Code

To run the submitted code, use the `run.py` file as such:

```
$ python run.py small 1
```

Here, `small` is the name of the data directory, and `1` is the level of verbosity. Currently verbosity may be set either on or off (0 or 1).

Introduction

The following relates to the practical trial on January 15, 2018. The applicant (Josh) was sent a directory of code and data relating to an implementation of the AX discrimination score. This writeup is to accompany the code modified by the applicant.

What I did

- **Variable Renaming** The code was modified such that items `A`, `B`, and `X` are evident. The exception to this is the internal variables of the added function `get_DTW()`, which uses `X` and `Y` so as to be agnostic to input.
- **Code Correction** The `if` statement in the choosing of item `X` was modified to avoid picking item `A` again.
- **Improve Structure of Code** The repeated `DTW` calculation used to estimate the ABX measure was double hard-coded. This modified code makes the calculation a stand-alone function.
- **Document Code** Commenting was used to break the code into logical parts, such as data loading, and DTW calculation.
- **Profile Code** Time stamps are used to show length of time used by I/O or calculations.

What I did not do

Optimize Code for Time Calculating DTW with Numpy would surely be faster than mutliple nested forloops, but I did not manage to implement it.

Comments

The major problems I see with this code relate to scaling to larger files (i.e. more dimensions per item), and more files. Having more files will tax the program on I/O time, and more dimensions will tax the distance calculations. To solve the former problem, the data can be saved on disc in a format best for reading and writing, modified for Python (i.e. there may be better formats than HDF5). As for computation, implementing everything with linear algebra libraries would be ideal.