Megaman: Scalable Manifold Learning in Python

James McQueen JMCQ@UW.EDU
Department of Statistics
University of Washington
Seattle, WA 98195-4322, USA

Marina Meilä MMP@STAT.WASHINGTON.EDU
Department of Statistics
University of Washington
Seattle, WA 98195-4322, USA

Jacob VanderPlas JAKEVDP@UW.EDU
e-Science Institute
University of Washington
Seattle, WA 98195-4322, USA

Zhongyue Zhang ZHANGZ@CS.WASHINGTON.EDU
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-4322, USA

Editor: Alexandre Gramfort

Abstract

Manifold Learning (ML) is a class of algorithms seeking a low-dimensional non-linear representation of high-dimensional data. Thus, ML algorithms are most applicable to high-dimensional data and require large sample sizes to accurately estimate the manifold. Despite this, most existing manifold learning implementations are not particularly scalable. Here we present a Python package that implements a variety of manifold learning algorithms in a modular and scalable fashion, using fast approximate neighbors searches and fast sparse eigendecompositions. The package incorporates theoretical advances in manifold learning, such as the unbiased Laplacian estimator introduced by Coifman and Lafon (2006) and the estimation of the embedding distortion by the Riemannian metric method introduced by Perrault-Joncas and Meila (2013). In benchmarks, even on a single-core desktop computer, our code embeds millions of data points in minutes, and takes just 200 minutes to embed the main sample of galaxy spectra from the Sloan Digital Sky Survey—consisting of 0.6 million samples in 3750-dimensions—a task which has not previously been possible.

Keywords: manifold learning, dimension reduction, Riemannian metric, graph embedding, scalable methods, python

1. Motivation

We propose megaman, a new Python package for scalable manifold learning. This package is designed for performance, while inheriting the functionality of scikit-learn’s well-designed API (Buitinck et al., 2013).
2. Downloading and installation

megaman is publicly available at: https://github.com/mmp2/megaman. megaman’s required dependencies are numpy, scipy, and scikit-learn, but for optimal performance FLANN, cython, pyamg and the C compiler gcc are also required. For unit tests and integration megaman depends on nose. The most recent megaman release can be installed along with its dependencies using the cross-platform conda package manager:

```
$ conda install megaman --channel=conda-forge
```

Alternatively, megaman can be installed from source by downloading the source repository and running:

```
$ python setup.py install
```

With nosetests installed, unit tests can be run with:

```
$ make test
```

3. Logical structure and classes overview

embeddings The manifold learning algorithms are implemented in their own classes inheriting from a base class. Included are SpectralEmbedding, which implements Laplacian Eigenmaps (Belkin and Niyogi 2002) and Diffusion Maps (Nadler et al. 2006), LTSA (Zhang and Zha, 2004), LocallyLinearEmbedding (Roweis and Saul 2000), and Isomap (Bernstein et al. 2000). Geometric operations common to many or all embedding algorithms (such as computing distances, Laplacians) are implemented by the Geometry class. A Geometry object is passed or created inside every embedding class. In particular, RiemannianMetric produces the estimated Riemannian metric via the method of Perrault-Joncas and Meila (2013).

eigendecomposition (module) provides a unified (function) interface to the different eigendecomposition methods provided in scipy.

For background of manifold learning, as well as megaman’s design philosophy, please see McQueen et al. (2016).

4. Quick start

```
from megaman.geometry import Geometry
from megaman.embedding import SpectralEmbedding
from sklearn.datasets import make_swiss_roll

X = make_swiss_roll(10000) # generate input data
radius = 1.1 # kernel bandwidth and for graph construction
# a Geometry object encapsulates generic geometric operations
geom = Geometry(
    adjacency_kwds = {'radius':3*radius}, # neighborhood radius
    adjacency_method = 'cyflann', # fast approximate neighbors
```

1. Conda can be downloaded at http://conda.pydata.org/miniconda.html
affinity_method = 'gaussian', # Gaussian kernel
affinity_kwds = {'radius': radius}, # kernel bandwidth
laplacian_method = 'geometric') # unbiased Laplacian
SE = SpectralEmbedding( # embedding algorithm & params
    n_components=2, # embed into 2 dimensions
eigen_solver='amg',
geom=geom) # pass the geometric information
Y = SE.fit_transform(X) # perform embedding

The last two instructions are identical to their analogous instructions in scikit-learn. Full documentation is available from the megaman website at: http://mmp2.github.io/megaman/

5. Benchmarks

The one other popular comparable implementation of manifold learning algorithms is the scikit-learn package. To make the comparison as fair as possible, we choose the SpectralEmbedding method for the comparison, with radius-based neighborhoods and the Locally-Optimized Block-Preconditioned Conjugate Gradient (lobpcg) eigensolver. Note, too, that with the default settings, scikit-learn would perform slower than in our experiments.

