機械学習に基づく地震信号分類器と観測点アソシエーション手法

寒河江 皓大[1], 矢部 優[1], 内出 崇彦[1]

[1]地質調査総合センター 活断層・火山研究部門

1. はじめに

本資料では, 地震・テクトニック微動 (以下, 微動)・ノイズの信号分類を行うために設計した機械学習モデル **DiET** の重みパラメータと, それに関する Python コードを提供する. また, 複数観測点の検出結果を取りまとめる方法 **GrASP** に関する Python コードも提供する. 手法の詳細については, Sagae et al. (2025, preprint) を参考にしていただきたい.

2. 環境構築

本プログラムは Python で書かれており, 以下のパッケージを必要とする.

- `python (3.11+)`
- `tensorflow (2.15.0)`
- `keras (2.15.0)`
- `numpy`
- `scipy`
- `scikit-learn`
- `pandas`
- `pyproj`
- `matplotlib`

付属の zip ファイル (`GSJ_open.zip`) を解凍し,`GSJ_open` ディレクトリを作成する. 環境構築には `Conda` を用いることを推奨している. `GSJ_open` ディレクトリ下にある `environment.yml` ファイルを用いて, `Shell` で以下のコマンドを実行し, 本プログラム用の仮想環境を構築することができる.

```
# Please move to the `GSJ_open` directory
conda env create -n [environment name] -f environment.yml
conda activate [environment name]
```

3. 使用方法

DiET と **GrASP** に関する Python テストコードを `GSJ_open/Examples` ディレクトリに付属する. Python テストコードで使用する関数は, `Methods` ディレクトリ下の `DiET` と `GrASP` パッケージにそれぞれ格納されている. 以下では, それぞれの Python テストコードの使用例を説明する.

3.1. Discriminator for Earthquake and Tremor (DiET)

DiET は, 3成分スペクトログラムを入力として, 地震・微動・ノイズの判別確率を出力する機械学習モデルである. そこでまず, スペクトログラムを作成する Python コードの例を以下に示す.

`Show_spectrogram.py`

```
# Please move to the `Examples/DiET` directory
import sys
sys.path.append("../../Methods")
from DiET import Make_Spec

# Make spectrogram
Spec_scale, t_sample, f_sample = Make_Spec(Waveform_data)
```

使用者は, 機器特性を取り除いた 1 分間の波形データ (`Waveform_data`) を `Make_Spec()` 関数に入力して, スペクトログラム (`Spec_scale`) を計算できる. 例として, `Examples/DiET` ディレクトリ下の `Show_spectrogram.py` を実行することで, ランダムに生成されたデータのスペクトログラムを計算し, 表示することが可能である. `Methods/DiET` ディレクトリの `Params_DiET.py` には, スペクトログラム作成のパラメータが記載されてい

る．ただし，波形データのサンプリング周波数 (`sampling_freq`) 以外のパラメータは変更しないこと．

引き続き，**DiET** モデルの読み込みと適用方法について，Python コードの例を示す．

`Model_apply.py`

```
import numpy as np
import sys
sys.path.append("../../Methods")
from DiET import Params_DiET, get_model

# Load model
model = get_model()

Station_number = 2
# Spectrogram with all pixel values of 1 is predicted as noise.
Input_data=np.ones((Station_number, Params_DiET.frequency_size,
Params_DiET.time_size, Params_DiET.num_channels))
# Remain only the spectrogram power in the 2-8 Hz.
Input_data[0,25:73,:,:] = 0.2

# Model prediction
pred_result = model.predict(Input_data, batch_size =
Station_number, verbose = 0)
```

DiET モデルは，`get_model()` 関数で読み込むことができる．モデルを使って推論を行う際は，以下のデータを入力する．

- `Input_data`: 4階のテンソルであり，各次元は，観測点数，周波数 (73)，時間 (71)，速度成分 (3) に対応する．速度成分は VX, VY, VZ の順で入力する．特に VZ には，上下動成分を入力することを推奨する．
- `batch_size`: 観測点数を入力．

推論の結果, 出力 (`pred_result`) の1, 2, 3列目には, **地震**, **微動**, **ノイズ**の順で判別確率が出力される. 本プログラムでは, `Examples/DiET` ディレクトリ下の `Model_apply.py` を実行することで, 擬似データに対する判別確率を出力することができる.

3.2. Graph-based Associator with Signal Probability (GrASP)

GrASP は, 観測点ごとの判別確率をもとに信号を検出した観測点群を抽出する方法である. **GrASP** の準備として, 使用者は観測点の位置情報ファイルを以下のフォーマットで作成する必要がある.

