Advanced Artificial Intelligence

Lab 09

Outline

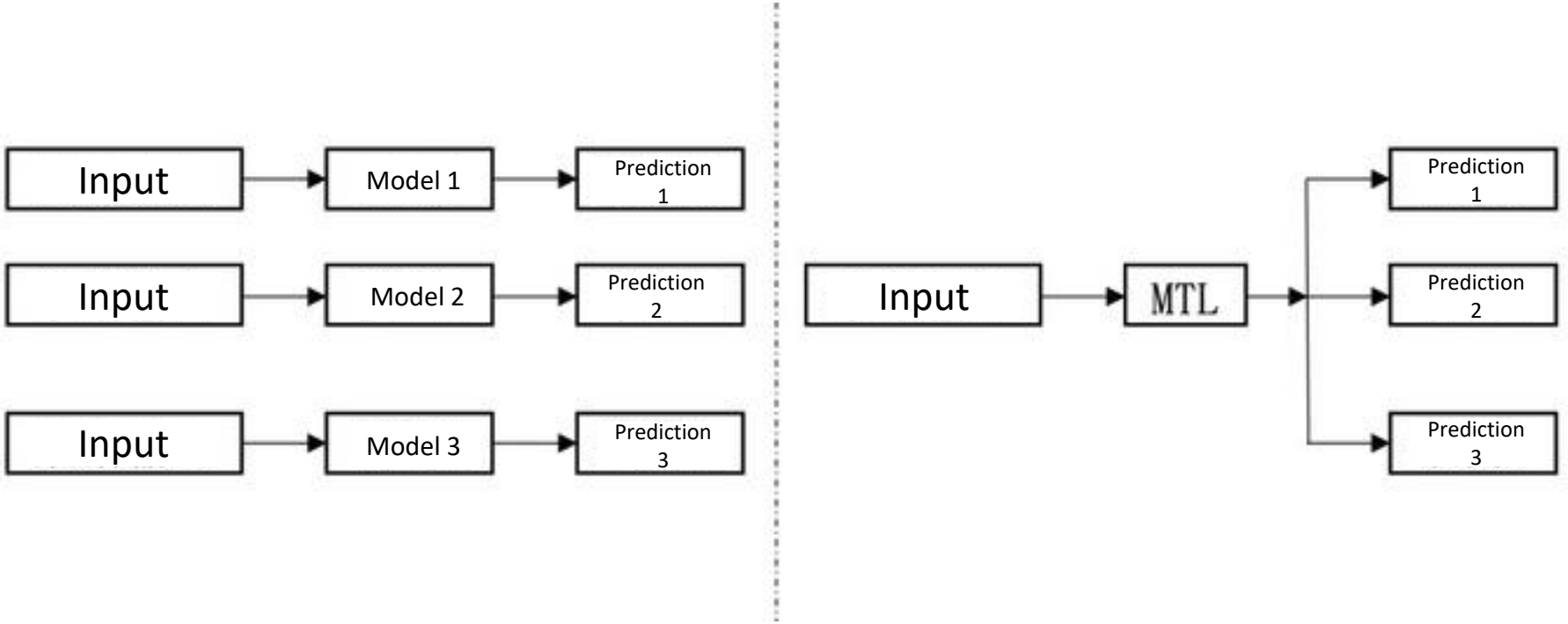
- Introduction of MGDA on multi-task learning
- Exercise

Multi-Task Learning

- Single-task learning: one loss and one task, such as depth prediction or image segmenation in computer visison, can generally be called single-task learning.
- Multi-task learning: in short, multi-task learning is defined as learning multiple objective functions loss at the same time. For example, both depth prediction or image segmenation can be learned with one model.

Multi-Task Learning

Single task learning vs. multi-task learning:



Advantages of multi-task learning

- From the **perspective of machine learning**:

multi-task learning improves the effect of the model by **providing an inductive knowledge**, which is provided by adding auxiliary tasks (specifically, adding a loss).

