Python Documentation

version

October 20, 2020

Contents

Welcome to moseq2-pca's documentation!

moseq2_pca package

CLI Module

moseq2-pca

```
moseq2-pca [OPTIONS] COMMAND [ARGS]...
```

Options

--version
Show the version and exit. [default: False]

apply-pca

Computes PCA Scores of extraction data given a pre-trained PCA

```
moseq2-pca apply-pca [OPTIONS]
```

Options

--chunk-size <chunk_size>
Number of frames per chunk [default: 4000]

--h5-mask-path <h5_mask_path>
Path to log-likelihood mask in h5 files [default: /frames_mask]

--h5-path <h5_path>
Path to data in h5 files [default: /frames]

--config-file <config_file>
Path to configuration file

-o, --output-dir <output_dir>
Directory to store results [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs/_pca]

-i, --input-dir <input_dir>
Directory to find h5 files [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs]

--cluster-type <cluster_type>
Cluster type [default: local]

Options: local|slurm|nodask

--timeout <timeout>
Time to wait for workers to initialize before proceeding (minutes) [default: 5]

-w, --wall-time <wall_time>
Wall time for workers [default: 06:00:00]

-m, --memory <memory>
Total RAM usage per worker [default: 15GB]

-p, --processes <processes>
Number of processes to run on each worker [default: 1]

-c, --cores <cores>
Number of cores per worker [default: 1]

-n, --nworkers <nworkers>
Number of workers [default: 10]

-q, --queue <queue>
Cluster queue/partition for submitting jobs [default: debug]

--dask-port <dask_port>
Port to access dask dashboard [default: 8787]

-d, --dask-cache-path <dask_cache_path>
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2_pca]

--output-file <output_file>
Name of h5 file for storing pca results [default: pca_scores]

--pca-path <pca_path>
Path to pca components [default: /components]

--pca-file <pca_file>
Path to PCA results

--fill-gaps <fill_gaps>
Fill dropped frames with nans [default: True]

--fps <fps>
Fps (only used if no timestamps found) [default: 30]

--detrend-window <detrend_window>
Length of detrend window (in seconds, 0 for no detrending) [default: 0]

-v, --verbose
Print sessions as they are being loaded. [default: False]

clip-scores

Clips specified number of frames from PCA scores at the beginning or end

```
moseq2-pca clip-scores [OPTIONS] PCA_FILE CLIP_SAMPLES
```

Options

--from-end
[default: False]

Arguments

PCA_FILE
Required argument

CLIP_SAMPLES
Required argument

compute-changepoints

Computes the Model-Free Syllable Changepoints based on the PCA/PCA_Scores

```
moseq2-pca compute-changepoints [OPTIONS]
```

Options

--chunk-size <chunk_size>
Number of frames per chunk [default: 4000]

--h5-mask-path <h5_mask_path>
Path to log-likelihood mask in h5 files [default: /frames_mask]

--h5-path <h5_path>
Path to data in h5 files [default: /frames]

--config-file <config_file>
Path to configuration file

-o, --output-dir <output_dir>
Directory to store results [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs/_pca]

-i, --input-dir <input_dir>
Directory to find h5 files [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs]

--cluster-type <cluster_type>
Cluster type [default: local]

Options: local|slurm|nodask

--timeout <timeout>
Time to wait for workers to initialize before proceeding (minutes) [default: 5]

-w, --wall-time <wall_time>
Wall time for workers [default: 06:00:00]

-m, --memory <memory>
Total RAM usage per worker [default: 15GB]

-p, --processes <processes>
Number of processes to run on each worker [default: 1]

-c, --cores <cores>
Number of cores per worker [default: 1]

-n, --nworkers <nworkers>
Number of workers [default: 10]

-q, --queue <queue>
Cluster queue/partition for submitting jobs [default: debug]

--dask-port <dask_port>
Port to access dask dashboard [default: 8787]

-d, --dask-cache-path <dask_cache_path>
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2_pca]

--output-file <output_file>
Name of h5 file for storing pca results [default: changepoints]

--pca-file-components <pca_file_components>
Path to PCA components

--pca-file-scores <pca_file_scores>
Path to PCA results

--pca-path <pca_path>
Path to pca components [default: /components]

--neighbors <neighbors>
Neighbors to use for peak identification [default: 1]

--threshold <threshold>
Peak threshold to use for changepoints [default: 0.5]

-k, --klags <klags>
Lag to use for derivative calculation [default: 6]

-s, --sigma <sigma>
Standard deviation of gaussian smoothing filter [default: 3.5]

-d, --dims <dims>
Number of random projections to use [default: 300]

