Homework 7: Dimensionality Reduction

Fundamentals of Data Science 2016/2017

The goal is to write a tool that applies Achlioptas' dimensionality reduction algorithm to a (large) input dataset. You should write and submit

- a library named `libID.py`
- a script named `ID.py`

where `ID` is your student ID, or if you do not have one, your last name without accents.

Note: the script should take at most 100s when invoked as in the example reported below (see the last page); after this time it will be automatically killed.

1 `libID.py`

It should implement the following methods:

`alldist(X)`

Return the Euclidean distance matrix between all pairs of rows in the NumPy array `X`, as an n-by-n NumPy array where n is the number of rows of `X`.

`achmat(D, d)`

Generate the D-by-d random matrix used by Achlioptas' algorithm. The i-th column of the matrix specifies the weights with which the coordinates of a D-dimensional vector are combined to produce the i-th entry of its d-dimensional transform. Each entry is chosen independently and uniformly at random in $\{-1, 1\}$.

`reduce(X, d)`

Perform dimensionality reduction on a set of points. Takes in input a set of n points in $\mathbb{R}^D$, in the form of an n-by-D matrix X, and reduces each point to have dimensionality d. It returns the n-by-d NumPy array containing the n reduced points as rows. Recall that the linear map that reduces point $\mathbf{x}$ is $f(\mathbf{x}) = d^{-0.5}\,\mathbf{x}^T\mathbf{A}$, where $\mathbf{A}$ is the matrix obtained with `achmat()`.

`distortion(dm1, dm2)`

Compute the distance distortion between all pairs of points, according to the distances in the matrices `dm1` and `dm2`. Recall that the distance distortion between point i and point j is simply `dm2(i,j)/dm1(i,j)`. The result is a NumPy array of length $n(n-1)/2$ containing, for each pair of points (i, j) with $1 \leq i < j \leq n$, the ratio between their distance according to `dm1` and their distance according to `dm2`. (Note that the distance between a point and itself is not considered).

2 `ID.py`

It should perform the following operations. Receive a filename and a list of positive integers $d_1, \ldots, d_k$ from the command line,

```
$ python ID.py data.csv d1 d2 ... dk
```

where `data.csv` has the format of `ratings.csv`, which is abailable on the course website; read into a `pandas.DataFrame`, it shall look like this:

```
   userId  movieId  rating   timestamp
0       1        2     3.5  1112486027
1       1       29     3.5  1112484676
2       1       32     3.5  1112484819
3       1       47     3.5  1112484727
4       1       50     3.5  1112484580
```

Pivot the dataset (see `pandas.DataFrame.pivot()`) to employ the values of the first column as index, the values of the second column as columns, and the values of the third column as values (note that the columns might have any name):

```
movieId  1       2       3       4       5       6       7       8       \
userId
1           NaN     3.5     NaN     NaN     NaN     NaN     NaN     NaN
2           NaN     NaN     4.0     NaN     NaN     NaN     NaN     NaN
3           4.0     NaN     NaN     NaN     NaN     NaN     NaN     NaN
4           NaN     NaN     NaN     NaN     NaN     3.0     NaN     NaN
5           NaN     3.0     NaN     NaN     NaN     NaN     NaN     NaN
```

Replace each `NaN` value with the average value of the column it belongs to. For each d in $d_1, d_2, \ldots, d_k$, use `reduce()` to reduce the dimensionality of the rows of the dataframe to dimension d, obtaining k different reduced datasets $R_1, \ldots, R_k$.

Then, from the original dataset and from each reduced dataset $R_1, \ldots, R_k$, sample a subset of $\min\{250, n\}$ random points, where n is the number of points in each dataset. **Note**: the same entries must be selected in all sets, i.e. if the i-th point is selected from the original set then the i-th point is also selected from each reduced set $R_1, \ldots, R_k$. **Hint:** look at the methods of `pandas.DataFrame`. Then, use `alldist()` to compute the all-to-all distance matrices on each subset of points, and apply `distortion()` to get the array of distance distortions (the ratio between each distance after reduction and the original one).

