analyze_batch

January 13, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import pysta
import stc
%load_ext autoreload
%autoreload 2
```

1 run for all cells (OFF LINE)

run

`python3 stc_batch.py [DATASET]`

datasets * 20180618 * 20180621 * 20180626 * 20180828

1.1 load data

[2]:
```
# load data

# load stim and spike data
#dataset_name = "20180618"
dataset_name = "20180621"
#dataset_name = "20180626"
#dataset_name = "20180828"
dataset_filename = "data/{}.mat".format(dataset_name)

stim, spike_train, info = pysta.load_data(dataset_filename)

channel_names = [ch.replace("ch_","") for ch in info["channel_names"]]
# info["channel_names"]

# load cell type
cell_type = pd.read_csv("data/{}_cell_type.csv".format(dataset_name))
```

```
List of arrays in this file:
```

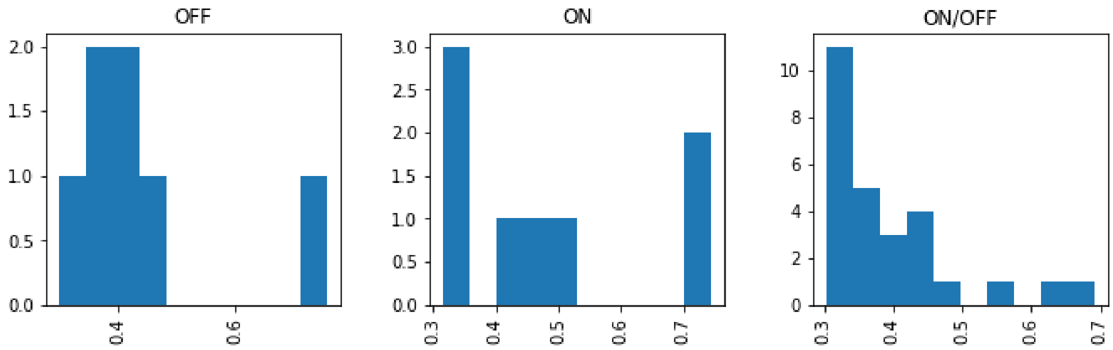
```
<KeysViewHDF5 ['#refs#', 'channel_names', 'height', 'sampling_rate',
'spike_train', 'stim', 'width']>
Shape of the array stim:  (64, 9000)
Shape of the array spike_train:  (115, 9000)
length of the list channel_names:  115
sampling_rate:  10.0
```

1.2 result - eigenvalues

[3]:
```
# read eigenvalus
all_eig_values = dict()
# eigen_values = list()
largest_eig_values = list()

folder_name = "{}_stc_tap8".format(dataset_name)
#folder_name = "stc_tap10_center_half"
for channel_name in channel_names: #info["channel_names"]:
    filename = "{}/ch_{}_eig_val.txt".format(folder_name,channel_name)
    eig_val = np.loadtxt(filename)

    all_eig_values[channel_name] = eig_val
#     eigen_values.append(eig_val)
    largest_eig_values.append(eig_val[0])

    #print(channel_name)
# plt.hist(largest_eig_values)

# all_eig_values
```

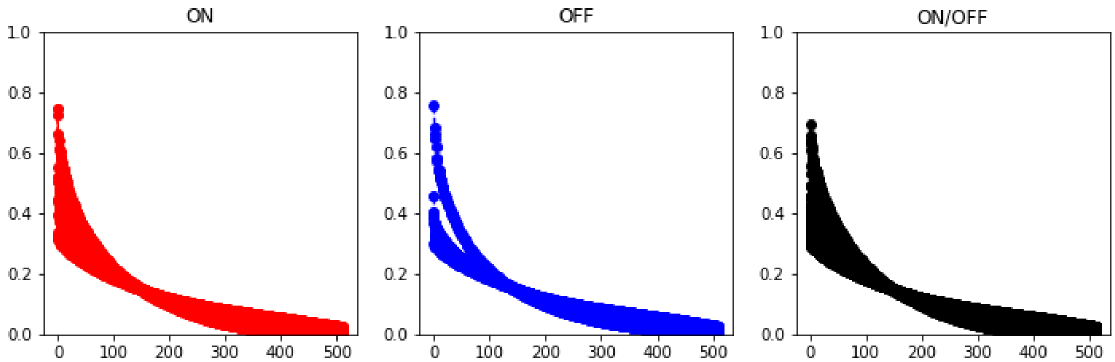
[4]:
```
# convert to DataFrame
eig = pd.DataFrame({"channel_name": channel_names, "largest_eig_values":
 ↪largest_eig_values})

results = cell_type.merge(eig, on="channel_name")
results.hist(column=["largest_eig_values"], by=["cell_type"], layout=(1,3),␣
 ↪figsize=(11,3))
```

[4]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x1a220ecb10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x10c8bb290>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x10c7bbed0>],
      dtype=object)
```



