Journal of
open research software

Gaspar, H A 2018 ugtm: A Python Package for Data Modeling and Visualization Using Generative Topographic Mapping. *Journal of Open Research Software,* 6: 26. DOI: https://doi.org/10.5334/jors.235

SOFTWARE METAPAPER

ugtm: A Python Package for Data Modeling and Visualization Using Generative Topographic Mapping

Héléna Alexandra Gaspar[1,2]

[1] Social, Genetic and Developmental Psychiatry (SGDP) Centre, Institute of Psychiatry, Psychology and Neuroscience, King's College London, UK

[2] National Institute for Health Research Biomedical Research Centre, South London and Maudsley National Health Service Trust, UK

hgaspar.chemoinfo@gmail.com

ugtm is a Python package that implements generative topographic mapping (GTM), a dimensionality reduction algorithm by Bishop, Svensén and Williams. Because of its probabilistic framework, GTM can also be used to build classification and regression models, and is an attractive alternative to t-distributed neighbour embedding (t-SNE) or other non-linear dimensionality reduction methods. The package is compatible with scikit-learn, and includes a GTM transformer (eGTM), a GTM classifier (eGTC) and a GTM regressor (eGTR). The input and output of these functions are numpy arrays. The package implements supplementary functions for GTM visualization and kernel GTM (kGTM). The code is under MIT license and available on GitHub (https://github.com/hagax8/ugtm). For installation instructions and documentation, cf. https://ugtm.readthedocs.io.

Keywords: generative topographic mapping; dimensionality reduction; machine learning; data visualization; data analysis; regression; classification

Funding statement: HG acknowledges funding from the US National Institute of Mental Health (PGC3: U01 MH109528).

(1) Overview

Introduction

ugtm (v2.0) is a package for multidimensional space analysis based on the generative topographic mapping (GTM). A complete documentation with API reference and tutorials is available online (https://ugtm.readthedocs.io). GTM is a non-linear manifold-based dimensionality reduction method introduced by Bishop et al [1]. The ugtm package contains an implementation of GTM, and also kernel GTM (kGTM), the kernel version of the algorithm introduced by Olier et al [2].

GTM maps are similar to self-organizing maps [3] but provide a probabilistic framework that can be used to "color" the map and generate class maps or landscapes. These colored maps can then be used to build regression and classification models. GTM regression (GTR) [4] and GTM classification (GTC) [5] algorithms are implemented in ugtm. Considering that scikit-learn [6] is now widely used for machine learning tasks, ugtm provides scikit-learn-compatible classes for data transformation (the eGTM transformer), classification (eGTC classifier), and regression (eGTR regressor).

Other implementations of the core algorithm of GTM are available online. The netlab package [7] implemented in Matlab was the first implementation with published source code. GTMapTool, a software written in Free Pascal, is available as a web application on the website of the Laboratoire de Chémoinformatique in Strasbourg (http://infochim.u-strasbg.fr/mobyle-cgi/portal.py#forms::gtmaptool). ugtm provides a Python implementation of the core GTM algorithm similar to both netlab and GTMapTool, and also includes predictive modelling frameworks for classification and regression compatible with scikit-learn. The ugtm code, open to collaboration, is freely available on GitHub (https://github.com/hagax8/ugtm) under MIT license.

Implementation and architecture

(1) Architecture

ugtm v2.0 is a package implemented in pure Python. The API reference is accessible online (https://ugtm.readthedocs.io/en/latest/api.html). The main modules and their relationships are described in **Figure 1**. The eGTM, eGTC and eGTR classes implemented in ugtm_sklearn

