Dynamical aspects of Deep Learning

Yacine Chitour
Joint work with Z. Liao and R. Couillet
DEUSTO-TECH, Bilbao

L2S, CentraleSupélec
Université Paris-Saclay
Paris, France

January 21, 2019

CentraleSupélec

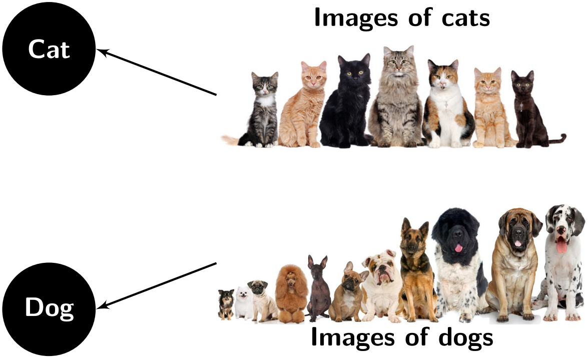
Outline

Outline

Motivation: "learn" to automatically classify images

Machine Learning:

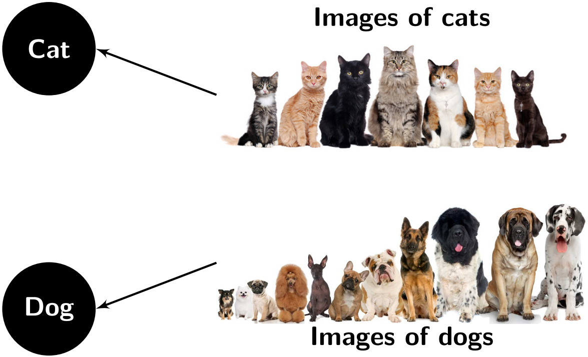
- given m images of cats $x_1^{cat}, x_2^{cat}, \ldots, x_m^{dog}$ and dogs $x_1^{dog}, x_2^{dog}, \ldots, x_m^{dog}$ of labels y_{cat} and y_{dog} ($y_{cat} \neq y_{dog}$), respectively.



Motivation: "learn" to automatically classify images

Machine Learning:

- given m images of cats $x_1^{cat}, x_2^{cat}, \ldots, x_m^{dog}$ and dogs $x_1^{dog}, x_2^{dog}, \ldots, x_m^{dog}$ of labels y_{cat} and y_{dog} $(y_{cat} \neq y_{dog})$, respectively.



- **Goal**: for a new image $x_{new}^{?}$ with $? \in \{cat, dog\}$, predict ?=cat or ?=dog.

How to "learn" to classify?

Learning phase: find W that minimizes $\sum_{i,j} \|y_{cat} - Wx_i^{cat}\|^2 + \|y_{dog} - Wx_j^{dog}\|^2$, where $x_i^{cat}, x_j^{dog} \in \mathbb{R}^{d_x}$ and $y_{cat}, y_{dog} \in \mathbb{R}^{d_y}$ **chosen by user**.

How to “learn” to classify?

Learning phase: find W that minimizes $\sum_{i,j} \|y_{cat} - Wx_i^{cat}\|^2 + \|y_{dog} - Wx_j^{dog}\|^2$, where $x_i^{cat}, x_j^{dog} \in \mathbb{R}^{d_x}$ and $y_{cat}, y_{dog} \in \mathbb{R}^{d_y}$ **chosen by user**.

- **Input**: (X, Y), images $X = [x_1^{cat}, \dots, x_1^{dog}, \dots] \in \mathbb{R}^{d_x \times 2m}$ with associated labels $Y = [y_{cat}, \dots, y_{dog}, \dots] \in \mathbb{R}^{d_y \times 2m}$.

How to "learn" to classify?

Learning phase: find W that minimizes $\sum_{i,j} \|y_{cat} - Wx_i^{cat}\|^2 + \|y_{dog} - Wx_j^{dog}\|^2$, where $x_i^{cat}, x_j^{dog} \in \mathbb{R}^{d_x}$ and $y_{cat}, y_{dog} \in \mathbb{R}^{d_y}$ **chosen by user**.