[5]:
```
# plot eigenvalues for cell type

plt.figure(figsize=(12,3.5))
ax=plt.subplot(131)
for channel_name in cell_type.loc[cell_type["cell_type"] ==␣
 ↪"ON"]["channel_name"]:
    #print(channel_name)
    plt.plot(all_eig_values[channel_name], 'or--')
ax.set_ylim(0, 1)
plt.title("ON")

ax=plt.subplot(132)
for channel_name in cell_type.loc[cell_type["cell_type"] ==␣
 ↪"OFF"]["channel_name"]:
    #print(channel_name)
    plt.plot(all_eig_values[channel_name], 'ob--')
ax.set_ylim(0, 1)
plt.title("OFF")

ax=plt.subplot(133)
for channel_name in cell_type.loc[cell_type["cell_type"] == "ON/
 ↪OFF"]["channel_name"]:
    #print(channel_name)
    plt.plot(all_eig_values[channel_name], 'ok--')
ax.set_ylim(0, 1)
plt.title("ON/OFF")
```

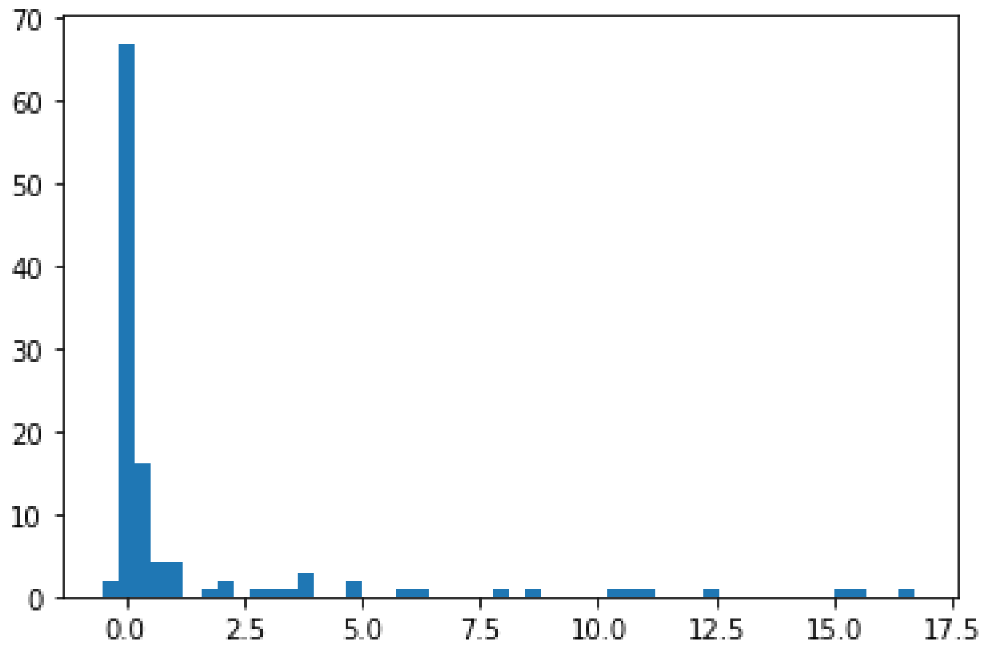
[5]:
```
Text(0.5, 1.0, 'ON/OFF')
```



1.3 result - kurtosis

[6]:
```
# load kurtosis
#tap = 5
Ks = np.loadtxt("{}/kurtosis.txt".format(folder_name))
plt.hist(Ks,50)

# store into a DataFrame
# remove "ch_" from channel names
kurtosis = pd.DataFrame({"channel_name": channel_names, "kurtosis": Ks})

kurtosis
```

[6]:
```
    channel_name  kurtosis
0            12a  0.047361
1            13a  0.048693
2            13b  0.054572
3            13c  0.351167
4            14a  0.119820
..           …         …
110          85c -0.078257
111          86a  0.007039
112          86b  0.162774
113          87a  0.099374
114          87b  1.940998

[115 rows x 2 columns]
```



