NICOLAUS COPERNICUS UNIVERSITY IN TORUŃ
Faculty of Physics, Astronomy and Informatics

Programming in Python

Final Project

Adam Linek

Reducing data dimensionality and projecting it onto two-dimensional space

August 2, 2020

1 Introduction

Often the result of an experiment is a high-dimensional data tensor, but only part of it is relevant. Knowledge of the phenomena occurring in the experiment usually allows to reduce the dimensionality of the data and simplify further analysis. However sometimes these phenomena are unknown and a conscious reduction of data dimensionality is not possible. For this purpose, machine learning is used, which by analyzing the provided multidimensional data allows to reduce the number of dimensions using various algorithms. The aim of the project is to check whether the selected data reduction algorithm can achieve effects comparable to those achieved through conscious reduction based on the experience and knowledge of the experimenter. Therefore, a Python script has been prepared which using the t-SNE algorithm reduces multidimensional data downloaded from the public FTP server of the Free University of Berlin. Then, using the matplotlib package, the data reduced to two dimensions were plotted and compared with the graph in the project description.

2 Script description

Script uses five different packages.

- mdshare - a package which provides a python-based downloader for molecular dynamics (MD) data from a public FTP server at FU Berlin. The "alanine-dipeptide-3x250ns-heavy-atom-distances.npz" data is used in the script.
- numpy - a package which adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- matplotlib - a plotting package for the Python programming language and its numerical mathematics extension NumPy.
- argparse - the package used to prepare interface for using the script with command-line mode.
- t-SNE - a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.

There are three functions defined in the script. First of them `loadData()` downloads a data from from FTP server or loads it from file if already downloaded and then returns it as ndarray. The `fit(X)` function reduces dimensionality of the given ndarray using t-SNE package. It is done by function `TSNE()` provided with the package. This function takes several arguments, but the only parameter required is the number of dimensions to which data is to be reduced. In this case, the function returns a two-dimensional array. This data is then scaled to be in the range from $-\pi$ to π and returned by `fit(X)` function. The last function `visualize(Y)` takes a two-dimensional array and then plots it on a two-dimensional square scatter graph. The execution part of the script involves downloading data from an FTP server via the `loadData()` function, then reducing its dimension to two using the `fit(X)` function and finally plotting it on a 2D graph with `visualize(Y)` (Figure 1).

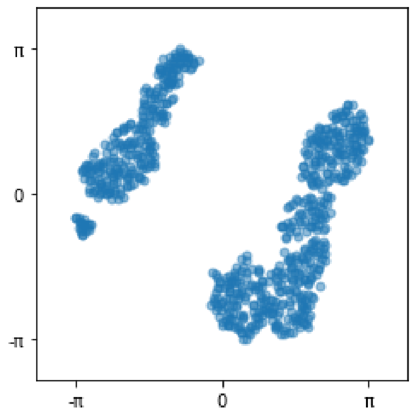
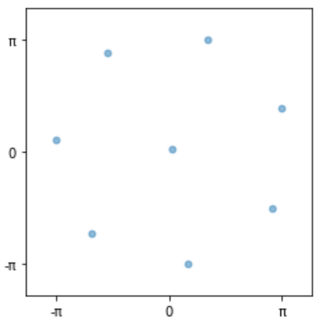
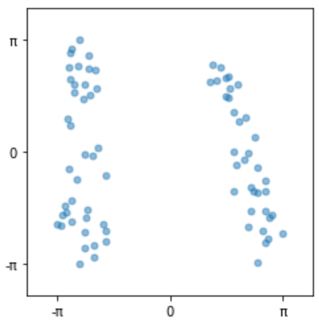


Figure 1: Example of the plotted data projected onto 2-dim space.

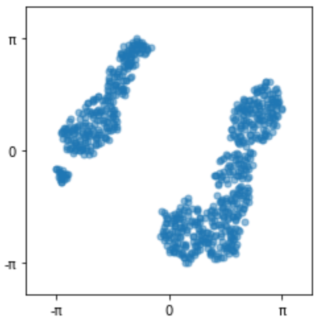
The script uses "alanine-dipeptide-3x250ns-heavy-atom-distances.npz" data, which are very large. It turns out to be important to introduce a step with which these data will be included in the reducing algorithm. It was necessary to determine the optimal step value in order to balance the number of data to be taken into account, which should be as large as possible, and the calculation time, which should be as short as possible. To this purpose, several graphs for different step values were generated and compared with each other. The graphs obtained for values above 1000 (Figure 2a & 2b) have too few points. A step set at 1000 (Figure 2c) makes the density of points on the graph seem to be sufficient, but nevertheless it was decided that the default value of the step is 500 (Figure 2d), which in effect certainly gives a graph with the appropriate number of points. It is evident that further decrease of the step (Figure 2e, 2f & 2g) does not significantly affect the graph, but significantly extends the calculation time.



