Source localization with MNE

MEG skills - Source localization with MNE

MNE : Basics of source localization

Author : Alexandre Gramfort alexandre.gramfort@telecom-paristech.fr

In this session we're going to **compute source estimates** on the **somatosensory dataset** used on monday for preprocessing. We will use the boundary element method (BEM) for forward modeling.
The lines below assume that you have already:

- generated the BEM surfaces e.g. using mne_watershed_bem
- done the coregistration with mne_analyze which generated the -trans.fif file (somstim_raw-trans.fif)
- generated a source space e.g. using mne_setup_source_space --oct 6. This generated the file called daniel-oct-6-src.fif
- computed the forward operator (gain matrix) `somstim-meg-oct-6-fwd.fif` using `mne_setup_forward_model --homog --surf --ico 4` and `mne_do_forward_solution --mindist 5 --spacing oct-6 --meas somstim_raw.fif --mri somstim_raw-trans.fif --bem daniel-5120 --megonly --overwrite --fwd somstim-meg-oct-6-fwd.fif`

A full script looks like this:

```
#!/usr/bin/env bash

export SUBJECTS_DIR=$PWD
export SUBJECT=daniel

# Generate BEM models
mne_watershed_bem --overwrite
cd ${SUBJECTS_DIR}/${SUBJECT}/bem
ln -s watershed/${SUBJECT}_inner_skull_surface${SUBJECT}-
inner_skull.surf
ln -s watershed/${SUBJECT}_outer_skin_surface${SUBJECT}-outer_skin.surf
ln -s watershed/${SUBJECT}_outer_skull_surface${SUBJECT}-
outer_skull.surf
cd -

# Source space
mne_setup_source_space --ico -6 --overwrite

# Prepare for forward computation
mne_setup_forward_model --homog --surf --ico 4

# Generate morph maps for morphing between daniel and fsaverage
mne_make_morph_maps --from ${SUBJECT} --to fsaverage
```

```
# compute forward models for both experimental data

cd ../MEG
mne_do_forward_solution --mindist 5 --spacing oct-6 \
        --meas somstim_raw.fif \
        --mri somstim_raw-trans.fif \
        --bem daniel-5120 --megonly \
        --overwrite --fwd somstim-meg-oct-6-fwd.fif
```

Now you can try to run these commands on your machine or use the ones generated for you and available in the MEG folder. Once you're done you can start with what is next.

```
In [1]:  # add plot inline in the page (not necessary in Spyder)
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

import mne
mne.set_log_level('WARNING')
```

Process MEG data

```
In [2]:  data_path = '/Users/alex/Sync/karolinska_teaching/'
raw_fname = data_path + '/MEG/somstim_raw.fif'

raw = mne.fiff.Raw(raw_fname)
print raw
<Raw  |  n_channels x n_times : 324 x 826000>
```

Apply fix...

```
In [3]:  def fix_info(raw):
    raw.info['chs'][raw.ch_names.index('BIO001')]['kind'] =
mne.fiff.constants.FIFF.FIFFV_EOG_CH
    raw.info['chs'][raw.ch_names.index('BIO002')]['kind'] =
mne.fiff.constants.FIFF.FIFFV_EOG_CH
    raw.info['chs'][raw.ch_names.index('BIO003')]['kind'] =
mne.fiff.constants.FIFF.FIFFV_ECG_CH
    raw.info['chs'][raw.ch_names.index('BIO004')]['kind'] =
mne.fiff.constants.FIFF.FIFFV_ECG_CH

fix_info(raw)
```

Looking at meta data, a.k.a. measurement info, such sampling frequency, channels etc.

```
In [4]:  print raw.info['sfreq']
1000.0
```

Define epochs and compute ERP/ERF

First look for events / triggers

```
In [5]: events = mne.find_events(raw, stim_channel='STI101',
verbose=True)
Reading 0 ... 825999  =      0.000 ...   825.999 secs...
[done]
400 events found
Events id: [1]

In [6]: event_id = dict(som=1)  # event trigger and conditions
tmin = -0.1  # start of each epoch (200ms before the trigger)
tmax = 0.3  # end of each epoch (500ms after the trigger)

picks = mne.fiff.pick_types(raw.info, meg=True, eeg=False, eog=True,
                            exclude='bads')

baseline = (None, 0)  # means from the first instant to t = 0
reject = dict(grad=4000e-13, mag=4e-12, eog=150e-6)

epochs = mne.Epochs(raw, events, event_id, tmin, tmax, proj=True,
                    picks=picks, baseline=baseline, reject=reject,
                    preload=True)
print epochs
<Epochs  |  n_events : 367 (all good), tmin : -0.1 (s), tmax : 0.3 (s),
baseline : (None, 0)>
```

