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 las imágenes dMRI contrastadas con métodos de tractografía pueden ayudarnos a comprender los patrones de conexión entre las distintas regiones del cebrero y a caracterizar trastornos neurológicos.

Debido a la enorme cantidad de información producida por dichos análisis y 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.

Proponemos explorar el conocido algoritmo de clustering K-means y una alternativa recientemente propuesta para tratar el agrupamiento de vías neuronales: QuickBundles [7].

Proponemos un procedimiento eficiente para asociar K-means con un Modelo de Densidad de Puntos (PDM, por sus siglas en inglés), una métrica propuesta recientemente para analizar estructuras geométricas.

Analizamos la performance y la usabilidad de estos algoritmos en datos simulados y en una base de datos de 10 sujetos.

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

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 contrasted with 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.

We propose to explore the well-known K-means clustering algorithm and QuickBundles, an alternative algorithm that has been proposed recently to deal with tract clustering [7].

We propose an efficient procedure to associate K-means with Point Density Model, a recently proposed metric to analyze geometric structures.

We analyze the performance and usability of these algorithms on simulated data and a database of 10 subjects.

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

1.1 Introduction

In the mid 1970s an image modality called Magnetic Resonance Imaging (MRI) was designed and is now widely used in radiology to visualize the body's internal structures. This technique, in vivo and non invasive, makes use of the nuclear magnetic resonance (NMR) phenomenon to obtain high quality images of tissues' structures. These images are nowadays used by clinicians and scientists 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 problems that MRI scans help to detect.

Other MRI modelities appeared in the 90's, two of the most important ones are functional MRI (fMRI) and diffusion MRI (dMRI). The latter measures the movement, the diffusion, of water molecules in the tissues. Their movement is not free, but it shows their interaction with many obstacles as macromolecules, tracts, membranes, etc. Therefore the pattern of water molecules diffusion can reveal microscopic details of the tissues' architecture, like the brain's complex tract network.

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 of the grey matter.

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

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

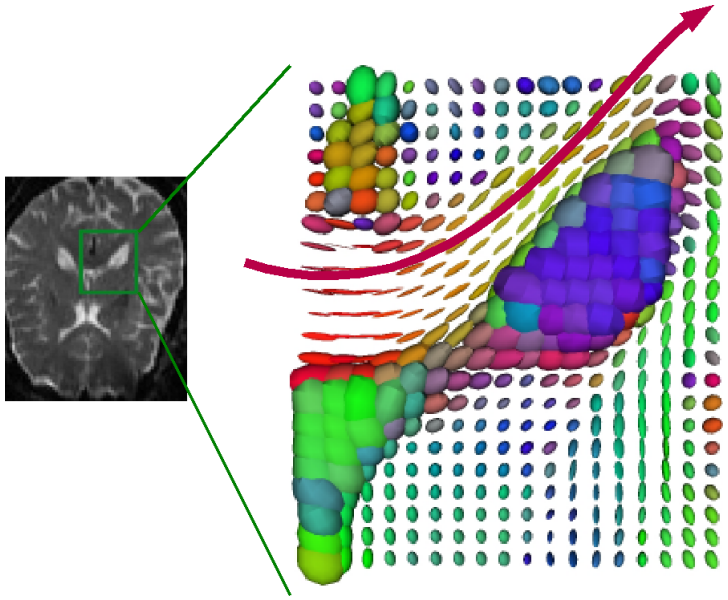


Fig. 1.1: Diffusion MRI provides local models of water diffusion that can be used to detect tracts using tractography algorithms.

Motivation

Tracts not only provide information about the connections between the different cortical and subcortical areas, but they are also used to detect certain anomalies (e.g. tumors), to compare the brains of groups of patients (both diseased and healthy), etc. Scientists have recently started to use them to improve registration algorithms applied to brain images [26, 35] (*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 the second image).

Tractography methods only recover part of the true brain tracts. Moreover, 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.

Fig. 1.2: Brain tracts estimated by a full brain tractography. The color indicates the main direction of the movement 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 viceversa) and blue along the Z-axis (foot-to-head or viceversa).

For these reasons we aim at grouping tracts with similar properties or characteristics. This is useful for various reasions: 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 simplifes subsequent algorithmic steps. Furthermore, it makes them robust against aberrant tractography results, noise, and outlier tracts.

Methods

Clustering of simple structures like n-dimensional points is a vastly explored area, while clustering more complicated objects (e.g. in non-linear space) is much more challenging.

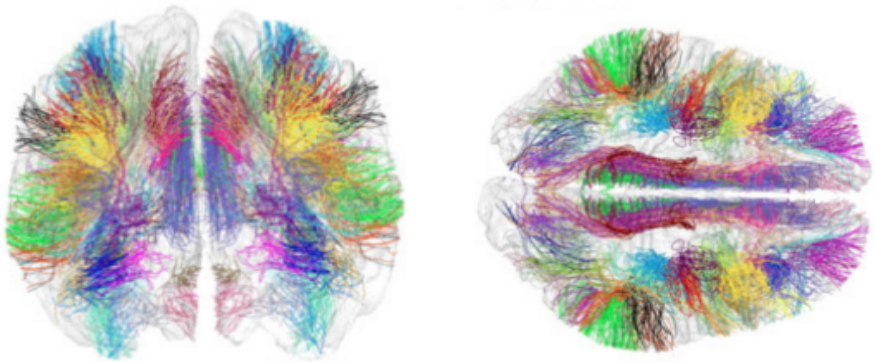
The main problem that we face is to define a relevant metric to assess the similarity of tracts. Another one is to pick the right number of classes, which is a very hard problem in general as we are in an unsupervised setting. We will not try to solve this issue in particular in this thesis.

We will approach the problem firstly by using known algorithms which have been proven to give good results like K-Means and DBSCAN [5]. Different other approaches have been proposed in the literature [30, 14, 31, 2, 32, 35, 11, 13, 33, 18] 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 some trade-offs: one can opt for a fast metric (in terms of computational cost), or one that helps maintain the geometric shape of the cluster's tracts, or one which seek to keep the cluster compact, etc. One can also use atlases to group tracts in well-known bundles. Some approaches are the Euclidian and Hausdorff distances, Mean Closest Point, Currents [4], Point Density Model [17], and Gaussian Processes [32].

We will evaluate the implications of choosing a given metric and how it combines with the different algorithms, which will in turn be analyzed regarding speed and suitability to the task. The scalability of both the clustering algorithm and the metrics, although not critical for our needs, will also be taken into account.

Once the clusters are obtained it is necessary to evaluate qualitatively and quantitatively if the result is satisfactory [16]. Moreover, the algorithm should ideally show some consistency of the resulting clusters across individuals and with existing atlases (if they were not used during the clustering). We would also like to measure the impact of the clusters in further processing steps such as brain coregistration.



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

Fig. 1.3: Brain tracts clusters by P. Guevara et al. [9, 8]

Supervised vs. Unsupervised Learning

Machine learning algorithms are described as either "supervised" or "unsupervised". The distinction is drawn from how the learner classifies data. In supervised algorithms, the classes are predetermined. These classes can be conceived as a finite set, previously obtained by a human or any kind of oracle. In practice, a certain segment of data will be

labelled with these classifications. The machine learner's task is to search for patterns and construct mathematical models. These models then are evaluated on the basis of their predictive ability in relation to measures of variance in the data itself. Examples of supervised learning techniques are decision tree induction, naïve Bayes, etc..

Unsupervised learners are not provided with classifications. In fact, the basic task of unsupervised learning is to develop classification labels automatically. Unsupervised algorithms seek out similarity between pieces of data in order to determine whether they can be characterized as forming a group. These groups are termed clusters, and there are a whole family of clustering machine learning techniques.

In unsupervised classification, often known as "cluster analysis" the machine is not told how the items are grouped. Its task is to arrive at some grouping of the data. In Kmeans, one of the most common algorithms for cluster analysis, the machine is told in advance how many clusters it should form – a potentially difficult and arbitrary decision to make.

It is apparent from this minimal account that the optimization step involved in unsupervised classification is somewhat ill-posed. Some initialization is taken more or less arbitrarily and the learning algorithms reach iteratively a stable configuration that makes sense. The results vary widely and may be completely off if the first steps are wrong. On the other hand, cluster analysis has great potential to reveal unexpected details of the data. And it has considerable corroborative power if its internal comparisons of low-level features of the items lead to groupings that make sense at a higher interpretative level or that one had suspected but deliberately withheld from the machine. Thus cluster analysis is a very promising tool for the exploration of relationships among groups.

2. METHODS

The core of this project is based on the *scikit-learn* [19], an open source machine learning library for Python. The tractographies and its visualizations were done thanks to medInria [28], an open source medical image processing and visualization software: `http://med.inria.fr/`.

2.1 Metrics

A metric is a function that measures distances in a given space. Besides the well-known Euclidean distances many other metrics have been invented to fulfil different needs. In the case of tracts, lines in a 3D space, we consider the four metrics described hereafter.

Given two tracts represented as a sequence of points in a 3-dimensional space $X = \{x_1, x_2, ..., x_{n_k}\}$ and $Y = \{y_1, y_2, ..., y_{y_k}\}$ where $x_i, y_j \in \mathbb{R}^3 \ \forall 1 \leq i \leq x_k, \forall 1 \leq j \leq y_k$ we define four candidate metrics in the next subsections.

Two of these metrics need us to resample tracts X and Y so they have the same amount of points each. We assume from now on that an initial discretization of tracts has been applied, so that all tracts have the same number of points K. This is a achieved by a simple linear interpolation. Tracts are now $X = \{x_1, x_2, ..., x_k\}$ and $Y = \{y_1, y_2, ..., y_k\}$ where $x_i, y_j \in \mathbb{R}^3 \ \forall 1 \leq i, j \leq k$.

Hausdoff and Mean Closes 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 [7] for QuickBundles.

Additionally, we introduce Point Density Model:

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 [25].

We propose the 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: Given a tract X, we represent it as the sum of Dirac concentrated at each tract point: $\frac{1}{k}\sum_{i=1}^{k}\delta_{x_i}$ (resp. Y).

Let K_σ be a Gaussian kernel with scale parameter σ, we can conveniently define the scalar product between two tracts as follows:

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

The definition of the Gaussian Kernel K is:

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

The Point Density Model distance is thus defined as:

$$PDM^2(X,Y) = \|X\|^2 + \|Y\|^2 - 2\langle X, Y\rangle$$

This distance captures misalignment and shape dissimilarities at the resolution σ. Distances much larger or much smaller than σ do not influence the metric.

Point Density Model is $\mathcal{O}(n^2)$: 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, it is actually NP-hard. 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 which offers faster running times in exchange of small looses in cluster quality. We will also experiment with DBSCAN which is strong where K-means is weak.

Our proposed solution, explained in 2.5 uses K-means at its core. We'll compare our results againts QuickBundle, a trac-oriented clustering algorithm proposed by Garyfallidis et al [7] in 2012.

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

Ideally we would simply want to run these algorithms, using the metrics described in section 2.1, on the tracts outputted by the tractography procedure. Unfortunately given the time-complexity of the resulting algorithm and the size of the input sets, this is unviable as a single run can take days.

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

2.3 Multidimensional Scaling

To-do/Question: rewrite and remove the following items

- que es
- para que se usa
- para que lo usamos nosotros
- aclaracion: no reducimos dimensiones. podriamos haber probado eso. agregar a seccion future work.

Multidimensional Scaling 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. The dimension of the generated objects is a parameter in the algorithm which means that the original objects that generated the distace matric, which could have been in a n-dimensional space, 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.