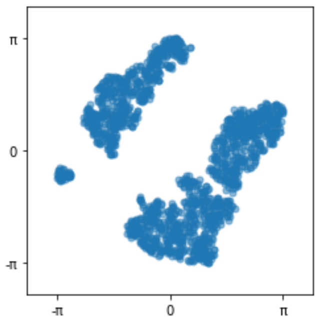
(a) Step set to 100000.



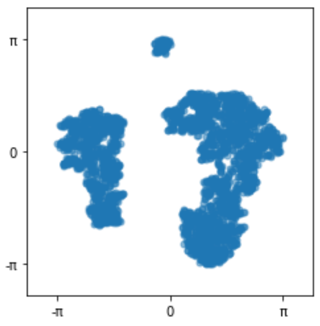
(b) Step set to 10000.



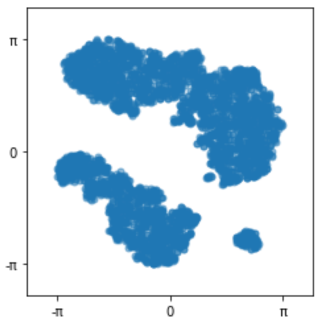
(c) Step set to 1000.



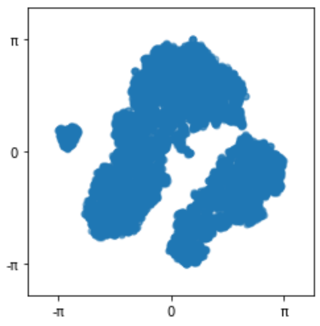
(d) Step set to 500.



(e) Step set to 300.



(f) Step set to 200.



(g) Step set to 100.

Figure 2: Comparison of the impact of the step value on data reduction.

In the case of a high density of points in the plot it is difficult to determine their highest concentration. Therefore, the user can call the `visualize(Y)` function in an extended mode which, in addition to generating a scatter plot, will superimpose a dense grid on it and, based on the calculated distance of each data point from the grid nodes, will determine the concentration of points and present it using a colorful contourf plot. However, calculating the distance of each data point from each grid node is an extremely time consuming process, so this option is not enabled by default. An example of a generated graph is shown in Figure (3).

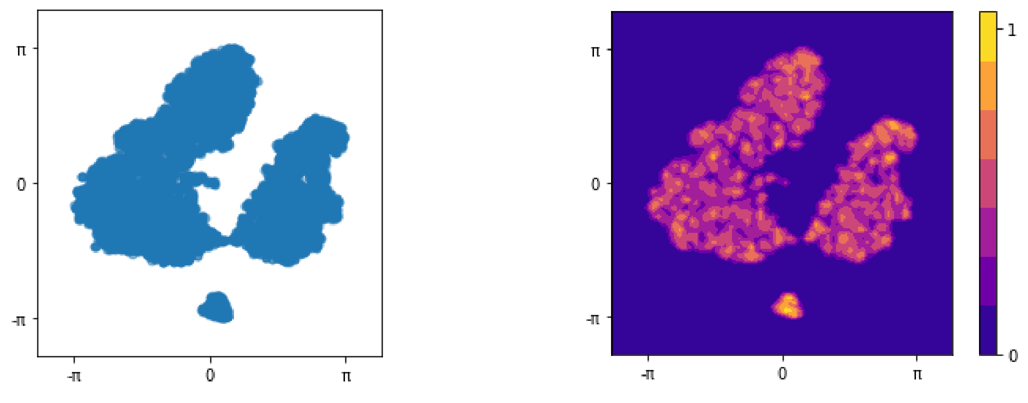


Figure 3: a) Plotted data projected onto 2-dim space with the step set to 100. b) Generated plot with normalized density estimation based on plot a).

3 Command-line mode

The script is supposed to work in the command-line mode. For this purpose, an argparse package was used to prepare an interface with which the user can call a program with different parameters. In the command line, one can use the `/cd` command to go to the folder containing the script. To run the program with default values, one can use the `/python script.py` command. However one can use the `/python script.py -h` command which will call the help describing all the parameters taken by the script. In particular there are

- `-s` (`-step`) - int - set the step of the samples. Higher number means more samples will be skiped (500 by default).
- `-p` (`-perplexity`) - float - the perplexity is related to the number of nearest neighbors that is used in other manifold learning algorithms. Larger datasets usually require a larger perplexity. Consider selecting a value between 5 and 50. Different values can result in significantly different results (30. by default).
- `-e` (`-early_exaggeration`) - float - controls how tight natural clusters in the original space are in the embedded space and how much space will be between them. For larger values, the space between natural clusters will be larger in the embedded space. Again, the choice of this parameter is not very critical. If the cost function increases during initial optimization, the early exaggeration factor or the learning rate might be too high (12. by default).
- `-l` (`-learning_rate`) - float - the learning rate for t-SNE is usually in the range [10.0, 1000.0]. If the learning rate is too high, the data may look like a 'ball' with any

point approximately equidistant from its nearest neighbours. If the learning rate is too low, most points may look compressed in a dense cloud with few outliers. If the cost function gets stuck in a bad local minimum increasing the learning rate may help (200. by default).

