Mathematics of Deep Learning
Lecture 7: Implicit bias of descent algorithms

Bruno Loureiro
Département d'Informatique, École Normale Supérieure - PSL & CNRS, France

28/02/2025

Typos, comments or suggestions? Get in touch at: bruno.loureiro@di.ens.fr

1 Motivation

So far, our discussion has focused mostly on the approximation and estimation properties of statistical models. However, one central ingredient in modern machine learning has been missing from our discussion: the training algorithm.

For strictly convex problems (such as the ridge regression with $\lambda > 0$), the training algorithm indeed plays a minor role: the minimiser is unique, and any "good enough" algorithm should convergence to it — with the only difference being the computational efficiency. The situation, however, is very different for non-convex problems where more than one minima can be present. In this case, different choices of algorithm (including here initialisation and choices of hyperparameter, e.g. learning rate schedule, mini-batch sampling, stopping time, etc.) can lead to different estimators with potentially drastic differences in the generalisation performance, see e.g. (Liu et al., 2020). This can be particularly striking in problems with more than a global minima, for instance in overparametrised networks which can be trained down to achieve zero training loss (i.e. perfectly interpolate the training data). In this case, different algorithms might converge to different interpolators, all achieving zero loss, but which can have different generalisation performances — see fig. 1 for an illustration.

The fact that different choices of algorithm lead to predictors with different statistical properties is known as the *implicit bias* of algorithms.[1] Characterising the implicit biases of widely used algorithms such as gradient descent and stochastic gradient descent is an active research field. In this lecture, we will study two of the simplest examples.

The discussion that follows was greatly inspired by a post in the blog of Francis Bach, written by Pillaud-Vivien and Pesme (2022), as well as Scott Pesme's PhD manuscript.

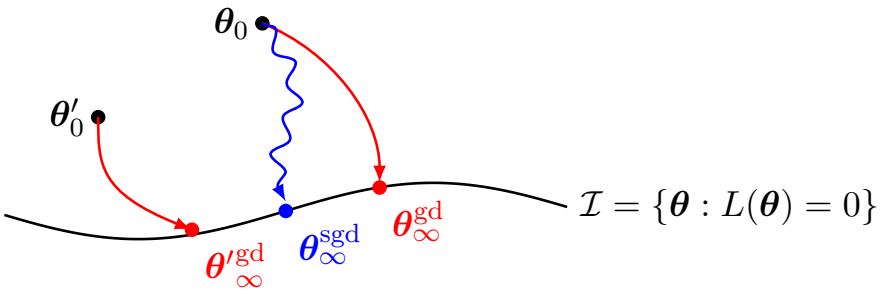

Figure 1

[1] Perhaps worryingly, the naming suggests that these are biases which often get forgotten.

2 Implicit bias in least-squares regression

As motivated above, algorithmic bias is mostly relevant in problems for which the loss has more than a single critical point. Arguably the simplest such problem is least-squares regression in the overspecified regime ($d > n$), which we now review.

2.1 Recap of OLS

Consider a supervised learning regression problem with training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^{d+1} : i \in [n]\}$. Assume for simplicity that the data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is full-rank. The lest-squares problem is defined as:

$$\min_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta}) \coloneqq \frac{1}{2n} \sum_{i=1}^{n} (y_i - \langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle)^2 \tag{2.1}$$

The Hessian of the empirical risk is simply the empirical covariance of the data:

$$\nabla^2 \hat{R}_n(\boldsymbol{\theta}) = \frac{1}{n} \boldsymbol{X}^\top \boldsymbol{X} =: \hat{\boldsymbol{\Sigma}}_n \tag{2.2}$$

which is positive semi-definite. This implies that $\hat{R}_n$ is a convex function of $\boldsymbol{\theta} \in \mathbb{R}^d$. However, it is not necessarily strictly convex: this is only the case for $n \geq d$ for which $\boldsymbol{X}^\top \boldsymbol{X}$ is positive-definite. For $n < d$, the empirical risk is just convex, meaning it can have more than one global minimum. How do global minimum look like?

Overdetermined regime ($n \geq d$) — The empirical risk is non-negative $\hat{R}_n(\boldsymbol{\theta}) \geq 0$, with equality $\hat{R}_n(\boldsymbol{\theta}) = 0$ for a predictor that perfectly interpolates the training data $\boldsymbol{X}\hat{\boldsymbol{\theta}} = \boldsymbol{y}$. Finding an interpolator is equivalent to solving a system of n equations with d independent variables (since we assumed $\boldsymbol{X}$ full rank). In particular, this is not possible when $n > d$: there are more equations than variables, and the system is *overdetermined*. Nevertheless, the unique solution to the OLS problem in eq. (2.1) is given by:

$$\hat{\boldsymbol{\theta}}_{\text{ols}}(\boldsymbol{X}, \boldsymbol{y}) = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top y, \qquad n \geq d \tag{2.3}$$

which, for $n > d$ has strictly positive training error $\hat{R}_n(\hat{\boldsymbol{\theta}}_{\text{ols}}) > 0$. For $n = d$, $\boldsymbol{X}$ becomes invertible and the unique interpolator given by $\hat{\boldsymbol{\theta}}_{\text{ols}}(\boldsymbol{X}, \boldsymbol{y}) = \boldsymbol{X}^{-1} \boldsymbol{y}$.

Overdetermined regime ($n < d$) — It is easy to see that for $n < d$ the solution:

$$\hat{\boldsymbol{\theta}}_{\text{ols}}(\boldsymbol{X}, \boldsymbol{y}) = \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{X})^{-1} \boldsymbol{y}, \qquad n \leq d \tag{2.4}$$

is an interpolator. Note it is precisely the limit of the ridge regressor when we take $\lambda \to 0^+$. However, this interpolator is not unique. Indeed, for any vector $\boldsymbol{v} \in \text{Ker}(\boldsymbol{X})$,[2] the sum $\hat{\boldsymbol{\theta}}_{\text{ols}} + \boldsymbol{v}$ is also an interpolator, since by definition $\boldsymbol{X}\boldsymbol{v} = \boldsymbol{0}$. This means that the space of interpolators define an affine space, which can be explicitly written as:

$$\begin{aligned} \mathcal{I} &= \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^d : \boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{y}\} = \{\boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \boldsymbol{y} + \boldsymbol{v} : \boldsymbol{v} \in \text{Ker}(\boldsymbol{X})\} \\ &= \hat{\boldsymbol{\theta}}_{\text{ols}} + \text{Ker}(\boldsymbol{X}) \end{aligned} \tag{2.5}$$

