Unsupervised Clustering of Neural Pathways

Sergio Medina

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Unsupervised Clustering of Neural Pathways

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Sergio Medina

Director: Bertrand Thirion
Codirector: Gaël Varoquaux
Parietal Team, Inria Saclay, Francia, 2013

AGRUPAMIENTO NO SUPERVISADO DE VÍAS NEURONALES

Las Imágenes de Resonancia Magnética de Difusión (dMRI, por sus siglas en inglés) pueden revelar la micro-estructura de la materia blanca del cerebro.

El análisis de la anisotropía observada en imágenes dMRI y destacada por métodos de tractografía puede ayudarnos a comprender el patrón de conexiones entre las distintas regiones del cerebro y a caracterizar trastornos neurológicos.

Debido a la cantidad de información producida por dichos análisis y a la cantidad de errores acarreados de las etapas de reconstrucción, una simplificación de los datos es necesaria.

Los algoritmos de clustering puede ser usados para agrupar muestras que son similares según una métrica dada.

En este trabajo abordamos preguntas importantes relacionadas al uso de clustering para vías neuronales obtenidas mediante dMRI, concretamente: *i)* cuál es la métrica más adecuada para cuantificar la similitud entre dos vías neuronales? *ii)* Cómo seleccionar el mejor algoritmo de clustering y una paramentrización entre las posibilidades estandar?

Mientras intentamos responder estas preguntas, llevamos a cabo una nueva contribución: mostramos como combinar al conocido algoritmo de clustering K-means con varias métricas manteniendo el procedimiento eficiente.

Analizamos el rendimiento y la usabilidad de los algoritmos resultantes en un base de datos de diez sujetos.

Mostramos que la asociación de K-mens con Point Density Model, una métrica recientemente propuesta para analizar estructuras geométricas, se desempeña mejor que otras soluciones usadas actualmente.

Palabras claves: Agrupamiento de vías neuronales - Aprendizaje no Supervisado - Imágenes DWI - Agrupamiento DTI - Modelo de Densidad de Puntos.

UNSUPERVISED CLUSTERING OF NEURAL PATHWAYS

Diffusion-weighted Magnetic Resonance Imaging (dMRI) can unveil the microstructure of the brain white matter.

The analysis of the anisotropy observed in the dMRI and highlighted by tractography methods can help to understand the pattern of connections between brain regions and characterize neurological diseases.

Because of the amount of information produced by such analyses and the errors carried by the reconstruction step, it is necessary to simplify this output.

Clustering algorithms can be used to group samples that are similar according to a given metric.

In this work, we address important questions related to the use of clustering for brain fibers obtained by dMRI, namely: *i)* what is the most adequate metric to quantify the similarity between brain fiber tracts? *ii)* How to select the best clustering algorithm and parametrization among standard possibilities?

While trying to solve these questions, we perform a new contribution: we show how to combine the well-known K-means clustering algorithm with various metrics while keeping an efficient procedure.

We analyze the performance and usability of the ensuing algorithms on a dataset of ten subjects.

We show that the association of K-means with Point Density Model, a recently proposed metric to analyze geometric structures, outperforms other state-of-the-art solutions.

Keywords: Tract Clustering - Unsupervised Learning - Point Density Model - DWI Imaging - DTI Clustering.

AGRADECIMIENTOS

A mis padres Héctor y Adriana y mi hermana Andrea, que son los primeros y mayores responsables de que haya comenzado y terminado una carrera.

A Martín y Chapa, ya que mi decisión de seguir Computación en Exactas siguió a la suya.

A la *gente de la facu*: Bruno, Chapa, Eddy, Martín, Facu, Fer, Giga, Javi, Leo R., Leo S., Luis, Maxi, Mati, Nati, Nelson, Pablo, Palla, Tara, Vivi y Zaiden, ya que fue gracias a ellos que no tiré la toalla a mitad de camino, al empujarnos todos juntos siempre nos dimos fuerzas para seguir.

A los profesores y ayudantes, cuya energía y motivación se veia a cada clase, en todo momento.

A la UBA, por una educación de calidad, de acceso libre, y gratuita.

REMERCIEMENTS

À Pierre, qui après un tout court entretien a décidé de m'embaucher et m'amener en France.

À Vivi, qui tout au début a transferé l'email de l'offre d'emploi et à la fin a collaboré à ce mémoire.

À Bertrand, mon directeur, qui m'a donné l'opportunité de travailler sur ce projet. J'ai dit déjà mille fois merci, mais je sens encore que c'est pas assez.

À Gaël, qui a toujours eu le temps de me donner son précieux avis.

À Régine, Valérie et Élodie, qui ont réussi la difficile, longue et fatigante tâche d'amener un Argentin en France.

À Parietal: Bertrand, Gaël, Alex Jr., Alex G., Michael, Virgile, Fabi, Yannick, Vivi, Jaques, Elvis, Benoît, Bernard et Pierre, qui ont fait de mon endroit de travail un deuxième «chez moi».

CONTENTS

1. INTRODUCTION

Magnetic Resonance Imaging (MRI) is an image modality widely used in radiology to visualize the body's internal structures ever since it was designed in the mid 70's. This non-invasive modality makes use of the nuclear magnetic resonance (NMR) phenomenon to obtain high quality images of tissues' structures. Nowadays clinicians and scientists use these images to detect diseases and analyze the function of organs. Cerebral aneurysms, cerebrovascular accidents, traumas, tumors, rips and inflammations in different organs are some of the main diseases that MRI scans help to characterize or diagnose.

New MRI modalities were developed during the 80's and 90's, Diffusion MRI (dMRI) being one of the most important. It measures the diffusion of water molecules in tissues. Their movement is not free, it shows their interaction with many obstacles as macromolecules, tracts and membranes. The patterns formed by those interactions can reveal microscopic details of the tissues' structure, like the brain's complex tract network. Figure 1.1 shows a dMRI scan and its extracted water diffusion model.

Tracts connect parts of the nervous system and consist of bundles of elongated axons of neurons; they constitute the brain's white matter. These tracts connect distant areas of the brain, on top of the local connections in the gray matter.

Tractography is a process that obtains a model of the neural pathways from the information of the diffusion of water molecules.

Researchers and clinicians use different names to refer to what we call tracts: neural pathways, streamlines, brain fibers, or simply pathways. We will mostly use the term *tract* but we might use any of the others indistinctly.

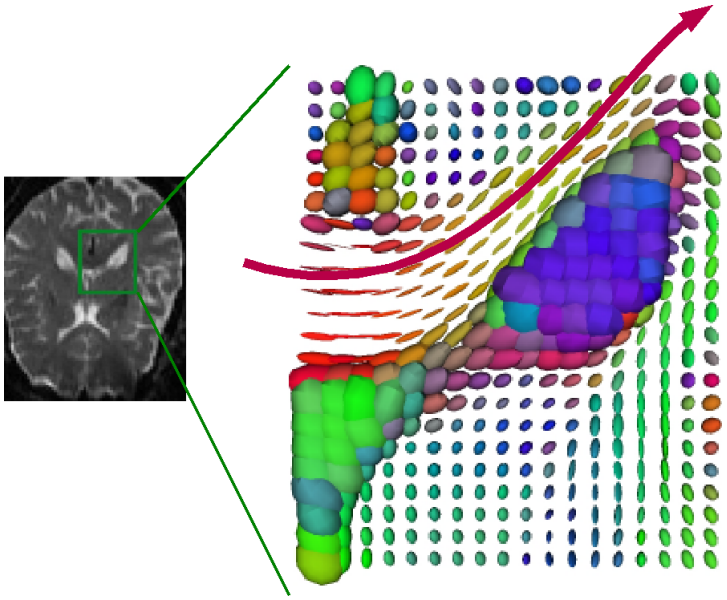


Fig. 1.1: Diffusion MRI provides local models of water diffusion that can be used to detect tracts using tractography algorithms.

1.1 Motivation

Tracts provide information about the connections between the different cortical and subcortical areas, but they can also be used to detect certain anomalies (e.g. tumors) and to compare the brains of groups of patients (both diseased and healthy). Scientists have recently started to use the geometry of fiber tracts to improve registration algorithms applied to brain images [25, 33] (*registering* images means finding a common coordinate system that allows one to consider that a certain point in one image corresponds to another point in a second image).

Tractography methods only recover part of the true brain tracts. However, to provide a sufficient sampling of the connectivity structure across all brain regions tens of thousands of tracts are necessary. Working with so many tracts is inconvenient, both for clinicians who visualize them and for researchers who use them in further processing. Figure 1.2 shows the output of a tractography algorithm.

Fig. 1.2: Brain tracts estimated by a full brain tractography. The color indicates the main direction of the diffusion of water molecules: red indicates movement along the X-axis (either right-to-left or left-to-right), green along the Y-axis (posterior-to-anterior or vice-versa) and blue along the Z-axis (foot-to-head or vice-versa).

For these reasons we aim at grouping tracts with similar properties or characteristics. This is useful for various reasons: for example it is convenient for a functional post-analysis to have tracts grouped according to the areas that they connect, while for registration algorithms grouping them depending on the clusters' shape makes the complex non-linear registration techniques easier to run and helps to interpret the (lack of) correspondences.

Using these clusters helps to compare tracts across individuals. It makes it possible to work with the mean of a single group, or a representative of a group, hence it simplifies subsequent algorithmic steps. Furthermore, by organizing the tracts according to their similarity, it makes other analysis steps robust against aberrant tractography results, noise, and outlier tracts.

1.2 Methods

Clustering of simple structures like n-dimensional points is a vastly explored area, while clustering more complicated objects (for instance objects whose description involves non-linear operations) is much more challenging.

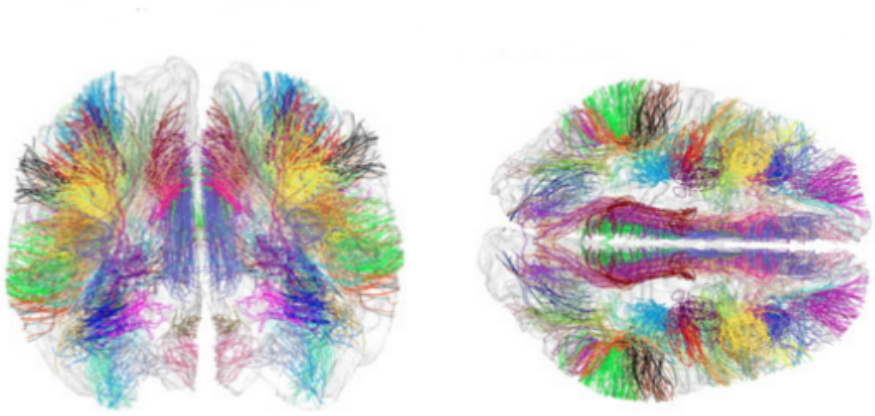
The main problem that we have to face is to define a relevant metric to assess the similarity of tracts. Another one is to pick the right number of clusters, which is a hard problem in general as we are in an unsupervised setting. We will not try to solve this issue in particular in this thesis.

We will approach the problem firstly by using known algorithms which have been proven to give good results like K-means and DBSCAN [5]. Other approaches have been proposed in the literature [2, 10, 12, 13, 17, 29, 30, 31, 32, 33] and yield alternative solutions.

These algorithms take as an input either n-dimensional points or distance matrices, therefore what matters are not the objects themselves but the distances between them. We call *metric* the function that quantifies the distance between tracts. Metric specification entails efficiency/accuracy trade-offs: one can opt for a fast metric (in terms of computational cost), one that helps to maintain the geometric shape of the cluster's tracts, or one that seeks to keep the cluster compact. One can also use atlases to group tracts in well-known bundles. Examples of metrics are the Euclidean and Hausdorff distances, Mean Closest Point, Currents [4], Point Density Model [16], and Gaussian Processes [31].

We will evaluate the implications of choosing a given metric and how it combines with the different algorithms, regarding speed and suitability to the task. The scalability of both the clustering algorithm and the metrics, although not critical on small datasets, will also be taken into account as some tractographies produce from thousands to millions of tracts.

Once the clusters are obtained it is necessary to evaluate qualitatively and quantitatively if the result is satisfactory [15]. Moreover, the algorithm should ideally show consistency regarding the resulting clusters across individuals and with existing atlases (if they were not used during the clustering). Figure 1.3 shows an example of a clustering performed by Guevara et al. [7, 8].



