Memory Storing in Meta-Learning

Chen Shangyu

April 22, 2019

Meta-Learning

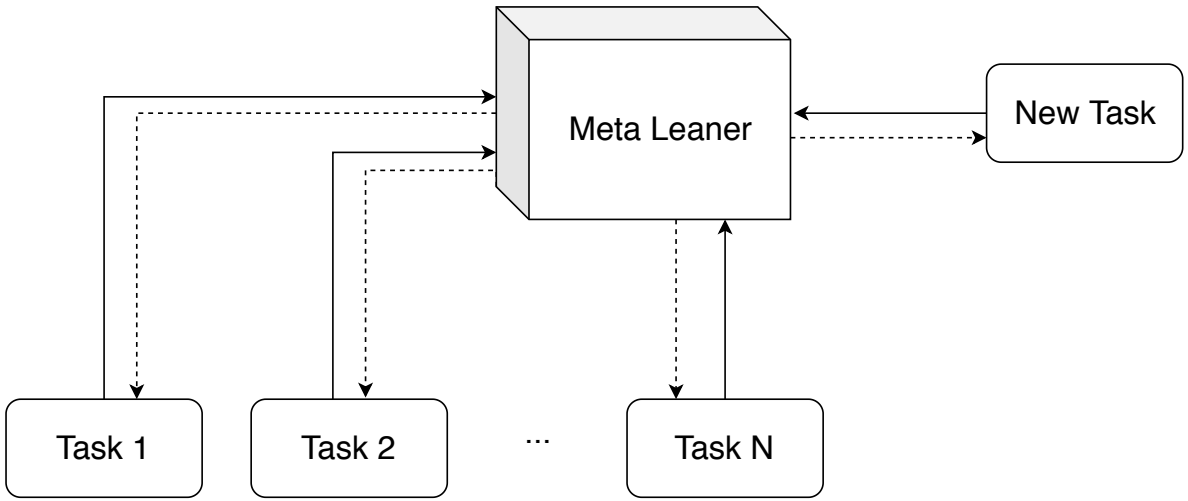


Figure: Illustration of meta learning

Meta-learning suggests framing the learning problem at two levels:

- Quick acquisition of knowledge within each separate task presented.
- Slower extraction of information learned across all the tasks.

Meta-Learning as Task Knowledge Extraction / Storing

- Knowledge stored in external.
- Knowledge stored in weights / gradients.

Meta-Learning with Memory-Augmented Neural Networks (MANN) (ICML2016)

Problem Definition

- Only a few training examples are presented one-by-one.

Motivation

- Meta learning enables rapid learning from sparse data within a task.
- This learning is guided by knowledge accrued across task.

Neural networks with a memory capacity:

- Information must be stored in memory in a representation that is both **stable** and **element-wise addressable**.
- The number of parameters should not be tied to the size of the memory.