The interpolator $\hat{\boldsymbol{\theta}}_{\text{ols}}$ is also known as the *minimum-ℓ_2 norm solution*, since by the triangular inequality:

$$||\hat{\boldsymbol{\theta}}_{\text{ols}} + \boldsymbol{v}||_2^2 \geq ||\hat{\boldsymbol{\theta}}_{\text{ols}}||_2^2 + ||\boldsymbol{v}||_2^2 \tag{2.6}$$

[2] Recall that for $n < d$, $\text{Ker}(\boldsymbol{X}) \neq \emptyset$ and that it is isomorphic to a $d - n$ space.

which implies it has the smallest Euclidean norm of all elements of $\mathcal{I}$. In other words, it is the solution of:

$$\min ||\boldsymbol{\theta}||_2^2, \text{ such that } \boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{y} \tag{2.7}$$

In learning theory, it is common to use norms $||\cdot||_p$ as a proxy for the complexity of a hypothesis class, and a common objective is to find predictors with low complexity, which are often associated with better generalisation. From this perspective, the minimum-ℓ_2 norm predictor $\hat{\boldsymbol{\theta}}_{\text{ols}}$ is the least complex interpolator with respect to $||\cdot||_2$.

⚠ We don't mean that the minimum-ℓ_2 norm predictor is the one with best generalisation in $\mathcal{I}$. Indeed, whether $\hat{\boldsymbol{\theta}}_{\text{ols}}$ generalises better than, e.g. the minimum-ℓ_1 predictor, will crucially depends on the target function $f_\star(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{x}]$.

Now consider minimising eq. (2.1) in the underdetermined regime $n < d$ using a descent-based algorithm, such as gradient descent. To which interpolator in $\mathcal{I}$ will it converge to? We start by answering this question in the context of gradient flow.

2.2 Implicit bias of gradient flow

Consider the gradient flow algorithm for the OLS problem:

$$\dot{\boldsymbol{\theta}}(t) = -\nabla \hat{R}_n(\boldsymbol{\theta}) = \frac{1}{n}\boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \tag{2.8}$$

with initial condition $\boldsymbol{\theta}(t) = \boldsymbol{\theta}_0$. This is a system of d coupled ODEs. To decouple them, let $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$ denote the SDV of $\boldsymbol{X}$:

$$\dot{\boldsymbol{\theta}}(t) = \frac{1}{n}\boldsymbol{V}\left(\boldsymbol{D}^\top\boldsymbol{U}^\top\boldsymbol{y} - \boldsymbol{D}^\top\boldsymbol{D}\boldsymbol{V}^\top\boldsymbol{\theta}(t)\right) \tag{2.9}$$

Hence, defining $\boldsymbol{\beta} = \boldsymbol{V}^\top\boldsymbol{\theta}(t)$ and $\tilde{\boldsymbol{y}} = \boldsymbol{U}^\top\boldsymbol{y}$, we have an autonomous system:

$$\dot{\beta}_j(t) = \frac{\sigma_j}{n}(\tilde{y}_j - \sigma_j\beta_j) \tag{2.10}$$

Note that, depending on $\text{rank}(\boldsymbol{X}) = \min(n, d)$, the equation above take a different shape.

Overspecified regime ($n \geq d$) — For $n \geq d$, $\text{rank}(\boldsymbol{X}) = d$ and hence $\sigma_j > 0$ for all $j \in [d]$. The solution is therefore given by:

$$\beta_j(t) = \frac{\tilde{y}_j}{\sigma_j} + e^{-\frac{\sigma_j^2 t}{n}}\left(\beta_{0,j} - \frac{\tilde{y}_j}{\sigma_j}\right) \tag{2.11}$$

where $\boldsymbol{\beta}_0 = \boldsymbol{V}^\top\boldsymbol{\theta}_0$. Or, in terms of $\boldsymbol{\theta}$:

$$\begin{aligned}\boldsymbol{\theta}(t) &= \sum_{j=1}^{d}\left[\frac{\langle \boldsymbol{u}_j, \boldsymbol{y}\rangle}{\sigma_j}\boldsymbol{v}_j + e^{-\frac{\sigma_j^2 t}{n}}\left(\boldsymbol{\theta}_0 - \frac{\langle \boldsymbol{u}_j, \boldsymbol{y}\rangle}{\sigma_j}\boldsymbol{v}_j\right)\right] \\ &= (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} + \sum_{j=1}^{d} e^{-\frac{\sigma_j^2 t}{n}}\left(\boldsymbol{\theta}_0 - \frac{\langle \boldsymbol{u}_j, \boldsymbol{y}\rangle}{\sigma_j}\boldsymbol{v}_j\right)\end{aligned} \tag{2.12}$$

As expected, this converges exponentially fast to $\hat{\boldsymbol{\theta}}_{\text{ols}}$ in eq. (2.3) — no surprise, this is the unique global minimum since the problem is strictly convex in this regime.

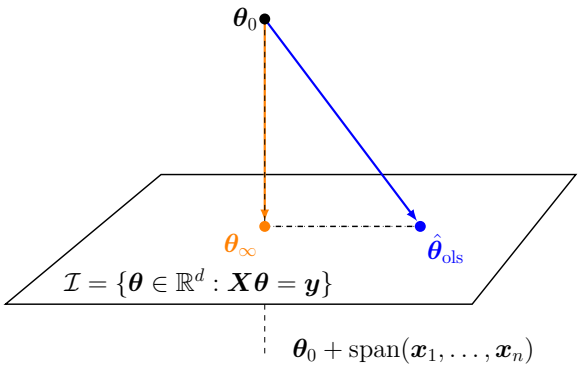

Figure 2: Implicit bias of gradient flow for in the underdetermined regime $n < d$. Gradient flow converges to the orthogonal projection of the initial condition on the linear subspace of interpolators.

