Introduction aux mathématiques pour l'apprentissage de réseaux de neurones

François Malgouyres

Institut de Mathématiques de Toulouse, Université Paul Sabatier, ANITI

Plan

Introduction

Il existe un couple de variables aléatoires (X, Y)

- On connaît $\mathscr{X}$ et $\mathscr{Y}$ tels que $\mathbb{P}(X \in \mathscr{X}) = \mathbb{P}(Y \in \mathscr{Y}) = 1$
- On sait que l'on va observer x d'une réalisation (x, y) de (X, Y)
- On veut construire une fonction $g : \mathscr{X} \longrightarrow \mathscr{Y}$ prédisant y

Pour une fonction coût $L : \mathscr{Y} \times \mathscr{Y} \longrightarrow \mathbb{R}$, on cherche

$$g^* \in \operatorname{argmin}_g R(g)$$

où

$$R(g) = \mathbb{E}(L(g(X), Y))$$

Introduction

- **Classification** : $\mathscr{Y} = \{1, \cdots, n_{out}\}$, pour $n_{out} \geq 2$
  On construit
  $$f : \mathscr{X} \longrightarrow \mathbb{R}^{n_{out}}$$
  on retourne
  $$g(x) \in \operatorname{argmax}_{i=1..n_{out}} f_i(x)$$
  Typiquement :
  $$L(y, y') = \mathbb{1}_{y \neq y'}$$
- **Régression** : $\mathscr{Y} = \mathbb{R}^{n_{out}}$, pour $n_{out} \geq 1$
  On construit
  $$g : \mathscr{X} \longrightarrow \mathbb{R}^{n_{out}}$$
  Typiquement :
  $$L(y, y') = \|y - y'\|^2$$

Introduction

- On observe $(x_i, y_i)_{i=1..n}$ un échantillon i.i.d. de même loi que (X, Y)
- On considère une famille $\mathscr{F}$ de fonctions $g_{\mathbf{w}}$ paramétrées par des paramètres $\mathbf{w}$ d'un espace Euclidien.
  - Dans cet exposé : les **réseaux de neurones**
- On cherche à résoudre

$$w^* \in \operatorname{argmin}_{\mathbf{w}} \; R(g_{\mathbf{w}})$$

Une approche (naïve) consiste à résoudre

$$\widehat{w} \in \operatorname{argmin}_{\mathbf{w}} \; \widehat{R}(g_{\mathbf{w}})$$

où

$$\widehat{R}(g_{\mathbf{w}}) \; = \; \frac{1}{n} \sum_{i=1}^{n} L(g_{\mathbf{w}}(x_i), y_i)$$

Introduction

ATTENTION : Cette modélisation **ne** tient **pas** compte :

- du **biais** dans les données: $(x_i, y_i)_{i=1..n}$ est rarement i.i.d. selon (X, Y)
  - x_i peut ne couvrir que partiellement le domaine, être corrompue, etc
  - y_i peut être une décision biaisée, bruitée, etc
- du besoin de **robustesse**
  - On impose que y_i soit prédit pour tout $x_i + e$, où $\|e\| \leq \varepsilon$
- d'une **régularisation** du réseau
  - **w** est parfois **quantifié** $\Longrightarrow$ **interpretabilité**, faible **coût calculatoire**
  - **w** est tel que $g_{\mathbf{w}}$ est **Lipschitz** $\Longrightarrow$ garantie de **robustesse**
  - **Régularisations explicites** : poids parcimonieux, pénalisation quadratique, dropout
  - Il peut y avoir une **régularisation implicite**
- Problèmes plus complexes. Ex: "embedding de mots"

Introduction

- $R^* = \inf_g R(g)$ le risque optimal
- Pour $\varepsilon > 0$, on fixe $\mathbf{w}^*$ et $\widehat{\mathbf{w}}$ tels que :

$$R(g_{\mathbf{w}^*}) \leq \inf_{\mathbf{w}} R(g_{\mathbf{w}}) + \varepsilon \qquad \text{et} \qquad \widehat{R}(g_{\widehat{\mathbf{w}}}) \leq \inf_{\mathbf{w}} \widehat{R}(g_{\mathbf{w}}) + \varepsilon$$

- Pour $\mathbf{w}$ retourné par un algorithme

On **décompose**

$$\begin{array}{rlcll} 0 \leq & R(g_{\mathbf{w}}) & - & R^* & \text{(l'excès de risque)} \\ = & R(g_{\mathbf{w}}) & - & \widehat{R}(g_{\mathbf{w}}) & \text{(erreur de généralisation)} \\ + & \widehat{R}(g_{\mathbf{w}}) & - & \widehat{R}(g_{\widehat{\mathbf{w}}}) & \text{(erreur d'optimisation)} \\ + & \widehat{R}(g_{\widehat{\mathbf{w}}}) & - & \widehat{R}(g_{\mathbf{w}^*}) & \leq \varepsilon \\ + & \widehat{R}(g_{\mathbf{w}^*}) & - & R(g_{\mathbf{w}^*}) & \text{(erreur de généralisation)} \\ + & R(g_{\mathbf{w}^*}) & - & R^* & \text{(erreur d'approximation)} \end{array}$$

Introduction : Erreur de généralisation

- **Convergence uniforme** : Quelles conditions sur le réseau et n garantissent que

$$\widehat{R}(g_{\mathbf{w}}) \simeq R(g_{\mathbf{w}}),$$

pour tout **w**?

  - N. Harvey, C. Liaw, A. Mehrabian. "Nearly-tight VC-dimension and Pseudodimension Bounds for Piecewise Linear Neural Networks." COLT 2017.
    - ★ VC-dimension des réseaux de neurones ReLU feedforward à H couches, W paramètres est $O(WH\log(W))$.
  - P. Bartlett, D. Foster, M. Telgarsky. "Spectrally-normalized margin bounds for neural networks." NeurIPS 2017.
  - N. Golowich, A. Rakhlin, O. Shamir. "Size-independent sample complexity of neural networks." COLT 2018.
  - etc

N'expliquent pas les performances des réseaux de neurones lorsque $n \leq W$

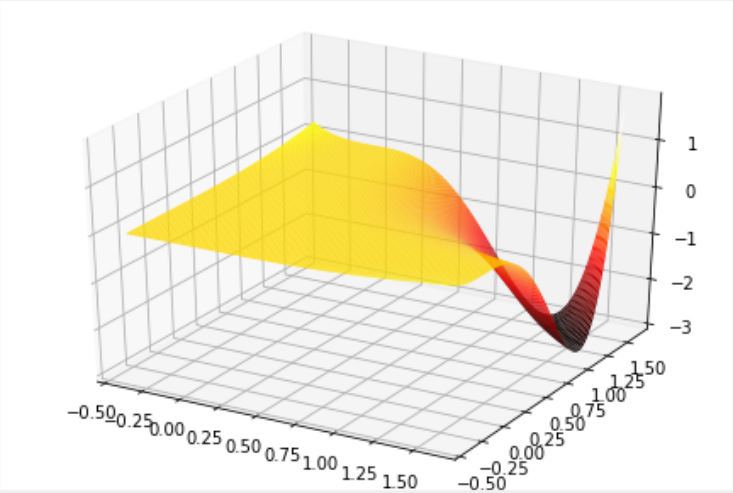
Introduction : Erreur d'approximation

- Est-ce que
$$\inf_{\mathbf{w}} R(g_{\mathbf{w}}) \simeq \inf_{g} R(g)?$$
- Quelle architecture de réseau permet d'approximer une certaine classe de fonctions, avec une précision ε?