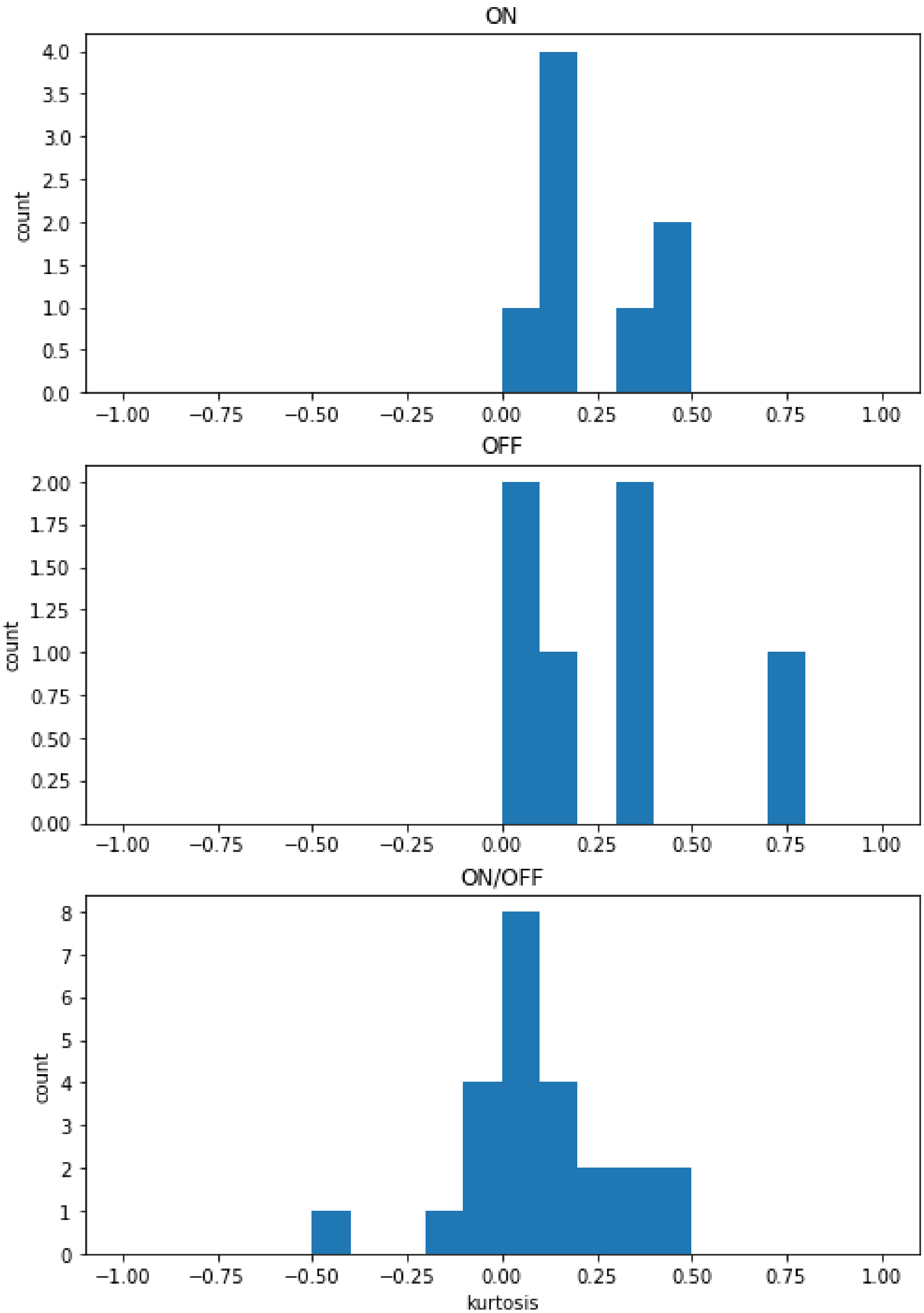
[7]:
```
# merge with cell_type
#cell_type
results = results.merge(kurtosis, on="channel_name", how="outer")
#results = cell_type.merge(kurtosis, on="channel_name")
# results.hist(column=["kurtosis"], by=["cell_type"], layout=(1,3),␣
↪figsize=(12,3.5))
```

[8]:
```
results
```

[8]:
```
    channel_name cell_type  largest_eig_values   kurtosis
0            12a    ON/OFF            0.455981   0.047361
1            13c    ON/OFF            0.381917   0.351167
2            16a    ON/OFF            0.315114  -0.001641
3            21b    ON/OFF            0.318001   0.133269
4            22b    ON/OFF            0.313448   0.105355
..           …        …                 …          …
110          83b       NaN                 NaN   2.042236
111          83c       NaN                 NaN  15.476388
112          85c       NaN                 NaN  -0.078257
113          86a       NaN                 NaN   0.007039
114          87a       NaN                 NaN   0.099374

[115 rows x 4 columns]
```

[9]:
```
k_on = results.loc[results["cell_type"]=="ON","kurtosis"]
k_off = results.loc[results["cell_type"]=="OFF","kurtosis"]
k_on_off = results.loc[results["cell_type"]=="ON/OFF","kurtosis"]

bins = np.linspace(-1,1,21)
# plt.hist(k_on, bins)
# plt.hist(k_off, bins)
# plt.hist(k_on_off, bins)

# plot separately
plt.figure(figsize=(8,12))
plt.subplot(3,1,1)
plt.hist(k_on, bins)
plt.title("ON")
# plt.xlabel("kurtosis")
plt.ylabel("count")

plt.subplot(3,1,2)
plt.hist(k_off, bins)
plt.title("OFF")
# plt.xlabel("kurtosis")
plt.ylabel("count")

plt.subplot(3,1,3)
plt.hist(k_on_off, bins)
plt.title("ON/OFF")
plt.xlabel("kurtosis")
plt.ylabel("count")
```

