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"  # NEED CELL TYPE
#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:
```

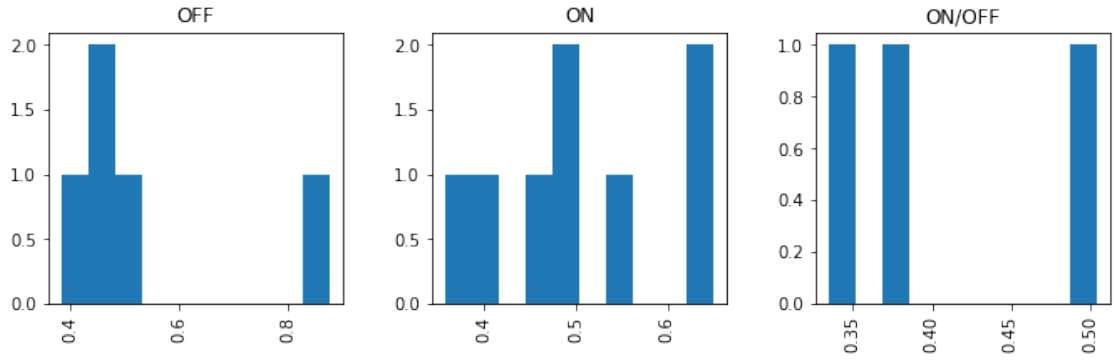
```
 <KeysViewHDF5 ['#refs#', 'channel_names', 'height', 'sampling_rate',
'spike_train', 'stim', 'width']>
Shape of the array stim:  (64, 9000)
Shape of the array spike_train:  (94, 9000)
length of the list channel_names:  94
sampling_rate:  10.0
```

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
```

