Continual Learning

Meta Continual Learning / Task Free Settings

邹笑寒

2020/08

Goals

Previous	Some current works
○ Deal with catastrophic forgetting.	○ Deal with catastrophic forgetting.
○ Learn current task.	○ Exploit existing knowledge to accelerate future learning. (Achieved by Meta-Learning) ○ Get rid of task boundaries. (Task-Free)

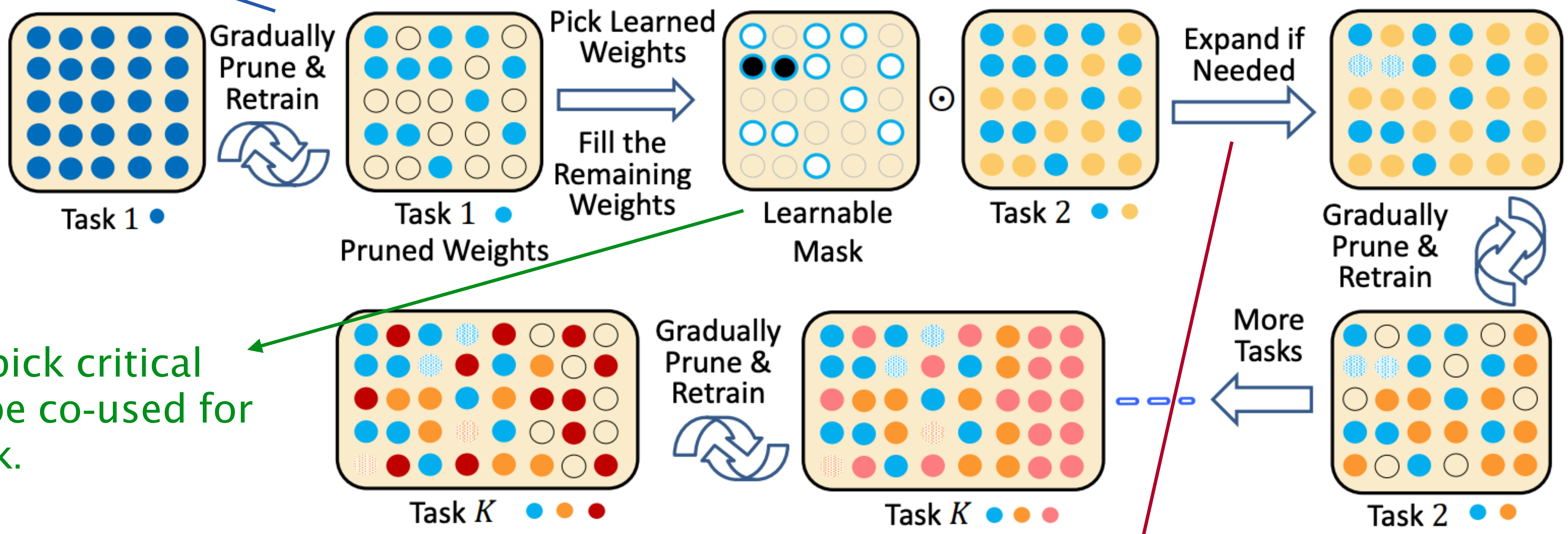
Common Approaches

- **Regularization**: Impose constraints on the update of the weights to retain knowledge.
- **Rehearsal**:
  - **Extra Memory**: Use extra memory to store data from previous tasks.
  - **Generative Replay**: Mimic past data by generative models (GAN, VAE, etc).
- **Dynamic Expansion**: Increase network capacity to handle new tasks.

Dynamic Expansion

Compacting, Picking and Growing for Unforgetting Continual Learning. NIPS 2019.

Model Compression (Gradually Prune): remove the model redundancy to the reduce the complexity



Train a **0-1 mask** to pick critical old-task weights to be co-used for learning the new task.

New weights for new tasks.

Very limited model expansion.

