Big Data

François Malgouyres

Institut de Mathématiques de Toulouse, Université Paul Sabatier

2025-2026

Plan

Outline

Classification et régression

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

Plus précisément, pour une fonction coût $L : \mathscr{Y} \times \mathscr{Y} \longrightarrow \mathbb{R}$, on veut trouver

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

On appelle $R(g) = \mathbb{E}(L(g(X), Y))$ le risque de g.

Car : La personnes utilisant g observe le coût

$$R_{test}(g) \simeq \frac{1}{n'} \sum_{i=1}^{n'} L(g(x'_i), y'_i)$$

pour un échantillon i.i.d $(x'_i, y'_i)_{i=1..n'}$ suivant la loi (X, Y), **indépendant de l'échantillon d'apprentissage**.
La loi des grands nombres conduit à considérer R.

Classification

En classification, $\mathscr{Y}$ est fini.

- Classification binaire : $\mathscr{Y} = \{-1, +1\}$
- Classification multi-classes : $\mathscr{Y} = \{1, \cdots, C\}$, $C \in \mathbb{N}$, $C > 2$

En deep-learning, pour $C \geq 2$, on construit C fonctions

$$g_c : \mathscr{X} \longrightarrow \mathbb{R} \qquad , c = 1..C$$

on retourne

$$y \in \text{argmax}_{c=1..C}\ g_c(x)$$

$$L(y, y') = \mathbb{1}_{y \neq y'} \qquad \text{ou la perte logistique, etc}$$

Exemples:

- $\mathscr{X} = \mathbb{R}^2$, $\mathscr{Y} = \{-1, +1\}$
- $\mathscr{X} = \mathbb{R}^n$: n est le nombre de pixels d'une image,
  $\mathscr{Y} = \{1, \cdots, C\}$: chaque sortie est une "note" pour la présence d'un objet.
- $\mathscr{X} = \mathbb{R}^n$: n est la taille d'un "état" (ex: l'écran d'un jeu Atari),
  $\mathscr{Y} = \{1, \cdots, C\}$: chaque sortie est l'espérance de gain de l'action $c = 1..C$.

Régression

En régression,

- $\mathscr{Y} = \mathbb{R}$
- $\mathscr{Y} = \mathbb{R}^C$

On construit C fonctions

$$g_c : \mathscr{X} \longrightarrow \mathbb{R} \qquad , c = 1..C$$

$$L(y, y') = \|y - y'\|^2 \qquad \text{(Son choix est une composante de la modélisation)}$$

Exemples:

- $\mathscr{X} = \mathbb{R}$, $\mathscr{Y} = \mathbb{R}$
- $\mathscr{X} = \mathbb{R}^d$: d est le nombre de mesures concernant une situation économique actuelle,
  $\mathscr{Y} = \mathbb{R}^C$: différentes quantités économiques futures $c = 1..C$.
- $\mathscr{X} = \mathbb{R}^d$: d la taille d'un vocabulaire, $(0, \ldots, 0, 1, 0, \ldots, 0)$ est un mot
  $\mathscr{Y} = \mathbb{R}^C$: espace d'une représentation des mots du vocabulaire.

Classification et régression: Erreur Bayésienne

Décision Bayésienne

Sous des hypothèses faibles sur la loi de (X, Y) et si elle existe, la fonction définie pour $x \in \mathscr{X}$ par

$$g_b(x) \in \operatorname{argmin}_{y \in \mathscr{Y}} \ \mathbb{E}(L(y, Y)|X = x)$$

minimise le risque :

$$\forall g : \mathscr{X} \to \mathscr{Y}, \qquad R(g) \ \geq \ R(g_b)$$

On appelle g_b la **décision Bayésienne**.

Remarques :

- On n'a pas forcément

  $$\forall x \in \mathscr{X}, \exists a \in \mathscr{Y}, \qquad \mathbb{P}\left(Y = a|X = x\right) = 1.$$

  En général,

  $$R(g_b) \neq 0.$$

- On ne peut pas calculer g_b car:
  - On ne connaît pas la loi de (X, Y)
  - On devrait représenter g_b avec un ordinateur

Classification et régression : les principes

- 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_w paramétrés par des paramètres w d'un espace Euclidien.
  - Pour nous : elle est basée sur les **réseaux de neurones**
- On voudrait résoudre

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

- La **minimisation du risque empirique** consiste à résoudre

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

où

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

Le contrôle du risque

- Pour toute fonction g, on appelle **risque en population** et **risque empirique** de g :

$$R(g) = \mathbb{E}\left(L(g(X), Y)\right) \qquad \text{et} \qquad \widehat{R}(g) = \frac{1}{n}\sum_{i=1}^{n} L(g(x_i), y_i)$$

- $R^* = \inf_g R(g)$ 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**

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

Puis on **majore** indépendamment chaque terme.