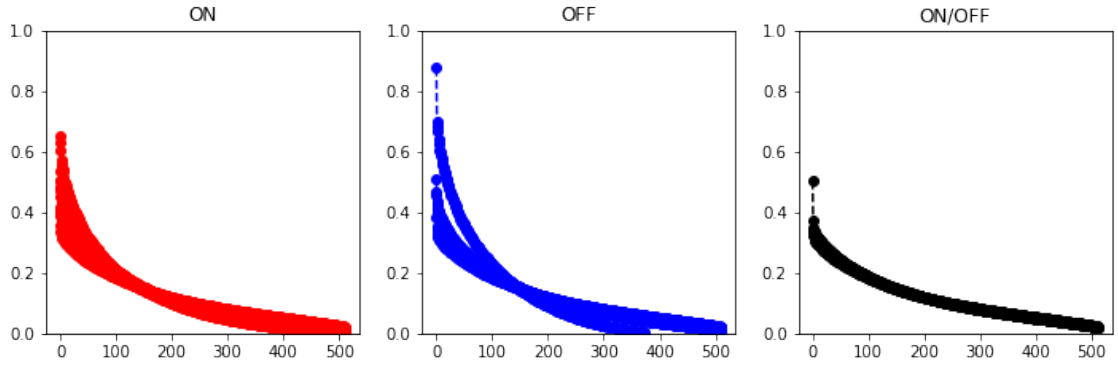
[4]:
```
# convert to DataFrame
eig = pd.DataFrame({"channel_name": channel_names, "largest_eig_values":
 ↪largest_eig_values})

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 0x1a1b940d50>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x10601e890>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x1060d7b50>],
      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')



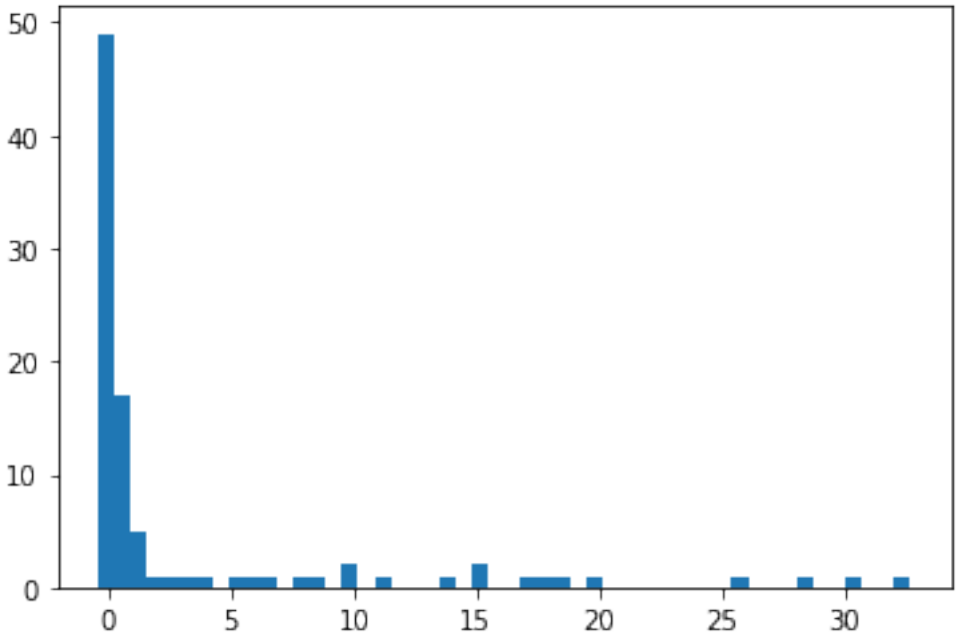
1.3 result - kurtosis

