# Script Demo for EEG/MEG based on NeuroRA

by [Zitong Lu](zitonglu1996.github.io)

April 09, 2020

Users can learn how to use [NeuroRA](https://zitonglu1996.github.io/NeuroRA/) to do research by representational analysis based on EEG/MEG data.

This demo is based on the publicly available visual-92-categories-task MEG datasets. (*Reference:* [Cichy, R. M., Pantazis, D., & Oliva, A. “Resolving human object recognition in space and time.” Nature neuroscience (2014): 17(3), 455-462.](https://www.nature.com/articles/nn.3635)) [MNE-Python](https://mne.tools/stable/index.html) has been used to load this dataset.

# -\*- coding: utf-8 -\*-

' a demo based on visual-92-categories-task MEG data '

# Users can learn how to use Neurora to do research based on EEG/MEG etc data.

\_\_author\_\_ = 'Zitong Lu'

**import** **numpy** **as** **np**

**import** **os.path** **as** **op**

**from** **pandas** **import** read\_csv

**import** **mne**

**from** **mne.io** **import** read\_raw\_fif

**from** **mne.datasets** **import** visual\_92\_categories

**from** **neurora.nps\_cal** **import** nps

**from** **neurora.rdm\_cal** **import** eegRDM

**from** **neurora.rdm\_corr** **import** rdm\_correlation\_spearman

**from** **neurora.corr\_cal\_by\_rdm** **import** rdms\_corr

**from** **neurora.rsa\_plot** **import** plot\_rdm, plot\_corrs\_by\_time, plot\_nps\_hotmap, plot\_corrs\_hotmap

"""\*\*\*\*\*\*\*\*\*\* Section 1: loading example data \*\*\*\*\*\*\*\*\*\*"""

""" Here, we use MNE-Python toolbox for loading data and processing """

""" you can learn this process from MNE-Python (https://mne-tools.github.io/stable/index.html) """

data\_path = visual\_92\_categories.data\_path()

fname = op.join(data\_path, 'visual\_stimuli.csv')

conds = read\_csv(fname)

conditions = []

**for** c **in** conds.values:

cond\_tags = list(c[:**2**])

cond\_tags += [('not-' **if** i == **0** **else** '') + conds.columns[k]

**for** k, i **in** enumerate(c[**2**:], **2**)]

conditions.append('/'.join(map(str, cond\_tags)))

event\_id = dict(zip(conditions, conds.trigger + **1**))

print(event\_id)

sub\_id = [**0**, **1**, **2**]

megdata = np.zeros([**3**, **92**, **306**, **1101**], dtype=np.float32)

subindex = **0**

**for** id **in** sub\_id:

fname = op.join(data\_path, 'sample\_subject\_'+str(id)+'\_tsss\_mc.fif')

raw = read\_raw\_fif(fname)

events = mne.find\_events(raw, min\_duration=.**002**)

events = events[events[:, **2**] <= **92**]

subdata = np.zeros([**92**, **306**, **1101**], dtype=np.float32)

**for** i **in** range(**92**):

epochs = mne.Epochs(raw, events=events, event\_id=i + **1**, baseline=**None**,

tmin=-**0.1**, tmax=**1**, preload=**True**)

data = epochs.average().data

subdata[i] = data

megdata[subindex] = subdata

subindex = subindex + **1**

# the shape of MEG data: megdata is [3, 92, 306, 1101]

# n\_subs = 3, n\_conditions = 92, n\_channels = 306, n\_timepoints = 1101 (-100ms to 1000ms)

"""\*\*\*\*\*\*\*\*\*\* Section 2: Preprocessing \*\*\*\*\*\*\*\*\*\*"""

# shape of megdata: [n\_subs, n\_cons, n\_chls, n\_ts] -> [n\_cons, n\_subs, n\_chls, n\_ts]

megdata = np.transpose(megdata, (**1**, **0**, **2**, **3**))

# shape of megdata: [n\_cons, n\_subs, n\_chls, n\_ts] -> [n\_cons, n\_subs, n\_trials, n\_chls, n\_ts]

# here data is averaged, so set n\_trials = 1

megdata = np.reshape(megdata, [**92**, **3**, **1**, **306**, **1101**])

