MAGIC Documentation

Release 3.0.0

Scott Gigante and Daniel Dager, Krishnaswamy Lab, Yale Univers

Mar 11, 2021

Contents

Markov Affinity-based Graph Imputation of Cells (MAGIC) is an algorithm for denoising high-dimensional data most commonly applied to single-cell RNA sequencing data. MAGIC learns the manifold data, using the resultant graph to smooth the features and restore the structure of the data.

To see how MAGIC can be applied to single-cell RNA-seq, elucidating the epithelial-to-mesenchymal transition, read our publication in Cell.

David van Dijk, et al. Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. 2018. Cell.

CHAPTER 1

Installation

1.1 Python installation

1.1.1 Installation with *pip*

The Python version of MAGIC can be installed using:

```
pip install --user magic-impute
```

1.1.2 Installation from source

The Python version of MAGIC can be installed from GitHub by running the following from a terminal:

```
git clone --recursive git://github.com/KrishnaswamyLab/MAGIC.git
cd MAGIC/python
python setup.py install --user
```

1.2 MATLAB installation

1. The MATLAB version of MAGIC can be accessed using:

   ```
   git clone git://github.com/KrishnaswamyLab/MAGIC.git
   cd MAGIC/Matlab
   ```

2. Add the MAGIC/Matlab directory to your MATLAB path and run any of our *run* or *test* scripts to get a feel for MAGIC.

1.3 R installation

In order to use MAGIC in R, you must also install the Python package.

If *python* or *pip* are not installed, you will need to install them. We recommend Miniconda3 to install Python and *pip* together, or otherwise you can install *pip* from https://pip.pypa.io/en/stable/installing/.

1.3.1 Installation from CRAN

In R, run this command to install MAGIC and all dependencies:

```
install.packages("Rmagic")
```

In a terminal, run the following command to install the Python repository:

```
pip install --user magic-impute
```

1.3.2 Installation from source

The latest source version of MAGIC can be accessed by running the following in a terminal:

```
git clone https://github.com/KrishnaswamyLab/MAGIC.git
cd MAGIC/Rmagic
R CMD INSTALL .
cd ../python
python setup.py install --user
```

If the *Rmagic* folder is empty, you have may forgotten to use the *–recursive* option for *git clone*. You can rectify this by running the following in a terminal:

```
cd MAGIC
git submodule init
git submodule update
cd Rmagic
R CMD INSTALL
cd ../python
python setup.py install --user
```

CHAPTER 2

Tutorial

To run MAGIC on your dataset, create a MAGIC operator and run *fit_transform*. Here we show an example with an artificial test dataset located in the MAGIC repository:

```
import magic
import matplotlib.pyplot as plt
import pandas as pd
X = pd.read_csv("MAGIC/data/test_data.csv")
magic_operator = magic.MAGIC()
X_magic = magic_operator.fit_transform(X, genes=['VIM', 'CDH1', 'ZEB1'])
plt.scatter(X_magic['VIM'], X_magic['CDH1'], c=X_magic['ZEB1'], s=1, cmap='inferno')
plt.show()
magic.plot.animate_magic(X, gene_x='VIM', gene_y='CDH1', gene_color='ZEB1',␣
↪operator=magic_operator)
```

A demo on MAGIC usage for single cell RNA-seq data can be found in this notebook: http://nbviewer.jupyter.org/github/KrishnaswamyLab/magic/blob/master/python/tutorial_notebooks/emt_tutorial.ipynb

A second tutorial analyzing myeloid and erythroid cells in mouse bone marrow is available here: http://nbviewer.jupyter.org/github/KrishnaswamyLab/magic/blob/master/python/tutorial_notebooks/bonemarrow_tutorial.ipynb

CHAPTER 3

API

3.1 MAGIC

Markov Affinity-based Graph Imputation of Cells (MAGIC)

Authors: Scott Gigante <scott.gigante@yale.edu>, Daniel Dager <daniel.dager@yale.edu> (C) 2018 Krishnaswamy Lab GPLv2

class magic.magic.**MAGIC**(*knn=5*, *knn_max=None*, *decay=1*, *t=3*, *n_pca=100*, *solver='exact'*, *knn_dist='euclidean'*, *n_jobs=1*, *random_state=None*, *verbose=1*)

Bases: sklearn.base.BaseEstimator

MAGIC operator which performs dimensionality reduction.

