Modular Meta-Learning

Authors: Ferran Alet, TomásLozano-Perez, Leslie P. Kaelbling

Presenters: Nathaniel Chong

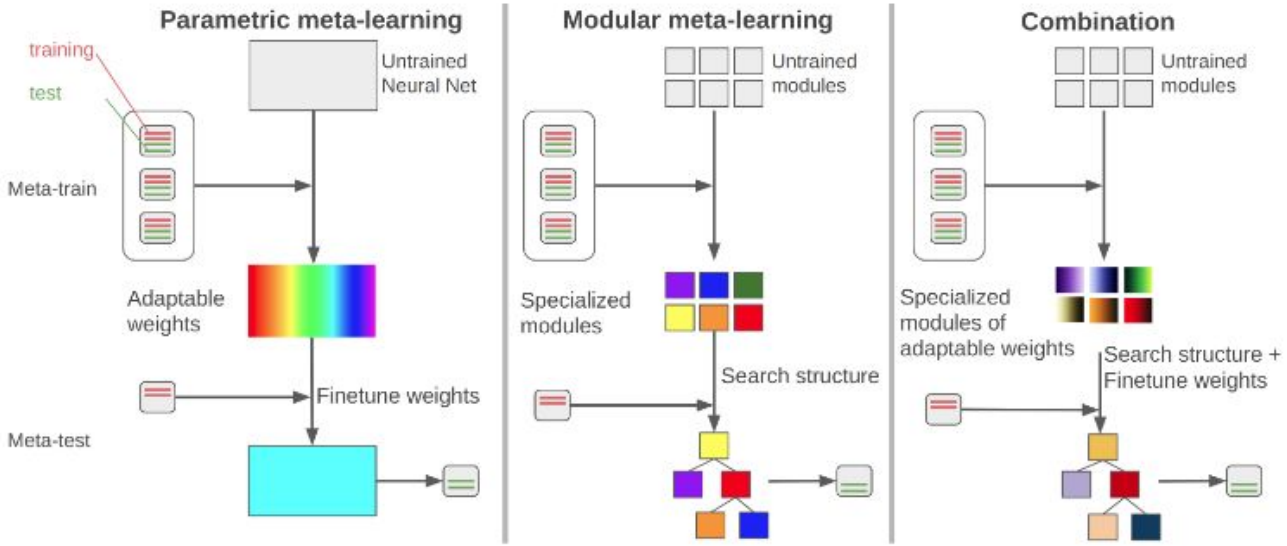
Background - Meta Learning

- Robot learning is expensive
- Helpful to learn a prior from related tasks to reduce training overhead
- **Prior Work** :
  - Find distributions or initial values of parameters
  - MAML
  - Parametric meta learning methods

This Paper: A modular framework for meta learning

Modular Meta Learning

- Learn repertoire of modules instead of set of initial weights/representation!
- **Combinatorial Generalization**: Combine finite set of modules to solve new tasks



Motivation & Related Work

- Modular approaches successful in structured tasks
- Increased interpretability: Observe what modules utilized in new tasks
- **Related Work** :
  - PathNET:
    - Layered modules, evolutionary algorithm to gate connections
  - Neural Module Networks
  - Structured Networks
    - Master policy composes sub-policies, single fixed scheme of composition

Problem Setup

- Context: Supervised learning problems with training/validation input pairs
- Underlying distribution over set of tasks
- **Given:** Data from m training tasks, and a small amount of data from a meta-test task
- **Find:** Hypothesis ***h*** that minimizes expected loss on meta-test task

Hypothesis Space

Represented by **(C, *F*, *Θ*)**

- **F:** Basis set of modules represented as neural networks f_i
- ***Θ*:** Weights of each neural network (module)
- **C:** Compositional scheme that describes how modules can be combined to form complex ones. Initial structure and set of local modification operations

$$h(x) = f_i(x)$$

Single Module

$$h(x) = \sum_{l=1}^{m} \frac{e^{f_{i_l}(x)}}{\sum_{l'=1}^{m} e^{f_{i_{l'}}(x)}} g_{j_l}(x)$$

Weighted Ensemble

Approach

- **Phase 1:** Off-line meta-learning phase

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

  - Find Θ for each module using train/validation splits
  - Minimize generalization performance of hypothesis chosen in (2)

- **Phase 2:** Online meta-test learning phase

$$S_\Theta^* = \arg\min_{S \in \mathbb{S}} e(D_{meta\text{-}test}^{train}, S, \Theta)$$