2.3.1 Mathematical Definition

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

The data to be analyzed is a collection of F 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:

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

The goal of MDS is, given Δ, to find F' tract $x_1, \ldots, x_I \in \mathbb{R}^N$ such that

$$\|x_i - x_j\| \approx \delta_{i,j} \forall i, j \in F$$

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, for instance we could use the metrics described above.

In other words, MDS attempts to find an embedding from the F tracts into $\mathbb{R}^N$ such that distances are 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 $(x_1, \ldots, x_I)$ is found as a minimizer of some cost function, for example:

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

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.

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 Combining MDS with metrics

The metrics described in section 2.1 better capture the distance between two tracts but they have an enormous disadvantage: they are very costly computational-wise. Point Density Model and Hausdorff for instance are quadratic on the number of points per tract k. 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 neural pathways we decided to combine them with MDS to maintain the algorithm whitin a reasonable running time while incorporating the information provided by the metrics.

The algorithm goes as follows: say we have a group F of n tracts $f_1, \ldots, f_n$. Each tract f_i has already been resampled so it has a resolution of k (i.e. 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 size $\langle n, p \rangle$. We then apply MDS to this matrix to obtain a new set F' of n tract-like points (i.e. points belonging to $\mathbb{R}^{3k}$). These new points will capture the original information of the distance between them, which is the information that will be use afterward by the clustering algorithm.

By using a random sample of p elements, where $p << N$, we decrease the complexity to $\mathcal{O}(Npk^2)$ which in turn decreases the running time in a significant amount. Note that as a first approach we do not intend to reduce the dimension of the points, this is something we wanted to explore but we unfortunately did not have the time during this project.

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 clustering algorithm

1: F ←remove tracts shorter than *ff* mm from F_{orig}
2: resample every tract in F to exactly k points each
3: $S \leftarrow$ obtain random sample of F taking p% of the tracts
4: $M \leftarrow$ compute full distance matrix of tracts from S to F using metric m
5: $F' \leftarrow$ apply multi dimensional scaling, keep the same dimension $= 3k$
6: $C \leftarrow$ cluster F' with algorithm c (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 maps 1-to-1 to the original set F and approximately preserves the input distances, therefore information from the costly tract metric. MDS starts with a distance matrix and assigns a location to each tract in N-dimensional space (in our case, number of points wanted per tract times the dimension of the points, 3 as we are in R^3). Here we use it asymmetrically, using the classical Nyström's approach for efficient dimension reduction [6].

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 increase.

Both QB and Kmeans+metric running times are sensitive to the number of clusters, however QuickBundles' time complexity is $O(NCk)$ and Kmeans+metric $O(NSk^2+NCk)$, 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 $Sk > C$.

3. VALIDATION SCHEME

Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular any evaluation metric should not take the absolute values of the cluster labels into account but rather if this clustering define separations of the data similar to some ground truth set of classes or satisfying some assumption such that members belong to the same class are more similar that members of different classes according to some similarity metric.

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, rand index and its variants and mutual information and its variants.

3.1 Unsupervised Setting Scores

As we expained before, the most complicated scenario is when we do not have the real bundles to compare our clusters with. Unfortunately this is also the most common scenario. Below we explain the Inertia and the Silhouette scores, which makes it possible to rate clustering results under an unsupervised setting.

The most famous unsupervised score is the Inertia (explained in appendix C) but we'll be using mostly the Silhouette Coefficient.

3.1.1 Silhouette Coefficient

The Silhouette Coefficient [22] measures how close a tract is to its own cluster in comparison to the rest of the clusters, i.e. whether there is another cluster that might represent it 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 big drawback of this score is that an exact computation of it will require 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 reduces the computation of the distances by approximating. We take a random sample s of the full tract set S and the labeling l and compute a partial distance matrix D of size $\langle S \times s \rangle$ 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 a random tract set (see section 4.2.2).

3.2 Supervised Setting Scores

Given a set $F = \{f_1, \ldots, f_n\}$ of N tracts, the knowledge of the ground truth classes/bundles, i.e. a partition $B = \{B_1, \ldots, B_r\}$ of the original set of tracts F and our clustering algorithm assignments, another partition of F, $C = \{C_1, \ldots, C_s\}$ a handful of scores that describe how similar both partitions are can be defined.

There are many scores for supervised settings, we'll be using the Rand Index and it's adjusted for chance version the Adjusted Rand Index (ARI). Correctness, Completeness, and V-measure. The Normalized ARI and the Weighted Normalized ARI, two modifications of the ARI by Moberts et al [16] specific for clusterings of brain tracts. And the Mutual Information score, and its adjusted for change version the Adjusted Mutual Information Score. The definition of all these scores can be found in appendix C.

4. EXPERIMENTS AND RESULTS

4.1 Section Outline

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

1. Point Density Model metric has a free parameter σ, it's resolution. We will 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 impacts the quality of the output clusters and the running time of the algorithm. We'll explore it's impact in section 4.2.2 and we will fix it for all our experiments.

3. The first step when experimenting with problems whose solution is unknown is to run the candidate algorithms on a reduced dataset, one where the solution is known, i.e. a selection of fibers whose original clusterization is defined beforehand by us, hence a "Manually Labelled Dataset". As with this data set the expected clusterization, the solution, is known, we'll be able to use the scores for supervised settings scores we explained in section 3.2. We will also see if the results provided by these scores are aligned with the ones provided by the Silhouette Coefficient, the score we will use when running the algorithm of a full brain tractography, therefore on an unsupervised setting.

4. DBSCAN: we ran DBSCAN first on the manually labelled 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 siginificantly faster than K-means while generating clusters which are slightly worse. We ran 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 labelled dataset introduced in section 4.3 and then on full tractographies with thousands of tracts.

4.2 Parameter Exploration

Our algorithm, described in section 2.5, has 2 free paramaters: p, the size of the random sample used by MDS, and σ, the resolution of Point Density Model metric. We'll analyse their impact in this section.

4.2.1 Sigma: PDM Kernel Resolution

We performed a parameter selection test over 10 subject to analyze the impact of the kernel size for K-means with Point Density Model. We vary σ from 10 to 60mm and the number of clusters from 200 to 1200. We noticed that after $\sigma = 42$mm the quality of the

clusters stop increasing in a significant amount, as shown in Figure 4.1. For the following experiments we fixed $\sigma = 42$mm.

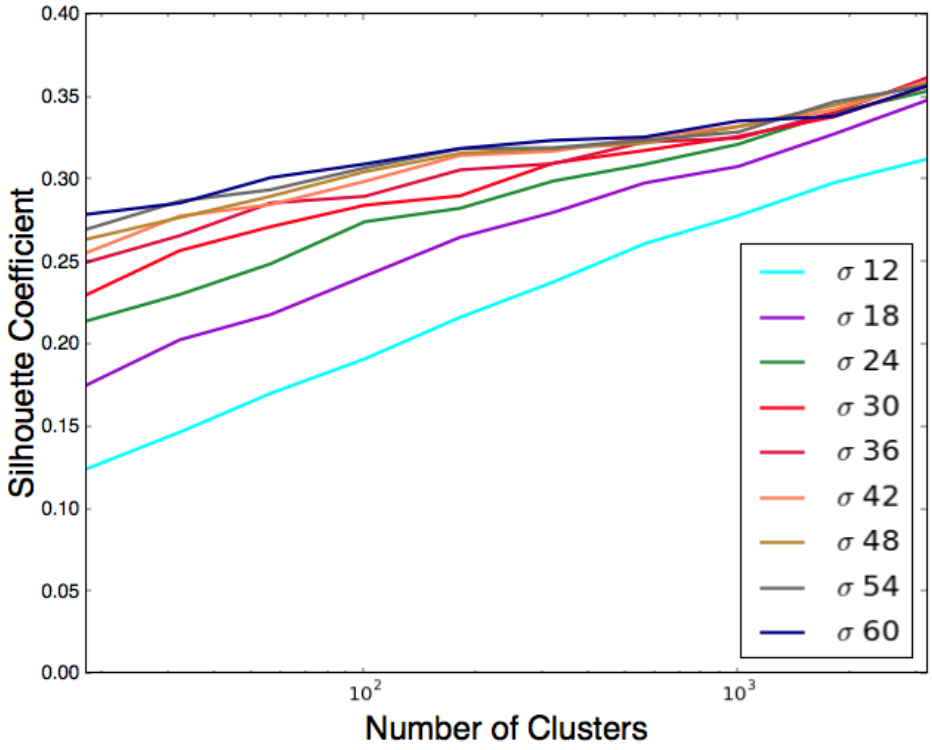


Fig. 4.1: σ pdm kernel resolution.

4.2.2 MDS random sample and approximation of Silhouette

To-do/Question: explicar por que p se relaciona con la aproximacion de silhouette To-do/Question: no se bien como hacer la estadistica, o el t-test

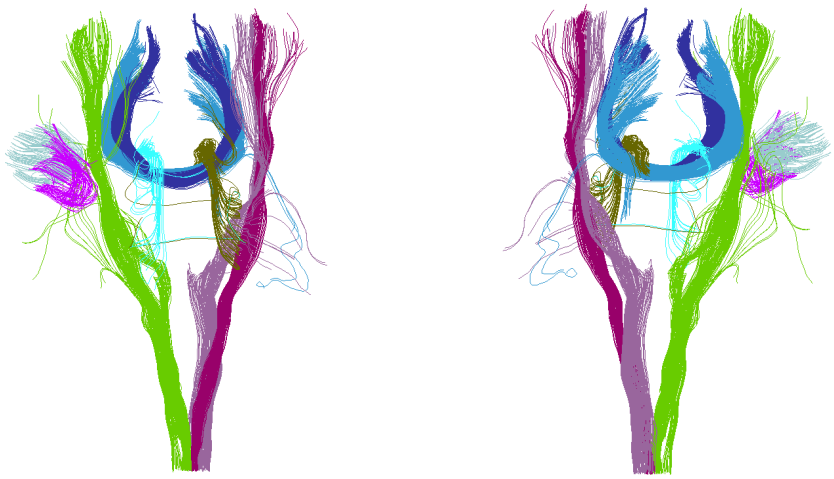
We wanted to see how the different choices for the sample size used by the MDS step impacted the silhouette score. In order to measure it, we ran the full algorithm on 10 arbitrary chosen subject using 50% of the tracts for the sample size, then 30%, 20%, 10% and finally only 5%. The biggest difference in silhouette scores was 10^{-2} we therefore safely consider that it is not significant.