Underspecified regime ($n < d$) — In this case, $\text{rank}(\boldsymbol{X}) = n$. Assuming that $\sigma_j \geq 0$ is arranged in non-increasing order, we have $\sigma_j = 0$ for all $j > n$. This means that the solution of eq. (2.10) is now given by:

$$\beta_j(t) = \begin{cases} \frac{\tilde{y}_j}{\sigma_j} + e^{-\frac{\sigma_j^2 t}{n}} \left(\beta_{0,j} - \frac{\tilde{y}_j}{\sigma_j}\right) & \text{for } 1 \leq j \leq n \\ \beta_{0,j} & \text{for } n < j \leq d \end{cases} \tag{2.13}$$

which means that only the components of the initial condition which are in $\text{span}(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n) \simeq \mathbb{R}^n$ change under the gradient flow, with the remaining components remaining constant. More precisely, recall that we can always decompose $\mathbb{R}^d = \text{range}(\boldsymbol{X}^\top) \oplus \ker(\boldsymbol{X})$, which induces the following orthogonal decomposition of the initial condition $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{0,\parallel} + \boldsymbol{\theta}_{0,\perp}$ with $\boldsymbol{\theta}_{0,\parallel} \in \text{range}(\boldsymbol{X}^\top)$ and $\boldsymbol{\theta}_{0,\perp} \in \ker(\boldsymbol{X})$. Therefore, we can write:

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}_{0,\perp} + \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{y} + \sum_{j=1}^{d} e^{-\frac{\sigma_j^2 t}{n}} \left(\boldsymbol{\theta}_{0,\parallel} - \frac{\langle \boldsymbol{u}_j, \boldsymbol{y}\rangle}{\sigma_j}\boldsymbol{v}_j\right) \tag{2.14}$$

In the long-time limit, this leads to:

$$\begin{aligned} \boldsymbol{\theta}_\infty \coloneqq \lim_{t\to\infty} \boldsymbol{\theta}(t) &= \boldsymbol{\theta}_{0,\perp} + \boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{y} \\ &= \boldsymbol{\theta}_{0,\perp} + \hat{\boldsymbol{\theta}}_{\text{ols}}(\boldsymbol{X}, \boldsymbol{y}) \end{aligned} \tag{2.15}$$

This corresponds to a particular interpolator $\mathcal{I}$: the one which is closest (in Euclidean distance) to the initial condition $\boldsymbol{\theta}_0$:

$$\boldsymbol{\theta}_\infty = \underset{\boldsymbol{\theta}\in\mathcal{I}}{\arg\min}\ ||\boldsymbol{\theta}_0 - \boldsymbol{\theta}||_2^2 \tag{2.16}$$

which is only equal to $\hat{\boldsymbol{\theta}}_{\text{ols}}$ if $\boldsymbol{\theta}_0 = \boldsymbol{0}$! See Figure 2 for an illustration.

2.3 Other descent algorithms

A natural question is whether gradient descent (the discretisation of gradient flow), or other descent-based algorithms, such as SGD, have different implicit biases than the one discussed above. Consider mini-batch SGD:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \nabla \hat{R}_{b_k}(\boldsymbol{\theta}_k), \quad \text{with} \quad \hat{R}_{b_k}(\boldsymbol{\theta}) = \frac{1}{2b}\sum_{i\in b_k}(y_i - \langle \boldsymbol{\theta}, \boldsymbol{x}_i\rangle)^2 \tag{2.17}$$

where η_k is a learning rate schedule and $b_k \subset [n]$ is a mini-batch of size $b := |b_k| \leq n$, which here we assume is sampled uniformly and independently at each iteration. Note that GD is a particular case of the above, obtained by choosing $b = n$ at every step. The key observation is the gradient of the empirical risk always lies in the span of the training data:

$$\nabla \hat{R}_{b_k}(\boldsymbol{\theta}_k) = -\frac{1}{|b_k|} \sum_{i \in b_k} (y_i - \langle \boldsymbol{\theta}_k, \boldsymbol{x}_i \rangle) \boldsymbol{x}_i \in \text{span}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \tag{2.18}$$

Therefore, for any iterate k, we have:

$$\boldsymbol{\theta}_k \in \boldsymbol{\theta}_0 + \text{span}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \tag{2.19}$$

which is an (affine) n-dimensional space. Assuming that the iterates of this algorithm converge to an interpolator $\boldsymbol{\theta}_k \to \boldsymbol{\theta}_\infty \in \mathcal{I}$[3], and recalling our characterisation of $\mathcal{I}$ from eq. (2.5):[4]

$$\mathcal{I} = \hat{\boldsymbol{\theta}}_{\text{ols}} + \ker(\boldsymbol{X}) = \hat{\boldsymbol{\theta}}_\star + \text{span}(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\perp. \tag{2.20}$$

where in the second equality we used that, by definition, the kernel is orthogonal to the image. This implies that:

$$\boldsymbol{\theta}_\infty - \boldsymbol{\theta}_0 \perp \boldsymbol{\theta}_\infty - \hat{\boldsymbol{\theta}}_{\text{ols}} \tag{2.21}$$

See fig. 2 for an illustration. Therefore:

$$||\hat{\boldsymbol{\theta}}_{\text{ols}} - \boldsymbol{\theta}_0||_2^2 = ||\hat{\boldsymbol{\theta}}_{\text{ols}} - \boldsymbol{\theta}_\infty||_2^2 + ||\hat{\boldsymbol{\theta}}_\infty - \boldsymbol{\theta}_0||_2^2 \geq ||\hat{\boldsymbol{\theta}}_\infty - \boldsymbol{\theta}_0||_2^2 \tag{2.22}$$

In other words, we must have:

$$\boldsymbol{\theta}_\infty = \arg\min_{\boldsymbol{\theta} \in \mathcal{I}} ||\boldsymbol{\theta}_0 - \boldsymbol{\theta}||_2^2 \tag{2.23}$$

Which is the same implicit bias of gradient flow. This means that, for the least-squares problem and from the perspective of the implicit bias, there is no difference between GD and SGD. A similar conclusion can be proven for more complex descent algorithms, such as SGD with momentum. However, it is important to stress that this property is specific to OLS, and is a direct consequence of the fact that the gradient lives in the span of the data, which for $n < d$ is a linear subspace of $\mathbb{R}^d$. As we will see next, more complicated architectures or loss functions will behave differently.

3 Implicit bias of diagonal linear networks

While the least-squares problem is instructive, it is too simple, missing some important features related to the way neural network are parametrised. We now turn our attention to another problem that remains simple enough so that an explicit mathematical analysis can be carried out, but that has additional structure that will shed light on an important aspect of implicit bias: the interplay between the network architecture and the training algorithm.