Advantages of multi-task learning

- Advantages
  - **Efficient:**

    one model can handle multiple tasks, which is very friendly to the industry. The **time cost, calculation cost, storage cost** and even model maintenance cost predicted single models are greater than those of one model.

  - **Regularization:**

    alleviate the over fitting of the model to a certain extent and **improve the generalization ability** of the model.

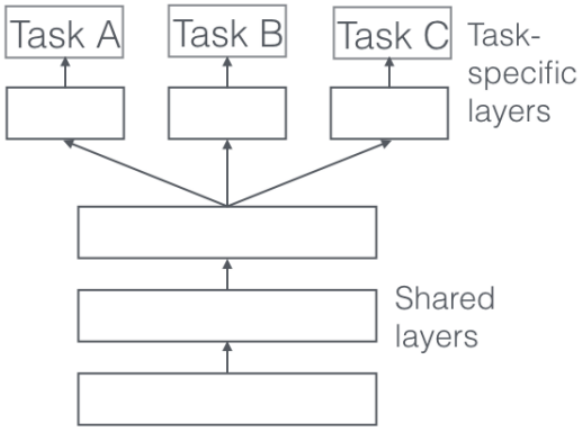
  - **Data enhancement**:

    different tasks have different noises. Learning together will offset part of the noise to a certain extent, making the learning effect better and the model more **robust**.

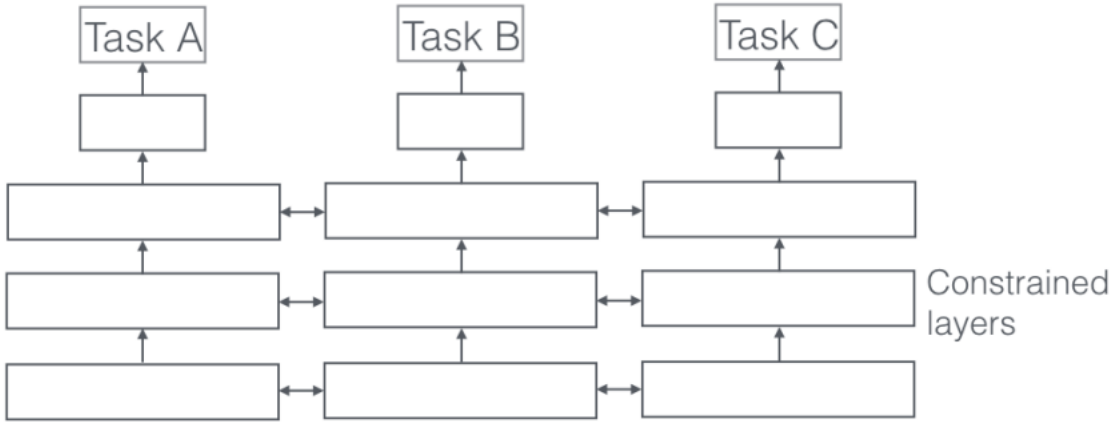
Basic framework of multi-task learning

- Hard parameter sharing: Most parameters are shared. For example, the bottom parameters are shared uniformly, and the top parameters are independent of each model.
- Soft parameter sharing: Each task has its own model and corresponding parameters. Compared with single task learning, different tasks restrict their model parameters through some way of information sharing.