Markov Affinity-based Graph Imputation of Cells (MAGIC) is an algorithm for denoising and transcript recover of single cells applied to single-cell RNA sequencing data, as described in van Dijk et al, 2018[1].

Parameters

- **knn** (*int, optional, default: 5*) – number of nearest neighbors from which to compute kernel bandwidth
- **knn_max** (*int, optional, default: None*) – maximum number of nearest neighbors with nonzero connection. If *None*, will be set to 3 * *knn*
- **decay** (*int, optional, default: 1*) – sets decay rate of kernel tails. If None, alpha decaying kernel is not used
- **t** (*int, optional, default: 3*) – power to which the diffusion operator is powered. This sets the level of diffusion. If 'auto', t is selected according to the Procrustes disparity of the diffused data
- **n_pca** (*int, optional, default: 100*) – Number of principal components to use for calculating neighborhoods. For extremely large datasets, using n_pca < 20 allows neighborhoods to be calculated in roughly log(n_samples) time.

[1] Van Dijk D *et al.* (2018), *Recovering Gene Interactions from Single-Cell Data Using Data Diffusion*, Cell.

- **solver** (*str, optional, default: 'exact'*) – Which solver to use. "exact" uses the implementation described in van Dijk et al. (2018)[1]. "approximate" uses a faster implementation that performs imputation in the PCA space and then projects back to the gene space. Note, the "approximate" solver may return negative values.
- **knn_dist** (*string, optional, default: 'euclidean'*) – Distance metric for building kNN graph. Recommended values: 'euclidean', 'cosine'. Any metric from *scipy.spatial.distance* can be used. Custom distance functions of form *f(x, y) = d* are also accepted
- **n_jobs** (*integer, optional, default: 1*) – The number of jobs to use for the computation. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n_jobs below -1, (n_cpus + 1 + n_jobs) are used. Thus for n_jobs = -2, all CPUs but one are used
- **random_state** (*integer or numpy.RandomState, optional, default: None*) – The generator used to initialize random PCA If an integer is given, it fixes the seed Defaults to the global *numpy* random number generator
- **verbose** (*int* or *boolean*, optional (default: 1)) – If *True* or > *0*, print status messages

X
Input data

Type array-like, shape=[n_samples, n_features]

X_magic
Output data

Type array-like, shape=[n_samples, n_features]

graph
The graph built on the input data

Type graphtools.BaseGraph

Examples

```
>>> import magic
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> X = pd.read_csv("../../data/test_data.csv")
>>> X.shape
(500, 197)
>>> magic_operator = magic.MAGIC()
>>> X_magic = magic_operator.fit_transform(X, genes=['VIM', 'CDH1', 'ZEB1'])
>>> X_magic.shape
(500, 3)
>>> magic_operator.set_params(t=7)
MAGIC(a=15, k=5, knn_dist='euclidean', n_jobs=1, n_pca=100,
   random_state=None, t=7, verbose=1)
>>> X_magic = magic_operator.transform(genes=['VIM', 'CDH1', 'ZEB1'])
>>> X_magic.shape
(500, 3)
>>> X_magic = magic_operator.transform(genes="all_genes")
>>> X_magic.shape
(500, 197)
>>> plt.scatter(X_magic['VIM'], X_magic['CDH1'],
...             c=X_magic['ZEB1'], s=1, cmap='inferno')
```

(continues on next page)