4.3 Manually Labelled Dataset

We tested the algorithms on a subset of 1163 real tracts previously identified from the corpus callosum, corticospinal tract, u-shape, and fronto-occipital. This group was selected for the following reasons:

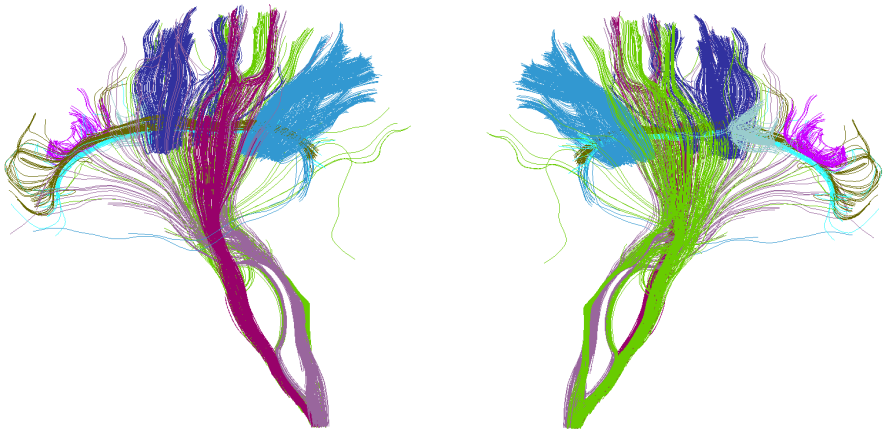
- It has similar tracts (in shape and trajectory) that belong to different hemispheres (they are "mirrored") and therefore should be clustered separately
- It contains some well-known tract bundles of the brain, like the corticospinal tract and the corpus callosum

- The 2 small u-shaped bundles are close enough and similar enough for some algorithms to cluster them together, but they should be in different clusters
- It contains tracts that we would like the algorithm to detect as outliers (of course not when we force the number of clusters to exactly 9 but when using more 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



(a) Anterior or Rostral (Front Surface) *(b)* Posterior or Caudal (Back Surface)

Fig. 4.2: Front and back views

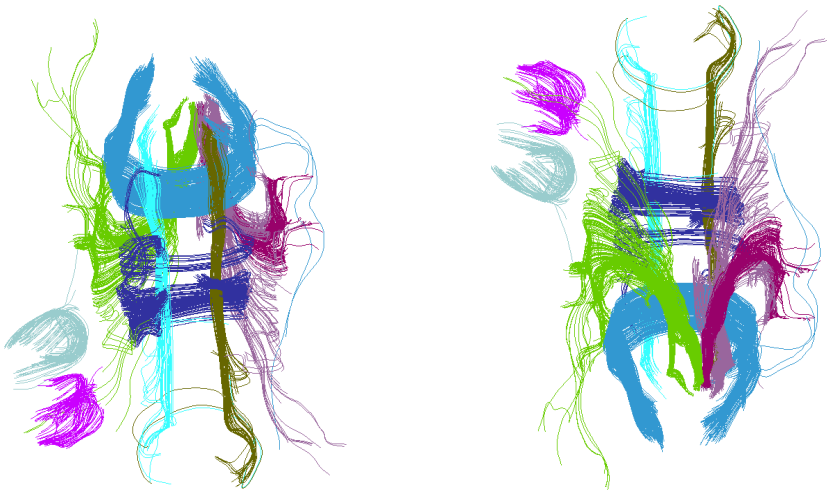


(a) Lateral Left *(b)* Lateral Right

Fig. 4.3: Lateral views

4.4 DBSCAN experiments and results

We first ran DBSCAN on the manually labelled dataset, where we knew beforehand the amount of clusters we wanted to recover. The algorithm could always recover the clusters,



(a) Superior or Dorsal (Top Surface)

(b) Inferior or Ventral (Bottom Surface)

Fig. 4.4: Top and bottom views

presenting scores 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. About 1200 tracts were analyzed and when scores were at their best about 30 to 50 tracts were considered as outliers. We therefore proceeded to run DBSCAN with each of the four metrics on 10 healty subjects.

The results were very similar across all the subjects and for every metric, i.e. while they differed in some specific values they did follow the same trend.

Below we present the results for one subject chosen at random. The metric used was Hausdorff and the total amount of tracts was 27066. In figure 4.5 we show the total number of clusters obtained with respect to DBSCAN's `eps` parameter (the dotted blue line); as explained in section B.0.6, `eps` is the neighborhood-defining distance.

We can see that the maximum number of clusters DBSCAN could obtain was almost 350 that is below what one could expect. In order to obtain a useful separation, researchers usually pick 500 clusters as a minimum (and up to 1000). With fewer, some areas of the brain that are expected to be separated end up being grouped together.

Moreover, we also need to look at the amount of tracts that were considered as outliers and therefore not clustered. The full red line in the same figure 4.5 shows the total numbers of clusters obtained. Let us also keep in mind that before clustering with DBSCAN we already removed tracts smaller than 40mm, which we consider as outliers. For instance, when DBSCAN obtained 347 clusters, only 12974 tracts out of a total of 24240 were clustered and not considered as outliers (and 2826 were smaller than 40mm).

Our conclusion is that DBSCAN isn't very well suited for the task of clustering neural pathways. Therefore we won't use it.

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

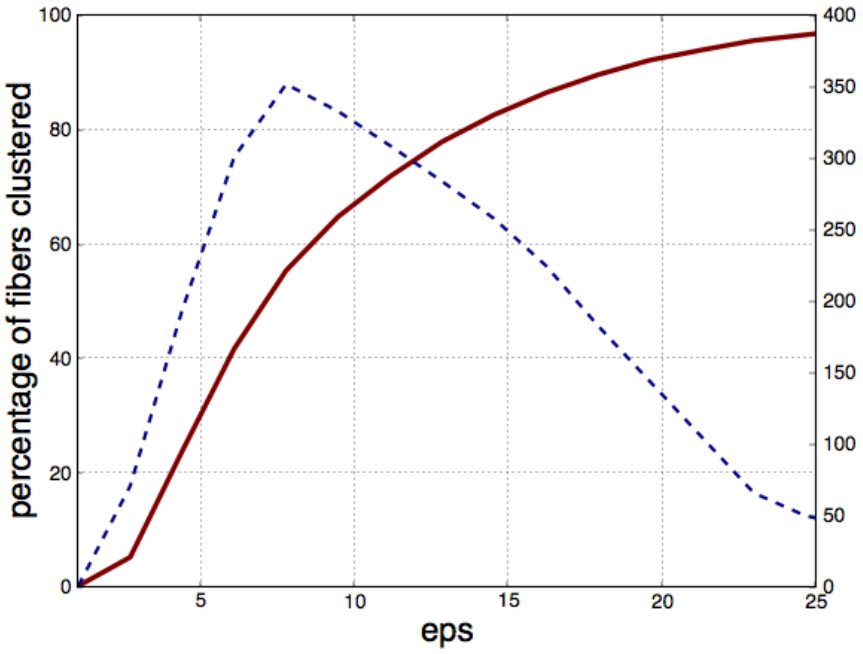


Fig. 4.5: DBSCAN: total number of clusters obtained (dotted blue line) and percentage of tracts that were clustered (full red line)

slightly worse than those obtained with vanilla Kmeans.

We tested both algorithms with all the metrics on 5 random subjects, taking random subsets of the full set of tracts and varying the number of clusters using a logarithmic scale (100, 180, 320 and 560). As both algorithms produce slightly different clusters depending on the random initialization, we ran each experiment 5 times and averaged the results. We compared the differences regarding silhouette score, inertia and running time.

Figure 4.6 shows the differences in running time. As explained in section 2.5 most of the processing time is spent generating the new tract set that the clustering algorithms will 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. Even if for 30000 tracts we can afford to wait around 5 minutes, as our application does not focus on speed, it is evident that for larger number of tracts Kmeans turns unaffordable.

Next we compare the inertia of the resuls of both algorithms. All the metrics followed the same trend so we picked one at random, averaged the results and plotted the difference between Mini Batch Kmeans' inertia and Kmeans' inertia. We can see that for fewer clusters (100, 180 and 320) the difference is smaller but Kmeans always has a lower inertia. For 560 clusters the difference gets larger. As the inertia is not a bounded score is more complicated to understand what do these differences mean. The Silhouette scores, bounded between -1 and 1, will illustrate it better.

Finally we plot in figure 4.8 the difference in silhouette score, like with inertia Kmeans yields a slight better silhouette for 100, 180 and 320 clusters, for 560 the difference can be as large as 0.1 which is definitely not insignificant.

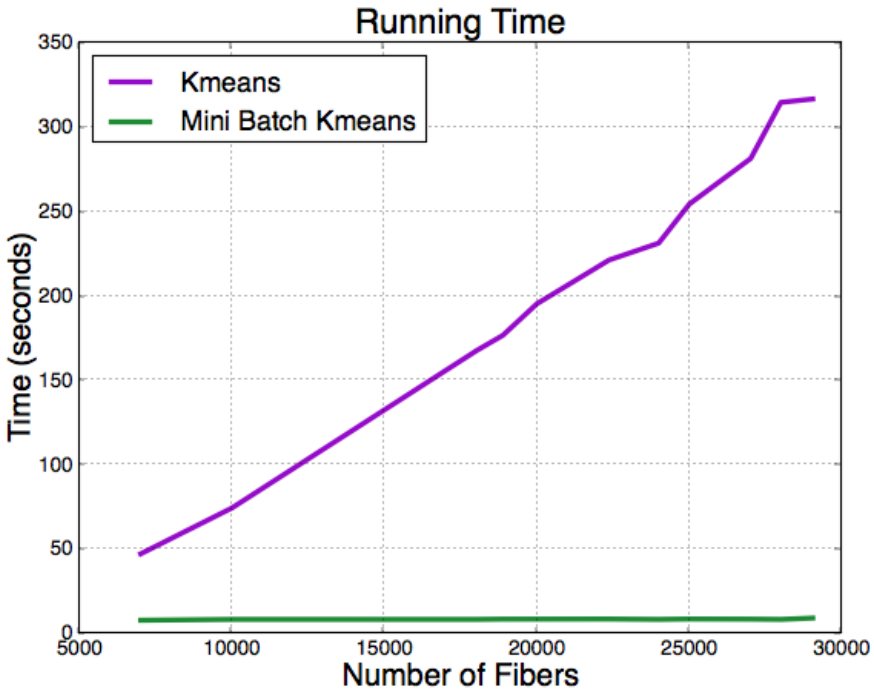


Fig. 4.6: Differences in running time for 560 clusters.

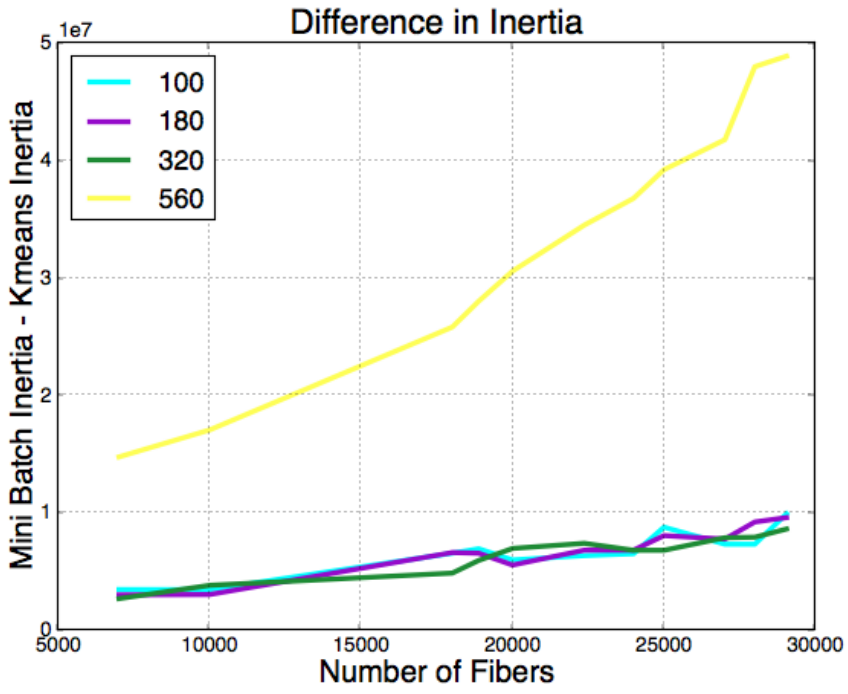


Fig. 4.7: Mini Batch Kmeans' Inertia - Kmeans' Inertia for 100, 180, 320 and 560 clusters.

