\frac{\nu_m}{2}\right) \right) \tag{3.12}$$

For future computations, it will be useful to write $\mathbf{s}$ as:

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_{11} \\ \text{vec}(\mathbf{S}_{21}) \\ s_{31} \\ s_{41} \\ \vdots \\ \mathbf{s}_{1M} \\ \text{vec}(\mathbf{S}_{2M}) \\ s_{3M} \\ s_{4M} \end{bmatrix}, \tag{3.13}$$

with $\mathbf{s}_{1m} \in \mathbb{R}^M$, $\mathbf{S}_{2m}\mathbb{R}^{M\times M}$, and s_{3m}, s_{4m} scalars for each $m \in \{1 \dots M\}$.

3.3 Online EM for a single $\mathcal{MST}$ component

As developed in [10] EM can already be used for a single MST, we give below the online EM version according to [4].

3.3.1 Maximization step

We recall that the Maximization step consists in finding the maximum of $Q(\mathbf{s};\boldsymbol{\theta}) = \mathbf{s}^T\phi(\boldsymbol{\theta}) - \psi(\theta)$ with respect to $\boldsymbol{\theta}$. To do so, we can find a stationary point of Q by solving:

$$\frac{\partial\phi}{\partial\boldsymbol{\theta}}\mathbf{s} - \frac{\partial\psi}{\partial\boldsymbol{\theta}}(\boldsymbol{\theta}) = 0. \tag{3.14}$$

After some computation and derivation, we find:

$$\boldsymbol{\mu} = \mathbf{D}\mathbf{S}_3^{-1}\mathbf{v}, \tag{3.15}$$

with $\mathbf{S}_3 = \text{diag}(s_{31} \dots s_{3M})$, and $\mathbf{v}^T = (\mathbf{d}_1^T\mathbf{s}_{11} \dots \mathbf{d}_M^T\mathbf{s}_{1M})$.

For each $m \in \{1 \dots M\}$ we also have

$$A_m = \mathbf{d}_m^T \mathbf{S}_{2m} \mathbf{d}_m - \frac{(\mathbf{d}_m^T \mathbf{s}_{1m})^2}{s_{3m}}, \tag{3.16}$$

with the notation $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_M]$. Equation 3.16 has been obtained by using the formula of the optimal $\boldsymbol{\mu}$ found in Equation 3.15. And using the previous two equations, we also find an optimization for $\mathbf{D}$:

$$\mathbf{D} = \underset{\mathbf{D}\mathbf{D}^T = \mathbf{I}_M}{\arg\min} \sum_{m=1}^{M} \log \left(\mathbf{d}_m^T \left(\mathbf{S}_{2m} - \frac{\mathbf{s}_{1m} \mathbf{s}_{1m}^T}{s_{3m}} \right) \mathbf{d}_m \right). \tag{3.17}$$

This problem of finding $\mathbf{D}$ is tricky, the problem is not convex and the constraints are that we need $\mathbf{D}$ to be orthogonal, this is why in subsubsection 3.3.2, we will discuss of a efficient method to find $\mathbf{D}$.

Finally the last parameter ν can be optimized separately by solving, for each $m \in \{1 \dots M\}$ the equation:

$$s_{4m} - s_{3m} - \Psi^{(0)}(\frac{\nu_m}{2}) + \log(\frac{\nu_m}{2}) + 1 = 0, \tag{3.18}$$

with $\Psi^{(0)}$ the digamma function (first derivative of the *log-gamma* function).

3.3.2 Details on the M-step for D

During the M step of the online EM for a $\mathcal{MST}$, we face the optimization problem, with $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_M]$,

$$\mathbf{D} = \underset{\mathbf{D}\mathbf{D}^T = \mathbf{I}_M}{\arg\min} \sum_{m=1}^{M} \log \left(\mathbf{d}_m^T \left(\mathbf{S}_{2m} - \frac{\mathbf{s}_{1m} \mathbf{s}_{1m}^T}{s_{3m}} \right) \mathbf{d}_m \right). \tag{3.19}$$

Problem Equation 3.19 can be seen as an orthogonality constrained optimization over the set of matrices of size $M \times M$. Another view consists in approaching this issue by seeing the problem as a minimization over the Stiefel manifold $St(M, M)$.

To solve Equation 3.19, we use optimization on manifolds, and a version of the conjugate gradient algorithm designed for manifolds Algorithm 2 with the Polak-Ribiere line search Equation 7 computing the Stiefel gradient using Equation A.9.

All the details and references are given in Appendix A.

3.3.3 Expectation step

Now that we developed the maximization step, we still need to update the statistics at each iteration as in Equation 3.4, the computations are very similar to [10]. To do so, we need to compute at each iteration $\bar{\mathbf{s}}(\mathbf{y}; \theta) = \mathbb{E}_\theta[s(\mathbf{X})|\mathbf{Y} = \mathbf{y}]$.

Given the relation find in Equation 3.10, we need to compute $\mathbb{E}_\theta[W_m|\mathbf{Y} = \mathbf{y}]$ and $\mathbb{E}_\theta[\log W_m|\mathbf{Y} = \mathbf{y}]$. We then denote:

$$u_{im}^{i-1} = \mathbb{E}_{\theta^{i-1}}[W_m|\mathbf{Y} = \mathbf{y}_i] = \frac{\alpha_m^{(i-1)}}{\beta_m^{(i-1)}} \tag{3.20}$$

and

$$\tilde{u}_{im}^{(i-1)} = \mathbb{E}_{\theta^{(i-1)}}[\log W_m | \mathbf{Y} = \mathbf{y}_i] = \Psi^{(0)}(\alpha_m^{(i-1)}) - \log \beta_m^{(i-1)}, \tag{3.21}$$

where

$$\alpha_m^{(i-1)} = \frac{\nu_m^{(i-1)} + 1}{2} \tag{3.22}$$

and

$$\beta_m^{(i-1)} = \frac{\nu_m^{(i-1)}}{2} + \frac{\left(\mathbf{d}_m^{(i-1)T}(\mathbf{y}_i - \boldsymbol{\mu}^{(i-1)})\right)^2}{2A_m^{(i-1)}} \tag{3.23}$$

Finally, the update for the statistics given the notation introduced in Equation 3.13, we obtain:

$$\mathbf{s}_{1m}^{(i)} = \gamma_i u_{1m}^{(i-1)} \mathbf{y}_i + (1 - \gamma_i)\mathbf{s}_{1m}^{(i-1)} \tag{3.24}$$
$$\mathbf{S}_{2m}^{(i)} = \gamma_i u_{im}^{(i-1)} \mathbf{y}_i \mathbf{y}_i^T + (1 - \gamma_i)\mathbf{S}_{2m}^{(i-1)} \tag{3.25}$$
$$s_{3m}^{(i)} = \gamma_i u_{im}^{(i-1)} + (1 - \gamma_i) s_{3m}^{(i-1)} \tag{3.26}$$
$$s_{4m}^{(i)} = \gamma_i \tilde{u}_{im}^{(i-1)} + (1 - \gamma_i) s_{4m}^{(i-1)}, \tag{3.27}$$

with:

$$\mathbf{s}_{1m}^{(i)} = u_{1m}^{(0)} \mathbf{y}_1 \tag{3.28}$$
$$\mathbf{S}_{2m}^{(i)} = u_{1m}^{(0)} \mathbf{y}_1 \mathbf{y}_1^T \tag{3.29}$$
$$s_{3m}^{(i)} = u_{1m}^{(0)} \tag{3.30}$$
$$s_{4m}^{(i)} = \tilde{u}_{1m}^{(0)}, \tag{3.31}$$

for initialization.

The mini-batch version Previously we have seen an update method that only use **one** observation per iteration. Implementing this in Python sometimes gives us some overflow errors due to some high values. The idea is then to derive a mini-batch version of the online version. This derivation is very simple (*e.g* [22]), and given a mini-batch of size B, the new statistics update is:

$$\mathbf{s}^{(i)} = \gamma_i \frac{1}{B} \sum_{b=0}^{B-1} \bar{s}(\mathbf{y}_{B \times i + b}; \boldsymbol{\theta}^{i-1}) + (1 - \gamma_i)\mathbf{s}^{(i-1)} \tag{3.32}$$

3.4 Online EM for mixture of $\mathcal{MST}$ ($\mathcal{MMST}$)

So far, we have seen an algorithm capable of estimating the parameters of a $\mathcal{MST}$ distribution, but our final goal is to estimate the parameters of a finite mixture of $\mathcal{MST}$ distributions. General formulas for a Gaussian mixture is given in [23].

3.4.1 Maximization step

For a mixture of K components, we note Z the random variable taking values in $\{1 \dots K\}$ such that $\mathbb{P}\{Z = k\} = \pi_k$, and we denote by $\boldsymbol{\theta}_{\mathcal{M}}$ the parameters of the mixture containing the pairs $(\boldsymbol{\theta}_k, \pi_k)$, with $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \mathbf{A}_k, \mathbf{D}_k, \boldsymbol{\nu}_k)$, and the mixture p.d.f is then:

$$p(\mathbf{y}; \boldsymbol{\theta}_{\mathcal{M}}) = \sum_{k=1}^{K} \pi_k \mathcal{MST}(\mathbf{y}; \boldsymbol{\theta}_k). \tag{3.33}$$

In the same idea that with a $\mathcal{MST}$ distribution, we consider this time the pairs $\mathbf{X}_i^T = (\mathbf{Y}_i^T, \mathbf{W}_i^T, Z_i)$, and after a small computation, the complete-data likelihood for the mixture is then [23]:

$$f_c(\mathbf{x}, \boldsymbol{\theta}_{\mathcal{M}}) = h(\mathbf{y}, \mathbf{w}) \exp\left(\sum_{k=1}^{K} \mathbf{1}_{z=k} \left[\log \pi_k + s(\mathbf{y}, \mathbf{w})^T \phi(\boldsymbol{\mu}_k, \mathbf{D}_k, \mathbf{A}_k, \boldsymbol{\nu}_k) - \psi(\boldsymbol{\mu}_k, \mathbf{D}_k, \mathbf{A}_k, \boldsymbol{\nu}_k) \right] \right), \tag{3.34}$$

which is well written in the exponential family form, then the online EM applies with $h_{\mathcal{M}}(\mathbf{x}) = h(\mathbf{y}, \mathbf{w})$, $\psi_{\mathcal{M}}(\boldsymbol{\theta}) = 0$,

$$s_{\mathcal{M}}(\mathbf{x}) = \begin{bmatrix} \mathbf{1}_{z=1} \\ \mathbf{1}_{z=1} s(\mathbf{y}, \mathbf{w}) \\ \vdots \\ \mathbf{1}_{z=K} \\ \mathbf{1}_{z=K} s(\mathbf{y}, \mathbf{w}) \end{bmatrix},$$

and

$$\phi_{\mathcal{M}}(\boldsymbol{\theta}_{\mathcal{M}}) = \begin{bmatrix} \log \pi_1 - \psi(\boldsymbol{\theta}_1) \\ \phi(\boldsymbol{\theta}_1) \\ \vdots \\ \log \pi_K - \psi(\boldsymbol{\theta}_K) \\ \phi(\boldsymbol{\theta}_K) \end{bmatrix}.$$

For simplicity, we introduce, the notation $\mathbf{s}_{\mathcal{M}}^T = (s_{01}, \mathbf{s}_{\mathcal{M}_1}^T, \dots, s_{0K}, \mathbf{s}_{\mathcal{M}_K}^T)$, and $\mathbf{s}_{\mathcal{M}_k}^T = (\mathbf{s}_{1k}, \dots, \mathbf{s}_{qk})$, with q the dimension of the previous vector $\mathbf{s}$ for a simple $\mathcal{MST}$ distribution, and $k \in \{1 \dots K\}$.

