3_EncoderMap_full_length_FAT10

June 17, 2025

[1]:
```
from tqdm import tqdm
import mdtraj as md
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import encodermap as em
```

```
/home/leonf/git/encoder_map_private/encodermap/loading/features.py:36:
UserWarning: Features need Unittests
  warnings.warn("Features need Unittests")
```

1 EncoderMap embedding for the closeness centralities of full-lenght FAT10

1.1 Loading closeness centralities

Closeness centralities were calculated in 1_Calculating_Closeness_Centralities_full_length_FAT10.ipynb

Shape: (n_frames, n_residues) –> (300150,165)

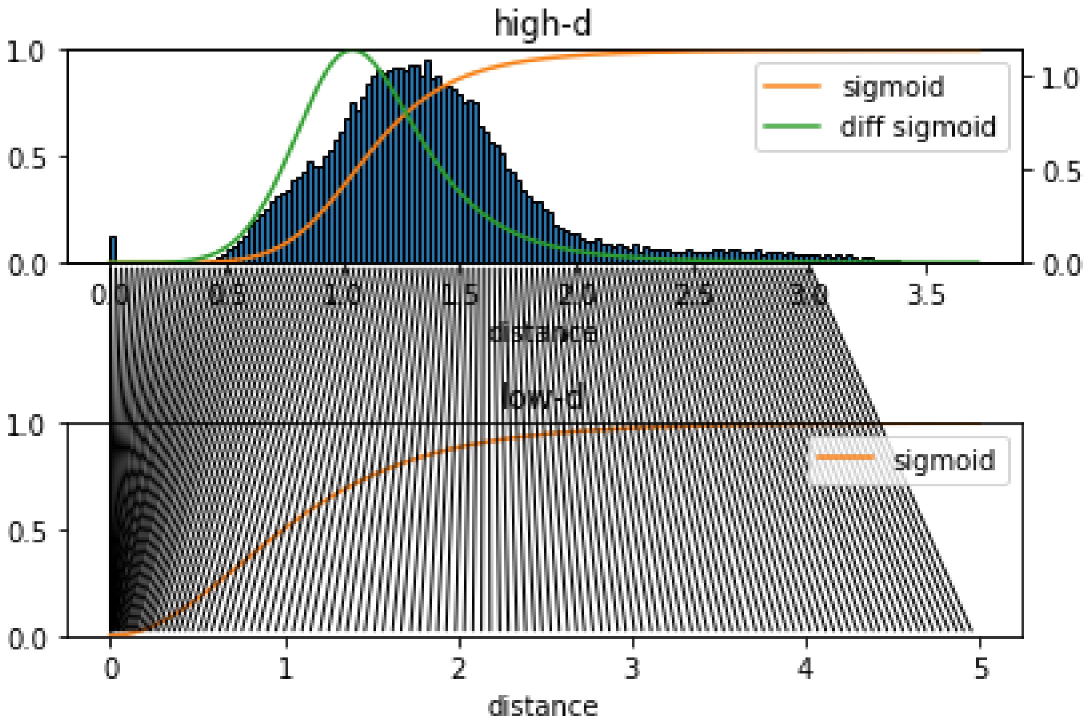
[2]:
```
closeness = np.load("./Closeness_Centralities_full_length_FAT10.npy")
closeness = closeness/np.max(closeness) #normalizing to have max = 1 in case␣
↪this was not done upon calculation. This can aid in obtaining better␣
↪EncoderMap results.
```

1.2 Setting parameters for EncoderMap

Please refer to the original publications of EncoderMap (Paper I and Paper II) and the python package documentation for details on using EncoderMap and parameter selection.