(continued from previous page)

```
>>> plt.show()
>>> magic.plot.animate_magic(X, gene_x='VIM', gene_y='CDH1',
...                          gene_color='ZEB1', operator=magic_operator)
>>> dremi = magic_operator.knnDREMI('VIM', 'CDH1', plot=True)
```

References

diff_op
The diffusion operator calculated from the data

fit(*X*, *graph=None*)
Computes the diffusion operator

Parameters

- **X** (*array, shape=[n_samples, n_features]*) – input data with *n_samples* samples and *n_features* dimensions. Accepted data types: *numpy.ndarray*, *scipy.sparse.spmatrix*, *pd.DataFrame*, *anndata.AnnData*.
- **graph** (*graphtools.Graph*, optional (default: None)) – If given, provides a precomputed kernel matrix with which to perform diffusion.

Returns **magic_operator** – The estimator object

Return type *MAGIC*

fit_transform(*X*, *graph=None*, ***kwargs*)
Computes the diffusion operator and the denoised gene expression

Parameters

- **X** (*array, shape=[n_samples, n_features]*) – input data with *n_samples* samples and *n_features* dimensions. Accepted data types: *numpy.ndarray*, *scipy.sparse.spmatrix*, *pd.DataFrame*, *anndata.AnnData*.
- **graph** (*graphtools.Graph*, optional (default: None)) – If given, provides a precomputed kernel matrix with which to perform diffusion.
- **genes** (*list or {"all_genes", "pca_only"}, optional (default: None)*) – List of genes, either as integer indices or column names if input data is a pandas DataFrame. If "all_genes", the entire smoothed matrix is returned. If "pca_only", PCA on the smoothed data is returned. If None, the entire matrix is also returned, but a warning may be raised if the resultant matrix is very large.
- **t_max** (*int, optional, default: 20*) – maximum t to test if *t* is set to 'auto'
- **plot_optimal_t** (*boolean, optional, default: False*) – If true and *t* is set to 'auto', plot the disparity used to select t
- **ax** (*matplotlib.axes.Axes, optional*) – If given and *plot_optimal_t* is true, plot will be drawn on the given axis.

Returns **X_magic** – The gene expression values after diffusion

Return type array, shape=[n_samples, n_genes]

get_params(*deep=True*)
Get parameters for this estimator.

Parameters **deep** (*bool, default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type dict

knnDREMI(*gene_x*, *gene_y*, *k=10*, *n_bins=20*, *n_mesh=3*, *n_jobs=1*, *plot=False*, ***kwargs*)
Calculate kNN-DREMI on MAGIC output

Calculates k-Nearest Neighbor conditional Density Resampled Estimate of Mutual Information as defined in Van Dijk et al, 2018.[1]

Note that kNN-DREMI, like Mutual Information and DREMI, is not symmetric. Here we are estimating I(Y|X).

Parameters

- **gene_x** (*array-like, shape=[n_samples]*) – Gene shown on the x axis (independent feature)
- **gene_y** (*array-like, shape=[n_samples]*) – Gene shown on the y axis (dependent feature)
- **k** (*int, range=[0:n_samples), optional (default: 10)*) – Number of neighbors
- **n_bins** (*int, range=[0:inf), optional (default: 20)*) – Number of bins for density resampling
- **n_mesh** (*int, range=[0:inf), optional (default: 3)*) – In each bin, density will be calculcated around (mesh ** 2) points
- **n_jobs** (*int, optional (default: 1)*) – Number of threads used for kNN calculation
- **plot** (*bool, optional (default: False)*) – If True, DREMI create plots of the data like those seen in Fig 5C/D of van Dijk et al. 2018. (doi:10.1016/j.cell.2018.05.061).
- ****kwargs** (additional arguments for *scprep.stats.plot_knnDREMI*) –

Returns **dremi** – kNN condtional Density resampled estimate of mutual information

Return type float

set_params(***params*)
Set the parameters on this estimator.

Any parameters not given as named arguments will be left at their current value.