[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	0.589665
1	12b	-0.075471
2	12c	0.236041
3	12d	-0.132510
4	13a	0.239566
..	…	…
89	83a	28.587453
90	83b	5.070189
91	83c	-0.397590
92	86a	0.244055
93	87a	6.856264

[94 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
```

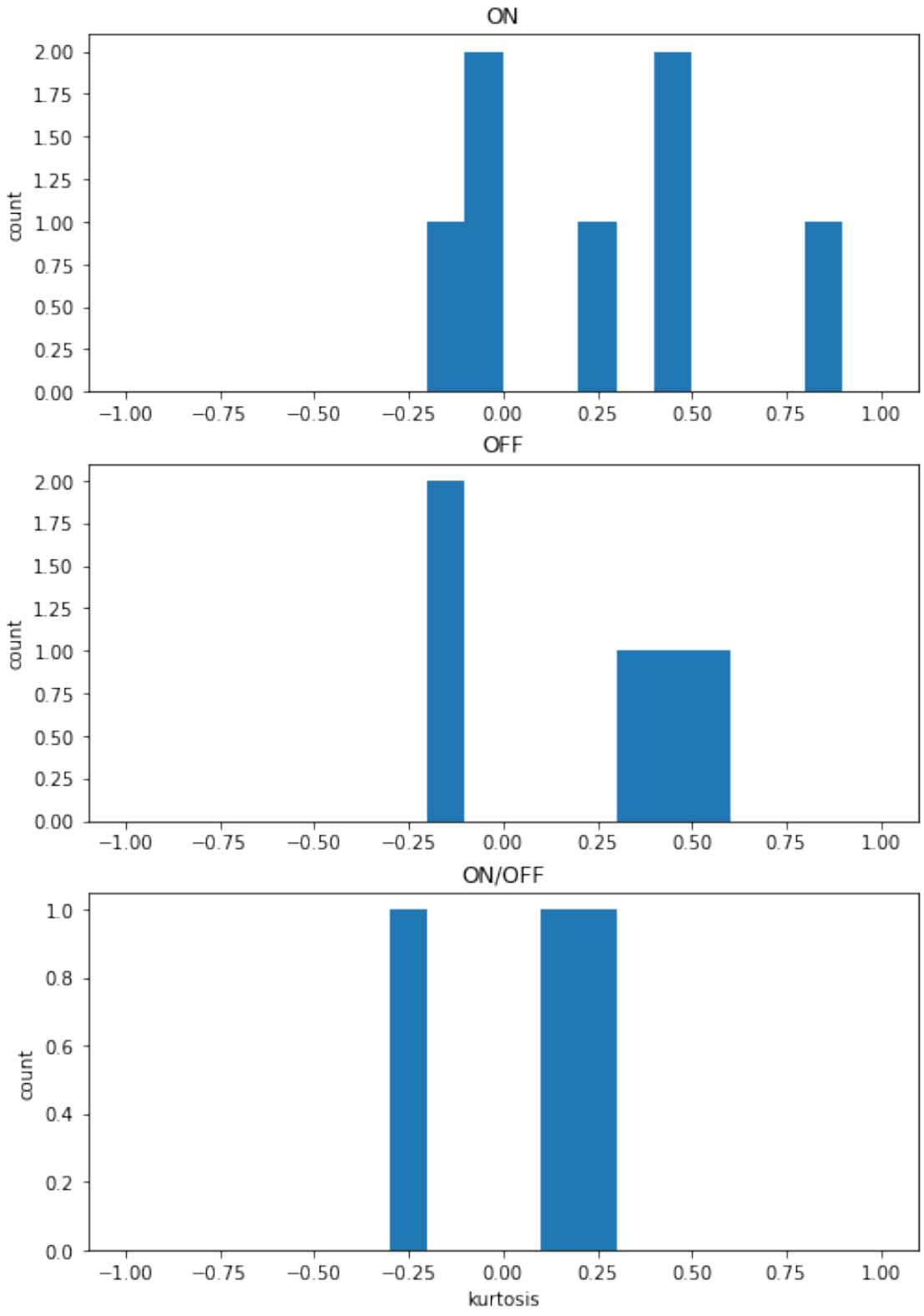
[8]:

	channel_name	cell_type	largest_eig_values	kurtosis
0	12a	OFF	0.458282	0.589665
1	12c	ON	0.536263	0.236041
2	12d	ON	0.413047	-0.132510
3	13a	ON/OFF	0.334611	0.239566
