Phecomp data analysis scripts quick guide

Isabel Fernández

Raw data files obtained with Phecomp cages have .mtb extension:
20090302.mtb

Steps to pre-process the data:

1- Keep only the useful data, eg. Food intake information (*f* option) and keep it in a second by second acquisition format. Data is splitted into 2 output files, one for each type of cages (only chow food, SC, and chow+choc ,CD).

```
python raw2matrix.py f 20090302.mtb
```

Output files: 20090302_CD_f.mtx, 20090302_SC_f.mtx

2- Convert the data into eating amounts (current acquired value is compared to next acquired value, and keep the difference) keeping second by second acquisition format.

```
python matrix2increment.py a 20090302_CD_f.mtx
```

Using option *a* (all), both _SC and CD files are processed at time.
If all eating amounts are considered, use `matrix2increment_all.py`
If only eating events are useful, convert it to binary (1, eat event, 0 no eat event) using `matrix2binary.py`

Output files: 20090302_SC_f_inc.mtx, 20090302_CD_f_inc.mtx

3- Keep the time intervals between each eating event and join all the files that compose the data set. Intervals are converted into its corresponding bin (interval distribution is divided into bins such as each bin contains roughly the same number of elements)

```
python  joindata_interval.py  m  20090302_SC_f_inc.mtx  20090309_SC_f_inc.mtx
20090313_SC_f_inc.mtx   20090317_SC_f_inc.mtx 20090323_SC_f_inc.mtx
```

Using option *m* (merge) joins all entries of the same cage in a single sequence.
All files that are to be joint must be listed after the option (m in the example)
`joindata_interval.py` filters out big intervals.
If input data comes from CD cages and split between chow and choc food is needed, use `joindata_interval_choc_chow.py`. This will add an extra line per cage with the eating amounts and the eating type (e.g. 0.02c, for choc, 0.02w, for chow)
If no filtering is desired use `joindata_interval_multiple_all.py`
If input data is in eating amount format use `joindata.py`
If input data is in eating amount format and compression of non-eating periods is desired, use `joindata_compress.py.`
To obtain both the interval and eating amount information use
`joindata_getincrement_and_interval.py`

To obtain only the eating amount information, without any time reference use `joindata_getincrement.py`

Output files: 20090302_SC_f_inc_m_int.jnd (contains interval sequences), 20090302_SC_f_inc_m_bin.jnd (contains bin sequences)

Train the HMM using the pre-processed data:

Once the data is already pre-processed, it is ready for training the model. To train a 2 states model. The training is done with file containing the bins:

```
hmmtrain_homogeneous.py 2 20090302_SC_f_inc_m_bin.jnd
```

This uses all input sequences in .jnd file and trains a 2 states default model using Baum-Welch algorithm. This script uses scaling variables in the computation of the model in order to avoid underflow.
The version without scaling variables is also available, `hmmtrain_homogeneous_noscaling.py`, although it is not recommended unless no underflow is assured.
In case that the input data has labels (states) available, use `hmmtrain_labeled.py`

Output files: 20090302_SC_f_inc_m_bin_2states.hmm

Decode data using an HMM model:

Once the model is trained it can be used to decode other sequences. Since the training data is in 'bin' format, the data to be decoded must also be in this format.

```
hmmpath.py 20090302_SC_f_inc_m_bin_2states.hmm 20090724_SC_f_inc_m_bin.jnd
```

Output files: 20090724_CD_f_inc_m_bin.pth

If instead of training with Baum-Welch, posterior decoding is preferred:

```
hmmpath_posterior.py 20090302_SC_f_inc_m_bin_2states.hmm
20090724_SC_f_inc_m_bin.jnd
```

This second option provides with a file with the posterior probabilities for each state and a path file, similar to the one obtained with Baum-Welch, that shows the state corresponding to the states that own the higher probability in posterior decoding.

Output files: 20090724_CD_f_inc_m_bin.pdg (posterior probabilities)
20090724_CD_f_inc_m_bin_posterior.pth (posterior path)