MoSeq2-Model Documentation

version

Datta Lab

August 23, 2021

Contents

Welcome to moseq2-model's documentation!

moseq2_model Package

CLI Module

moseq2-model

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

Options

--version
Show the version and exit.

Default: False

count-frames

Counts number of frames in given h5 file (pca_scores)

```
moseq2-model count-frames [OPTIONS] INPUT_FILE
```

Options

--var-name <var_name>
Variable name in input file with PCs

Default: scores

Arguments

INPUT_FILE
Required argument

kappa-scan

Batch fit multiple models scanning over different syllable length probability prior.

```
moseq2-model kappa-scan [OPTIONS] INPUT_FILE OUTPUT_DIR
```

Options

-i, --index <index>
Path to moseq2-index.yaml for group definitions (used only with the separate-trans flag)

Default:

--out-script <out_script>
Name of bash script file to save model training commands.

Default: train_out.sh

--n-models <n_models>
Number of models to train in kappa scan.

Default: 10

--prefix <prefix>
Batch command string to prefix model training command (slurm only).

Default:

--cluster-type <cluster_type>
Platform to train models on

Default: local
Options: local|slurm

--scan-scale <scan_scale>
Scale to scan kappa values at.

Default: log
Options: log|linear

--min-kappa <min_kappa>
Minimum kappa value to begin scan from.

--max-kappa <max_kappa>
Maximum kappa value to end scan on.

--memory <memory>
RAM (slurm only)

Default: 5GB

--wall-time <wall_time>
Wall time (slurm only)

Default: 3:00:00

--partition <partition>
Partition name (slurm only)

Default: short

--get-cmd
Print scan command strings.

Default: False

--run-cmd
Run scan command strings.

Default: False

--check-every <check_every>
Increment to record training and validation log-likelihoods.

Default: 5

--robust
Use robust AR-HMM model. More tolerant to noise

Default: False

--separate-trans
Use separate transition matrix for each group

Default: False

--nlags <nlags>
Number of lags to use

Default: 3

--noise-level <noise_level>
Additive white gaussian noise for regularization. Not generally used

Default: 0

-a, --alpha <alpha>
Alpha; hierarchical dirichlet process hyperparameter (try not t ochange it).

Default: 5.7

-g, --gamma <gamma>
Gamma; hierarchical dirichlet process hyperparameter (try not to change it).

Default: 1000.0

--load-groups <load_groups>
If groups should be loaded with the PC scores.

Default: True

--percent-split <percent_split>
Training-validation split percentage used when not holding out data and when this parameter > 0.

Default: 0

-p, --progressbar <progressbar>
Show model progress

Default: True

-w, --whiten <whiten>
Whiten PCs: (e)each session (a)ll combined or (n)o whitening

Default: all

--npcs <npcs>
Number of PCs to use

Default: 10

-m, --max-states <max_states>
Maximum number of states

Default: 100

--save-model
Save model object at the end of training

Default: False

-s, --save-every <save_every>
Increment to save labels and model object (-1 for just last)

Default: -1

--e-step
Compute the expected state values for each animal

Default: False

--var-name <var_name>
Variable name in input file with PCs

Default: scores

-n, --num-iter <num_iter>
Number of times to resample model

Default: 100

-c, --ncpus <ncpus>
Number of cores to use for resampling

Default: 0

--nfolds <nfolds>
Number of folds for split

Default: 5

--hold-out-seed <hold_out_seed>

Random seed for holding out data (set for reproducibility)

Default: -1

-h, --hold-out
Hold out one fold (set by nfolds) for computing heldout likelihood

Default: False

Arguments

INPUT_FILE
Required argument

OUTPUT_DIR
Required argument

learn-model

Trains ARHMM on PCA Scores with given training parameters

```
moseq2-model learn-model [OPTIONS] INPUT_FILE DEST_FILE
```

Options

--check-every <check_every>
Increment to record training and validation log-likelihoods.