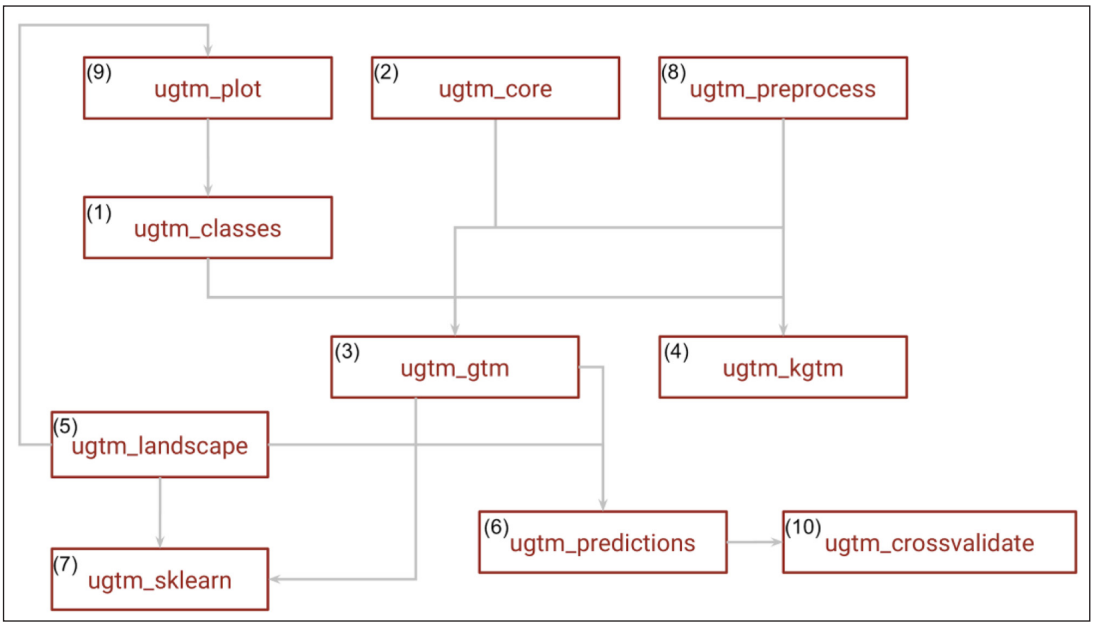

Figure 1: Graph of ugtm v2.0 modules: (1) ugtm_classes: classes for generative topographic mapping (GTM) models, (2) ugtm_core: kernel GTM (kGTM) and GTM core functions, (3) ugtm_gtm: expectation-maximization algorithm for GTM, (4) ugtm_kgtm: expectation-maximization algorithm for kGTM, (5) ugtm_landscape: functions for colouring maps, (6) ugtm_predictions: GTM-based prediction algorithms, (7) ugtm_sklearn: sklearn-compatible eGTM transformer, eGTC classifier, and eGTR regressor, (8) ugtm_preprocess: preprocessing functions for data scaling and PCA preprocessing, using sklearn, (9) ugtm_plot: plotting functions for GTM maps, using matplotlib and mpld3, (10) ugtm_crossvalidate: cross-validation workflows.

inherit from scikit-learn classes TransformerMixin (eGTM), ClassifierMixin (eGTC), and RegressorMixin (eGTR).

(2) Installation

The ugtm package can be downloaded from PyPI using pip, by typing "pip install ugtm" in a terminal. In the Python console, the package can be imported by typing "import ugtm".

(3) eGTM: GTM algorithm

The basic GTM dimensionality reduction is implemented into the scikit-learn-compatible eGTM class. The `eGTM.fit()` function fits a GTM model to a data matrix (a numpy array). The `eGTM.transform()` function uses the fitted model to generate a 2D projection for new data (also a numpy array):

```
from ugtm import eGTM
import numpy as np
```

```
# Generate dummy train and test sets
X_train = np.random.randn(100, 50)
X_test = np.random.randn(50, 50)
```

```
# eGTM transformer: fit a map using X_train and
project X_test
eGTM().fit(X_train).transform(X_test,
model="means")
```

```
# Model with different hyperparameters
eGTM(k=10,m=5,s=1,regul=1).fit(X_train)
```

The four GTM hyperparameters (*regul*, *k*, *m*, and *s*) can be tuned: *k* is used for tuning the GTM resolution (a GTM map is discretized into a grid of [*k*, *k*] nodes), *m* is the number of RBF functions (defining an [*m*, *m*] grid), *s* is the RBF function width factor, and *regul* is the regularization coefficient. Implementation details can be found in the API description.

A data point can be represented in 3 ways using GTMs: responsibilities, means and modes. *Responsibilities* represent the probability distribution of a data point on the map; each data point is associated with a vector of k^2 responsibilities (k^2 = number of nodes on the GTM grid). In neural network terminology, a responsibility vector is called a feature vector and can be seen as a processed representation of a datum. These responsibilities can be used to compute the *mean* position of a data point on a GTM, or its *mode* (the node with largest responsibility).

(4) eGTC and eGTR: classification and regression using GTM

The eGTC and eGTR classes implement GTM-based classification and regression algorithms. eGTC and eGTR algorithms are based on class maps and landscapes, which are different ways of coloring a GTM. GTM class maps are constructed using discrete labels and GTM landscapes using continuous labels. New data can be projected onto these colored maps to predict labels. A GTM landscape for the S curve dataset is shown in **Figure 2**, and a GTM class map for the UCI handwritten digits dataset [8] in **Figure 3**, with other data projections using t-distributed stochastic neighbor embedding (t-SNE) [9], multidimensional scaling (MDS) [10] and locally linear embedding (LLE) [11]. The visualizations were produced using ugtm, scikit-learn [6] and altair [12].

