THE ELEMENTARY LIBMAXDIV USER GUIDE

1. WHAT IS LIBMAXDIV?

`libmaxdiv` is an implementation of the "Maximally Divergent Intervals" (MDI) algorithm for detection of anomalous intervals in spatio-temporal time-series. The implementation is written in `C++` and provides a `C`-style interface, so that it may be used with any other programming language. Bindings for Python are already available and shipped with `libmaxdiv`. For non-spatial, purely temporal time-series, a convenient graphical user interface (GUI) is provided to facilitate experimentation with your data without having to write a single line of code.

This document is primarily intended as installation guide for `libmaxdiv` and user manual for the GUI. If you want to know about the details of the MDI algorithm, please consult the following article:

> *Erik Rodner, Björn Barz, Yanira Guanche, Milan Flach, Miguel Mahecha, Paul Bodesheim, Markus Reichstein, Joachim Denzler. "Maximally Divergent Intervals for Anomaly Detection". ICML Workshop on Anomaly Detection (ICML-WS). 2016.*

If you want to use `libmaxdiv` programmatically in another application, take a look at the library interface documentation in the file `maxdiv/libmaxdiv/libmaxdiv.h` of the source code.

2. INSTALLATION

If you haven't done this already, download an archive with the source code of `libmaxdiv` from GITHUB-URL-GOES-HERE and extract it to a directory of your choice.

The installation of all required dependencies of the software and the compilation of the software itself from the source code depends on your operating system. Windows users can skip the next paragraph.

FOR LINUX

For building libmaxdiv, you will need a `C++` compiler capable of `C++11` (e.g., `g++` >= 4.9) and CMake >= 3.1, which you can download here: https://cmake.org/

After that, open a terminal and switch to the directory `maxdiv/libmaxdiv` in the source tree of libmaxdiv. There, run the following commands to download the `Eigen` library and to compile `libmaxdiv`:

```
wget http://bitbucket.org/eigen/eigen/get/3.2.9.tar.gz
tar -xzf 3.2.9.tar.gz && rm 3.2.9.tar.gz
mv eigen-eigen-dc6cfdf9bcec eigen-3.2.9
mkdir bin && cd bin
cmake -D CMAKE_CXX_FLAGS=-I../eigen-3.2.9 ..
make
```

This should compile the `libmaxdiv` library and the command line interface `maxdiv_cli`.

Make sure to use the version of `CMake` installed before and not the system's version which may be outdated. So, you may have to replace "cmake" with the path to your personal `CMake` binary.

To run the graphical user interface, you will need Python. A Python interpreter usually belongs to the default tools of the Linux distributions. Just make sure, that it is at least version 2.7 or higher (Python 3.x is fine too).

In addition, you need the following Python packages:

- `numpy`
- `matplotlib`
- `scipy`
- `scikit-learn`
- `PIL` or `Pillow`

On distributions derived from Debian, you should be able to install these packages for Python 2 by running the following command in a terminal:

```
sudo apt-get install python-numpy python-matplotlib python-scipy python-sklearn python-pil
```

Use the following command for Python 3:

```
sudo apt-get install python3-numpy python3-matplotlib python3-scipy python3-sklearn python3-pil
```

After that, just run the Python script `launch-gui.py` located in the root directory of the `libmaxdiv` source tree. On some systems, a double-click on that file may suffice, otherwise you may need to run the following command from that directory:

```
python launch-gui.py
```

FOR WINDOWS

We provide pre-compiled binaries of the `libmaxdiv` library and the `maxdiv_cli` command line interface for Microsoft Windows, so you don't have to compile the source code by yourself.

However, to run the graphical user interface, you will need Python version 2.7 or higher (Python 3.x is fine too). You can download an installer for Python from https://www.python.org/. Make sure to install a version of python that matches your system architecture, i.e., 64-bit Python for 64-bit Windows and 32-bit Python for 32-bit Windows. You can find out which version of Windows you're using by going to the System Control Panel and then "System".