4	14a	ON	0.501542	0.448637
..	…	…	…	…
89	83a	NaN	NaN	28.587453
90	83b	NaN	NaN	5.070189
91	83c	NaN	NaN	-0.397590
92	86a	NaN	NaN	0.244055
93	87a	NaN	NaN	6.856264

[94 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
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
OFF
count
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
ON/OFF
count
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
kurtosis

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

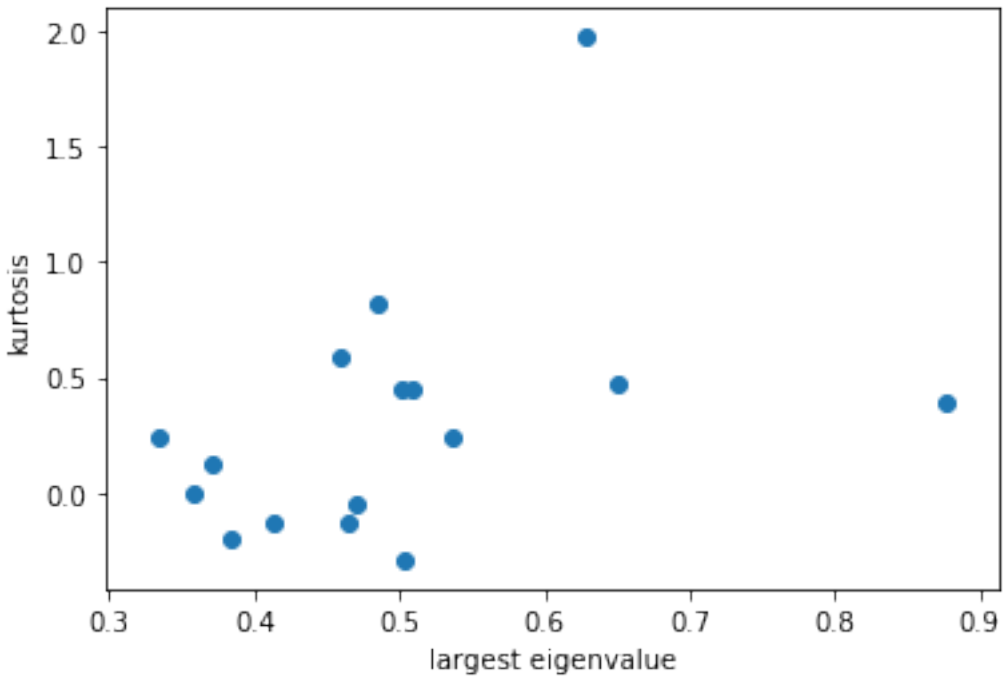
