Meta-Learning Symmetries with sparse reparametrization

***Abstract*—Some Deep Learning architectures have been designed to be equivariant to certain transformation. The most common example is the convolutional neural network which is equivariant to translations of the input. When the underlying symmetry is known, such networks improve generalization while reducing the model size. However, as this might not be the case, symmetries could be learnt by data. This idea was elaborated in [1], in which they meta-learn a common representation of the symmetries among tasks. However, this representation is subject to instabilities and requires a huge amount of tasks in the learning process. We propose to make a step toward stability and generalization by enforcing sparsity to the symmetry representation.**

***Index Terms*—meta-learning, sparsity, symmetry, equivariance**

I. Introduction

A. Meta-Learing Symmetries by Reparametrization

[1] propose to incorporate a sharing pattern in fully connected layers aiming at generalizing equivariant layers to multiple symmetry transformation. Instead of having independently for each task a fully connected layer $\phi : x \in \mathbb{R}^n \rightarrow Wx \in \mathbb{R}^m$ parameterized by its own weight matrix $W \in \mathbb{R}^{m \times n}$, the weight matrix is factorized in the following way :

$$\text{vec}(W) = Uv,\ v \in \mathbb{R}^k, U \in \mathbb{R}^{mn \times k} \tag{1}$$

where U is the so called "symmetry matrix" shared among tasks and v the task specific "filter parameters".

The symmetry matrix can become very expensive when m,n or k become larger. Thus, the authors used the Kronecker factorization for larger k in their experiments. Basically, they reparameterized the weight matrix using the Kronecker product of two smaller matrices for representing U :

$$\text{vec}(W) = (U_1 \otimes U_2)V,\ V \in \mathbb{R}^{k \times l}, U_1 \in \mathbb{R}^{n \times l}, U_2 \in \mathbb{R}^{m \times k} \tag{2}$$

Hence, the number of entries for the symmetry matrix shrinks from $mnkl$ to $mk + nl$.

The authors propose to interpret this reparametrization as a layer implementing group convolution. Given a finite group $G = \{g_1, .., g_m\}$, $U^G \in \mathbb{R}^{\gtrdot \ltimes \times \times \ltimes}$ is the symmetry matrix storing the representations $\pi(g_i) \in \text{GL}_n(\mathbb{R})$ for each $i = 1..m$. They prove that for any filter $v \in \mathbb{R}^n$, the layer with weights $U^G v$ regarded in $\mathbb{R}^{m \times n}$ implements G-convolution, and that any G-convolution can be represented by weights $U^G v$ for some $v \in \mathbb{R}^n$.

B. Sparsity in Neural Networks

Numerous methods have been proposed to impose sparsity in neural networks parameters, in order to have a more compact and efficient network. [2] propose to exploit both positive and negative correlations between features to impose sparsity and remove redundancies. Their idea is to combine the 2-norm aiming at reducing the variance of the model and reducing the generalization error, and the 1-norm which promotes sparsity.

Grouping model weights allows to have a targeted regularization of the parameters. One can group the weights from the same feature to different latent dimension, or conversely those from the same latent dimension and different features.

Let $\mathbf{W}$ be the set of weights, $g \in \mathcal{G}$ a weight group and $w_{g,i}$ the weight at index i of group g. The (2,1)-norm or group sparsity regularization is defined as :

$$\Omega(\mathbf{W}) = \sum_g \sqrt{\sum_i w_{g,i}^2} \tag{3}$$

Group sparsity enforces competition between groups. In case each group is associated to a feature, then it would result in discarding some features.

The (1,2)-norm or exclusive sparsity regularization is defined as :

$$\Omega(\mathbf{W}) = \frac{1}{2} \sum_g \left(\sum_i |w_{g,i}| \right)^2 \tag{4}$$

Exclusive sparsity however enforces competition within groups and results in even weights among the groups.

As both of these losses include non-differentiable terms, one can not use usual gradient descend optimization algorithms. The proximal gradient algorithm can be used in such cases where the final loss is a combination of smooth terms such as the loss of the training objective, and non-smooth terms such as the regularization terms.

This method consists in taking one step using the gradient of the objective loss $\mathcal{L}(W, \mathcal{D})$ with respect to the parameters W_t giving $W_{t+1/2}$, and then performing an Euclidean projection with respect to the following expression :

$$\min_{W_{t+1}} \Omega(W_{t+1}) + \frac{1}{2\lambda s} ||W_{t+1} - W_{t+1/2}||_2^2 \tag{5}$$

The solution to this projection can be computed in closed form with respect to both group and exclusive sparsity regularization which gives :

$$prox_{GL}(W) = \left(1 - \frac{\lambda}{||w_g||_2}\right) w_{g,i} \tag{6}$$

$$prox_{EL}(W) = sign(w_{g,i})(1 - \lambda ||w_g||_1)_+ \tag{7}$$

II. Method

The idea of this method is to incorporate inductive bias in the learning process of the symmetry matrix. Firstly, we observe that due to the huge number of entries in the matrix, Secondly, by interpreting the symmetry matrix as a representation of a finite group such as in [1], one might expect to have some sharing pattern among the entries. Indeed, as a group is stable by composition, representation of elements of the group the group tend to be similar in a certain way. For common groups, the representation of their elements can be very sparse, such as for the translation groups of sifting symmetries. Thus, we would like to enforce sparsity and parameter sharing to improve stability and generalization.