After that, you need to install the following Python packages:

- `numpy`
- `matplotlib`
- `scipy`
- `scikit-learn`
- `PIL` or `Pillow`

Recent versions of python should allow you to do this from the command line by running:

```
pip install numpy matplotlib scipy scikit-learn pillow
```

If that does not work for you, you can find pre-built binaries for each package at http://www.lfd.uci.edu/~gohlke/pythonlibs/. For each of the packages mentioned above, download the corresponding `.whl` file from that site that fits your Python version and system architecture. You can then install those files by running `pip install <path-to-file>`, where `<path-to-file>` has to be replaced with the full path of the downloaded package file you would like to install.

After having installed Python and the packages mentioned above, just run `launch-gui.py` in the `win32` directory located in the root directory of the `libmaxdiv` source tree. Don't confuse this with the `launch-gui.py` file located directly in the root directory! On some systems, a double-click on that file may suffice, otherwise you may need to run the following command from the command line in the `win32` directory:

```
python launch-gui.py
```

3. USING THE GRAPHICAL USER INTERFACE

3.1. LOADING DATA

Immediately after launching the GUI you will be asked to select a file with time-series data. That file has to be in the CSV format, which stands for "comma-separated values". That means, it is a simple text file with each line containing the values of the attributes measured at a specific time-step (with the lines being in chronological order). Those values are separated by commas, but different delimiters such as semicolons should also be detected by the application automatically. Most data management tools should be able to export data as CSV files (e.g., Microsoft Excel).

The first attribute of the data may be a continuous list of timestamps which may also be given as date and time in the following format: `YYYY-MM-DD HH:MM:SS` or `YYYY-MM-DD`

The first line in the file may be a comma-separated list of attribute names.

Missing values in the data can be encoded as "nan" (without the quotes).

If there is more than one attribute, the application will allow you to select a subset of the attributes and/or a sub-range of the time-series to be loaded after you have selected the file.

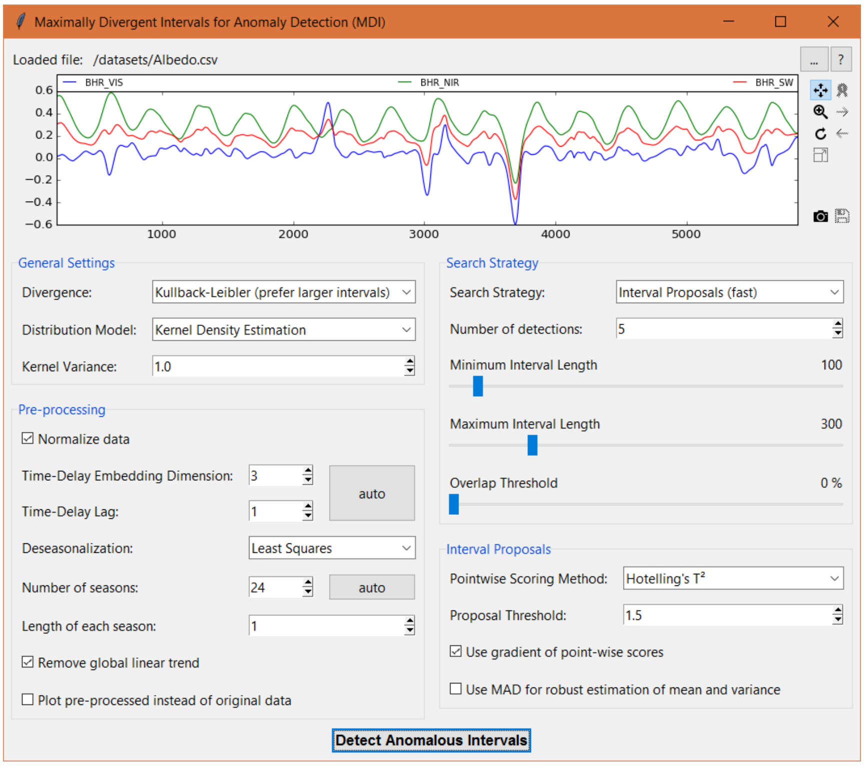
Note that though the `libmaxdiv` library does also support *spatio*-temporal data, the capabilities of the GUI are limited to purely temporal time-series for the time being.

