Deep learning based time window selection algorithm for adjoint tomography

Full waveform inversion is an efficient tool to retrieve high resolution subsurface properties by matching observed seismic data with synthetic iteratively. A sophisticated preprocessing workflow must be applied to observed and synthetic data to avoid cycle skipping problem. Specifically, the mismatched and noise-dominated data cannot be involved in the inversion. We design a fully convolutional network to automatically capture complex phases from the seismograms and reject cycle skipped and/or noise-dominated events, thereby mitigating the nonlinearity of the inversion.

The program is divided into 3 parts:

1. Data preprocessing module: preprocess.py

Input: a single component (Z component) of the seismogram.

The data preprocessing includes: normalization, bandpass filter, and resampling.

1. Normalization: normalization of the maximum value of the absolute value of amplitude.
2. Bandpass filter: three frequency bands: 1~6Hz、4~10Hz、8~15Hz.
3. Resampling: uniformly resample the data set to 4992 sampling points.

2. Training module: fcn\_test.py

The neural network consists of 16 convolutional layers, which includes eight down-sampling stages and eight up-sampling stages. The down-sampling layer extracts and shrinks useful features from the input data, and the up-sampling layer expands the condensed layer and converts it to the probability distribution. 1D convolutions and rectified linear unit (ReLU) activation function are utilized in each layer, and the final convolution layer output the predicted result with the sigmoid function. The kernel size of all convolutional layers is 3. The number of channels determines feature size of each convolutional layer, which increases from 1 to 512 and then decreased from 512 to 1. To regularize the problem and avoid overfitting, we insert dropout layers with a dropout rate of 0.5 to each convolutional layer.

3. Prediction module: predict\_parameters.py

Output: 1D array represents the probability distribution of the complex phases.

In this module, users can predict a single seismogram or multiple seismic records according to the training result ‘unet.hdf5’ obtained by the training module.

Example:

Build environment: anaconda 3 and python 3.7.

1. Train: set up number of trainnum and testnum (3:1); set up batch size (adjust according to your own GPU performance); set up epoch number of the network. Then run the FCN code (python3 fcn\_test.py). Run CUDA\_VISIBLE\_DEVICES=1 python fcn\_test.py to use GPU to accelerate the program.
2. Prediction: set trainnum and testnum (trainnum=0 in this case). User can choose the output format of prediction result ( “SAC” or “MAT”), meanwhile, you need modify your output path, then run your file “python3 predict\_parameters.py” Run CUDA\_VISIBLE\_DEVICES=1 python predict\_parameters.py to use GPU to accelerate the program.

ENVIRONMENT: We perform the tests based on Keras 2.3.1 and CUDA 9.6.

Check your CUDA version：

cat /usr/local/cuda/version.txt

Check your cuDNN version:

cat /usr/local/cuda/include/cudnn.h | grep CUDNN\_MAJOR -A 2

View the corresponding tensorflow version:

Install tensorflow:

pip3 install tensorflow-gpu==2.0.0 or conda install tensorflow-gpu==2.0.0

Install keras:

pip3 install keras-gpu==2.3.1 or conda install keras-gpu==2.3.1

Check if the version you installed is successful：

(base) [users@root ~]$ python

Python 3.7.1 | packaged by conda-forge | (default, Nov 13 2018, 18:33:04)

[GCC 7.3.0] :: Anaconda, Inc. on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import keras

Using TensorFlow backend.

>>>import tensorflow

>>>