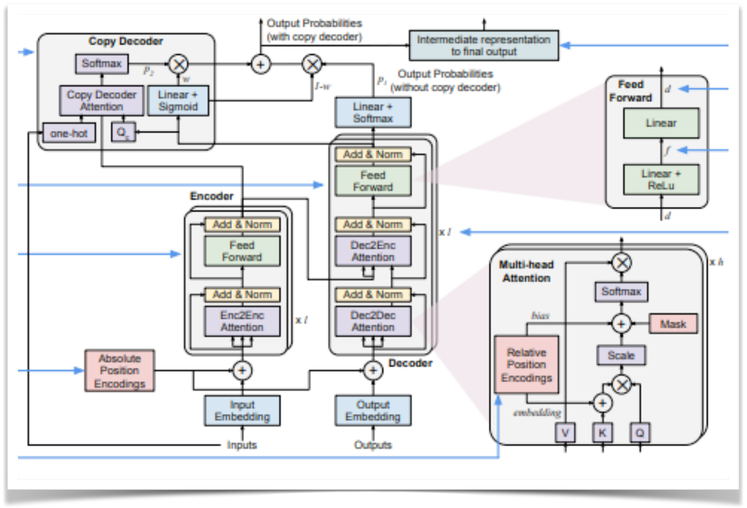
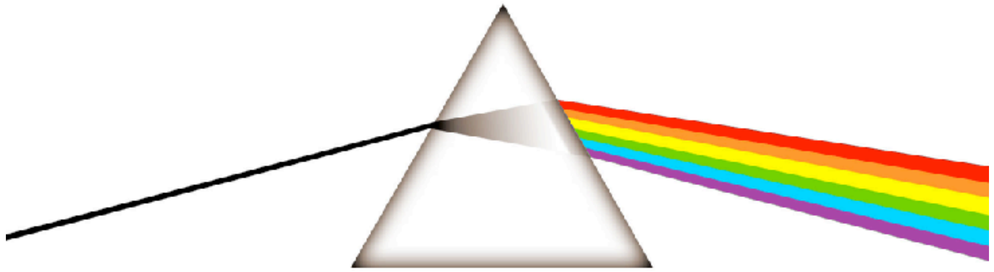
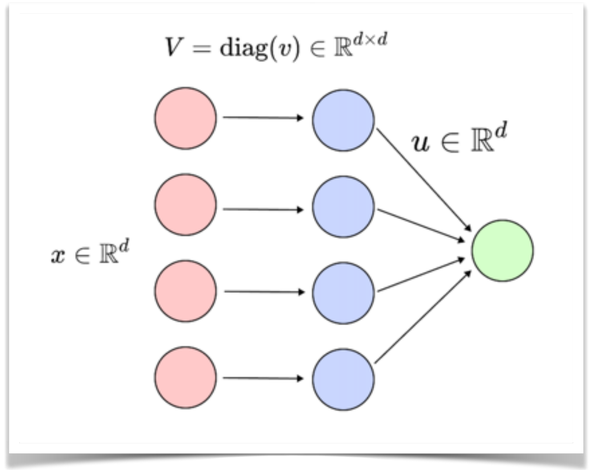
(Old school) Deep Learning Theory Through the Lens of Diagonal Linear Networks







Scott Pesme,
PostDoc at Inria Grenoble with Julien Mairal

(Old school) Deep Learning Theory
Through the Lens of Diagonal Linear Networks

Loucas Pillaud-Vivien
Assistant professor, Ecole des Ponts

Scott Pesme,
PostDoc at Inria Grenoble with Julien Mairal

Mathieu Even
PostDoc, Inria Montpellier

Radu-Alexandru Dragomir
Assistant professor, Telecom Paris

Nicolas Flammarion
Assistant professor, EPFL

Hristo Papazov
PhD, EPFL

Suriya Gunasekar
Researcher, Microsoft Redmond

Context: a deep learning breakthrough

Fact: Neural networks are everywhere and they lead to outstanding results

Ingredients: Data + Compute power + **Architectures and training tricks**

However: We don't understand why and how they work.

What could have possibly gone wrong?!

"From the formal point of view one cannot guarantee that neural networks generalize well, ***since according to theory****, in order to control generalisation ability one should control two factors:* ***the value of the empirical risk*** *and* ***the value of the confidence interval****. Neural networks, however, cannot control either two."*

Vapnik (1999)

The Vapnik-Jackel Bet in 1995:

1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand
quantitatively why big neural nets working on large databases are not so bad.
(Understanding means that there will be clear conditions and bounds)

Vapnik bets (one fancy dinner) that Jackel is wrong.

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

****

2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong

3/14/95
V. Vapnik

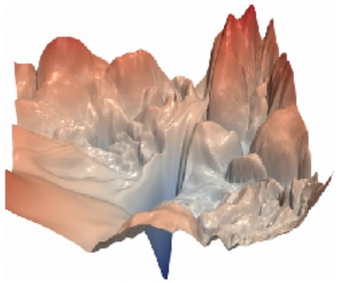
3/14//95
L. Jackel

3/14/95
Witnessed by Y. LeCun

Two hurdles:

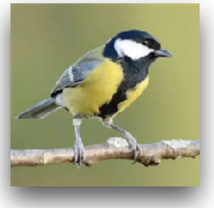
Optimisation

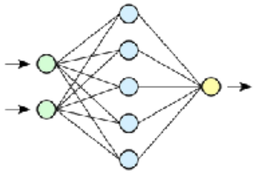
—> Non convexity



Why do gradient methods work?

Generalisation





Mésange charbonnière

Why does it generalise to unseen data?

How does the trained network make a prediction?

Insights from empirical works:

Deep neural nets can 'easily' be trained to minimum training loss

=> non-convexity is not too problematic

Though there exist `bad' global minima, gradient methods converge towards `good' ones

=> the training algorithm must be taken into account

Importance of taking the training algorithm into account:

Cifar-10 dataset, ResNet-18 (with batch-normalisation)

	Train accuracy	Test accuracy
'Adversarial minimum'	100%	~0%
Gradient Descent	100%	~7 (??)
Stochastic Gradient Descent	100%	8
SGD + momentum	100%	
GD + mom + DA + ℓ_2	100%	87%
SGD + mom + DA + ℓ_2	100%	95%

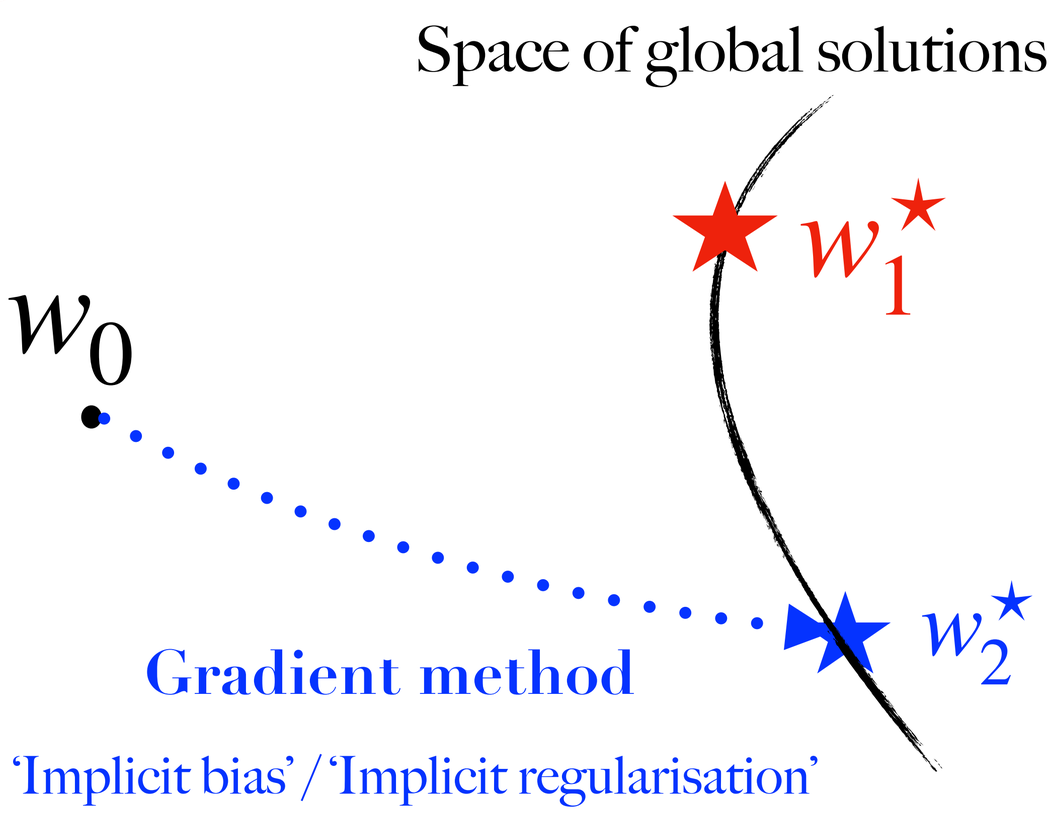
"Old school" Pre-LLM observations!

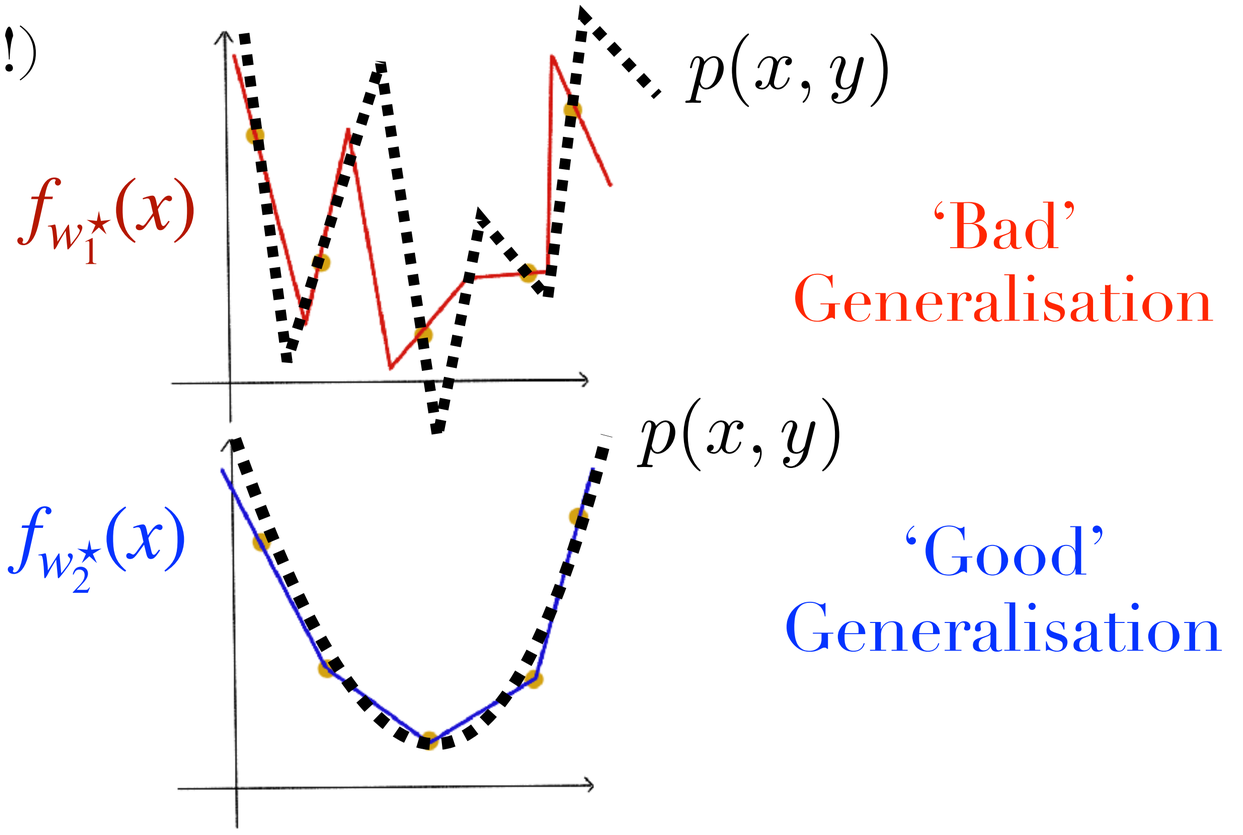
Revisiting Small Batch Training For Deep Neural Networks, Masters and Luschi 2018
Bad Global Minima Exist and SGD Can Reach Them, Liu et al. 2020
Stochastic Training is Not Necessary for Generalization, Geiping et al. 2021

Vapnik (1999): 'the designers of neural networks compensate the mathematical shortcomings with the high art of engineering'

The algorithm and its hyperparameters are crucial to understand deep learning performances

Insights from empirical works (with a drawing)





Depending on the true data distribution:

"Structure 1" ≲ "Structure 2".

