Supplementary Material for Learning Algebraic Representation for Systematic Generalization in Abstract Reasoning

Chi Zhang[⋆,1], Sirui Xie[⋆,1], Baoxiong Jia[⋆,1]
Ying Nian Wu[1], Song-Chun Zhu[1,2,3,4], and Yixin Zhu[2]

[1] University of California, Los Angeles, CA 90095, USA
[2] Peking University, Haidian, Beijing 10080, China
[3] Tsinghua University, Haidian, Beijing 10080, China
[4] Beijing Institute for General Artificial Intelligence, Haidian, Beijing 10080, China
⋆ indicates equal contribution
{chi.zhang,srxie}@ucla.edu, {ywu,sczhu}@stat.ucla.edu,
yixin.zhu@pku.edu.cn
Project Website: http://wellyzhang.github.io/project/alans.html

1 Inducing and Executing Operators

In the main text, we exemplify the induction and the execution process using a binary operator. Here, we discuss other details regarding the formulation for all three types of operators, *i.e.*, unary, binary, and ternary.

Unary Operator To induce the unary operator $\mathcal{T}_u^a$ for an attribute a, we solve the following optimization problem

$$\mathcal{T}_u^a = \arg\min_{\mathcal{T}} \ell_u^a(\mathcal{T}), \tag{1}$$

where

$$\begin{aligned} \ell_u^a(\mathcal{T}) = 1/5 \times \Big(& \mathbb{E}\left[\left\|M(b_{o,1}^a)\mathcal{T} - M(b_{o,2}^a)\right\|_F^2\right] + \\ & \mathbb{E}\left[\left\|M(b_{o,2}^a)\mathcal{T} - M(b_{o,3}^a)\right\|_F^2\right] + \\ & \mathbb{E}\left[\left\|M(b_{o,4}^a)\mathcal{T} - M(b_{o,5}^a)\right\|_F^2\right] + \\ & \mathbb{E}\left[\left\|M(b_{o,5}^a)\mathcal{T} - M(b_{o,6}^a)\right\|_F^2\right] + \\ & \mathbb{E}\left[\left\|M(b_{o,7}^a)\mathcal{T} - M(b_{o,8}^a)\right\|_F^2\right] \Big) + \\ & \lambda_u^a \left\|\mathcal{T}\right\|_F^2 . \end{aligned} \tag{2}$$

The indexing follows the row / column major. By taking the derivative with respect to $\mathcal{T}$ and setting it to be $\mathbf{0}$, we have the following solution,

$$\mathcal{T}_u^a = A^{-1}B \tag{3}$$

where, assuming independence,

$$\begin{aligned} A =&\mathbb{E}\left[M(b^a_{o,1})^T M(b^a_{o,1})\right] + \mathbb{E}\left[M(b^a_{o,2})^T M(b^a_{o,2})\right] + \\ &\mathbb{E}\left[M(b^a_{o,4})^T M(b^a_{o,4})\right] + \mathbb{E}\left[M(b^a_{o,5})^T M(b^a_{o,5})\right] + \\ &\mathbb{E}\left[M(b^a_{o,7})^T M(b^a_{o,7})\right] + 5\lambda^a_u I \end{aligned} \tag{4}$$

and

$$\begin{aligned} B =&\mathbb{E}\left[M(b^a_{o,1})^T\right] \mathbb{E}\left[M(b^a_{o,2})\right] + \\ &\mathbb{E}\left[M(b^a_{o,2})^T\right] \mathbb{E}\left[M(b^a_{o,3})\right] + \\ &\mathbb{E}\left[M(b^a_{o,4})^T\right] \mathbb{E}\left[M(b^a_{o,5})\right] + \\ &\mathbb{E}\left[M(b^a_{o,5})^T\right] \mathbb{E}\left[M(b^a_{o,6})\right] + \\ &\mathbb{E}\left[M(b^a_{o,7})^T\right] \mathbb{E}\left[M(b^a_{o,8})\right]. \end{aligned} \tag{5}$$

Note that as long as $\lambda^a_u > 0$, A is a symmetric positive definite matrix and hence is invertible. Compared to the binary case, the unary operator can be regarded as a special binary operator where one of the operand is a constant, absorbed into operator learning, and jointly solved.