- **Input**: (X, Y), images $X = [x_1^{cat}, \ldots, x_1^{dog}, \ldots] \in \mathbb{R}^{d_x \times 2m}$ with associated labels $Y = [y_{cat}, \ldots, y_{dog}, \ldots] \in \mathbb{R}^{d_y \times 2m}$.
- **Output**: $W \in \mathbb{R}^{d_y \times d_x}$ that minimizes the difference $\|Y - WX\|^2$

How to “learn” to classify?

Learning phase: find W that minimizes $\sum_{i,j} \|y_{cat} - Wx_i^{cat}\|^2 + \|y_{dog} - Wx_j^{dog}\|^2$, where $x_i^{cat}, x_j^{dog} \in \mathbb{R}^{d_x}$ and $y_{cat}, y_{dog} \in \mathbb{R}^{d_y}$ **chosen by user**.

- **Input**: (X, Y), images $X = [x_1^{cat}, \dots, x_1^{dog}, \dots] \in \mathbb{R}^{d_x \times 2m}$ with associated labels $Y = [y_{cat}, \dots, y_{dog}, \dots] \in \mathbb{R}^{d_y \times 2m}$.
- **Output**: $W \in \mathbb{R}^{d_y \times d_x}$ that minimizes the difference $\|Y - WX\|^2$

Prediction phase: for a new image $x_{new}^?$ (of unknown true label $x_{new}^?$), predicts:

How to "learn" to classify?

Learning phase: find W that minimizes $\sum_{i,j} \|y_{cat} - Wx_i^{cat}\|^2 + \|y_{dog} - Wx_j^{dog}\|^2$, where $x_i^{cat}, x_j^{dog} \in \mathbb{R}^{d_x}$ and $y_{cat}, y_{dog} \in \mathbb{R}^{d_y}$ **chosen by user**.

- **Input**: (X, Y), images $X = [x_1^{cat}, \dots, x_1^{dog}, \dots] \in \mathbb{R}^{d_x \times 2m}$ with associated labels $Y = [y_{cat}, \dots, y_{dog}, \dots] \in \mathbb{R}^{d_y \times 2m}$.
- **Output**: $W \in \mathbb{R}^{d_y \times d_x}$ that minimizes the difference $\|Y - WX\|^2$

Prediction phase: for a new image $x_{new}^?$ (of unknown true label $x_{new}^?$), predicts:

- $x_{new}^?$ to be a cat if $\|y_{cat} - Wy_{new}^?\| < \|y_{dog} - Wx_{new}^?\|$

How to "learn" to classify?

Learning phase: find W that minimizes $\sum_{i,j} \|y_{cat} - Wx_i^{cat}\|^2 + \|y_{dog} - Wx_j^{dog}\|^2$, where $x_i^{cat}, x_j^{dog} \in \mathbb{R}^{d_x}$ and $y_{cat}, y_{dog} \in \mathbb{R}^{d_y}$ **chosen by user**.

- **Input**: (X, Y), images $X = [x_1^{cat}, \ldots, x_1^{dog}, \ldots] \in \mathbb{R}^{d_x \times 2m}$ with associated labels $Y = [y_{cat}, \ldots, y_{dog}, \ldots] \in \mathbb{R}^{d_y \times 2m}$.
- **Output**: $W \in \mathbb{R}^{d_y \times d_x}$ that minimizes the difference $\|Y - WX\|^2$

Prediction phase: for a new image $x_{new}^?$ (of unknown true label $x_{new}^?$), predicts:

- $x_{new}^?$ to be a cat if $\|y_{cat} - Wy_{new}^?\| < \|y_{dog} - Wx_{new}^?\|$
- $y_{new}^?$ to be a dog otherwise

From Linear Regression to Deep Neural Networks

Objective: given (X, Y), find W that minimizes the difference $\|Y - WX\|^2$.

$\Rightarrow$"Best" solution: linear regression $W = YX^\mathsf{T}(XX^\mathsf{T})^{-1}$ if XX^T invertible. However,

Objective: given (X, Y), find W that minimizes the difference $\|Y - WX\|^2$.