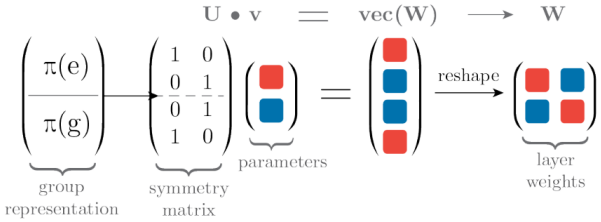

Fig. 1: Reparametrization with group symmetry matrix

Algorithm 1: MSR : Meta-Learning with regularization

input: $\{\mathcal{T}_j\}_{j=1}^{N} \sim p(\mathcal{T})$:Tasks
input: U, v: Random symmetry and filter matrices
input: α, η, λ: Inner and outer learning rates
input: λ: Regularization parameter
while *not done* **do**
  sample minibatch $\{\mathcal{T}_i\}_{i=1}^{n} \sim \{\mathcal{T}_j\}_{j=1}^{N}$
  for $\mathcal{T}_i \in \mathcal{T}_i\}_{i=1}^{n}$ **do**
    $\{\mathrm{D}_i^{tr}, \mathcal{D}_i^{val}\} \leftarrow \mathcal{T}_i$
    $\delta_i \leftarrow \nabla_v \mathcal{L}(U, v, \mathcal{D}_i^{tr})$
    v' $\leftarrow v - \alpha\delta_i$
    $G_i \leftarrow \frac{d}{dU}\mathcal{L}(U, v, \mathcal{D}_i^{tr})$
  U $\leftarrow U - \eta \sum_i G_i$
  $\underline{\mathrm{U}} \leftarrow prox_*(U)$

III. Experiments

A. Locally connected layer

In this experiment, the data is synthetically generated using a simple locally connected layer and we are aiming for recovering transitional symmetries. For each task, one generates random one-dimentional samples that are fed into a locally connected layer having filter sizes $s = 3$ or 5. The filters at each position are a random linear combination of a common basis of cardinality $r = 1, 2$ or 5, denoted as the rank. When the rank is 1, we recover a classical convolution layer (strictly speaking, the filters are just collinear). When applying the method, the symmetry matrix U matrix should recover the structure of the locally connected layer which is common to all tasks, and the parameter v should learn task-specific information related to the filters.

Data type	Base			
	20	50	100	1000
rank1 kernel3	0.675	**0.297**	**0.342**	0.294
rank2 kernel3	0.919	**0.471**	**0.67**	**0.409**
rank5 kernel3	1.364	0.909	**0.389**	
rank2 kernel5	3.146	1.957	0.976	**1.08**
rank5 kernel5	1.814	1.397	1.096	**0.628**

(a) Without regularization

Data type	Sparsity			
	20	50	100	1000
rank1 kernel3	**0.322**	1.816	0.452	**0.263**
rank2 kernel3	**0.515**	0.543	0.671	0.596
rank5 kernel3	**0.611**	**0.5**	0.416	
rank2 kernel5	**2.26**	**1.413**	**0.916**	1.244
rank5 kernel5	**1.204**	**0.895**	**0.764**	0.669

(b) With exclusive sparsity regularization

TABLE I: Losses on test dataset

The sparsity inducing method was tested using with both group regularization techniques : exclusive sparsity and group sparsity. We decided to group the weights by input feature, that is each each weight that is multiplied by a particular dimension of the input. Exclusive sparsity appeared to have much better results than group sparsity. Indeed, this method results in weights involving a particular feature to compete. Thus output features will tend to be influenced by few input features and induce sparsity. This coherent with the construction of locally connected layers in which output features depend on neighbouring corresponding input features. On the other side group sparsity would enforce different output features to compete against each other. In this setup however, there is no reason that any output feature gets more influence than others, especially if we consider multiple tasks.

For each of the problems and for 20, 50, 100 and 1000 we generated a dataset containing 20 data samples per task. Then we trained two models : on with exclusive sparsity regularization and one without. In [**?**], we display the absolute weights of the symmetry matrix learnt across the dimension n and m based on the notation on the first section, where we took the maximum among the dimension k. [**?**] shows the distribution of the absolute weights. In [**?**], we compared the loss values obtained on the test dataset.