Voir l'exposé de **Rémi Gribonval**, à 14h.

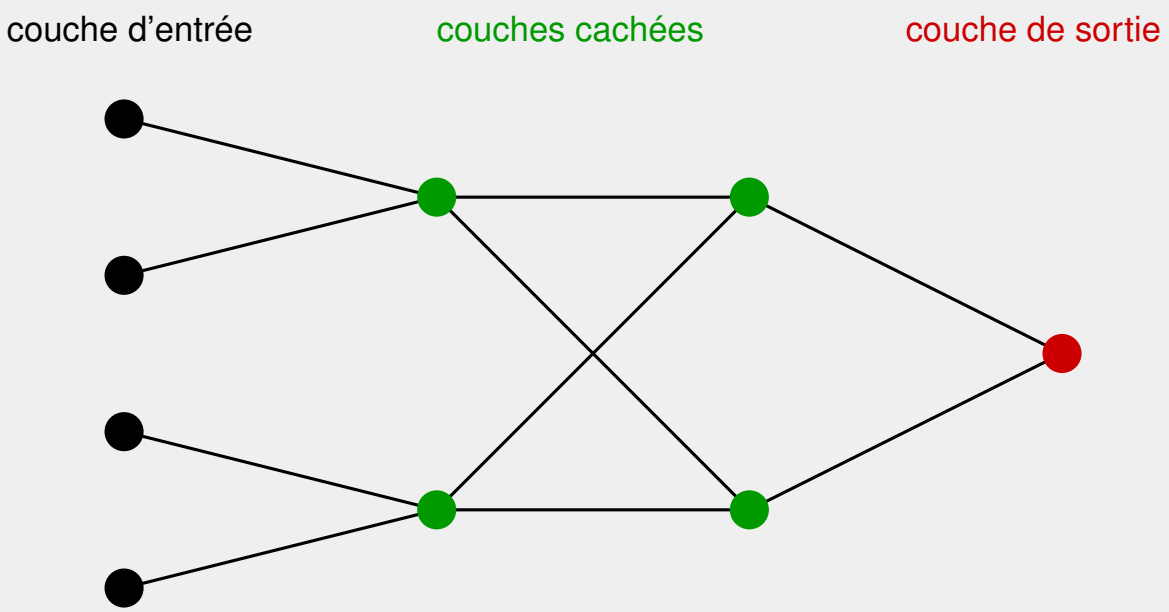
Introduction : Erreurs d'optimisation

- le calcul de la fonction objectif dépend de **beaucoup d'entrées** (n grand)
- La fonction objectif est souvent **non-différentiable**
- **w** vit dans un espace de **grande dimension**
- les **propriétés de la fonction objectif** ne se prêtent pas forcément à l'optimisation. On parle de **paysage de la fonction objectif**.



Plan

La structure Feedforward/Le perceptron



- Couche 0: couche d'entrée;
  Couches 1 et 2: couches cachées;
  Couche 3: couche de sortie.
- Chaque sommet (= neurone) contient un réel
- Chaque arête contient un poids

La structure Feedforward/Le perceptron

- H couches (on dit aussi $H-1$ couches cachées)
- les tailles $n_0, n_1, \cdots, n_H \in \mathbb{N}$ des différentes couches
- On note $f_h(x)$: le résultat obtenu en calculant le contenu de la couche h pour l'entrée $x \in \mathbb{R}^{n_0}$
- On note $W_h \in \mathbb{R}^{n_h \times n_{h-1}}$: la matrice contenant les poids sur les arcs entre la couche $h-1$ et la couche h
- On note $b_h \in \mathbb{R}^{n_h}$: le biais ajouté à la couche h
- On note σ: la fonction d'activation appliquée à chaque couche
  - elle applique la même fonction à chaque entrée d'un vecteur
  - dépend de h, en général

La prédiction/l'inférence : la fonction $f_H : \mathbb{R}^{n_0} \longrightarrow \mathbb{R}^{n_H}$

$$\begin{cases} f_0(x) = x \\ f_h(x) = \sigma\left(W_h f_{h-1}(x) + b_h\right) & , \forall h = 1, \cdots, H \end{cases}$$

La structure Feedforward/Le perceptron: Exemple

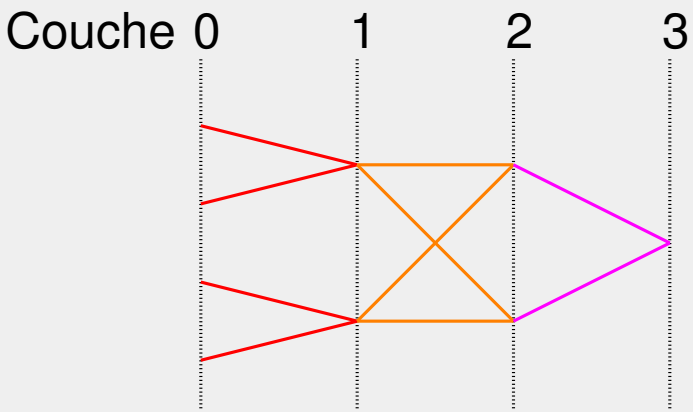


Figure: Feedforward

Ci-dessous, chaque $*$ est un nombre

$$W_1 = \begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \end{pmatrix}, b_1 = \begin{pmatrix} * \\ * \end{pmatrix}, W_2 = \begin{pmatrix} * & * \\ * & * \end{pmatrix}, b_2 = \begin{pmatrix} * \\ * \end{pmatrix}, W_3 = \begin{pmatrix} * & * \end{pmatrix},$$

$$b_3 = \begin{pmatrix} * \end{pmatrix},$$

$$f_3(x) = \sigma\Big(W_3\sigma\big(W_2\sigma(W_1x+b_1)+b_2\big)+b_3\Big)$$

La structure Feedforward/Le perceptron: Vocabulaire et variantes

- **Fully-connected** : Tous les facteurs sont “pleins”
- **Fonctions d’activation** :
  - Dépendent de la couche
  - Une zoologie importante (tanh, heavyside, identité, sigmoïde, etc)
  - La plus utilisée est ReLU : $\sigma(t) = \max(0, t)$
  - Des non-locales : group-sort
  - Souvent, celle associée à la dernière couche est l’identité.
- **“Batch-normalisation”** : Au lieu d’optimiser les biais **b**, on règle un couple $(\mathrm{diag}(\gamma), \mathbf{b})$ permettant centrer les données et fixer leur variance.