"""\*\*\*\* Section 2: Calculating the neural pattern similarity \*\*\*\*"""

# Get data under different condition

# Here we calculate the neural pattern similarity (NPS) between two stimulus

# Seeing Humanface vs. Seeing Non-Humanface

# get data under "humanface" condtion

megdata\_humanface = megdata[**12**:**24**]

# get data under "nonhumanface" condition

megdata\_nonhumanface = megdata[**36**:**48**]

# Average the data

avg\_megdata\_humanface = np.average(megdata\_humanface, axis=**0**)

avg\_megdata\_nonhumanface = np.average(megdata\_nonhumanface, axis=**0**)

# Create NPS input data

# Here we extract the data from first 5 channels between 0ms and 1000ms

nps\_data = np.zeros([**2**, **3**, **1**, **5**, **1000**]) # n\_cons=2, n\_subs=3, n\_chls=5, n\_ts=1000

nps\_data[**0**] = avg\_megdata\_humanface[:, :, :**5**, **100**:**1100**] # the start time of the data is -100ms

nps\_data[**1**] = avg\_megdata\_nonhumanface[:, :, :**5**, **100**:**1100**] # so 100:1200 corresponds 0ms-1000ms

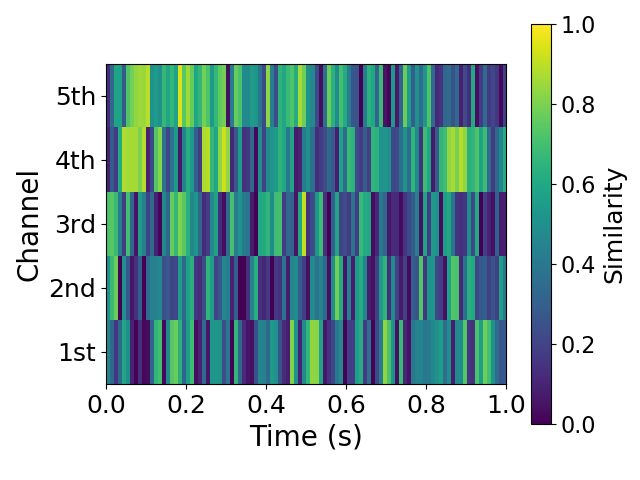
# Calculate the NPS with a 10ms time-window

# (raw sampling requency is 1000Hz, so here time\_win=10ms/(1s/1000Hz)/1000=10)

nps = nps(nps\_data, time\_win=**10**)

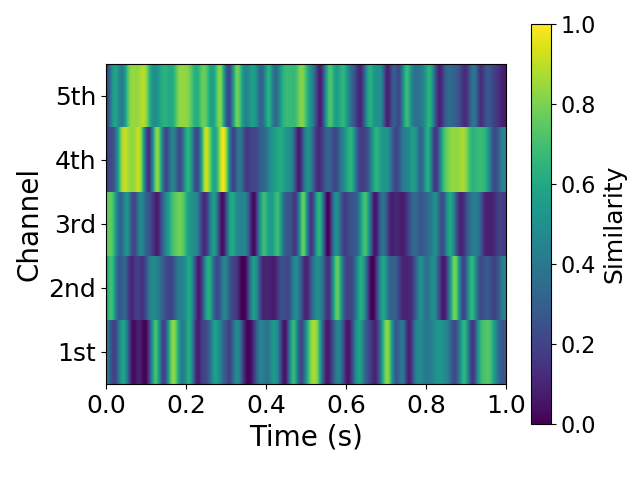
# Plot the NPS results

plot\_nps\_hotmap(nps, time\_unit=[**0**, **0.01**], abs=**True**)



# Smooth the results and plot

plot\_nps\_hotmap(nps, time\_unit=[**0**, **0.01**], abs=**True**, smooth=**True**)



