BCIComp_IVa : LinkPred Documentation

Paper

- Systematic Enchancement of Functional Connectivity In Brain-Computer Interfacing Using Common Spatial Patterns and Tangent Space Mapping

Dataset

- IVa
    - The epochs are pre-processed at [8, 30] Hz using a 4th order butterworth filter and segmented from the trigger onset to 3.5 seconds. The constants file contains channel related information and sampling rate. To load the numpy files, use the following command: np.load(filename, allow_pickle=True)
    - To retrieve individual information, use:
        - Example: get the epochs from the numpy file for subject aa: test = np.load(path + '/preprocessed_epochs_aa.npy', allow_pickle=True) epochs = test.item().get('epochs')

Working Log

- https://shimowendang.com/docs/38d8PTDxhTKcrxXW/ 「IVa-Comp: LinkPred Working Log」，可复制链接后用石墨文档 App 或小程序打开 (Before Oct.)
- 【腾讯文档】IVaComp Working Log https://docs.qq.com/doc/DU1RmTXV3b1ZCb01V

Ref:

- GCN-GAN: https://arxiv.org/abs/1901.09165

Explanation: BCIComp_IVa-LinkPred Repo

IVaComp

Data

We start with the assumption that the dynamical pattern changes of EEG signal as a time series data could differentiate the tasks, meanwhile be resistant to subjects variance to be consistent across subjects. So we started by the problem of link prediction.

There are several works on this IVa Benchmark. mainly concluded in this paper: https://arxiv.org/pdf/2103.10977.pdf (Dokur et al., 2021)

- adj_dict: fully connected adjacency matrices.
- adj_to_csv: adj matrices reorganized as saved as csv files
- spaese_adj_dict: sparsened adjacency matrices
- epcohs: raw data.

- adj_gen.py : For each trial, based on non-overlapping window size of 35 (timesteps / trial: 350), segment the epoch/subject into 10 snapshots. Calculating the PLV value for each pair of nodes (# of nodes: 118), resulted in a fully-connected diagonal adjacency matrix, with each value in [0, 1]
- adj_to_csv.py: Adjacent matrices constructed (adj_gen.py) reorganized as DataFrames (columns: src(source node id), dst(destination node id), weight(plv value of src and dst node), rows: number of edges)
- sparse_adj.py: Waiting

- **LinkPred**

  We want to capture the link dynamic changes across time. By referring the paper(Lei et al., 2019) which raised up a non-linear temporal link prediction model: **GCN-GAN** for weighted dynamic networks, tailor the model for IVa Dataset. NOTE in the paper (Page 6):

  ”””

  ... **The sparsity of link weights**: Average portion of zeros elements in alll the adjacency matrices for UCSB, KASIT, BJ-Taxi and NumFabric are 0.52, 0.92, 0.94, 9.50, respectively. ...

  ”””

  ```
  # Defined mismatch rate
  def MissRate(input, target):
      num = 1
      for s in input.size():
          num = num * s
      mask1 = (input > 0) & (target == 0)
      mask2 = (input == 0) & (target > 0)
      mask = mask1 | mask2
      return mask.sum().item() / num
  ```

  Therefore, sparsened rather than fully connected matrix according to above.

  GCN-GAN model: two parts: generator and discriminator.

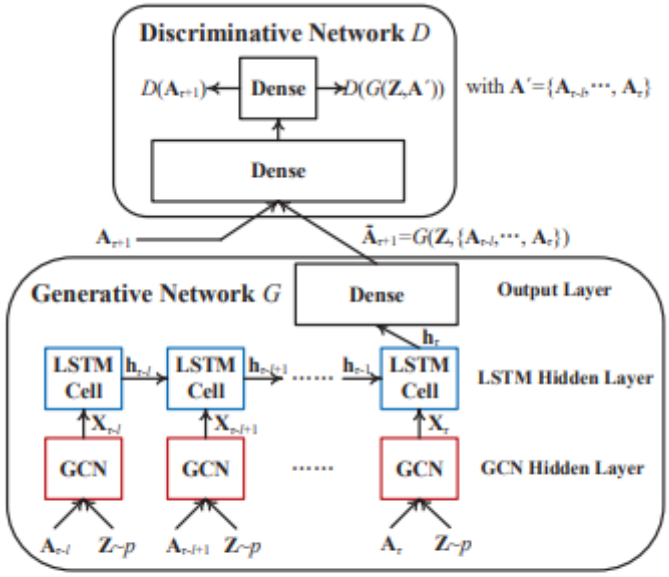


Fig. 1. The architecture of the GCN-GAN temporal link prediction model with a generative network G (bottom side) and a discriminative network D (top side). The generative network consists of a GCN hidden layer, an LSTM hidden layer and a full-connected output layer, while the discriminative network takes the form of full-connected feedforward network.

