meld Documentation

Release 1.0.0

Daniel Burkhardt

Dec 10, 2020

CONTENTS

MELD is a Python package for quantifying the effects of experimental perturbations. For an in depth explanation of the algorithm, read our manuscript on BioRxiv: https://www.biorxiv.org/content/10.1101/532846v2

The goal of MELD is to identify populations of cells that are most affected by an experimental perturbation. Rather than clustering the data first and calculating differential abundance of samples within clusters, MELD provides a density estimate for each scRNA-seq sample for every cell in each dataset. Comparing the ratio between the density of each sample provides a quantitative estimate the effect of a perturbation at the single-cell level. We can then identify the cells most or least affected by the perturbation.

CHAPTER

ONE

INSTALLATION

```
pip install --user git+git://github.com/KrishnaswamyLab/MELD.git#subdirectory=python
```

CHAPTER

TWO

REQUIREMENTS

MELD requires Python >= 3.6. All other requirements are installed automatically by *pip*.

CHAPTER

THREE

REFERENCE

3.1 MELD density estimation

class meld.meld.**MELD**(*beta=60*, *offset=0*, *order=1*, *filter='heat'*, *solver='chebyshev'*, *chebyshev_order=50*, *lap_type='combinatorial'*, *sample_normalize=True*, *anisotropy=1*, *n_landmark=None*, ***kwargs*)

MELD operator for filtering signals over a graph.

Parameters

- **beta** (*int, optional, Default: 60*) – Amount of smoothing to apply. Default value of 60 determined through analysis of simulated data using Splatter.
- **offset** (*float, optional, Default: 0*) – Amount to shift the MELD filter in the eigenvalue spectrum. Recommend using an eigenvalue from the graph based on the spectral distribution. Should be in interval [0,1]
- **order** (*int, optional, Default: 1*) – Falloff and smoothness of the filter. High order leads to square-like filters.
- **filter** (*str, optional, Default: 'heat'*) – Filter type to use. Should be in ['heat', 'laplacian']
- **solver** (*string, optional, Default: 'chebyshev'*) – Method to solve convex problem. 'chebyshev' uses a chebyshev polynomial approximation of the corresponding filter. 'exact' uses the eigenvalue solution to the problem
- **chebyshev_order** (*int, optional, Default: 50*) – Order of chebyshev approximation to use.
- **lap_type** (*('combinatorial', 'normalized'), Default: 'combinatorial'*) – The kind of Laplacian to calculate
- **sample_normalize** (*boolean, optional, Default: True*) – If True, the sample indicator vectors are column normalized to sum to 1

property beta

Amount of smoothing to apply. Default value of 60 determined throughanalysis of simulated data using Splatter

property chebyshev_order

Order of chebyshev approximation to use.

property filter

Filter type to use. Should be in ['heat', 'laplacian']

fit_transform(*X*, *sample_labels*, ***kwargs*)

Builds the MELD filter over a graph built on data *X* and estimates density of each sample in *sample_labels*

Parameters

- **X** (*array-like, shape=[n_samples, m_features]*) – Data on which to build graph to perform data smoothing over.
- **sample_labels** (*array-like, shape=[n_samples, p_signals]*) – 1- or 2-dimensional array of non-numerics indicating the sample origin for each cell.
- **kwargs** (*additional arguments for graphtools.Graph*) –

Returns **sample_densities** – Density estimate for each sample over a graph built from X

Return type ndarray, shape=[n_samples, p_signals]

property lap_type
The kind of Laplacian to calculate

property offset
Amount to shift the MELD filter in the eigenvalue spectrum.Recommend using an eigenvalue from the graph based on thespectral distribution. Should be in interval [0,1]

property order
Falloff and smoothness of the filter.High order leads to square-like filters.

property sample_densities
Density associated with each sample

property solver
Method to solve convex problem.'chebyshev' uses a chebyshev polynomial approximation of the correspondingfilter. 'exact' uses the eigenvalue solution to the problem

transform(*sample_labels*)
Filters a collection of sample_indicators over the data graph.

Parameters **sample_indicators** (*ndarray [n, p]*) – 1- or 2-dimensional sample indicator array to filter.

Returns **sample_densities** – A density estimate for each sample.

Return type ndarray [n, p]

3.2 Vertex Frequency Clustering

class meld.cluster.**VertexFrequencyCluster**(*n_clusters=10*, *likelihood_bias=1*, *window_count=9*, *window_sizes=None*, *sparse=False*, *suppress=False*, *random_state=None*, ***kwargs*)

Bases: sklearn.base.BaseEstimator

Performs Vertex Frequency clustering for data given a raw experimental signal and enhanced experimental signal.

Parameters

- **n_clusters** (*int, optional, default: 10*) – The number of clusters to form.
- **likelihood_bias** (*float, optional, default: 1*) – A normalization term that biases clustering towards the likelihood (higher values) or towards the spectrogram (lower values)

- **window_count** (*int, optional, default: 9*) – Number of windows to use if window_sizes = None
- **window_sizes** (*None, optional, default: None*) – ndarray of integer window sizes to supply to t
- **sparse** (*bool, optional, default: False*) – Use sparse matrices. This is significantly slower, but will use less memory
- **suppress** (*bool, optional*) – Suppress warnings
- **random_state** (*int or None, optional (default: None)*) – Random seed for clustering
- ****kwargs** – Description

Raises **NotImplementedError** – Window functions are not implemented

Examples

fit(*G*)
Sets eigenvectors and windows.

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 mapping of string to any

predict(*n_clusters=None*, ***kwargs*)
Runs KMeans on the spectrogram.

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

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Parameters ****params** (*dict*) – Estimator parameters.

Returns **self** – Estimator instance.

Return type object

transform(*sample_indicator*, *likelihood=None*, *center=True*)
Calculates the spectrogram of the graph using the sample_indicator

CHAPTER

FOUR

QUICK START

You can use *meld* as follows:

```
import numpy as np
import meld

# Create toy data
n_samples = 500
n_dimensions = 100
data = np.random.normal(size=(n_samples, n_dimensions))
sample_labels = np.random.choice(['treatment', 'control'], size=n_samples)

# Estimate density of each sample over the graph
sample_densities = meld.MELD().fit_transform(data, sample_labels)

# Normalize densities to calculate sample likelihoods
sample_likelihoods = meld.utils.normalize_densities(sample_densities)
```

CHAPTER

FIVE

HELP

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

PYTHON MODULE INDEX

m

INDEX