--fps <fps>
Fps (only used if no timestamps found) [default: 30]

-v, --verbose
Print sessions as they are being loaded. [default: False]

train-pca

Trains PCA on all extracted results (h5 files) in input directory

```
moseq2-pca train-pca [OPTIONS]
```

Options

--chunk-size <chunk_size>
Number of frames per chunk [default: 4000]

--h5-mask-path <h5_mask_path>
Path to log-likelihood mask in h5 files [default: /frames_mask]

--h5-path <h5_path>
Path to data in h5 files [default: /frames]

--config-file <config_file>
Path to configuration file

-o, --output-dir <output_dir>
Directory to store results [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs/_pca]

-i, --input-dir <input_dir>
Directory to find h5 files [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs]

--cluster-type <cluster_type>
Cluster type [default: local]

Options: local|slurm|nodask

--timeout <timeout>
Time to wait for workers to initialize before proceeding (minutes) [default: 5]

-w, --wall-time <wall_time>
Wall time for workers [default: 06:00:00]

-m, --memory <memory>
Total RAM usage per worker [default: 15GB]

-p, --processes <processes>
Number of processes to run on each worker [default: 1]

-c, --cores <cores>
Number of cores per worker [default: 1]

-n, --nworkers <nworkers>
Number of workers [default: 10]

-q, --queue <queue>
Cluster queue/partition for submitting jobs [default: debug]

--dask-port <dask_port>
Port to access dask dashboard [default: 8787]

-d, --dask-cache-path <dask_cache_path>
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2_pca]

--gaussfilter-space <gaussfilter_space>
Spatial filter for data (Gaussian) [default: 1.5, 1]

--gaussfilter-time <gaussfilter_time>
Temporal filter for data (Gaussian) [default: 0]

--medfilter-space <medfilter_space>
Median spatial filter [default: 0]

--medfilter-time <medfilter_time>
Median temporal filter [default: 0]

--missing-data
Use missing data PCA [default: False]

--missing-data-iters <missing_data_iters>
Missing data PCA iterations [default: 10]

--mask-threshold <mask_threshold>
Threshold for mask (missing data only) [default: -16]

--mask-height-threshold <mask_height_threshold>

Threshold for mask based on floor height [default: 5]

--min-height <min_height>
Min mouse height from floor (mm) [default: 10]

--max-height <max_height>
Max mouse height from floor (mm) [default: 100]

--tailfilter-size <tailfilter_size>
Tail filter size [default: 9, 9]

--tailfilter-shape <tailfilter_shape>
Tail filter shape [default: ellipse]

--use-fft
Use 2D fft [default: False]

--recon-pcs <recon_pcs>
Number of PCs to use for missing data reconstruction [default: 10]

--rank <rank>
Rank for compressed SVD (generally>>nPCS) [default: 25]

--output-file <output_file>
Name of h5 file for storing pca results [default: pca]

--local-processes <local_processes>
Used with a local cluster. If True: use processes, If False: use threads [default: False]

GUI Module

GUI front-end operations. This module contains all the functionality and configurable parameters users can alter to most accurately process their data.

Note: These functions perform jupyter notebook specific preprocessing, loads in corresponding parameters from the CLI functions, then call the corresponding wrapper function with the given input parameters.

moseq2_pca.gui.**apply_pca_command** (progress_paths, output_file)
Compute PCA Scores given trained PCA using Jupyter Notebook.

Parameters:
- **progress_paths (dict)** (*dictionary containing notebook progress paths*)
- **output_file (str)** (*name of output pca file.*)

Returns: **(str)**

Return type: success string.

moseq2_pca.gui.**compute_changepoints_command** (input_dir, progress_paths, output_file)
Compute Changepoint distribution using Jupyter Notebook.

Parameters:
- **input_dir (str)** (*path to directory containing training data*)
- **progress_paths (dict)** (*dictionary containing notebook progress paths*)
- **output_file (str)** (*name of output pca file.*)

Returns: **(str)**

Return type: success string.

moseq2_pca.gui.**train_pca_command** (progress_paths, output_dir, output_file)
Train PCA through Jupyter notebook, and updates config file.

Parameters:
- **progress_paths (dict)** (*dictionary containing notebook progress paths*)
- **output_dir (str)** (*path to output pca directory*)
- **output_file (str)** (*name of output pca file.*)

Returns:

Return type: None

Utilities Module

Utility and helper functions for traversing directories to find and read files, filtering operations,

Dask initialization, and changepoint helper functions.

`moseq2_pca.util.`**`check_timestamps`** (`h5s`)

Helper function to determine whether timestamps and/or metadata is missing from extracted files. Function will emit a warning if either pieces of data are missing.