Task Free

Task-Free Continual Learning. CVPR 2019. (Regularization)

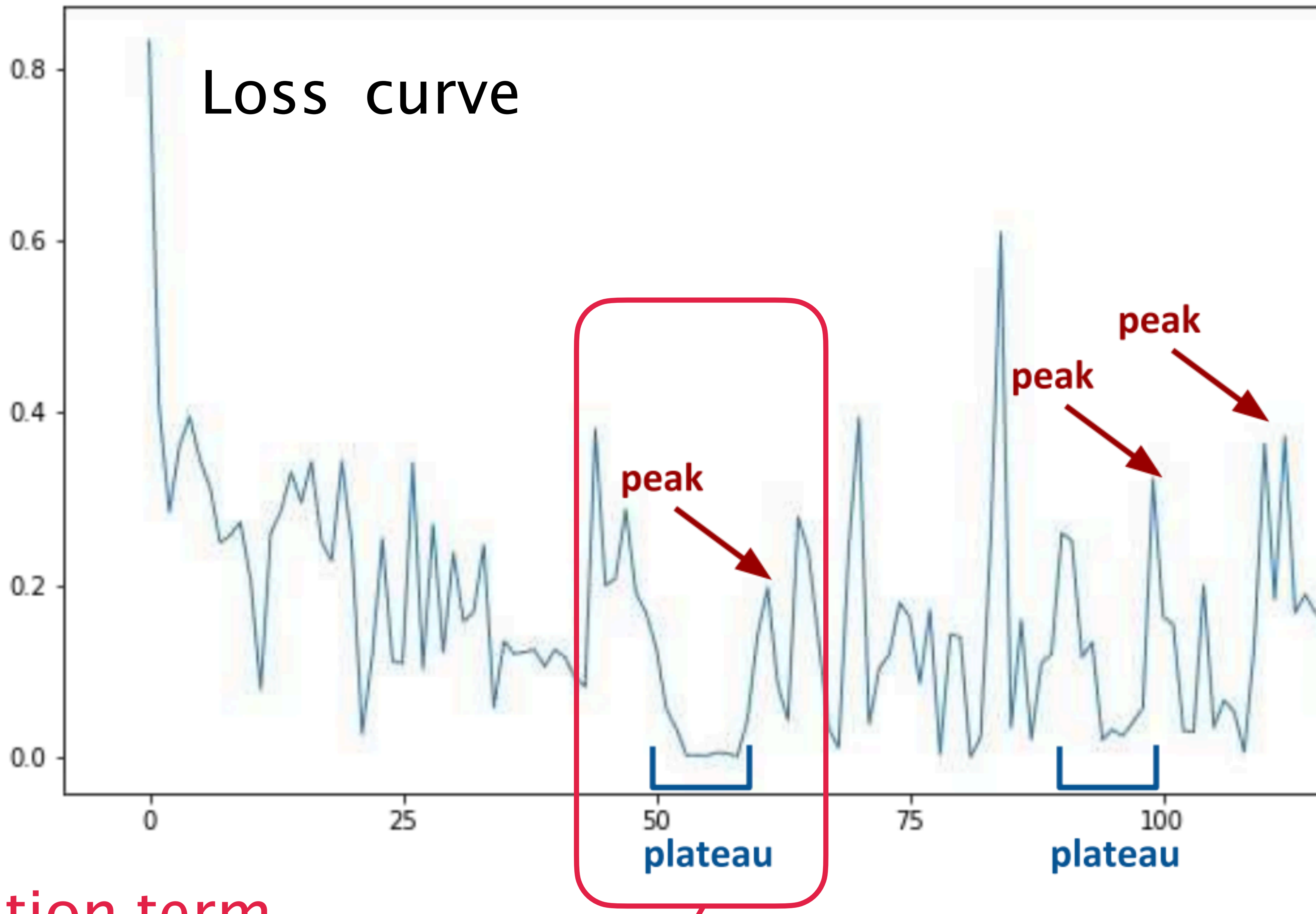
Memory Aware Synapses: Learning What (Not) to Forget. ECCV 2018 (Regularization)

$$F(x_k;\theta+\delta) - F(x_k;\theta) \approx \sum_i g_i(x_k)\delta_i \Rightarrow g_i(x_k) = \frac{\partial F(x_k)}{\partial \theta_i}$$

importance weight: $\Omega_i = \frac{1}{N}\sum_{k=1}^{N} \|g_i(x_k)\|$

regularization term

final loss: $L(\theta) = L_n(\theta) + \frac{\lambda}{2}\sum_i \Omega_i(\theta_i - \theta_i^*)^2$



a sliding window for detecting plateaus (task boundaries)

Task Free

Continual Unsupervised Representation Learning. NIPS 2019.

