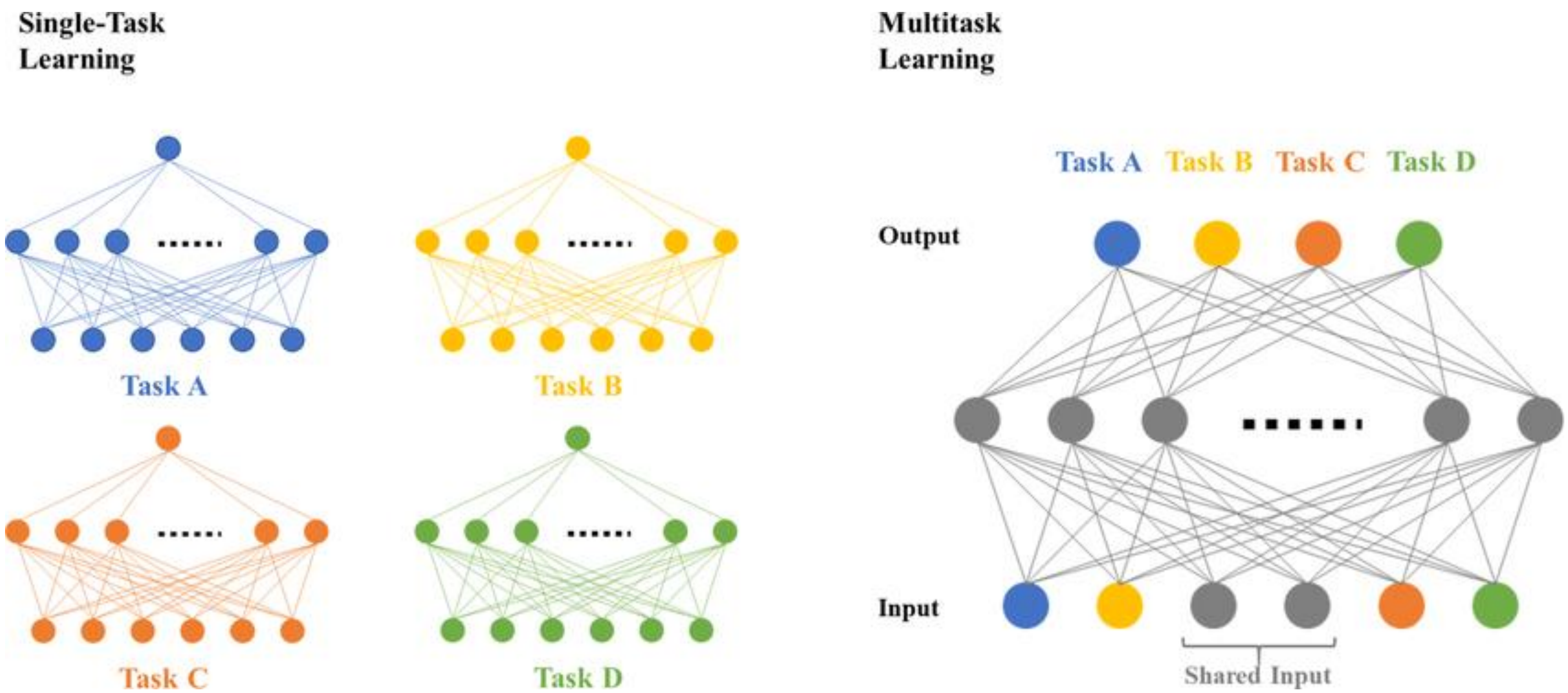
Ways to balance multiple losses in multi-task learning