The new function to maximize with respect to $\boldsymbol{\theta}_{\mathcal{M}}$ is this time, $Q_{\mathcal{M}}(\mathbf{s}_{\mathcal{M}}, \boldsymbol{\theta}_{\mathcal{M}}) = \mathbf{s}_{\mathcal{M}}^T \phi_{\mathcal{M}}(\boldsymbol{\theta}_{\mathcal{M}})$, to simplify the maximization process, we can optimize each term of $\boldsymbol{\theta}_{\mathcal{M}}$ separately, and the maximization with respect to π_k yields

$$\pi_k(\mathbf{s}_{\mathcal{M}}) = \frac{s_{0k}}{\sum_{\zeta=1}^{K} s_{0\zeta}}. \tag{3.35}$$

In addition, as we previously said, we can optimize with respect to each $\boldsymbol{\theta}_k$ separately giving:

$$\frac{\partial Q_{\mathcal{M}}}{\partial \boldsymbol{\theta}_k}(\mathbf{s}_{\mathcal{M}}, \boldsymbol{\theta}_{\mathcal{M}}) = -s_{0k}\frac{\partial \psi}{\partial \boldsymbol{\theta}_k}(\boldsymbol{\theta}_k) + J_\phi(\boldsymbol{\theta}_k)\mathbf{s}_{\mathcal{M}k} = s_{0k}\frac{\partial Q}{\partial \boldsymbol{\theta}_k}\left(\left[\frac{\mathbf{s}_{\mathcal{M}_k}}{s_{0k}}\right], \boldsymbol{\theta}_k\right) \tag{3.36}$$

.

The Equation 3.36 means that for each $k \in \{1 \dots K\}$, the maximization of $Q_{\mathcal{M}}$ with respect to $\boldsymbol{\theta}_k$ is the same as maximizing $Q(\left[\frac{\mathbf{s}_{\mathcal{M}_k}}{s_{0k}}\right], .)$. Hence with previous notations (the ones for the $\mathcal{MST}$ distribution), we have:

$$\bar{\boldsymbol{\theta}}_{\mathcal{M}}(s_{\mathcal{M}}) = \begin{bmatrix} \bar{\pi}_1(\mathbf{s}_{\mathcal{M}}) \\ \bar{\boldsymbol{\theta}}(\mathbf{s}_{\mathcal{M}_1}/s_{01}) \\ \vdots \\ \bar{\pi}_K(\mathbf{s}_{\mathcal{M}}) \\ \bar{\boldsymbol{\theta}}(\mathbf{s}_{\mathcal{M}_K}/s_{0K}) \end{bmatrix}. \tag{3.37}$$

3.4.2 Expectation step

In this step, we need to compute the relevant statistics $\bar{\mathbf{s}}_{\mathcal{M}}(\mathbf{y}; \boldsymbol{\theta}_{\mathcal{M}}) = \mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}}[\mathbf{s}_{\mathcal{M}}(\mathbf{X})|\mathbf{Y} = \mathbf{y}]$. To do so, we need to compute,

$$r_{ik} = \mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}}[\mathbf{1}_{Z=k}|\mathbf{Y} = \mathbf{y}_i] = \mathbb{P}\{Z = k|\mathbf{y}_i\} = \frac{\pi_k \mathcal{MST}(\mathbf{y}_i; \boldsymbol{\theta}_k)}{p(\mathbf{y_i}; \boldsymbol{\theta}_{\mathcal{M}})},$$

$$\mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}}[\mathbf{1}_{Z=k}s(\mathbf{Y}, \mathbf{W})|\mathbf{Y} = \mathbf{y}_i] = r_{ik}\mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}}[\mathbf{s}(\mathbf{Y}, \mathbf{W})|\mathbf{Y} = \mathbf{y}_i, Z = k].$$

The last expectation requires to compute for each $m \in \{1 \dots M\}$ (M being the dimension of the data), and $k \in \{1 \dots K\}$, $u_{imk} = \mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}}[W_m|\mathbf{Y} = \mathbf{y}_i, Z = k]$ and $\tilde{u}_{imk} = \mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}}[\log W_m|\mathbf{Y} = \mathbf{y}_i, Z = k]$. And we have:

$$u_{imk}^{(i-1)} = \mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}^{(i-1)}}[W_m|\mathbf{Y} = \mathbf{y}_i, Z = k] = \frac{\alpha_{mk}^{(i-1)}}{\beta_{mk}^{(i-1)}}, \tag{3.38}$$

$$\tilde{u}_{imk}^{(i-1)} = \mathbb{E}_{\boldsymbol{\theta}_{\mathcal{M}}^{(i-1)}}[\log W_m|\mathbf{Y} = \mathbf{y}_i, Z = k] = \Psi^{(0)}(\alpha_{mk}^{(i-1)}) - \log \beta_{mk}^{(i-1)}), \tag{3.39}$$

with

$$\alpha_{mk}^{i-1} = \frac{\nu_{mk}^{(i-1)} + 1}{2}, \tag{3.40}$$

$$\beta_{mk}^{(i-1)} = \frac{\nu_{mk}^{(i-1)}}{2} + \frac{\left(\mathbf{d}_{mk}^{(i-1)T}(\mathbf{y}_i - \boldsymbol{\mu}_k^{(i-1)})\right)^2}{2A_{mk}^{(i-1)}} \tag{3.41}$$

.

The update of the statistics is then very similar to the update for a $\mathcal{MST}$ distribution, meaning that for each $m \in \{1 \dots M\}$, and each $k \in \{1 \dots K\}$:

$$
\begin{aligned}
s_{0k}^{(i)} &= \gamma_i r_{ik}^{(i-1)} + (1-\gamma_i) s_{0k}^{(i-1)} \\
\mathbf{s}_{2mk}^{(i)} &= \gamma_i u_{1mk}^{(i-1)} \mathbf{y}_i + (1-\gamma_i) \mathbf{s}_{1mk}^{(i-1)} \\
\mathbf{S}_{2mk}^{(i)} &= \gamma_i u_{imk}^{(i-1)} \mathbf{y}_i \mathbf{y}_i^T + (1-\gamma_i) \mathbf{S}_{2mk}^{(i-1)} \\
s_{3mk}^{(i)} &= \gamma_i u_{im}^{(i-1)} + (1-\gamma_i) s_{3mk}^{(i-1)} \\
s_{4mk}^{(i)} &= \gamma_i \tilde{u}_{imk}^{(i-1)} + (1-\gamma_i) s_{4mk}^{(i-1)},
\end{aligned}
$$

4 Illustrations on simulated data

In order to assess the performance of the online EM algorithm, we just consider simulated data, and we also compare results with the standard EM where the data is in a single batch.

4.1 Code source and useful packages

The complete code in python is available on Github[1], and we use Pymanopt for automatic differentiation and calculation of Riemaniann gradient [30].

4.2 Results with the mini-batch version

To assess the online EM implementation, we chose to simulate data that follows a $\mathcal{MMST}$ distribution using Equation 4.1. They are two points we can check:

- The convergence of the parameters: $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, $\mathbf{A}$, $\mathbf{D}$, $\boldsymbol{\nu}$.
- The clustering that can be assessed visually and by using some metrics such as the accuracy and the F1-score.

For the tests, we choose to simulate data in 2 and 3 dimensions and to generate a mixture of four components. In each case we chose random values for each of the parameters, and we simulate 10^6 points.

For the initialization of $\boldsymbol{\mu}$, we use the Trimmed KMeans algorithm [6] which seems more appropriate than the KMeans algorithm due to the heavy tail of a $\mathcal{MST}$ distribution. We also compute $\mathbf{D}$ and $\mathbf{A}$ using the spectral decomposition of the empirical covariance of the training set, and we finally set the $\boldsymbol{\nu}$ coefficients to 20.

The learning rate defined in Equation 3.4 is chosen as in [4] with $\gamma_n = n^{-0.6}$.

In addition, in order to accelerate the convergence of the stochastic approximation made for the EM algorithm, we choose to use Polyak-Ruppert averaging [25]. If we note by Θ_n the iterates of the updates, we average the iterates starting from a chosen $n_0 > 1$:

$$
\bar{\Theta}_n = \frac{1}{n - n_0 + 1} \sum_{k=n_0}^{n} \theta_n. \tag{4.1}
$$

[1] `https://github.com/geoffroyO/OLEmMMST`

This means can also be computed recursively as $\bar{\Theta}_n = (1 - \frac{1}{n-n_0+1})\bar{\Theta}_{n-1} + \frac{1}{n-n_0+1}\Theta_n$. This solves some problems encountered with the Robbins-Monro update giving "smoother" results. In [25], it has been demonstrated that we have indeed a convergence of the iterates Θ_n towards the optimal value.

In addition, we should also remind that the decomposition of $\mathbf{\Sigma}$ in Equation 2.2, gives a particular order for $A_1 \dots A_M$, and for the columns of $\mathbf{D}$. But a permutation of the eigenvalues and the eigenvectors of $\mathbf{\Sigma}$ in the decomposition will still gives the same law, this is why after convergence, we will have to make the correction to assess to convergence of the parameters. More specifically, we look at the output $\boldsymbol{\pi}$ of the online EM algorithm and look for the permutation between the $\boldsymbol{\pi}$ we use to simulate data and $\boldsymbol{\pi}$ output.

4.2.1 $\mathcal{MMST}$ in 2D with 4 components

Firstly, we generate 10^6 points of a $\mathcal{MMST}$ with random parameters as in Figure 3. We can see that this example is not too hard because the centers of the components are well separated, but it is not too simple either because of the superposition of the tails of each component.