To predict the answer representation, we solve another optimization problem, *i.e.*,

$$\widehat{M^a_u} = \arg\min_M \ell^a_u(M) = \mathbb{E}\left[\left\|M(b^a_{o,8})\mathcal{T}^a_u - M\right\|^2_F\right]. \tag{6}$$

Taking its derivative and setting it to $\mathbf{0}$, we have

$$\widehat{M^a_u} = \mathbb{E}\left[M(b^a_{o,8})\right]\mathcal{T}^a_u. \tag{7}$$

Note that this is exactly the execution of the learned operator.

Binary Operator The optimization problem for the binary case can be formulated as

$$\mathcal{T}^a_b = \arg\min_{\mathcal{T}} \ell^a_b(\mathcal{T}), \tag{8}$$

where

$$\begin{aligned} \ell^a_b(\mathcal{T}) = 1/2 \times \Big(&\mathbb{E}\left[\left\|M(b^a_{o,1})\mathcal{T}M(b^a_{o,2}) - M(b^a_{o,3})\right\|^2_F\right] + \\ &\mathbb{E}\left[\left\|M(b^a_{o,4})\mathcal{T}M(b^a_{o,5}) - M(b^a_{o,6})\right\|^2_F\right]\Big) + \\ &\lambda^a_b \left\|\mathcal{T}\right\|^2_F. \end{aligned} \tag{9}$$

We note that, assuming independence, the solution satisfies

$$\begin{aligned} &\mathbb{E}\left[M(b^a_{o,1})^T M(b^a_{o,1})\right]\mathcal{T}\mathbb{E}\left[M(b^a_{o,2}) M(b^a_{o,2})^T\right] + \\ &\mathbb{E}\left[M(b^a_{o,4})^T M(b^a_{o,4})\right]\mathcal{T}\mathbb{E}\left[M(b^a_{o,5}) M(b^a_{o,5})^T\right] + 2\lambda^a_b\mathcal{T} \\ =&\mathbb{E}\left[M(b^a_{o,1})^T\right]\mathbb{E}\left[M(b^a_{o,3})\right]\mathbb{E}\left[M(b^a_{o,2})^T\right] + \\ &\mathbb{E}\left[M(b^a_{o,4})^T\right]\mathbb{E}\left[M(b^a_{o,6})\right]\mathbb{E}\left[M(b^a_{o,5})^T\right]. \end{aligned} \tag{10}$$

This is a linear matrix equation and can be turned into a linear equation by vectorization. Using $\mathrm{vec}(A\mathcal{T}B) = A \otimes B\, \mathrm{vec}(\mathcal{T})$ [4], where $\otimes$ denotes the Kronecker product, we have

$$\mathrm{vec}(\mathcal{T}_b^a) = A^{-1}B, \tag{11}$$

where

$$\begin{aligned} A =& \mathbb{E}\left[M(b_{o,1}^a)^T M(b_{o,1}^a)\right] \otimes \mathbb{E}\left[M(b_{o,2}^a) M(b_{o,2}^a)^T\right] + \\ & \mathbb{E}\left[M(b_{o,4}^a)^T M(b_{o,4}^a)\right] \otimes \mathbb{E}\left[M(b_{o,5}^a) M(b_{o,5}^a)^T\right] + \\ & 2\lambda_b^a I \end{aligned} \tag{12}$$

and

$$\begin{aligned} B =& \mathrm{vec}\left(\mathbb{E}\left[M(b_{o,1}^a)^T\right] \mathbb{E}\left[M(b_{o,3}^a)\right] \mathbb{E}\left[M(b_{o,2}^a)^T\right]\right) + \\ & \mathrm{vec}\left(\mathbb{E}\left[M(b_{o,4}^a)^T\right] \mathbb{E}\left[M(b_{o,6}^a)\right] \mathbb{E}\left[M(b_{o,5}^a)^T\right]\right). \end{aligned} \tag{13}$$

Note that A is also symmetric positive definite given positive λ_b^a and hence invertible.

