Neural Relational Inference with Fast Modular Meta-Learning

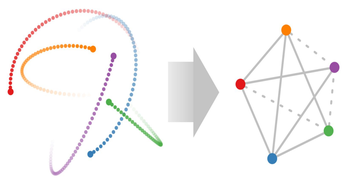
Ferran Alet, Erica Weng, Tomás Lozano-Pérez, Leslie P. Kaelbling

alet-etal.com

Key ideas

1. Graph Neural Networks with multiple types of entities and relations are useful for modeling dynamical systems.
2. We meta-learn a set of neural modules that allow us to model many dynamical systems after inferring their structure.
3. Model-based approach to relational inference is more data efficient and allows inferences for which it wasn't trained.
4. We scale up modular meta-learning, from 100 to 50k datasets, by learning a proposal function for Simulated Annealing.

Neural Relational Inference (Kipf et al. '18)



Training data

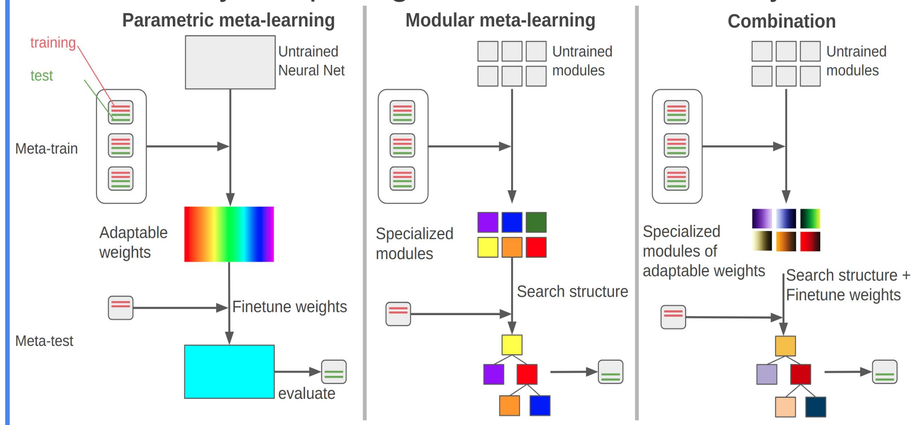
$$s_1^{1:T}, \ldots, s_n^{1:T}$$

Test data

$$s_1^{T+1:T+k}, \ldots, s_n^{T+1:T+k}$$

Modular meta-learning (Alet et al. '18)

Generalize by composing modules in different ways

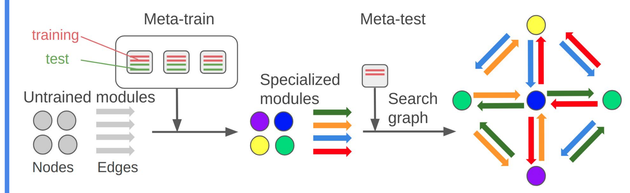


procedure BOUNCE($S_1, \ldots, S_m, D_1^{train}, \ldots, D_m^{train}, T, \mathcal{S}, \Theta$)
  for $j = 1 \ldots m$ **do**
    $S'_j = Propose_{\mathcal{S}}(S_j, \Theta)$
    if $Accept(e(D_j^{train}, S'_j, \Theta), e(D_j^{train}, S_j, \Theta), T)$ **then** $S_j = S'_j$
procedure GRAD($\Theta, S_1, \ldots, S_m, D_1^{val}, \ldots, D_m^{val}, \eta$)
  $\Delta = 0$
  for $j = 1 \ldots m$ **do**
    $(x, y) = \text{rand\_elt}(D_j^{val})$; $\Delta = \Delta + \nabla_\Theta L(S_{j_\Theta}(x), y)$
  $\Theta = \Theta - \eta\Delta$

BounceGrad

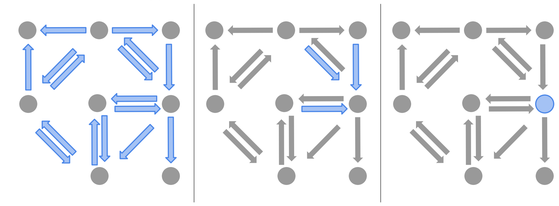
Meta-learn module weights and structures by alternating between Simulated Annealing and Gradient Descent