Yifei Hu

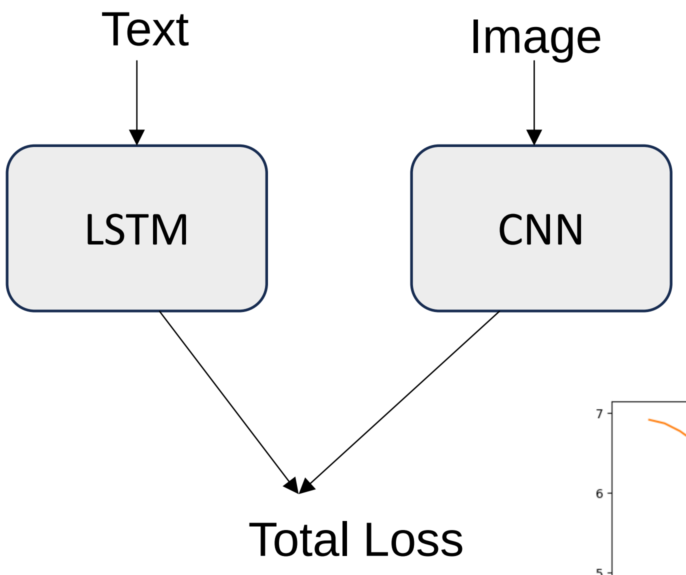
2024/09/13

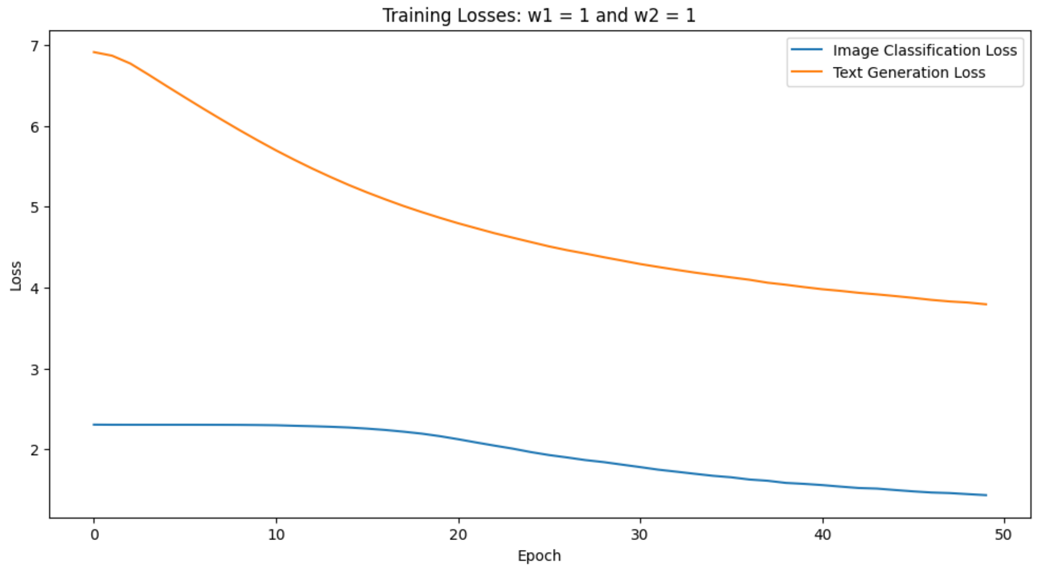
What is multi-task learning?