$\Rightarrow$"Best" solution: linear regression $W = YX^\intercal(XX^\intercal)^{-1}$ if $XX^\intercal$ invertible. However,

- linear regression may easily **overfit**: **"learned" W too adapted to the given pair (X, Y) and $\|y^?_{new} - Wx^?_{new}\|$ large if $x^?_{new} \notin X$.**

From Linear Regression to Deep Neural Networks

Objective: given (X, Y), find W that minimizes the difference $\|Y - WX\|^2$.

$\Rightarrow$"Best" solution: linear regression $W = YX^\intercal(XX^\intercal)^{-1}$ if $XX^\intercal$ invertible. However,

- linear regression may easily **overfit**: **"learned" W too adapted to the given pair (X, Y) and $\|y^?_{new} - Wx^?_{new}\|$ large if $x^?_{new} \notin X$.**
- does not work well for difficult problems (e.g., cat & dogs classification, face recognition, etc.)

From Linear Regression to Deep Neural Networks

⇒(Brain-inspired) LINEAR neural network models (back to [Rosenblatt, 1958])

From Linear Regression to Deep Neural Networks

⇒(Brain-inspired) LINEAR neural network models (back to [Rosenblatt, 1958])

From Linear Regression to Deep Neural Networks

⇒(Brain-inspired) LINEAR neural network models (back to [Rosenblatt, 1958])

⟹

⇒(Brain-inspired) LINEAR neural network models (back to [Rosenblatt, 1958])

$\Longrightarrow$

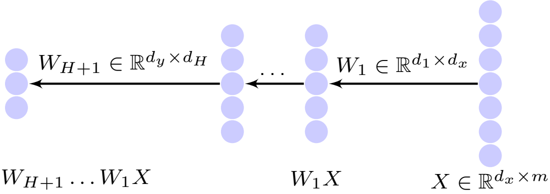


Figure: Illustration of H-hidden-layer linear neural network

$$\boxed{W = W_{H+1} W_H \cdots W_1}$$

⇒(Brain-inspired) LINEAR neural network models (back to [Rosenblatt, 1958])

⟹

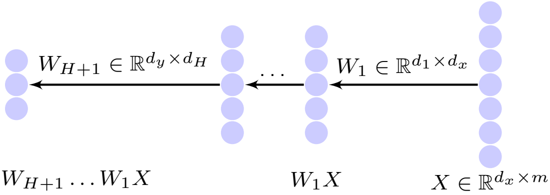


Figure: Illustration of H-hidden-layer linear neural network

$$\boxed{W = W_{H+1} W_H \cdots W_1}$$

- "deeper" structures brings better performance but are computationally more difficult

⇒(Brain-inspired) LINEAR neural network models (back to [Rosenblatt, 1958])

$\Longrightarrow$

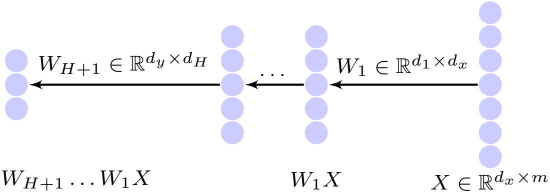


Figure: Illustration of H-hidden-layer linear neural network

$$\boxed{W = W_{H+1} W_H \cdots W_1}$$

- "deeper" structures brings better performance but are computationally more difficult
- with rapid development of modern computation hardwares (Graphics Processing Units, etc)

From Linear Regression to Deep Neural Networks

⇒(Brain-inspired) LINEAR neural network models (back to [Rosenblatt, 1958])

⟹

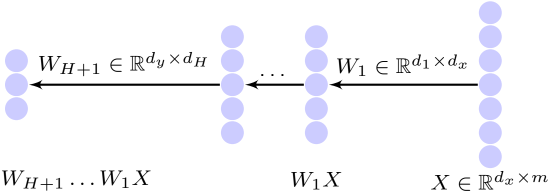


Figure: Illustration of H-hidden-layer linear neural network

$$\boxed{W = W_{H+1} W_H \cdots W_1}$$

- "deeper" structures brings better performance but are computationally more difficult
- with rapid development of modern computation hardwares (Graphics Processing Units, etc)

[Lecun 1998] 5-layer of $60K$ parameters to [He, 2015] 152-layer of $60M$ parameters

- NONLINEAR neural networks:

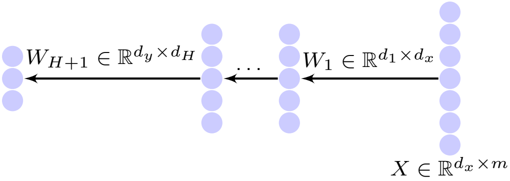


Figure: Illustration of H-hidden-layer nonlinear neural network

From Linear Regression to Deep Neural Networks

- NONLINEAR neural networks:

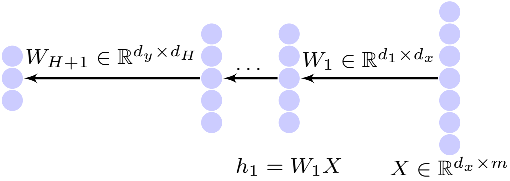


Figure: Illustration of H-hidden-layer nonlinear neural network

- NONLINEAR neural networks:

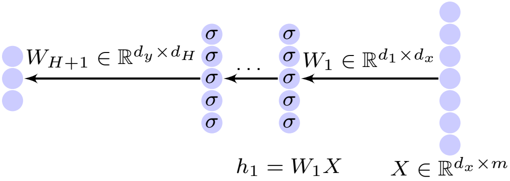


Figure: Illustration of H-hidden-layer nonlinear neural network

- NONLINEAR neural networks:

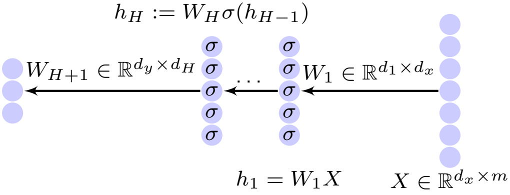


Figure: Illustration of H-hidden-layer nonlinear neural network

From Linear Regression to Deep Neural Networks

- NONLINEAR neural networks:

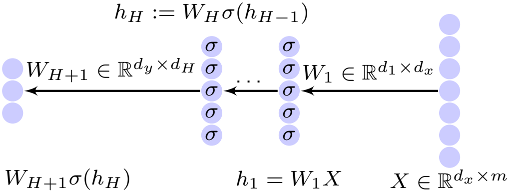


Figure: Illustration of H-hidden-layer nonlinear neural network

with (nonlinear) *activation function* $\sigma(z)$: $\mathrm{ReLU}(z) = \max(z, 0)$, Leaky ReLU $\max(z, az)$ (a¿0) or sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$.

$$\boxed{W \cdot X = W_{H+1}\sigma(W_H\sigma(W_{H-1}\sigma(\cdots W_1 X))).}$$

- NONLINEAR neural networks:

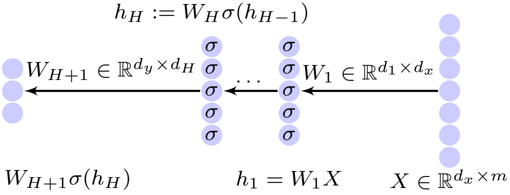
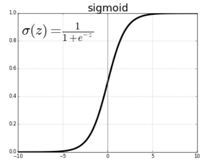


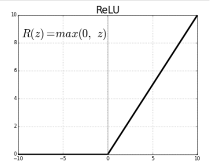
Figure: Illustration of H-hidden-layer nonlinear neural network

with (nonlinear) *activation function* $\sigma(z)$: $\mathrm{ReLU}(z) = \max(z, 0)$, Leaky ReLU $\max(z, az)$ (a¿0) or sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$.

$$\boxed{W \cdot X = W_{H+1}\sigma(W_H \sigma(W_{H-1}\sigma(\cdots W_1 X))).}$$

$\Longrightarrow$





- NONLINEAR neural networks:

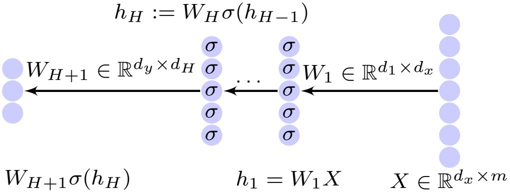
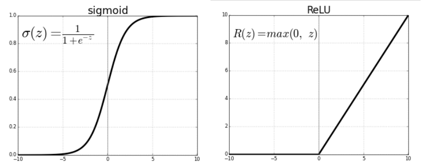


Figure: Illustration of H-hidden-layer nonlinear neural network

with (nonlinear) *activation function* $\sigma(z)$: $\mathrm{ReLU}(z) = \max(z, 0)$, Leaky ReLU $\max(z, az)$ (a¿0) or sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$.

$$\boxed{W \cdot X = W_{H+1}\sigma(W_H \sigma(W_{H-1}\sigma(\cdots W_1 X))).}$$

$\Longrightarrow$



- Or more elaborate structures: convolution, recurrent, residual, etc.