Réseaux convolutifs

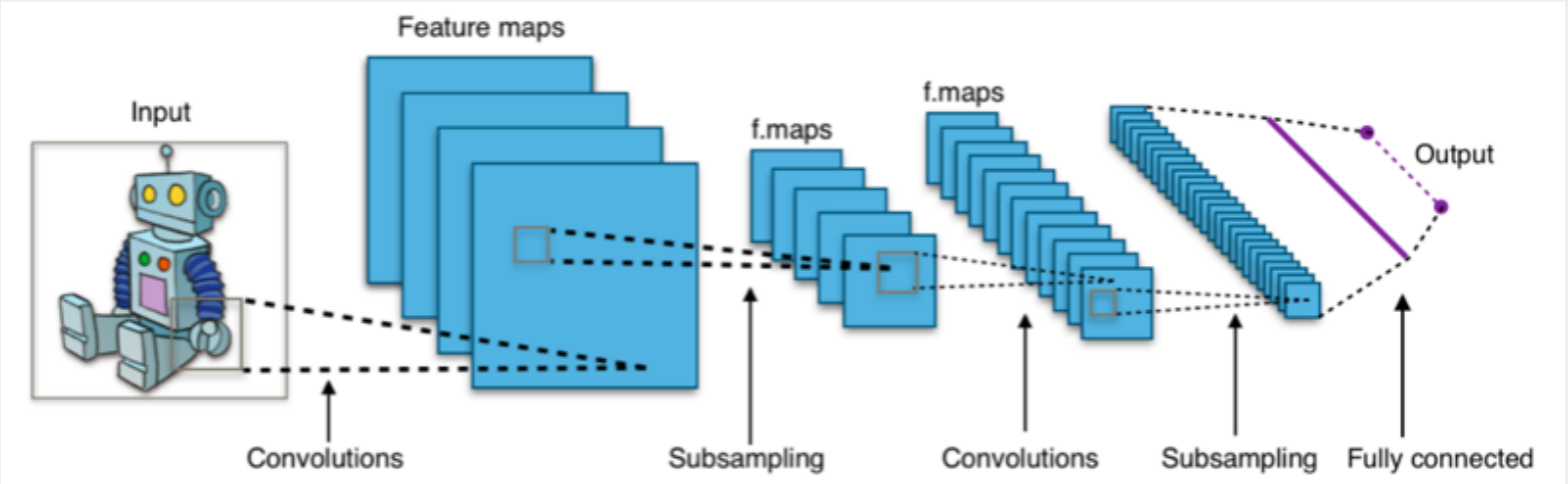
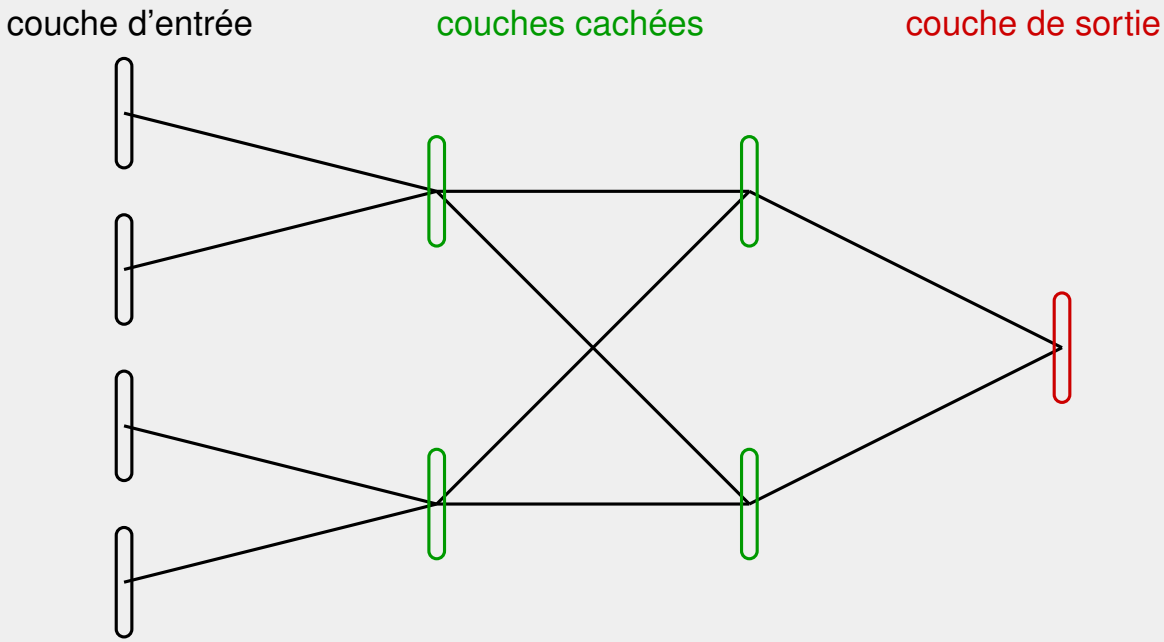


Figure: Source : wikipedia page "apprentissage profond"

Réseaux convolutifs



- Chaque sommet contient un signal/une image. On parle de **canal**.
- Les canaux de la couche d'entrée correspondent aux canaux de x.

Réseaux convolutifs

- **des signaux de taille N**
- H couches (on dit aussi $H-1$ couches cachées)
- les tailles $n_0, n_1, \cdots, n_H \in \mathbb{N}$ des différentes couches ($= N\times$ nombre de canal)
- On note $f_h(x)$: le résultat obtenu en calculant le contenu de la couche h pour l'entrée x
- On note $W_h \in \mathbb{R}^{n_h \times n_{h-1}}$: la matrice **concaténant des matrices de convolutions** pour passer de la couche $h-1$ et la couche h
- On note $b_h \in \mathbb{R}^{n_h}$: le biais ajouté à la couche h
- On note σ: la fonction d'activation appliquée à chaque couche
  - elle applique la même fonction à chaque entrée d'un vecteur
  - dépend de h, en général

L'action du réseau : la fonction f_H

$$\begin{cases} f_0(x) = vect(x) \\ f_h(x) = \sigma\left(W_h f_{h-1}(x) + b_h\right) & , \forall h = 1, \cdots, H \end{cases}$$

Réseaux convolutifs: Matrice circulante, Toeplitz, bloc circulante... calculant des convolutions

