Content:

Section I: The clustering problem

In *unsupervised learning*[1] one wants to obtain (or "learn") a function or an underlying law describing the dataset without having been previously exposed to labelled examples. The idea is the one of being capable of spotting general patterns directly into the raw data. Related techniques make for instance use of concepts like similarity within the data, in which case we talk about *clustering*[1], or try to discover latent variables within the input set by projecting data from a higher to a lower dimensional space, known as *dimensionality reduction*.

The problem of clustering has several extremely relevant practical applications, e.g. in image segmentation and compression[1], but its theoretical study is notoriously hampered by the lack of the very same formal definition of what a cluster is[2]. Historically, the problem was first addresses in the 50s with the introduction of an objective cost function, considering the sum of the squared Euclidean distances between each point and its respective cluster centre[3]. These centre locations would then be iteratively reassigned in order to minimize the cost function. The need to average over these k intracluster set of distances gave the name *k-mean* (KM) to the algorithm. It is customary to group under the name of *partitional* all algorithms making use of an objective function optimization[2].
Since the locus of points equidistant from two points (e.g. two cluster centres) is a straight line, one of the problems related with the KM algorithm is that it works only for linearly separable datasets. A mapping of each of the N datapoints $\mathbf{x}_i$ via $\mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$ into a *feature space* where the mapped points might be linearly separable was therefore introduced. The approach, called *Kernel k-mean* (KKM), owes its name to the fact that quadratic distances in feature space can be computed via the sole knowledge of the N^2 ij-pairwise products $K_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, with K_{ij} being the Kernel matrix.
Two major limitations are related with the KM and KKM algorithms: on one hand, number and initial centres of the clusters have to be specified[3], while on the other hand the choice of a Kernel matrix still defines a shape for the detectable clusters. In order to solve the problem of initial centres assignment, and therefore make the minimization of the objective function a deterministic process, the authors in Ref. 3 introduced the *Global Kernel k-mean* (GKKM) algorithm. In GKKM, each point in the dataset is progressively considered as a candidate cluster centre by evaluating the KKM clustering error (CE) resulting from the cost function. The one clustering solution with k clusters and the lowest CE, out of the total N possible, is then considered as the optimal one. The GKKM algorithm is deterministic, as the solution for k clusters is built progressively starting from the $k-1$ one. Although GKKM solves cluster initialization, it still requires to input the total number of clusters and it still makes use of a specific kernel; furthermore, the algorithm complexity is kN times larger than KKM[3]. Importantly, methods have been devised in order to reduce the GKKM complexity. A very effective one runs GKKM only over a set of $P \ll N$ initial cluster centres candidates or *exemplars*. These candidates are selected via likelihood maximization of a gaussian-like mixture[1] in feature space, with the P exemplars being the best estimates for the mixture centres. The resulting algorithm, called *Global Kernel k-mean with Convex Mixture Models*[3] (GKKM-CMM), reduces the complexity by a factor P/N with respect to GKKM.

Even when provided with the GKKM-CMM algorithm, the user still has to specify the total number of clusters and the shape of a detectable cluster is still dependent on the Kernel choice. In order to overcome these limitations, the authors in Ref. 2 propose a scheme for cluster detection that aims at starting from the very same definition of what a cluster is. The inclusion (exclusion) of two points $\mathbf{x}_i, \mathbf{x}_j$ to (from) the same cluster is described by the ij-weights of a connectivity matrix w_{ij} being 1 (or 0). Such weights are *adaptive*, meaning that they get updated by progressively looking at bigger length scales in the vicinity of $\mathbf{x}_i$ and $\mathbf{x}_j$ and get changed to zero ($w_{ij} = 0$) is the number of points in between the pair is below a threshold value. The threshold is controlled by an hyperparameter λ for the model, the only free parameter of the analysis. The algorithm gets the name of *Adaptive Weights Clustering* (AWC), with the iterative search technique called *propagation-separation*[2].

Section II: Overview of the *clustering* package

The *clustering* Python package available at fcasola/Clustering-Algorithms contains implementations of 4 partitional: the KM, KKM, GKKM, GKKM-CMM algorithms and one propagation-separation: the AWC algorithm. Whereas a technical guide to the use of the package is present in the README file, an extensive testing of all the implemented algorithms is present in the analysis folder, in the form of interactive jupyter notebook.

IV: Bibliography and References:

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer Science (2006).
[2] K. Efimov and L. Adamyan, *Adaptive Nonparametric Clustering*, arXiv:1709.09102v1 (2017).
[3] G. F. Tzortzis and A. C. Likas, *The Global Kernel k-Means Algorithm for Clustering in Feature Space*, IEEE Trans. on Neur. Net. **20**, 7 (2009).