We conclude that for the specific domain of tract clustering, Kmeans' better results over Mini Batch Kmeans are what we want to keep. As time performance is not key in our study we can afford waiting about 6/7 minutes for Kmeans to finish. Moreover let's keep

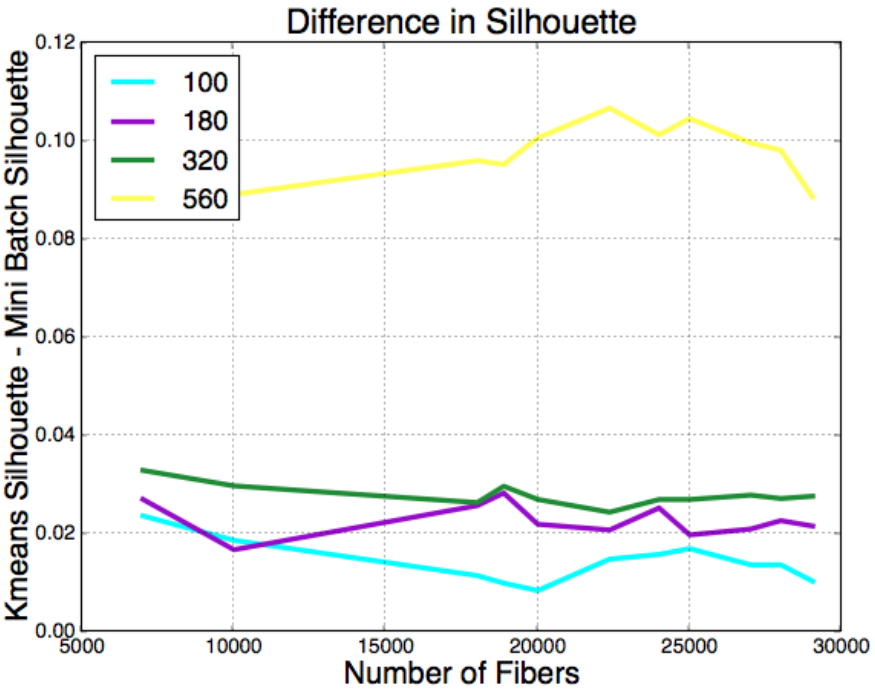


Fig. 4.8: Kmeans' Silhouette - Mini Batch Kmeans' Silhouette for 100, 180, 320 and 560 clusters.

in mind that in our algorithm the generation of the matrix consumes most of the time, and not the clustering algorithm.

4.6 Our algorithm: experiments and results

We ran our algorithm, described in section 2.5 first on the manualle labelled dataset described in section 4.3 and then on full tractographies with thousands of tracts.

To-do/Question: dejar solo 2 scores y poner los demas en apendices?

4.6.1 Experiments on Manually Labelled Data

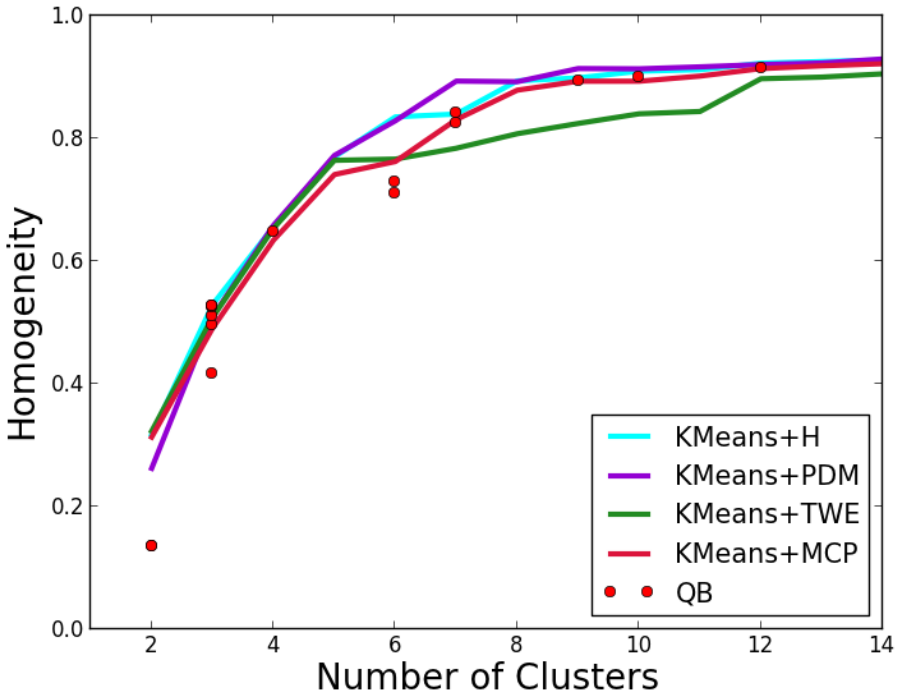
We tested first the algorithms on the manually labelled data described above. As we knew the real clusters beforehand we were able to run the validation criteria that need a reference or *ground truth*: Homogeneity, Completeness, V-Measure, Adjusted Mutual Information, Adjusted Rand Index, Normalized Adjusted Rand Index, Weighted Normalized Adjusted Rand Index, and also the Silhouette Coefficient and the Inertia. 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.

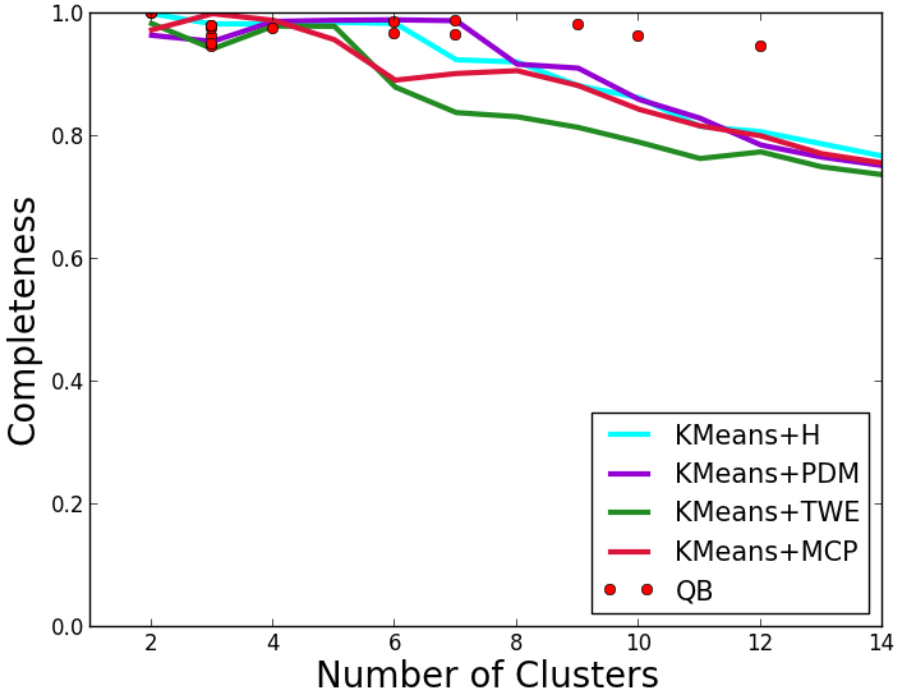
Analysis of the Scores

Below are the plots showing the obtained scores for each of the 5 combinations/techniques: Kmeans + Two-ways Euclidean, Kmeans + Point Density Model (σ = 42mm), Kmeans

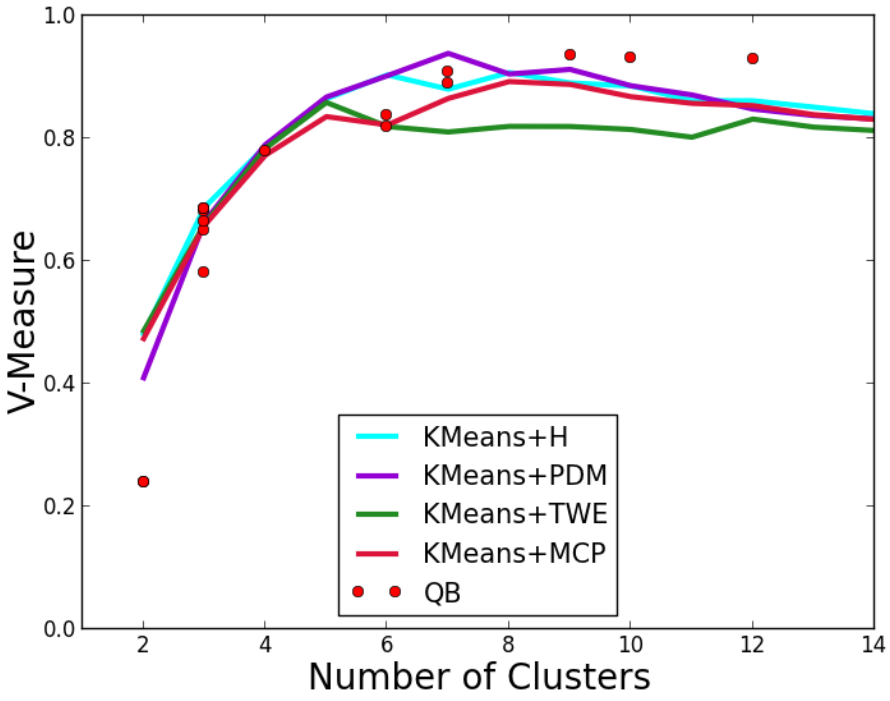
+ Mean Closest Point, Kmeans + Hausdorff, and QuickBundles. An analysis is presented afterward.



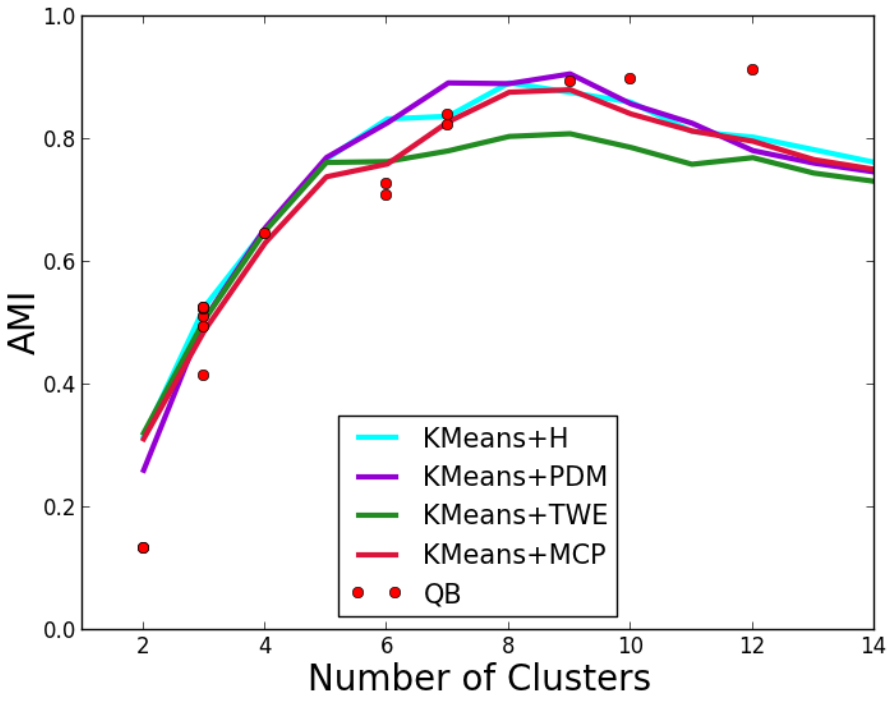
(a) Homogeneity (Correctness)