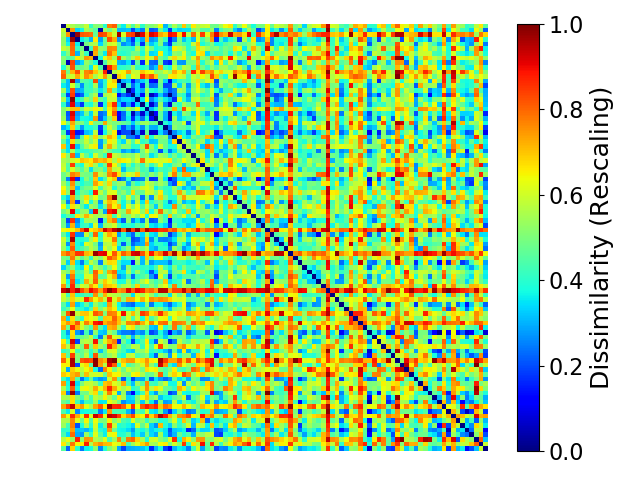
"""\*\*\*\*\* Section 3: Calculating single RDM and Plotting \*\*\*\*\*"""

# Calculate the RDM based on the data during 190ms-210ms

rdm = eegRDM(megdata[:, :, :, :, **290**:**310**])

# Plot this RDM

plot\_rdm(rdm, rescale=**True**)



"""\*\*\*\*\*\*\*\* Section 4: Calculating RDMs and Plotting \*\*\*\*\*\*\*\*"""