We can see that the regularization helps to capture a better representation of the real generation scheme, as it includes much more sparsity in the symmetry matrix. Also, it achieves better results on the test dataset when the problems are more complex and when we have fewer tasks.

References

[1] A. Zhou, T. Knowles and C. Finn, "Meta-Learing Symmetries by Reparametrization," *ICLR 2021*.

[2] J. Yoon, and S. J. Hwang, "Combined Group and Exclusive Sparsity for Deep Neural Networks,"*PMLR 2017*.

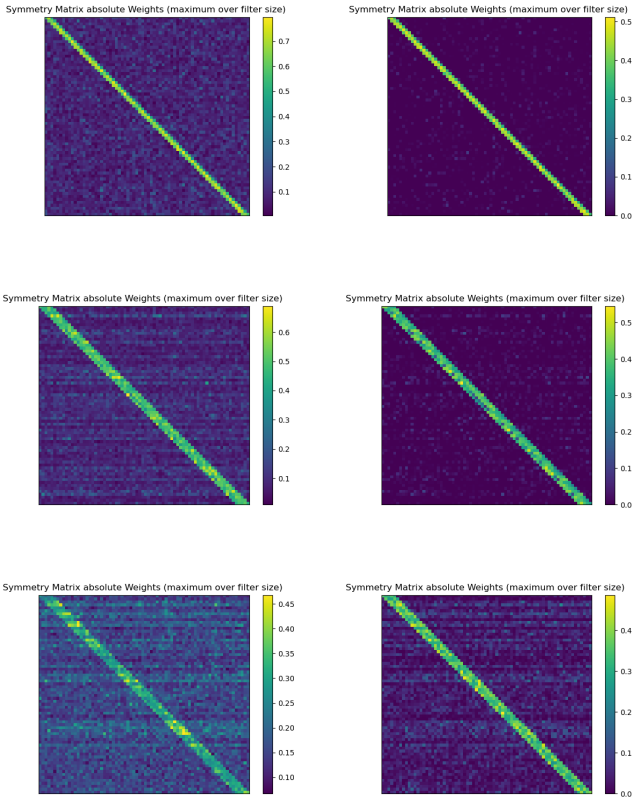

Fig. 2: Visualisation of symmetry matrices learnt with (right) and without (left) regularization. The data corresponding to the upper figures includes 20 tasks of locally connected having filter size 3 and rank 1 (shared filter). For the middle figures, the data includes 50 tasks of locally connected having filter size 5 and rank 2. For the lower figures, the data includes 50 tasks of locally connected having filter size 5 and rank 5.

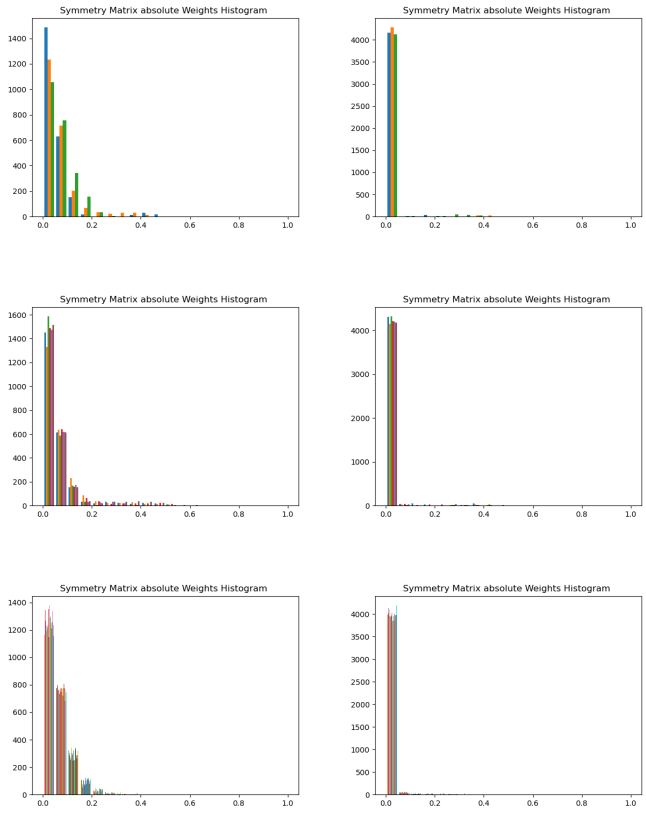

Fig. 3: Distribution of the absolute weights learnt with (right) and without (left) regularization. The data corresponding to the upper figures includes 20 tasks of locally connected having filter size 3 and rank 1 (shared filter). For the middle figures, the data includes 50 tasks of locally connected having filter size 5 and rank 2. For the lower figures, the data includes 50 tasks of locally connected having filter size 5 and rank 5. Each task include 20 samples.