[9]:
```
Text(0, 0.5, 'count')
```


ON
count
OFF
count
ON/OFF
count
kurtosis

[10]:
```
results.loc[results["kurtosis"]<0]
```

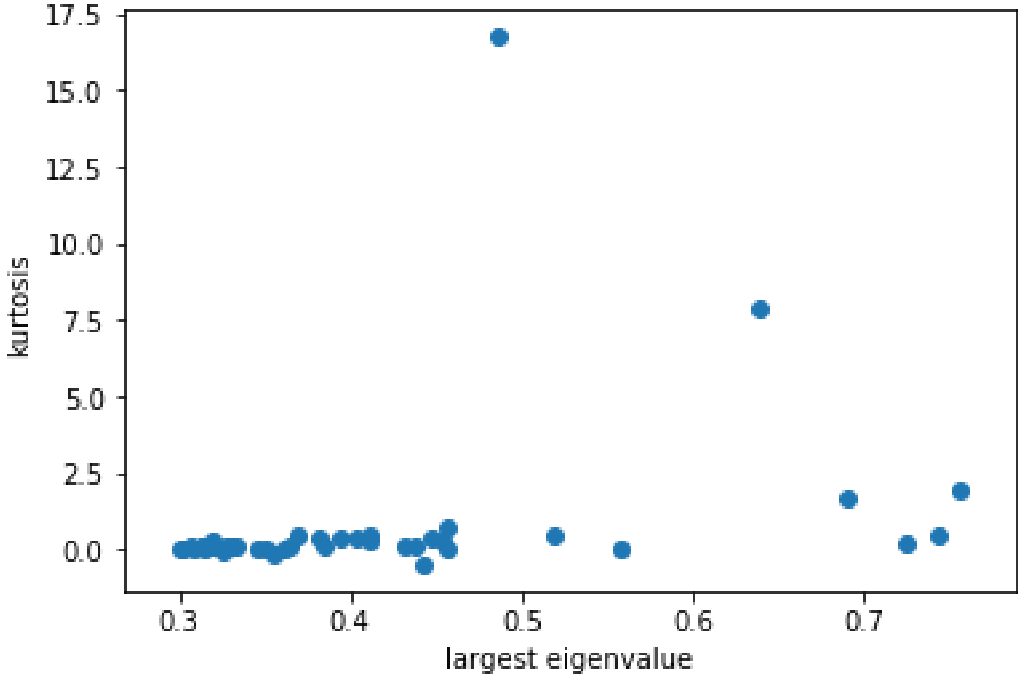
[10]:

	channel_name	cell_type	largest_eig_values	kurtosis
2	16a	ON/OFF	0.315114	-0.001641
5	22d	ON/OFF	0.361678	-0.022156
12	26c	ON/OFF	0.325113	-0.052811
23	42b	ON/OFF	0.354532	-0.142625
34	67a	ON/OFF	0.346075	-0.012561
39	85b	ON/OFF	0.443079	-0.481039
48	17b	NaN	NaN	-0.007362
51	22a	NaN	NaN	-0.025896
55	26a	NaN	NaN	-0.037126
58	28b	NaN	NaN	-0.117339
60	33a	NaN	NaN	-0.003678
61	33c	NaN	NaN	-0.024368
64	36a	NaN	NaN	-0.051979
67	36d	NaN	NaN	-0.098224
76	43b	NaN	NaN	-0.102695
79	45b	NaN	NaN	-0.005283
82	46b	NaN	NaN	-0.061190
101	71d	NaN	NaN	-0.126725
103	72b	NaN	NaN	-0.051334
112	85c	NaN	NaN	-0.078257

1.4 eigenvalues & kurtosis

[11]:
```
plt.scatter(results["largest_eig_values"], results["kurtosis"])
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")
```

[11]: Text(0, 0.5, 'kurtosis')

