Alma Mater Studiorum · Università di Bologna

SCHOOL OF SCIENCE

Master Degree in Mathematics

Curriculum: Advanced Mathematics for Applications

Dynamical Evolution of a Neural Network in the Online Learning Regime and the Effect of Low-Rank Adaptation on Catastrophic Forgetting.

Master Thesis in Satistical Mechanics applied to Machine Learning

Supervisor:
Dott.ssa Federica Gerace

Presented by:
Filippo Alessandroni

Cosupervisors:
Dott. Alessandro Breccia
Dott. Théo Marchetta

Academic Year 2024-2025

He who knows all the answers
has not been asked all the questions.

Colui che conosce tutte le risposte
non ha fatto tutte le domande.

Abstract

The widespread integration of artificial intelligence into everyday life has raised fundamental questions about the mathematical principles underlying neural network learning. Modern architectures are rarely trained from scratch for each new task; instead, they leverage knowledge acquired from previously learned tasks. This paradigm, known as transfer learning, consists in initializing a model for a target task using a network previously trained on a related source task. For example, a network trained to distinguish cars from motorcycles can be fine-tuned to classify bicycles and trucks, building upon previously learned feature representations. When the objective is not only to learn the target task but also to retain performance on the source task, the problem falls within the framework of continual learning. In this setting, forgetting quantifies the degradation in performance on the source task after training on the target task. A well-known challenge is catastrophic forgetting: during fine-tuning, performance on the original task can deteriorate dramatically. A central question is whether specific fine-tuning strategies can mitigate catastrophic forgetting. Although the phenomenon has been extensively studied and its causes widely analyzed (see e.g. [1, 2, 3]), it remains unclear to what extent it can be controlled through tailored adaptation techniques. In this work, we investigate the impact of Low-Rank Adaptation (LoRA) on catastrophic forgetting. LoRA is a parameter-efficient fine-tuning method in which the pretrained weight matrix is kept frozen and updated by adding a low-rank perturbation, expressed as the product of two trainable matrices. Since the original weights remain unchanged, this approach may have significant implications for memory retention. We address this question within the theoretical framework of online learning introduced in [4], which enables the study of learning dynamics by tracking the time evolution of macroscopic order parameters (overlaps), such as weight norms. In online learning, the weights of a network are updated after each individual data point, which is subsequently discarded. In the regime of large datasets, this one-pass stochastic gradient descent approximation becomes realistic, as the probability of encountering the same example twice is negligible. This perspective allows us to analyze neural networks as dynamical systems. Focusing

on committee machines [5, 6], we derive differential equations that govern the learning dynamics and study their equilibrium configurations, plateau phases, and generalization curves. Through this analytical approach, we identify the conditions under which LoRA mitigates catastrophic forgetting, highlighting the role of source–target similarity, architectural choices for teacher and student networks, and the interplay between the hyperparameters introduced by LoRA.

Abstract

La diffusione su larga scala dell'intelligenza artificiale ha sollevato importanti questioni sui fondamenti matematici del processo di learning delle reti neurali. Al giorno d'oggi, le reti neurali vengono raramente allenate da zero; nella pratica, si sfrutta la conoscenza che altre reti neurali hanno acquisito quando allenate su altri compiti. Questo paradigma, noto come transfer learning, consiste nell'inizializzare un modello, detto target, utilizzando una rete precedente allenata su un compito affine. Tale modello preallenato è detto source. Ad esempio, una rete allenata per distinguere auto e moto può essere modificata (*fine tuning*) per classificare camion e biciclette, basandosi su caratteristiche comuni tra queste due classi di oggetti. Se l'obiettivo fosse non solo imparare il nuovo task, ma anche mantenere una buona performance sul primo, si starebbe parlando del cosiddetto continual learning. In tal caso, si chiama *forgetting* quella quantità che misura la perdita di performance sul primo task una volta che si è iniziato ad allenare la rete per risolvere il secondo. Tuttavia, questa procedura soffre di una problematica molto nota detta *catastrophic forgetting*: la performance sul source deteriora in maniera drammatica. Una domanda cruciale è se esistano delle tecniche di fine-tuning che possano arginare il catastrophic forgetting. Sebbene il fenomeno sia stato ampiamente studiato, così come le sue cause (si vedano ad esempio [1, 2, 3]), rimane ancora poco chiaro se e in che misura esso si possa controllare usando delle specifiche tecniche di fine-tuning. In questa tesi, studieremo l'impatto della Low-Rank Adaptation (LoRA) sul catastrophic forgetting. LoRA è un metodo di fine-tuning computazionalmente molto efficente, in cui la matrice del source è mantenuta fissa durante tutto il training sul secondo task e ad essa viene aggiunta una perturbazione di rango basso, espressa come il prodotto di due matrici, le quali vengono imparate usando la discesa del gradiente sulla funzione obiettivo che vogliamo minimizzare (loss function). Poiché i pesi originali rimangono immodificati, questo approccio potrebbe avere effetti rilevanti sul forgetting. Tutto questo studio verrà condotto utilizzando le tecniche di apprendimento online introdotte in [4]. Questo approccio permette lo studio della dinamica dell'apprendimento, poiché permette lo studio dell'evoluzione nel tempo di alcune quantità macroscopiche, dette parametri d'ordine

(od overlaps), come ad esempio la norma dei pesi della rete. Nell'apprendimento online, i pesi della rete vengono aggiornati mediante discesa dei gradienti, in cui la funzione obiettivo viene minimizzata utilizzando un singolo dato in input, che viene successivamente eliminato. In un regime di sovrabbondanza di dati, questa discesa del gradiente su singolo input è realistico, giacché la probabilità di incontrare più volte lo stesso dato è pressoché trascurabile. Questa prospettiva ci permette di analizzare le reti neurali come fossero sistemi dinamici. Focalizzandoci sullo studio delle committee machines [5] [6], deriveremo delle equazioni differenziali che governano la dinamica di apprendimento, permettendoci di studiare configurazioni di equilibrio, presenza di eventuali fasi sub-ottimali di apprendimento e permettendo di determinare analiticamente le curve di apprendimento. Utilizzando questo approccio, identificheremo le condizioni sotto le quali LoRA mitiga gli effetti del catastrophic forgetting, mettendo in luce l'importanza della similarità tra compiti, della scelta delle architetture insegnante e studente e il rapporto con gli iperparametri introdotti da LoRA.

Contents

Chapter 1

Dynamical equations for Committee Machines.

In this chapter, we will introduce a specific artificial neural architecture: the *committee machine*. Inspired by the work in [4, 7], we will introduce a set of ordinary differential equations that will describe the learning process of a neural network in the famous teacher-student setting [6] under synthetic input data, i.e. where the input data are not real-world data-points, but are drawn from a known probability distribution, which for us will be a gaussian distribution with a specific mean and covariance. This chapter is primarily devoted to the presentation of the theoretical results in [4, 7, 8], for which we will provide explicit computations of the formulas in [4, 7], together with the reproduction of some numerical experiments. Some notation may differ from those in the literature and is made to make the presentation more uniform.

1.1 What is a neural network

This section is a basic introduction to the main concepts of neural network, inspired by [9]. Neural networks are algorithms whose goal is to learn from data to accomplish a specific task. For example, we can design a neural architecture that, given an image of a cat or a dog, outputs the animal in that image. More precisely, we can give the following:

Definition 1.1.1. *A (deep) neural network with L layers is described by a sequence of matrices $W_1, \dots, W_L$, called weights, and functions $g_1, \dots, g_L$, called activation functions and are typically non-linear. The output of the network is given by a sequence of transformations of an input vector ξ as follows:*

$$z_1 = g_1(W_1\xi)$$

where $W_1\xi$ is a matrix-vector product and g_1 is applied component-wise. Then we define recursively for any $i = 2, \dots, L$:

$$z_i = g_i(W_i z_{i-1})$$

and the output of the network is z_L, which we can also denote as $\sigma(\xi) := z_L$.

Remark. *For our purposes, we will consider the case where z_L is a scalar.*

Once the model is introduced, we need to talk about the data. A *data set* is a collection $(\xi^\mu, \tau^\mu)_{\mu=1,\dots,P}$ of data, where ξ^μ is the input (e.g. image) and τ^μ is the label associated to that input (e.g. "cat" or "dog"). We will use the data to "train" the network to associate the correct label to *any* input. In order to do this, we need a function that measures how well the network is labeling a specific input vector.

Definition 1.1.2. *A loss function is a function $e : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.*

Remark. *Typically, e is non-negative and can be thought of as a distance function.*

Remark. *Given a data point (ξ^μ, τ^μ) and a deep neural network with output $z_L = \sigma(\xi^\mu)$, the loss on input ξ^μ is defined as $e(\xi^\mu) := e(\sigma(\xi^\mu), \tau^\mu)$. The idea is that, the lower the loss, the higher the ability of the network to associate the true label to the given data.*

The idea is then to minimize a loss function, called *training loss* that is an average of the losses computed on the data set $(\xi^\mu, \tau^\mu)_{\mu=1,\dots,P}$ (called *training set*) and defined as a function of the weights $W = [W_1, \dots, W_L]$ of the network as follows:

$$\mathcal{E}_{train}(W) = \frac{1}{|B|} \sum_{\mu \in B} e(\sigma(\xi^\mu), \tau^\mu)$$

where $B \subseteq \{1, \dots, P\}$ is called *batch* and is a collection of *some* data points. In order to minimize the *training loss* we can use the Gradient Descent algorithm (GD), that is, we can update the weights using the following relation:

$$W_{k+1} = W_k - \eta \nabla \mathcal{E}_{train}(W_k)$$

with $\eta > 0$ called *learning rate*. Using many batches at a time allows us to determine a sequence of weights W_k that can eventually converge to a minimizer of the training loss, W^*.

Remark. *Convergence to an optimal value is not always guaranteed, as the loss function may present many local minima, making GD only locally optimal.*

Finally, once the training procedure has finished, we can test how good the trained network W is, by computing the so-called *test loss*:

$$\mathcal{E}_{test}(W) = \frac{1}{P_{test}} \sum_{\mu=1}^{P_{test}} e(\sigma(\hat{\xi}^\mu), \hat{\tau}^\mu)$$

where $(\hat{\xi}^\mu, \hat{\tau}^\mu)_{\mu=1,\dots,P_{test}}$ is the *test set*, possibly different from the train set.

1.2 Committee Machines

A *committee machine* is a two-layered neural architecture in which the output of the network is provided as a linear combination [5] of the outputs of more simple networks. Those simpler units are called *perceptrons*.

Definition 1.2.1. *Let $N \in \mathbb{N}$. Let $J \in \mathbb{R}^N$ and $g : \mathbb{R} \to \mathbb{R}$ be a function. A perceptron is a couple $\Pi = (J, g)$. The output of a perceptron $\Pi = (J, g)$ on the input data $\xi \in \mathbb{R}^N$ is defined as:*

$$O(J, g\,; \xi) = g\left(\frac{J \cdot \xi}{\sqrt{N}}\right). \tag{1.1}$$

We refer to g as the activation function.

Remark. *The reason why we are scaling using $\sqrt{N}$ will be clear in the following of this chapter. This is due to the so-called thermodynamic limit, i.e. the case where $N \to +\infty$, which is the main case of interest in statistical mechanics, as neural networks typically have a large number of neurons and synaptic connections, as happens in the human brain.*

Going back to our description of a committee machine, we can give the following

Definition 1.2.2. *Let $N, K \in \mathbb{N}$. Let $\{J_i\}_{i=1,\dots,K} \subseteq \mathbb{R}^N$, $h \in \mathbb{R}^K$ and $g : \mathbb{R} \to \mathbb{R}$ be a function. A committee machine is the data $\mathfrak{C} = (\boldsymbol{J}, \boldsymbol{h}, g)$, where $\mathbf{J} = \Big[J_1, \dots, J_K\Big]$ is a matrix whose columns are the vectors J_i's. The output of a committee machine is defined as:*

$$\Omega(\boldsymbol{J}, \boldsymbol{h}, g\,; \xi) = \sum_{i=1}^{K} h_i\, g\left(\frac{J_i \cdot \xi}{\sqrt{N}}\right). \tag{1.2}$$

K is the number of hidden units, $\mathbf{J}$ is called input-to-hidden weight matrix, while $\mathbf{h}$ is called the hidden-to-output weight vector. Again, g is called activation function.

Remark. *As we said before, one can rephrase the definition of output of a committee machine in terms of those of many perceptrons as follows:*

$$\Omega(\boldsymbol{J}, \boldsymbol{h}, g\,; \xi) = \sum_{i=1}^{K} h_i\, O(J_i, g\,; \xi) \tag{1.3}$$

Remark. *In the following, when possible, we will use the shortcut* $\Omega(\mathfrak{C};\xi)$ *for* $\Omega(\boldsymbol{J},\boldsymbol{h},g\,;\xi)$ *, where* $\mathfrak{C}=(\mathbf{J},\mathbf{h},g)$*.*

Remark. *Working with committee machines allows us to retrieve the simpler case of a perceptron by considering the case* $K=1$*.*

There are many reasons why committee machines should be examined in depth, as they represent a powerful architecture in many learning scenarios. Firstly, unlike the perceptron, committee machines represent a more complex class of neural networks and therefore provide a wider range of applications [10]; for example, if we want to solve a classification task where the data is not linearly separable, a perceptron is not able to do so, as the perceptron parametrizes just a hyperplane defined by the vector $\mathbf{J}$. Moreover, committee machines proved to be universal approximators [11].

1.3 The teacher-student setting

The teacher-student setting [6] is a well-known learning scenario in statistical mechanics of neural networks, as it is a non-trivial yet mathematically tractable framework in which the data are labeled according to a rule defined by a neural network, called *teacher*, and another network, the *student*, is trained to associate the correct label to the data. The idea behind this setting is the following:
Given an input data $\xi\in\mathbb{R}^N$, the label associated with this input is provided by a *teacher network* as

$$\tau(\xi):=\Omega(\mathfrak{T}\,;\xi),$$

where $\mathfrak{T}=(\mathbf{B},\mathbf{v},g)$ is a committee machine with M hidden units. In the following, we will use a specific activation function, the scaled error function

$$g(z)=\frac{1}{\sqrt{2\pi}}\int_{-z}^{z}e^{-\frac{t^2}{2}}\,dt.$$

This function allows us to compute many quantities of interest analytically [4, 7]. When considering the output of a committee machine, we will keep the activation function implicit.

Remark. *The scaled error function is an odd-symmetric function, it is* $[-1,1]$*-valued and has derivative* $g'(z)=\sqrt{\frac{2}{\pi}}\,e^{-\frac{z^2}{2}}$*.*

Now, we can introduce the student network, which is a committee machine $\mathfrak{S}=(\mathbf{J},\mathbf{h},g)$ with K hidden units, where the weights $\mathbf{J}$ and $\mathbf{h}$ are tunable, meaning that can

change in order to learn the rule provided by the teacher network, that is we would like to have

$$\sigma(\xi) := \Omega(\mathfrak{C}; \xi) = \tau(\xi) \text{ for all } \xi \in X,$$

where $X \subseteq \mathbb{R}^N$ is the input space.

Remark. *One should not always expect the student to be able to perfectly generalize the rule, meaning to be able to assign the true label to every input data. This is a key point in machine learning, as not every network is able to generalize every task, since the underlying rule may be too complex to be perfectly learned by small architectures. Much work has been done in this sense [1, 4, 7, 12, 13], both in the general learning scenario and in the case of committee machines.*

Based on this observation, we may give a more flexible, yet effective, concept of generalization of a task [4, 7]. Let ξ be an N-dimensional random variable, which represents our input data. Let $\mathfrak{T}, \mathfrak{S}$ be the teacher and student committee machines, respectively. The *generalization error* associated with $\mathfrak{S}$ is defined as:

$$\mathcal{E}(\mathfrak{S}) = \langle e(\tau(\xi), \sigma(\xi)) \rangle, \tag{1.4}$$

where $\langle \cdot \rangle$ represents the average over the distribution of ξ, while $e(\cdot)$ is a loss function, which measures the mismatch between the true label associated to ξ (i.e. $\tau(\xi)$) and the prediction of the student $\sigma(\xi)$. In the following, we will work with the squared loss function

$$e(\tau, \sigma) = \frac{1}{2}(\tau - \sigma)^2.$$

We can then write the generalization error explicitly:

$$\begin{aligned}
\mathcal{E}(\mathfrak{S}) &= \langle e(\tau(\xi), \sigma(\xi)) \rangle \\
&= \frac{1}{2} \langle (\Omega(\mathfrak{T}\,; \xi) - \Omega(\mathfrak{C}; \xi))^2 \rangle \\
&= \frac{1}{2} \langle (\sum_{m=1}^{M} v_m \, g\left(\frac{B_m \cdot \xi}{\sqrt{N}}\right) - \sum_{i=1}^{K} h_i \, g\left(\frac{J_i \cdot \xi}{\sqrt{N}}\right))^2 \rangle \\
&= \frac{1}{2} \sum_{m,n=1}^{M} v_m v_n \langle g(Y_m) g(Y_n) \rangle - \sum_{i=1}^{K} \sum_{n=1}^{M} h_i v_n \langle g(Y_n) g(X_i) \rangle \\
&+ \frac{1}{2} \sum_{i,k=1}^{K} h_i h_k \langle g(X_i) g(X_k) \rangle.
\end{aligned} \tag{1.5}$$

Here we used the shortcuts $X_i := \frac{J_i \cdot \xi}{\sqrt{N}}$ and $Y_m := \frac{B_m \cdot \xi}{\sqrt{N}}$, that are typically called *preactivations* [4, 7].

Remark. *Equation* (1.5) *provides a very powerful tool for analyzing the behavior of the network with different weights. Indeed, as long as the average* $\langle\cdot\rangle$ *contains only functions involving* X_i, Y_m*, it is not necessary to compute the average over the* N*-dimensional distribution of* ξ*, which may be extremely difficult with* N *large. As long as the functions in* $\langle\cdot\rangle$ *only contain preactivations and do not explicitly depend on* ξ *(see e.g. last equality in 1.5), we can compute the average implicitly using a low-dimensional distribution, which is the* $(K+M)$*-dimensional joint distribution of the preactivations [4, 7].*

Remark. *Note that the generalization error in* (1.4) *is an average over a probability distribution of some random vector. A major issue in machine learning is given by the fact that the distribution of the data points is rarely at hand, meaning that the explicit computation of the generalization error is usually not possible.*

Remark. *The generalization error measures how well the network is able to solve the task on average.*

1.4 The learning phase

The learning phase of a neural network is when the weights of the network are tuned to minimize a prescribed *training loss*. In this thesis, inspired by [4, 7], we will exploit the main characteristic of the *online learning algorithm*, where a Gradient Descent step is made on one input at a time, i.e., at each step $\mu = 1, \dots, P$, we draw independently one input vector ξ^μ and compute the loss only on this input, i.e., we compute

$$e(\Omega(\mathfrak{T}\,;\xi^\mu), \Omega(\mathfrak{C}^\mu\,;\xi^\mu)) = \frac{1}{2}(\sum_{m=1}^{M} v_m\, g\left(\frac{B_m\cdot\xi^\mu}{\sqrt{N}}\right) - \sum_{i=1}^{K} h_i^\mu\, g\left(\frac{J_i\cdot\xi^\mu}{\sqrt{N}}\right))^2,$$

where $\mathfrak{C}^\mu = (\mathbf{J}^\mu, \mathbf{h}^\mu)$. Then, we compute the gradient with respect to the weights. More explicitly, the gradient with respect to the i-th column of $\mathbf{J}$ is the following:

$$\nabla_{J_i} e(\Omega(\mathfrak{T}\,;\xi^\mu), \Omega(\mathfrak{C}^\mu\,;\xi^\mu)) = \Delta^\mu\, h_i^\mu\, g'(X_i^\mu)\,\frac{\xi^\mu}{\sqrt{N}} \tag{1.6}$$

where we denote: $\Delta^\mu := \sum_{i=1}^{K} h_i^\mu\, g\,(X_i^\mu) - \sum_{m=1}^{M} v_m\, g\,(Y_m^\mu)$ and X_i^μ, Y_m^μ are the preactivations on input ξ^μ.

Moreover, the gradient with respect to the i-th component of the head vector is:

$$\nabla_{h_i} e(\Omega(\mathfrak{T};\xi^\mu), \Omega(\mathfrak{C}^\mu;\xi^\mu)) = \Delta^\mu\, g(X_i^\mu). \tag{1.7}$$

Finally, the gradient update for the weights of the student is made explicitly as follows

$$J_i^{\mu+1} = J_i^\mu - \frac{\eta_J}{\sqrt{N}}\Delta^\mu\, h_i^\mu\, g'(X_i^\mu)\,\xi^\mu \tag{1.8}$$

$$h_i^{\mu+1} = h_i^\mu - \frac{\eta_h}{N}\Delta^\mu\, g(X_i^\mu). \tag{1.9}$$

Remark. *The scaling of a factor N of the learning rate in* (1.9) *is needed later to perform the thermodynamic limit [1, 4, 7].*

1.5 The order parameters

When talking about complex systems, one must take into account the role played by the so-called *order parameters*: some low-dimensional quantities that can be computed to analyze the current state of the system. In the context of committee machines trained in the online regime, as in [4, 7] we define the following quantities:

$$Q_{ik} = \frac{J_i \cdot J_k}{N} \quad \text{for all} \quad i,k \in \{1,\dots,K\} \tag{1.10}$$

$$R_{in} = \frac{J_i \cdot B_n}{N} \quad \text{for all} \quad i \in \{1,\dots,K\} \quad n \in \{1,\dots,M\} \tag{1.11}$$

$$T_{mn} = \frac{B_m \cdot B_n}{N} \quad \text{for all} \quad m,n \in \{1,\dots,M\} \tag{1.12}$$

Here, Q measures the overlap between the student's synaptic connections, while R measures how much each student's perceptron is aligned with those of the teachers. Many learning scenarios have been studied in the literature [4, 7, 8] both with and without noise in the data. In the following of this chapter, we will focus on the matched scenario, that is, the setting where $K = M$, for which perfect generalization is expected, as the student has enough resources to learn the rule [4, 7].

1.6 Dynamical equations through synthetic data

Let us now introduce the distribution of the data points. Following the works of Saad and Solla [4, 7, 14] we are able to compute dynamical equations for the order parameters by explicitly computing the averages that arise from taking the thermodynamic limit. Let us now give some information about the input space and in particular about the probability distribution. Assume now that $\xi \sim \mathcal{N}(0, I_N)$ be an N-dimensional gaussian vector with uncorrelated components and unitary variance. As a consequence, the distribution of the random vector of preactivations $Z = (X_1,\dots,X_K,Y_1,\dots,Y_M)$ is a $(K+M)$-dimensional gaussian with zero mean and covariance matrix

$$\tilde{C} = \begin{bmatrix} Q & R \\ R^T & T \end{bmatrix},$$

where we have defined $Q = (Q_{ik})_{i=1\ldots,K}^{k=1\ldots,K}$, $R = (R_{in})_{i=1\ldots,K}^{n=1\ldots,M}$, $T = (T_{mn})_{m=1\ldots,M}^{n=1\ldots,M}$. We then re-label the preactivations X_i, Y_n as follows:

$$\begin{aligned} &X_1 \mapsto Z_1 \\ &\cdots \\ &X_K \mapsto Z_K \\ &Y_1 \mapsto Z_{K+1} \\ &\cdots \\ &Y_M \mapsto Z_{K+M} \end{aligned}$$

so that we can compute the averages in (1.5) by computing:

$$I_2(i,j) = \langle g(Z_i)g(Z_j)\rangle \overset{(A.2)}{=} \frac{2}{\pi}\arcsin\left(\frac{C_{ij}}{\sqrt{C_{ii}+1}\sqrt{C_{jj}+1}}\right)$$

where the mean is a two-dimensional average over the marginal distribution of (Z_i, Z_j) (case $i \neq j$), or over the one-dimensional distribution of Z_i (case $i = j$). In both cases, the distribution is a gaussian with zero mean and covariance matrix, C the projection of $\tilde{C}$ with respect to the indices i and j, that is, $C = (\tilde{C}_{rs})_{r,s\in\{i,j\}}$. We refer to appendix A.2 for the explicit computation of I_2 (which is found in [4, 7]), which allows us to write explicitly the generalization error in terms of the order parameters [4, 7] as follows:

$$\begin{aligned} \mathcal{E}(\mathfrak{S}) = \frac{1}{\pi}\{ &\sum_{m,n=1}^{M} v_m v_n \arcsin\frac{T_{mn}}{\sqrt{(1+T_{mm})(1+T_{nn})}} \\ &- 2\sum_{i=1}^{K}\sum_{n=1}^{M} h_i v_n \arcsin\frac{R_{in}}{\sqrt{(1+Q_{ii})(1+T_{nn})}} \\ &+ \sum_{i,k=1}^{K} h_i h_k \arcsin\frac{Q_{ik}}{\sqrt{(1+Q_{ii})(1+Q_{kk})}}\}. \end{aligned} \tag{1.13}$$

1.6.1 From the difference equation to the ODEs

In the same way as it was done in the literature ([1, 4, 7]) using the Gradient Descent update in (1.8) and (1.9), one can also write the order parameters at each step of the

learning process as follows:

$$\begin{aligned} J_i^{\mu+1} \cdot J_k^{\mu+1} &= (J_i^\mu - \frac{\eta_J}{\sqrt{N}}\Delta^\mu\, h_i^\mu\, g'(X_i^\mu)\,\xi^\mu) \cdot (J_k^\mu - \frac{\eta_J}{\sqrt{N}}\Delta^\mu\, h_k^\mu\, g'(X_k^\mu)\,\xi^\mu) \\ &= J_i^\mu \cdot J_k^\mu - \frac{\eta_J}{\sqrt{N}}\Delta^\mu\, h_i^\mu\, g'(X_i^\mu)\, J_k^\mu \cdot \xi^\mu - \frac{\eta_J}{\sqrt{N}}\Delta^\mu\, h_k^\mu\, g'(X_k^\mu)\, J_i^\mu \cdot \xi^\mu \\ &+ \frac{\eta_J^2}{N}(\Delta^\mu)^2\, h_i^\mu\, g'(X_i^\mu)\, h_k^\mu\, g'(X_k^\mu)\,\xi^\mu \cdot \xi^\mu \\ &= J_i^\mu \cdot J_k^\mu - \eta_J\,\Delta^\mu\, h_i^\mu\, g'(X_i^\mu)\, X_k^\mu - \eta_J\,\Delta^\mu\, h_k^\mu\, g'(X_k^\mu)\, X_i^\mu \\ &+ \eta_J^2(\Delta^\mu)^2\, h_i^\mu\, g'(X_i^\mu)\, h_k^\mu\, g'(X_k^\mu)\,\frac{||\xi^\mu||^2}{N}. \end{aligned} \tag{1.14}$$

If we divide everything by N we find that:

$$\begin{aligned} \delta Q_{ik} = Q_{ik}^{\mu+1} - Q_{ik}^\mu = &- \frac{\eta_J}{N}\,\Delta^\mu\, h_i^\mu\, g'(X_i^\mu)\, X_k^\mu \\ &- \frac{\eta_J}{N}\,\Delta^\mu\, h_k^\mu\, g'(X_k^\mu)\, X_i^\mu \\ &+ \frac{\eta_J^2}{N}(\Delta^\mu)^2\, h_i^\mu\, g'(X_i^\mu)\, h_k^\mu\, g'(X_k^\mu)\,\frac{||\xi^\mu||^2}{N}, \end{aligned} \tag{1.15}$$

which is a difference equation for the discrete evolution of the overlap Q_{ik}. This is a powerful tool for analyzing the dynamical evolution of the system. In principle, one could even avoid training the network, but still have a formula that computes the order parameter during a training, provided that an initial condition was given. However, one must take into account the stochasticity that arises from the terms that depend explicitly on ξ^μ. In order to get rid of this randomness, we can perform a thermodynamic limit in (1.15). By doing so, we can think of $\tau = \mu/N$ as a continuous-time variable and retrieve the differential equation [4, 7]:

$$\begin{aligned} \frac{\mathrm{d}Q_{ik}}{\mathrm{d}\tau} = &- \eta_J\, h_i\, \langle g'(X_i)\, X_k\, \Delta\rangle \\ &- \eta_J\, h_k\, \langle g'(X_k)\, X_i\, \Delta\rangle \\ &+ \eta_J^2\, h_i\, h_k\, \langle g'(X_i) g'(X_k) \Delta^2\rangle. \end{aligned} \tag{1.16}$$

Remark. *In principle, taking the thermodynamic limit should not make the average $\langle\cdot\rangle$ appear. The reason of this appearance is a self-averaging property, deeply analyzed in the literature [10, 15]. More specifically, a quantity is self-averaging if its value in the N-large limit does not depend on the value it takes at the single point ξ^μ, but can be written through an average over the whole distribution of the data ξ.*

Remark. *Recall that ξ is an N-dimensional vector with uncorrelated components, zero mean and unitary variance, meaning that in the thermodynamic limit we have that $\frac{||\xi||^2}{N} \xrightarrow{N\to+\infty} 1$ almost surely, by the Kolmogorov's Strong Law of Large Numbers (see e.g. [16, Theorem 20.2]).*

Remark. *Equation* (1.16) *provides a powerful tool for understanding the behavior of the network. Moreover, since the averages above only involve functions of the preactivations, they can be computed using the low-dimensional distribution we have used so far.*

In particular, one can observe that the quantity above can be written as:

$$\begin{aligned}\frac{\mathrm{d}Q_{ik}}{\mathrm{d}\tau} =& \eta_J\, h_i \left[\sum_{m=1}^{M} v_m \langle g'(X_i)\, X_k\, g\,(Y_m)\rangle - \sum_{j=1}^{K} h_j \langle g'(X_i)\, X_k\, g\,(X_j)\rangle\right] \\ & \eta_J\, h_k \left[\sum_{m=1}^{M} v_m \langle g'(X_k)\, X_i\, g\,(Y_m)\rangle - \sum_{j=1}^{K} h_j \langle g'(X_k)\, X_i\, g\,(X_j)\rangle\right] \\ & + \eta_J^2\, h_i\, h_k \left[\sum_{j,l=1}^{K} h_j h_l \langle g'(X_i) g'(X_k) g(X_j) g(X_l)\rangle\right. \\ & + \sum_{m,n=1}^{M} v_m v_n \langle g'(X_i) g'(X_k) g(Y_m) g(Y_n)\rangle \\ & \left. -2\sum_{j=1}^{K}\sum_{m=1}^{M} h_j v_m \langle g'(X_i) g'(X_k) g(X_j) g(Y_m)\rangle\right],\end{aligned} \tag{1.17}$$

which involves only averages of the type

$$I_3(i,j,k) = \langle g'(Z_i)\, Z_j\, g(Z_k)\rangle \text{ with } i,j \in \{1,\dots,K\},\, k \in \{1,\dots,K+M\}, \tag{1.18}$$

and

$$I_4(i,j,k,l) = \langle g'(Z_i)\, g'(Z_j)\, g(Z_k)\, g(Z_l)\rangle \text{ with } i,j \in \{1,\dots,K\},\, k,l \in \{1,\dots,K+M\}, \tag{1.19}$$

and can be rewritten as

$$\begin{aligned}\frac{\mathrm{d}Q_{ik}}{\mathrm{d}\tau} =& \eta_J\, h_i \left[\sum_{m=1}^{M} v_m\, I_3(i,k,K+m) - \sum_{j=1}^{K} h_j\, I_3(i,k,j)\right] \\ & \eta_J\, h_k \left[\sum_{m=1}^{M} v_m\, I_3(k,i,K+m) - \sum_{j=1}^{K} h_j\, I_3(k,i,j)\right] \\ & + \eta_J^2\, h_i\, h_k \left[\sum_{j,l=1}^{K} h_j\, h_l\, I_4(i,k,j,l)\right. \\ & + \sum_{m,n=1}^{M} v_m\, v_n\, I_4(i,k,K+m,K+n) \\ & \left. -2\sum_{j=1}^{K}\sum_{m=1}^{M} h_j\, v_m\; I_4(i,k,j,K+m)\right],\end{aligned} \tag{1.20}$$

We refer to A.3 and A.4 for the computation of this quantities (firstly introduced in [4, 7]), that can be performed explicitly, making equation (1.17) a closed-form ODE that can be solved numerically as it was done in [4, 7]. We rely on appendix B for the algebraic steps that lead to the differential equation of R and h:

$$\frac{\mathrm{d}R_{in}}{\mathrm{d}\tau} = \eta_J\, h_i \left[\sum_{m=1}^{M} v_m\, I_3(i, K+n, K+m) - \sum_{j=1}^{K} h_j\, I_3(i, K+n, j)\right], \tag{1.21}$$

$$\frac{\mathrm{d}h_i}{\mathrm{d}\tau} = \eta_h \left[\sum_{m=1}^{M} v_m\, I_2(K+m, i) - \sum_{j=1}^{K} h_j\, I_2(j, i)\right]. \tag{1.22}$$

Remark. *We remark that the above differential equations are exact in the thermodynamic limit, meaning that when applying all the theoretical results presented so far, one must take into account the effect of possibly small values of N, making the ODEs less accurate.*

1.7 Numerical simulations

In this section, we provide a comparison between the dynamical equations found in 1.6.1 and the behavior of the network during an actual training session using the backpropagation of error in the online learning regime. Firstly, we will study the case of a perceptron learning a task provided by a perceptron, in order to reproduce some results in [8]. Then, we will study the behavior of the so-called soft-committee machine [7], i.e., a committee machine where the head weights are constant during the whole procedure. In particular, we will study the case where both teacher and student's head are initialized randomly. Later on we will discuss the main issues that can arise in the simulations. Finally, as in [17], we will analyze the general framework of a committee machine, where the heads are to be learned, too. Our analysis will focus on the noiseless case, although many of our theoretical results also work for the noisy setting, where noise is assumed on the input or on the output [14].

1.7.1 Perceptron

Simulations with the perceptron [8] are perhaps the easiest to implement and yet provide valuable information on the evolution of a neural network during training. The differential equations for a perceptron can be retrieved from those of a generic committee machine, by setting $K = M = 1$ and $h_1 = v_1$.
Simulation parameters and initializations:

- Input dimension: $N = 1000$;
- Learning rate: $\eta_J = 1$ (and $\eta_h = 0$);
- Optimizer: SGD optimizer with batch size = 1 (equivalent to Gradient Descent);
- Loss: scaled MSE loss;
- Teacher weight initialization: element-wise uncorrelated normal gaussians;
- Teacher head initialization: $v_1 = 1$;
- Student weight initialization: element-wise uncorrelated Gaussian with zero mean and variance $\sigma^2 = 0.01$;
- Student head initialization: $h_1 = 1$;

Using one simulation with input dimension $N = 1000$ shows that learning a perceptron without noise is a perfectly realizable scenario. We can rewrite Equation (1.13) for $K = M = 1$ and $h_1 = v_1 = 1$, and obtain

$$\mathcal{E}(\Pi) = \frac{1}{\pi}\{\arcsin\frac{T}{1+T} - 2\arcsin\frac{R}{\sqrt{(1+Q)(1+T)}} + \arcsin\frac{Q}{1+Q}\} \tag{1.23}$$

which shows that for $Q = R = T$ we have $\mathcal{E}(\Pi) = 0$. The results shown in Figures 1.1, 1.2, 1.3 confirm this trend, suggesting that perfect generalization is achieved when the teacher and the student have the exact same weight as the following holds:

$$\frac{||J-B||^2}{N} = \frac{||J||^2}{N} - 2\frac{J\cdot B}{N} + \frac{||B||^2}{N} = Q - 2R + T \overset{Q=R=T}{=} 0$$

yielding $J = B$.
Also note that the point $(Q, R) = (T, T)$ is an equilibrium point of the dynamical equations 1.20, 1.21, i.e., $\frac{\mathrm{d}Q}{\mathrm{d}\tau} = \frac{\mathrm{d}R}{\mathrm{d}\tau} = 0$ [8]. Indeed, such equations can be simplified to

$$\begin{aligned}
\frac{\mathrm{d}Q}{\mathrm{d}\tau} &= \eta_J\,[\,I_3(1,1,2) - I_3(1,1,1)] \\
&+ \eta_J\,[\,I_3(1,1,2) - I_3(1,1,1)] \\
&+ \eta_J^2\,[I_4(1,1,1,1) + I_4(1,1,2,2) - 2\,I_4(1,1,1,2)] \\
&= 2\,\eta_J\,[I_3(1,1,2) - I_3(1,1,1)] \\
&+ \eta_J^2\,[I_4(1,1,1,1) + I_4(1,1,2,2) - 2\,I_4(1,1,1,2)]\,.
\end{aligned} \tag{1.24}$$

After computing the integrals above using A.3, A.12 we obtain:

$$\begin{aligned}\frac{\mathrm{d}Q}{\mathrm{d}\tau} &= \eta_J \frac{4}{\pi(1+Q)} \left[\frac{R}{\sqrt{(1+Q)(1+T)-R^2}} - \frac{Q}{\sqrt{1+2Q}}\right] \\ &+ \eta_J^2 \frac{4}{\pi^2\sqrt{1+2Q}} \left[\arcsin\left(\frac{Q}{1+3Q}\right)\right. \\ &- 2\arcsin\left(\frac{R}{\sqrt{1+3Q}\sqrt{(1+2Q)(1+T)-2R^2}}\right) \\ &\left.+ \arcsin\left(\frac{(1+2Q)T-2R^2}{(1+2Q)(1+T)-2R^2}\right)\right].\end{aligned} \tag{1.25}$$

Similarly, the evolution of R is given by:

$$\frac{\mathrm{d}R}{\mathrm{d}\tau} = \eta_J \left[I_3(1,2,2) - I_3(1,2,1)\right] \tag{1.26}$$

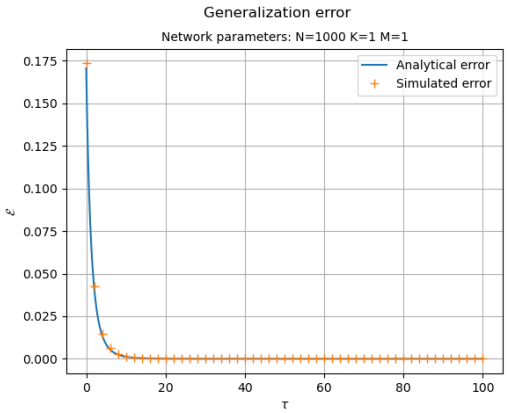
which becomes:

$$\frac{\mathrm{d}R}{\mathrm{d}\tau} = \eta_J \frac{2}{\pi(1+Q)} \left[\frac{(1+Q)T-R^2}{\sqrt{1+Q+T+QT-R^2}} - \frac{R}{\sqrt{1+2Q}}\right]. \tag{1.27}$$

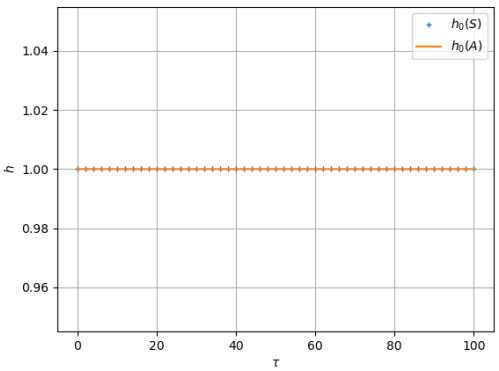
This is the same result achieved in [8], where $T = 1$ was studied.

Remark. *The choice of the learning rate is crucial, both for the learning procedure and for numerical integration. Many theoretical results have been proposed to select the appropriate values for this parameter [7, 8]. Recent works have proposed applying control theory to tackle this problem and find optimal values for this parameter [3].*

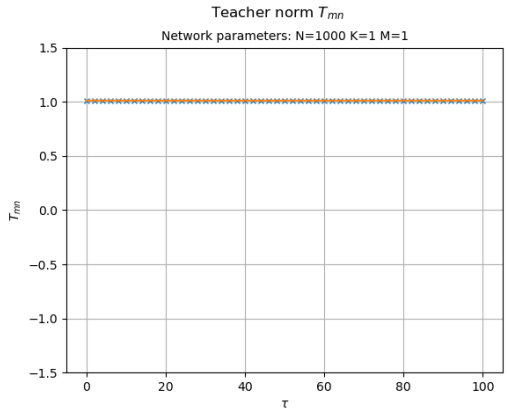
Figure 1.1a illustrates the generalization error obtained by both solving the differential equations and using a simulation on a network. As expected, perfect generalization is achieved really fast. Moreover, Figures 1.1b, 1.1c, 1.1d show the parameters of the teacher network, together with the head of the student network. Figures 1.2a, 1.3a show that the behavior of the student network is characterized, as we said before, by the convergence of the order parameters towards the equilibrium point $(Q, R) = (T, T)$. Also note that since the differential equations that we computed in Section 1.6.1 are exact in the thermodynamic limit [4, 7], we compute the absolute error between the simulation and the analytical curve and find that the displacement is of order $\frac{1}{\sqrt{N}}$.



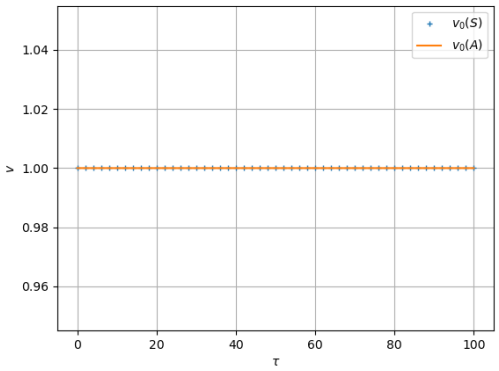
(a) Generalization error.



(b) Student head.

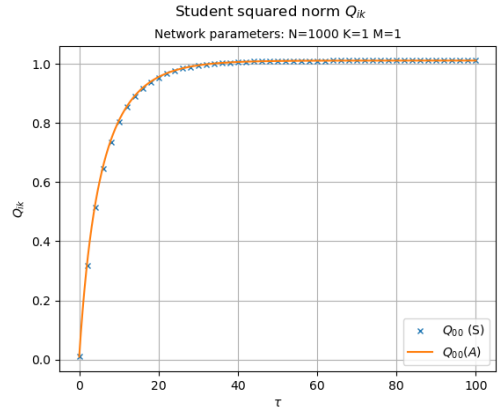


(c) Teacher norm.

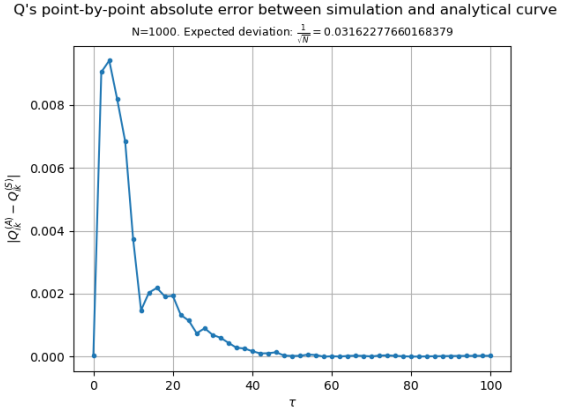


(d) Teacher head.

Figure 1.1: Generalization error and parameters of the perceptrons. ODEs (A) and Simulations (S) are shown.

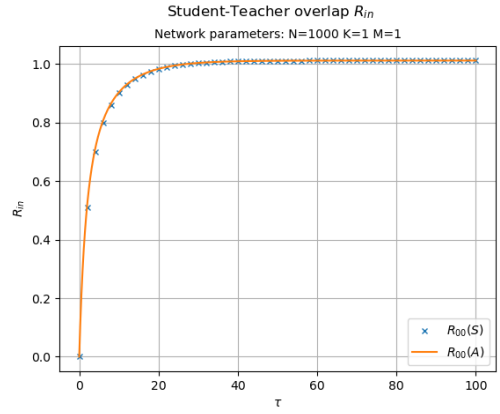


(a) Evolution of the Q_{ik} parameters during training

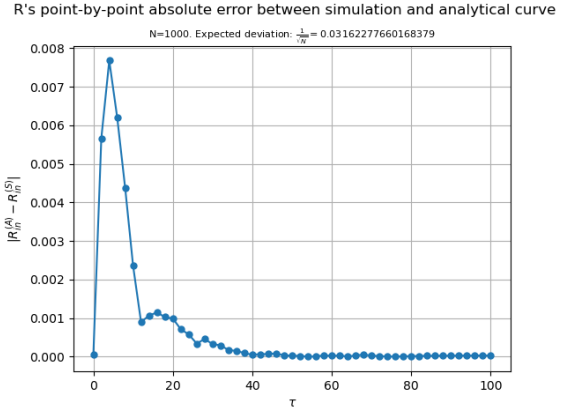


(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.2: Evolution of the student norm. Crossed symbols indicate order parameter computed during one training procedure with input dimension $N = 1000$. Learning rate $\eta_J = 1$ was used. ODEs (A) and Simulations (S) are shown.



(a) Evolution of the R_{in} parameters during training



(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.3: Evolution of the student-teacher overlap. Crossed symbols indicate order parameter computed during one training procedure with input dimension $N = 1000$. Learning rate $\eta_J = 1$ was used. ODEs (A) and Simulations (S) are shown.

1.7.2 Soft committee machine

In this section, we analyze the convergence to generalization of two committee machines with $K = M = 3$ hidden units. Firstly, we analyze the phenomenology of the so-called soft committee machine [7]: it is a committee machine where only the input-to-hidden weights are trainable parameters, while the hidden-to-output weights are fixed and are shared between the student and the teacher. From the combinatorial point of view, we must point out the following fact: when the heads of both the teacher and the student are set to be equal to the same constant c (i.e. $h_i = v_i = c$) this induces a symmetry in the problem that must be taken into account. Indeed, there is no reason to think that the input-to-hidden weights of the students (the columns of the matrix $\mathbf{J}$) will align with those of the teacher following the same ordering. Let us consider the case $K = M = 2$, so that the matrices of weights are:

$$\mathbf{J} = \begin{bmatrix} J_1 & J_2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} B_1 & B_2 \end{bmatrix}.$$

At convergence to perfect generalization, one may find that $J_1 = B_1$ and $J_2 = B_2$, but this is not always the case, since also $J_1 = B_2$ and $J_2 = B_1$ are a possibility, and indeed this may happen during simulations. This is more true than ever in the common setting in which the teacher's weights are orthogonal to each other and of unit norm. This is also referred to as the ungraded teacher setting [7], for which teacher weights are taken as $T_{nm} = \delta_{nm}$ and δ_{nm} is the Kronecker delta.

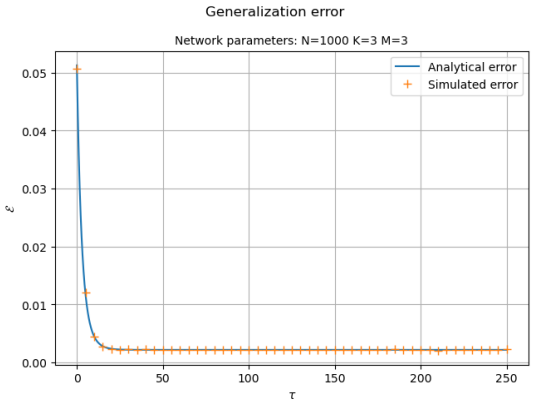
As can be seen in Figure 1.4a, a plateau (i.e., a constant value of the error that is not zero yet) is reached and maintained for a long time in a suboptimal solution. This is because the symmetry affects how the student's weights align to the teacher, leaving some residual information that is shared among vectors [7]. This can be seen in Figure 1.5a as the mixed overlaps Q_{ik} for $k \neq i$ are not zero, unlike those of the teacher (see Figures 1.5, 1.6). If we go deeper in time, one can see that after a long symmetric trapping [7], the generalization error goes to zero, and the student weights specialize to those of the teacher (see, e.g., [7]).

Simulation parameters and initializations:

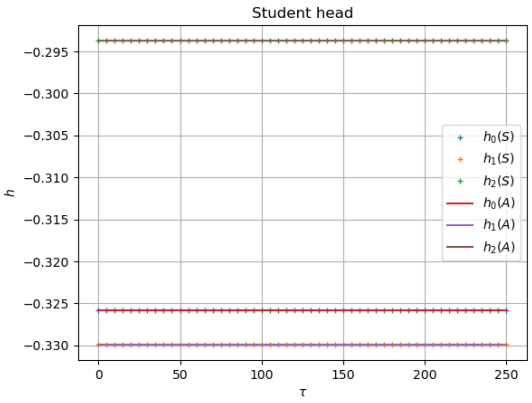
- $K = M = 3$;
- Learning rate: $\eta_J = 1$ (and $\eta_h = 0$);
- Optimizer: SGD optimizer with batch size = 1 (equivalent to Gradient Descent);
- Loss: scaled MSE loss;

For the ungraded case (Figures 1.4, 1.5, 1.6):

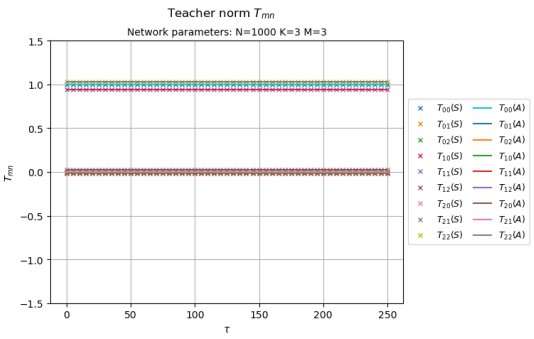
- Input dimension: $N = 1000$;
- Teacher weight initialization: element-wise uncorrelated normal gaussian;
- Teacher head initialization: $v_i = g_i$ with g_i random realization of a normal gaussian;
- Student weight initialization: element-wise uncorrelated gaussian with zero mean and variance $\sigma^2 = 0.01$;
- Student head initialization: $h_i = v_i$ for any i.



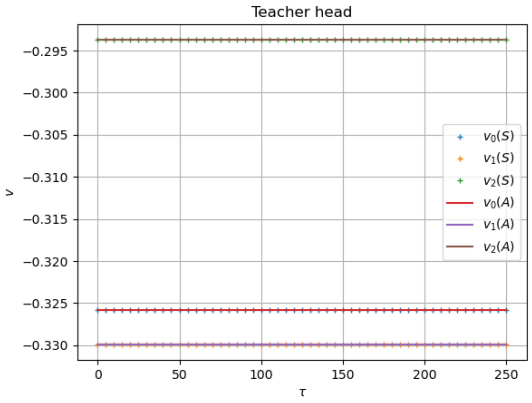
(a) Generalization error.



(b) Student head.

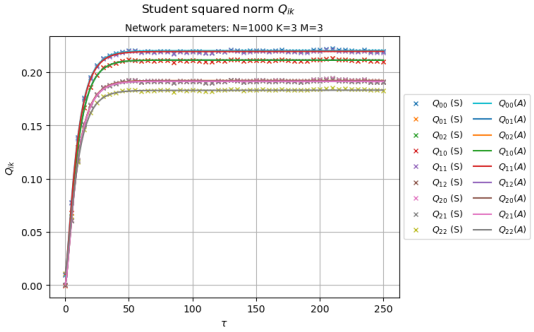


(c) Teacher norm.

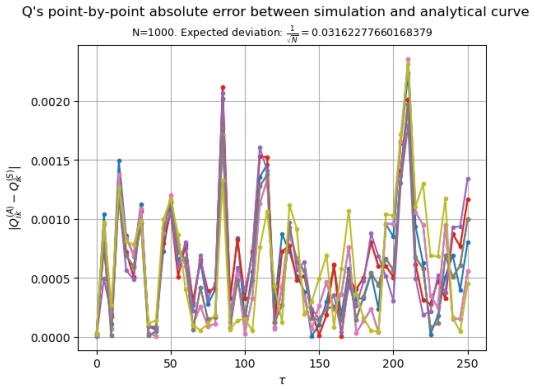


(d) Teacher head.

Figure 1.4: Generalization error and constants of the soft committee machines. The setting is that of an ungraded teacher (i.e. $T_{nm} = \delta_{nm}$). ODEs (A) and Simulations (S) are shown.

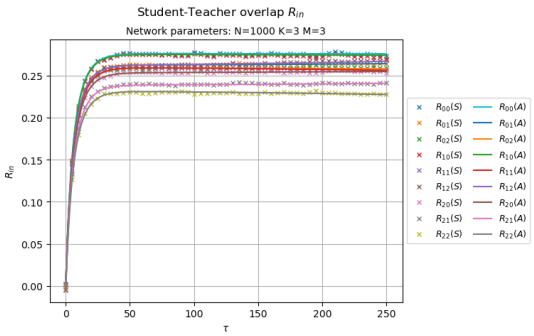


(a) Evolution of the Q_{ik} parameters during training

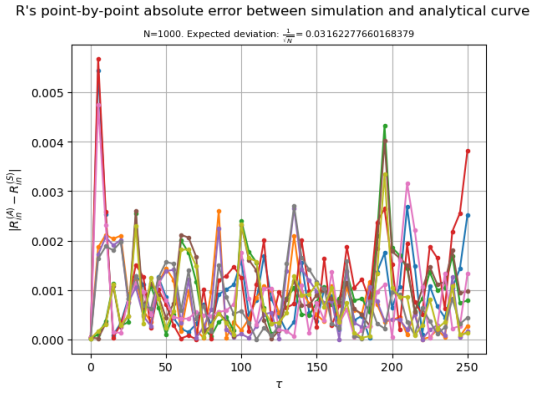


(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.5: Evolution of the student norm. Crossed symbols indicate results of training, averaged over 10 training procedures. The setting is that of an ungraded teacher (i.e. $T_{nm} = \delta_{nm}$). ODEs (A) and Simulations (S) are shown.



(a) Evolution of the R_{in} parameters during training



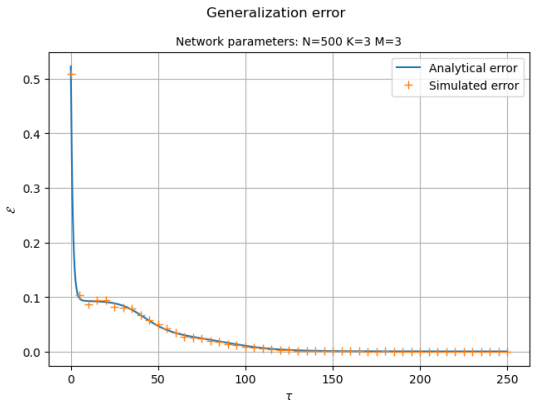
(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.6: Evolution of the student overlap. Crossed symbols indicate results of training, averaged over 10 training procedures. The setting is that of an ungraded teacher (i.e. $T_{nm} = \delta_{nm}$). ODEs (A) and Simulations (S) are shown.

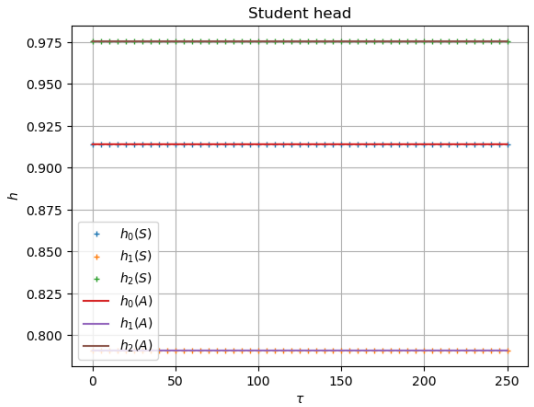
For the graded case (Figures 1.7, 1.8, 1.9):

- Input dimension: $N = 500$;
- Teacher weight initialization: element-wise uncorrelated normal gaussian, normalized to have $T_{nm} = n\,\delta_{nm}$;
- Teacher head initialization: $v_i = 1 + g_i$ with g_i random realization of a gaussian with zero mean and variance $\sigma^2 = 0.01$;
- Student weight initialization: element-wise uncorrelated gaussian with zero mean and variance $\sigma^2 = 0.01$;
- Student head initialization: $h_i = v_i$ for any i.

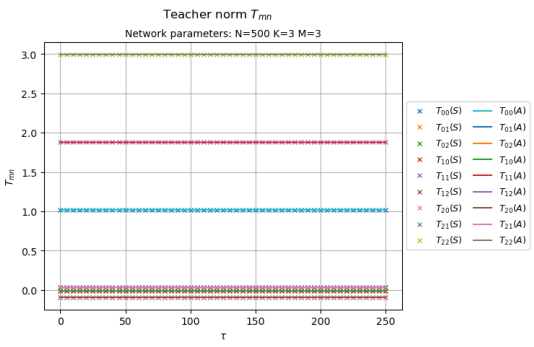
In the graded teacher case (Figures 1.7, 1.8, 1.9) we observe the typical behavior that we saw for the perceptron: each of the student's vector aligns with one of the teacher, and mixed terms Q_{ik} and R_{in} go to zero, as the orthogonality of the teacher's vectors suggests. Also note that no symmetric trapping is observed and even in the short-time period, generalization is reached [4]. Again, since the ODEs are exact in the thermodynamic limit, one can compute the point-wise error between the simulations and analytical solutions of the ODEs and find that the displacement is always of order $\frac{1}{\sqrt{N}}$.



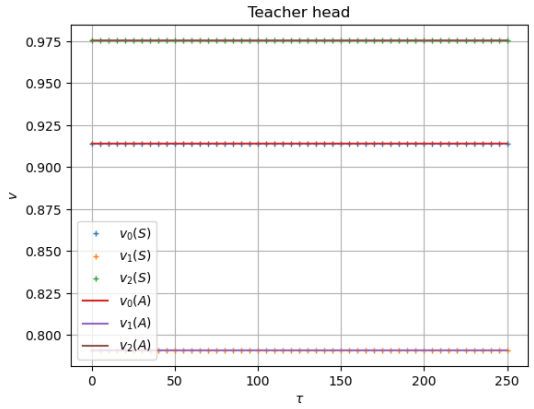
(a) Generalization error.



(b) Student head.

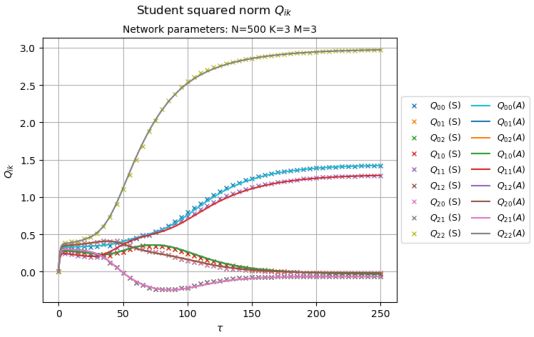


(c) Teacher norm.

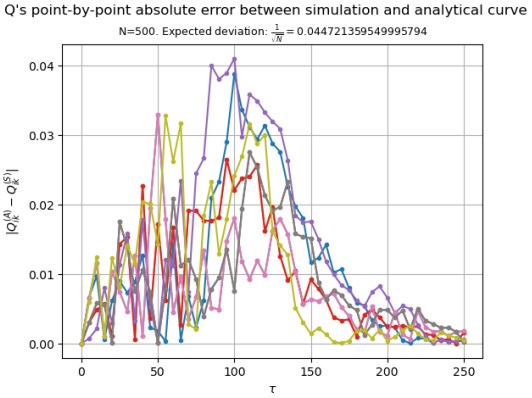


(d) Teacher head.

Figure 1.7: Generalization error and constants of the soft committee machines in the graded teacher setting (i.e. $T_{nm} = n\,\delta_{nm}$). ODEs (A) and Simulations (S) are shown.

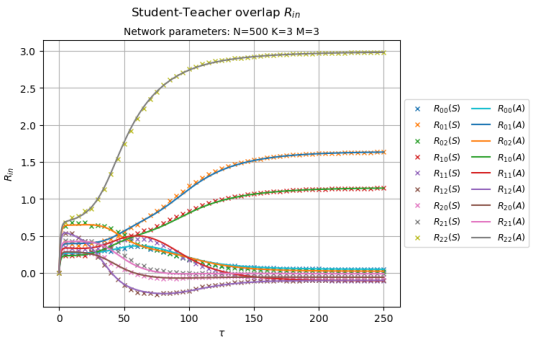


(a) Evolution of the Q_{ik} parameters during training

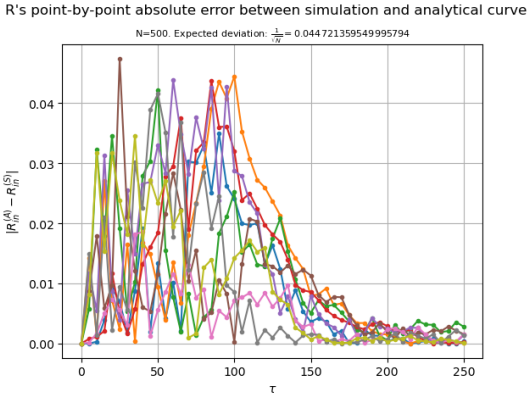


(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.8: Evolution of the student norm. Crossed symbols indicate results of training. The setting is that of a graded teacher (i.e. $T_{nm} = n\,\delta_{nm}$). ODEs (A) and Simulations (S) are shown.



(a) Evolution of the R_{in} parameters during training



(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.9: Evolution of the student norm. Crossed symbols indicate results of training. The setting is that of a graded teacher (i.e. $T_{nm} = n\,\delta_{nm}$). ODEs (A) and Simulations (S) are shown.

1.7.3 Committee machine

In this section, we study the evolution of a committee machine, for which also the head weights are trained. All the observations that we have made so far fit perfectly to the general case. The setting is that of a graded teacher (i.e. $T_{nm} = n\,\delta_{nm}$). We will consider case $K = M = 2$ to ease the discussion, but all subsequent observations are generalizable to a more general case. When training both layers of the student network, an observation is to be made. Since the error function is an odd function, a change of sign in a student vector is compensated by a change of sign in the corresponding hidden unit (see for example the flip of sign in R_{01} and h_0 in Figures 1.13a, 1.14a with respect to the sign of the corresponding head of the teacher). That is, assume that the student's weights are:

$$\mathbf{J} = \begin{bmatrix} J_1 & J_2 \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} h_1 & h_2 \end{bmatrix}$$

then this network produces the output

$$\Omega(\mathbf{J}, \mathbf{h}; \xi) = h_1\, g\left(\frac{J_1 \cdot \xi}{\sqrt{N}}\right) + h_2\, g\left(\frac{J_2 \cdot \xi}{\sqrt{N}}\right).$$

On the other hand, let us consider the following student:

$$\hat{\mathbf{J}} = \begin{bmatrix} -J_1 & J_2 \end{bmatrix} \quad \hat{\mathbf{h}} = \begin{bmatrix} -h_1 & h_2 \end{bmatrix}$$

which produces the output

$$\Omega(\hat{\mathbf{J}}, \hat{\mathbf{h}}; \xi) = -h_1\, g\left(\frac{-J_1 \cdot \xi}{\sqrt{N}}\right) + h_2\, g\left(\frac{J_2 \cdot \xi}{\sqrt{N}}\right)$$

which is equal to $\Omega(\mathbf{J}, \mathbf{h}; \xi)$, by oddity. This is exactly what happens in our simulation (Figures 1.11, 1.13a, 1.14a).
Simulation parameters and initializations:

- Input dimension $N = 1000$;
- $K = M = 2$;
- Learning rate: $\eta_J = 1$ and $\eta_h = 1$;
- Optimizer: SGD optimizer with batch size = 1 (equivalent to Gradient Descent);
- Loss: scaled MSE loss;
- Teacher weight initialization: element-wise uncorrelated normal gaussian, normalized to have $T_{nm} = n\,\delta_{nm}$;
- Teacher head initialization: element-wise normal gaussian;

- Student weight and head initialization: element-wise uncorrelated gaussian with zero mean and variance $\sigma^2 = 0.01$;

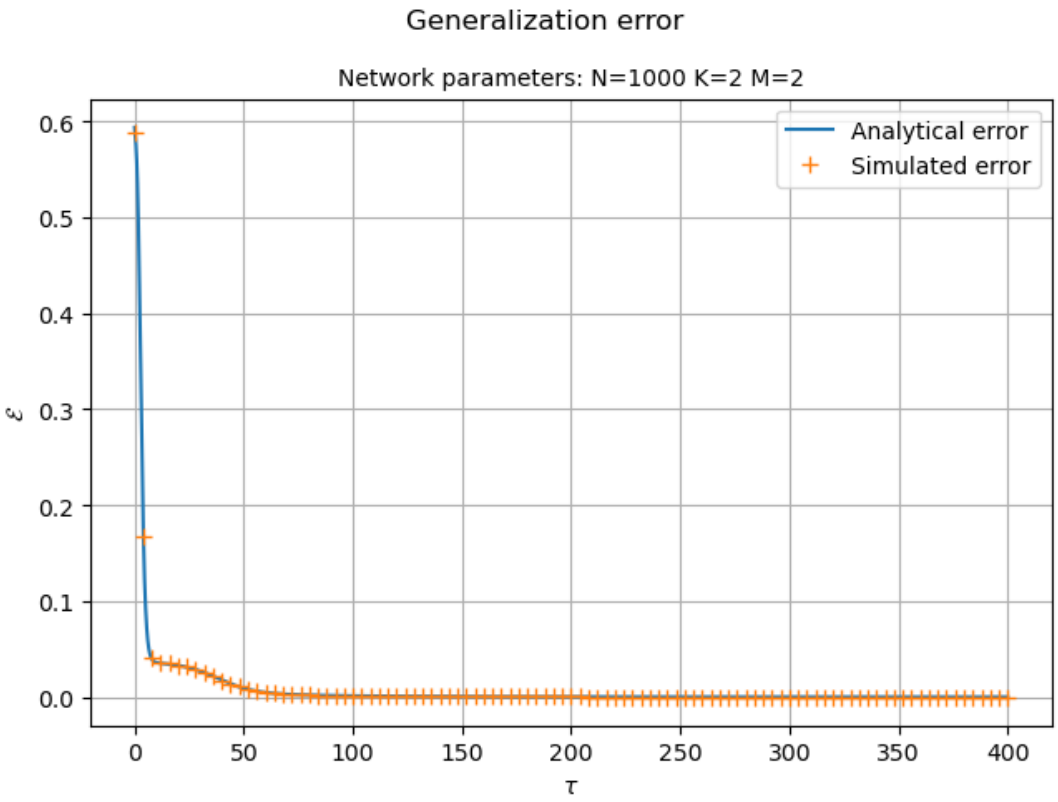
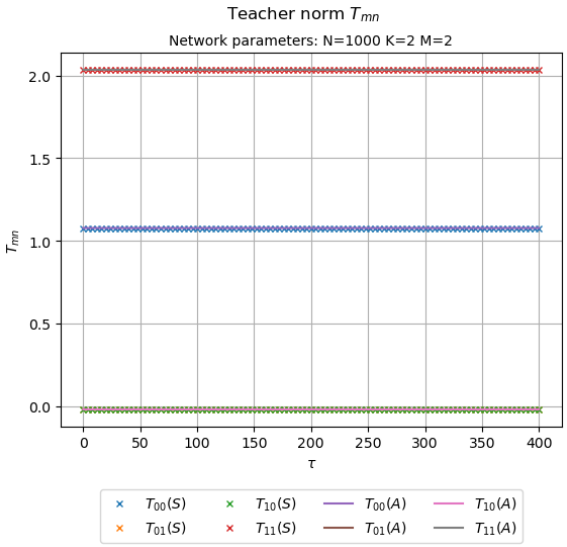
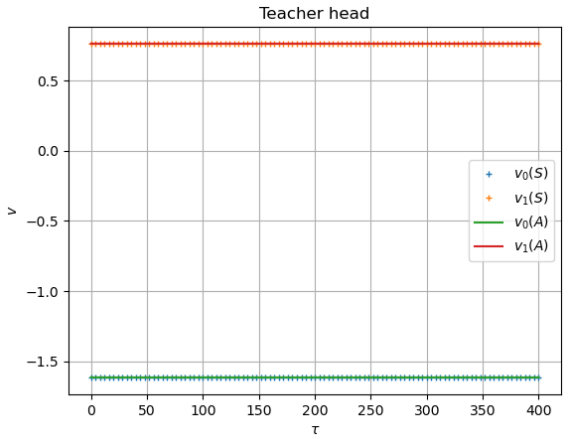


Figure 1.10: Generalization error. ODEs (A) and Simulations (S) are shown.

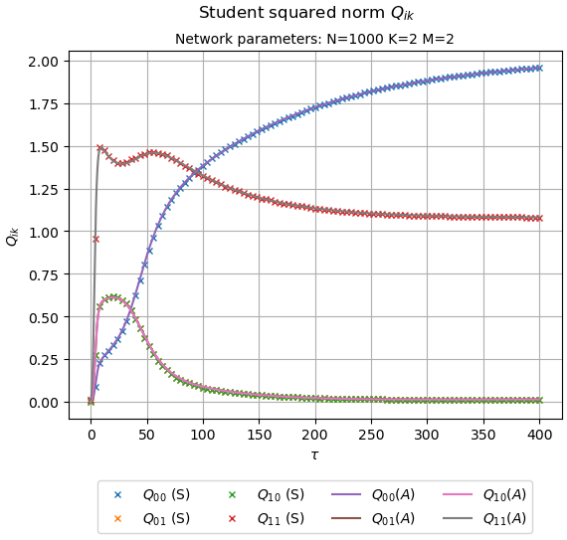


(a) Teacher norm.

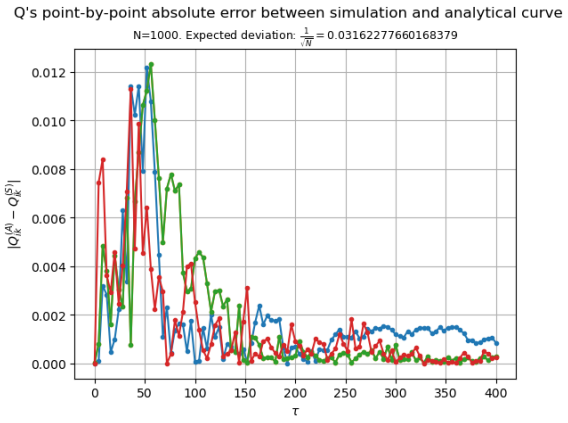


(b) Teacher head.

Figure 1.11: Constants of the teacher committee machine. ODEs (A) and Simulations (S) are shown.

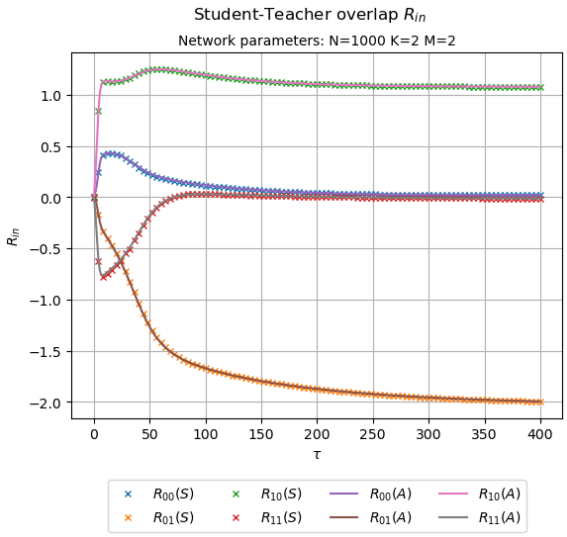


(a) Evolution of the Q_{ik} parameters during training

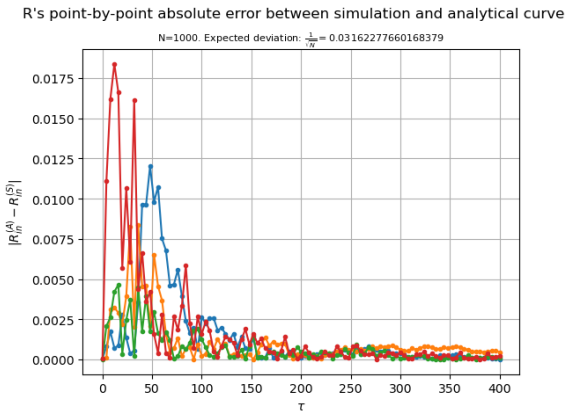


(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.12: Evolution of the student norm and absolute error.

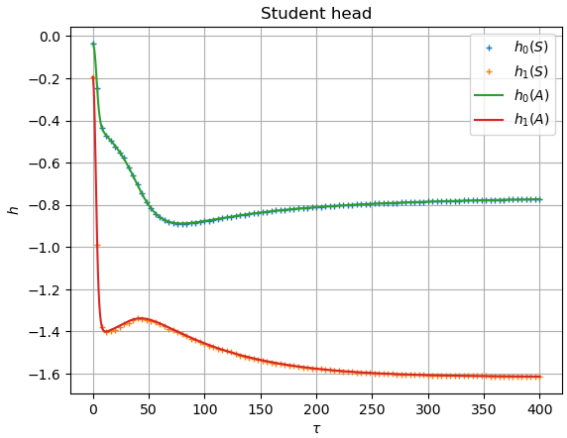


(a) Evolution of the R_{in} parameters during training

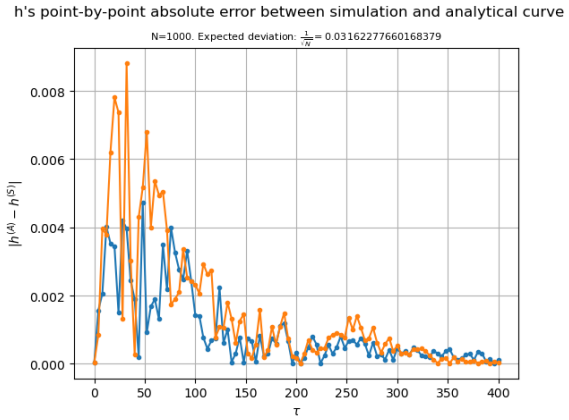


(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.13: Evolution of the student norm and absolute error. ODEs (A) and Simulations (S) are shown.



(a) Evolution of the h_i parameters during training.



(b) Error between the simulation and the analytical solution of the ODE.

Figure 1.14: Evolution of the student norm and absolute error. ODEs (A) and Simulations (S) are shown.

Chapter 2

Continual learning in neural networks

Transfer learning [9] is one of the prominent techniques of modern machine learning. The key idea is the following: assume that we have a pre-trained neural network at hand. Assume that this network was trained to solve a task (task A, also called *source*). When we want to save time in solving another task (task B, also called *target*), which is somehow related to the first one, we can use the *source* as initialization for the network that is learning the *target*. This is far from unreasonable, as common sense suggests that learning task B from scratch is more difficult than learning it from previous knowledge in a related field (task A). *Continual learning* is the learning scenario in which learning B and not forgetting A are pursued at the same time. The concept of *task switch* plays a crucial role in continual learning: it is the learning step at which one should start learning task B and stopping the update of the network for task A. This chapter is devoted to reproducing the main results in [1].

2.1 Multi-headed approach to transfer learning

In this work, we will follow a multi-headed approach to continual learning [18]. In this new setting, we will exploit all the quantities in [1] that are needed to analyze the behavior of the networks under the continual learning framework. Following [1], we define two teachers, each of which defines a different task. We represent them as committee machines $\mathfrak{T}^{\diamondsuit} = (\mathbf{B}^{\diamondsuit}, \mathbf{v}^{\diamondsuit}, g)$ (for task A) and $\mathfrak{T}^{\heartsuit} = (\mathbf{B}^{\heartsuit}, \mathbf{v}^{\heartsuit}, g)$ (for task B) and with the same number of hidden units M. Then, following the multi-headed approach, the source and target student networks share the same input-to-hidden matrix weight $\mathbf{J}$, but have different head weights. Unlike previous chapters, $\mathbf{J}$ is a $K \times N$ matrix whose rows are the

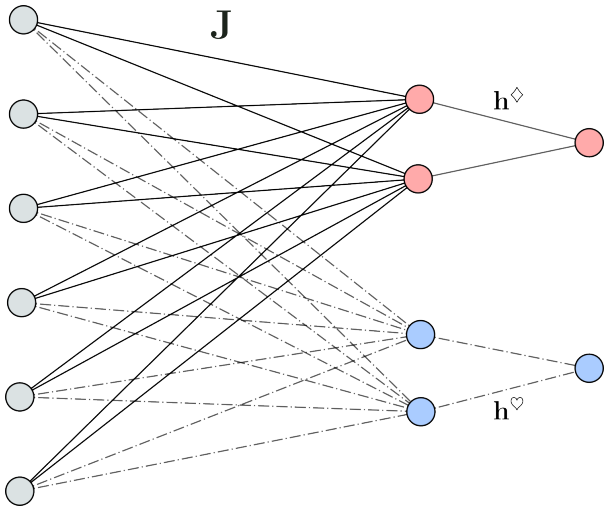


Figure 2.1: Pictorial representation of a multi-headed student. Dash-dotted lines represent the same weights $\mathbf{J}$ as the continuous lines, but when task B is studied.

N dimensional weights that define the i-th hidden unit preactivation; this will ease the computations and avoid dimensional mismatch when computing the order parameters. The same holds for $\mathbf{B}$. Then, we define a unique student network $\mathfrak{S} = (\mathbf{J}, \mathbf{h}, g)$ with K hidden units, where $\mathbf{h} = \begin{bmatrix} \mathbf{h}^{\diamondsuit} & \mathbf{h}^{\heartsuit} \end{bmatrix}$. We will write $\mathfrak{S}^{\diamondsuit}$ whenever task A is active, i.e. when we are studying the committee machine $(\mathbf{J}, \mathbf{h}^{\diamondsuit}, g)$. The same consideration works by writing $\mathfrak{S}^{\heartsuit}$. Thus, the output of the student network $\mathfrak{C}$ is a two-valued output, one for each task, that we define as:

$$\Omega(\mathfrak{C}; \xi) = \begin{bmatrix} \Omega(\mathfrak{C}^{\diamondsuit}; \xi) & \Omega(\mathfrak{C}^{\heartsuit}; \xi) \end{bmatrix}.$$

This above is more of a design detail that one can use when doing simulations. Then, at each training step we can define the generalization error, which also depends on the active task, as:

$$\mathcal{E}(\mathfrak{C}^*) = \frac{1}{2} \langle (\Omega(\mathfrak{C}^*; \xi) - (\mathfrak{T}^*; \xi))^2 \rangle$$

and $* \in \{\diamondsuit, \heartsuit\}$.

All quantities and formulas retrieved in Chapter 1 are the same for this new setting, with a clear distinction between the formulas that belong to task A and those belonging to task B.

2.1.1 Order parameters in continual learning

Like before, we can now define the order parameters in the continual learning setting [1]. To this end, define:

$$Q = \frac{\mathbf{J}\mathbf{J}^T}{N} \tag{2.1}$$

$$R = \frac{\mathbf{J}(\mathbf{B}^{\diamondsuit})^T}{N} \tag{2.2}$$

$$T = \frac{\mathbf{B}^{\diamondsuit}(\mathbf{B}^{\diamondsuit})^T}{N} \tag{2.3}$$

$$U = \frac{\mathbf{J}(\mathbf{B}^{\heartsuit})^T}{N} \tag{2.4}$$

$$S = \frac{\mathbf{B}^{\heartsuit}(\mathbf{B}^{\heartsuit})^T}{N} \tag{2.5}$$

$$V = \frac{\mathbf{B}^{\diamondsuit}(\mathbf{B}^{\heartsuit})^T}{N}. \tag{2.6}$$

V is typically called *task similarity* and we will discuss its role in continual learning in Section 2.2, inspired by [1, 2, 3].

2.1.2 Generalization error and ODEs

Before task switch (which we denote by $\hat{\tau}$), we are studying task A. We write explicitly all the quantities of interest, that are found in [1]. The generalization error is as follows:

$$\begin{aligned}\mathcal{E}(\mathfrak{S}^{\diamondsuit}) = \frac{1}{\pi}\{ &\sum_{m,n=1}^{M} v_m^{\diamondsuit} v_n^{\diamondsuit} \arcsin \frac{T_{mn}}{\sqrt{(1+T_{mm})(1+T_{nn})}} \\ &- 2\sum_{i=1}^{K}\sum_{n=1}^{M} h_i^{\diamondsuit} v_n^{\diamondsuit} \arcsin \frac{R_{in}}{\sqrt{(1+Q_{ii})(1+T_{nn})}} \\ &+ \sum_{i,k=1}^{K} h_i^{\diamondsuit} h_k^{\diamondsuit} \arcsin \frac{Q_{ik}}{\sqrt{(1+Q_{ii})(1+Q_{kk})}}\}.\end{aligned} \tag{2.7}$$

The covariance matrix of the low-dimensional averages is:

$$\tilde{C} = \begin{bmatrix} Q & R \\ R^T & T \end{bmatrix},$$

and for $\tau \leq \hat{\tau}$:

$$\begin{aligned}\frac{\mathrm{d}Q_{ik}}{\mathrm{d}\tau} =& \eta_J\, h_i^{\diamondsuit} \left[\sum_{m=1}^{M} v_m^{\diamondsuit}\, I_3(i,k,K+m) - \sum_{j=1}^{K} h_j^{\diamondsuit}\, I_3(i,k,j)\right] \\ & \eta_J\, h_k^{\diamondsuit} \left[\sum_{m=1}^{M} v_m^{\diamondsuit}\, I_3(k,i,K+m) - \sum_{j=1}^{K} h_j^{\diamondsuit}\, I_3(k,i,j)\right] \\ & + \eta_J^2\, h_i^{\diamondsuit}\, h_k^{\diamondsuit} \left[\sum_{j,l=1}^{K} h_j^{\diamondsuit}\, h_l^{\diamondsuit}\, I_4(i,k,j,l)\right. \\ & + \sum_{m,n=1}^{M} v_m^{\diamondsuit}\, v_n^{\diamondsuit}\, I_4(i,k,K+m,K+n) \\ & \left. -2\sum_{j=1}^{K}\sum_{m=1}^{M} h_j^{\diamondsuit}\, v_m^{\diamondsuit}\; I_4(i,k,j,K+m)\right],\end{aligned} \tag{2.8}$$

$$\frac{\mathrm{d}R_{in}}{\mathrm{d}\tau} = \eta_J\, h_i^{\diamondsuit} \left[\sum_{m=1}^{M} v_m^{\diamondsuit}\, I_3(i,K+n,K+m) - \sum_{j=1}^{K} h_j^{\diamondsuit}\, I_3(i,K+n,j)\right], \tag{2.9}$$

$$\frac{\mathrm{d}h_i^{\diamondsuit}}{\mathrm{d}\tau} = \eta_h \left[\sum_{m=1}^{M} v_m^{\diamondsuit}\, I_2(K+m,i) - \sum_{j=1}^{K} h_j^{\diamondsuit}\, I_2(j,i)\right]. \tag{2.10}$$

After task switch, i.e., when $\tau > \hat{\tau}$ the corresponding quantities are written as follows.

$$\begin{aligned}\mathcal{E}(\mathfrak{S}^{\heartsuit}) = \frac{1}{\pi}\{ & \sum_{p,q=1}^{M} v_p^{\heartsuit} v_q^{\heartsuit} \arcsin \frac{S_{pq}}{\sqrt{(1+S_{pp})(1+S_{qq})}} \\ & - 2\sum_{i=1}^{K}\sum_{p=1}^{M} h_i^{\heartsuit} v_p^{\heartsuit} \arcsin \frac{U_{ip}}{\sqrt{(1+Q_{ii})(1+S_{pp})}} \\ & + \sum_{i,k=1}^{K} h_i^{\heartsuit} h_k^{\heartsuit} \arcsin \frac{Q_{ik}}{\sqrt{(1+Q_{ii})(1+Q_{kk})}}\}.\end{aligned} \tag{2.11}$$

The covariance matrix of the low-dimensional averages is:

$$\tilde{C} = \begin{bmatrix} Q & U \\ U^T & S \end{bmatrix},$$

and for $\tau > \hat{\tau}$:

$$\begin{aligned}\frac{\mathrm{d}Q_{ik}}{\mathrm{d}\tau} =& \eta_J\, h_i^{\heartsuit} \left[\sum_{p=1}^{M} v_p^{\heartsuit}\, I_3(i,k,K+p) - \sum_{j=1}^{K} h_j^{\heartsuit}\, I_3(i,k,j)\right] \\ & \eta_J\, h_k^{\heartsuit} \left[\sum_{p=1}^{M} v_p^{\heartsuit}\, I_3(k,i,K+p) - \sum_{j=1}^{K} h_j^{\heartsuit}\, I_3(k,i,j)\right] \\ & + \eta_J^2\, h_i^{\heartsuit}\, h_k^{\heartsuit} \left[\sum_{j,l=1}^{K} h_j^{\heartsuit}\, h_l^{\heartsuit}\, I_4(i,k,j,l)\right. \\ & + \sum_{p,q=1}^{M} v_p^{\heartsuit}\, v_q^{\heartsuit}\, I_4(i,k,K+p,K+q) \\ & \left. -2\sum_{j=1}^{K}\sum_{p=1}^{M} h_j^{\heartsuit}\, v_p^{\heartsuit}\; I_4(i,k,j,K+p)\right],\end{aligned} \tag{2.12}$$

$$\frac{\mathrm{d}U_{ip}}{\mathrm{d}\tau} = \eta_J\, h_i^{\heartsuit} \left[\sum_{q=1}^{M} v_q^{\heartsuit}\, I_3(i,K+p,K+q) - \sum_{j=1}^{K} h_j^{\heartsuit}\, I_3(i,K+p,j)\right], \tag{2.13}$$

$$\frac{\mathrm{d}h_i^{\heartsuit}}{\mathrm{d}\tau} = \eta_h \left[\sum_{p=1}^{M} v_p^{\heartsuit}\, I_2(K+p,i) - \sum_{j=1}^{K} h_j^{\heartsuit}\, I_2(j,i)\right]. \tag{2.14}$$

2.2 Task similarity and impact on catastrophic forgetting

In this section, we will analyze how transfer and forgetting are affected by the similarity between tasks reproducing the work in [1]. First of all, we distinguish between similarity at level of features, i.e. similarity between first layers' weights, and similarity on the read-out level, i.e. similarity between hidden-to-output weights. Here, we will focus on the similarity between feature weights.

2.2.1 Transfer and forgetting

When talking about continual learning, two main quantities arise: transfer and forgetting. Transfer is the long-term capacity to learn a new task, while forgetting is the attitude of a network to forget the previous task. Let $\mathcal{E}_t^{\diamondsuit}$ be the generalization error at time t for task A, and $\mathcal{E}_t^{\heartsuit}$ be that of task B. Then we define [1]:

$$\text{Transfer at time } t\text{:} \quad \mathcal{T}_t = \mathcal{E}_{\hat{\tau}}^{\heartsuit} - \mathcal{E}_{\hat{\tau}+t}^{\heartsuit} \tag{2.15}$$

$$\text{Forgetting at time } t\text{:} \quad \mathcal{F}_t = \mathcal{E}_{\hat{\tau}+t}^{\diamondsuit} - \mathcal{E}_{\hat{\tau}}^{\diamondsuit} \tag{2.16}$$

Transfer could also be defined as the difference between the generalization error obtained with continual learning and the generalization error obtained by learning from scratch. Here we prefer the definitions given in (2.15),(2.16) as they facilitate comparison. As in [1, 19, 20], we will show that intermediate task similarity leads to higher forgetting, also known as *catastrophic forgetting*. On the other hand, transfer will prove to be monotonically increasing with task similarity.

2.2.2 Perturbing the pre-trained model

As an intermediate step towards Low Rank Adaptation, we are interested in the effect of perturbing the source model and use this perturbed matrix as initial condition for the training on task B. Our analysis showed what follows: when the task similarity between the source and the target is zero, using the source, or a perturbation of it, as initial condition, does not affect convergence, as the pre-trained model is arguably useless for the second task, whose rule (matrix of weights of the teacher) is orthogonal. As task similarity increases, perturbing the source slows the convergence, but does not introduce any plateau in the generalization curve. Let us describe better how the perturbation is performed.

Let $\mathbf{J}_{\hat{\tau}}$ be the source model, i.e. the weight matrix obtained at the end of learning task A. Let $\sigma > 0$ and P be a $K \times N$ matrix with entries that are i.i.d. normal Gaussian random variables. Then we define our (random) initial condition for task B as:

$$\mathbf{J_0} := \mathbf{J}_{\hat{\tau}} + \sigma P.$$

We now analyze the effect of this perturbation on the order parameters. The overlap Q at task switch, after perturbation, is given by:

$$Q = \frac{\mathbf{J_0}(\mathbf{J_0})^T}{N} = \underbrace{\frac{\mathbf{J}_{\hat{\tau}}(\mathbf{J}_{\hat{\tau}})^T}{N}}_{Q_{\hat{\tau}}} + 2\sigma\frac{\mathbf{J}_{\hat{\tau}}(\mathbf{P})^T}{N} + \sigma^2\frac{\mathbf{P}(\mathbf{P})^T}{N}$$

where $Q_{\hat{\tau}}$ is the self-overlap of the source model $\mathbf{J}_{\hat{\tau}}$. By the law of large numbers, in the thermodynamic limit, the scalar product has no effect on Q, as it goes to zero. On the other hand, being that $\frac{\mathbf{P}(\mathbf{P})^T}{N} \to I_K$, where I_K is the identity matrix of order K, the perturbation has the clear effect of increasing the diagonal of $Q_{\hat{\tau}}$ of a summand σ^2. In order to mitigate the effect of this perturbation on the order parameters, we can re-normalize Q to have the same norm of $Q_{\hat{\tau}}$. This idea has a clear geometric interpretation. At task switch, the self-overlap matrix of the source model is $Q_{\hat{\tau}}$, and the diagonal entries of this matrix are the norm of the rows of $\mathbf{J}_{\hat{\tau}}$, meaning that each of them are confined on a sphere of radius $(Q_{\hat{\tau}})_{ii}$. After the perturbation, we translate each vector and change

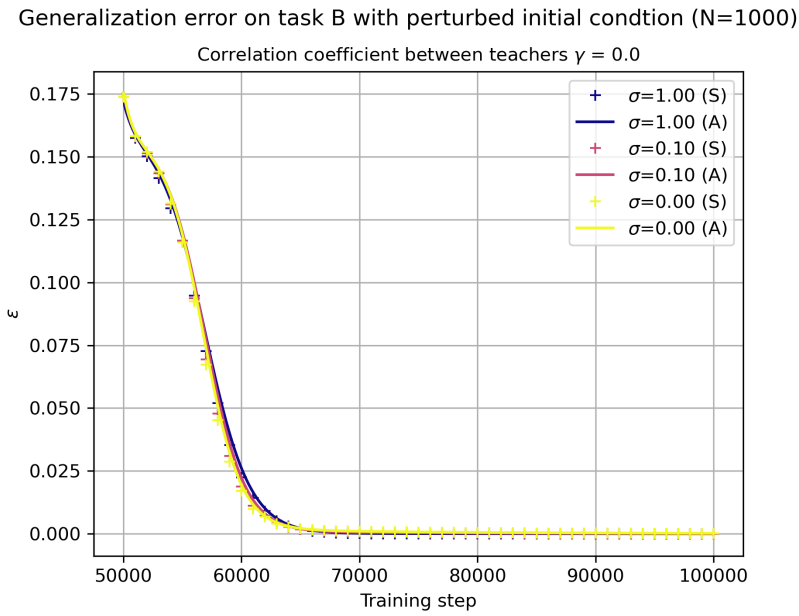


Figure 2.2: Generalization curve on second task for different values of the perturbation of the source. Orthogonal tasks $V = 0$. ODEs (A) and Simulations (S) are shown.

its direction and norm. Normalization is performed so that the new rows of $\mathbf{J}$ may change direction but belong to the same sphere as the rows of $\mathbf{J}_\tau$. The setting we are considering now is $K = 2$ and $M = 1$, so the task similarity is a scalar quantity (γ in the figures) that represents the cosine (up to renormalization) between the teachers. Consider Figure 2.2, where the two teachers defining the rules are orthogonal, i.e. where task similarity is $V = 0$. We observe that perturbing the pre-trained weights and using this perturbation as an initial condition for the student network on the second task does not affect the convergence to generalization. As one should expect, when the source is trained on a task that is completely uncorrelated to the new task, previous information is pretty much useless. On the other hand, increasing similarity between tasks makes the pre-trained model more and more suitable for the new task, enforcing the network to converge faster when the pre-trained model is not corrupted by noise. This is precisely what happens in Figures 2.3 and 2.4: when task similarity increases, the unperturbed initialization is the one that makes convergence faster. All the figures mentioned above contain also the analytical curves that come from the differential equations, proving that there is a perfect match between simulations and theoretical results.

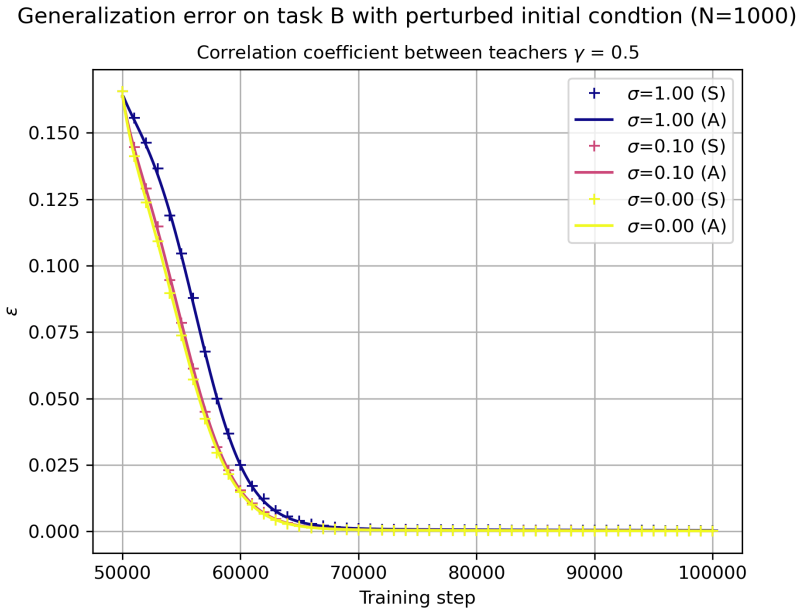


Figure 2.3: Generalization curve on second task for different values of the perturbation of the source. Intermediate task similarity, $V = 0.5$. ODEs (A) and Simulations (S) are shown.

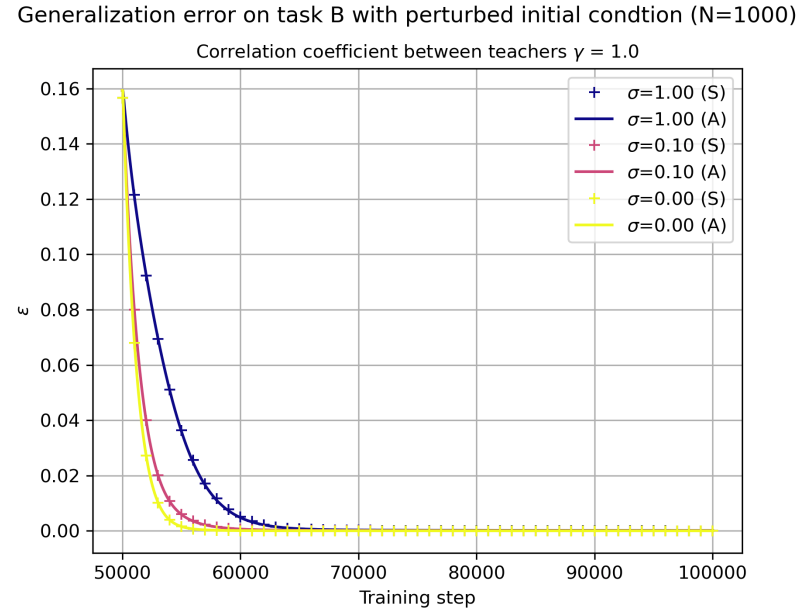


Figure 2.4: Generalization curve on second task for different values of the perturbation of the source. Task similarity $V = 1$. ODEs (A) and Simulations (S) are shown.

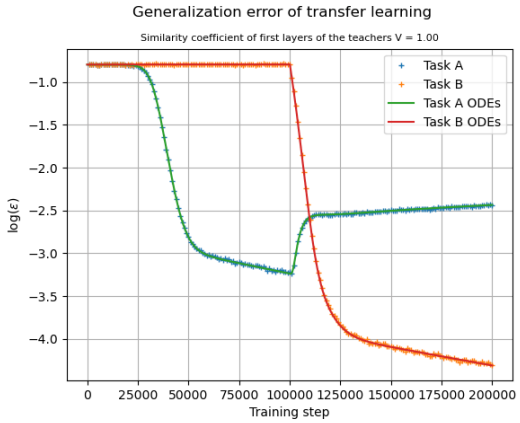
2.3 Numerical simulations

In this section, we want to reproduce some results in [1] to understand how task similarity affects forgetting. We will show, as the literature suggests, that intermediate task similarity is the one that suffers from catastrophic forgetting.

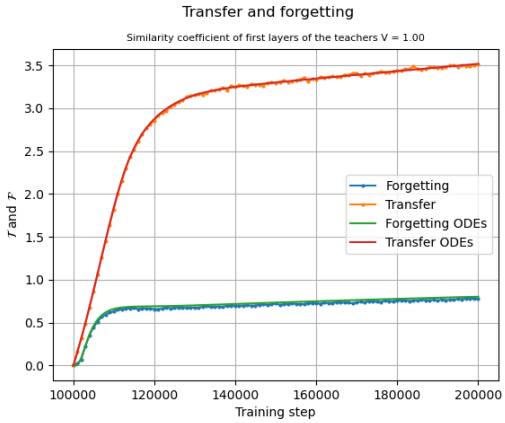
2.3.1 Theoretical results and simulations.

Before doing so, we show that the dynamical equations cited in section 2.1.2, together with the generalization errors, perfectly match with simulations. Consider the toy model $K = 2$ and $M = 1$. In this setting, generalization of both tasks is expected, as the student has sufficient resources to learn from both teachers. Moreover, for $M = 1$ the similarity 2.6 between teachers is a scalar number. In addition, let $\mathfrak{T}^{\diamondsuit} = (\mathbf{B}^{\diamondsuit}, \mathbf{v}^{\diamondsuit}, g)$ and $\mathfrak{T}^{\heartsuit} = (\mathbf{B}^{\heartsuit}, \mathbf{v}^{\heartsuit}, g)$ be the teachers and assume that $\|\mathbf{B}^{\diamondsuit}\| = \|\mathbf{B}^{\heartsuit}\| = 1$. Then, the similarity is given by the cosine of the angle between the teachers, and therefore ranges in $[-1, 1]$. For our analysis, following [1], we consider the setting $\mathbf{v}^{\diamondsuit} = 1$ and $\mathbf{v}^{\heartsuit} = -1$, thereby disentangling the similarity at the level of the second layer. Figures 2.5-2.7 show the comparison between analytical solutions and simulations, and we find the same results of Figure 1 and 2 in [1]. Theoretical and simulated curves perfectly match (within the expected deviation). Parameters of the simulation:

- Input dimension $N = 2000$;
- $K = 2$, $M = 1$;
- Learning rate: $\eta_J = \eta_h = 0.9$;
- Task similarity: $V = 1$;
- Optimizer: SGD optimizer with batch size = 1 (equivalent to Gradient Descent);
- Loss: scaled MSE loss;
- Teacher weight initialization: element-wise uncorrelated normal gaussian;
- Teacher head initialization: $\mathbf{v}^{\diamondsuit} = 1$ and $\mathbf{v}^{\heartsuit} = -1$;
- Student weight and head initialization: element-wise uncorrelated gaussian with zero mean and variance $\sigma^2 = 0.001$;

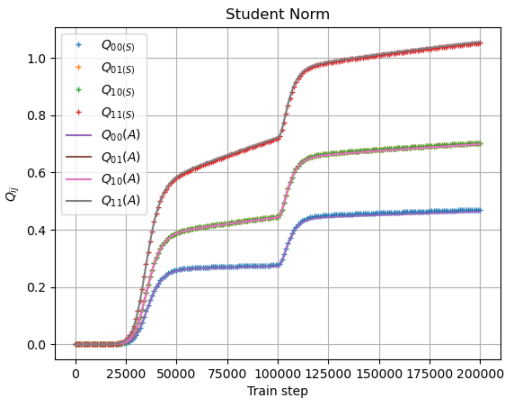


(a) Generalization loss: analytical solution and simulation.



(b) Transfer and Forgetting after task switch: analytical solution and simulation.

Figure 2.5: Generalization curves.



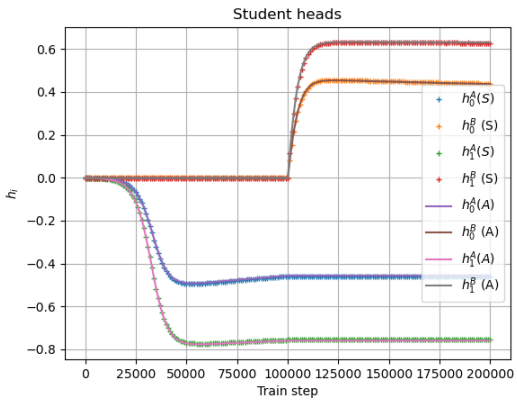


Figure 2.6: Evolution of the student's parameters during training. ODEs (A) and Simulations (S) are shown.

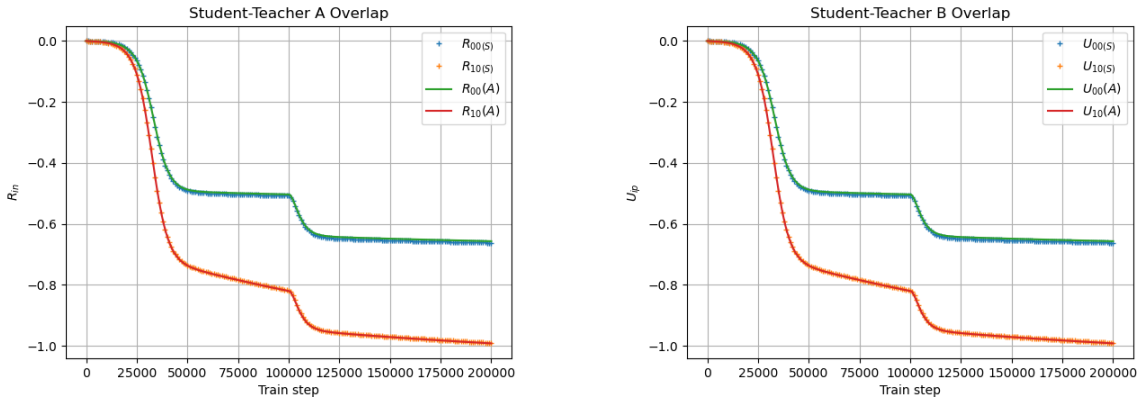


Figure 2.7: Evolution of the overlap between student and teacher A and B, respectively. ODEs (A) and Simulations (S) are shown.

2.3.2 Impact of task similarity

We are ready to discuss how task similarity affects catastrophic forgetting (inspired by [1]). Figures 2.8,2.9 show how the generalization loss for both tasks changes with task similarity V. They reproduce Figure 3 in [1] perfectly. Recall that, since we are working with an odd activation function, varying V in $[-1, 0)$ is not necessary, as a negative sign can always be absorbed by the hidden-to-output weights [1]. Again, our analysis is carried out with the following parameters:

- Input dimension $N = 1000$;
- $K = 2$, $M = 1$;
- Learning rate: $\eta_J = \eta_h = 0.9$;
- Time horizon $s = 100$, task switch at $s = 50$
- Optimizer: SGD optimizer with batch size = 1 (equivalent to Gradient Descent);
- Loss: scaled MSE loss;
- Teachers' input-to-hidden vectors initialization: see C;
- Teacher head initialization: $\mathbf{v}^{\diamondsuit} = 1$ and $\mathbf{v}^{\heartsuit} = -1$;
- Student weight and head initialization: element-wise uncorrelated gaussian with zero mean and variance $\sigma^2 = 0.001$;

Figure 2.8 shows that even with strongly correlated teacher vectors, forgetting is present. In the same figure, we quantify forgetting at $s = 75000$ (i.e. in the middle of the training procedure of the second task) for different values of similarity. Surprisingly, forgetting is higher when the task similarity is intermediate, while it is lower when the teachers

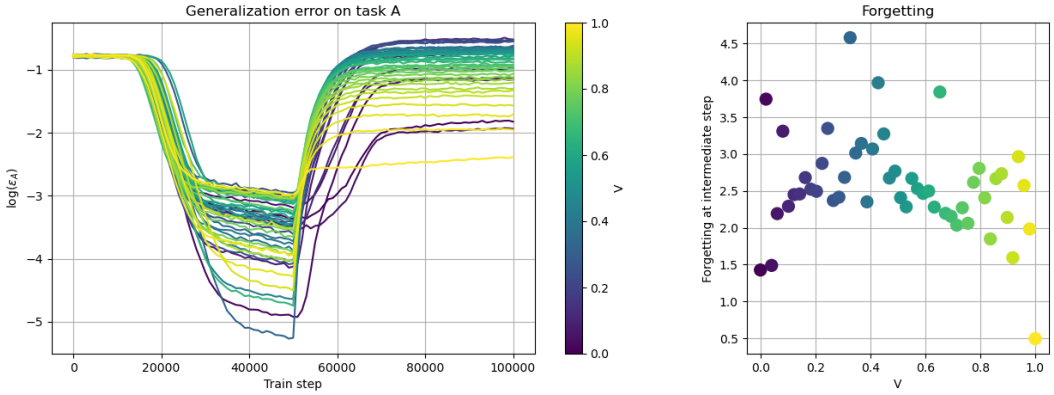


Figure 2.8: Generalization loss on task A for different values of the similarity coefficient. Forgetting at intermediate step of the training procedure.

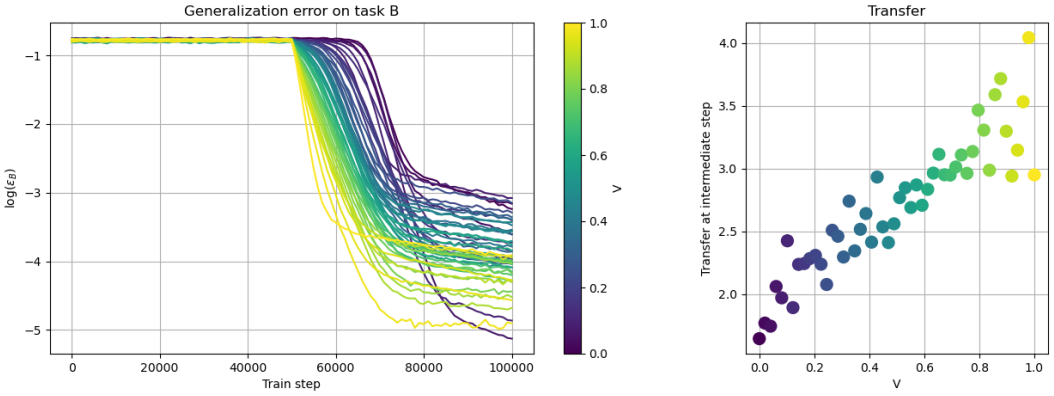


Figure 2.9: Generalization loss on task B for different values of the similarity coefficient. Transfer at intermediate step of the training procedure.

are not correlated, and takes the lowest values when the teachers are strongly aligned. On the other hand, Figure 2.9 shows that transfer is monotonically increasing with task similarity.

Chapter 3

Low-Rank Adaptation for Continual Learning

This thesis introduces a novel approach by deriving differential equations for the order parameters of the low-rank matrices introduced by Low-Rank Adaptation (LoRA), and investigating its effect on catastrophic forgetting.

3.1 Fine-tuning

Fine-tuning is a widely used paradigm in transfer learning in which the pre-trained model (*source*, denoted by $J^{(s)}$) is kept frozen while learning the second task (*target*) by adding a tuning matrix ΔJ to $J^{(s)}$ and learning only ΔJ.

3.2 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) is a state of the art parameter-efficient fine-tuning technique, first introduced in [21], where the tuning matrix ΔJ is the product of two. We will use LoRA for continual learning in the teacher-student setting in order to study the main characteristics of this technique in this elementary, yet effective, setting. The teacher-student setting in online learning will also be useful in deriving differential equations for the order parameters, as was done in the literature for other learning paradigms ([1, 4]), which also inspired previous chapters. Let $J^{(s)}$ be the input-to-hidden weight matrix obtained at the end of the training procedure on the first task. If the student is a multi-headed committee machine, we have $J^{(s)} \in \mathbb{R}^{K\times N}$. Then the matrix of weights

on the second task is defined by:

$$J^{(t)} := J^{(s)} + \frac{\beta}{\sqrt{r}}\mathbf{DA}$$

with $\beta \in \mathbb{R}$, $r \in \mathbb{N}$, $\mathbf{D} \in \mathbb{R}^{K\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times N}$. r is the low rank, while β is a scaling factor. A key idea is to understand whether the parameter β, which acts as a learning rate, is learnable or can become a control variable. In the following, we will refer to $\mathbf{D}$ as the *intensive* matrix or *intensive* weight (whose dimensions do not grow with N), while we will refer to $\mathbf{A}$ as the *extensive* matrix or *extensive* weight. Using the same notation that we have used so far for the multi-headed approach to continual learning, we define the student network $\mathfrak{S} = (\mathbf{J}, \mathbf{D}, \mathbf{A}, \mathbf{h}, g)$ with K hidden units, where $\mathbf{h} = \begin{bmatrix}\mathbf{h}^{\diamondsuit} & \mathbf{h}^{\heartsuit}\end{bmatrix}$. When the first task is being learned, it is sufficient to keep $\mathbf{D}$ and $\mathbf{A}$ fixed. At task switch, $\mathbf{J}$ is frozen and $\mathbf{D}$ and $\mathbf{A}$ are updated using gradient descent.

3.2.1 Gradient update of D

Let ξ^μ be the input vector. Let $\gamma := \beta/\sqrt{r}$. The output of the student network for task B is:

$$\Omega(\mathfrak{C}^\mu; \xi^\mu) = \sum_{i=1}^{K} h_i^\mu g\left(\frac{J_i\xi^\mu + \gamma D_i^\mu A^\mu \xi^\mu}{\sqrt{N}}\right) \tag{3.1}$$

where we suppressed the superscript $\heartsuit$ (i.e. $\mathfrak{C} = \mathfrak{C}^{\heartsuit}$ and $h_i = h_i^{\heartsuit}$) to keep the equations clean. Here, J_i and D_i are the i-th rows of $\mathbf{J}$ and $\mathbf{D}$, respectively. We also define $\tilde{J}_i = J_i + \gamma D_i \mathbf{A}$. The loss function is then

$$e(\Omega(\mathfrak{T}\,;\xi^\mu), \Omega(\mathfrak{C}^\mu\,;\xi^\mu)) = \frac{1}{2}(\sum_{m=1}^{M} v_m\, g\left(\frac{B_m\xi^\mu}{\sqrt{N}}\right) - \sum_{i=1}^{K} h_i^\mu\, g\left(\frac{J_i\xi^\mu + \gamma D_i^\mu A^\mu\xi^\mu}{\sqrt{N}}\right))^2. \tag{3.2}$$

we define, as before, the preactivations as follows:

$$X_i := \frac{J_i\xi}{\sqrt{N}} \tag{3.3}$$

$$Y_m := \frac{B_m\xi}{\sqrt{N}} \tag{3.4}$$

$$W_s := \frac{A_s\xi}{\sqrt{N}} \tag{3.5}$$

$$\tilde{X}_i := X_i + D_i\frac{A\xi}{\sqrt{N}} \tag{3.6}$$

where A_s is the s-th row of A and by setting

$$\mathbf{W} = \begin{bmatrix} W_1 \\ \vdots \\ W_r \end{bmatrix}$$

we can rewrite 3.6 as

$$\tilde{X}_i := X_i + \gamma D_i \mathbf{W}. \tag{3.7}$$

Using this notation, we can compute the partial derivative of 3.2 with respect to D_{js} as:

$$\begin{aligned} \partial_{D_{js}} e(\Omega(\mathfrak{T}\,;\xi^\mu), \Omega(\mathfrak{C}^\mu\,;\xi^\mu)) &= \Delta^\mu \partial_{D_{js}} \left(\sum_{i=1}^{K} h_i^\mu \, g\left(X_i^\mu + \gamma \sum_{t=1}^{r} D_{it}^\mu W_t^\mu \right) \right) \\ &= \gamma \Delta^\mu h_j^\mu g'\left(\tilde{X}_j^\mu\right) W_s^\mu. \end{aligned} \tag{3.8}$$

The gradient update for $\mathbf{D}$ is therefore

$$D_{js}^{\mu+1} = D_{js}^\mu - \frac{\eta_D}{N} \gamma \Delta^\mu h_j^\mu g'\left(\tilde{X}_j^\mu\right) W_s^\mu \tag{3.9}$$

and the scaling $1/N$ follows the same idea as the one for the head weights of the student, allowing for a well-defined thermodynamic limit.

3.2.2 Gradient update of A

In order to compute this update, we need to rewrite 3.2 as (we suppress the superscript μ for simplicity):

$$e = \frac{1}{2} \left(\sum_{m=1}^{M} v_m \, g\left(Y_m\right) - \sum_{i=1}^{K} h_i \, g\left(X_i + \frac{\gamma}{\sqrt{N}} \sum_{l=1}^{r} D_{il} \sum_{n=1}^{N} A_{ln} \xi_n \right) \right)^2. \tag{3.10}$$

Then the partial derivative with respect to A_{sp} results in

$$\begin{aligned} \partial_{A_{sp}} e &= \Delta \, \partial_{A_{sp}} \left(\sum_{i=1}^{K} h_i \, g\left(X_i + \frac{\gamma}{\sqrt{N}} \sum_{l=1}^{r} D_{il} \sum_{n=1}^{N} A_{ln} \xi_n \right) \right) \\ &= \Delta \sum_{i=1}^{K} h_i \, \partial_{A_{sp}} g\left(X_i + \frac{\gamma}{\sqrt{N}} \sum_{l=1}^{r} D_{il} \sum_{n=1}^{N} A_{ln} \xi_n \right) \\ &= \Delta \sum_{i=1}^{K} h_i \, g'\left(\tilde{X}_i\right) \frac{\gamma}{\sqrt{N}} D_{is} \xi_p. \end{aligned} \tag{3.11}$$

In matrix form, if A_s is the s-th row of $\mathbf{A}$:

$$A_s^{\mu+1} = A_s^\mu - \frac{\eta_A}{\sqrt{N}} \gamma \Delta^\mu \left(\sum_{i=1}^{K} h_i^\mu g'\left(\tilde{X}_i^\mu\right) D_{is}^\mu \right) (\xi^\mu)^T \tag{3.12}$$

3.3 The new order parameters

Recall that computing an average over the distribution of the input $\xi \sim \mathcal{N}(0, I_N)$ is not needed when the functions involved depend only on the preactivations. Let us

now focus on the joint distribution of such preactivations. Although $\tilde{X}_i$ is just a linear transformation of X_i and W_s, it is recommended to consider the higher dimensional gaussian distribution of the vector

$$Z = (X_1, \ldots, X_K, \tilde{X}_1, \ldots, \tilde{X}_K, Y_1, \ldots, Y_M, W_1, \ldots, W_r). \tag{3.13}$$

The idea behind this augmented gaussian distribution will become clear in a while. Let us now compute the covariance matrix of this $(2K+M+r)$-dimensional gaussian vector. As before, we find:

$$\langle X_i X_j \rangle = \frac{J_i J_i^T}{N} = Q_{ij} \tag{3.14}$$

$$\langle X_i Y_n \rangle = \frac{J_i B_n^T}{N} = R_{in} \tag{3.15}$$

$$\langle Y_m Y_n \rangle = \frac{B_m B_n^T}{N} = T_{mn}. \tag{3.16}$$

Then we compute the remaining interactions:

$$\langle W_s W_t \rangle = \frac{A_s A_t^T}{N} = \Phi_{st} \tag{3.17}$$

$$\langle Y_m W_s \rangle = \frac{B_m A_s^T}{N} = \Gamma_{ms} \tag{3.18}$$

$$\langle X_i W_s \rangle = \frac{J_i A_s^T}{N} = \Xi_{is} \tag{3.19}$$

$$\langle \tilde{X}_i W_s \rangle = \frac{(J_i + \gamma\, D_i A) A_s^T}{N} = \frac{J_i A_s^T}{N} + \gamma\, D_i \frac{A A_s^T}{N} = \Xi_{is} + \gamma\, D_i\, \Phi_{s:}^T \tag{3.20}$$

$$\langle \tilde{X}_i Y_m \rangle = \frac{(J_i + \gamma\, D_i A) B_m^T}{N} = \frac{J_i B_m^T}{N} + \gamma\, D_i \frac{A B_m^T}{N} = R_{im} + \gamma D_i \Gamma_{m:}^T \tag{3.21}$$

$$\langle X_i \tilde{X}_j \rangle = \frac{J_i (J_j + \gamma D_j A)^T}{N} = \frac{J_i J_j^T}{N} + \gamma \frac{J_i A^T D_j^T}{N} = Q_{ij} + \gamma\, \Xi_{i:}\, D_j^T \tag{3.22}$$

$$\begin{aligned} \langle \tilde{X}_i \tilde{X}_j \rangle &= \frac{\tilde{J}_i \tilde{J}_i}{N} = \frac{1}{N} \left(J_i + \gamma\, D_i A\right) \left(J_j^T + \gamma\, A^T D_j^T\right) \\ &= \frac{J_i J_j^T}{N} + \gamma\, \frac{J_i A^T}{N} D_j^T + \gamma\, D_i\, \frac{A J_j^T}{N} + \gamma^2\, D_i\, \frac{A A^T}{N} D_j^T \\ &= Q_{ij} + \gamma\, G_{i:} D_j^T + \gamma\, D_i G_{j:}^T + \gamma^2\, D_i \Phi D_j^T \end{aligned} \tag{3.23}$$

where the subscript "s :" denotes the s-th row of the corresponding matrix. In summary, we have the following order parameters:

$$
\begin{aligned}
&D \in \mathbb{R}^{K\times r},\\
&Q = \frac{\mathbf{J}\mathbf{J}^T}{N} \in \mathbb{R}^{K\times K},\\
&R = \frac{\mathbf{J}\mathbf{B}^T}{N} \in \mathbb{R}^{K\times M},\\
&T = \frac{\mathbf{B}\mathbf{B}^T}{N} \in \mathbb{R}^{M\times M},\\
&\Phi = \frac{\mathbf{A}\mathbf{A}^T}{N} \in \mathbb{R}^{r\times r},\\
&\Gamma = \frac{\mathbf{B}\mathbf{A}^T}{N} \in \mathbb{R}^{M\times r},\\
&\Xi = \frac{\mathbf{J}\mathbf{A}^T}{N} \in \mathbb{R}^{K\times r}
\end{aligned}
$$

and the vector 3.13 is gaussian with zero mean and covariance matrix:

$$
\tilde{C} = \begin{bmatrix}
Q & Q+\gamma\,\Xi\,D^T & R & \Xi\\
Q^T+\gamma\,D\Xi^T & Q+\gamma(\Xi D^T + D\Xi^T)+\gamma^2 D\Phi D^T & R+\gamma\,D\Gamma^T & \Xi+\gamma\,D\Phi^T\\
R^T & R^T+\gamma\,\Gamma D^T & T & \Gamma\\
G^T & G^T+\gamma\,\Phi D^T & \Gamma^T & \Phi
\end{bmatrix}.
$$

We can now write the generalization error in terms of the new preactivations, since:

$$\begin{aligned}
\langle e\left(\Omega(\mathfrak{T}\,;\xi),\Omega(\mathfrak{C}^{\heartsuit}\,;\xi)\right)\rangle_{\xi} &= \frac{1}{2}\langle\left(\sum_{m=1}^{M} v_m\, g\left(\frac{B_m\xi}{\sqrt{N}}\right)-\sum_{i=1}^{K} h_i\, g\left(\frac{J_i\xi+\gamma\, D_iA\xi}{\sqrt{N}}\right)\right)^2\rangle_{\xi}\\
&= \frac{1}{2}\langle\left(\sum_{m=1}^{M} v_m\, g\left(Y_m\right)-\sum_{i=1}^{K} h_i\, g\left(\tilde{X}_i\right)\right)^2\rangle_{\xi}\\
&= \frac{1}{2}\sum_{m,n}^{M} v_m v_n\langle g(Y_m)g(Y_n)\rangle_Z\\
&- \sum_{m}^{M}\sum_{i=1}^{K} v_m h_i\langle g(Y_m)g(\tilde{X}_i)\rangle_Z\\
&+ \frac{1}{2}\sum_{i,j}^{K} h_i h_j\langle g(\tilde{X}_i)g(\tilde{X}_j\rangle_Z\\
&= \frac{1}{2}\sum_{m,n}^{M} v_m v_n I_2(2K+m,2K+n)\\
&- \sum_{m}^{M}\sum_{i=1}^{K} v_m h_i I_2(2K+m,K+i)\\
&+ \frac{1}{2}\sum_{i,j}^{K} h_i h_j I_2(K+i,K+j)
\end{aligned} \tag{3.24}$$

which results in

$$\mathcal{E}(\mathfrak{S}^{\heartsuit}) = \frac{1}{2}\sum_{m,n}^{M} v_m v_n I_2(2K+m,2K+n) \tag{3.25}$$

$$- \sum_{m}^{M}\sum_{i=1}^{K} v_m h_i I_2(2K+m,K+i) \tag{3.26}$$

$$+ \frac{1}{2}\sum_{i,j}^{K} h_i h_j I_2(K+i,K+j). \tag{3.27}$$

The same computation can be carried out for the loss on the first task, but more order parameters are needed, which goes beyond the scope of this work. At the end of this long but straightforward computation, we remark on the reason behind the augmented gaussian distribution. If we had not introduced the new variable $\tilde{X}_i$, we would have had to compute the expected value of functions involving $g(X_i+\gamma\, D_iW)$. This could have introduced a completely new setting and would have not allowed us to use the quantities (e.g. I_2, I_3, I_4) in [7]. Introducing $\tilde{X}_i$ helps us overcome this issue and allows us to use all the quantities mentioned so far.

3.4 Differential equations for the order parameters

This section derives new differential equations for the order parameters of the network in the LoRA paradigm, using the tools of [7]. In the following, when writing v, h or B_m, we refer to the quantities $v^\heartsuit, h^\heartsuit$ or $B_m^\heartsuit$ that define the second task.

3.4.1 Differential equation for D

From 3.9 we obtain:

$$\frac{D_{js}^{\mu+1} - D_{js}^{\mu}}{1/N} = -\eta_D \, \gamma \, \Delta^\mu h_j^\mu g'\left(\tilde{X}_j^\mu\right) W_s^\mu$$

which in the thermodynamic limit results in the differential equation

$$\frac{\mathrm{d}D_{js}}{\mathrm{d}\tau} = -\eta_D \, \gamma \, h_j \langle g'\left(\tilde{X}_j\right) \, \Delta \, W_s \rangle$$

which we can expand to

$$\begin{aligned}\frac{\mathrm{d}D_{js}}{\mathrm{d}\tau} &= \eta_D \, \gamma \, h_j \left[\sum_m^M v_m \langle g'\left(\tilde{X}_j\right) \, W_s \, g\left(Y_m\right)\rangle - \sum_i^K h_i \langle g'\left(\tilde{X}_j\right) \, W_s \, g\left(\tilde{X}_i\right)\rangle\right] \\ &= \eta_D \, \gamma \, h_j \left[\sum_m^M v_m I_3(K+j, 2K+M+s, 2K+m)\right. \\ &\qquad \left. - \sum_i^K h_i I_3(K+j, 2K+M+s, K+m)\right].\end{aligned} \tag{3.28}$$

3.4.2 Differential equation for Φ

By 3.12 we have:

$$A_s^{\mu+1} = A_s^\mu - \frac{\eta_A}{\sqrt{N}} \gamma \Delta^\mu \left(\sum_{i=1}^K h_i^\mu g'\left(\tilde{X}_i^\mu\right) D_{is}^\mu\right) (\xi^\mu)^T.$$

Then:

$$\begin{aligned}A_s^{\mu+1}(A_t^{\mu+1})^T = A_s^\mu (A_t^\mu)^T &- \eta_A \, \gamma \, \Delta^\mu \left(\sum_{i=1}^K h_i^\mu g'\left(\tilde{X}_i^\mu\right) D_{is}^\mu\right) \left(\frac{A_t^\mu \xi}{\sqrt{N}}\right)^T \\ &- \eta_A \, \gamma \, \Delta^\mu \left(\sum_{i=1}^K h_i^\mu g'\left(\tilde{X}_i^\mu\right) D_{it}^\mu\right) \left(\frac{A_s^\mu \xi}{\sqrt{N}}\right) \\ &+ \eta_A^2 \, \gamma^2 \, (\Delta^\mu)^2 \left(\sum_{i=1}^K h_i^\mu g'\left(\tilde{X}_i^\mu\right) D_{is}^\mu\right) \left(\sum_{i=1}^K h_i^\mu g'\left(\tilde{X}_i^\mu\right) D_{it}^\mu\right) \frac{\|\xi\|^2}{N}\end{aligned}$$

which can be written as:

$$\begin{aligned}\frac{\Phi_{st}^{\mu+1}-\Phi_{st}^{\mu}}{1/N} =& -\eta_A\,\gamma\,\left(\sum_{i=1}^{K} h_i^{\mu} D_{is}^{\mu} g'\left(\tilde{X}_i^{\mu}\right)\,W_t^{\mu}\,\Delta^{\mu}\right)\\ &-\eta_A\,\gamma\,\left(\sum_{j=1}^{K} h_j^{\mu} D_{jt}^{\mu} g'\left(\tilde{X}_j^{\mu}\right)\,W_s^{\mu}\,\Delta^{\mu}\right)\\ &+\eta_A^2\,\gamma^2\,\left(\sum_{i,j}^{K} h_i^{\mu} h_j^{\mu} D_{is}^{\mu} D_{jt}^{\mu}\,g'\left(\tilde{X}_i^{\mu}\right) g'\left(\tilde{X}_j^{\mu}\right)(\Delta^{\mu})^2\right)\frac{\|\xi\|^2}{N}\end{aligned}$$

which in the thermodynamic limit becomes:

$$\begin{aligned}\frac{\mathrm{d}\Phi_{st}}{\mathrm{d}\tau} =& -\eta_A\,\gamma\,\left(\sum_{i=1}^{K} h_i D_{is}\langle g'\left(\tilde{X}_i\right)\,W_t\,\Delta\rangle\right)\\ &-\eta_A\,\gamma\,\left(\sum_{i=j}^{K} h_j D_{jt}\langle g'\left(\tilde{X}_j\right)\,W_s\,\Delta\rangle\right)\\ &+\eta_A^2\,\gamma^2\,\left(\sum_{i,j}^{K} h_i h_j D_{is} D_{jt}\langle g'\left(\tilde{X}_i\right) g'\left(\tilde{X}_j\right)(\Delta)^2\rangle\right)\end{aligned}$$

then expanding the definition of Δ and Δ^2 we have:

$$\begin{aligned}\frac{\mathrm{d}\Phi_{st}}{\mathrm{d}\tau} =& \eta_D\gamma\left[\sum_{i,m} h_i v_m D_{is}\langle g'(\tilde{X}_i)\,W_t\,g(Y_m)\rangle - \sum_{i,k} h_i h_k D_{is}\langle g'(\tilde{X}_i)\,W_t\,g(\tilde{X}_k)\rangle\right]\\ &+\eta_D\gamma\left[\sum_{i,m} h_i v_m D_{it}\langle g'(\tilde{X}_i)\,W_s\,g(Y_m)\rangle - \sum_{i,k} h_i h_k D_{it}\langle g'(\tilde{X}_i)\,W_s\,g(\tilde{X}_k)\rangle\right]\\ &+\eta_D^2\gamma^2\left[\sum_{i,j,k,k} h_i h_j h_k h_l D_{it} D_{js}\langle g'(\tilde{X}_i) g'(\tilde{X}_j) g(\tilde{X}_k) g(\tilde{X}_l)\rangle\right.\\ &-2\sum_{i,j,k,m} h_i h_j h_k v_m D_{it} D_{js}\langle g'(\tilde{X}_i) g'(\tilde{X}_j) g(\tilde{X}_k) g(Y_m)\rangle\\ &\left.+\sum_{i,j,m,n} h_i h_j v_m v_n D_{it} D_{js}\langle g'(\tilde{X}_i) g'(\tilde{X}_j) g(Y_m) g(Y_n)\rangle\right]\end{aligned}$$

which is written in terms of I_3 and I_4 as:

$$\begin{aligned}\frac{\mathrm{d}\Phi_{st}}{\mathrm{d}\tau} &= \eta_D\gamma\left[\sum_{i,m} h_i v_m D_{is} I_3(K+i, 2K+M+t, 2K+m)\right.\\ &\quad\left.-\sum_{i,k} h_i h_k D_{is} I_3(K+i, 2K+M+t, K+k)\right]\\ &\quad+\eta_D\gamma\left[\sum_{i,m} h_i v_m D_{it} I_3(K+i, 2K+M+s, 2K+m)\right.\\ &\quad\left.-\sum_{i,k} h_i h_k D_{it} I_3(K+i, 2K+M+s, K+k)\right]\\ &\quad+\eta_D^2\gamma^2\left[\sum_{i,j,k,l} h_i h_j h_k h_l D_{it} D_{js} I_4(K+i, K+j, K+k, K+l)\right.\\ &\quad-2\sum_{i,j,k,m} h_i h_j h_k v_m D_{it} D_{js} I_4(K+i, K+j, K+k, 2K+m)\\ &\quad\left.+\sum_{i,j,m,n} h_i h_j v_m v_n D_{it} D_{js} I_4(K+i, K+j, 2K+m, 2K+n)\right].\end{aligned} \tag{3.29}$$

3.4.3 Differential equation for Ξ

Again, from the gradient update 3.12, we have that:

$$J_j(A_s^{\mu+1})^T = J_j(A_s^{\mu})^T - \eta_A\gamma\Delta^{\mu}\left(\sum_i^K h_i^{\mu}\, g'(\tilde{X}_i^{\mu}) D_{is}^{\mu}\right)\underbrace{\frac{J_j\xi}{\sqrt{N}}}_{X_j}$$

and substituting the definition of Ξ we have:

$$\frac{\Xi_{js}^{\mu+1} - \Xi_{js}^{\mu}}{1/N} = -\eta_A\gamma\Delta^{\mu}\left(\sum_i^K h_i^{\mu}\, g'(\tilde{X}_i^{\mu}) D_{is}^{\mu}\, X_j\right)$$

which, in the thermodynamic limit, becomes:

$$\begin{aligned}\frac{\mathrm{d}\Xi_{js}}{\mathrm{d}\tau} &= -\eta_A\,\gamma\left[\sum_i^K h_i D_{is}\langle g'(\tilde{X}_i) X_j \Delta\rangle\right]\\ &= \eta_A\,\gamma\left[\sum_{i,m} h_i v_m D_{is}\langle g'(\tilde{X}_i)\, X_j\, g(Y_m)\rangle - \sum_{i,l} h_i h_l D_{is}\langle g'(\tilde{X}_i)\, X_j\, g(\tilde{X}_l)\rangle\right].\end{aligned}$$

Finally, we can rewrite this differential equation in terms of I_3 as follows:

$$\begin{aligned}\frac{\mathrm{d}\Xi_{js}}{\mathrm{d}\tau} &= \eta_A\,\gamma\left[\sum_{i,m} h_i v_m D_{is} I_3(K+i, j, 2K+m)\right.\\ &\quad\left.-\sum_{i,l} h_i h_l D_{is} I_3(K+i, j, K+l)\right].\end{aligned} \tag{3.30}$$

3.4.4 Differential equation for Γ

From the gradient update 3.12, we have that:

$$B_m(A_s^{\mu+1})^T = B_m(A_s^{\mu})^T - \eta_A \gamma \Delta^\mu \left(\sum_i^K h_i^\mu \, g'(\tilde{X}_i^\mu) D_{is}^\mu \right) \underbrace{\frac{B_m \xi}{\sqrt{N}}}_{Y_m}$$

and substituting the definition of Ξ we have:

$$\frac{\Gamma_{ms}^{\mu+1} - \Gamma_{ms}^{\mu}}{1/N} = -\eta_A \gamma \Delta^\mu \left(\sum_i^K h_i^\mu \, g'(\tilde{X}_i^\mu) D_{is}^\mu \, Y_m \right)$$

which, in the thermodynamic limit, becomes:

$$\begin{aligned} \frac{\mathrm{d}\Gamma_{ms}}{\mathrm{d}\tau} &= -\eta_A \, \gamma \, \left[\sum_i^K h_i D_{is} \langle g'(\tilde{X}_i) Y_m \Delta \rangle \right] \\ &= \eta_A \, \gamma \, \left[\sum_{i,n} h_i v_n D_{is} \langle g'(\tilde{X}_i) \, Y_m \, g(Y_n) \rangle - \sum_{i,l} h_i h_l D_{is} \langle g'(\tilde{X}_i) \, Y_m \, g(\tilde{X}_l) \rangle \right]. \end{aligned}$$

Finally, we can rewrite this differential equation in terms of I_3 as follows:

$$\begin{aligned} \frac{\mathrm{d}\Gamma_{ms}}{\mathrm{d}\tau} = \eta_A \, \gamma \, \Bigg[& \sum_{i,n} h_i v_n D_{is} I_3(K+i, 2K+m, 2K+n) \\ & - \sum_{i,l} h_i h_l D_{is} I_3(K+i, 2K+m, K+l) \Bigg]. \end{aligned} \tag{3.31}$$

3.4.5 Differential equation for h

From the loss on the second task computed in 3.2 we can compute the gradient update for $\mathbf{h}^\heartsuit$ as:

$$\partial_{h_j} e = \Delta^\mu g(\tilde{X}_j^\mu)$$

and the gradient update reads as:

$$h_j^{\mu+1} = h_j^\mu - \frac{\eta_h}{N} \Delta^\mu g(\tilde{X}_j^\mu). \tag{3.32}$$

Thus, rearranging the terms and taking the thermodynamic limit, we get:

$$\begin{aligned} \frac{\mathrm{d}h_j}{\mathrm{d}\tau} &= -\eta_h \langle \Delta g(\tilde{X}_j) \rangle \\ &= \eta_h \left[\sum_m v_m \, \langle g(Y_m) \, g(\tilde{X}_j) \rangle - \sum_i h_i \, \langle g(\tilde{X}_i) \, g(\tilde{X}_j) \rangle \right] \end{aligned}$$

which becomes:

$$\frac{\mathrm{d}h_j}{\mathrm{d}\tau} = \eta_h \left[\sum_m v_m \, I_2(2K+m, K+j) - \sum_i h_i \, I_2(K+i, K+j) \right] \tag{3.33}$$

Chapter 4

Results

In this chapter, we present the main results and conclusions on the effect of Low-Rank Adaptation on catastrophic forgetting, as well as a comparison between simulations and analytical solutions of the differential equations found in 3.4.

4.1 Effects of LoRA on catastrophic forgetting

In this section, we study the main features introduced by LoRA in the continual learning setting, through simulations. In order to do this, we will make comparisons with the work of [1] to exploit the main differences between the two approaches. Thus, every plot will present results given by

1. the setting in [1], which was also studied in 2, which we will refer to as LGS;
2. the study of continual learning using LoRA;

In order to make sure to exploit interesting characteristics, we will study an *over-realizable* scenario ($K > M$, [7]) with $K = 10$ and $M = 2$. Simulation details:

- Input dimension $N = 1000$;
- $K = 10$, $M = 2$;
- Low rank $r = 1$;
- $\beta = 1$;
- Learning rate: $\eta_J = \eta_h = \eta_D = \eta_A = 0.9$;
- Optimizer: SGD optimizer with batch size = 1 (equivalent to Gradient Descent);
- Loss: scaled MSE loss;
- Teachers' input-to-hidden vectors' initialization: prescribed similarity obtained using the procedure described in C;
- Teacher head initialization: $\mathbf{v}^{\diamondsuit} = 1$ and $\mathbf{v}^{\heartsuit} = -1$ (entry-wise);

- Student weight and head initialization: element-wise uncorrelated gaussian with zero mean and variance $\sigma^2 = 0.001$;

4.1.1 Task similarity $V = 0.1$

From Figure 4.1, we observe that whenever the two tasks are almost orthogonal, using Low-Rank Adaptation does not cause forgetting at all, but also no transfer is observed as the loss on task B does not decrease after task switch.

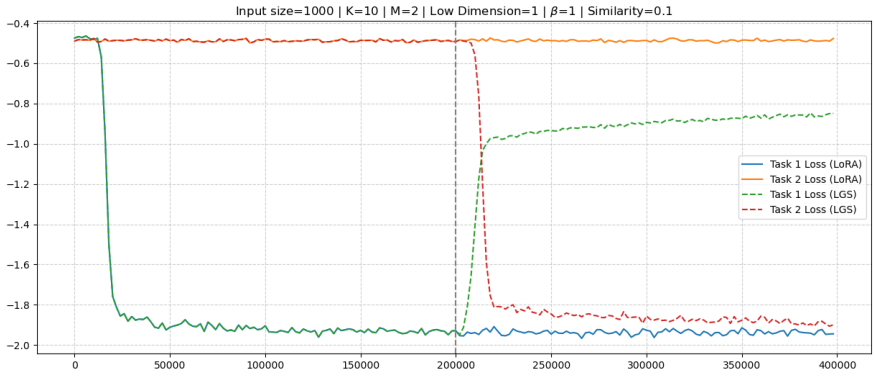


Figure 4.1: Generalization errors on both tasks using LoRA and LGS. Task similarity $V = 0.1$.

4.1.2 Task similarity $V = 0.5$

Figure 4.2 shows an interesting effect of LoRA. An initial plateau (between step 200000 and 250000 approximately) is observed, in which the generalization loss on task B decreases, but the loss on task A does not increase, i.e. no forgetting is present at that point. An intuition on the reason why this plateau appears will be given in the following section, where the differential equations and the order parameters are analyzed. Figure 4.2 also shows that after step 275000, loss A increases rapidly. However, the asymptotic value of the loss on task A given by LoRA is smaller than the one obtained without LoRA.

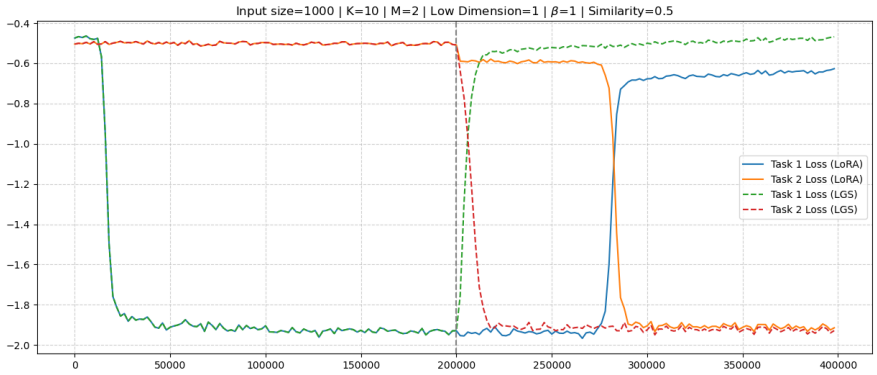


Figure 4.2: Generalization errors on both tasks using LoRA and LGS. Task similarity $V = 0.5$.

4.1.3 Task similarity $V = 0.9$

Figure 4.3 shows that highly correlated tasks preserve the plateau for loss B given by LoRA, which was also observed at intermediate similarity. Moreover, the plateau is obtained at a significantly lower value of the loss function. This is a key property that can be used to truly mitigate catastrophic forgetting, at least when the correlation between tasks is high.

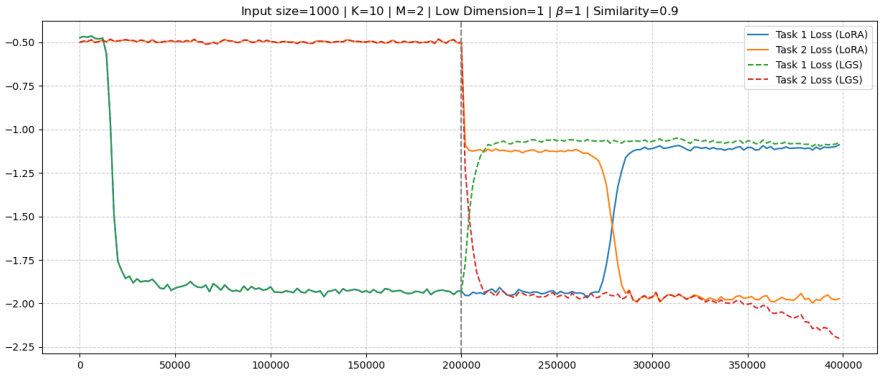
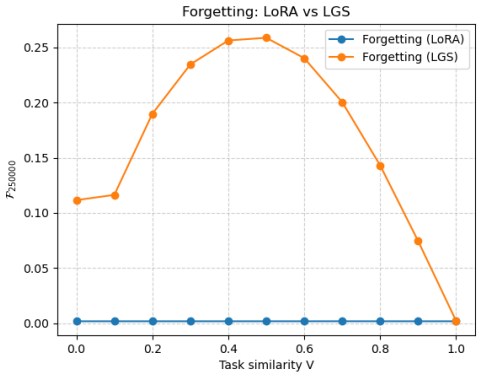


Figure 4.3: Generalization errors on both tasks using LoRA and LGS. Task similarity $V = 0.9$.

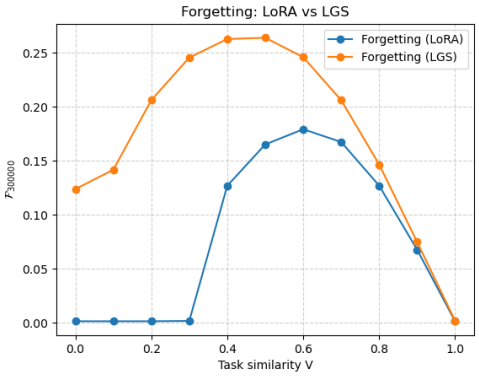
4.1.4 Forgetting under LoRA setting

Our simulations showed that, despite task similarity, Low-Rank Adaptation provides lower forgetting than the classical continual learning setting in [1], even at intermediate training steps. Indeed, all subfigures in 4.4 show that, together with an initial phase in

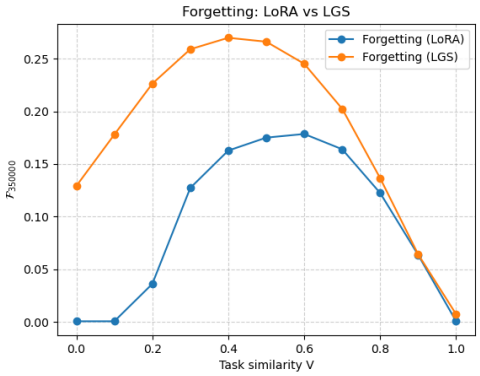
which LoRA does not show forgetting (4.4-(a) and (b)), forgetting is always lower than the one obtained with (LGS).



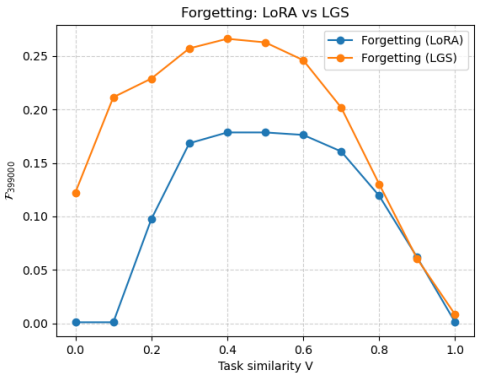
(a) Forgetting at 500 steps after task switch.



(b) Forgetting at 50000 training steps after task switch.



(c) Forgetting at 100000 steps after task switch.



(d) Forgetting at 199000 steps after task switch, nearly at the end of the training procedure.

Figure 4.4: Forgetting at different time steps, for different values of the similarity.

4.1.5 The effect of β

In this section, we will study the effect of β on the generalization error. As can also be seen from the dynamical equations found in 3, the rescaling factor β acts as a learning rate. Indeed, when computing the gradient with respect to $\mathbf{D}$ and $\mathbf{A}$ (see 3.8, 3.11), the factor $\gamma = \beta/\sqrt{r}$ appears as a factor in the gradient. Simulations proved that increasing the value of β speeds up convergence to a stationary value of generalization errors. However, small values of β slow down this behavior. Here we present, for different values of task similarity, the generalization plots obtained using LoRA, varying the value

of β. In every plot, the higher the β, the faster the plateau of task B identified in 4.1 appears, but the faster the forgetting. Figure 4.5 also suggests that the value of β can also show significantly diverse asymptotic values for generalization errors. Further work may investigate how to optimally choose β to reduce forgetting. As task similarity increases, this effect reduces.

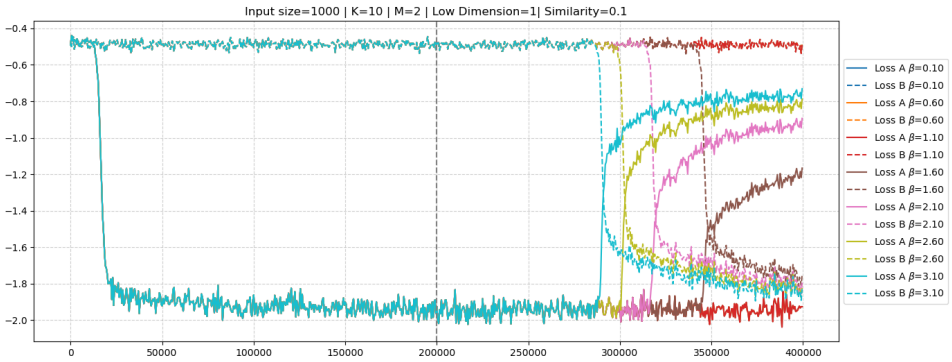


Figure 4.5: Generalization errors for different values of β. Task similarity $V = 0.1$.

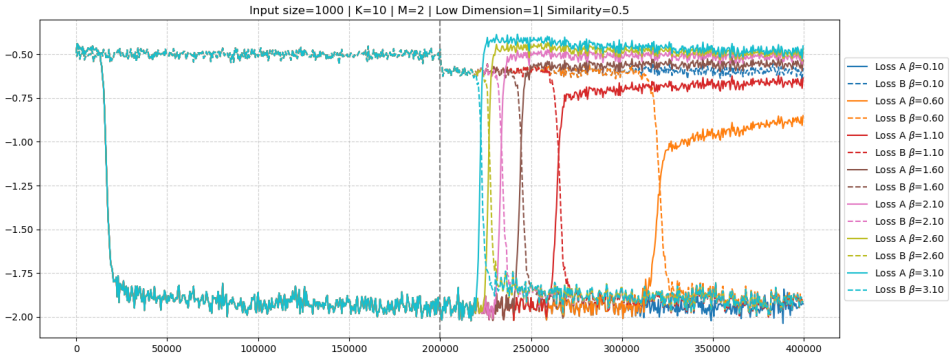


Figure 4.6: Generalization errors for different values of β. Task similarity $V = 0.5$.

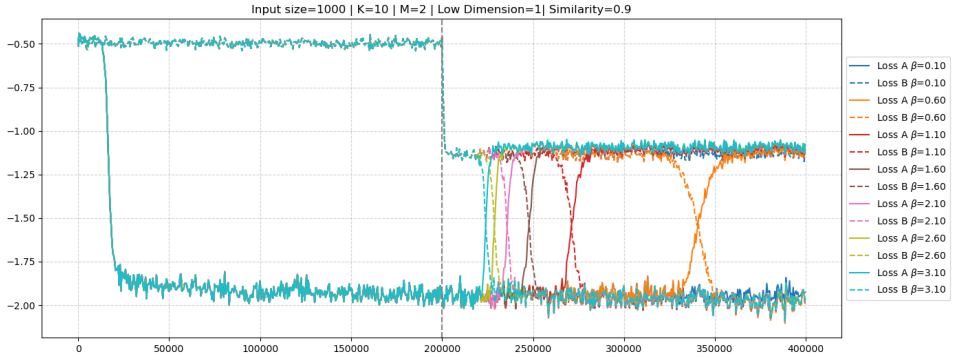


Figure 4.7: Generalization errors for different values of β. Task similarity $V = 0.9$.

4.2 ODEs and simulations

Since LoRA showed an interesting phenomenology from the point of view of catastrophic forgetting, a key idea is now to study the differential equations found in 3.4. Showing that there is a match between the differential equations and the simulations allows us to study the solution of the ODEs directly, which is faster than an actual training session even with a modest input size. We remark that the differential equations and the order parameters refer to the networks learning task B. One may also find the order parameters for the networks on task A, but this is beyond the goal of this thesis. As a final result, in this section we show that the differential equations found in 3.4 match the order parameters computed during the training procedure with the online approach. We will consider task similarities $V \in \{0.2, 0.5, 0.9\}$. The dynamical equations are studied starting at task switch. As we said, one may also compute the dynamics before switching task, but new dynamical equations for the order parameter Ξ and a new computation of the loss on task A should be studied, as Ξ also depends on the source. Here we make a comparison between the differential equations and the simulations at task switch, only, i.e. we train the network on task A and then we employ the dynamical equations using the source provided by the simulation, starting at task switch. Simulation details:

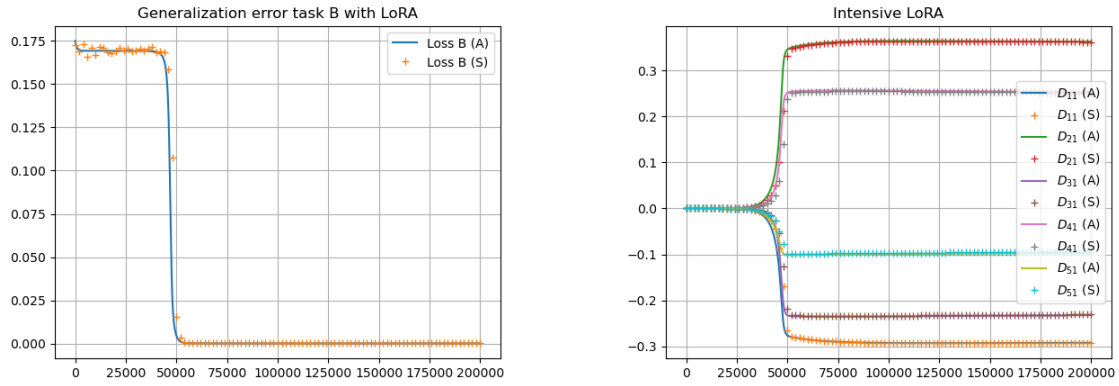
- Input dimension $N = 1000$;
- $K = 5$, $M = 1$;
- Low rank: $r = 1$;
- $\beta = 1$ (for $V \in \{0.5, 0.9\}$) and $\beta = 5$ for $V = 0.2$;
- Learning rate: $\eta_J = \eta_h = \eta_D = \eta_A = 0.9$;
- Optimizer: SGD optimizer with batch size = 1 (equivalent to Gradient Descent);
- Loss: scaled MSE loss;

- Teachers' input-to-hidden vectors initialization: prescribed similarity obtained using the procedure described in C;
- Teacher head initialization: $\mathbf{v}^{\diamondsuit} = 1$ and $\mathbf{v}^{\heartsuit} = -1$;
- Student weight and head initialization: element-wise uncorrelated gaussian with zero mean and variance $\sigma^2 = 0.001$;

Remark. *The reason behind the choice $\beta = 5$ when tasks are almost orthogonal ($V = 0.2$) is that keeping information on the source can decrease the performance on the second task. Thus, amplifying the effect of the LoRA update is needed.*

Solving the dynamical equations confirms the presence of an initial plateau in the generalization error for task B. Here, we can also see that this plateau is given by a trapping of the order parameters of LoRA to constant values, while the head weights change instantaneously starting from task switch. As long as the LoRA order parameters change, meaning that the network is learning the main features at level of the input-to-hidden layer, the generalization error starts to decrease.

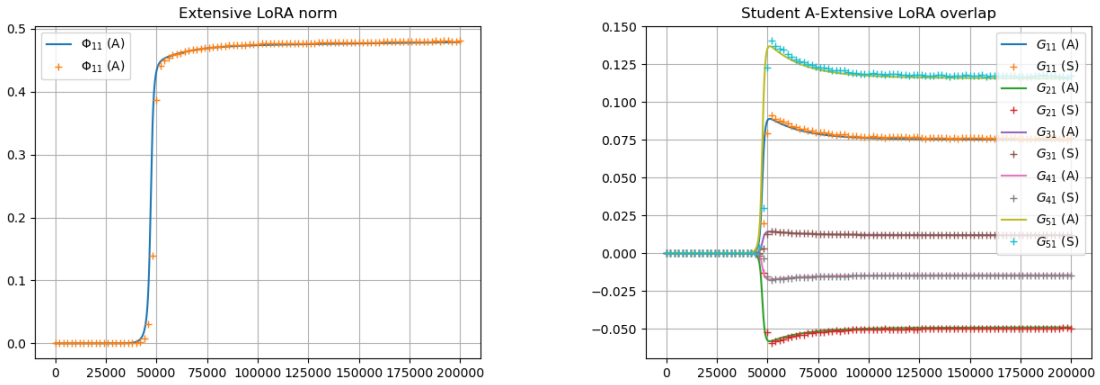
4.2.1 Similarity $V = 0.2$



(a) Generalization error on task B.

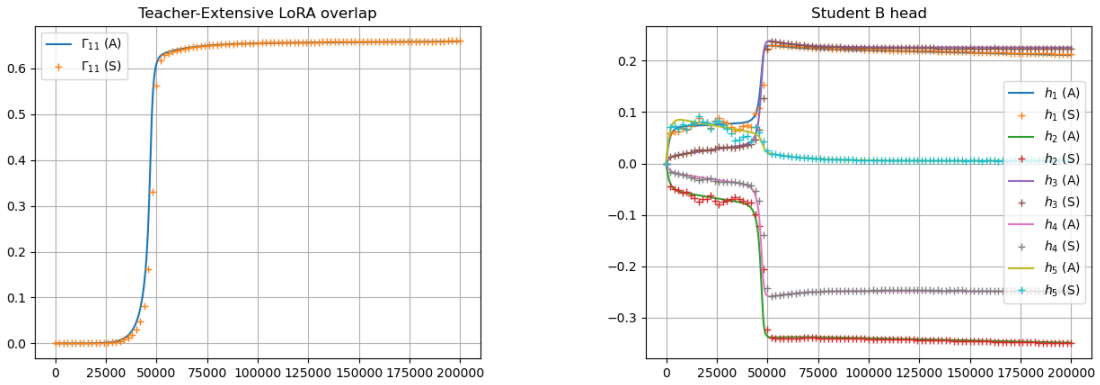
(b) Evolution of $\mathbf{D}$.

Figure 4.8: ODEs (A) and Simulations (S). Task Similarity $V = 0.2$.



(a) Evolution of Φ. (b) Evolution of Ξ.

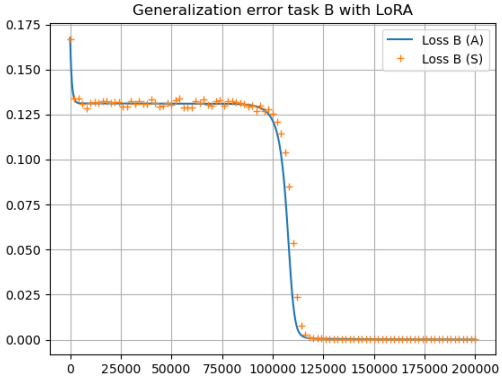
Figure 4.9: ODEs (A) and Simulations (S). Task Similarity $V = 0.2$.



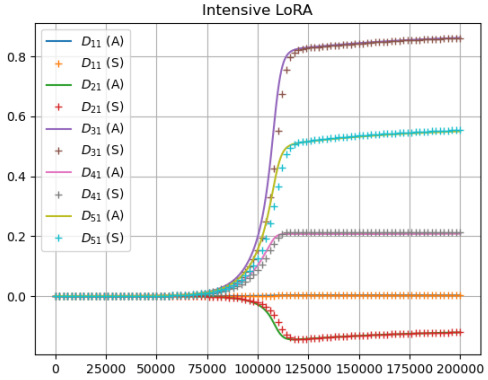
(a) Evolution of Γ. (b) Evolution of $\mathbf{h}^{\heartsuit}$.

Figure 4.10: ODEs (A) and Simulations (S). Task Similarity $V = 0.2$.

4.2.2 Similarity $V = 0.5$

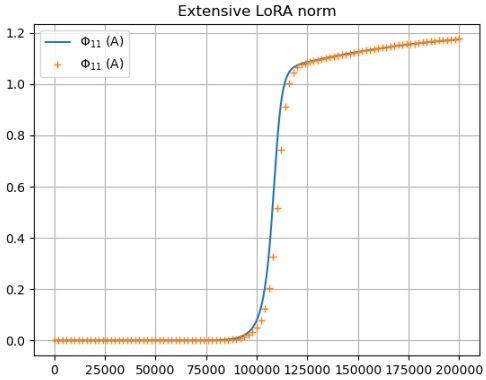


(a) Generalization error on task B.

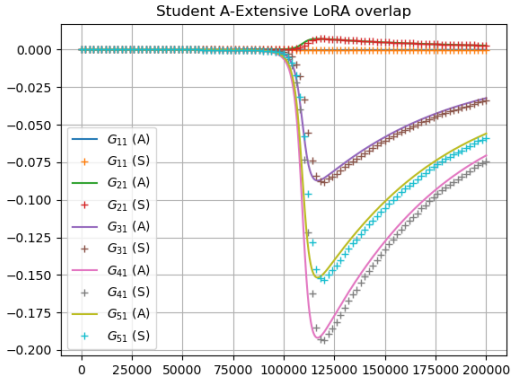


(b) Evolution of $\mathbf{D}$.

Figure 4.11: ODEs (A) and Simulations (S). Task Similarity $V = 0.5$.

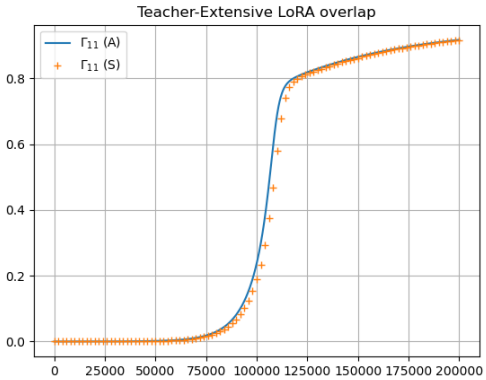


(a) Evolution of Φ.

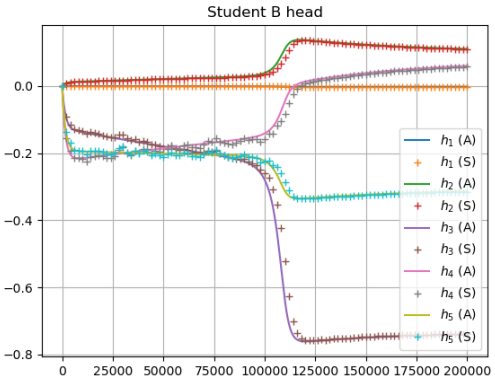


(b) Evolution of Ξ.

Figure 4.12: ODEs (A) and Simulations (S). Task Similarity $V = 0.5$.



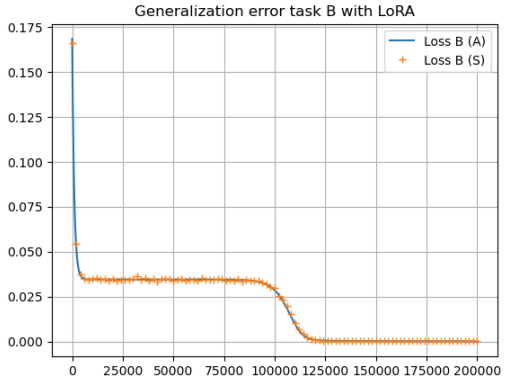
(a) Evolution of Γ.



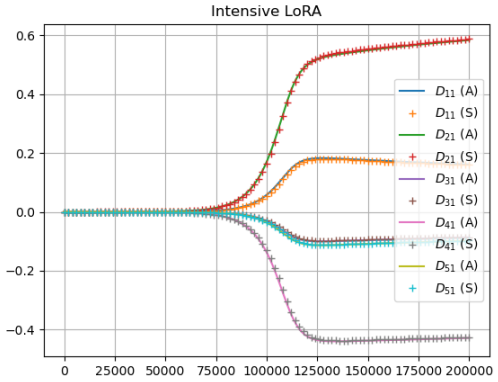
(b) Evolution of $\mathbf{h}^{\heartsuit}$

Figure 4.13: ODEs (A) and Simulations (S). Task Similarity $V = 0.5$.

4.2.3 Similarity $V = 0.9$

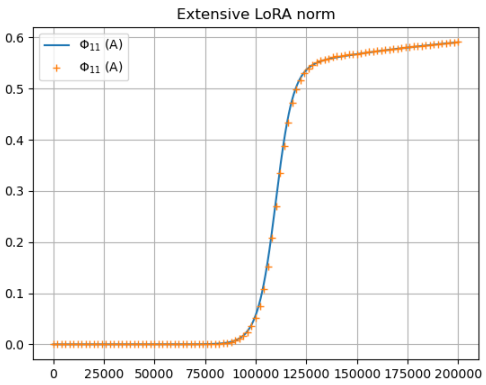


(a) Generalization error on task B.

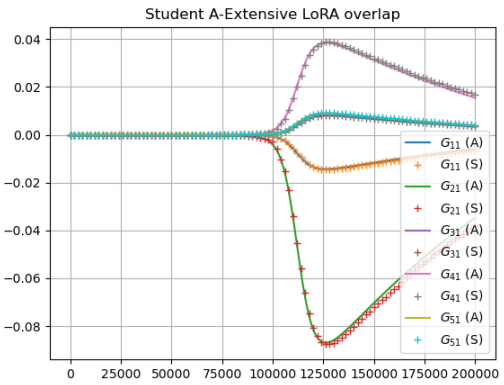


(b) Evolution of $\mathbf{D}$.

Figure 4.14: ODEs (A) and Simulations (S). Task Similarity $V = 0.9$.

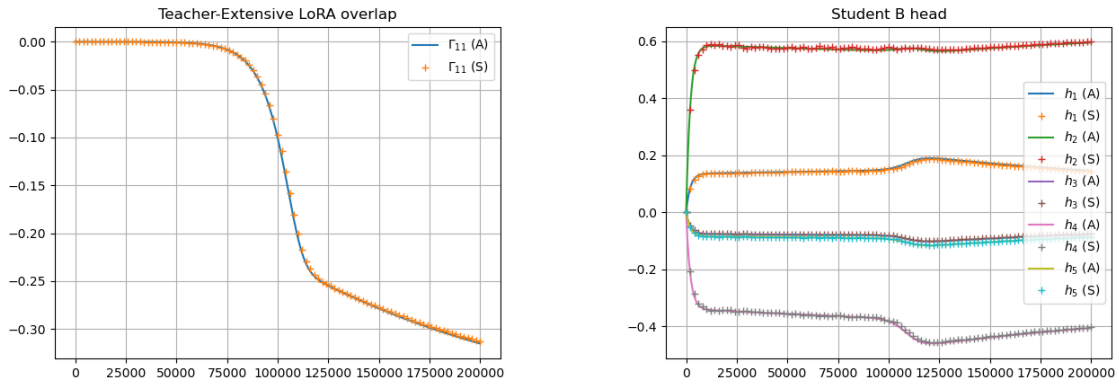


(a) Evolution of Φ.



(b) Evolution of Ξ.

Figure 4.15: ODEs (A) and Simulations (S). Task Similarity $V = 0.9$.



(a) Evolution of Γ.

(b) Evolution of $\mathbf{h}^{\heartsuit}$.

Figure 4.16: ODEs (A) and Simulations (S). Task Similarity $V = 0.9$.

Appendix A

Integral computations

A.1 Preliminary formulas

Let us give some preliminary results on integrals involving the error function. All formulas are written in two versions, one using the error function $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$, and the other using its rescaling $g(x) = \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) = \sqrt{\frac{2}{\pi}} \int_0^x e^{-\frac{t^2}{2}}\, dt$. Observe that $g'(x) = \sqrt{\frac{2}{\pi}} e^{-\frac{x^2}{2}}$.

Proposition A.1.1. *For any $a > 0$ and any $b \in \mathbb{R}$:*

$$\int_{\mathbb{R}} x e^{-ax^2} g(bx) dx = \frac{b}{a\sqrt{2a + b^2}} \tag{A.1}$$

$$\int_{\mathbb{R}} x e^{-ax^2} \operatorname{erf}(bx) dx = \frac{b}{a\sqrt{a + b^2}} \tag{A.2}$$

Proof. The proof of A.2 follows from A.1 with $b = \sqrt{2}\hat{b}$.
By integration by parts:

$$\int_{\mathbb{R}} x e^{-ax^2} g(bx) dx = -\frac{1}{2a} e^{-ax^2} g(bx) \Big|_{-\infty}^{+\infty} + \frac{b}{2a} \sqrt{\frac{2}{\pi}} \int_{\mathbb{R}} e^{-\frac{x^2}{2}(2a+b^2)}\, dx$$

Since $\left| e^{-ax^2} g(bx) \right| \leq \sqrt{\frac{2}{\pi}} \left| e^{-ax^2} bx \right| \to 0$ as $|x| \to +\infty$, and using the definition of a gaussian density, the above expression is equal to

$$\begin{aligned} &= \frac{b}{2a} \sqrt{\frac{2}{\pi}} \frac{\sqrt{2\pi}}{\sqrt{2a + b^2}} \\ &= \frac{b}{a\sqrt{2a + b^2}}. \end{aligned}$$

□

The following formula is needed to compute some quantities of interest.

Proposition A.1.2 ([22, Formula 13, Section 4.2])**.**

$$\int_{\mathbb{R}} e^{-(ax+b)^2} \operatorname{erf}(x)\, dx = -\frac{\sqrt{\pi}}{a} \operatorname{erf}\left(\frac{b}{\sqrt{a^2+1}}\right).$$

A sketch of the proof:

Proof. Define

$$I(b) = \int_{\mathbb{R}} e^{-(ax+b)^2} \operatorname{erf}(x)\, dx.$$

Rewrite $I(b)$ using a change of variable

$$I(b) = \int_{\mathbb{R}} e^{-(az)^2} \operatorname{erf}(z - \frac{b}{a})\, dz.$$

Now differentiate with respect to b:

$$\begin{aligned}
\frac{\mathrm{d}I(b)}{\mathrm{d}b} &= \frac{-2}{a\sqrt{\pi}} \int_{\mathbb{R}} e^{-a^2z^2} e^{-(z-\frac{b}{a})^2}\, dz \\
&= \frac{-2e^{-\frac{b^2}{a^2}}}{a\sqrt{\pi}} \int_{\mathbb{R}} e^{-((a^2+1)z^2 - \frac{2bz}{a})}\, dz \\
&= \frac{-2e^{-\frac{b^2}{a^2} + \frac{b^2}{a^2(a^2+1)}}}{a\sqrt{\pi}} \underbrace{\int_{\mathbb{R}} e^{-(\sqrt{a^2+1}z - \frac{b}{a\sqrt{a^2+1}})^2}\, dz}_{\sqrt{\frac{2\pi}{2(a^2+1)}}} \\
&= \frac{-2e^{-\frac{b^2}{a^2+1}}}{a\sqrt{a^2+1}}.
\end{aligned}$$

Let then $R > b$. By the fundamental theorem of calculus:

$$I(R) - I(b) = -2 \int_b^R \frac{e^{-\frac{\beta^2}{a^2+1}}}{a\sqrt{a^2+1}}\, d\beta = -\frac{2}{a} \int_{\frac{b}{\sqrt{a^2+1}}}^{\frac{R}{\sqrt{a^2+1}}} e^{-t^2}\, dt.$$

Let then $R \to +\infty$ and observe that $I(R) \xrightarrow{R\to+\infty} 0$ to obtain:

$$I(b) = \frac{2}{a} \int_{\frac{b}{\sqrt{a^2+1}}}^{+\infty} e^{-t^2}\, dt.$$

Use the same computation for $R < b$ and observe that $I(R) \xrightarrow{R\to-\infty} 0$ as well and get:

$$I(b) = -\frac{2}{a} \int_{-\infty}^{\frac{b}{\sqrt{a^2+1}}} e^{-t^2}\, dt.$$

Summing the two expressions and using the change of variable $z = -t$ in the integral with $+\infty$ as extreme, we get the claim. □

Corollary A.1.1.

$$\int_{\mathbb{R}} e^{-(ax+b)^2} g(cx)\,dx = -\frac{\sqrt{\pi}}{a}\operatorname{erf}\left(\frac{bc}{\sqrt{2a^2+c^2}}\right) = -\frac{\sqrt{\pi}}{a} g\left(\frac{\sqrt{2}bc}{\sqrt{2a^2+c^2}}\right). \tag{A.3}$$

$$\int_{\mathbb{R}} e^{-(ax+b)^2} \operatorname{erf}(cx)\,dx = -\frac{\sqrt{\pi}}{a}\operatorname{erf}\left(\frac{bc}{\sqrt{a^2+c^2}}\right). \tag{A.4}$$

The last property of interest of the error function is the following and is a more general result of [8, Equation (A3)]:

Proposition A.1.3. *For any $a, b \in \mathbb{R}$ and any $c > 0$ the followings hold true:*

$$\int_{\mathbb{R}} e^{-\frac{cx^2}{2}} g(ax)\, g(bx)\,dx = \frac{2\sqrt{2}}{\sqrt{\pi c}} \arcsin\left(\frac{ab}{\sqrt{a^2+c}\sqrt{b^2+c}}\right) \tag{A.5}$$

$$\int_{\mathbb{R}} e^{-\frac{cx^2}{2}} \operatorname{erf}(ax)\operatorname{erf}(bx)dx = \frac{2\sqrt{2}}{\sqrt{\pi c}} \arcsin\left(\frac{2ab}{\sqrt{2a^2+c}\sqrt{2b^2+c}}\right) \tag{A.6}$$

Proposition A.1.4. *Let (X, Y) be a 2-dimensional gaussian vector with zero mean and covariance matrix C. Let A be the inverse of C and let $\langle\cdot\rangle$ be the expected value with respect to $\mathcal{N}(0, C)$. Let g be the scaled error function $g(x) = \operatorname{erf}(x/\sqrt{2})$ Then*

$$\langle g(X)\cdot g(Y)\rangle = \frac{2}{\pi}\arcsin\left(\frac{C_{12}}{\sqrt{C_{11}+1}\sqrt{C_{22}+1}}\right). \tag{A.7}$$

Proof.

$$\begin{aligned}
\langle g(X)\cdot g(Y)\rangle &= \int_{\mathbb{R}^2} g(x)g(y)\frac{e^{-\frac{1}{2}\left(A_{11}x^2+2xyA_{12}+A_{22}y^2\right)}}{\sqrt{(2\pi)^2\det(C)}}\,dxdy \\
&= \frac{1}{2\pi\sqrt{\det(C)}}\int_{\mathbb{R}} g(x)\cdot e^{-\frac{1}{2}x^2\frac{\det(A)}{A_{22}}}\int_{\mathbb{R}} g(y)\cdot e^{-\left(\sqrt{\frac{A_{22}}{2}}y+\frac{A_{12}}{\sqrt{2A_{22}}}x\right)^2}\,dy\,dx \\
&\overset{(A.3)}{=} -\frac{1}{\sqrt{2\pi A_{22}\det(C)}}\int_{\mathbb{R}} e^{-\frac{x^2}{2}\frac{\det(A)}{A_{22}}} g(x)\cdot g\left(\frac{A_{12}}{\sqrt{A_{22}^2+A_{22}}}x\right)dx \\
&\overset{(A.5)}{=} -\frac{2}{\pi}\arcsin\left(\frac{A_{12}}{\sqrt{\det(A)+A_{11}}\sqrt{\det(A)+A_{22}}}\right) \\
&= \frac{2}{\pi}\arcsin\left(\frac{C_{12}}{\sqrt{C_{11}+1}\sqrt{C_{22}+1}}\right)
\end{aligned}$$

where the last equality follows from the fact that:

$$A_{12} = -C_{12}\det(A)$$

$$A_{11} = C_{22}\det(A)$$

$$A_{22} = C_{11}\det(A)$$

□

A.2 Computation of I_2

This is a straightforward application of proposition A.1.4. The conclusion of formula (1.13) follows from observing that:

$$\begin{aligned}
&C_{ij} = R_{in} \quad \text{for } i \in \{1,\dots,K\} \text{ and } j \in \{K+1,\dots,M\} \text{ where } n = j-K \\
&C_{ij} = T_{mn} \quad \text{for } i,j \in \{K+1,\dots,M\} \text{ where } m = i-K,\ n = j-K, \\
&C_{ij} = Q_{ik} \quad \text{for } i,k \in \{1,\dots,K\}.
\end{aligned}$$

A.3 Computation of I_3

Let us now compute the integral in (15), that is we compute:

$$I_3 = \langle g'(Z_i)\, Z_j\, g(Z_k) \rangle,$$

where $i \in \{1,\dots,K\}$ and $j,k \in \{1,\dots,K+M\}$. Let C be the projected matrix. Let us denote by A the inverse of C and recall that the cofactor matrix of a symmetric one can be written as:

$$A_{ij} = \frac{(-1)^{i+j}}{\det(C)} C_{ij},$$

where C_{ij} is the matrix obtained from C by deleting the i-th row and the j-th column.

Using this notation, we have that

$$\begin{aligned}
I_3 &= \int_{\mathbb{R}^3} g'(x) \cdot y \cdot g(z) \cdot \gamma(x,y,z)\, dx\, dy\, dz \\
&= \frac{1}{2\pi^2\sqrt{\det(C)}} \times \\
&\times \int_{\mathbb{R}^3} y \cdot g(z) \cdot e^{-\frac{1}{2}\left((A_{11}+1)x^2 + A_{22}y^2 + A_{33}z^2 + 2xyA_{12} + 2yzA_{23} + 2xzA_{13}\right)}\, dx\, dy\, dz \\
&= \frac{1}{\sqrt{2\pi^3(A_{11}+1)\det(C)}} \times \\
&\times \int_{\mathbb{R}^2} y \cdot g(z) \cdot e^{-\frac{1}{2}\left(\frac{A_{22}(A_{11}+1)-A_{12}^2}{A_{11}+1} y^2 + 2\frac{A_{23}(A_{11}+1)-A_{13}A_{12}}{A_{11}+1} yz + \frac{A_{33}(A_{11}+1)-A_{13}^2}{A_{11}+1} z^2\right)}\, dy\, dz
\end{aligned}$$

Using the following shortcuts for the quantities above:

$$D_{11} = \frac{A_{22}(A_{11}+1) - A_{12}^2}{A_{11}+1}$$

$$D_{12} = -\frac{A_{23}(A_{11}+1) - A_{13}A_{12}}{A_{11}+1}$$

$$D_{22} = \frac{A_{33}(A_{11}+1) - A_{13}^2}{A_{11}+1}$$

we compute

$$\begin{aligned}
&\int_{\mathbb{R}^2} y \cdot g(z) \cdot e^{-\frac{1}{2}\left(D_{11}y^2 - 2D_{12}yz + D_{22}z^2\right)}\, dy\, dz \\
=&\int_{\mathbb{R}} y \cdot e^{-\frac{1}{2}\left(\frac{D_{11}D_{22}-D_{12}^2}{D_{22}}\right)y^2} \int_{\mathbb{R}} g(z) \cdot e^{-\left(\sqrt{\frac{D_{22}}{2}}z - \frac{D_{12}}{\sqrt{2D_{22}}}y\right)^2} dz\, dy \\
=&\sqrt{\frac{2\pi}{D_{22}}} \int_{\mathbb{R}} y \cdot e^{-\frac{D_{11}D_{22}-D_{12}^2}{2D_{22}}y^2} \cdot g\left(\frac{D_{12}}{\sqrt{D_{22}^2 + D_{22}}}y\right) dy \\
\overset{(A.1)}{=}&\frac{2\sqrt{2\pi}D_{12}}{(D_{11}D_{22} - D_{12}^2)\sqrt{D_{11}(D_{22}+1) - D_{12}^2}}.
\end{aligned}$$

Putting everything together:

$$I_3 = \frac{2}{\pi} \frac{D_{12}}{(D_{11}D_{22} - D_{12}^2)\sqrt{(D_{11}(D_{22}+1) - D_{12}^2)\det(C)(A_{11}+1)}}$$

After some computations, we can write explicitly the quantities above as follows:

$$\begin{aligned}
D_{11}D_{22} - D_{12}^2 &= \frac{(A_{11}+1)(A_{22}A_{33} - A_{23}^2) + 2A_{13}A_{23}A_{12} - A_{22}A_{13}^2 - A_{33}A_{12}^2}{A_{11}+1} \\
&= \frac{A_{22}A_{33} - A_{23}^2 + A_{11}(A_{22}A_{33} - A_{23}^2) - A_{12}(A_{12}A_{33} - A_{13}A_{23})}{A_{11}+1} \\
&+ \frac{A_{13}(A_{23}A_{12} - A_{22}A_{13})}{A_{11}+1} \\
&= \frac{\det(A) + A_{22}A_{33} - A_{23}^2}{A_{11}+1}
\end{aligned}$$

$$\frac{D_{12}}{D_{11}D_{22} - D_{12}^2} = \frac{A_{13}A_{12} - A_{23}A_{11} - A_{23}}{\det(A) + A_{22}A_{33} - A_{23}^2}$$

By the cofactors' matrix, one can see that:

$$\begin{aligned}
A_{13}A_{12} - A_{23}A_{11} &= C_{23}\det(A) \\
-A_{23} &= (C_{22}C_{33} - C_{23}^2)\det(A) \\
A_{22}A_{33} - A_{23}^2 &= C_{11}\det(A)
\end{aligned}$$

so that

$$\frac{D_{12}}{D_{11}D_{22} - D_{12}^2} = \frac{C_{23}(C_{11}+1) - C_{12}C_{13}}{C_{11}+1}$$

Equivalently:

$$\begin{aligned}
D_{11}(D_{22}+1) - D_{12}^2 &= D_{11} + D_{11}D_{22} - D_{12}^2 \\
&= \frac{A_{22}(A_{11}+1) - A_{12}^2 + \det(A) + A_{22}A_{33} - A_{23}^2}{A_{11}+1}
\end{aligned}$$

hence

$$\begin{aligned}&\sqrt{(D_{11}(D_{22}+1)-D_{12}^2)\det(C)(A_{11}+1)}\\ =&\sqrt{\det(C)(A_{22}+A_{11}A_{22}-A_{12}^2+A_{33}A_{22}-A_{23}^2+\det(A))}\\ =&\sqrt{(C_{11}+1)(C_{33}+1)-C_{13}^2}\end{aligned}$$

where the last equality follows from:

$$A_{11}A_{22}-A_{12}^2=C_{33}\det(A),$$

$$A_{22}=(C_{11}C_{33}-C_{13}^2)\det(A),$$

$$\det(C)\det(A)=\det(C)\det(C^{-1})=1.$$

Finally, we have

$$I_3=\frac{2}{\pi}\frac{1}{\sqrt{(C_{11}+1)(C_{33}+1)-C_{13}^2}}\frac{C_{23}(C_{11}+1)-C_{12}C_{13}}{C_{11}+1}$$

which concludes the proof of the formula for I_3 that can be found in [7, Equations 15-16].

A.4 Computation of I_4

Let us now compute the last integral of interest, i.e.

$$I_4=\langle g'(Z_i)\,g'(Z_j)\,g(Z_k)\,g(Z_l)\rangle$$

where the average is over the gaussian distribution with zero mean and covariance matrix C, the projection of $\tilde{C}$ with respect to the indexes of interest. Let us now prove the quantity of interest. Let C be the four-dimensional covariance matrix of our gaussian random vector, and let A be its inverse. Then

$$\begin{aligned}I_4&=\frac{2}{\pi\sqrt{(2\pi)^4\det(C)}}\int_{\mathbb{R}^4}e^{-\frac{x^2}{2}}\cdot e^{-\frac{y^2}{2}}\cdot g(s)\cdot g(t)\cdot e^{-\frac{1}{2}(x,y,s,t)A(x,y,s,t)^T}\,dx\,dy\,dt\,ds\\ &=\frac{2\sqrt{\det(\hat{C})}}{\pi\sqrt{\det(C)}}\int_{\mathbb{R}^4}g(s)\cdot g(t)\cdot e^{-\frac{1}{2}(x,y,s,t)\hat{A}(x,y,s,t)^T}\frac{dx\,dy\,dt\,ds}{\sqrt{(2\pi)^4\det(\hat{C})}}\\ &=\frac{2\sqrt{\det(\hat{C})}}{\pi\sqrt{\det(C)}}\langle\langle\, g(X)g(Y)\,\rangle\rangle\end{aligned}$$

where $\langle\langle \cdot \rangle\rangle$ is the mean with respect to the gaussian distribution $(Z, W, X, Y) \sim \mathcal{N}(0, \hat{C})$, where $\hat{C} = \hat{A}^{-1}$ and

$$\hat{A} = A + E, \quad \text{where } E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that although the last average is over a 4-dimensional gaussian distribution, the function does not depend on Z and W at all, meaning that the average must be computed over the 2-dimensional gaussian distribution with zero mean and covariance matrix, the sub-matrix of $\hat{C}$

$$\begin{bmatrix} \hat{C}_{33} & \hat{C}_{34} \\ \hat{C}_{34} & \hat{C}_{44} \end{bmatrix}.$$

By (A.7) we have that

$$I_4 = \frac{4}{\pi^2} \frac{\sqrt{\det(\hat{C})}}{\sqrt{\det(C)}} \arcsin\left(\frac{\hat{C}_{34}}{\sqrt{\hat{C}_{33}+1}\sqrt{\hat{C}_{44}+1}} \right)$$

We will now give an expression for all the quantities above in terms of values of C only. First of all, note that:

$$\frac{\sqrt{\det(\hat{C})}}{\sqrt{\det(C)}} = \frac{1}{\sqrt{\det(\hat{A})\det(C)}} = \frac{1}{\sqrt{\det(\hat{A} \cdot C)}} = \frac{1}{\sqrt{\det((A+E)\cdot C)}} = \frac{1}{\sqrt{I + EC}}$$

After some simple computation, we find that $\Lambda_4 := \det(I+EC) = (C_{11}+1)(C_{22}+1) - C_{12}^2$. We can also compute the values in $\hat{C}$ in terms of that of C using the following matrix identities:

$$\hat{C} = \hat{A}^{-1} = (A+E)^{-1} = (C^{-1}+E)^{-1} = (C^{-1}(I+CE))^{-1} = (I-CE)^{-1}C$$

Therefore, in order to compute $\hat{C}_{33}, \hat{C}_{34}, \hat{C}_{44}$ we can use the following identities:

$$\hat{C}_{33} = \sum_{i=1}^{4} (I+CE)^{-1}_{3i} C_{i3}$$

$$\hat{C}_{34} = \sum_{i=1}^{4} (I+CE)^{-1}_{3i} C_{i4}$$

$$\hat{C}_{44} = \sum_{i=1}^{4} (I+CE)^{-1}_{4i} C_{i4}$$

Using the co-factors' matrix to compute the above-mentioned coefficients of $(I+CE)^{-1}$ one finds that:

$$(I+CE)^{-1}_{31} = \frac{C_{12}C_{23} - C_{13}(C_{22}+1)}{\Lambda_4}$$

$$(I+CE)^{-1}_{32} = \frac{C_{12}C_{13} - C_{23}(C_{11}+1)}{\Lambda_4}$$

$$(I+CE)^{-1}_{33} = 1$$

$$(I+CE)^{-1}_{34} = 0$$

$$(I+CE)^{-1}_{41} = \frac{C_{12}C_{24} - C_{13}(C_{22}+1)}{\Lambda_4}$$

$$(I+CE)^{-1}_{42} = \frac{C_{12}C_{14} - C_{23}(C_{11}+1)}{\Lambda_4}$$

$$(I+CE)^{-1}_{43} = 0$$

$$(I+CE)^{-1}_{44} = 1$$

And finally compute:

$$\hat{C}_{34} = \frac{\Lambda_0}{\Lambda_4} = \frac{\Lambda_4 C_{34} - C_{13}C_{14}(C_{22}+1) - C_{23}C_{24}(C_{11}+1) + C_{12}C_{14}C_{23} + C_{12}C_{24}C_{13}}{\Lambda_4}.$$

$$\hat{C}_{33} + 1 = \frac{\Lambda_1}{\Lambda_4} = \frac{\Lambda_4(C_{33}+1) - C_{13}^2(C_{22}+1) - C_{23}^2(C_{11}+1) + 2C_{12}C_{13}C_{23}}{\Lambda_4}$$

$$\hat{C}_{44} + 1 = \frac{\Lambda_2}{\Lambda_4} = \frac{\Lambda_4(C_{44}+1) - C_{14}^2(C_{22}+1) - C_{24}^2(C_{11}+1) + 2C_{12}C_{14}C_{24}}{\Lambda_4}$$

Putting everything together, we retrieve the formula in [7, Equations 16-19]. That is, defining:

$$\Lambda_4 = (C_{11}+1)(C_{22}+1) - C_{12}^2 \tag{A.8}$$

$$\Lambda_0 = \Lambda_4 C_{34} - C_{13}C_{14}(C_{22}+1) - C_{23}C_{24}(C_{11}+1) + C_{12}C_{14}C_{23} + C_{12}C_{24}C_{13} \tag{A.9}$$

$$\Lambda_1 = \Lambda_4(C_{33}+1) - C_{13}^2(C_{22}+1) - C_{23}^2(C_{11}+1) + 2C_{12}C_{13}C_{23} \tag{A.10}$$

$$\Lambda_2 = \Lambda_4(C_{44}+1) - C_{14}^2(C_{22}+1) - C_{24}^2(C_{11}+1) + 2C_{12}C_{14}C_{24} \tag{A.11}$$

we obtain:

$$I_4 = \frac{4}{\pi^2\sqrt{\Lambda_4}} \arcsin\left(\frac{\Lambda_0}{\sqrt{\Lambda_1}\sqrt{\Lambda_2}}\right) \tag{A.12}$$

Appendix B

Overlaps ODEs

In this section, we compute explicitly the differential equations for the order parameters as it was done in [4, 1].

B.1 Differential equation for R

Using the Gradient update for J_i (1.8) we find that:

$$J_i^{\mu+1} \cdot B_n = J_i^{\mu} \cdot B_n - \eta_J \Delta^{\mu} \, h_i^{\mu} \, g'(X_i^{\mu}) \, \frac{B_n \cdot \xi^{\mu}}{\sqrt{N}} \tag{B.1}$$

that is

$$\delta R_{in} = R_{in}^{\mu+1} - R_{in}^{\mu} = \frac{J_i^{\mu+1} \cdot B_n}{N} - \frac{J_i^{\mu} \cdot B_n}{N} = -\frac{\eta_J}{N} \, \Delta^{\mu} \, h_i^{\mu} \, g'(X_i^{\mu}) \, Y_n^{\mu} \tag{B.2}$$

Again, performing the thermodynamic limit and regarding μ/N as a continuous time we obtain:

$$\begin{aligned} \frac{\mathrm{d}R_{in}}{d\tau} &= -\eta_J \, h_i \, \langle \Delta \, g'(X_i) \, Y_n \rangle \\ &= \eta_J \, h_i \left[\sum_{m=1}^{M} v_m \langle g'(X_i) \, Y_n \, g(Y_m) \rangle - \sum_{j=1}^{K} h_j \langle g'(X_i) \, Y_n \, g(X_j) \rangle \right] \end{aligned} \tag{B.3}$$

which can be written in terms of I_3 as follows:

$$\frac{\mathrm{d}R_{in}}{\mathrm{d}\tau} = \eta_J \, h_i \left[\sum_{m=1}^{M} v_m \, I_3(i, K+n, K+m) - \sum_{j=1}^{K} h_j \, I_3(i, K+n, j) \right] \tag{1.21}$$

B.2 Differential equation for h

Now we apply the same reasoning as before directly to the gradient update (1.9) and write:

$$\delta h_i = h_i^{\mu+1} - h_i^{\mu} = -\frac{\eta_h}{N} \, \Delta^{\mu} \, g(X_i^{\mu}). \tag{B.4}$$

In the thermodynamic limit we get:

$$\begin{aligned}\frac{\mathrm{d}h_i}{d\tau} &= -\eta_h \langle g(X_i)\, \Delta \rangle \\ &= \eta_h \left[\sum_{m=1}^{M} v_m \langle g(Y_m)\, g(X_i) \rangle - \sum_{j=1}^{K} h_j \langle g(X_j)\, g(X_i) \rangle \right]\end{aligned} \tag{B.5}$$

which can be written in terms of the integrals I_2 as:

$$\frac{\mathrm{d}h_i}{d\tau} = \eta_h \left[\sum_{m=1}^{M} v_m\, I_2(K+m, i) - \sum_{j=1}^{K} h_j\, I_2(j, i) \right]. \tag{1.22}$$

Appendix C

Generate teacher networks

C.1 Scalar case

We now describe how to generate two teacher vectors with the prescribed overlap $V \in [0, 1]$. First, randomly generate an $N \times N$ matrix and consider Q the orthogonal matrix of its QR decomposition. Being Q_i the i-th row of Q, define:

$$B^{\diamondsuit} = \sqrt{N} Q_1$$

where the rescaling is performed to make each entry of $B^{\diamondsuit}$ of order $\mathcal{O}(1)$. Then, we define

$$B^{\heartsuit} = V B^{\diamondsuit} + \sqrt{1 - V^2}\, \hat{Q}_2$$

and $\hat{Q}_2 = \sqrt{N}\, Q_2$.

C.2 Matrix case

Assume now $M > 1$. Let V be the mutual overlap matrix of the teachers. We now need $B^{\diamondsuit}, B^{\heartsuit} \in \mathbb{R}^{M \times K}$ so that

$$V = \frac{\mathbf{B}^{\diamondsuit} (\mathbf{B}^{\heartsuit})^T}{N}.$$

Consider the singular value decomposition of V:

$$V = U \Sigma Z^T$$

where $\Sigma = diag(\sigma_1, \ldots, \sigma_M)$ is a diagonal matrix, whose diagonal contains the singular values of V, ordered in descending order. Let now Q be the orthogonal matrix of the QR decomposition of a gaussian matrix G of dimension N (i.e. a random square matrix). Then define:

$$B^{\diamondsuit} = \sqrt{N} U (\Sigma)^{1/2} Q_{1:M},$$

$$B^{\heartsuit} = \sqrt{N} Z (\Sigma)^{1/2} Q_{1:M},$$

$Q_{1:M}$ is the $M \times N$ matrix whose rows are the first M rows of Q and $(\Sigma)^{1/2} = diag(\sqrt{\sigma_i})$. Then:

$$\frac{\mathbf{B}^{\diamondsuit}(\mathbf{B}^{\heartsuit})^T}{N} = U(\Sigma)^{1/2} \underbrace{Q_{1:M}(Q_{1:M})^T}_{I_M} (\Sigma)^{1/2} Z^T = U \Sigma Z^T = V.$$

Another way of characterizing the similarity is by considering:

$$B^{\heartsuit} = V B^{\diamondsuit} + \sqrt{1 - V^2} B_{orth}$$

where B_{orth} is orthogonal to $B^{\diamondsuit}$ and V is a scalar quantity that can be thought as a scalar similarity between matrices. This is the method used in the simulations with $M > 1$.

List of Figures

Bibliography

[1] Sebastian Lee, Sebastian Goldt, and Andrew Saxe. Continual learning in the teacher-student setup: Impact of task similarity, 2021.

[2] Sebastian Lee, Stefano Sarao Mannelli, Claudia Clopath, Sebastian Goldt, and Andrew Saxe. Maslow's hammer for catastrophic forgetting: Node re-use vs node activation, 2022.

[3] Francesco Mori, Stefano Sarao Mannelli, and Francesca Mignacco. Optimal protocols for continual learning via statistical physics and control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2025(8):084004, August 2025.

[4] David Saad and Sara A. Solla. Exact solution for on-line learning in multilayer neural networks. *Phys. Rev. Lett.*, 74:4337–4340, May 1995.

[5] Volker Tresp. Committee machines. In Yu Hen Hu and Jenq-Neng Hwang, editors, *Handbook of Neural Network Signal Processing*. CRC press, 2018.

[6] E Gardner and B Derrida. Three unfinished works on the optimal storage capacity of networks. *Journal of Physics A: Mathematical and General*, 22(12):1983, jun 1989.

[7] David Saad and Sara A. Solla. On-line learning in soft committee machines. *Phys. Rev. E*, 52:4225–4243, Oct 1995.

[8] Michael Biehl and H Schwarze. Learning by on-line gradient descent. *Journal of Physics A: Mathematical and General*, 28:643–656, 02 1995.

[9] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning, 2023.

[10] Michael Biehl, Nestor Caticha, and Peter Riegler. Statistical mechanics of on-line learning. In Michael Biehl, Barbara Hammer, Michel Verleysen, and Thomas Vill-

mann, editors, *Similarity-Based Clustering: Recent Developments and Biomedical Applications*, pages 1–22. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, December 1989.

[12] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, November 1995.

[13] Sebastian Goldt and Udo Seifert. Thermodynamic efficiency of learning a rule in neural networks. *New Journal of Physics*, 19(11):113001, November 2017.

[14] David Saad and Sara A. Solla. Learning with noise and regularizers in multilayer neural networks. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

[15] Georg Reents and Robert Urbanczik. Self-averaging and on-line learning. *Physical Review Letters*, 80:5445–5448, 1998.

[16] Jean Jacod and Philip Protter. *Probability Essentials*. Universitext. Springer Berlin, Heidelberg, 2 edition, 2004.

[17] Sebastian Goldt, Madhu S Advani, Andrew M Saxe, Florent Krzakala, and Lenka Zdeborová. Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124010, December 2020.

[18] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence*, page 556–572. Springer International Publishing, 2018.

[19] Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning, 2023.

[20] Vinay V. Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics, 2020.

[21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[22] Edward W. Ng and Murray Geller. A table of integrals of the error functions. *Journal of research of the National Bureau of Standards - B. Mathematical Sciences*, 73B(1):643–656, 02-03 1969.