Parameters

- **knn** (*int, optional, default: 5*) – number of nearest neighbors on which to build kernel
- **decay** (*int, optional, default: 1*) – sets decay rate of kernel tails. If None, alpha decaying kernel is not used
- **t** (*int, optional, default: 3*) – power to which the diffusion operator is powered. This sets the level of diffusion. If 'auto', t is selected according to the R squared of the diffused data
- **n_pca** (*int, optional, default: 100*) – Number of principal components to use for calculating neighborhoods. For extremely large datasets, using n_pca < 20 allows neighborhoods to be calculated in roughly log(n_samples) time.

- **knn_dist** (*string, optional, default: 'euclidean'*) – recommended values: 'euclidean', 'cosine' Any metric from *scipy.spatial.distance* can be used distance metric for building kNN graph.
- **n_jobs** (*integer, optional, default: 1*) – The number of jobs to use for the computation. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n_jobs below -1, (n_cpus + 1 + n_jobs) are used. Thus for n_jobs = -2, all CPUs but one are used
- **random_state** (*integer or numpy.RandomState, optional, default: None*) – The generator used to initialize random PCA If an integer is given, it fixes the seed Defaults to the global *numpy* random number generator
- **verbose** (*int* or *boolean*, optional (default: 1)) – If *True* or > *0*, print status messages

Returns

Return type self

transform(*X=None*, *genes=None*, *t_max=20*, *plot_optimal_t=False*, *ax=None*)

Computes the values of genes after diffusion

Parameters

- **X** (*array, optional, shape=[n_samples, n_features]*) – input data with *n_samples* samples and *n_features* dimensions. Not required, since MAGIC does not embed cells not given in the input matrix to *MAGIC.fit()*. Accepted data types: *numpy.ndarray*, *scipy.sparse.spmatrix*, *pd.DataFrame*, *anndata.AnnData*.
- **genes** (*list or {"all_genes", "pca_only"}, optional (default: None)*) – List of genes, either as integer indices or column names if input data is a pandas DataFrame. If "all_genes", the entire smoothed matrix is returned. If "pca_only", PCA on the smoothed data is returned. If None, the entire matrix is also returned, but a warning may be raised if the resultant matrix is very large.
- **t_max** (*int, optional, default: 20*) – maximum t to test if *t* is set to 'auto'
- **plot_optimal_t** (*boolean, optional, default: False*) – If true and *t* is set to 'auto', plot the disparity used to select t
- **ax** (*matplotlib.axes.Axes, optional*) – If given and *plot_optimal_t* is true, plot will be drawn on the given axis.

Returns **X_magic** – The gene expression values after diffusion

Return type array, shape=[n_samples, n_genes]

3.2 Plotting

magic.plot.**animate_magic**(*data*, *gene_x*, *gene_y*, *gene_color=None*, *t_max=20*, *delay=2*, *operator=None*, *filename=None*, *ax=None*, *figsize=None*, *s=1*, *cmap='inferno'*, *interval=200*, *dpi=100*, *ipython_html='jshtml'*, *verbose=False*, ***kwargs*)

Animate a gene-gene relationship with increased diffusion

Parameters

- **data** (*array-like*) – Input data matrix
- **gene_x** (*int or str*) – Gene to put on the x axis

- **gene_y** (*int or str*) – Gene to put on the y axis
- **gene_color** (*int or str, optional (default: None)*) – Gene to color by. If None, no color vector is used
- **t_max** (*int, optional (default: 20)*) – maximum value of t to include in the animation
- **delay** (*int, optional (default: 5)*) – number of frames to dwell on the first frame before applying MAGIC
- **operator** (magic.MAGIC, *optional (default: None)*) – precomputed MAGIC operator. If None, one is created.
- **filename** (*str, optional (default: None)*) – If not None, saves a .gif or .mp4 with the output
- **ax** (*matplotlib.Axes* or None, optional (default: None)) – axis on which to plot. If None, an axis is created
- **figsize** (*tuple, optional (default: None)*) – Tuple of floats for creation of new *matplotlib* figure. Only used if *ax* is None.
- **s** (*int, optional (default: 1)*) – Point size
- **cmap** (*str or callable, optional (default: 'inferno')*) – Matplotlib colormap
- **interval** (*float, optional (default: 30)*) – Time in milliseconds between frames
- **dpi** (*int, optional (default: 100)*) – Dots per inch (image quality) in saved animation)
- **ipython_html** (*{'html5', 'jshtml'}*) – which html writer to use if using a Jupyter Notebook
- **verbose** (*bool, optional (default: False)*) – MAGIC operator verbosity
- ***kwargs** (*arguments for MAGIC*) –

