Information Sharing Across Tasks

From Multi-tasking to Meta-learning

Anshul Thakur

Department of Engineering Science
University of Oxford

8th Nov, 2022

Table of Contents

Table of Contents

Multi-tasking Architectures

- Multi-tasking architectures are characterised by:
  - A shared feature extractor "trunk"
  - Tasks-specific layers working on same feature representation

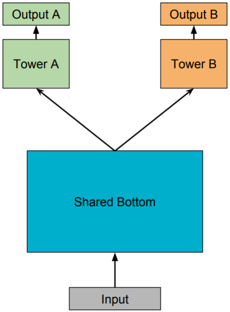


Figure: Illustration of a typical multi-tasking architecture.

Multi-tasking: Under the hood

- $\mathbf{x}$: input examples. $(y_1, y_2, \ldots, y_n)$: corresponding task labels
- Model $f_{\theta,\phi_i}()$ outputs task-specific predictions: $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$
- θ: shared parameters, ϕ_i: task-specific parameters for ith task
- Loss function is an aggregate of task-specific losses:

$$\mathcal{L} = \sum_{i=1}^{n} \alpha_i \mathcal{L}_i(y_i, \mathbf{p}_i), \tag{1}$$

where α_i is the coefficient for task i.

Multi-tasking: Under the hood

- Computing gradients for shared parameters:

$$\nabla_\theta \mathcal{L} = \sum_{i=1}^{n} \alpha_i \nabla_\theta \mathcal{L}_i(y_i, \mathbf{p}) \tag{2}$$

- α_i are usually constrained to be the convex coefficients

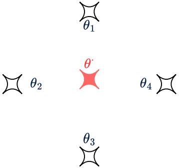
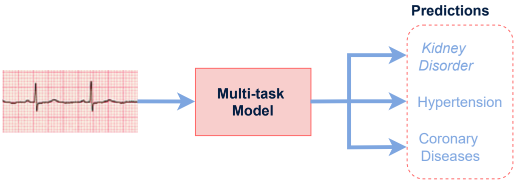


Figure: Deviation between optimal parameters and shared parameters learned by MTL.

Why do we need multi-tasking?



- Predicting multiple outcomes for a patient from a single input:
  - Avoids firing multiple models
  - Lesser storage complexity
- Regularises the training
  - Avoids over-fitting
  - Root cause of this regularisation is learning common representation
- Another way to look at it:
  - MTL adds noise to noisy gradients
  - Further increasing the chances of arriving at wider local minima

Table of Contents

Challenges in Multi-tasking: Shared Representation

- Is it always possible to learn a good shared representation?
  - Of-course not!
- A good shared representation could be obtained if tasks are similar
  - Our and machine's perception of task similarity could be different!
- Less similar tasks result in shared parameters that are not be helpful to any task
- **Solution:** Task Affinity Groupings[1]
  - Identify similar tasks and use them for MTL
  - Strategy: Update shared layers using a task-specific loss and analysing the impact of the updated representation on other losses

[1] Fifty et al., Efficiently Identifying Task Groupings for Multi-Task Learning, Neurips 2021

Challenges in Multi-tasking: Shared Representation

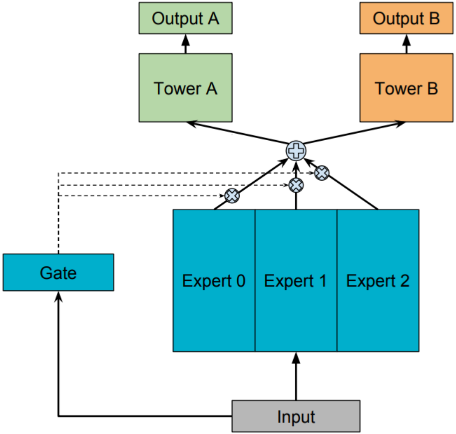


Figure: Mixture of Experts for multi-tasking

Challenges in Multi-tasking: Optimisation Issues

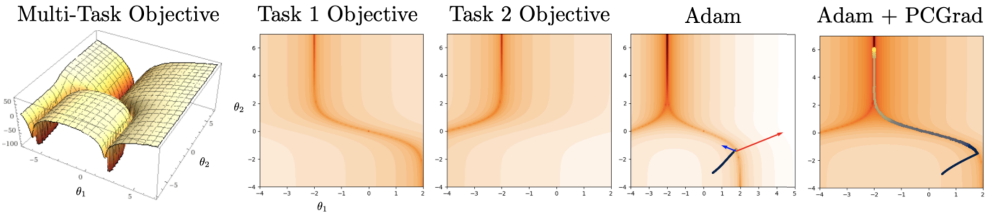


Figure: Gradient conflicts

Source: Yu et al., Gradient Surgery for Multi-Task Learning, Neurips 2020.