# Calculate the RDMs by a 10ms time-window

# (raw sampling requency is 1000Hz, so here time\_win=10ms/(1s/1000Hz)/1000=10)

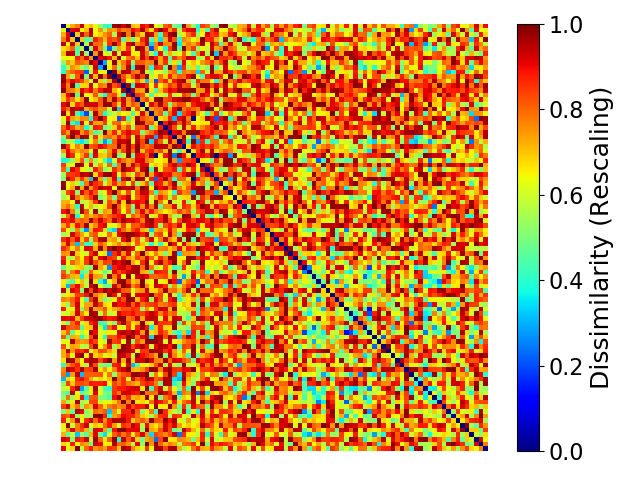
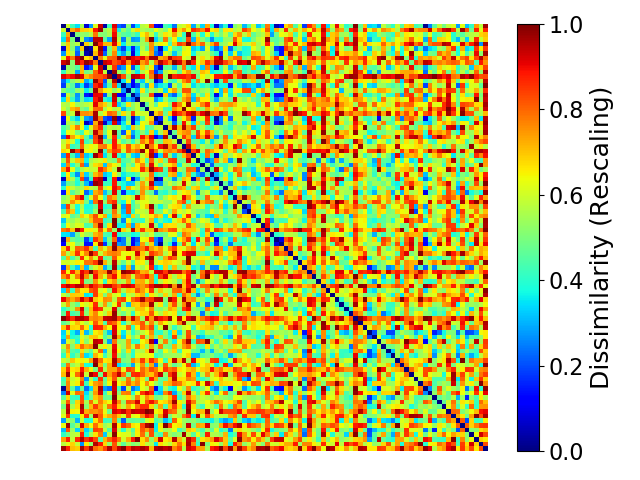
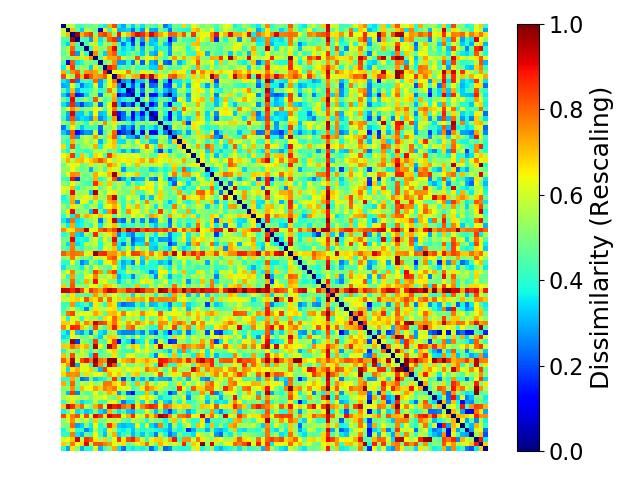
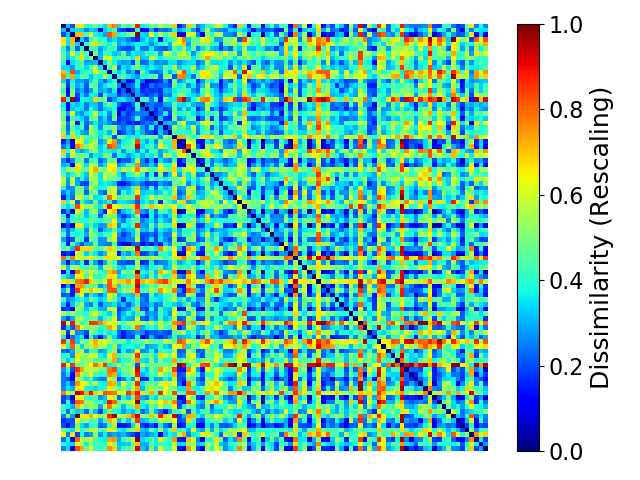
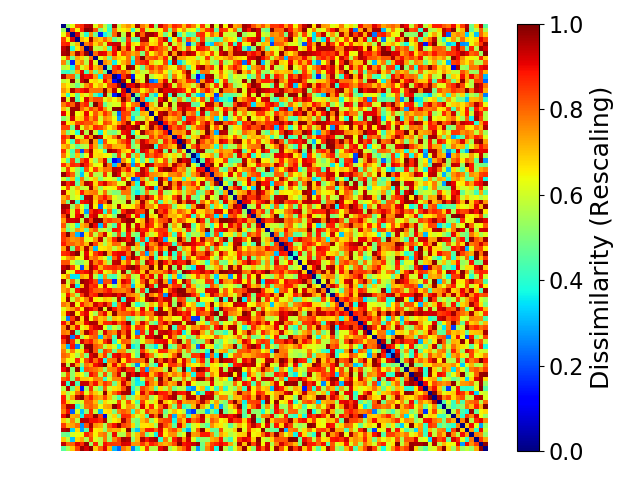
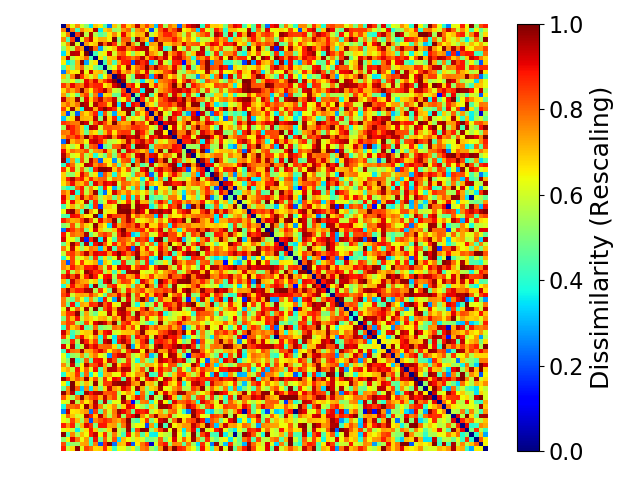
rdms = eegRDM(megdata, time\_win=**10**, time\_opt=**1**)

# Plot the RDM of 0ms, 50ms, 100ms, 150ms, 200ms

times = [**0**, **10**, **20**, **30**, **40**, **50**]

**for** t **in** times:

plot\_rdm(rdms[t], rescale=**True**)



"""\*\*\*\* Section 5: Calculating the Similarity between two RDMs \*\*\*\*"""

# RDM of 200ms

rdm\_sample1 = rdms[**30**]

