MLDG

RP Group 33

1. Introduction

- Humans encounter new task conditions all the time
- Used to develop strategies which are robust to different task contexts
- We want artificial learning agents to do the same
- More versatility
- Better out the box performance

2. MLDG Algorithm

High-level pseudocode of the MLDG algorithm can be found in Figure 2. One can intuitively interpret that the parameters (Theta) are corrected by the loss function of the meta-test set (G). Therefore, the model will not overfit on a set of domains.

Explanation:

line 2 the algorithm starts off by defining the domains S.
line 3 the initial model parameters (Theta) and hyperparameters are set.
line 4 denotes the start of the iterations.
line 5 the training data are split in a meta-train set and a meta-test set (Figure 3)
line 6 the gradients for meta-train are calculated using the loss function (F).
line 7 the proposed updated parameters can be calculated for the meta train set.
line 8 the loss function (G) is calculated for the meta-test set as well.
line 9 the updated parameters (Theta) are calculated based on the meta-train set loss function (F) and the meta-test set loss function (G) times a constant gamma.

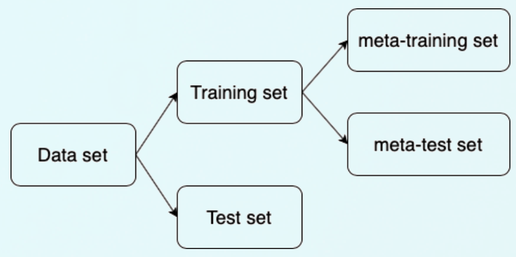
Algorithm 1 Meta-Learning Domain Generalization

1: **procedure** MLDG
2: **Input**: Domains $\mathcal{S}$
3: **Init**: Model parameters Θ. Hyperparameters α, β, γ.
4: **for** ite **in** iterations **do**
5: **Split**: $\mathcal{S}$ and $\check{\mathcal{S}} \leftarrow \mathcal{S}$
6: **Meta-train**: Gradients $\nabla_\Theta = \mathcal{F}'_\Theta(\mathcal{S}; \Theta)$
7: Updated parameters $\Theta' = \Theta - \alpha \nabla_\Theta$
8: **Meta-test**: Loss is $\mathcal{G}(\check{\mathcal{S}}; \Theta')$.
9: **Meta-optimization**: Update Θ

$$\Theta = \Theta - \gamma \frac{\partial(\mathcal{F}(\bar{\mathcal{S}}; \Theta) + \beta \mathcal{G}(\check{\mathcal{S}}; \Theta - \alpha \nabla_\Theta))}{\partial \Theta}$$

10: **end for**
11: **end procedure**

Figure 1

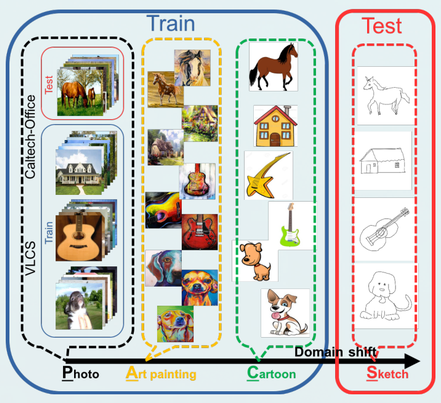


3. Experiment to Reproduce

- Recognize object in one domain, while only training in other domains
- PACS multi-domain recognition benchmark
- 9991 images across 7 categories spread over 4 domains (see figure 3)
- Results to be reproduced are shown in table 1

Categories
- Dog
- Elephant
- Giraffe
- Guitar
- House
- Horse
- Person

Domains
- Photo
- Art painting
- Cartoon
- Sketch



4. Reproducability Efforts

Code changes
- Rewrite existing code to Python 3
- Rewrite code to utilize PyTorch

Extra
- Investigate hyperparam robustness
- Investigate loss over time

Table 1: Cross-domain recognition accuracy (Multi-class accuracy) on the PACS dataset. Best performance in bold.

	D-MTAE (Ghifary et al. 2015)	Deep-all	DSN (Bousmalis et al. 2016)	AlexNet+TF (Li et al. 2017)	MLDG (CNN)
art_painting	60.27	64.91	61.13	62.86	**66.23**
cartoon	58.65	64.28	66.54	**66.97**	66.88
photo	**91.12**	86.67	83.25	89.50	88.00
sketch	47.86	53.08	58.58	57.51	**58.96**
Ave.	64.48	67.24	67.37	69.21	**70.01**

5. Intermediate Reproducability Results

- Single run with 3 iterations
- Original paper performed 5 runs with 3 iterations each
- Reproduced accuracy is expected to converge towards table 1 for more runs

MLP (Baseline)

```
ite: 0 loss: 5.83397 lr: 0.0005
ite: 5000 loss: 0.7646471 lr: 0.0005
ite: 10000 loss: 0.41197827 lr: 0.0005
ite: 15000 loss: 0.21348761 lr: 5e-06
ite: 20000 loss: 0.23729184 lr: 5e-05
ite: 25000 loss: 0.23310997 lr: 5e-05
ite: 30000 loss: 0.23491922 lr: 5.000000000000002e-07
ite: 35000 loss: 0.20772612 lr: 5.000000000000001e-06
ite: 40000 loss: 0.2116398 lr: 5.000000000000001e-06
ite: 45000 loss: 0.19000785 lr: 5.000000000000001e-08
```

Average accuracy: 69.5%

MLDG

```
ite: 0 meta_train_loss: 3.9915504 meta_val_loss: 7.7315373 lr: 0.0005
ite: 5000 meta_train_loss: 0.38368234 meta_val_loss: 0.7135035 lr: 0.0005
ite: 10000 meta_train_loss: 0.47828954 meta_val_loss: 0.22636156 lr: 0.0005
ite: 15000 meta_train_loss: 0.3709153 meta_val_loss: 0.018917829 lr: 5e-06
ite: 20000 meta_train_loss: 0.33163258 meta_val_loss: 0.014834516 lr: 5e-05
ite: 25000 meta_train_loss: 0.3221491 meta_val_loss: 0.039074518 lr: 5e-05
ite: 30000 meta_train_loss: 0.34363568 meta_val_loss: 0.03459864 lr: 5.000000000000002e-07
ite: 35000 meta_train_loss: 0.3272212 meta_val_loss: 0.0419287 lr: 5.000000000000001e-06
ite: 40000 meta_train_loss: 0.031871844 meta_val_loss: 0.29184628 lr: 5.000000000000001e-06
ite: 45000 meta_train_loss: 0.3590644 meta_val_loss: 0.09930536 lr: 5.000000000000001e-08
```

Average accuracy: 71.7%