(Generative Replay + Dynamic Expansion)

Generative Replay via VAE:

$$\left.\begin{aligned} &y \sim \mathrm{Cat}(\pi)\text{: current task id} \\ &z \sim N(\mu_z(y), \sigma_z^2(y))\text{: task-specific latent variable} \\ &x \sim \mathrm{Bernoulli}(\mu_z(z))\text{: input data} \end{aligned}\right\} \quad \log p(x) = \sum_z p(x, y, z) \geq L$$

reconstruct data

$$\text{ELBO (maximize): } \mathcal{L} \approx \boxed{\sum_{k=1}^{K} \overbrace{q(\mathbf{y}=k \mid \mathbf{x})}^{\text{component posterior}} \Big[\overbrace{\log p(\mathbf{x} \mid \widetilde{\mathbf{z}}^{(k)})}^{\text{component-wise reconstruction loss}} \overbrace{- \mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}, \mathbf{y}=k) \,||\, p(\mathbf{z} \mid \mathbf{y}=k))}^{\text{component-wise regulariser}} \Big]}$$

$$\boxed{- \underbrace{\mathrm{KL}(q(\mathbf{y} \mid \mathbf{x}) \,||\, p(\mathbf{y}))}_{\text{Categorical regulariser}}}$$

perform task clustering

[CURL]

Task Free

Continual Unsupervised Representation Learning. NIPS 2019.

(Generative Replay + Dynamic Expansion)

Dynamic expansion:

$$
\begin{aligned}
&\text{if ELBO}(x) < c_{new} \text{ (a threshold) :}\\
&\qquad x \rightarrow D_{new} \text{ (a new task)}\\
&\qquad \text{if } N(D_{new}) \geq N_{new} \text{ :}\\
&\qquad\qquad \theta^{(K+1)} = \theta^{(k^*)} \quad (k^* = \arg\max_{k \in \{1,2,\ldots,K\}} \sum_{x \in D_{new}} q(y = k \mid x))
\end{aligned}
$$

[CURL]

Task Free

Task Agnostic Continual Learning Using Online Variational Bayes. arXiv 2019.

(Regularization)

Bayes' rule:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

incremental Bayes' rule:

$$p(\theta|D_n) = \frac{p(D_n|\theta)p(\theta|D_{n-1})}{p(D_n)}$$

Doesn't care about task boundaries.

Algorithm 1 Bayesian Gradient Descent (BGD)

Initialize μ, σ, η, K

Repeat learning rate

$$\mu_i \leftarrow \mu_i - \eta\sigma_i^2 \mathbb{E}_\varepsilon\left[\frac{\partial \mathrm{L}_n(\boldsymbol{\theta})}{\partial \theta_i}\right]$$

$$\sigma_i \leftarrow \sigma_i \sqrt{1 + \left(\frac{1}{2}\sigma_i \mathbb{E}_\varepsilon\left[\frac{\partial \mathrm{L}_n(\boldsymbol{\theta})}{\partial \theta_i}\varepsilon_i\right]\right)^2} - \frac{1}{2}\sigma_i^2 \mathbb{E}_\varepsilon\left[\frac{\partial \mathrm{L}_n(\boldsymbol{\theta})}{\partial \theta_i}\varepsilon_i\right]$$

Until convergence criterion is met.

Unimportant weights have a large uncertainty σ_i, which means a large learning rate.

The expectations are estimated using Monte Carlo method, with $\theta_i^{(k)} = \mu_i + \varepsilon_i^{(k)}\sigma_i$:

$$\mathbb{E}_\varepsilon\left[\frac{\partial \mathrm{L}_n(\boldsymbol{\theta})}{\partial \theta_i}\right] \approx \frac{1}{K}\sum_{k=1}^{K} \frac{\partial \mathrm{L}_n\left(\boldsymbol{\theta}^{(k)}\right)}{\partial \theta_i}$$

$$\mathbb{E}_\varepsilon\left[\frac{\partial \mathrm{L}_n(\boldsymbol{\theta})}{\partial \theta_i}\varepsilon_i\right] \approx \frac{1}{K}\sum_{k=1}^{K} \frac{\partial \mathrm{L}_n\left(\boldsymbol{\theta}^{(k)}\right)}{\partial \theta_i}\varepsilon_i^{(k)}$$

[BGD]

Meta Continual Learning

Meta Continual Learning. arXiv 2018.

Meta-Learning: train a neural network h_ϕ (MLP) to be optimizer, which can predict update steps using existing knowledge instead of based on current gradients only.