On suppose $v, x \in \mathbb{R}^{N+1}$. Le signal v est supposé $(N+1)-$périodique de période:

$$v_{-n'} = v_{N+1-n'} \qquad \text{pour tout } n' = 0..N$$

On a:

$$\begin{pmatrix} v_0 & v_N & \cdots & \cdots & \cdots & v_1 \\ v_1 & v_0 & v_N & \cdots & \cdots & v_2 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & v_N \\ v_N & \cdots & \cdots & \cdots & v_1 & v_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} \sum_{n'=0}^{N} v_{0-n'} x_{n'} \\ \sum_{n'=0}^{N} v_{1-n'} x_{n'} \\ \vdots \\ \vdots \\ \vdots \\ \sum_{n'=0}^{N} v_{N-n'} x_{n'} \end{pmatrix} = v * x$$

Réseaux convolutifs: Exemple

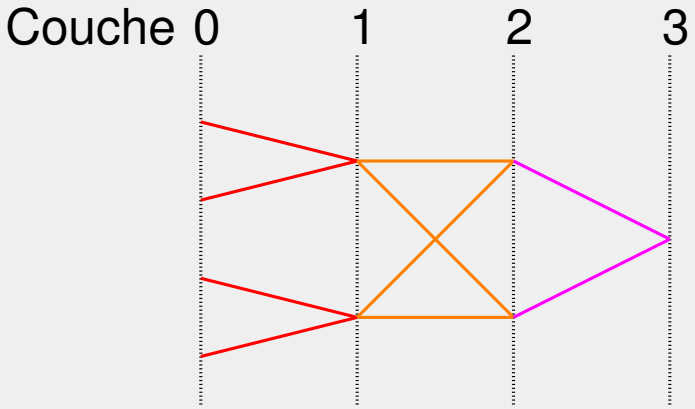


Figure: Réseau convolutif

Ci-dessous,

- chaque $*$ est une matrice composant **une convolution** et **un échantillonnage**
- chaque $\circ$ est un vecteur colonne de taille N (cas sans sous-échantillonnage)

$$W_1 = \begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \end{pmatrix}, b_1 = \begin{pmatrix} \circ \\ \circ \end{pmatrix}, W_2 = \begin{pmatrix} * & * \\ * & * \end{pmatrix}, b_2 = \begin{pmatrix} \circ \\ \circ \end{pmatrix}, W_3 = \begin{pmatrix} * & * \end{pmatrix},$$
$$b_3 = \begin{pmatrix} \circ \end{pmatrix},$$

$$f_3(x) = \sigma\Big(W_3\sigma\big(W_2\sigma(W_1x+b_1)+b_2\big)+b_3\Big)$$

Réseaux convolutifs: Propriétés

Remarque

Les formules sont les mêmes que dans le cas feedforward. Seules les façons de construire les W_h et $f_0(x)$ diffèrent.
Un réseau convolutif est un réseau feedforward dans lequel on impose une structure.

Il existe beaucoup de variantes:

- Une topologie contenant plusieurs type de structures (convolutif, feed-forward ...)
- Couches "Res-net":

$$f_h(x) = \sigma\Big(W_h\ \sigma\big(W_{h-1} f_{h-2}(x) + b_{h-1}\big)\ + b_h + f_{h-2}(x)\Big)$$

- "Max pooling"
- Recurrent neural networks
- Recursive neural networks
- etc

Les combinaisons de réseaux

- **Autoencoder** : Un réseau compresse, un réseau décompresse
- **Création d'embeddings** : Deux réseaux envoient une image et un texte dans un même espace de caractéristiques. Le mapping permet d'interroger le contenu d'images.
- **Generative Adversarial Networks (GAN)** : un réseau génère des données; un réseau discrimine les données générées de vraies données.
- **Réduction de biais dans les données (FairGAN)** : Un réseaux envoie les données dans un espace de caractéristiques et ses poids sont optimisés pour qu'un réseau classifiant sur un critère pertinent fonctionne, un réseau classifiant sur un critère non-pertinent échoue.
- **etc**

Plan

Outline

Premières propriétés

- On dispose d'un échantillon $(x_i, y_i)_{i=1..n}$.
- On dispose d'un réseau de profondeur H, de topologie fixé, dont
  - les arcs sont paramétrés par $\mathbf{w} \in \mathbb{R}^{H \times S}$
  - les nœuds sont paramétrés par $\mathbf{b} \in \mathbb{R}^{n_1 + \cdots + n_H}$
  - la fonction d'activation est notée σ
- On connait : $W_h : \mathbb{R}^S \longrightarrow \mathbb{R}^{n_h \times n_{h-1}}$
- La prédiction est la fonction

$$f_{\mathbf{w},\mathbf{b}}(x) = \sigma\Big(W_H(\mathbf{w}_H) \cdots \sigma\big(W_2(\mathbf{w}_2)\sigma\big(W_1(\mathbf{w}_1)x + b_1\big) + b_2\big) \cdots + b_H\Big)$$

ou, dans le cas feedforward,

$$f_{\mathbf{w},\mathbf{b}}(x) = \sigma\Big(\mathbf{W}_H \cdots \sigma\big(\mathbf{W}_2\sigma\big(\mathbf{W}_1 x + b_1\big) + b_2\big) \cdots + b_H\Big)$$

- Cas de la **régression**
- On considère une fonction objectif définie par

$$\begin{array}{rcl} E : \mathbb{R}^{H \times S} \times \mathbb{R}^{n_1 + \cdots + n_H} & \longrightarrow & \mathbb{R} \\ (\mathbf{w}, \mathbf{b}) & \longmapsto & E(\mathbf{w}, \mathbf{b}) = \sum_{i=1}^n L\big(f_{\mathbf{w},\mathbf{b}}(x_i) - y_i\big) \end{array}$$

pour une fonction coût $L : \mathscr{Y} \longrightarrow \mathbb{R}$.

Premières propriétés

Proposition: Non-coercivité dans le cas ReLU

Pour tout réseau, avec le fonction d'activation ReLU, pour tout échantillon d'apprentissage, la fonction E **n'est pas coercive**.

Preuve utilisant l'homogénéité:
On considère $\mathbf{w} \in \mathbb{R}^{H \times S}$ et, pour tout $\lambda > 0$, on définit $\mathbf{w}^\lambda \in \mathbb{R}^{H \times S}$ par

$$\mathbf{w}_1^\lambda = \lambda^{H-1}\mathbf{w}_1 \qquad \text{et} \qquad \mathbf{w}_h^\lambda = \lambda^{-1}\mathbf{w}_h, \forall h = 2..H$$

Comme pour $\sigma(\lambda t) = \lambda\sigma(t)$ et, pour tout h, $W_h(\lambda \mathbf{w}_h) = \lambda W_h(\mathbf{w}_h)$ on a

$$f_{\mathbf{w},0}(x) = f_{\mathbf{w}^\lambda,0}(x) \qquad \text{, pour tout } x \text{ et tout } \lambda > 0.$$

Donc $E(\mathbf{w}, 0) = E(\mathbf{w}^\lambda, 0)$, pour tout $\lambda > 0$.
Donc E n'est pas coercive.