(a) All bundles (Frontal view) *(b)* All bundles (Top view)

Fig. 1.3: Brain tracts clusters by P. Guevara et al. [7, 8]

1.3 Supervised and Unsupervised Learning

Machine learning algorithms are either "supervised" or "unsupervised", depending on how they classify data. In a supervised setting, the classes are a predetermined finite set. In practice, a learning data set is labeled with the classifications. The algorithm's task is to find predictive patterns and build mathematical models to relate those to the known classification. These models are then evaluated depending on their capacity to predict new classifications.

Unsupervised algorithms do not start with a classification. The task of the algorithm consists in developing a data-driven classification. Unsupervised algorithms search for similarities in the data to determine if they can be characterized as belonging to the same group.

In an unsupervised setting, or "cluster analysis", the algorithm does not rely on any information on how elements are grouped and its task is to group them. In K-means, one of the most used algorithms for cluster analysis, the algorithm is given how many clusters it should form. Deciding how many clusters should be formed is a difficult problem in general and we have chosen not to address it in this thesis.

The algorithm initialization is performed more or less arbitrarily and the algorithm must reach a stable configuration after a certain number of iterations. Results vary largely and they can be poor if the algorithm's initialization is poor. This means that the cluster analysis problem is somehow ill-posed.

On the other hand, cluster analysis has great potential in uncovering unexpected details of the data. Most importantly, it is a necessary step to create meaningful visualizations of the data, and thus to assess the quality and usability of tractography outputs. Cluster analysis is thus a necessary tool to study relationships among groups of subjects.

2. METHODS

The main contributions of this project rely on *scikit-learn* [18], an open source machine learning library for Python. The tractographies and the visualizations were done thanks to medInria [27], an open source medical image processing and visualization software: `http://med.inria.fr/`.

In this chapter we will first introduce all the different bricks that we used to build our clustering algorithm: metrics, well-known clustering algorithms and a technique called Multi-Dimensional Scaling. Lastly we will explain how and why all these bricks were put together to create our clustering algorithm described in section 2.5.

2.1 Metrics

A metric is a function that measures distances between samples in a given space. Besides the well-known Euclidean distances many other metrics have been invented to fulfill different needs. In the case of tracts, lines in a 3D space, we will consider four metrics: Hausdorff, Mean Closest Point, Two-Ways Euclidean (TWE) and Point Density Model (PDM). As Garyfallidis et al [6] called TWE Minimum Direct Flip (MDF), we will use TWE or MDF interchangeably.

Hausdorff and Mean Closest Point are widely-used, well-known metrics. We explain them in Appendix A along with Minimum Direct Flip, the metric used by Garyfallidis et al. [6] for their clustering algorithm QuickBundles.

Additionally, we introduce Point Density Model, a metric that has never been used so far to measure distances between brain tracts.

MDF and PDM require us to resample tracts so that they have the same amount of points each. Hausdorff and Mean Closest Point work for tracts with different amounts of points. For simplicity we resample the fibers to k points each.

Given two tracts represented as a sequence of points in a 3-dimensional space $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_m\}$, we consider that an initial resampling has been applied, so that all tracts have the same number k of points. This is a achieved by a linear interpolation. Tracts are now $X = \{x_1, x_2, ..., x_k\}$ and $Y = \{y_1, y_2, ..., y_k\}$ where $x_i, y_j \in \mathbb{R}^3 \ \forall i, j1 \leq i, j \leq k$.

2.1.1 Point Density Model

We propose to use the Point Density Model (PDM) metric which has been used previously for representing sulcal lines [1] but has never been used to measure inter-tract distances.

The oriented version of Point Density Model called *Currents* has been used to represent tracts in a registration scheme in [24].

We propose Point Density Model to better capture the tracts' shape. PDM is sensitive to the tracts' form and position, and is quite robust to missing tract segments. This last property is much desired as tracts are often mis-segmented due to noise and crossing tracts issues.

Definition: The similarity between tracts $X = \{x_1, x_2, ..., x_k\}$ and $Y = \{y_1, y_2, ..., y_k\}$ is given by

$$sim_\sigma(X, Y) = \frac{1}{k^2} \sum_{\substack{i=1 \\ j=1}}^{k} K_\sigma(x_i, y_j)$$

The similarity is symmetric but is not reflexive. K_σ is a Gaussian kernel with scale parameter σ

$$K_\sigma(p, q) = e^{-\frac{||p-q||^2}{2\sigma^2}}$$

finally, the squared PDM distance is

$$PDM_\sigma(X, Y)^2 = sim_\sigma(X, X) + sim_\sigma(Y, Y) - 2sim_\sigma(X, Y)$$

This distance captures misalignment and shape dissimilarities at the resolution σ. Distances much larger or much smaller than σ do not influence the metric. In section 4.2.1 we analyze different values for σ.

The computational complexity of Point Density Model is $\mathcal{O}(k^2)$: quadratic on the number of points per tract.

2.2 Clustering Algorithms

Clustering a data set in an unsupervised setting is a complex and well-known problem, however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum.

In this thesis we will use the well-known K-means algorithm and one of its variants, Mini-Batch K-means [22], which offers faster running times in exchange of small looses in cluster quality. We will also experiment with DBSCAN [5] which is strong where K-means is weak.

Our proposed solution, explained in section 2.5 uses K-means at its core. We will compare our results against QuickBundles, a clustering algorithm specially designed for clustering brain tracts, proposed by Garyfallidis et al [6] in 2012.

An explanation of these algorithms can be found in Appendix B.

Ideally we would simply want to run these algorithms, using the metrics listed in section 2.1. Unfortunately given the time-complexity of this combination and the size of the input sets, this is not viable as a single run can take days.

In order to use both concepts in a reasonable amount of time we combine them by using a Multidimensional Scaling technique.

2.3 Multidimensional Scaling

Given a set of n tracts of k-points each, which can also be seen as a coordinates matrix in $\mathbb{R}^{n \times k}$, and a metric, we can compute all the pairwise distances creating a full distance matrix in $\mathbb{R}^{n \times n}$.

Multidimensional Scaling (MDS) is a technique that computes a coordinates matrix out of a distance matrix so that the the between-object distances are preserved as well as

possible. It is important to note that resulting objects are not the same as the original objects, only the distance between them is approximately preserved.

MDS allows us to select the dimension of the generated objects (it is a parameter in the algorithm) which means that the original objects that generated the distance matrix, which could have been in a k-dimensional space endowed with a non-Euclidean distance, can be reduced to a 2- or 3-dimensional space, which is very useful for plotting. For this reason MDS is a technique widely used in information visualization and data analysis.

Given a set of n tracts, MDS can also output a full coordinates matrix (in $\mathbb{R}^{n\times k}$) out of a partial distance matrix (in $\mathbb{R}^{n\times m}$), in other words it works also if we only have the distances from a subset of m tracts to all the original n tracts.

We now present the mathematical definition of MDS and we explain right after how we use MDS to combine the metrics explained in section 2.1 with K-means.

2.3.1 Mathematical Definition

Adapted from the Wikipedia article on Multidimensional scaling `http://en.wikipedia.org/wiki/Multidimensional_scaling`.

The data to be analyzed is a collection of n tracts on which a distance function is defined. $\delta_{i,j}$ is the distance between the i^{th} and j^{th} tracts. These distances are the entries of the distance matrix:

$$\Delta = \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,n} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,n} \\ \vdots & \vdots & & \vdots \\ \delta_{n,1} & \delta_{n,2} & \cdots & \delta_{n,n} \end{pmatrix}.$$

The goal of MDS is, given Δ, to find n representatives $z_1, \ldots, z_n \in \mathbb{R}^q$ such that

$$\|z_i - z_j\| \approx \delta_{i,j} \forall i, j 1 \leq i, j \leq n$$

where $\|\cdot\|$ is a vector norm. In classical MDS, this norm is the Euclidean distance, but, in a broader sense, it may be a metric or arbitrary distance function.

In other words, MDS attempts to find an embedding from the n tracts into $\mathbb{R}^q$ such that distances are best preserved. The resulting vectors x_i are not unique: with the Euclidean distance for instance, they may be arbitrarily translated, rotated, and reflected, since these transformations do not change the pairwise distances $\|x_i - x_j\|$.

There are various approaches to determining the vectors x_i. Usually, MDS is formulated as an optimization problem, where $(x_1, \ldots, x_n)$ is found as a minimizer of some cost function, for example:

$$\min_{x_1,\ldots,x_n} \sum_{i<j} (\|x_i - x_j\| - \delta_{i,j})^2.$$

A solution may then be found by numerical optimization techniques. For some particularly chosen cost functions, minimizers can be stated analytically in terms of matrix eigendecompositions.

This corresponds to a canonical dimension reduction setting. When starting from a partial distance matrix in $\mathbb{R}^{n\times p}$, only the distances that are in the distance matrix are preserved. Depending on the domain of the problem and the characteristics of the data points, this can be a good approximation.

To sum up, thanks to MDS we can go from a distance matrix to a set of objects whose Euclidean distances closely approximates the distances in the distance matrix. We will combine this with the metrics introduced in section 2.1.

2.4 Use of MDS with tract metrics

The metrics described in section 2.1 have an enormous disadvantage: they are very costly computational-wise.

Hausdorff for instance is quadratic on the number k of points per tract . If we want to compute all the pairwise distances for n tracts it would cost us $\mathcal{O}(n^2k^2)$. Computing a full matrix of pairwise distances for a small set of 20,000 tracts using Hausdorff on a single-core machine takes about 5 and a half hours, if we double the amount of tracts to 40,000 then it takes about 21 hours.

In order to be able to use these metrics that are more suitable for brain tracts we decided to combine them with MDS to maintain the algorithm within a reasonable running time while incorporating the information provided by the metrics.

The algorithm goes as follows: let F be a group of n tracts $f_1, \ldots, f_n$. Each tract f_i has already been resampled so it has a resolution of k (exactly k 3-dimensional points each). We take a random sample S of p tracts from F and then we compute a given distance d between all the tracts in S and F, creating a distance matrix M of shape (n, p). We then apply MDS to this matrix to obtain a new set F' of n representatives (objects belonging to $\mathbb{R}^q$). These new objects will capture the original distance information between them, which is used afterward by the clustering algorithm.

By using a random sample of p elements, where $p \ll n$, we decrease the computational complexity to $\mathcal{O}(npk^2 + np^2)$ which in turn decreases the running time by a significant amount. Note that as a first approach we do not intend to reduce the dimension of the tracts, this is something we would like to explore in future works.

The analysis of the size of p so as to keep the best possible trade-off between speed gain and information loss is treated in section 4.2.2.

2.5 The Algorithm

Below is the pseudo-code of the full algorithm. The input parameters are:

- a set F_{orig} of n tracts, each tract being a list of points in $\mathbb{R}^3$
- *ff*: tracts with length $<$ *ff* will be discarded
- p: percentage of the random sample
- m: the metric that will be used
- c: the desired number of clusters

and the pseudo-code:

```
1: F ← remove tracts shorter than ff millimeters from F_orig
2: resample every tract in F to exactly k points each
3: S ← obtain random sample of F taking p tracts
```

4: M ← compute a distance matrix of tracts from S to F using metric m
5: F' ← apply multi dimensional scaling to M (keep dimension $q = 3k$)
6: C ← process F' with K-means and obtain c clusters
(using Euclidean distance)
7: Return C

The innovative part of this algorithm lies in steps 4, 5 and 6. With MDS we obtain a new set of transformed samples F' which approximately preserves the between-object distances of F, therefore information from the costly tract metric.

We discard tracts that are shorter than *ff* for mainly two reasons: firstly because they are probably outliers of the tractography. Secondly because we do not aim at clustering every single tract in the output of a tractography, but rather to have bundles that make sense and help to understand the underlying structure at a glance. By leaving aside small, short tracts, the quality of the clusters is improved. If eventually we would like to include these fibers, we could simply, at the end of the algorithm, put them in the nearest cluster.

Both this algorithm and QuickBundles are sensitive to the number of clusters c, however QuickBundles' time complexity is $O(nck)$ and our algorithm $O(npk^2 + nck)$, where c is the number of clusters, k the tract resolution and S the sample size. In this algorithm, the creation of the partial distance matrix dominates the time complexity as long as $pk > c$.

3. VALIDATION SCHEME

Evaluating the performance of a clustering algorithm is not as easy as counting the number of errors or the precision of a classification. In particular, any evaluation method shall not take into account the absolute values of the labels of the clusters (i.e. whether tract 1 is in cluster 'A' or 'B') but rather if the clustering defines a separation of the data in a way that members of the same group are more similar among them (according to some metric) than members of other groups.