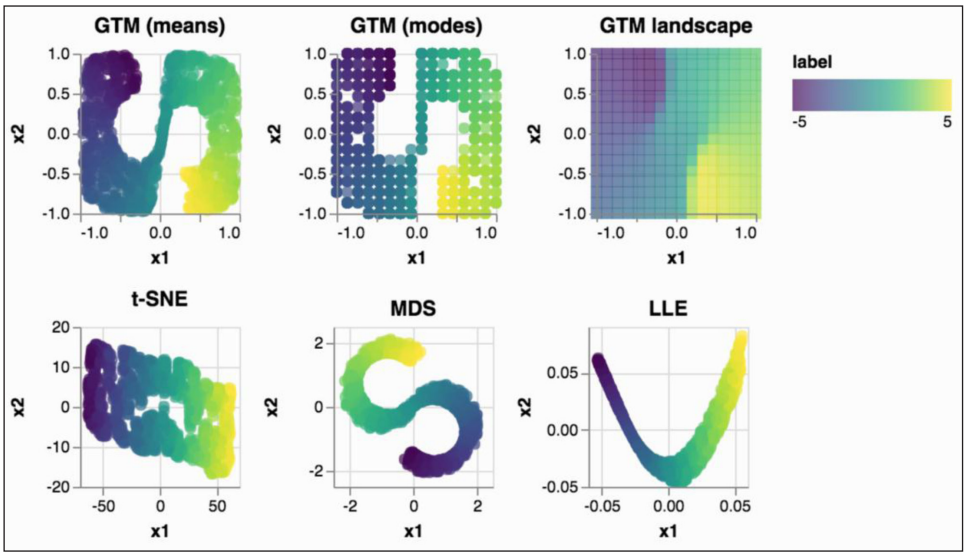

Figure 2: Generative topographic mapping (GTM) representations of the S curve dataset (downloaded from sklearn): mean positions, modes, and landscape for continuous labels. The code to reproduce this plot is accessible online (https://ugtm.readthedocs.io/en/latest/visualization_examples.html). The GTM projection can be compared to t-distributed stochastic neighbor embedding (t-SNE), multidimensional scaling (MDS) or locally linear embedding (LLE). The axes x1 and x2 are latent axes found by the corresponding algorithm.

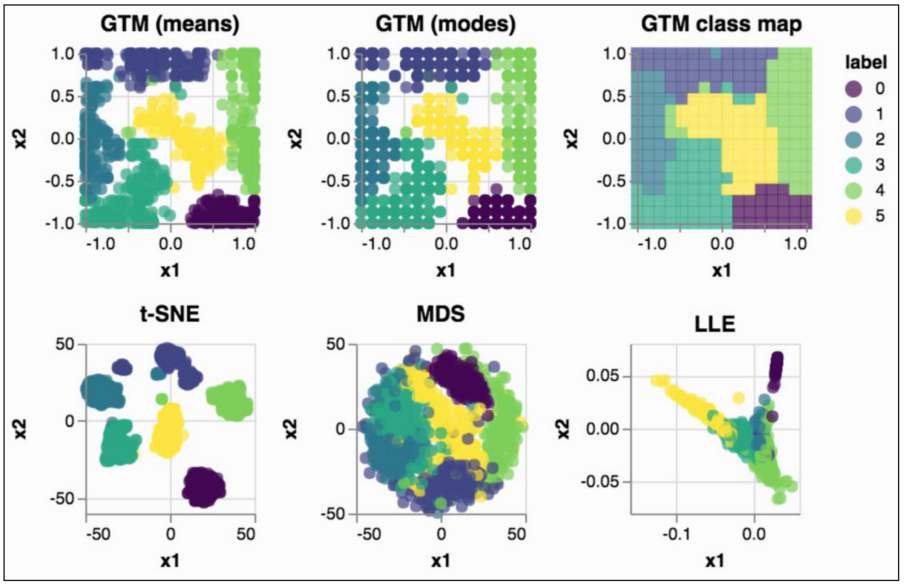

Figure 3: GTM representations of the hand-written digits dataset (digits 0 to 5, from the UCI database): mean positions, modes, and class map for discrete labels. The code to reproduce this plot is accessible online (https://ugtm.readthedocs.io/en/latest/visualization_examples.html). The GTM projection can be compared to t-distributed stochastic neighbor embedding (t-SNE), multidimensional scaling (MDS) or locally linear embedding (LLE). The axes x1 and x2 are latent axes found by the corresponding algorithm.