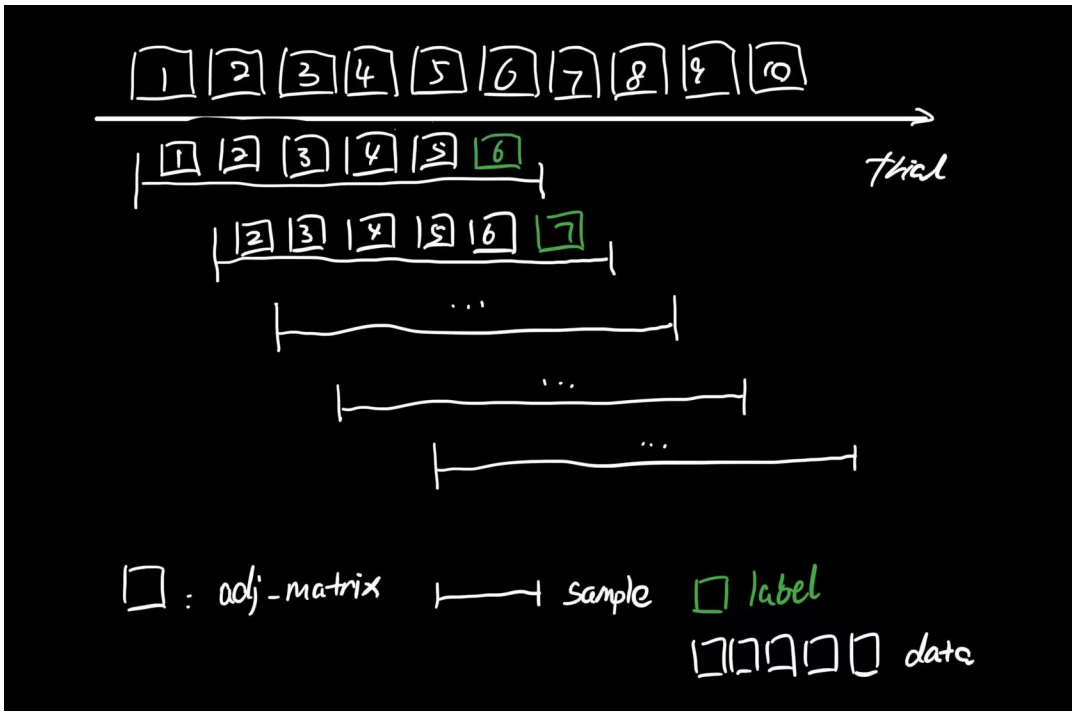
- dataset
- result: learned model by generator saved as pkl and pth / subject. test result saved.
- model.py
- peprocess.py

  For each trial / subject, we have:

  fully connected adjacency matrix: (118, 118)

  adjacency matrix id: 1 2 3 4 5 6 7 8 9 10

  Reorganize them as samples, 5 samples / trial, 5 * 280 = 1400 samples.

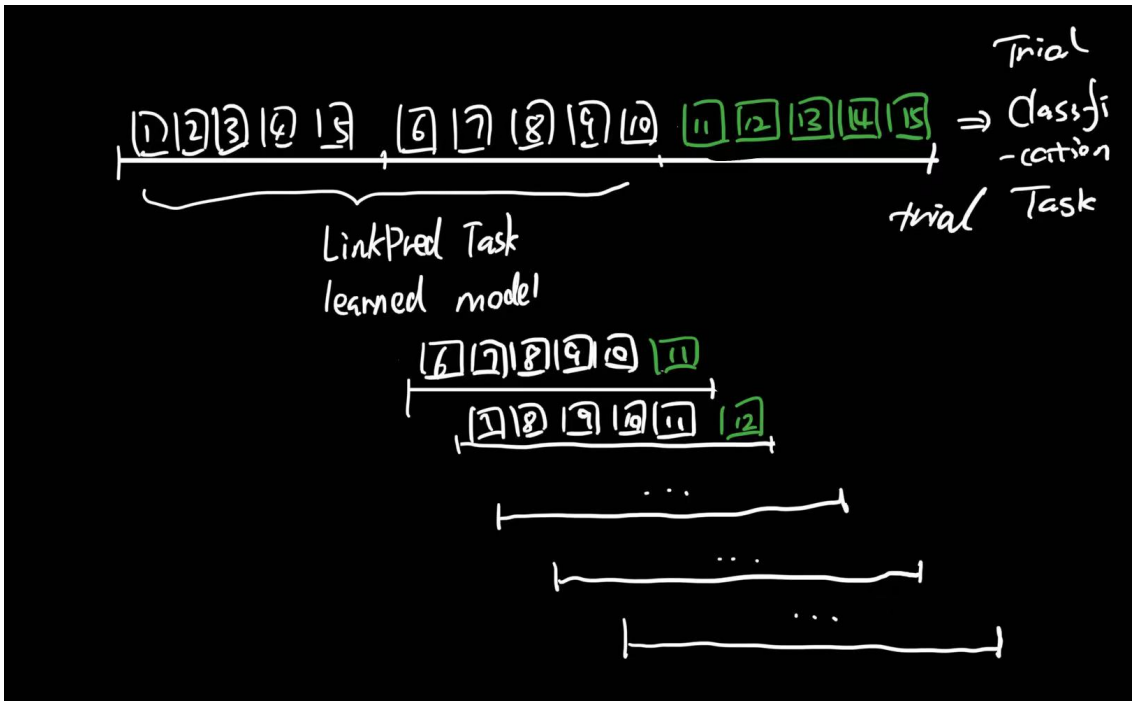


- train.py

- test
- utils.py: LPDataset classes, MSE, and EdgeWiseKL and MissRate function built for training utility.
- config.yml

- **TrialPred**

  We plan to generate 5 more graphs(sparsened adjacency matrix) from the **LinkPred** task for each trail, using the learned model parameters and graph 6-10 in each trial, then concatenate them to 10 graphs get from the original trial. So we did a data augmentation in this step and could use 15 graphs as a sample / trail. That's how we think the two tasks can be connected.



  - dataset
  - gcn_lstm_model.py
  - preprcess.py:

    organizing 10 graphs/ trial as a sample with label 0(right_hand) or 1(right_foot), have 280 samples, 200 for training and 80 for test. (*organizing 15 graphs: Waiting.)
  - train_and_test.py
  - utils.py

Lei

replication of Lei's work in Pytorch, found here https://github.com/jiangqn/GCN-GAN-pytorch and modified little problems