The predicted answer representation is given by

$$\widehat{M_b^a} = \arg\min_M \ell_b^a(M) = \mathbb{E}\left[\left\| M(b_{o,7}^a) \mathcal{T}_b^a M(b_{o,8}^a) - M \right\|_F^2\right], \tag{14}$$

which can be solved by executing the induced binary operator

$$\widehat{M_b^a} = \mathbb{E}\left[M(b_{o,7}^a)\right] \mathcal{T}_b^a \mathbb{E}\left[M(b_{o,8}^a)\right]. \tag{15}$$

Ternary Operator A ternary operation can be regarded as a unary operation on elements defined on rows / columns. Specifically, we propose to construct the algebraic representation of a row / column by concatenating the algebraic representation of each panel in it, *i.e.*,

$$M(b_{o,i}^a, b_{o,i+1}^a, b_{o,i+2}^a) = [M(b_{o,i}^a); M(b_{o,i+1}^a); M(b_{o,i+2}^a)]. \tag{16}$$

Then the ternary operator can be solved by

$$\mathcal{T}_t^a = \arg\min_{\mathcal{T}} \ell_t^a(\mathcal{T}), \tag{17}$$

where

$$\begin{aligned} \ell_t^a(\mathcal{T}) =& \mathbb{E}\left[\left\| M(b_{o,1}^a, b_{o,2}^a, b_{o,3}^a)\mathcal{T} - M(b_{o,4}^a, b_{o,5}^a, b_{o,6}^a) \right\|_F^2\right] + \\ & \lambda_t^a \left\| \mathcal{T} \right\|_F^2. \end{aligned} \tag{18}$$

Similar to the unary case discussed above,

$$\mathcal{T}_t^a = A^{-1}B \tag{19}$$

where

$$A = \mathbb{E}\left[M(b_{o,1}^a, b_{o,2}^a, b_{o,3}^a)^T M(b_{o,1}^a, b_{o,2}^a, b_{o,3}^a)\right] + \lambda_t^a I \tag{20}$$

and

$$B = \mathbb{E}\left[M(b_{o,1}^a, b_{o,2}^a, b_{o,3}^a)^T\right] \mathbb{E}\left[M(b_{o,4}^a, b_{o,5}^a, b_{o,6}^a)\right]. \tag{21}$$

Correspondingly, the answer representation can be obtained by first executing the ternary operator $\mathbb{E}\left[M(b^a_{o,4}, b^a_{o,5}, b^a_{o,6})\right] \mathcal{T}^a_t$ and slicing it from the result.

To compute the operator distribution, we model it based on the fitness of each operator type,

$$P(\mathcal{T}^a = \mathcal{T}^a_u \mid \{I_{o,i}\}_{i=1}^8) \propto \exp(-\ell^a_u(\mathcal{T}^a_u)) \tag{22}$$

$$P(\mathcal{T}^a = \mathcal{T}^a_b \mid \{I_{o,i}\}_{i=1}^8) \propto \exp(-\ell^a_b(\mathcal{T}^a_b)) \tag{23}$$

$$P(\mathcal{T}^a = \mathcal{T}^a_t \mid \{I_{o,i}\}_{i=1}^8) \propto \exp(-\ell^a_t(\mathcal{T}^a_t)). \tag{24}$$

2 Systematic Splits

In the original work of Zhang *et al.* [7] and Hu *et al.* [2], there are four operators: Constant, Progression, Arithmetic, and Distribute of Three. Progression is parameterized by its step size (±1/2). Arithmetic includes addition and subtraction. And Distribute of Three is implemented as shifting and can be either a left shift or a right one. Note that Constant can be regarded as special Progression with a step size of 0. In this work, we group all four operators into three types: unary (Constant and Progression), binary (Arithmetic), and ternary (Distribute of Three).