Compute the evoked response

```
In [7]: evoked = epochs.average()

evoked.save(data_path + '/MEG/somstim-ave.fif')

evoked.plot()

# Ugly hack due to acquisition problem when specifying the channel
types
layout = mne.layouts.read_layout('Vectorview-mag.lout')
layout.names = mne.utils._clean_names(layout.names,
remove_whitespace=True)

fig = evoked.plot_topomap(times=np.linspace(0.05, 0.12, 5),
ch_type='mag', layout=layout)
```

One can observe a clean (slightly rotating) dipolar pattern first and then bilateral dipoles. The objective now is to locate these early and later components.

Compute noise covariance

Inverse modeling requires the estimation of a noise covariance matrix. This is used to spatially whiten the data and typically allows to combine different types of sensors (magnetometers, gradiometers, EEG) for source localization.

```
In [8]: noise_cov = mne.compute_covariance(epochs, tmax=-0.01) # stay
away from the stim artifact
print noise_cov.data.shape
(306, 306)

In [9]: figs = mne.viz.plot_cov(noise_cov, raw.info)
```

```
In [10]:  # regularize noise covariance
```

```
noise_cov = mne.cov.regularize(noise_cov, epochs.info,
              mag=0.1, grad=0.1, eeg=0.1, proj=True)
```

Exercise

Recompute the noise covariance without setting proj=True when creating Epochs. What do you observe? Why?

Inverse modeling: MNE and dSPM on evoked and raw data

Import the required functions:

```
In [11]: from mne.forward import read_forward_solution
from mne.minimum_norm import make_inverse_operator, apply_inverse, \
write_inverse_operator
```

Read the forward solution and compute the inverse operator

```
In [12]: fname_fwd = data_path + '/MEG/somstim-meg-oct-6-fwd.fif'
fwd = mne.read_forward_solution(fname_fwd, surf_ori=True)

# compute the inverse operator
info = evoked.info
inverse_operator = make_inverse_operator(info, fwd, noise_cov,
                                         loose=0.2, depth=0.8)

# save the inverse operator to disk
fname_inv = data_path + '/MEG/somsen-meg-oct-6-inv.fif'
write_inverse_operator(fname_inv, inverse_operator)
```

At this point one can you mne_analyze for interactive analysis:
http://martinos.org/mne/stable/manual/analyze.html

Compute the source estimates

The acronym of sources estimates in the MNE software is stc which stands for **source time courses**. The stc files can be visualized with mne_analyze by loading it as overlays.

```
In [13]: method = "dSPM"
snr = 3.
lambda2 = 1. / snr ** 2
stc = apply_inverse(evoked, inverse_operator, lambda2,
                    method=method, pick_ori=None)
print stc
<SourceEstimate  |  8196 vertices, subject : daniel, tmin : -100.0
(ms), tmax : 300.0 (ms), tstep : 1.0 (ms), data size : 8196 x 401>

In [14]: stc.data.shape
```

```
# we have 8196 cortical locations and 401 time points

Out[14]: (8196, 401)

In [15]: print stc.times.shape, np.min(stc.times), np.max(stc.times)
(401,) -0.1 0.3
```

you're done. The lines below show you have to visualize in Python and script figure generation. You'll find exercises at the bottom to go further.
You can browse the examples on inverse modeling at:
http://martinos.org/mne/stable/auto_examples/index.html#inverse-problem-and-source-analysis

Show the result

```
In [16]: # %matplotlib qt4
subjects_dir = data_path + '/subjects'
brain = stc.plot(surface='inflated', hemi='rh',
subjects_dir=subjects_dir)
brain.set_data_time_index(144)  # 221 for S2
brain.scale_data_colormap(fmin=4, fmid=8, fmax=12, transparent=True)
brain.show_view('lateral')
INFO:surfer:Updating smoothing matrix, be patient..
INFO:surfer:Smoothing matrix creation, step 1
INFO:surfer:Smoothing matrix creation, step 2
INFO:surfer:Smoothing matrix creation, step 3
INFO:surfer:Smoothing matrix creation, step 4
INFO:surfer:Smoothing matrix creation, step 5
INFO:surfer:Smoothing matrix creation, step 6
INFO:surfer:Smoothing matrix creation, step 7
INFO:surfer:Smoothing matrix creation, step 8
INFO:surfer:Smoothing matrix creation, step 9
INFO:surfer:Smoothing matrix creation, step 10
INFO:surfer:colormap: fmin=5.00e+00 fmid=1.00e+01 fmax=1.50e+01
transparent=1
INFO:surfer:colormap: fmin=4.00e+00 fmid=8.00e+00 fmax=1.20e+01
transparent=1

Out[16]:((-7.0167092985348768e-15, 90.0, 569.22845458984375, array([
0.,  0.,  0.])), -90.0)

In [17]: brain.save_image('dspm.jpg')
from IPython.display import Image
Image(filename='dspm.jpg', width=600)

Out[17]:
```