[10]:

	channel_name	cell_type	largest_eig_values	kurtosis
2	12d	ON	0.413047	-0.132510
5	22a	ON	0.357162	-0.005451
6	22b	OFF	0.384337	-0.197462
11	31b	ON	0.469850	-0.044110
12	31c	OFF	0.464234	-0.126053
14	32c	ON/OFF	0.503917	-0.289700
16	12b	NaN	NaN	-0.075471
23	24b	NaN	NaN	-0.050109
25	24d	NaN	NaN	-0.166838
27	25b	NaN	NaN	-0.171246
28	25d	NaN	NaN	-0.023206
42	33b	NaN	NaN	-0.043693
43	33c	NaN	NaN	-0.027868
46	36c	NaN	NaN	-0.151526
59	47a	NaN	NaN	-0.402670
66	51b	NaN	NaN	-0.129846
71	57a	NaN	NaN	-0.161648
77	62a	NaN	NaN	-0.036036
78	62b	NaN	NaN	-0.192468
82	73a	NaN	NaN	-0.092588
91	83c	NaN	NaN	-0.397590

1.4 eigenvalues & kurtosis

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

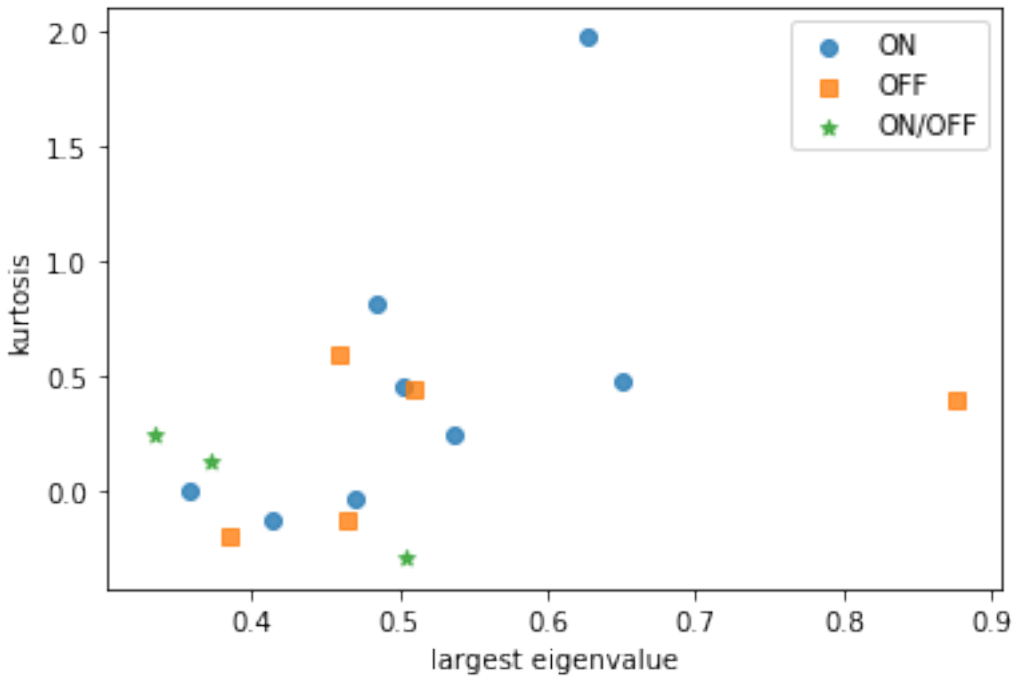
[11]:
```
Text(0, 0.5, 'kurtosis')
```



[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")
```

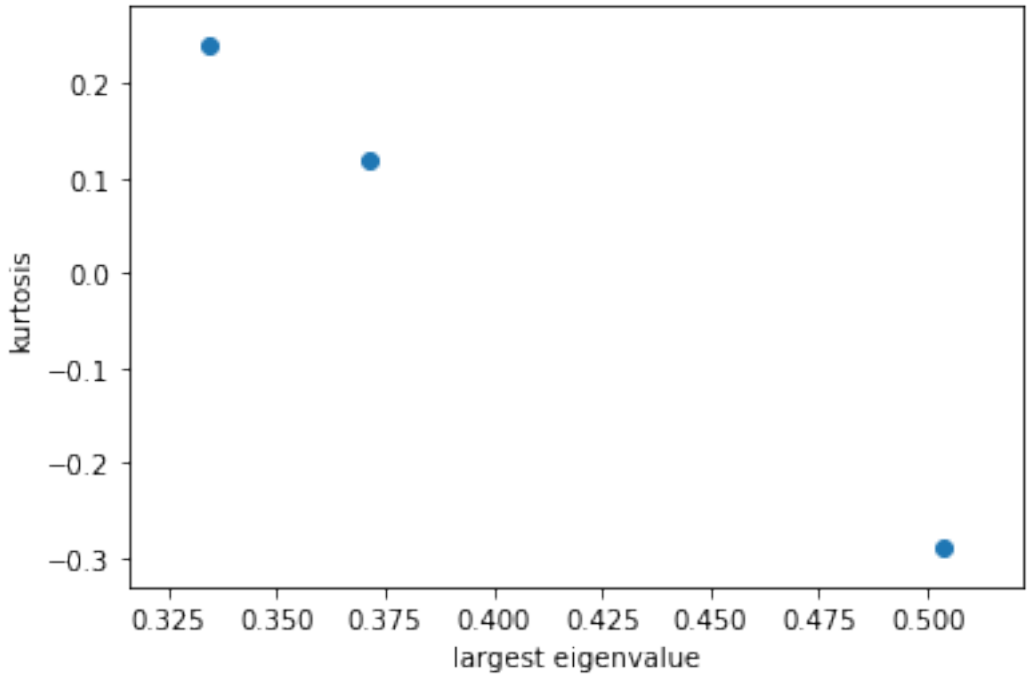
```
plt.legend(["ON", "OFF", "ON/OFF"])
```

[12]: <matplotlib.legend.Legend at 0x1a1cfd88d0>