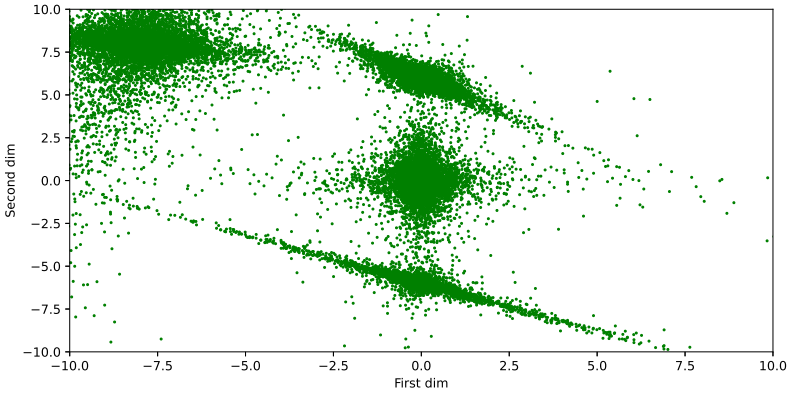


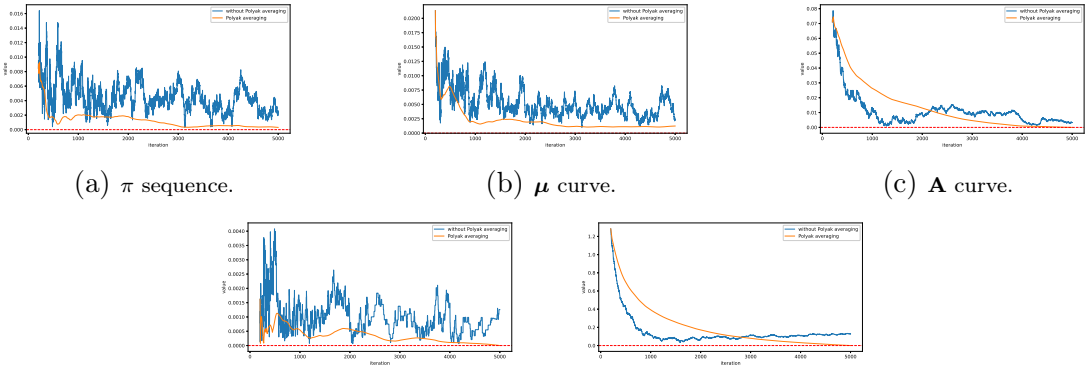
Figure 3: $\mathcal{MMST}$ simulation in 2D with 4 components.

In a second part, we choose to assess the convergence of the parameters. For each parameter, we compute at each iteration the distance between the current estimation and the real value.

For instance for π, we simply compute $||\pi^i - \pi_{true}||_2$. But for $\boldsymbol{\mu} = (\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_K)$, $\mathbf{A}$, $\mathbf{D}$, and $\boldsymbol{\nu} = (\boldsymbol{\nu}_1 \dots \boldsymbol{\nu}_K)$, we compute for each cluster its distance with its true value and then we compute the mean of the distances. Namely, for $\mathbf{D}$, with K clusters we have to consider $\mathbf{D}_1 \dots \mathbf{D}_K$. Then we compute $\frac{1}{K}\sum_{k=1}^{K} ||\mathbf{D}_k^i - \mathbf{D}_k||_{fro}$ at each iteration i.

The results are in Figure 4, we use a batch size of 200 and we notice that convergence appears around 500 iterations with a tolerance fixed to 10^{-2}, it means that we used $200 = 50000$ points before convergence, this result will be considered when we will compare the online EM with the standard EM algorithm. For information, processing the 10^6 points with a batch size of 200 took 5 minutes, this measure is simply informative mostly because it is very dependant on the implementation,

nevertheless you can find in Table 1 some computation time results compare with standard EM until convergence with a fixed tolerance at 10^{-2}.



(a) π sequence. (b) $\boldsymbol{\mu}$ curve. (c) $\mathbf{A}$ curve.

(d) $\mathbf{D}$ curve. (e) ν curve.

Figure 4: Convergence results for the parameters. (Iterations values in blue, Polyak averaging in orange)

Now that we have illustrated the convergence result, it would be interesting to assess the quality of the clustering. The clustering is an essential step of this report, where we will have, in the application to cluster voxels in *de novo* Parkinsonian patients. In Figure 5, we can visually see the result of the clustering, each components of the mixture seems well assigned, this is also verified with an overall 99% accuracy and *f1-score* for each class. Note that here the number of components is fixed to the true one.

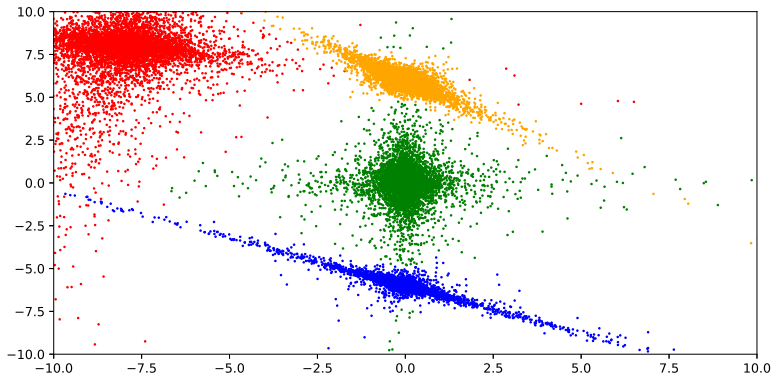


Figure 5: Clustering results with 4 components obtained with online EM.

In parallel and as an illustration, we have in Figure 6 the obtained clustering using a Gaussian Mixture Model with 4 components initialized using the Trimmed Kmeans too. We see that the algorithm performs really bad modelling the long tails of each $\mathcal{MST}$ component with infinite covariance.

4.2.2 $\mathcal{MMST}$ in 3D with 4 components

We will now display the results in three dimensions. In this case, we use the same method as in the previous section: firstly we simulate 10^6 points, then we assess the

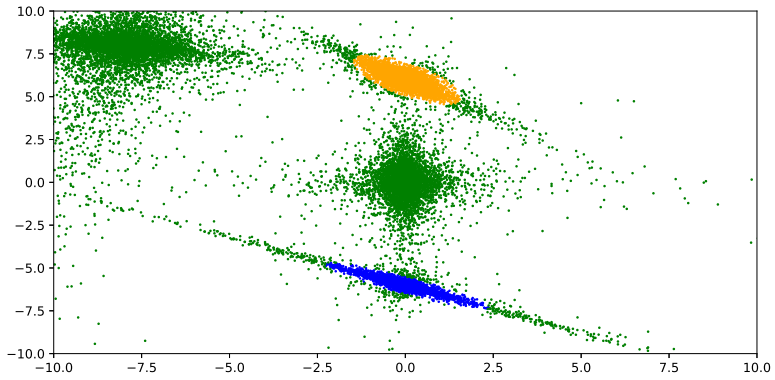


Figure 6: Clustering with 4 components with a Gaussian mixture model.

convergence of the parameters and finally the performance of the clustering.

To simulate the 10^6 point, the same method is used as in Equation 2.2, giving the 3D plot in Figure 7.

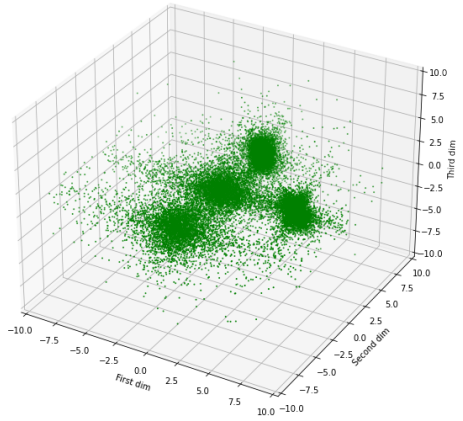
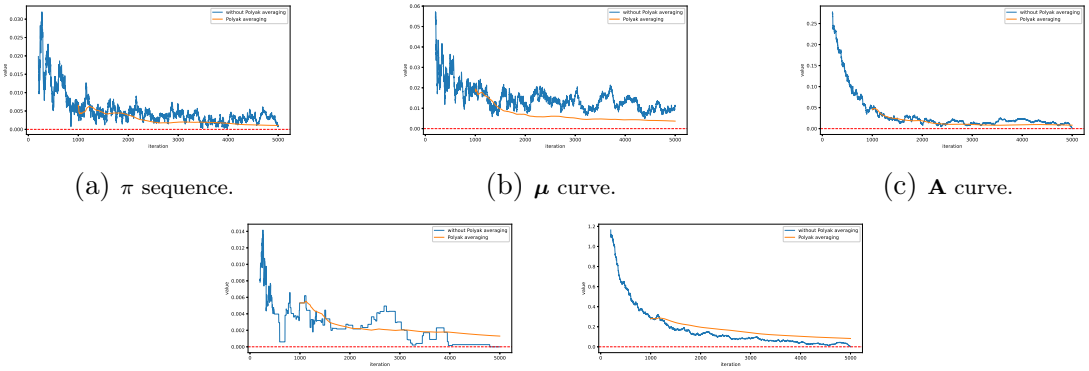


Figure 7: Simulation of 10^6 points following a $\mathcal{MMST}$ in 3 dimensions with 4 components.

The next step is now to assess the convergence of the parameters. The results are in Figure 8. As in two dimensions, we note that we attain convergence after the 700^{th} iteration, still with a tolerance fixed at 10^{-2}, the same batch size of 200 is used here. You can find computation time results in Table 1. We can notice that an increased budget is needed for convergence compared to the two dimensional case

Finally the clustering in Figure 9 is as good as in 2 dimensions with 99% accuracy and *f1-score* for each cluster.



(a) π sequence. (b) $\boldsymbol{\mu}$ curve. (c) $\mathbf{A}$ curve.

(d) $\mathbf{D}$ curve. (e) ν curve.

Figure 8: Convergence results for the parameters. (Iterations values in blue, Polyak averaging in orange)

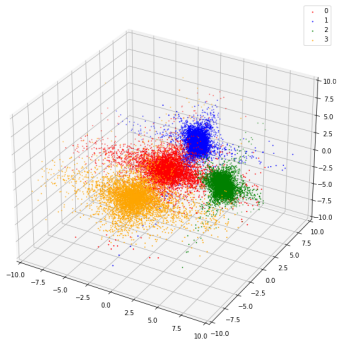


Figure 9: Clustering with 4 components obtained with online EM.

4.3 Comparison with standard EM

Now that we have illustrated some convergence results with the online version of the EM algorithm, it could be interesting to compare the behaviour of the standard EM algorithm for $\mathcal{MMST}$ described in [10] with the one of the online version.

Intuitively we can think that the standard EM algorithm will converge faster (in term of number of iterations), and the convergence curves will be smoother than the ones of the online version. Indeed, this is what we observed. In Figure 10, we have sampled 5 million points from a predefined $\mathcal{MMST}$, then we applied to this data the standard EM algorithm along with the online one, and we plot the same curves as in Figure 9.

The clustering in Figure 12 seems also as efficient between the standard and the online EM, both have 99% accuracy on the test set. Finally, the main difference we have between both implementations is in term of time complexity, the standard EM is a slower than the online one before convergence as seen in Table 1, but this is very dependent on the implementation, the real advantage of the online version is that it avoids RAM issues when we have many data.