  - Fixed Θ, find best structure/hypothesis/composition of modules

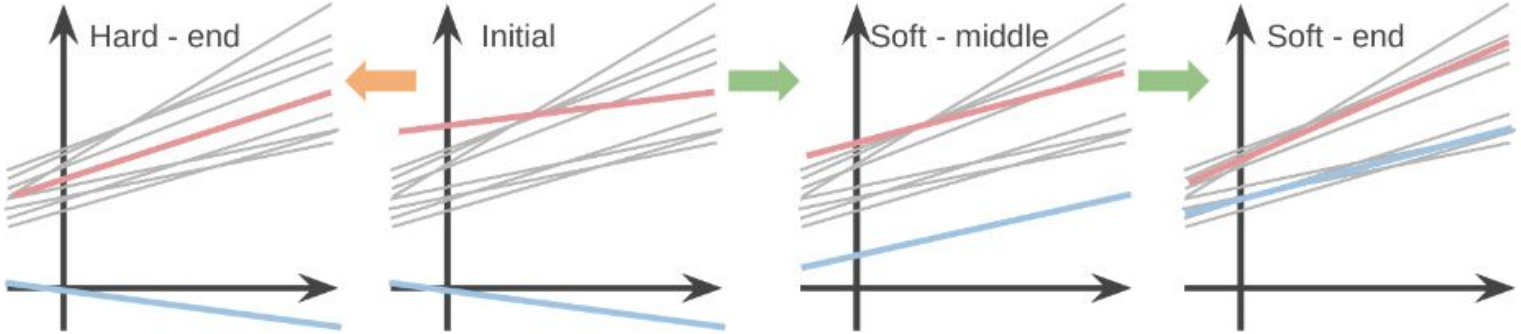
- **Phase 3:** Online parameter-tuning, incorporating MAML objective

Difficulty of Meta-Training

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

- Chicken-Egg problem: We don't know module weights, nor best structure
- To choose best structure, need good modules
- Cannot train modules, if you don't know how they should interact
- Few points per task

This causes a risk of premature optimization and local optima!



Meta-Test Learning Cont. (Phase 2)

- **Recall:** Find optimal structure given fixed Θ

$$S^*_\Theta = \arg\min_{S \in \mathbb{S}} e(D^{train}_{meta\text{-}test}, S, \Theta)$$

- Employ simulated annealing on training set of meta-test task
  - Stochasticity to combat local optima
  - Asymptotic optimality guarantees

procedure ONLINE($D^{train}_{meta-test}$, $\mathbb{S}$, Θ, T_0, Δ_T, T_{end})
  S = random simple structure from $\mathbb{S}$
  for $T = T_0;\ T = T - \Delta_T;\ T < T_{end}$ **do**
    $S' = \text{PROPOSE}_{\mathbb{S}}(S)$
    if ACCEPT($e(D, S', \Theta), e(D, S, \Theta), T$) **then** $S = S'$
  return S
procedure ACCEPT(v', v, T)
  return $v' < v$ or rand(0, 1) $< \exp\{(v - v')/T\}$

Meta Learning Cont. (Phase 1)

- **Recall:** Find Θ for each module using train/validation splits $J(\Theta) = \sum_{j=1}^{m} e(D_j^{test}, \arg\min_{S \in \mathbb{S}} e(D_j^{train}, S, \Theta), \Theta)$
- Smooth local optima by changing objective function

$$\widehat{J}(\Theta, T) = \sum_{j=1}^{m} \mathbb{E}_{S \sim \mathrm{MC}(\mathbb{S}, v(s;\Theta), T)} e(D_j^{test}, S, \Theta)$$

- Find best Θ in expectation given distribution of structures induced by Markov Chain
- Markov Chain obtained from simulated annealing procedure

Meta Learning - Bounce

procedure BOUNCE($S_1, \ldots, S_m, D_1^{train}, \ldots, D_m^{train}, T, \mathbb{S}, \Theta$)
  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** $S_j = S'_j$

- Sample structures based on simulated annealing step (Markov Chain)
- Uses current Θ values, for fixed temperature T

Meta Learning - Grad

- Given particular structure from Bounce, we obtain a feed-forward differentiable function
- Adjust Θ using simple gradient descent

procedure GRAD($\Theta, S_1, \ldots, S_m, D_1^{test}, \ldots, D_m^{test}, \eta$)