Consider the following model, known as a linear *diagonal neural network*:

$$f(\boldsymbol{x}; \boldsymbol{u}, \boldsymbol{v}) = \sum_{j=1}^{d} u_j v_j x_j = \langle \boldsymbol{u} \odot \boldsymbol{v}, \boldsymbol{x} \rangle \tag{3.1}$$

where the $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ are two vectors, and $\odot$ denote the entry-wise product between vectors, also known as the Hadamard product. This hypothesis can be equivalently seen in two ways:

[3] This can be proven under small enough constant learning rate $\eta_k = \eta$ in the $d > n$ regime. We won't do it here, but it is a reasonable assumption given that the problem is strongly convex when restricted to $\boldsymbol{\theta}_k \in \boldsymbol{\theta}_0 + \text{span}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$.

[4] More generally, note this is true for any reference interpolator $\hat{\boldsymbol{\theta}}_\star \in \mathcal{I}$.

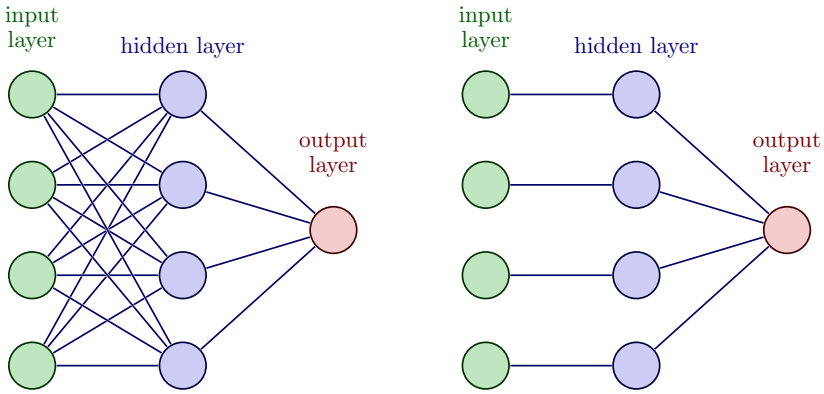

Figure 3: (**Left**) Standard fully connected two-layer neural network with $d = p = 4$ hidden-units and a single output. (**Right**) Diagonal two-layer neural network with $d = 4$ input nodes and a single output.

- First, it can be seen as a linear predictor $f(\boldsymbol{x}, \boldsymbol{\theta}) = \langle \boldsymbol{\theta}, \boldsymbol{x} \rangle$ with a particular parametrisation of the weights $\boldsymbol{\theta} = \boldsymbol{u} \odot \boldsymbol{v} \in \mathbb{R}^d$. Therefore, the statistical properties of our predictor are the same as the linear predictor. However, as we will going to see below, the fact that we parametrise it in a particular way has important consequences for optimisation.
- Alternatively, it can see it as a two-layer neural network $f(\boldsymbol{x}; \boldsymbol{\theta}) = \langle \boldsymbol{u}, \sigma(\boldsymbol{W}\boldsymbol{x}) \rangle$ with linear activation $\sigma(z) = z$, $p = d$ hidden-units and first-layer weights which are constrained to be a diagonal matrix $\boldsymbol{W}_{jk} = v_j \delta_{jk}$ — hence the name diagonal neural network. See fig. 3 for an illustration.

This model was introduced in Woodworth et al. (2020), and was motivated by previous work on implicit bias of algorithms in matrix factorisation Gunasekar et al. (2017).

Remark 1 (Symmetry)**.** This parametrisation has an obvious rescaling symmetry. Indeed, for any non-zero vector $\boldsymbol{b} \in \mathbb{R}^d$ we have:

$$f(\boldsymbol{x}; \boldsymbol{b} \odot \boldsymbol{u}, \boldsymbol{b}^{-1} \odot \boldsymbol{v}) = f(\boldsymbol{x}; \boldsymbol{u}, \boldsymbol{v}) \tag{3.2}$$

where the inverse is applied component-wise. This symmetry will play an important role in what follows.

As in the previous section, we will be interested in the empirical risk minimisation problem over a batch of training data:

$$\min_{\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d} L(\boldsymbol{u}, \boldsymbol{v}) \coloneqq \frac{1}{2n} \sum_{i=1}^{n} (y_i - \langle \boldsymbol{u} \odot \boldsymbol{v}, \boldsymbol{x}_i \rangle)^2 \tag{3.3}$$

The first important observation is that although the empirical risk is a convex function of the product $\boldsymbol{\theta} = \boldsymbol{u} \odot \boldsymbol{v}$, it is not a convex function of $\boldsymbol{u}, \boldsymbol{v}$! This means that in principle $L(\boldsymbol{u}, \boldsymbol{v})$ can have several critical points. To get some intuition, it is instructive to look at the $n = d = 1$ case, where the loss read:

$$\min_{(u,v) \in \mathbb{R}^2} L(u, v) \coloneqq \frac{1}{2}(y - uv)^2 \tag{3.4}$$

where without loss of generality we set $x = 1$. It is clear that the global minima manifold is given by the hyperbola $uv = 1$. Since $\partial_u L = -(y - uv)v$ and $\partial_v L = -(y - uv)u$, the only other critical point is the saddle-point $u = v = 1$. See fig. 4.

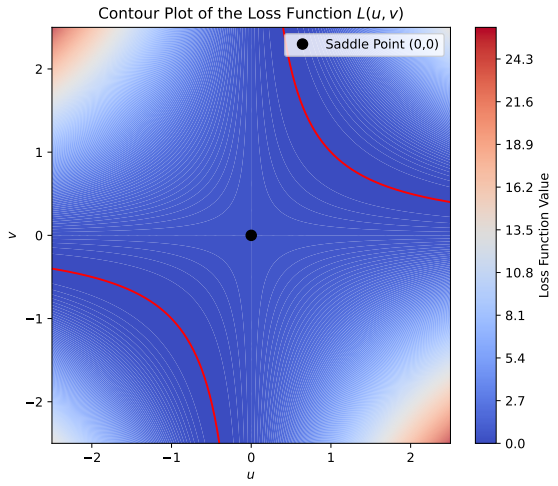

Figure 4: Contour plot of the loss $L(u, v) = 1/2(y - uv)^2$ for $y = 1$. The interpolation manifold is given by the hyperbola $uv = 1$, shown in solid red. The only other critical point is a saddle point at $u = v = 0$.