The `eGTC.fit()` and `eGTR.fit()` functions take as input two numpy arrays: a data matrix and a corresponding label vector. The `eGTC.transform()` and `eGTR.transform()` functions return predicted outcomes as numpy arrays:

```
from ugtm import eGTC, eGTR
import numpy as np
```

```
# Generate dummy train and test sets
X_train = np.random.randn(100, 50)
X_test = np.random.randn(50, 50)
y_train = np.random.choice([1, 2, 3], size=100)
```

```
# eGTC: predict labels for X_test
y_pred = eGTC().fit(X_train,y_train).
transform(X_test)
```

```
# eGTR: predict labels for X_test
y_pred = eGTR().fit(X_train,y_train).
transform(X_test)
```

(5) External resources

The package uses the following external resources:

1. scikit-learn [6], a machine learning library that also provides data preprocessing and statistical evaluation functions.
2. numpy and scipy [13–15], the main scientific packages in Python, used here for linear algebra operations and statistics.
3. matplotlib [16], used to construct plots.
4. mpld3 (http://mpld3.github.io/), which bridges matplotlib and the javascript library D3.js [17] to generate interactive web visualizations.

Quality control

Several examples are provided in the online documentation (https://ugtm.readthedocs.io). Unit tests were conducted using the unittest Python package. The test scripts are available on GitHub (https://github.com/hagax8/ugtm/tree/master/tests). The main test scripts are the core GTM test (test_ugtm_gtm.py), the workflow test (test_ugtm_workflow.py), and scikit-learn compatibility tests (test_ugtm_sklearn.py):

1. Core GTM algorithm (test_ugtm_gtm.py): the core GTM test checks matrix dimensions, convergence of the log likelihood function, and the projection of new data on the map. If the training set and test set are the same, responsibilities of the training and test sets should also be the same.
2. Scikit-learn-compatible classes (test_ugtm_sklearn.py): these tests check the output dimensions of the eGTM transformer, eGTC classifier and eGTR regressor. It also checks for correct projection of new data on the GTM map.
3. Workflow test (test_ugtm_workflow.py): the workflow test script was designed to test all possible workflows, for label-free data, categorical labels, and continuous labels.

Supplementary tests were implemented for plots (test_ugtm_plot.py), printing results (test_ugtm_write.py), classification models (test_ugtm_GTC.py), regression models (test_ugtm_GTR.py), and kernel algorithm (test_ugtm_kgtm.py).

These tests were carried out with Python 2.7.14 and Python 3.4.6 on Scientific Linux 6.6 and macOS High Sierra 10.13.2.

(2) Availability

Operating system

ugtm was tested on Scientific Linux 6.6 and macOS High Sierra 10.13.2 but not on Windows. ugtm is written in pure Python and should be available on any operating system supporting Python frameworks.

Programming language

Python >= 2.7 (tested on Python 3.4.6 and Python 2.7.14).

Additional system requirements

ugtm does not require any supplementary data. The amount of required active memory depends on the input data and on the map size (hyperparameters k and m).

Dependencies

scikit-learn >= 0.20
numpy >= 1.13.1
matplotlib >= 2.0.2
scipy >= 0.19.1
mpld3 >= 0.3

List of contributors

Héléna A. Gaspar

Software location

Archive

Name: ugtm v2.0.0
Persistent identifier: https://doi.org/10.5281/zenodo.1489295
Licence: MIT
Publisher: Héléna A. Gaspar
Version published: 2.0.0
Date published: 15/11/2018

Code repository

Name: GitHub
Identifier: https://github.com/hagax8/ugtm
Licence: MIT
Date published: 15/11/2018

Language

ugtm was developed in English.

(3) Reuse potential

Support for ugtm is available on GitHub (https://github.com/hagax8/ugtm) – users can post issues through the GitHub platform (https://github.com/hagax8/ugtm/issues) or contribute directly to the code.

The eGTM (data transformation), eGTC (classification) and eGTR (regression) classes implemented in ugtm are fully compatible with scikit-learn and can be used in scikit-learn pipelines for visualization, regression or classification. GTM hyperparameters can be optimized using scikit-learn grid search for regression and classification tasks – examples are provided in the online documentation (https://ugtm.readthedocs.io).