- 1列目: 観測点名
- 2列目: 緯度 [deg]
- 3列目: 経度 [deg]

具体的には, `Data` ディレクトリの `Station_info.csv` を参考にしてもらいたい. また, **DiET** モデルが出力した判別確率を以下のフォーマットで保存しておく必要がある.

- 1列目: 年
- 2列目: 月
- 3列目: 日
- 4列目: 時
- 5列目: 分
- 6列目以降: 観測点ごとの判別確率 (`Station_info.csv` の行順と同じ)

具体的には, `Data/Probability` ディレクトリにある `Probability_T.csv` と `Probability_EQ.csv` を参考にしてもらいたい.

微動について, **GrASP** で観測点を取りまとめる Python コードの例を以下に示す.

`GrASP_Tremor.py`

```
# Please move to the `Examples/GrASP` directory
import numpy as np
import pandas as pd
```

```
 import pyproj
 import sys
 sys.path.append("../../Methods")
 from GrASP import Params_GrASP, kNN_graph, Make_Stadis_array,
Apply_Tremor

 # Load Station location
 Station_loc = pd.read_csv('../../Data/Station_info.csv')
 Station_loc_data = Station_loc.values

 # Make adjacency matrix (permanent adjacency)
 Stadis_array_original = Make_Stadis_array(Station_loc_data)
 Stadis_array = kNN_graph(Stadis_array_original, k_neareast =
 Params_GrASP.k_neareast)

 # Converted to XY coordinates
 transformer=pyproj.Transformer.from_crs(Params_GrASP.EPSG_base,
 Params_GrASP.EPSG_proj, always_xy=True)
 Station_locx,Station_locy=transformer.transform(Station_loc_dat
 a[:, 2], Station_loc_data[:, 1])
 Station_loc_data_XY = np.vstack((Station_loc_data[:, 0],
 Station_locx, Station_locy))
 Station_loc_data_XY = Station_loc_data_XY.T

 # Load Tremor probability
 Read_Detection=pd.read_csv('../../Data/Probability/Probability_
T.csv')
 Data = Read_Detection.values

 # GrASP for Tremor
 Save_Candidate = Apply_Tremor(Data, Stadis_array,
Station_loc_data_XY)
```

`Apply_Tremor()` 関数で，微動に関する **GrASP** を実行することができる．入力するデータは以下のとおりである．

- `Data`: 微動の判別確率 (`Probability_T.csv`) を読み込んで入力.
- `Stadis_array`: 距離行列を `kNN_graph()` 関数に入力して得られる隣接行列. 距離行列は, `Station_info.csv` を `Make_Stadis_array()` 関数に入力して計算する.
- `Station_loc_data_XY`: 平面直角座標系における観測点の位置.

特に, 距離行列 (`Stadis_array_original`) は何度も使用するため, 別ファイルに保存しておくことを推奨する (`Data/Stadis_array_original.csv`). 出力の `Save_Candidate` は, `Probability_T.csv` と同じフォーマットをもち, 抽出された観測点群が列挙される. 本プログラムでは, `Examples/GrASP` ディレクトリ下の `GrASP_Tremor.py` で上記の例を実行することができる.

次に, **地震**について, **GrASP** で観測点を取りまとめる Python コードの例を以下に示す.

`GrASP_EQ.py`

```
import sys
sys.path.append("../../Methods")
from GrASP import Params_GrASP, kNN_graph, Apply_Nocut

# Loading EQ probability
Read_Detection=pd.read_csv('../../Data/Probability/Probability_EQ.csv')
Data = Read_Detection.values

# GrASP for EQ
Save_Candidate = Apply_Nocut(Data, Stadis_array)
```

`Apply_Nocut()` 関数で, グラフ分割なしの **GrASP** を実行することができる. 入力するデータは以下のとおりである.

- `Data`: 地震の判別確率 (`Probability_EQ.csv`) を読み込んで入力.
- `Stadis_array`: `Apply_Tremor()` 関数で入力したものと同じ.