Remark 2. The phenomenology above is quite general. Indeed, by introducing a constraint $\boldsymbol{\theta} = \boldsymbol{u} \odot \boldsymbol{v}$ on the overparametrised space $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{2d}$ we introduce a symmetry that effectively degenerates the global minima of the original problem into a full orbit of the group.

As we will see, when $n, d > 1$, things are similar, except that we have additional saddle-points. For concreteness, we focus the discussion that follows to the underspecified regime $d > n$.

3.1 Properties of the landscape

As a starting point, we show that all the extremisers of $\hat{R}_n$ must be global minima. To see this, consider the map:

$$\boldsymbol{\theta} : (\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{2d} \mapsto \boldsymbol{u} \odot \boldsymbol{v} \in \mathbb{R}^{d}., \tag{3.5}$$

and note that $L(\boldsymbol{u}, \boldsymbol{v}) = (\hat{R}_n \circ \boldsymbol{\theta})(\boldsymbol{u}, \boldsymbol{v})$. Since the map $\boldsymbol{\theta}$ is differentiable at $\mathbb{R}^{2d}$, by the chain rule we have:

$$\nabla_{(\boldsymbol{u},\boldsymbol{v})} L(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{J}_{\boldsymbol{\theta}}(\boldsymbol{u}, \boldsymbol{v})^\top \nabla_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta}) \tag{3.6}$$

where $\boldsymbol{J}_{\boldsymbol{\theta}}(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{d \times 2d}$ is the Jacobian matrix of $\boldsymbol{\theta}$ at $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{2d}$. This implies that critical points of $L(\boldsymbol{u}, \boldsymbol{v})$ are either critical points of $\hat{R}(\boldsymbol{\theta})$ — which by convexity are global minima — or elements or in the kernel of the Jacobian. Since we have:

$$\partial_{u_j} \theta_k = v_k \delta_{jk}, \qquad \partial_{v_j} \theta_k = u_k \delta_{jk} \tag{3.7}$$

the Jacobian is a full-rank matrix $\operatorname{rank}(\boldsymbol{J}_{\boldsymbol{\theta}}(\boldsymbol{u}, \boldsymbol{v})) = d$ whenever the coordinates of $\boldsymbol{u}$ and $\boldsymbol{v}$ are not simultaneously zero (check this!). In other words, the critical points of L which are not critical points of $\hat{R}_n$ necessarely have $(u_j, v_j) = (0, 0)$ for some $j \in [d]$. However, these points cannot be extremisers of the loss, since the loss is flat across these directions. To see this, consider $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^d$ such that $(u_j, v_j) = (0, 0)$ for some $j \in [d]$. Then, it is easy to check that the $L(\boldsymbol{u}, \boldsymbol{v} + \alpha \boldsymbol{e}_j) = L(\boldsymbol{u}, \boldsymbol{v})$ where $\boldsymbol{e}_j \in \mathbb{R}^d$ is the basis vector (similarly for $\boldsymbol{u} + \alpha \boldsymbol{e}_j$, by symmetry). Recalling that by definition local

minima (maxima) are such that moving in their neighbourhood increase (decrease) the loss, we can conclude that the only local extremisers of $L(\boldsymbol{u}, \boldsymbol{v})$ are the global minima (i.e. the extremisers of $\hat{R}_n$).

Recall from section 2 that in the underspecified regime $n < d$, these are the interpolators $\mathcal{I} = \{\boldsymbol{\theta} \in \mathbb{R}^d : \boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{y}\}$. Therefore, accounting for the symmetry eq. (3.2), we can write the global minimisers of $L(\boldsymbol{u}, \boldsymbol{v})$ as:

$$\underset{(\boldsymbol{u},\boldsymbol{v})\in\mathbb{R}^{2d}}{\arg\min}\; L(\boldsymbol{u}, \boldsymbol{v}) = \left\{ \left(\operatorname{sign}(\boldsymbol{\theta}_\star)\sqrt{|\boldsymbol{\theta}_\star|} \odot \boldsymbol{b}, \sqrt{|\boldsymbol{\theta}_\star|} \odot \boldsymbol{b}^{-1} \right) : \boldsymbol{\theta}_\star \in \mathcal{I}, \boldsymbol{b} \in \mathbb{R}^d, \boldsymbol{b} \neq \boldsymbol{0} \right\}. \tag{3.8}$$

where the non-linear operations are understood entry-wise. Now let's look at the remaining critical points. From the discussion above, we know these are saddle-points $(\boldsymbol{u}_c, \boldsymbol{v}_c) \in \mathbb{R}^{2d}$ with a subset of coordinates $(u_{c,j}, v_{c,j}) = (0, 0)$, and hence $\theta_{c,j} \coloneqq \theta_j(\boldsymbol{u}_c, \boldsymbol{v}_c) = 0$. Consider the gradient of L:

$$\begin{aligned} \nabla_{\boldsymbol{u}} L(\boldsymbol{u}, \boldsymbol{v}) &= -\frac{1}{n}\boldsymbol{v} \odot \boldsymbol{X}^\top \left(\boldsymbol{y} - \boldsymbol{X}^\top(\boldsymbol{u} \odot \boldsymbol{v}) \right) = \boldsymbol{v} \odot \nabla_{\boldsymbol{\theta}} \hat{R}(\boldsymbol{u} \odot \boldsymbol{v}) \\ \nabla_{\boldsymbol{v}} L(\boldsymbol{u}, \boldsymbol{v}) &= -\frac{1}{n}\boldsymbol{u} \odot \boldsymbol{X}^\top \left(\boldsymbol{y} - \boldsymbol{X}^\top(\boldsymbol{u} \odot \boldsymbol{v}) \right) = \boldsymbol{u} \odot \nabla_{\boldsymbol{\theta}} \hat{R}(\boldsymbol{u} \odot \boldsymbol{v}) \end{aligned} \tag{3.9}$$

To have a critical point, we need $\nabla_{\boldsymbol{u}} L = \nabla_{\boldsymbol{v}} L = \boldsymbol{0}$. An obvious choice is $(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{0}, \boldsymbol{0})$. For $(\boldsymbol{u}, \boldsymbol{v}) \neq (\boldsymbol{0}, \boldsymbol{0})$, since a subset of the coordinates is zero, this is automatically satisfied for these coordinates. In the remaining coordinates, we effectively have an OLS problem. To formalise this, define the support of a vector:

$$\operatorname{supp}(\boldsymbol{\theta}) = \{j \in [d] : \theta_j \neq 0\}. \tag{3.10}$$