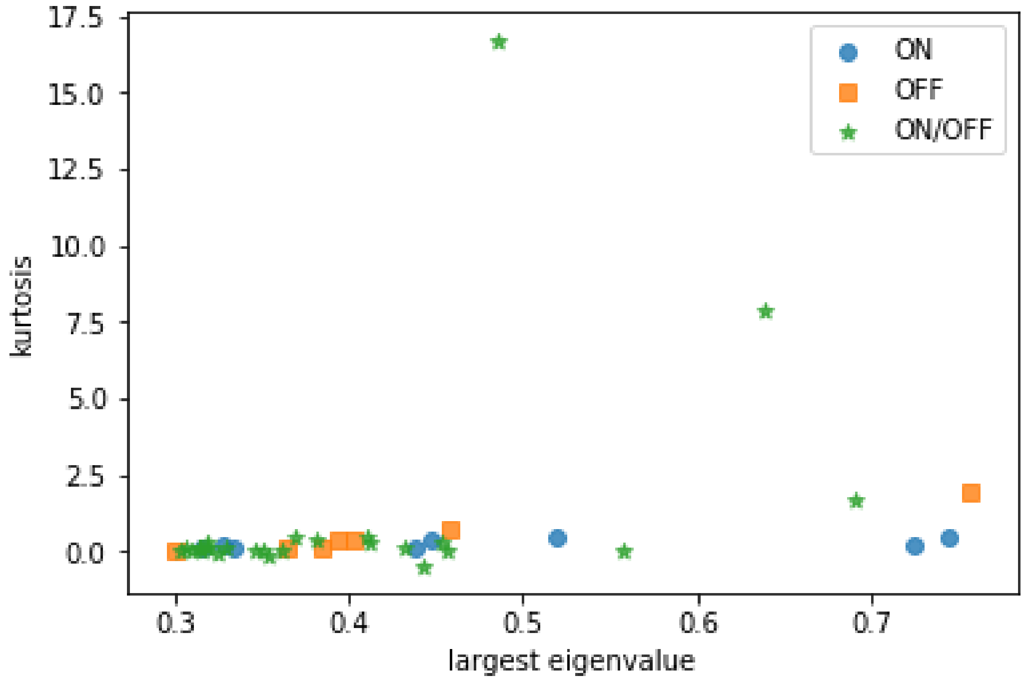


[12]:

```
# plot for each cell type
results_on = results.loc[results["cell_type"]=="ON"]
results_off = results.loc[results["cell_type"]=="OFF"]
results_on_off = results.loc[results["cell_type"]=="ON/OFF"]

# plt.figure(figsize=(12,3))
# plt.subplot(131)
ax=plt.scatter(results_on["largest_eig_values"], results_on["kurtosis"],␣
 ↪marker="o", alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")

# plt.subplot(132)
plt.scatter(results_off["largest_eig_values"], results_off["kurtosis"],␣
 ↪marker="s", alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")

# plt.subplot(133)
plt.scatter(results_on_off["largest_eig_values"], results_on_off["kurtosis"],␣
 ↪marker="*", alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")
```

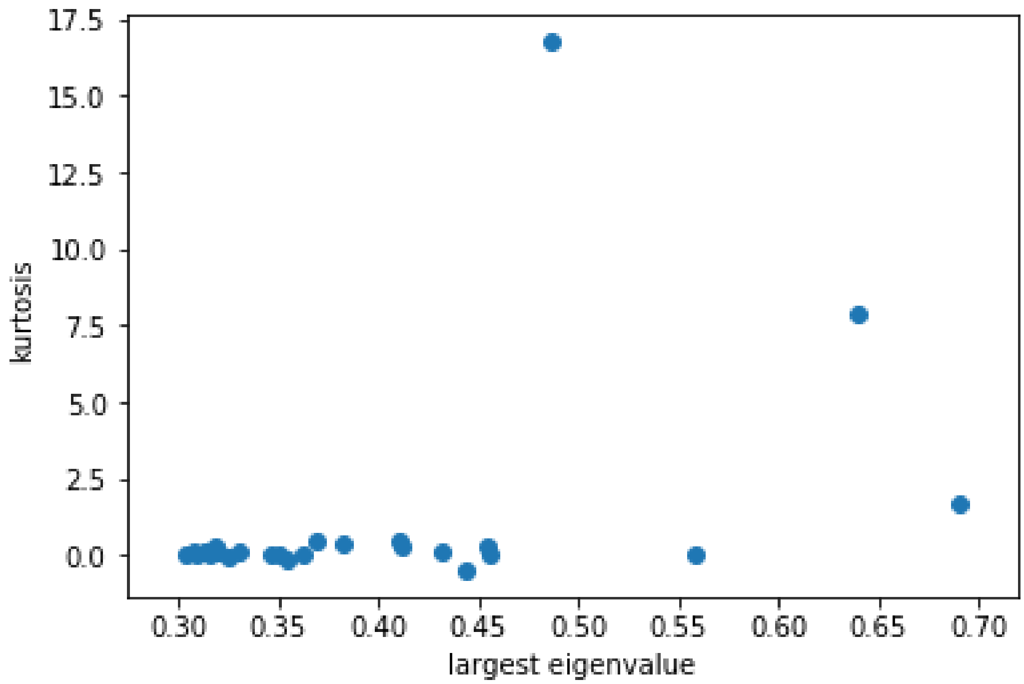
```
plt.legend(["ON", "OFF", "ON/OFF"])
```

[12]: <matplotlib.legend.Legend at 0x1a241ddc50>



[13]:
```
plt.scatter(results_on_off["largest_eig_values"], results_on_off["kurtosis"],␣
 ↪marker="o")# , alpha=0.8)
plt.xlabel("largest eigenvalue")
plt.ylabel("kurtosis")

# plt.legend(["ON", "OFF", "ON/OFF"])
```

[13]: Text(0, 0.5, 'kurtosis')



