Meta-Learning that Memorizes (M&M)

Alex Boulton McKeehan

MAML

Goal: Learn optimal initialization parameters for any model.

Input: Batch of tasks (datasets) split into train and test.

Outer loop update: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$

(backprop meta-parameters over total training loss over all tasks)

Inner loop update: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

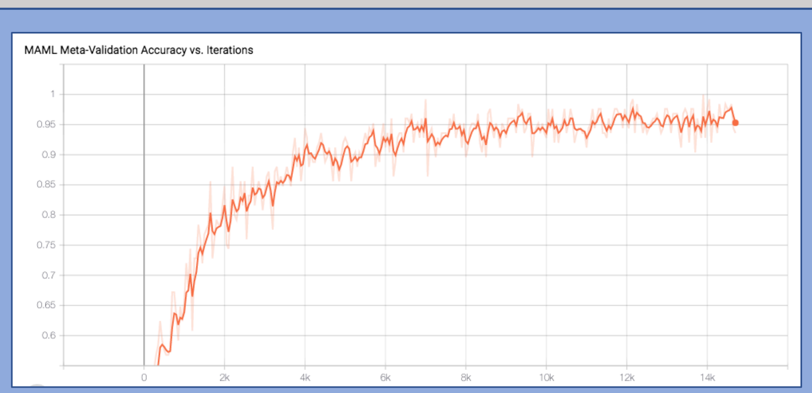
(standard stochastic gradient descent)

Limitations

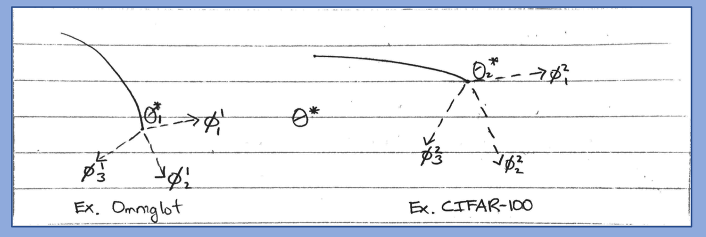
1. Training performance highly stochastic.
- Limits threshold of maximum attainable performance.
- Ex: highest possible meta-accuracy ~97% in (1).

2. Catastrophic forgetting.
- The optimum in meta-parameter space over all task families may be far from all optima.
- Learning one task family (ex. animal classification) after another (ex. Facial recognition) erodes performance on the former.

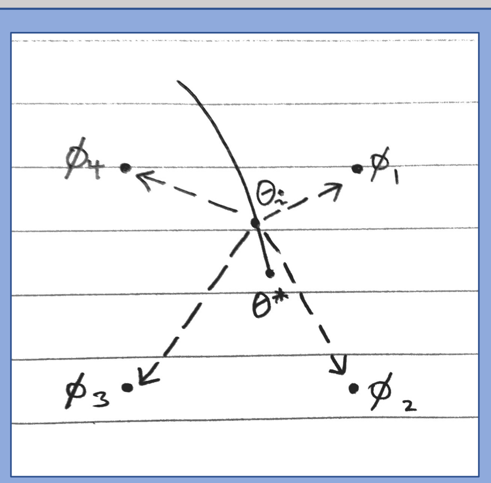
3. Meta-Gradient thrashing.
- Meta-gradient updates may conflict and yield updates in opposite directions.



(1) Meta-validation accuracy of MAML trained on Omniglot with meta-lr of 0.4. Notice the high stochasticity, even at the end of training.



(2) Example of catastrophic forgetting for two task families, where reaching either meta-optimum will induce poor performance on tasks from the other task family.

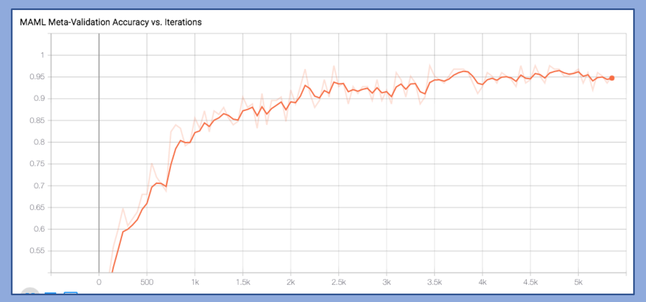


(3) Example of gradient thrashing, where gradient updates lead the meta-gradient in orthogonal directions through meta-parameter space.

Attempt 1: Learn Inner-Loop Learning Rate

Implementation
- Learn inner-loop learning rate per weight layer and inner loop update iteration.

Advantages
- Enables meta-gradient to converge efficiently.
  - Don't overshoot; speed up to reduce loss most efficiently.

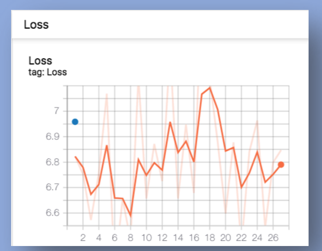


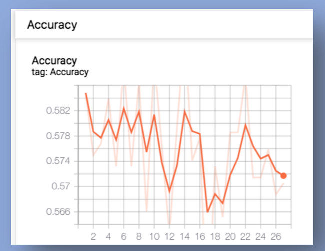
Performance on Omniglot with inner-loop learning rate initialized to 0.04. Compare to earlier graph in terms of convergence and final performance.

Attempt 2: Hessian regularization

Implementation
- Encourage meta-gradient updates to take a similar trajectory by encouraging similarity in second-order updates.

Unclear results; need more compute time.





Loss and accuracy after running on Omniglot for a small number of epochs. The failure of the loss to decrease suggests the regularization term coefficient (0.01) is too high.

$$\theta \leftarrow \theta - \nabla \mathcal{L}(\theta - \nabla \mathcal{L}(\theta, \mathcal{D}_i^{\text{train}}), \mathcal{D}_i^{\text{test}})$$

$$\mathcal{L}_H = \sum_i ||\mathbf{H}[\mathcal{L}(\theta', \mathcal{D}_i^{\text{train}})]||_F^2$$

Loss and accuracy after running on Omniglot for a small number of epochs. The failure of the loss to decrease suggests the regularization term coefficient (0.01) is too high.

Attempt 3: Meta-PCGrad

Implementation
- Group meta-batches into sets of meta-batch tasks.
- Modify original PCGrad algorithm to project meta-gradients instead of standard gradients.

Algorithm 1 PCGrad Update Rule

Require: Current model parameters θ
Sample mini-batch of tasks $\mathcal{B} = \{\mathcal{T}_k\} \sim p(\mathcal{T})$
for $\mathcal{T}_i \sim \mathcal{B}$ in sequence **do**
  Compute gradient $\mathbf{g}_i$ of $\mathcal{T}_i$ as $\mathbf{g}_i = \nabla_\theta \mathcal{L}_i(f_\theta)$
  for $\mathcal{T}_j \overset{\text{uniformly}}{\sim} \mathcal{B}$ in random order **do**
    Compute gradient $\mathbf{g}_j$ of task $\mathcal{T}_j$ as $\mathbf{g}_j = \nabla_\theta \mathcal{L}_j(f_\theta)$
    Compute cosine similarity between $\mathbf{g}_i$ as $\mathbf{g}_j$ as $\cos(\phi_{ij}) = \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\|\|\mathbf{g}_j\|}$.
    if $\cos(\phi_{ij}) < 0$ **then**
      Set $\mathbf{g}_i = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \mathbf{g}_j$ *// Subtract the projection of $\mathbf{g}_i$ onto $\mathbf{g}_j$*
    end if
  end for
  Store $\mathbf{g}_i^{\text{proj}} = \mathbf{g}_i$
end for
return update $\Delta\theta = \sum_i \mathbf{g}_i^{\text{proj}}$

Conclusion

- Meta-learning fails to reach top accuracies due to gradient thrashing.
- These effects can be ameliorated by algorithmic fine-tuning.
- Optimal meta parameters still cannot generalize to different task families.

  Next Steps: Generative model, MADAM.

References

[1] Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. International Conference on Machine Learning (ICML), 2017.
[2] Parisotto, E., Ba, J. L., and Salakhutdinov, R. Deep multitask and transfer reinforcement learning. International Conference on Learning Representa- tions (ICLR), 2016.
[3] α (double-blind review) Gradient Surgery for Multi-Task Learning (2019). [4] β (double-blind review) Mint: Matrix-Interleaving for Multi-Task Learning (2019).
[4] Rosenbaum, C., Klinger, T., and Riemer, M. Routing Networks: Adaptive Selection of Non-linear Functions for Multi-Task Learning. International Conference on Learning Representations (ICLR), 2018.
[5]

Yu, Tianhe, et al. "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning." *arXiv preprint arXiv:1910.10897* (2019).