$\Delta = 0$

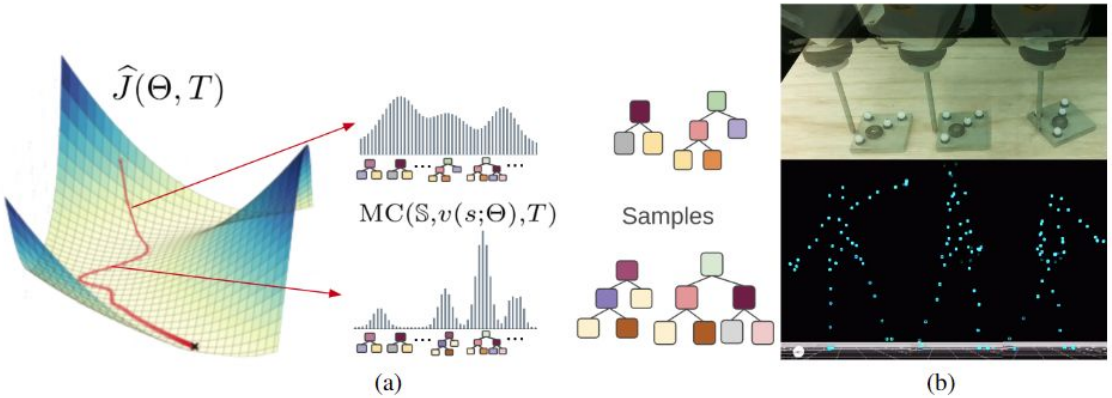
for $j = 1 \ldots m$ **do**

$(x, y) = \text{rand\_elt}(D_j^{test})$; $\Delta = \Delta + \nabla_\Theta L(S_{j\Theta}(x), y)$

$\Theta = \Theta - \eta\Delta$

Meta Learning - BOUNCEGRAD

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($S_1, \ldots, S_m, D_1^{train}, \ldots, D_m^{train}, T, \mathbb{S}, \Theta$)
    GRAD($\Theta, S_1, \ldots, S_m, D_1^{test}, \ldots, D_m^{test}, \eta$)



Parameter Tuning - MOMA (Phase 3)

- After BOUNCEGRAD, can run more gradient steps on Θ to finetune on training data
- Incorporate MAML objective into BOUNCEGRAD (Phase 1) and ONLINE (Phase 2)
- $e_{\text{MAML}}(D, S, \Theta) = \sum_{\{(x,y)\in D\}} L(S_{O(\Theta,D,S)}(x), y)$ $\quad O(\Theta, D, S) = \Theta - \eta \nabla_\Theta e(D, S, \Theta)$
- Use this e_{MAML} in BOUNCEGRAD and ONLINE

Experiments - Simple Function Relations

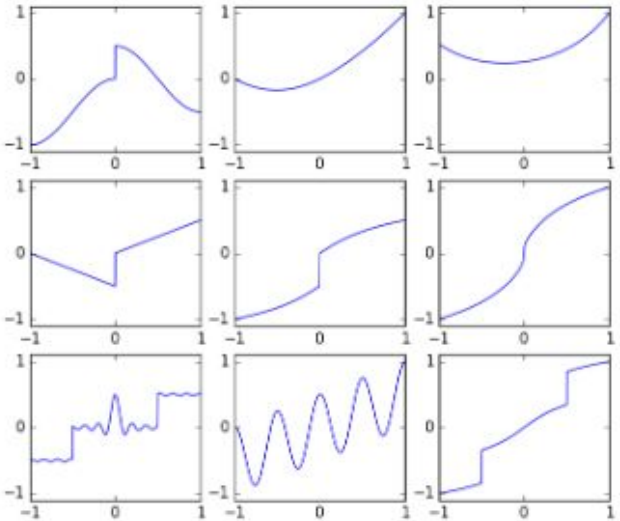
- Predict values of functions given input values
- *F* consists of 20 feed forward neural networks (half 1 layer, half 2 layers)
- 230 randomly generated tasks for meta-learning
- Sine waves and common non-linear functions studied
- *C:* $h(x) = f_i(f_j(x))$ $\quad h(x) = f_i(x) + f_j(x)$