[14]:
```
results_on_off
```

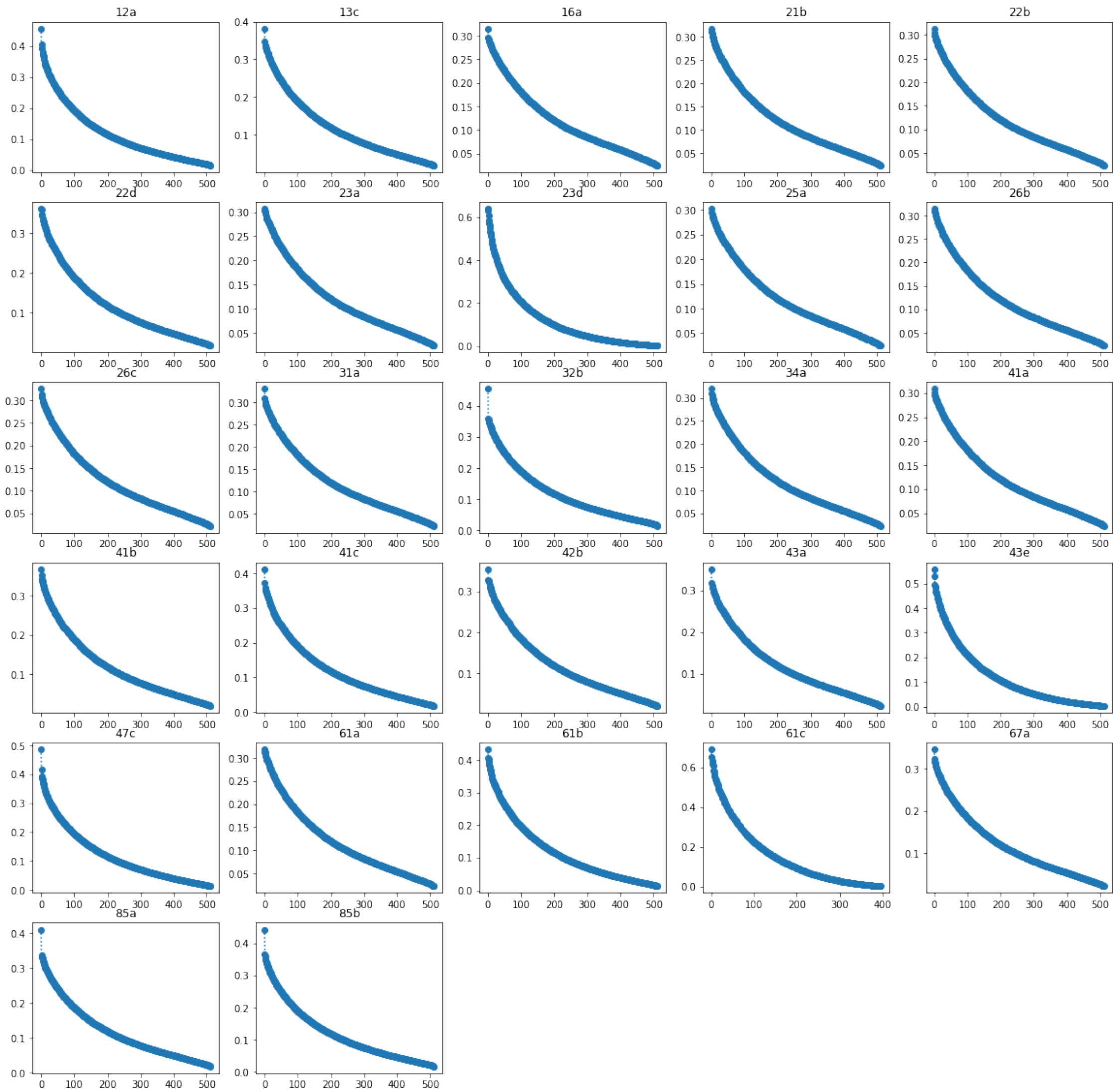
[14]:

	channel_name	cell_type	largest_eig_values	kurtosis
0	12a	ON/OFF	0.455981	0.047361
1	13c	ON/OFF	0.381917	0.351167
2	16a	ON/OFF	0.315114	-0.001641
3	21b	ON/OFF	0.318001	0.133269
4	22b	ON/OFF	0.313448	0.105355
5	22d	ON/OFF	0.361678	-0.022156
6	23a	ON/OFF	0.307121	0.122073
8	23d	ON/OFF	0.638946	7.884196
9	25a	ON/OFF	0.302862	0.006930
11	26b	ON/OFF	0.314308	0.072191
12	26c	ON/OFF	0.325113	-0.052811
14	31a	ON/OFF	0.329710	0.095240
16	32b	ON/OFF	0.453828	0.312954
18	34a	ON/OFF	0.319379	0.071693
19	41a	ON/OFF	0.308823	0.049786
20	41b	ON/OFF	0.368720	0.452510
21	41c	ON/OFF	0.411937	0.251246
23	42b	ON/OFF	0.354532	-0.142625
24	43a	ON/OFF	0.350669	0.031971
25	43e	ON/OFF	0.557959	0.003227
29	47c	ON/OFF	0.485867	16.724623