On LINEAR Deep Nonlinear Neural Networks

Set $d_{H+1} := d_y$, $d_0 := d_x$ and consider

$$X \in \mathbb{R}^{d_0 \times m} \quad Y \in \mathbb{R}^{d_{H+1}}.$$

Goal: find $\mathbf{W} = (W_{H+1}, \cdots, W_1)$ that minimizes the function (depending on (X, Y) !!)

$$F(\mathbf{W}) := \|Y - WX\|^2, \quad W = W_{H+1} W_H \cdots W_1,$$

where

$$W_j \in \mathbb{R}^{d_j \times d_{j-1}}, \quad 1 \leq j \leq H+1.$$

On LINEAR Deep Nonlinear Neural Networks

Set $d_{H+1} := d_y$, $d_0 := d_x$ and consider

$$X \in \mathbb{R}^{d_0 \times m} \quad Y \in \mathbb{R}^{d_{H+1}}.$$

Goal: find $\mathbf{W} = (W_{H+1}, \cdots, W_1)$ that minimizes the function (depending on (X, Y) !!)

$$F(\mathbf{W}) := \|Y - WX\|^2, \quad W = W_{H+1} W_H \cdots W_1,$$

where

$$W_j \in \mathbb{R}^{d_j \times d_{j-1}}, \quad 1 \leq j \leq H+1.$$

Define state space $\mathcal{W}$ (recall $d_y = d_{H+1}$ and $d_0 = d_x$)

$$\mathcal{W} = \mathbb{R}^{d_{H+1} \times d_H} \times \cdots \mathbb{R}^{d_1 \times d_0}.$$

and **Gradient Descent associated with** F

$$(GD)_{(X,Y)} \qquad \frac{d\mathbf{W}}{dt} = -\nabla F(\mathbf{W}), \quad \mathbf{W} \in \mathcal{W}.$$

On LINEAR Deep Nonlinear Neural Networks

Set $d_{H+1} := d_y$, $d_0 := d_x$ and consider

$$X \in \mathbb{R}^{d_0 \times m} \quad Y \in \mathbb{R}^{d_{H+1}}.$$

Goal: find $\mathbf{W} = (W_{H+1}, \cdots, W_1)$ that minimizes the function (depending on (X, Y) !!)

$$F(\mathbf{W}) := \|Y - WX\|^2, \quad W = W_{H+1} W_H \cdots W_1,$$

where

$$W_j \in \mathbb{R}^{d_j \times d_{j-1}}, \quad 1 \leq j \leq H+1.$$

Define state space $\mathcal{W}$ (recall $d_y = d_{H+1}$ and $d_0 = d_x$)

$$\mathcal{W} = \mathbb{R}^{d_{H+1} \times d_H} \times \cdots \mathbb{R}^{d_1 \times d_0}.$$

and **Gradient Descent associated with** F

$$(GD)_{(X,Y)} \qquad \frac{d\mathbf{W}}{dt} = -\nabla F(\mathbf{W}), \quad \mathbf{W} \in \mathcal{W}.$$

Conjecture ($\Longleftrightarrow$ Overfitting Problem)

(OP): For a.e. (X, Y) and $\mathbf{W}_0 \in \mathcal{W}$, traj. of $(GD)_{(X,Y)}$ starting at $\mathbf{W}_0$ CV to a glob. minimum of F.