Then, the previous condition can be written $\nabla_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta})_j = 0$ for $j \notin \operatorname{supp}(\boldsymbol{\theta}_c)$. This allow us to characterise the saddle-points of L directly in terms of $\hat{R}_n$ as:

$$\boldsymbol{\theta}_c \in \underset{\theta_j = 0 \text{ for } j \notin \operatorname{supp}(\theta_c)}{\arg\min} \hat{R}_n(\boldsymbol{\theta}) \tag{3.11}$$

⚠ Note that due to the symmetry eq. (3.2), all points in the orbit of a saddle-point $(\boldsymbol{u}_c, \boldsymbol{v}_c) \in \mathbb{R}^{2d}$ are also saddle points.

We can summarise the properties of the loss landscape in the following proposition.

Proposition 1. The critical points of the empirical risk eq. (3.3) in the underspecified regime $d > d$ are characterised as follows.

- The only extremisers of $L(\boldsymbol{u}, \boldsymbol{v})$ are global minima, which can be written as:

$$\underset{(\boldsymbol{u},\boldsymbol{v})\in\mathbb{R}^{2d}}{\arg\min}\; L(\boldsymbol{u}, \boldsymbol{v}) = \left\{ \left(\operatorname{sign}(\boldsymbol{\theta}_\star)\sqrt{|\boldsymbol{\theta}_\star|} \odot \boldsymbol{b}, \sqrt{|\boldsymbol{\theta}_\star|} \odot \boldsymbol{b}^{-1} \right) : \boldsymbol{\theta}_\star \in \mathcal{I}, \boldsymbol{b} \in \mathbb{R}^d, \boldsymbol{b} \neq \boldsymbol{0} \right\}. \tag{3.12}$$

  where $\mathcal{I} = \{\boldsymbol{\theta} \in \mathbb{R}^d : \boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{y}\}$ are the set of OLS interpolators.

- All the other critical points are saddle-points, given by $\boldsymbol{\theta}_c = \boldsymbol{\theta}(\boldsymbol{u}_c, \boldsymbol{v}_c)$ such that:

$$\boldsymbol{\theta}_c \in \underset{\theta_j = 0 \text{ for } j \notin \operatorname{supp}(\theta_c)}{\arg\min} \hat{R}_n(\boldsymbol{\theta}) \tag{3.13}$$

3.2 The implicit bias of gradient flow

We now consider gradient flow for the linear diagonal network:

$$\dot{\boldsymbol{u}}(t) = -\nabla_{\boldsymbol{u}} L(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{n}\boldsymbol{v}(t) \odot \boldsymbol{X}^\top \boldsymbol{r}(t) \tag{3.14}$$

$$\dot{\boldsymbol{v}}(t) = -\nabla_{\boldsymbol{v}} L(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{n}\boldsymbol{u}(t) \odot \boldsymbol{X}^\top \boldsymbol{r}(t) \tag{3.15}$$

where for notational convenience we defined the displacement vector $\boldsymbol{r}(t) = \boldsymbol{y} - \boldsymbol{X}^\top(\boldsymbol{u}(t) \odot \boldsymbol{v}(t))$. In particular, we would like to compare how this dynamics differs from the gradient flow on $\hat{R}_n$ in eq. (2.8). For that, we can attempt to reconstruct the trajectory $\boldsymbol{\theta}(t) = \boldsymbol{u}(t) \odot \boldsymbol{v}(t)$ from the above:

$$\begin{aligned}\dot{\boldsymbol{\theta}} &= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{u} \odot \boldsymbol{v}) = \dot{u} \odot \boldsymbol{v} + \boldsymbol{u} \odot \dot{\boldsymbol{v}} \\ &= \frac{1}{n}\boldsymbol{X}^\top \boldsymbol{r}(t) \odot (\boldsymbol{v}^2 + \boldsymbol{u}^2)\end{aligned} \tag{3.16}$$

where the squares are understood entry-wise. In principle, it is not clear how to close this equation in $\boldsymbol{\theta}$. In order to achieve this, we note that the following quantity:

$$\boldsymbol{I}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^2 - \boldsymbol{v}^2}{2} \tag{3.17}$$

is an integral of motion, i.e.:

$$\dot{\boldsymbol{I}} = \boldsymbol{u} \odot \dot{\boldsymbol{u}} - \boldsymbol{v} \odot \dot{\boldsymbol{v}} = 0 \tag{3.18}$$

it is therefore conserved along the flow. With this, we can rewrite:

$$\boldsymbol{u}^2 + \boldsymbol{v}^2 = \sqrt{\boldsymbol{\theta}^2 + \boldsymbol{I}^2} \tag{3.19}$$

where we remind the reader the square-root is understood component-wise. Therefore:

$$\begin{aligned}\dot{\boldsymbol{\theta}}(t) &= -\frac{2}{n}\boldsymbol{X}^\top \boldsymbol{r}(t) \odot \sqrt{\boldsymbol{\theta}(t)^2 + \boldsymbol{I}^2} \\ &= -2\sqrt{\boldsymbol{\theta}(t)^2 + \boldsymbol{I}^2} \odot \nabla_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta})\end{aligned} \tag{3.20}$$

which is remarkably different from the simple gradient flow $\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} \hat{R}_n(\boldsymbol{\theta})$. Unfortunately, solving this non-convex flow explicitly is hard. In order to characterise the implicit bias, we will make the following rewriting. Define the potential function:

$$\phi_{\boldsymbol{I}}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{j=1}^{d}\left(\theta_j \sinh^{-1}\left(\frac{\theta_j}{I_j}\right) - \sqrt{\theta_j^2 + I_j^2} + I_j\right) \tag{3.21}$$

Noting that:

$$\partial_{\theta_j}\phi = \sinh^{-1}\left(\frac{\theta_j}{I_j}\right) \tag{3.22}$$

It is easy to check that we can rewrite the flow in eq. (3.20) as:

$$\frac{\mathrm{d}\nabla_{\boldsymbol{\theta}}\phi}{\mathrm{d}t} = -\nabla_{\boldsymbol{\theta}}\hat{R}_n(\boldsymbol{\theta}) \tag{3.23}$$