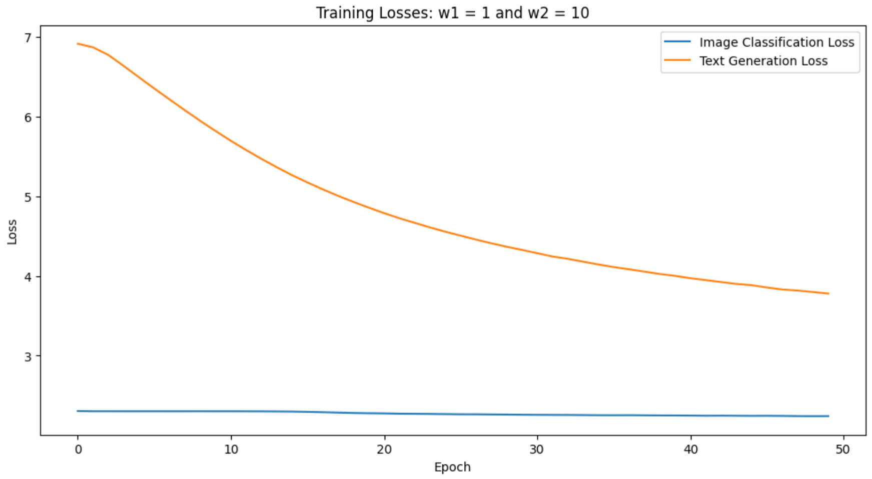


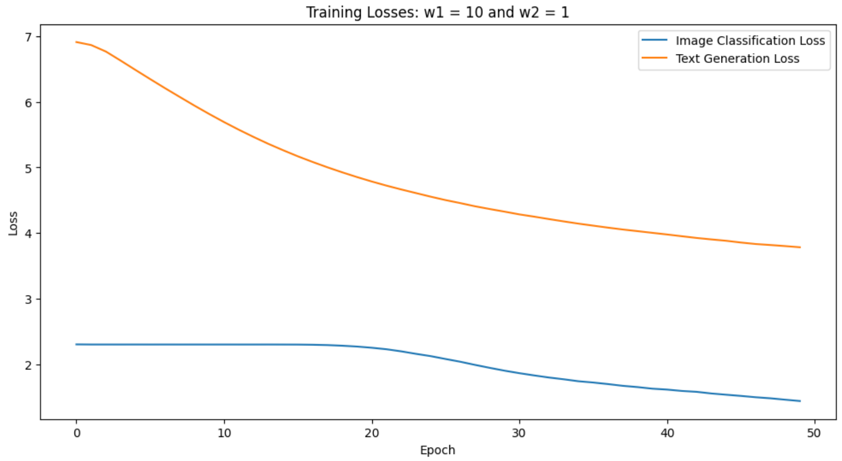
$$L_{total} = \sum_i w_i L_i$$

Multi-task learning example









How to find the appropriate w_i ?

Normalize the weight

1. Use initial value as the weight

$$\mathcal{L} = \sum_{i=1}^{n} \frac{\mathcal{L}_i}{\mathcal{L}_i^{(\text{init})}} \rightarrow \mathcal{L} = \sum_{i=1}^{n} \frac{\mathcal{L}_i}{\mathcal{L}_i^{(\text{prior})}}$$

- Scale all the loss to the same level
- Get the value by running several batches of data or roughly estimate(for K-class classification, the initial loss will be $logK$)
- Cannot weight how “difficult” the task is.