Parameters: **h5s (list)** (*List of paths to all extracted h5 files.*)

Returns:

Return type: None

`moseq2_pca.util.`**`clean_frames`** (`frames`, `medfilter_space=None`, `gaussfilter_space=None`, `medfilter_time=None`, `gaussfilter_time=None`, `detrend_time=None`, `tailfilter=None`, `tail_threshold=5`)

Filters spatial/temporal noise from frames using Median and Gaussian filters, given kernel sizes for each respective requested filter.

Parameters:

- **frames (3D numpy array)** (*frames to filter.*)
- **medfilter_space (list)** (*median spatial filter kernel.*)
- **gaussfilter_space (list)** (*gaussian spatial filter kernel.*)
- **medfilter_time (list)** (*median temporal filter.*)
- **gaussfilter_time (list)** (*gaussian temporal filter.*)
- **detrend_time (int)** (*number of frames to lag for.*)
- **tailfilter (int)** (*size of tail-filter kernel.*)
- **tail_threshold (int)** (*threshold value to use for tail filtering*)

Returns: **out (3D numpy array)**

Return type: filtered frames.

`moseq2_pca.util.`**`close_dask`** (`client`, `cluster`, `timeout`)

Shuts down the Dask client and cluster. Dumps all cached data.

Parameters:

- **client (Dask Client)** (*Client object*)
- **cluster (Dask Cluster)**
- **timeout (int)** (*Time to wait for client to close gracefully (minutes)*)

Returns:

Return type: None

`moseq2_pca.util.`**`command_with_config`** (`config_file_param_name`)

`moseq2_pca.util.`**`gauss_smooth`** (`signal`, `win_length=None`, `sig=1.5`, `kernel=None`)

Perform Gaussian Smoothing on a 1D signal.

Parameters:

- **signal (1d numpy array)** (*signal to perform smoothing*)
- **win_length (int)** (*window_size for gaussian kernel filter*)
- **sig (float)** (*variance of 1d gaussian kernel.*)
- **kernel (tuple)** (*kernel size to use for smoothing*)

Returns: **result (1d numpy array)**

Return type: smoothed signal

`moseq2_pca.util.`**`gaussian_kernel1d`** (`n=None`, `sig=3`)

Get 1D gaussian kernel.

Parameters:
- **n (int)** (*number of points to use.*)
- **sig (int)** (*variance of kernel to use.*)

Returns: **kernel (1d array)**

Return type: 1D numpy kernel.

`moseq2_pca.util.`**`get_changepoints`** (`scores`, `k=5`, `sigma=3`, `peak_height=0.5`, `peak_neighbors=1`, `baseline=True`, `timestamps=None`)

Compute changepoints distribution and CP Curve.

Parameters:
- **scores (3D numpy array)** (*nframes * r * c*)
- **k (int)** (*klags - Lag to use for derivative calculation.*)
- **sigma (int)** (*Standard deviation of gaussian smoothing filter.*)
- **peak_height (float)** (*user-defined peak Changepoint length.*)
- **peak_neighbors (int)** (*number of peaks in the CP curve.*)
- **baseline (bool)** (*normalize data.*)
- **timestamps (array)** (*loaded timestamps.*)

Returns: **cps (2D numpy array)** (*array of values for CP curve*) **normed_df (1D numpy array)** (*array of values for bar plot*)

`moseq2_pca.util.`**`get_env_cpu_and_mem`** ()

Reads current system environment and returns the amount of available memory and CPUs to allocate to the created cluster.

Returns: **mem (float)** (*Optimal number of memory (in bytes) to allocate to initialized dask cluster*) **cpu (int)** (*Optimal number of CPUs to allocate to dask*)

`moseq2_pca.util.`**`get_metadata_path`** (`h5file`)

Return path within h5 file that contains the kinect extraction metadata.

Parameters: **h5file (str)** (*path to h5 file.*)

Returns: **(str)**

Return type: path to acquistion metadata within h5 file.

`moseq2_pca.util.`**`get_rps`** (`frames`, `rps=600`, `normalize=True`)

Get random projections of frames.

Parameters:
- **frames (2D or 3D numpy array)** (*Frames to get dimensions from.*)
- **rps (int)** (*Number of random projections.*)
- **normalize (bool)** (*indicates whether to normalize frames.*)

Returns: **rproj (2D or 3D numpy array)**

Return type: Computed random projections with same shape as frames

`moseq2_pca.util.`**`get_timestamp_path`** (`h5file`)

Return path within h5 file that contains the kinect timestamps

Parameters: **h5file (str)** (*path to h5 file.*)

Returns: **(str)**

Return type: path to metadata timestamps within h5 file

`moseq2_pca.util.`**`h5_to_dict`** (`h5file`, `path`)

Reads all contents from h5 and returns them in a nested dict object.