3.2. PARAMETERS

The GUI provides numerous options (see Figure 1) which influence the behavior of the algorithm and can have significant impact on the resulting detections. This section gives a short explanation for each configuration option:

General Settings

- **Divergence:** The divergence measure to be used for comparing the distributions of the data in each possible interval with the rest of the time-series. The following divergence measures are available:
    - <u>Kullback-Leibler (prefer larger intervals):</u> The original Kullback-Leibler divergence (see below) has a systematic bias that prefers smaller intervals. This variant, called the "unbiased Kullback-Leibler divergence" avoids this deficiency by multiplying the KL divergence with the length of the interval, but slightly tends to prefer larger intervals.


Maximally Divergent Intervals for Anomaly Detection (MDI)
...
?
BHR_VIS
BHR_NIR
BHR_SW
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
1000
2000
3000
4000
5000
auto
auto
Detect Anomalous Intervals

Loaded file: /datasets/Albedo.csv

General Settings

Divergence: Kullback-Leibler (prefer larger intervals)

Distribution Model: Kernel Density Estimation

Kernel Variance: 1.0

Pre-processing

☑ Normalize data

Time-Delay Embedding Dimension: 3

Time-Delay Lag: 1

Deseasonalization: Least Squares

Number of seasons: 24

Length of each season: 1

☑ Remove global linear trend

☐ Plot pre-processed instead of original data

Search Strategy

Search Strategy: Interval Proposals (fast)

Number of detections: 5

Minimum Interval Length 100

Maximum Interval Length 300

Overlap Threshold 0 %

Interval Proposals

Pointwise Scoring Method: Hotelling's T^2

Proposal Threshold: 1.5

☑ Use gradient of point-wise scores

☐ Use MAD for robust estimation of mean and variance

FIGURE 1: SCREENSHOT OF THE GUI WITH ALL AVAILABLE PARAMETERS. NOTE THAT THE CONFIGURATION SHOWN IN THIS SCREENSHOT IS ARBITRARY AND BY NO MEANS A RECOMMENDATION.

  - Kullback-Leibler (prefer smaller intervals): The original Kullback-Leibler divergence. Note that this divergence has a systematic bias towards small intervals, which often results in multiple contiguous detection for a single anomaly. Thus, the “unbiased KL divergence” (see above) is usually a better alternative.
  - Cross Entropy (prefer larger intervals): The cross entropy between the two distributions, multiplied with the size of the interval. In contrast to the “unbiased KL divergence” (see above), there is no theoretic justification for doing so and the results are usually strongly biased towards large intervals. Thus, the ordinary cross entropy (see below) is usually a better alternative.
  - Cross Entropy (prefer smaller intervals): The cross entropy between the two distributions. If the entropy of the data does not vary significantly over time, this might be superior to the Kullback-Leibler divergence when combined with the Gaussian distribution model (see next parameter).
  - Jensen-Shannon: The Jensen-Shannon divergence is an extension of the Kullback-Leibler divergence, which is symmetric and bounded, so that anomaly scores can’t get infinitely high. However, it is often inferior in practice when combined with Gaussian models (see next parameter), but may be useful for other distribution models. Yet, the computation will be comparatively slow.

- **Distribution Model:** The method used to model the distribution of the data in each interval and the rest of the time series. The following models are available:
  - Gaussian (Full Covariance): A multivariate normal distribution. This usually works best, but should not be combined with the Jensen-Shannon divergence.
  - Gaussian (Global Covariance): A multivariate normal distribution with a global, shared covariance matrix. This restriction might be useful for data with many attributes if the intervals of interest are comparatively small.
  - Gaussian (Identity Covariance): A multivariate normal distribution with an identity covariance matrix. This will result in a simple comparison of the mean vectors of a given interval and the remainder of the time-series.

