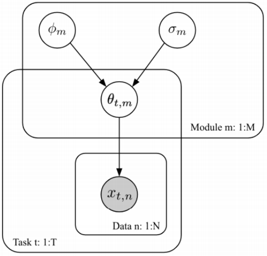
Modular Meta-learning: Variants of MAML

Reference - *Modular Meta-Learning with Shrinkage*

In general $\mathrm{w} = \{\theta_1 \dots \theta_M\}$ e.g., different layers of a network. Variants of MAML learn a prior for w that is adapted for each task; but do all layers need to adapt? E.g. if only one layer is adapted, perhaps it could be trained for many more steps per task without risk of over-fitting. This paper learns to *differently* adapt each layer: assuming each θ_m is normally distributed as $\mathcal{N}(\phi_m, \sigma_m^2)$. **Layers with small or zero σ_m^2 will not adapt.** To learn ϕ, σ^2 we take Bayesian view:

$$p(\mathrm{w}^{1:T}, \mathcal{D}|\phi, \sigma^2) = \prod_{t=1}^{T} \prod_{m=1}^{M} \mathcal{N}(\theta_m^t|\phi_m, \sigma_m^2) \prod_{t=1}^{T} p(\mathcal{D}_t|_t)$$



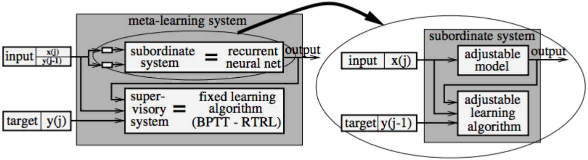
using the MAML approach to update ϕ, σ^2:
the inner loop computes:
$\hat{\theta}^t(\phi, \sigma^2) \equiv \operatorname*{argmin}_{\mathrm{w}^t} \left[-\log p(\mathcal{D}_t^{Train}|\mathrm{w}^t) - \log p(\mathrm{w}^t|\phi, \sigma^2)\right]$

& the outer loop minimizes $\frac{1}{T}\sum_{t=1}^{T} -\log p(\mathcal{D}_t^{Test}|\hat{\mathrm{w}}^t)$.

Model-based Meta-learning

Reference - *Learning to Learn using Gradient Descent*

Early paper that introduces the idea of inputting a *dataset* to a recurrent network as a sequence with labels. A 'supervisory procedure' - gradient descent - is used to train such a network: basically, an LSTM is used as a meta-learner taking datasets $D_k = \{(x_1^k, y_0^k)\dots(x_n^k, y_{n-1}^k)\}$ as input with targets $\{y_1^k \dots y_n^k\}$ The network learns to predict y_{n+1} for any new unlabeled labeled input $(x_{n+1}, \_)$



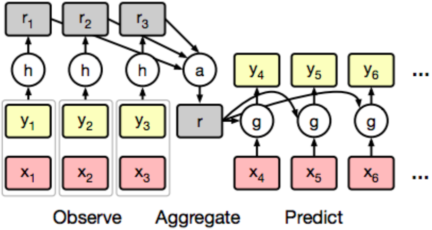
$h_t, c_t = LSTM([x_t, y_{t-1}], h_{t-1}, c_{t-1})$. Predict $y_i = g(x_i, LSTM([x_i, y_k], h_k, c_k))$, $k = i - 1$ for $x_i \in D_{Train}$, and $k = n$ for $x_i \in D_{Test}$

Update parameters of $LSTM$ *and* g using gradient of loss over D_{Train}*

Model-based Meta-learning (cont) 1

Reference - *Contitional Neural Processes*

Training examples are passed through an MLP to generate representations; class-specific representations are aggregated and passed to a second classification MLP concatenated with query examples (from both test and train). Entire network is trained end-to-end on multiple tasks. Recent paper and seems to beat many baselines.

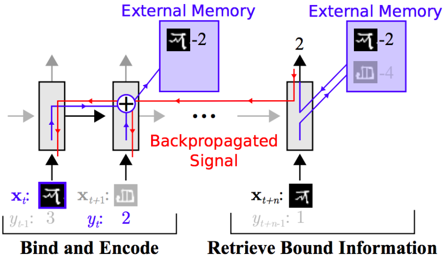


Model-based Meta-learning (cont) 2

Reference - *Meta-Learning with Memory-Augmented Neural Networks*

Datasets are presented as sequences, $\{(x_t, y_{t-1})\}$ as in the RNN-based approach. A 'memory-augmented LSTM/FF network' is the meta-learner.

Network updates rows of a memory matrix M with keys $\mathrm{k}_t = \phi(x_t)^T$:
$M_t = M_{t-1} + \mathrm{w}^w \mathrm{k}_t$; w^w = write weights
Retrieved memory $r_t = \mathrm{w}^r M_t$ used to predict $\hat{y}_t$ using a feedforward layer.
(read weights $\mathrm{w}^r = Softmax(M_t \cdot k_t)$)
usage weights: $\mathrm{w}^u_t = \gamma \mathrm{u}^u_{t-1} + \mathrm{w}^r_t + \mathrm{w}^w_t$
used to compute w^w as follows:



'least used' $w^{lu} = 1$ for t smallest elements of w^u, 0 otherwise
$\mathrm{w}^w_t = \delta w^r_{t-1} + (1-\delta)\mathrm{w}^{lu}_{t-1}$; prior to writing the least used row of M is zeroed.
Read/write to 'memory' - we can view this as a 'neural Turing machine'.