Algorithm 1 Meta continual learning for training h_ϕ

procedure META-CONTINUAL-LEARNING($f_\theta, h_\phi, \mathcal{T}_0$)
  for all $\mathcal{T}_{0,j>1}$ in $\mathcal{T}_0$ **do**
    for Epochs 1,2,3, ... **do**
      $\theta^*_{0,j-1} \leftarrow$ train f_θ for one epoch on $\mathcal{T}_{0,j-1}$ using `Adam`
      $\theta \leftarrow \theta^*_{0,j-1}$
      $\boxed{g_{j-1} \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_{0,j-1}))}$
      for Epochs 1,2,3, ... **do**
        $g \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_{0,j}))$
        $\theta \leftarrow \theta - \eta h_\phi(g_{j-1}, g, \theta^*_{0,j-1}, \mathcal{I})$
        $\boxed{\phi \leftarrow \texttt{Adam}\,(\nabla_\phi \mathcal{L}(f_\theta(\mathcal{T}_{0,j-1} \cup \mathcal{T}_{0,j})))}$
      end for
    end for
  end for
end procedure

T_0: an independent meta-training dataset, containing subtasks similar to the continual learning tasks

because there is nothing to preserve

g_{j-1}: average squared gradients of task $j-1$.

the optimizer h_ϕ's optimizer is Adam

leverage info from both the current and previous tasks to prevent forgetting

Meta Continual Learning

Meta Continual Learning. arXiv 2018.

Continual Learning: use the trained optimizer h_ϕ

Algorithm 2 Continual learning using the trained h_{ϕ^*}

procedure CONTINUAL-LEARNING($f_\theta, h_{\phi^*}, \{\mathcal{T}_1, \mathcal{T}_2, \cdots\}$)
    for all $\mathcal{T}_i$ **do**
        if $i = 1$ **then**
            $\theta_1^* \leftarrow$ Train f_{θ_1} on $\mathcal{T}_1$ using `Adam` because there is nothing to preserve
        else
            $\theta \leftarrow \theta_{i-1}^*$
            $g_{i-1} \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_{i-1}))$
            for Epochs 1,2,3, ... **do**
                $g \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_i))$
                $\theta \leftarrow \theta - \eta h_{\phi^*}(\underline{g_{i-1}, g}, \theta_{i-1}^*, \mathcal{I})$
            end for
            $\theta_i^* \leftarrow \theta$
        end if leverage info from both the current and previous tasks to prevent forgetting
    end for
end procedure

