**Supplementary material for the manuscript**

# Data Expressiveness and Its Use in Data-centric AI

Parichit Sharma, Hasan Kurban, M.M. Dalkilic

Computer Science Department
The Luddy School of Informatics, Computing, and Engineering Indiana University, Bloomington
Luddy Hall (700 N. Woodlawn Ave) 3056
Email: parishar@iu.edu

# INDEX

## 1. Purpose of the document

This document serves as the supplementary text for the manuscript – '**Data Expressiveness and Its Use in** Data-centric AI'. It is submitted as additional information accompanying the primary article and contains the information that could not be described due to constraints on page limit. For the reviewers, the document provides additional information on data and preprocessing, specific details of experiments and how to reproduce the results, should the reviewer choose to.

## 2. Experimental details

Experiments are designed to isolate and observe the impact of specific properties for example by only varying the data size (without changing the number of dimensions or number of clusters) and observing the effect on execution time, number of iterations, and classification accuracy. This exercise not only demonstrates the broad range of effectiveness, but also illustrates the otherwise complex relationship between the run-time behavior of iterative learning algorithms and various properties of big data.

Each experiment is repeated three times and the results present the averaged values. In order to minimize variations due to the choice of initial centroids, same initialization points are used across all the algorithms for a given execution. This ensures that the experiments are safeguarded against differences in the initialization protocol etc. Section 6 provides more details about the system specifications.

## 3. Note on arbitrary number of clusters

Our implementation of EM-DC, EM* and EM-T allows for partitioning data into any arbitrary number of blocks. While it is straightforward to compare the training time and number of iterations, classification accuracy is calculated by labelling the predicted clusters as one of the original cluster labels. When the true number of blocks is known, for any arbitrary number of blocks (obtained by partitioning the data), the predicted blocks are assigned the class labels based on the Euclidean distance (a popular choice of metric) between the true and predicted centroids. The Adjusted Rand Index (ARI) [1] (library link) is then used to

compare the classification accuracy by using the known ground truth and the predicted labels.

In our experiments – data was not divided into training or test sets because we wanted to determine the classification accuracy on entire data set thus used whole data for clustering. However, in principal it's also possible to use a test data for prediction and our published CRAN package DCEM (web – implementation of EM* algorithm in R) provides the option to predict on test data (data not seen by the algorithm).

## 4. Dataset and preprocessing

We used the cropland mapping dataset originally published in [1]. The data combines features from two sources – optical readings from a satellite and Unmanned Aerial Vehicle Synthetic Aperture Radar (UAVSAR) system (Radar). There are 325,834 records in 175 dimensions and 7 classes. The classification task is to predict the label (i.e., type of crop) for a given data point. The data is freely accessible from the UCI machine learning repository [2]. Download the data from (download data link), unzip the file and place the 'dataset' folder under the project repo (see section 5 to know how to access the project repo.)

### 4.1 Preprocessing

The original data is highly skewed as almost 98% of the records in the data belongs to 5 classes (where the class labels are 1,3,4,5,6), whereas the remaining classes i.e. (class labels 2 and 7) only accounts for 2% of the records. Hence, to prevent skewness from impacting the clustering accuracy - experiments are done after removing the records belonging to minority classes leaving us with ~321,000 records in the data. The original data was further processed to create data sets for each of the experiment as described below.

**Experiments on large number of clusters:** To observe the effect of increasing the number of clusters on overall performance (first plot in Figure-1.png in the manuscript) – we created 'crop.csv' (found under folder ***clustering_data*** in the project's GIT repo) that has ~321,000 records in 10 dimensions. PCA on the original data revealed that most of the variance is captured by the first 20-25 PCs. Since we wanted to isolate the impact of large number of clusters - so we decided to use the first 10 principal components, as using many more

dimensions would have made it difficult to isolate the differences in the performance to number of clusters or number of dimensions, whereas using too few dimensions would have rendered the clustering task trivial.

**Experiments on large number of dimensions:** This experiment (second plot from top in Figure-1.png in the manuscript) compares the relative performance when the number of dimensions are increased. We gradually increased the number of principal components to create different datasets with 10, 20, 50, 80 and 100 dimensions (available under **dimensionality_data** folder in project's GIT repo). For this experiment, we fixed the number of clusters as 10 and used all the datapoints i.e. 321,000. The different datasets are appended with the number of dimensions in that file for example crop_10.csv has 10 dimensions and crop_100.csv has 100 dimensions.