TLDL: A theory of deep learning cannot exclude the training algorithm

Algorithm

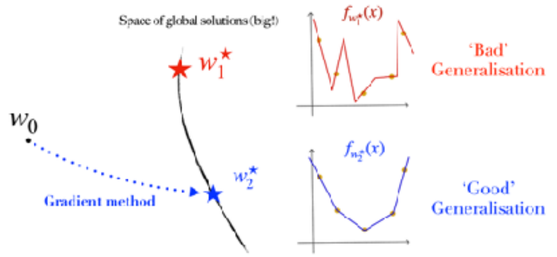
Data

Architecture

	Train accuracy	Test accuracy
'Adversarial minimum'	100%	0%
GD	100%	~72% (??)
SGD	100%	85%
SGD + momentum	100%	89%
GD + mom + DA + ℓ_2	100%	87%
SGD + mom + DA + ℓ_2	100%	95%

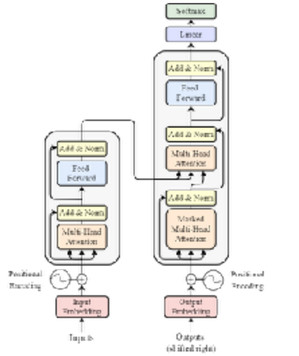
Goal:

Gain a theoretical understanding of these empirical insights.



Adopted strategy:

Transformer



$$\mathrm{Attn}(h)(t) = h(t) + \sum_{i=1}^{h} W_o^i \sum_{s=1}^{t} \sigma[(W_Q^i h(t))^\top W_K^i h(s)]\, W_V^i h(s)$$

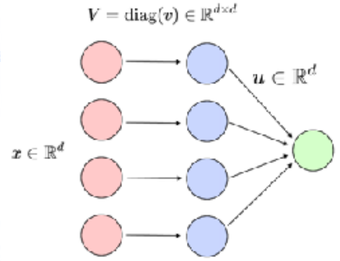
$$\mathrm{FF}(h)(t) = f(h(t)),$$

Practical relevance (for now)

Theoretical tractability (for now)

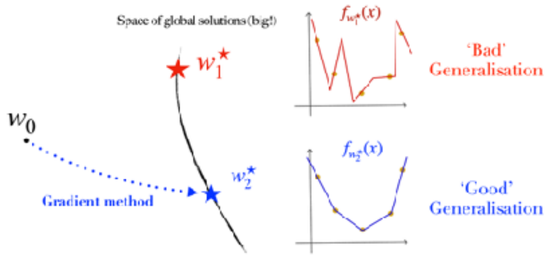
$x \mapsto \langle x, \beta \rangle$

2-layer diagonal linear network



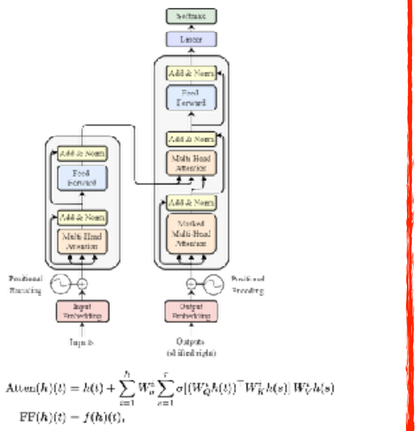
Goal:

Gain a theoretical understanding of these empirical insights.



Adopted strategy:

Transformer



$$\mathrm{Attn}(h)(t) = h(t) + \sum_{i=1}^{h} W_o^i \sum_{s=1}^{t} \sigma\big((W_Q^i h(t))^\top W_K^i h(s)\big)\, W_V^i h(s)$$

$$\mathrm{FF}(h)(t) = f(h(t)),$$

Practical relevance (for now)

Theoretical tractability (for now)

2-layer diagonal linear network

$V = \mathrm{diag}(v) \in \mathbb{R}^{d \times d}$

$u \in \mathbb{R}^d$

$x \in \mathbb{R}^d$

$x \mapsto \langle x, \beta \rangle$

Linear regression $x \mapsto \langle x, \beta \rangle$

Overparametrised least squares:

$$L(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \langle \beta,\ x_i \rangle)^2 \qquad d \gg n \quad \text{hence} \qquad \arg\min L = \{\beta^\star \in \mathbb{R}^d, \langle \beta^\star, x_i \rangle = y_i, \forall i \in [n]\}$$

is a 'big' affine space

Mini-batch SGD:

$$\beta_{t+1} = \beta_t + \gamma \underbrace{\frac{1}{b} \sum_{i \in \mathcal{B}_t} (y_i - \langle x_i, \beta_t \rangle) x_i}_{\in \text{span}(x_1, \ldots, x_n)}$$

Independant of:
- Stepsize
- Batch-size
- (Momentum)

Pythagorean theorem

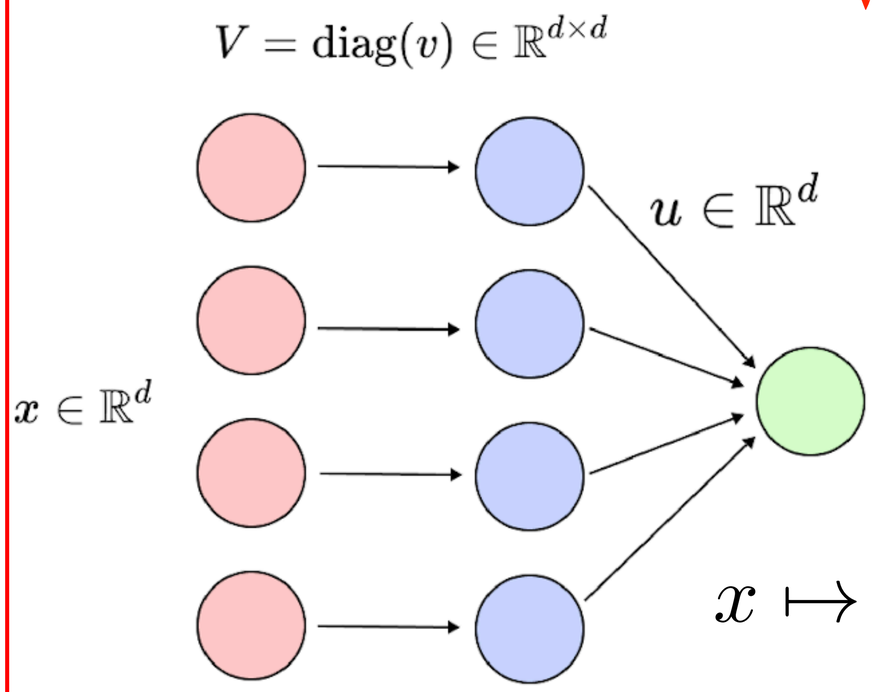
Pythagoras et al. 500 BC

implies

Implicit regularisation

$$\beta_\infty^{(\mathrm{S})\mathrm{GD}} = \underset{\forall i,\ \langle \beta^\star, x_i \rangle = y_i}{\operatorname{argmin}} \|\beta^\star - \beta_0\|_2^2$$

The 2-layer diagonal linear network



"Weights"

$$w = \begin{pmatrix} u \\ v \end{pmatrix} \in \mathbb{R}^{2d}$$

Element-wise multiplication

$$x \mapsto \langle \underbrace{u \odot v}_{\beta_w \in \mathbb{R}^d}, \ x \rangle$$

"Predictors"

Linear regression setting with $\beta_w = u \odot v$ reparametrisation:

$$F(w) = L(u \odot v) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \langle u \odot v, \ x_i \rangle)^2$$

'Overparametrised' setting:

(Dimension d) > (Number of samples n)

Many zero-loss training solutions!

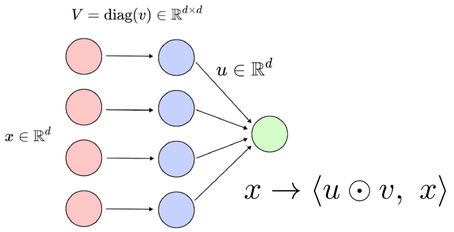
Proxy model which is theoretically tractable

Sparse recovery literature: *Hoff [2016]*, *Zhao et al [2019]*, *Vaskevicius et al. [2019]*, *Poon and Peyré [2021]*

ML literature: *Gunasekar et al. [2017]*, *Woodworth et al. [2019]*, *(…)*

Gradient flow over a 2-layer diagonal linear network

Woodworth et al., Kernel and Rich Regimes in Overparametrized Models, COLT 2020



"Weights": $w = \begin{pmatrix} u \\ v \end{pmatrix} \in \mathbb{R}^{2d}$

"Predictors": $\beta_w = u \odot v \in \mathbb{R}^d$

Square loss:

$$F(w) = L(u \odot v) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \langle u \odot v,\ x_i\rangle)^2$$

$$L(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \langle \beta,\ x_i\rangle)^2$$

Weights follow the GF:

$$\mathrm{d}w_t = -\nabla F(w_t)\mathrm{d}t$$

Simplified Initialisation:

$$u^\alpha_{t=0} = \alpha\mathbf{1}$$
$$v^\alpha_{t=0} = \mathbf{0}$$

$\beta^\alpha_t = u^\alpha_t \odot v^\alpha_t$

Predictors follow a mirror flow:

$$\mathrm{d}\nabla\phi_\alpha(\beta^\alpha_t) = -\nabla L(\beta^\alpha_t)\mathrm{d}t$$

Initialisation

$$\beta^\alpha_{t=0} = \mathbf{0}$$

Convergence of the iterates towards:

$$\beta^{\mathrm{GF}}_\alpha = \underset{y_i = \langle x_i, \beta^\star\rangle, \forall i}{\arg\min}\ \phi_\alpha(\beta^\star)$$

Hypentropy potential:

$$\phi_\alpha(\beta) = \frac{1}{2}\sum_{i=1}^{d}\left(\beta_i \operatorname{arcsinh}\left(\frac{\beta_i}{\alpha^2}\right) - \sqrt{\beta_i^2 + \alpha^4} + \alpha^2\right)$$

Gradient flow over a 2-layer diagonal linear network:

Convergence of the iterates to

$$\beta_\alpha^{GF} = \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \ \phi_\alpha(\beta^\star)$$

$$\phi_\alpha(\beta) = \frac{1}{2}\sum_{i=1}^{d}\left(\beta_i \operatorname{arcsinh}\left(\frac{\beta_i}{\alpha^2}\right) - \sqrt{\beta_i^2 + \alpha^4} + \alpha^2\right)$$

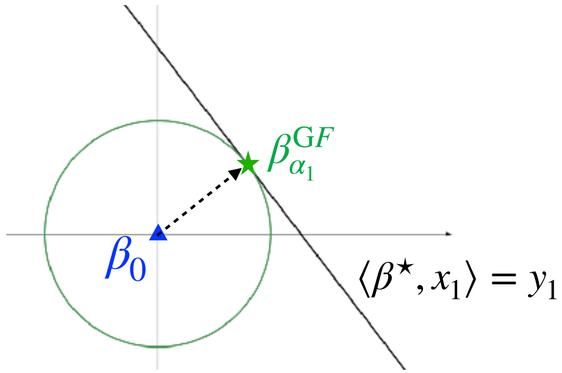
$$\underset{\alpha \to \infty}{\sim} \frac{1}{4\alpha^2}\|\cdot\|_2^2 \qquad \underset{\alpha \to 0}{\sim} \ln(1/\alpha)\ \|\cdot\|_1$$

$$\beta_\alpha^{GF} \underset{\alpha \to \infty}{\longrightarrow} \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \|\beta^\star\|_2^2$$

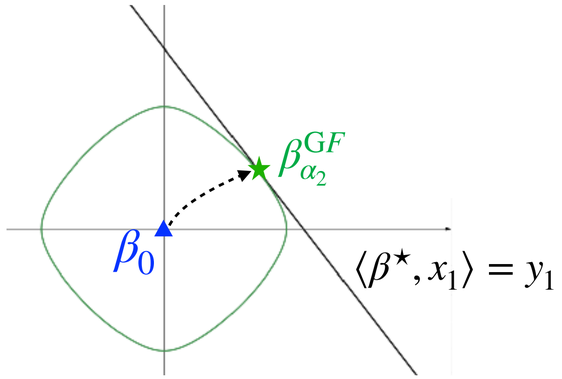
Two asymptotic regimes

$$\beta_\alpha^{GF} \underset{\alpha \to 0}{\longrightarrow} \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \|\beta^\star\|_1$$

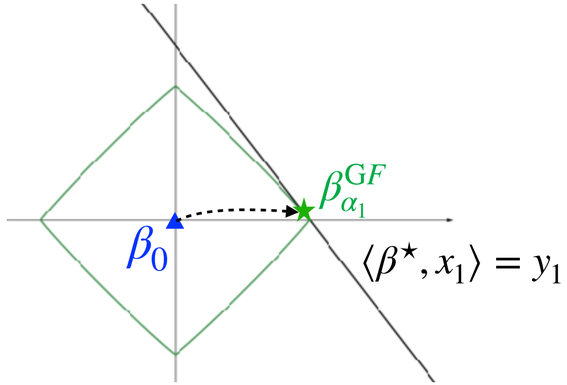
"Big" initialisation α_1



"Intermediate" initialisation α_2



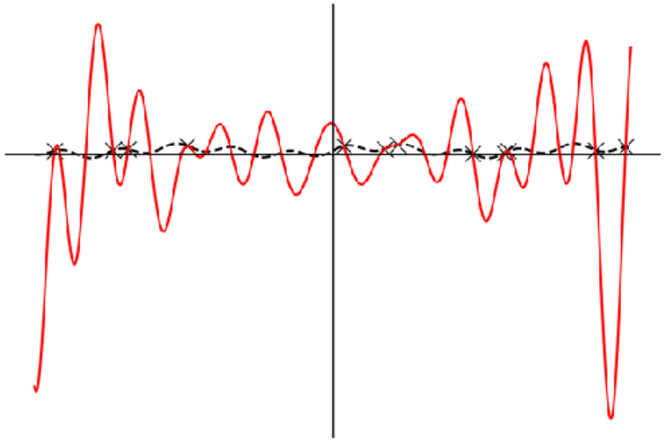
"Small" initialisation α_3



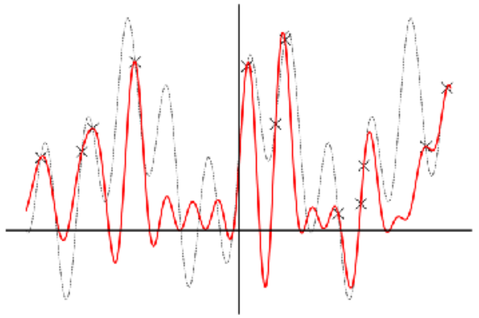
Toy experiments: **Dataset:** $(x_i, y_i)_{1\leq i\leq n} \in \mathbb{R}\times\mathbb{R}$ $(n,d) = (12, 60)$

