Modular Meta Learning

Chen Zhao

CoRL 2018 (`https://arxiv.org/pdf/1806.10166.pdf`)

1 Motivation

- The objective of learning a handful of data is to extract information that will substantially reduce the training-data requirements for a new task.
- Previous approaches to meta-learning have focused on finding distributions over or initial values of parameters, based on a set of "training tasks," that will enable a new "test task" to be learned with many fewer training examples. This paper focuses on finding a set of reusable modules that can form components of a solution to a new task.
- Authors wish to address problems that may benefit from a modular decomposition but do not provide any task-level input from which the structure of a solution may be derived. But they adopt a similar modular structure and parameter adaptation method for learning reusable modules, but use a general-purpose simulated-annealing search strategy to find an appropriate structural decomposition for each new task.

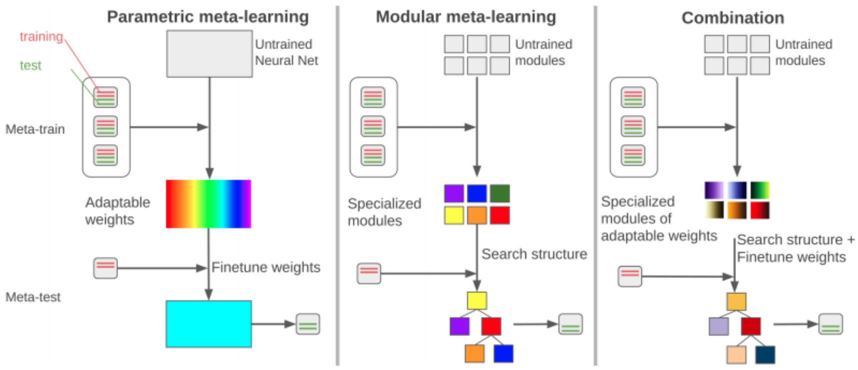

Figure 1: All methods train on a set of related tasks and obtain some flexible intermediate representation. Parametric strategies such as MAML (left) learn a representation that can be quickly adjusted to solve a new task. Our modular meta-learning method (middle) learns a repertoire of modules that can be quickly recombined to solve a new task. A combination of MAML and modular meta-learning (right) learn initial weights for modules that can be combined and adapted for a new task.

2 Problem Formulation and Learning Algorithm

The approach has two phases: an off-line meta-learning phase and an on-line meta-test learning phase. In the meta-learning phase, the proposed approach takes training and validation data sets for tasks $1, ..., k$ as input and generate a parametrization for each module, $\Theta = (\theta_1, ..., \theta_k)$ as output; the objective is to construct modules that will work together as good building blocks for future tasks. In the metatest learning phase, a set of possible structures $\mathbb{S}$ and Θ are taken as input; the output is a compositional from $S \in \mathbb{S}$ which includes a selection of modules $f_1, ..., f_{m_s}$ to be used in that form.

At meta-learning time, $\mathbb{S}$ is specified, and the objective is to find parameter values Θ that constitute a set of modules that can be recombined to effectively solve each of the training tasks. The training objective is to find Θ that minimizes the average generalization performance using parameter set Θ.

$$J(\Theta) = \sum_{j=1}^{m} e(D_j^{test}, \arg\min_{S \in \mathbb{S}} e(D_j^{train}, S, \Theta), \Theta)$$

The overall learning algorithm for optimizing:

procedure BOUNCEGRAD($\mathbb{S}$, $D_1^{train}, \ldots, D_m^{train}$, $D_1^{test}, \ldots, D_m^{test}$, η, T_0, Δ_T, T_{end})
  $S_1, \ldots, S_m$ = random simple structures from $\mathbb{S}$; Θ = neural-network weight initialization
  for $T = T_0;\ T = T - \Delta_T;\ T < T_{end}$ **do**
    BOUNCE($S_1, \ldots, S_m$, $D_1^{train}, \ldots, D_m^{train}$, T, $\mathbb{S}$, Θ)
    GRAD(Θ, $S_1, \ldots, S_m$, $D_1^{test}, \ldots, D_m^{test}$, η)

procedure BOUNCE($S_1, \ldots, S_m$, $D_1^{train}, \ldots, D_m^{train}$,$T$, $\mathbb{S}$, Θ)
  for $j = 1 \ldots m$ **do**
    $S'_j = Propose_{\mathbb{S}}(S_j, \Theta)$
    if $Accept(e(D_j^{train}, S'_j, \Theta), e(D_j^{train}, S_j, \Theta), T)$ **then** $S_j = S'_j$

procedure GRAD(Θ, $S_1, \ldots, S_m$, $D_1^{test}, \ldots, D_m^{test}$, η)
  $\Delta = 0$
  for $j = 1 \ldots m$ **do**
    $(x, y) = \mathrm{rand\_elt}(D_j^{test})$; $\Delta = \Delta + \nabla_\Theta L(S_{j_\Theta}(x), y)$
  $\Theta = \Theta - \eta\Delta$