To study systematic generalization in abstract relation learning, we use the Raven's Progressive Matrices (RPM) generation method proposed in Zhang *et al.* [7] and Hu *et al.* [2] and carefully split data into three regimes:

- Systematicity: The training set and the test set contain all three types of operators but disjoint instances. Specifically, the training set has Constant, Progression of ±1, addition in Arithmetic, and left shift in Distribute of Three, while in the test set there are Progression of ±2, subtraction in Arithmetic, and right shift in Distribute of Three.
- Productivity: The training set contains only unary operators and the test set only binary operators. Specifically, the training set has Constant and all instances of Progression, while the test set all instances of Arithmetic.
- Localism: The training set contains only binary operators and the test set only unary operators. Specifically, the training set has all instances of Arithmetic and the test set Constant and all instances of Progression.

Please see Figures 1 to 3 for examples in the three splits.

3 Implementation Details

3.1 Network Architecture

We use a LeNet-like architecture [5] for each branch of the object CNN. See Table 1 for the design. Note that the object CNN consists of four branches, including objectiveness, type, size, and color. The parameters for Convolution denote the output channel size, kernel size, and stride, respectively. A BatchNorm layer is parameterized by the number of channels, whereas a MaxPool layer by its stride. An output size is used to specify parameters of a Linear layer. m equals 2, 5, 6, 10 for objectiveness, type, size, and color, respectively. For numerical stability, we use LogSoftMax to turn a probability simplex into its log space.

Operator	Parameters
Convolution	$[6, 5, 1]$
BatchNorm	6
SoftPlus	
MaxPool	2
Convolution	$[16, 5, 1]$
BatchNorm	16
SoftPlus	
MaxPool	2
Linear	120
SoftPlus	
Linear	84
SoftPlus	
Linear	m
LogSoftMax	

Table 1. The network architecture used for each branch of the object CNN.

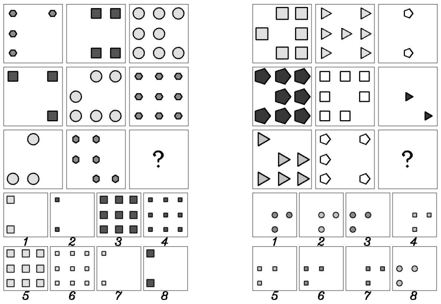

Fig. 1. A training example (left) and a test example (right) in the systematicity split. Note that in the training example, the arithmetic relation (in number) is addition and the shifting is always a left shift (in type, size, and color). In the test example, the shifting becomes a right shift (in type), the size progression has a step of 2, and the color arithmetic becomes subtraction.

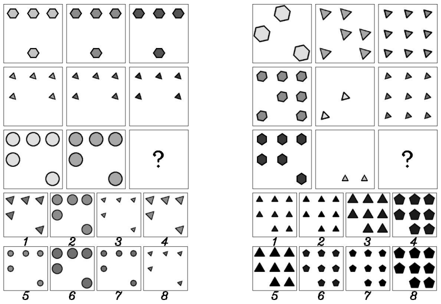

Fig. 2. A training example (left) and a test example (right) in the productivity split. Note that in the training example, the constant rule is applied to number, type, and size, while the progression rule is applied on color. In the testing example, the arithmetic rule is applied on all attributes.

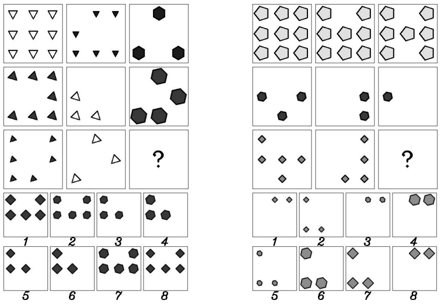

Fig. 3. A training example (left) and a test example (right) in the localism split. Note that in the training example, the arithmetic rule is on all attributes. In the test example, the progression rule is applied on number and the constant rule on all other attributes.

3.2 Other Hyperparameters

For the inner regularized linear regression, we set different regularization coefficients for different attributes but, for the same attribute, we keep them the same across all three types of operators. For position, $\lambda = 10^{-4}$. For number, $\lambda = 10^{-6}$. For type, $\lambda = 10^{-6}$. For size, $\lambda = 10^{-6}$. For color, $\lambda = 5 \times 10^{-7}$. All of the regularization terms in Eq. (10) in the main text are set to be 1 and $\{M_0^a\}$ and $\{M^a\}$ are initialized as 2×2 square matrices.