Default: 5

--robust
Use robust AR-HMM model. More tolerant to noise

Default: False

--separate-trans
Use separate transition matrix for each group

Default: False

--nlags <nlags>
Number of lags to use

Default: 3

--noise-level <noise_level>
Additive white gaussian noise for regularization. Not generally used

Default: 0

-a, --alpha <alpha>
Alpha; hierarchical dirichlet process hyperparameter (try not t ochange it).

Default: 5.7

-g, --gamma <gamma>
Gamma; hierarchical dirichlet process hyperparameter (try not to change it).

Default: 1000.0

--load-groups <load_groups>
If groups should be loaded with the PC scores.

Default: True

--percent-split <percent_split>
Training-validation split percentage used when not holding out data and when this parameter > 0.

Default: 0

-p, --progressbar <progressbar>

Show model progress

Default: True

-w, --whiten <whiten>
Whiten PCs: (e)each session (a)ll combined or (n)o whitening

Default: all

--npcs <npcs>
Number of PCs to use

Default: 10

-m, --max-states <max_states>
Maximum number of states

Default: 100

--save-model
Save model object at the end of training

Default: False

-s, --save-every <save_every>
Increment to save labels and model object (-1 for just last)

Default: -1

--e-step
Compute the expected state values for each animal

Default: False

--var-name <var_name>
Variable name in input file with PCs

Default: scores

-n, --num-iter <num_iter>
Number of times to resample model

Default: 100

-c, --ncpus <ncpus>
Number of cores to use for resampling

Default: 0

--nfolds <nfolds>
Number of folds for split

Default: 5

--hold-out-seed <hold_out_seed>
Random seed for holding out data (set for reproducibility)

Default: -1

-h, --hold-out
Hold out one fold (set by nfolds) for computing heldout likelihood

Default: False

-k, --kappa <kappa>
Kappa; hyperparameter used to set syllable duration. Larger k = longer syllable lengths

--checkpoint-freq <checkpoint_freq>
checkpoint the training after N iterations

Default: -1

--use-checkpoint
indicate whether to use previously saved checkpoint

Default: False

-i, --index <index>
Path to moseq2-index.yaml for group definitions (used only with the separate-trans flag)

Default:

--default-group <default_group>
Default group name to use for separate-trans

Default: n/a

-v, --verbose
Print syllable log-likelihoods during training.

Default: False

Arguments

INPUT_FILE
Required argument

DEST_FILE
Required argument

GUI Module

GUI front-end function for training ARHMM.

moseq2_model.gui.**learn_model_command** (progress_paths, hold_out=False, nfolds=2, num_iter=100, max_states=100, npcs=10, scan_scale='log', kappa=None, min_kappa=None, max_kappa=None, n_models=5, alpha=5.7, gamma=1000.0, separate_trans=True, robust=True, checkpoint_freq=- 1, use_checkpoint=False, check_every=5, select_groups=False, percent_split=0, output_dir=None, out_script='train_out.sh', cluster_type='local', get_cmd=True, run_cmd=False, prefix='', memory='16GB', wall_time='3:00:00', partition='short', verbose=False)
Trains ARHMM from within a Jupyter notebook. Note that the configuration file will be overriden with the function parameters.

Parameters:

- **progress_paths (dict)** (*notebook progress dict that contains paths to the pca scores, config, and index files.*)
- **hold_out (bool)** (*indicate whether to hold out data during training.*)
- **nfolds (int)** (*number of folds to hold out.*)
- **num_iter (int)** (*number of training iterations.*)
- **max_states (int)** (*maximum number of model states.*)
- **npcs (int)** (*number of PCs to include in training.*)
- **kappa (float)** (*hyperparameter for setting syllable duration. Larger kappa = longer syllable durations.*)
- **min_kappa (float)** (*Minimum kappa to train model on (used in kappa scan).*)
- **max_kappa (float)** (*Maximum kappa to train model on (used in kappa scan).*)
- **n_models (int)** (*Number of models to spawn for kappan scan*)
- **scan_scale (str)** (*Scale factor to generate scanning kappa values. ['log', 'linear']*)
- **separate_trans (bool)** (*indicate whether to compute separate syllable transition matrices for each experimental group.*)
- **robust (bool)** (*indicate whether to use a t-distributed robust ARHMM distribution. More tolerant to noise.*)
- **checkpoint_freq (int)** (*frequency at which to save model checkpoints.*)
- **use_checkpoint (bool)** (*flag to load a previously saved training checkpoint.*)
- **alpha (float)** (*scaling parameter for hierarchical dirichlet process (it's recommended to leave this parameter alone).*)
- **gamma (float)** (*scaling parameter for hierarchical dirichlet process (it's recommended to leave this parameter alone).*)
- **select_groups (bool)** (*flag to interactively display all sessions and choose subset of groups to model alone.*)
- **check_every (int)** (*perform log-likelihood check every check_every iterations.*)
- **get_cmd (bool)** (*flag to return the kappa scan learn-model commands.*)
- **run_cmd (bool)** (*flag to run the kappa scan learn-model commands.*)
- **percent_split (int)** (*train-validation data split ratio percentage.*)
- **output_dir (str)** (*directory to store multiple trained models via kappa-scan*)
- **out_script (str)** (*name of the script containing all the kappa scanning commands.*)
- **cluster_type (str)** (*name of cluster to run model training on; either ['local', 'slurm']*)
- **prefix (str)** (*slurm command prefix with job specification parameters (slurm only).*)
- **memory (str)** (*amount of memory in GB to allocate to each training job (slurm only).*)
- **wall_time (str)** (*maximum time for a slurm job to run (slurm only).*)
- **partition (str)** (*slurm partition name to run training jobs on (slurm only).*)
- **verbose (bool)** (*compute modeling summary - can slow down training.*)

Returns:

Return type: None or kappa scan command

General Utilities Module

Utility functions for handling loading and saving models and their respective metadata.

`moseq2_model.util.`**`copy_model`** (model_obj)

Return a new shallow copy of the ARHMM that doesn't contain the training data.

Parameters: **model_obj (ARHMM)** (*model to copy.*)

Returns: **cp (ARHMM)**

Return type: copy of the model

`moseq2_model.util.`**`count_frames`** (data_dict=None, input_file=None, var_name='scores')

Counts the total number of frames loaded from the PCA scores file.

Parameters:

- **data_dict (OrderedDict)** (*Loaded PCA scores OrderedDict object.*)
- **input_file (str)** (*Path to PCA Scores file to load data_dict if not already data_dict is None*)
- **var_name (str)** (*Path within PCA h5 file to load scores from.*)

Returns: **total_frames (int)**

Return type: total number of counted frames.

`moseq2_model.util.`**`create_command_strings`** (input_file, output_dir, config_data, kappas, model_name_format='model-{:03d}-{}.p')

Creates the CLI learn-model command strings with parameter flags based on the contents of the configuration dict. Each model will a use different kappa value within the specified range.

Parameters:

- **input_file (str)** (*Path to PCA Scores*)
- **index_file (str)** (*Path to index file*)
- **output_dir (str)** (*Path to directory to save models in.*)
- **config_data (dict)** (*Configuration parameters dict.*)
- **kappas (list)** (*List of kappa values for model training commands.*)
- **model_name_format (str)** (*Filename string format string.*)

Returns: **command_string (str)**

Return type: CLI learn-model command strings with the requested parameters separated by newline characters

`moseq2_model.util.`**`dict_to_h5`** (h5file, export_dict, path='/')

Recursively save dicts to h5 file groups. # https://codereview.stackexchange.com/questions/120802/recursively-save-python-dictionaries-to-hdf5-files-using-h5py

Parameters:

- **h5file (h5py.File)** (*opened h5py File object.*)
- **export_dict (dict)** (*dictionary to save*)
- **path (str)** (*path within h5 to save to.*)

Returns:

Return type: None

`moseq2_model.util.`**`get_current_model`** (use_checkpoint, all_checkpoints, train_data, model_parameters)

Checks to see whether user is loading a checkpointed model, if so, loads the latest iteration. Otherwise, will instantiate a new model.

