hmmkay

Release 0.0.1

Aug 25, 2019

Contents

Hmmkay is a basic library for discrete Hidden Markov Models that relies on numba's just-in-time compilation. It supports decoding, likelihood scoring, fitting (parameter estimation), and sampling.

Hmmkay accepts sequences of arbitrary length, e.g. 2d numpy arrays or lists of iterables. Hmmkay internally converts lists of iterables into Numba typed lists of numpy arrays (you might want to do that yourself to avoid repeated convertions using *hmmkay.utils.check_sequences()*)

Scoring and decoding example:

```
>>> from hmmkay.utils import make_proba_matrices
>>> from hmmkay import HMM

>>> init_probas, transition_probas, emission_probas = make_proba_matrices(
...     n_hidden_states=2,
...     n_observable_states=4,
...     random_state=0
... )
>>> hmm = HMM(init_probas, transition_probas, emission_probas)

>>> sequences = [[0, 1, 2, 3], [0, 2]]
>>> hmm.log_likelihood(sequences)
-8.336
>>> hmm.decode(sequences)  # most likely sequences of hidden states
[array([1, 0, 0, 1], dtype=int32), array([1, 0], dtype=int32)]
```

Fitting example:

```
>>> from hmmkay.utils import make_observation_sequences
>>> sequences = make_observation_sequences(n_seq=100, n_observable_states=4, random_
↪state=0)
>>> hmm.fit(sequences)
```

Sampling example:

```
>>> hmm.sample(n_obs=2, n_seq=3)  # return sequences of hidden and observable states
(array([[0, 1],
        [1, 1],
        [0, 0]]), array([[0, 2],
        [2, 3],
        [0, 0]]))
```

CHAPTER 1

API Reference

1.1 HMM class

class hmmkay.**HMM**(*init_probas*, *transitions*, *emissions*, *n_iter=10*)

Discrete Hidden Markov Model.

The number of hidden and observable states are determined by the shapes of the probability matrices passed as parameters.

Parameters

- **init_probas** (*array-like of shape (n_hidden_states,)*) – The initial probabilities.
- **transitions** (*array-like of shape (n_hidden_states, n_hidden_states)*) – The transition probabilities. `transitions[i, j] = P(st+1 = j / st = i)`.
- **emissions** (*array-like of shape (n_hidden_states, n_observable_states)*) – The probabilities of symbol emission. `emissions[i, o] = P(Ot = o / st = i)`.
- **n_iter** (*int, default=10*) – Number of iterations to run for the EM algorithm (in `fit()`).

decode(*sequences*, *return_log_probas=False*)

Decode sequences with Viterbi algorithm.

Given a sequence of observable states, return the sequence of hidden states that most-likely generated the input.

Parameters

- **sequences** (*array-like of shape (n_seq, n_obs) or list (or numba typed list) of iterables of variable length*) – The sequences of observable states

- **return_log_probas** (*bool, default=False*) – If True, log-probabilities of the joint sequences of observable and hidden states are returned

Returns

- **best_paths** (*ndarray of shape (n_seq, n_obs) or list of ndarray of variable length*) – The most likely sequences of hidden states.
- **log_probabilities** (*ndarray of shape (n_seq,)*) – log-probabilities of the joint sequences of observable and hidden states. Only present if `return_log_probas` is True.

fit (*sequences*)

Fit model to sequences.

The probabilities matrices `init_probas`, `transitions` and `emissions` are estimated with the EM algorithm.

Parameters **sequences** (*array-like of shape (n_seq, n_obs) or list (or numba typed list) of iterables of variable length*) – The sequences of observable states

Returns **self**

Return type HMM instance

log_likelihood (*sequences*)

Compute log-likelihood of sequences.

Parameters **sequences** (*array-like of shape (n_seq, n_obs) or list (or numba typed list) of iterables of variable length*) – The sequences of observable states

Returns **log_likelihood**

Return type array of shape (n_seq,)

sample (*n_seq=10*, *n_obs=10*, *random_state=None*)

Sample sequences of hidden and observable states.

Parameters

- **n_seq** (*int, default=10*) – Number of sequences to sample
- **n_obs** (*int, default=10*) – Number of observations per sequence
- **random_state** (*int or np.random.RandomState instance, default=None*) – Controls the RNG, see scikt-learn glossary for details.

Returns

- **hidden_states_sequences** (*ndarray of shape (n_seq, n_obs)*)
- **observable_states_sequences** (*ndarray of shape (n_seq, n_obs)*)

1.2 Utils

The utils module contains helpers for input checking, parameter generation and sequence generation.

hmmkay.utils.**make_observation_sequences** (*n_seq=10*, *n_observable_states=3*, *n_obs_min=10*, *n_obs_max=None*, *random_state=None*)

Generate random observation sequences.

Parameters

- **n_seq** (*int, default=10*) – Number of sequences to generate
- **n_observable_states** (*int, default=3*) – Number of observable states.
- **n_obs_min** (*int, default=10*) – Minimum length of each sequence.
- **n_obs_max** (*int or None, default=None*) – If None (default), all sequences are of length `n_obs_min` and a 2d ndarray is returned. If an int, the length of each sequence is chosen randomly with `n_obs_min <= length < n_obs_max`. A numba typed list of arrays is returned in this case.
- **random_state** (*int or np.random.RandomState instance, default=None*) – Controls the RNG, see scikt-learn glossary for details.

Returns **sequences** – The generated sequences of observable states

Return type ndarray of shape (n_seq, n_obs_min,) or numba typed list of ndarray of variable length

hmmkay.utils.**make_proba_matrices**(*n_hidden_states=4*, *n_observable_states=3*, *random_state=None*)

Generate random probability matrices.

Parameters

- **n_hidden_states** (*int, default=4*) – Number of hidden states
- **n_observable_states** (*int, default=3*) – Number of observable states
- **random_state** (*int or np.random.RandomState instance, default=None*) – Controls the RNG, see scikt-learn glossary for details.

Returns

- **init_probas** (*array-like of shape (n_hidden_states,)*) – The initial probabilities.
- **transitions** (*array-like of shape (n_hidden_states, n_hidden_states)*) – The transition probabilities. `transitions[i, j] = P(st+1 = j / st = i)`.
- **emissions** (*array-like of shape (n_hidden_states, n_observable_states)*) – The probabilities of symbol emission. `emissions[i, o] = P(Ot = o / st = i)`.

hmmkay.utils.**check_sequences**(*sequences*, *return_longest_length=False*)

Convert sequences into appropriate format.

This helper is called before any method that uses sequences. It is recommended to convert your sequences once and for all before using the `HMM` class, to avoid repeated convertions.

Parameters

- **sequences** (*array-like of shape (n_seq, n_obs) or list/typed list of iterables of variable length.*) – Lists of iterables are converted to typed lists of numpy arrays, which can have different lengths. 2D arrays are untouched (all sequences have the same length).
- **return_longest_length** (*bool, default=False*) – If True, also return the length of the longest sequence.

Returns **sequences** – The sequences converted either to ndarray or numba typed list of ndarrays.

Return type ndarray of shape (n_seq, n_obs) or typed list of ndarray of variable length

Python Module Index

h

Index