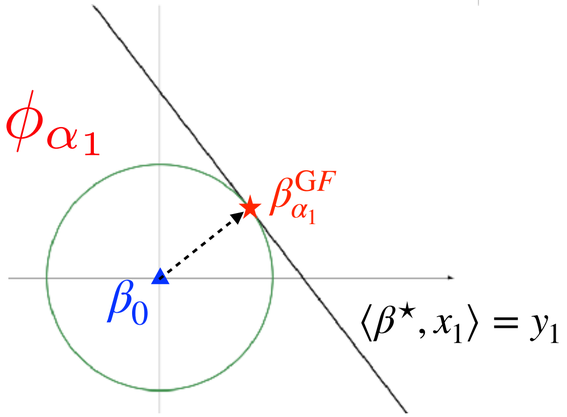
Feature map: $\varphi(x) = (\cos(\pi k x), \sin(\pi k x))_{0\leq k\leq d/2} \in \mathbb{R}^d$

DLN: $F(w) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \langle u\odot v,\ \varphi(x_i)\rangle)^2$ **GF with init:** $u_0^\alpha = \alpha\mathbf{1}$, $v_0^\alpha = \mathbf{0}$

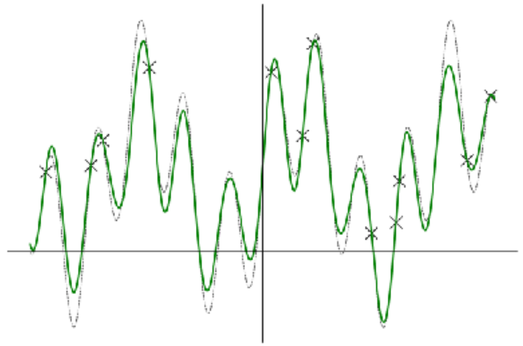


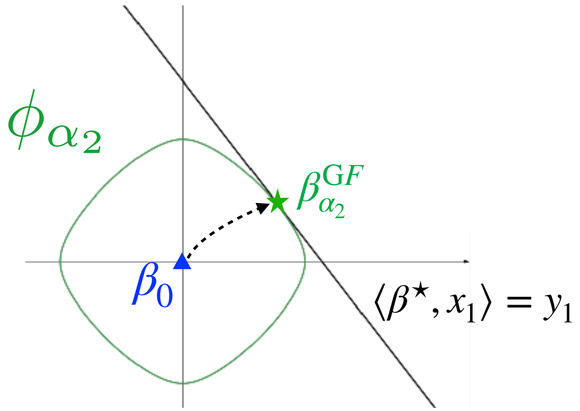
"Big" initialisation α_1



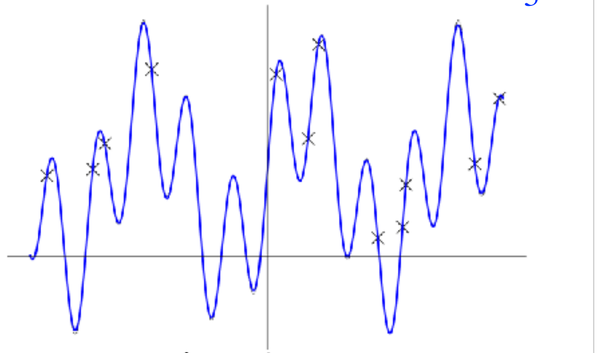


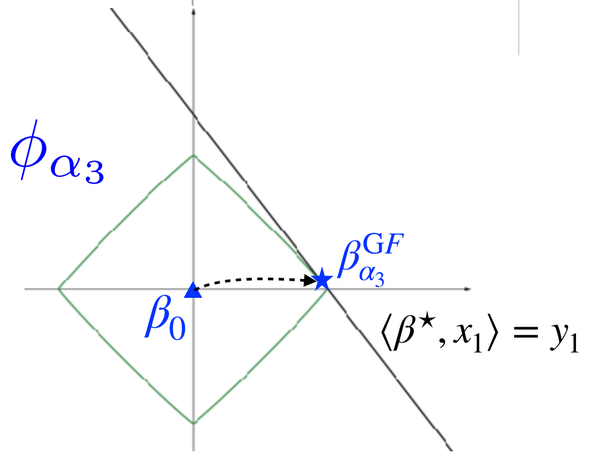
"Medium" initialisation α_2





"Small" initialisation α_3





Impact of the initialisation in "real" deep learning

"Large initialisation"

Lazy / NTK regime = Learning with random features

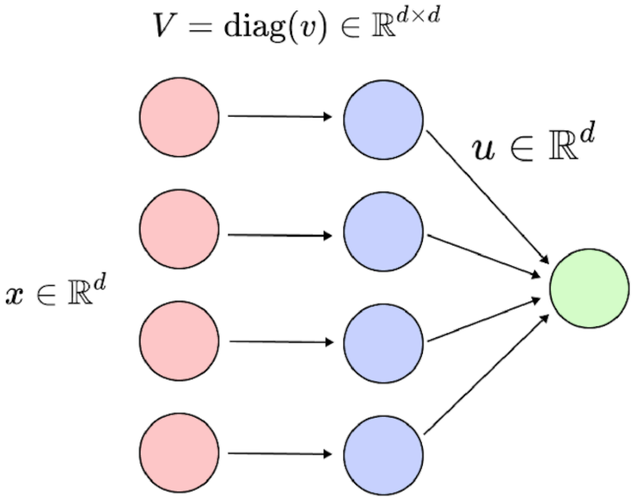
=> Poor generalisation

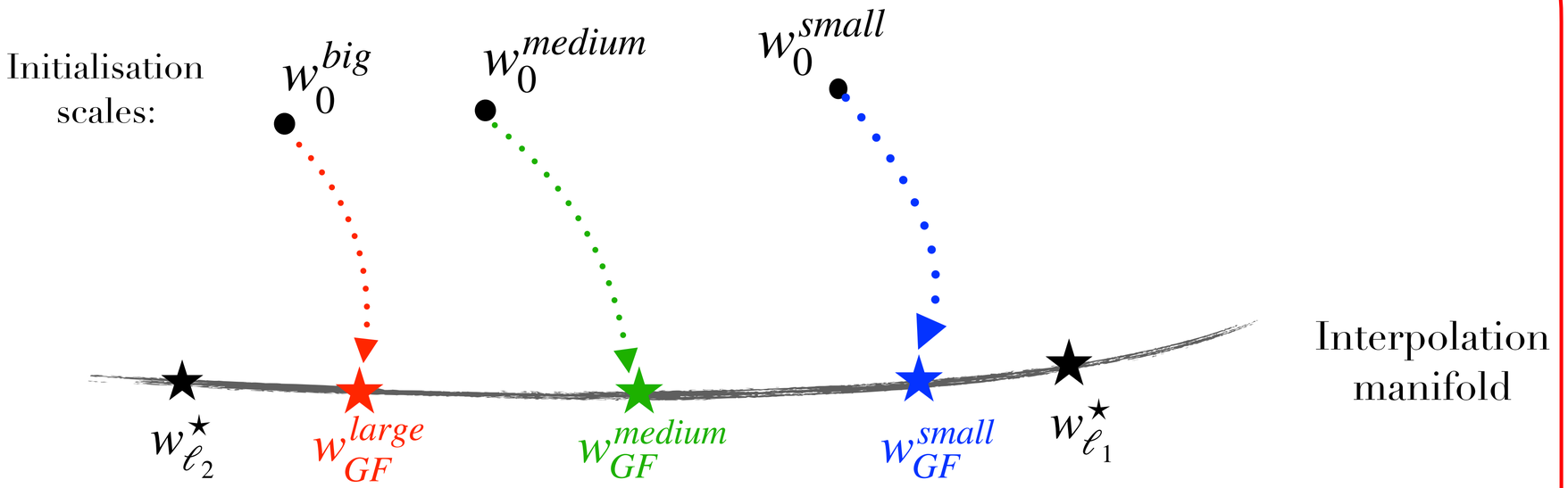
"Small initialisation"

Feature learning / Rich regime = "Features" are learnt

=> good generalisation

Conclusion # 1: Gradient flow and impact of initialisation





What about (S)GD?

‘Real’ deep learning: impact of noise and of the step size.

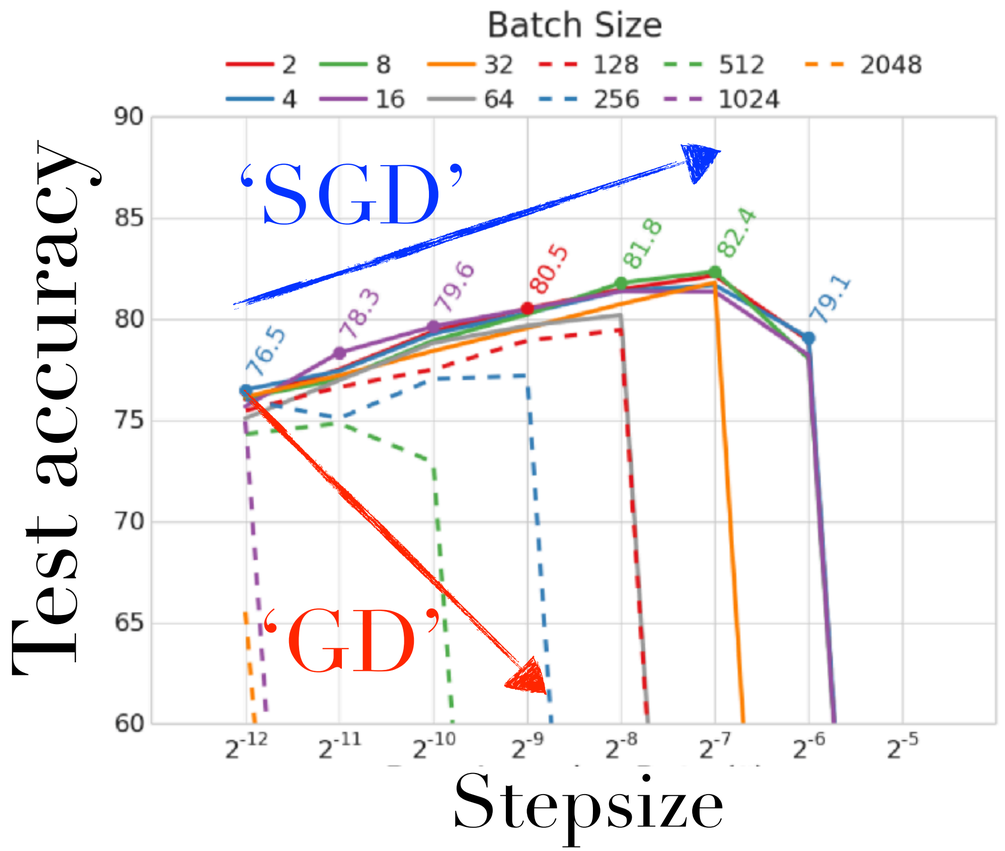
Empirical observations

D. Masters and C. Luschi (2018)

Mini-batch SGD

AlexNet on Cifar 10

↑ Architecture ↑ Dataset



In terms of test performance:

Large stepsize SGD >> “Gradient flow” >> Large stepsize GD

The same tendencies occur when training the diagonal linear network!