Parameters:

- **use_checkpoint (bool)** (*CLI input parameter indicating user is loading a checkpointed model*)
- **all_checkpoints (list)** (*list of all found checkpoint paths*)
- **train_data (OrderedDict)** (*dictionary of uuid-PC score key-value pairs*)
- **model_parameters (dict)** (*dictionary of required modeling hyperparameters.*)

Returns: **arhmm (ARHMM)** (*instantiated model object including loaded data*) **itr (int)** (*starting iteration number for the model to begin training from.*)

`moseq2_model.util.`**`get_loglikelihoods`** (arhmm, data, groups, separate_trans, normalize=True)

Computes the log-likelihoods of the training sessions.

Parameters:
- **arhmm (ARHMM)** (*ARHMM model.*)
- **data (dict)** (*dict object with UUID keys containing the PCS used for training.*)
- **groups (list)** (*list of assigned groups for all corresponding session uuids. Only used if*) – separate_trans == True.
- **separate_trans (bool)** (*flag to compute separate log-likelihoods for each modeled group.*)
- **normalize (bool)** (*if set to True this function will normalize by frame counts in each session*)

Returns: **ll (list)**

Return type: list of log-likelihoods for the trained model

`moseq2_model.util.`**`get_parameter_strings`** (config_data)

Creates the CLI learn-model command using the given config_data dict contents, which can be used to run the modeling step. Function checks for the following paramters: [npcs, num_iter, separate_trans, robust, e_step, hold_out, max_states, converge, tolerance].

Parameters:
- **index_file (str)** (*Path to index file.*)
- **config_data (dict)** (*Configuration parameters dict.*)

Returns: **parameters (str)** (*String containing CLI command parameter flags.*) **prefix (str)** (*Prefix string for the learn-model command (Slurm only).*)

`moseq2_model.util.`**`get_parameters_from_model`** (model)

Get parameter dictionary from model.

Parameters: **model (ARHMM)** (*model to get parameters from.*)

Returns: **parameters (dict)**

Return type: dictionary containing all modeling parameters

`moseq2_model.util.`**`get_scan_range_kappas`** (data_dict, config_data)

Helper function that returns the kappa values to train models on based on the user's selected scanning scale range. Default values will be selected if min/max_kappa are None.
An example: scan_scale = 'log'; nframes = 1800; min_kappa = 10e3; max_kappa = 10e5; n_models = 10; >>> kappas = [1000, 1668, 2782, 4641, 7742, 12915, 21544, 35938, 59948, 100000]
Another Exmaple: nframes = 1800 'scan_scale': 'linear', 'min_kappa': None, 'max_kappa': None, 'n_models': 10 min(kappas) == 18 max(kappas) == 18000000 >>> kappas == [18, 20016, 40014, 60012, 80010, 100008, 120006, 140004, 160002, 180000]

Parameters:
- **data_dict (OrderedDict)** (*Loaded PCA score dictionary.*)
- **config_data (dict)** (*Configuration parameters dict.*)

Returns: **kappas (list)**

Return type: list of ints corresponding to the kappa value for each model.

`moseq2_model.util.`**`get_session_groupings`** (data_metadata, train_list, hold_out_list)

Creates a list or tuple of assigned groups for training and (optionally) held out data.

Parameters:
- **data_metadata (dict)** (*dict containing session group information*)
- **groups (list)** (*list of all session groups*)
- **all_keys (list)** (*list of all corresponding included session UUIDs*)
- **hold_out_list (list)** (*list of held-out uuids*)

Returns: **groupings (tuple)** (*2-tuple containing lists of train groups*) *and held-out groups (if held_out_list exists)*

`moseq2_model.util.`**`h5_to_dict`** (h5file, path: str = '/') → dict

Load h5 data to dictionary from a user specified path.

Parameters:
- **h5file (str or h5py.File)** (*file path to the given h5 file or the h5 file handle*)
- **path (str)** (*path to the base dataset within the h5 file*)

Returns: **out (dict)**

Return type: a dict with h5 file contents with the same path structure

`moseq2_model.util.`**`is_uuid`** (string)
checks to see if string is a uuid. Returns True if it is.