Challenges in Multi-tasking: Optimisation Issues

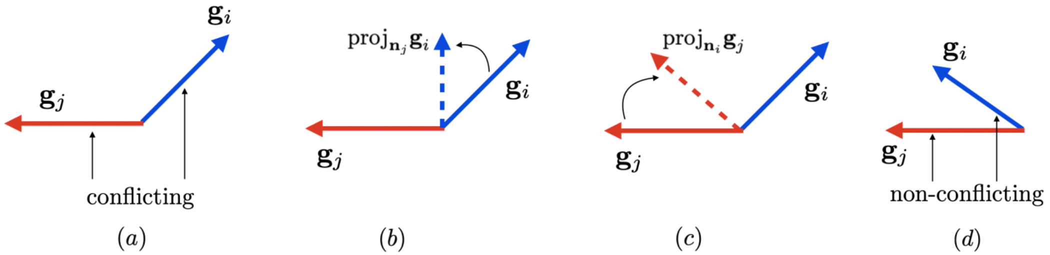


Figure: Gradient conflicts

Source: Yu et al., Gradient Surgery for Multi-Task Learning, Neurips 2020.

Challenges in Multi-tasking: Gradient Washout

- Loss of one of the tasks may overwhelm the other losses[2]
- Multi-tasking objective will mostly be optimised for this dominating tasks
- **Solution**: Meta-learn coefficients to make sure every task gets importance

[2] Navon et al., Multi-Task Learning as a Bargaining Game, ICML 2022.

Challenges in Multi-tasking: Label Availability

- Availability of all labels for an example is rare
- In healthcare informatics, it’s possible that a patient may not exhibit all the outcomes that we are modelling
- MTL can only use a subset of all the available data

Table of Contents

Meta-learning: Learning To Learn

- **Interpretation:** Training a model using gradients computed by other models
- **Purpose:** Learning a common model across multiple tasks
  - Global/common model can be quickly adapted to new unseen tasks
  - Proposed to mimic the learning in humans
- **Why are we interested in meta-learning?:**
  - Provides common/shared model across multiple tasks
  - Alleviate example-labels requirement of MTL
  - Information sharing across tasks in an example-independent manner

First Order Meta-learning: REPTILE[3]

- $f_\theta()$: Global model parameterised by θ
- $\mathcal{D}_t = \{\mathbf{x}_i, y_i\}_{i=1}^n$: Task t dataset

---

for $t \leftarrow 1 : T$ **do**

$\theta_t = \theta$

$\mathcal{B} \leftarrow$ SAMPLE-BATCHES($\mathcal{D}_t$)

for any $(\mathbf{b}, \mathbf{l}) \in \mathcal{B}$ **do**

$L = \mathcal{L}(f_{\theta_t}(\mathbf{b}, \mathbf{l}))$

$\theta_t = \theta_t - \alpha \nabla_{\theta_t} L$

$\phi = \sum_{t=1}^{T} (\theta_t - \theta)$

$\theta = \theta + \alpha\phi$

---

[3] https://openai.com/blog/reptile/

Multi-tasking to Meta-learning

- All inner-loop updates can be summaries as one single gradient update
- Meta-grad for task t, $\phi_t = -(\theta - \theta_t) = \nabla_{\theta_t} L$
- REPTILE update can be written as:

$$\boxed{\nabla \mathcal{L}_\theta = \sum_{t=1}^{T} \nabla_{\theta_t} L = \sum_{t=1}^{T} \alpha_i \nabla_{\theta_t} L} \tag{3}$$

- REPTILE/MAML, MTL and FedAvg are identical w.r.t. optimisation

Meta-learning as Information Sharing Mechanism

- All tasks are trained independently
- No requirement of one example and multiple labels
- We can share information across tasks even if patients are different
- All other problems of MTL also exist in MAML/REPTILE

Table of Contents

An Ideal Information Sharing Framework

- **Example Independent:** Unlike multi-tasking, like meta-learning
- **No Optimisation Issues:** Unlike multi-tasking and meta-learning
- **No Gradient Aggregation Issues :** Unlike multi-tasking, meta-learning and FedAvg
- **Heterogeneity:** Unlike multi-tasking, meta-learning and FedAvg
- **Domain-agnostic**
- Models performing multiple tasks is not the target

Epilogue

- At CHI lab, we are working on realising an ideal information sharing framework
- Federated learning, Meta-learning and Multi-tasking deal with same issues
- Reach out at anshul.thakur@eng.ox.ac.uk for collaboration