IS

Wednesday, 19 November 2025 11:28 AM

Bilevel optimization in MAML

meta_learner → θ (meta-parameters)

fast_learner → copies θ & adapts on each task

$$\theta' = \theta - \alpha \frac{dL_{train}}{d\theta}$$

→ meta-steps – how many times algo is run → 2000 in our case
↳ outer loop

→ Sample tasks from distribution $p(\tau)$

- → compute val loss before adaptation
- → k inner loop steps
  ↳ produces adapted θ'

each task has
training data – D_τ^{train}
Validation data – D_τ^{val}

For each method :

inner loop (Task Adaptation)

$$\theta'_\tau = \theta - \alpha \nabla_\theta L^{T}_{train}(\theta)$$

↳ do this for each task

k steps

$$\theta^{(0)} = \theta$$
$$\theta^{(1)} = \theta^{(0)} - \alpha \nabla_\theta L_{train}(\theta^{(0)})$$
$$\theta^{(2)} = \theta^{(1)} - \alpha \nabla_\theta L_{train}(\theta^{(1)})$$
$$\vdots$$
$$\theta^{(k)} = \theta^{(k)} - \alpha \nabla L_{train}(\theta^{(k-1)})$$

parameters update

Start with θ , predict output
compute loss wrt this loop's parameter θ
backward pass (to find gradient)
gradient descent update.

final $\theta^{(k)}$ → task adapted parameters

$$\hat{y}_{val} = f_{\theta_\tau^{(k)}}(x^{val})$$

val loss after adaptation :

$L_{val}(\theta_\tau^{(k)})$ → how good meta initialization θ was at letting us adapt to task

Now in explicit maml:

for meta-gradient step :

$$\nabla_\theta L_{val}(\theta_\tau^{(k)})$$

since θ_τ^k depends on θ through k inner steps:

$$\theta_\tau^{(k)} = \theta - \alpha \nabla_\theta L_{train}(\theta) - \alpha \nabla_{\theta^{(1)}} L_{train}(\theta^{(1)}) \dots$$

$\Rightarrow \frac{\partial L_{val}}{\partial \theta}$ chain rule through all k steps

This introduces Hessian terms:

$$\nabla_\theta L_{val}(\theta_\tau^{(k)}) = \frac{\partial L_{val}}{\partial \theta_\tau^{(k)}} \cdot \frac{\partial \theta_\tau^{(k)}}{\partial \theta}$$

⇒ Repeat this process for each task and average the meta-gradients:

$$g_{meta} = \frac{1}{B} \sum_\tau \nabla_\theta L_{val}(\theta_\tau^{(k)})$$

Then update meta parameters :

$$\theta \leftarrow \theta - \beta g_{meta}$$

↳ meta learning rate

One meta-iteration
our code runs for 2000 meta-steps

pre-update loss – loss before inner loop gradient step
post update loss – loss after inner loop k steps
} train

val loss for each outer loop. ↳ how well model fits task after adaptation averaged over all tasks

FOMAML :

full maml meta-gradient :

$$\nabla_\theta L_{val}(\theta') = \nabla_{\theta'} L_{val}(\theta') \cdot [I - \alpha \nabla_\theta^2 L_{train}\, \theta)$$

FOMAML : drops the Hessians :

$$\nabla_\theta L_{val}(\theta') \approx \nabla_{\theta'} L_{val}(\theta')$$

Implicit MAML :

solves a minimization problem
we treat θ' → adapted parameters as solution to an optimization problem θ^*
so gradient $= 0$

$$\theta_\tau^* = \arg\min_{\theta'} \left[L^\tau_{train}(\theta') + \frac{\lambda}{2} \|\theta' - \theta\|^2 \right]$$

so now :

$$\nabla_{\theta'} L_{train}(\theta_\tau^*) + \lambda(\theta_\tau^* - \theta) = 0$$

differentiating wrt θ :

$$\nabla^2_{\theta'} L_{train}(\theta_\tau^*) \cdot \frac{\partial \theta_\tau^*}{\partial \theta} + \lambda \left(\frac{\partial \theta_\tau^*}{\partial \theta} - I \right) = 0$$

This gives a linear system:

$$(\nabla^2_{\theta'} L_{train} + \lambda I) \cdot \frac{\partial \theta_\tau^*}{\partial \theta} = \lambda I$$

$$\frac{\partial \theta_\tau^*}{\partial \theta} = (H_{train} + \lambda I)^{-1} \lambda I$$

plug this jacobian into our gradient :

$$\nabla_\theta L^\tau_{val}(\theta_\tau^*) = \lambda \underbrace{(H + \lambda I)^{-1} \nabla_{\theta'} L^\tau_{val}(\theta_\tau^*)}$$

Computed by cg solver and Hessian vector product

Reptile :

1. Starts with θ
2. K inner loops
   → get adapted parameter ϕ
3. outer update

$$\theta \leftarrow \theta + \epsilon(\phi - \theta)$$

Reptile is very close to FOMAML

Neumann :

$$\nabla_\theta L_{val}(\phi) = \nabla_\phi L_{val}(\phi) \frac{\partial \phi}{\partial \theta}$$

$$\frac{\partial \phi}{\partial \theta} = (I - \alpha H)^{-1}$$

↳ Hessian of training loss

$$(I - A)^{-1} \approx I + A + A^2 \dots$$

$$\Rightarrow (I - \alpha H)^{-1} \approx I + \alpha H + \alpha^2 H^2 \dots$$

$$\Rightarrow \nabla_\theta L_{val}(\phi) = g_{val} + \alpha H g_{val} + \alpha^2 H^2 g_{val} + \dots$$

↳ uses Hessian vector product

Penalty method:

bilevel approximation where meta objective is :

$$L_{meta}(\theta) = L_{val}(\phi) + \lambda \|\nabla_\phi L_{train}(\phi)\|^2$$

if inner loop converged perfectly , then :

$$\nabla_\phi L_{train}(\phi^*) = 0$$

uses Hessians vector product