group of machine
gmum
learning research

Continual Learning with Guarantees via Weight Interval Constraints

***Maciej Wołczyk*, *Karol Piczak*,**
Bartosz Wójcik, Łukasz Pustelnik, Paweł Morawiecki,
Jacek Tabor, Tomasz Trzciński, Przemysław Spurek

Background and motivation

group of machine gmum learning research

- **Continual learning** – learning from a stream of data while:
  - reducing forgetting,
  - maximizing knowledge transfer,
  - and satisfying other desiderata on computation and memory.
- The no-forgetting is usually a soft constraint, but sometimes remembering might be critical!
- In this work, we devise a method which can put guarantees on forgetting.
- **Main idea: define a region of viable parameters for the current task, and stay within it for the rest of the training.**

Definition

- In this more restrictive setting, the learning objective for task T_j can be written as [1]:

$$\arg\min_{\theta} \ell(T_j, \theta) \text{ satisfying } \ell(T_m, \theta) \leq \ell(T_m, \theta_m^*)$$

for all $m = 1, \ldots, j-1$

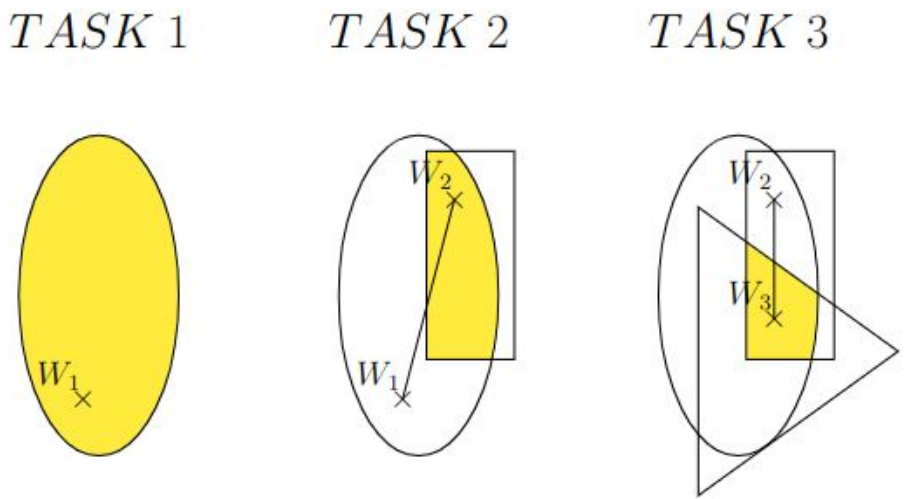
where $\boldsymbol{\theta}_m^*$ are the parameters obtained directly after learning task m

and ℓ is the cross-entropy loss over the whole task dataset.

- Alternatively, we look for parameter of the new task $\boldsymbol{\theta}$ within the set of parameters Θ_m that satisfies the inequality condition.

[1] Chaudhry, et al. *Efficient lifelong learning with a-gem,* ICLR 2019.

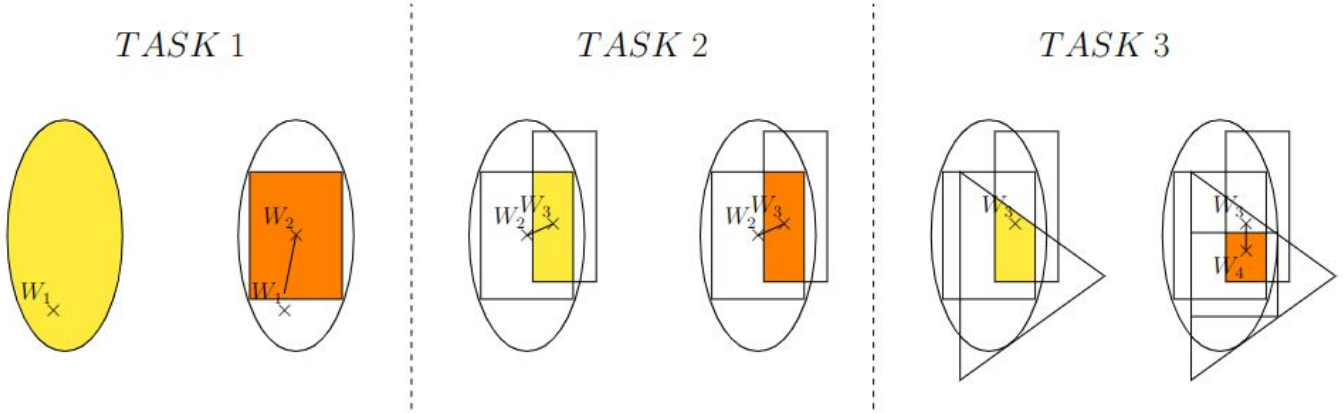
Perfect CL is NP-hard



- Problem: the viable parameter region Θ_m is highly irregular
- Previous work [2] showed that even if we assume that the regions are polytopes, it's still NP-hard.
- In our work, we try to solve this by considering a simpler setting with additional assumptions.

[2] Knoblauch et al, *Optimal Continual Learning has Perfect Memory and is NP-hard,* ICML 2020.

Intersection of parameter regions



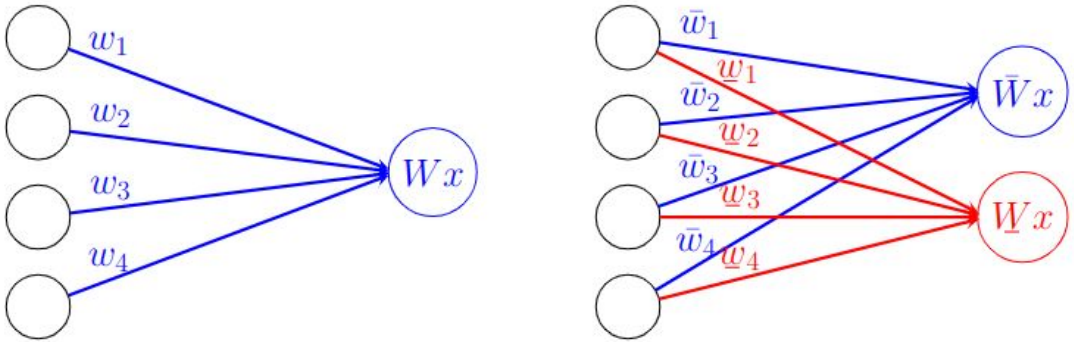
- At each point, we try to find the largest hyperrectangle that fits in the viable region Θ_m.
- Finding intersections between hyperrectangles can be done in polynomial time.
- In our work, we show how to implement this idea efficiently through interval arithmetic.

Practical implementation - intervals

- In practice, we implement the hyperrectangle region by representing each weight as an interval rather than a point in space:

$$[\underline{W}_k, \bar{W}_k] = [W_k - \varepsilon_k, W_k + \varepsilon_k]$$

- We use the tools from interval arithmetic to implement interval neural networks:



Interval upper bound

group of machine gmum learning research

- We can find an upper bound $\hat{\ell}(T_j, \Theta)$ on the worst-case loss for task T_j on the parameter region Θ represented by the hyperrectangle:

$$\hat{\ell}(T_j, \Theta) >= \max_{\theta \in \Theta_m} \ell(T_j, \theta)$$

- As long as we stay within Θ during the training, the loss won't be higher than $\hat{\ell}(T_j, \Theta)$.
- This is easy to compute and differentiable wrt. parameters.
- We can minimize it to find a region of parameters with low loss everywhere

InterContiNet

group of machine gmum learning research

Algorithm 1 InterContiNet training procedure for a given task

Input: model trained on the previous task (weights W^*, radii ε^*), current task T_j.
Reparameterize W, ε using Eq. (5).
for epoch in $1 \ldots$ center_epochs **do**
  Update μ by minimizing $\ell(T_j, \Theta)$ {Train centers}
end for
Initialize ε as largest possible within the previous interval.
for epoch in $1 \ldots$ radii_epochs **do**
  Update ν by minimizing $\hat{\ell}(T_j, \Theta)$ {Train radii}
  if $\hat{\text{acc}} \geq \text{acc} \cdot \text{acc\_thresh}$ **then**
    break
  end if
end for
return W, ε

Our method works in two stages:

1. **Find the centers *W*** that minimize cross-entropy loss while staying within the parameter region.
2. **Shrink the radii *ε*** to minimize the worst-case loss.

Experiments

group of machine gmum learning research

Table 1. The average accuracy across all five tasks of the split MNIST protocol, evaluated after learning the whole sequence. Each value is the average of five runs (with standard deviations).

Method	Incremental task	Incremental domain	Incremental class
SGD	96.27 ± 0.38	64.58 ± 0.26	19.01 ± 0.04
Adam	95.53 ± 3.16	59.32 ± 1.08	19.74 ± 0.01
L2	96.31 ± 0.41	72.21 ± 0.17	18.88 ± 0.18
EWC	97.01 ± 0.13	76.90 ± 0.41	18.90 ± 0.06
oEWC	97.01 ± 0.13	77.02 ± 0.51	18.89 ± 0.07
SI	96.19 ± 0.63	80.62 ± 0.17	17.94 ± 0.57
MAS	96.52 ± 0.14	**84.41 ± 0.42**	17.38 ± 4.19
LwF	97.03 ± 0.05	82.76 ± 0.17	**49.37 ± 0.68**
InterContiNet	**98.93 ± 0.05**	77.77 ± 1.24	40.73 ± 3.26
Offline	99.74 ± 0.03	99.03 ± 0.04	98.49 ± 0.02

Table 2. The average accuracy across all five tasks of the split FashionMNIST protocol, evaluated after learning the whole sequence. Each value is the average of five runs (with standard deviations).

Method	Incremental task	Incremental domain	Incremental class
SGD	92.22 ± 3.06	82.77 ± 0.44	19.91 ± 0.01
Adam	88.13 ± 5.64	78.48 ± 0.47	19.96 ± 0.01
L2	97.36 ± 0.17	**92.65 ± 0.09**	26.91 ± 1.23
EWC	97.53 ± 0.15	92.12 ± 0.18	19.90 ± 0.01
oEWC	96.70 ± 0.57	88.83 ± 0.30	19.87 ± 0.02
SI	97.00 ± 0.25	91.45 ± 0.07	19.97 ± 0.34
MAS	97.43 ± 0.14	91.74 ± 0.19	10.00 ± 0.00
LwF	98.10 ± 0.07	88.63 ± 0.12	**39.51 ± 1.45**
InterContiNet	**98.37 ± 0.06**	**92.65 ± 0.40**	35.11 ± 0.02
Offline	97.98 ± 0.05	96.39 ± 0.06	82.54 ± 0.13

Ablation studies

- By tuning the hyperparameters we can balance **plasticity** (accuracy on the current task, top row) and **stability** (accuracy on the first task, bottom row).
- **Check out our paper for more details!**