Parameters:
- **h5file (str)** (*path to h5 file*)
- **path (str)** (*path to group within h5 file*)

Returns: **ans (dict)**

Return type: dictionary of all h5 group contents

moseq2_pca.util.**initialize_dask**(nworkers=50, processes=1, memory='4GB', cores=1, wall_time='01:00:00', queue='debug', local_processes=False, cluster_type='local', timeout=10, cache_path='/Users/aymanzeine/moseq2_pca', dashboard_port='8787', data_size=None, **kwargs)

Initialize dask client, cluster, workers, etc.

Parameters:
- **nworkers (int)** (*number of dask workers to initialize*)
- **processes (int)** (*number of processes per worker*)
- **memory (str)** (*amount of memory to allocate to dask cluster*)
- **cores (int)** (*number of cores to use.*)
- **wall_time (str)** (*amount of time to allow program to run*)
- **queue (str)** (*logging mode*)
- **local_processes (bool)** (*flag to use processes or threads when using a local cluster*)
- **cluster_type (str)** (*indicate what cluster to use (local or slurm)*)
- **scheduler (str)** (*indicate what scheduler to use*)
- **timeout (int)** (*how many minutes to wait for workers to initialize*)
- **cache_path (str or Pathlike)** (*path to store cached data*)
- **dashboard_port (str)** (*port number to find dask statistics*)
- **data_size (float)** (*size of the dask array in number of bytes.*)
- **kwargs** (*extra keyward arguments*)

Returns: **client (dask Client)** (*initialized Client*) **cluster (dask Cluster)** (*initialized Cluster*) **workers (dask Workers)** (*intialized workers*)

moseq2_pca.util.**insert_nans**(timestamps, data, fps=30)

Fills NaN values with 0 in timestamps.

Parameters:
- **timestamps (1D array)** (*timestamp time-strs*)
- **data (1D array)** (*timestamp values*)
- **fps (int)** (*frames per second*)

Returns: **filled_data (1D array)** (*filled missing timestamp values.*) **data_idx (1D array)** (*indices of inserted 0s*) **filled_timestamps (1D array)** (*filled timestamp-strs*)

moseq2_pca.util.**read_yaml**(yaml_file)

Reads yaml file and returns dictionary representation of file contents.

Parameters: **yaml_file (str)** (*path to yaml file*)

Returns: **return_dict (dict)**

Return type: dict of yaml file contents

moseq2_pca.util.**recursive_find_h5s**(root_dir='/Users/aymanzeine/Desktop/moseq/moseq2-pca/docs', ext='.h5', yaml_string='{}.yaml')

Recursively find h5 files, along with yaml files with the same basename

Parameters:
- **root_dir (str or os.Pathlike)** (*path to directory to start recursive search*)
- **ext (str)** (*extension to search for, e.g. .h5*)
- **yaml_string (str)** (*a format to use to name yaml files*)

Returns: **h5s (list)** (*list of h5 file paths*) **dicts (list)** (*list of dicts containing metadata file contents*) **yamls (list)** (*list of yaml file paths*)

moseq2_pca.util.**select_strel**(string='e', size=(10, 10))

Selects Structuring Element Shape

Parameters:
- **string (str)** (*e for Ellipse, r for Rectangle*)
- **size (tuple)** (*size of StructuringElement*)

Returns: **strel (cv2.StructuringElement)**

Return type: returned StructuringElement with specified size.

`moseq2_pca.util.`**`set_dask_config`**(`memory={'pause': False, 'spill': False, 'target': 0.85, 'terminate': 0.95}`)

Sets initial dask configuration parameters

Parameters: **memory (dict)**

Visualization Module

Visualization operations for plotting computed PCs, a Scree Plot, and the Changepoint PDF histogram.

`moseq2_pca.viz.`**`changepoint_dist`**(`cps`, `headless=False`)

Creates bar plot describing computed Changepoint Distribution.

Parameters:
- **cps (np.ndarray)** (*changepoints to graph*)
- **headless (bool)** (*trim first element in PC list*)

Returns: **plt (plt.figure)** (*figure to save/graph*) **ax (plt.ax)** (*figure axis variable*)

`moseq2_pca.viz.`**`display_components`**(`components`, `cmap='gray'`, `headless=False`)

Creates grid of computed Principal Components.