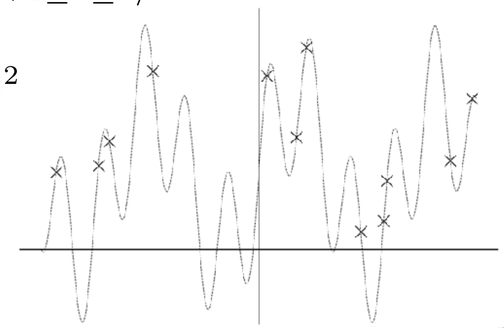
Same setting as before but (S)GD

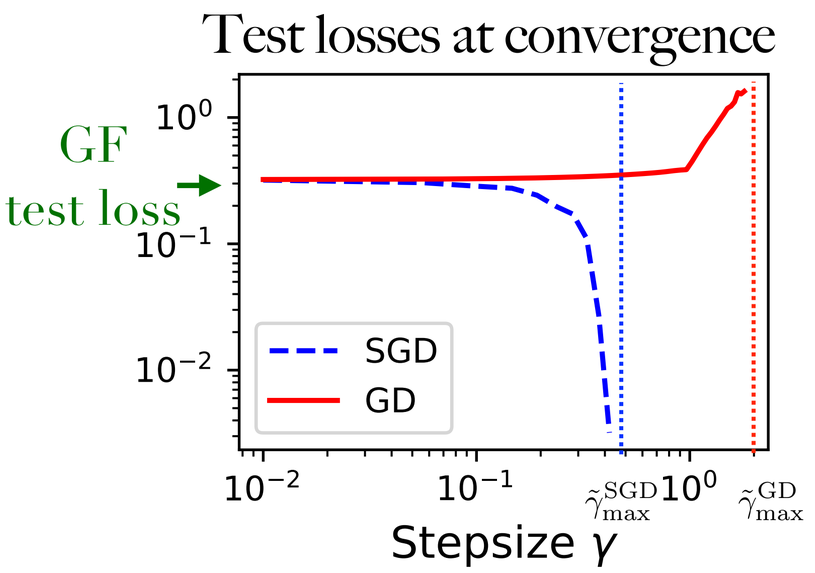
Dataset: $(x_i, y_i)_{1 \le i \le n} \in \mathbb{R} \times \mathbb{R} \qquad (n, d) = (12, 60)$

Feature map: $\varphi(x) = (\cos(\pi k x), \sin(\pi k x))_{0 \le k \le d/2} \in \mathbb{R}^d$

DLN: $F(w) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \langle u \odot v,\ \varphi(x_i) \rangle)^2$

Fixed Init: $u_0^\alpha = \alpha \mathbf{1}$, $v_0^\alpha = \mathbf{0}$





The same tendencies occur when training the diagonal linear network!

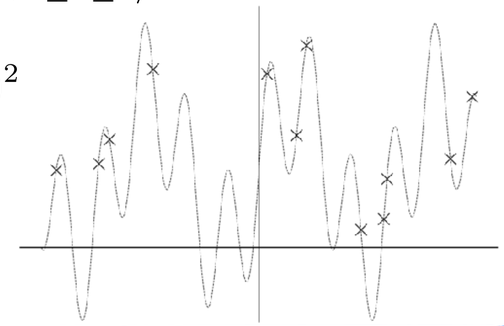
Same setting as before but (S)GD

Dataset: $(x_i, y_i)_{1\leq i\leq n} \in \mathbb{R}\times\mathbb{R}$ $\qquad (n,d) = (12, 60)$

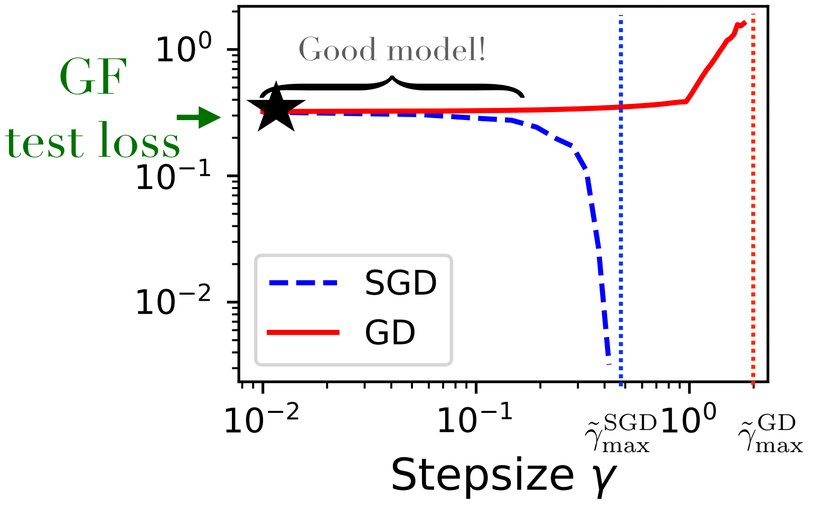
Feature map: $\varphi(x) = (\cos(\pi k x), \sin(\pi k x))_{0\leq k\leq d/2} \in \mathbb{R}^d$

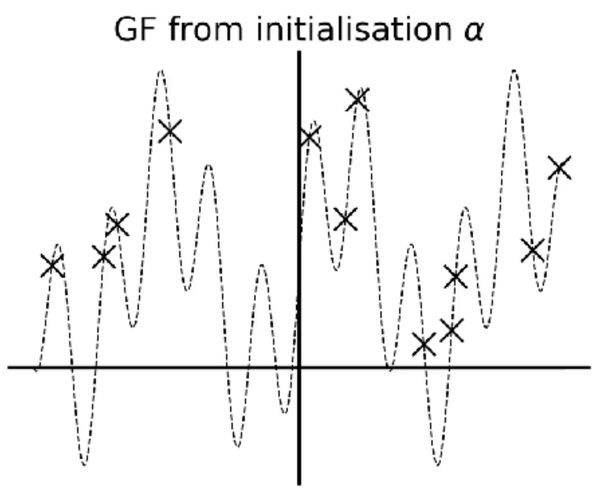
DLN: $F(w) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \langle u\odot v,\ \varphi(x_i)\rangle)^2$

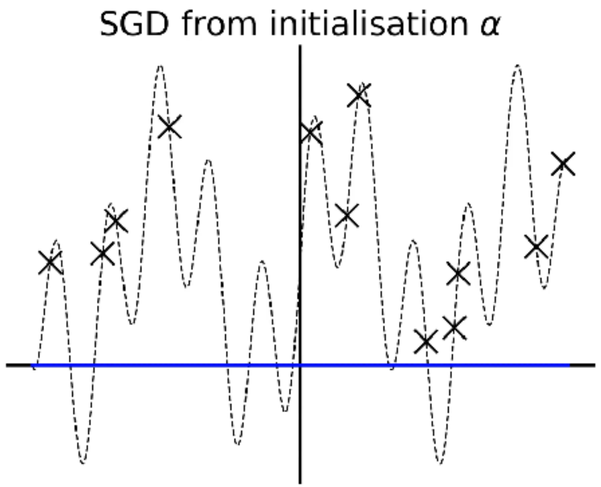
Fixed Init: $u_0^\alpha = \alpha\mathbf{1}$, $v_0^\alpha = \mathbf{0}$

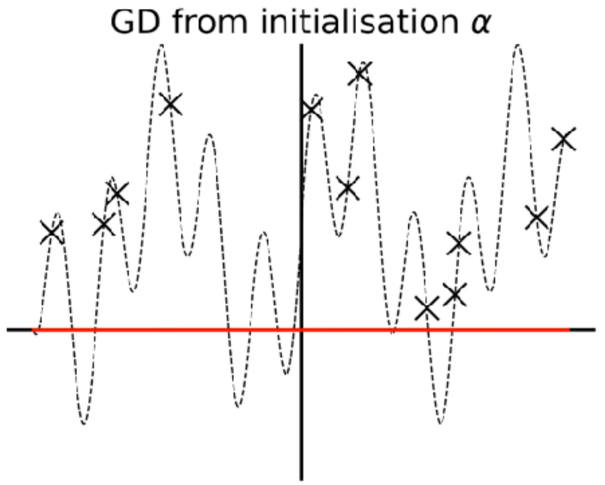


Test losses at convergence









The same tendencies occur when training the diagonal linear network!

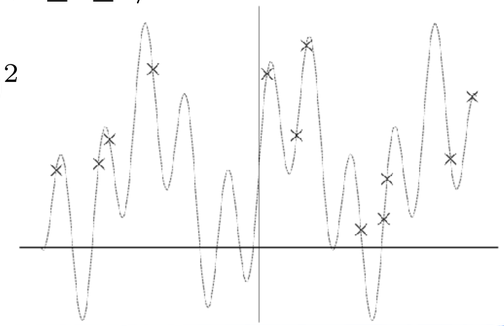
Same setting as before but (S)GD

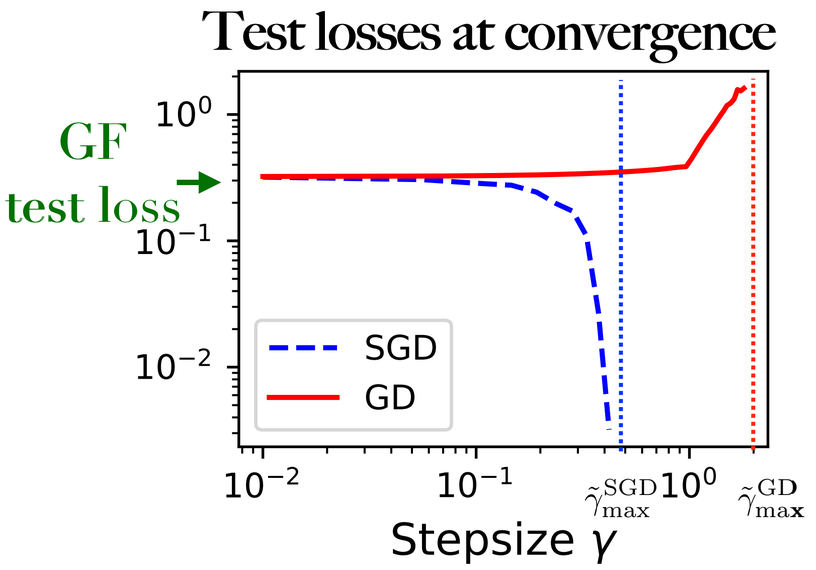
Dataset: $(x_i, y_i)_{1\le i\le n} \in \mathbb{R}\times\mathbb{R}$ $\qquad (n,d) = (12, 60)$

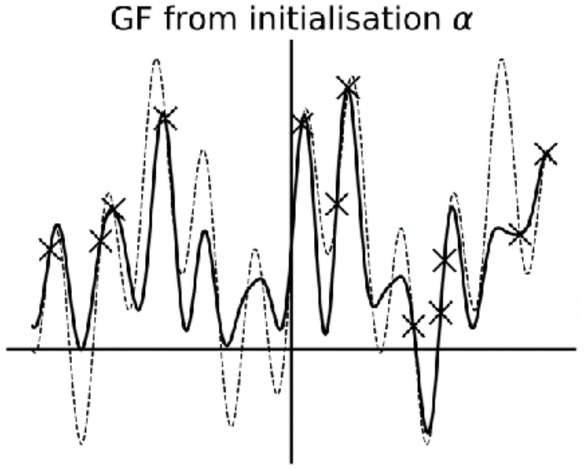
Feature map: $\varphi(x) = (\cos(\pi k x), \sin(\pi k x))_{0\le k\le d/2} \in \mathbb{R}^d$

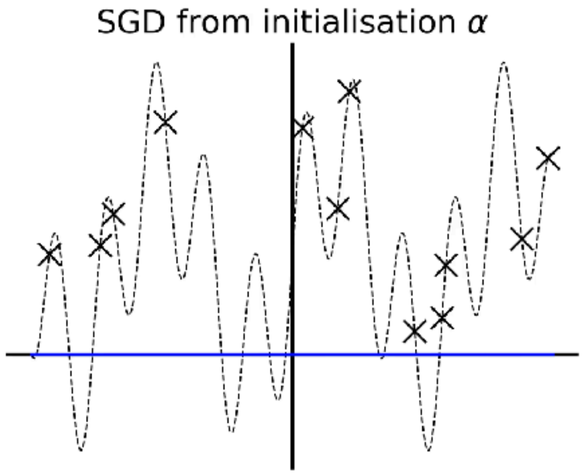
DLN: $F(w) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \langle u\odot v,\ \varphi(x_i)\rangle)^2$

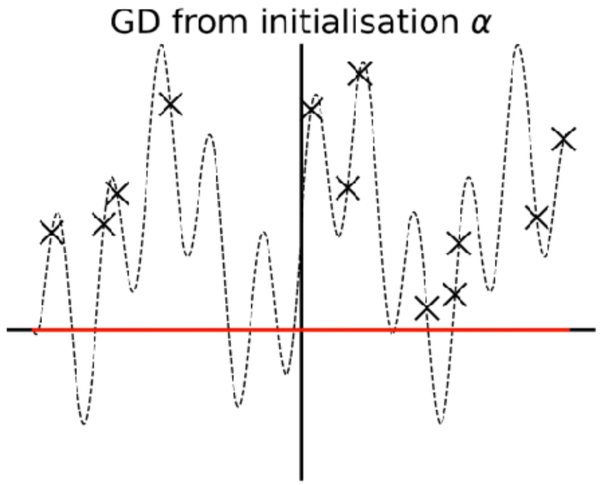
Fixed Init: $u_0^\alpha = \alpha\mathbf{1}$, $v_0^\alpha = \mathbf{0}$











The same tendencies occur when training the diagonal linear network!