`moseq2_model.util.`**`load_arhmm_checkpoint`** (filename: str, train_data: dict) → dict
Load an arhmm checkpoint and re-add data into the arhmm model checkpoint.

Parameters:
- **filename (str)** (*path that specifies the checkpoint.*)
- **train_data (OrderedDict)** (*an OrderedDict that contains the training data*)

Returns: **mdl_dict (dict)**

Return type: a dict containing the model with reloaded data, and associated training data

`moseq2_model.util.`**`load_cell_string_from_matlab`** (filename, var_name='uuids')
Load cell strings from MATLAB file.

Parameters:
- **filename (str)** (*path to .mat file*)
- **var_name (str)** (*variable name to read*)

Returns: **return_list (list)**

Return type: list of selected loaded variables

`moseq2_model.util.`**`load_data_from_matlab`** (filename, var_name='features', npcs=10)
Load PC Scores from a specified variable column in a MATLAB file.

Parameters:
- **filename (str)** (*path to MATLAB (.mat) file*)
- **var_name (str)** (*variable to load*)
- **npcs (int)** (*number of PCs to load.*)

Returns: **data_dict (OrderedDict)**

Return type: loaded dictionary of uuid and PC-score pairings.

`moseq2_model.util.`**`load_pcs`** (filename, var_name='features', load_groups=False, npcs=10)
Load the Principal Component Scores for modeling.

Parameters:
- **filename (str)** (*path to the file that contains PC scores*)
- **var_name (str)** (key where the pc scores are stored within `filename`)
- **load_groups (bool)** (*Load metadata group variable*)
- **npcs (int)** (*Number of PCs to load*)

Returns: **data_dict (OrderedDict)** (*key-value pairs for keys being uuids and values being PC scores.*)
metadata (OrderedDict) (*dictionary containing lists of index-aligned uuids and groups.*)

`moseq2_model.util.`**`save_arhmm_checkpoint`** (filename: str, arhmm: dict)
Save an arhmm checkpoint and strip out data used to train the model.

Parameters:
- **filename (str)** (*path that specifies the checkpoint*)
- **arhmm (dict)** (*a dictionary containing the model obj, training iteration number,*) – log-likelihoods of each training step, and labels for each step.

Returns:

Return type: None

`moseq2_model.util.`**`save_dict`** (filename, obj_to_save=None)
Save dictionary to file.

Parameters:
- **filename (str)** (*path to file where dict is being saved.*)
- **obj_to_save (dict)** (*dict to save.*)

Returns:

Return type: None

Subpackages

moseq2_model.helpers Package

Helpers - Data Module

Helper functions for reading data from index files, and preparing metadata prior to training.

`moseq2_model.helpers.data.`**`get_heldout_data_splits`** (data_dict, train_list, hold_out_list)
Split data by session UUIDs into training and held out datasets.

Parameters:
- **data_dict (OrderedDict)** (*dictionary of all PC scores included in the model*)
- **train_list (list)** (*list of keys included in the training data*)
- **hold_out_list (list)** (*list of keys included in the held out data*)

Returns: **train_data (OrderedDict)** (*dictionary of uuid to PC score key-value pairs for uuids in train_list*) **test_data (OrderedDict)** (*dictionary of uuids to PC score key-value pairs for uuids in hold_out_list.*)

`moseq2_model.helpers.data.`**`get_training_data_splits`** (split_frac, data_dict)
Split the data into a training and held out dataset by splitting each session by some fraction *percent_split*.

Parameters:
- **split_frac (float)** (*fraction to split each session into training and held out data. A value of 0.9*) – means 90% of the data will be preserved for training.
- **data_dict (OrderedDict)** (*dict of uuid-PC Score key-value pairs for all data included in the model.*)

Returns: **training_data (OrderedDict)** (*the split percentage of the training data.*) **validation_data (OrderedDict)** (*the split percentage of the validation data*)

`moseq2_model.helpers.data.`**`graph_modeling_loglikelihoods`** (config_data, iter_lls, iter_holls, model_dir)
Graphs model training performance progress throughout modeling. Will only run if verbose == True

Parameters:
- **config_data (dict)** (*dictionary of model training parameters.*)
- **iter_lls (list)** (*list of training log-likelihoods for each training iteration*)
- **iter_holls (list)** (*list of held out log-likelihoods for each training iteration*)
- **model_dir (str)** (*path to the directory the model is saved in.*)

Returns: **img_path (str)**

Return type: path to saved graph.