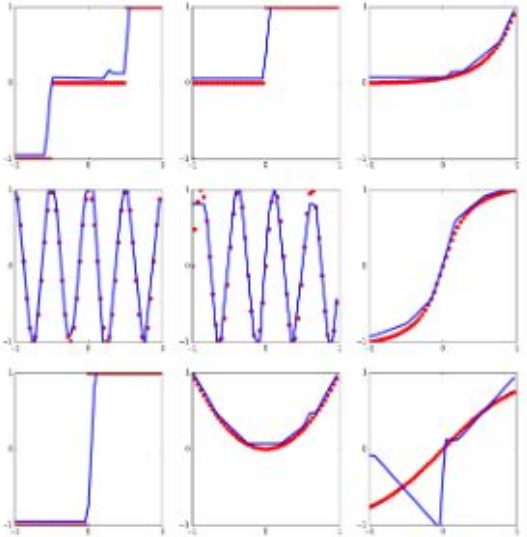
Experiments - Simple Function Relations

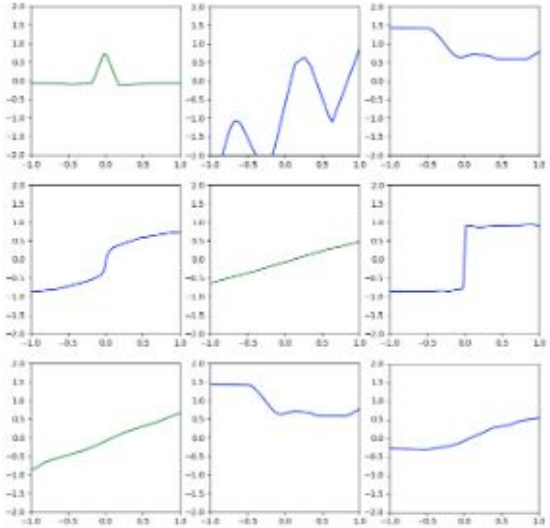
Dataset	POOLED	MAML	BOUNCEGRAD	MOMA	Structure
Parametrized sines	98.1	26.5	32.5	**19.8**	composition
Sum of functions (1-4pts)	32.7	19.7	**12.8**	18.0	sum
Sum of functions (16pts)	31.9	8.0	**0.4**	**0.4**	sum
MIT push: known objects	21.5	18.5	16.7	**14.9**	attention
MIT push: new objects	18.4	**16.9**	**17.1**	**17.0**	attention
Berkeley MoCap: known actions	35.7	35.5	**32.2**	**31.9**	concatenate
Berkeley MoCap: new actions	79.5	77.7	77.0	**73.8**	concatenate

Table 1: Summary of results; lower is better; bold results are not significantly different from best.

Learned Modules





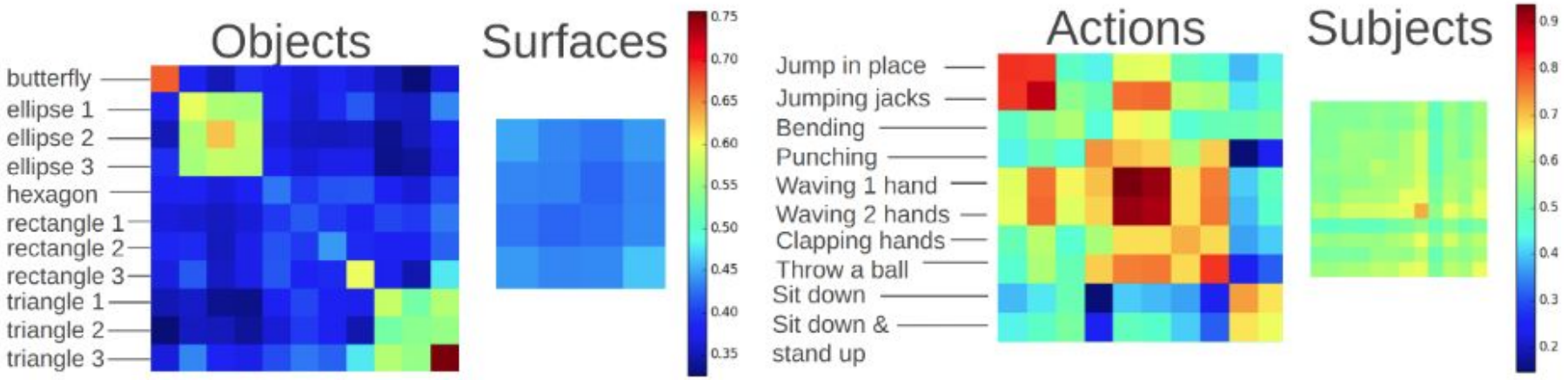


Experiments - Predict Pushing Action

- MIT push-dataset: Results for pushing 11 objects on 4 surfaces w/ manipulator hand
- **Goal**: Predict 3D change in object position + orientation
- **Results**: For previously unseen objects, MOMA is best, BOUNCE GRAD > MAML
- Otherwise, all 3 methods perform generally the same