[3]:
```
%matplotlib inline
parameters = em.Parameters()
parameters.main_path = em.misc.run_path("encodermap_runs_FAT10")
parameters.n_steps = 25000
parameters.dist_sig_parameters = (1.1 ,6 ,6 , 1, 2, 6)
parameters.learning_rate = 0.00001
parameters.l2_reg_constant = 0.00001
parameters.periodicity = float('inf')
# if your data set is large you should not try to calculate
# the pairwise distance histogram with the complete data.
em.plot.distance_histogram(closeness[::1000],  # e.g. use high_d_data[::100] to␣
 ↪use every 100th point
                           parameters.periodicity,
                           parameters.dist_sig_parameters)
```

[3]:
```
(<AxesSubplot:title={'center':'high-d'}, xlabel='distance'>,
 <AxesSubplot:label='55ca058e-366c-437c-922b-e2023400cb3d'>,
 <AxesSubplot:title={'center':'low-d'}, xlabel='distance'>)
```



1.3 Running EncoderMap

[4]:
```
e_map = em.EncoderMap(parameters, closeness)
e_map.train()

low_d_projection = e_map.encode(closeness)
```

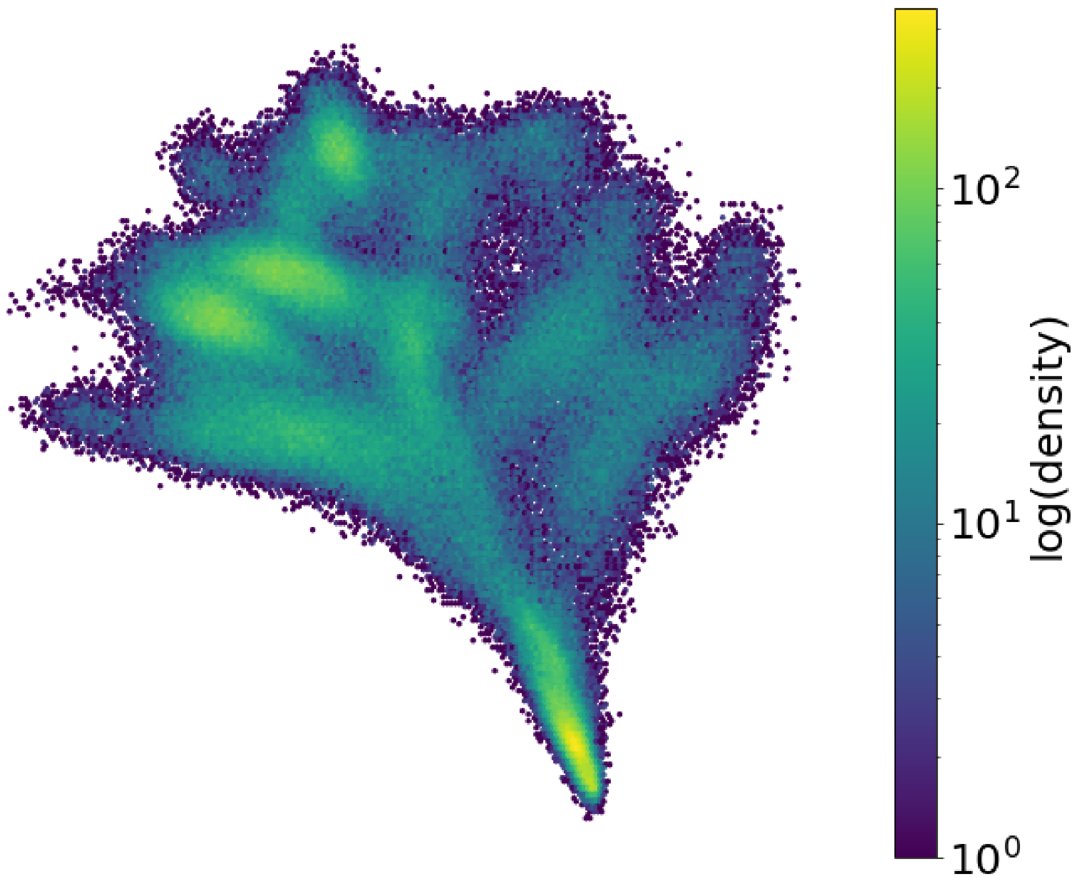
```
Output files are saved to encodermap_runs_FAT10/run0 as defined in 'main_path'
in the parameters.
```

```
 20%|          | 4999/25000 [00:30<01:55, 173.90it/s, Loss after step
5000=8.61]WARNING:absl:Function `_wrapped_model` contains input name(s)
```

```
Encoder_0_input with unsupported characters which will be renamed to
encoder_0_input in the SavedModel.
WARNING:absl:Function `_wrapped_model` contains input name(s) Decoder_0_input
with unsupported characters which will be renamed to decoder_0_input in the
SavedModel.
 40%|        | 9999/25000 [00:59<01:25, 176.25it/s, Loss after step
10000=7.95]WARNING:absl:Function `_wrapped_model` contains input name(s)
Encoder_0_input with unsupported characters which will be renamed to
encoder_0_input in the SavedModel.
WARNING:absl:Function `_wrapped_model` contains input name(s) Decoder_0_input
with unsupported characters which will be renamed to decoder_0_input in the
SavedModel.
 60%|       | 14999/25000 [01:28<00:56, 176.10it/s, Loss after step
15000=7.4] WARNING:absl:Function `_wrapped_model` contains input name(s)
Encoder_0_input with unsupported characters which will be renamed to
encoder_0_input in the SavedModel.
WARNING:absl:Function `_wrapped_model` contains input name(s) Decoder_0_input