The problem of evaluating models in unsupervised settings is notoriously difficult. Here we consider a set of standard criteria: the inertia of the clusters, the silhouette coefficient and some measures that require a ground truth: completeness, homogeneity, and rand index and its variants. We will focus specially in the Weighted Normalized Adjusted Rand Index or WNARI, described in appendix C.6. It was introduced by Moberts et al. [15] as a score specifically designed for brain tracts.

3.1 Unsupervised Setting Scores

As we explained before, the most complicated scenario is when we do not have the true bundles to compare our clusters with. Unfortunately this is also the most common scenario. The Inertia and the Silhouette scores make it possible to rate clustering results under an unsupervised setting.

The Inertia (explained in appendix C) has the disadvantage that it decreases along with the number of clusters, so it is not useful when comparing clusterings with different numbers of components. It is also the minimizing target of K-means, the algorithm we used, so it is preferable to rely on an independent score.

3.1.1 Silhouette Coefficient

The Silhouette Coefficient [21] measures how close a tract is to its own cluster in comparison to the rest of the clusters. In other words, it measures whether there is another cluster that might represent the tract better or as well.

The silhouette score for a given tract is defined as:

$$silhouette = \frac{b - a}{max(a, b)}$$

where a is the mean intra-cluster distance (the mean distance between a sample and all other points in the same class) and b the mean nearest-cluster distance (the mean distance between a sample and all other points in the next nearest cluster).

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

The main drawback of this score is that an exact computation of the silhouette requires the calculation of a full distance matrix (in order to find the nearest cluster) which renders it prohibitive for our use. Using multi-dimensional scaling (MDS) method once again we can reduce the computation time by approximating the distances. We take a random sample S

of the full tract set F and compute a partial distance matrix D of shape $(\#S \times \#F)$, where $\#S$ and $\#F$ denote the cardinal of S and F respectively, and we use that to approximate the silhouette score. With 10% of the tracts, the relative error between the silhouette scores computed using the full distance matrix and the approximated one was smaller than 10^{-2} on various random tract sets (see section 4.2.2).

3.2 Supervised Setting Scores

When we know how the tracts should be grouped (the true classes/bundles the tracts belong to) it is easy to evaluate a clustering. Many scores has been created to use in a supervised setting.

When running our algorithm on a manually labeled data set, in which we know the true classes, we use this type of scores to rate the clusterings generated by out algorithm. We use a variation of the Adjusted Rand Index called Weighted Normalized ARI or WNARI, created by Moberts et al [15] specific for clusterings of brain tracts. All the definitions and explanations can be found in appendix C.

4. EXPERIMENTS AND RESULTS

4.1 Chapter Outline

In this chapter we present the experiments that we performed and the conclusions that we drafted out of them.

1. Point Density Model metric has a free parameter σ that represents the resolution. We show how we fix it in section 4.2.1.

2. Our algorithm, presented in section 2.5, has also a free parameter p: the size of the random sample. It has an unknown impact on the quality of the output clusters as well as the running time of the algorithm. We explore its impact in section 4.2.2 and we fix it for all of our experiments.

3. Then, to assess the soundness of the proposed approach, we run the candidate algorithms on a reduced dataset, one where the solution is known: a selection of fibers whose original clustering is defined beforehand by us, hence we call it a "Manually Labeled Dataset". As with this data set the expected clustering, the solution, is known, we can use the scores for supervised settings that we presented in section 3.2. We can also check whether the results provided by these scores are consistent with those provided by the Silhouette Coefficient, the only score that we can use when running the algorithm of a full brain tractography in an unsupervised setting.

4. DBSCAN: we run DBSCAN first on the manually labeled dataset and then on full tractographies. We present the results and draft conclusions in section 4.4.

5. Mini Batch K-means is known to be significantly faster than K-means while generating clusters that are slightly worse. We run experiments to determine whether this is also the case in our domain: clustering of brain tracts. Conclusions are presented in section 4.5.

6. Finally we run the algorithm we proposed in section 2.5. First on the manually labeled dataset introduced in section 4.3 and then on full tractographies with thousands of tracts.

4.2 Parameters Setting

Our algorithm, described in section 2.5, has two free parameters: p, the size of the random sample used by MDS, and σ, the resolution of Point Density Model metric. We study their impact and decide how to set them in this section.

4.2.1 Sigma: PDM Kernel Resolution

We performed a parameter selection test over 10 subjects to analyze the impact of the kernel size for K-means with Point Density Model. We varied σ from 10 to 60 mm and the number of clusters from 500 to 1200. We noticed that after $\sigma = 42$ mm the silhouette coefficient stopped increasing in a significant amount. We therefore fixed $\sigma = 42$ mm.

4.2.2 Size of the random sample p

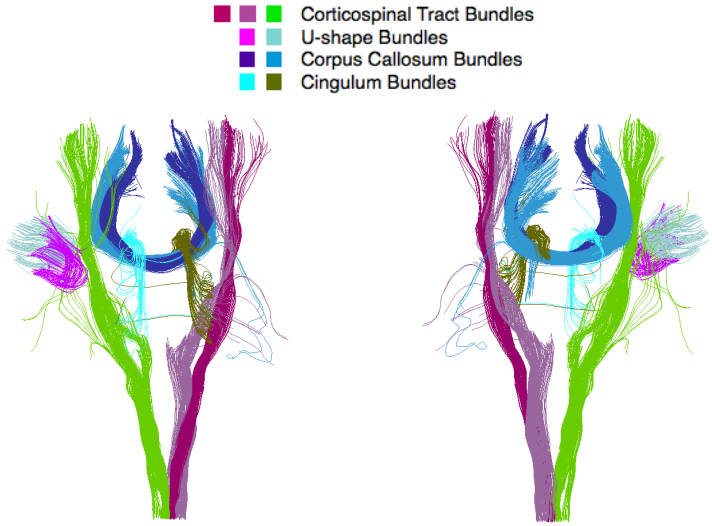
As explained in section 3.1.1 we will approximate the silhouette score by using a partial distance matrix instead of the full one. Therefore we wanted to see how the different choices for the sample size (the same sample size p used by the MDS step) impacted the silhouette score. In order to measure this, we ran the full algorithm on 10 subjects using all the tracts for the sample size, then 30% and 20% of the tracts. The largest difference in silhouette scores was 10^{-2} which is small enough for us to fix $p = 20\%$.

4.3 Manually Labeled Dataset

In order to first test the proposed algorithm on a known data set, (one where a reference definition of the bundles is available), we selected 9 known brain bundles containing 1163 tracts previously identified from the corpus callosum, corticospinal tract, u-shape, and fronto-occipital. This group of bundles was selected for the following reasons:

- It has similar tracts (in shape and trajectory) that belong to different hemispheres (they are "mirrored") and therefore should be clustered separately.
- It contains some well-known tract bundles of the brain, like the corticospinal tract and the corpus callosum.
- The two small u-shaped bundles are so close and similar that some algorithms may cluster them together, but they should actually be in different clusters.
- Tracts in the corticospinal tract can be hard to cluster as they start quite close to each other but then they spread wide and large.

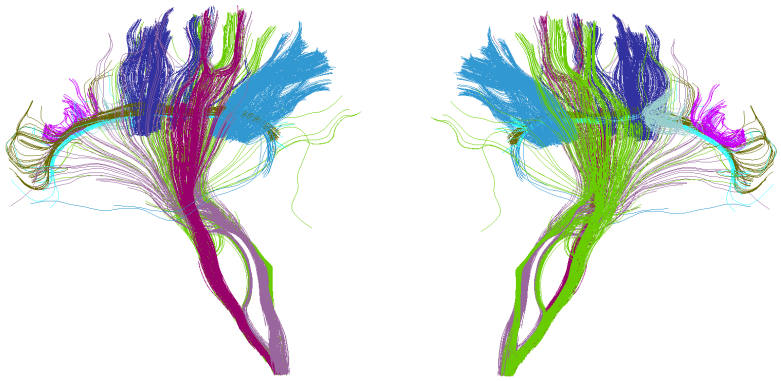
Figure 4.1 shows the front and back views of the manually labeled data set. The lateral views can be seen in figure 4.2 and the front and bottom views in figure 4.3.



(a) Anterior or Rostral (Front Surface)

(b) Posterior or Caudal (Back Surface)

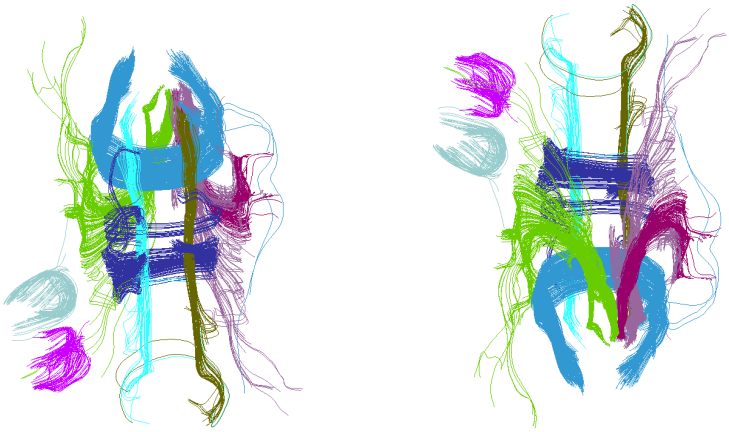
Fig. 4.1: Front and back views of the selected fiber tracts.



(a) Lateral Left

(b) Lateral Right

Fig. 4.2: Lateral views of the selected fiber tracts.



(a) Superior or Dorsal (Top Surface)

(b) Inferior or Ventral (Bottom Surface)

Fig. 4.3: Top and bottom views of the selected fiber tracts.

4.4 DBSCAN experiments and results

We first ran DBSCAN on the manually labeled dataset, where we knew beforehand the number of clusters that we would ideally recover. The algorithm always recovered the clusters, presenting a WNARI score of an average of 0.8 and reaching 0.9 when varying the metrics and their parameters. The algorithm also identified outliers, setting aside some spurious tracts. Out of the ∼1200 tracts in the manually labeled data set, about 30 to 50 tracts were considered as outliers when scores were at their best. We therefore proceeded to run DBSCAN with each of the four metrics on 10 subjects.

The results were very similar across all the subjects and for every metric, while they differed in some specific values they did follow the same trend: DBSCAN leaves many fibers out as outliers and generates large clusters.

Figure 4.4 shows the results for one subject chosen at random. We can see the total number of clusters obtained with respect to DBSCAN's `eps` parameter (the dotted blue line); as explained in appendix B.3, `eps` is the neighborhood-defining distance. The maximum number of clusters DBSCAN could obtain was almost 350 which is below what is expected. In order to obtain a useful separation, researchers usually pick 500 clusters as a minimum (and up to 1000). With fewer, some tracts that should be separated end up being grouped together.

Moreover, we also need to consider the amount of tracts that were considered as outliers and therefore not clustered. The full red line in the same figure 4.4 shows the percentage of fibers that were clustered: when 350 clusters were obtained, more than 40% of the fibers were left out as outliers.

We conclude that DBSCAN is not very well suited for the task of clustering neural pathways. Therefore we will not use it.

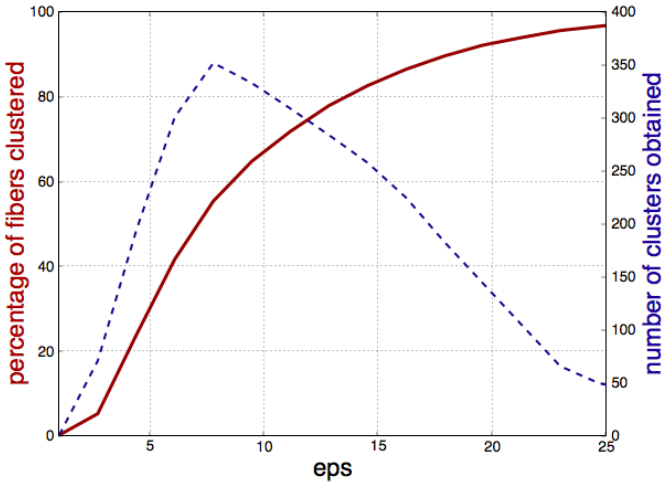


Fig. 4.4: DBSCAN: total number of clusters obtained (dotted blue line) and percentage of tracts that were clustered (full red line)

4.5 Mini Batch Kmeans experiments and results

We wanted to better understand the differences between Kmeans and its variant Mini Batch Kmeans. The latter is known to be significantly faster while obtaining clusters slightly worse than those obtained with vanilla Kmeans [22].