The reader who is familiar with convex optimisation will recognise this is a *Mirror descent flow*, the zero learning rate limit of the *Mirror descent* algorithm. Mirror descent is an algorithm for constrained optimisation that generalises the Euclidean projection into the constraint set to other geometries, in this case implicitly defined by the Bregman divergence of a potential function ϕ:

$$D_\phi(\boldsymbol{\theta}, \boldsymbol{\theta}') = \phi(\boldsymbol{\theta}) - \phi(\boldsymbol{\theta}') - \langle \nabla\phi(\boldsymbol{\theta}), \boldsymbol{\theta} - \boldsymbol{\theta}' \rangle \tag{3.24}$$

Note that the Euclidean projection is recovered with the quadratic potential $\phi(\boldsymbol{\theta}) = {}^1\!/_2||\boldsymbol{\theta}||_2^2$. The advantage of writing this as a mirror flow is that, in the case in which the potential is a strictly convex function, which is the case here:

$$\partial_{\theta_j}\partial_{\theta_k}\phi_{\boldsymbol{I}}(\boldsymbol{\theta}) = \frac{1}{2\sqrt{\theta_j^2 + I_j^2}}\delta_{jk} > 0, \tag{3.25}$$

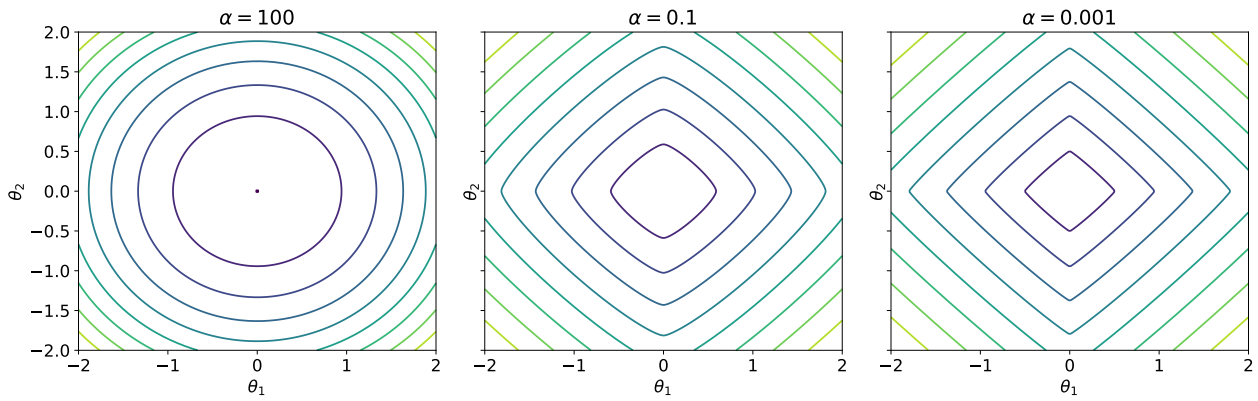

Figure 5: Contour plot of the potential ϕ_α defined in eq. (3.21) with $\boldsymbol{I} = \alpha^2 \mathbf{1}_d$ in the (θ_1, θ_2)-plane for $\alpha \in \{10^{-3}, 0.1, 100\}$

strong guarantees from convex optimisation apply. For instance, it can be proved that the flow in eq. (3.23) converge $\boldsymbol{\theta} \to \boldsymbol{\theta}_\infty$, and that in the regime $d > n$ this must necessarily be an interpolator $\boldsymbol{\theta}_\infty \in \mathcal{I}$ of $\hat{R}_n$. By the same argument as in section 2.2, since $\nabla_{\boldsymbol{\theta}} \hat{R}_n \in \text{span}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$, eq. (3.23) implies that for any $t > 0$:

$$\nabla_{\boldsymbol{\theta}} \phi_{\boldsymbol{I}}(\boldsymbol{\theta}(t)) \in \nabla_{\boldsymbol{\theta}} \phi_{\boldsymbol{I}}(\boldsymbol{\theta}_0) + \text{span}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \tag{3.26}$$

Moreover, we can repeat the argument in eq. (2.22) with the Bregman divergence instead of the Euclidean norm. Indeed, for any interpolator $\hat{\boldsymbol{\theta}} \in \mathcal{I}$

$$\begin{aligned} D_\phi(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}_0) &= D_\phi(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}_\infty) + D_\phi(\boldsymbol{\theta}_\infty, \boldsymbol{\theta}_0) + \langle \nabla \phi_{\boldsymbol{I}}(\boldsymbol{\theta}_\infty) - \underbrace{\nabla \phi_{\boldsymbol{I}}(\boldsymbol{\theta}_0), \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_\infty}_{=0} \rangle \\ &\geq D_\phi(\boldsymbol{\theta}_\infty, \boldsymbol{\theta}_0) \end{aligned} \tag{3.27}$$

which implies that:

$$\boldsymbol{\theta}_\infty = \underset{\boldsymbol{\theta} \in \mathcal{I}}{\arg\min}\ D_\phi(\boldsymbol{\theta}, \boldsymbol{\theta}_0) \tag{3.28}$$

Or in words: GD converges to the interpolator which has minimal Bregman divergence to the initial condition. Note that, as expected, this result recovers eq. (2.23) when $\phi(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2^2$. Interpolators of the type eq. (3.28) can be quite different from the ones in eq. (2.23) — we now discuss these differences in detail.

