ClusterinClusters

> Welcome to use `Clusterin'Clusters` -code! It is a program for Structure selection by k-means clustering algorithm

Structure Selection

In configurational sampling procedure structure selection is used to reduce the amount of calculation for the next level of theory -step. This can be achieved simply by discarding some structures from the process. `ClusterinClusters` uses k-means clustering for determining which structures are similar to each other. Similarity measure is Euclidian distance by default. The structures are represented as descriptors generated by the `DScribe` -library.

Usage:

Standalone:

1. Make input data in same format than the example files (or edit the `read_data()` -function in `dataio.py` to fit your data)
2. Edit the parameter input in `input4Clustering.csv` . See below for further parameter details.
3. Run `StructureSelection.py`
4. See selected structures in SelectedXXX.csv

In JKCS

1. Run JKCS up to part 4.
2. *Run `JKCS5_filter` with some option?*

Input parameters

The settings for `StructureSelection.py` can be edited by editing the values in `input4Clustering.csv` .

parameter	value	name	description
`n`	1	`n_jobs`	"Provide the number of parallel jobs"
`k`	15	`n_clusters_init`	"Provide the number of initial clusters (k) for k-means"
`c`	3	`n_clusters_out`	"Provide the number of clusters selected from k-means"
`m`	4	`n_molecules`	"Number of molecules in the given system"
`r`	True	`sampling`	"Whether to select a random sample from the best n_clusters_out"
`s`	20	`n_structures_out`	"Provide the number of structures = local minima outputted"
`l`	DFT	`level`	"[XTB]/[DFT] Whether to perform selection on XTB or DFT structures"
`e`	True	`normEd`	"Normalise energies and convert to kcal/mol"
`v`	True	`verbose`	"Whether to print out some progress"
`d`	MBTR	`descname`	"[CM]/[MBTR]/[SOAP] Which descriptor to use"
`pd`	True	`plotDescs`	"Whether to plot Descriptors"

parameter	value	name	description
`pc`	True	`plotClustering`	"Whether to visualize clustering results. Uses hierarchical clustering and rather greedy t-SNE"

Main program

The main program performs all the functionalities of ClusterinClusters by calling the functions from the modules. It reads in the values `input4Clustering.csv` and passes them on to the functions as parameters.

Outputs

The user can choose verbose on/off in `input4Clustering.csv` parameter `v` by providing `True` / `False` .

Plots

The program creates a folder " `plots` " for visual representation of descriptors and clustering results. User can choose not to plot descriptors or clustering with the parameters `pd` and `pc` respectively.

Clustering results are visualised using t-SNE and plotted both in 2D and in interactive 3D plot (`.html`) that can be opened in a browser.
Note: t-SNE is a rather greedy algorithm so if there's thousands of structures or time is of the essence then `pc` = `False` is recommended.

Selected Structures

The structures are selected from the clustering results. The code calculates mean energies for each k-means cluster and prefers the clusters with smallest mean energies. Parameter `c` defines how many of $k$ clusters are selected. If random sampling is not used, all structures that belong to the selected clusters are outputted. Random sampling takes a sample of structures from the selected clusters and outputs them. Parameter `r` defines whether random sampling is used and parameter `s` defines the size of the sample *ie.* the number of structures outputted. Random sampling is recommended when saving time and calculation resources is of interest.

The list of selected structures is outputted as `selectedXXX.csv` .

Modules

`dataio.py`

This module is used for reading in the data.

function	description
:-------------.	:-------------
makedir()	This function is used inside `init_files()` to create a folder if it does not exist.
init_files()	This function creates a folder for plots inside the `wrkdir`
read_data()	This function reads in the data from `JKCS` output. It needs the `level` parameter defined correctly.
read_xyz()	This function reads the xyz structures from `JKCS` output. It outputs the structures as a DataFrame of ASE objects.
get_structure()	This function is used to choose one structure from ASE structures in order to get more relevant info from the structure. It can be used in conjunction with the `struct2img()` or `plotDescs()` functions to visualise chosen structure or its descriptors.

`descriptors.py`

This module is used for transforming the structures from xyz-input to a descriptor. In the beginning the descriptor hyperparameters are defined and can be tuned if necessary.

function	description
setupDescs()	This function needs the DataFrame of structures represented as ASE objects and it outputs the structures represented with the descriptors in a DataFrame.
plotDescs()	This function has the same input as `setupDescs()` and it sets up the descriptors in a way suitable for plotting. The function can save the plots and show them. It will not run if parameter `pd` is set to `False`.

`selection.py`

This module is used for structure selection functionalities.

function	description
calcKmeans()	Runs k-means algorithm on structures. Returns a cluster labels for each structure as an array.
calcEAvg()	Calculates an average energies for all clusters. Returns cluster labels and their respective anerage energies as a DataFrame.
getBestClusters()	Returns the structures that belong to the best clusters. Makes a random sample if requested by parameter `r`

`visualize.py`

This module is used for visualisation purposes.

function	description
makeDend()	Cluster structures with hierarchical clustering and either save the dendrogram plot or show only.
makeTsne_2D()	Visualise k-means results in 2D. Needs cluster labels as parameters. Can either save the plot or show only.
plotTsneE_3D()	Visualise k-means results with energy as z-axis. Uses plotly to make interactive plot. Can either show the plot or save it as html that can be opened later in browser
struct2img()	Draw an 2D image of an ASE object. A lousy way of visualising the structures.

`own_colormap.py`

This module is used for defining the colors used in plotting.

function	description
own_cmap()	A function that defines colors to be used in plotting. Colors are chosen from Matplotlib library and can be edited. The function returns a list of colornames with length of `n_clusters_init`
visualise_colors()	This function provides a way of visualising the colors with a barplot. Gets number of colors as a parameter.