A Theoretical Analysis of Catastrophic Forgetting through the NTK Overlap Matrix

Thang Doan 1 2    Mehdi Bennani 3    Bogdan Mazoure 1 2
Guillaume Rabusseau 2 4    Pierre Alquier 5

Abstract

Continual learning (CL) is a setting in which an agent has to learn from an incoming stream of data during its entire lifetime. Although major advances have been made in the field, one recurring problem which remains unsolved is that of Catastrophic Forgetting (CF). While the issue has been extensively studied empirically, little attention has been paid from a theoretical angle. In this paper, we show that the impact of CF increases as two tasks increasingly align. We introduce a measure of task similarity called the NTK overlap matrix which is at the core of CF. We analyze common projected gradient algorithms and demonstrate how they mitigate forgetting. Then, we propose a variant of Orthogonal Gradient Descent (OGD) which leverages structure of the data through Principal Component Analysis (PCA). Experiments support our theoretical findings and show how our method can help reduce CF on classical CL datasets.

1 Introduction

Continual learning (CL) or lifelong learning [Thrun 1995, Chen and Liu 2018] has been one of the most important milestones on the path to building artificial general intelligence [Silver 2011]. This setting refers to learning from an incoming stream of data, as well as leveraging previous knowledge for future tasks (through forward-backward transfer [Lopez-Paz and Ranzato 2017]). While the topic has seen increasing interest in the past years [De Lange et al. 2019, Parisi et al. 2019] and a number of sophisticated methods have been developed [Kirkpatrick et al. 2017, Lopez-Paz and Ranzato 2017, Chaudhry et al. 2018, Aljundi et al. 2019b], a yet unsolved central challenge remains: Catastrophic Forgetting (CF) [Goodfellow et al. 2013, McCloskey and Cohen 1989].

CF occurs when past solutions degrade while learning from new incoming tasks according to non-stationary distributions. Previous work either investigated this phenomenon empirically at different granularity levels (task level [Nguyen et al. 2019], neural network representations level [Ramasesh et al. 2020]), or proposed a quantitative metric [Farquhar and Gal 2018, Kemker et al. 2017, Nguyen et al. 2020].

Despite the vast set of existing works on CF, there is still few theoretical works studying this major topic. Recently, Bennani et al. [2020] propose a framework to study Continual Learning in the NTK regime then derive generalization guarantees of CL under the Neural Tangent Kernel [Jacot et al. 2018, NTK] for Orthogonal Gradient Descent [Lopez-Paz and Ranzato 2017]. Following on this work, we propose a theoretical analysis of Catastrophic Forgetting for a family of projection algorithms including OGD, GEM [Lopez-Paz and Ranzato 2017]. Our contributions can be summarized as follows:

- We provide a general definition of Catastrophic Forgetting, and examine the special case of CF under the Neural Tangent Kernel (NTK) regime. Our definition leverages the similarity between the source and target task.
- We derive the expression of the forgetting error for a family of orthogonal projection methods based on the NTK overlap matrix. This matrix reduces

1McGill University 2Mila 3Aqemia 4Université de Montréal 5RIKEN AIP
corresponding author: thang.doan@mail.mcgill.ca
2020] studied the designed
2018]. In [Pan et al., 2020], the authors propose a
2017, Farajtabar et al., 2020] or parameters isolations
and studied its impact on the forgetting. Mirzadeh
nomenon at the neural network layers level. [Xie et al.,
forgetting, [Ramasesh et al., 2020] analyzed this phe-
empirically studied the impact of tasks similarity on the
CF [Toneva et al., 2018]. While Nguyen et al. [2019]
Recently, a lot of efforts has been put toward dissecting
Forgetting, its underlying theory remains unclear.
Defying Catastrophic Forgetting [McCloskey and Co-
1989] has always been an important challenge
for CL under the NTK regime for the infinite memory
cause. Our work relaxes this constraint to the finite memory
case, which is more applicable in the empirical
setting.
3 Preliminaries
3.1 Notations
We use $\| \cdot \|_2$ to denote the Euclidian norm of a vector
or the spectral norm of a matrix. We use $\langle \cdot, \cdot \rangle$ for the
Euclidean dot product, and $\langle \cdot, \cdot \rangle_H$ the dot product in
the Hilbert space $\mathcal{H}$. We index the task ID by $\tau$. Learn-
able parameter are denoted $\omega$ and when indexing as
$\omega_\tau$ correspond to the training during task $\tau$. Moreover
$\star$ represents the variable at the end of a given task,
\text{i.e} $w_\star^\tau$ represents the learned parameters at the end of
task $\tau$.
We denote $\mathbb{N}$ the set of natural numbers, $\mathbb{R}$ the space
of real numbers and $\mathbb{N}^*$ for the set $\mathbb{N} \setminus \{0\}$.
3.2 Continual Learning
Let $\mathcal{X}$ be some feature space of interest (we take $\mathcal{X} = \mathbb{R}^p$), and let $\mathcal{Y}$ be the space of labels (we let $\mathcal{Y} = \mathbb{R}$, but $\mathcal{Y} = \Delta^K$ can be used for classification). In CL, we
receive a stream of supervised learning tasks $T_\tau$, $\tau \in [T]$
where $T_\tau = \{x_{\tau,j}, y_{\tau,j}\}_{j=1}^{n_\tau}$ with $T \in \mathbb{N}^*$. While $X^\star \in \mathbb{R}^{n_\star \times p}$ ($p$ being the number of features) represents the
dataset of task $T_\tau$ and $x_{\tau,j}, j = 1, ..., n_\tau \in \mathcal{X}$ is a sample
with its corresponding label $y_{\tau,j} \in \mathcal{Y}$. The goal is to
learn a predictor $f_\omega : \mathcal{X} \times T \rightarrow \mathcal{Y}$ with $\omega \in \mathbb{R}^p$
the parameters that will perform a prediction as accurate
as possible. In the framework of CL, one cannot recover
samples from previous tasks unless storing them in a
memory buffer [Lopez-Paz and Ranzato] 2017 [Parisit
et al. 2019].
\footnote{\Delta^K$ denotes the vertices of the probability simplex of
dimension $K$}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig.png}
\caption{Unlike SGD, the projection methods reduce
the forgetting by projecting the source and target
tasks on a residual subspace.}
\end{figure}
3.3 NTK framework for Continual Learning

Lee et al. [2019] recently proved that under the NTK regime neural networks evolve as a linear model:

\[ f^*_\tau(x) = f^*_{\tau-1}(x) + (\nabla_x f_0(x), \omega^*_\tau - \omega^*_{\tau-1}) \]

with \( \omega^*_\tau \) being the final weight after training on task \( \tau \). The latter formulation implies the feature maps \( \phi(x) = \nabla_x f_0(x) \in \mathbb{R}^{1 \times p} \) is constant over time. Under that framework, Bennani et al. [2020] show that CL models can be expressed as a recursive kernel regression and prove generalization and performance guarantee of OGD under infinite memory setting. We build up on this theoretical framework to study CF and quantify how the tasks similarity imply forgetting through the lens of eigenvalues and singular values decomposition (PCA and SVD).

4 Analysis of Catastrophic Forgetting in finite memory

In this section, we propose a general definition of Catastrophic Forgetting (CF). Casted in the NTK framework, this definition allows to understand what are the main sources of CF. Namely, CF is likely to occur when two tasks align significantly. Finally, we investigate CF properties for the vanilla case (SGD) and projection based methods such as OGD and a variant of GEM. We then introduce a new algorithm called PCA-OGD, an extension of OGD which reduces CF.

4.1 A definition of Catastrophic Forgetting under the NTK regime

A natural quantity to characterize CF is the change in predictions for the same input between a source task \( \tau_S \) and target task \( \tau_T \).

Definition 1 (Drift).

Let \( \tau_S \) (respectively \( \tau_T \)) be the source task (respectively target task), \( \mathcal{D}_{\tau_S} \) the source test set, the CF of task \( \tau_S \) after training on all the subsequent tasks up to the target task \( \tau_T \) is defined as:

\[
\delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) = \left( f^*_{\tau_T}(x) - f^*_{\tau_S}(x) \right)_{(x,y) \in \mathcal{D}_{\tau_S}}
\]

(1)

Note that \( \delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) \) is a vector in \( \mathbb{R}^{n_{\tau_S}} \) that contains the changes of predictions for any input \( x \) in the task \( \tau_S \). In the case of classification, we take the \( k \)-output of \( f^*_\tau \) such that \( y_k = 1 \). In order to quantify the overall forgetting on this task, we use the squared norm of this vector.

Definition 2 (Catastrophic Forgetting).

Let \( \tau_S \) (respectively \( \tau_T \)) be the source task (respectively target task), \( \mathcal{D}_{\tau_S} \) the source test set, the CF of task \( \tau_S \) after training on all subsequent tasks up to task \( \tau_T \) is defined as:

\[
\Delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) = \| \delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) \|_2^2 = \sum_{(x,y) \in \mathcal{D}_{\tau_S}} (f^*_{\tau_T}(x) - f^*_{\tau_S}(x))^2
\]

(2)

The above expression is very general but has an interesting linear form under the NTK regime and allows us to get insight on the behavior on the variation of the forgetting.

Lemma 1 (CF under NTK regime).

Let \( \{\omega^*_\tau, \forall \tau \in [T]\} \) be the weight at the end of the training of task \( \tau \), the Catastrophic Forgetting of a source task \( \tau_S \) with respect to a target task \( \tau_T \) is given by:

\[
\Delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) = \| \delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) \|_2^2 = \| \phi(X^{\tau_S})(\omega^*_\tau - \omega^*_S) \|_2^2
\]

(3)

Proof. See Appendix Section 8.1

Lemma 1 expresses the forgetting as a linear relation between the kernel \( \phi(X^{\tau_S}) \) (which is assumed to be constant) and the variation of the weights from the source task \( \tau_S \) until the target task \( \tau_T \).

Remark 1. Note that, from Equation 4, two cases are possible when \( \Delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) = 0 \). The trivial case happens when \( \forall \tau \in [T] \):

\[
\left( f^*_{\tau+1}(x) - f^*_\tau(x) \right)_{(x,y) \in \mathcal{D}_{\tau_S}} = 0
\]

In this case, there is no drift at all. However, it is also possible that some tasks induce a drift on \( X^{\tau_S} \) that is compensated by subsequent tasks. Indeed, for \( \forall \tau \in [T] \):

\[
0 = \delta^{\tau_S \rightarrow \tau_T}(X^{\tau_S}) = \left( f^*_{\tau_T}(x) - f^*_{\tau_S}(x) \right)_{(x,y) \in \mathcal{D}_{\tau_S}} = \left( f^*_{\tau_T}(x) - f^*_\tau(x) + f^*_\tau(x) - f^*_{\tau_S}(x) \right)_{(x,y) \in \mathcal{D}_{\tau_S}}
\]

simply implies, for any \( (x,y) \in \mathcal{D}_{\tau_S} \),

\[
f^*_{\tau_T}(x) - f^*_\tau(x) = -(f^*_\tau(x) - f^*_{\tau_S}(x)).
\]

This would be an example of no forgetting due to a forward/backward transfer in the sense of Lopez-Paz and Ranzato [2017].

Now that we have defined the central quantity of this study, we will gain deeper insights by investigating SGD which is the vanilla algorithm.
4.2 High correlations across tasks induce forgetting for vanilla SGD

In this section, we derive the Catastrophic Forgetting expression for SGD. This will be the starting point to derive CF for the projection based methods (OGD, GEM and PCA-OGD).

**Theorem 1.** (Catastrophic Forgetting for SGD) Let $U\tau \Sigma_{\tau} V^{T}_{\tau}$ be the SVD of $\phi(X^{\tau})$ for each $\tau \in [T]$, and let $\lambda > 0$ the weight decay regularizer. The CF from task $\tau_{S}$ up until task $\tau_{T}$ is then given by:

$$\Delta_{\tau_{S} \rightarrow \tau_{T}}(X^{\tau_{S}}) = \left\| \sum_{k=\tau_{S}+1}^{T} U_{\tau_{S}} \Sigma_{\tau_{S}} O_{\tau_{S} \rightarrow k}^{\tau_{S} \rightarrow -k} M_{k} \hat{y}_{k} \right\|_{2}^{2}$$

where:

$$O_{\tau_{S} \rightarrow k}^{\tau_{S} \rightarrow -k} = V^{T}_{\tau_{S}} V_{k}$$

$$M_{k} = \Sigma_{k} [S^{2}_{k} + \lambda n_{k}]^{-1} U^{T}_{k}$$

$$\hat{y}_{k} = y_{k} - f_{k-1}^{*}(x^{k})$$

**Proof.** See Appendix Section 8.2

Theorem 1 describes the Catastrophic Forgetting for SGD on the task $T_{\tau_{S}}$, after training on the subsequent tasks up to the task $T_{\tau_{T}}$. The CF is expressed as a function of the overlap between the subspaces of the subsequent tasks and the reference task, through what we call the NTK overlap matrices $\{O_{\tau_{S} \rightarrow k}^{\tau_{S} \rightarrow -k}, k \in [\tau_{S}+1, \tau_{T}]\}$. High overlap between tasks increases the norm of the NTK overlap matrix which implies high forgetting.

More formally, the main elements of Catastrophic Forgetting are:

- $\Sigma_{\tau_{S}}$ encodes the importance of the principal components of the source task. Components with high magnitude contribute to forgetting since they imply high variation along those directions.

- $\{O_{\tau_{S} \rightarrow k}^{\tau_{S} \rightarrow -k}, k \in [\tau_{S}+1, \tau_{T}]\}$ encodes the similarity of the principal components between the source task and a subsequent task $k$. High norm of this matrix means high overlap between tasks and leads to high risk of forgetting. This forgetting occurs because the previous knowledge along a given component may be erased by the new dataset.

- $\hat{y}_{k}$ encodes the residual that remains to be learned by the current model. A null residual implies that the previous model predicts perfectly the new task, therefore there is no learning hence no forgetting.

- $M_{k}$ is a rotation of the residuals weighted by the principal components space. The rotated residuals $M_{k} \hat{y}_{k}$ can be interpreted as the residuals along each principal component.

- $\|\sum_{k=\tau_{S}+1}^{T} \|$ encodes that the forgetting can be canceled by other tasks by learning again forgotten knowledge.

We will see in what follows that the matrix $O_{\tau_{S} \rightarrow \tau_{T}}^{\tau_{S} \rightarrow -k}$ captures the alignment between the source task $\tau_{S}$ and the target task $\tau_{T}$. More formally, the singular values of $O_{\tau_{S} \rightarrow \tau_{T}}^{\tau_{S} \rightarrow -k}$ are the cosines of the principal angles between the spaces spanned by the source data $\phi(X^{\tau_{S}})$ and the target data $\phi(X^{\tau_{T}})$ [Wedin, 1983].

**Corollary 1** (Bounding CF with angle between source and target subspace).

Let $\Theta_{\tau_{S} \rightarrow \tau_{T}}^{\tau_{S} \rightarrow -k}$ be the diagonal matrix of singular values of $O_{\tau_{S} \rightarrow \tau_{T}}^{\tau_{S} \rightarrow -k}$ (each diagonal element $\cos(\theta_{\tau_{S}, \tau_{T}}^{\tau_{S} \rightarrow -k})_{i}$ is the cosine of the $i$-th principal angle between $\phi(X^{\tau_{S}})$ and $\phi(X^{\tau_{T}})$). Let $\sigma_{\tau_{S},1} \geq \sigma_{\tau_{S},2} \geq \ldots \geq \sigma_{\tau_{S},n_{\tau_{S}}}$ be the singular values of $\phi(X^{\tau_{S}})$ (i.e. the diagonal elements of $\Sigma_{\tau_{S}}$).

The bound of the forgetting from a source task $\tau_{S}$ up until a target task $\tau_{T}$ is given by:

$$\Delta_{\tau_{S} \rightarrow \tau_{T}}(X^{\tau_{S}}) \leq \sigma_{\tau_{S},1}^{2} \sum_{k=\tau_{S}+1}^{T} \|\Theta^{\tau_{S} \rightarrow -k}\|_{2}^{2} \|M_{k} \hat{y}_{k}\|_{2}^{2}$$

**Proof.** See Appendix Section 8.3

Corollary 1 bounds the CF by the sum of the cosines of the first principal angles between the source task $\tau_{S}$ and each subsequent task until the target task $\tau_{T}$ (represented by the diagonal matrix $\Theta_{\tau_{S} \rightarrow \tau_{T}}^{\tau_{S} \rightarrow -k}$) and a coefficient $\sigma_{\tau_{S},1}^{2}$ from the source task $\tau_{S}$.

- $\{\Theta_{\tau_{S} \rightarrow \tau_{T}}^{\tau_{S} \rightarrow -k}, k \in [\tau_{S}+1, \tau_{T}]\}$ is the diagonal matrix where each element represents the cosine angle between subspaces $\tau_{S}$ and $k$: $\cos(\theta_{\tau_{S},k})_{i}$. If the principal angle between two tasks is small (i.e. the two tasks are aligned), the cosine will be large which implies a high risk of forgetting.

- $\sigma_{\tau_{S},1}$ is the variance of the data of task $\tau_{S}$ along its principal direction of variation. Intuitively, $\sigma_{\tau_{S},1}$ measures the spread of the data for task $\tau_{S}$.

In the end, a potential component responsible for CF in the Vanilla SGD case is the projection from the source task onto the target task. This phenomenon is best characterized by the eigenvalues of $O_{\tau_{S} \rightarrow \tau_{T}}^{\tau_{S} \rightarrow -k}$ which acts as a similarity measure between the tasks. One avenue to mitigate the CF can be to project orthogonally to the source task subspace which are the main insight from OGD [Farajtabar et al., 2020] and GEM [Lopez-Paz and Ranzato, 2017].
4.3 The effectiveness of the orthogonal projection against Catastrophic Forgetting

Now, we study the GEM and OGD algorithms, we identify these two algorithms as projection based algorithms. We extend the previous analysis to study the effectiveness of these algorithms against Catastrophic Forgetting.

Recap OGD [Farajtabar et al., 2020] stores the feature maps of arbitrary samples from each task, then projects the update gradient orthogonally to these feature maps. The idea is to preserve the subspace spanned by the previous samples ([Yu et al., 2020] proposed a similar variant for multi-task learning).

GEM [Lopez-Paz and Ranzato, 2017] computes the gradient of the train loss over each previous task, by storing samples from each task. While OGD performs an orthogonal projection to the gradients of the model, GEM projects orthogonally to the space spanned by the losses gradients. The idea is to update the model under the constraint that the train loss over the previous tasks does not increase.

GEM-NT: Decoupling Forward/Backward Transfer from Catastrophic Forgetting OGD has been extensively studied by [Bennani et al., 2020], therefore we perform the analysis for the GEM algorithm, then highlight the similarities with OGD. Also, in order to decouple CF from Forward/Backward Transfer, we study a variant of GEM with no transfer at all, which we call GEM No Transfer (GEM-NT).

Similarly to GEM, GEM-NT maintains an episodic memory containing $d$ samples from each previous tasks seen so far. During each gradient step of task $\tau + 1$, GEM samples from the memory $d$ elements from each previous task then compute the average loss function gradient:

$$g_k = \frac{1}{d} \sum_{j=1}^{d} \nabla_x L_k(x_j^k), \quad \forall k = 1, ..., \tau$$

If the proposed update during task $\tau + 1$ can potentially degrades former solutions (i.e. $(g_{\tau+1}, g_k) < 0, \forall k \leq \tau$) then the proposed update is projected orthogonally to these gradients $g_k, \forall k \leq \tau$.

As opposed to GEM, which performs the orthogonal projection conditionally on the impact of the gradient update on the previous training losses, GEM-NT project orthogonally to $g_k, \forall k \leq \tau$ at each step irrespectively of the sign of the dot product. The algorithm pseudo-code can be found in Appendix Section 2.

The effectiveness of GEM-NT against CF Denote $G_{\tau} \in \mathbb{R}^{p \times \tau}$ the matrix where each columns represents $g_k, \forall k = 1, ..., \tau$, the orthogonal projection matrix is then defined as $T_{\tau} = I_o - G_{\tau} G_{\tau}^T$. This represents an orthogonal projection whatever the sign of the dot product $(g_{\tau+1}, g_k)$ in order to decouple the forgetting from transfer.

We are now ready to provide the CF of GEM-NT.

**Corollary 2** (CF for GEM-NT).

Using the previous notations, The CF from task $\tau_S$ up until task $\tau_T$ for GEM-NT given by:

$$\Delta_{\tau_S \rightarrow \tau_T} (X_{\tau_S}) = \left\| \sum_{k=\tau_S+1}^{\tau_T} U_{\tau_S} \Sigma_{\tau_S} \Theta_{GEM-NT,k} M_k y_k \right\|_2^2 (7)$$

where:

$$\Theta_{GEM-NT,k} = V_{\tau_S}^T G_{k-1} G_{k-1}^TV_k$$

$$M_k = \Sigma_k U_k^T (\phi(X_k)\bar{\phi}(X_k)^T + \lambda I_{n_k})^{-1}$$

$\bar{\phi}(X_k) = \phi(X_k)T_{k-1}$

(Differences with the vanilla case SGD are highlighted in color)

**Proof.** See Appendix Section 8.4

The difference for GEM-NT lies in the double projection of the source and target task onto the subspace $G_{\tau}$ which contain elements orthogonal to $g_k, \forall k = 1, ..., \tau - 1$.

Similarly to Corollary 1 we can bound each projection matrix $(V_{\tau_S}^T G_{k-1}^T$ and $G_{k-1}^TV_k, \forall k \in [\tau_S + 1, \tau_T])$ by their respective matrices of singular values $(\Theta_{GEM-NT,k-1}$ and $\Theta_{k-1}^T, \forall k \in [\tau_S + 1, \tau_T])$. This leads us to the following upper-bound for the CF of GEM-NT:

$$\Delta_{\tau_S \rightarrow \tau_T} (X_{\tau_S}) \leq$$

$$\sigma_{\tau_S,1}^2 \sum_{k=\tau_S+1}^{\tau_T} \left\| \Theta_{\tau_S \rightarrow \tau_{k-1}} \right\|_2^2 \left\| \phi^{k \rightarrow \tau_{k-1}} \right\|_2^2 \left\| M_k y_k \right\|_2^2 $$(8)

Connection of GEM-NT to OGD For the analysis purpose, let’s suppose that the memory per task is 1, $\lambda = 0, \forall \tau \in [T]$ and assume a mean square loss error function. In that case:

$$g_k = \begin{cases} 
\nabla_x f_k(x)(f_k(x) - y_k) & \text{(GEM-NT)} \\
\nabla_x f_k(x) & \text{(OGD)}
\end{cases}$$

- unlike OGD, GEM-NT weights the orthogonal projection with the residuals $(f_k(x) - y_k) = (y_k + \delta^{k \rightarrow \tau}(x_k))$ which represents the difference between the new prediction (due to the drift) for $x_k$ under model $\tau$ and the target $y_k$. 
• Previous tasks that are well learned (small residuals) will contribute less to the orthogonal projection to the detriment of tasks with large residuals (badly learned then). This seems counter-intuitive because by doing so, the projection will not be orthogonal to well learned tasks (in the edge case of zero residuals) then unlearning can happen for those tasks.

While OGD and GEM-NT are more robust to CF than SGD through the orthogonal projection, they do not leverage explicitly the structure in the data. We can then compress this information through dimension reduction algorithms such as SVD in order to both maximise the information contained in the memory as well as mitigating the CF.

4.4 PCA-OGD: leveraging structure by projecting orthogonally to the top d principal directions

Unlike OGD that stores randomly d samples from each task $k = 1,\ldots,\tau$ of $\{\nabla \omega f(x^k_j)\}^{d}_{j=1}$, at the end of each task $\tau$, PCA-OGD samples randomly $s > d$ elements from $\{\nabla \omega f(x^\tau)\}$ denoted $v_{\tau,i}$, $i = 1,\ldots,d$. These are the directions that capture the most variance of the data. If we denote by $P_{\tau,:d}$ the matrix where each columns represents $v_{\tau,i}$, $k = 1,\ldots,\tau$, $i = 1,\ldots,d$ then the orthogonal matrix projection can be written as:

$$T_{\tau,:d} = I_p - P_{\tau,:d} P_{\tau,:d}^T = R_{\tau,:d} R_{\tau,:d}^T \quad (10)$$

where the columns of $R_{\tau,:d}$ form an orthonormal basis of the orthogonal complement of the span of $P_{\tau,:d}$. For the terminology, $P_{\tau,:d} \in \mathbb{R}^{p \times (\tau - d)}$ (respectively $R_{\tau,:d} \in \mathbb{R}^{p \times (\tau - d)}$) represents the top subspace (respectively the residuals subspace) of order $d$ for task $1$ until $\tau$. A pseudo-code of PCA-OGD is given in Alg. 1 (the computational overhead can be found in the Appendix).

We are now ready to provide the CF of PCA-OGD.

**Corollary 3** (Forgetting for PCA-OGD).

For each $\tau \in \mathbb{T}$, let $\hat{\phi}(x^\tau) = \phi(x^\tau) T_{\tau-1,:d}$ and let $U_{\tau} \Sigma_{\tau} V_{\tau}^T$ be the SVD of $\hat{\phi}(x^\tau)$. The CF for PCA-OGD is given by:

$$\Delta^{T,S \rightarrow T,T}(x^{T_S}) = \left\| \sum_{k=T_S+1}^{T} U_{\tau} \Sigma_{\tau} O_{PCA}^{T_S \rightarrow k} M_k y_k \right\|^2$$

where:

$$O_{PCA}^{T_S \rightarrow k} = V_{\tau} R_{k-1,:d} R_{k-1,:d}^T V_k$$

$$M_k = \Sigma_k U_k [\hat{\phi}(x^k) \hat{\phi}(x^k)^T + \lambda I_{n_k}]^{-1}$$

$$\hat{\phi}(x^k) = \phi(x^k) T_{k-1,:d}$$


**Algorithm 1:** PCA-OGD (Differences with OGD in red)

**Input:** A task sequence $T_1, T_2, \ldots$, learning rate $\eta$, PCA samples $s$, components to keep $d$

1. Initialize $S_J \leftarrow \{\}$ ; $\omega \leftarrow \omega_0$

2. for Task ID $\tau = 1, 2, 3, \ldots$

    repeat

    \[
    g \leftarrow \text{Stochastic Batch Gradient for } \tau \text{ at } \omega;
    \]

    \[
    \text{// Orthogonal updates}
    \]

    \[
    \tilde{g} = g - \sum_{v \in S_J} \text{proj}_V(g);
    \]

    \[
    \omega \leftarrow \omega - \eta \tilde{g}
    \]

    until convergence;

    \[
    \text{// Gram-Schmidt orthogonalization}
    \]

    for $(x, y) \in D_\tau$ and $k \in [1, c]$ s.t. $y_k = 1$

    \[
    u \leftarrow \nabla f_\tau(x; \omega) - \sum_{v \in S_J} \text{proj}_V(\nabla \omega f_\tau(x; \omega))
    \]

    \[
    S_J \leftarrow S_J \cup \{u\}
    \]

end for

\[
\text{// SVD}
\]