Graph Neural Networks



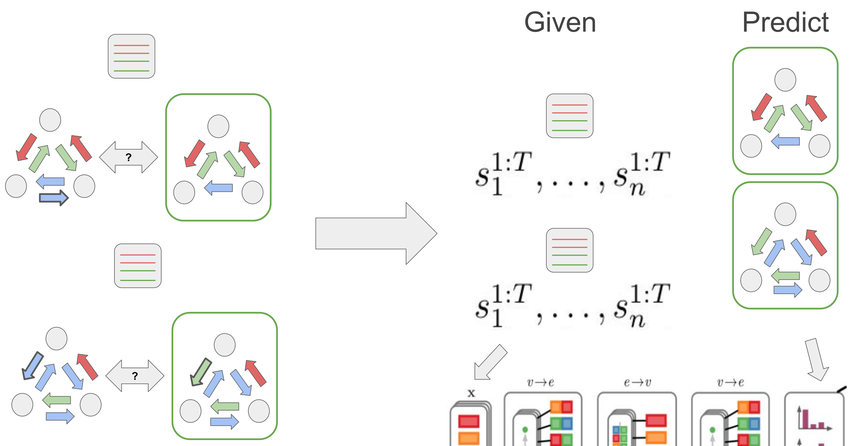
procedure PROPOSESTRUCTURE($S, \mathcal{G}, \mathbb{G}, \mathbb{H}$)
  $P \leftarrow S$
  if Bernouilli$(1/2)$ **then**
    $idx \leftarrow random\_elt(\mathcal{G}.nodes)$; $P.m_{n_{idx}} \leftarrow$random_elt($\mathbb{G}$)
  else
    $idx \leftarrow random\_elt(\mathcal{G}.edges)$; $P.m_{e_{idx}} \leftarrow$random_elt($\mathbb{H}$)
  return P
procedure EVALUATE($\mathcal{G}, S, \mathcal{L}, \boldsymbol{x}, \boldsymbol{y}$)
  $\boldsymbol{p} \leftarrow f_{out}\left(\text{MP}^{(T)}(\mathcal{G}, S)(f_{in}(\boldsymbol{x}))\right)$
  return $\mathcal{L}(\boldsymbol{y}, \boldsymbol{p})$

MP:message passing



Learning a Proposal Function

Instead of proposing random changes, a neural network learns to make good proposals by imitating current structures found by Simulated Annealing.



Simulated Annealing uses the proposal function to make good proposals for each dataset, accepting them depending on performance.

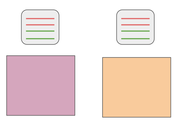
Neural network learns to predict structures from datasets, providing a prior for proposal function.

Virtuous loop similar to that of AlphaZero

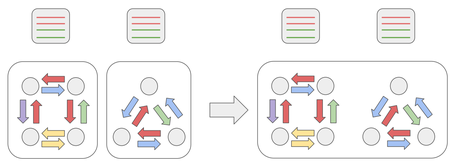
slow but consistent Simulated Annealing trains a fast neural network, which speeds up Simulated Annealing

Batching Multiple Datasets

Most non-modular meta-learning approaches cannot batch evaluations between datasets



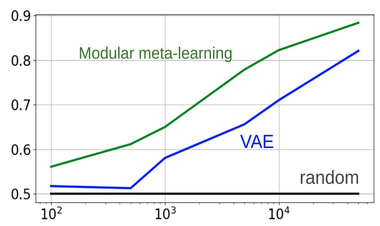
Modular representation:
batch into one graph, evaluate together



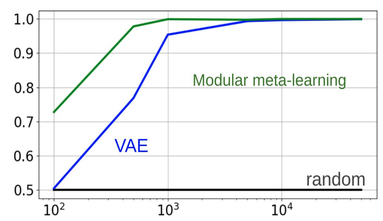
Same idea works for other modular composition, being easy to implement and giving ~5x speed-up

Higher Data Efficiency

Charged

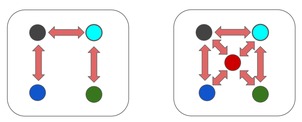


Springs



Inferring Unseen Entities

Observed | Predicted | Infer new entity



Our model-based approach allows us to infer other aspects of a scene for which we did not train, such as the trajectory of an unseen entity that affects observed entities, by putting it inside our inner optimization.

Related Work

Modular meta-learning; Alet et al. CoRL 2018

Neural relational inference; Kipf et al. NeurIPS 2018

Mastering the game of Go without human knowledge; Silver et al. Nature '17

Automatically composing representation... ; Chang et al. ICLR 2019

Future Work

Vehicle interaction dataset

Understanding the intentions of drivers