`moseq2_model.helpers.data.`**`prepare_model_metadata`** (data_dict, data_metadata, config_data)
Sets model training metadata parameters, whitens data, if hold_out is True, will split data and return list of heldout keys, and updates all dictionaries.

Parameters:
- **data_dict (OrderedDict)** (*loaded data dictionary.*)
- **data_metadata (OrderedDict)** (*loaded metadata dictionary.*)
- **config_data (dict)** (*dictionary containing all modeling parameters.*)

Returns: **data_dict (OrderedDict)** (*optionally whitened and updated data dictionary.*) **model_parameters (dict)** (*model parameters used to initialize the ARHMM*) **train_list (list)** (*list of session uuids to include for training.*) **hold_out_list (list)** (*list of session uuids to hold out (if hold_out == True)*)

`moseq2_model.helpers.data.`**`process_indexfile`** (index, data_metadata, default_group='n/a', select_groups=False)

Reads index file (if it exists) and returns dictionaries containing metadata in the index file. The data_metadata will also be updated with the information read from the index file

Parameters:
- **index (str or None)** (*path to index file.*)
- **data_metadata (dict)** (*loaded metadata containing uuid and group information.*)
- **default_group (str)** (*default group name to supply to data without assigned group labels*)
- **select_groups (bool)** (*when True, print metadata describing group selection*)

Returns: **index_data (dict)** (*loaded index file.*) **data_metadata (dict)** (*updated metadata dictionary.*)

`moseq2_model.helpers.data.`**`select_data_to_model`** (index_data, data_dict, data_metadata, select_groups=False)

Prompts user to select data to model via the data uuids/groups and paths located in the index file if the select_groups flag is True. Otherwise, it will use all data to model behavior.

Parameters:
- **index_data (dict)** (*loaded dictionary from index file*)
- **data_dict (dict)** (*dictionary containing PC scores for all sessions*)
- **data_metadata (dict)** (*dictionary containing metadata associated with the*) – recording sessions
- **select_groups (bool)** (*flag to solicit user input on which groups to select for modeling*)

Returns: **data_dict (dict)** (*dictionary to model containing data from the selected*) – session uuids **data_metadata (dict)** (*updated metadata containing the selected uuids and*) – groups

Helpers - Wrappers Module

Wrapper functions for all functionality included in MoSeq2-Model 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_model.helpers.wrappers.`**`kappa_scan_fit_models_wrapper`** (input_file, config_data, output_dir)

Wrapper function that spools multiple model training commands for a range of kappa values.

Parameters:
- **input_file (str)** (*Path to PCA Scores*)
- **config_data (dict)** (*Dict containing model training parameters*)
- **output_dir (str)** (*Path to output directory to save trained models*)

Returns: **command_string (str)**

Return type: CLI command string for model training commands.

`moseq2_model.helpers.wrappers.`**`learn_model_wrapper`** (input_file, dest_file, config_data)

Function used to train ARHMM on PCA data.

Parameters:
- **input_file (str)** (*path to pca scores file.*)
- **dest_file (str)** (*path to save model to.*)
- **config_data (dict)** (*dictionary containing the modeling parameters.*)

Returns:

Return type: None

moseq2_model.train Package

Train - Fit Module

Train - Label Utilities Module

Train - Model Module

ARHMM model initialization utilities.

`moseq2_model.train.models.`**`ARHMM`** (data_dict, kappa=1000000.0, gamma=999, nlags=3, alpha=5.7, K_0_scale=10.0, S_0_scale=0.01, max_states=100, empirical_bayes=True, affine=True, model_hypparams={}, obs_hypparams={}, sticky_init=False, separate_trans=False, groups=None, robust=False, silent=False)

Initializes ARHMM and adds data and group labels to the ARHMM model.