Parameters:
- **components (2D np.ndarray)** (*components to graph*)
- **cmap (str)** (*color map to use*)
- **headless (bool)** (*trim first element in PC list*)

Returns: **plt (plt.figure)** (*figure to save/graph*) **ax (plt.ax)** (*figure axis variable*)

`moseq2_pca.viz.`**`plot_pca_results`**(`output_dict`, `save_file`, `output_dir`)

Convenience function to graph and save Trained PCA results.

Parameters:
- **output_dict (dict)** (*Dict object containing PCA training results*)
- **save_file (str)** (*Path to write images to.*)
- **output_dir (str)** (*Directory containing logger*)

Returns:

Return type: None

`moseq2_pca.viz.`**`scree_plot`**(`explained_variance_ratio`, `headless=False`)

Creates Scree plot describing principal components.

Parameters:
- **explained_variance_ratio (1D np.array)** (*explained variance ratio of each principal component*)
- **headless (bool)** (*trim first element in PC list*)

Returns: **plt (plt.figure)**

Return type: figure to save/graph

Subpackages

moseq2_pca.helpers package

Helpers - Data Module

Helper functions for reading files and directories in preparation for changepoint analysis or apply pca.

`moseq2_pca.helpers.data.`**`get_pca_paths`**(`config_data`, `output_dir`)

Helper function for changepoints_wrapper to perform data-path existence checks. Returns paths to saved pre-trained PCA components and PCA Scores files.

Parameters:
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)

Returns: **config_data (dict)** (*updated config_data dict with the proper paths*) **pca_file_components (str)** (*path to trained pca file*) **pca_file_scores (str)** (*path to pca_scores file*)

`moseq2_pca.helpers.data.`**`get_pca_yaml_data`**(`pca_yaml`)

Reads PCA yaml file and returns metadata

Parameters: **pca_yaml (str)** (*path to pca.yaml*)

Returns: **use_fft (bool)** (*indicates whether to use FFT*) **clean_params (dict)** (*dict of image filtering parameters*) **mask_params (dict)** (*dict of mask parameters)*) **missing_data (bool)** (*indicates whether to use mask_params*)

`moseq2_pca.helpers.data.`**`load_pcs_for_cp`**(`pca_file_components`, `config_data`)

Load computed Principal Components for Model-free Changepoint Analysis.

Parameters:
- **pca_file_components (str)** (*path to pca h5 file to read PCs*)
- **config_data (dict)** (*config parameters*)

Returns: **pca_components (str)** (*path to pca components*) **changepoint_params (dict)** (*dict of relevant changepoint parameters*) **cluster (dask Cluster)** (*Dask Cluster object.*) **client (dask Client)** (*Dask Client Object*) **workers (dask Workers)** (*intialized workers or None if cluster_type = 'local'*) **missing_data (bool)** (*Indicates whether to use mask_params*) **mask_params (dict)** (*Mask parameters to use when computing CPs*)

Helpers - Wrapper Module

Wrapper functions for all functionality included in MoSeq2-PCA that is accessible via CLI or GUI. Each wrapper function executes the functionality from end-to-end given it's dependency parameters are inputted. (See CLI Click parameters)

`moseq2_pca.helpers.wrappers.`**`apply_pca_wrapper`**(`input_dir`, `config_data`, `output_dir`, `output_file`)

Wrapper function to obtain PCA Scores.

Parameters:
- **input_dir (int)** (*path to directory containing all h5+yaml files*)
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)
- **output_file (str)** (*pca model filename*)
- **kwargs (dict)** (*dictionary containing loaded h5s, yamls and dicts found in given input_dir*)

Returns: **config_data (dict)**

Return type: updated config_data variable to write back in GUI API

`moseq2_pca.helpers.wrappers.`**`clip_scores_wrapper`**(`pca_file`, `clip_samples`, `from_end=False`)

Clips PCA scores from the beginning or end. Note that scores are modified *in place*.

Parameters:
- **pca_file (str)** (*Path to PCA scores.*)
- **clip_samples (int)** (*number of samples to clip from beginning or end*)
- **from_end (bool)** (*if true clip from end rather than beginning*)

`moseq2_pca.helpers.wrappers.`**`compute_changepoints_wrapper`**(`input_dir`, `config_data`, `output_dir`, `output_file`)

Wrapper function to compute model-free (PCA based) Changepoints.