Returns

Return type A Matplotlib animation showing diffusion of an edge with increased t

CHAPTER 4

Quick Start

To run MAGIC on your dataset, create a MAGIC operator and run *fit_transform*. Here we show an example with a small, artificial dataset located in the MAGIC repository:

```
import magic
import pandas as pd
import matplotlib.pyplot as plt
X = pd.read_csv("MAGIC/data/test_data.csv")
magic_operator = magic.MAGIC()
X_magic = magic_operator.fit_transform(X, genes=['VIM', 'CDH1', 'ZEB1'])
plt.scatter(X_magic['VIM'], X_magic['CDH1'], c=X_magic['ZEB1'], s=1, cmap='inferno')
plt.show()
magic.plot.animate_magic(X, gene_x='VIM', gene_y='CDH1', gene_color='ZEB1',␣
↪operator=magic_operator)
```

CHAPTER 5

Help

If you have any questions or require assistance using MAGIC, please contact us at https://krishnaswamylab.org/get-help

class magic.**MAGIC**(*knn=5*, *knn_max=None*, *decay=1*, *t=3*, *n_pca=100*, *solver='exact'*, *knn_dist='euclidean'*, *n_jobs=1*, *random_state=None*, *verbose=1*)

MAGIC operator which performs dimensionality reduction.

Markov Affinity-based Graph Imputation of Cells (MAGIC) is an algorithm for denoising and transcript recover of single cells applied to single-cell RNA sequencing data, as described in van Dijk et al, 2018[1].

Parameters

- **knn** (*int, optional, default: 5*) – number of nearest neighbors from which to compute kernel bandwidth
- **knn_max** (*int, optional, default: None*) – maximum number of nearest neighbors with nonzero connection. If *None*, will be set to 3 * *knn*
- **decay** (*int, optional, default: 1*) – sets decay rate of kernel tails. If None, alpha decaying kernel is not used
- **t** (*int, optional, default: 3*) – power to which the diffusion operator is powered. This sets the level of diffusion. If 'auto', t is selected according to the Procrustes disparity of the diffused data
- **n_pca** (*int, optional, default: 100*) – Number of principal components to use for calculating neighborhoods. For extremely large datasets, using n_pca < 20 allows neighborhoods to be calculated in roughly log(n_samples) time.
- **solver** (*str, optional, default: 'exact'*) – Which solver to use. "exact" uses the implementation described in van Dijk et al. (2018)[1]. "approximate" uses a faster implementation that performs imputation in the PCA space and then projects back to the gene space. Note, the "approximate" solver may return negative values.
- **knn_dist** (*string, optional, default: 'euclidean'*) – Distance metric for building kNN graph. Recommended values: 'euclidean', 'cosine'. Any metric from

[1] Van Dijk D *et al.* (2018), *Recovering Gene Interactions from Single-Cell Data Using Data Diffusion*, Cell.

scipy.spatial.distance can be used. Custom distance functions of form *f(x, y) = d* are also accepted

- **n_jobs** (*integer, optional, default: 1*) – The number of jobs to use for the computation. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n_jobs below -1, (n_cpus + 1 + n_jobs) are used. Thus for n_jobs = -2, all CPUs but one are used
- **random_state** (*integer or numpy.RandomState, optional, default: None*) – The generator used to initialize random PCA If an integer is given, it fixes the seed Defaults to the global *numpy* random number generator
- **verbose** (*int* or *boolean*, optional (default: 1)) – If *True* or > *0*, print status messages