Parameters:

- **data_dict (OrderedDict)** (*training data to add to model*)
- **kappa (float)** (*hyperparameter for setting syllable duration. Larger kappa = longer syllable durations*)
- **gamma (float)** (*scaling parameter for hierarchical dirichlet process (it's recommended to leave this parameter alone)*)
- **nlags (int)** (*number of lag frames to add to sessions*)
- **alpha (float)** (*scaling parameter for hierarchical dirichlet process (it's recommended to leave this parameter alone)*)
- **K_0_scale (float)** (*Standard deviation of lagged data*)
- **S_0_scale (float)** (*scale standard deviation initialization (don't touch this parameter unless necessary)*)
- **max_states (int)** (*Maximum number of model states*)
- **empirical_bayes (bool)** (*Use empirical bayes to initialize sigma*)
- **affine (bool)** (*Use affine transformation in the AR processes*)
- **model_hypparams (dict)** (*other model parameters (don't touch this parameter unless necessary)*)
- **obs_hypparams (dict)** (*observed parameters nu_0, S_0, M_0, and K_0 (don't touch this parameter unless necessary)*)
- **sticky_init (bool)** (*Initialize the model with random states*)
- **separate_trans (bool)** (*use separate transition matrices for each group*)
- **groups (list)** (*list of groups to model*)
- **robust (bool)** (*use student's t-distributed AR model*)
- **silent (bool)** (*flag to print out model information.*)

Returns: **model (ARHMM)**

Return type: initialized model object

Train - General Utilities Module

ARHMM utility functions

`moseq2_model.train.util.`**`get_crosslikes`** (arhmm, frame_by_frame=False)

Gets the cross-likelihoods, a measure of confidence in label segmentation, for each model label.

Parameters:

- **arhmm** (*the ARHMM model object*)
- **frame_by_frame (bool)** (*if True, the cross-likelihoods will be computed for each frame.*)

Returns: **All_CLs (list)** (*a dictionary containing cross-likelihoods for each syllable pair.*) if `frame_by_frame=True`, it will contain a value for each frame **CL (np.ndarray)** (*the average cross-likelihood for each syllable pair*)

`moseq2_model.train.util.`**`get_labels_from_model`**(model)
Grabs model labels for each training dataset and places them in a list.

Parameters: **model (ARHMM)** (*trained ARHMM model*)
Returns: **labels (list)**
Return type: An array of predicted syllable labels for each training session

`moseq2_model.train.util.`**`get_model_summary`**(model, groups, train_data, val_data, separate_trans)
Computes log-likelihood of train_data and val_data (if not None). aIs only run if verbose = True.

Parameters:
- **model (ARHMM)** (*model to compute lls.*)
- **groups (list)** (*list of session group names.*)
- **train_data (OrderedDict)** (*Ordered dict of training data*)
- **val_data** (*(OrderedDict or None): Ordered dict of validation/held-out data*)
- **separate_trans (bool) indicates whether to separate lls for each group.**

Returns: **train_ll (float)** (*normalized average training log-likelihood across all recording sessions.*) **val_ll (float)** (*normalized average held-out log-likelihood across all recording sessions.*)

`moseq2_model.train.util.`**`rleslices`**(seq)
Get changepoint slices

Parameters: **seq (list)** (*list of labels*)
Returns: **(map generator)**
Return type: slices of syllable changepoints

`moseq2_model.train.util.`**`run_e_step`**(arhmm)
Computes the expectation for each state across all frames of the training dataset and places them in a list.

Parameters: **arhmm (ARHMM)** (*model to compute expected states from.*)
Returns: **e_states (list)**
Return type: list of expected states

`moseq2_model.train.util.`**`slices_from_indicators`**(indseq)
Compute start and stop indices (slices) for each contiguous sequence of True values in *indseq*.

Parameters: **indseq (list)** (*Indicator array, containing True and False values*)
Returns: **(list)**
Return type: list of slices from *indseq*.