Gradient Descent for Linear Neural Networks - First reductions

(Usual) working assumptions

$$X, Y \text{ full rank }, m \geq \max(d_i) \leq \min(d_i) = d_y.$$

Up to SVD and computations, can assume

$$X = Id_{d_x} \text{ (i.e. } m = d_x), \quad Y = \begin{pmatrix} D_Y & 0 \end{pmatrix}, \quad D_Y \in \mathbb{R}^{d_y \times d_y} \text{ diagonal } > 0.$$

Gradient Descent for Linear Neural Networks - First reductions

(Usual) working assumptions

$$X, Y \text{ full rank }, m \geq \max(d_i) \leq \min(d_i) = d_y.$$

Up to SVD and computations, can assume

$$X = Id_{d_x} \text{ (i.e. } m = d_x), \quad Y = \begin{pmatrix} D_Y & 0 \end{pmatrix}, \quad D_Y \in \mathbb{R}^{d_y \times d_y} \text{ diagonal } > 0.$$

$\boxed{Notation.}$

$$(\Pi W)_i^j = W_j \cdots W_i, \quad 1 \leq i \leq j \leq H+1, \quad M = Y - (\Pi W)_1^{H+1}.$$

Gradient Descent for Linear Neural Networks - First reductions

(Usual) working assumptions

$$X, Y \text{ full rank }, m \geq \max(d_i) \leq \min(d_i) = d_y.$$

Up to SVD and computations, can assume

$$X = Id_{d_x} \text{ (i.e. } m = d_x), \quad Y = \begin{pmatrix} D_Y & 0 \end{pmatrix}, \quad D_Y \in \mathbb{R}^{d_y \times d_y} \text{ diagonal } > 0.$$

$\boxed{Notation.}$

$$(\Pi W)_i^j = W_j \cdots W_i, \quad 1 \leq i \leq j \leq H+1, \quad M = Y - (\Pi W)_1^{H+1}.$$

Gradient dynamics, $1 \leq j \leq H+1$

$$(GD)_{D_Y} \qquad \frac{dW_j}{dt} = (\Pi W)_{j+1}^{H+1} M (\Pi W)_1^{j-1}.$$

Gradient Descent for Linear Neural Networks - First reductions

(Usual) working assumptions

$$X, Y \text{ full rank }, m \geq \max(d_i) \leq \min(d_i) = d_y.$$

Up to SVD and computations, can assume

$$X = Id_{d_x} \text{ (i.e. } m = d_x), \quad Y = \begin{pmatrix} D_Y & 0 \end{pmatrix}, \quad D_Y \in \mathbb{R}^{d_y \times d_y} \text{ diagonal } > 0.$$

Notation.

$$(\Pi W)_i^j = W_j \cdots W_i, \quad 1 \leq i \leq j \leq H+1, \quad M = Y - (\Pi W)_1^{H+1}.$$

Gradient dynamics, $1 \leq j \leq H+1$

$$(GD)_{D_Y} \qquad \frac{dW_j}{dt} = (\Pi W)_{j+1}^{H+1} M (\Pi W)_1^{j-1}.$$

Definition. Critical points $\nabla F(\mathbf{W}) = 0$

$$\mathrm{Crit}(F) = \{\mathbf{W} = (W_{H+1}, \cdots, W_1) \in \mathcal{W}, \; (\Pi W)_{j+1}^{H+1} M (\Pi W)_1^{j-1} = 0\}.$$

Candidates for limit points of trajectories.

Gradient Descent for Linear Neural Networks - Convergence

Theorem (C., Liao, Couillet '18)

Every traj. of $(GD)_{D_Y}$ *converges to a element of* $\mathrm{Crit}(F)$*.*

Gradient Descent for Linear Neural Networks - Convergence

Theorem (C., Liao, Couillet '18)

Every traj. of $(GD)_{D_Y}$ converges to a element of $\mathrm{Crit}(F)$.

PROOF

Key (and obvious) remark: $(GD)_{D_Y}$ analytic $\Longrightarrow$ Lojasiewicz's theorem can be used

Proposition (Lojasiewicz 50s')

Every **BOUNDED** *traj. of analytic gradient system converges to critical point.*

Gradient Descent for Linear Neural Networks - Convergence

Theorem (C., Liao, Couillet '18)

Every traj. of $(GD)_{D_Y}$ *converges to a element of* $\mathrm{Crit}(F)$.

PROOF

Key (and obvious) remark: $(GD)_{D_Y}$ analytic $\Longrightarrow$ Lojasiewicz's theorem can be used

Proposition (Lojasiewicz 50s')

Every **BOUNDED** *traj. of analytic gradient system converges to critical point.*

Proof reduces to show that trajectories are bounded.

