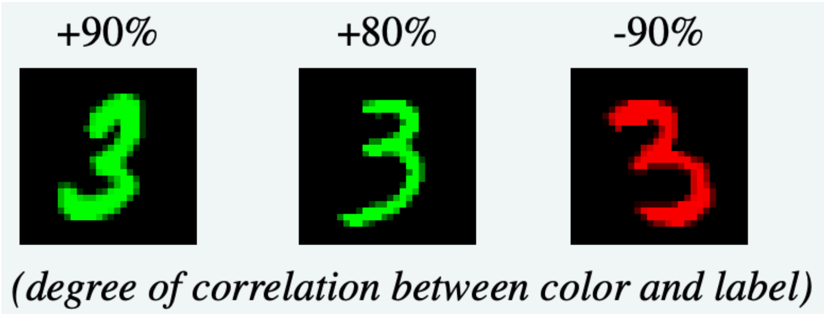
AISG Programming Assignment ColoredMNIST

Dataset Description



ColoredMNIST [1] is a variant of MNIST, the classical digit recognization dataset. *Concept shift* is artificially produced by coloring of the images that creates spurious correlation between the label and the color. The transformation process contains several steps:

1. Define three environments (two training, one test) from MNIST, each with 20,000 samples.
2. For each sample, assign a preliminary binary label $\tilde{y}$ to the image based on the digit. $\tilde{y} = 0$ for digits 0-4 and $\tilde{y} = 1$ for 5-9.
3. Obtain the final label y by flipping $\tilde{y}$ with probability $p_f = 0.25$.
4. Sample the color id z by flipping y with probability p_e , where $p_e = 0.1$ in the first training environment, $p_e = 0.2$ in the second, and $p_e = 0.9$ in the test environment. Color the image red if $z = 1$ or green if $z = 0$.

As a result, the color is positively correlated with the label in both training environments, though with different degrees of correlation. The correlation is negated in the test environment.

We adapt the dataset to a slightly easier version for this assignment by resetting the probabilities.

$$p_f = 0.2, \quad p_e = 0.1, 0.4 \text{ for training and } 0.9 \text{ for test.} \tag{1}$$

Algorithms

There are two algorithms to be implemented for this assignment: Group DRO and IRM, which accounts for 15'' and 5'', respectively. The total score is 20''.

Empirical Risk Minimization

We have already implemented *Empirical Risk Minimization (ERM)*, by combining the binary cross entropy loss of all training environments $e \in E$ with sample sizes of N_e. Labels and predictions are denoted by y and $\hat{y}$, respectively.

$$\begin{aligned} \mathcal{L}_{\text{ERM}} &= \frac{1}{|E|} \sum_e \mathcal{L}_e && (2) \\ &= \frac{1}{|E|} \sum_e \frac{1}{N_e} \sum_{i=1}^{N_e} \Big[-y_i^{(e)} \log \hat{y}_i^{(e)} - (1 - y_i^{(e)}) \log(1 - \hat{y}_i^{(e)}) \Big]. && (3) \end{aligned}$$

TODO: Group DRO (15'')

Group DRO [3] trains distributionally robust neural networks for group shifts, by reweighting groups to focus on risky subpopulations. The algorithm is detailed as follows.

Algorithm 1 Mini batch optimization algorithm for group DRO

Require: Step size η; Learning rate ϵ; Data distribution P; Group set G

Initialize $\theta^{(0)}$ and $q^{(0)}$

for $t = 1, \ldots, T$ **do**

  $\{(x_i, y_i, e_i)\}_{i=1}^{B} \sim P$ ▷ Sample a batch of data.

  $q' \leftarrow q^{(t-1)}$;

  for $g \in G$ **do**

    $q'_g \leftarrow q'_g \exp\left\{\eta \frac{\sum_i \mathbf{1}[e_i=g]\ell(\theta^{(t-1)}; x_i, y_i)}{\sum_i \mathbf{1}[e_i=g]}\right\}$ ▷ Update weights for group g

  $q^{(t)} \leftarrow q' / \sum_g q'_g$ ▷ Renormalize q

  $\theta^{(t)} \leftarrow \theta^{(t-1)} - \epsilon \sum_g q_g^{(t)} \nabla_\theta \frac{\sum_i \mathbf{1}[e_i=g]\ell(\theta^{(t-1)}; x_i, y_i)}{\sum_i \mathbf{1}[e_i=g]}$ ▷ Use q to update θ

We suggest $\eta = 0.05$ for this assignment.

Grading Criteria:

- (15'') Test accuracy $\geq 60\%$.
- (10'') Test accuracy $\geq 55\%$.