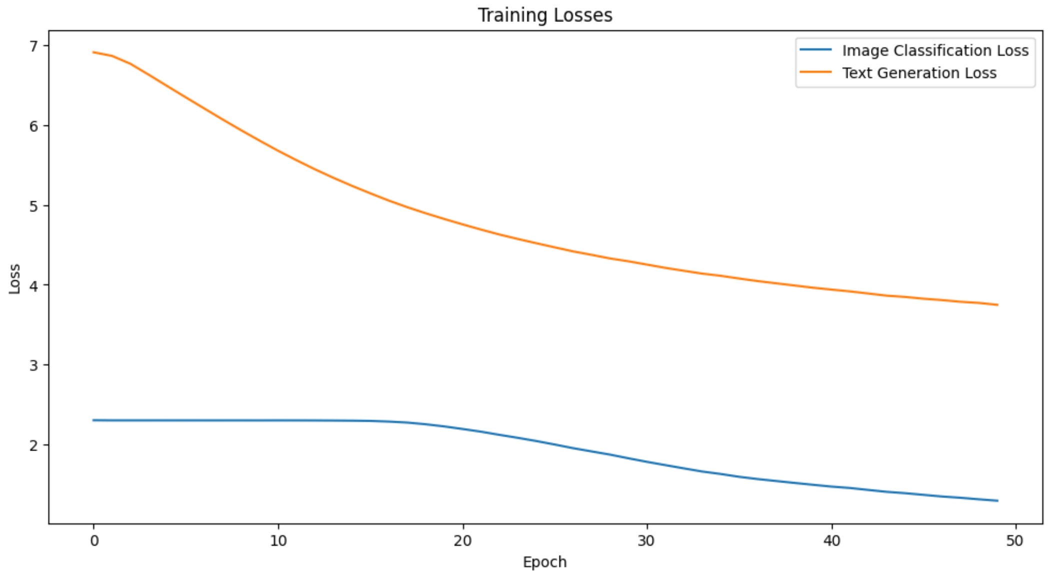
- Apply prior distribution to the loss

Normalize the weight

2. Change the weight dynamically during training

$$\mathcal{L} = \sum_{i=1}^{n} \frac{\mathcal{L}_i}{\mathcal{L}_i^{(\mathrm{sg})}}$$

```
loss = loss1/loss1.detach() + loss2/loss2.detach()
```



Another view to see

$$\mathcal{L} = \sum_{i=1}^{n} \frac{\mathcal{L}_i}{\mathcal{L}_i^{(sg)}}$$

$$\nabla_\theta \left(\frac{\mathcal{L}_i}{\mathcal{L}_i^{(sg)}} \right) = \frac{\nabla_\theta \mathcal{L}_i}{\mathcal{L}_i^{(sg)}} = \frac{\nabla_\theta \mathcal{L}_i}{\mathcal{L}_i} = \nabla_\theta \log \mathcal{L}_i$$

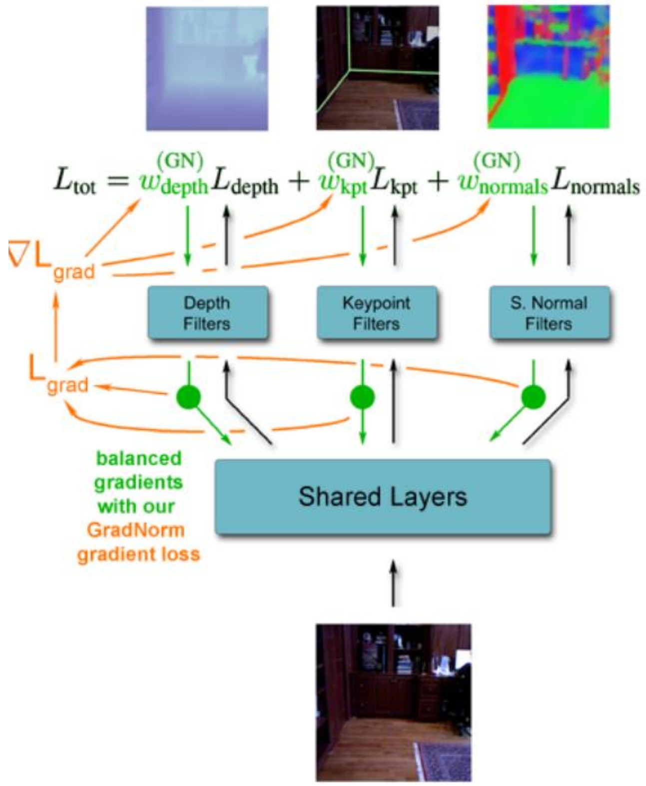
$$\mathcal{L} \equiv \sum_{i=1}^{n} \log \mathcal{L}_i = n \log \sqrt[n]{\prod_{i=1}^{n} \mathcal{L}_i}$$

Notice here the log function is a monotonous function, so instead we can also choose

$$\mathcal{L} = \sqrt[n]{\prod_{i=1}^{n} \mathcal{L}_i}$$

For further extension we can use $\mathcal{L}(\gamma) = \sqrt[\gamma]{\prod_{i=1}^{\gamma} \mathcal{L}_i}$ and we only need to fine tune γ