For training, we first train for 10 epochs parameters regarding objectiveness, including the objectiveness branch, and the representation matrices on position and number. We then perform 2 rounds of cyclic training on parameters regarding type, size, and color, each of which experiences 10 epochs of updates in a round. Finally, we fine-tune all parameters for another 10 epochs, totaling up to 80 training epochs. The entire system is optimized using ADAM [3] with a learning rate of 9.5×10^{-5}.

Of note, we realize that it's still hard for the model to converge despite of scheduled training. We recommend warm-starting the perception component with a limited number of annotations: 5 to 8 samples should suffice.

4 Marginalization for Other Attributes

For the attribute of position, we denote its value as R^o, a binary vector of length N, with each entry corresponding to one of the N windows. Then

$$P(\text{Position} = R^o) = \prod_{j=1}^{N} P(r_j^o = R_j^o), \tag{25}$$

where $P(r_j^o)$ denotes the jth region's estimated objectiveness distribution returned by a CNN as in the main text.

For the attribute of type, the panel attribute of type being k is evaluated as

$$P(\text{Type} = k) = \sum_{R^o} \left(\prod_{j, R_j^o = 1} P(r_j^t = k) \right) P(\text{Position} = R^o), \tag{26}$$

where $P(r_j^t)$ denotes the jth region's estimated type distribution returned by a CNN.

The computation for size and color is exactly the same as type, except that we use the region's estimated size and color distribution returned by a CNN.

5 More on Neural Visual Perception

- **Why not train a CNN to predict the position and number of objects?** The CNN is trained to predict the type, size, color, and object existence in a window. The object existence in windows is marginalized to be a

Number distribution and Position distribution. This is a light-weight method for object detection. Nevertheless, it is also possible to use a Fast-RCNN like method to predict object positions (this implies number) directly. However, in this way, the framework loses the probabilistic interpretation (the object proposal branch is currently still deterministic), and we cannot perform end-to-end learning.

- **How does the CNN predict the presence of an object, its type, size, and color given that it is not trained to do that?** For each window, the CNN outputs 4 softmaxed vectors, corresponding to the probability distributions of object existence, object type, object size, and object color. The spaces for these attributes are pre-defined. CNN's weights are then jointly trained in the framework. Such a design follows recent neuro-symbolic methods [1,6] that also rely on the implicitly trained representation. In short, we assign semantics to the implicitly trained representation (probability distributions for attributes), performs marginalization and reasoning **as if** they are ground-truth attribute distributions, and jointly train using only the problem's target label.

6 More on Algebraic Abstract Reasoning

- **Are the learned operators interpretable and aligned with ground-truth?** As different operators are of different shapes and are represented as matrices with multiple entries, it's hard to quantitatively evaluate the fitness in general. However, we have noticed the following patterns of the learned matrices in our case-by-case examination: (1) Approximate permutation matrices for the shifting operation in unary operators (Progression in position) and ternary operators (Distribute of Three). Algebraically, shifting elements in a vector is exactly implemented as the product with a permutation matrix in the representation theory. (2) Approximate identity matrix that keeps the distribution the same and it corresponds to the Constant relation. (3) Approximate successor matrices that can be understood as Progression when multiplied.

References

1. Han, C., Mao, J., Gan, C., Tenenbaum, J., Wu, J.: Visual concept-metaconcept learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
2. Hu, S., Ma, Y., Liu, X., Wei, Y., Bai, S.: Hierarchical rule induction network for abstract visual reasoning. arXiv preprint arXiv:2002.06838 (2020)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2014)
4. Lancaster, P.: Explicit solutions of linear matrix equations. SIAM Review **12**(4), 544–566 (1970)
5. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
6. Mao, J., Gan, C., Kohli, P., Tenenbaum, J.B., Wu, J.: The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In: International Conference on Learning Representations (ICLR) (2019)
7. Zhang, C., Gao, F., Jia, B., Zhu, Y., Zhu, S.C.: Raven: A dataset for relational and analogical visual reasoning. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2019)