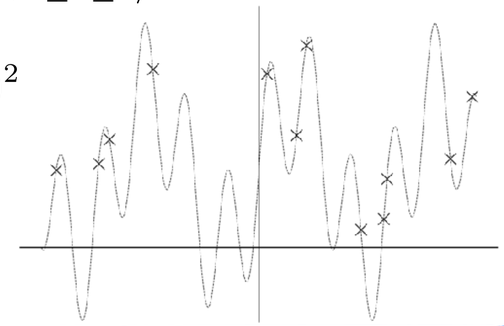
Same setting as before but (S)GD

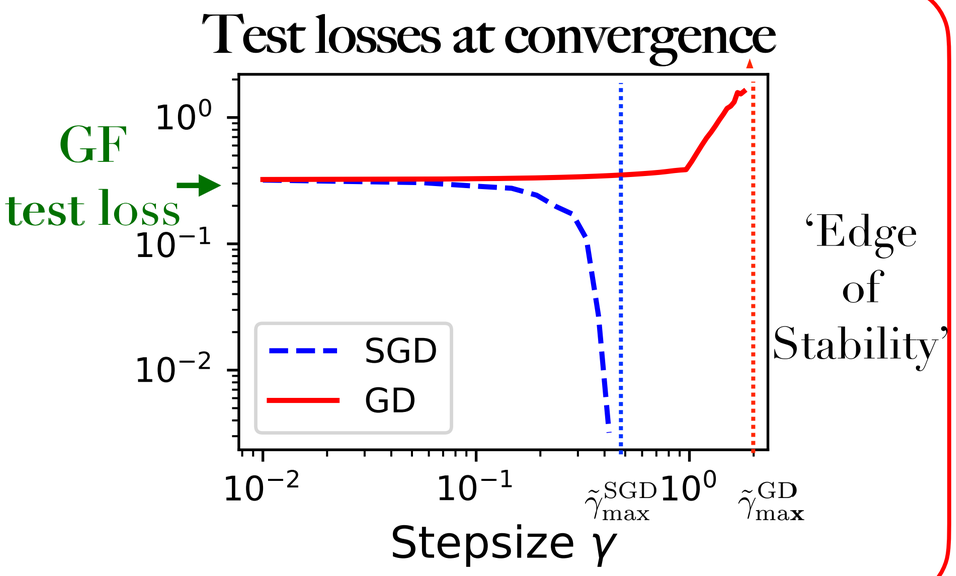
Dataset: $(x_i, y_i)_{1\le i\le n} \in \mathbb{R}\times\mathbb{R}$ $\qquad (n,d) = (12, 60)$

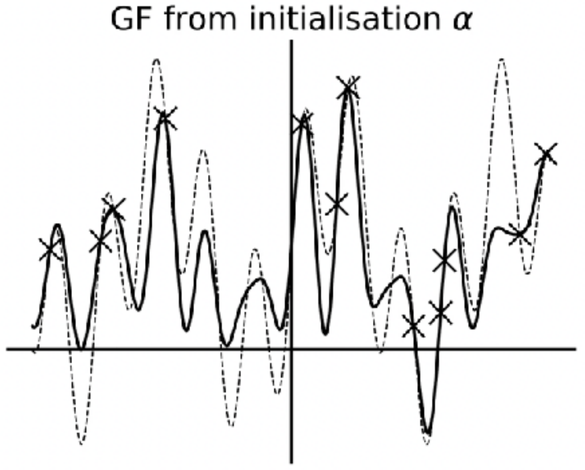
Feature map: $\varphi(x) = (\cos(\pi k x), \sin(\pi k x))_{0\le k\le d/2} \in \mathbb{R}^d$

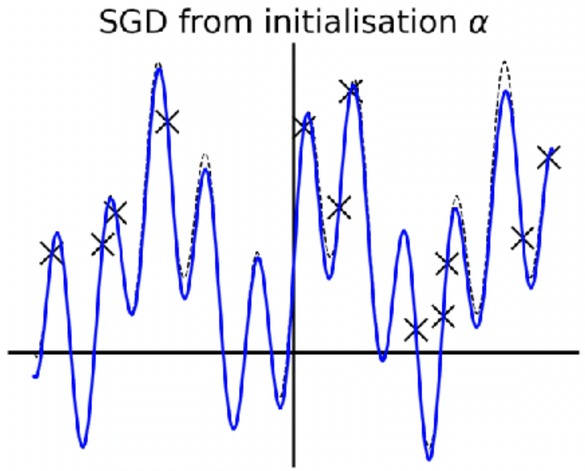
DLN: $F(w) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \langle u\odot v,\ \varphi(x_i)\rangle)^2$

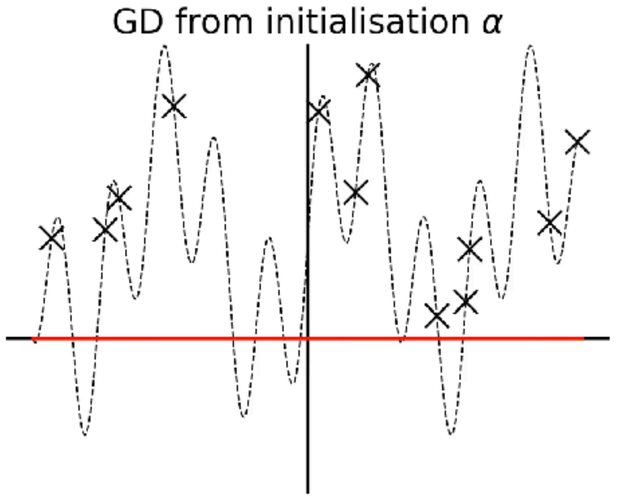
Fixed Init: $u_0^\alpha = \alpha\mathbf{1}$, $v_0^\alpha = \mathbf{0}$

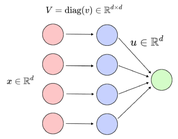












GF

Woodworth et al. 2020

SGF

PPVF 2021

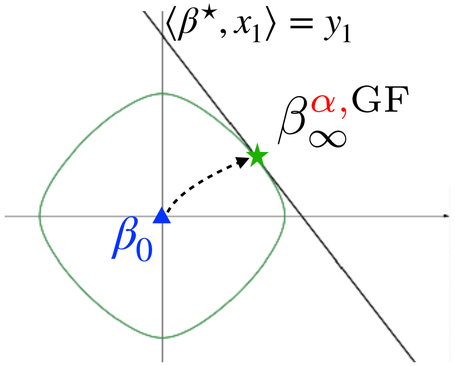
Gradient methods with initialisation $(u_0, v_0) = (\alpha \mathbf{1}, 0)$

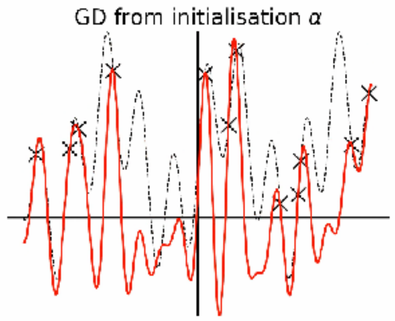
Convergence of the iterates to:

Visual Explanation:

$$\mathrm{d}w_t = -\nabla F(w_t)\mathrm{d}t$$

$$\beta^{\mathrm{GF}} = \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \ \phi_\alpha(\beta^\star)$$



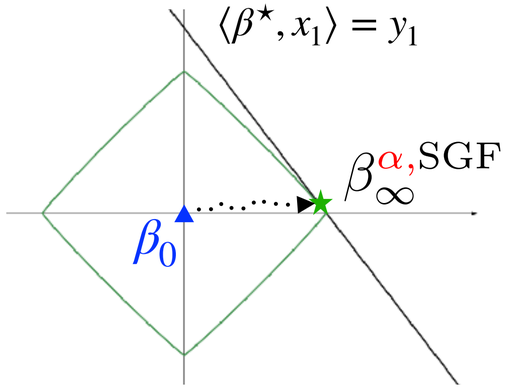


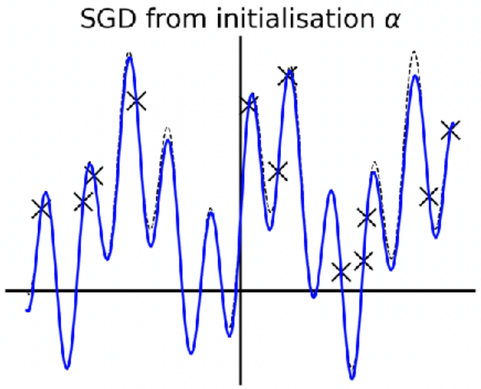
$$\mathrm{d}u_t = -\nabla_u F(w_t)\mathrm{d}t + 2\sqrt{\gamma L(\beta_t)} v_t \odot [X^\top \mathrm{d}B_t]$$

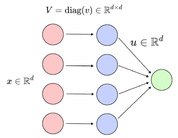
$$+2\sqrt{\gamma}\mathrm{d}B_t$$

$$\beta^{\mathrm{SGF}} = \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \ \phi_{\alpha_\infty}(\beta^\star)$$

$$\alpha_\infty = \alpha \exp\left(-\gamma \mathrm{diag}(X^\top X) \int_0^{+\infty} L(\beta_s)\mathrm{d}s \right)$$
$$< \alpha$$







Gradient method with initialisation $(u_0, v_0) = (\alpha \mathbf{1}, 0)$

Convergence of the iterates to:

GF

Woodworth et al. 2020

$$\mathrm{d}w_t = -\nabla F(w_t)\mathrm{d}t \quad \longrightarrow \quad \beta^{\mathrm{GF}} = \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \ \phi_\alpha(\beta^\star)$$

SGF

PPVF 2021

$$\mathrm{d}u_t = -\nabla_u F(w_t)\mathrm{d}t + 2\sqrt{\gamma L(\beta_t)} v_t \odot [X^\top \mathrm{d}B_t]$$

$$+2\sqrt{\gamma}\mathrm{d}B_t$$

$$\longrightarrow \quad \beta^{\mathrm{SGF}} = \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \ \phi_{\alpha_\infty}(\beta^\star)$$

$$\alpha_\infty = \alpha \exp\left(-\gamma \mathrm{diag}\left(X^\top X\right) \int_0^{+\infty} L(\beta_s)\mathrm{d}s \right)$$

(S)GD

EPGF 2023

$$w_{k+1} = w_k - \gamma \nabla F_{\mathcal{B}_k}(w_k) \quad \longrightarrow \quad \beta^{(\mathrm{S})\mathrm{GD}} = \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \ D_{\phi_{\alpha_\infty}}(\beta^\star, \tilde{\beta}_0)$$

$$\alpha_\infty = \alpha \exp\left(-\sum_{k=0}^{\infty} q(\gamma \nabla L_{\mathcal{B}_k}(\beta_k)) \right)$$

$$q(z) = -\frac{1}{2}\ln((1-z^2)^2) \underset{z\to 0}{\sim} z^2$$

MGF

PPF 2024

$$\lambda \ddot{w}_t + \dot{w}_t + \nabla F(w_t) = 0 \ \text{ where } \lambda = \gamma/(1-\beta)^2$$

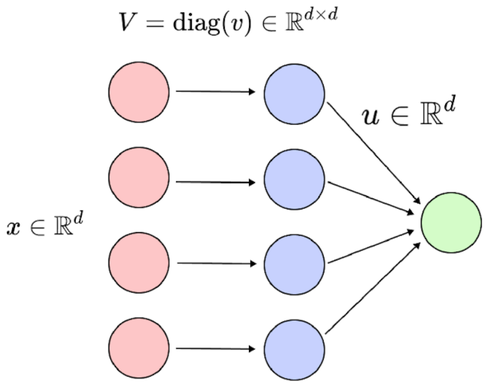
$$\left(w_{k+1} = w_k - \gamma \nabla F(w_k) + \beta(w_k - w_{k-1})\right)$$

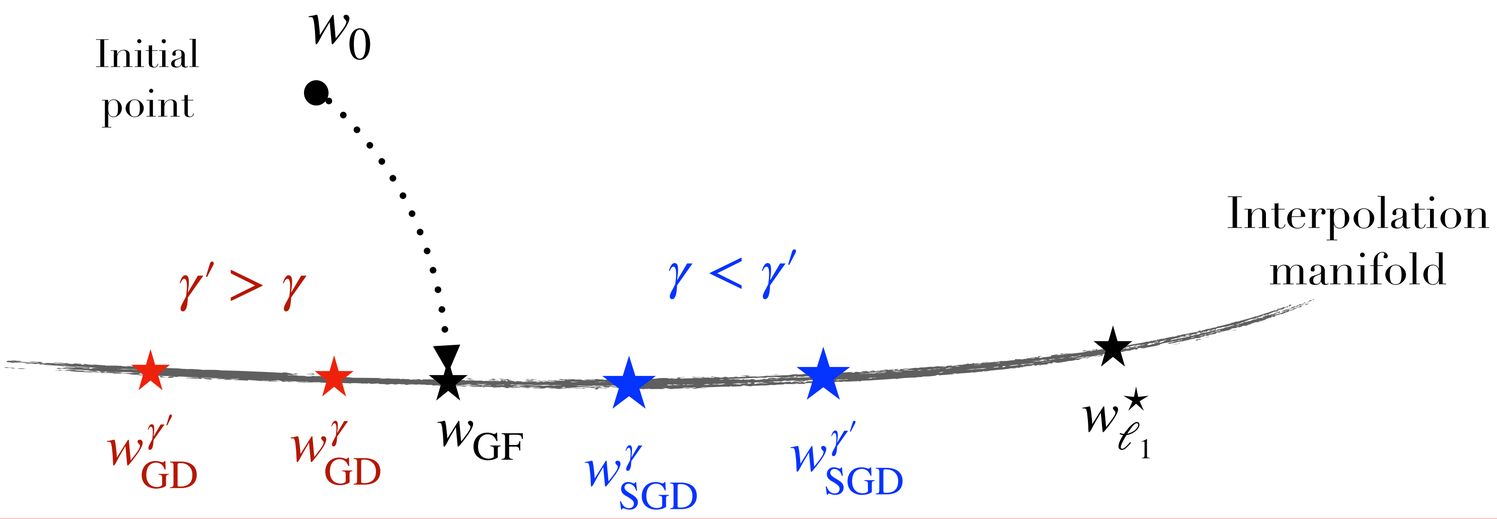
$$\longrightarrow \quad \beta^{\mathrm{MGF}} = \underset{y_i = \langle x_i, \beta^\star \rangle, \forall i}{\arg\min} \ D_{\phi_{\alpha_\infty}}(\beta^\star, \tilde{\beta}_0)$$