Parameters:
- **input_dir (int)** (*path to directory containing all h5+yaml files*)
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)
- **output_file (str)** (*pca model filename*)
- **kwargs (dict)** (*dictionary containing loaded h5s, yamls and dicts found in given input_dir*)

Returns: **config_data (dict)**

Return type: updated config_data variable to write back in GUI API

`moseq2_pca.helpers.wrappers.`**`load_and_check_data`**(`input_dir`, `output_dir`, `changepoints=False`)

Executes initialization functionality that is common among all 3 PCA related operations. Function will load relevant h5 and yaml files found in given input directory, then check for timestamps and warn the user if they are missing.

Parameters:
- **input_dir (str)** (*input directory containing h5 files to find*)
- **output_dir (str)** (*directory name to save pca results*)
- **changepoints (bool)** (*boolean for whether to find data from the aggregate_results directory*)

Returns: **output_dir (str)** (*absolute output directory path*) **h5s (list)** (*list of found h5 files*) **yamls (list)** (*list of corresponding yaml files*) **dicts (list)** (*list of corresponding metadata.json files*)

`moseq2_pca.helpers.wrappers.`**`train_pca_wrapper`**(`input_dir`, `config_data`, `output_dir`, `output_file`)

Wrapper function to train PCA.

Parameters:
- **input_dir (int)** (*path to directory containing all h5+yaml files*)
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)
- **output_file (str)** (*pca model filename*)
- **kwargs (dict)** (*dictionary containing loaded h5s, yamls and dicts found in given input_dir*)

Returns: **config_data (dict)**

Return type: updated config_data variable to write back in GUI API

moseq2_pca.pca package

PCA - Utilties Module

Utility functions for all PCA-related operations.

`moseq2_pca.pca.util.`**`apply_pca_dask`**(`pca_components`, `h5s`, `yamls`, `use_fft`, `clean_params`, `save_file`, `chunk_size`, `mask_params`, `missing_data`, `client`, `fps=30`, `h5_path='/frames'`, `h5_mask_path='/frames_mask'`, `verbose=False`)

"Apply" trained PCA on input frame data to obtain PCA Scores using Distributed Dask cluster.

Parameters:

- **pca_components (np.array)** (*array of computed Principal Components*)
- **h5s (list)** (*list of h5 files*)
- **yamls (list)** (*list of yaml files*)
- **use_fft (bool)** (*indicate whether to use 2D-FFT*)
- **clean_params (dict)** (*dictionary containing filtering options*)
- **save_file (str)** (*path to pca_scores filename to save*)
- **chunk_size (int)** (*size of chunks to process*)
- **mask_params (dict)** (*dictionary of masking parameters (if missing data)*)
- **missing_data (bool)** (*indicates whether to use mask arrays.*)
- **fps (int)** (*frames per second*)
- **h5_path (str)** (*path to frames within selected h5 file (default: '/frames')*)
- **h5_mask_path (str)** (*path to masked frames within selected h5 file (default: '/frames_mask')*)
- **verbose (bool)** (*print session names as they are being loaded.*)

Returns:

Return type: None

`moseq2_pca.pca.util.`**`apply_pca_local`**(`pca_components`, `h5s`, `yamls`, `use_fft`, `clean_params`, `save_file`, `chunk_size`, `mask_params`, `missing_data`, `fps=30`, `h5_path='/frames'`, `h5_mask_path='/frames_mask'`, `verbose=False`)

"Apply" trained PCA on input frame data to obtain PCA Scores using local cluster/platform.

Parameters:

- **pca_components (np.array)** (*array of computed Principal Components*)
- **h5s (list)** (*list of h5 files*)
- **yamls (list)** (*list of yaml files*)
- **use_fft (bool)** (*indicate whether to use 2D-FFT*)
- **clean_params (dict)** (*dictionary containing filtering options*)
- **save_file (str)** (*path to pca_scores filename to save*)
- **chunk_size (int)** (*size of chunks to process*)
- **mask_params (dict)** (*dictionary of masking parameters (if missing data)*)
- **missing_data (bool)** (*indicates whether to use mask arrays.*)
- **fps (int)** (*frames per second*)
- **h5_path (str)** (*path to frames within selected h5 file (default: '/frames')*)
- **h5_mask_path (str)** (*path to masked frames within selected h5 file (default: '/frames_mask')*)
- **verbose (bool)** (*print session names as they are being loaded.*)

Returns:

Return type: None

`moseq2_pca.pca.util.`**`compute_explained_variance`**(`s`, `nsamples`, `total_var`)

Computes the explained variance and explained variance ratio contributed by each computed Principal Component.