30	61a	ON/OFF	0.318565	0.298006
31	61b	ON/OFF	0.431893	0.111418
32	61c	ON/OFF	0.690777	1.681480
34	67a	ON/OFF	0.346075	-0.012561
38	85a	ON/OFF	0.410610	0.461787
39	85b	ON/OFF	0.443079	-0.481039

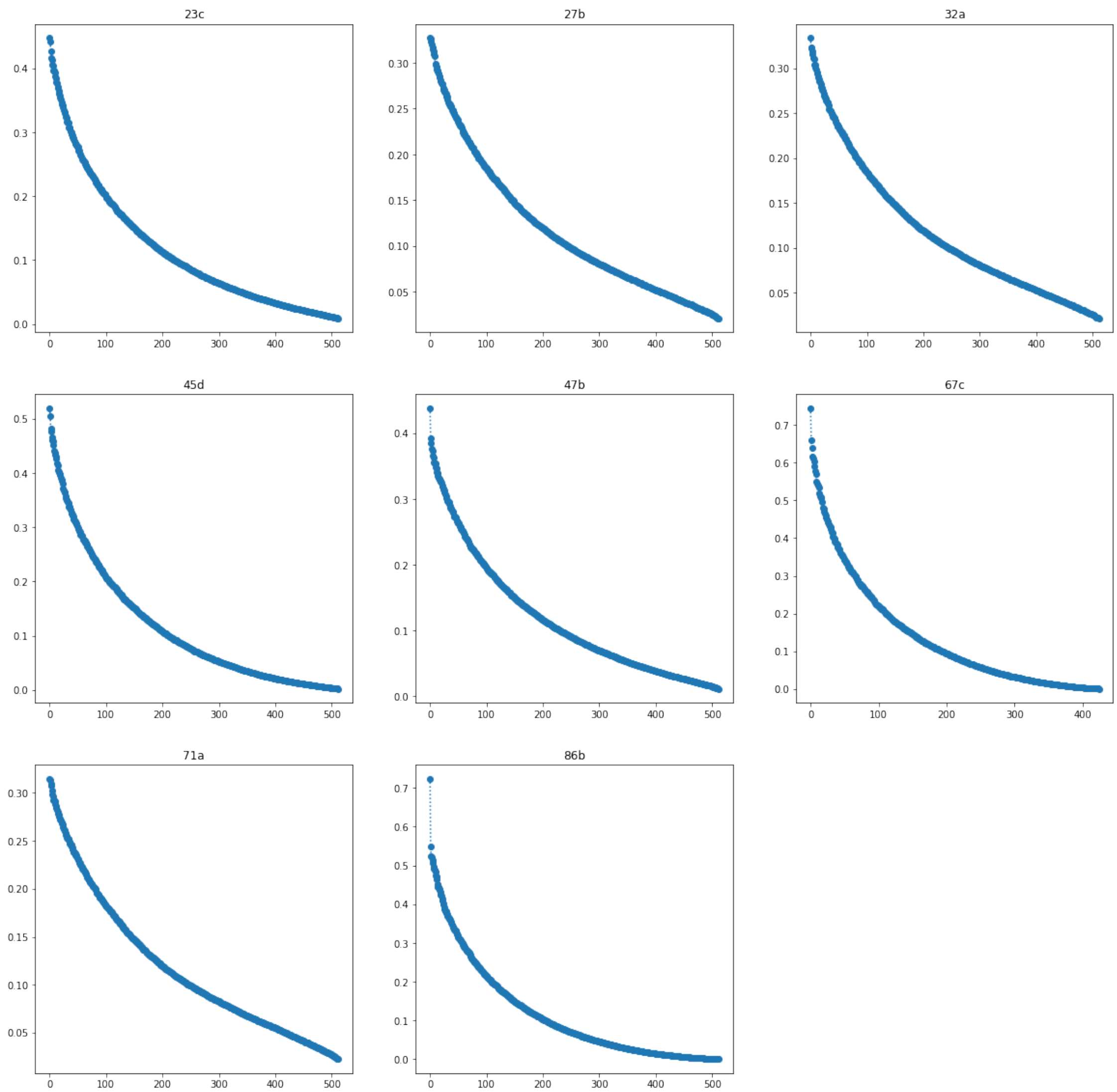
[15]:

```
# plot eigenvalues for ON/OFF cells

def plot_eigenvalues(all_eigen_values, channel_names):

    num_subplots=len(channel_names)
    num_row = int(np.ceil(np.sqrt(num_subplots)))
    num_col = int(np.ceil(num_subplots / num_row))

    plt.figure(figsize=(20,20))
    for i, channel_name in enumerate(channel_names):
        plt.subplot(num_row, num_col,i+1)
        plt.plot(all_eig_values[channel_name],"o:")
        plt.title(channel_name)

plot_eigenvalues(all_eig_values, results_on_off["channel_name"])
plt.savefig("{}/eigenvalues_on_off.png".format(folder_name))
```