(1) Hard parameter sharing



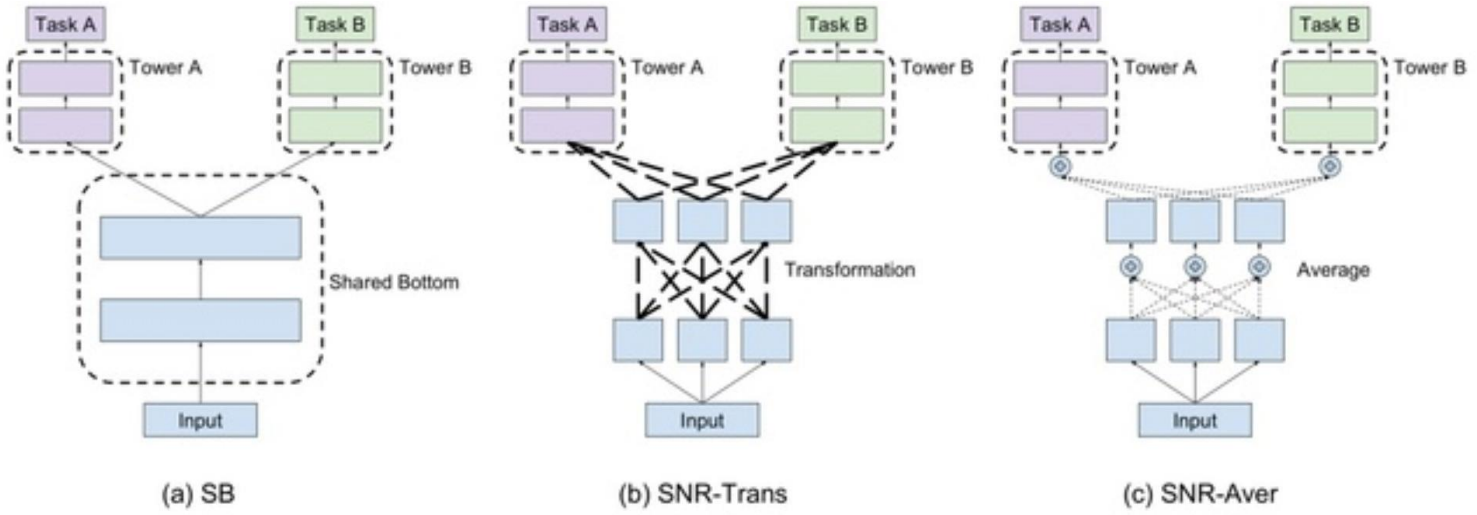
(2) Soft parameter sharing



Main research problems of multi-task learning

- Model architecture design: in addition to the hard parameter sharing and soft parameter sharing mentioned above, researchers have also proposed many different models for different problem scenarios, hoping to achieve better results and minimize the number of parameters of the model.

Example:
SNR model
proposed by Google



Main research problems of multi-task learning

- **Design and improvement of loss function of multi-task learning**: since multiple tasks are put together, the data distribution and importance of different tasks are often different. In most cases, it is not very appropriate to directly sum the loss of all tasks and then optimize the response gradient propagation.
- Generally, the loss function of multi-task learning can be written as, that is, the weighted sum of loss of multiple tasks:

$$\mathcal{L}_{MTL} = \sum_i w_i \cdot \mathcal{L}_i.$$

Main research problems of multi-task learning

How to design the weights of multiple losses:

- The simplest weight design method is to artificially give each task a weight;
- Calculate the weight according to the uncertainty of the task. For details, refer to multi task learning using uncertainty to weight losses for scene geometry and semantics.
- Try to adjust the weights of losses so that each loss contributes equally to the update of shared parameters.
- Introduce methods in other fields, such as multi-objective optimization