**Experiments on large data size:** For this experiment (plot-2 from top in Figure-1.png in the manuscript), we created different sub datasets by selecting pre-defined number of records from the original data i.e. (100,000, 200,000, … , 321,000 records). While selecting the records from raw data, we selected roughly the same number of records from each class to ensure a fair representation of all the classes in the subset data. The files for this data are appended with the size for example, **crop_100.csv contains 100, 000** data points. This data can be accessed under the ***scalability_data*** folder in project's GIT repo.

Note: The creation of data does not use random seeds.

## 5. How to replicate the experiment results

Since running all the experiments can take long period of time (also depend on factors such as other resource intensive tasks running on the system, available resources – memory etc.) - please take note of the approximate time limits for running each experiment in case you choose to re-run only specific experiment.

| S.no | Experiment type | Time |
|---|---|---|
| 1 | Clustering experiments | 3-4 hours |
| 2 | Dimensionality experiments | 4-5 hours |
| 3. | Scalability experiments | 2-3 hours |

Follow the steps below to re-run the experiments and generate the plots.

0) Download the dataset from data-download. Unzip the file and move it under the project root folder i.e., EM-DC-NEURIPS_2021/datasets

1) Clone the project's GIT repo from (https://github.com/parichit/EM-DC-NEURIPS_2021). Please ensure that you have the datasets folder present inside the project's root i.e., EM-DC-NEURIPS_2021/datasets

2) Make sure to change the directory to **benchmarking_scripts** folder inside the project folder.

3) Although it's not mandatory but keep a safe copy for reference purposes of the current output folders i.e. benchmark_clus, benchmark_dims, benchmark_scal folders. Re-running the scripts will create new output folders by the same name (if old folders are found they will be automatically copied by a different name with timestamp in the project directory).

4) You can also run the following commands directly but it's better to run them as background task (i.e., use nohup) to prevent job killing due to system shutdown or internet connection issues (if you are connected to the server via terminal)

   **4.1 Clustering Experiments**

   ```
   nohup python3 benchmark_clusters.py >
   output_clus.log 2>&1 &
   ```

   output files will be under EM-DC-NEURIPS_2021/benchmark_clus

   **4.2 Dimensionality experiments**

   ```
   nohup python3 benchmark_dimensions.py >
   output_dims.log 2>&1 &
   ```

output files will be under EM-DC-NEURIPS_2021/benchmark_dims

**4.3 Scalability experiments**

```
nohup python3 benchmark_scal.py > output_scal.log
2>&1 &
```

output files will be under EM-DC-NEURIPS_2021/benchmark_scal

5) **Generating the plot** – After the runs are complete – you can check that by ensuring that all output files i.e., emdc_res.txt, emstar_res.txt and emt_res.txt are present in the output folders (you should expect these files in each of the folder corresponding to the experiment you ran). Change the directory to project folder and run the below command.

```
Rscript plotting.R
```

output can be found under EM-DC-NEURIPS_2021/Figure-1.png

## 6. System Details

All experiments were done on a dedicated SICE (School of Informatics, Computing and Engineering) server Silo. Silo is a 64-bit system running the Ubuntu operating system. It has 24 AMD cores with 512 GB of main memory and 32GB of virtual memory. The version of python is 3.8.7.

## 7. Additional User Manual

As a proof of concept - our existing implementation of EM* [4] (that use a partial order data structure) can be downloaded from CRAN. The package user manual and vignettes can be accessed at [15] and [16] respectively. The user manual contains comprehensive information on each function call and their parameters as supported by the package. It also lists useful code samples to illustrate the

functionality of each function call. The Vignettes act as a quick start guide to help users in getting started with the package.

**References**

[1] Hubert, L., Arabie, P. Comparing partitions. Journal of Classification 2, 193–218 (1985). https://doi.org/10.1007/BF01908075

[2] Iman Khosravi, Abdolreza Safari & Saeid Homayouni (2018) MSMD: maximum separability and minimum dependency feature selection for cropland classification from optical and radar data, International Journal of Remote Sensing, 39:8, 2159-2176, DOI: 10.1080/01431161.2018.1425564

[3] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[4] Sharma Parichit, Kurban Hasan, Jenne Mark and Dalkilic Mehmet (2020). DCEM: Clustering Big Data using Expectation Maximization Star (EM*) Algorithm. R package version 2.0.4. https://CRAN.R-project.org/package=DCEM

[5] https://cran.r-project.org/web/packages/DCEM/DCEM.pdf

[6] https://cran.r-project.org/web/packages/DCEM/vignettes/DCEM.html

**[End of Document]**