X
Input data

Type array-like, shape=[n_samples, n_features]

X_magic
Output data

Type array-like, shape=[n_samples, n_features]

graph
The graph built on the input data

Type graphtools.BaseGraph

Examples

```
>>> import magic
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> X = pd.read_csv("../../data/test_data.csv")
>>> X.shape
(500, 197)
>>> magic_operator = magic.MAGIC()
>>> X_magic = magic_operator.fit_transform(X, genes=['VIM', 'CDH1', 'ZEB1'])
>>> X_magic.shape
(500, 3)
>>> magic_operator.set_params(t=7)
MAGIC(a=15, k=5, knn_dist='euclidean', n_jobs=1, n_pca=100,
   random_state=None, t=7, verbose=1)
>>> X_magic = magic_operator.transform(genes=['VIM', 'CDH1', 'ZEB1'])
>>> X_magic.shape
(500, 3)
>>> X_magic = magic_operator.transform(genes="all_genes")
>>> X_magic.shape
(500, 197)
>>> plt.scatter(X_magic['VIM'], X_magic['CDH1'],
...             c=X_magic['ZEB1'], s=1, cmap='inferno')
>>> plt.show()
>>> magic.plot.animate_magic(X, gene_x='VIM', gene_y='CDH1',
...                          gene_color='ZEB1', operator=magic_operator)
>>> dremi = magic_operator.knnDREMI('VIM', 'CDH1', plot=True)
```

References

diff_op
The diffusion operator calculated from the data

fit (*X*, *graph=None*)
Computes the diffusion operator

Parameters

- **X** (*array, shape=[n_samples, n_features]*) – input data with *n_samples* samples and *n_features* dimensions. Accepted data types: *numpy.ndarray*, *scipy.sparse.spmatrix*, *pd.DataFrame*, *anndata.AnnData*.
- **graph** (*graphtools.Graph*, optional (default: None)) – If given, provides a precomputed kernel matrix with which to perform diffusion.

Returns **magic_operator** – The estimator object

Return type *MAGIC*

fit_transform (*X*, *graph=None*, ***kwargs*)
Computes the diffusion operator and the denoised gene expression

Parameters

- **X** (*array, shape=[n_samples, n_features]*) – input data with *n_samples* samples and *n_features* dimensions. Accepted data types: *numpy.ndarray*, *scipy.sparse.spmatrix*, *pd.DataFrame*, *anndata.AnnData*.
- **graph** (*graphtools.Graph*, optional (default: None)) – If given, provides a precomputed kernel matrix with which to perform diffusion.
- **genes** (*list or {"all_genes", "pca_only"}, optional (default: None)*) – List of genes, either as integer indices or column names if input data is a pandas DataFrame. If "all_genes", the entire smoothed matrix is returned. If "pca_only", PCA on the smoothed data is returned. If None, the entire matrix is also returned, but a warning may be raised if the resultant matrix is very large.
- **t_max** (*int, optional, default: 20*) – maximum t to test if *t* is set to 'auto'
- **plot_optimal_t** (*boolean, optional, default: False*) – If true and *t* is set to 'auto', plot the disparity used to select t
- **ax** (*matplotlib.axes.Axes, optional*) – If given and *plot_optimal_t* is true, plot will be drawn on the given axis.

Returns **X_magic** – The gene expression values after diffusion

Return type array, shape=[n_samples, n_genes]

knnDREMI (*gene_x*, *gene_y*, *k=10*, *n_bins=20*, *n_mesh=3*, *n_jobs=1*, *plot=False*, ***kwargs*)
Calculate kNN-DREMI on MAGIC output

Calculates k-Nearest Neighbor conditional Density Resampled Estimate of Mutual Information as defined in Van Dijk et al, 2018.[1]

Note that kNN-DREMI, like Mutual Information and DREMI, is not symmetric. Here we are estimating I(Y|X).