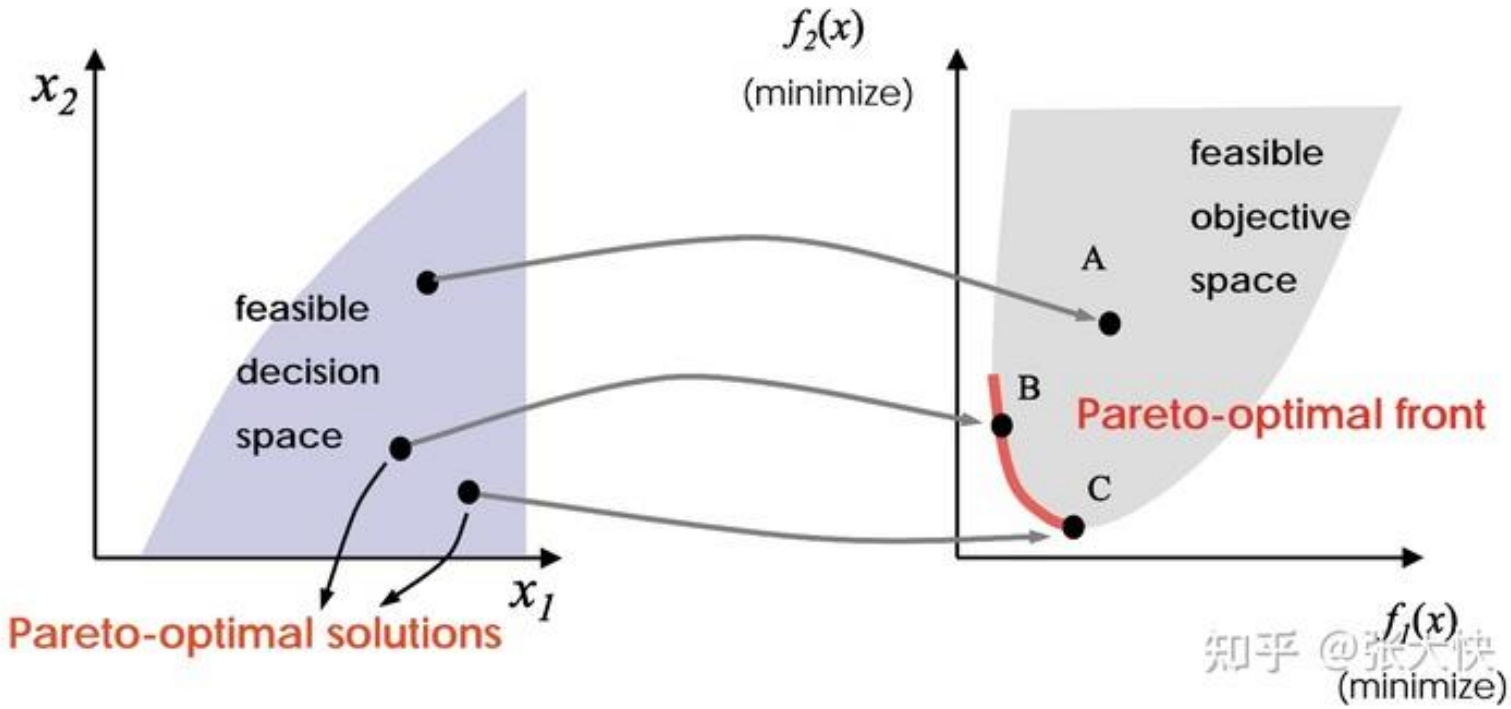
MGDA

- It was first proposed to apply the multi-objective optimization algorithm to multi-task learning: < multi-task learning as multi-objective optimization>
- Usually, the loss function of multi-task learning is a weighted linear combination of multi-task loss. However, when there is a **competitive** relationship between multi tasks, this method may no longer work.
- **multi-task learning is regarded as a multi-objective optimization problem**. The optimization goal is to **find the Pareto optimal solution**.

Refer to:https://proceedings.neurips.cc/paper/2018/hash/432aca3a1e345e339f35a30c8f65edce-Abstract.html

MGDA

- Pareto Front: the set composed of target value vectors corresponding to each solution in the Pareto optimal set is called Pareto optimal front.



MGDA

Consider a multi-task learning (MTL) problem over an input space $\mathcal{X}$ and a collection of task spaces $\{\mathcal{Y}^t\}_{t\in[T]}$, such that a large dataset of i.i.d. data points $\{\mathbf{x}_i, y_i^1, \ldots, y_i^T\}_{i\in[N]}$ is given where T is the number of tasks, N is the number of data points, and y_i^t is the label of the t^{th} task for the i^{th} data point.[1] We further consider a parametric hypothesis class per task as $f^t(\mathbf{x}; \boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) : \mathcal{X} \to \mathcal{Y}^t$, such that some parameters ($\boldsymbol{\theta}^{sh}$) are shared between tasks and some ($\boldsymbol{\theta}^t$) are task-specific. We also consider task-specific loss functions $\mathcal{L}^t(\cdot,\cdot) : \mathcal{Y}^t \times \mathcal{Y}^t \to \mathbb{R}^+$.