MANN Overflow

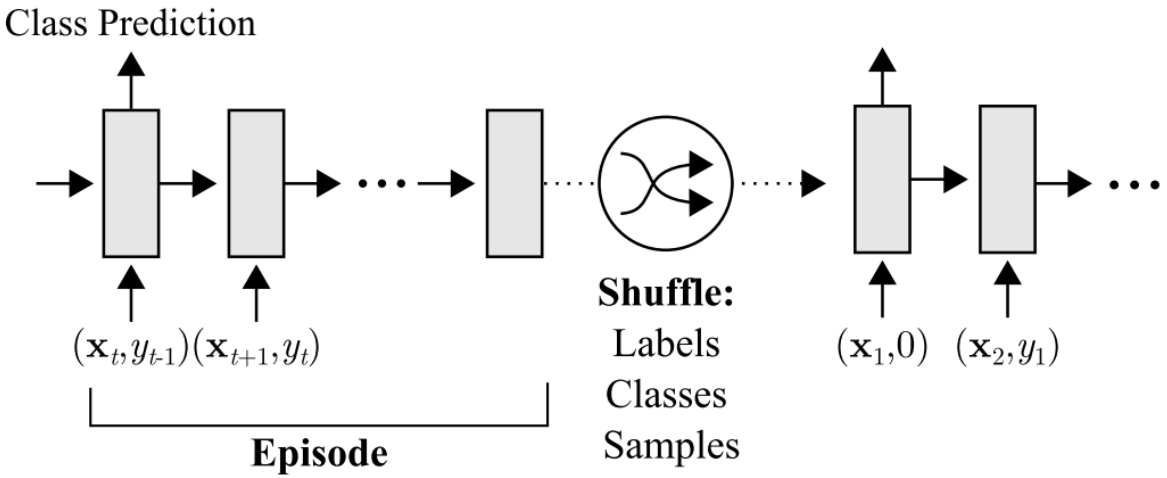


Figure: Overflow of MANN

- Episode: contains data from some datasets (Tasks)
- y_t as target is presented as input along with x_t, in a temporally offset manner.
- Controller is a neural turing machine, which generate memory as the input to a classifier, and as an additional input for the next controller state.

Neural Turing Machines (NTM)

Memory in NTM:

- A matrix $\mathbf{M}_t$.
- Read head: memory retrieval.
- Write head: memory encode and update.

Process of Memory Retrieval:

- Given some input, x_t, the controller produces a key, k_t:

$$K(k_t, \mathbf{M}_t(i)) = \frac{k_t \cdot \mathbf{M}_t(i)}{||k_t|| \times ||\mathbf{M}_t(i)||} \tag{1}$$

- Read-weight vector w_t^r:

$$w_t^r(i) = \frac{\exp(K(k_t, \mathbf{M}_t(i)))}{\sum_j \exp(K(k_t, \mathbf{M}_t(j)))} \tag{2}$$

- Memory retrieved:

$$\mathbf{r}_t = \sum_i w_t^r(i)\mathbf{M}_t(i) \tag{3}$$

Neural Turing Machines (NTM) (Cont.)

Process of Memory Update:

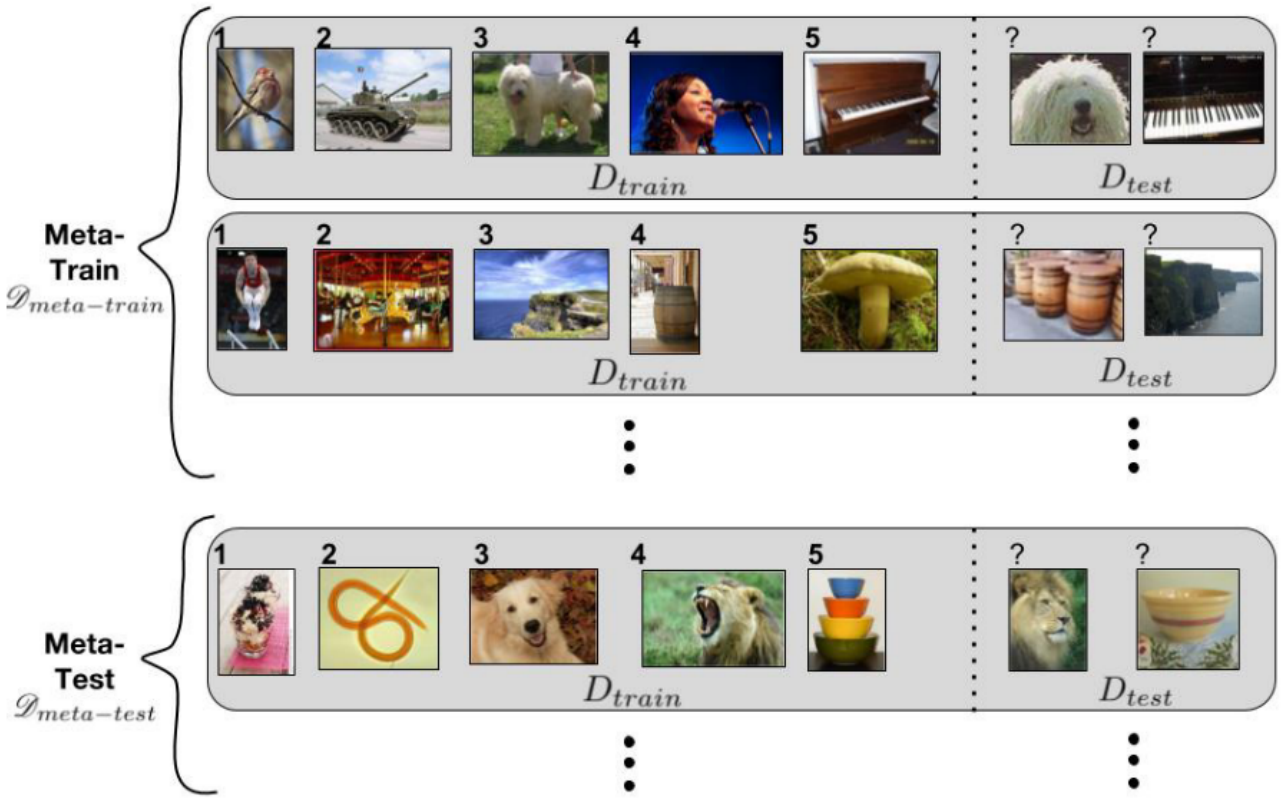
- Usage weight: $w_t^u = \gamma w_{t-1}^u + w_t^r + w_t^w$
- Least-used weight w_t^{lu}:

$$w_t^{lu}(i) = \begin{cases} 0 & \text{if} \quad w_t^u(i) > m(w_t^u, n) \\ 1 & \text{if} \quad w_t^u(i) \leq m(w_t^u, n) \end{cases} \tag{4}$$

  where n is set to equal the number of reads to memory.
- Write weights: $w_t^w = \sigma(\alpha) \times w_{t-1}^r + (1 - \sigma(\alpha)) w_{t-1}^{lu}$
- Memory update:

$$\mathbf{M}_t(i) = \mathbf{M}_{t-1}(i) + w_t^w(i) k_t \tag{5}$$

Optimization as a Model for Few-Shot Learning (ICLR2016)



$$\theta_t = f_t \odot \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} \mathcal{L}_t \tag{6}$$

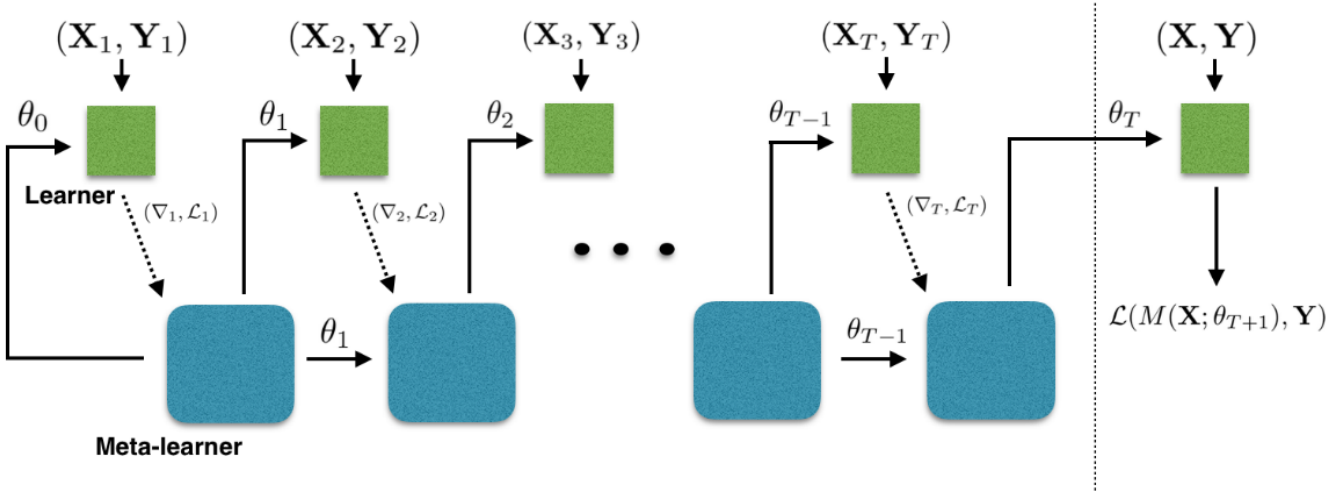
LSTM as Meta Learner

- Propose a meta-learner LSTM to learn update rule, specially, learning rate α and weight decay f_t:

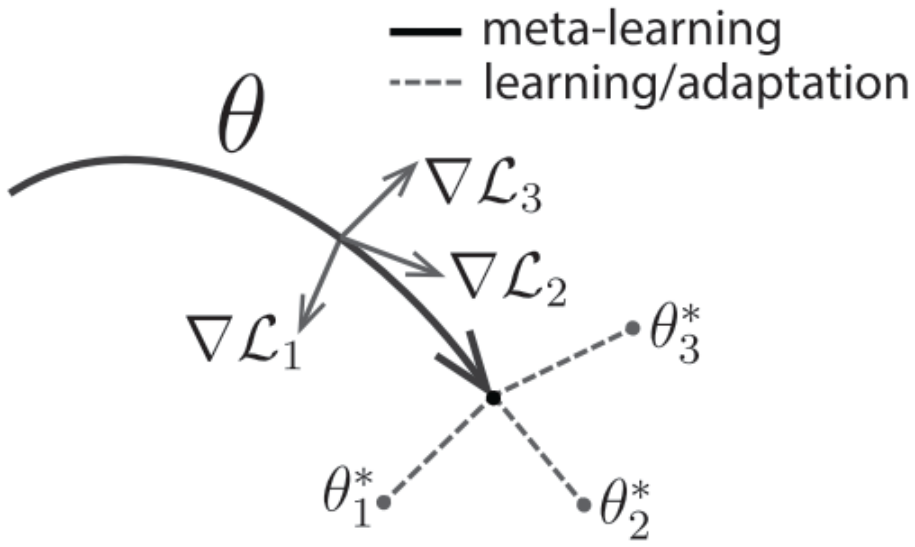
$$i_t = \sigma(\mathbf{W}_I \cdot [\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}] + \mathbf{b}_I) \quad (7)$$

$$f_t = \sigma(\mathbf{W}_F \cdot [\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}] + \mathbf{b}_F) \quad (8)$$

- Parameters in meta-learner is updated by $D_{test} \in \mathcal{D}_{meta-train}$:



Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks (ICML2017)



Learning to Optimize (ICLR2017)

Optimization algorithm (using first-order gradient only) can be recasted as:

$$\pi(f, \{x^0, ..., x^{i-1}\}) = \begin{cases} -\gamma \nabla f(x^{i-1}) \\ -\gamma \left(\sum_{j=0}^{i-1} \alpha^{i-1-j} \nabla f(x^j) \right) \end{cases} \tag{9}$$

- Learn a policy $\pi(f, \{x^0, ..., x^{i-1}\})$ by guided policy search.
- State: current location, previous gradients, improvements in the objective value
- Action: Step vector that is used to update the current location.
- Loss: objective value.

Hypernetwork (ICLR2017)

Motivation

- Learn / Generate weights ($K^j \in \mathbb{R}^{N_{in} \times N_{out} \times K \times K}$) by a hypernetwork instead of loss.
- Hypernetwork is a 2-layer neural network.

Process:

- Given a layer embedding $z^j \in \mathbb{R}^{N_z}$ (learnable)
- Layer 1: $a_i^j = W_i z^j + B_i,\ i = 1, ..., N_{in}$
- Layer 2: $K_i^j = \left\langle W_{out}, a_i^j \right\rangle + B_{out}$
- $K^j = \left(K_1^j, K_2^j, ..., K_{N_{in}}^j \right)$