Utility-based Perturbed Gradient Descent: An Optimizer for Continual Learning

UNIVERSITY OF ALBERTA

Mohamed Elsayed, A. Rupam Mahmood

NEURAL INFORMATION PROCESSING SYSTEMS

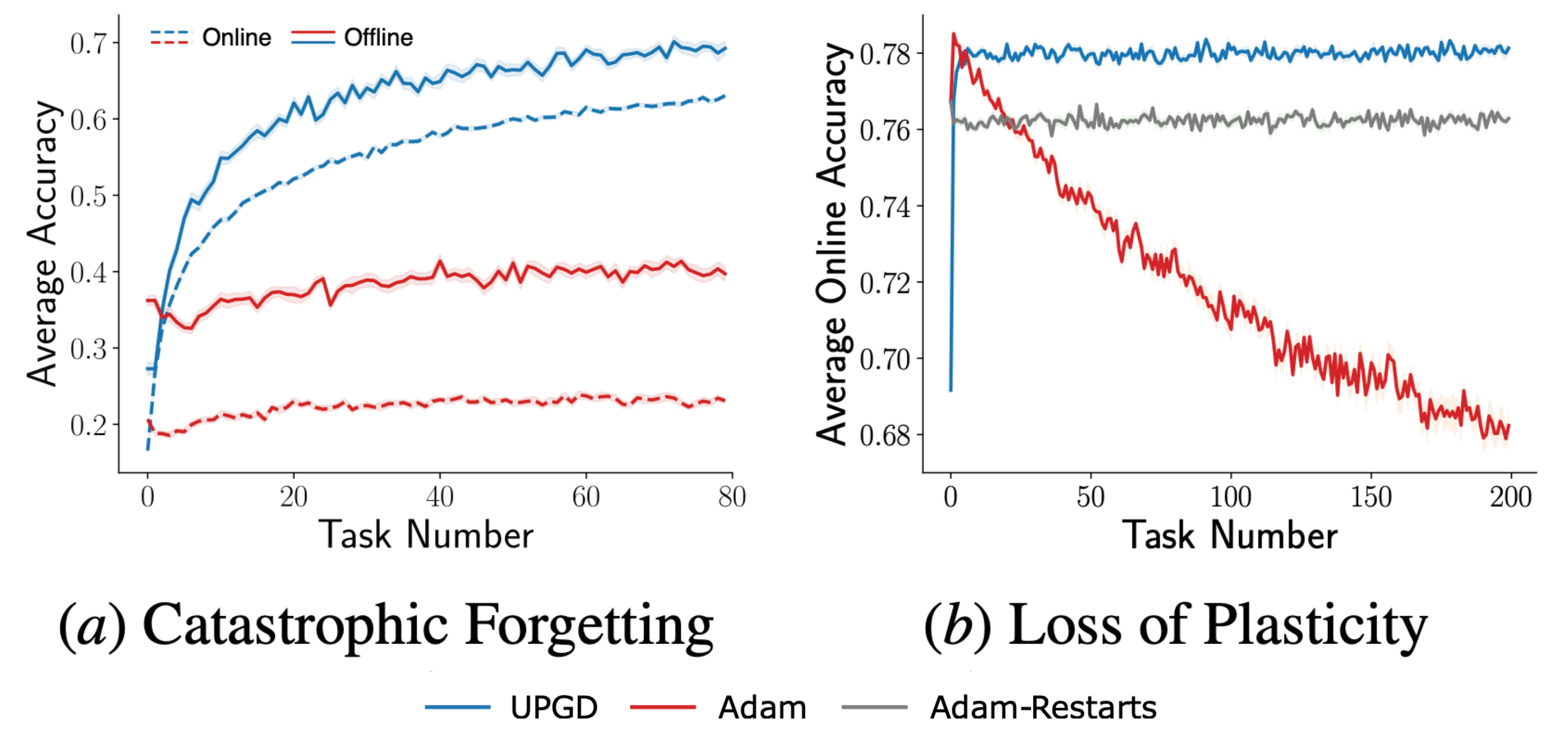
OPT Workshop on Optimization for Machine Learning

Abstract

- Deep representation learning methods struggle with continual learning, suffering from both **catastrophic forgetting** of useful units and **loss of plasticity**, often due to rigid and unuseful units.
- While many methods address these two issues separately, only a few currently deal with both issues of continual learning.
- *Utility-based Perturbed Gradient Descent* (UPGD) protect useful weights and perturb less useful ones, addressing both issues.
- We use the setup of **streaming learning** as the testing ground and design continual learning problems with **hundreds of non-stationarities** and **unknown task boundaries**.
- **UPGD improves performance on non-stationary** problems, being **demonstrably capable of addressing both issues**.

Streaming Learning Setting

- The learner is provided with a stream of samples (x_t, y_t) generated using a **non-stationary *target function*** f_t such that $y_t = f_t(x_t)$.
- The learner observes the input $x_t \in \mathbb{R}^d$, outputs the prediction $\hat{y} \in \mathbb{R}^m$, and then observes the true output $y_t \in \mathbb{R}^m$, strictly in this order. The learner is then evaluated immediately based on an online metric $E(y_t, \hat{y}_t)$ (e.g., accuracy).
- The learner uses a neural network for prediction to learn immediately **without storing the sample**.
- Due to the non-stationarity of the target function, learners encounter loss of plasticity and catastrophic forgetting



(a) Catastrophic Forgetting (b) Loss of Plasticity

(a) Adam suffers from catastrophic forgetting on label-permuted extended MNIST and hence hardly improves performance.
(b) Adam loses plasticity on input-permuted MNIST as newer tasks are presented and performs much worse than Adam with restarts later.

Method

Idea: *to retain useful units while modifying the rest, we need a metric to assess their utility or usefulness.*

- The **utility** of a weight is defined as the change in loss when setting the weight to zero. The utility $U_{l,i,j}(Z)$ of the weight i, j at layer l is:

$$U_{l,i,j} \doteq \mathscr{L}(\mathscr{W}_{\neg[l,i,j]}, Z) - \mathscr{L}(\mathscr{W}, Z)$$

where Z is the sample, $\mathscr{L}(\mathscr{W}, Z)$ is the sample loss given $\mathscr{W}$, and $\mathscr{L}(\mathscr{W}_{\neg[l,i,j]}, Z)$ is a counterfactual loss where $\mathscr{W}_{\neg[l,i,j]}$ is the same as $\mathscr{W}$ except the weight $W_{l,i,j}$ is set to 0.

- This is **computationally expensive** since it requires many additional forward passes. Thus, we aim to approximate it such that no additional forward passes are needed.
- We expand the counterfactual loss $\mathscr{L}(\mathscr{W}_{\neg[l,i,j]}, Z)$ around the current weight $W_{l,i,j}$ and evaluate at weight zero, the result is:

$$U_{l,i,j} \approx -\frac{\partial \mathscr{L}}{\partial W_{l,i,j}} W_{l,i,j} + \frac{1}{2}\frac{\partial^2 \mathscr{L}}{\partial W_{l,i,j}^2} W_{l,i,j}^2$$

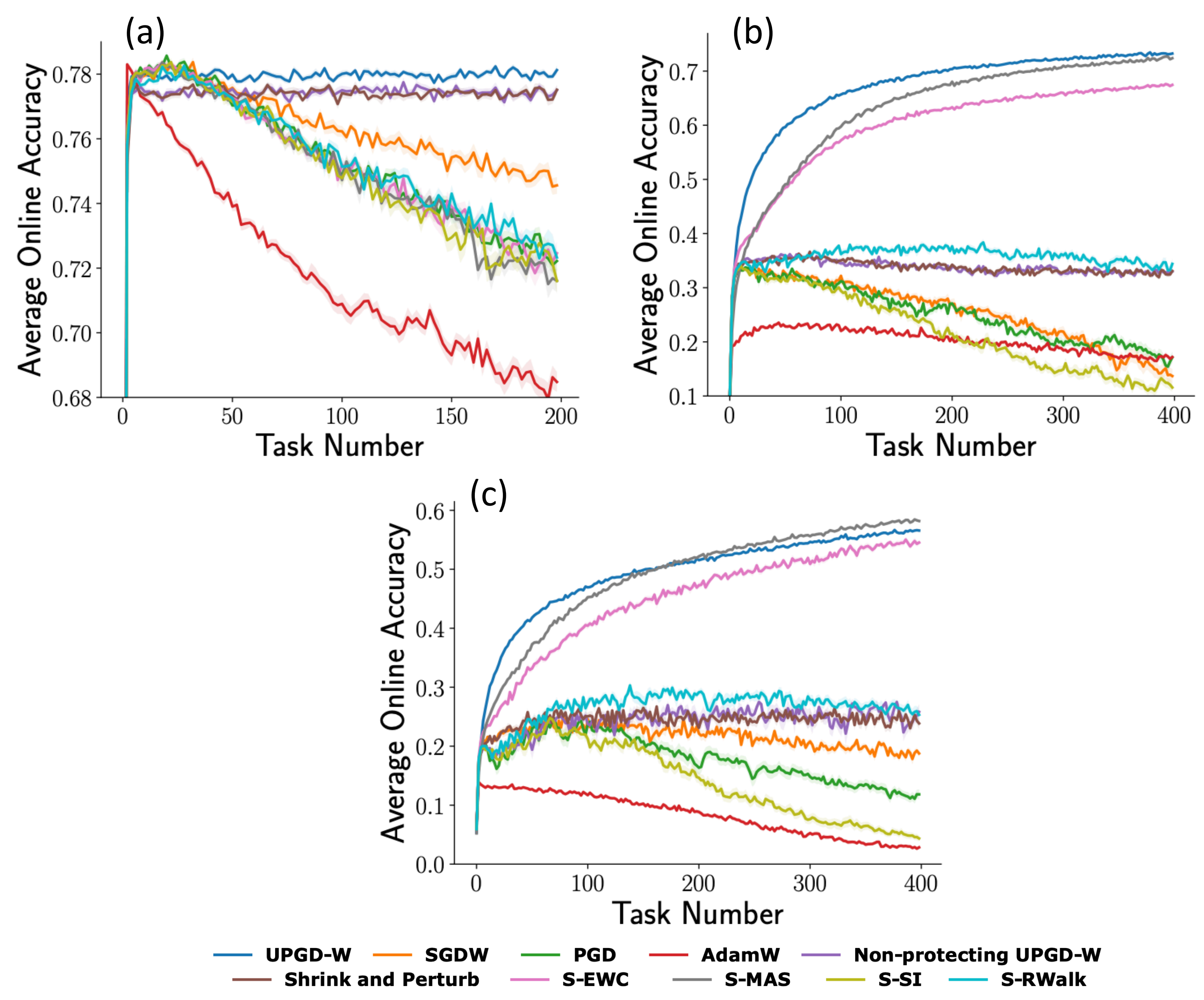
- We write an update equation that that performs gradient-based learning guided by utility-based information.
- The utility is used as a **gate** for the gradients to prevent large updates to already useful weights, *addressing forgetting,* and help perturb unuseful weights which become difficult to change through gradients, addressing loss of plasticity. UPGD rule is given by:

$$W_{l,i,j} \leftarrow W_{l,i,j} - \alpha \left(\frac{\partial \mathscr{L}}{\partial w_{l,i,j}} + \xi \right) \left(1 - \bar{U}_{l,i,j} \right)$$

where $\xi \sim \mathcal{N}(0,1)$ is the noise sample, α is the step size, and $\bar{U}_{l,i,j} \in [0,1]$ is a scaled utility. For $\bar{U}_{l,i,j} = 1$, the weight remains unaltered even by gradient descent, and for $\bar{U}_{l,i,j} = 0$, get updated by both perturbation and gradient descent.

- We scale the utility by the maximum utility of all weights (e.g., instantaneous or trace), which is given by $\bar{U}_{l,i,j} = \phi(U_{l,i,j}/\eta)$, where ϕ is the scaling function, for which we use sigmoid.

Results and Discussion



Performance of methods on Input-permuted MNIST (a), Label-permuted EMNIST (b), and Label-permuted mini-ImageNet (c).

- *Input-Permuted* MNIST is a problem where only loss of plasticity is present. Learned representations in a task become irrelevant to the new task, so the it's a suitable problem to study loss of plasticity.
- UPGD has no decaying performance, mitigating loss of plasticity.
- *Label-permuted EMNIST/mini-ImageNet* are problems where loss of plasticity and catastrophic forgetting are present. Learned representations in one task are still useful for the next task, making these problems suitable for studying catastrophic forgetting.
- UPGD continually improves its online accuracy, showing that it improves its representations by protecting useful weights from large updates, mitigating catastrophic forgetting.

Conclusion

UPGD mitigates continual learning issues in streaming learning through protecting useful weights and perturbing less useful ones.

{mohamedelsayed, armahmood}@ualberta.ca