Although many hypothesis classes and loss functions have been proposed in the MTL literature, they generally yield the following empirical risk minimization formulation:

$$\min_{\substack{\boldsymbol{\theta}^{sh},\\ \boldsymbol{\theta}^1,\ldots,\boldsymbol{\theta}^T}} \quad \sum_{t=1}^{T} c^t \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \tag{1}$$

for some static or dynamically computed weights c^t per task, where $\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ is the empirical loss of the task t, defined as $\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \triangleq \frac{1}{N}\sum_i \mathcal{L}\big(f^t(\mathbf{x}_i; \boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t), y_i^t\big)$.

MGDA

- Multi-task learning into multi-objective optimization：

Alternatively, MTL can be formulated as multi-objective optimization: optimizing a collection of possibly conflicting objectives. This is the approach we take. We specify the multi-objective optimization formulation of MTL using a vector-valued loss $\mathbf{L}$:

$$\min_{\substack{\boldsymbol{\theta}^{sh},\\ \boldsymbol{\theta}^1,\ldots,\boldsymbol{\theta}^T}} \mathbf{L}(\boldsymbol{\theta}^{sh},\boldsymbol{\theta}^1,\ldots,\boldsymbol{\theta}^T) = \min_{\substack{\boldsymbol{\theta}^{sh},\\ \boldsymbol{\theta}^1,\ldots,\boldsymbol{\theta}^T}} \left(\hat{\mathcal{L}}^1(\boldsymbol{\theta}^{sh},\boldsymbol{\theta}^1),\ldots,\hat{\mathcal{L}}^T(\boldsymbol{\theta}^{sh},\boldsymbol{\theta}^T)\right)^\intercal. \tag{2}$$

The goal of multi-objective optimization is achieving Pareto optimality.

Definition 1 (Pareto optimality for MTL)

(a) *A solution* $\boldsymbol{\theta}$ *dominates a solution* $\bar{\boldsymbol{\theta}}$ *if* $\hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh},\boldsymbol{\theta}^t) \leq \hat{\mathcal{L}}^t(\bar{\boldsymbol{\theta}}^{sh},\bar{\boldsymbol{\theta}}^t)$ *for all tasks* t *and* $\mathbf{L}(\boldsymbol{\theta}^{sh},\boldsymbol{\theta}^1,\ldots,\boldsymbol{\theta}^T) \neq \mathbf{L}(\bar{\boldsymbol{\theta}}^{sh},\bar{\boldsymbol{\theta}}^1,\ldots,\bar{\boldsymbol{\theta}}^T)$.

(b) *A solution* $\boldsymbol{\theta}^\star$ *is called Pareto optimal if there exists no solution* $\boldsymbol{\theta}$ *that dominates* $\boldsymbol{\theta}^\star$.

The set of Pareto optimal solutions is called the Pareto set ($\mathcal{P}_{\boldsymbol{\theta}}$) and its image is called the Pareto front ($\mathcal{P}_{\mathbf{L}} = \{\mathbf{L}(\boldsymbol{\theta})\}_{\boldsymbol{\theta}\in\mathcal{P}_{\boldsymbol{\theta}}}$). In this paper, we focus on gradient-based multi-objective optimization due to its direct relevance to gradient-based MTL.

MGDA

- Multi-task learning into multi-objective optimization:

$$\min_{\theta^{sh},\theta^{i},i=1,\ldots,T} L(\theta^{sh},\theta^{1},\ldots,\theta^{T}) = \min_{\theta^{sh},\theta^{i},i=1,\ldots,T} (\hat{\mathcal{L}}^{t}(\theta^{sh},\theta^{1}),\ldots,\hat{\mathcal{L}}^{t}(\theta^{sh},\theta^{T}))^{\top}$$

- Different from the traditional setting of multi-task learning loss function, the goal here is not scalar, but **vector**. The goal of the solution is **Pareto optimality**.

MGDA

As in the single-objective case, multi-objective optimization can be solved to local optimality via gradient descent. In this section, we summarize one such approach, called the multiple gradient descent algorithm (MGDA) (Désidéri, 2012). MGDA leverages the Karush-Kuhn-Tucker (KKT) conditions, which are necessary for optimality (Fliege and Svaiter, 2000; Schäffler et al., 2002; Désidéri, 2012). We now state the KKT conditions for both task-specific and shared parameters:

- There exist $\alpha^1, \ldots, \alpha^T \geq 0$ such that $\sum_{t=1}^{T} \alpha^t = 1$ and $\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) = 0$
- For all tasks t, $\nabla_{\boldsymbol{\theta}^t} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) = 0$

Any solution that satisfies these conditions is called a Pareto stationary point. Although every Pareto optimal point is Pareto stationary, the reverse may not be true. Consider the optimization problem

$$\min_{\alpha^1,\ldots,\alpha^T} \left\{ \left\| \sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t) \right\|_2^2 \middle| \sum_{t=1}^{T} \alpha^t = 1, \alpha^t \geq 0 \quad \forall t \right\} \tag{3}$$

Désidéri (2012) showed that either the solution to this optimization problem is 0 and the resulting point satisfies the KKT conditions, or the solution gives a descent direction that improves all tasks. Hence, the resulting MTL algorithm would be gradient descent on the task-specific parameters followed by solving (3) and applying the solution ($\sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}}$) as a gradient update to shared parameters. We discuss how to solve (3) for an arbitrary model in Section 3.2 and present an efficient solution when the underlying model is an encoder-decoder in Section 3.3.

MGDA

- For example, for two tasks, the optimization problem is transformed into:

$$\min_{\alpha\in[0,1])} ||\alpha\nabla_{\theta^{sh}}\hat{L}^1(\theta^{sh},\theta^1)+(1-\alpha)\nabla_{\theta^{sh}}\hat{L}^2(\theta^{sh},\theta^2)||_2^2$$

$$\hat{\alpha}=[\frac{(\nabla_{\theta^{sh}}\hat{L}^2(\theta^{sh},\theta^2)-\nabla_{\theta^{sh}}\hat{L}^1(\theta^{sh},\theta^1))^T\nabla_{\theta^{sh}}\hat{L}^2(\theta^{sh},\theta^2)}{||\nabla_{\theta^{sh}}\hat{L}^1(\theta^{sh},\theta^1)-\nabla_{\theta^{sh}}\hat{L}^2(\theta^{sh},\theta^2)||_2^2}]_{+,\frac{1}{T}}$$

$$[a]_{+,\frac{1}{T}}=\max(min(a,1),0)$$

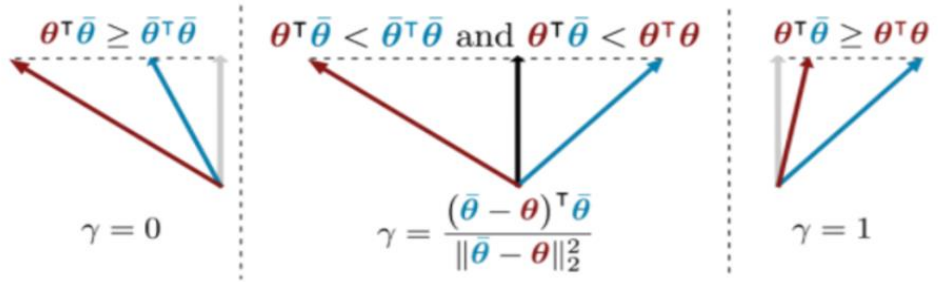


Figure 1: Visualisation of the min-norm point in the convex hull of two points ($\min_{\gamma\in[0,1]} \|\gamma\boldsymbol{\theta}+(1-\gamma)\bar{\boldsymbol{\theta}}\|_2^2$). As the geometry suggests, the solution is either an edge case or a perpendicular vector.