Now that we have access to very large amounts of data, dimensionality reduction methods are becoming more and more popular. It is often necessary to get an overview of a dataset that exists in a space of hundreds or thousands of dimensions (data features). GTM can be a nice alternative to other dimensionality reduction algorithms such as t-SNE, MDS or LLE. It also provides a probabilistic framework that can be used to obtain a comprehensive overview of a dataset. For example, GTM can be useful to visualize and cluster multidimensional data for health research, and investigate very large datasets in chemistry or in genomics – single cell data, genotypes, endophenotypes, or polygenic risk scores. At the moment, t-SNE is very popular for these purposes but presents a major drawback: new data cannot be easily projected onto a pre-trained t-SNE map. Another possible application for GTM could be the visualization of feature vectors from deep neural networks. Hopefully, ugtm should make it easier to use GTM for these applications.

ugtm also provides opportunities for further developments. Future enhancements could include the implementation of a mini-batch version (to process data block by data block) – for now, the entire data matrix is processed in one batch. A multispace version of GTM (Stargate GTM [18]) could also be added in the future, as well as data projection functions for the kernel GTM algorithm. Support for other probability distributions could also be included.

Acknowledgements
Thanks to Prof. Alexandre Varnek and Prof. Igor I. Baskin for their help in designing the GTC and GTR algorithms.

Competing Interests
The author has no competing interests to declare.

References

1. **Bishop, C M, Svensén, M** and **Williams, C K I** 1998 GTM: The Generative Topographic Mapping. *Neural Computation*, 10(1): 215–34. DOI: https://doi.org/10.1162/089976698300017953
2. **Olier, I, Vellido, A** and **Giraldo, J** 2010 Kernel generative topographic mapping. *ESANN 2010, 18th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28–30, 2010, Proceedings*.
3. **Kohonen, T** 2001 Self-Organizing Maps. Springer Science & Business Media. DOI: https://doi.org/10.1007/978-3-642-56927-2
4. **Gaspar, H A, Marcou, G, Horvath, D, Arault, A, Lozano, S, Vayer, P,** et al. 2013 Generative topographic mapping-based classification models and their applicability domain: Application to the biopharmaceutics Drug Disposition Classification System (BDDCS). *Journal of Chemical Information and Modeling*, 53(12): 3318–25. DOI: https://doi.org/10.1021/ci400423c
5. **Gaspar, H A, Baskin, I I, Marcou, G, Horvath, D** and **Varnek, A** 2015 GTM-Based QSAR Models and Their Applicability Domains. *Molecular Informatics*, 34(6–7): 348–56. DOI: https://doi.org/10.1002/minf.201400153
6. **Buitinck, L, Louppe, G, Blondel, M, Pedregosa, F, Mueller, A, Grisel, O,** et al. 2013. API design for machine learning software: Experiences from the scikit-learn project. arXiv [cs.LG].
7. **Nabney, I** 2002 NETLAB: Algorithms for Pattern Recognition. Springer Science & Business Media.
8. **Dheeru, D** and **Taniskidou, E K** 2017 UCI machine learning repository (2017). URL: https://archive.ics.uci.edu/ml.
9. **van der Maaten, L** and **Hinton, G** 2008 Visualizing Data using t-SNE. *Journal of Machine Learning Research: JMLR*, 9: 2579–605. Nov.
10. **Cox, T F** and **Cox, M A A** 2000 Multidimensional scaling. Chapman and hall/CRC.
11. **Roweis, S T** and **Saul, L K** 2000 Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500): 2323–6. DOI: https://doi.org/10.1126/science.290.5500.2323
12. n.d. altair. Github.
13. **Oliphant, T E** 2015 Guide to NumPy: 2nd Edition. 2 edition, CreateSpace Independent Publishing Platform.
14. **Jones, E, Oliphant, T** and **Peterson, P** 2001 SciPy: Open Source Scientific Tools for Python.
15. **Oliphant, T E** 2007 Python for Scientific Computing. *Computing in Science Engineering*, 9(3): 10–20. DOI: https://doi.org/10.1109/MCSE.2007.58
16. **Hunter, J D** 2007 Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3): 90–5. DOI: https://doi.org/10.1109/MCSE.2007.55
17. **Bostock, M** n.d. D3.js – Data-Driven Documents. https://d3js.org/ [accessed May 31, 2018].
18. **Gaspar, H A, Baskin, I I, Marcou, G, Horvath, D** and **Varnek, A** 2015 Stargate GTM: Bridging Descriptor and Activity Spaces. *Journal of Chemical Information and Modeling*, 55(11): 2403–10. DOI: https://doi.org/10.1021/acs.jcim.5b00398

How to cite this article: Gaspar, H A 2018 ugtm: A Python Package for Data Modeling and Visualization Using Generative Topographic Mapping. *Journal of Open Research Software,* 6: 26. DOI: https://doi.org/10.5334/jors.235

Submitted: 06 June 2018 **Accepted:** 27 November 2018 **Published:** 19 December 2018