Proposition (Invariants)

For $1 \leq j \leq H$, *following quantities are conserved along traj. of* $(GD)_{D_Y}$

$$W_{j+1}^{\mathsf{T}} W_{j+1} - W_j W_j^{\mathsf{T}} = \left(W_{j+1}^{\mathsf{T}} W_{j+1} - W_j W_j^{\mathsf{T}}\right)\Big|_{t=0}.$$

$\Longrightarrow \|W_j(t)\|_F^2 = \|W_{H+1}\|_F^2 + C_j \quad t \geq 0,\ 1 \leq j \leq H.$

Set $g(t) = \|W_{H+1}\|_F^2$. Given a traj. of $(GD)_{D_Y}$, one proves that there exists $C_0, C_1 > 0$

$$\frac{dg}{dt} \leq -C_0 g^{H+1}(t) + C_1\left(1 + g^H(t)\right), \quad \forall t \geq 0.$$

Definition

For $\mathbf{W} \in Crit(F)$ define

$$R(\mathbf{W}) = (\Pi W)_2^{H+1}, \quad r(\mathbf{W}) = \mathrm{rank}\, R(\mathbf{W}) \in [0, d_y],$$

$$Z(\mathbf{W}) = (\Pi W)_2^{H} \quad r_Z(\mathbf{W}) = \mathrm{rank}\, Z(\mathbf{W}) \geq R(\mathbf{W}).$$

Then

$$\mathrm{Crit}(F) = \cup_{r=0}^{d_y} \mathrm{Crit}_r(F), \quad \mathrm{Crit}_r(F) = \{\mathbf{W} \in \mathrm{Crit}(F),\ r(\mathbf{W}) = r\}.$$

$CrV(F)$ = Set of critical values of $F = \{F(\mathbf{W}), \mathbf{W} \in \mathrm{Crit}(F)\}$.

Definition

For $\mathbf{W} \in Crit(F)$ define

$$R(\mathbf{W}) = (\Pi W)_2^{H+1}, \quad r(\mathbf{W}) = \operatorname{rank} R(\mathbf{W}) \in [0, d_y],$$

$$Z(\mathbf{W}) = (\Pi W)_2^{H} \quad r_Z(\mathbf{W}) = \operatorname{rank} Z(\mathbf{W}) \geq R(\mathbf{W}).$$

Then

$$\mathrm{Crit}(F) = \cup_{r=0}^{d_y} \mathrm{Crit}_r(F), \quad \mathrm{Crit}_r(F) = \{\mathbf{W} \in \mathrm{Crit}(F),\ r(\mathbf{W}) = r\}.$$

$CrV(F)$ = Set of critical values of $F = \{F(\mathbf{W}), \mathbf{W} \in \mathrm{Crit}(F)\}$.

Proposition (Landscape of Deep Linear Networks)

Assume $Y = \begin{pmatrix} D_Y & 0 \end{pmatrix}$ *has* **two by two distinct eigenvalues**.

i) $CrV(F) =$ *(finite) set of half sums of squares of any subset of singular values of* Y.

ii) $\mathrm{Crit}_{d_y}(L)$ = *set of local (and global) minima with* $F = 0$ *and* $M = 0$.

iii) *For* $0 \leq r \leq d_y - 1$, $\mathrm{Crit}_r(F)$ *algebraic variety of dim.* > 0 *made of saddle points.*
If $r_Z > r \geq 0$, $Hessian(F)(\mathbf{W})$ *has at least one negative eigenvalue.*

Gradient Descent for Linear Neural Networks - Case $H = 1$

Reformulation of Conjecture (OP)

Conjecture (New formulation of (OP))

For a.e. (X, Y), the union of the basins of attraction of saddles points is of measure zero.

Reformulation of Conjecture (OP)

Conjecture (New formulation of (OP))

For a.e. (X, Y), the union of the basins of attraction of saddles points is of measure zero.

Proposition (C., Liao, Couillet '18)

Conjecture (OP) true if $H = 1$.

Argument relies on concept of **Normal Hyperbolicity** (due to Fenichel 1972).