"Morphing" data to an average brain for group studies

```
In [18]: stc_fsaverage = stc.morph(subject_to='fsaverage',
subjects_dir=subjects_dir)
```

Visualize on the average brain

```
In [19]: brain_fsaverage =stc_fsaverage.plot(surface='inflated',
hemi='rh',subjects_dir=subjects_dir)
brain_fsaverage.set_data_time_index(171)
brain_fsaverage.scale_data_colormap(fmin=5, fmid=10, fmax=15,
transparent=True)
brain_fsaverage.show_view('lateral')
INFO:surfer:Updating smoothing matrix, be patient..
INFO:surfer:Smoothing matrix creation, step 1
INFO:surfer:Smoothing matrix creation, step 2
INFO:surfer:Smoothing matrix creation, step 3
INFO:surfer:Smoothing matrix creation, step 4
INFO:surfer:Smoothing matrix creation, step 5
INFO:surfer:Smoothing matrix creation, step 6
INFO:surfer:Smoothing matrix creation, step 7
INFO:surfer:Smoothing matrix creation, step 8
INFO:surfer:Smoothing matrix creation, step 9
INFO:surfer:Smoothing matrix creation, step 10
INFO:surfer:colormap: fmin=5.00e+00 fmid=1.00e+01 fmax=1.50e+01
transparent=1
INFO:surfer:colormap: fmin=5.00e+00 fmid=1.00e+01 fmax=1.50e+01
transparent=1

Out[19]: ((-7.0167092985348768e-15, 90.0, 430.92617797851562, array([
0.,  0.,  0.])), -90.0)

In [20]: brain_fsaverage.save_image('dspm_fsaverage.jpg')
from IPython.display import Image
Image(filename='dspm_fsaverage.jpg', width=600)

Out[20]:
```

Solving the inverse problem on raw data or epochs

```
In [21]: fname_label = data_path +
'/subjects/daniel/label/lh.BA1.label'
label = mne.read_label(fname_label)
```

Compute inverse solution during the first 15s:

```
In [22]: from mne.minimum_norm import apply_inverse_raw,
apply_inverse_epochs

start, stop = raw.time_as_index([0, 15])  # read the first 15s of data

stc = apply_inverse_raw(raw, inverse_operator, lambda2, method, label,
start, stop)
```

Plot the dSPM time courses in the label

```
In [23]: plt.plot(stc.times, stc.data.T)
plt.xlabel('time (s)')
plt.ylabel('dSPM value')

Out[23]: <matplotlib.text.Text at 0x140b66410>
```

And on epochs:

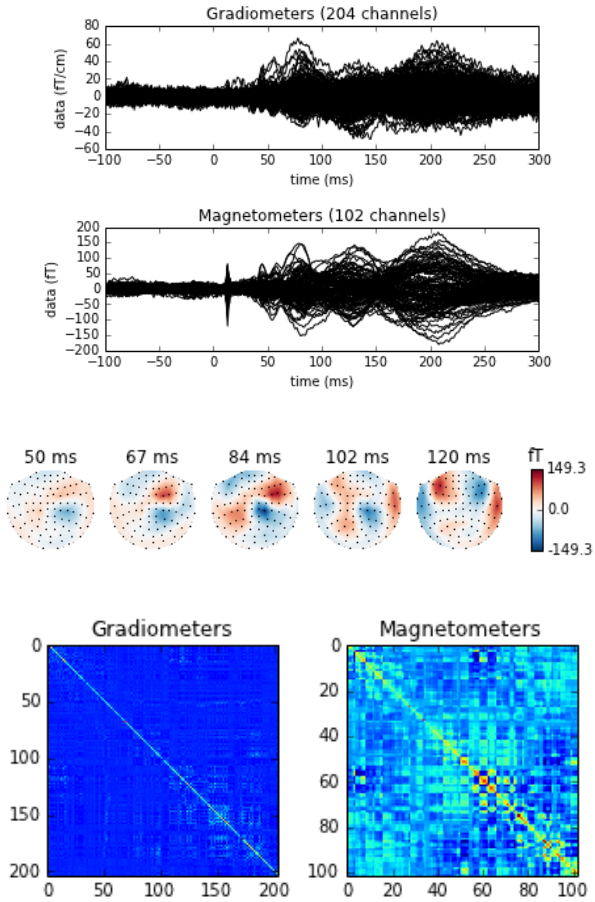
```
In [24]: from mne.minimum_norm import apply_inverse_epochs

# run it on 10 epochs only to avoid allocating too much memory
stcs = apply_inverse_epochs(epochs[:10], inverse_operator, lambda2,
method, label)
print "Number of stcs : %d" % len(stcs)
print stcs[:3]
Number of stcs : 10
```