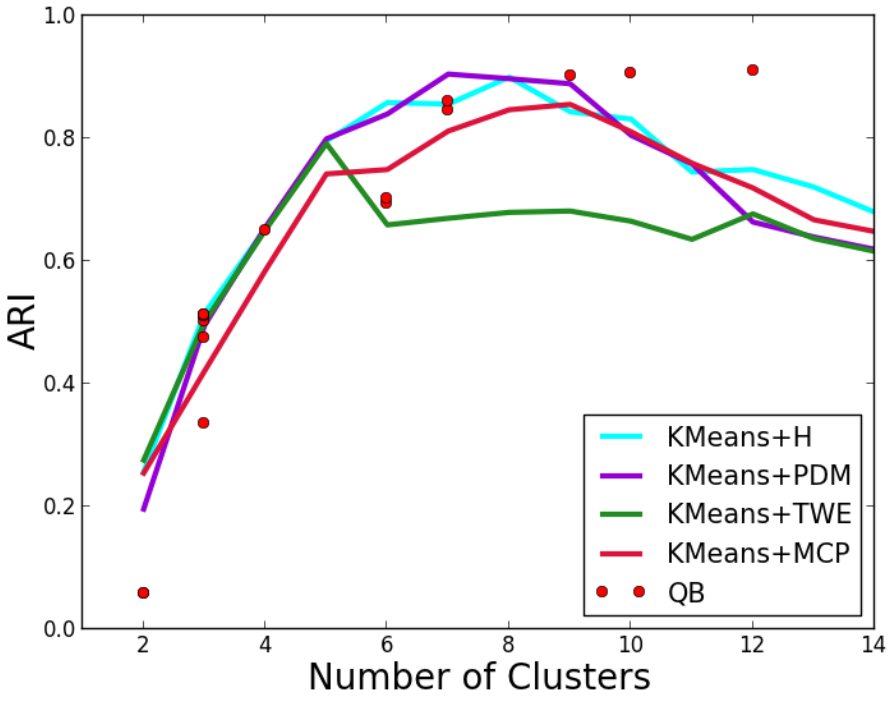
(b) Completeness



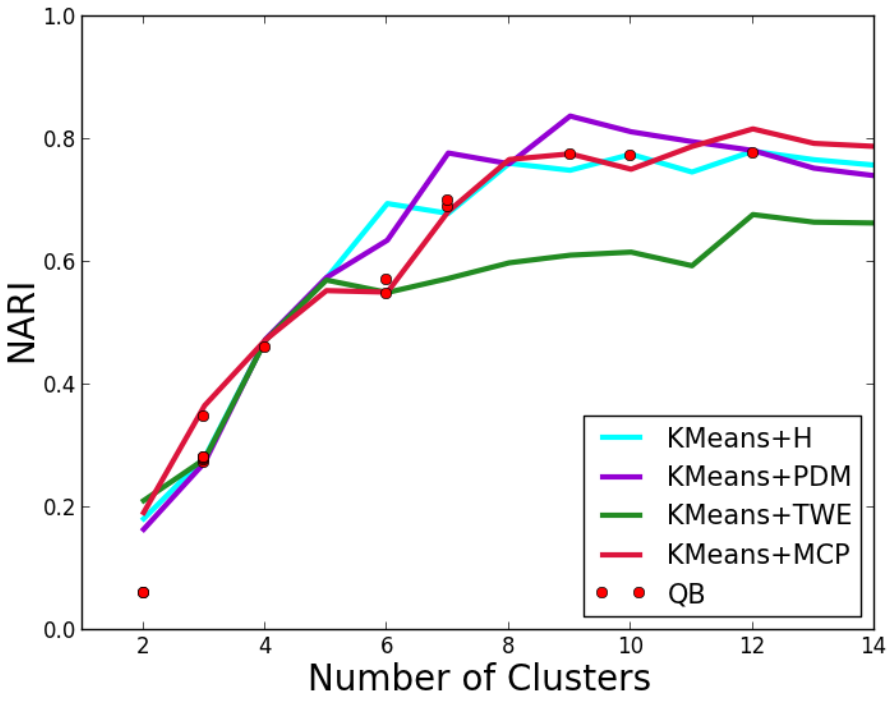
(a) V-Measure



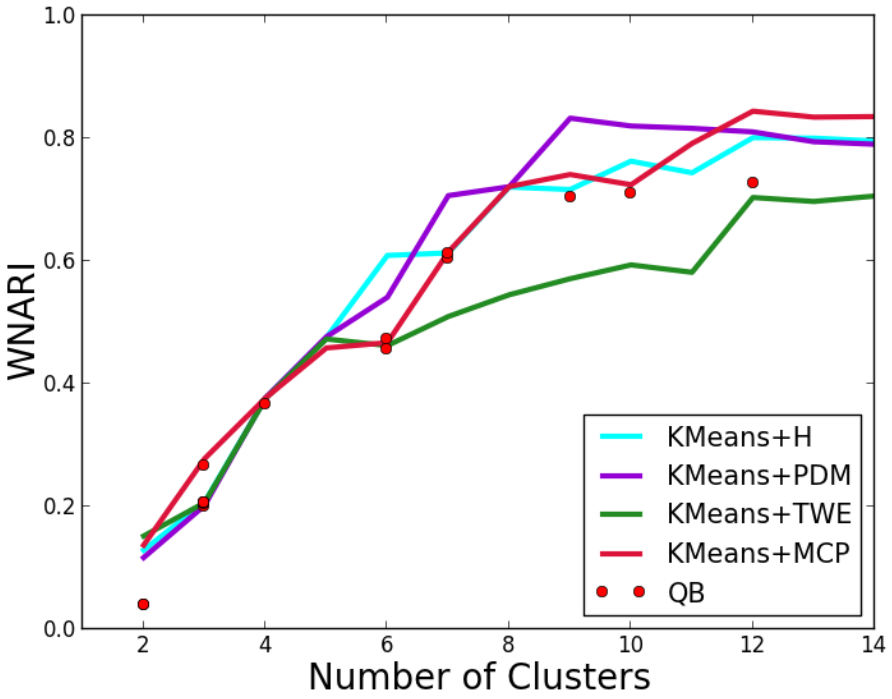
(b) Adjusted Mutual Information



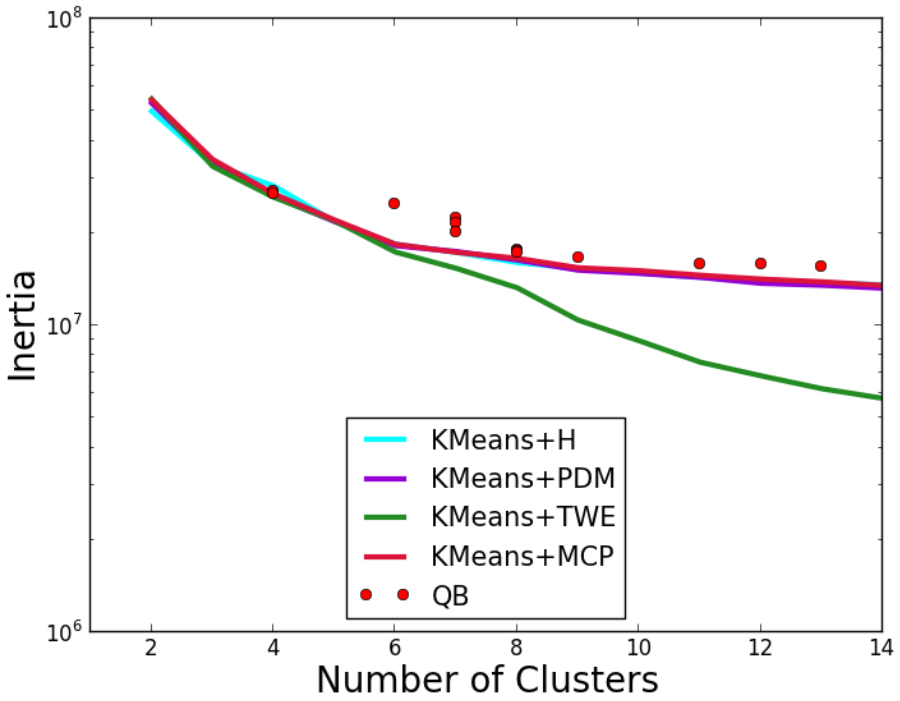
(a) Adjusted Rand Index



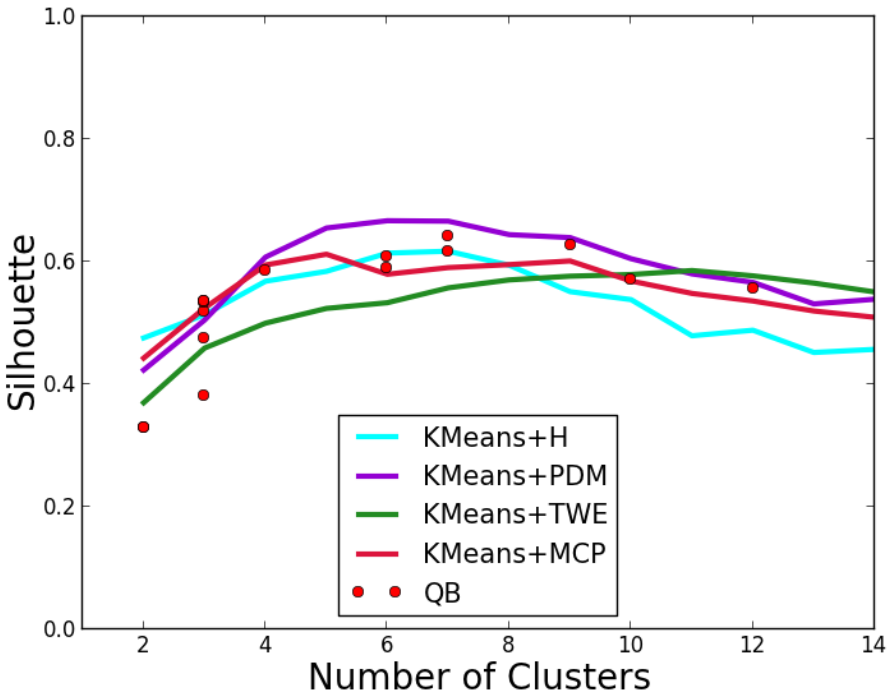
(b) Normalized Adjusted Rand Index



(a) Weighted Normalized Adjusted Rand Index



(b) inertia



(a) Silhouette Coefficient

Fig. 4.5: Scores of the manually labelled data

First one can observe the scores that need a ground truth, i.e. all but Inertia and Silhouette. The closer they are to 1 the more similar the obtained clusters are to our 9 manually labelled bundles, described in section 4.3.

The first three plots show the Correctness (Homogeneity), Completeness and V-Measure. As explained in section C.4, homogeneity penalizes the clustering scheme in which samples from different classes are clustered together and Completeness the ones in which samples from the same class are clustered separately, V-measure is the harmonic mean of both. If we only look at the V-measure we observe that for the desired amount of clusters, between 8 and 10, QuickBundles behaves slightly better. By breaking it 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. As we have explained in section C.6 this produces clusters of a lower quality.

The Adjusted Mutual Information and Adjusted Rand Index present similar graphs: up to 9 clusters KMeans+PDM clearly obtains higher scores but for larger numbers QuickBundles obtains clusters that are more similar to the ground truth. Again, as explained in section C.6 these scores are very general and, while useful, don't necessarily agree with clinicians' opinions.