Role of the initialisation — To study the differences in a concrete setting, we consider the initial condition $\boldsymbol{\theta}_0 = \mathbf{0}$. Recall from section 2.2 that in this case, gradient flow on the square loss converges to $\hat{\boldsymbol{\theta}}_{\text{ols}} = \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \boldsymbol{y}$, the minimum ℓ_2-norm interpolator. What about the diagonal neural network flow? The initial condition $\boldsymbol{\theta}_0 = \boldsymbol{u}_0 \odot \boldsymbol{v}_0 = \mathbf{0}$ corresponds to a full family of initial conditions on $(\boldsymbol{u}_0, \boldsymbol{v}_0)$. Again, for concreteness we focus our attention on a particular subclass of solutions parametrised by a single scalar parameter $\alpha \geq 0$:

$$\boldsymbol{u}_0 = \sqrt{2}\alpha \mathbf{1}_d, \qquad \boldsymbol{v}_0 = \mathbf{0} \tag{3.29}$$

In this case, the integral of motion is given by:

$$\boldsymbol{I}(\boldsymbol{u}_0, \boldsymbol{v}_0) = \alpha^2 \mathbf{1}_d \tag{3.30}$$

A simple asymptotic expansion allow us to see how the potential $\phi_\alpha \equiv \phi_{\alpha^2 \mathbf{1}_d}$ looks like when α is small or large:

$$\phi_\alpha(\boldsymbol{\theta}) \underset{\alpha \to 0^+}{\asymp} \log(1/\alpha) \cdot ||\boldsymbol{\theta}||_1, \qquad \phi_\alpha(\boldsymbol{\theta}) \underset{\alpha \to \infty}{\asymp} \frac{1}{4\alpha^2} ||\boldsymbol{\theta}||_2^2. \tag{3.31}$$

In other words, α continuously interpolate $\boldsymbol{\phi}_\alpha$ between the ℓ_1 and the ℓ_2 norm! See fig. 5 for a contour plot of ϕ_α at different values of α. Consequently, from eq. (3.28) it can be shown that the solution $\boldsymbol{\theta}_\infty(\alpha)$ will inherit a similar implicit bias:

$$\boldsymbol{\theta}_\infty(\alpha) \underset{\alpha \to 0^+}{\to} \underset{\boldsymbol{\theta} \in \mathcal{I}}{\arg\min} ||\boldsymbol{\theta}||_1, \qquad \boldsymbol{\theta}_\infty(\alpha) \underset{\alpha \to \infty}{\to} \underset{\boldsymbol{\theta} \in \mathcal{I}}{\arg\min} ||\boldsymbol{\theta}||_2 \tag{3.32}$$

⚠ Proving this result requires showing uniform convergence of ϕ_α as $\alpha \to 0/\infty$ on a compact of $\mathbb{R}^d$ and that $\boldsymbol{\theta}_\infty(\alpha)$ is bounded for all $\alpha \geq 0$. A detailed proof can be found in Proposition 6 of Pesme (2024).

Remark 3 (Relationship to lazy training)**.** Recall from Lecture 4 our discussion on how the scale of initialisation determines whether our network behaves as a kernel method or whether it learn features. Kernel methods are just linear models on Hilbert space, and therefore the implicit bias of GF/GF/SGD for kernels is the same as least-squares, which for vanishing initialisation $\boldsymbol{\theta}_0 = \mathbf{0}$ is the minimum-ℓ_2 norm. Indeed, this is coherent with the $\alpha \to \infty$ limit of our diagonal linear network.

Interestingly, this analogy also hold in the opposite, $\alpha \to 0^+$ limit. Indeed, this limit is akin to the mean-field limit discussed in Lectures 3 & 4, where the wide network can be seen as an integral over a limiting probability measure over the hidden units. The implicit bias of gradient flow on the square loss corresponds in this case to the minimum Barron norm interpolator, which is akin to a ℓ_1-penalty on the weights.

Remark 4 (Relation to generalisation)**.** It is important to stress that, whether ℓ_1, ℓ_2 or in between, the implicit regularisation above is a property of the architecture and training algorithm, and hence is independent of generalisation. Indeed, we have made no assumption on the data distribution, and whether a particular regularisation is "good" in terms of generalisation will crucially depend on the properties of the target function. For instance, if the underlying predictor is a linear function $y_i = \langle \boldsymbol{\theta}_\star, \boldsymbol{x}_i \rangle + \varepsilon_i$ with sparse weights $||\boldsymbol{\theta}_\star||_0 \ll d$, we expect the minimum-$\ell_1$ norm predictor to generalise better than the minimum-ℓ_2 norm predictor. But the converse can be true if $||\boldsymbol{\theta}_\star||_2^2 = d$ instead.

4 To go further

4.1 Benefits of noise

As we discussed in section 2.3, in the OLS problem the implicit bias is the same whether we use gradient flow, GD or SGD — a consequence of the fact that in these three algorithms problems the gradient lives in the span of the covariates. Interestingly, the situation is very different for linear diagonal neural networks. As shown in (Pesme et al., 2021), adding a stochastic term to gradient flow leads to an algorithm with similar potential, but with an effective initialisation scale $\alpha_{\text{sgf}} < \alpha_{rmgf}$. This implies that for a fixed choice of initialisation scale α, these two algorithms will converge to different interpolators with different implicit biases — a situation closer to fig. 1. In particular, since $\alpha_{\text{sgf}} < \alpha_{rmgf}$, $\boldsymbol{\theta}_\infty^{\text{sgf}}$ will tend to be sparser than $\boldsymbol{\theta}_\infty^{\text{gf}}$. The role of the step-size (which quantifies the difference between GF and GD) has also been studied in (Nacson et al., 2022).

4.2 Implicit bias in binary classification

Our discussion in this lecture has focused on results for the square loss function. There is a similar line of work characterising the implicit bias of algorithms for for binary classification with the logistic loss. Soudry et al. (2018) has shown that, when data is linearly separable (hence interpolators exist), GD or SGD for logistic regression converge to the minimum margin interpolator, independently of the initial condition.

References

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. *Advances in neural information processing systems*, 30, 2017.

Shengchao Liu, Dimitris Papailiopoulos, and Dimitris Achlioptas. Bad global minima exist and sgd can reach them. *Advances in Neural Information Processing Systems*, 33:8543–8552, 2020.

Mor Shpigel Nacson, Kavya Ravichandran, Nathan Srebro, and Daniel Soudry. Implicit bias of the step size in linear diagonal neural networks. In *International Conference on Machine Learning*, pages 16270–16295. PMLR, 2022.

Scott Pesme. *Deep Learning Theory Through the Lens of Diagonal Linear Networks*. PhD thesis, Lausanne, 2024. URL https://infoscience.epfl.ch/handle/20.500.14299/208225.

Scott Pesme, Loucas Pillaud-Vivien, and Nicolas Flammarion. Implicit bias of sgd for diagonal linear networks: a provable benefit of stochasticity. *Advances in Neural Information Processing Systems*, 34:29218–29230, 2021.

Loucas Pillaud-Vivien and Scott Pesme. Rethinking sgd's noise - ii: Implicit bias, Sep 2022. URL https://francisbach.com/implicit-bias-sgd/.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, 2020.