Parameters:

- **s (1d array)** (*computed singular values.*)
- **nsamples (int)** (*number of included samples.*)
- **total_var (float)** (*total variance captured by principal components.*)

Returns: **explained_variance (1d-array)** (*list of floats denoting the explained variance per PC.*) **explained_variance_ratio (1d-array)** (*list of floats denoting the explained variance ratios per PC.*)

`moseq2_pca.pca.util.`**`compute_svd`** (`dask_array`, `mean`, `rank`, `iters`, `missing_data`, `mask`, `recon_pcs`, `min_height`, `max_height`, `client`)

Runs Singular Vector Decomposition on the inputted frames of shape (nframes, nfeatures). Data is centered by subtracting it by the mean value of the data. If missing_data == True, It will iteratively recompute the svd on the mean-centered data to reconstruct the PCs from the missing data until it converges.

Parameters:

- **dask_array (dask 2d-array)** (*Reshaped input data array of shape (nframes x nfeatures)*)
- **mean (1d array)** (*Means of each row in dask_array.*)
- **rank (int)** (*Rank of the desired thin SVD decomposition.*)
- **iters (int)** (*Number of SVD iterations*)
- **missing_data (bool)** (*Indicates whether to compute SVD with a masked array*)
- **mask (dask 2d-array)** (*None if missing_data == False, else mask array of shape dask_array*)
- **recon_pcs (int)** (*Number of PCs to reconstruct for missing data.*)
- **min_height (int)** (*Minimum height of mouse above the ground, used to filter reconstructed PCs.*)
- **max_height (int)** (*Maximum height of mouse above the ground, used to filter reconstructed PCs.*)
- **client (dask Client)** (*Dask client to process batches.*)

Returns: **s (1d array)** (*computed singular values (eigen-values).*) **v (2d array)** (*computed principal components (eigen-vectors).*) **mean (1d array)** (*updated mean of dask array if missing_data == True.*) **total_var (float)** (*total variance captured by principal components.*)

`moseq2_pca.pca.util.`**`copy_metadatas_to_scores`** (`f`, `f_scores`, `uuid`)

Copies metadata from individual session extract h5 files to the PCA scores h5 file.

Parameters:

- **f (read-open h5py File)** (*open "results_00.h5" h5py.File object in read-mode*)
- **f_scores (read-open h5py File)** (*open "pca_scores.h5" h5py.File object in read-mode*)
- **uuid (str)** (*uuid of inputted session h5 "f".*)

Returns:

Return type: None

`moseq2_pca.pca.util.`**`get_changepoints_dask`** (`changepoint_params`, `pca_components`, `h5s`, `yamls`, `save_file`, `chunk_size`, `mask_params`, `missing_data`, `client`, `fps=30`, `pca_scores=None`, `progress_bar=False`, `h5_path='/frames'`, `h5_mask_path='/frames_mask'`, `verbose=False`)

Computes model-free changepoints using PCs and PC Scores on distributed dask cluster.

Parameters:
- **changepoint_params (dict)** (*dict of changepoint parameters*)
- **pca_components (np.array)** (*computed principal components*)
- **h5s (list)** (*list of h5 files*)
- **yamls (list)** (*list of yaml files*)
- **save_file (str)** (*path to save changepoint files*)
- **chunk_size (int)** (*size of chunks to process in dask.*)
- **mask_params (dict)** (*dict of missing_data mask parameters.*)
- **missing_data (bool)** (*indicate whether to use mask_params*)
- **client (dask Client)** (*initialized Dask Client object*)
- **fps (int)** (*frames per second*)
- **pca_scores (np.array)** (*computed principal component scores*)
- **progress_bar (bool)** (*display progress bar*)
- **h5_path (str)** (*path to frames within selected h5 file (default: '/frames')*)
- **h5_mask_path (str)** (*path to masked frames within selected h5 file (default: '/frames_mask')*)
- **verbose (bool)** (*print session names as they are being loaded.*)

Returns:

Return type: None

`moseq2_pca.pca.util.`**`get_timestamps`**(`f`, `frames`, `fps=30`)

Reads the timestamps from a given h5 file.

Parameters:
- **f (read-open h5py File)** (*open "results_00.h5" h5py.File object in read-mode*)
- **frames (3d-array)** (*list of 2d frames contained in opened h5 File.*)
- **fps (int)** (*frames per second.*)

Returns: **timestamps (1d array)**

Return type: array of timestamps for inputted frames variable

`moseq2_pca.pca.util.`**`mask_data`**(`original_data`, `mask`, `new_data`)

Create a mask subregion given a boolean mask if missing data flag is used.

Parameters:
- **original_data (3d np.ndarray)** (*input frames*)
- **mask (3d boolean np.ndarray)** (*mask array*)
- **new_data (3d np.ndarray)** (*frames to use*)

Returns: **output (3d np.ndarray)**

Return type: masked data array

`moseq2_pca.pca.util.`**`train_pca_dask`**(`dask_array`, `clean_params`, `use_fft`, `rank`, `cluster_type`, `client`, `mask=None`, `iters=10`, `recon_pcs=10`, `min_height=10`, `max_height=100`)

Train PCA using dask arrays.