# RDM of 800ms

rdm\_sample2 = rdms[**90**]

# calculate the correlation coefficient between these two RDMs

corr = rdm\_correlation\_spearman(rdm\_sample1, rdm\_sample2, rescale=**True**)

print(corr)

SpearmanrResult(correlation=0.02665680483550596, pvalue=0.08462337954774739)

"""\*\*\*\*\* Section 6: Calculating the Similarity and Plotting \*\*\*\*\*"""

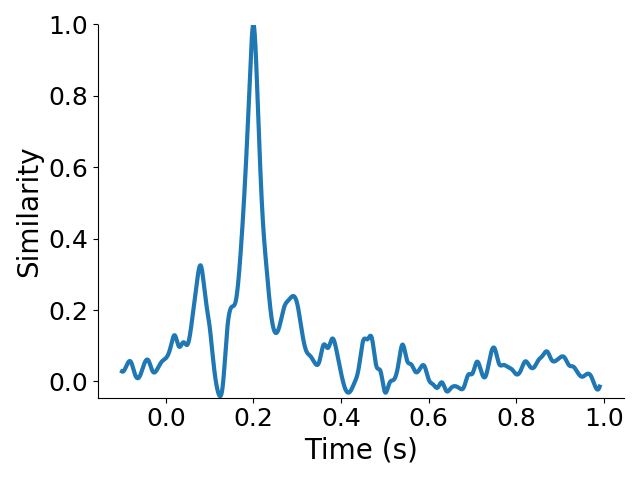
# Calculate the representational similarity between 200ms and all the time points

corrs1 = rdms\_corr(rdm\_sample1, rdms)

# Plot the corrs1

corrs1 = np.reshape(corrs1, [**1**, **110**, **2**])

plot\_corrs\_by\_time(corrs1, time\_unit=[-**0.1**, **0.01**])



# Calculate and Plot multi-corrs

corrs2 = rdms\_corr(rdm\_sample2, rdms)

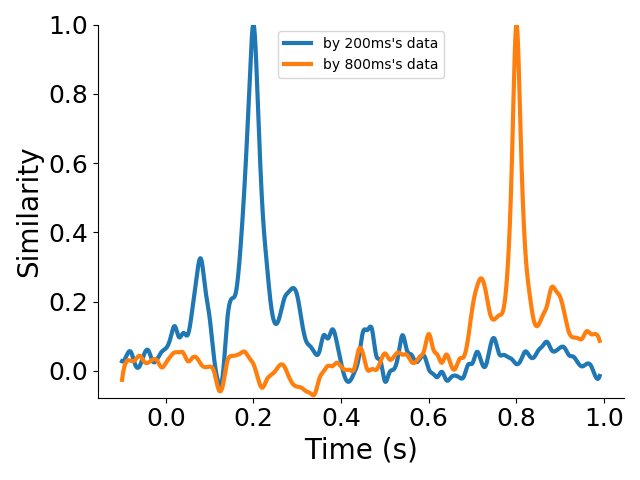
corrs = np.zeros([**2**, **110**, **2**])

corrs[**0**] = corrs1

corrs[**1**] = corrs2

labels = ["by 200ms's data", "by 800ms's data"]

plot\_corrs\_by\_time(corrs, labels=labels, time\_unit=[-**0.1**, **0.01**])



"""\*\*\*\*\* Section 7: Calculating the RDMs for each channels \*\*\*\*\*"""

# Calculate the RDMs for the first six channels by a 10ms time-window between 0ms and 1000ms

rdms\_chls = eegRDM(megdata[:, :, :, :**6**, **100**:**1100**], chl\_opt=**1**, time\_opt=**1**, time\_win=**10**)

# Create a 'human-related' coding model RDM

model\_rdm = np.ones([**92**, **92**])