`moseq2_model.train.util.`**`train_model`**(model, num_iter=100, ncpus=1, checkpoint_freq=None, checkpoint_file=None, start=0, progress_kwargs={}, train_data=None, val_data=None, separate_trans=False, groups=None, verbose=False, check_every=2)
ARHMM training: Resamples ARHMM for inputted number of iterations, and optionally computes loglikelihood scores for each iteration if verbose is True.

Parameters:
- **model (ARHMM)** (*model to train*)
- **num_iter (int)** (*total number of resampling iterations*)
- **ncpus (int)** (*number of cpus used to resample the model*)
- **checkpoint_freq (int)** (*frequency (iterations) to save a checkpoint of the model*)
- **checkpoint_file (str)** (*path to save new checkpoint file*)
- **start (int)** (*starting iteration index used to resume modeling. Default is 0*)
- **progress_kwargs (dict)** (*keyword arguments for progress bar*)
- **train_data (OrderedDict)** (*dict of training data used for getting log-likelihods*) – if verbose is True
- **val_data (OrderedDict)** (*dict of validation data used for getting validation*) – log-likelihoods if verbose is True.
- **separate_trans (bool)** (*use separated transition matrices for each group*)
- **groups (list)** (*list of groups included in modeling used for getting log-likelihoods*) – if verbose is True
- **verbose (bool)** (get log-likelihoods at *check_every* interval)
- **check_every (int)** (*frequency (iterations) to record model training/validation*) – log-likelihoods during training

Returns: **model (ARHMM)** (*trained model.*) **log_likelihood (list)** (*list of training log-likelihoods per session after modeling.*) **labels (list)** (*list of labels predicted per session after modeling.*) **iter_lls (list)** (list of training log-likelihoods for each *check_every* iteration.) **iter_holls (list)** (list of held-out log-likelihoods for each *check_every* iteration.) **interrupt (bool)** (*flag to notify the caller of this function if a keyboard interrupt happened*)

`moseq2_model.train.util.`**`training_checkpoint`** (model, itr, checkpoint_file)
Formats the model checkpoint filename and saves the model checkpoint

Parameters:
- **model (ARHMM)** (*Model being trained.*)
- **itr (itr)** (*Current modeling iteration.*)
- **checkpoint_file (str)** (*Model checkpoint file name.*)

`moseq2_model.train.util.`**`whiten_all`** (data_dict, center=True)
Whitens the PC Scores (with Cholesky decomposition) using all the data to compute the covariance matrix.

Parameters:
- **data_dict (OrderedDict)** (*Training dataset*)
- **center (bool)** (*Indicates whether to center data by subtracting the mean PC score.*)

Returns: **data_dict (OrderedDict)**

Return type: Whitened training data dictionary

`moseq2_model.train.util.`**`whiten_each`** (data_dict, center=True)
Whiten the PC scores for each training dataset separately.

Parameters:
- **data_dict (OrderedDict)** (*Training dataset*)
- **center (bool)** (*Indicates whether to center data by subtracting the mean PC score.*)

Returns: **data_dict (OrderedDict)**

Return type: Whitened training data dictionary

`moseq2_model.train.util.`**`zscore_all`** (data_dict, npcs=10, center=True)
z-score the PC Scores altogether.

Parameters:
- **data_dict (OrderedDict)** (*Training dictionary*)
- **npcs (int)** (*number of pcs included*)
- **center (bool)** (*Indicates whether to center data by subtracting the mean PC score.*)

Returns: **data_dict (OrderedDict)**

Return type: z-scored training data dictionary

`moseq2_model.train.util.`**`zscore_each`** (data_dict, center=True)

z-score each set of PC Scores separately

Parameters:
- **data_dict (OrderedDict)** (*Training dictionary*)
- **center (bool)** (*Indicates whether to center data by subtracting the mean PC score.*)

Returns: **data_dict (OrderedDict)**

Return type: z-scored training data dictionary

Index

- **genindex**

Index

Symbols

A

C

D

G

O

P

R

S

T

W

Z

Python Module Index

m