Parameters:

- **dask_array (dask array)** (*chunked frames to train PCA*)
- **clean_params (dict)** (*dictionary containing filtering parameters*)
- **use_fft (bool)** (*indicates whether to use 2d-FFT on images.*)
- **rank (int)** (*Matrix rank to use*)
- **cluster_type (str)** (*indicates which cluster to use.*)
- **client (Dask.Client)** (*client object to execute dask operations*)
- **mask (dask array)** (*dask array of masked data if missing_data parameter==True*)
- **iters (int)** (*number of SVD iterations*)
- **recon_pcs (int)** (*number of PCs to reconstruct. (if missing_data = True)*)
- **min_height (int)** (*minimum mouse height from floor in (mm)*)
- **max_height (int)** (*maximum mouse height from floor in (mm)*)

Returns: **output_dict (dict)**

Return type: dictionary containing PCA training results.

Indices and tables

- `genindex`
- `modindex`
- `search`

Index

Symbols

--nworkers <nworkers>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
--output-dir <output_dir>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
--output-file <output_file>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
--pca-file <pca_file>	moseq2-pca-apply-pca command line option
le-components <pca_file_components>	moseq2-pca-compute-changepoints command line option
a-file-scores <pca_file_scores>	moseq2-pca-compute-changepoints command line option
--pca-path <pca_path>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option
--processes <processes>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
--queue <queue>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
--rank <rank>	moseq2-pca-train-pca command line option
--recon-pcs <recon_pcs>	moseq2-pca-train-pca command line option
--sigma <sigma>	moseq2-pca-compute-changepoints command line option
-tailfilter-shape <tailfilter_shape>	moseq2-pca-train-pca command line option
--tailfilter-size <tailfilter_size>	moseq2-pca-train-pca command line option
--threshold <threshold>	moseq2-pca-compute-changepoints command line option
--timeout <timeout>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
--use-fft	moseq2-pca-train-pca command line option
--verbose	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca command line option
--version	
--wall-time <wall_time>	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
-c	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
-d	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option [1] moseq2-pca-train-pca command line option
-i	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
-k	moseq2-pca-compute-changepoints command line option
-m	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option
-n	moseq2-pca-apply-pca command line option moseq2-pca-compute-changepoints command line option moseq2-pca-train-pca command line option

--cluster-type <cluster_type>

--config-file <config_file>

--cores <cores>

--dask-cache-path <dask_cache_path>

--dask-port <dask_port>

--detrend-window <detrend_window>

--fill-gaps <fill_gaps>

--fps <fps>

--h5-mask-path <h5_mask_path>

--h5-path <h5_path>

--input-dir <input_dir>

--memory <memory>

--nworkers <nworkers>

--output-dir <output_dir>

--output-file <output_file>

--pca-file <pca_file>

--pca-path <pca_path>

--processes <processes>

--queue <queue>

--timeout <timeout>

--verbose

--wall-time <wall_time>

-c

-d

-i

-m

-n

-o

-p

-q

-v

-w

moseq2-pca-clip-scores command line option

--from-end

CLIP_SAMPLES

PCA_FILE

moseq2-pca-compute-changepoints command line option

--chunk-size <chunk_size>

--cluster-type <cluster_type>

--config-file <config_file>

--cores <cores>

--dask-cache-path <dask_cache_path>

--dask-port <dask_port>

--dims <dims>

--fps <fps>

--h5-mask-path <h5_mask_path>

--h5-path <h5_path>

--input-dir <input_dir>

--klags <klags>

--memory <memory>

--neighbors <neighbors>

--nworkers <nworkers>

--output-dir <output_dir>

--output-file <output_file>

--pca-file-components <pca_file_components>

--pca-file-scores <pca_file_scores>

--pca-path <pca_path>

--processes <processes>

--queue <queue>

--sigma <sigma>

--threshold <threshold>

--timeout <timeout>

--verbose

--wall-time <wall_time>

-c

-d [1]

-i

-k

-m

-n

-o

-p

-q

-s

-v

-w

moseq2-pca-train-pca command line option

--chunk-size <chunk_size>

--cluster-type <cluster_type>

--config-file <config_file>

--cores <cores>

--dask-cache-path <dask_cache_path>

--dask-port <dask_port>

--gaussfilter-space <gaussfilter_space>

P

R

S

T

Python Module Index

m