Workshop at the 7th Symposium on Advances in Approximate Bayesian Inference (non-archival), 2025 1–9

Compact Memory for K-prior Based Continual Learning

Yohan Jung* YOHAN.JUNG@RIKEN.JP
Hyungi Lee*† LHK2708@KAIST.AC.KR
Wenlong Chen*† WENLONG.CHEN21@IMPERIAL.AC.UK
Thomas Möllenhoff THOMAS.MOELLENHOFF@RIKEN.JP
Yingzhen Li YINGZHEN.LI@IMPERIAL.AC.UK
Juho Lee JUHOLEE@KAIST.AC.KR
Mohammad Emtiyaz Khan EMTIYAZ.KHAN@RIKEN.JP

Abstract

Despite recent progress, continual lifelong learning cannot yet match the performance of batch learning. This is partly because the regularization methods used for continual learning are not as effective as the stochastic gradients sampled from the whole data. Replay methods reconstruct past gradients and can work better but the memory-buffer size can grow quite large with the number of tasks and make the method slow. Here, we propose a new method to build a compact memory to accurately reconstruct the past gradients. We use the framework by Khan and Swaroop (2021) who prove the existence of optimal memory to perfectly reconstruct the gradients. We show that, for linear regression, the optimal memory is obtained by Hessian matching and use this to propose an extension to logistic regression by using the probabilistic PCA method. We confirm our findings on small-scale classification problems. Overall, we hope to encourage future research on compact memory for continual learning.

1. Introduction

Continual lifelong learning aims to enable continual acquisition of new skills while retaining and reusing past knowledge, but this requires a delicate balance of the past and future (Mermillod et al., 2013) in order to avoid harmful interference (Sutton, 1986). In other words, future learning must be regularized by the past knowledge otherwise the past can be forgotten (Kirkpatrick et al., 2017). Designing a regularization method is crucial but, despite a lot of recent progress, there is no satisfactory solution yet. Regularization towards the past is a fundamental principle, appearing in optimization (Bertsekas, 2011), online learning (Cesa-Bianchi and Lugosi, 2006; Shalev-Shwartz et al., 2012) and Bayesian-like updating (Hoeven et al., 2018). Many such regularization-based methods have been adapted to continual learning for neural networks (Kirkpatrick et al., 2017; Li and Hoiem, 2017; Lee et al., 2017; Zenke et al., 2017; Ebrahimi et al., 2018; Ritter et al., 2018). However, these approaches appear to perform much worse when compared to memory or output-based rehearsal strategies (Rebuffi et al., 2001; Nguyen et al., 2017; Titsias et al., 2019; Pan et al., 2020; Buzzega et al., 2020). While sophisticated regularization-based strategies can mitigate forgetting to

∗. Equal contribution.
†. This was conducted during the RIKEN ABI team internship program.

some extent (Daxberger et al., 2023), their performance still lags far behind simple stochastic gradient batch learning which has access to all the data at once. Addressing this gap is an important problem to not only make progress in continual lifelong learning, but also to reduce the costs and environmental impacts of AI (Paleyes et al., 2022).

One way to obtain performance similar to batch learning is to design regularizers to accurately reconstruct past gradients. This principle is proposed by Khan and Swaroop (2021) where they design a regularizer, referred to as K-prior $\mathcal{K}_t(\boldsymbol{\theta})$, by using the parameter $\boldsymbol{\theta}_t$ after task t and a subset $\mathcal{M}_t$ (or memory) of all past data $\mathcal{D}_{1:t}$. The form of the prior and memory $\mathcal{M}_t$ is chosen such that the $\mathcal{K}_t(\boldsymbol{\theta})$ reconstructs the gradient of the batch learning,

$$\nabla \mathcal{K}_t(\boldsymbol{\theta}) \approx \sum_{j=0}^{t} \nabla \bar{\ell}_t(\boldsymbol{\theta}), \tag{1}$$

where $\bar{\ell}_t$ is the loss function over the dataset $\mathcal{D}_t$ of task t, and $\bar{\ell}_0$ is the regularizer. Within K-prior, we can combine weight-regularization, function-regularization, and experience replay to ensure a good reconstruction of the gradient (Daxberger et al., 2023).

A major issue with K-prior is that the memory size can be large because $\mathcal{M}_t$ in practice is chosen to be a subset of the data. Moreover, if we define the memory as a subset, it remains unclear which specific subsets should be chosen to minimize the gradient reconstruction error while preserving compactness. This is not an issue of the framework that allows arbitrary inputs to be employed as memory (even those without labels), but this property has not been exploited yet to build a more compact memory. In fact, Khan and Swaroop (2021, App. 1) theoretically proves the existence of such a compact memory for logistic regression, but achieving such memory is difficult in practice.

Here, we propose to build a compact representation in the feature space (instead of the input space). The key idea is to start with linear regression and show that the problem can be solved by Hessian Matching. In such a case, perfect reconstruction of the gradient is possible by simply looking at the non-zero singular values of the feature matrix. We generalize the method to logistic regression by posing it as a maximum likelihood estimation of Probabilistic Principal Component Analysis (PPCA; Tipping and Bishop, 1999). In our formulation the reconstruction is not perfect, but a near-optimal solution can still be found by increasing the memory size a bit further. We present some preliminary results on small-scale classification problems in support of our results. The hope of this work is to promote future work on compact memory for continual lifelong learning.

2. Continual learning with gradient reconstruction

Continual lifelong learning requires a delicate balance of past and future knowledge to enable continual acquisition of new knowledge without interfering with old knowledge. One of the simplest settings to study this problem is the supervised learning case where each task data $\mathcal{D}_t$ consists of N_t input-output pairs $(\mathbf{x}_i, \mathbf{y}_i)$. The goal can be to learn a parametric model $\mathbf{f}_i^{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^D$ to predict the i'th output y_i from input x_i.

In this setup, we start with a parameter $\boldsymbol{\theta}_0$ and, after seeing $\mathcal{D}_1$, update to get $\boldsymbol{\theta}_1$, and continue to do so to get a sequence of $\boldsymbol{\theta}_t$. In this online setting, we assume that, when updating $\boldsymbol{\theta}_{t+1}$, we have access to all dataset $\mathcal{D}_{t+1}$ of current task and the old parameters $\boldsymbol{\theta}_t$. The old parameter can be used in a weight-regularizer but we could also simply use a

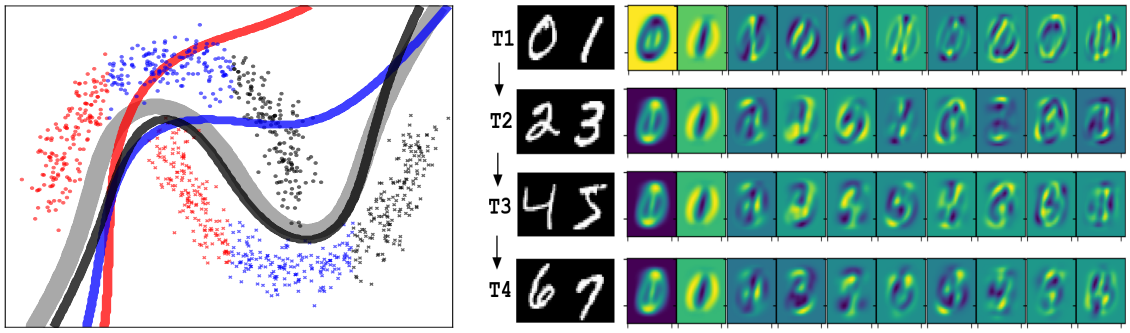

Figure 1: **Left figure** illustrates the results of two-moon classification with 3 tasks (shown with red, blue, and black) where the classifier gets close results to the truth (the gray line) when we use a memory of just 7 points/tasks. **Right figure** visualizes the memories for Split-MNIST as more tasks are added. Each row shows the eigen-images sorted in decreasing eigenvalues from left to right. We see that the first 2-3 images are learned quickly and we only observe variational in images with smaller eigenvalues (4th image onward).

memory buffer $\mathcal{M}_t$ of old data $\mathcal{D}_{1:t}$ to perform experience replay. These two options are shown below in the left and right respectively,

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \bar{\ell}_{t+1}(\boldsymbol{\theta}) + \tfrac{1}{2}\delta\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2, \qquad \boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \bar{\ell}_{t+1}(\boldsymbol{\theta}) + \sum_{j\in\mathcal{M}_t} \bar{\ell}_j(\boldsymbol{\theta}), \tag{2}$$

where $\delta > 0$ is a coefficient controlling the strength of the weight regularizer. An intermediate option is to use function regularization where we match the output of the current output $\mathbf{f}_j^{\boldsymbol{\theta}}$ to the old output $\mathbf{f}_j^{\boldsymbol{\theta}_t}$. These are complementary methods that can be combined, but have yet to match the performance of batch learning.

One way to obtain performance similar to batch learning is to design regularizers to accurately reconstruct the gradient over the past data, as shown in Eq.(6). This principle is proposed by Khan and Swaroop (2021) who design a regularizer called Knowledge-Adaptation prior (K-prior) which aims to reconstruct the past gradients. For instance, consider a linear-regression problem with predictors $f_i^{\boldsymbol{\theta}} = \boldsymbol{\phi}_i^\top \boldsymbol{\theta}$ for a feature map $\boldsymbol{\phi}_i \in \mathbb{R}^P$ of the input $\mathbf{x}_i$. Then, we can show that the following K-prior recovers the past gradients

$$\mathcal{K}_t(\boldsymbol{\theta}) = \tfrac{1}{2}\delta\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2 + \sum_{j\in\mathcal{M}_t} \tfrac{1}{2}\left(f_j^{\boldsymbol{\theta}} - f_j^{\boldsymbol{\theta}_t}\right)^2, \tag{3}$$

arbitrarily accurately as $\mathcal{M}_t \to \mathcal{D}_{1:t}$ (the regularizer is assumed to be $\frac{\delta}{2}\|\boldsymbol{\theta}\|^2$). Essentially, K-priors suggest regularizing both the parameters and function outputs. Khan and Swaroop (2021) generalize this to a generic loss function (for example, logistic regression), and present extensions covering many existing adaptation strategies as special cases of this strategy, including Elastic Weight-Consolidation (EWC), Knowledge Distillation, Support Vector Machines, Sparse Gaussian Process, etc. Daxberger et al. (2023) further extend to include Experience Replay and obtain good results on ImageNet dataset.

One major issue with K-prior is that the memory $\mathcal{M}_t$ in practice is chosen as a subset of data, selected in a heuristic fashion. The memory size can be larger than desired,

for instance, Daxberger et al. (2023) require to store 7% of the ImageNet dataset, which increases the overhead significantly when compared to the weight-regularization method. The K-prior framework itself does not require the memory inputs to be restricted to the data because it can regularize the outputs using arbitrary inputs without relying on their labels. Khan and Swaroop (2021) argue that this is an advantage of the method which can be exploited to reduce the memory size, however, there is no existing work on this so far.

Most importantly, Khan and Swaroop (2021, App. A) shows the existence of an optimal memory size for logistic regression case. Here, the memory consists of K_t vectors $\mathbf{u}_i$ where K_t is the rank of the input matrix $\mathbf{X}_{1:t}$ (the matrix formed by all the inputs in $\mathcal{D}_{1:t}$). They prove that a perfect gradient reconstruction is possible by simply matching K_t predictors $f_k^{\boldsymbol{\theta}} = \mathbf{u}_k^\top \boldsymbol{\theta}$. Specifically, there exist scalars $w_{k|t}^*$ such that the following K-prior perfectly recovers the gradients for the cross-entropy loss $\ell(y_i, \sigma(f_i^{\boldsymbol{\theta}}))$,

$$\mathcal{K}_t(\boldsymbol{\theta}) = \tfrac{1}{2}\delta\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2 + \sum_{k=1}^{K_t} w_{k|t}^* \, \ell\left(\sigma(f_k^{\boldsymbol{\theta}}),\, \sigma(f_k^{\boldsymbol{\theta}_t})\right). \tag{4}$$

In practice, it is difficult to find $w_{k|t}^*$ exactly, but we aim to learn the vectors $\mathbf{u}_k$ to get a good reconstruction while keeping the memory size K_t small. The goal of this work is to propose a method to achieve this.

3. Compact memory for K-prior

We propose to build a compact memory in the space of features $\boldsymbol{\phi}_i \in \mathbb{R}^P$, which is in contrast to the methods that use a subset of data examples. Our goal is to maintain a matrix $\mathbf{U}_t \in \mathbb{R}^{P \times K_t}$ with K_t columns, each denoted by $\mathbf{u}_{k|t} \in \mathbb{R}^P$:

$$\text{Memory Set: } \mathbf{U}_t = \begin{bmatrix} \mathbf{u}_{1|t} & \mathbf{u}_{2|t} & \ldots\ldots & \mathbf{u}_{K_t|t} \end{bmatrix}$$

At each step, we update it to find a new $\mathbf{U}_{t+1}$ with K_{t+1} columns denoted by $\mathbf{u}_{k|t+1}$. We will slowly grow K_t, although this is not necessary. Assuming that $\mathbf{U}_t$ can accurately reconstruct the gradient at task t, we will estimate $\mathbf{U}_{t+1}$ such that it maintains this property at task $t+1$. We will now demonstrate for linear regression that this task can be accomplished by a simple Hessian matching, and then we will extend this idea to logistic regression.

3.1. Compact memory for linear regression via Hessian matching

For simplicity, we assume the regularizer $\ell_0(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2$, but other convex regularizers can also be used. Now, suppose that we have a set $\mathbf{U}_t$ which can perfectly reconstruct the gradients over past data as shown in Eq.(6). If this is true, then we can safely recover the solution that considers the data up to $t+1$ at once, i.e. *batch solution*, by solving

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \underbrace{\sum_{j=1}^{N_{t+1}} \tfrac{1}{2}\left(f_j^{\boldsymbol{\theta}} - y_j\right)^2}_{=\bar{\ell}_{t+1}(\boldsymbol{\theta})} + \underbrace{\tfrac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^2 + \sum_{k=1}^{K_t} \tfrac{1}{2}\left(f_k^{\boldsymbol{\theta}} - f_k^{\boldsymbol{\theta}_t}\right)^2}_{=\mathcal{K}_t(\boldsymbol{\theta};\mathbf{U}_t)}. \tag{5}$$

The first term here is the loss over the task $t+1$ and the rest of the terms belong to the K-prior, which is similar to Eq.(4), but now customized for the squared loss instead of

cross-entropy loss. For regression, $w^*_{k|t} = 1$ in Eq.(4) naturally arises by changing the form of the log of Gaussian likelihood.

Given such $\mathbf{U}_t$, our goal is to find $\mathbf{U}_{t+1}$, which also satisfies the perfect gradient reconstruction property, as shown in Eq.(6). However, because the old K-prior $\mathcal{K}_t(\boldsymbol{\theta}; \mathbf{U}_t)$ also recovers the gradients perfectly, we can use the recursion to write the following

$$\nabla \mathcal{K}_{t+1}(\boldsymbol{\theta}; \mathbf{U}_{t+1}) = \nabla \bar{\ell}_{t+1}(\boldsymbol{\theta}) + \nabla \mathcal{K}_t(\boldsymbol{\theta}; \mathbf{U}_t). \tag{6}$$

The size of the gradients is $P \times 1$, so it is not possible to infer $\mathbf{U}_{t+1}$ from here whose size is $P \times K_{t+1}$. However, if we differentiate both sides again, then we get an equation that has enough degrees of freedom to determine $\mathbf{U}_{t+1}$,

$$\mathbf{U}_{t+1}\mathbf{U}_{t+1}^\top = \boldsymbol{\Phi}_{t+1}\boldsymbol{\Phi}_{t+1}^\top + \mathbf{U}_t\mathbf{U}_t^\top \tag{7}$$

where $\boldsymbol{\Phi}_{t+1} \in \mathbb{R}^{P \times N_{t+1}}$ is the feature matrix containing columns $\boldsymbol{\phi}_j$ corresponding to all inputs j in $\mathcal{D}_{t+1}$. Essentially, for linear regression, gradient matching implies Hessian matching as well and this equation is independent of $\boldsymbol{\theta}$. We propose to use this to estimate $\mathbf{U}_{t+1}$.

Eq.(7) implies that $\mathbf{U}_{t+1}$ should be a low-rank reconstruction of a dataset formed by concatenating $\mathbf{U}_t$ and $\boldsymbol{\phi}_{t+1}$, and thus can be computed by performing SVD on the concatenated dataset and taking the first K_{t+1} columns (only those with non-zero singular values). A numerically easier alternative is the Expectation Maximization (EM) procedure for Probabilistic PCA by Tipping and Bishop (1999). In this procedure, we try to estimate the latent $\mathbf{Z}_{t+1} \in \mathbb{R}^{K_{t+1} \times (N_t + K_t)}$ to be able to predict $\mathbf{U}_t$ and $\boldsymbol{\Phi}_{t+1}$ from the future $\mathbf{U}_{t+1}$,

$$[\mathbf{U}_t,\ \boldsymbol{\Phi}_{t+1}] \approx \mathbf{U}_{t+1}\mathbf{Z}_{t+1}$$

where each column of $\mathbf{Z}_{t+1}$ is drawn from a standard normal. This has a simple interpretation that Hessian matching suggests using the memory set that can predict both past and future features well. The EM procedure is also easy and cheap to implement.

3.2. Extension to generalized linear model

The procedure above can be extended to any generalized linear model. We will only discuss logistic regression and leave the general case aside due to space limitation. We will use the K-prior shown in Eq.(4) but with unknown coefficients $w_{k|t}$. Our goal is to estimate both coefficients $w_{k|t+1}$ and $\mathbf{U}_{t+1}$, assuming that good estimates of $w_{k|t}$ and $\mathbf{U}_t$, are given.

Even for generalized linear models, the principle of Hessian matching remains unchanged, resulting in an equation structurally similar to that of the linear regression case, except for additional diagonal terms as below:

$$\mathbf{U}_{t+1}\mathbf{W}_{t+1}\mathbf{U}_{t+1}^\top = \boldsymbol{\Phi}_{t+1}\mathbf{B}_{t+1}\boldsymbol{\Phi}_{t+1}^\top + \mathbf{U}_t\mathbf{W}_t\mathbf{U}_t^\top \tag{8}$$

where $\mathbf{W}_{t+1}, \mathbf{W}_t$, and $\mathbf{B}_{t+1}$ are the diagonal matrices whose respective diagonal entries are

$$\mathbf{W}^{kk}_{t+1} = w_{k|t+1} \cdot \sigma'(f^{\boldsymbol{\theta}}_{k|t+1}), \qquad \mathbf{W}^{kk}_{t} = w_{k|t} \cdot \sigma'(f^{\boldsymbol{\theta}}_{k|t}), \qquad \mathbf{B}^{jj}_{t} = \sigma'(f^{\boldsymbol{\theta}}_{j}), \tag{9}$$

where $\sigma'(f)$ is the derivative of the sigmoid at f. The only difficulty now is that the diagonal matrices depend on $\boldsymbol{\theta}$. A reasonable choice to do Hessian matching at $\boldsymbol{\theta}_{t+1}$. The EM algorithm, used for regression, can be also easily generalized to solve for both $\mathbf{u}_{k|t+1}$ and $w_{k|t+1}$. Essentially, instead of estimating $\mathbf{u}_{k|t+1}$ done in regression, we can estimate $\tilde{\mathbf{u}}_{k|t+1} = w^{1/2}_{k|t+1}\mathbf{U}_{t+1}$ directly. Then, we set $w_{k|t+1}$ to be the norm of $\tilde{\mathbf{u}}_{k|t+1}$ and set $\mathbf{u}_{k|t+1}$ to be with norm 1. The final algorithm is summarized in Algorithm 1.

4. Experimental results

4.1. Multi output linear regression

We demonstrate whether our EM algorithm for hessian matching achieves compact memory. To this end, we conduct a continual multi-label regression task on **Split-MNIST** (Zenke et al., 2017), where the MNIST dataset is divided into five subsets, each corresponding to a binary classification task with label pairs (0,1), (2,3), (4,5), (6,7), and (8,9). To formulate the binary classification task as a regression problem, we map each label $k \in \{0, \dots, 9\}$ into a vector $\mathbf{1}_k - \frac{1}{10}\mathbf{1}$ where $\mathbf{1}_k \in \mathbb{R}^{10}$ is a one-hot vector with only the kth component set to one and all others set to zero, and $\mathbf{1} \in \mathbb{R}^{10}$ is a vector of ones. For the feature map, we use $\phi(x) = x \in \mathbb{R}^{784}$, i.e., the vectorized pixel intensities of an image. For the memory, we use constant memory size per task and accumulate its size as the model trains on new tasks. After training, we measure the classification accuracy using the entire test dataset for evaluation.

Results. The figure on the right in Figure 1 illustrates how the top-10 eigenvectors of Hessian $\mathbf{U}_{t+1}\mathbf{U}_{t+1}^\top$ in Eq.(7), obtained by our EM algorithm, change as our memory is updated after training each task. In each row, the eigenvectors are placed based on their eigenvalues (left: high eigenvalue). This result implies that new features, looking as mixture of (2,3), (4,5), and (6,7), appear in each row whenever the memories are updated. Figure 2 compares our EM method with the SVD approach over three seeds across varying memory sizes; the SVD approach obtains $\mathbf{U}_{t+1}$ by applying SVD to the right side of Eq.(7), and the joint training denotes that all tasks are trained without sequentially updating the memory. This shows that our EM method is more effective than the SVD approach when using a smaller number of memory.

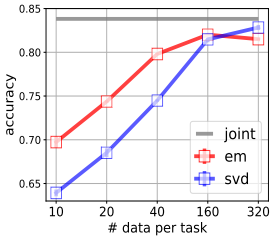

Figure 2: EM vs SVD

4.2. Logistic regression

We aim to demonstrate that our method can find a compact memory for logistic regression. To this end, we conduct a continual binary task on the **two-moon dataset**, where the dataset is split into three binary tasks according to the inputs, and the model is sequentially trained for a given task with BCE loss. For a feature map, We use $\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3] \in \mathbb{R}^7$. We set 7 memories per task and accumulate the size of memories ($7 \rightarrow 14 \rightarrow 21$) as the task proceeds.

Results. The left figure in Figure 1 shows how the classifier's decision boundary changes as it trains on each task sequentially (red → blue → black). We confirm that the decision boundary gets close to its oracle (the gray line), obtained by joint training, in the end.

5. Conclusion

In this work, we note that hessian matching builds a compact memory to accurately reconstruct the past gradients in linear regression and generalize this approach to logistic regression by using the EM algorithm of the probabilistic PCA method. Empirically, we demonstrate that our approach works on small-scale classification tasks. In future work, we will extend our method for multi-label classification and make it feasible for neural networks.

References

Dimitri P Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. 2011.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. Advances in neural information processing systems, 33:15920–15930, 2020.

Nicolo Cesa-Bianchi and Gábor Lugosi. Prediction, learning, and games. Cambridge university press, 2006.

Erik Daxberger, Siddharth Swaroop, Kazuki Osawa, Rio Yokota, Richard E Turner, José Miguel Hernández-Lobato, and Mohammad Emtiyaz Khan. Improving continual learning by accurate gradient reconstructions of the past. Transactions on Machine Learning Research, 2023.

Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning in bayesian neural networks–extended abstract. In Proc. IEEE Conf. Comput. Vis. Pattern Recognition (CVPR), 2018.

Dirk Hoeven, Tim Erven, and Wojciech Kotłowski. The many faces of exponential weights in online learning. In Conference On Learning Theory, pages 2067–2092. PMLR, 2018.

Mohammad Emtiyaz E Khan and Siddharth Swaroop. Knowledge-adaptation priors. Advances in neural information processing systems, 34:19757–19770, 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526, 2017.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. Advances in neural information processing systems, 30, 2017.

Zhizhong Li and Derek Hoiem. Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12):2935–2947, 2017.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. arXiv preprint arXiv:1710.10628, 2017.

Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. Challenges in deploying machine learning: a survey of case studies. ACM Computing Surveys, 55(6):1–29, 2022.

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. Incremental classifier and representation learning. In Conference on Computer Vision and Pattern Recognition (CVPR), 2001.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. Advances in Neural Information Processing Systems, 2018.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. Foundations and Trends® in Machine Learning, 4(2):107–194, 2012.

Richard S Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. In Proceedings of the Annual Meeting of the Cognitive Science Society, volume 8, 1986.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. Journal of the Royal Statistical Society Series B: Statistical Methodology, 61(3):611–622, 1999.

Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. arXiv preprint arXiv:1901.11356, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Adam: a method for stochastic optimization. In International Conference on Machine Learning (ICML), 2017.

Appendix A. Appendix

A.1. Hessian matching for logistic regression via EM algorithm

Algorithm 1 Learning a memory in logistic regression through PPCA-EM algorithm.

Require: Task $t+1$ dataset $\mathcal{D}_{t+1}$, model parameter $\boldsymbol{\theta}_t$, regularizer parameter of K-prior δ memory $\mathbf{U}_t$, and weight $\mathbf{W}_t$ trained

// **K-prior:** Train $\boldsymbol{\theta}_{t+1}$ for task $\mathcal{D}_{t+1}$ with memory $\mathbf{U}_t$ and weight $\mathbf{W}_t$

$\boldsymbol{\theta}_{t+1} \leftarrow \text{argmin}_{\boldsymbol{\theta}}\ \bar{\ell}_{t+1}(\boldsymbol{\theta}) + \mathcal{K}_t(\boldsymbol{\theta})$ in eq. (4)

// **EM:** Train memory $\{\mathbf{u}_{k|t+1}\}$, and weight $\{w_{k|t+1}\}$ trained at task $t+1$

Define $\mathbf{U}_{t+1} = [\mathbf{u}_{1|t+1}, \ldots, \mathbf{u}_{K_{t+1}|t+1}]$ with $\mathbf{u}_{t|k+1}$ initialized from $\mathcal{D}_{t+1}$

Define $\mathbf{W}_{t+1} = \text{diag}(\{w_{k|t+1} \cdot \sigma'(f^{\boldsymbol{\theta}}_{k|t+1})\})$ with $w_{k|t+1} = 1$

Define $\mathbf{B}_{t+1} = \text{diag}(\{\sigma'(f^{\boldsymbol{\theta}}_j)\})$

Define $\widehat{\mathbf{U}}_t = [\boldsymbol{\Phi}_{t+1}\mathbf{B}^{1/2}_{t+1}\ ;\ \mathbf{U}_t\mathbf{W}^{1/2}_t]$ // column concatenation

while not converged **do**

  $\widetilde{\mathbf{U}}_{t+1} \leftarrow \mathbf{U}_{t+1}\mathbf{W}^{1/2}_{t+1}$

  $\mathbf{D}_{t+1} \leftarrow \text{diag}\{\sigma'(f^{\boldsymbol{\theta}}_{k|t+1}\}$ // set non-linear term of $\sigma'(f^{\boldsymbol{\theta}}_{k|t+1})$ using updated $\mathbf{U}_{t+1}$.

  while not converged **do**

    $\mathbf{S} \leftarrow \frac{1}{\sigma^2}\widetilde{\mathbf{U}}^\top_{t+1}\widetilde{\mathbf{U}}_{t+1} + \mathbf{I}$

    $\mathbf{M} \leftarrow \frac{1}{\sigma^2}\mathbf{S}^{-1}\widetilde{\mathbf{U}}^\top_{t+1}\widehat{\mathbf{U}}_t$

    $\mathbf{A} \leftarrow \widehat{\mathbf{U}}_t\mathbf{M}^\top$

    $\mathbf{B} \leftarrow (\mathbf{S}^{-1} + \mathbf{M}\mathbf{M}^\top)^{-1}$

    $\widetilde{\mathbf{U}}_{t+1} \leftarrow \mathbf{A}\mathbf{B}$

  end while

  $\mathbf{Z}_{t+1} \leftarrow \widetilde{\mathbf{U}}_{t+1}\mathbf{D}^{-1/2}_{t+1}$

  $\mathbf{U}_{t+1} \leftarrow \mathbf{Z}_{t+1}\text{diag}([\mathbf{Z}_{t+1}]_{\text{cols}})^{-1}$, // column norm $[\mathbf{Z}_{t+1}]_{\text{cols}} := \{\|[\mathbf{Z}_{t+1}]_{:,k}\|\}$

  $\mathbf{W}_{t+1} \leftarrow \text{diag}(\{w_{k|t+1} \cdot \sigma'(f^{\boldsymbol{\theta}}_{k|t+1})\})$ with $w_{k|t+1} = [\mathbf{Z}_{t+1}]^2_{:,k}$

end while

Set $[\mathbf{u}_{1|t+1}; ..; \mathbf{u}_{K_{t+1}|t+1}] = \mathbf{U}_{t+1}$ and $[w^{1/2}_{1|t+1}, .., w^{1/2}_{K_{t+1}|t+1}] = [\mathbf{Z}_{t+1}]_{\text{cols}}$

Algorithm 1 describes how to learn the model parameter $\boldsymbol{\theta}_{t+1}$ by K-prior and learn the memory $\mathbf{U}_{t+1}$ using the PPCA-EM algorithm. Once $\mathbf{U}_{t+1}$ and $\mathbf{W}_{t+1}$ satisfying

$$\mathbf{U}_{t+1}\mathbf{W}_{t+1}\mathbf{U}^\top_{t+1} \approx \boldsymbol{\Phi}_{t+1}\mathbf{B}_{t+1}\boldsymbol{\Phi}^\top_{t+1} + \mathbf{U}_t\mathbf{W}_t\mathbf{U}^\top_t \tag{10}$$

is obtained where $\mathbf{W}_{t+1}, \mathbf{W}_t$, and $\mathbf{B}_{t+1}$ are the diagonal matrices whose respective diagonal entries are

$$\mathbf{W}^{kk}_{t+1} = w_{k|t+1} \cdot \sigma'(f^{\boldsymbol{\theta}}_{k|t+1}), \qquad \mathbf{W}^{kk}_t = w_{k|t} \cdot \sigma'(f^{\boldsymbol{\theta}}_{k|t}), \qquad \mathbf{B}^{jj}_t = \sigma'(f^{\boldsymbol{\theta}}_j), \tag{11}$$

with $\sigma'(f^{\boldsymbol{\theta}}_{k|t+1}) = \sigma(\mathbf{u}^\top_{k|t+1}\boldsymbol{\theta}_{t+1})(1-\sigma(\mathbf{u}^\top_{k|t+1}\boldsymbol{\theta}_{t+1}))$, and $\sigma'(f^{\boldsymbol{\theta}}_j) = \sigma(\phi^\top_j\boldsymbol{\theta}_{t+1})(1-\sigma(\phi^\top_j\boldsymbol{\theta}_{t+1}))$, we can employ them for K-prior regularizer to be used for learning new task.