The Normalized ARI and the Weighted ARI are more adequate for our domain, the former giving all bundles the same weight independently of the amount of tracts they contain and the latter also giving more importance to clusters' correctness over completeness. Their plots show that QuickBundles behaves quite poorly and Kmeans combined with PDM or MCP perform best. This also makes us wonder about the distribution of the tracts among the many clusters: it seems that QuickBundles is yielding some big clusters. We will analyse it later in section 4.6.2.

In order to be able to compare the algorithms and metrics using the inertia criterion we decided to compute it, for all cases, using the Euclidean distance. This is why TWE presents the lowest inertia score, Kmeans aims at minimizing the clusters' inertia computed also using the euclidean distances and TWE is very similar to it. By looking at the plot, we can effectively confirm that QuickBundles' clusters have a higher variance than Kmeans combined with the other metrics.

Lastly, the Silhouette scores puts Kmeans+PDM as the clear winner when below 11 clusters and then TWE shows some advantage.

On average it can be observed that Kmeans with Point Density Model metric obtains good results for all cases. This was expected and reinforced by these experiments, as we believe that this metric better captures the distance between tracts, taking into consideration disaligments and shape dissimilarities.

Analysis of the Clusters

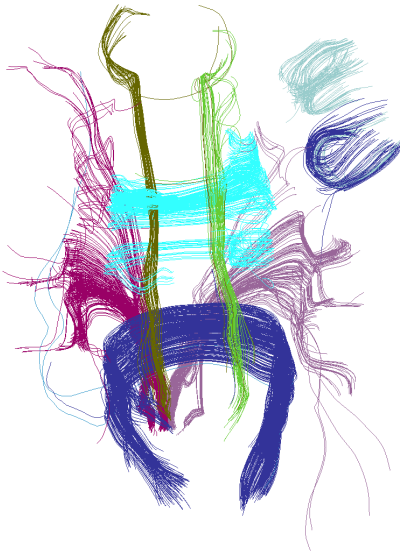
Below we present the clusters recovered by Kmeans when setting the number of cluster to 9 and by QuickBundles when settings the threshold parameter to 29.2mm (produced 9 clusters).

The 2 corpus callosum tract clusters were well identified by every algorithm except for TWE which mixed some other tracts in too. 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

Hausdorff

Two-ways Euclidean

Fig. 4.6: Manually labelled data results: Hausdorff and Two-ways Euclidean



QuickBundles

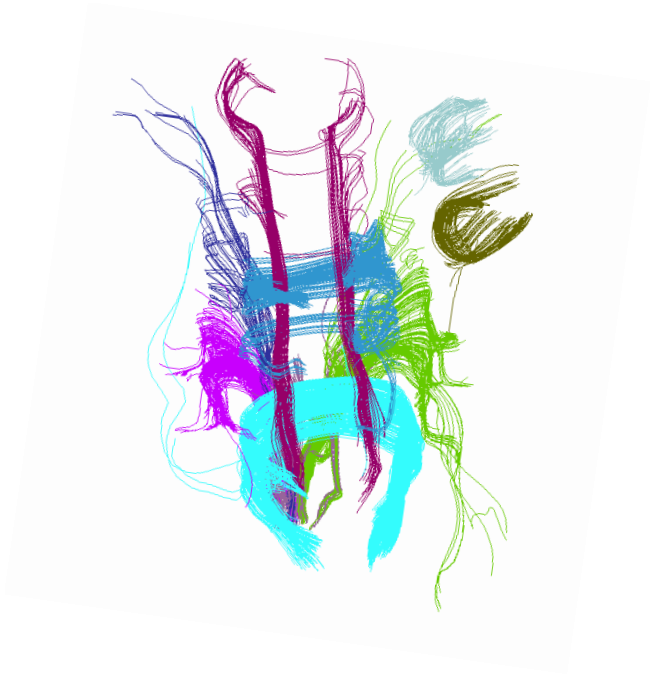
Point Density Model

Fig. 4.7: Manually labeled data results (cont.): QuickBundles and Point Density Model

trac (the two that were in the same hemisphere) and Point Density Model succeded to obtain all the original clusters.

4.6.2 Experiments on Full Tractographies

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 18 to 3200. Additionally we compared their output to QuickBundles (QB) [7]. However,



Mean Closest Point

Fig. 4.8: Manually labeled data results (cont.): Mean Closest Point

as we explained in B.0.7 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 40mm, and selected threshold values based on the number of clusters obtained to run them over the 10 subjects.

On real data, we can only use the fully unsupervised criteria, such as inertia and the silhouette criterion. 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. Results are given in Fig. 4.9 for each of the aforementioned criteria and algorithms.

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 number of clusters (over 3000) curves between QB and KMeans+PDM seem to converge in terms of cluster quality.

The behaviuor of MCP is strange and requires special attention. A simple explanation is the ill-posed optimization: MCP is not a very well-behaved distance and K-means may perform poorly in that case. In other words, K-means is not a well-behaved cluster estimator: it tries to solve a non-convex problem. 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, what it relies on is the distance between the tracts and, as explained in section A.1.2, MCP is not a distance, which yields a bad behaviour overall.

T-test : For 560 clusters, the average Silhouette Score of clusters obtained with K-means+PDM is 0.382087 and the standard deviation 0.00762463833581, for QB the mean is 0.330673 and the standard deviation 0.00933340378468. A T-test is a statistics that cheks if two means are reliably different from each other (and aren't different just be-

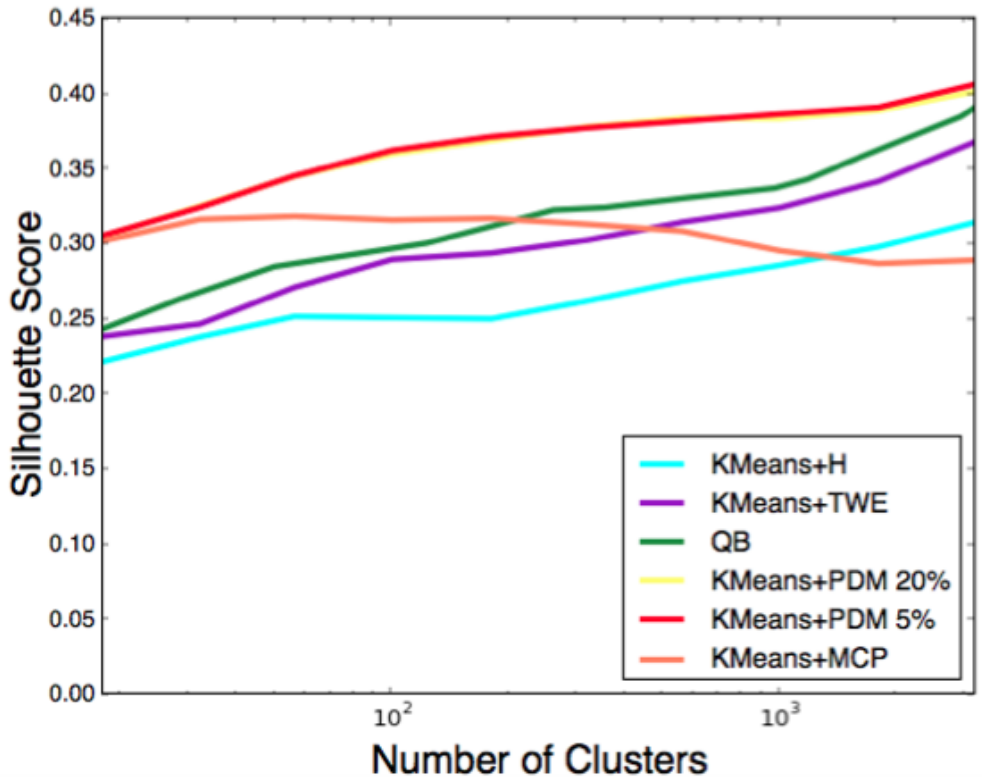


Fig. 4.9: **Silhouette score on real data**: Comparison of KMeans with PDM, TWE, MCP, and Hausdorff metrics, and QuickBundles. Each curve shows the average score of the ten subjects. KMeans+PDM is used with two sizes of the learning set in MDS step.

cause of chance). A paired-samples T-test was used to check whether our experiments are statistically significant: t(9) = 11.77, p = 9.07e-07, no significant difference was found.

Note that a given number of cluster can correspond to strikingly different structures in the data, depending on the algorithm and metric: In Figure 4.10 we show the result of the full brain tract clustering for all algorithms on an arbitrary chosen subject. The number of clusters was set to 560 for 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 on the corpus callosum and the corticospinal tract.

In figure 4.11 we can see the histograms on the clusters sizes. We see that QB has the biggest amount of small clusters, that likely correspond to outlier tracts, and 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 time are sensitive to the number of clusters, however QuickBundles' time complexity is $O(NCk)$ and KMeans+PDM $O(NSk^2+NCk)$, where C is the number of clusters, k the tract resolution and S the sample size. In k-means+PDM, the creation of the partial distance matrix dominates the time complexity as long as $Sk > C$.

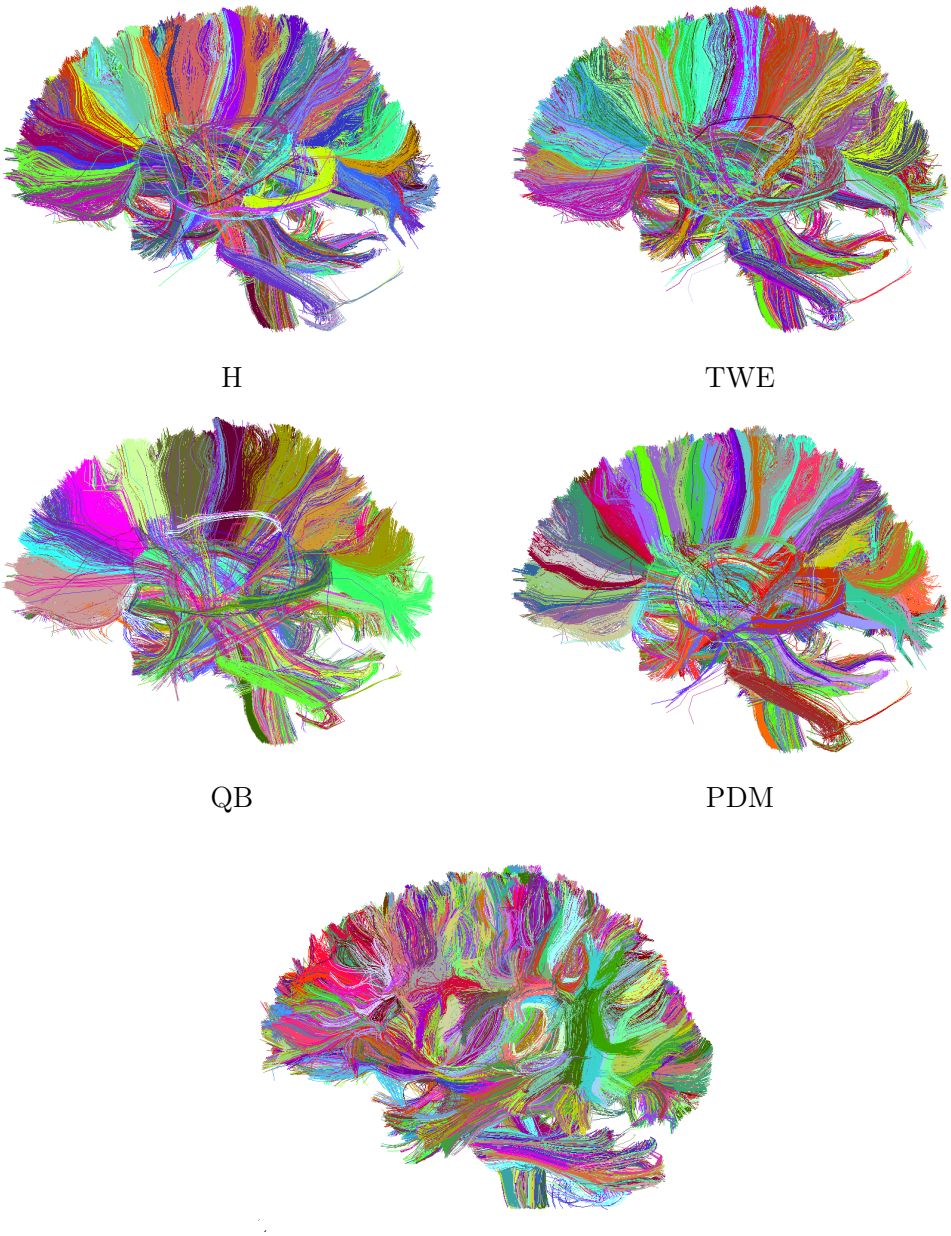


Fig. 4.10: **560 clusters on brain**: Qualitative algorithm comparison for the resulting tract clusters on an arbitrary chosen subject.