- ○ Kernel Density Estimation (KDE): The probability density of all samples in an interval $I = [a, b]$ is estimated according to $p_I(x_t) = \frac{1}{|I|}\sum_{t' \in I} k(x_t, x_{t'})$ with a Gaussian kernel $k(x, y) = (2\pi\sigma^2)^{-\frac{d}{2}} \cdot \exp(-\frac{\|x-y\|^2}{2\sigma^2})$.
- ○ Ensemble of Random Projection Histograms (ERPH): This method is designed for extremely high-dimensional data with hundreds of attributes. It generates a fixed number of random projections of the data onto 1-dimensional sub-spaces, computes a histogram for each projection and estimates the probability density as the geometric mean of the 1-d probability densities. In practice, however, it is often better to reduce the dimensionality of the data using PCA and employ a Gaussian model.

- **Kernel Variance:** The value of the parameter σ^2 used by KDE (see above).
- **Histograms:** The number of random projections to be used by the ERPH model (see above).
- **Bins:** The number of bins for each histogram in the ERPH model (see above). This parameter can be set to 0 for an individual and automatic determination of the number of bins by maximizing penalized log-likelihood, but we have found fixed small numbers of bins to often give better results.

Pre-processing

- **Normalize data:** If checked, the data will be normalized first by attribute-wise subtraction of the mean and division by the maximum value.
- **Time-Delay Embedding Dimension & Time-Delay Lag:** In order to take the context of each sample into account, time-delay embedding is applied as a pre-processing step. It transforms a time-series $(x_t)_{t=1}^{n}, x_t \in \mathbb{R}^d$, into another time-series by stacking shifted copies of it together, so that each sample incorporates the values of some samples at previous time-steps:
$$x'_t = (x_t \quad x_{t-\tau} \quad x_{t-2\tau} \quad \cdots \quad x_{t-(\kappa-1)\tau})$$
The parameter κ is called *embedding dimension* and specifies the number of samples to stack together, while the *time lag* τ defines the distance between each time-step to be aggregated as context. An example for $\kappa = 3, \tau = 4$ is shown in Figure 2.
The number of attributes in the resulting time-series is κd, i.e., a high embedding dimension does not only increase the information available at each time-step, but also the data size and, thus, memory consumption and processing time. The time lag may be used to take a wider range of context into account without increasing the embedding dimension too far, but a high time lag runs the risk of missing relevant context.
In general, there is no rule of thumb for choosing these two parameters, though the output of the MDI algorithm is very sensitive to them. One could choose the embedding dimension as high as affordable and adjust the time lag accordingly to include a sufficient amount of context, but we suggest experimenting with different configurations.
The button "auto" will try to find good values automatically by considering the speed of the decrease of mutual information when the distance between time-steps is increased, but that method often yields sub-optimal values for the parameters.

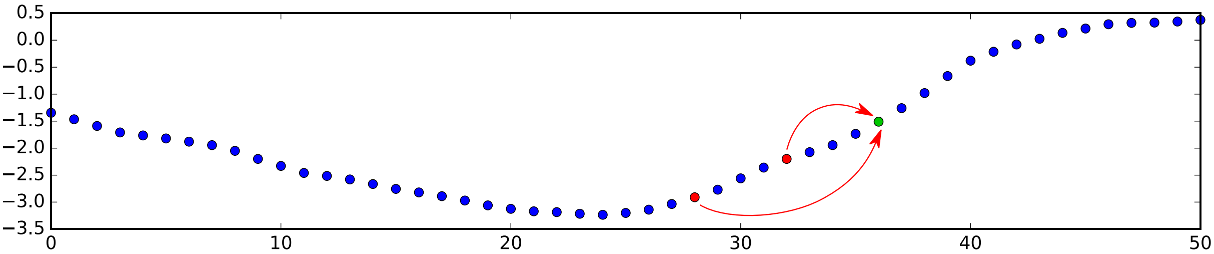


FIGURE 2: A VISUALIZATION OF TIME-DELAY EMBEDDING WITH $\kappa = 3, \tau = 4$.