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

if $\boldsymbol{\theta}^\intercal\bar{\boldsymbol{\theta}} \geq \boldsymbol{\theta}^\intercal\boldsymbol{\theta}$ **then**
  $\gamma = 1$
else if $\boldsymbol{\theta}^\intercal\bar{\boldsymbol{\theta}} \geq \bar{\boldsymbol{\theta}}^\intercal\bar{\boldsymbol{\theta}}$ **then**
  $\gamma = 0$
else
  $\gamma = \frac{(\bar{\boldsymbol{\theta}}-\boldsymbol{\theta})^\intercal\bar{\boldsymbol{\theta}}}{\|\boldsymbol{\theta}-\bar{\boldsymbol{\theta}}\|_2^2}$
end if

MGDA

- Of course, the number of tasks equal to 2 is only a special case. When the number of tasks is greater than or equal to 2, the author proposes that Frank-Wolfe algorithm can be used to solve the constrained optimization problem.
- Frank-Wolfe algorithm refer to:
  - < Revisiting Frank-Wolfe: Projection-free sparse convex optimization>, http://proceedings.mlr.press/v28/jaggi13.html
  - <An algorithm for quadratic programming>, https://onlinelibrary.wiley.com/doi/10.1002/nav.3800030109

Results

Table 3: Performance of MTL algorithms on MultiMNIST. Single-task baselines solve tasks separately, with dedicated models, but are shown in the same row for clarity.

	Left digit accuracy [%]	Right digit accuracy [%]
Single task	**97.23**	**95.90**
Uniform scaling	96.46	94.99
Kendall et al. 2018	96.47	95.29
GradNorm	96.27	94.84
Ours	**97.26**	**95.90**

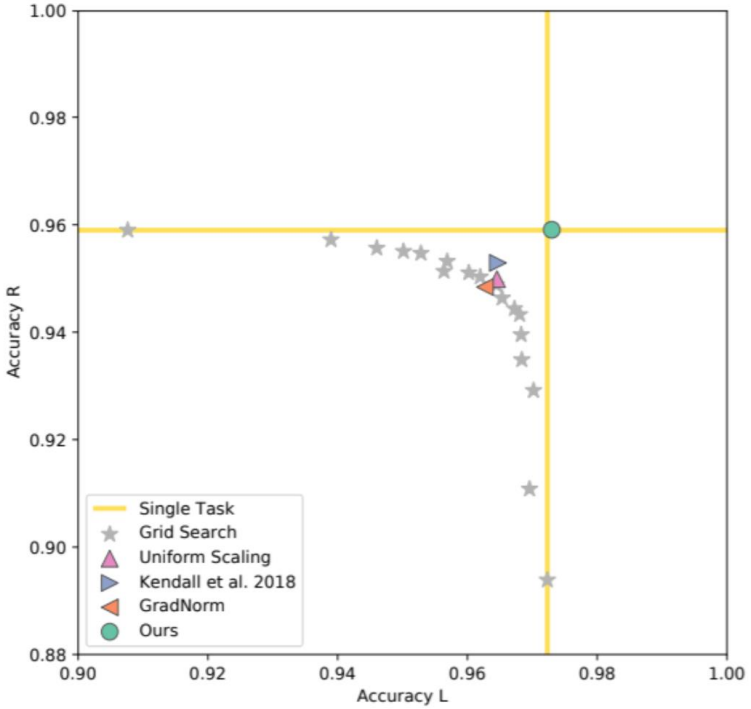


Figure 3: **MultiMNIST accuracy profile.** We plot the obtained accuracy in detecting the left and right digits for all baselines. The grid-search results suggest that the tasks compete for model capacity. Our method is the only one that finds a solution that is as good as training a dedicated model for each task. Top-right is better.

Exercise

- Read the original paper <Multi-Task Learning as Multi-Objective Optimization> . Can you put forward your own thinking in connection with the multi-objective optimization knowledge learned in the classroom?
- Refer to https://github.com/isl-org/MultiObjectiveOptimization , download and run the open source code and observe the results.