- `-n` (`-n_iter`) - int - maximum number of iterations for the optimization. Should be at least 250 (1000 by default).
- `-v` (`-verbose`) - int - verbosity level (0 by default).
- `-a` (`-angle`) - float - the trade-off between speed and accuracy for Barnes-Hut T-SNE. 'angle' is the angular size of a distant node as measured from a point. If this size is below 'angle' then it is used as a summary node of all points contained within it. This method is not very sensitive to changes in this parameter in the range of 0.2 - 0.8. Angle less than 0.2 has quickly increasing computation time and angle greater 0.8 has quickly increasing error (0.5 by default).
- `-c` (`-contour`) - bool - if set to True it will generate contour plot in addition to scatter plot. Warning: May take some time if step is small (False by default).

  With this knowledge one can use the command together with proper parameters and their values to use the program in desired way. For instance, to use a program with `step` set to 1000, `learning_rate` set to 100 and `n_iter` set to 2000 one should use `python script.py -s 1000 -l 100 -n 2000` command.

4 Summary

The aim of the project was to investigate whether data projection carried out using machine learning can replace the experimenter's experience and knowledge of the experimental system. Figure (4) shows a two-dimensional projection of high-dimensional data, which was prepared thanks to the knowledge of the phenomena occurring in the experiment.

Comparing this picture with figure (5) showing the projection of the same data but obtained using machine learning and the t-SNE algorithm, it is evident that these graphics differ significantly. While in both cases it is visible that the data projection is concentrated around certain areas, these areas do not overlap between the images. As a result, it can be argued that machine learning and an algorithm at this level of advancement are extremely useful, but still cannot fully replace the experimenter's experience and knowledge of phenomena occurring in the experimental system.

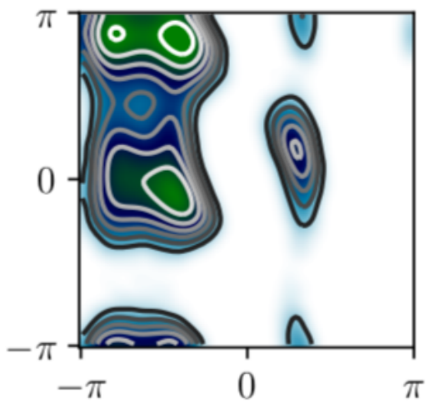


Figure 4: Example of the plotted data projected onto 2-dim space.

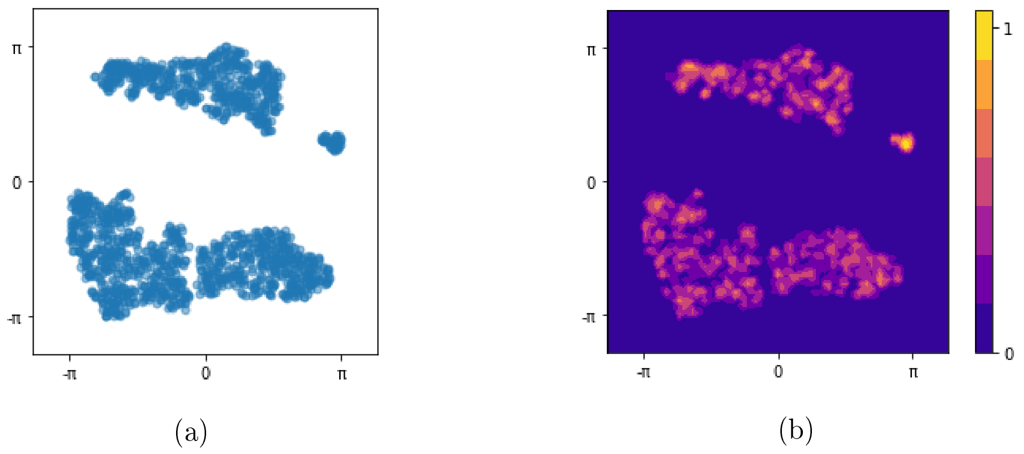


Figure 5: a) Plotted data projected onto 2-dim space with the step set to 500. b) Generated plot with normalized density estimation based on plot a).