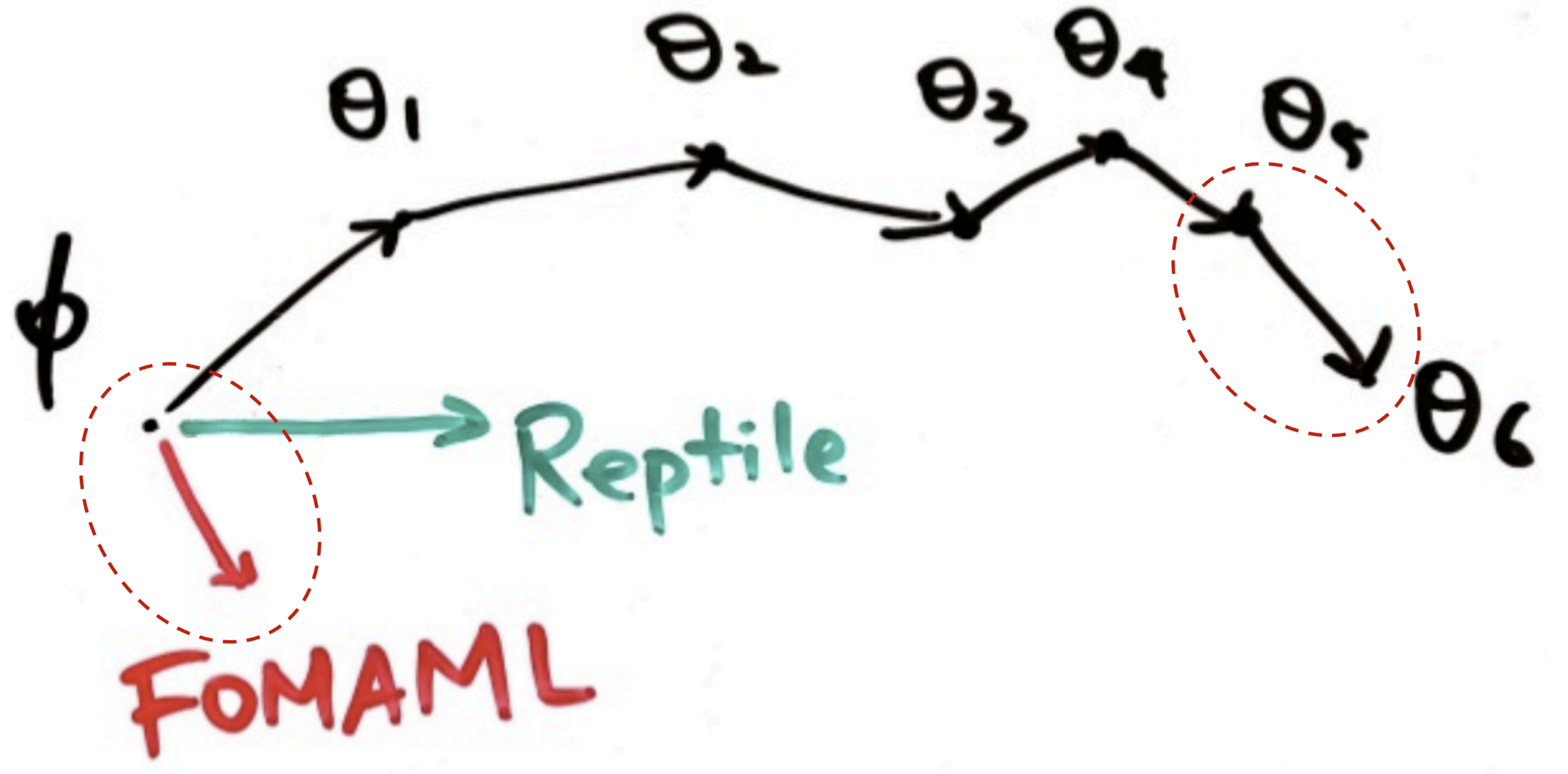
Meta Continual Learning

Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference. ICLR 2019. (Reptile + Experience Replay)

Reptile (**On First-Order Meta-Learning Algorithms. arXiv 2018.**)

Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for



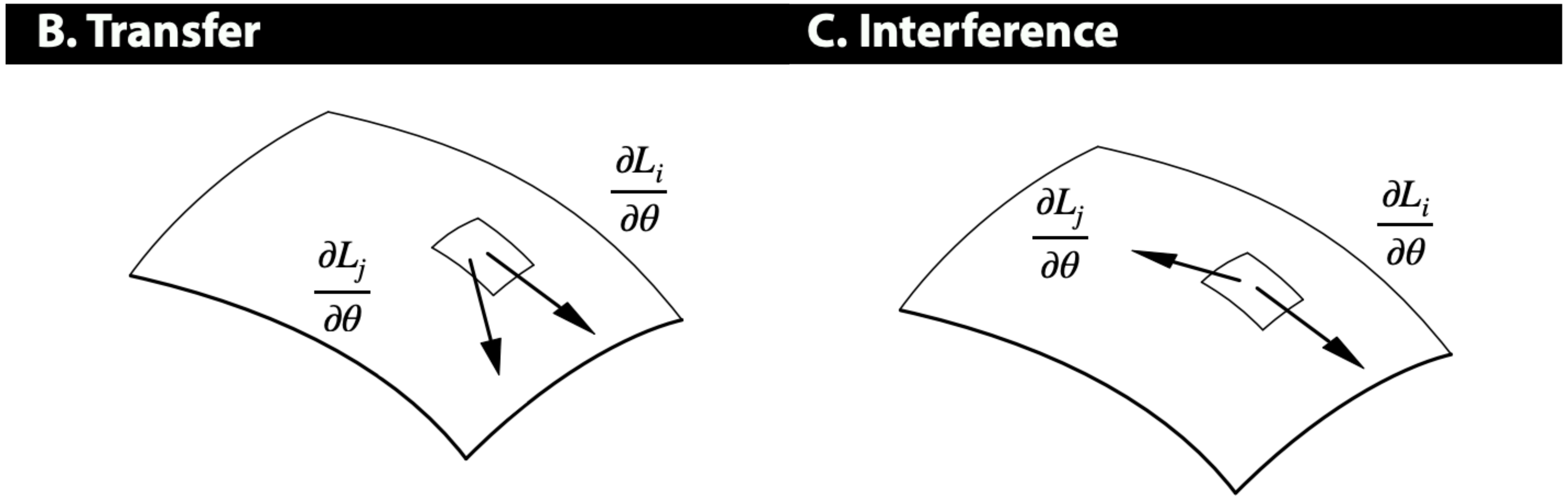
[MER]