Parameters

- **gene_x** (*array-like, shape=[n_samples]*) – Gene shown on the x axis (independent feature)

- **gene_y** (*array-like, shape=[n_samples]*) – Gene shown on the y axis (dependent feature)
- **k** (*int, range=[0:n_samples), optional (default: 10)*) – Number of neighbors
- **n_bins** (*int, range=[0:inf), optional (default: 20)*) – Number of bins for density resampling
- **n_mesh** (*int, range=[0:inf), optional (default: 3)*) – In each bin, density will be calculcated around (mesh ** 2) points
- **n_jobs** (*int, optional (default: 1)*) – Number of threads used for kNN calculation
- **plot** (*bool, optional (default: False)*) – If True, DREMI create plots of the data like those seen in Fig 5C/D of van Dijk et al. 2018. (doi:10.1016/j.cell.2018.05.061).
- ****kwargs** (additional arguments for *scprep.stats.plot_knnDREMI*) –

Returns **dremi** – kNN condtional Density resampled estimate of mutual information

Return type float

set_params(***params*)

Set the parameters on this estimator.

Any parameters not given as named arguments will be left at their current value.

Parameters

- **knn** (*int, optional, default: 5*) – number of nearest neighbors on which to build kernel
- **decay** (*int, optional, default: 1*) – sets decay rate of kernel tails. If None, alpha decaying kernel is not used
- **t** (*int, optional, default: 3*) – power to which the diffusion operator is powered. This sets the level of diffusion. If 'auto', t is selected according to the R squared of the diffused data
- **n_pca** (*int, optional, default: 100*) – Number of principal components to use for calculating neighborhoods. For extremely large datasets, using n_pca < 20 allows neighborhoods to be calculated in roughly log(n_samples) time.
- **knn_dist** (*string, optional, default: 'euclidean'*) – recommended values: 'euclidean', 'cosine' Any metric from *scipy.spatial.distance* can be used distance metric for building kNN graph.
- **n_jobs** (*integer, optional, default: 1*) – The number of jobs to use for the computation. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n_jobs below -1, (n_cpus + 1 + n_jobs) are used. Thus for n_jobs = -2, all CPUs but one are used
- **random_state** (*integer or numpy.RandomState, optional, default: None*) – The generator used to initialize random PCA If an integer is given, it fixes the seed Defaults to the global *numpy* random number generator
- **verbose** (*int* or *boolean*, optional (default: 1)) – If *True* or > *0*, print status messages

Returns

Return type self

transform(*X=None*, *genes=None*, *t_max=20*, *plot_optimal_t=False*, *ax=None*)
Computes the values of genes after diffusion

Parameters

- **X** (*array, optional, shape=[n_samples, n_features]*) – input data with *n_samples* samples and *n_features* dimensions. Not required, since MAGIC does not embed cells not given in the input matrix to *MAGIC.fit()*. Accepted data types: *numpy.ndarray*, *scipy.sparse.spmatrix*, *pd.DataFrame*, *anndata.AnnData*.
- **genes** (*list or {"all_genes", "pca_only"}, optional (default: None)*) – List of genes, either as integer indices or column names if input data is a pandas DataFrame. If "all_genes", the entire smoothed matrix is returned. If "pca_only", PCA on the smoothed data is returned. If None, the entire matrix is also returned, but a warning may be raised if the resultant matrix is very large.
- **t_max** (*int, optional, default: 20*) – maximum t to test if *t* is set to 'auto'
- **plot_optimal_t** (*boolean, optional, default: False*) – If true and *t* is set to 'auto', plot the disparity used to select t
- **ax** (*matplotlib.axes.Axes, optional*) – If given and *plot_optimal_t* is true, plot will be drawn on the given axis.

Returns **X_magic** – The gene expression values after diffusion

Return type array, shape=[n_samples, n_genes]

Python Module Index

m

Index