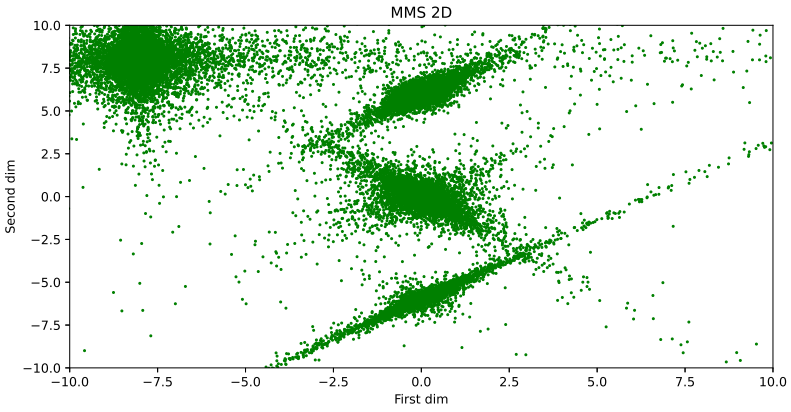
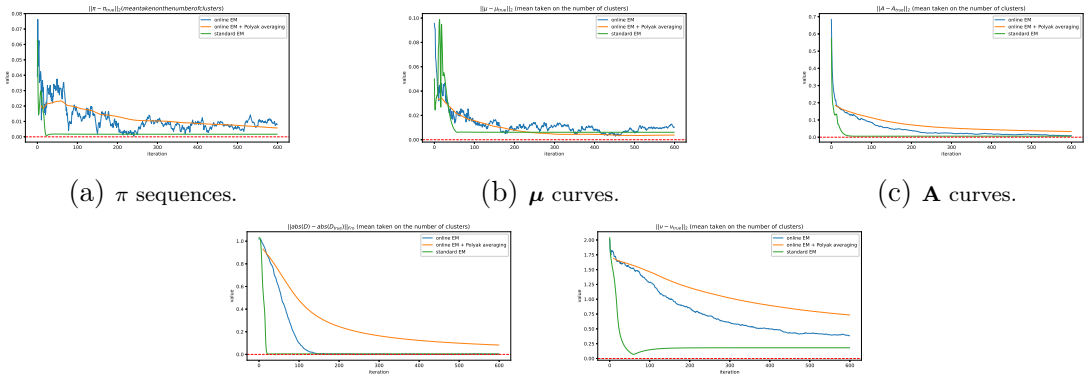


Figure 10: $\mathcal{MMST}$ simulation in 2D with 4 components.



(a) π sequences. (b) $\boldsymbol{\mu}$ curves. (c) $\mathbf{A}$ curves.

(d) $\mathbf{D}$ curves. (e) ν curves.

Figure 11: Convergence results for the parameters. (Online EM iterations values in blue, Online EM Polyak averaging in orange and Standard EM iterations values in green)

	Standard EM	Online EM
2D	30 sec	27 sec
3D	110 sec	53 sec

Table 1: Online EM vs standard EM computation time complexities in 2D and 3D with 4 components until convergence with fixed tolerance at 10^{-2}.

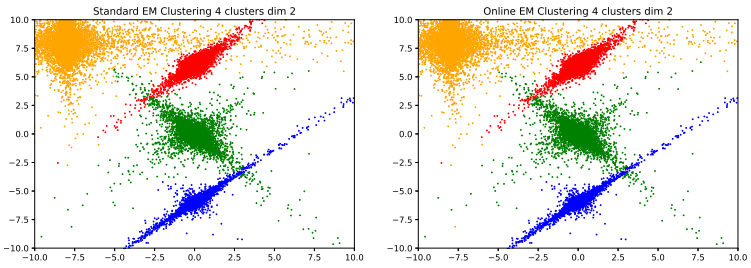


Figure 12: Standard EM (Left) vs online EM (Right) clustering result.

5 Unsupervised anomaly detection: principle & methods

In this section, we explain the general framework of UAD and the different methods that we test / use.

5.1 Measure of coherence

If we recall the context, we know have a reference model ($\mathcal{MMST}$) with trained parameters (online EM). The idea here is, given a random point y, to be able to answer to the question:

$$\textit{Is } y \textit{ an anomaly?}$$

To do so, we need to find a sort of distance between y and the reference model, that we will preferably call a measure of coherence towards the reference model.

Empirically the lower the likelihood the more abnormal is an observation, and using Equation 3.6, we remark that the weights w of this mixture:

$$\begin{aligned} w_m^y &= \mathbb{E}\{w_m | Y = y\} \\ &= \sum_{k=1}^{K} \mathbb{P}\{Z = k | Y = y\} \mathbb{E}\{w_m | Y = y, Z = k\} \\ &= \sum_{k=1}^{K} r_{yk} u_{mk}, \end{aligned}$$

can be used as a measure of proximity of the m^{th} dimension of the point y with r_{yk} and u_{mk} defined in subsubsection 3.4.2. We also note that the higher the weight of a point, the better it is, this is why for the rest of the study we will assume that if a point has a high weight in one dimension then it is a normal point so that our final proximity measure is defined as:

$$w^y = \max w_1^y \dots w_m^y.$$

Indeed as in Figure 13 we make the choice to characterize a point as a regular point as soon as it is a regular point for one of the dimensions.

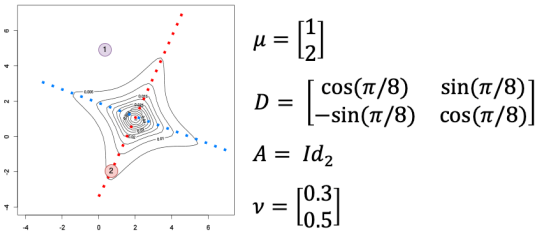


Figure 13: $\mathcal{MST}$ distribution with one outlier (1) and an extreme "regular" point (2).

5.2 Design of an anomaly detection rule

Now that we have defined our coherence measure, we need to find a threshold for this measure so that if $w^y < tresh$ then y is an anomaly. This can be defined by fixing an acceptable False Positive Rate (FPR) α, and finding the $1-\alpha$ quantile, of the weights distribution on all the standard points used for the training such that for any point p, $\mathbb{P}\{w_p \leq t_\alpha\} = \alpha$. Then we have the following rule for the abnormal voxel detection:

Given a point y and its coherence value w^y, y is abnormal if $w^y \leq t_\alpha$.

Another measure of coherence, the log-score: During our study, we also tried to use what we call the log-likelihood score, which is for a point y and with f, the likelihood function obtained after the EM algorithm, $\log f(y)$. Intuitively, the lower is this value, the better the chances are that y is an anomaly. Unfortunately, in the application it appears that this measure gives worst results that the measure based on the weights.

6 Application to anomaly detection in medical images

6.1 Parkinson disease (PD) Overview

Parkinson's disease (PD) is an ever-present neurodegenerative condition with a prevalence of 66-1500 per 100,000 in Europe and 10,000,000 cases worldwide [13]. In 1817, James Parkinson described first the disease where patients have "Involuntary tremulous motion, with lessened muscular power, in parts not in action and even when supported; with a propensity to bend the trunk forward and to pass from a walking to a running pace: the senses and intellects being uninjured" [11]. Age is an important factor of the disease, which is tagged as rare before 50 years and increasing prevalence after 60 years, and the disease affects more the male with a 3:2 ratio male-female [13]. For more than 200 years now, tremendous effort has been made to make the life of PD patients easier lowering the symptoms burden of the disease. The mortality has been dramatically reduced passing from a 3:1 ratio of deaths to a 1.52:1 ratio between 1990 and 2010. The motor symptoms of the disease can appear 20 years after the first symptoms as depression, rapid eye movement disorder that makes a very early diagnosis very hard to detect. Usually, as in Figure 14, the diagnosis appears as the first motor symptoms appears. Today, there is no way for radiologist or neurologist to detect PD patients before the early stage stated in Figure 14. This is why a lot of effort is put in the ability to find new biomarkers for the early detection of Parkinson's disease so that the medical staff can tackle the disease earlier. In our work, based on MR imaging, we aim to:

- Be able to diagnose early stage PD patients.
- To find new biomarkers (lesion location in the brain) for the early diagnosis of Parkinson's disease.

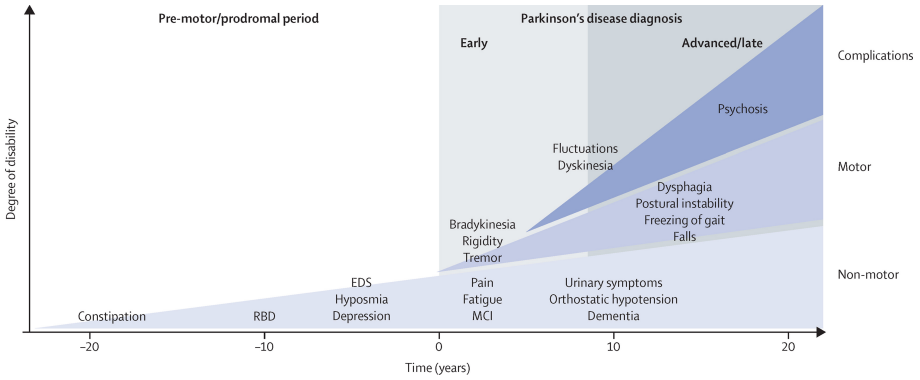


Figure 14: Clinical symptoms and time course of Parkinson's disease progression. [13]

6.2 Medical Imaging: MRI Qualitative Imaging

Anatomical MR images provide detailed information about the shape and size of brain regions *in vivo*. The MRI has been possible thanks to the work of Paul Lauterbur and Peter Mansfield that greatly contributed to it. The main steps of MRI are illustrated in Figure 15.

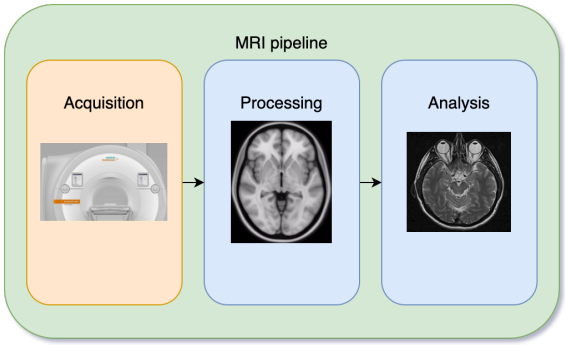


Figure 15: MRI pipeline illustrated.

The acquisition part is not under study here. The processing part will be study and the analysis is the core of this report.

