Interleaving

1 Introduction

From my understanding, the goal of interleaving learning is to find an architecture that perform well on both two datasets, such as CIFAR-10 and CIFAR-100. After training, the weights for two tasks of the architecture have not to be the same. During the training iteration, the purpose of initializing architecture weights with the other one's weights, is to accelerate the convergence of the model.

The optimization problem is as follows, where A denotes continuous representation of neural architecture which is same as DARTS for two tasks; W denotes the weights of the architecture; H denotes the weights of different parts of two tasks such as the final output layer for CIFAR-10 and CIFAR-100; D denotes the datasets.

$$
\begin{aligned}
\min_{A} \quad & L(A, \widetilde{W}_1^{(3)}(A), H_1^*; D_1^{(val)}) + L(A, \widetilde{W}_2^{(4)}(A), H_2^*; D_2^{(val)}) \\
\text{s.t.} \quad & \widetilde{W}_2^{(4)}(A), H_2^* = \min_{W_2, H_2} L(A, W_2, H_2; D_2^{(tr)}) + \lambda \|W_2 - \widetilde{W}_1^{(3)}(A)\|_F^2 \\
& \widetilde{W}_1^{(3)}(A), H_1^* = \min_{W_1, H_1} L(A, W_1, H_1; D_1^{(tr)}) + \lambda \|W_1 - \widetilde{W}_2^{(2)}(A)\|_F^2 \\
& \widetilde{W}_2^{(2)}(A) = \min_{W, H_2} L(A, W_2, H_2; D_2^{(tr)}) + \lambda \|W_2 - \widetilde{W}_1^{(1)}(A)\|_F^2 \\
& \widetilde{W}_1^{(1)}(A) = \min_{W_1, H_1} L(A, W_1, H_1; D_1^{(tr)})
\end{aligned}
\tag{1}
$$

Consider the approximate architecture gradient used in DARTS, we have:

$$
\begin{aligned}
& \nabla_A L(A, W_1, H_1; D_1^{(val)}) \\
\approx\ & \nabla_A L(A, W_1 - \xi \nabla_{W_1} \left[L(A, W_1, H_1; D_1^{(tr)}) + \lambda \|W_1 - \widetilde{W}_2'\|_F^2 \right], H_1 - \xi \nabla_{H_1} L(A, W_1, H_1; D_1^{(tr)}); D_1^{(val)}) \\
=\ & \nabla_A L(A, W_1', H_1'; D_1^{(val)}) \\
& - \xi \nabla_{A, W_1}^2 L(A, W_1, H_1; D_1^{(tr)}) \cdot \nabla_{W_1'} L(A, W_1', H_1'; D_1^{(val)}) \\
& - \xi \nabla_{A, H_1}^2 L(A, W_1, H_1; D_1^{(tr)}) \cdot \nabla_{H_1'} L(A, W_1', H_1'; D_1^{(val)})
\end{aligned}
\tag{2}
$$

where $W_1' = W_1 - \xi\nabla_{W_1}\left[L(A, W_1, H_1; D_1^{(tr)}) + \lambda\|W_1 - \widetilde{W}_2'\|_F^2\right]$; $H_1' = H_1 - \xi\nabla_{H_1}L(A, W, H_1; D_1^{(tr)})$. If there is no W_2', which means the training process just beginning, then set $W_2' = \mathbf{O}$. For simplicity, the terms ξ here are same, but actually they can be different.

Since we use W_1' to approximate $\widetilde{W}_1^{(1)}(A)$, then W_1' can be regarded as a fixed tensor when initializing W_2.

$$
\begin{aligned}
&\nabla_A L(A, W_2, H_2; D_2^{(val)}) \\
&\approx \nabla_A L(A, W_2 - \xi\nabla_{W_2}\left[L(A, W_2, H_2; D_2^{(tr)}) + \lambda\|W_2 - \widetilde{W}_1'\|_F^2\right], H_2 - \xi\nabla_{H_2}L(A, W_2, H_2; D_2^{(tr)}); D_2^{(val)}) \\
&= \nabla_A L(A, W_2', H_2'; D_2^{(val)}) \\
&\quad - \xi\nabla_{A,W_2}^2 L(A, W_2, H_2; D_2^{(tr)}) \cdot \nabla_{W_2'} L(A, W_2', H_2'; D_2^{(val)}) \\
&\quad - \xi\nabla_{A,H_2}^2 L(A, W_2, H_2; D_2^{(tr)}) \cdot \nabla_{H_2'} L(A, W_2', H_2'; D_2^{(val)})
\end{aligned}
\tag{3}
$$

where $W_2' = W_2 - \xi\nabla_{W_2}\left[L(A, W_2, H_2; D_2^{(tr)}) + \lambda\|W_2 - \widetilde{W}_1'\|_F^2\right]$; $H_2' = H_2 - \xi\nabla_{H_2}L(A, W, H_2; D_2^{(tr)})$. Then W_2' can be used to initialize W_1.

Besides, for the matrix-vector product, the complexity can be further reduced. Let ϵ be a small scalar and $W^{\pm} = W \pm \epsilon\nabla_{W'}L(A, W', H'; D^{(val)})$, $H^{\pm} = H \pm \epsilon\nabla_{H'}L(A, W', H'; D^{(val)})$:

$$
\begin{aligned}
&\nabla_{A,W}^2 L(A, W, H; D^{(tr)}) \cdot \nabla_{W'} L(A, W', H'; D^{(val)}) \\
&\approx \frac{\nabla_A L(A, W^+, H; D^{(tr)}) - \nabla_A L(A, W^-, H; D^{(tr)})}{2\epsilon} \\
&\nabla_{A,H}^2 L(A, W, H; D^{(tr)}) \cdot \nabla_{H'} L(A, W', H'; D^{(val)}) \\
&\approx \frac{\nabla_A L(A, W, H^+; D^{(tr)}) - \nabla_A L(A, W, H^-; D^{(tr)})}{2\epsilon}
\end{aligned}
\tag{4}
$$

2 Algorithm

The iterative procedure is outlined as follows. The mixed operations $\overline{o}^{(i,j)}$ are defined as same as DARTS.

Algorithm 1: Interleaving Differential Architecture Search

Create mixed operation $\overline{o}^{(i,j)}$ parametrized by $A^{(i,j)}$ for edge (i,j) ;

while *not converged* **do**

1. Update architecture A by equation (2) and equation (3)
2. Update weights W_1, H_1, W_2, H_2
3. Compute objective $L(A, W_1, H_1; D_1^{(val)}) + L(A, W_2, H_2; D_2^{(val)})$

end

Derive the final architecture based on the learned A.