We tested both algorithms with all the metrics on 5 random subjects, varying the number of clusters using a logarithmic scale. In order to measure the running time with respect to the number of fibers to cluster, we took random subsets of incrementing sizes, out of the full set of tracts. As both algorithms produce slightly different clusters depending on the random initialization, we ran each experiment 5 times. We then compared the running times and the silhouette scores.

Figure 4.5 shows the differences in running time. As explained in section 2.5 most of the processing time is spent generating the new tract set F' (the partial distance matrix and the MDS step) that the clustering algorithms consume, therefore we only plot the running time of the clustering algorithms (which is independent from the metric used). The curves were all similar for all the different number of clusters we tried, we picked 560 clusters for the plot. We can see that indeed Mini Batch Kmeans is significantly faster.

We then compared the silhouette score: when creating less than 400 clusters Kmeans always obtained better scores, up to 0.04 points better. When the number of clusters increased, up to 600 clusters, the difference in silhouette increased and Kmeans always obtained between 0.08 and 0.11 points more than Mini Batch Kmeans. A paired t-test was conducted, yielding a t-value of 51.22 and a p-value of 1.94e-13 therefore the difference is significant.

We conclude that for our objectives, we will use Kmeans so as to maximize the quality of our clusters. As time performance is not key in our study we can afford waiting about 6/7 minutes for Kmeans to complete. Moreover let us keep in mind that in our algorithm the generation of the distance matrix takes most of the time, and not the clustering algorithm. Mini Batch Kmeans could be reconsidered for tractographies with millions of tracts or for implementations that are time sensitive.

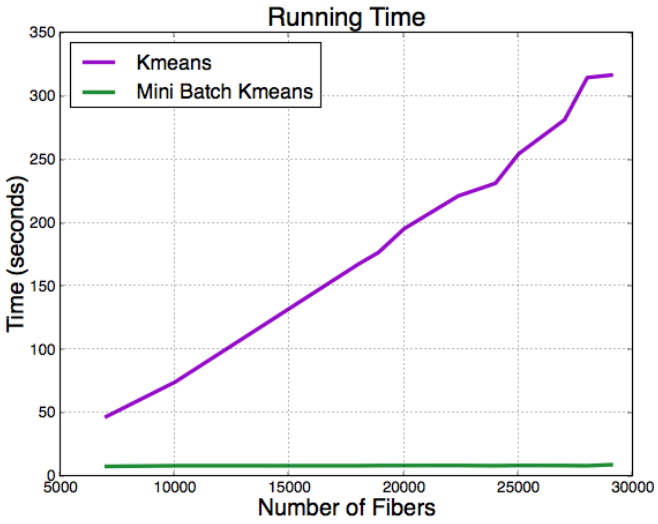


Fig. 4.5: Differences in running time between K-means and Mini Batch K-means for 560 clusters. This is an average across five datasets, based on five initializations.

4.6 Our algorithm: experiments and results

We ran the algorithm described in section 2.5, with the 4 different metrics, first on the manually labeled dataset described in section 4.3 and then on full tractographies with thousands of tracts. We also compared the results to QuickBundles.

4.6.1 Experiments on Manually Labeled Data

We tested first the algorithms on the manually labeled data described in section 4.3. As we knew the true bundles beforehand we were able to run the validation criteria that need a reference or *ground truth* as well as the Silhouette Coefficient. We varied the number of clusters for K-means and the threshold parameter for QuickBundles (represented with dots).

Each experiment was run 10 times while randomly removing 1/8 of the tracts, to sample variable configurations.

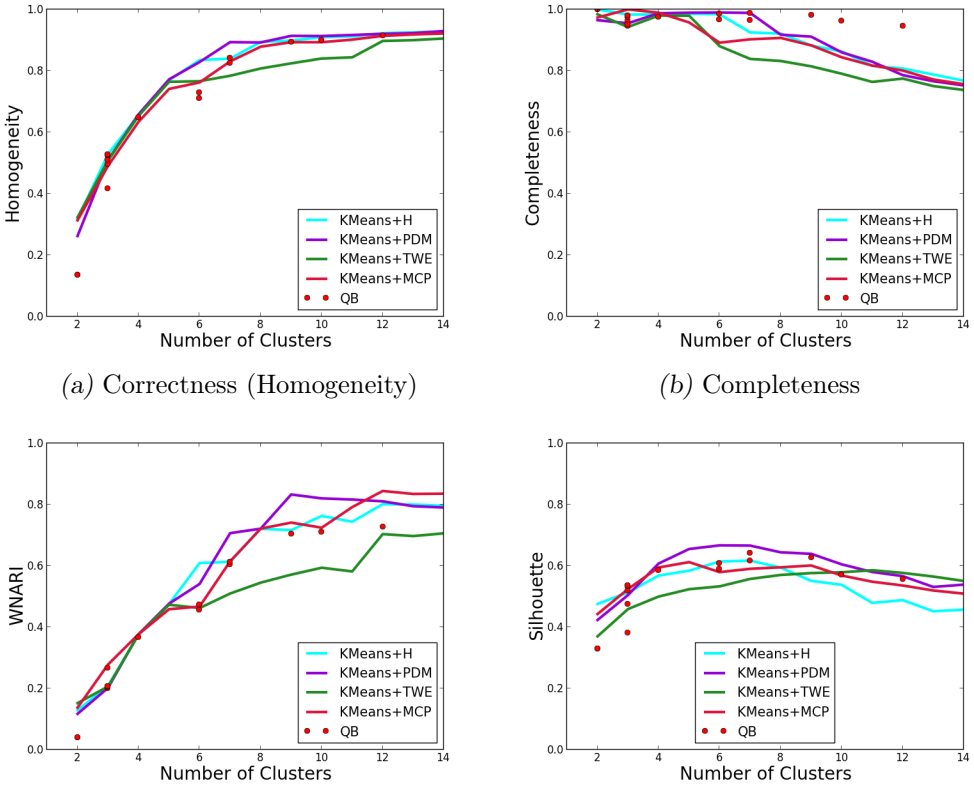
Analysis of the Scores

We used the Weighted Normalized Adjusted Rand Index, or WNARI, which according to the research conducted by Moberts et al [15] is the supervised score that matches best the opinion of clinicians regarding what makes a good clustering.

The WNARI evaluates the correctness and completeness of the clusters. Correctness checks that each cluster contains only members of a single class. Tracts belonging to different anatomical structures should not be clustered together. Correctness penalizes the clustering schemes in which samples from different classes are clustered together. Completeness verifies that all members of a given class are assigned to the same cluster. Tracts belonging to the same anatomical structure should be clustered together. It penalizes the clustering schemes in which samples from the same class are clustered separately (see appendix C.5).

WNARI gives more importance to correctness than to completeness: it is indeed preferable to have two clusters that actually belong to the same bundle, than to have to separate bundles grouped in the same cluster. The idea behind this is that, when considering the final result, it is easier to merge two clusters than to sub-divide a given cluster.

WNARI, Correctness and Completeness are shown in figure 4.6 along with the Silhouette Coefficient.



(a) Correctness (Homogeneity)

(b) Completeness

(c) WNARI

(d) Silhouette Coefficient

Fig. 4.6: Scores of the manually labeled data

Figure 4.6c shows that Kmeans+PDM obtains the best scores, with QuickBundles performing as well as Kmeans + Hausdorff and MCP. By breaking WNARI into Correctness and Completeness we can see that QuickBundles behaves extremely well regarding completeness but not so well on homogeneity. On the other hand, KMeans+PDM obtained high homogeneity but lower completeness, indicating that clusters contain tracts from the same structure but that they are not complete, which means that some structures can be split. This makes us wonder about the distribution of the tracts among the many clusters: it seems that QuickBundles is yielding some large clusters. We will analyze this later in section 4.6.2.

Lastly, the Silhouette scores puts Kmeans+PDM as the winner below 11 clusters (the simulated ground truth comprises 9 clusters).

It can be observed that Kmeans with Point Density Model metric obtains good results in all cases. This was expected, as we believe that this metric better captures the distance between tracts, taking into consideration misalignments and shape dissimilarities. The

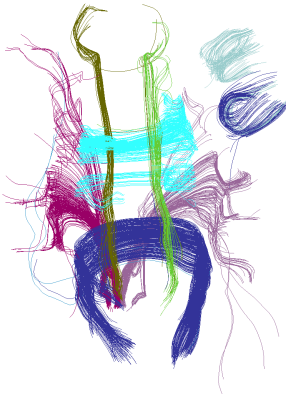
results confirm our prior hypothesis.

Analysis of the Clusters

Figure 4.7 shows the clusters recovered by Kmeans when setting the number of clusters to 9 and by QuickBundles when setting the threshold parameter to 29.2 mm (produced 9 clusters). The 2 corpus callosum tract clusters (the big C-shaped bundles in the middle, refer to figure 4.1a for a legend associating the color code with the bundles) were well identified by every algorithm except for TWE that also mixed some other tracts. The fronto-occipital tracts (the vertical, elongated ones) from both hemispheres were classified as a unique cluster for Hausdorff and Mean Closest Point. QuickBundles mixed together 2 of the 3 bundles of the corticospinal tract (the two that were in the same hemisphere) and Point Density Model succeeded to obtain all the original clusters.

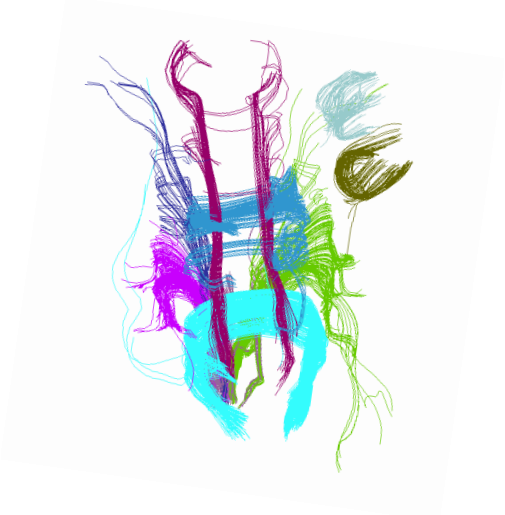
Hausdorff

Two-ways Euclidean



QuickBundles

Point Density Model



Mean Closest Point

Fig. 4.7: Manually Labeled Data - Clusters obtained with different clustering methods.

4.6.2 Experiments on Full Tractographies

After running the algorithms on the Manually Labeled Data we ran them all on full tractographies, i.e. models of brain tracts of about 35,000 to 50,000 tracts each. We exhaustively tested over ten subjects Kmeans with Point Density Model, Hausdorff, Two-ways Euclidean and Mean Closest Point while varying the number of clusters from 120 to 3200. Additionally we compared their output to QuickBundles (QB) [6]. However, as we explained in B.4 in QB the resulting number of clusters is guided by a threshold value. Therefore we ran QB over one subject varying the threshold from 5 to 40 mm, and selected threshold values based on the number of clusters obtained to run them over the 10 subjects.

In an unsupervised setting, we can only use the unsupervised scores, such as inertia and the silhouette coefficient. We focus on the latter as it is bounded between -1 and 1 (therefore is easier to compare) and it less affected by the number of clusters. Figure 4.8 shows the results obtained for each of the aforementioned criteria and algorithms.

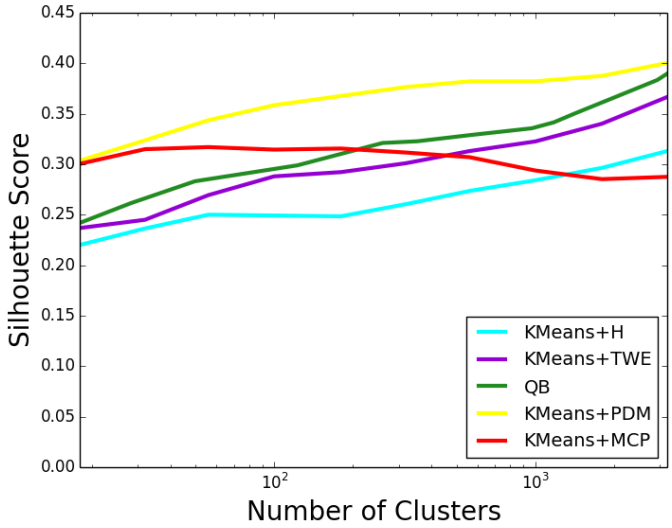


Fig. 4.8: **Silhouette score on full tractographies**: Comparison of KMeans with PDM, TWE, MCP, and Hausdorff metrics, and QuickBundles. Each curve shows the average score of the ten subjects. KMeans+PDM is used with two sizes of the learning set in MDS step.