The MRI data are quite complex and depend on a certain number of factors. First, the data are liable of a number of artifacts as head movement during the acquisition. There are also obviously a lot of brain anatomical variability between individuals. The MRI data are also very large, indeed typically a voxel (the 3D version of a pixel) represents a volume of order 1mm $\times$ 1mm $\times$ 1mm, hence for one individual it is possible to have millions of voxels. A major part of MRI analysis is to deal with those issues, including (none exhaustive list) [24]:

- **Quality control**: presence of artifacts
- **Motion correction**: realignment of scans for one individual
- **Spatial normalization**: Transport all the images of different individuals in one common referential space where we can compare them all, and conduct a group study.

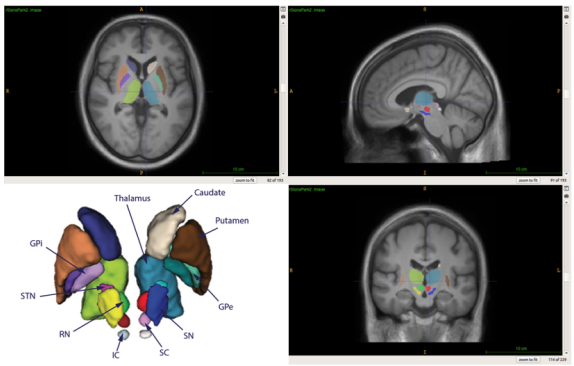


Figure 16: Subcortical structures. [31]

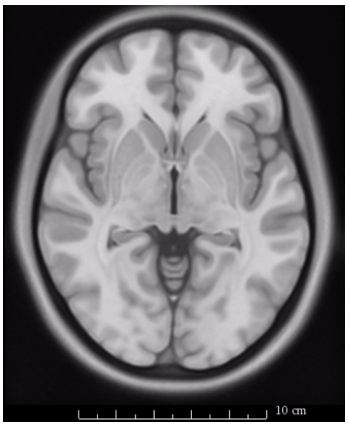


Figure 17: MNI space.

- **Spatial smoothing**: Voluntary blurring of the data to reduce the noise in the images and inter-individual variability.

MR images are made by the response to a magnetic field of protons of water molecules present in the human body to particular excitation. Since protons from different tissues will react differently, the different structures of the brain can be contrasted.

What we call T1-weighted images differentiate tissues by their longitudinal relaxation times. Fat quickly realigns its longitudinal magnetization with the magnetic field thus it appears bright, on the contrary, cerebrospinal fluid presents in sulci and ventricules, has a much slower longitudinal magnetization realignment and therefore appears dark. T1-weighted images are largely used for anatomical exploration since the tissues and structures in the brain are clearly distinguishable.

However, subcortical structures, like the Substantia Nigra, the subthalamic nucleus, the globus pallidus and the red nucleus (see Figure 16) that are areas of interest for Parkinson's Disease [26] are not really distinguishable in standard T1-weighted images.

Diffusion Imaging monitors the displacement of water molecules in the brain, again, the diffusion of water molecules depends on the histological properties of the tissues [15]. It provides unique information about myelin fibers that constitute the white matter.

What we call Diffusion Tensor Imaging (DTI) characterizes the direction of displacement. DTI requires the acquisition of at least six diffusions weighted images, each obtained with different orientations. From there, we can obtain two other quantitative measures. The Mean Diffusivity (MD) which is a scalar value that quantify the rotationally invariant magnitude of water diffusion. And the Fractional Anisotropy (FA) that is also a scalar value describing the degree of anisotropy in water displacements.

The MNI space

When we work on a group study, there are a lot of different individuals with all their particularities that makes their brain unique. That is why all the images are different, a voxel at position (x, y, z) is not necessarily belonging to the same structures for every subject of the study. We thus need a way to normalize all those images so that a particular voxel can be compared between all the subjects [3].

To do so the MNI space has been computed by using 152 subjects, it is roughly a representative image of the normal brain. Then given a potential subject, the idea is to project the MRI image onto this MNI space, this often implies an interpolation (to create a new image based on the deformation field computed to move the image from the individual space to the MNI space) and thus, a loss of information. In Figure 17, there is an illustration of one axial slice (fixed z coordinate) of the MNI space, we note that it is quite blurred. Intuitively this blurring seems normal if we interpret the MNI space as the average brain computed on 152 different brains.

6.3 Datasets

6.3.1 PPMI

The Progression Parkinson's Marker Initiative (PPMI[2]) [16] is a landmark study collaborating with partners around the world to create a robust open-access data set and biosample library to speed scientific breakthroughs and new treatments. This dataset contains 57 healthy subjects (named controls) and 130 *de novo* PD patients (*ie*: Recently diagnosed) for each of the subject a diffusion image is available, from which we can extract the Mean Diffusivity and the Fractional Anisotropy.

As we have seen each patient is unique, this is why so as to be able to compare the subjects between them, we need to put all the subjects in the same space. We then choose the MNI space for our study group, this way, we are now able to compare each voxel value at the same common location in tissues or structures.

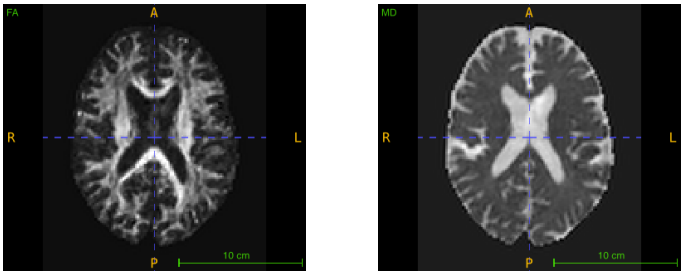


Figure 18: FA (Left) and MD (Right) for a subject from the PPMI dataset.

6.3.2 AGIR-Park study

This dataset is a lot smaller compared to PPMI with only 8 controls subjects and 20 patients. All images have been acquired on a Phillips 3T scanner at the IRMaGe platform in Grenoble.

This study is composed of 11 females and 9 males *de novo* PD patients that are between 53 and 92 years old, and 4 healthy females and males who are between 47 and 82 years old.

For each subject, we have:

- Anatomical T1 image
- DTI scan from which FA and MD features have been previously extracted

[2] `https://www.ppmi-info.org/about-ppmi`

- Blood perfusion scan: defined as the volume of blood passing through a given amount of brain tissue per unit of time.

As pre-processing we chose to co-register all individual images on the corresponding T1 scan and we normalize the Neuromorphometric Atlas onto the T1 space of each subject. An Atlas is simply a segmentation of the brain generally in the MNI space delimiting its different structures.

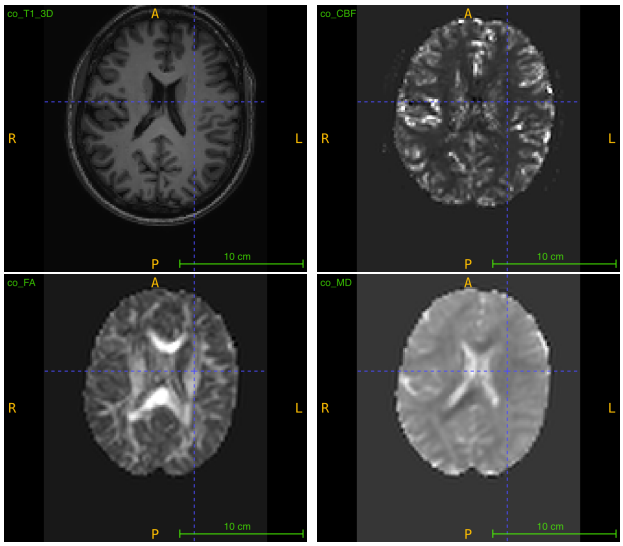


Figure 19: T2 (Top Left), CBF (Top Right), FA (Bottom Left), and MD (Bottom Right) for a subject from the Agir-Park dataset.

Note that the post-processing applied for the AGIR-Park data set is better than the one applied for the PPMI data set since we do not bring all the subjects in one common spatial space, we only transform the atlas to the spatial space of the subject.

6.4 Unsupervised anomaly detection: proposed method

According to previous work as in [2], using mixtures of multivariate generalized students distributions has been pretty successful in unsupervised detection of large tumor in brain rats. But the case of large tumors is "easier" to detect in the sense that the human eye can easily remark them. In [27], it has been shown that Diffusion and Perfusions imaging contains good characteristics in detecting anomalies in the brain of *de novo* Parkinsonian patients.

During her thesis [19] Veronica Munoz Ramirez, have been unable to use this approach on large large data sets. Indeed, detecting anomalies using a mixture model implies using the EM algorithm to estimate the mixture parameters. But in the standard EM version all the data need to be passed throw the algorithm in one shot. With a small number of subjects, this is achievable, but when the data grow loading all the data into the RAM becomes difficult.

[19] also use a completely different method by using Variationnal Autoencoders (VAE) [20]. By using VAE, [19] got compelling results with a good accuracy implying that the network was indeed detecting lesion from PD in the brain.

Now, given the sequential version of the EM algorithm we have seen previously for $\mathcal{MMST}$ model, our work will be to use all the possible data of PPMI so as to compute a mixture model, and this time we shouldn't be annoyed by RAM issues or memory complexity problems. Then we will be able to compare the results with the VAE method.

6.4.1 Estimation of the reference model (PPMI)

For this step the goal is to estimate what we call an *reference model*. In the PPMI dataset, we have two different kinds of subjects: the PD patients and the control subjects (*ie*: healthy subjects).

Let's denote a voxel v and $\mathcal{V}_H$ the set of voxels that belong to control subjects. We also note $Y_H = \{y_v | v \in \mathcal{V}_H\}$ as the set of features vectors of each voxel for each control, with $y_v = (FA_v, MD_v) \in \mathbb{R}^2$.

When we refer to estimate a *healthy* model, it means estimating the parameters of the supposed distribution of the y_v.

A first visualization for a control in Figure 20 shows that we can't simply model the data with a Gaussian Mixtures mostly because of the high tail that appears in this plot that would probably be seen as another cluster with a Gaussian mixtures model.

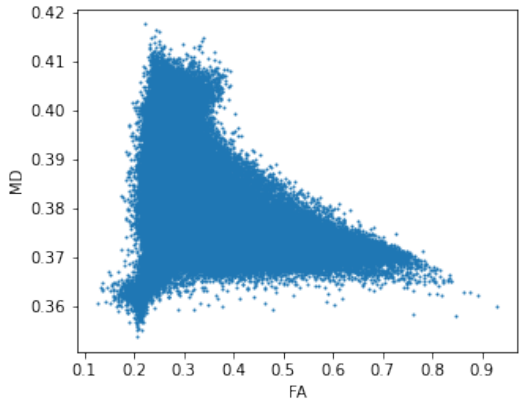


Figure 20: FA - MD scatter plot for a control.

The observation in Figure 20 shows why we use mixture of $\mathcal{MST}$ distributions to model the data. In addition we will be able to use the sequential version of the EM algorithm.