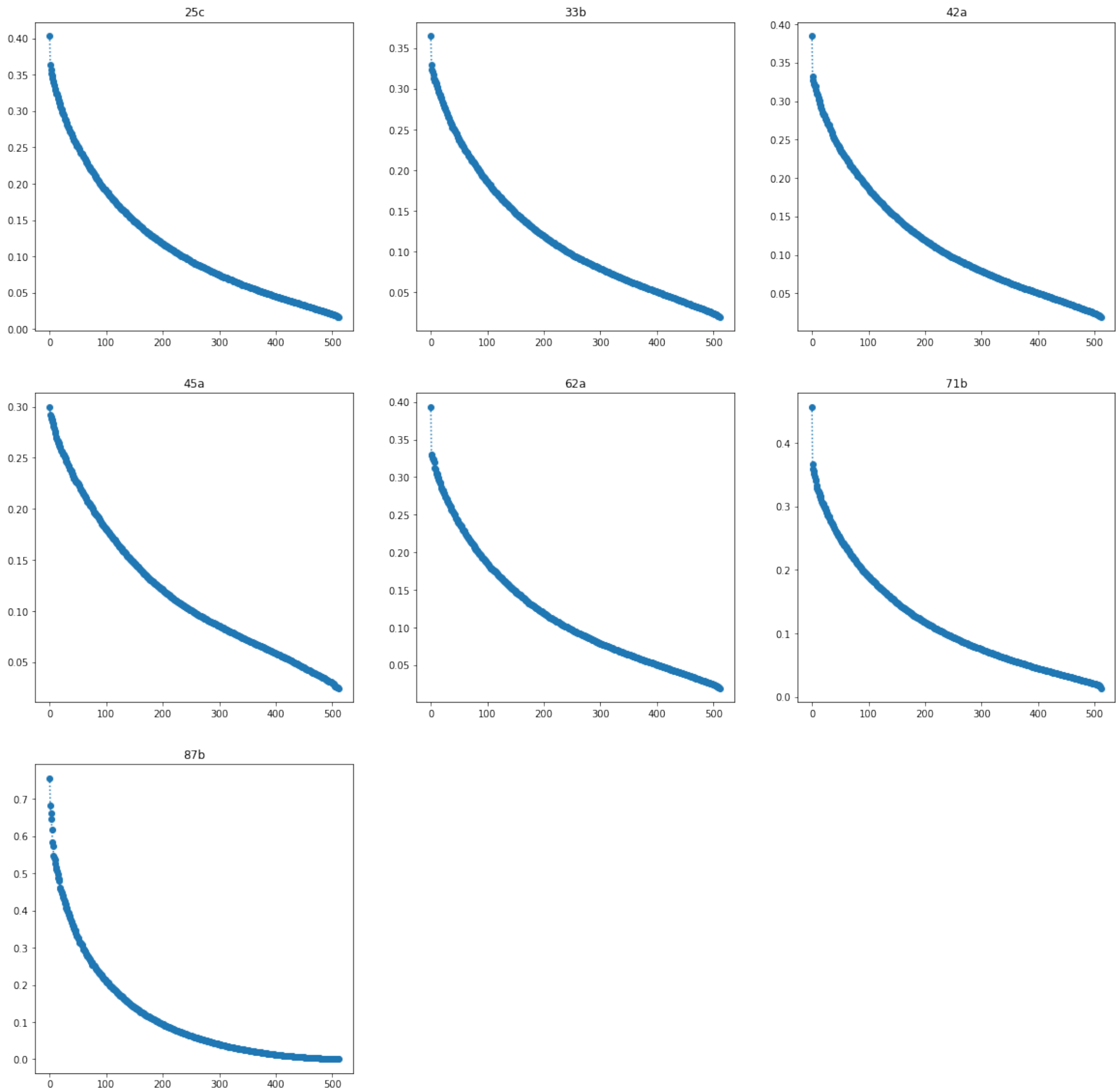


[16]:

```
plot_eigenvalues(all_eig_values, results_on["channel_name"])
plt.savefig("{}/eigenvalues_on.png".format(folder_name))
```



[17]:
```
plot_eigenvalues(all_eig_values, results_off["channel_name"])
plt.savefig("{}/eigenvalues_off.png".format(folder_name))
```


25c
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40
0 100 200 300 400 500
33b
0.05 0.10 0.15 0.20 0.25 0.30 0.35
0 100 200 300 400 500
42a
0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40
0 100 200 300 400 500
45a
0.05 0.10 0.15 0.20 0.25 0.30
0 100 200 300 400 500
62a
0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40
0 100 200 300 400 500
71b
0.0 0.1 0.2 0.3 0.4
0 100 200 300 400 500
87b
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
0 100 200 300 400 500

[ ]: