Task Adaptive Parameter Sharing for Multi-Task Learning

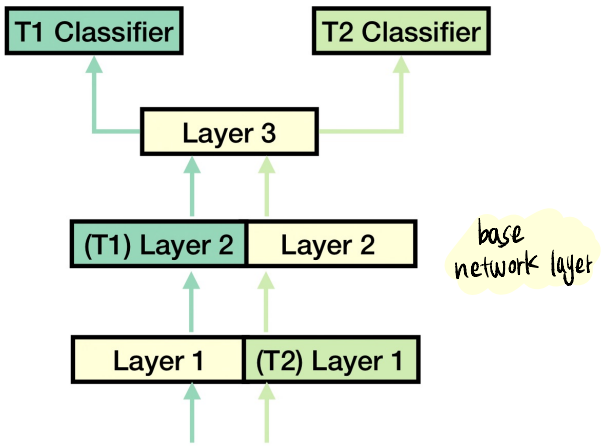
[Methodology]

Given:

- Neural Networks w/ L layers of weights
- K target tasks: $\{T_1, T_2, \ldots, T_K\}$

Goal: select the minimal necessary subset of layers that needs to be tuned to achieve the best (or close to the best) performance.

Idea: To relax the combinatorial problem into a continuous one, which will ultimately give us a simple joint loss function to find both the optimal task-specific layers to tune and optimize parameters of those layers.



(b) Our Method

⟨Weight parametrization⟩

S_i: scoring parameter for layer i

reparametrization of the weights for each layer

$$W_i = \bar{W}_i + I_T(S_i)\,\delta W_i$$

$\bar{W}_i$: the shared weights of the pretrained network

δW_i: a trainable parameter which describes a task-specific perturbation of the base network

$$I_T(S_i) = \begin{cases} 1 & \text{if } S_i \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

← layer is transformed to a task-specific layer (i.e., make its parameters task-specific)

← the layer is the same as the base network and no new parameters are introduced.

⟨Joint Optimization⟩

Recast the initial combinatorial problem as a joint optimization problem over weight deltas and scores.

tries to optimize the task-specific parameters to achieve the best performance on the task

$$(\delta W, S) = \underset{W,S}{\operatorname{argmin}} \mathcal{L}_D(W) + \frac{\lambda}{L} \sum_{i=1}^{L} |s_i|$$

Sparsity inducing regularizer on S which penalizes large values of s_i, encouraging layer sharing rather than introducing task-specific parameters.

$S = (s_1, \cdots, s_L)$

$\delta W = (\delta w_1, \cdots, \delta w_L)$

$W = (w_1 \cdots w_L)$

$\mathcal{L}_D(W)$: the loss of model on dataset D

⟨Straight-through gradient estimation⟩

To make the problem learnable, we utilize a straight-through gradient estimator.
i.e., modify the backward pass and redefine the gradient update as:

$$\nabla_{s_i} w_i = \delta w_i$$

$$(\text{i.e., } \nabla_{s_i} w_i = \nabla_{s_i} \{\overline{w}_i + s_i \delta w_i\})$$

〈Joint MTL〉

How to learn a pretrained representation that reduces the number of task-specific layers that need to be learned to obtain optimal performance:

▨ Loss function

$$(\bar{w}, \delta w_1, \ldots, \delta w_K, s_1, \ldots, s_K) = \arg\min_{\bar{w}, \delta w^k, s} \sum_{k=1}^{K} \left(\mathcal{L}_{D_k}(\bar{w}, \delta w^k) + \frac{1}{L} \sum_{i=1}^{L} |s_i^k| \right)$$

- k: # of tasks
- $\bar{w}$: common weights
- $\delta w^k, s^k$: task specific parameters

↳ encourages learning common weights $\bar{w}$ in such a way that the number of task specific parameters is minimized, due to the L1 penalty on s

⊕ Batch Normalization

[Experiments]