We can see that KMeans+PDM is always 0.05 points ahead from QB, which is followed by TWE and a bit farther there is Hausdorff. Nonetheless when going to large numbers of clusters (over 3000), curves between QB and KMeans+PDM seem to converge in terms of cluster quality.

The behavior of MCP is unexpected and requires some attention. A simple explanation is the ill-posed optimization: MCP does not enforce the definition of a distance and K-means may perform poorly in that case. In other words, K-means is not a reliable cluster estimator: it tries to solve a non-convex problem with a greedy approach. As a consequence, it almost surely does not find the optimal solution, and this behavior becomes even worse when we increase the number of clusters. Moreover, it relies on the distance between the tracts and MCP is not a proper distance, (it does not enforce the triangle inequality) which yields a bad behavior overall (see appendix A.2).

T-test : For 560 clusters, the average Silhouette Score of clusters obtained with K-means+PDM is 0.38 and the standard deviation 0.008, for QB the mean is 0.33 and the standard deviation 0.009. A t-test is a statistic that checks if two means are reliably different from each other (and not simply due to chance). A paired-samples t-test was used to check whether our experiments are statistically significant: t(9) = 11.77, p = 9.07e-07, the difference is thus highly significant.

Note that a given number of clusters can correspond to strikingly different structures in the data, depending on the algorithm and metric: In Figure 4.9 we show the result of the full brain tract clustering for all algorithms on an arbitrarily chosen subject. The number of clusters was set to 560 in all of them. Resulting clusters on H, MCP, TWE and QB seem to be wider and more heterogeneous than PDM, showing that PDM metric can indeed better capture shape of tracts. Homogeneity of clusters in comparison to H, MCP, TWE and QB can clearly be seen in the corpus callosum and the corticospinal tract.

In figure 4.10 we can see the histograms of the clusters sizes. QuickBundles has the largest amount of small clusters, that likely correspond to outlier tracts and, as the results of the Manually Labeled data set suggested, it also formed very large clusters. KMeans+PDM also seems to generate a few small clusters, in contrary to KMeans+H, TWE or MCP that have no small clusters, meaning that spurious tracts are included in clusters, and not rejected as outliers.

Both QB and Kmeans+PDM running times are sensitive to the number of clusters, however QuickBundles' time complexity is $O(nck)$ and KMeans+PDM $O(npk^2 + nck)$, where n is the number of tracts, c is the number of clusters, k the tract resolution and p the sample size. In K-means+PDM, the creation of the partial distance matrix dominates the time complexity as long as $pk > c$.

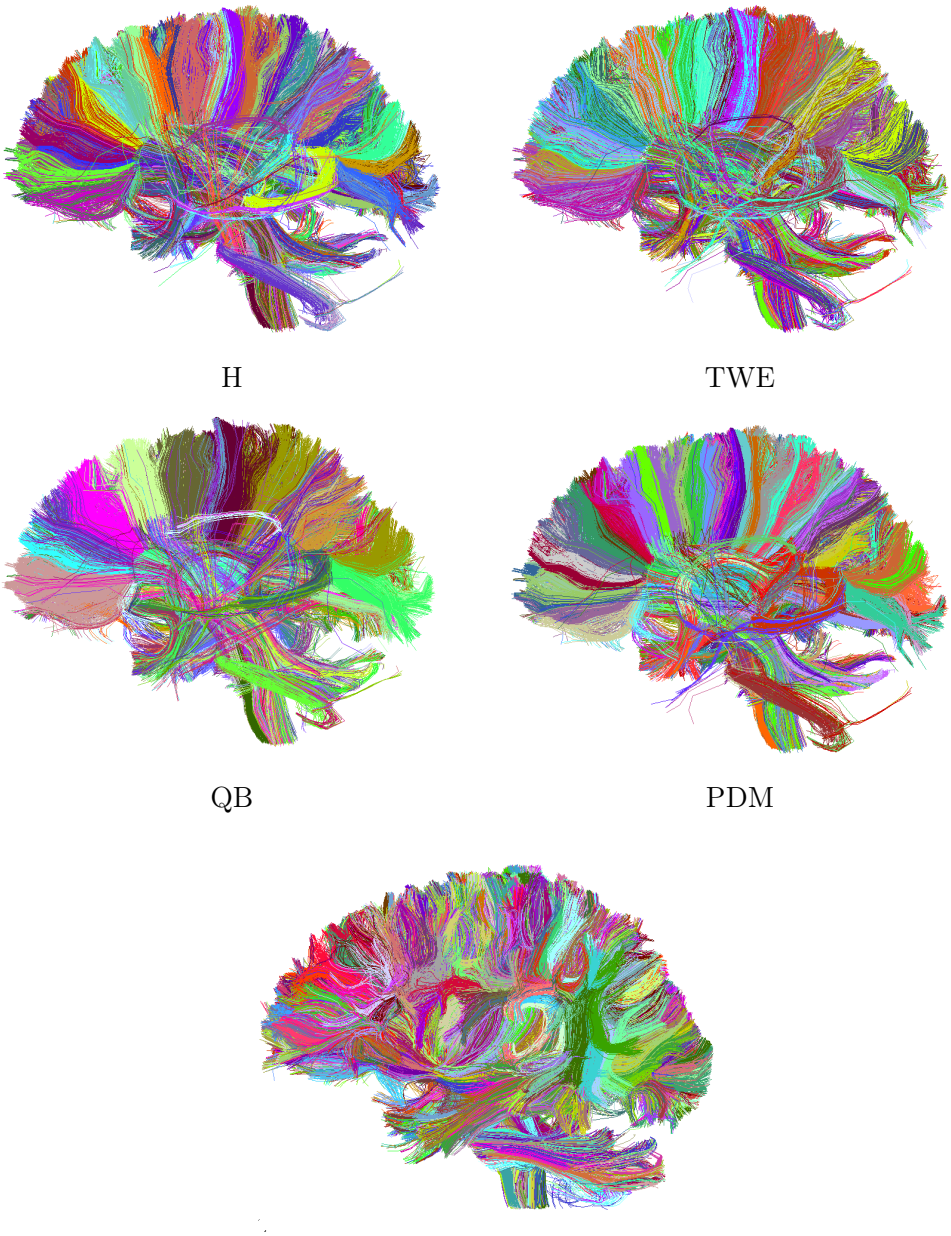


Fig. 4.9: **560 clusters of fiber tracts**: Qualitative algorithm comparison for the resulting tract clusters on an arbitrary chosen subject.

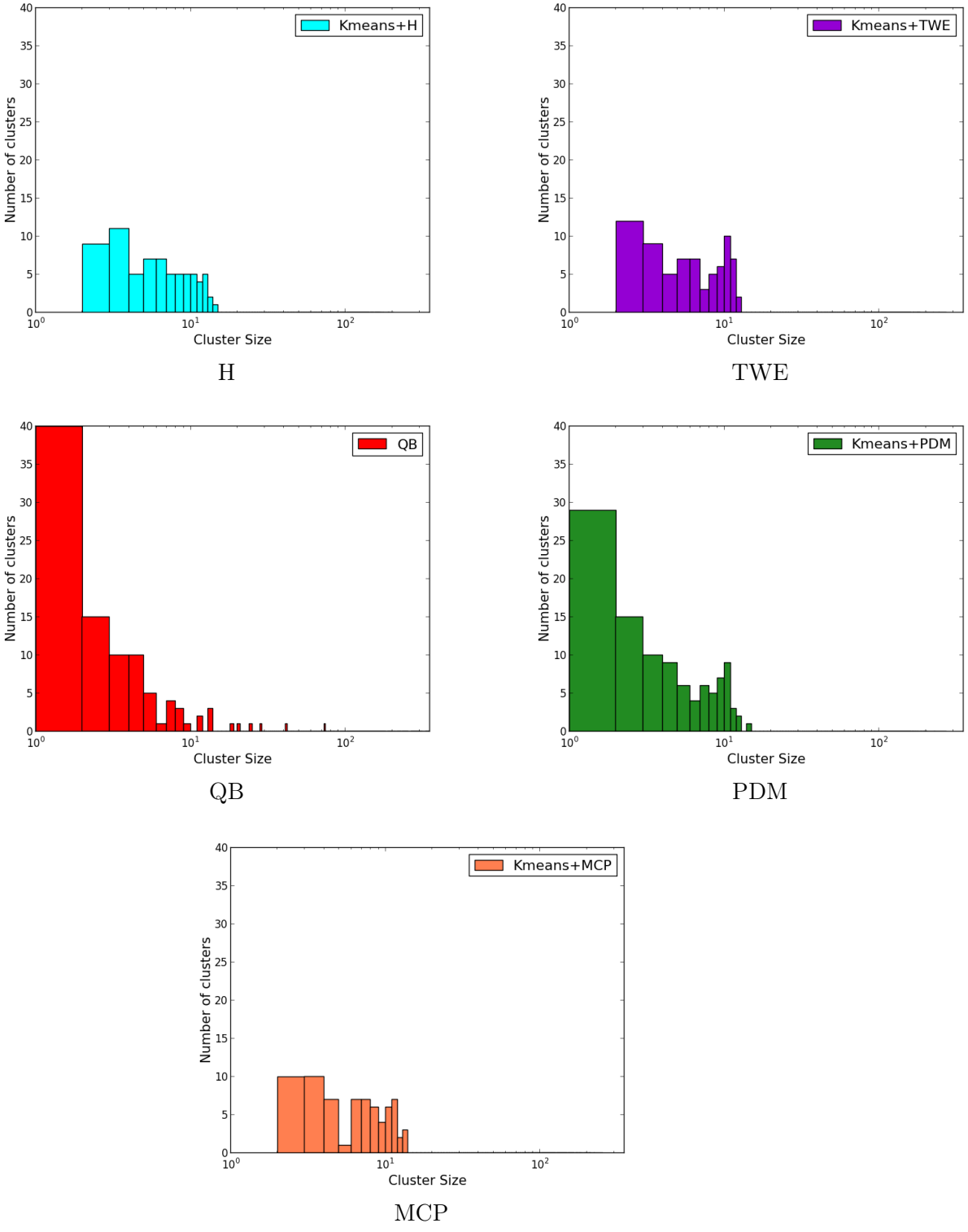


Fig. 4.10: Histogram of the cluster sizes for the different algorithms.

5. CONCLUSIONS

We have presented an analysis and comparison of some of the techniques most commonly used on tract clustering. Clustering is a tool of choice to simplify the complicated structure of brain tracts, assuming that we obtain homogeneous clusters that can easily be represented by the cluster centroid.

We performed a new contribution: we show how to combine the well-known K-means clustering algorithm with various metrics while keeping an efficient procedure. By introducing multi dimensional scaling techniques we could include and compare the available metrics on the literature for measuring distances between tracts, incorporating PDM which has been used recently to represent geometric structures in the brain, but never for tract clustering.

We show different behaviors of the methods depending on the number of clusters: while QB is good at isolating outlier tracts in small clusters, it requires a large number of clusters to represent effectively the whole set of tracts. Kmeans+PDM has a better compression power, but is less robust against outlier tracts. It outperforms other metrics.

We believe that this method along with a posterior tract registration [24] can be a consistent tool for white matter group analysis. In the future, it could be applied for the analysis of white matter in neurological settings [3].

6. FUTURE WORK

During the development of this thesis we could not explore all the possible options and develop every possible improvement of the basic clustering set-up. Some particularly relevant questions to address further are:

- **Test more metrics**, like "Currents" [4], the oriented version of Point Density Model. This requires to somehow choose an orientation for each of the tracts, which is not an easy task. We also wanted to try using Gaussian Processes [31] which seemed to have yielded good results to Wasserman et al. [31].
- Experiment with **Brain Atlases** to improve the clustering algorithms, a promising option, explored by Pamela Guevara et al [7].
- Measure the **performance of Mini Batch Kmeans** with huge tractographies and millions of tracts, and compare it with the QuickBundles method.
- Explore **reducing the dimension of the set obtained by MDS**. As this reduces computation time, we wonder what impact it has on the clusters quality.
- Explore the impact of **further reducing the sample size**. Does the cluster quality decrease significantly? How much is the speed gain?

In almost every case, the exploration of accuracy/efficiency tradeoffs is at the core of the development of practical solutions to fiber clustering.