**for** i **in** range(**92**):

**for** j **in** range(**92**):

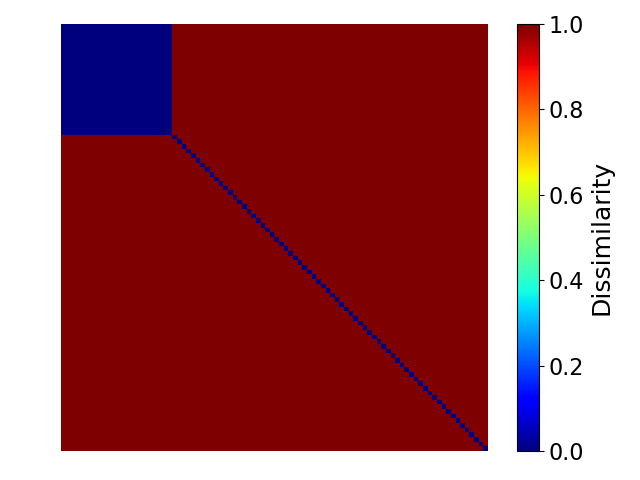
**if** (i < **24**) **and** (j < **24**):

model\_rdm[i, j] = **0**

model\_rdm[i, i] = **0**

# Plot this coding model RDM

plot\_rdm(model\_rdm)

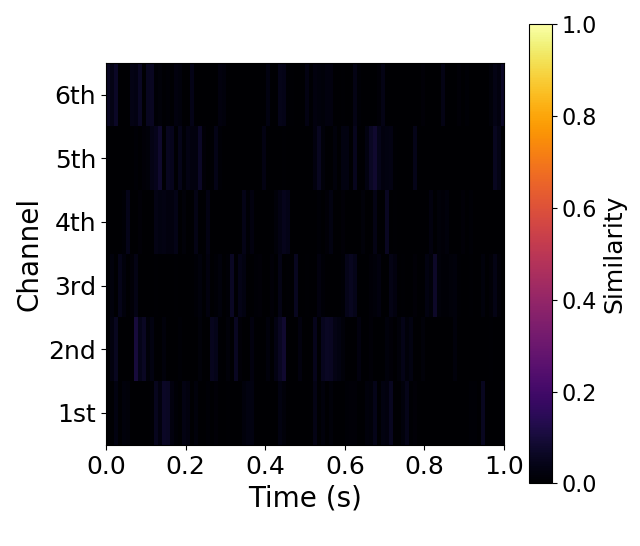


# Calculate the representational similarity between the neural activities and the coding model for each channel

corrs\_chls = rdms\_corr(model\_rdm, rdms\_chls)

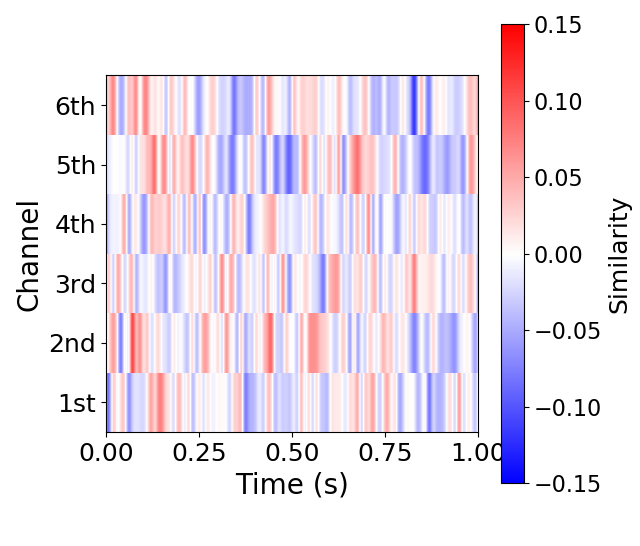
# Plot the representational similarity results

plot\_corrs\_hotmap(corrs\_chls, time\_unit=[**0**, **0.01**])



# Set more parameters and re-plot

plot\_corrs\_hotmap(corrs\_chls, time\_unit=[**0**, **0.01**], lim=[-**0.15**, **0.15**], smooth=**True**, cmap='bwr')



Users can see more details below:

GitHub Website: <https://github.com/neurora/NeuroRA>

NeuroRA Website: <https://neuroragithub.io/NeuroRA/>

PyPi Website: <https://pypi.org/project/neurora/>

BioRxiv Paper: <https://doi.org/10.1101/2020.03.25.008086>