The number of clusters that is an input parameter of the EM algorithm can be determined by using the Bayesian Information Criterion (BIC) , that introduces a penalty on the amount of data to the maximum log-likelihood value so as to prevent overfiting.

After running the online EM algorithm we obtain the parameter $\boldsymbol{\theta}_{\mathcal{M}}$ that is the maximum likelihood estimator so that the features vectors y_v from healthy voxels approximately follow a $\mathcal{MMST}$ distribution of parameter $\boldsymbol{\theta}_{\mathcal{M}}$, we also have an assigned cluster for each voxel.

For a control and for each voxel we can then assign a cluster resulting in a complete segmentation of the brain as in Figure 21 , where we used $K_{BIC} = 9$. We

note that the segmentation highlights some structures of the brain as the ventricles and some part of the white matter.

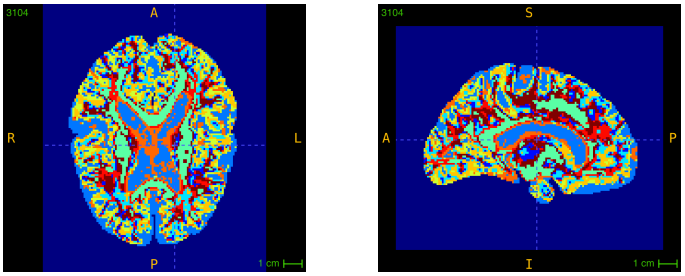


Figure 21: Segmentation result $K = 9$ on a control subject.

Finally, we have seen that the pre-processing done in the PPMI dataset can greatly affect the boundaries of the structures, and empirically, we find that the boundaries of the ventricles are often badly processed. But in Figure 21, it appears that those boundaries form a specific class, hence they should not be detected as anomalies in further steps.

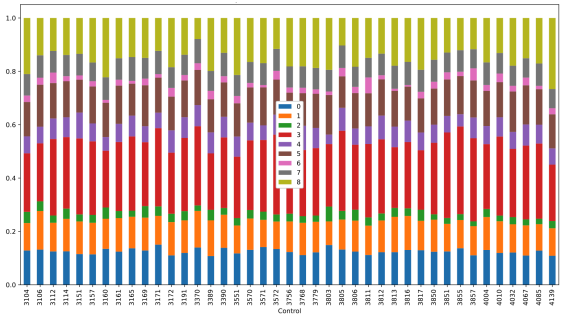


Figure 22: Signatures of each controls from PPMI dataset used for training giving the proportion of each class for each subject.

6.4.2 Anomalies detection by comparison to the healthy model (with PPMI).

Now, we have a voxel-wise model trained on healthy subjects, and we would like to discriminate PD subjects from the healthy ones. An idea is then to detect abnormal voxels for each subject, then we will define a threshold counting the number of abnormal voxels that will characterize PD subjects from healthy subjects.

First there is a need to define what is an abnormal voxel. To do so we use the rule described in subsection 5.1.

Given the anomaly detection rule, for each test controls and patients, we can count the number of abnormal voxels and find the best threshold N_{ab} (in number of anomalies) that maximize a chosen metric such that if a subject as N abnormal voxels with $N > N_{ab}$ then the subject is classified as PD and *vice-versa*.

6.4.3 Validation and testing (PPMI)

To be able to compare our result with the VAE method in [20], we are going to describe and use the exact same pipeline for the validation of the model.

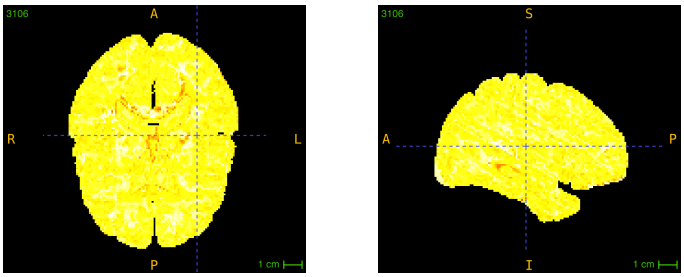


Figure 23: Weights map with $K = 9$ on a control subject.

The method is to validate the model using a 10-folds cross-validation. Since we are using the exact same data as in [20], we are also going to use the exact same folds. Then, for each fold we are using 42 controls to train the model and 15 controls to test the model, finally all the patients are used for the testing part. Once the model is trained, we compute for all the voxels and all the subjects their weights.

Then, knowing that we also want to possibly characterize structures that are possibly more affected by PD (we already know that white matter and subcortical structures as the Substantia Nigra, Red Nucleus are quite affected by PD), we are going for each structure s to choose an appropriate threshold t_{α^s}, in our case for a 9 class model, we choose $\alpha^s = 0.25\% = \alpha$ for all structures and t_α is computed empirically on the training controls. We finally compute the number of abnormal voxels for all the test controls and the patients.

Now, we note that we have a very imbalanced testing data set with a 15:130 test controls-patients ratio so we have to find metrics that will compensate this imbalance, for instance choosing the accuracy will be a very bad choice. In addition, a good choice would be to choose the G-mean metric which is appropriate for imbalance data sets [14], with $gmean = \sqrt{Sensitivity \times Specificity}$ and $Sensitivity$ = True Positive Rate, $Specificity$ = True Negative Rate, the ROC-AUC is also a good choice since it also takes into account the True Positive Rate along with the False Positive Rate. Hence, those two metrics can gives us a good intuition on how well our model can differentiate PD patients from control subjects. The choice of N_{ab} is done in accordance with the maximization of the g-mean metric. In Figure 24 we got good result compared with the VAE method [20], with a small increase given the g-mean metric and smaller boxes, meaning less variations between the folds, but a small decrease concerning the AUC metric but still with fewer variations between folds. The global results are as expected with some structures that stands out for PD detection as the Substantia Nigra, the White Matter but also the parietal lobe, that is what the authors of [20] noticed also.

What we have seen is that our model can detect more or less PD patients from healthy subjects. What we can do is then to look at the location of the detected anomalies. Intuitively, we can suppose that the "anomalies" detected in the controls are going to be edges of structures, but for the patients, we expect to find grouped anomalies and not simply isolated voxels. This intuition is confirmed for some PD subjects as it is illustrated in Figure 26 where The anomalies for control subjects are detected on the edges of the structures but not for for PD subjects.

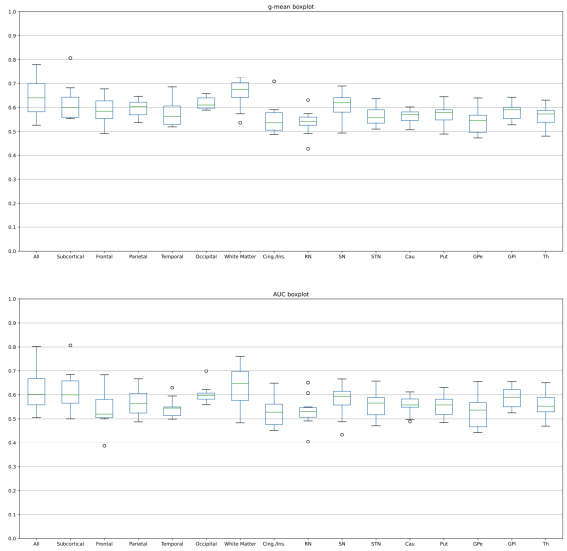


Figure 24: Boxplots representing the results for the g-mean and AUC metric by structures.

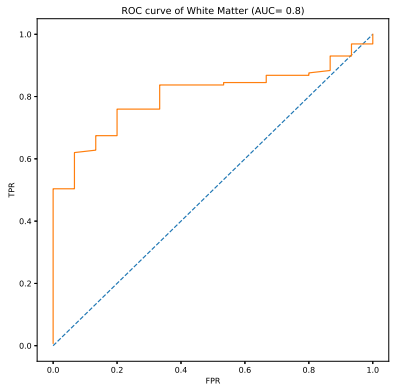


Figure 25: Best ROC curve obtained in the white matter.

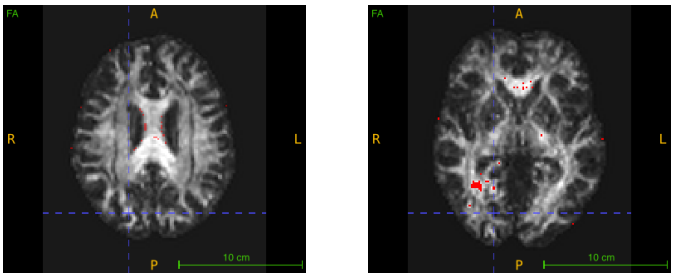


Figure 26: Anomalies in red for a control (Left) and a PD patient (Right).

6.5 Results for AgirPark data set

In order to know if the method is generalisable, we used data from AgirPark study. We have indeed a lot less of data for training and testing but we, this time have another perfusion feature (the CBF). Then, we use the same pipeline as with PPMI, by noting $\mathcal{V}_H$ the set of voxels belonging to healthy subject, with Y_H the set of features containing the $y_v = (FA_v, MD_v, CBF_v) \in \mathbb{R}^3$ for each voxel v. For validation and testing, since we have less data we only did a 5-folds cross-validation, with 6 controls for the training, 2 for the testing and all the patients for the testing. The results in Figure 27 suggests there is a great variability between the folds, controls are often in the right class but it varies a lot for the patients.

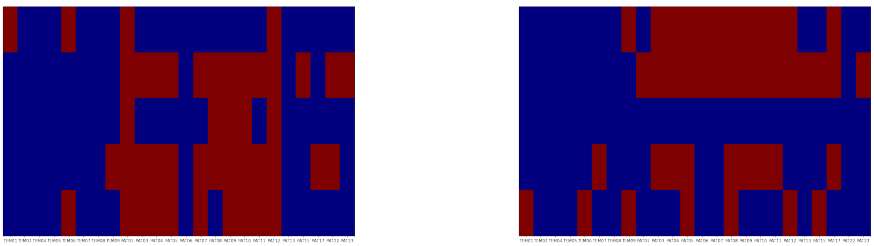


Figure 27: Binary classification results of the subjects from AgirPark (x-axis) over the 5 folds (y-axis) for the entire brain (Left) and the white matter (Right). Red color means wrong classification, blue color means good classification.

7 Conclusion

7.1 Future work

At the end of this report, many lines of thought to continue the work opened up.

Online EM Part of them consists in the improvement of the online EM algorithm. Starting by exploring acceleration methods for convergence influencing on the choice of the step size for the update of the statistics in Equation 3.2, but also by acting on acceleration of the stochastic approximation using methods as ZAP [18]. Finally it would also be interesting to study the influence of the batch size, and the way in which to pass the data in the algorithm. Indeed, in our case we could think about passing data several times by "working with epoch" in the algorithm.

Features improvement In our study, we only studied the data using the quantities FA, MD, CBF. However we could take advantage of the work done in [20] using more complex features learned by the VAE to feed the EM algorithm.