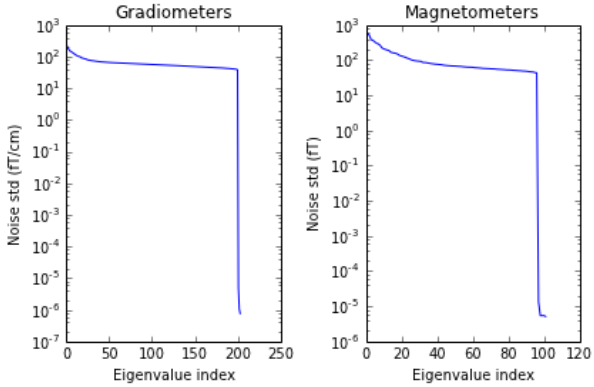
```
[<SourceEstimate  |  104 vertices, subject : daniel, tmin : -100.0
(ms), tmax : 300.0 (ms), tstep : 1.0 (ms), data size : 104 x 401>,
<SourceEstimate  |  104 vertices, subject : daniel, tmin : -100.0 (ms),
tmax : 300.0 (ms), tstep : 1.0 (ms), data size : 104 x 401>,
<SourceEstimate  |  104 vertices, subject : daniel, tmin : -100.0 (ms),
tmax : 300.0 (ms), tstep : 1.0 (ms), data size : 104 x 401>]
```

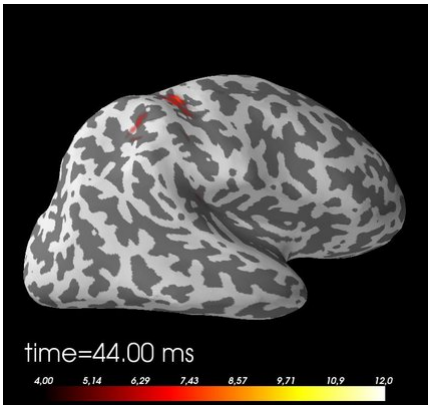
Exercises

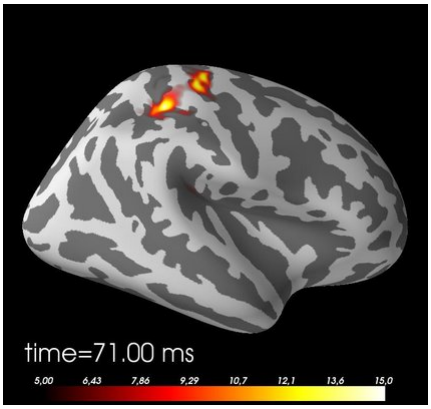
- Can you see the secondary somatosensory cortex (S2) if you look at 120ms?
- Run sLORETA on the same data and compare source localizations
- Run an LCMV beamformer on the same data and compare source localizations. Have a look at http://martinos.org/mne/stable/auto_examples/inverse/plot_lcmv_beamformer.html

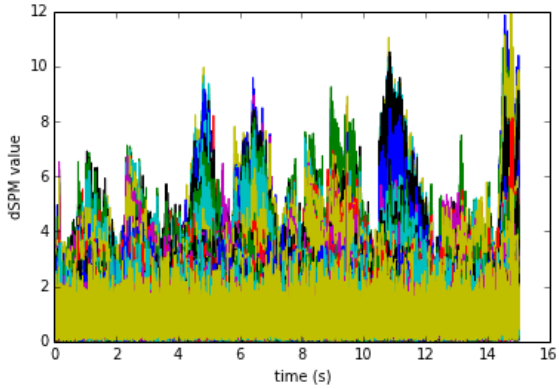
In [24]:




Gradiometers
Magnetometers
Noise std (fT/cm)
Noise std (fT)
Eigenvalue index
Eigenvalue index


time=44.00 ms
4,00 5,14 6,29 7,43 8,57 9,71 10,9 12,0


time=71.00 ms
5,00 6,43 7,86 9,29 10,7 12,1 13,6 15,0


dSPM value
time (s)