JOINT COVARIATE-ALIGNMENT AND CONCEPT-ALIGNMENT: A FRAMEWORK FOR OUT-OF-DISTRIBUTION GENERALIZATION

Anonymous

Anonymous

ABSTRACT

Due to the limited space and the double-blind review process, this report provides the details of implementation and how to reproduce the numerical results of paper "*Joint covariate-alignment and concept-alignment: a framework for out-of-distribution generalization*", submission 18 at IEEE International Workshop on Machine Learning for Signal Processing (MLSP) 2022. The final version of this code will be uploaded on our personal GitHub website after the end of reviewing process.

1. IMPLEMENTATION DETAILS

1.1. CS-CMNIST

For the CS-CMNIST dataset, we follow the learning model and tuning procedures proposed in [1].

Particularly, the learning model contains three convolutional layers with the corresponding feature dimensions of 256, 128, and 64. Each convolutional layer is followed by a ReLU activation and batch normalization layer. The last layer is a linear layer that aims to classify the digit back into 10 classes. We use a stochastic gradient descent optimizer for training, the batch size is set to 128, the learning rate is 10^{-1} and decays after 600 steps while the total number of steps is set to $2,000$. The hyperparameters α, β are selected from $(10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4})$. Noting that three algorithms CEM, MMD-CEM, and CORAL-CEM are based on the CEM algorithm [2], thus, they have one more additional hyper-parameter that controls the conditional entropy term. For consistency purposes, this hyper-parameter also is selected from $(10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4})$.

We use the training-domain validation set procedure [3] for model selection i.e., selecting the model with the highest validation accuracy on the validation set sampled from seen domains. We repeat our entire experiment five times and only report the average accuracy together with its standard deviation.

1.2. CMNIST

For the CMNIST dataset, we follow the learning model and tuning procedures proposed in [3].

Particularly, following [3], we use MNIST-ConvNet as the Neural network architecture for learning from the CMNIST dataset. The details of MNIST-ConvNet can be found in Fig. 1 (Table 7 [3]). The default learning rate is 0.001, the default batch size is 64, the hyperparameters α, β are randomly selected from $[10^{-1}, 10^{5}]$. Noting that three algorithms CEM, MMD-CEM, and CORAL-CEM are based on the CEM algorithm [2], thus, they have one more additional hyper-parameter that controls the conditional entropy term. For consistency purposes, this hyper-parameter is randomly selected from $[10^{-1}, 10^{5}]$. Finally, for each algorithm, 10 pairs of hyper-parameters are randomly selected in the range of $[10^{-1}, 10^{5}]$ for testing. Noting that in [3], the authors use 20 pairs of of hyper-parameters to select the best model, however, the code runs slowly and we decide to reduce to 10 pairs.

Table 7: Details of our MNIST ConvNet architecture. All convolutions use 3×3 kernels and "same" padding.

#	Layer
1	Conv2D (in=d, out=64)
2	ReLU
3	GroupNorm (groups=8)
4	Conv2D (in=64, out=128, stride=2)
5	ReLU
6	GroupNorm (8 groups)
7	Conv2D (in=128, out=128)
8	ReLU
9	GroupNorm (8 groups)
10	Conv2D (in=128, out=128)
11	ReLU
12	GroupNorm (8 groups)
13	Global average-pooling

Fig. 1. Architecture of MNIST-ConvNet for CMNIST dataset.

Even though there are three model selection methods proposed in [3], and the results of all methods are recorded in our implementation, we decide to report the results from the training-domain validation set model selection method in our paper. This is not only because the training-domain validation is the most common selection method but also because of the consistency of the report in CS-CMNIST dataset. Since run-

978-1-6654-8547-0/22//$31.00 ©2022 IEEE

ning the experiment for the CMNIST dataset requires a lot of time and computer resources, we only repeat our entire experiment two times and report the average accuracy together with its standard deviation. Noting that in [3], the entire experiment is repeated five times for the final accuracy.

2. REPRODUCING THE RESULTS

2.1. CS-CMNIST dataset

Our implementation is based on the source code at `https://github.com/ahujak/IB-IRM` [1].

First, you need to install some packages for satisfying all the requirements below:

1. torch 1.6.0
2. torchvision 0.7.0
3. numpy 1.19.1
4. tqdm 4.41.1

Second, please take a look at the tested algorithms in the file "*algorithm.py*". The algorithms are denoted from 1, 2, ..., up to 8 in order of ERM, IRM, IB-ERM, IB-IRM, CORAL-CEM, MMD-CEM, MMD-IRM, and CEM. In "*algorithm.py*", we have different names for ERM, IRM, IB-ERM algorithms so these algorithms can run together without conflict. However, to keep the same number of hyper-parameters and utilize the existing source code, five algorithms IB-IRM, CORAL-CEM, MMD-CEM, MMD-IRM, and CEM are under the same name of "IBIRM" because they are very similar, just having a bit of change at the loss function. Therefore, please select only one algorithm from five algorithms IB-IRM, CORAL-CEM, MMD-CEM, MMD-IRM, and CEM to run at a time, and please comment out other ones. In the default code, we are using the MMD-CEM algorithm and commented out IB-IRM, CORAL-CEM, MMD-IRM, and CEM.