- **Deseasonalization:** In many domains, the distribution of data is subject to periodic changes due to seasonal or diurnal patterns, which may disturb the algorithm and disguise the actual anomalies. `libmaxdiv` provides three methods for removing such seasonal patterns. Which of them works best usually depends on the data. Moreover, deseasonalization might not be necessary at all even if there is a seasonal pattern, depending on the application and the objective.
    - Z-Score: The data is divided into seasonal groups that are normalized separately.
    - Least Squares: A time-series $(x_t)_{t=1}^{n}, x_t \in \mathbb{R}^d$, is modelled according to
      $$x_t = a_0 + b_0 \cdot t + a_{s(t)} + b_{s(t)} \cdot t + x_t', \qquad t = 1, \dots, n,$$
      where $s(t)$ denotes the season that the time-step t belongs to. Every term of that model has a specific interpretation: a_0 is an overall global offset, $b_0 \cdot t$ is a global linear trend, $a_{s(t)}$ is the offset specific to the respective season, $b_{s(t)} \cdot t$ is a linear trend of that seasonal effect, and the residuals x_t' make up the deseasonalized time-series.
    - Fourier Transform: The time-series is transformed into the frequency domain using the Discrete Fourier Transform. Then, Fourier coefficients with an absolute value greater than 3 standard deviations above the mean are set to zero in order to remove frequencies belonging to seasonal effects from the Fourier spectrum. The inverse transformation of the modified spectrum gives the deseasonalized time-series.

      This method can handle multiple overlapping seasonal effects and does not require any prior knowledge about the seasonality, as opposed to the two methods described above. In turn, it may result in artifacts, especially at the borders, due to the hard removal of frequencies and the periodicity assumption of the Fourier transform.
- **Number of Seasons:** The number of seasonal groups for the Z-Score and the Least Squares deseasonalization method. For example, a value of 24 would be appropriate four hourly data with a diurnal pattern.

  The button “auto” tries to detect the number of seasons automatically by inspecting the frequency domain of the data.
- **Length of each season:** The length of each season for the Least Squares deseasonalization method. This allows multiple consecutive time-steps to be assigned to the same season. For example, if the data has a diurnal pattern and a temporal resolution of 1 minute, one may set the length of each season to 60 and the number of seasons to 24.
- **Remove global linear trend:** Specifies whether to include the term $a_0 + b_0 \cdot t$ in the Least Squares model for deseasonalization.
- **Plot pre-processed instead of original data:** If this is checked, the plot in the GUI will show the data after normalization and deseasonalization have been applied instead of the original data loaded from the file. This is useful for getting an insight into the effect of different deseasonalization parameters. This flag does not change the behavior of the actual algorithm at all.

Search Strategy

- **Search Strategy:**
    - Full Scan: Check every possible interval for an anomaly. This is accurate, but might take some time if the time-series is long.
    - Interval Proposals: Only look at “interesting” intervals. Which intervals are interesting is determined based on a fast point-wise anomaly detection method. Since only a subset of intervals is checked in detail, this strategy leads to a significant speed-up, but might also miss an anomaly. In our experience, completely missing an anomaly is rare, but often the localization of the anomalies is not as accurate as with a full scan.
- **Number of detections:** The number of highest-scoring detections to be visualized.
- **Minimum & Maximum Interval Length:** The minimum and maximum length of the intervals to be checked. If application-specific knowledge about the size of the anomalies to expect is available, this can be used to avoid checking too small or too large intervals in order to decrease

computation time.
The minimum interval length cannot be smaller than 10 time-steps, since a robust estimation of the probability distribution of the data would not be possible otherwise. If the data contain more than one attribute, we recommend even larger values.
The maximum length of the intervals is restricted to be at most ¼ of the length of the time-series.

- **Overlap Threshold:** The maximum overlap allowed between detected intervals.