7. PUBLICATION

The paper version of this research *A Comparison of Metrics and Algorithms for Fiber Clustering* [23] has been published at the 3^{rd} International Workshop on Pattern Recognition in NeuroImaging (PRNI) in June 2013, held at the University of Pennsylvania, Philadelphia, US (`http://www.rad.upenn.edu/sbia/PRNI2013/`).

Appendices

Appendix A

METRICS

Brain tracts are modeled as a list of points in $\mathbb{R}^3$. Some of the metrics in this appendix require the tracts to have the same amount of points, so we always resample the tracts and reduce their number of points to k (this is achieved by a linear interpolation). Tracts are now $X = \{x_1, x_2, ..., x_k\}$ and $Y = \{y_1, y_2, ..., y_k\}$ where $x_i, y_j \in \mathbb{R}^3 \ \forall 1 \leq i, j \leq k$.

A.1 Hausdorff

From its Wikipedia article: `http://en.wikipedia.org/wiki/Hausdorff_distance`.

The Hausdorff distance measures how far two sets of points are from each other. Informally, two sets are close in the Hausdorff distance if every point of either set is close to some point of the other set. The Hausdorff distance is the longest distance you can be forced to travel by an adversary who chooses a point in one of the two sets, from where you then must travel to the other set. In other words, it is the greatest of all the distances from a point in one set to the closest point in the other set. Figure A.1 shows the components of the calculation of the Hausdorff distance for 2 lines. The definition is:

$$hausdorff\,(X, Y) = \max(h(X, Y), h(Y, X))$$

$$h(X, Y) = \max_{i=1..k}(\min_{j=1..k}(||x_i - y_j||_2)))$$

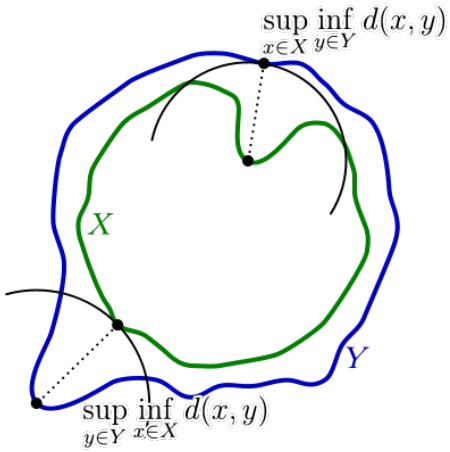


Fig. A.1: Components of the calculation of the Hausdorff distance between the green line X and the blue line Y.

The Hausdorff metric handles tracts as clouds of points, therefore it does not take advantage of the tracts' direction. Its complexity is $\mathcal{O}(k^2)$, quadratic on the number of points per tract.

A.2 Mean Closest Point (MCP)

Another common practice when measuring distances in the 3D space is taking a combination of the distances of the closest points of the different subsets.

$$mcp = \frac{\sum_{i=1}^{k} \min_{j=1..k} ||x_i - y_j||_2}{k} + \frac{\sum_{j=1}^{k} \min_{i=1..k} ||x_i - y_j||_2}{k}$$

MCP complexity is $\mathcal{O}(k^2)$, quadratic on the number of points per tract. Note that it is not a proper metric, as it does not enforce the triangle inequality. This can lead to poor behavior in practical applications.

A.3 Minimum Direct Flip / Two-ways Euclidean Distance

Adapted from QuickBundles' paper "QuickBundles, a method for tractography simplification" by Garyfallidis et al [6].

The Euclidean distance on vectors of stacked coordinates is a metric used widely for clustering, yet it can yield very different results depending on the chosen orientation for the tract.

Having a consistent orientation for all tracts across the brain is an extremely difficult task without previously segmenting the brain. To overcome this issue, TWE/MDF evaluates the distance in both directions. Therefore the TWE/MDF distance is simply averaging the Euclidean distances of each pair of points of X and Y, repeat this but reversing the points of Y (thus changing the direction of the tract) and keep the minimum of both values.

Two-Ways Euclidean (TWE), or Minimum Direct Flip (MDF) as Garyfallidis et al [6] call it, is the distance used by their algorithm QuickBundles (see B.4). To define MDF, they first define $d_{direct}(s, s')$ and $d_{flipped}(s, s')$, as shown in figure A.2, that are both Euclidean distances.

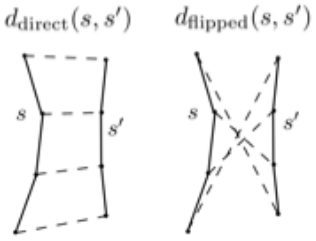


Fig. A.2: Euclidean distances used to calculate MDF.

$$d_{direct}(X, Y) = \frac{1}{k} \sum_{i=1}^{k} ||x_i - y_i||_2$$

$$MDF(X, Y) = \min(d_{direct}(X, Y), d_{direct}(X, flipped(Y))) = \min(d_{direct}(X, Y), d_{flipped}(X, Y))$$

The main advantage of this distance is that it is very fast to compute (between two tracts of k points it requires the calculation of just $2k$ inter-point distances), it takes account of tract direction issues through consideration of both direct and flipped tracts. Moreover,

its behavior is easy to understand, from the simplest case of parallel equi-length tracts to the most complicated with very divergent tracts. Another advantage of the MDF distance function is that it separates short tracts from long tracts; a tract s that is a portion of a tract f will be relatively poorly matched on MDF to f. On the other hand it does not take into account the 3D-shape of the tracts which is one of their major characteristics.

Appendix B

ALGORITHMS

B.1 K-means

From Matteo Matteucci's clustering tutorial: `http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html` and K-means Wikipedia page: `http://en.wikipedia.org/wiki/K-means_clustering`.

K-means clustering is a well-known method of cluster analysis which aims to partition a set N of n observations into c clusters in which each observation belongs to the cluster with the nearest mean, where c is fixed a priori. This results in a partitioning of the data space into Voronoi cells. In our particular case each "observation" is a brain tract.

Probably K-means is the most famous unsupervised learning algorithm that solves the well-known clustering problem. The term "K-means" was coined by MacQueen in 1967 [11] however the ideas goes back as far as 1956 with Hugo Steinhaus [26]. The algorithm was proposed also by S. Loyd and E. Forgy and this is why sometimes it is referred as the Loyd-Forgy method in the computer science community. Hereafter we will refer to it simply as K-means.

The procedure follows a clear way to classify a given data set through c clusters fixed a priori. The main idea is to define c centroids, one for each cluster. These centroids should be placed in a clever way because different placements cause different results. The next step is to take each point belonging to a given data set and associate it with the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to recompute c new centroids as barycenters of the clusters resulting from the previous step. After we have these c new centroids, a new assignment of each sample to the nearest centroid is done. This results in an update/assignment loop. As a result of this loop we may notice that the c centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

The mathematical definition is: given a set of observations $(x_1, \ldots, x_n)$, where each observation is a d-dimensional real vector, K-means clustering aims to partition N into c sets $(c \leq n)$ $C = \{C_1, \ldots, C_c\}$ so as to minimize their *inertia* (the within-cluster sum of squares, see appendix C.1).

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{c} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mu_i\|^2$$

where μ_i is the cluster centroid (the mean of points in C_i).

B.1.1 Algorithm

The algorithm has three important phases: Initialization, Assignment and Update:

```
1: State Initialization: choose c centroids by drawing randomly c observations
                         among n without replacement
do
    2: State Assignment: Assign each observation to its closest cluster
    3: State Update: Update the centroids of each cluster
while (Centroids moved)
```

Initialization: Place c points $m_1^{(1)}, \ldots, m_c^{(1)}$ into the space represented by the objects that are being clustered. These points represent the initial group centroids.

Assignment Step: Assign each observation to the cluster whose mean is closest to it (i.e. partition the observations according to the Voronoi diagram generated by the means). t is the current iteration.

$$C_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\| \leq \left\| x_p - m_j^{(t)} \right\| \; \forall \; 1 \leq j \leq c \right\}$$

where each x_p is assigned to exactly one $C^{(t)}$, even if it could be is assigned to two or more of them.

Update step: Calculate the new means to be the centroids of the observations in the new clusters.

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{\mathbf{x}_j \in C_i^{(t)}} \mathbf{x}_j$$

When the centroids no longer move the algorithm has converged.

Although it can be proved that the procedure will always terminate, the K-means algorithm does not necessarily find the optimal configuration, corresponding to the global objective function minimum. The algorithm is actually sensitive to the initial randomly selected cluster centers. K-means is usually run multiple times to reduce this effect.

The complexity of the algorithm is $O(n \times c \times I \times d)$, where n is the number of samples, c is the number of clusters we want, I is the number of iterations and d is the number of features in a particular sample (in our case, $d = 3k$, where k is the number of points in the brain tract).

K-means is a simple algorithm that has been adapted to many problem domains. In this thesis it is also adapted to our specific needs.

B.2 Mini Batch K-means

From Scikit-learn's Mini Batch K-means page `http://scikit-learn.org/stable/modules/clustering.html` and Siddharth Agrawal's machine learning tutorial `http://algorithmicthoughts.wordpress.com`.

Mini Batch K-means [22] is a variant of the K-means algorithm that uses mini-batches to drastically reduce the computation time, while still attempting to optimize the same objective function. Mini-batches are subsets of the input data, randomly sampled in each training iteration. These mini-batches drastically reduce the amount of computation required to converge to a local solution. In contrast to other algorithms that reduce the

convergence time of K-means, and depending on the domain, mini-batch K-means produces results that are generally only slightly worse than the standard K-means algorithm.

B.2.1 Algorithm

The algorithm iterates between two major steps, similar to vanilla K-means. In the first step, b samples are drawn randomly from the dataset, to form a mini-batch. These are then assigned to the nearest centroid. In the second step, the centroids are updated. In contrast to K-means, this is done on a per-sample basis. For each sample in the mini-batch, the assigned centroid is updated by taking the streaming average of the sample and all previous samples assigned to that centroid. This has the effect of decreasing the rate of change for a centroid over time. These steps are performed until convergence or a predetermined number of iterations is reached.

```
Given: mini-batch size b, iterations I, data set N
Initialize each ct ∈ C with an x picked randomly from N
v ← 0
for i = 1 to I do
    M ← b examples picked randomly from N
    for x in M do
        d[x] ← f(C, x)              // Cache the center nearest to x
    end for
    for x in M do
        ct ← d[x]                   // Get cached center for this x
        v[ct] ← v[ct] + 1           // Update per-center counts
        lr ← 1 / v[ct]              // Get per-center learning rate
        ct ← (1 - lr)ct + lr * x    // Take gradient step
    end for
end for
```

Mini Batch K-means converges faster than K-means, but the quality of the results is reduced. In practice this difference in quality could be small enough to be overweight by the speed gain.

Altogether, Mini Batch K-means presents all the advantages and disadvantages explained above for K-means, thus it is used only to benefit from the increased computation speed.

B.3 DBSCAN

From DBSCAN's Wikipedia article: `http://en.wikipedia.org/wiki/DBSCAN`.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm proposed in 1996 by Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaoping Xu [5]. DBSCAN along with K-means are among the most common clustering algorithms and also most cited in scientific literature.

B.3.1 Concepts

DBSCAN's definition of a cluster is based on the notion of *density reachability*. Basically, a point q is directly density-reachable from a point p if it is not farther away than a

given distance ϵ (i.e. is part of its ϵ-neighborhood) and if p is surrounded by sufficiently many points such that one may consider p and q to be part of a cluster. q is called *density-reachable* (note the distinction from "*directly* density-reachable") from p if there is a sequence $p_1, \ldots, p_n$ of points with $p_1 = p$ and $p_n = q$ where each p_i is directly density-reachable from p_{i-1}.

Note that the relation of density-reachable is not symmetric. q might lie on the edge of a cluster, having insufficiently many neighbors to count as dense itself. This would halt the process of finding a path that stops with the first non-dense point. By contrast, starting the process with q would lead to p (though the process would halt there, p being the first non-dense point). Due to this asymmetry, the notion of *density-connected* is introduced: two points p and q are density-connected if there is a point o such that both p and q are density-reachable from o. Density-connectedness is symmetric.

A cluster satisfies two properties:

1. All points within the cluster are mutually density-connected.
2. If a point is density-connected to any point of the cluster, it is part of the cluster as well.

B.3.2 Algorithm

DBSCAN requires two parameters: ϵ (eps) and the minimum number of points required to form a cluster (minPts). It starts with an arbitrary starting point that has not been visited. This point's ϵ-neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized ϵ-environment of a different point and hence be made part of a cluster.