Dataset	POOLED	MAML	BOUNCEGRAD	MOMA	Structure
MIT push: known objects	21.5	18.5	16.7	**14.9**	attention
MIT push: new objects	18.4	**16.9**	**17.1**	**17.0**	attention

Learned Modules



Experiments - Skeletal Configurations

- Robots should be able to predict/understand human motion
- Berkeley MHAD motion dataset
- **Goal:** Given 3 previous skeletal configurations, predict next
- **Results:** MOMA and BounceGRAD outperform rest, MOMA prevails in unseen case

Dataset	POOLED	MAML	BOUNCEGRAD	MOMA	Structure
Berkeley MoCap: known actions	35.7	35.5	**32.2**	**31.9**	concatenate
Berkeley MoCap: new actions	79.5	77.7	77.0	**73.8**	concatenate

Conclusion

- Proved modular compositional structure is a useful basis to solve new tasks
- Modular meta learning performs on par or better with existing meta learning methods
- Modular meta learning provides additional insight to the structure and nature of the problem. Boosts interpretability

Discussion Questions

- When humans learn new tasks, our past experiences with similar tasks and environmental semantic knowledge aids us. Does this align more with parametric or modular meta learning methods?
    - **Example:** Baseball player hitting a golf ball.
    - **In Class:** Task dependent, and probably a combination of both methods.
- What are some ways to incorporate domain-specific prior knowledge into this framework?
- Given this paper and the previous, how can you apply the modular framework to RL problems?
- Can modular meta learning or meta learning in general be applied beyond robot cognition into motion/locomotion? What would this entail?

Piazza Discussion: Combinatorial Generalization@113_f2

The concept of ”infinite use of finite means” in language comes from the idea that although our vocabulary may be finite, the combinations can lead to infinite meaningful expressions. In this paper, the means are the different modules, and we compose these modules to generalize to new tasks.

In this method for meta-learning, can we generalize these 'means' well enough to make them work for the new tasks if the tasks varied greatly from the original means? I'm sure that it will be a great basis for the transfer learning problems, but is it enough?

- How much can the test-time task vary from the training time task distribution?
- Different modules for generalization?
- **In Class**: Not sure if this approach will successfully generalize to vastly different problems. There is nothing during training to encourage the modules to be distinct! This is a question of meta-learning as a whole.

Piazza Discussion: Module Hypertuning @113_f1, @113_f6

I noticed in some of their experiments the paper trained neural nets on 16 different functions, which served as the modules that could be combined to form a meta-model. It would be interesting to see how the performance would change if there were more or less modules available or if increasing the allowed complexity of the structures would be better. But there would probably be some trade-off as the online meta-test learning step might take longer if there was a larger structure search space.

- Is there a rule of thumb/better way to define modules? Maybe for particular problem categories.

Piazza Discussion: Module Hypertuning @113_f3

It seemed that one of the major challenges that the authors faced was the avoidance of local minima, leading to the use of simulated annealing as a search algorithm and the development of BounceGrad. I'm wondering if anyone has any insight into what makes this task so unsmooth or why there are so many minima to be caught in? Is it due to the fact that some structures vastly outperform others? Wondering if anyone has any insights.

- Search space is high-dimensional. Since space of possible architectures is complex, optimization landscape is also non-convex.
- Cast this problem to graph neural networks?

Piazza Discussion: Continual Learning/Generalization @113_f5

One advantage of the modular process that the author proposed is it encourages continual learning by dividing complex robot tasks into easily manageable modules. Robot learning task is limited by the 3D world, and therefore knowledge are commonly transferable. If all the researches related to robot learning could share and integrate model in the proposed modular way, our knowledge would be advanced more efficiently.

- Approach encourages continual learning and allows for knowledge transfer across related tasks.
- One direct application is in a warehouse automation system

In Class Discussion: Comparison to Boosting

- Modular meta-learning is similar to boosting and can be thought of as an ensemble method
- Instead of linear classifiers as individual units, the individual units are neural networks (module)
- Modular meta-learning combines units with more complex structures such as attention, composition, and addition to ensemble the individual units

Thank You