Interval Proposals (only visible if "Search Strategy" is set to "Interval Proposals")

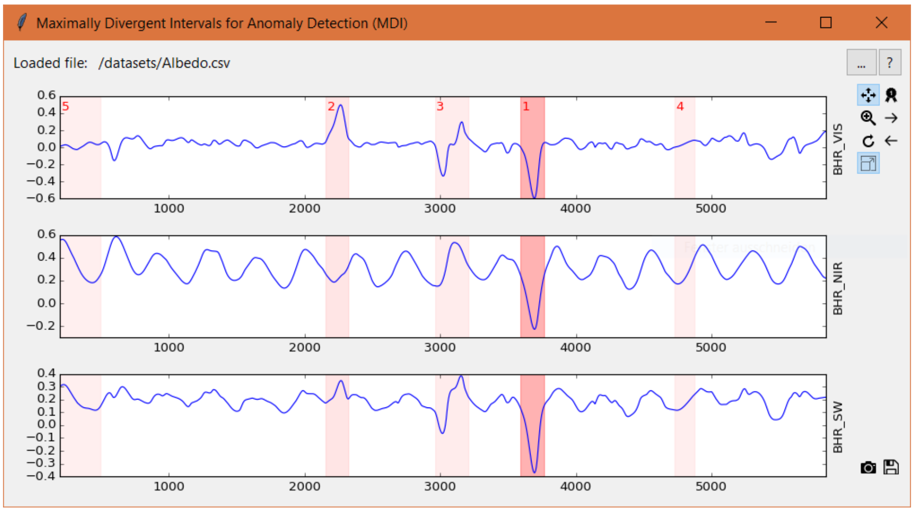
- **Pointwise Scoring Method:** The point-wise scoring method to be used for the generation of interval proposals. Possible choices are Hotelling's T^2 and point-wise KDE. The former one usually works better.
- **Proposal Threshold:** A threshold which controls how many proposals are generated. Higher thresholds will increase the speed of the algorithm, but also the number of anomalies that are not detected, though they would be detected by a full scan.
- **Use gradient of point-wise scores:** If checked, the gradient of the point-wise scores will be used for proposal generation instead of the original scores. This should be used for the point-wise KDE method. For Hotelling's T^2, it probably does not make a significant difference.
- **Use MAD for robust estimation of mean and variance:** The threshold operation for proposal generation is based on the mean and the variance of the point-wise scores. If this option is checked, the median and the median absolute deviation above median (MAD) are used instead.

3.3. INSPECTING THE DETECTIONS

After having loaded the data and configured the parameters, the algorithm can be run by clicking on the button “Detect Anomalous Intervals”. This can take some time. After it has finished, the selected number of detections will be visualized in the plot at the top of the window as filled rectangles. The intensity of the red fill color of the detections corresponds to their anomalousness determined by the algorithm. In addition, the rank of the top 9 detections is shown in the top-left corner of the interval.

To inspect and export the results of the algorithm, you can interact with the plot using the buttons right next to it:

- ✥ Switch to *pan mode*. In this mode, you may click on the plot and hold the left mouse button to move it around (assuming it has been zoomed before, so that the entire time-series does not fit in the frame).
- 🔍 Switch to *zoom mode*. In this mode, you may left-click on a point in the plot to zoom in, while a right-click will zoom out.
- ↻ Reset the plot to show the entire time-series.
- 🏅 Zoom in to the highest-scoring detection.
- → Zoom in to the next detection.
- ← Zoom in to the previous detection.
- ⧉ Switch between *normal view* and *expanded view*. In expanded view, the parameter configuration is hidden and each dimension is plotted separately (see Figure 3).
- 📷 Save a snapshot of the current plot as image file.
- 💾 Export the detections returned by the algorithm as CSV file. The three values in each line of the file will be the timestamp of the first and the last point inside of the anomalous interval, followed by its anomaly score.



Maximally Divergent Intervals for Anomaly Detection (MDI)

Loaded file: /datasets/Albedo.csv

FIGURE 3: SCREENSHOT OF THE GUI IN EXPANDED VIEW SHOWING THE TOP 5 DETECTIONS.