Premières propriétés

Proposition: Structure de la fonction objectif: le cas ReLU

Pour tout réseau, avec le fonction d'activation ReLU, pour tout $x \in \mathbb{R}^{n_0}$, pour tout $j = 1 \cdots n_H$, la fonction

$$\begin{array}{rcl} \mathbb{R}^{H \times S} \times \mathbb{R}^{n_1 + \cdots + n_H} & \longrightarrow & \mathbb{R} \\ (\mathbf{w}, \mathbf{b}) & \longmapsto & \left[f_{\mathbf{w},\mathbf{b}}(x) \right]_j \end{array}$$

est **continue**. C'est un **polynôme de degré** H **par morceaux**.
Il y a au plus $2^{n_1 + \cdots + n_H}$ morceaux.
Si L est quadratique, la fonction E est **continue et c'est un polynôme de degré** $2H$ **par morceaux**.

Ex.: Le bord $\partial C = \overline{C} - \overset{\circ}{C}$ d'un morceau C est toujours de mesure nulle.

La Rétropropagation

On suppose:

- L est C^1.
- On suppose que σ est C^1

Sous ces hypothèses: E **est différentiable.**

- On ne considère qu'un unique exemple (x, y):
  - On somme si besoin les gradients
  - Cas de l'algorithme du gradient stochastique

Rétropropagation: cas feedforward complètement connecté

On note $f^h_{\mathbf{w},\mathbf{b}}(x)$, le contenu de la couche h, et

$$d^h_{\mathbf{w},\mathbf{b}}(x) = \sigma'\left(\mathbf{W}_h f^{h-1}_{\mathbf{w},\mathbf{b}}(x) + \mathbf{b}_h\right)$$

Proposition: Gradient pour un réseau feedforward

On a pour $h = 1..H$, $i = 1..n_h$, $j = 1..n_{h-1}$

$$\frac{\partial E}{\partial \mathbf{W}_{h,i,j}}(\mathbf{w},\mathbf{b}) = \left[f^{h-1}_{\mathbf{w},\mathbf{b}}(x)\right]_j \left[d^h_{\mathbf{w},\mathbf{b}}(x)\right]_i \Delta^h_i(x)$$

$$\frac{\partial E}{\partial \mathbf{b}_{h,i}}(\mathbf{w},\mathbf{b}) = \left[d^h_{\mathbf{w},\mathbf{b}}(x)\right]_i \Delta^h_i(x)$$

où $\Delta^h(x) \in \mathbb{R}^{n_h}$ est défini par

$$\Delta^h(x) = \begin{cases} \nabla L(f_{\mathbf{w},\mathbf{b}}(x) - y) & \text{, si } h = H \\ \mathbf{W}^T_{h+1}.\mathrm{diag}\left(d^{h+1}_{\mathbf{w},\mathbf{b}}(x)\right).\Delta^{h+1}(x) & \text{, sinon} \end{cases}$$

Rq: $\mathbf{W}^T_{h+1}$ remonte d'un niveau dans le réseau: **Rétropropagation**.

Rétropropagation: Problèmes connus

- **"Vanishing/Exploding gradient"** :
  - Si $\mathbf{W}_{h+1}^T . \mathrm{diag}\left(d_{\mathbf{w},\mathbf{b}}^{h+1}(x)\right)$ est systématiquement une contraction $\Longrightarrow$ "Vanishing gradient"
  - Si $\mathbf{W}_{h+1}^T . \mathrm{diag}\left(d_{\mathbf{w},\mathbf{b}}^{h+1}(x)\right)$ augmente systématiquement la norme $\Longrightarrow$ "Exploding gradient"
- **Dans certaines régions de l'espace, la fonction objectif est très irrégulière** :
  Pour une fonction d'activation homogène[1] (notamment ReLU):
  Pour $\lambda > 0$, $\lambda \sim 0$

$$\mathbf{W}_1^\lambda = \lambda^{H-1}\mathbf{W}_1 \qquad \text{et} \qquad \mathbf{W}_h^\lambda = \lambda^{-1}\mathbf{W}_h, \forall h = 2..H$$

$$\mathbf{b}_1^\lambda = \lambda^{H-1}\mathbf{b}_1 \qquad \text{et} \qquad \mathbf{b}_h^\lambda = \lambda^{-h}\mathbf{b}_h, \forall h = 2..H$$

  On a, pour tout x,

$$f_{\mathbf{w}^\lambda,\mathbf{b}^\lambda}(x) = f_{\mathbf{w},\mathbf{b}}(x)$$

  mais (génériquement) $\frac{\partial E}{\partial \mathbf{W}_2}$ est grand, la constante Lipschitz de $\frac{\partial E}{\partial \mathbf{W}_1}$ est très faible en $\mathbf{W}^\lambda$.

[1] Attention à la non-différentiabilité

Outline

Paysage de la fonction objectif : Introduction

Rappel, pour tout[2] $\mathbf{w}$:

$$
\begin{array}{rlcll}
0 \leq & R(f_{\mathbf{w}}) & - & R^* & \text{(l'excès de risque)} \\
= & R(f_{\mathbf{w}}) & - & \widehat{R}(f_{\mathbf{w}}) & \text{(erreur de généralisation)} \\
+ & \widehat{R}(f_{\mathbf{w}}) & - & \widehat{R}(f_{\widehat{\mathbf{w}}}) & \text{(erreur d'optimisation)} \\
+ & \widehat{R}(f_{\widehat{\mathbf{w}}}) & - & \widehat{R}(f_{\mathbf{w}^*}) & \leq \varepsilon \\
+ & \widehat{R}(f_{\mathbf{w}^*}) & - & R(f_{\mathbf{w}^*}) & \text{(erreur de généralisation)} \\
+ & R(f_{\mathbf{w}^*}) & - & R^* & \text{(erreur d'approximation)}
\end{array}
$$

On utilise un algorithme d'optimisation pour trouver un $\mathbf{w}^{calc}$, on veut que

$$\widehat{R}(f_{\mathbf{w}^{calc}}) - \inf_{\mathbf{w}} \widehat{R}(f_{\mathbf{w}})$$

soit le plus faible possible.

Idéalement, on voudrait que cette quantité soit nulle ou au moins on voudrait la borner supérieurement.

[2] Pour alléger les notations, on oublie $\mathbf{b}$.

