**MDC PARAMETERS**

**pca**: String. The options to apply or not, a Principal Component Analysis (PCA) on the data before it is fed into the MDC training. (Default: *“auto”*)

“*auto*”: Use of 2 principal components.

“*elbow*”: MDC automatically chooses the optimal no. of principal components based on the elbow rule on the normalized PCA explained variance plot. The selection is made based on the elbow point of 45 degrees to the x axis.

“*none*”: No use of PCA.

Tips

* Applying PCA hugely improves training time.

Advanced

* With the “*auto*” parameter value, the user can also manually select, if they so choose to, the no. of principal components, by changing the “2” in the source code (line ~83 in pca.py) with the desired number.
* The user can plot the PCA explained variance plot through the source code and obtain more visual information (lines ~27-28 in pca.py).

**neighbors**: Boolean. The option to also update or not the weights of the neighbors of the BMU neuron. (Default: *True*)

*True*: Updates the weights of the neighbors of the BMU neuron.

*False*: Only the BMU weights will be updated.

**epochs**: Positive integer. The number of epochs of the neural network training. It determines how many times the whole dataset will be seen by the network. (Default: *5000*)

Tips

* Literature suggests at least 500 iterations for every neuron [Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, *21*(1-3), 1-6].

**lr**: Positive float. The value of the initial learning rate of the neural network training. It adjusts how much the neuron weights will be altered to mimic the data points. (Default: *0.3*)

**n\_neurons:** The number of neurons of the neural network. It determines the number of clusters. (Default: -1)

Positive (>=2) integer: User specified number of neurons.

-1: Automatic selection of the no. of neurons based on the elbow rule on the normalized Sum of Squared Errors (SSE) plot. The selection is made based on the elbow point of 45 degrees to the x axis.

The number of neurons up to which the training will run is determined by the “max\_n\_neurons” parameter.

Advanced

* The user can plot the SSE plot through the source code and obtain more visual information (lines ~57-58 in auto\_neuron\_number\_selection.py).

**max\_n\_neurons:** Positive (>2) integer.The number of neurons up to which the automatic neuron selection will run. (Default: 8)

Tips

* This value cannot be greater than the number of sample class labels in the dataset.

**neuron\_init**: The neuron initialization technique. (Default: “points”)

“random”: The neuron weights are initialized randomly from a uniform distribution over [0, 1).

“points”: Every neuron is initialized as a randomly selected existing data point.

No same data point can be selected for two neurons.

By selecting “points”, MDC randomly selects data points equal to the no. of neurons and calculates their average in-between Euclidean distance. The combination with the highest average distance is selected as the chosen data points to initialize the neurons. This way, MDC initializes the neurons by trying to utilize the largest possible span in the time-related, high dimensionality space. The number of different combinations to be calculated is dictated by the “depth” parameter.

Tips

* Due to randomness, a good practice is to redo the training a couple of times for whichever neuron initialization technique.

**depth:** The number of different combinations for MDC to calculate the Euclidean distance dispersion in order to initialize the neurons. (Default: 10000)

Positive integer: The number of different combinations.

“auto”: It automatically selects and calculates every possible combination.

Tips

* Very high values (> 500000) can result in very high running times. This depends on the user’s hardware and the number of neurons.
* Caution is needed when high values of this parameter along with n\_neurons = -1 is used.
* A higher value results in a higher chance that the algorithm will choose the optimal data points for neuron initialization.

Advanced

* Being able to select and calculate the dispersion of every possible combination of data points, means the exact same initialization for every time MDC is used. This eliminates the stochastic nature of neuron initialization and results in better consistency.
* The number of possible combinations without repetitions of a given dataset with samples and number of neurons is  . For a given dataset, if this number does not exceed the computational threshold depending on the hardware (e.g., 500000), it is advised to use the “auto” parameter value.

**t1**: Positive integer. Constant value, which controls the exponential decrease of the learning rate. (Default: *epochs / 2*)

Tips

* A good practice is to solve the *exp* decrease function (equation 2 in “Parts of the MDC Algorithm”) for different *t1* and *i* (current iteration) values and assess the value of the learning rate at that iteration.

**t2**: Positive integer. Constant value, which controls the exponential decrease of the neighborhood function. (Default: *epochs*)

Tips

* Higher *t2* corresponds to higher and more lasting cooperation between neurons. Solving the neighborhood function (equation 3 in “Parts of the MDC Algorithm”) can give insight to the constant’s desired value.

**verbose**: Positive integer. Determines the way the output will be presented to the user. (Default: 2)

0: No console output.

1: Prints out a list of the cluster labels for all data points.

2: Prints out the clusters along with every label member.

**random\_state**: Positive integer (<=10000). Determines the random state of the algorithm. Choosing a value eliminates the stochastic nature of the “points” neuron initialization and chooses the same combinations every time MDC is run as a whole. (Default: a random value each time MDC is run)