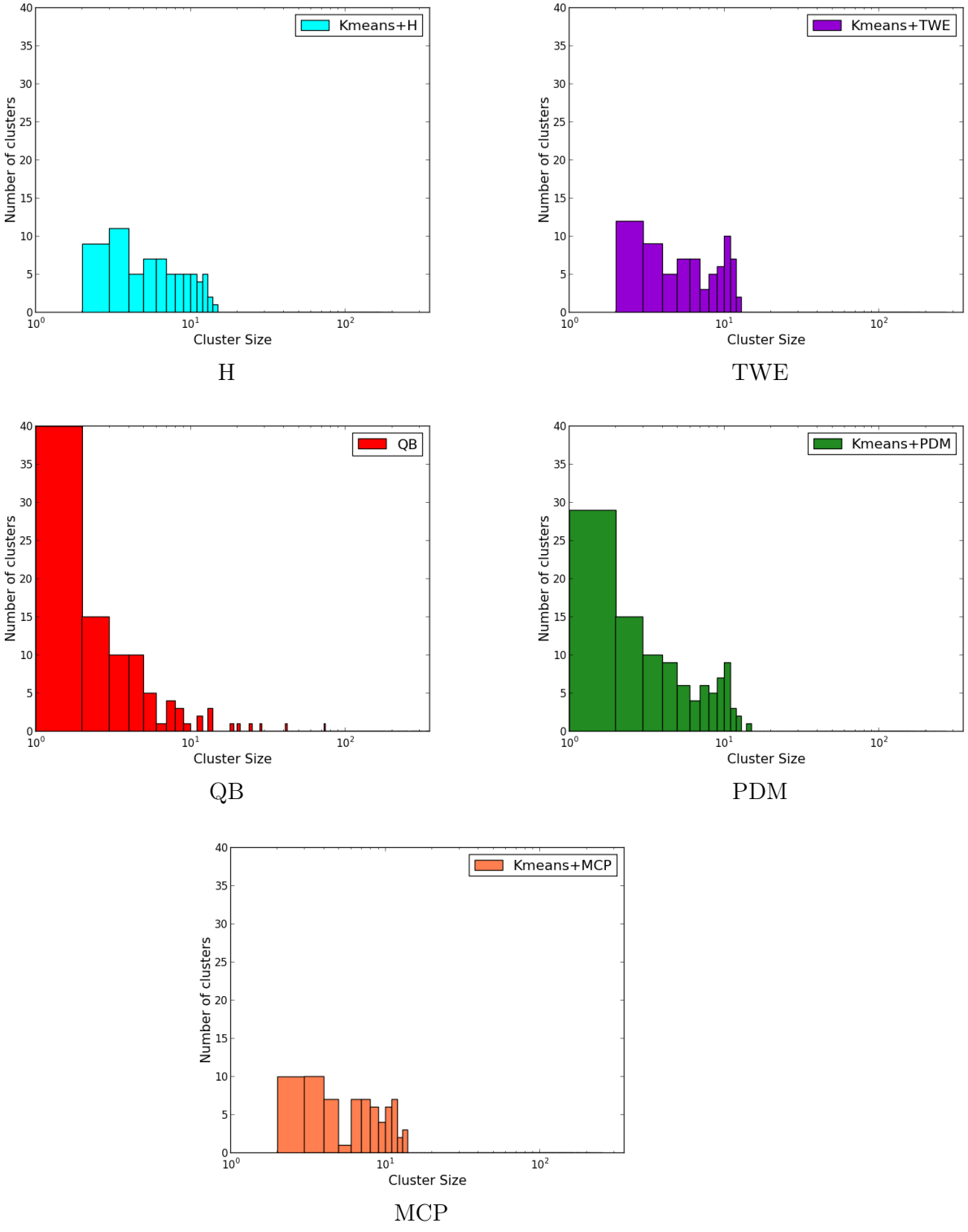


Fig. 4.11: Histogram of the cluster sizes for the different algorithms.

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.

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 clusters to represent effectively the whole set of tracts. Kmeans+PDM has a better compression power, but is less robust against outlier tracts. It clearly outperforms other metrics.

We believe that this method along with a posterior tract registration [25] 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 didn't have the time to explore all the possible options and to develop all the ideas that arose. 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 [32] which seemed to have given good results to Wasserman et al. [32].
- Experiment with **Brain Atlases** to improve the clustering algorithms, a promising option, explored by Pamela Guevara et al [8].
- Measure the **performance of Mini Batch Kmeans** with huge tractographies and millions of tracts, and compare it with QuickBundeles
- Explore reducing the dimension of the set obtained by MDS. What is the impact in cluster quality and running time?
- Explore the impact of **reducing the sample size** below 5%. Does the cluster quality decrease significantly? How much is the speed gain?

7. PUBLICATION

The paper version of this research *A Comparison of Metrics and Algorithms for Fiber Clustering* [24] 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

A.1 Metrics

A.1.1 Hausdorff

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

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.

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

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

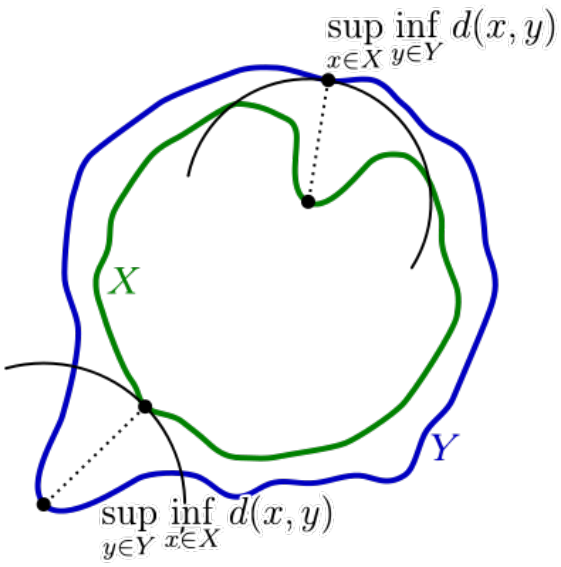


Fig. A.1: Components of the calculation of the Hausdorff distance between the green line X and the blue line Y.

The Hausdorff metric does not require the tracts to have the same number of points. It handles tracts as clouds of points, therefore it does not take advantage of the tracts' direction.

Hausdorff is $\mathcal{O}(n^2)$: quadratic on the number of points per tract.

A.1.2 Mean Closest Point

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.

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

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

A.1.3 Minimum Direct Flip / Two-ways Euclidean Distance

Some sections and images are taken from QuickBundle's paper "QuickBundles, a method for tractography simplification" [7].

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 we evaluate the distance in both directions. Therefore the Two-ways Euclidean 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.

TWE, or Minimum Direct Flip (MDF) as Moberts et al [7] call it, is the distance used by their algorithm QuickBundles (see section B.0.7). They first define $d_{direct}(s, s')$ and $d_{flipped}(s, s')$ that are both Euclidean distances:

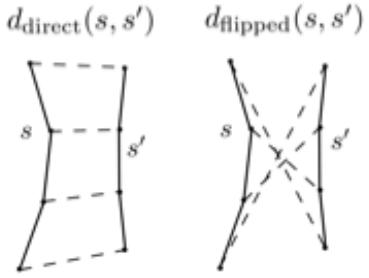


Fig. A.2: Euclidean distances used to calculate MDF.

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

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

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, and that 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.

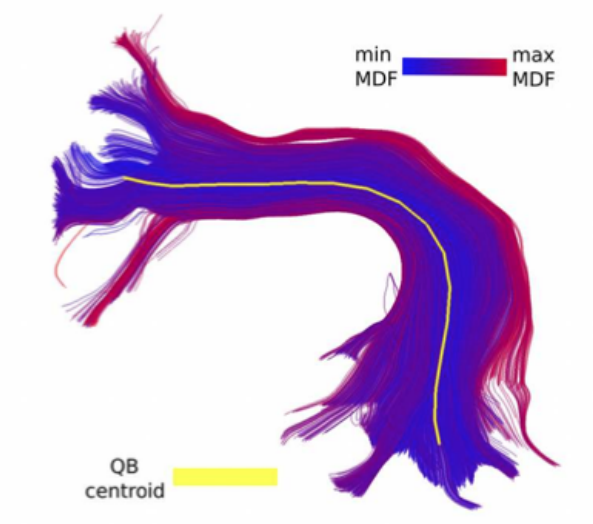


Fig. A.3: MDF distance from all the tracts in the cluster to the cluster centroid.

Appendix B

ALGORITHMS

B.0.4 K-means

From Matteo Matteucci's clustering tutorial: `http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html`.

K-means clustering is a well-known method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, where k is fixed a priori. This results in a partitioning of the data space into Voronoi cells. 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 [12] however the ideas goes back as far as 1956 with Hugo Steinhaus [27]. 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 it simply as K-means.

The procedure follows a clear way to classify a given data set through k clusters fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids shoud 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 k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new assignment of the data set points to the nearest centroid is done. This results in an update/assignment loop. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

The matemathical defitions 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 the n observations into k sets $(k \leq n)$ $S = \{S_1, \ldots, S_k\}$ so as to minimize their *inertia* (the within-cluster sum of squares, see section C.1).

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \mu_i\|^2$$

where μ_i is the cluster centroid, i.e. the mean of points in S_i.

Algorithm

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

1: Initializitation: choose k centroids by drawing randomly k observations among n without replacement
2: **do**
3: Assigment: Assign each observation to its closest cluster
4: Update: Update the centroids of each cluster
5: **while** Centroids moved

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

Assigment 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).

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

where each x_p is assigned to exactly one $S^{(t)}$, 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.

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

When the centroids ne longer move the algorithm has converged.

Analysis & Complexity

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

The complexity of the algorithm is $O(n \times k \times I \times f)$, where n is the number of samples, k is the number of clusters we want, I is the number of iterations and f is the number of features in a particular sample. It can be clearly seen that the algorithm won't scale for huge number of objects to be clustered.