[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')

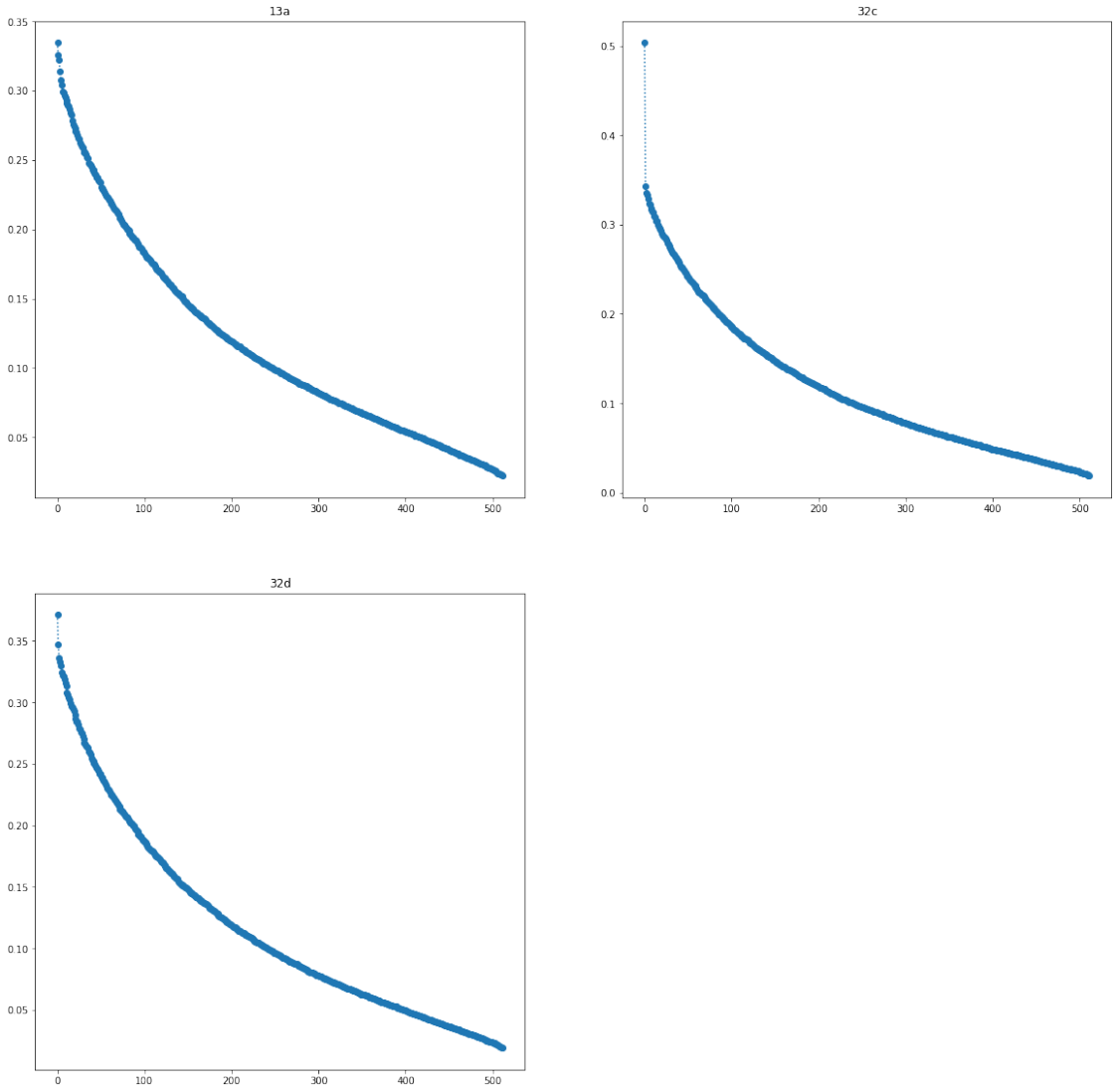


[14]:
```
results_on_off
```

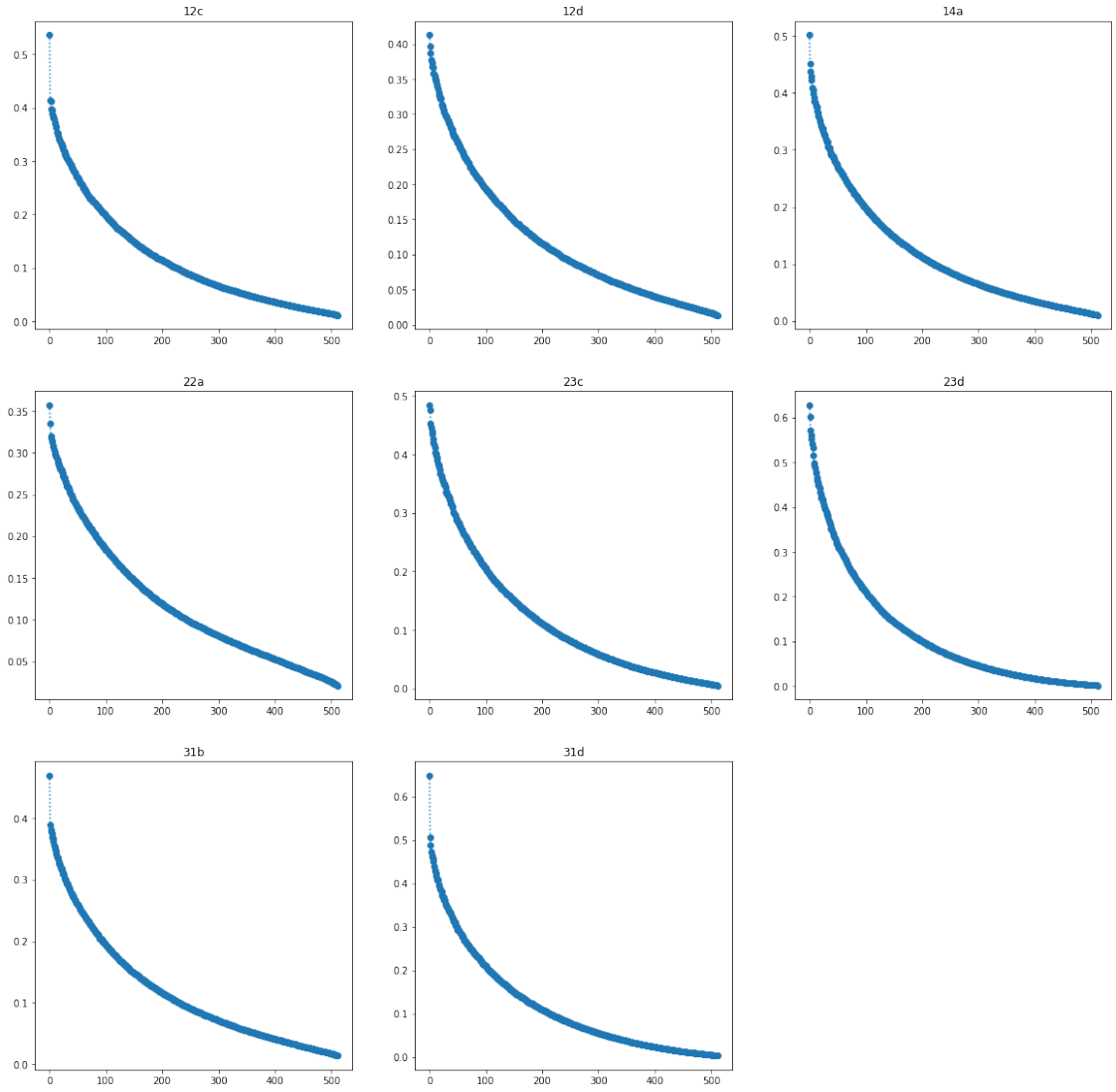
[14]:

	channel_name	cell_type	largest_eig_values	kurtosis
3	13a	ON/OFF	0.334611	0.239566
14	32c	ON/OFF	0.503917	-0.289700
15	32d	ON/OFF	0.371361	0.119760

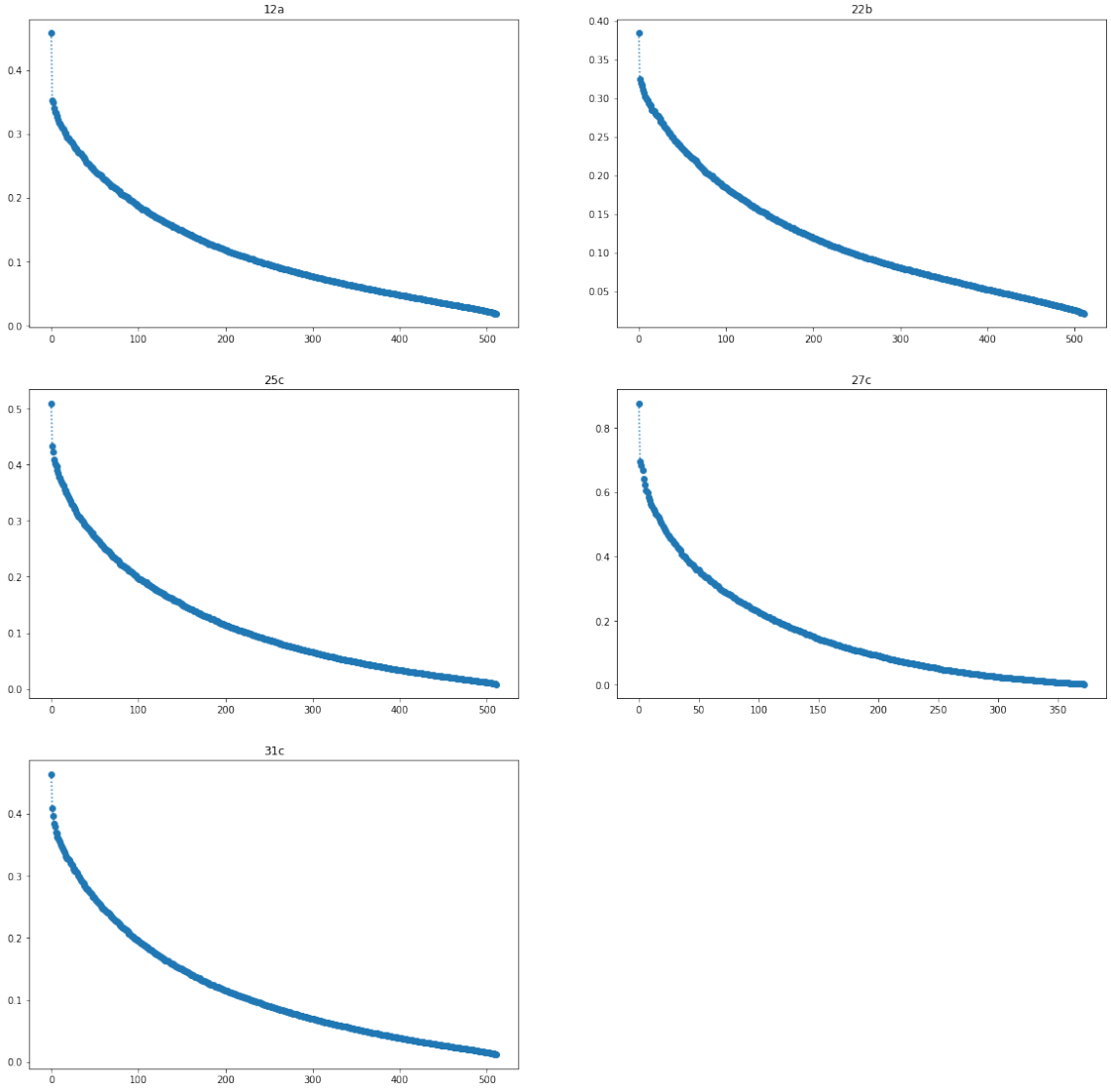
[15]:
```
# plot eigenvalues for ON/OFF cells

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))
```


12a
22b
25c
27c
31c

[ ]: