A PAC-Bayes bound for deterministic classifiers

Eugenio Clerico, George Deligiannidis, Benjamin Guedj, Arnaud Doucet

A PAC-Bayes bound for deterministic classifiers

Eugenio Clerico*[1], George Deligiannidis[1], Benjamin Guedj[2, 3], and Arnaud Doucet[1]

[1] *Department of Statistics, University of Oxford*
[2] *Centre for Artificial Intelligence, University College London*
[3] *Inria London*

Abstract

We establish a disintegrated PAC-Bayesian bound, for classifiers that are trained via continuous-time (non-stochastic) gradient descent. Contrarily to what is standard in the PAC-Bayesian setting, our result applies to a training algorithm that is deterministic, conditioned on a random initialisation, without requiring any *de-randomisation* step. We provide a broad discussion of the main features of the bound that we propose, and we study analytically and empirically its behaviour on linear models, finding promising results.

1 Introduction

Effectively upper-bounding the generalisation error of modern learning algorithms is one of the main open challenges for the statistical learning theory community (Zhang et al., 2016). Originally, properties of the hypothesis space, such as VC dimension and Rademacher complexity (Bousquet et al., 2004; Vapnik, 2000; Shalev-Shwartz and Ben-David, 2014), were used to establish *worst-case* generalisation bounds, holding uniformly over all the possible algorithms and datasets. However, these results are often vacuous when applied to the current over-parameterised models, the modern perspective focuses on algorithm- and data-dependent bounds (McAllester, 1998; Bousquet and Elisseeff, 2002; Dwork and Roth, 2014; Hardt et al., 2016; Xu and Raginsky, 2017; Clerico et al., 2022b; Lugosi and Neu, 2022). Among the various approaches proposed in the literature, the PAC-Bayesian framework (Guedj, 2019; Alquier, 2021) has obtained promising empirical results for deep networks (Dziugaite and Roy, 2017; Zhou et al., 2019; Pérez-Ortiz et al., 2021; Clerico et al., 2022a).

Typically, the PAC-Bayesian bounds are upper bounds on the expected population loss of stochastic classifiers, holding with high probability on the random draw of the training dataset. This setting, where the model's parameters are required to be intrinsically random, does not match with the more standard scenario of a non-randomised predictor. Although most available empirical PAC-Bayesian studies on neural networks focus on specifically designed stochastic architectures (Dziugaite and Roy, 2017; Zhou et al., 2019; Pérez-Ortiz et al., 2021; Clerico et al., 2022a), there are also *de-randomisation* methods that allow to apply PAC-Bayesian ideas to deterministic classifiers. For instance, one can approximate a deterministic model with a noisy version, whose parameters are stochastically perturbed. Under suitable stability assumptions, a PAC-Bayesian analysis of the stochastic model can bring a generalisation bound for the deterministic one (Neyshabur et al., 2018; Nagarajan and Kolter, 2019; Miyaguchi, 2019). Alternatively, one can average a stochastic algorithm to get a deterministic one. Then, PAC-Bayesian bounds on the loss of this averaged predictor can be obtained by Jensen's inequality (Germain et al., 2009; Letarte et al., 2019; Biggs and Guedj, 2021) (if the loss function is convex), or through margin-based arguments (Langford and Shawe-Taylor, 2002; Banerjee et al., 2020; Biggs and Guedj, 2022). Finally, besides the PAC-Bayesian bounds in expectation, there are disintegrated bounds that hold with high probability with respect to a random realisation of the stochastic algorithm. For instance, in the case of a stochastic network, this means that we have a generalisation bound for the deterministic classifier obtained by drawing all the parameters once (Catoni, 2007; Blanchard and Fleuret, 2007; Rivasplata et al., 2020; Viallard et al., 2021).

*Correspondence: *clerico@stats.ox.ac.uk*

When trying to apply the PAC-Bayesian framework to study a given non-stochastic model, the first of the methods proposed above is the most natural. However, in the present work, we show that for this purpose it is also possible to leverage the disintegrated PAC-Bayesian bounds. Our starting point is noticing that even training a *deterministic* classifier does usually involve some randomness: the standard procedure when training a neural network almost always starts with the random draw of the initial weights from some known simple distribution (Goodfellow et al., 2016). For a model trained via continuous-time (non-stochastic) gradient descent, we show that it is possible to exploit this source of stochasticity and obtain a disintegrated PAC-Bayes bound that holds with high probability on the random training dataset and initialisation (Proposition 2). Evaluating this bound only requires to be able to evaluate the density for the initial and final configurations of the network, and to compute the integral along the trajectory of the training objective's Laplacian. Although the bound necessitates the *ideal* situation of continuous training dynamics, to our knowledge this is the first PAC-Bayesian result that directly apply to a standard non-stochastic neural networks with a fully deterministic training, without strong assumptions on the architecture and the need of the introduction of any auxiliary stochastic model. Besides, we find our bound remarkably easy-to-prove, clean, and simple, with no strong hypotheses made on the training objective, which is only required to be twice differentiable.

The paper is organised as follows. First, we set up the learning framework and introduce the notations in Section 2. Section 3 gives a quick overview on the disintegrated PAC-Bayesian bounds, while in Section 4 we establish our main result (Proposition 2). Section 5 discusses the main features of our bound, and finally Section 6 presents some analytical and empirical quantitative results.

2 Notation and setting

In a standard supervised learning framework, we consider pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where x denotes an example and y its label. For simplicity, we can imagine that there is a mapping $f^\star : \mathcal{X} \to \mathcal{Y}$ that associates to each x a unique correct label $y = f^\star(x)$. A learning algorithm takes a labelled training dataset $s = \{x_1 \dots x_m\}$, made of m inputs whose labels $\{f^\star(x_1) \dots f^\star(x_m)\}$ are known, and outputs a function $f : \mathcal{X} \to \mathcal{Y}$, which ideally is a good approximation of $f^\star$. More generally, the algorithm will return an hypothesis $h \in \mathcal{H}$ (*e.g.*, values of a neural network's parameters), which is understood to parameterise a function $f_h : \mathcal{X} \to \mathcal{Y}$. For simplicity, we will always assume that $\mathcal{H} \subseteq \mathbb{R}^N$, for some dimension $N > 0$. We call the algorithm stochastic when its output h is a random variable on $\mathcal{H}$, whose law can depend on s.

The performance of each hypothesis can be assessed via a loss function ℓ, which takes a pair of labels (y, y') and returns a real value, representing how "far apart" y and y' are. In practice, for any hypothesis h, one can always compute the empirical loss on the training dataset

$$\mathcal{L}_s(h) = \frac{1}{m} \sum_{x \in s} \ell(f_h(x), f^\star(x)) \,.$$

However, we are essentially interested in how well h can predict the label of an x which does not belong to s. Assuming that the population of examples follows a distribution μ, the actual relevant quantity is the population loss

$$\mathcal{L}_{\mathcal{X}}(h) = \int_{\mathcal{X}} \ell(f_h(x), f^\star(x)) \mathrm{d}\mu(x) \,.$$

Upper-bounding $\mathcal{L}_{\mathcal{X}}$, having access to $\mathcal{L}_s$ only, is usually referred to as the generalisation problem. Typically, the training dataset is made of m i.i.d. samples from μ (*i.e.*, $s \sim \mu^m = \mu^{\otimes m}$). We are interested in upper bounds on $\mathcal{L}_{\mathcal{X}}(h)$ (where h is the hypothesis picked by the algorithm) that holds with high probability on the random draw of s (or of (h, s) in the stochastic setting).

We will consider an algorithm whose output is obtained by optimising an objective $\mathcal{C}_s : \mathcal{H} \to \mathbb{R}$ via continuous-time gradient descent. Clearly, $\mathcal{C}_s$ depends on the training dataset s and, in practice, it can coincide with the empirical loss, although this is not necessarily the case, as one might use a surrogate loss for the training or add some regularisation term. We require that the objective is twice differentiable, and we define the gradient flow $(h_0, t) \mapsto \Phi_t(h_0)$ as the solution of the dynamics

$$\partial_t \Phi_t(h_0) = -\nabla \mathcal{C}_s(\Phi_t(h_0)) \,; \qquad \Phi_0(h_0) = h_0 \,.$$

As h_0 is fixed before starting the training, we will often omit the explicit dependence on it, and simply write the solution of the above dynamics as h_t. Fixed a time horizon $T > 0$, our algorithm's output is given by h_T.

We will focus on the case of a random initialisation, with law ρ_0. We define the distribution ρ_t as the push-forward of ρ_0 under the gradient flow, that is

$$\rho_t = \Phi_t^{\#} \rho_0 \,.$$

In this way, if we draw h_0 from ρ_0, then we automatically get that $h_t \sim \rho_t$. Note that h_t is non-random when conditioned on the initialisation h_0. In particular, although the algorithm is in fact stochastic, we can see it as a *deterministic* model that has only a random initialisation. This is actually the common setting for modern neural networks, whose initial configuration is usually drawn from some simple distribution (*e.g.*, He et al. (2015)).

3 Disintegrated PAC-Bayes bounds

The main focus in the development of the PAC-Bayesian theory has been on bounds in expectation, and only recently Rivasplata et al. (2020) and Viallard et al. (2021) have brought back interest in the disintegrated bounds, which actually date back to the works of Catoni (2007) and Blanchard and Fleuret (2007). We refer to Alquier (2021) for a detailed introductory exposition of the PAC-Bayesian framework that also discusses some disintegrated results.

We consider a stochastic model, which takes a sample $s \sim \mu^m$ and returns a random hypothesis $h \sim \rho^s$, where the superscript ^{s} that we use in this section stresses explicitly that ρ depends on s.[1] We denote the joint law of (s, h) as $\mu^m * \rho^s$, that is $\mathrm{d}(\mu^m * \rho^s)(s, h) = \mathrm{d}\mu^m(s)\mathrm{d}\rho^s(h)$. As it is standard in the PAC-Bayesian literature, we will call ρ^s the *posterior* distribution. We denote as π a *prior* distribution on $\mathcal{H}$, whose only requirement is to be data-agnostic, in the sense that it cannot depend on the specific dataset s used for the training. We write $\mu^m \otimes \pi$ for the law of a pair (s, h), where $s \sim \mu^m$ and $h \sim \pi$ are independent.

A disintegrated PAC-Bayesian bound is an upper bound on $\mathcal{L}_{\mathcal{X}}(h)$, where h is the output of a stochastic algorithm, that holds with high probability over $(s, h) \sim \mu^m * \rho^s$. The next result, from Rivasplata et al. (2020), allows to derive several of these bounds. We provide a short proof for completeness.

Proposition 1. *Let $\Psi : \mathbb{R}^2 \to \mathbb{R}$ be an arbitrary measurable function, and define ξ as the expectation of $e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h))}$ under $(s, h) \sim \mu^m \otimes \pi$. For any $\delta \in (0, 1)$, for any posterior distribution ρ^s absolutely continuous with respect to the prior π, with probability at least $1 - \delta$ on $(s, h) \sim \mu^m * \rho^s$ we have that*

$$\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h)) \leq \log \frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h) + \log \frac{\xi}{\delta} \,.$$

Proof. By Markov's inequality, for $(s, h) \sim \mu^m * \rho^s$ we have that with probability at least $1 - \delta$

$$e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h)) - \log \frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h)} \leq \frac{1}{\delta} \int_{\mathcal{X}^m \times \mathcal{H}} e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h)) - \log \frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h)} \mathrm{d}\mu^m(s)\mathrm{d}\rho^s(h) \,.$$

By noticing that for all $s \in \mathcal{X}^m$

$$\int_{\mathcal{H}} e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h)) - \log \frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h)} \mathrm{d}\rho^s(h) = \int_{\mathcal{H}} \frac{e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h))}}{\frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h)} \chi_{\frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h) > 0} \mathrm{d}\rho^s(h) = \int_{\mathcal{H}} e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h))} \mathrm{d}\pi(h) \,,$$

we easily get the desired result. □

Choosing $\Psi(u, v) = 2m(u - v)^2$ one finds the disintegrated version of a standard PAC-Bayes bound from McAllester (1999). Alternatively, defining $\mathrm{kl}(u\|v) = u \log \frac{u}{v} + (1 - u) \log \frac{1-u}{1-v}$ (the relative entropy between two binary Bernoulli distributions) and setting $\Psi(u, v) = m\mathrm{kl}(u\|v)$, we get a tighter bound, which is the disintegrated counterpart of a bound due to Langford and Seeger (2001) and Maurer (2004). These results follow from the fact that in both cases ξ can be upper-bounded by $2\sqrt{m}$, if the loss ℓ is bounded in $[0, 1]$ (Bégin et al., 2016).

[1] To be rigorous, one should actually require that $s \mapsto \rho^s$ is a Markov kernel.

Corollary 1. *Assume that the loss ℓ is bounded in $[0,1]$. Then, for any $\delta \in (0,1)$, for any posterior distribution ρ^s absolutely continuous with respect to the prior π, each of the following bounds hold with probability at least $1-\delta$ on $(s,h) \sim \mu^m * \rho^s$*

$$\mathcal{L}_{\mathcal{X}}(h) \leq \mathcal{L}_s(h) + \sqrt{\frac{\log \frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h) + \log \frac{2\sqrt{m}}{\delta}}{2m}}\,;$$

$$\mathcal{L}_{\mathcal{X}}(h) \leq \mathrm{kl}^{-1}\left(\mathcal{L}_s(h) \middle| \frac{\log \frac{\mathrm{d}\rho^s}{\mathrm{d}\pi}(h) + \log \frac{2\sqrt{m}}{\delta}}{m}\right),$$

where $\mathrm{kl}^{-1}(u|c) = \sup\{v \in [0,1] : \mathrm{kl}(u\|v) \leq c\}$.

4 Disintegrated bounds under gradient flow dynamics

We state here our main result, a disintegrated bound for an algorithm trained by continuous gradient-descent optimisation. We recall some notation from Section 2. We denote as $\mathcal{C}_s$ a twice differentiable training objective, and we consider a random initial state $h_0 \sim \rho_0$. The dynamics are described by

$$\partial_t h_t = -\nabla \mathcal{C}_s(h_t)\,,$$

where we omit the explicit dependence of h_t on s and h_0. The algorithm's output is given by h_T, with $T > 0$ some fixed time horizon. We define ρ_t as the push-forward of ρ_0 under the gradient flow, so that $h_t \sim \rho_t$. For simplicity, for all t we assume that ρ_t admits a density with respect to the Lebesgue measure, which we will again denote as ρ_t, with a slight abuse of notation.

It is natural to suppose that ρ_0 is chosen without relying on the specific dataset s used for the training, so that, in particular, we can select ρ_0 as our PAC-Bayesian *prior*. On the other hand, ρ_T depends on s through the objective $\mathcal{C}_s$, and plays the role of the *posterior*. Sampling the initialisation h_0 of the model from ρ_0, and then following the deterministic training dynamics up to T, we get a final hypothesis h_T, which can be seen as a sample from ρ_T. In particular, saying that an event holds with probability at least $1-\delta$ under $(s,h_T) \sim \mu^m * \rho_T$ is equivalent to state that it has probability at least $1-\delta$ under $(s,h_0) \sim \mu^m \otimes \rho_0$. We can thus use Proposition 1 to obtain a PAC-Bayesian generalisation bound for an algorithm that, once drawn s and h_0, is deterministic.

So far, we have a bound that involves the term $\rho_T(h_T)/\rho_0(h_T)$. Typically, the density ρ_0 is known, and evaluating $\rho_0(h_T)$ does not pose any problem. On the other hand, computing $\rho_T(h_T)$ is less straightforward. However, thanks to the continuity of the gradient descent dynamics, it is possible to keep track of the evolution of $\rho_t(h_t)$ along the training trajectory via a simple trick, usually referred to as "instantaneous change of variables" in the normalising flow literature (Chen et al., 2018).

Proposition 2. *Consider the dynamics $\partial_t h_t = -\nabla \mathcal{C}_s(h_t)$, where $\mathcal{C}_s : \mathcal{H} \to \mathbb{R}$ is twice differentiable. Let $\Psi : \mathbb{R}^2 \to \mathbb{R}$ be a measurable function. Fixed $T > 0$[2] and $\delta \in (0,1)$, with probability at least $1-\delta$ on the random draw $(s,h_0) \sim \mu^m \otimes \rho_0$, we have*

$$\Psi(\mathcal{L}_s(h_T), \mathcal{L}_{\mathcal{X}}(h_T)) \leq \log \frac{\rho_0(h_0)}{\rho_0(h_T)} + \int_0^T \Delta \mathcal{C}_s(h_t)\mathrm{d}t + \log \frac{\xi}{\delta}\,,$$

where Δ denotes the Laplacian with respect to h and $\xi = \int_{\mathcal{X}^m \times \mathcal{H}} e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h))} \mathrm{d}\mu^m(s)\mathrm{d}\rho_0(h)$.

Proof. Since sampling (s,h_T) from $\mu^m * \rho_T$ is equivalent to drawing $(s,h_0) \sim \mu^m * \rho_0$ and following the dynamics up to T, Proposition 1 yields directly that, with probability at least $1-\delta$ on $(s,h_0) \sim \mu^m \otimes \rho_0$,

$$\Psi(\mathcal{L}_s(h_T), \mathcal{L}_{\mathcal{X}}(h_T)) \leq \log \frac{\rho_T(h_T)}{\rho_0(h_T)} + \log \frac{\xi}{\delta}\,.$$

The gradient flow's continuity equation states that, for all $h \in \mathcal{H}$,

$$\partial_t \rho_t(h) = \nabla \cdot (\rho_t(h) \nabla \mathcal{C}_s(h)) = \nabla \rho_t(h) \cdot \nabla \mathcal{C}_s(h) + \rho_t(h) \Delta \mathcal{C}_s(h)\,.$$

[2] To be fully rigorous, one should explicitly assume that the solution $t \mapsto h_t$ of the dynamics is well defined on $[0,T]$, for all initial conditions $h_0 \in \mathcal{H}$.

It follows that

$$\partial_t(\rho_t(h_t)) = \partial_t\rho_t(h_t) + \nabla\rho_t(h_t)\cdot\partial_t h_t = \rho_t(h_t)\Delta\mathcal{C}_s(h_t)\,,$$

which yields

$$\log\frac{\rho_T(h_T)}{\rho_0(h_0)} = \int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t\,.$$

Hence

$$\log\frac{\rho_T(h_T)}{\rho_0(h_T)} = \log\frac{\rho_0(h_0)}{\rho_0(h_T)} + \int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t\,,$$

and we conclude. □

Note that the time horizon T must be chosen a priori, as it cannot depend on the specific s and h_0 used for the training. However, through a classical union bound argument, one can slightly loosen the bound while allowing to select the optimal T from a finite pool of possible values. Concretely, it is possible to pick the best T from a set containing K potential choices at the cost of replacing the term $\log\frac{\xi}{\delta}$ with $\log\frac{K\xi}{\delta}$, in the bounds of Propositions 1 and 2.

Clearly, for $\ell\subseteq[0,1]$, the choices of Ψ that lead to Corollary 1 will again bring explicit bounds. Setting $\Psi(u,v)=2m(u-v)^2$ yields

$$\mathcal{L}_{\mathcal{X}}(h_T) \leq \mathcal{L}_s(h_T) + \sqrt{\frac{\log\frac{\rho_0(h_0)}{\rho_0(h_T)} + \log\frac{2\sqrt{m}}{\delta} + \int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t}{2m}}\,, \tag{1}$$

while from $\Psi(u,v)=m\mathrm{kl}(u|v)$ we get

$$\mathcal{L}_{\mathcal{X}}(h_T) \leq \mathrm{kl}^{-1}\left(\mathcal{L}_s(h_T)\middle|\frac{\log\frac{\rho_0(h_0)}{\rho_0(h_T)} + \log\frac{2\sqrt{m}}{\delta} + \int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t}{m}\right)\,, \tag{2}$$

where each of the above two bounds holds with probability higher than $1-\delta$ on the random initialisation and training dataset.

We remark that Proposition 2 holds also for time-dependent objectives (as long as they do not depend on ρ_t), as the gradient flow's continuity equation is still valid. Leveraging this, we can specialise the result for training with (potentially randomly selected) batches. We refer to Section 5.5 for further details. Finally, an interesting alternative scenario is when the continuous-time dynamics are stochastic. For instance, one might want to look at the Langevin's diffusion $\mathrm{d}h_t = -\nabla\mathcal{C}_s(h_t) + \sigma\mathrm{d}B_t$, where $\sigma>0$ is a fixed coefficient and B_t a Brownian motion. However, in this case we do not end up anymore with an easily tractable exact expression for $\log\frac{\rho_T(h_T)}{\rho_0(h_0)}$ (see Appendix A).

5 Analysis and discussion

5.1 A few comments on the Laplacian term

An interesting feature of the bound in Proposition 2 is the integral of the Laplacian of the optimisation objective along the training path:

$$\int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t\,. \tag{3}$$

Under the gradient flow dynamics, $\Delta\mathcal{C}_s(h_t)$ is keeping track of how the local probability density is locally varying. Indeed, from a PAC-Bayesian point of view, for the bound to be small we need to end up in some point where the *posterior* density is not too high compared to the initial *prior*. If we are following a trajectory characterised by high values of the Laplacian, we will see a sharp increase of the density. Intuitively, we can picture the situation as if we were attracting the nearby paths and bringing further *probability mass* around us. In this case we are likely to end with a final ρ_T that is much larger than the initial ρ_0, and hence, potentially, a loose bound.

It is worth noticing that, in the one-dimensional case, (3) takes a particularly simple form, as it only depends on the initial and final "velocities". To see this, note that $\partial_t\mathcal{C}_s'(h_t) = \mathcal{C}_s''(h_t)\partial_t h_t = -\mathcal{C}_s''(h_t)\mathcal{C}_s'(h_t)$,

with $\mathcal{C}'_s$ and $\mathcal{C}''_s$ denoting the first and second derivatives of $\mathcal{C}_s$. It follows that $\mathcal{C}''_s(h_t) = -\partial_t \log |\mathcal{C}'_s(h_t)|$, which yields

$$\int_0^T \mathcal{C}''_s(h_t)\mathrm{d}t = \log \frac{\mathcal{C}'_s(h_0)}{\mathcal{C}'_s(h_T)} \,.$$

On the other hand, things are not that simple in the multidimensional case. With a few algebraic manipulations, we can write

$$\int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t = \log \frac{\|\nabla\mathcal{C}_s(h_0)\|}{\|\nabla\mathcal{C}_s(h_T)\|} - \int_{h_{[0:T]}} \nabla \cdot \tau(h) \, \|\delta h\| \,, \tag{4}$$

where $\tau(h)$ stands for the unit tangent vector to the trajectory in h, and $\int_{h_{[0:T]}} \dots \|\delta h\|$ denotes the line integral along the path $h_{[0:T]}$. The last term has a clear interpretation, as it quantifies how much $h_{[0:T]}$ is attracting the nearby trajectories. We refer to Appendix B for the derivation of (4).

Although we are not aware of any generalisation bound that involves the exact term (3), the Laplacian's integral might find a natural interpretation through the connection between flatness in the loss landscape and good generalisation. It dates back to Hochreiter and Schmidhube (1997) the belief that an algorithm, able to find wide local optima of the loss, performs well on unseen inputs. Re-popularised by Keskar et al. (2017), this idea has been the object of an intense debate, attracting some criticisms as the definitions of *flatness* are often susceptible to re-parameterisations of the hypothesis space (Dinh et al., 2017; Neyshabur et al., 2017). However, numerous recent works have provided empirical and theoretical support, as well as new insights, to the flat minima's argument (Izmailov et al., 2018; He et al., 2019; Dziugaite and Roy, 2017; Neu et al., 2021). Going back to our setting, if we assume that the objective $\mathcal{C}_s$ coincides with the empirical loss, we clearly see that sharper loss's local optima correspond to higher values of the Laplacian term. As a consequence, converging towards flatter minima will typically yield a tighter bound.

5.2 Role of the time horizon

Most of the standard PAC-Bayesian bounds (in expectation) contain the Kullback-Leibler divergence between the *prior* and the *posterior*. This quantity is infinity if the *posterior* is a Dirac delta, and this is why these results cannot apply directly to the deterministic setting. Even the disintegrated framework is not exempt from this pathological behaviour, which is reflected by the fact that in Proposition 2 we cannot let the time horizon T to be infinite even if h_t converges, as the *posterior* would be degenerate and typically consist in a sum of Dirac deltas centred on the local optima. Concretely, we can easily see that if an algorithm is converging to a local minimum (where $\mathcal{C}_s$ is approximately quadratic), then (1) will asymptotically go as

$$\mathcal{L}_{\mathcal{X}}(h) - \mathcal{L}_s(h) \lesssim O(\sqrt{T})$$

for large T (see Appendix C for more details). However, this is not the only possible asymptotic behaviour. For instance, we can consider situations where h_t is not converging but diverging. As we will see in Section 6, this is what happens for linear models trained to optimise a linear objective or the cross-entropy.

Although characterising the typical spectrum of the Hessian of the loss during the training evolution is a challenging open problem, recent empirical studies suggest that after an initial phase where the Laplacian can potentially be negative, later stages of the dynamics are characterised by a few dominant positive Hessian's eigenvalues (Sagun et al., 2018; Ghorbani et al., 2019). As a consequence, in most situations the Laplacian's integral will soon become monotonically increasing with T. Moreover, we can expect that $\log \frac{\rho_0(h_0)}{\rho_0(h_T)}$ will grow with T as well, since h_0 is likely to lie in a region where the *prior* density is high, and the training will typically push h_t farther and farther from it. Overall, in order to achieve non-vacuous bounds, we need to stop the training at a finite time horizon T, which will play the role of a trade-off between how well we can fit the training dataset, and how much we can keep the bound small.

A final remark, already mentioned earlier, is that T must be chosen independently of the dataset s and initial state h_0 used for the training. However, adding a penalty $\log K$ to the bounds of Propositions 1 and 2 allows us to pick the best time horizon among a set $\{T_1 \dots T_K\}$ of K possible choices. This is the consequence of an elementary union argument, as for each T_k we can consider a bound holding with probability at least $1 - \delta/K$.

5.3 Scaling with the dimension of the hypothesis space

When dealing with generalisation bounds, a natural question is how they scale with the dimension N of the hypothesis space. In our case, the answer is not easy, as it is hard to say how the different components of bound depend on N. For instance, we can expect the Laplacian to scale linearly with N, as it is the sum of N terms. If $T = O(1)$ and the Laplacian's integral dominates the bound, (1) becomes $O(\sqrt{N/m})$, which is a standard behaviour. On the other hand, if the log-density term $\log \frac{\rho_0(h_0)}{\rho_0(h_T)}$ is the main contribution to the bound, the scaling with N might be more complex, as it will depend on the particular choice of the *prior* density.

Interestingly, the optimal time horizon T can scale with N as well, and empirical results on MNIST show that for a linear model we can have $T \sim O(1/N)$, and hence obtain non-vacuous bounds even in a highly over-parameterised setting (see Figure 1). Intuitively, when N is large, the algorithm can quickly find an hypothesis that fit well the training data and is "close" to h_0, hence requiring a low training time to be reached. On the other hand, fewer learnable parameters might need a longer training time before reaching low values of the empirical loss. Despite the simple setting of our experiment, we believe that this is a promising result as the bound seems unaffected by the *curse of dimensionality*. However, a more detailed empirical and theoretical analysis is needed in order to gauge how well our bound performs with more complex models, such as deep networks.

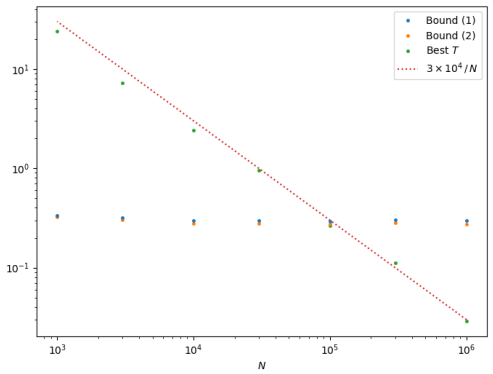

Figure 1: Behaviour of best time horizon T with the model dimension N, from experiments on MNIST for linear models trained to optimise a cross-entropy objective. The optimal T was found to go roughly as $1/N$ (*cf.* dotted line), preventing the bounds (1) and (2) from exploding, even in highly over-parameterised settings. Note that the results reported here were obtained via discretised gradient dynamics (*cf.* Section 5.4 and Figure 2). We refer to Appendix E.3 for further details on the experiment.

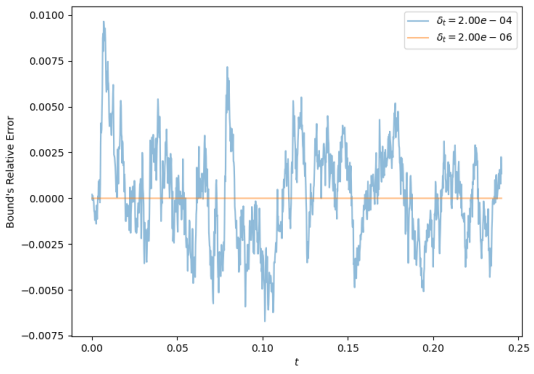

Figure 2: Relative error $(B_1 - B_2)/B_2$, for a linear model with 10^5 parameters trained on MNIST via discretised gradient steps on a cross-entropy objective. Here, B_1 and B_2 are the values of the bound (2) obtained with the time-steps $\delta_{t,1} = 2 \times 10^{-4}$ and $\delta_{t,2} = 2 \times 10^{-6}$ respectively. The small relative error suggests that the corrections to the bound due to the time-discretised dynamics are likely to be negligible. We refer to Appendix E.3 for further details on the experiment.

5.4 Discretised dynamics

One of the main limitations of the present work is that the bound of Proposition 2 requires a continuous-time gradient descent. Apart from a few simple settings, where analytic solutions are available, the standard gradient-based optimisation algorithms consider discretised dynamics, whose trajectories are made of jumps and only approximate the continuous flow. As a consequence, in order to leverage our results and get rigorous generalisation bounds for the standard gradient-descent optimisers, one would need to control the errors introduced by the time discretisation via a careful analysis, which is beyond the scope of the present work. However, we can at least conjecture that, for small enough learning rates, the results obtained via the discretised dynamics are a good approximation of the exact solution of the gradient flow. In Figure 2, the relative error between the values of the bound (2), obtained from two different time discretisations, stays small throughout the training, suggesting that the discretised results can be good approximations of the actual continuous dynamics.

Interestingly, Barrett and Dherin (2021) remarked that the solutions of the discretised dynamics $h_{k+1} - h_k = -\varepsilon \nabla \mathcal{C}_s(h_k)$ follow closely the continuous paths yielded by $\partial_t h_t = -\nabla(\mathcal{C}_s(h_t) + \varepsilon \|\nabla \mathcal{C}_s(h_t)\|^2)$, with an error after a single step of order $O(\varepsilon^3)$[3]. Consequently, if we aim at studying the discrete dynamics, we could obtain a more precise approximation of an actual generalisation bound by replacing the term $\Delta\mathcal{C}_s$ with $\Delta(\mathcal{C}_s(h_t) + \varepsilon \|\nabla \mathcal{C}_s(h_t)\|^2)$. If higher accuracy is required, one can add further correction terms to $\mathcal{C}_s$, as it is standard in the backward error analysis of Runge-Kutta methods (Hairer et al., 2006).

5.5 Training via batches

So far, we have considered the case of a training objective $\mathcal{C}_s$ which is fixed throughout the training. However, the bound of Proposition 2 holds also for time-dependent objectives. In particular, we can apply our result to the case in which the training is performed by optimising the empirical loss over different batches of the training dataset. Let us denote as $s_1 \dots s_M$ the (temporarily ordered) sequence of batches used for the training, where a batch needs to appears multiple times if it is used more than once. It is worth noticing that the batches and their order are allowed to be selected randomly, as it is usually the standard stochastic gradient descent optimisation. Then, we can suppose that from $t = 0$ to $t = t_1$ we use $\mathcal{C}_{s_1}$ as optimisation objective, from $t = t_1$ to $t = t_2$, we use $\mathcal{C}_{s_2}$, and so on until $t_M = T$. We can get a bound as the one of Proposition 2, with the only difference that $\int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t$ will now be replaced by

$$\int_0^{t_1} \Delta\mathcal{C}_{s_1}(h_t)\mathrm{d}t + \int_{t_1}^{t_2} \Delta\mathcal{C}_{s_2}(h_t)\mathrm{d}t + \dots + \int_{t_{M-1}}^{t_M} \Delta\mathcal{C}_{s_M}(h_t)\mathrm{d}t\,. \tag{5}$$

5.6 Data-dependent priors

For most of PAC-Bayesian results, the *prior* is required to be chosen independently of the dataset used to evaluate the bound. However, we can use an additional dataset s' (independent of s) to learn the *prior*. This strategy was first proposed by Seeger (2002) and has lead to the tightest currently available empirical PAC-Bayesian bounds on MNIST and CIFAR10 (Pérez-Ortiz et al., 2021; Clerico et al., 2022a).

In our setting we might proceed in a similar way. Till a fixed time t_0, we follow the dynamics $\partial_t h_t = -\nabla\mathcal{C}_{s'}(h_t)$, and we select ρ_{t_0} as the PAC-Bayesian *prior*. Then, we take $\mathcal{C}_s$ as training objective and we go on as usual, starting from h_{t_0} and following the gradient flow until the time horizon T. In order to evaluate the bound, we need to compute $\log \frac{\rho_T(h_T)}{\rho_{t_0}(h_T)}$. Proceeding as in the proof of Proposition 2, we can get

$$\log \frac{\rho_T(h_T)}{\rho_{t_0}(h_T)} = \log \frac{\rho_{t_0}(h_{t_0})}{\rho_{t_0}(h_T)} + \int_{t_0}^{T} \Delta\mathcal{C}_s(h_t)\mathrm{d}t = \log \frac{\rho_0(h_{t_0})}{\rho_{t_0}(h_T)} + \int_0^{t_0} \Delta\mathcal{C}_{s'}(h_t)\mathrm{d}t + \int_{t_0}^{T} \Delta\mathcal{C}_s(h_t)\mathrm{d}t\,.$$

Now, the only problem is to evaluate $\log \rho_{t_0}(t_T)$, as we do not have a closed form for ρ_{t_0}. However, it is possible to exactly compute this term as

$$\log \rho_{t_0}(h_T) = \log \rho_0(\hat{h}_{T-t_0}) + \int_0^{t_0} \Delta\mathcal{C}_{s'}(\hat{h}_{T-t_0+t})\mathrm{d}t\,, \tag{6}$$

where $\hat{h}_t$ is the solution of $\partial_t \hat{h}_t = -\nabla\mathcal{C}_{s'}(\hat{h}_t)$ satisfying $\hat{h}_T = h_T$. We hence get

$$\log \frac{\rho_T(h_T)}{\rho_{t_0}(h_T)} = \log \frac{\rho_0(h_{t_0})}{\rho_0(\hat{h}_{T-t_0})} + \int_0^{t_0} (\Delta\mathcal{C}_{s'}(h_t) - \Delta\mathcal{C}_{s'}(\hat{h}_{T-t_0+t}))\mathrm{d}t + \int_{t_0}^{T} \Delta\mathcal{C}_s(h_t)\mathrm{d}t\,.$$

Plugging this expression in Proposition 1 we readily obtain a bound whose prior ρ_{t_0} is data-dependent. We refer to Appendix D for an explicit statement of the bound and a derivation of (6).

[3] For $\partial_t h_t = -\nabla\mathcal{C}_s(h_t)$ the single-step error is of order $O(\varepsilon^2)$.

6 Linear models

To give an idea of the behaviour of the bounds that we propose, we report here some analytic and empirical results. We consider models linear in the parameters, in the form

$$F_h^i(x) = \sum_j h^{ij} \Phi^j(x) \,,$$

where the Φ^j are some fixed mappings $\mathbb{R} \to \mathbb{R}$. For binary classification tasks, where $y \in \{-1, +1\}$, we let F_h be a real function and the model's prediction f_h corresponds to the sign of F_h. Conversely, for general multi-class problems the predicted label f_h is the index of the largest component of F_h.

As it is standard in the classification setting, we can assume that ℓ is the 01-loss[4], so that $\mathcal{L}_s$ and $\mathcal{L}_\mathcal{X}$ represent the empirical and population error. Unless otherwise stated, we will always consider an optimisation objective in the form $\mathcal{C}_s(h) = \hat{\mathcal{L}}_s(h) = \frac{1}{m} \sum_{x \in s} \hat{\ell}(F_h(x), f^\star(x))$, with $\hat{\ell}$ denoting some twice differentiable surrogate loss function. Applying the chain rule to $\hat{\ell}$, we find that

$$\Delta \hat{\ell} = \sum_i \frac{\partial \hat{\ell}}{\partial F^i} \Delta F^i + \sum_{ii'} \frac{\partial^2 \hat{\ell}}{\partial F^i \partial F^{i'}} \nabla F^i \cdot \nabla F^{i'} \,,$$

where Δ and ∇ always refer to derivatives with respect to h. In the linear models that we are considering $\Delta F = 0$ and $\nabla F^i \cdot \nabla F^{i'} = \delta_{ii'} \|\Phi\|^2$. Thus

$$\Delta \hat{\ell} = \|\Phi\|^2 \Delta_F \hat{\ell} \,, \tag{7}$$

with $\Delta_F \hat{\ell} = \sum_i \partial^2_{F^i} \hat{\ell}$, the Laplacian of $\hat{\ell}$ with respect to F.

6.1 Linear loss

A first simple example that we can solve analytically is a binary classification problem with

$$\hat{\ell}(F, y) = -Fy \,.$$

The surrogate loss is linear in F, and so (7) yields $\Delta \hat{\ell} = 0$. In particular, the Laplacian's integral in our bounds vanishes. We can easily solve the evolution dynamics and find that

$$h_T = h_0 + \gamma T \,,$$

where, recalling that $f^\star(x)$ is the correct label of the example x, we have defined $\gamma = \frac{1}{m} \sum_{x \in s} f^\star(x) \Phi(x)$. Assuming that under ρ_0 the components of h are all independent centred random variables, with variance $1/N$ (N being the dimension of h), we find that $\log \frac{\rho_0(h_0)}{\rho_0(h_T)} = \frac{N}{2}(\|h_T\|^2 - \|h_0\|^2)$. So, (1) becomes

$$\mathcal{L}_\mathcal{X}(h_T) - \mathcal{L}_s(h_T) \leq \sqrt{\frac{N}{4m} \left(2T h_0 \cdot \gamma + T^2 \|\gamma\|^2\right) + \frac{1}{2m} \log \frac{2\sqrt{m}}{\delta}} \,.$$

Tipycally, $\|h_0\| \sim O(1)$, and we expect $\|h_T\| \sim T\|\gamma\|$ to dominate the dynamics. Notice that the bound scales linearly with T. This is different from the $O(\sqrt{T})$ behaviour that we had encountered in section 5.2. Indeed, here h is not approaching a local optimum, but diverging at constant speed γ.

We tested empirically the behaviour of the bounds (1) and (2) with a linear surrogate loss, by training a model where the learnable vector h had 10000 components. The dataset was made of 500 datapoints, coming from Gaussian clusters in $\mathbb{R}^5$ and labelled into two categories. The map Φ was in the form $x \mapsto \phi(Wx)$, with ϕ the ReLU function and W a (5×10000)-matrix whose entries were independent draws from a centred Gaussian with variance $1/5$. We refer to Appendix E.1 for further experimental details and results. Figure 3 compares the behaviour of the two bounds with the test errors obtained from a held-out dataset, for 50 different possible values of the time horizon[5] and for $\delta = 5 \times 10^{-3}$. Note that for large T (1) is linear, as expected.

[4] $\ell(y, y') = 0$ if $y = y'$; 1 otherwise.

[5] The penalty factor $\log 50$ (see end of Section 5.2) is already included in the values reported.

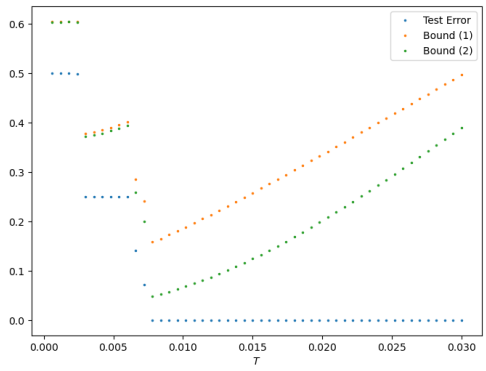

Figure 3: Behaviour of the bounds (1) and (2) for a linear model with 10000 parameters, trained with a linear surrogate loss, for different time horizons T. The data came from eight different Gaussian distributions on $\mathbb{R}^5$ that had been assigned binary labels. The training dataset counted 500 datapoints and a held-out test dataset was used to evaluate the test errors reported in the plot. We refer to Appendix E.1 for further details.

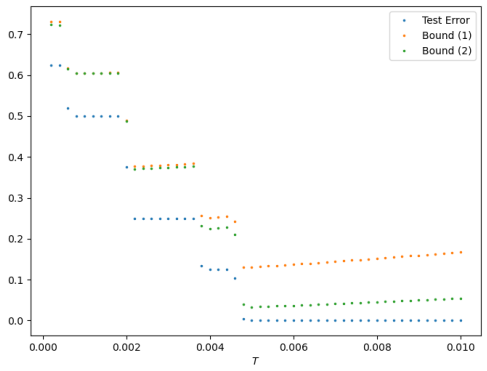

Figure 4: Behaviour of the bounds (1) and (2) for a linear model with 10000 parameters, trained with a quadratic surrogate loss, for different time horizons T. The learning task was the same as in Figure 3. Again, we used a training dataset made of 500 datapoints and we evaluated the test errors on a held-out test dataset. The parameters α and β in $\mathcal{C}_s$ were both set to 1. We refer to Appendix E.2 for further details.

6.2 Quadratic loss

Another natural situation where we can solve things explicitly is the case of a quadratic surrogate loss

$$\hat{\ell}(F, y) = \frac{1}{2}(F - \beta y)^2 \,.$$

Here, β is some fixed positive value, and we are still considering a binary classification problem, where the true labels are $+1$ or -1 and the network's prediction is the sign of F. Now, $\Delta_F \hat{\ell} = 1$ and hence (7) yields $\Delta\hat{\ell} = \|\Phi\|^2$. We consider a training objective in the form

$$\mathcal{C}_s(h) = \frac{\alpha\|h\|^2}{2N} + \frac{1}{m}\sum_{x\in s} \hat{\ell}(F_h(x), f^\star(x)) \,,$$

with $\alpha \geq 0$ and N the dimension of h. The regularising term $\alpha\|h\|^2/(2N)$ is here introduced to computationally stabilise the algorithm.[6] We easily find that

$$\int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t = \Gamma T \,,$$

with $\Gamma = \alpha + \frac{1}{m}\sum_{x\in s}\|\Phi(x)\|^2$. The training dynamics are now governed by $\partial_t h_t = \beta\gamma - \Theta h_t$, where again we have defined $\gamma = \frac{1}{m}\sum_{x\in s} f^\star(x)\Phi(x)$, while $\Theta = \frac{\alpha}{N}\,\mathrm{Id} + \frac{1}{m}\sum_{x\in s}\Phi(x)\Phi(x)^\top$ is a $N \times N$ matrix. Hence, assuming that Θ is invertible, we have

$$h_T = h_0 + (\mathrm{Id} - e^{-T\Theta})(\Theta^{-1}\beta\gamma - h_0) \,.$$

If again we consider a Gaussian initialisation, (1) becomes

$$\mathcal{L}_{\mathcal{X}}(h_T) - \mathcal{L}_s(h_T) \leq \sqrt{\frac{\frac{N}{2}(\|h_0\|^2 - \|h_T\|^2) + \Gamma T}{2m}} \,.$$

As the objective is quadratic, for large T this bound will behave as $O(\sqrt{T})$, as it happens more generally every time the algorithm is converging to a local minimum of the objective (see Section 5.2).

Figure 4 shows the bounds obtained by optimising a quadratic objective with $\alpha = \beta = 1$. The dataset and the setting were those that we used for the linear loss's training, which we described in the Section 6.1. Again, we used $\delta = 5 \times 10^{-3}$. We refer to Appendix E.2 for further experimental details and results.

[6] In the unregularised case $\alpha = 0$, the matrix inversion Θ^{-1}, necessary to compute h_T, can be source of significant computational errors. Introducing a factor $\alpha > 0$ makes the algorithm more stable.

6.3 Cross-entropy loss

For a general classification task, which can possibly have more than two labels, when $f_h = \operatorname{argmax}_i F_h^i$ a common optimisation objective is the cross-entropy loss

$$\hat{\ell}(F, y) = \log \sum_i e^{F^i} - F^y \,.$$

In this case we do not have simple analytic dynamics, and we will need to rely on some numerical solution. We have

$$\partial_t h_t^{ij} = -\frac{1}{m} \sum_{x \in s} \left(\frac{e^{(h_t \Phi(x))^i}}{\sum_k e^{(h_t \Phi(x))^k}} - \delta^{i f^\star(x)} \right) \Phi^j(x) \,,$$

where $\delta^{i f^\star(x)}$ denotes the Kronecker delta between the labels i and $f^\star(x)$, and

$$\Delta \hat{\ell} = \|\Phi\|^2 \left(1 - \frac{\sum_k e^{2F^k}}{(\sum_k e^{F^k})^2} \right) .$$

Contrarily to what happened with the quadratic and linear losses, where the objective's Laplacian stayed constant during the training, here we are in the interesting situation where the Laplacian tends to decrease during the training. Indeed, for $\hat{\ell}(F, y)$ to be small, we need F^y to be much larger than the other components of F, so that $\sum_k e^{F^k} \simeq e^{F^y}$. But this also means that the Laplacian is small, since $(\sum_k e^{2F^k})/(\sum_k e^{F^k})^2 \simeq 1$. This is a different behaviour from what happens when the algorithm is converging to a local minimum of the objective, as as now we can expect that the Lapacian's integral will have a sub-linear behaviour in T. Indeed, in the present situation, we are not approaching a local optimum. Conversely, h_t is slowly diverging, reaching flatter and flatter regions of $\mathcal{H}$.

We trained linear models of different sizes on MNIST, using the cross-entropy loss as objective. Since the dynamics does not admit a simple analytic solution, we performed the classical gradient-descent algorithm on batches of the training dataset, with finite time-steps. As discussed in Section 5.4, what we obtain in this way are not rigorous generalisation bounds, as we are not following the continuous dynamics. However, we believe that the result that we get are a good approximation of the ones that one would obtain with the exact solution of the gradient flow (see Figure 2 and the discussion in Section 5.4). The feature mapping Φ that we used was again the composition of a ReLU function and a linear transformation $x \mapsto Wx$, where the components of W were independently sampled from a normal distribution, with variance $1/784$ (784 being the dimension of the input). We refer to Appendix E.3 for further experimental details.

Table 1: Bounds on MNIST

Dimension N	Bound (2)	Test Error
10^3	.3251	.2049
3×10^3	.3034	.1967
10^4	.2786	.1680
3×10^4	.2795	.1541
10^5	.2751	.1602
3×10^5	.2812	.1534
10^6	.2757	.1568

Table 1 compares the bound (2) with the test error evaluated on a held-out dataset, for different dimensions of the hypothesis space $\mathcal{H}$, and always with $\delta = 5 \times 10^{-3}$. We chose the best bound among those obtained with 25 different values of T, at the price of adding a penalty factor $\log 25$ (see Section 5.2). As we had already stressed in Figure 1 (which refers to the same experiment), note that the bounds are not negatively affected by the high over-parameterisation, essentially because the optimal time horizon T decreases roughly as $1/N$. The tightest bound was achieved for a network of width 10000 (*i.e.*, $N = 10^5$). For this model, Figure 5 shows the evolution of the bounds (1) and (2) during the training. The values of the empirical error and of the penalty factor $(\log \frac{\rho_0(h_0)}{\rho_0(h_T)} + \log \frac{25\sqrt{m}}{\delta} + \int_0^T \Delta \mathcal{C}_s(h_t) \mathrm{d}t)/m$ are also reported. Figures 6 compares the contributions of the Laplacian's integral and the log-density term, showing that in this case it is the latter that dominates.

As a final remark, we notice that the bounds of Table 1 are comparable to those reported by Clerico et al. (2021) for a stochastic single-hidden-layer network. Notably, the results therein were obtained by using a PAC-Bayesian bound as optimisation objective, while here we are directly optimising the empirical cross-entropy loss, without any regularisation term.

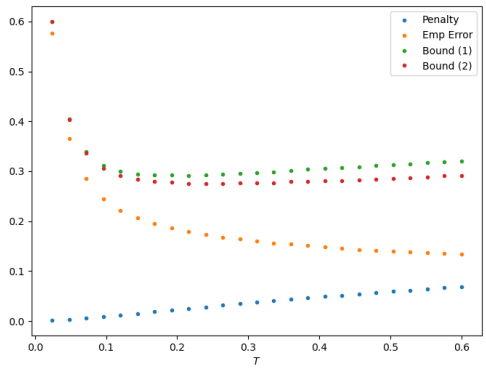

Figure 5: Behaviour of the bounds (1) and (2) at different time horizons T, for a linear model with 10^5 parameters trained with the cross-entropy loss on MNIST. The training dataset counted $m = 60000$ datapoints and a held-out test dataset was used for the test errors reported. The plot also shows the values of the penalty factor $(\log \frac{\rho_0(h_0)}{\rho_0(h_T)} + \log \frac{25\sqrt{m}}{\delta} + \int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t)/m$, where $\delta = 5 \times 10^{-3}$. See Appendix E.3 for more experimental details.

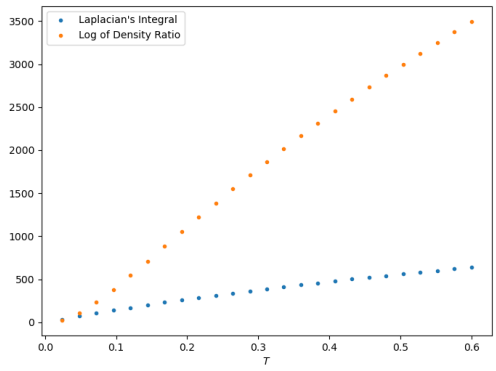

Figure 6: For the same experiment illustrated in Figure 5, comparison of the log density term $\log \frac{\rho_0(h_0)}{\rho_0(h_T)}$ and the Laplacian's integral $\int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t)/m$. The former clearly dominates on the latter. Note that the contribution of $\log \frac{25\sqrt{m}}{\delta} \simeq 14.02$ to the bound is negligible.

7 Conclusion

We established disintegrated PAC-Bayesian bounds for deterministic algorithms, holding with high probability on the random training dataset and initialisation. We tested our findings empirically on linear models, obtaining promising results that are non-vacuous even for highly over-parameterised settings. It is worth noticing that most of empirical PAC-Bayesian results in the literature are derived by using a PAC-Bayesian bound as the optimisation objective (Dziugaite and Roy, 2017; Pérez-Ortiz et al., 2021; Clerico et al., 2021, 2022a), while our bounds directly apply to the standard (continuous-time) empirical risk optimisation. Yet, our results on MNIST closely match the bounds obtained by Clerico et al. (2021). We defer to future work a more extensive empirical analysis, focusing on data-dependent *priors* and non-linear models.

A main limitation of our framework is the necessity of continuous-time dynamics. However, it might be possible to obtain rigorous results for the standard discretised gradient-descent optimisation via a careful numerical analysis. Alternatively, a scenario which might be worth exploring is the limit of infinite width, where the NTK dynamics (Jacot et al., 2018) might provide interesting continuous-time exact results. Finally, we mention as possible further developments a deeper study of stochastic training dynamics, where the gradient flow's continuity relation is replaced by the Fokker-Planck equation, and the case of a training objective that depends on the bound itself.

Acknowledgments: This work has been partially supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through the grants EP/R513295/1 (DTP scheme) and CoSInES EP/R034710/1. The authors would like to thank Umut Şimşekli, Tyler Farghly, and Patrick Rebeschini for the valuable comments and suggestions.

References

P. Alquier. User-friendly introduction to PAC-Bayes bounds. *arXiv:2110.11216*, 2021.

A. Banerjee, T. Chen, and Y. Zhou. De-randomized PAC-Bayes margin bounds: Applications to non-convex and non-smooth predictors. *arXiv:2002.09956*, 2020.

D.G.T. Barrett and B. Dherin. Implicit gradient regularization. *ICLR*, 2021.

F. Biggs and B. Guedj. Differentiable PAC-Bayes objectives with partially aggregated neural networks. *Entropy*, 23(10), 2021.

F. Biggs and B. Guedj. On margins and derandomisation in PAC-Bayes. *AISTATS*, 2022.

G. Blanchard and F. Fleuret. Occam's hammer. *COLT*, 2007.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2, 2002.

O. Bousquet, S. Boucheron, and G. Lugosi. *Introduction to Statistical Learning Theory.* Springer, 2004.

L. Bégin, P. Germain, F. Laviolette, and J.F. Roy. PAC-Bayesian bounds based on the Rényi divergence. *AISTATS*, 2016.

O. Catoni. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *IMS Lecture Notes Monograph Series*, 2007.

R.T.Q. Chen, Y. Rubanova, J. Bettencourt, and D.K. Duvenaud. Neural ordinary differential equations. *NeurIPS*, 2018.

E. Clerico, G. Deligiannidis, and A. Doucet. Wide stochastic networks: Gaussian limit and PAC-Bayesian training. *arxiv:2106.09798*, 2021.

E. Clerico, G. Deligiannidis, and A. Doucet. Conditionally Gaussian PAC-Bayes. *AISTATS*, 2022a.

E. Clerico, A. Shidani, G. Deligiannidis, and A. Doucet. Chained generalisation bounds. *COLT*, 2022b.

L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. *ICML*, 2017.

C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9, 2014.

G.K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *UAI*, 2017.

P. Germain, A. Lacasse, F. Laviolette, and M. Marchand. PAC-Bayesian learning of linear classifiers. *ICML*, 2009.

B. Ghorbani, S. Krishnan, and Y. Xiao. An investigation into neural net optimization via Hessian eigenvalue density. *ICML*, 2019.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

B. Guedj. A primer on PAC-Bayesian learning. *Proceedings of the Second Congress of the French Mathematical Society*, 2019.

E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration.* Springer-Verlag, 2006.

M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. *ICML*, 2016.

H. He, G. Huang, and Y. Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *NeurIPS*, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, 2015.

S. Hochreiter and J. Schmidhube. Flat minima. *Neural Computations*, 9, 1997.

P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A.G. Wilson. Averaging weights leads to wider optima and better generalization. *UAI*, 2018.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS*, 2018.

N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P.T.P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *ICLR*, 2017.

J. Langford and M. Seeger. Bounds for averaging classifiers. *CMU technical report*, 2001.

J. Langford and J. Shawe-Taylor. PAC-Bayes & margins. *NeurIPS*, 2002.

G. Letarte, P. Germain, B. Guedj, and F. Laviolette. Dichotomize and generalize: PAC-Bayesian binary activated deep neural networks. *NeurIPS*, 2019.

G. Lugosi and G. Neu. Generalization bounds via convex analysis. *COLT*, 2022.

A. Maurer. A note on the PAC Bayesian theorem. *arXiv:0411099*, 2004.

D.A. McAllester. Some PAC-Bayesian theorems. *COLT*, 1998.

D.A. McAllester. PAC-Bayesian model averaging. *COLT*, 1999.

K. Miyaguchi. PAC-Bayesian transportation bound. *arXiv:1905.13435*, 2019.

V. Nagarajan and J.Z. Kolter. Deterministic PAC-Bayesian generalization bounds for deep networks via generalizing noise-resilience. *ICLR*, 2019.

G. Neu, G.K. Dziugaite, M. Haghifam, and D. M. Roy. Information-theoretic generalization bounds for stochastic gradient descent. *COLT*, 2021.

B. Neyshabur, S. Bhojanapalli, D.A. McAllester, and N. Srebro. Exploring generalization in deep learning. *NeurIPS*, 2017.

B. Neyshabur, S. Bhojanapalli, and N. Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *ICLR*, 2018.

M. Pérez-Ortiz, O. Risvaplata, J. Shawe-Taylor, and C. Szepesvári. Tighter risk certificates for neural networks. *Journal of Machine Learning Research*, 22, 2021.

O. Rivasplata, I. Kuzborskij, C. Szepesvári, and J. Shawe-Taylor. PAC-Bayes analysis beyond the usual bounds. *NeurIPS*, 2020.

L. Sagun, U. Evci, V.U. Güney, Y. Dauphin, and L. Bottou. Empirical analysis of the Hessian of over-parametrized neural networks. *ICLR Workshop*, 2018.

M. Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 3, 2002.

S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv:2011.13456*, 2020.

V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 2000.

P. Viallard, P. Germain, A. Habrard, and E. Morvant. A general framework for the disintegration of PAC-Bayesian bounds. *arXiv:2102.08649*, 2021.

A. Xu and M. Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *NeurIPS*, 2017.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *Communications of the ACM*, 64, 11 2016.

W. Zhou, V. Veitch, M. Austern, R.P. Adams, and P. Orbanz. Non-vacuous generalization bounds at the ImageNet scale: a PAC-Bayesian compression approach. *ICLR*, 2019.

Appendix

A Langevin dynamics

We briefly discuss what happens if we try to use a Langevin dynamics instead of a deterministic gradient method, that is

$$\mathrm{d}h_t = -\nabla \mathcal{C}_s(h_t)\mathrm{d}t + \sigma \mathrm{d}B_t \,. \tag{8}$$

Here, $\sigma > 0$ is a diffusion coefficient and B_t denotes the standard Brownian motion. Note that the introduction of noise prevents the bound from diverging at large T. Indeed, under suitable regularity conditions for $\mathcal{C}_s$, ρ_t will converge to the Gibbs equilibrium distribution

$$\rho_\infty(h) = \frac{1}{Z} e^{-2\mathcal{C}_s(h)/\sigma^2} \,,$$

with Z the normalising constant. As ρ_∞ is bounded by $1/Z$, asymptotically $\log \frac{\rho_T(h_T)}{\rho_0(h_T)} \lesssim -\log(Z\rho_0(h_T))$ does not diverge with T.

For (9), the Fokker-Planck equation yields

$$\partial_t \rho_t(h) = \nabla(\rho_t(h)\nabla \mathcal{C}_s(h)) + \frac{\sigma^2}{2}\Delta\rho_t(h) = \nabla\rho_t(h)\cdot\nabla\mathcal{C}_s(h) + \rho_t(h)\Delta\mathcal{C}_s(h) + \frac{\sigma^2}{2}\Delta\rho_t(h) \,. \tag{9}$$

Applying Itō's lemma we get

$$\mathrm{d}(\rho_t(h_t)) = (\rho_t(h_t)\Delta\mathcal{C}_s(h_t) + \sigma^2\Delta\rho_t(h_t))\mathrm{d}t + \sigma\nabla\rho_t(h_t)\cdot \mathrm{d}B_t \,.$$

Now, writing $u_t = \log\rho_t$, we have

$$\mathrm{d}u_t(h_t) = \left(\Delta\mathcal{C}_s(h_t) + \frac{\sigma^2}{2}\|\nabla u_t(h_t)\|^2 + \sigma^2\Delta u_t(h_t)\right)\mathrm{d}t + \sigma\nabla u_t(h_t)\cdot \mathrm{d}B_t \,, \tag{10}$$

where we used $\frac{\Delta\rho_t}{\rho_t} = \|\nabla u_t\|^2 + \Delta u_t$. This expression cannot easily be integrated in t, as it happened for the deterministic dynamics, since now we have a dependence on u_t (and hence on ρ_t) in the RHS. In order to obtain a computable bound, one would probably need to upper-bound the intractable quantities in the RHS, an approach that we defer to future work.

An alternative way to tackle the stochastic dynamics consists in converting the SDE (8) in an ODE inducing the same marginal distributions (an idea already exploited in Song et al. (2020)). In fact, consider the deterministic flow

$$\partial_t \hat{h}_t = -\nabla\left(\mathcal{C}_s(\hat{h}_t) + \frac{\sigma^2}{2}\log\hat{\rho}_t(\hat{h}_t)\right) \,, \tag{11}$$

where we denote as $\hat{\rho}_t$ the density of $\hat{h}_t$ and we impose $\hat{\rho}_0 = \rho_0$. The continuity equation now reads

$$\partial_t\hat{\rho}_t(h) = \nabla\cdot\left(\hat{\rho}_t(h)\,\nabla\left(\mathcal{C}_s(h) + \frac{\sigma^2}{2}\log\hat{\rho}_t(h)\right)\right) = \nabla(\hat{\rho}_t(h)\nabla\mathcal{C}_s(h)) + \frac{\sigma^2}{2}\Delta\hat{\rho}_t(h) \,. \tag{12}$$

Comparing (12) with (9), we immediately see that ρ_t and $\hat{\rho}_t$ undergo the same dynamics and, since $\rho_0 = \hat{\rho}_0$, we have that $\rho_t = \hat{\rho}_t$ for all t. In practice, this means that we have two equivalent ways to get $h_T \sim \rho_T$: given an initial point from ρ_0, we can follow either the stochastic dynamics (8), or the deterministic ones (11), up to time T. Moreover, in order to evaluate the factor $\log\rho_T(h_T)$, needed for our PAC-Bayesian bound, instead of using (10) we can leverage the continuity equation of the deterministic flow to derive

$$\log\rho_T(h_T) = \log\rho_0(\hat{h}_0) + \int_0^T \Delta\left(\mathcal{C}_s(\hat{h}_t) + \frac{\sigma^2}{2}\log\rho_t(\hat{h}_t)\right)\mathrm{d}t \,,$$

where $\hat{h}_t$ denotes the solution of (11) satisfying $\hat{h}_T = h_T$. Although this expression looks easier than (10), its actual computation is typically unfeasible in practice, due to the RHS's dependence on $\log\rho_t$ (both explicit and implicit, through $\hat{h}_t$). In the generative modelling context, score matching and neural networks techniques are used to approximate $\nabla\log\rho_t$, but this introduces approximation errors difficult to quantify (Song et al., 2020).

As a final remark, we mention that the dynamics (11) also appear when trying to build an algorithm that directly optimises a PAC-Bayesian bound. Indeed, for $\ell \subseteq [0,1]$, the choice $\Psi(u,v) = \frac{2}{\sigma^2}(u-v)$ (for an arbitrary $\sigma > 0$) in Proposition 1 leads to a bound in the form

$$\mathcal{L}_{\mathcal{X}}(h_T) \leq \mathcal{L}_s(h_T) + \frac{1}{4m\sigma^2} + \frac{\sigma^2}{2}\left(\log\frac{\rho_T(h_T)}{\rho_0(h_T)} + \log\frac{1}{\delta}\right).$$

Using this bound as optimisation objective is equivalent to follow the dynamics (11), provided that we set $\mathcal{C}_s(h) = \mathcal{L}_s(h) + \frac{1}{4m\sigma^2} + \frac{\sigma^2}{2}(-\log\rho_0(h) + \log\frac{1}{\delta})$. We defer to future work a deeper analysis of this connection.

B Rewriting the Laplacian's integral

We explicitly derive here (4). First, notice that $\partial_t \log\|\nabla\mathcal{C}_s(h_t)\| = -\frac{\nabla\mathcal{C}_s(h_t)}{\|\nabla\mathcal{C}_s(h_t)\|}\cdot\nabla\|\nabla\mathcal{C}_s(h_t)\|$. Since, for all h,

$$\Delta\mathcal{C}_s(h) = \nabla\cdot\nabla\mathcal{C}_s(h) = \nabla\cdot\left(\frac{\nabla\mathcal{C}_s(h)}{\|\nabla\mathcal{C}_s(h)\|}\right)\|\nabla\mathcal{C}_s(h)\| + \frac{\nabla\mathcal{C}_s(h)}{\|\nabla\mathcal{C}_s(h)\|}\cdot\nabla\|\nabla\mathcal{C}_s(h)\|,$$

we get

$$\Delta\mathcal{C}_s(h_t) = \nabla\cdot\left(\frac{\nabla\mathcal{C}_s(h_t)}{\|\nabla\mathcal{C}_s(h_t)\|}\right)\|\nabla\mathcal{C}_s(h_t)\| - \partial_t\log\|\nabla\mathcal{C}_s(h_t)\|.$$

We conclude that

$$\int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t = \log\frac{\|\nabla\mathcal{C}_s(h_0)\|}{\|\nabla\mathcal{C}_s(h_T)\|} + \int_0^T \nabla\cdot\left(\frac{\nabla\mathcal{C}_s(h_t)}{\|\nabla\mathcal{C}_s(h_t)\|}\right)\|\nabla\mathcal{C}_s(h_t)\|\mathrm{d}t.$$

The integral in the RHS is a line-integral along the path $h_{[0,T]}$, as $\|\nabla\mathcal{C}_s(h)\|$ is the norm of the flow's "velocity" in h. Moreover, $\tau(h) = -\frac{\nabla\mathcal{C}_s(h)}{\|\nabla\mathcal{C}_s(h)\|}$ is the unit tangent vector to the gradient flow in h. We can thus write

$$\int_0^T \Delta\mathcal{C}_s(h_t)\mathrm{d}t = \log\frac{\|\nabla\mathcal{C}_s(h_0)\|}{\|\nabla\mathcal{C}_s(h_T)\|} - \int_{h_{[0:T]}} \nabla\cdot\tau(h)\|\delta h\|,$$

which is (4).

C Asymptotic behaviour around a local minimum

We consider the case in which the algorithm is approaching a quadratic local optimum of $\mathcal{C}_s$. We will hence say that for t larger than some $\hat{t}$ we have

$$\mathcal{C}_s(h_t) \simeq \mathcal{C}_s(h^\star) + \frac{1}{2}(h_t - h^\star)^\top H(h_t - h^\star),$$

where H is a positive definite matrix, namely the Hessian of $\mathcal{C}_s$ evaluated at $h^\star$. Hence, for $t \geq \hat{t}$ the objective's Laplacian is roughly constant,

$$\Delta\mathcal{C}_s(h_t) \simeq \mathrm{Tr}[H].$$

For a large enough time horizon T, we can asymptotically write $\int_0^T \Delta\mathcal{C}_s(h_t) \simeq T\mathrm{Tr}[H]$. Moreover, since the Laplacian's integral is diverging while $\frac{\rho_0(h_0)}{\rho_0(h_T)} \to \frac{\rho_0(h_\star)}{\rho_0(h_T)}$, asymptotically we find that (1) takes the form

$$\mathcal{L}_{\mathcal{X}}(w_T) - \mathcal{L}_s(w_T) \lesssim O\left(\sqrt{\frac{T\mathrm{Tr}[H]}{2m}}\right).$$

D Data-dependent priors

As we discussed in Section 5.6, if we want to use a *prior* ρ_{t_0}, learnt on a dataset s' (independent of s), we need to evaluate $\log\rho_{t_0}(h_T)$. Define $\hat{h}_t$ as the solution of $\partial_t\hat{h}_t = -\nabla\mathcal{C}_{s'}(\hat{h}_t)$ satisfying $\hat{h}_T = h_T$. We have that

$$\log\rho_{t_0}(h_T) = \log\rho_{t_0}(\hat{h}_T) = \log\rho_0(\hat{h}_{T-t_0}) + \int_0^T \partial_t(\log\rho_t(\hat{h}_{T-t_0+t}))\mathrm{d}t.$$

Since, for $t \in [0, t_0]$, ρ_t is the push-forward of ρ_0 under the gradient dynamics with objective $\mathcal{C}_{s'}$, we have that for all h

$$\partial_t \rho_t(h) = \rho_t(h) \Delta \mathcal{C}_{s'}(h) + \nabla \rho_t(h) \cdot \nabla \mathcal{C}_{s'}(h)$$

and so in particular

$$\partial_t(\rho_t(\hat{h}_t)) = \partial_t \rho_t(\hat{h}_t) + \nabla \mathcal{C}_{s'}(\hat{h}_t) \cdot \partial_t \hat{h}_t = \rho_t(\hat{h}_t) \Delta \mathcal{C}_{s'}(\hat{h}_t)\,.$$

We conclude that

$$\log \rho_{t_0}(h_T) = \log \rho_0(\hat{h}_{T-t_0}) + \int_0^T \Delta \mathcal{C}_{s'}(\hat{h}_{T-t_0+t}) \mathrm{d}t\,,$$

which is (6). We can hence state the following variant of Proposition 2 for data-dependent *priors*.

Proposition 3. *Let s and s' be independent training dataset, with $s \sim \mu^m$. Let $\mathcal{C}_s$ and $\mathcal{C}_{s'}$ be two twice differentiable training objectives, fix $T > 0$ and $t_0 \in [0, T)$. Consider an algorithm that, for an initial state h_0, follows the dynamics $\partial_t h_t = -\nabla \mathcal{C}_{s'}(h_t)$ for $t \in [0, t_0]$, and $\partial_t h_t = -\nabla \mathcal{C}_s(h_t)$ for $t \in [t_0, T]$, and finally outputs h_T. Then, for any measurable function $\Psi : \mathbb{R}^2 \to \mathbb{R}$, with probability at least $1 - \delta$ on the random draw $(s, h_0) \sim \mu^m \otimes \rho_0$, we have*

$$\Psi(\mathcal{L}_s(h_T), \mathcal{L}_{\mathcal{X}}(h_T)) \leq \log \frac{\rho_0(h_{t_0})}{\rho_0(\hat{h}_{T-t_0})} + \int_0^{t_0} (\Delta \mathcal{C}_{s'}(h_t) - \Delta \mathcal{C}_{s'}(\hat{t}_{T-t_0+t}))\mathrm{d}t + \int_{t_0}^T \Delta \mathcal{C}_s(h_t)\mathrm{d}t + \log \frac{\xi}{\delta}\,,$$

where $\hat{h}_t$ is the solution of $\partial_t \hat{h}_t = -\nabla \mathcal{C}_{s'}(\hat{h}_t)$ with $\hat{h}_T = h_T$, and $\xi = \int_{\mathcal{X}^m \times \mathcal{H}} e^{\Psi(\mathcal{L}_s(h), \mathcal{L}_{\mathcal{X}}(h))} \mathrm{d}\mu^m(s) \mathrm{d}\rho_{t_0}(h)$.

E Experimental details

E.1 Linear loss

For our experiments with the linear surrogate loss (see Section 6.1), we used a binary toy dataset, which was created as follows. First, we sampled eight independent points in $\mathbb{R}^5$, from a standard normal distribution. These eight points were then used as the means of eight Gaussian distributions, symmetric and with variance 0.1. We sampled eight Gaussian clusters of 5000 points from each of this eight distributions. We then assigned the label -1 to four clusters, and $+1$ to the remaining four. Finally, all the points were projected on the unit sphere. 500 of them, randomly chosen, were used as training dataset, while the remaining 39500 constituted a held-out test dataset.

The linear model that we considered was a neural network with a single-hidden layer, where the hidden nodes were kept fixed at their initial values and only the outer layer was trained. Explicitly, we had $F(x) = h \cdot \Phi(x)$, with $\Phi(x) = \phi(Wx)$. Here, W denotes a random $5 \times N$ matrix, whose components were independently drawn from a centred normal distribution with variance $1/5$, and ϕ is the ReLU function $\phi(x) = \max(0, x)$. We chose the distribution ρ_0 as a multivariate normal distribution, so that all the components of h_0 were independent draws from a Gaussian random variable with variance $1/N$.

We tried different values for the dimension N of h: 100, 1000, and 10000. We reported in Figure 3 in the main text the results obtained for $N = 10000$. The parameter δ was fixed at .005, so the bounds hold with probability higher than .995. Since the values reported are already comprehensive of the factor $\log 50$ due to the union bound, it is possible to select the horizon time T yielding the tightest result. Figure 7 compares the values of (2) for the different sizes N.

E.2 Quadratic loss

For the experiments with the quadratic surrogate loss (Section 6.2), we used the same binary dataset and models that we described in Appendix E.1. The parameters α and β that define $\mathcal{C}_s$ were both set to 1. Figure 4 in the main text reports the results obtained for $N = 10000$. The value reported are already taking into account the $\log 50$ factor (due to the union bound), and δ was again chosen to be .005. Figure 8 compares the values of the bound (2) for different dimensions N.

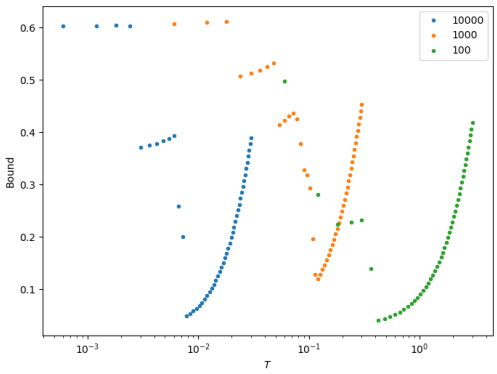

Figure 7: For the experiment with the linear surrogate loss, comparison of the evolution of the bound (2) for different sizes N of the the model. See Section 6.1 and Appendix E.1 for more details.

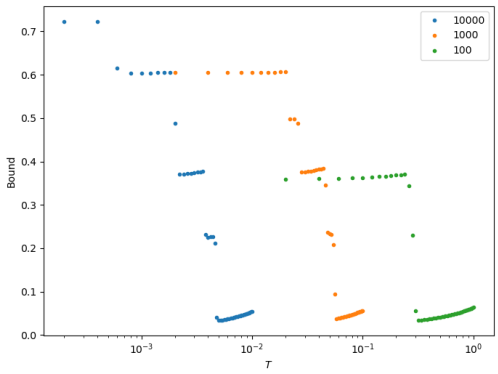

Figure 8: For the experiment with the quadratic surrogate loss, comparison of the evolution of the bound (2) for different sizes N of the the model. See Section 6.2 and Appendix E.2 for more details.

E.3 Cross-entropy loss

We performed our experiments with the cross-entropy as surrogate loss (see Section 6.3) on the MNIST dataset, with the standard split of 60000 images used for the training and 10000 for the testing of the model. The training was performed on batches of 500 elements. The images were flattened to one-dimensional arrays and renormalised. We used again linear architectures $F(x) = h\Phi(x)$, but now with h being a $10 \times w$ matrix (w denoting the width of the network). Φ was always in the form $\Phi(x) = \phi(Wx)$. Here ϕ acts component-wise as the ReLU function, and W is a $w \times 784$ matrix (784 being the dimension of the input datapoints), whose components were independently drawn from a centred normal distribution with variance $1/784$. The dimension of the hypothesis h was $N = 10w$, as the networks had 10 output nodes. We tested different values of N, as reported in Table 1. In all the bounds $\delta = .005$.

Since closed-form solutions for the flow are not available, we used discretised gradient dynamics. For each batch s_k we made an update $h_{t+\delta_t} = h_t + \delta_t \nabla \mathcal{C}_{s_k}(h_t)$, for some small δ_t. Recall that, when training by batches, the Laplacian's integral takes the form (5). Here we chose that $t_k = k\delta_t$, so that every time we switched to a new batch after a single time-step. In particular, we get $\int_{t_{k-1}}^{t_k} \Delta\mathcal{C}_{s_k}(h_t)\mathrm{d}t = \Delta\mathcal{C}_{s_k}(h_{t-1})\delta_t$ in (5). Table 2 reports the different values of time discretisations δ_t, used for the models that we trained. Figure 2 in the main text shows the relative error between the values of (2) obtained for the same model ($N = 10^5$) for two different time discretisations ($\delta_{t,1} = 2 \times 10^{-4}$ and $\delta_{t,2} = 2 \times 10^{-6}$). In both cases we trained for a time-interval of 2×10^{-4} before switching batch. This means that for $\delta_{t,1}$ each batch was used for a single time-step, before switching to the next one, while for $\delta_{t,2}$ 100 gradient-steps were performed on each batch before passing to the next one.

Table 2: Time discretisation

Dimension N	Final Time	δ_t
10^3	60	2.00×10^{-2}
3×10^3	20	6.67×10^{-3}
10^4	6	2.00×10^{-3}
3×10^4	2	6.67×10^{-4}
10^5	.6	2.00×10^{-4}
3×10^5	.2	6.67×10^{-5}
10^6	.06	2.00×10^{-5}

The values of (2) that we obtained for the different dimensions N are reported in Table 1, in the main text. The penalty $\log 25$ is already taken into account for the values reported, as for each width the best bound has been chosen among the values obtained with 25 different time horizons. The optimal T reported in Figure 1 are the values of the best time horizon (among the 25 ones that we tried) that we obtained for each model's width. Figures 5 and 6 refer to the model with size $N = 10^5$.