Grad Normalization



$$\nabla_\theta \mathcal{L} = \sum_{i=1}^{n} \frac{\nabla_\theta \mathcal{L}_i}{||\nabla_\theta \mathcal{L}_i||}$$

GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks (ICML 2018)

Weight by uncertainty

If We have tasks that the output follows Gaussian distribution

$$
\begin{aligned}
p(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{f}^W(\mathbf{x})) &= p(\mathbf{y}_1 | \mathbf{f}^W(\mathbf{x})) \cdot p(\mathbf{y}_2 | \mathbf{f}^W(\mathbf{x})) \\
&= \mathcal{N}(\mathbf{y}_1; \mathbf{f}^W(\mathbf{x}), \sigma_1^2) \cdot \mathcal{N}(\mathbf{y}_2; \mathbf{f}^W(\mathbf{x}), \sigma_2^2)
\end{aligned}
$$

Then:

$$
\begin{aligned}
\mathcal{L}(\mathbf{W}, \sigma_1, \sigma_2) &= -\log p(y_1, y_2 | \mathbf{f}^W(\mathbf{x})) \\
&\propto \frac{1}{2\sigma_1^2} \|\mathbf{y}_1 - \mathbf{f}^W(\mathbf{x})\|^2 + \frac{1}{2\sigma_2^2} \|\mathbf{y}_2 - \mathbf{f}^W(\mathbf{x})\|^2 + \log \sigma_1 \sigma_2 \\
&= \frac{1}{2\sigma_1^2} L_1(\mathbf{W}) + \frac{1}{2\sigma_2^2} L_2(\mathbf{W}) + \log \sigma_1 \sigma_2
\end{aligned}
$$

Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics (ICCV 2018)

Pareto Optimization

For gradient descend, we have

$$\mathcal{L}(\theta + \Delta\theta) \approx \mathcal{L}(\theta) + \langle \nabla_\theta \mathcal{L}, \Delta\theta \rangle$$

$$\Delta\theta = -\eta \nabla_\theta \mathcal{L}$$

Find θ to make $\Delta L < 0$

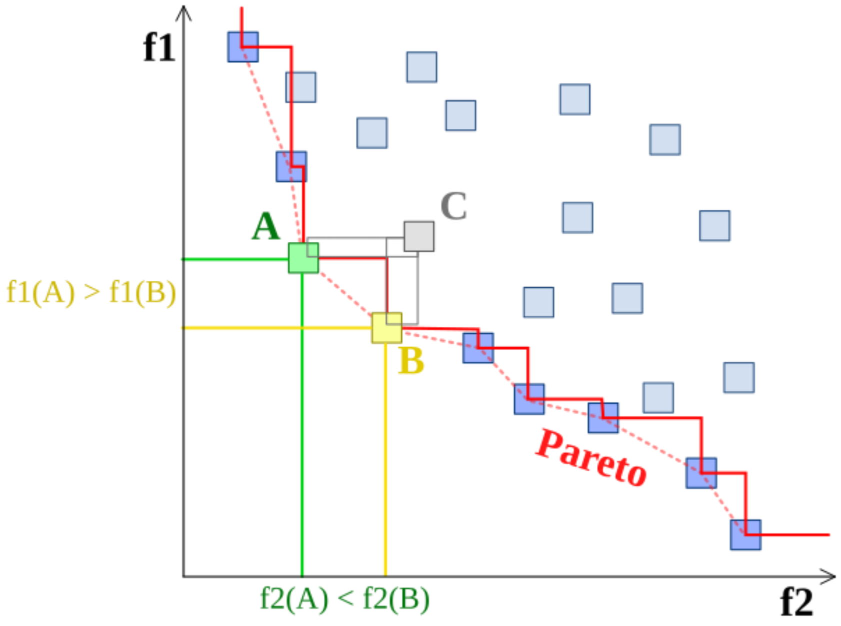
$$\Delta\theta = -\eta \nabla_\theta \mathcal{L}$$