If a point is found to be a dense part of a cluster, its ϵ-neighborhood is also part of that cluster. Hence, all points that are found within the ϵ-neighborhood are added, as is their own ϵ-neighborhood when they are also dense. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

B.3.3 Pseudocode

N is the set of the objects to be clustered, eps is ϵ and $minPts$ is the minimum number of points required to form a cluster.

```
DBSCAN(N, eps, minPts)
   C ← 0
   for each unvisited point P in dataset N
      mark P as visited
      NeighborPts ← regionQuery(P, eps)
      if sizeof(NeighborPts) < minPts
         mark P as NOISE
      else
         C ← next cluster
         expandCluster(P, NeighborPts, C, eps, minPts)
   end for
```

```
expandCluster(P, NeighborPts, C, eps, minPts)
   add P to cluster C
   for each point P' in NeighborPts
      if P' is not visited
         mark P' as visited
         NeighborPts' ← regionQuery(P', eps)
         if sizeof(NeighborPts') >= minPts
            NeighborPts ← NeighborPts joined with NeighborPts'
      if P' is not yet member of any cluster
         add P' to cluster C
   end for

regionQuery(P, eps)
   return all points within P's eps-neighborhood (including P)
```

B.3.4 Analysis

DBSCAN visits each point of the dataset, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the time complexity is mostly governed by the number of *regionQuery* invocations. DBSCAN executes exactly one such query for each point, and if an indexing structure is used that executes such a neighborhood query in $\mathcal{O}(\log n)$, an overall runtime complexity of $\mathcal{O}(n \cdot \log n)$ is obtained. Without the use of an accelerating index structure, the run time complexity is $\mathcal{O}(n^2)$. Often the distance matrix of size $(n^2 - n)/2$ is materialized to avoid distance re-computations. This however also needs $\mathcal{O}(n^2)$ memory.

DBSCAN's advantages are:

- It does not require one to specify the number of clusters in the data a priori, as opposed to K-means.
- It can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the $minPts$ parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.
- It has a notion of noise.
- It requires just two parameters and is mostly insensitive to the ordering of the points. (However, points sitting on the edge of two different clusters might swap cluster membership if the ordering of the points is changed, and the cluster assignment is unique only up to isomorphism.)

And the disadvantages:

- The quality of DBSCAN highly depends on the metric used in the function $regionQuery(P, \epsilon)$. This not only impacts the quality of the clusters but also the running time of the algorithm, slow metrics applied on big data sets can render the algorithm unusable.
- It cannot cluster data sets well with large differences in densities, since the $minPts{-}\epsilon$ combination cannot then be chosen appropriately for all clusters.

B.4 QuickBundles

From QuickBundles' paper "QuickBundles, a method for tractography simplification" by Garyfallidis et al. [6].

QuickBundles (QB) is a simple and very fast algorithm which can reduce tractography outputs to an accessible structure in a time that is linear in the number of tracts n.

Each tract is a fixed-length ordered sequence of points in $\mathbb{R}^3$, and QB uses comparison functions and amalgamations that take account of and preserve this structure. Moreover each item is either added to an existing cluster on the basis of the distances between the cluster descriptor of the item and the descriptors of the current list of clusters. Clusters are held in a list which is extended according to need. Unlike amalgamation clustering algorithms such as K-means there is no reassignment or updating phase in QB, once an item is assigned to a cluster it stays there, and clusters are not amalgamated. QB derives its speed from this idea.

QB uses the minimum average direct flip distance (MDF) described in appendix A.3. QB stores information about clusters in cluster nodes. Tracts are indexed with $i = 1 \dots n$ where s_i is the $k \times 3$ matrix representing tract i. A cluster node is defined as a triple $c = (In, h, cs)$ where In is the list of the integer indices of the tracts in that cluster, cs is the number of tracts in the cluster (cluster size), and h is the tract sum. h is a $k \times 3$ matrix which can be updated on the fly when a tract is added to a cluster and is equal to:

$$h = \sum_{i=1}^{cs} s_i$$

where s_i is the $k \times 3$ matrix representing i-th tract, and cs is the number of tracts in the cluster. The centroid tract is calculated as $v = h/cs$.

B.4.1 Algorithm

The algorithm proceeds as follows: at any one step in the algorithm there are M clusters. Select the first tract s_1 and place it in the first cluster $c_1 \leftarrow (1, s_1, 1)$. $M = 1$ at this point. For each remaining tract in turn $i = 3 \dots n$:

1. calculate the MDF distance between tract s_i and the centroid tracts v_e of all the current clusters c_e, $e = 1 \dots M$, where v is defined on the fly as $v = h/cs$
2. if any of the MDF values m_e are smaller than a clustering threshold θ add tract i to the cluster e with the minimum value for m_e; $c_e = (In, h, cs)$, and update $c_e \leftarrow (append(In, i), h + s, cs + 1)$
3. otherwise create a new cluster $c_{M+1} \leftarrow ([i], s_i, 1), M \leftarrow M + 1$.

Choice of orientation can become an issue when adding tracts together, because tracts can equivalently have their points ordered $1 \dots k$ or be flipped with order $k \dots 1$. A step in QB takes account of the possibility of needing to perform such a flip of a tract before adding it to a centroid tract according to which direction produced the lowest MDF value.

The complexity of QB is in the best case linear $\mathcal{O}(n)$ with the number of tracts n and worst case quadratic $\mathcal{O}(n^2)$ when every cluster contains only one tract. The average case is $\mathcal{O}(Mn)$ where M is the number of clusters. One of the reasons why QB has on average

linear time complexity derives from the structure of the cluster node: only the sum of current tracts h in the cluster is kept, and the sum is cumulative; moreover there is no recalculation of clusters, the tracts are passed through only once and a tract is assigned to one cluster only.

Appendix C

SCORES

C.1 Inertia

Adapted from from scikit-learn's section on Inertia `http://scikit-learn.org/stable/modules/clustering.html#inertia`

The cluster inertia is the variance of the cluster measured by the squared distance of each tract of the cluster to the cluster centroid (i.e. the sum of the squared distances of the samples to the nearest centroid). Inertia is Kmeans' minimization function (see B.1).

The cluster centroid is the mathematical central vector of the cluster, which may not necessarily be a member of the data set. The centroid of a tract set F is given by

$$F = \{f_1, \ldots, f_n\}$$
$$f_i = \{p_1^i, \ldots, p_k^i\}$$
$$p = \langle p_x, p_y, p_z \rangle$$

$$centroid = \left\{ \frac{\sum_{i=1}^{n} p_1^i}{n}, \ldots, \frac{\sum_{i=1}^{n} p_n^i}{n} \right\}$$

Logically, the less clusters, the more spread the tracts are, the larger the inertia is.

This holds true for every clustering algorithm and for every metric. Hence the inertia is not a model selection metric, it is a score that is useful only when comparing different cluster solutions when the number of clusters is fixed.

The inertia score presents two main drawbacks:

- It makes the assumption that clusters are convex and isotropic which is not always the case, for instance inertia is a useless metric to evaluate clustering algorithms that tries to identify nested circles on a 2D plane.
- Inertia is not a normalized metric: we just know that lower values are better and bounded by zero. One potential solution would be to adjust inertia for random clustering (assuming the number of ground truth classes is known).

C.2 Rand Index

From its Wikipedia Article "Rand Index" `http://en.wikipedia.org/wiki/Rand_index`

The Rand index [19] (named after its author William Rand) is a measure of the similarity between two clusterings.

Let F be our initial set of n brain tracts to cluster $F = \{f_1, \ldots, f_n\}$, the clusters obtained by the clustering algorithm are $C = \{C_1, \ldots, C_s\}$. We also know the *ground truth*: how the tracts should be clustered, the original bundles/classes: $B = \{B_1, \ldots, B_r\}$.

The Rand Index compares the clusters C and the bundles B, it first defines a, b, c and d as follows:

- $a = |\{(f_i, f_j)/f_i, f_j \in B_k, f_i, f_j \in C_l\}|$

  the number of pairs of elements in F that are in the same bundle in B and in the same cluster in C

- $b = |\{(f_i, f_j)/f_i \in B_{k_1}, f_j \in B_{k_2}, f_i \in C_{l_1}, f_j \in C_{l_2}\}|$

  the number of pairs of elements in F that are in different bundles in B and in different clusters in C

- $c = |\{(f_i, f_j)/f_i, f_j \in B_k, f_i \in C_{l_1}, f_j \in C_{l_2}\}|$

  the number of pairs of elements in F that are in the same bundle in B and in different clusters in C

- $d = |\{(f_i, f_j)|f_i \in B_{k_1}, f_j \in B_{k_2}, f_i, f_j \in C_l\}|$

  the number of pairs of elements in F that are in different bundles in B and in the same cluster in C

for some $1 \leq i, j \leq n, i \neq j, 1 \leq k, k_1, k_2 \leq r, k_1 \neq k_2, 1 \leq l, l_1, l_2 \leq s, l_1 \neq l_2$.
The Rand Index is thus defined as:

$$RI = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

Intuitively, $a+b$ can be considered as the number of agreements between B and C and $c+d$ as the number of disagreements between B and C.

The Rand index has a value between 0 and 1, with 0 indicating that the two data clusters do not agree on any pair of points and 1 indicating that the data clusters are exactly the same.

C.2.1 The contingency table

In order to compute the Rand index a contingency table is used. The overlap between B and C can be summarized in a contingency table $[n_{ij}]$ where each entry n_{ij} denotes the number of objects in common between B_i and C_j: $n_{ij} = |B_i \cap C_j|$.

B \ C	C_1	C_2	$\cdots$	C_s	Sums
B_1	n_{11}	n_{12}	$\cdots$	n_{1s}	u_1
B_2	n_{21}	n_{22}	$\cdots$	n_{s2}	u_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
B_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	u_r
Sums	v_1	v_2	$\cdots$	v_s	

We can now define a, b, c and d based on the contingency table:

	Same Cluster	Different Cluster	Sums
Same Bundle	$a = \sum_{i=1}^{r}\sum_{j=1}^{s}\binom{n_{ij}}{2}$	$b = \sum_{i=1}^{r}\binom{u_i}{2} - a$	m_1
Different Bundle	$c = \sum_{j=1}^{s}\binom{v_j}{2} - a$	$d = \binom{n}{2} - a - b - c$	$M - m_1$
Sums	m_2	$M - m_2$	M

were M is the total number of pairs $M = \binom{n}{2}$

Although the lower-limit of this index is 0.0, this value is rarely returned with real data [14]. This is because the Rand index is not corrected for agreement by chance.

C.3 Adjusted Rand Index

From its Wikipedia article Rand Index `http://en.wikipedia.org/wiki/Rand_index`.

A form of the Rand index may be defined that is adjusted for the chance grouping of elements, this is the adjusted Rand index [9, 28]. The adjusted Rand index can yield a value between -1 and +1.

The general form of a statistic S that is corrected for chance is:

$$S' = \frac{S - E(S)}{Max(S) - E(S)}$$

In this equation, $Max(S)$ is the upper-limit of S, and $E(S)$ is the expected value of S. In the case of the Rand index, for $E(RandIndex)$ a hypergeometric distribution [9] is assumed. If S returns its expected value then S' is 0.0, and if S returns a value of 1.0 then S' also returns 1.0.

Therefore the adjusted form of the Rand Index is defined as

$$ARI = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$$

$$ARI = \frac{\sum_{i=1}^{r}\sum_{j=1}^{s}\binom{n_{ij}}{2} - \left[\sum_{i=1}^{r}\binom{a_i}{2}\sum_{j=1}^{s}\binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i=1}^{r}\binom{a_i}{2} + \sum_{j=1}^{s}\binom{b_j}{2}\right] - \left[\sum_{i=1}^{r}\binom{a_i}{2}\sum_{j=1}^{s}\binom{b_j}{2}\right]/\binom{n}{2}}$$

which can be re-written to favor clarity in terms of the values a, b, c, d, m_1, m_2 and M:

$$ARI = \frac{\frac{a+d}{M} - E(\frac{a+d}{M})}{1 - E(\frac{a+d}{M})}$$

$$ARI = \frac{a - \frac{m_1 m_2}{M}}{\frac{m_1+m_2}{2} - \frac{m_1 m_2}{M}}$$

Moberts et al. [15], in their paper "Evaluation of fiber clustering methods for diffusion tensor imaging", compared the scores given by the ARI when applied on clusterings of brain tracts with the opinion of clinicians of what a good clustering is. They discovered that the ARI has two problems when used to measure tract clusterings: big bundles are more important that smaller bundles and completeness and correctness (two concepts introduced

in C.5) are weighted equally, while they should not. They addressed these 2 problems by modifying the ARI, creating two new scores: the Normalized Adjusted Rand Index, or NARI, and the Weighted Normalized Adjusted Rand Index, or WNARI.