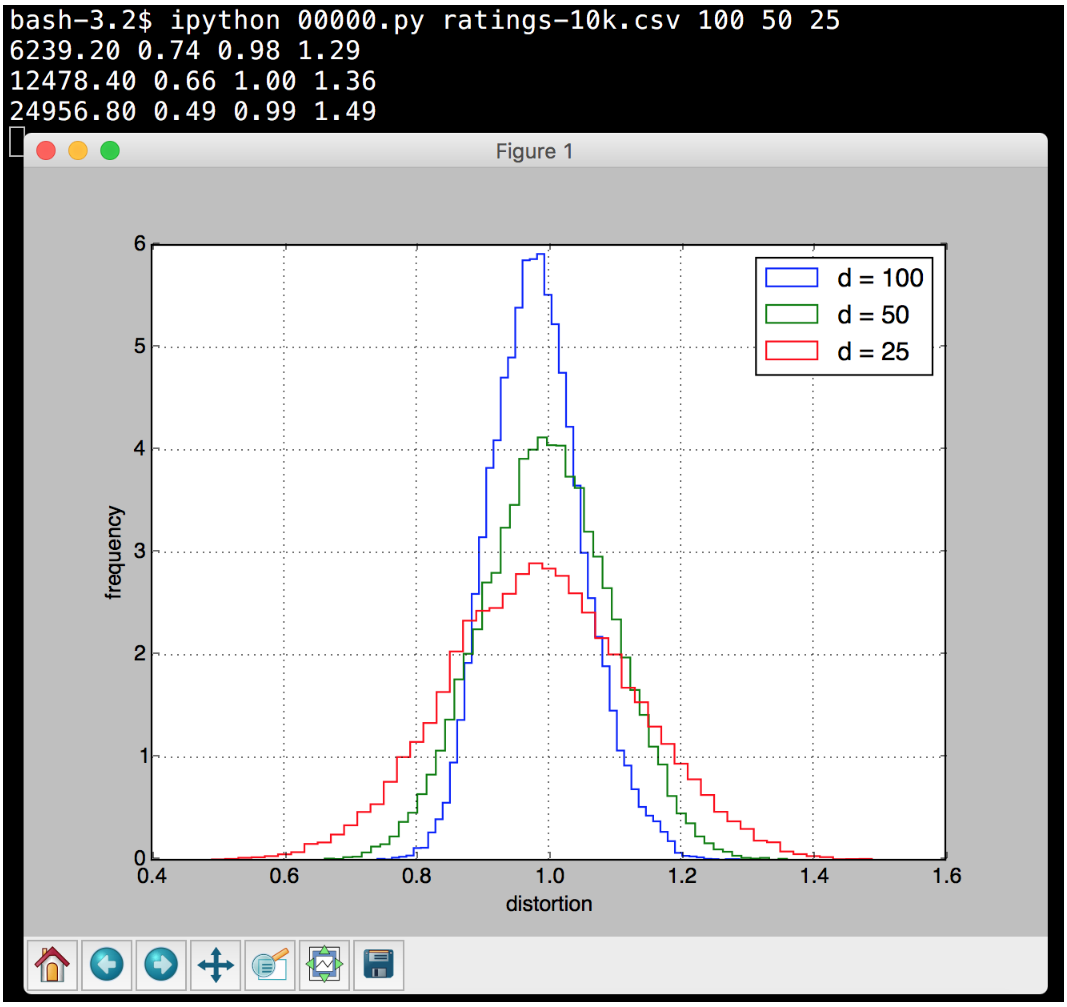
The script should then:

1. for each $j = 1, \dots, k$, print to screen on a single line: the ratio between the bytes taken by the original data and those taken by R_j, as well as the minimum, the average, and the maximum distance distortion obtained on the subset sampled from R_j. Hints: `pandas.DataFrame.memory_usage()`, `numpy.ndarray.nbytes`. Use only 2 digits after the decimal separator.

2. plot, on a single window, the histogram of the distortion frequency distribution for each d_j, $j = 1, \dots, k$. **Be sure that the window stays put on the screen until manually closed**, see e.g. the option `block=True`.

Example

Using the dataset `ratings-10k.csv`:



```
bash-3.2$ ipython 00000.py ratings-10k.csv 100 50 25
6239.20 0.74 0.98 1.29
12478.40 0.66 1.00 1.36
24956.80 0.49 0.99 1.49
```

Note the impressive reduction in the memory footprint – around 4 orders of magnitude – traded for just a little distortion.