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

B.0.5 Mini Batch K-means

From Scikit-learn's Mini Batch K-means page `http://scikit-learn.org/stable/modules/clustering.html#mini-batch-k-means` and Siddharth Agrawal's machine learning tutorial `http://algorithmicthoughts.wordpress.com/2013/07/26/machine-learning-mini-batch-k-means`

The Mini Batch K-Means [23] is a variant of the K-Means algorithm that uses mini-batches to drastically reduce the computation time, while still attempting to optimise 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 algorithm.

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

Comparison with Kmeans

Mini Batch K-Means converges faster than K-Means, but the quality of the results is reduced. In practice this difference in quality can be quite small, as shown in the following example:

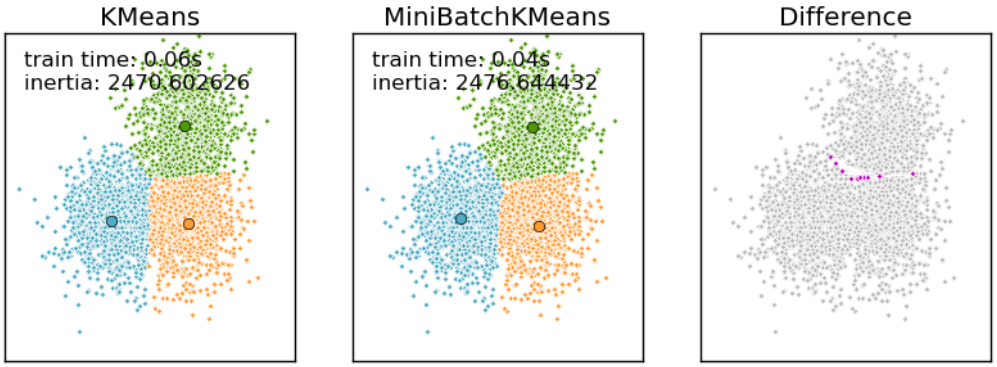


Fig. B.1: K-means vs Mini Batch K-means clustering the same data set

In section C.1 we will introduce the concept of *Inertia* but for now we can simply say that the lower, the better.

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

B.0.6 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 Xiaowei Xu [5]. It is a density-based algorithm because it finds a number of clusters starting from the estimated density distribution of corresponding nodes. DBSCAN along with K-means are ones of the most common clustering algorithms and also most cited in scientific literature.

Explanation

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:

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

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.

Pseudocode

D 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(D, eps, MinPts)
   C = 0
   for each unvisited point P in dataset D
      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)

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

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

Complexity

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 recomputations. This however also needs $\mathcal{O}(n^2)$ memory.

Advantages

- 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.)
- It is designed for use with structures that can accelerate region queries, e.g. using an R^* tree.

Disadvantages

- The quality of DBSCAN depends on the metric used in the function $regionQuery(P, \epsilon)$. The most common distance metric used is the Euclidean distance. Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse of dimensionality", making it difficult to find an appropriate value for ϵ. This effect, however, is also present in any other algorithm based on the Euclidean distance.
- 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.0.7 QuickBundles

From Garydallidis et al. paper "QuickBundles, a method for tractography simplification" [7].

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

In QB each item (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 [12, 27] or BIRCH [34], 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 before (tracts are resampled to k points each). 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 $c = (I, h, n)$ where I is the list of the integer indices $i = 1 \dots N$ of the tracts in that cluster, n is the number of tracts in the cluster, 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:

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

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

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 $v = h/n$
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; $c_e = (I, h, n)$, and update $c_e \leftarrow (append(I, i), h + s, n + 1)$
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.

Complexity

The complexity of QB is in the best case linear time $\mathcal{O}(N)$ with the number of tracts N and worst case $\mathcal{O}(N^2)$ when every cluster contains only one tracts. 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: they only save the sum of current tracts h in the cluster and the sum is cumulative; moreover there is no recalculation of clusters, the streamlines are passed through only once and a tract is assigned to one cluster only.

Appendix C

SCORES

C.1 Inertia

Some sections are 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 on 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 section B.0.4).

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

$$F = \{f_1, \ldots, f_n\}$$

$$f_i = \{p_1, \ldots, p_k\}$$

$$p_i = \langle p_x, p_y, p_z \rangle$$

$$C = \left\{ \frac{\sum_{i=1}^{k} p_1}{k}, \ldots, \frac{\sum_{i=1}^{k} p_k}{k} \right\}$$

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 labelizations 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 especially if the clusters are manifolds with weird shapes: for instance inertia is a useless metric to evaluate clustering algorithm 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 [20] (named after its author William Rand) is a measure of the similarity between two data clusterings. Using the sets F, B and C described above we define:

- $a = |\{(f_i, f_j)/f_i, f_j \in B_k, f_i, f_j \in C_l\}|$
  the number of pairs of elements in F that are in the same set in B and in the same set 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 sets in B and in different sets 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 set in B and in different sets 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 sets in B and in the same set 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 we use a contingency table. 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 [15]. 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 [10, 29]. 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 [10] 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

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

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

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

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

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

When using the AR for tract bundles we realize that it doesn't suit perfectly our needs. Two problems arise: big bundles are more important that smaller bundles and completeness and correctes are weighted equally, while they should not as Moberts et al. [16] explained.

C.4 Correctness (Homogeneity), Completeness and V-measure

From Scikit-learn's section `http://scikit-learn.org/stable/modules/clustering.html#homogeneity-completeness-and-v-measure`

Given the knowledge of the real bundles B, it is possible to define some intuitive metric using conditional entropy analysis.

In particular Rosenberg and Hirschberg (2007) [21] 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 separarely.

Homogeneity and completeness scores are formally given by:

$$correctness = 1 - \frac{H(B|C)}{H(C)}$$

$$completeness = 1 - \frac{H(C|B)}{H(B)}$$

where $H(B|C)$ is the conditional entropy of the classes given the cluster assignments and is given by:

$$H(B|C) = -\sum_{i=1}^{r}\sum_{j=1}^{s}\frac{n_{i,j}}{n} \times \log\frac{n_{i,j}}{n}$$

and $H(B)$ is the entropy of the classes and is given by:

$$H(B) = -\sum_{i=1}^{r}\frac{n_i}{n} \times \log\frac{n_i}{n}$$

The $n_{i,j}$ are values from the contigency table (see C.2.1). Rosenberg and Hirschberg further define V-measure as the harmonic mean of correctness and completeness:

$$V - measure = 2\frac{correctness \times completeness}{correctness + completeness}$$

While correctness and completeness are not symmetrical V-measure is, therefore it can be used to evaluate the agreement of two independent assignments of the same dataset.

All are bounded below by 0.0 and above by 1.0 (higher is better). Correctness = 1 means that no tract is clustered together with any tracts from other bundles.

The previously introduced metrics are not normalized with regards to random labeling: this means that depending on the number of samples, clusters and ground truth classes, a completely random labeling will not always yield the same values for homogeneity, completeness and hence v-measure. In particular random labeling won't yield zero scores especially when the number of clusters is large.

C.5 Normalized Adjusted Rand index

From Moberts el al's paper "Evaluation of fiber clustering methods for diffusion tensor imaging" [16]

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. [16] define a Normalized Adjusted Rand (NAR) index. 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 k and to multiply each entry n_{ij} by a factor $\frac{k}{u_i}$.

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

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

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

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

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

$$NAR = \lim_{k\to\infty} \frac{a' - \frac{m'_1 m'_2}{\binom{rk}{2}}}{\frac{m'_1 + m'_2}{2} - \frac{m'_1 m'_2}{\binom{rk}{2}}} = \frac{2f - 2rg}{2f - rf - r^2}$$

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.6 Weighted Normalized Adjusted Rand Index

From Moberts el al's paper "Evaluation of fiber clustering methods for diffusion tensor imaging" [16]

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:

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

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 WNR to weight both concepts:

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

Finally the adjust the WNR for chance agreement:

$$WNAR = \lim_{k \to \infty} \frac{WNR - E(WNR)}{1 - E(WNR)}$$

where

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

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

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 WNAR is:

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

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

2 clusters that belong to the same bundle (i.e. they should be in the same cluster) it is way easier to us to visually put it together, while it is very difficult to visually separate 1 cluster that contains 2 bundles.

C.7 Mutual Information

From Scikit-learn's page `http://scikit-learn.org/stable/modules/clustering.html#mutual-information-based-scores`.

Mutual Information is a function that measures the agreement of the two assignments, ignoring permutations. In probability theory and information theory, the mutual information of two random variables is a quantity that measures the mutual dependence of the two random variables, i.e. it measures the reduction of the uncertainty (the entropy) of a random variable X given the knowledge of another random variable Y. Informally we say that two variables are independent if the knowledge given by one does not provide information on the other.

Given the two sets B and C as described above, the entropy of either is the amount of uncertaintly for an array, and can be calculated as:

$$H(B) = \sum_{i=1}^{r} P(i) \log(P(i))$$

where $P(i) = \frac{|B_i|}{r}$ is the probability that an object picked at random from B falls into bundle B_i, likewise for C

$$H(C) = \sum_{j=1}^{s} P'(j) \log(P'(j))$$

with $P'(j) = \frac{|C_j|}{s}$. $H(B)$ and $H(C)$ are non-negative and take the value 0 only when there is no uncertainty determining an object's bundle or cluster membership, i.e. when there is only one bundle or cluster. The Mutual Information (MI) between B and C is calculated by:

$$MI(B, C) = \sum_{i=1}^{r} \sum_{j=1}^{s} P''(i, j) \left(\frac{P''(i, j)}{P(i)P'(j)} \right)$$

and now $P''(i, j) = \frac{|B_i \cap C_j|}{n}$ denotes the probability that a point belongs to both the bundle B_i and the cluster C_j. MI is a non-negative quantity upper bounded by the entropies $H(B)$ and $H(C)$. It quantifies the information shared by the two clusterings and thus can be employed as a clustering similarity measure.

C.8 Adjusted Mutual Information

From Scikit-learn's page `http://scikit-learn.org/stable/modules/clustering.html#mutual-information-based-scores`.

Adjusted Mutual Information (AMI) is an adjustement of the Mutual Information (MI) score to account for chance. It accounts for the fact that the MI is generally higher for two

clusterings with a larger number of clusters, regardless of whether there is actually more information shared.

As explained above, a statistic S that is corrected for chance is:

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

With $Max(S)$ is the upper-limit of S, and $E(S)$ is the expected value of S. The expected value for the mutual information can be calculated by adopting a hypergeometric model of randomness using the following equation from Vinh, Epps, and Bailey (2009) [29]:

$$E[\mathrm{MI}(B,C)] = \sum_{i=1}^{r}\sum_{j=1}^{s}\sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i,b_j)} \frac{n_{ij}}{N}\log\left(\frac{N.n_{ij}}{a_i b_j}\right) \times \frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}$$

where $(a_i + b_j - N)^+$ denotes $\max(0, a_i + b_j, N)$. The variables a_i and b_j are partial sums of the contingency table:

$$a_i = \sum_{j=1}^{s} n_{ij}$$

$$b_j = \sum_{i=1}^{r} n_{ij}$$

The adjusted measure [29] for the mutual information may then be defined to be:

$$AMI(B,C) = \frac{MI(B,C) - E[MI(B,C)]}{max(H(B),H(C)) - E[MI(B,C)]}$$

The AMI takes a value of 1 when the two partitions are identical and 0 when the MI between two partitions equals to that expected by chance.

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] C. Fowlkes, S. Belongie, Fan Chung, and J. Malik. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):214–225, 2004.

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

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

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

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases, 1996.

[35] 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.