Third, please run "*sweep_train.py*" to get the reported result. "*sweep_train.py*" scans over the algorithms and the hyper-parameters. The setting of hyper-parameters can be found at the end of "*sweep_train.py*". Now, we just commented out the sweeping functions for ERM, IRM, and IB-ERM while keeping the same sweeping function for IB-IRM, CORAL-CEM, MMD-CEM, MMD-IRM, and CEM. Please select the algorithm you want to sweep and comment out other ones.

In addition, there are some parameters that can be selected in "*sweep_train.py*". For example, "–test_val" to control the tuning procedures, "–env_seed" to select the model, "–holdout_fraction" to control the validation set. For now, we use the default values of 5% data for validation and use the train-validation tuning procedure. You may not need to adjust these parameters.

The checkpoints are not stored, and the final results will be printed after the whole run. Our code is based on `https://github.com/ahujak/IB-IRM` [1]. Due to the double-blind review process, we promise to upload our source code to GitHub after the end of the review process.

2.2. CMNIST dataset

The code for CMNIST dataset is based on `https://github.com/facebookresearch/DomainBed` [3].

First, you need to install some package for satisfying the below requirements (see "*requirements.txt*"):

1. numpy==1.20.3
2. wilds==1.2.2
3. imageio==2.9.0
4. gdown==3.13.0
5. torchvision==0.8.2
6. torch==1.7.1
7. tqdm==4.62.2
8. backpack==0.1
9. parameterized==0.8.1
10. Pillow==8.3.2

Second, you may not need to download the data since we already uploaded the MNIST dataset into this folder.

Third, launch a sweep via:

"python -m sweep –command launch
–data_dir=./domainbed/data/path
–output_dir=./domainbed/outputThuan/path
–command_launcher local
–algorithms IB_IRM
–datasets ColoredMNIST
–n_hparams 10
–n_trials 2"

where "n_trials" is the number of trials i.e, the number of random seeds, for each trial, the algorithm will scan over all the hyper-parameters and result in the accuracy for that trial. The final accuracy is averaged over all trials. "–n_hparams" is the number of hyper-parameter pairs which is randomly picked in a range of $(0.1, 10000)$. "–datasets" is the selected dataset, default by "ColoredMNIST", "–algorithms" is the selected algorithm, please select the algorithm you want to test.

After all jobs have either succeeded or failed, you can delete the data from failed jobs with:

"python -m sweep –command delete_incomplete

–data_dir=./domainbed/data/path
–output_dir=./domainbed/outputThuan/path
–command_launcher local
–algorithms IB_IRM
–datasets ColoredMNIST
–n_hparams 10
–n_trials 2”

and then re-launch them by running “python -m sweep –command launch” again.

To view the results of your sweep:

“python -m domainbed.scripts.collect_results
–input_dir=/my/sweep/output/path”

Note 1: Please take a look at the tested algorithms in “*algorithms.py*”. The algorithms are denoted from 1, 2, ..., up to 8 in order of ERM, IRM, IB-ERM, IB-IRM, CEM, MMD-IRM, MMD-CEM, and CORAL-CEM. We have different names for ERM, IRM, IB-ERM algorithms so these algorithms can run together without conflict. However, to keep the same number of hyper-parameters and utilize the existing source code, all algorithms IB-IRM, CEM, MMD-IRM, MMD-CEM, and CORAL-CEM are under the same name of ”IB_IRM” - because they are very similar, just having a bit of change at the loss function. Please select only one algorithm from five algorithms IB-IRM, CEM, MMD-IRM, MMD-CEM, and CORAL-CEM to run at a time and comment out other ones. In the default code, we are using the CEM algorithm and commented out IB-IRM, MMD-IRM, MMD-CEM, and CORAL-CEM.

Note 2: Please see “*README.md*” for the plain text version of this report, it may be easier to copy the comments from the plain text than the comment in this pdf file. We also encourage the reader to refer to `https://github.com/facebookresearch/DomainBed` [3] for more details.

3. LIMITATIONS OF THIS IMPLEMENTATION

Due to the limited time, the code is not prepared very clearly. We will fix the existing problems, and upload a new version to our GitHub webpage after the end of reviewing process. The new version will not require any manual settings like this version i.e., all algorithms can be run automatically.

Thank you for your consideration!
Authors

4. REFERENCES

[1] K. Ahuja, E. Caballero, D. Zhang, Y. Bengio, I. Mitliagkas, and I. Rish, “Invariance principle meets information bottleneck for out-of-distribution generalization,” *arXiv preprint arXiv:2106.06607*, 2021.

[2] T. Nguyen, B. Lyu, P. Ishwar, M. Scheutz, and S. Aeron, “Conditional entropy minimization principle for learning domain invariant representation features,” *arXiv preprint arXiv:2201.10460*, 2022.

[3] I. Gulrajani and D. Lopez-Paz, “In search of lost domain generalization,” *arXiv preprint arXiv:2007.01434*, 2020.