Coherence measure improvement One of the problematic aspect in the notion of measure of coherence developed in subsection 5.1 is that we are given a multi-dimensional model and we reduce it to one scalar value that may not be a proper distance. An idea we wish to elaborate on in the future is the notion of multivariate quantile [12]. Indeed, using multivariate quantile regression [7] could help us to define multivariate quantile regions so that we should loose less information than with the current coherence measure.

Account for the spatial information As we have seen, the reference model is currently calculated on all the voxels, without taking their locations into account. However, the location of a voxel is an important information, as we have seen in subsubsection 6.4.3, where we see that certain areas of the brain are more sensitive to Parkinson's disease than others. Taking into account the spatial dependency between voxels could be done using patches of voxels as the items of interest rather than each voxel alone. More traditionnaly, Hidden Markov Random Fields (HMRF) models are also good candidates for this goal. In particular, it would be interesting to generalize our use of online learning and stochastic approximation schemes in the

Markov field context. This would both save time in parameter estimation and allow larger data sets to be processed.

7.2 Contributions

This work allowed the implementation of the online EM algorithm based on the exponential form of the $\mathcal{MMST}$, and provided an efficient way to deal with the M-step for the D parameters (orthogonal matrices). It also lifted the limit on the amount of data that could be processed with the standard EM algorithm, dramatically reducing computing time and memory management issues. Finally, it proposed of an effective coherence measure for anomaly detection based on quantities that can be computed from the mixture model and that can be interpreted as weights.

Data and Code availability: Data used in this report were obtained from the Parkinson's Progression Markers Initiative (PPMI) database (www. ppmi-info.org/data). For up-to-date information on the study, visit www.ppmi-info.org. PPMI - a public-private partnership - is funded by the Michael J. Fox Foundation for Parkinson's Research and funding partners, including Abbvie, Allergan, Avid Radiopharmaceuticals, Biogen, BioLegend, Bristol-Myers Squibb, Celgene, Denali, GE Healthcare, Genentech, GlaxoSmithKline, Lilly, Lundbeck, Merck, Meso Scale Discovery, Pfizer, Piramal, Prevail Therapeutics, Roche, Sanofi Genzyme, Servier, Takeda, Teva, UCB, Verily, Voyager Therapeutics and Golub Capital.

The code that supports the findings of this report are available at Github[3].

[3] `https://github.com/geoffroyO/OLEmMMST`

Appendices

A Details on the M-step

A.1 Optimization manifolds

In this part we introduce some tools and notions to be able to solve the following problem:

$$\min_{x \in \mathcal{M}} f(x), \tag{A.1}$$

with $\mathcal{M}$ a manifold, and a function $f : \mathcal{M} \mapsto \mathbb{R}$.

The main references used are [1] and [9]. The explicit definition of a manifold will not be explained here and is given in [1].

A.1.1 Notions

Theorem 1. $\mathbb{R}^{n \times p}$ *(set of real matrix of size* $n \times p$*,* $\mathcal{S}_n(\mathbb{R})$ *(set of real symmetric matrices of size* $n \times n$*),* $St(p, n)$ *(set of real matrices of size* $n \times p$ *with orthogonal columns) are manifolds.*

In addition, we have

$$dim(St(p, n)) = np - \frac{1}{2}p(p + 1). \tag{A.2}$$

A simple proof can be done to show that $St(p, n)$ is a manifold using the *submersion theorem* given in [1].

A.1.2 Tangent space and derivation

To solve Equation A.1 we are tempted to use a kind of gradient descent. In $\mathbb{R}^n$, the direction of the steepest descent if given by

$$\eta = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{d \in \mathbb{R}^n, ||d|| < \epsilon}{\arg\min} f(x + d). \tag{A.3}$$

The main issue with this definition is that $x + d$ makes no sense in $St(p, n)$ since it is not a vector space. This is why we need to investigate for a method to define differentiability over a manifold.

The intuitive approach is to consider a smooth curve $\gamma : [0, 1] \mapsto \mathcal{M}$ on a manifold $\mathcal{M}$ through x (*ie:* $\gamma(0) = x$), and to generalize the directional derivative by using $\frac{d(f(\gamma(t)))}{dt}\Big|_{t=0}$. We can now define the tangent space of a manifold at some point.

Definition 1. We define the tangent space of a manifold $\mathcal{M}$ at x as the set,

$$T_x\mathcal{M} = \{\dot{\gamma}(0), \text{ such that } \gamma \text{ is a smooth curve and } \gamma(0) = x\}. \tag{A.4}$$

Now that we have define the tangent space of a manifold, we have to find a way to define a metric in the tangent space so that we are able to solve the steepest descent as in Equation A.3. This can be done by endowing the tangent space $T_x\mathcal{M}$ ($x \in \mathcal{M}$)with an inner product $< ., . >_x$. A manifold whose tangent spaces are all

endowed with a smoothly varying inner product is called a *Riemannian manifold*, and we note $g =< .,. >$ the associated metric. Also remark that a vector space endowed with an inner product is a *Riemannian manifold* called an *Euclidian space*.

We can now give a definition of the gradient of a function defined over a manifold,

Definition 2. The gradient at $x \in \mathcal{M}$ of a scalar function $f : \mathcal{M} \mapsto \mathbb{R}$ defined over a *Riemannian manifold*, $(\mathcal{M}, g)$ denoted by $\text{grad}_{\mathcal{M}} f(x)$ is the only vector (*Riesz representation theorem*) of $T_x\mathcal{M}$ that satisfies:

$$< \text{grad}_{\mathcal{M}} f(x), \xi >= Df(x)[\xi] \text{ for all } \xi \in T_x\mathcal{M}. \tag{A.5}$$

Now, we have seen that $St(p, n)$ is in fact a submanifold of $\mathbb{R}^{n\times p}$, its tangent spaces can then simply inherits from the inner product of $\mathbb{R}^{n\times p}$, it's say it the Froebenius inner product $< A, B >= tr(A^T B)$, and it is possible to define for each tangent spaces the orthogonal projection, for $X \in St(p, n)$, $P_X : T_X\mathbb{R}^{n\times p} \mapsto T_X St(p, n)$ such that for all $\xi \in T_X\mathbb{R}^{n\times p}$ (we can also remark that $T_X\mathbb{R}^{n\times p} = \mathbb{R}^{n\times p}$):

$$< \xi - P_X(\xi), P_X(\xi) >_x = 0. \tag{A.6}$$

Proposition 1. *We finally get using Equation A.6 and Definition Equation A.5 that:*

$$\text{grad}_{St(p,n)} f(X) = P_X(\text{grad}_{\mathbb{R}^{n\times p}} f(X)). \tag{A.7}$$

Proposition 2. *The orthogonal projection $P_X : T_X\mathbb{R}^{n\times p} \mapsto T_X St(p, p)$ is given by*

$$P_X(Z) = (\mathbf{I} - XX^T)Z + X\text{skew}(X^T Z), \tag{A.8}$$

with $\text{skew}(M) = \frac{1}{2}(M - M^T)$.

Doing the computation as in [9], finally gives us, by denoting $\widetilde{\nabla}$ the gradient on the Stiefel manifold the formula

$$\widetilde{\nabla}_X f = \nabla_X f - X\nabla_X f^T X. \tag{A.9}$$

A.1.3 Retraction and Parallel transport

In the context of optimization we are now going to define two points of interest that will help us to fully understand the concept of Riemaniann optimization:

- Retraction.
- Parallel transport of vector from a tangent space to another.

Definition 3 (geodesic)**.** A geodesic of a manifold $\mathcal{M}$ is a smooth curve with zero acceleration,

$$\frac{d^2\gamma(t)}{(dt)^2} = 0. \tag{A.10}$$

Without entering in the detail, we note the application that maps $T_x\mathcal{M} \mapsto \mathcal{M}$, and which associate at $\xi \in T_x\mathcal{M}$ the value $\gamma(1)$, with γ being the geodesic such that $\gamma(0) = x$ and $\dot{\gamma}(0) = \xi$, $exp_x(\xi) = \gamma(1)$ and it is called the *Riemaniann exponential*.

Retraction
So far we have seen that the standard steepest descent makes no sense in a Riemaniann manifold because it is not necessarily a vector space. But previously we have been able to define a metric and a proper definition of the tangent space of a manifold at a given point allowing us to define the gradient for scalar functions defined on manifold. Given the gradient and a stepsize we could obtain the steepest descent by using a retraction which is an application that maps vectors from $T_x\mathcal{M}$ to $\mathcal{M}$. The natural retraction that we can use is the *Riemaniann exponential*, but we have to be careful about the fact that computing geodesics can be very long.

Definition 4 (Retraction)**.** A retraction is an application that maps a vector of the tangent space at some point to an element of the manifold, it is noted $R_x : T_x\mathcal{M} \mapsto \mathcal{M}$.

Optimization
With what we have seen so far, we can now define the iterates of a Riemaniann Gradient descent of a function $f : \mathcal{M} \mapsto \mathbb{R}$, given a step $t_i > 0$ as:

$$x_{k+1} = R_{x_k}(-t_k \text{grad}_{\mathcal{M}} f(x_k)). \tag{A.11}$$

Parallel transport
In some optimization methods, we will sometimes need to compare the gradients of previous iteration with the actual gradient. The issue here, with manifolds is that those two gradients (the current and the previous ones) belongs to two different tangent spaces, so $\xi_{x_{k+1}} - \xi_{x_k}$ for $\xi_{x_{k+1}} \in T_{x_{k+1}}\mathcal{M}$ and $\xi_{x_k} \in T_{x_k}\mathcal{M}$ is ill defined. To counter that, intuitively we would like to have an operation that brings ξ_{x_k} into the current tangent space $T_{x_{k+1}}\mathcal{M}$.

Given the iterates in Equation A.11, we know that x_{k+1} is in fact equal to $R_{x_k}(\xi_k)$, we then only need to compare ξ_{x_k} with $\xi_{R_{x_k}(\xi_k)}$.

This is possible by using a vector transport for which we will not give a rigorous definition that you can find in [1].

Definition 5 (Transport Vector)**.** A Transport Vector is a smooth application $\mathcal{T}$ that given a point $x \in \mathcal{M}$, and given two tangent vectors $\xi, \eta \in T_x\mathcal{M}$ transport the vector η into the tangent space of $R_x(\xi)$.

A.2 Conjugate Gradient: the Euclidian case

In this paragraph we are considering the following problem:

$$\min_{x \in \mathbb{R}^n} x^T Q x - b^T x, \tag{A.12}$$

with $Q \in \mathbb{R}^{n \times n}$ positive definite.

Definition 6 (Q-conjugacy)**.** Let Q be a symmetric matrix, $(d_1, ..., d_k)$ a family of non-zero vectors in $\mathbb{R}^n$. This family is said to be Q-conjugate if for all $i \neq j$ we have:

$$d_i^T Q d_j = 0. \tag{A.13}$$

From this definition the most important property that makes the conjugate gradient method very useful for this kind of problems comes out.