C.4 Normalized Adjusted Rand Index (NARI)

From Moberts el al's paper "Evaluation of fiber clustering methods for diffusion tensor imaging" [15]

A problem with the Adjusted Rand index is that it does not account for bundles that are of widely varying sizes. That is, the Adjusted Rand index measures agreement on the level of tracts, not on the level of bundles. As a result, a bundle with a large number of tracts is weighted more than a bundle with a small number of tracts. It can be noticed that whenever big, important bundles like the corpus callosum are complete, the Adjusted Rand index returns a high value whatever the situation of the other bundles is.

To take into account that bundles should be weighted equally, Moberts et al. [15] define a Normalized Adjusted Rand Index (NARI). The idea is to modify the contingency table such that each bundle has the same weight. A way to achieve this is by setting the row sum u_i of each bundle B_i in the contingency table to some nonnegative value h and to multiply each entry n_{ij} by a factor $\frac{h}{u_i}$.

B \ C	C_1	C_2	$\cdots$	C_s	Sums
B_1	$n_{11}\frac{h}{u_1}$	$n_{12}\frac{h}{u_1}$	$\cdots$	$n_{1s}\frac{h}{u_1}$	h
B_2	$n_{21}\frac{h}{u_2}$	$n_{22}\frac{h}{u_2}$	$\cdots$	$n_{s2}\frac{h}{u_2}$	h
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
B_r	$n_{r1}\frac{h}{u_n}$	$n_{r2}\frac{h}{u_n}$	$\cdots$	$n_{rs}\frac{h}{u_n}$	h
Sums	v'_1	v'_2	$\cdots$	v'_s	Rh

As explained in C.2.1 $u_i = \sum_{j=1}^{s} n_{ij}$, it was the sum of the whole row before we multiplied by h/u_i.

Therefore the new values for a, b, c, d, m_1, m_2 are:

	Same Cluster	Different Cluster	Sums
Same Bundle	$a' = \sum_{i=1}^{r}\sum_{j=1}^{s}\binom{h\frac{n_{ij}}{u_i}}{2}$	$b' = r\binom{h}{2} - a'$	m'_1
Different Bundle	$c' = \sum_{j=1}^{s}\binom{v'_j}{2} - a'$	$d' = \binom{rh}{2} - a' - b' - c'$	$M' - m'_1$
Sums	m'_2	$M' - m'_2$	M'

Moberts et al. then propose to set $\lim_{h\to\infty}$, thereby pretending there are an infinite amount of tracts, which gives more stable results. Then the Normalized Adjusted Rand Index is defined as:

$$NARI = \lim_{h\to\infty} \frac{a' - \frac{m'_1 m'_2}{\binom{rh}{2}}}{\frac{m'_1 + m'_2}{2} - \frac{m'_1 m'_2}{\binom{rh}{2}}} = \frac{2f - 2rg}{2f - rf - r^2}$$

with

$$f = \sum_{j=1}^{s} \left(\sum_{i=1}^{r} \right)^2$$

$$g = \sum_{i=1}^{r} \sum_{j=1}^{s} \frac{n_{ij}^2}{u_i^2}$$

C.5 Correctness (Homogeneity) and Completeness

Rosenberg and Hirschberg (2007) [20] define the following two desirable objectives for any cluster assignment:

- Correctness (Homogeneity): each cluster contains only members of a single class. Tracts belonging to different anatomical structures should not be clustered together. Homogeneity penalizes the clustering scheme in which samples from different classes are clustered together.
- Completeness: all members of a given class are assigned to the same cluster. Tracts belonging to the same anatomical structure should be clustered together. It penalizes the clustering scheme in which samples from the same class are clustered separately.

C.6 Weighted Normalized Adjusted Rand Index (WNARI)

From Moberts el al's paper "Evaluation of fiber clustering methods for diffusion tensor imaging" [15]

Another problem is that physicians consider an incorrect clustering worse than an incomplete clustering. In an incorrect clustering, tracts belonging to different anatomical bundles are clustered together, which makes it difficult to distinguish between bundles. This makes a correct clustering visually more appealing than a complete clustering.

Moberts et al. propose a final modification to the Adjusted Rand Index that enables it to weigh correctness and completeness differently.

By defining the Rand Index in terms of the normalized contingency table we obtain:

$$NRI = \frac{a' + d'}{a' + b' + c' + d'} = 1 - \frac{b'}{M'} - \frac{c'}{M'}$$

The fraction $\frac{b'}{M'}$ indicates the incompleteness of the cluster and the fraction $\frac{c'}{M'}$ its incorrectness. By adding an extra parameter $0 \leq \alpha \leq 1$ they define the Weighted Normalized Index WNRI to weight both concepts:

$$WNRI = 1 - 2(1 - \alpha)\frac{b'}{M'} - 2\alpha\frac{c'}{M'}$$

Finally the adjust the WNRI for chance agreement:

$$WNARI = \lim_{h \to \infty} \frac{WNRI - E(WNRI)}{1 - E(WNRI)}$$

where

$$E(WNRI) = 1 - 2(1-\alpha)E\left(\frac{b'}{M'}\right) - 2\alpha E\left(\frac{c'}{M'}\right) \tag{C.1}$$

$$= 1 - 2(1-\alpha)\frac{m'_1(M'-m'_2)}{M'^2} - 2\alpha\frac{m'_2(M'-m'_1)}{M'^2} \tag{C.2}$$

since the expected value of b' is $\frac{m'_1(M'-m'_2)}{M'}$ and the expected value of c' is $\frac{m'_2(M'-m'_1)}{M'}$. Finally, the WNARI is:

$$WNARI = \frac{f - rg}{f - \alpha rf - r^2 - \alpha r^2}$$

with f and g as before:

$$f = \sum_{j=1}^{s}\left(\sum_{i=1}^{r}\right)^2$$

$$g = \sum_{i=1}^{r}\sum_{j=1}^{s}\frac{n_{ij}^2}{u_i^2}$$

Moberts et al. conducted an experiment in which they showed the results of different clusterings to physicians and discovered that a value of $\alpha = 0.75$ matches exactly their opinion of what a good cluster and a bad cluster were. Intuitively, if we are looking at two clusters that belong to the same bundle (they should be one cluster), it is way easier to us to visually put it together, while it is very difficult to visually separate one given cluster that contains two bundles.

BIBLIOGRAPHY

[1] Guillaume Auzias et al. Disco: A coherent diffeomorphic framework for brain registration under exhaustive sulcal constraints. In *LNCS*, volume 5761, pages 730–738, 2009.

[2] Matthan W A Caan, Lucas J van Vliet, Charles B L M Majoie, Maaike M van der Graaff, C. A. Grimbergen, and Frans M Vos. Nonrigid point set matching of white matter tracts for diffusion tensor image analysis. *IEEE Trans Biomed Eng*, 58(9):2431–2440, Sep 2011.

[3] Lynn E. DeLisi et al. Early detection of schizophrenia by diffusion weighted imaging. *Psychiatry Research: Neuroimaging*, 148(1):61 – 66, 2006.

[4] Stanley Durrleman, Pierre Fillard, Xavier Pennec, Alain Trouvé, and Nicholas Ayache. A statistical model of white matter fiber bundles based on currents. *Inf Process Med Imaging*, 21:114–125, 2009.

[5] Martin Ester, Hans peter Kriegel, Jörg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[6] Eleftherios Garyfallidis et al. Quickbundles, a method for tractography simplification. *Frontiers in Neuroscience*, 6(175), 2012.

[7] P. Guevara, D. Duclap, C. Poupon, L. Marrakchi-Kacem, P. Fillard, D. Le Bihan, M. Leboyer, J. Houenou, and J-F. Mangin. Automatic fiber bundle segmentation in massive tractography datasets using a multi-subject bundle atlas. *Neuroimage*, 61(4):1083–1099, Jul 2012.

[8] P. Guevara, C. Poupon, D. Rivière, Y. Cointepas, M. Descoteaux, B. Thirion, and J-F. Mangin. Robust clustering of massive tractography datasets. *Neuroimage*, 54(3):1975–1993, Feb 2011.

[9] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.

[10] Hai Li, Zhong Xue, Lei Guo, Tianming Liu, Jill Hunter, and Stephen T C Wong. A hybrid approach to automatic clustering of white matter fibers. *Neuroimage*, 49(2):1249–1258, Jan 2010.

[11] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, volume 1, pages 281–297. Univ. of Calif. Press, 1967.

[12] Mahnaz Maddah, W. Eric L Grimson, Simon K Warfield, and William M Wells. A unified framework for clustering and quantitative analysis of white matter fiber tracts. *Med Image Anal*, 12(2):191–202, Apr 2008.

[13] Mahnaz Maddah, James V Miller, Edith V Sullivan, Adolf Pfefferbaum, and Torsten Rohlfing. Sheet-like white matter fiber tracts: representation, clustering, and quantitative analysis. *Med Image Comput Comput Assist Interv*, 14(Pt 2):191–199, 2011.

[14] Glenn W. Milligan and Martha C. Cooper. A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research*, 21(4):441–458, 1986.

[15] B. Moberts, A. Vilanova, and J.J. van Wijk. Evaluation of fiber clustering methods for diffusion tensor imaging. In *Visualization, 2005. VIS 05. IEEE*, pages 65 – 72, oct. 2005.

[16] Andriy Myronenko, Xubo Song, and Miguel Carreira-Perpinan. Non-rigid point set registration: Coherent point drift. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS) 19*, pages 1009–1016. MIT Press, Cambridge, MA, 2007.

[17] L. J. O'Donnell, M. Kubicki, M. E. Shenton, M. H. Dreusicke, W. E L Grimson, and C. F. Westin. A method for clustering white matter fiber tracts. *AJNR Am J Neuroradiol*, 27(5):1032–1036, May 2006.

[18] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[20] Andrew Rosenberg and Julia Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[21] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *JCAM*, 20(0):53 – 65, 1987.

[22] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1177–1178, New York, NY, USA, 2010. ACM.

[23] V. Siless, S. Medina, G. Varoquaux, and B. Thirion. A comparison of metrics and algorithms for fiber clustering. In *Pattern Recognition in Neuroimaging (PRNI), 2013 International Workshop on*, pages 190–193, 2013.

[24] Viviana Siless et al. Joint T1 and brain fiber log-demons registration using currents to model geometry. In *MICCAI*, pages 57–65, 2012.

[25] Viviana Siless, Pamela Guevara, Xavier Pennec, and Pierre Fillard. Joint T1 and Brain Fiber Diffeomorphic Registration Using the Demons. In Tianming Liu, Dinggang Shen, Luis Ibanez, and Xiaodong Tao, editors, *Multimodal Brain Image Analysis*, volume 7012 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin / Heidelberg, 2011.

[26] H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804, 1956.

[27] N. Toussaint, J.C. Souplet, and P. Fillard. Medinria: Medical image navigation and research tool by inria. In *Proc. of MICCAI'07 Workshop on Interaction in medical image analysis and visualization*, Brisbane, Australia, 2007.

[28] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA, 2009. ACM.

[29] Qian Wang, Pew-Thian Yap, Guorong Wu, and Dinggang Shen. Application of neuroanatomical features to tractography clustering. *Hum Brain Mapp*, Mar 2012.

[30] Xiaogang Wang, W. Eric L Grimson, and Carl-Fredrik Westin. Tractography segmentation using a hierarchical dirichlet processes mixture model. *Neuroimage*, 54(1):290–302, Jan 2011.

[31] D. Wassermann, L. Bloy, E. Kanterakis, R. Verma, and R. Deriche. Unsupervised white matter fiber clustering and tract probability map generation: applications of a gaussian process framework for white matter fibers. *Neuroimage*, 51(1):228–241, May 2010.

[32] Yan Xia, U. Turken, Susan L Whitfield-Gabrieli, and John D Gabrieli. Knowledge-based classification of neuronal fibers in entire brain. *Med Image Comput Comput Assist Interv*, 8(Pt 1):205–212, 2005.

[33] Orly Zvitia, Arnaldo Mayer, Ran Shadmi, Shmuel Miron, and Hayit K Greenspan. Co-registration of white matter tractographies by adaptive-mean-shift and gaussian mixture modeling. *IEEE Trans Med Imaging*, 29(1):132–145, Jan 2010.