with unsupported characters which will be renamed to decoder_0_input in the
SavedModel.
 80%|      | 19999/25000 [01:57<00:28, 176.18it/s, Loss after step
20000=6.14]WARNING:absl:Function `_wrapped_model` contains input name(s)
Encoder_0_input with unsupported characters which will be renamed to
encoder_0_input in the SavedModel.
WARNING:absl:Function `_wrapped_model` contains input name(s) Decoder_0_input
with unsupported characters which will be renamed to decoder_0_input in the
SavedModel.
100%|      | 24999/25000 [02:26<00:00, 179.30it/s, Loss after step
25000=5.59]WARNING:absl:Function `_wrapped_model` contains input name(s)
Encoder_0_input with unsupported characters which will be renamed to
encoder_0_input in the SavedModel.
WARNING:absl:Function `_wrapped_model` contains input name(s) Decoder_0_input
with unsupported characters which will be renamed to decoder_0_input in the
SavedModel.
100%|      | 25000/25000 [02:27<00:00, 169.83it/s, Loss after step
25000=5.59]
WARNING:absl:Function `_wrapped_model` contains input name(s) Encoder_0_input
with unsupported characters which will be renamed to encoder_0_input in the
SavedModel.
WARNING:absl:Function `_wrapped_model` contains input name(s) Decoder_0_input
with unsupported characters which will be renamed to decoder_0_input in the
SavedModel.
```

1.4 Plotting the resulting low-dimensional embedding, colored by population density

[5]:
```
%matplotlib inline
plt.rcParams.update({'font.size': 25})
fig, ax = plt.subplots(figsize = (10,8),tight_layout = True)
hex_map = ax.hexbin(low_d_projection[:, 0],
                    low_d_projection[:, 1],
                    cmap = 'viridis',
                    bins = 'log',
                    mincnt = 1,
                    gridsize = 200)
ax.set_axis_off()
cb = plt.colorbar(hex_map, label = "log(density)")
cb.set_alpha(1)
cb.draw_all()
```



1.5 Note: EncoderMap is non-deterministic.

This means the resulting embeddings can deviate between runs. The embeddings may appear rotated or flipped or differ slightly in other details. This can lead to differing results when using HDBSCAN, such as deviations in cluster numbering. To ensure consistency with the analyses in the paper, we provide the coordinates of the EncoderMap that was used in the paper and proceed in the analyses with those.