Proposition 3. *If $Q \succ 0$, and $(d_1, ...d_k)$ with $k \leq n$ is a family of Q-conjugate vectors then they are linearly independent.*

Proof. We take $\alpha_1, ...\alpha_k \in \mathbb{R}$, and we suppose that $\alpha_1 d_1 + \cdots + \alpha_k d_k = 0$, we then have, multiplying by $d_1^T Q$, $\alpha_1 d_1^T Q d_1 + \cdots + \alpha_k d_1^T Q d_k = \alpha_1 d_1^T Q d_1$ by Q-conjugacy. But Q is positive definite and d_1 is non-zero vector so $d_1^T Q d_1 > 0$, then necessarily we have $\alpha_1 = 0$, doing the same for the other indices, we get $\alpha_1, \ldots, \alpha_k = 0$ and the family is linearly independent. □

We now see why this definition of Q-conjugacy is useful. It is because we can write the optimal solution x^* of Equation A.12 as a linear combination of $(d_1, \ldots, d_n)$ of Q-conjugate vectors that is also a basis of $\mathbb{R}^n$.

Also, because $Qx^* = b$ (that comes from the fact that x^* is a critical point then the gradient is null), we have if $x^* = \alpha_1 b_1 + \cdots + \alpha_n b_n$ that $\alpha_i = \frac{d_i^T b}{d_i^T Q d_i}$.

We remark that given a basis of $\mathbb{R}^n$ and $Q - conjugate$ vectors that x^* can be construct in only n iterations.

Finally the remaining issue, is that we would like to be able to construct the n Q-conjugate vectors step by step during the iterations. This gives algorithm Algorithm 1. It still remains to show that the α_k are the good ones for the d_k vectors and that the d_k vectors form a Q-conjugate family.

Algorithm 1 Conjugate Gradient Algorithm Quadratic function

$g_0 \leftarrow b - Qx_0$
$d_0 = -g_0$
for k=0...n-1 **do**
  $\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$
  $x_{k+1} = x_k + \alpha_k d_k$
  $g_{k+1} = Qx_{k+1} - b$
  if $k \neq n-1$ **then**
    $\beta_k = \frac{g_{k+1}^T Q d_k}{q_k^T Q d_k}$
    $d_{k+1} = -g_{k+1} + \beta_k d_k$
  end if
end for

Generalization of the Conjugate Gradient Algorithm

Our goal is now to solve the following problem, for a scalar function twice differentiable:

$$\min_{x \in \mathbb{R}^n} f(x). \tag{A.14}$$

The function f may not be quadratic but we can make the approximation, using the Taylor expansion:

$$f(x) \approx f(x_0) + \nabla f(x_0)^T(x - x_0) + (x - x_0)^T \nabla^2 f(x_0)(x - x_0). \tag{A.15}$$

Then, we can derive an conjugate gradient version to solve this new problem as in Algorithm 2. The issue being that the algorithm will not finish in n steps so what we can do is to repeat Algorithm 2 as much as needed by replacing $x_0 \leftarrow x_n$ for the new initialization.

Algorithm 2 Conjugate Gradient Algorithm Generalization

$g_0 \leftarrow \nabla f(x_0)$
$d_0 = -g_0$
for k=0...n-1 **do**
  $\alpha_k = -\frac{g_k^T d_k}{d_k^T [\nabla^2 f(x_k)] d_k}$
  $x_{k+1} = x_k + \alpha_k d_k$
  $g_{k+1} = \nabla f(x_{k+1})$
  if $k \neq n-1$ **then**
    $\beta_k = \frac{g_{k+1}^T [\nabla^2 f(x_k)] d_k}{d_k^T [\nabla^2 f(x_k)] d_k}$
    $d_{k+1} = -g_{k+1} + \beta_k d_k$
  end if
end for

We can note that the conjugate gradient method is in between the steepest descent and the second order Newton methods. The difference here with the Newton methods is that we don't need to compute the inverse of the Hessian.

There exists also some other rules for the iterates of β that are called the β-rules. They give the exact results in the quadratic case and avoid the computation of the Hessian.

Proposition 4 (Polak-Ribiere line search)**.** *The Polak-Ribiere β-rule gives the following update for the iterates of β:*

$$\beta_k = \frac{(g_{k+1} - g_k)^T g_{k+1}}{g_k^T g_k}. \tag{A.16}$$

A.3 Conjugate Gradient: the Riemanian case

A.3.1 The algorithm in the Riemanian case

With the previous notations and notions, we have introduced we are now able to fully derive the Riemanian Conjugate Gradient algorithm. The main differences compared to the Euclidian case is that for next iterate x_{k+1} we need to do a retraction, and the computation of the β_k is also slightly different since it implies the gradient of previous iterations which then lies on a different tangent space than the current one.

Definition 7 (The Riemanian Conjugate Gradient)**.** The iterates of the Riemanian Conjugate Gradient are the following:

$$x_{k+1} = R_{x_k}(\delta_k), \tag{A.17}$$

with $\delta_k = \text{grad}_{\mathcal{M}} f(x_k) + \beta_k \delta_{k-1}$.

By noting $P_{x_{k+1} \leftarrow x_k}$ the parallel transportation a vector from $T_{x_k}\mathcal{M}$ to $T_{x_{k+1}}\mathcal{M}$ we have the modified Polak-Ribiere line search:

$$\beta_k = \frac{< \text{grad}_{\mathcal{M}} f(x_{k+1}), \text{grad}_{\mathcal{M}} f(x_{k+1}) - P_{x_{k+1} \leftarrow x_k} \text{grad}_{\mathcal{M}} f(x_k) >_{x_{k+1}}}{||\text{grad}_{\mathcal{M}} f(x_k)||^2_{x_k}} \tag{A.18}$$

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. Optimization algorithms on matrix manifolds. In *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[2] Alexis Arnaud, Florence Forbes, Nicolas Coquery, Nora Collomb, Benjamin Lemasson, and Emmanuel L Barbier. Fully automatic lesion localization and characterization: Application to brain tumors using multiparametric quantitative mri data. *IEEE transactions on medical imaging*, 37(7):1678–1689, 2018.

[3] Matthew Brett, Ingrid S Johnsrude, and Adrian M Owen. The problem of functional localization in the human brain. *Nature reviews neuroscience*, 3(3):243–249, 2002.

[4] Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[5] Gilles Celeux and Gérard Govaert. Gaussian parsimonious clustering models. *Pattern recognition*, 28(5):781–793, 1995.

[6] Juan Antonio Cuesta-Albertos, Alfonso Gordaliza, and Carlos Matrán. Trimmed k-means: An attempt to robustify quantizers. *The Annals of Statistics*, 25(2):553–576, 1997.

[7] Eustasio del Barrio, Alberto Gonzalez Sanz, and Marc Hallin. Nonparametric multiple-output center-outward quantile regression. *arXiv preprint arXiv:2204.11756*, 2022.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[9] Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[10] Florence Forbes and Darren Wraith. A new family of multivariate heavy-tailed distributions with variable marginal amounts of tailweight: application to robust clustering. *Statistics and computing*, 24(6):971–984, 2014.

[11] Christopher Goetz. The history of parkinson's disease: early clinical descriptions and neurological therapies. *Cold Spring Harbor perspectives in medicine*, 1(1):a008862, 2011.

[12] Marc Hallin, Eustasio Del Barrio, Juan Cuesta-Albertos, and Carlos Matrán. Distribution and quantile functions, ranks and signs in dimension d: A measure transportation approach. *The Annals of Statistics*, 49(2):1139–1165, 2021.

[13] Lorraine V Kalia and Anthony E Lang. Parkinson's disease. *The Lancet*, 386(9996):896–912, 2015.

[14] Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, page 179. Citeseer, 1997.

[15] Denis Le Bihan and Heidi Johansen-Berg. Diffusion mri at 25: exploring brain tissue structure and function. *Neuroimage*, 61(2):324–341, 2012.

[16] Kenneth Marek, Danna Jennings, Shirley Lasch, Andrew Siderowf, Caroline Tanner, Tanya Simuni, Chris Coffey, Karl Kieburtz, Emily Flagg, Sohini Chowdhury, et al. The parkinson progression marker initiative (ppmi). *Progress in neurobiology*, 95(4):629–635, 2011.

[17] Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[18] Sean Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press, 2022.

[19] Veronica Munoz Ramirez. *Anomaly characterization in the MRI data of 'de novo 'Parkinson's patients*. PhD thesis, Université Grenoble Alpes, 12 2020.

[20] Verónica Muñoz-Ramírez, Virgilio Kmetzsch, Florence Forbes, Sara Meoni, Elena Moro, and Michel Dojat. Subtle anomaly detection in mri brain scans: Application to biomarkers extraction in patients with de novo parkinson's disease. *medRxiv*, 2021.

[21] Hien D Nguyen, Florence Forbes, and Geoffrey J McLachlan. Mini-batch learning of exponential family finite mixture models. *Statistics and Computing*, 30(4):731–748, 2020.

[22] Hien D Nguyen, Florence Forbes, and Geoffrey J McLachlan. Mini-batch learning of exponential family finite mixture models. *Statistics and Computing*, 30(4):731–748, 2020.

[23] Hien Duy Nguyen and Florence Forbes. Global implicit function theorems and the online expectation–maximisation algorithm. *Australian & New Zealand Journal of Statistics*, 2021.

[24] Russell A Poldrack, Jeanette A Mumford, and Thomas E Nichols. *Handbook of functional MRI data analysis*. Cambridge University Press, 2011.

[25] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[26] Nadya Pyatigorskaya, Cécile Gallea, Daniel Garcia-Lorenzo, Marie Vidailhet, and Stéphane Lehericy. A review of the use of magnetic resonance imaging in parkinson's disease. *Therapeutic advances in neurological disorders*, 7(4):206–220, 2014.

[27] Verónica Muñoz Ramírez, Florence Forbes, Julyan Arbel, Alexis Arnaud, and Michel Dojat. Quantitative mri characterization of brain abnormalities in de novo parkinsonian patients. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 1572–1575. IEEE, 2019.

[28] S Reimão, P Pita Lobo, D Neutel, L Correia Guedes, M Coelho, MM Rosa, J Ferreira, D Abreu, N Gonçalves, C Morgado, et al. Substantia nigra neuromelanin magnetic resonance imaging in de novo parkinson's disease patients. *European Journal of Neurology*, 22(3):540–546, 2015.

[29] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[30] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *arXiv preprint arXiv:1603.03236*, 2016.

[31] Yiming Xiao, Vladimir Fonov, Silvain Bériault, Fahd Al Subaie, M Mallar Chakravarty, Abbas F Sadikot, G Bruce Pike, and D Louis Collins. Multi-contrast unbiased mri atlas of a parkinson's disease population. *International journal of computer assisted radiology and surgery*, 10(3):329–341, 2015.