出力の `Save_Candidate` は,`Probability_EQ.csv` と同じフォーマットをもち, 抽出された観測点群が列挙される. また, `Apply_Nocut()` 関数は, 微動の判別確率を入力として受け取ることも想定している. これは, たとえば **GrASP** を別の観測網に適用する場合に有効である. このような場合, まずは一定期間の微動の判別確率を検証データとして `Apply_Nocut()` 関数に入力し, 抽出された観測点群の特徴 (例えば, Graph std: 観測点の空間的な広がり, Sagae et al. (2025, preprint) を参照) を調べることを推奨する. その特徴をもとに後述のパラメータを設定し, 以降は `Apply_Tremor()` 関数を使用して連続解析が可能である. 本プログラムでは,`Examples/GrASP` ディレクトリ下の `GrASP_EQ.py` で上記の例を実行することができる.

最後に, **GrASP** で抽出した微動の観測点群にカテゴリを付与する Python コードの例を示す.

`GrASP_Category.py`

```
import sys
sys.path.append("../../Methods")
from GrASP import Params_GrASP, kNN_graph, Apply_Categorization

# Load Tremor and EQ association results
Tremor_Association=pd.read_csv('../../Results/Tremor_candidate.
csv')
Data_T = Tremor_Association.values
EQ_Association = pd.read_csv('../../Results/EQ_candidate.csv')
Data_EQ = EQ_Association.values

# GrASP for Tremor Categorization
Save_result = Apply_Categorization(Stadis_array, Data_T,
Data_EQ)
```

`Apply_Categorization()` 関数を実行することで, 微動にカテゴリを付与することができる. 入力するデータは以下のとおりである.

- `Stadis_array`: `Apply_Tremor()` 関数で入力したものと同じ.
- `Data_T`, `Data_EQ`: 微動と地震それぞれに関する観測点の取りまとめ結果.

出力の `Save_result` には, `Data_T` の末列に 1 (probable tremor), または 2 (possible tremor) のカテゴリを付与したものが出力される. 本プログラムでは, `Examples/GrASP` ディレクトリ下の `GrASP_Category.py` で上記の例を実行することができる.

GrASP のパラメータは, `Methods/GrASP` ディレクトリの `Params_GrASP.py` に記載されている. **GrASP** の適用先を日本海溝から変更する場合には, 以下のパラメータを調整する必要がある.

- `EPSG_proj`: 適用地域が含まれる UTM 座標系の EPSG コードに変更 (EPSG コードは, https://spatialreference.org/ などを参照).
- `k_nearest`: 観測点数に合わせて調整 (はじめは $k \approx \sqrt{N}$ (N: 観測点数) に設定することを推奨).
- `Std_thres`: 遠地地震を除去するためのパラメータ. `Apply_Nocut()` 関数を用いて, 検証データから抽出された観測点群の特徴をもとに調整 (Sagae et al. (2025, preprint) に倣い, 観測点の空間的な広がりを示す Graph_std の 3σ を設定することを推奨).
- `Max_eigen`: 第二最小固有値の累積寄与率に関する閾値. グラフ分割が多発する場合には調整が必要.

4. 免責

- 本プログラムは Ubuntu 22.04 で実行した. その他の環境で正しく動作するかは保証できない.
- 本プログラムの使用により生じるいかなる損害も, 国立研究開発法人産業技術総合研究所は一切の責任を負わない.

5. 開発グループ

主要メンバー

- 開発者: 寒河江 皓大 (k.sagae@aist.go.jp)
- コードレビュー, 草稿チェック: 矢部 優 (s.yabe@aist.go.jp)
- プロジェクトリーダー: 内出 崇彦 (t.uchide@aist.go.jp)

協力者

- クイックレビュー: 加納 将行

6. 使用にあたって

- 本プログラムを使用する際は, 以下を引用すること.

寒河江皓大・矢部優・内出崇彦 (2025) 機械学習に基づく地震信号分類器と観測点アソシエーション手法, 地質調査総合センター研究資料集, no 767, 産業技術総合研究所地質調査総合センター.

- 不具合の報告等は開発グループにご連絡ください.

7. 謝辞

DiET モデルを作成するに当たっては, 国立研究開発法人 防災科学技術研究所が運営する日本海溝海底地震津波観測網 S-net の速度波形データを使用しました. また, 気象庁一元化処理地震カタログを使用しました. 本研究は JSPS 科研費 学術変革領域研究 (A)「Slow-to-Fast 地震学」課題番号: JP21H05205, JP21H05203) の助成を受けたものです.

8. ライセンス

9. 参考文献

Kodai Sagae, Masayuki Kano, Suguru Yabe, Takahiko Uchide, 2025 (preprint), Machine Learning-Based Detection and Localization of Tectonic Tremors in the Japan Trench. *ESS Open Archive* . February 19, 2025.
DOI: 10.22541/essoar.174000875.59692909/v1