Paysage de la fonction objectif : Introduction

On suppose que $\mathbf{w} \longmapsto \widehat{R}(f_{\mathbf{w}})$ est C^2 partout. On a

$$\widehat{R}(f_{\mathbf{w}}) = \widehat{R}(f_{\mathbf{w}^*}) + \langle \nabla_{\mathbf{w}} \widehat{R}(f_{\mathbf{w}^*}), \mathbf{w} - \mathbf{w}^* \rangle + \frac{1}{2} \langle \nabla^2_{\mathbf{w}} \widehat{R}(f_{\mathbf{w}^*})(\mathbf{w} - \mathbf{w}^*), \mathbf{w} - \mathbf{w}^* \rangle + o(\|\mathbf{w} - \mathbf{w}^*\|^2)$$

On distingue:

- **$\mathbf{w}^*$ est un minimiseur global:**
  - $\forall \mathbf{w}, \qquad \widehat{R}(f_{\mathbf{w}^*}) \leq \widehat{R}(f_{\mathbf{w}})$
  - $\widehat{R}(f_{\mathbf{w}^*}) = \min_{\mathbf{w}} \widehat{R}(f_{\mathbf{w}})$
- **$\mathbf{w}^*$ est un minimiseur local:**
  - Il existe un voisinage ouvert $\mathscr{O}$ de $\mathbf{w}^*$ tel que

$$\forall \mathbf{w} \in \mathscr{O}, \qquad \widehat{R}(f_{\mathbf{w}^*}) \leq \widehat{R}(f_{\mathbf{w}})$$

- **$\mathbf{w}^*$ est un point critique du second ordre:**
  - On a

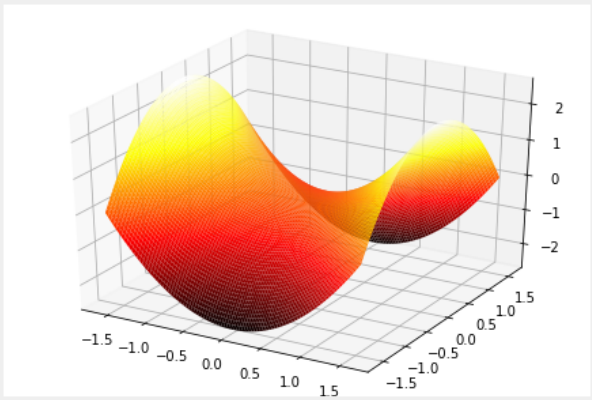
$$\nabla \widehat{R}(f_{\mathbf{w}^*}) = 0 \qquad \text{et} \qquad \nabla^2 \widehat{R}(f_{\mathbf{w}^*}) \succeq 0$$

- **$\mathbf{w}^*$ est un point critique du premier ordre:**
  - On a

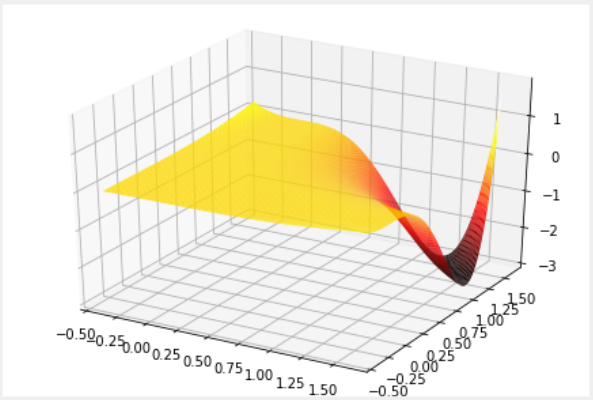
$$\nabla \widehat{R}(f_{\mathbf{w}^*}) = 0$$

Paysage de la fonction objectif : Introduction

- ***w**** **est un point selle** si c'est un point critique qui n'est ni un minimiseur local, ni un maximiseur local
  - **Un point selle *w**** **est strict :** si ce n'est pas un point critique du second ordre (i.e., le Hessian a une v.p. négative).
  - **Un point selle *w**** **est non-strict:** si c'est un point critique du second ordre (i.e. le Hessian est semi-defini positif et a une v.p. égale à 0. Typiquement, un terme d'ordre supérieur en fait un point selle.).



(a) Point selle strict



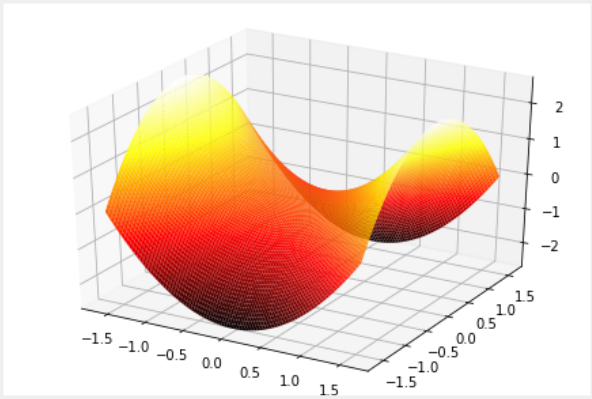
(b) Point selle non-strict

Paysage de la fonction objectif : Introduction

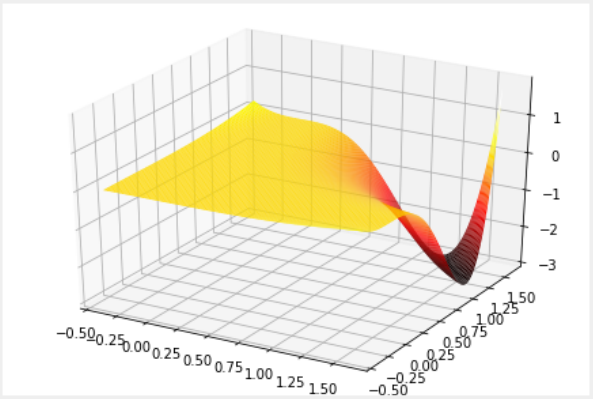
Optimisation non convexe avec les mains

Pour des fonctions non-convexes, on sait montrer que

- **dans un cadre assez vaste**, l'algorithme du gradient (ou gradient stochastique) **converge vers un point critique du premier ordre**
- **dans un cadre plus restreint**, l'algorithme du gradient **converge vers un point critique du second ordre**



(a) Point selle strict



(b) Point selle non-strict

- S. Gadat, F. Panloup, S. Saadane. "Stochastic heavy ball." Electronic Journal of Statistics 12.1 (2018): 461-529.
- J. Lee, M. Simchowitz, M. Jordan, B. Recht." Gradient Descent Converges to Minimizers." COLT 2016.

Paysage pour les réseaux larges

Différents énoncés décrits dans

- Gori, Tesi, "On the problem of local minima in backpropagation", IEEE PAMI, 1992
- Yu, Chen, "On the local minima free condition of backpropagation learning", IEEE Trans. Neural Networks, 1995
- Nguyen, Hein, "The loss surface of deep and wide neural networks", ICML, 2017
- ....