Where η is the learning rate for gradient descend

Pareto Optimization

For Pareto Optimization, we want to find a Pareto optimal solution among all tasks.

$$
\begin{cases}
\langle \nabla_\theta \mathcal{L}_1, \Delta\theta \rangle \leq 0 \\
\langle \nabla_\theta \mathcal{L}_2, \Delta\theta \rangle \leq 0 \\
\vdots \\
\langle \nabla_\theta \mathcal{L}_n, \Delta\theta \rangle \leq 0
\end{cases}
$$



For simplicity, here we make $\boldsymbol{g_i} = \nabla_\theta L_i$

Pareto Optimization

For this task we want to find a vector u that makes

$$\forall i, \langle g_i, u \rangle \geq 0 \quad \Leftrightarrow \quad \min_i \langle g_i, u \rangle \geq 0$$

Which is also to find a u which satisfied

$$\max_u \min_i \langle g_i, u \rangle$$

For stability, we add a regulation term,thus the goal is

$$\max_u \min_i \langle g_i, u \rangle - \frac{1}{2} \|u\|^2 \geq 0$$

Pareto Optimization

For this task we want to find a vector u that makes

$$\forall i, \langle g_i, u \rangle \geq 0 \quad \Leftrightarrow \quad \min_i \langle g_i, u \rangle \geq 0$$

Which is also to find a u which satisfied

$$\max_u \min_i \langle g_i, u \rangle$$

For stability, we add a regulation term,thus the objective is

$$\boxed{\max_u \min_i \langle g_i, u \rangle - \frac{1}{2} \|u\|^2 \geq 0}$$

Solution

Define a set of weight

$$\mathrm{P}^n = \{(\alpha_1, \alpha_2, \cdots, \alpha_n) | \alpha_1, \alpha_2, \cdots, \alpha_n \geq 0, \sum_i \alpha_i = 1\}$$

Easy to verify:

$$\min_i \langle \mathbf{g}_i, u \rangle = \min_{\alpha \in P^n} \langle \tilde{g}(\alpha), u \rangle, \quad \tilde{g}(\alpha) = \sum_i \alpha_i \mathbf{g}_i$$

The objective is equivalent to:

$$\max_u \min_{\alpha \in P^n} \langle \tilde{g}(\alpha), u \rangle - \frac{1}{2} \|u\|^2$$

Based on minmax theorem, the min and the max operation is interchangable:

$$\boxed{\min_{\alpha \in P^n} \max_u \langle \tilde{g}(\alpha), u \rangle - \frac{1}{2} \|u\|^2 = \min_{\alpha \in P^n} \frac{1}{2} \|\tilde{g}(\alpha)\|^2}$$

The problem becomes finding a weighted average of gradients such that its magnitude is minimized

Solution

Define a set of weight

$$\mathrm{P}^n = \{(\alpha_1, \alpha_2, \cdots, \alpha_n) | \alpha_1, \alpha_2, \cdots, \alpha_n \geq 0, \sum_i \alpha_i = 1\}$$

Easy to verify:

$$\min_i \langle \mathbf{g}_i, u \rangle = \min_{\alpha \in P^n} \langle \tilde{g}(\alpha), u \rangle, \quad \tilde{g}(\alpha) = \sum_i \alpha_i \mathbf{g}_i$$

The objective is equivalent to:

$$\max_u \min_{\alpha \in P^n} \langle \tilde{g}(\alpha), u \rangle - \frac{1}{2} \|u\|^2$$

Based on minmax theorem, the min and the max operation is interchangable:

$$\boxed{\min_{\alpha \in P^n} \max_u \langle \tilde{g}(\alpha), u \rangle - \frac{1}{2} \|u\|^2 = \min_{\alpha \in P^n} \frac{1}{2} \|\tilde{g}(\alpha)\|^2}$$