$$\alpha_\infty = \alpha \exp\left(-\int(\cdots) \right)$$

Conclusion # 2: Impact of hyper parameters

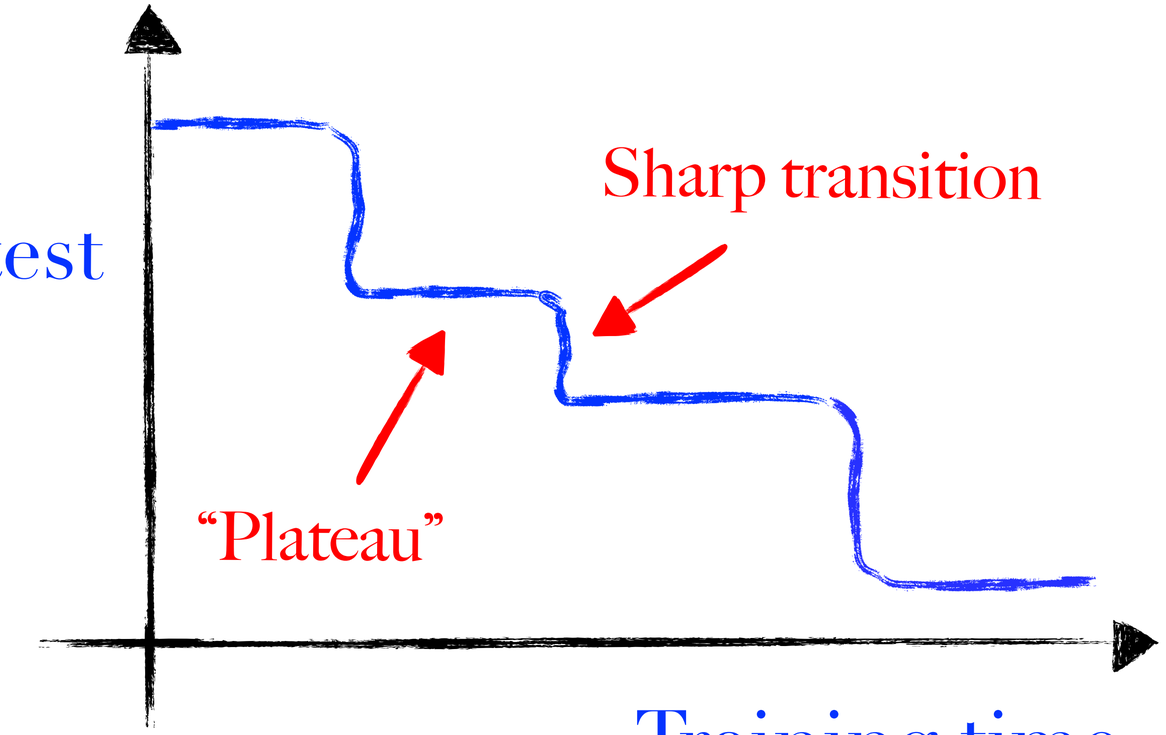
Impact of hyper parameters on the recovered solution





What can we say about the full trajectory?

Some peculiar training plots



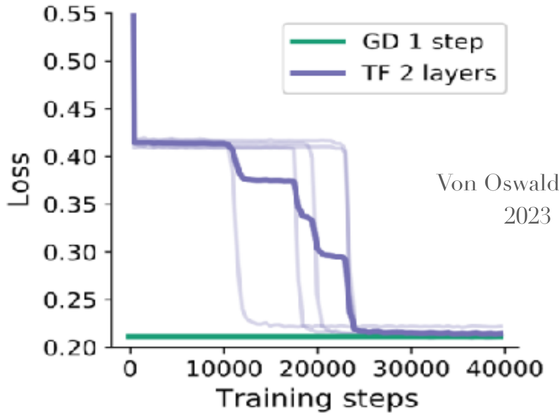
Gradient method on a neural net

No change of stepsize

The learning curve is piecewise constant!

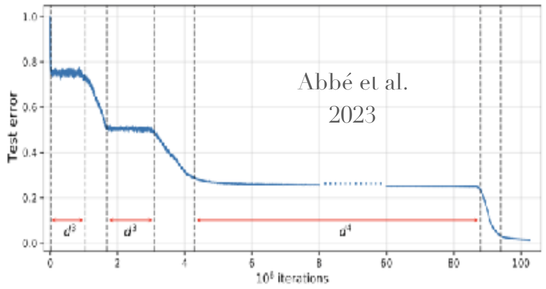
Training various architectures with small initialisation of the weights

GD over a transformer



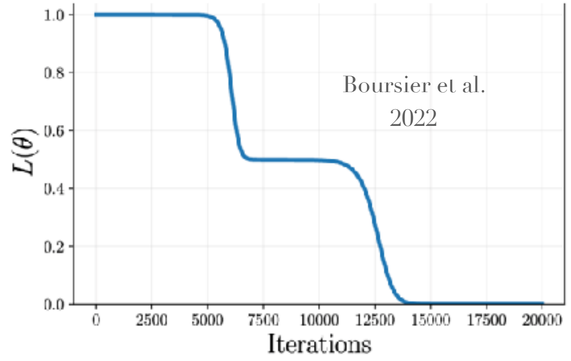
Von Oswald Et al.
2023

SGD over a 2-layer network with sigmoid activation



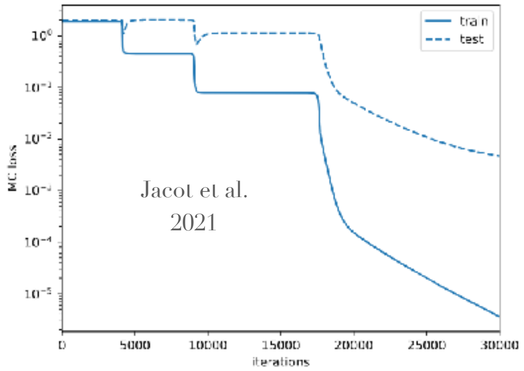
Abbé et al.
2023

GD over a 2-layer ReLU network



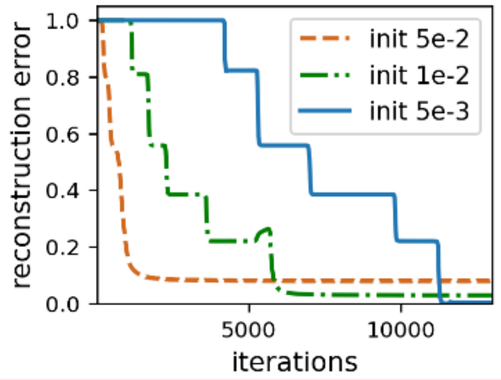
Boursier et al.
2022

GD over a deep linear network



Jacot et al.
2021

GD over a tensor factorisation problem



Razin et al.
2021

Why small initialisation?

Regime where networks perform well:
"rich regime" / "feature learning"

"Opposite" of the NTK regime
(Lazy regime)

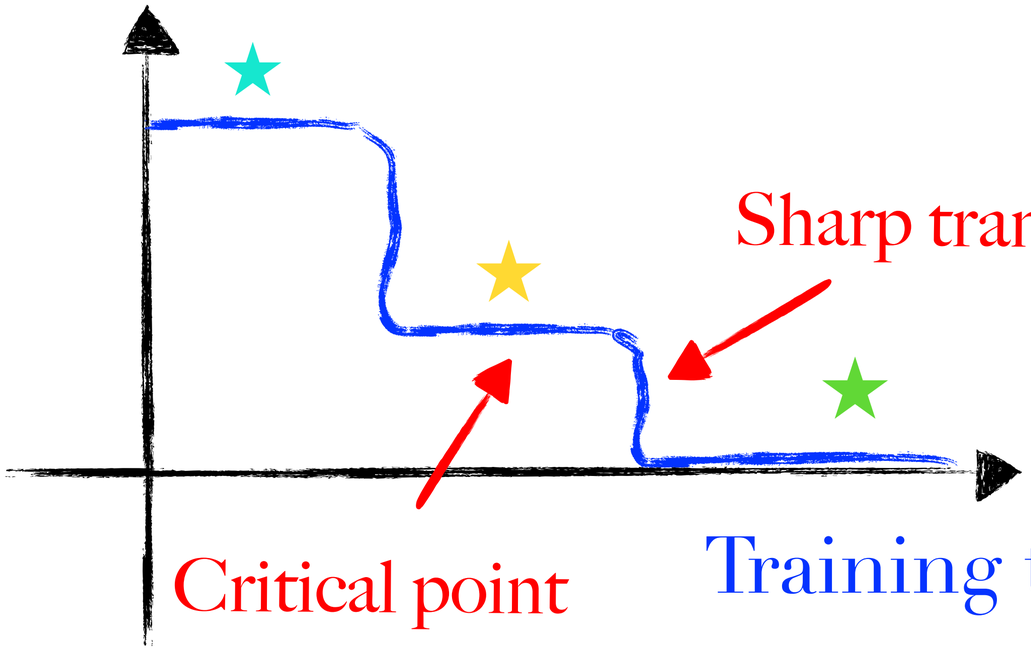
Each transition corresponds to "learning a feature"

What's (intuitively) happening?

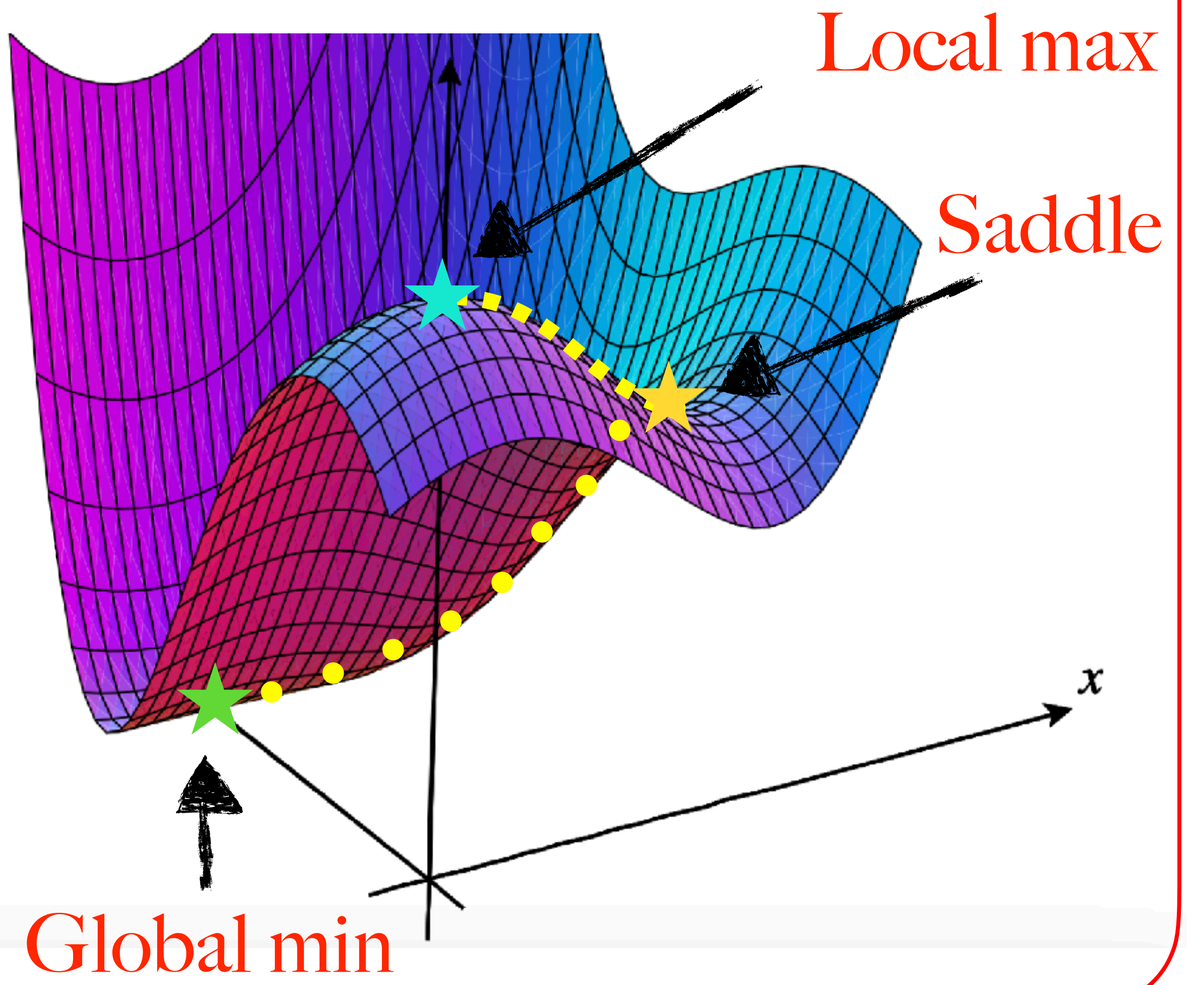
(S)GD over a loss F:

$$w_{k+1} = w_k - \gamma \nabla F(w_k)$$

Train loss



Loss surface plot $(F : \mathbb{R}^2 \to \mathbb{R})$



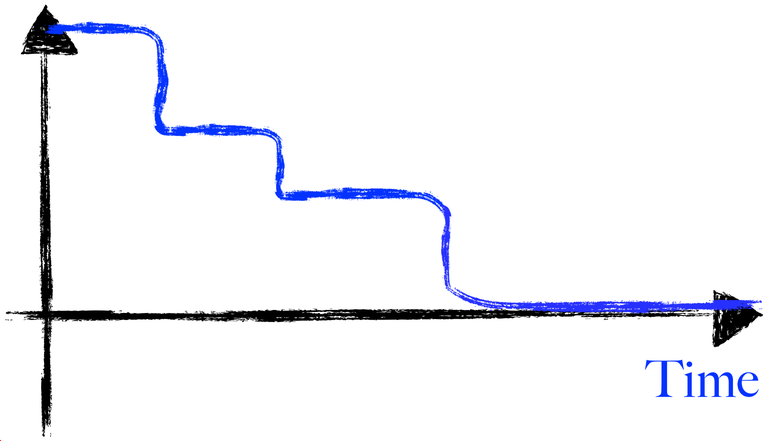
Training a 2-layer diagonal linear network with GD and small initialisation