On considère:

- un problème de régression
- un réseau feedforward complètement connecté
- des observations $(x_i, y_i)_{i=1..n}$

Paysage pour les réseaux larges

Le Paysage pour les réseaux larges

On suppose que σ est C^1 et que, pour tout $t \in \mathbb{R}$, $\sigma'(t) \neq 0$. On suppose que le coût L est C^1, à valeur dans $\mathbb{R}^+$ et tel que $L(0) = 0$. On suppose aussi que $\nabla L(y) = 0$ si et seulement si $y = 0$. On note $X = [x_1 \cdots x_n] \in \mathbb{R}^{n_0 \times n}$ et $A = \begin{pmatrix} X \\ \mathbb{1}_n^T \end{pmatrix}$.

On considère un point critique du premier ordre $(\mathbf{w}, \mathbf{b})$ de $\widehat{R}$.
On suppose que $\operatorname{rang}(A) = n$ et que, pour tout $h = 1..H$, $\operatorname{rang}(\mathbf{W}_h) = n_h$.
Alors, on a

$$\widehat{R}(f_{\mathbf{w},\mathbf{b}}) = 0$$

et $(\mathbf{w}, \mathbf{b})$ est un minimum global.

Les hypothèses fortes sont celles sur le rang. Elles impliquent notamment

$$n \leq n_0 + 1 \qquad \text{et} \qquad n_H \leq n_{H-1} \leq \cdots \leq n_0$$

Nb: Ci-dessus l'indice 0 est arbitraire car on pourrait supposer que les x_i sont le résultat des premières couches d'un réseau.

Paysage pour les réseaux linéaires

Pointeurs bibliographiques

- Baldi, Hornik, "Neural networks and principal component analysis: Learning from examples without local minima", Neural networks, 1989.
- Baldi, Hornik, "Learning in linear neural networks: A survey", IEEE Transactions on neural networks, 1995.
- Kawaguchi, "Deep learning without poor local minima", NIPS 2016
- .... (Notamment dans les groupes de S. Arora à Princeton, de F. Bach à l'ENS)
- En collaboration avec **E. Achour** (Polytechnique, M2 MVA, actuellement en 2^{de} année de thèse), **S. Gerchinovitz** (IRT Saint Exupéry)

- un problème de régression
- un réseau feedforward complètement connecté, linéaire (notamment sans biais)
- des observations $(x_i, y_i)_{l=1..n}$

Paysage pour les réseaux linéaires

On note

- $X \in \mathbb{R}^{d_x \times n}$ et $Y \in \mathbb{R}^{d_y \times n}$ les matrices contenant les données.
- $\widehat{R}(\mathbf{W}) = \sum_{i=1}^{m} \| W_H W_{H-1} \cdots W_2 W_1 x_i - y_i \|_2^2 = \| W_H \cdots W_1 X - Y \|^2$
- $$\Sigma_{XX} = \sum_{i=1}^{n} x_i x_i^T = XX^T \in \mathbb{R}^{d_x \times d_x} \quad , \quad \Sigma_{YY} = \sum_{i=1}^{n} y_i y_i^T = YY^T \in \mathbb{R}^{d_y \times d_y},$$
  $$\Sigma_{XY} = \sum_{i=1}^{n} x_i y_i^T = XY^T \in \mathbb{R}^{d_x \times d_y} \quad , \quad \Sigma_{YX} = \sum_{i=1}^{n} y_i x_i^T = YX^T \in \mathbb{R}^{d_y \times d_x},$$
- $$\Sigma^{1/2} = \Sigma_{YX} \Sigma_{XX}^{-1} X \in \mathbb{R}^{d_y \times n},$$

  sa SVD

  $$\Sigma^{1/2} = U \Delta V^T,$$

  avec $U \in \mathbb{R}^{d_y \times d_y}$, $\Delta = \mathrm{diag}\big((\delta_i)_{i=1..d_y}\big) \in \mathbb{R}^{d_y \times n}$, $V \in \mathbb{R}^{n \times n}$.
- $r_{max} = \min(d_x, n_1, \ldots, n_{H-1}, d_y)$

Paysage pour les réseaux linéaires

On suppose

- $d_y \leq d_x \leq n$
- Σ_{XX} est inversible et Σ_{XY} est de rang plein
- les valeurs singulières de $\Sigma^{1/2}$ sont distinctes

Lemme (Inférence et valeurs critiques)

Soit $\mathbf{W}$ *un point critique du premier ordre de* $\widehat{R}$. *On pose* $r = rk(W_H \cdots W_1)$.
Il existe un unique sous-ensemble $\mathscr{S} \subset [\![1, d_y]\!]$ *de taille* r *tel que:*

$$W_H \cdots W_1 = U_{\mathscr{S}} U_{\mathscr{S}}^T \Sigma_{YX} \Sigma_{XX}^{-1}.$$

La valeur critique correspondante vaut $\widehat{R}(\mathbf{W}) = \mathrm{tr}(\Sigma_{YY}) - \sum_{i \in \mathscr{S}} \delta_i^2$.
On dit que le point critique $\mathbf{W}$ *est associé à* $\mathscr{S}$.

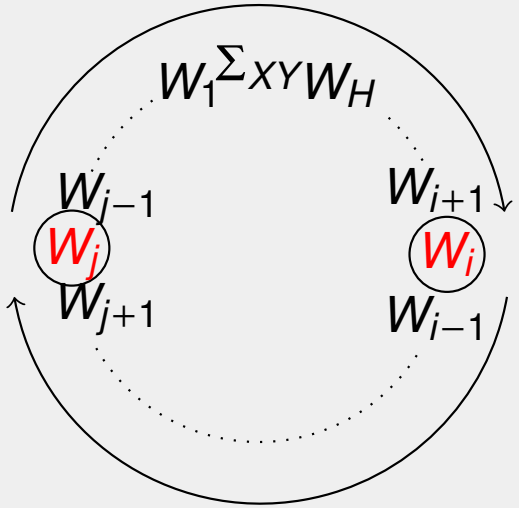
Proposition

Pour tout $\mathscr{S} \subset [\![1, d_y]\!]$ *de taille* $r \in [\![0, r_{max}]\!]$, *il existe un point critique* $\mathbf{W}$ *associé à* $\mathscr{S}$.