The problem becomes finding a weighted average of gradients such that its magnitude is minimized

Two-point situation

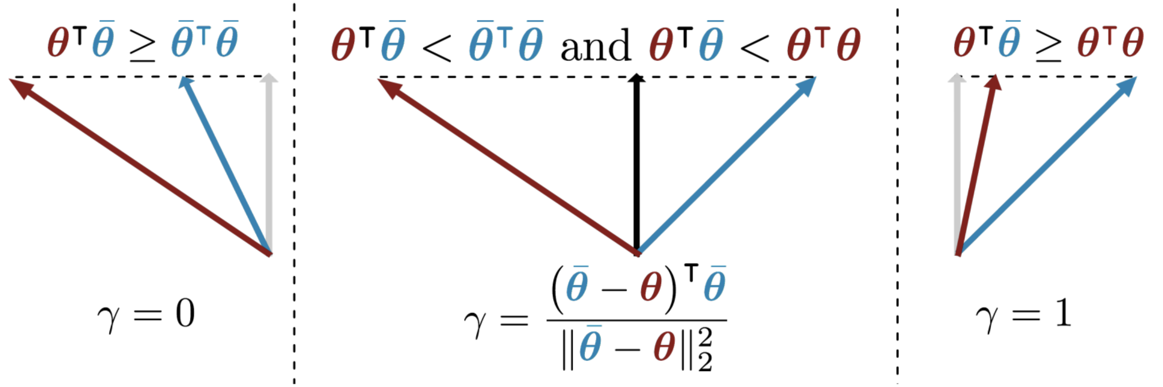


Figure 1: Visualisation of the min-norm point in the convex hull of two points ($\min_{\gamma\in[0,1]} \|\gamma\boldsymbol{\theta} + (1-\gamma)\bar{\boldsymbol{\theta}}\|_2^2$). As the geometry suggests, the solution is either an edge case or a perpendicular vector.

Algorithm 1 $\min_{\gamma\in[0,1]} \|\gamma\boldsymbol{\theta} + (1-\gamma)\bar{\boldsymbol{\theta}}\|_2^2$

1: **if** $\boldsymbol{\theta}^\intercal\bar{\boldsymbol{\theta}} \geq \boldsymbol{\theta}^\intercal\boldsymbol{\theta}$ **then**
2: $\quad\gamma = 1$
3: **else if** $\boldsymbol{\theta}^\intercal\bar{\boldsymbol{\theta}} \geq \bar{\boldsymbol{\theta}}^\intercal\bar{\boldsymbol{\theta}}$ **then**
4: $\quad\gamma = 0$
5: **else**
6: $\quad\gamma = \frac{(\bar{\boldsymbol{\theta}}-\boldsymbol{\theta})^\intercal\bar{\boldsymbol{\theta}}}{\|\boldsymbol{\theta}-\bar{\boldsymbol{\theta}}\|_2^2}$
7: **end if**

Multi-task learning as multi-objective optimization (NeurIPS 2018)

Frank-Wolfe algorithm

- If n > 2

$$\begin{cases} \tau = \text{argmin}_i \langle g_i, \tilde{g}(\alpha^{(k)}) \rangle \\ \gamma = \text{argmin}_\gamma \|\tilde{g}((1-\gamma)\alpha^{(k)} + \gamma e_\tau)\|^2 = \text{argmin}_\gamma \|(1-\gamma)\tilde{g}(\alpha^{(k)}) + \gamma g_\tau\|^2 \\ \alpha^{(k+1)} = (1-\gamma)\alpha^{(k)} + \gamma e_\tau \end{cases}$$

Take home message

- Various methods exist to optimize multi-task learning:
  - Single-task transformation (initial, prior, real-time loss values)
  - Uncertainty weighting/Gradient manipulation (GradNorm)
  - Pareto optimal approaches

Thank you for your listening!