Meta Continual Learning

Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference. ICLR 2019. (Reptile + Experience Replay)

Objective of Reptile:

$$\theta = \arg\min_{\theta} \mathbb{E}_{B_1,\ldots,B_s \sim D_t}\left[2\sum_{i=1}^{s}\left[L(B_i) - \sum_{j=1}^{i-1}\alpha \boxed{\frac{\partial L(B_i)}{\partial \theta}\cdot\frac{\partial L(B_j)}{\partial \theta}}\right]\right]$$



large inner product

small inner product

Maximize the inner product of gradients of two different mini batches for the same task.

[MER]

Meta Continual Learning

Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference. ICLR 2019. (Reptile + Experience Replay)

Reptile + Experience Replay:

$$\theta = \arg\min_{\theta} \mathbb{E}_{(x_{11},y_{11}),\ldots,(x_{sk},y_{sk})\sim \boxed{M}} \left[2\sum_{i=1}^{s} \sum_{j=1}^{k} \left[L(x_{ij}, y_{ij}) - \sum_{q=1}^{i-1} \sum_{r=1}^{j-1} \alpha \frac{\partial L(x_{ij}, y_{ij})}{\partial \theta} \cdot \frac{\partial L(x_{qr}, y_{qr})}{\partial \theta} \right] \right]$$

current example + past examples

Summary

- **Reptile:** accelerate future learning
- **Experience Replay:** prevent forgetting

[MER]

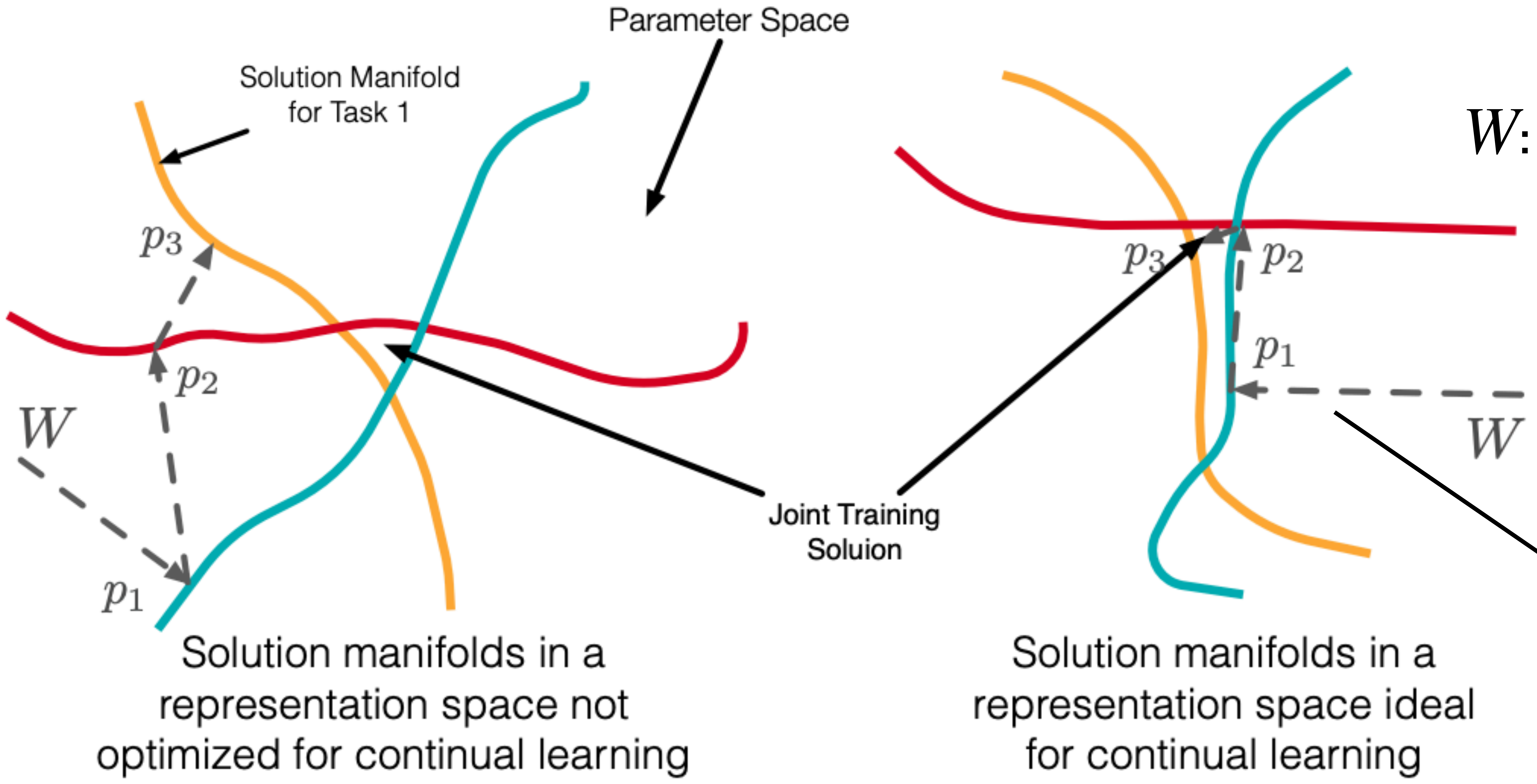
Meta Continual Learning

Meta-Learning Representations for Continual Learning. NIPS 2019.

(based on MAML)

× MAML: Find the best initial parameters

√ MRCL: Find the best representation (encoder)



W: Weights of the task-specific decoder

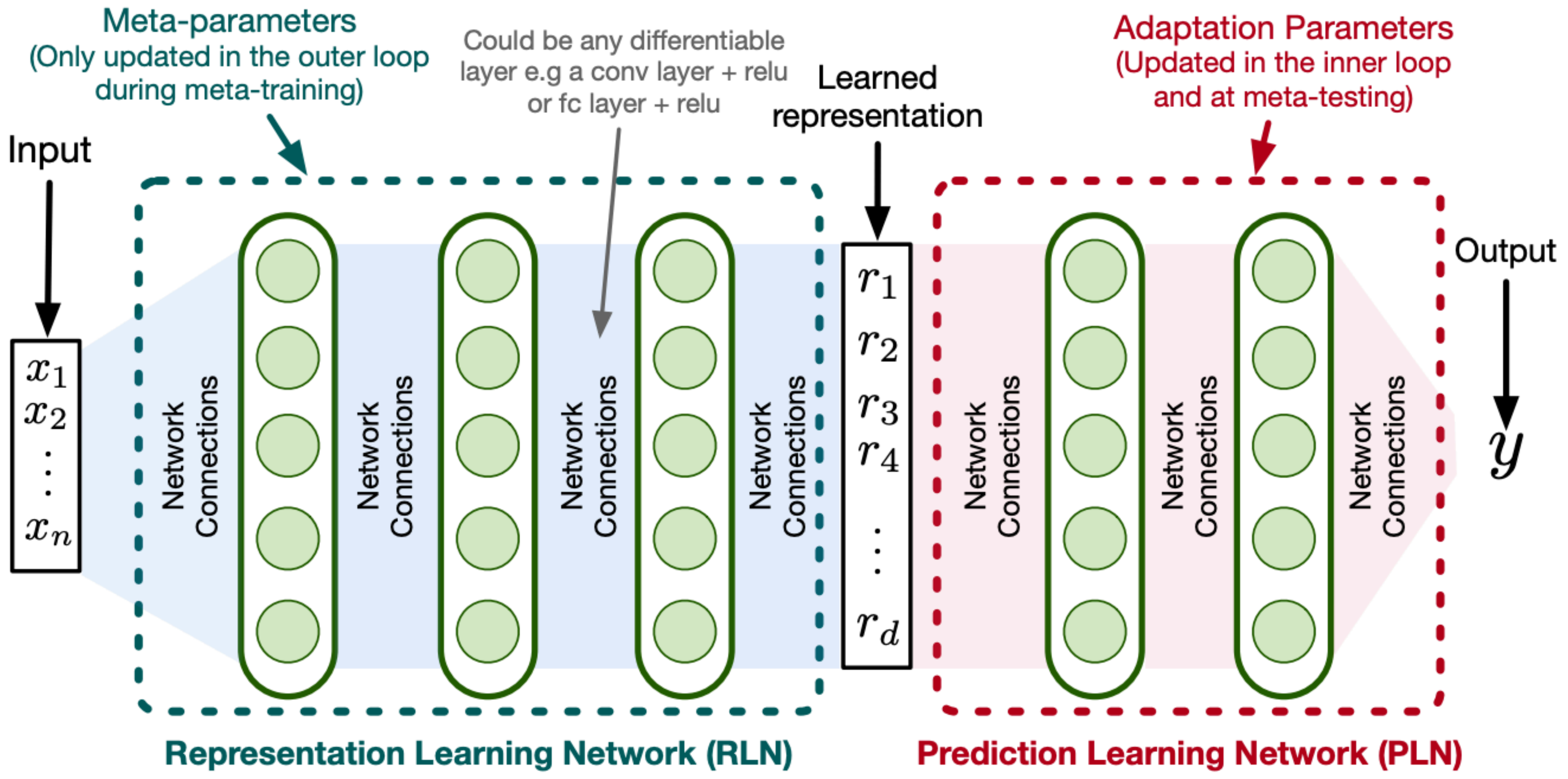
An optimal trajectory during fine-tuning (only decoder)

[MRCL]

Meta Continual Learning

Meta-Learning Representations for Continual Learning. NIPS 2019.

(based on MAML)



Algorithm 2: Meta-Training : OML

Require: $p(\mathcal{T})$: distribution over CLP problems
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  randomly initialize W
  Sample CLP problem $\mathcal{T}_i \sim p(\mathcal{T})$
  Sample $\mathcal{S}_{train}$ from $p(\mathcal{S}_k|\mathcal{T}_i)$
  $W_0 = W$ inner loop: update decoder on S_{train}
  for $j = 1, 2, \ldots, k$ **do**
    $(X_j, Y_j) = \mathcal{S}_{train}[j]$
    $W_j = W_{j-1} - \alpha \nabla_{W_{j-1}} \ell_i(f_{\theta, W_{j-1}}(X_j), Y_j)$
  end for
  Sample S_{test} from $p(\mathcal{S}_k|\mathcal{T}_i)$
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \ell_i(f_{\theta, W_k}(S_{test}[:, 0]), S_{test}[:, 1])$
end while   outer loop: update encoder on S_{test}

encoder (shared)   decoder (task-specific)

$$f_{W,\theta} = g_W(\phi_\theta(X))$$

[MRCL]

Task-Free Meta Continual Learning

Task Agnostic Continual Learning via Meta Learning. ICML 2020 LifelongML Workshop.

(Reptile + Regularization)

incremental Bayes' rule: $p(\theta \mid D_{0:t}) = \dfrac{p(D_t \mid \theta, D_{0:t-1})p(\theta \mid D_{0:t-1})}{p(D_t \mid D_{0:t-1})}$

objective:
$$q_t(\theta) = \arg\min_{q(\theta)} \mathrm{KL}(q_t(\theta) \| p(\theta \mid D_{0:t}))$$
$$= \arg\min_{q(\theta)} \mathbb{E}_{q(\theta)}[\log \boxed{p(D_t \mid \theta, D_{0:t-1})}] - \mathrm{KL}(q_t(\theta) \| q_{t-1}(\theta))$$

Reptile model

$$p(D_t \mid \theta, D_{0:t-1}) \approx p(D_t \mid \theta, \boxed{D_t^{ctx}}) = p(y_t \mid \boxed{f_\theta(x_t)})$$

$D_t^{ctx} = \{(x_{t-k}, y_{t-k}), \ldots, (x_{t-1}, y_{t-1})\}$ (a sliding window with a fixed length k)

Doesn't care about task boundaries.

[What & How]

Summary

online changepoint detection?

Gaussian Process Change Point Models. ICML 2010.

Bayesian Online Changepoint Detection. 2007.

Method	Task Free?	Meta-Learning?	Details
Task Free Continual Learning (CVPR 2019)	√		○ detect task boundaries by detecting plateaus on loss surface using a sliding window (?) ○ MAS (a regularization-based method)
CURL (NIPS 2019)	√		○ generative replay (VAE) ○ cluster samples into different tasks (dynamic expansion)
BGD (arXiv 2019)	√		○ online variational Bayes ○ without detecting task boundaries (?)
Meta Continual Learning (arXiv 2018)		√	○ train a neural network to be optimizer ○ use info of the previous task to prevent forgetting
MER (ICLR 2019)		√	○ Reptile + experience replay
MRCL (NIPS 2019)		√	○ MAML ○ train an encoder instead of a parameter initialization
What & How (ICLR 2020 Workshop)	√	√	○ Reptile + online variational Bayes ○ without detecting task changes (?)