Paysage pour les réseaux linéaires

- **Pivot** $(i,j) \in [\![1,H]\!]^2$, avec $i > j$
- **Blocs complémentaires du pivot** (i,j):

Premier bloc complémentaire : $W_{j-1} \cdots W_1 \Sigma_{XY} W_H \cdots W_{i+1}$



Second bloc complémentaire : $W_{i-1} \cdots W_{j+1}$

- **Pivot étreint pour W** : L'un des blocs complémentaires est de rang $\mathrm{rk}(W_H \dots W_1)$
- **W est point critique étreint**: **W** est un point critique et tous les pivots sont étreints pour **W**

Paysage pour les réseaux linéaires

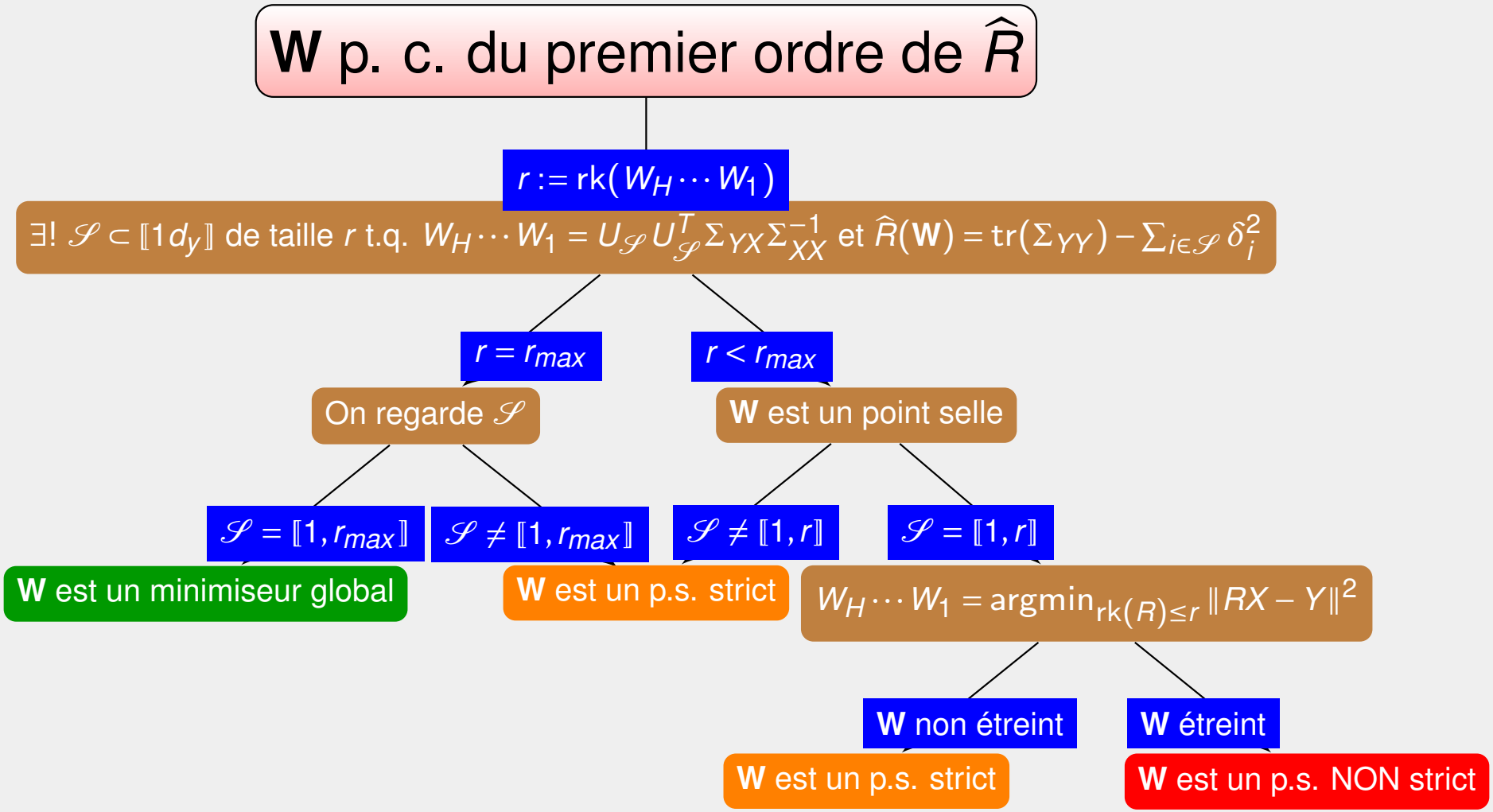


Figure: Theorem [E. Achour et al.]

Paysage pour les réseaux linéaires

Proposition (E. Achour et al.)

- *Pour $H = 2$, il n'existe pas de point selle non-strict.*
- *Pour $H \geq 3$, pour tout $r < r_{max}$, il existe des points critiques associés à $[\![1, r]\!]$ étreints et non-étreints.*

Proposition (E. Achour et al.)

On note $\mathscr{S}_{max} = [\![1, r_{max}]\!]$ et $Q_{max} = [\![1, d_y]\!] \setminus \mathscr{S}_{max} = [\![r_{max}+1, d_y]\!]$.
W *est un* **minimiseur global** *de $\widehat{R}$* **si et seulement si** *il existe des matrices inversibles $D_{H-1} \in \mathbb{R}^{d_{H-1} \times d_{H-1}}, \dots, D_1 \in \mathbb{R}^{d_1 \times d_1}$, et des matrices $A_R \in \mathbb{R}^{(d_y - r_{max}) \times (d_{H-1} - r_{max})}$, $(W_h)_{DR} \in \mathbb{R}^{(d_h - r_{max}) \times (d_{h-1} - r_{max})}$ pour $h \in [\![2, H-1]\!]$, et $M_D \in \mathbb{R}^{(d_1 - r_{max}) \times d_x}$ tels que:*

$$W_H = [U_{\mathscr{S}_{max}}, U_{Q_{max}} A_R] D_{H-1}^{-1}$$

$$W_h = D_h \begin{bmatrix} I_{r_{max}} & 0 \\ 0 & (W_h)_{DR} \end{bmatrix} D_{h-1}^{-1} \qquad \forall h \in [\![2, H-1]\!]$$

$$W_1 = D_1 \begin{bmatrix} U_{\mathscr{S}_{max}}^T \Sigma_{YX} \Sigma_{XX}^{-1} \\ M_D \end{bmatrix}.$$

Paysage pour les réseaux linéaires

Conclusion [E. Achour et al.]

- **Caractérisation** de l'ensemble: des minimiseurs globaux; points selles stricts; point selles non-stricts.
- Tout **point critique du second ordre** qui n'est pas un minimiseur global conduit à une solution de la régression linéaire sous **contrainte de rang**.
- Les points selles non-stricts sont associés avec $\mathbf{r_{max}}$ **valeurs plateau** pour le risque empirique
- **Paramétrisation des minimiseurs globaux**

Autres travaux sur les réseaux de neurones :

- Condition d'identifiabilité et de stabilité de **w** : SIAM data science, MSML, ITW, SPARS et thèse de **J. Bona-Pellissier** en cours
- Apprentissage certifié de fonction monotone : AAAI, workshop safeAI

Merci pour votre attention !