TODO: Invariant Risk Minimization (5'')

Invariant Risk Minimization (IRM) [1] learns a representation Φ of data, upon which the predictor $w \circ \Phi$ is simultaneously optimal across training environments. By taking a 1-dim representation $\Phi \in \mathbb{R}$ and **fixing** the predictor $w = 1.0$, the optimization of the IRM objective is tractable by regularization (IRMv1):

$$\mathcal{L}_{\text{IRM}} = \mathcal{L}_{\text{ERM}}(\Phi) + \lambda \cdot \frac{1}{|E|} \sum_e \left\| \nabla_{w|w=1.0} \mathcal{L}_e(w \circ \Phi) \right\|^2. \tag{4}$$

We suggest $\lambda = 50$ for this assignment.

Grading Criteria:

- (5'') Test accuracy $\geq 60\%$.

Assignment Instructions

Please complete the programming assignment based on the code framework. Two sections are to be implemented for Group DRO and IRM. Please search by `TODO`. The structure of the code framework is shown below.

```
└─hw_OOD
    ├─README.pdf
    │
```

```
    └src
        ├dataset.py             Dataloader.
        ├exp-mnist.py           Main file.
        ├LICENSE
        ├metric.py              Metrics (accuracy, AUC, macro F1).
        ├model.py               The MLP model and loss functions. (TODO)
        ├preprocessing.py       Download and generate the ColoredMNIST dataset.
        ├register.py
        ├requirements.txt
        ├run.sh                 The script to run all required experiments.
        └trainer.py             The training procedure.
```

Suggested steps complete the assignment:

1. Install `python` and the required packages.

```
pip install -r requirements.txt
```

2. Run the ERM baseline to check all packages are correctly configured.

```
python exp-mnist.py  --seed xxxx --trainer ERM > ERM_log.txt
```

The full log is saved at `ERM_log.txt`. The file should end with a summary of statistics. For example:

```
Summary: {'Accuracy_testmean': 0.5047, 'Accuracy_teststd': 0.055995907588561O3,
'AUC_testmean': 0.4348140903478652, 'AUC_teststd': 0.06026994797221927,
'F1_macro_testmean': 0.5034497162692105, 'F1_macro_teststd': 0.056703574819522484}
```

3. *Briefly* read the `exp-mnist.py` file to get a sense of the experimental design, including construction of the dataset, the model, the optimizer and the trainer. Three repetitive experiments are conducted and a summary of statistics is reported.
4. Read the `trainer.py` file to understand the training procedure, especially the computation of losses and regularization.
5. *Carefully* read the `model.py` file to understand the MLP model and its structure. Note that an `output_layer` is added for the model, which is **frozen** as 1 and passes the original output of the model. The `output_layer` can be exploited for computation of the IRM regularization.
6. Implement the class `groupDRO` and `IRM` which are subclasses of `Loss`.
7. Edit the file `run.sh` and set the `seed` argument to the **last four digits** of your student ID. Run the following commands to test your implementation:

```
chmod +x run.sh
./run.sh
```

The log is saved at `IRM_log.txt` and `groupDRO_log.txt`. Inspect the output to ensure the accuracy metric satisfies the requirement.

8. Submit the **code** and the **log** file. Follow the below structure and naming conventions and package them in `zip` format for upload, with the file name `studentID_name.zip`. Please **DO NOT** upload the

downloaded dataset.

```
└─studentID_name
    ├─README.pdf
    │
    └─src
        ├─dataset.py
        ├─exp-mnist.py
        ├─LICENSE
        ├─metric.py
        ├─model.py
        ├─preprocessing.py
        ├─register.py
        ├─requirements.txt
        ├─run.sh
        ├─IRM_log.txt
        ├─groupDRO_log.txt
        └─trainer.py
```

Please note:

- The ColoredMNIST dataset is randomly generated during the initial run, but remains consistent for all subsequent runs. Deleting the generated `ColoredMNIST` folder will trigger a new random generation.
- The code is run on GPU by default. Use the argument `--no_cuda` to run codes on CPU.
- The required accuracy can be obtained under the default hyperparameter if the algorithms are correctly implemented. Hyperparameter tuning is allowed but discouraged.
- If the ColoredMNIST dataset cannot be automatically downloaded. Access the dataset from here. and put the downloaded `.pt` files in the directory `data/ColoredMNIST`.
- The lab report is **not required**.
- For errors, ambiguities, and bugs in the documentation and experiment framework, please contact: lkh20@mails.tsinghua.edu.cn
- PyTorch online documentation: https://pytorch.org/docs/stable/index.html

References

[1] Arjovsky, M., Bottou, L., Gulrajani, I., & Lopez-Paz, D. (2019). Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.

[2] Gulrajani, I., & Lopez-Paz, D. (2020). In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*.

[3] Sagawa, Shiori, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization." *arXiv preprint arXiv:1911.08731* (2019).