We display total embedding time (including time to compute the graph $G$, the Laplacian matrix and the embedding $Y$) for megaman versus scikit-learn, as the number of samples $N$ varies or the data dimension $D$ varies (Figure 1). All benchmark computations were performed on a single desktop computer running Linux with 24.68GB RAM and a Quad-Core 3.07GHz Intel Xeon CPU. We use a relatively weak machine to demonstrate that our package can be reasonably used without high performance hardware. The experiments show that megaman scales considerably better than scikit-learn, even in the most favorable conditions for the latter; the memory footprint of megaman is smaller, even when scikit-learn uses sparse matrices internally. The advantages grow as the data size grows, whether it is w.r.t $D$ or to $N$.

We also report run times on two real world data sets. The first is the word2vec data set which contains feature vectors in 300 dimensions for about 3 million words and phrases, extracted from Google News. The vector representation was obtained via a multilayer neural network by Mikolov et al. (2013). The second data set contains galaxy spectra from the Sloan Digital Sky Survey (Abazajian et al., 2009), preprocessed as described in Telford et al. (2016).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Dimensions</th>
<th>Distances</th>
<th>Embedding</th>
<th>R. metric</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galaxies</td>
<td>0.7M</td>
<td>3750</td>
<td>190.5</td>
<td>8.9</td>
<td>0.1</td>
<td>199.5</td>
</tr>
<tr>
<td>Word2Vec</td>
<td>3M</td>
<td>300</td>
<td>107.9</td>
<td>44.8</td>
<td>0.6</td>
<td>153.3</td>
</tr>
</tbody>
</table>

2. For megaman we also compute the Riemannian metric estimate at each point; this time is negligible compared to the total time to obtain the embedding.

3. The word2vec data used were from GoogleNews-vectors-negative300.bin.gz which can be downloaded from https://code.google.com/archive/p/word2vec/

4. The Sloan Digital Sky Survey data can be downloaded from www.sdss.org
Figure 1: **Run time vs. data set size** $N$ for fixed $D = 100$ (left) and **Run time vs. data set dimension** $D$ for fixed $N = 50,000$ (right). The data is from a Swiss Roll (in 3 dimensions) with additional noise dimensions, embedded into $s = 2$ dimensions by the SpectralEmbedding algorithm. By $D = 10,000$ and $N = 1,000,000$ scikit-learn was unable to compute an embedding due to insufficient memory. All megaman run times (including time between distance and embedding) are faster than scikit-learn.

6. Conclusion

megaman puts in the hands of scientists and methodologists alike tools that enable them to apply state of the art manifold learning methods to data sets of realistic size. The package is extensible, modular, with an API familiar to scikit-learn users. Future development will be mainly in the direction of further scalability (Nystrom extension, parallelization) and expanding the data analytical tools (distance calculations, estimation of dimension, estimation of neighborhood radius, directed graph embedding).

We hope that by providing this package, non-linear dimension reduction will be benefit those who most need it: the practitioners exploring large scientific data sets.

Acknowledgments

We would like to acknowledge support for this project from the National Science Foundation (NSF grant IIS-9988642), the Multidisciplinary Research Program of the Department of Defense (MURI N00014-00-1-0637), the Department of Defense (62-7760 “DOD Unclassified Math”), the Moore/Sloan Data Science Environment grant, and funding from the Washington Research Foundation. This project grew from the Data Science Incubator program at the University of Washington eScience Institute.

5. For more information on Data Science Incubator program see [http://data.uw.edu/incubator/](http://data.uw.edu/incubator/)
References


O. Grace Telford, Jacob Vanderplas, James McQueen, and Marina Meila. Metric embedding of Sloan galaxy spectra. (in preparation), 2016.