Loss: $F(u,v) = L(u \odot v) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \langle u \odot v,\ x_i\rangle)^2$

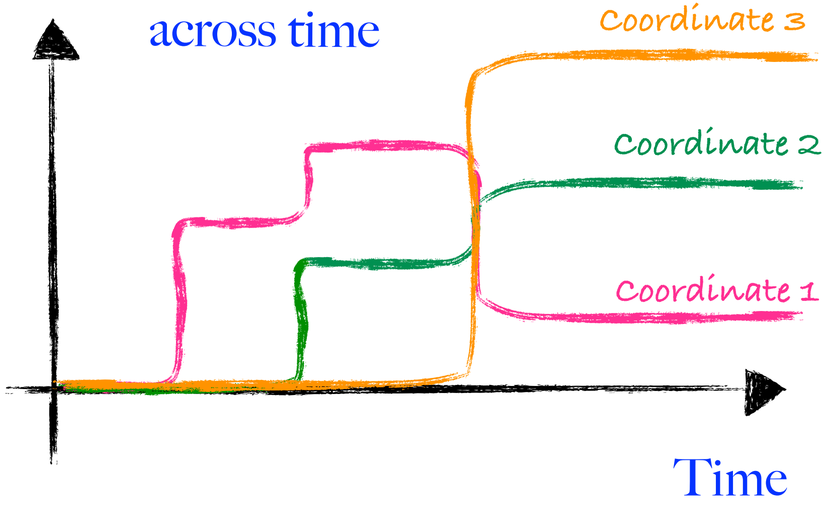
GD on weights: $(u_{k+1}, v_{k+1}) = (u_k, v_k) - \gamma \nabla F(u_k, v_k)$

Predictors: $\beta_k = u_k \odot v_k$

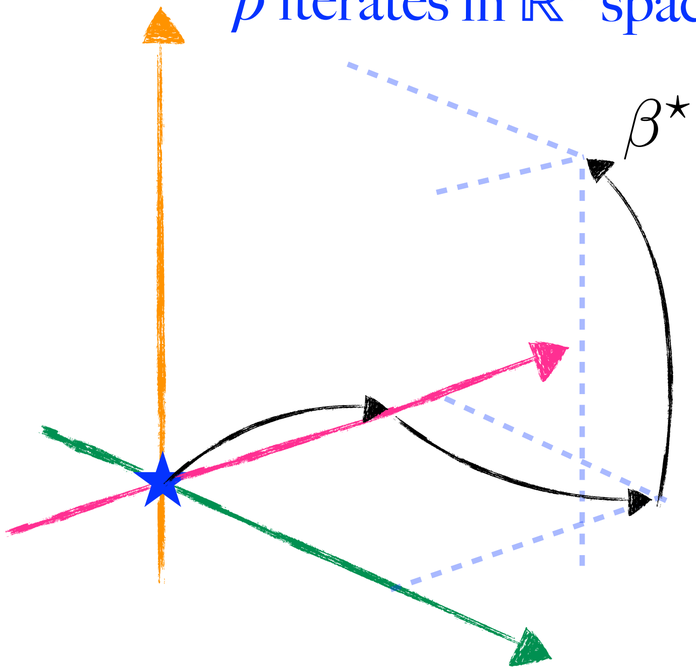
Train or test loss across time



Coordinates across time



β iterates in $\mathbb{R}^d$ space



"Simple" setting where the saddle-to-saddle process occurs!

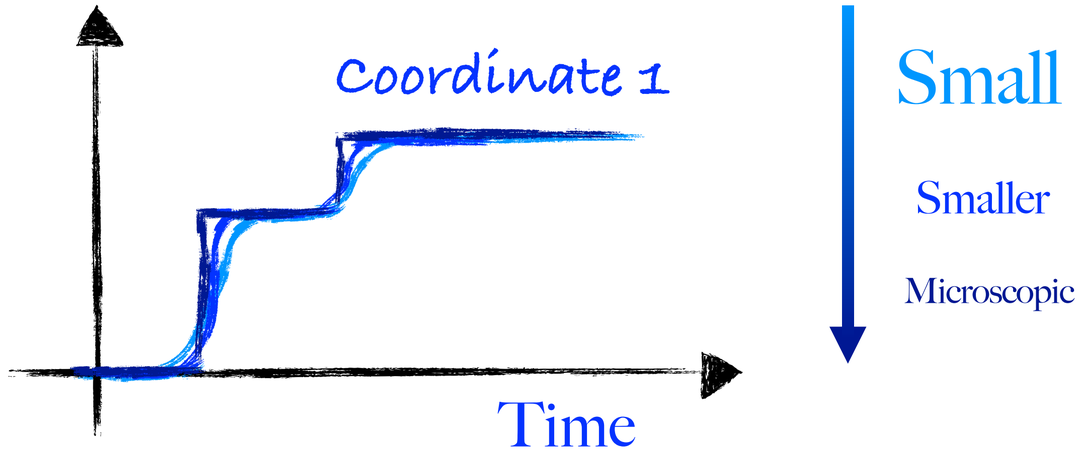
Coordinates activate successively

Let's prove this!

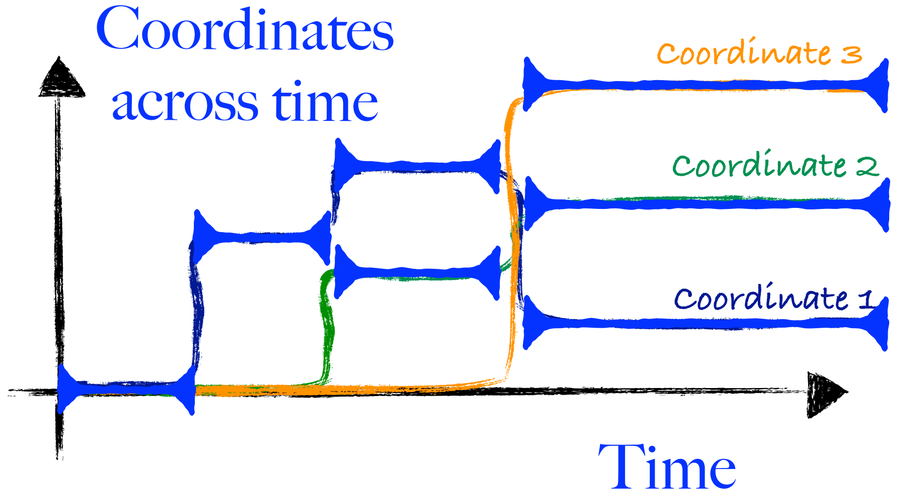
Informal Theorem:

For gradient flow, as the initialisation of the weights goes to 0 :

1. The (accelerated) iterates converge to a piecewise constant function



2. The visited saddles and jump times are given by an algorithm



Inverse scale space algorithm which describes the visited saddles and the jump times

Algorithm 1: Successive saddles and jump times of $\lim_{\alpha \to 0} \tilde{\beta}^{\alpha}$

Initialise: $(t, \beta, s) \leftarrow (0, \mathbf{0}, \mathbf{0})$;

while $\nabla L(\beta) \neq \mathbf{0}$ **do**

$$\mathcal{A} \leftarrow \{j \in [d], \nabla L(\beta)(j) \neq 0\}$$

Set of coordinates "which are unstable"

$$\Delta \leftarrow \inf \{\delta > 0 \text{ s.t. } \exists i \in \mathcal{A},\ s(i) - \delta \nabla L(\beta)(i) = \pm 1\}$$

Time spent on the saddle

$$(t,\ s) \leftarrow (t + \Delta,\ s - \Delta \cdot \nabla L(\beta))$$

$$\beta \leftarrow \arg\min\ L(\beta)\ \text{ where }\ \beta \in \left\{ \beta \in \mathbb{R}^d \ \text{ s.t. } \begin{array}{l} \beta_i \geq 0 \text{ if } s(i) = +1 \\ \beta_i \leq 0 \text{ if } s(i) = -1 \\ \beta_i = 0 \text{ if } s(i) \in (-1,1) \end{array} \right\}$$

Next saddle: least square solution (under sign constraints)

end

Output: Successive values of β and t

Burger, Möller, Benning and Osher, An adaptive inverse scale space method for compressed sensing *Mathematics of Computation*, 2013

Key insight:

Next visited saddle
$\neq$
Direction of most negative curvature of F

Related works

Various works in various settings, but basically assume that $\frac{1}{n}X^\top X \sim I_d$

- S. Arora, N. Cohen, W. Hu, and Y. Luo. *Implicit regularization in deep matrix factorization*. NeurIPS 2019.
- Z. Li, Y. Luo, and K. Lyu. *Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning*. ICLR 2021
- J. Jin, Z. Li, K. Lyu, S. S Du, and J. D Lee. *Understanding Incremental Learning of Gradient Descent: A Fine-grained Analysis of Matrix Sensing*. ICML 2023

Closest work but stronger assumptions on the data:

- R. Berthier. *Incremental learning in diagonal linear networks*. JMLR 2023

Extension to general diagonal architectures $f_{NN}(x, w) = h(x, u \odot v)$:

- E. Boix-Adsera, E. Littwin, E. Abbe, S. Bengio, and J. Susskind. *Transformers learn through gradual rank increase*. NeurIPS 2023

Under a RIP assumption, clear and simple incremental learning

Under a Restricted Isometry Property with a sparse gold predictor $\beta^\star = (\beta_1^\star, \dots, \beta_r^\star, 0, \dots, 0)$

$$\beta_1^\star > \dots > \beta_r^\star > 0$$

$$\text{“} \frac{1}{n} X^\top X \sim I_d \text{”}$$

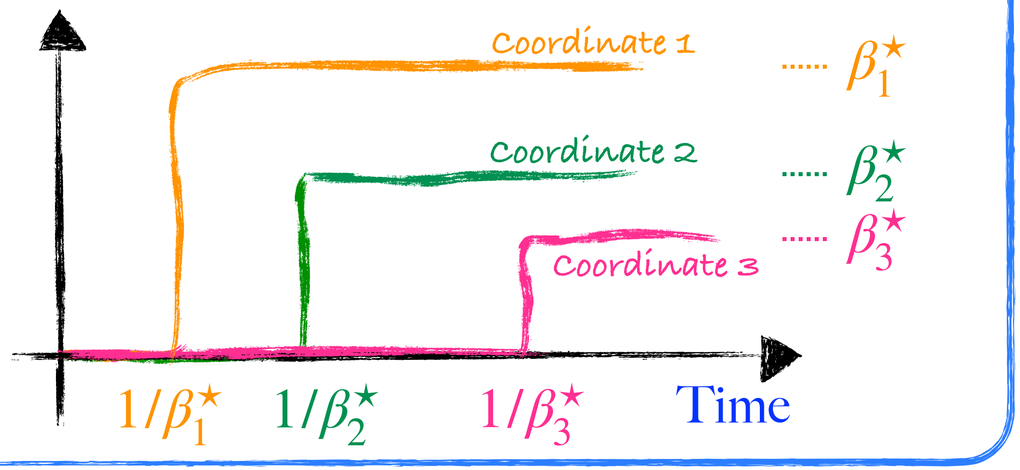
Then the algorithm simply becomes:

Exactly r saddles are visited, they are

$$\beta_i \approx (\beta_1^\star, \dots, \beta_i^\star, 0, \dots, 0)$$

The jump times are $t_i \approx 1/|\beta_i^\star|$

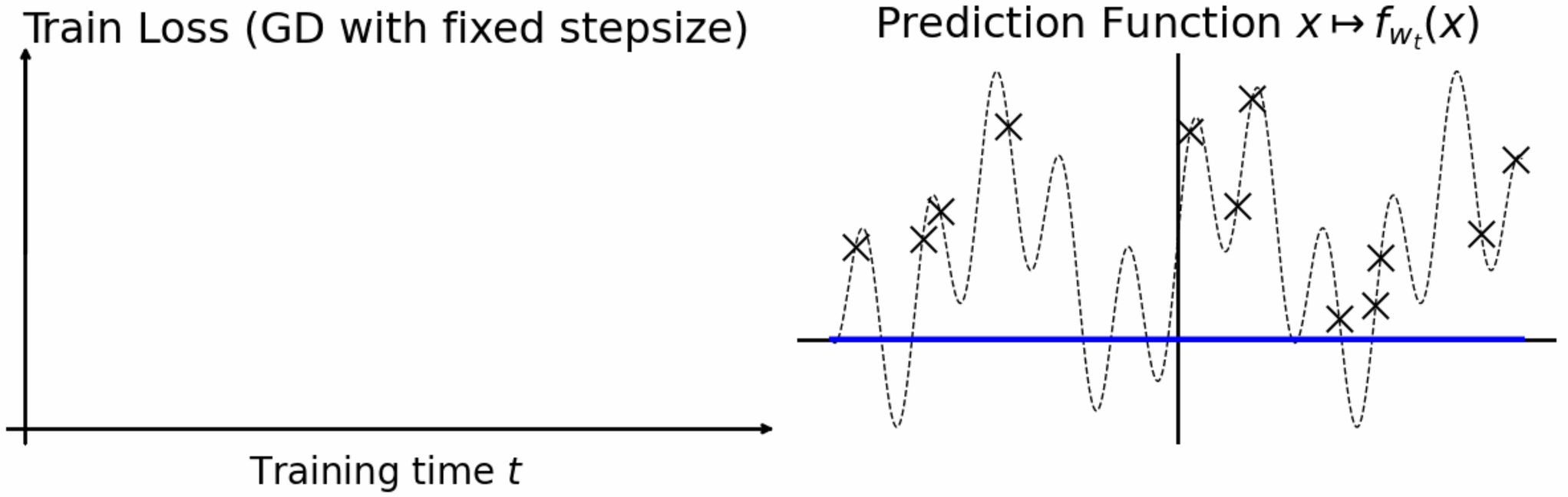
In picture:

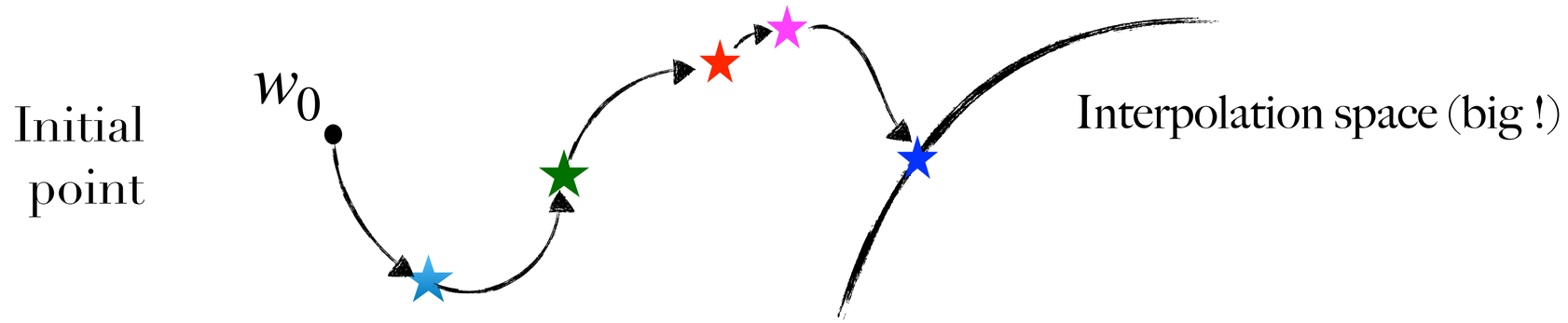


Incremental Learning

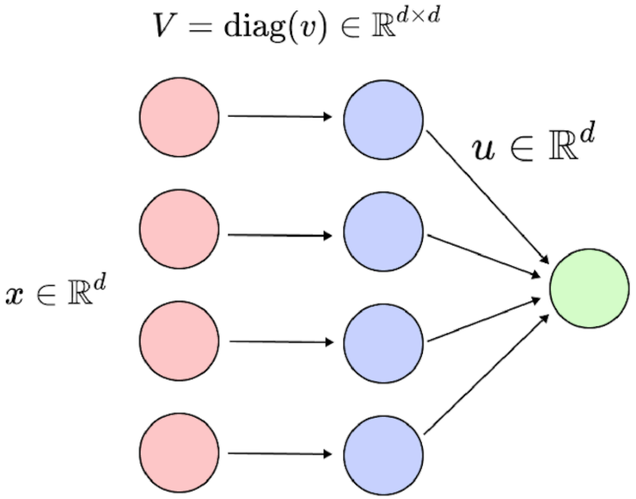
Feature map: $\varphi(x) = (\cos(\pi k x), \sin(\pi k x))_{0 \leq k \leq d/2} \in \mathbb{R}^d$ **DLN:** $F(w) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \langle u \odot v,\ \varphi(x_i) \rangle)^2$

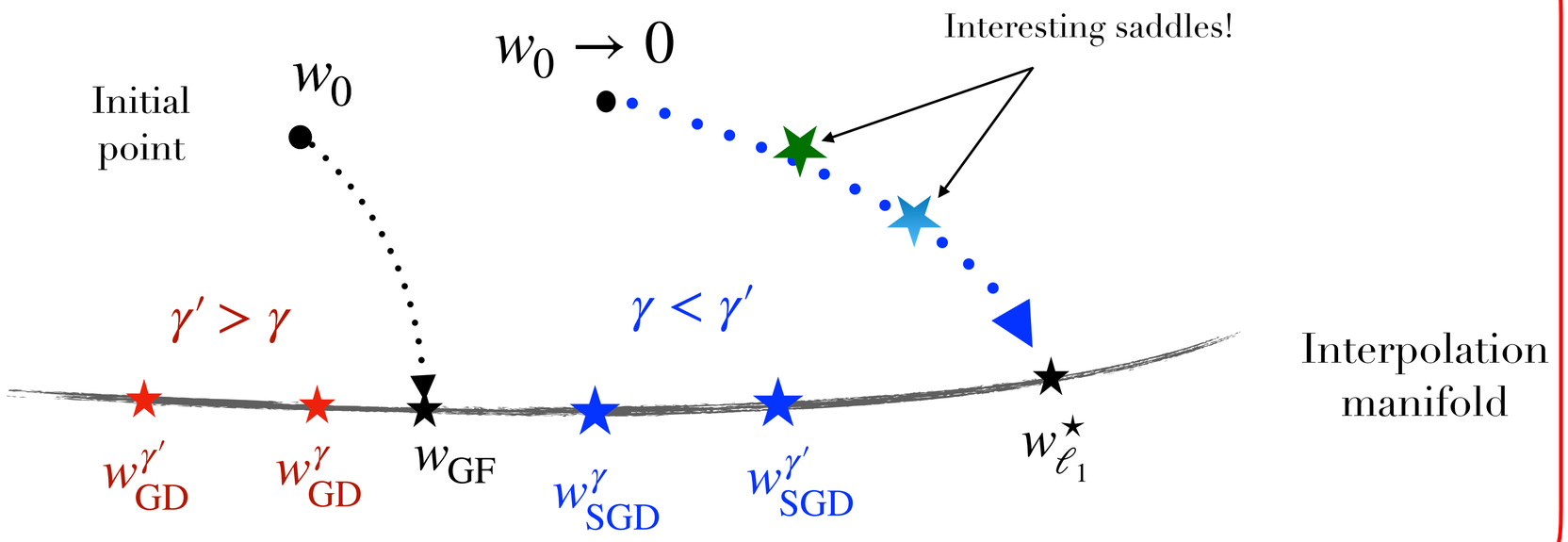
GD with small initialisation **Prediction functions:** $f_{w_t} : x \mapsto \langle u_t \odot v_t,\ \varphi(x) \rangle$





Conclusion # 3: Trajectory description





Regression!

Story is different for classification

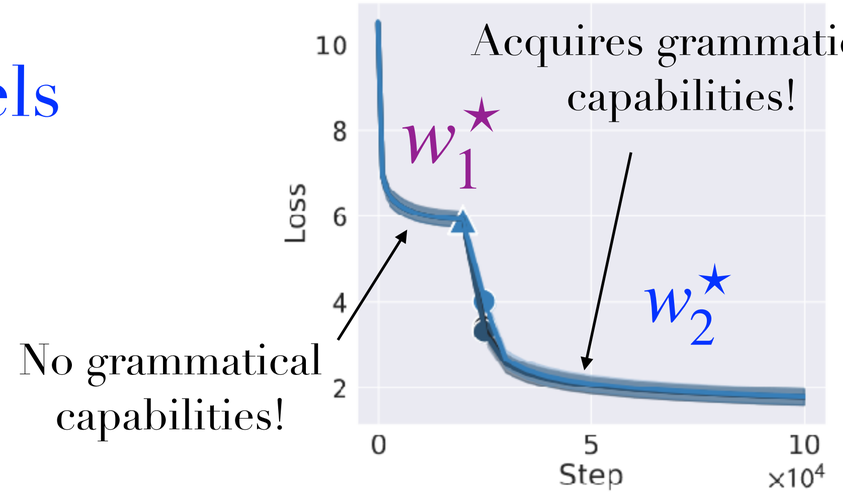
Understanding the training trajectories is crucial to understand (modern!) deep learning

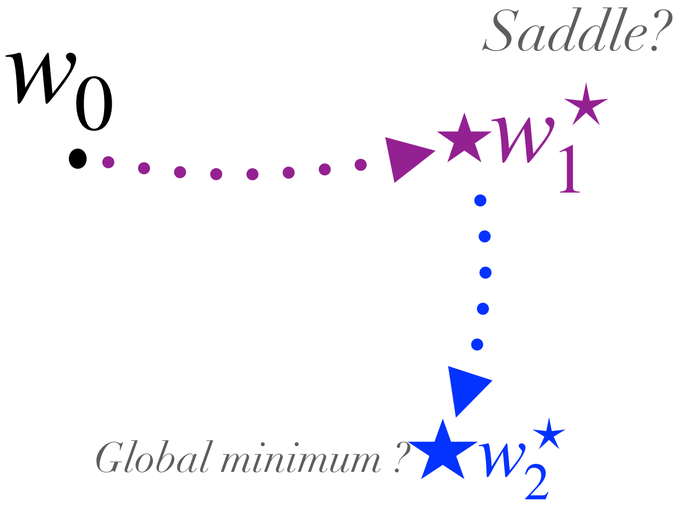
Empirical Observations

Large language models

Text corpus
Bert transformer model

Chen et al. ICLR 2024





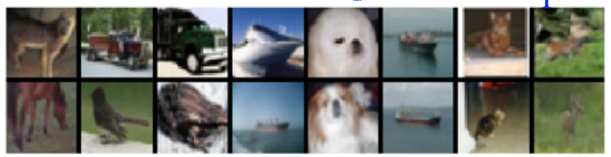
Generative Models

EDM
(diffusion model)

Gu et al. TMLR 2025

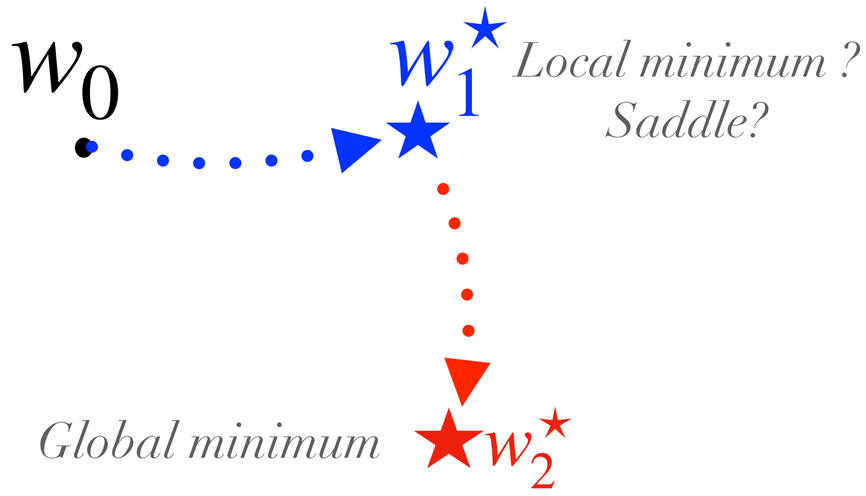
Generalisation capabilities $w_1^\star$

Images générées:
Données d'entraînement:



Harmful overfitting after long training $w_2^\star$

Images générées:
Données d'entraînement:



(Old school) Deep Learning Theory Through the Lens of Diagonal Linear Networks

Loucas Pillaud-Vivien
Assistant professor, Ecole des Ponts

Scott Pesme,
PostDoc at Inria Grenoble with Julien Mairal

Mathieu Even
PostDoc, Inria Montpellier

Radu-Alexandru Dragomir
Assistant professor, Telecom Paris

Nicolas Flammarion
Assistant professor, EPFL

Hristo Papazov
PhD, EPFL

Suriya Gunasekar
Researcher, Microsoft Redmond