Sample $s$ elements from $\tau$

\[
\text{top } d \text{ eigenvectors } \leftarrow \text{PCA}\{\nabla f_\tau(x^\tau_j)\}_{j=1}^s
\]

\[
S_J \leftarrow S_J \cup \{ \text{top } d \text{ eigenvectors} \}
\]

end for

**Proof.** See Appendix Section 8.5

Corollary 3 underlines the difference with GEM-NT as this time the double projection are on the residuals subspace $R_{k-1,:d}$ containing the orthogonol vector to the features map $\nabla \omega f(x)$ instead of the loss function gradient.

**Remark 2.**

- **PCA is helpful in datasets where the eigenvalues are decreasing exponentially since keeping a small number of components can leverage a large information and explain a great part of the variance. Projecting orthogonally to these main components will lead to small forgetting if $\sigma_{\tau,+1}$ is small.**

- **On the other hand, unfavourable situations where data are spread uniformly along all directions (i.e., eigenvalues are uniformly equals ) will requires to keep all components and a larger memory. As an example, we build a worst-case scenario in Appendix Section 8.12 where OGD is performing better than PCA-OGD.**

Similarly as the previous case, we can bound the double projection on $R_{k-1,:d}$ with the corresponding diagonal
matrix $\Theta_{\tau S \rightarrow R_k}$. Additionally, the CF is bounded by $\sigma_{\tau S, d+1}$ which is due to the orthogonal projection to the first $d$ principal directions. The upper bound of the CF is given by:

$$\Delta_{\tau S \rightarrow \tau T}(X^\tau) \leq \sigma_{\tau S, d+1}^2 \sum_{k=\tau S+1}^{\tau T} \left\| \Theta_{\tau S \rightarrow R_k} \right\|_2^2 \left\| \Theta_{k \rightarrow R_{k-1}} \right\|_2^2 \left\| M_k \tilde{y}_k \right\|_2^2$$

(12)

Note that in contrast with Eq. (8), the first term in the upper bound is the $(d + 1)$-th singular value of $\phi(X^\tau)$, which is due to the PCA step of PCA-OGD. A summary of the forgetting properties of the described methods can be found in Table 2 in Appendix.

5 Experiments

In this section, we study the impact of the NTK overlap matrix on the forgetting by validating Corollary 1. We then illustrate how PCA-OGD efficiently captures and compresses the information in datasets (Corollary 3). Finally, we benchmark PCA-OGD on standard CL baselines.

5.1 Low eigenvalue of the NTK overlap matrix induces smaller drop in performance

Objective: As presented in Corollary 1, we want to assess the effect of the eigenvalues of the NTK overlap matrix on the forgetting.

Experiments: We measure the drop in accuracy for task 1 until task 15 on Rotated MNIST with respect to the maximum eigenvalue of the NTK overlap matrix $O^{1 \rightarrow 15}$.

Results: Figure 2 shows the drop in accuracy between task 1 and task 15 for Rotated MNIST versus the largest eigenvalue of $O^{1 \rightarrow 15}$. As expected low eigenvalues lead to a smaller drop in accuracy and thus less forgetting. PCA-OGD improves upon OGD, having from 7% to 10% less drop in performance.

5.2 PCA-OGD reduces forgetting by efficiently leveraging structure in the data

Objective: We show how capturing the top $d$ principal directions helps reducing Catastrophic Forgetting (Corollary 3).

Experiments: We compare the spectrum of the NTK overlap matrix for different methods: SGD, GEM-NT, OGD and PCA-OGD, for different memory sizes.

Finally, the final accuracies on Rotated and Permuted MNIST are reported in Figure 4 for the first seven tasks. In Rotated MNIST, we can see that PCA-OGD is twice
more memory efficient than OGD: with a memory size of 100 PCA-OGD has comparable results to OGD with a memory size 200. Interestingly, while the marginal increase for PCA-OGD is roughly constant going from memory size 25 to 50 or 50 to 100, OGD incurs a high increase from memory size 100 to 200 while below that threshold the improvement is relatively small.

The later therefore project updates orthogonally to outdated gradients. This issue has also been mentioned in Bennani et al. 2020. Note the good performance of PCA-OGD in Split CIFAR where the dataset size is 2,500 (making the NTK assumption more realistic) and similar patterns are seen across tasks (CIFAR100 dataset is divided into 20 superfamilies within which we can count 5 subfamilies hence having a pattern across tasks. To examine this hypothesis, we plot the NTK changes for different datasets in Appendix 8.10. We can indeed see that the NTK does not vary anymore after 1 task for Split CIFAR while it increases linearly for MNIST datasets which confirms our hypothesis.

<table>
<thead>
<tr>
<th></th>
<th>SGD</th>
<th>EWC</th>
<th>A-GEM</th>
<th>OGD</th>
<th>PCA-OGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permuted MNIST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>72.34±1.26</td>
<td>70.4±1.26</td>
<td>81.64±0.44</td>
<td>80.95±0.85</td>
<td>81.42±0.82</td>
</tr>
<tr>
<td>F2</td>
<td>14.88±1.64</td>
<td>3.81±0.47</td>
<td>7.29±0.45</td>
<td>9.72±0.51</td>
<td>9.11±0.65</td>
</tr>
<tr>
<td>Rotated MNIST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>66.02±0.10</td>
<td>58.2±0.02</td>
<td>83.52±0.22</td>
<td>71.42±0.35</td>
<td>82.05±0.50</td>
</tr>
<tr>
<td>F2</td>
<td>29.57±0.56</td>
<td>13.44±0.82</td>
<td>9.86±0.28</td>
<td>16.52±0.46</td>
<td>11.67±0.65</td>
</tr>
<tr>
<td>Split MNIST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>95.1±1.08</td>
<td>94.60±1.46</td>
<td>99.29±1.02</td>
<td>96.05±0.34</td>
<td>95.96±0.20</td>
</tr>
<tr>
<td>F2</td>
<td>2.02±1.48</td>
<td>2.08±1.56</td>
<td>2.82±1.72</td>
<td>0.37±0.21</td>
<td>0.28±0.15</td>
</tr>
<tr>
<td>Split CIFAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>88.41±1.60</td>
<td>80.43±0.67</td>
<td>94.93±2.4</td>
<td>90.97±0.72</td>
<td>72.7±0.97</td>
</tr>
<tr>
<td>F2</td>
<td>22.69±2.18</td>
<td>6.56±1.49</td>
<td>31.36±2.58</td>
<td>8.27±0.51</td>
<td>5.39±0.85</td>
</tr>
</tbody>
</table>

Table 1: Average Accuracy and Forgetting for all baselines considered across the datasets (5 seeds).

5.3 General performance of PCA-OGD against baselines

Objective and Experiments: We compare PCA-OGD against other baseline methods: SGD, A-GEM Chaudhry et al. 2018 and OGD Farajtabar et al. 2020. Additionally to the final accuracies, we report the Average Accuracy $A_T$ and Forgetting Measure $F_T$ Lopez-Paz and Ranzato 2017 Chaudhry et al. 2018. We run AGEM instead of GEM-NT which is faster with comparable results Chaudhry et al. 2018 (since GEM-NT is solving a quadratic programming optimization at each iteration step). Definition of these metrics and full details of the experimental setup can be found in Appendix 8.11.

Results: The results are summarized in Table 1 (additional results are presented in Appendix 8.11). Overall, PCA-OGD obtains comparable results to A-GEM. A-GEM has the advantage of accounting for the NTK changes by updating it while PCA-OGD and OGD are storing the gradients from previous iteration.

Figure 4: Final accuracy on Rotated MNIST for different memory size (averaged over 5 seeds ±1 std). OGD needs twice as much memory as PCA-OGD in order to achieve the same performance (i.e compare OGD (200) and PCA (100)).

We ran OGD and PCA-OGD on a counter-example dataset (Permuted MNIST), where there is no structure within the dataset (see Appendix 8.7). In this case, PCA-OGD is less efficient since it needs to keep more principal components than in a structured dataset setting.

6 Conclusion

We present a theoretical analysis of CF in the NTK regime, for SGD and the projection based algorithms OGD, GEM-NT and PCA-OGD. We quantify the impact of the tasks similarity on CF through the NTK overlap matrix. Experiments support our findings that the overlap matrix is crucial in reducing CF and our proposed method PCA-OGD efficiently mitigates CF. However, our analysis relies on the core assumption of overparameterisation, an important next step is to account for the change of NTK over time. We hope this analysis opens new directions to study the properties of Catastrophic Forgetting for other Continual Learning algorithms.

7 Acknowledgments

The authors would like to thank Joelle Pineau for useful discussions and feedbacks. Finally, we thank Compute Canada for providing computational resources used in this project.
References