Le contrôle du risque

- **Erreur d'optimisation** : $|\widehat{R}(g_{\mathbf{w}}) - \widehat{R}(g_{\widehat{\mathbf{w}}})|$
  - Est-ce-que l'optimisation a réussi? Pbs: Paysage de la fonction objectif, n grand, d grand.
  - **En deep-learning :** Elle est petite pour les **"gros" réseaux**.
- **Erreur d'approximation** : $|R(g_{\mathbf{w}^*}) - R(g_b)|$, où $R(g_b) \simeq \inf_g R(g)$
  - Il suffit que $\|g_{\mathbf{w}^*} - g_b\|$ soit petit (le choix de la norme compte).
  - Quelles fonctions peut-on approximer avec notre classe de fonctions? Mots clefs: Expressivité, théorie de l'approximation etc
  - **En deep-learning :** Elle est petite pour les **"gros" réseaux**.
- **Erreur de généralisation** : $|R(g_{\mathbf{w}}) - \widehat{R}(g_{\mathbf{w}})|$
  - Quelles conditions sur le réseau et l'échantillon pour avoir la **convergence uniforme**

$$\sup_{\mathbf{w}} |\widehat{R}(g_{\mathbf{w}}) - R(g_{\mathbf{w}})| \qquad \text{soit petit?}$$

  - **En théorie :** Dimension de Vapnik-Chervonenkis $\simeq$ nombre de paramètres
  - **En pratique :** Reste faible même pour les **"gros" réseaux**

$\rightsquigarrow$ **régularisation implicite**.

La régularisation implicite : Visualisation

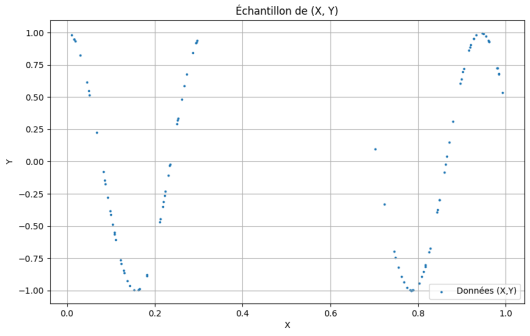


Figure: Source : demo_regularisation_implicite.ipynb

Apprentissage d'un réseau de neurones MLP, ReLU, de profondeur 6, de largeur 40.
Que va-t'il se passer dans $[0.3, 0.7]$?

La régularisation implicite : Visualisation

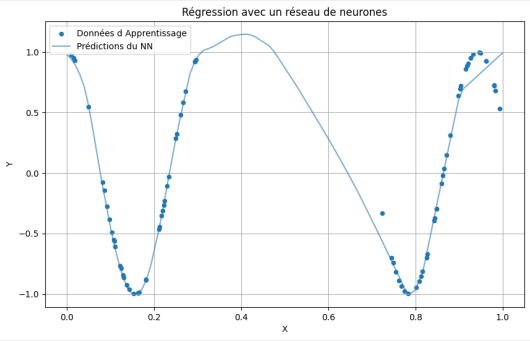


Figure: Source : demo_regularisation_implicite.ipynb

La régularisation implicite : pas de sur-apprentissage

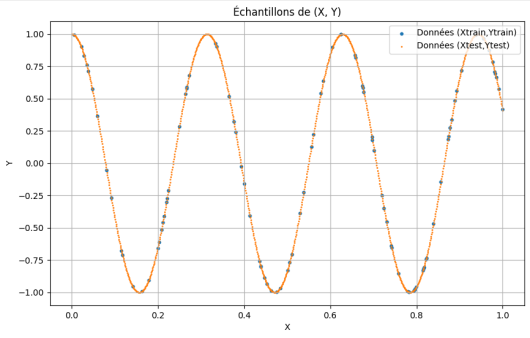


Figure: Source : demo_regularisation_implicite_overfit.ipynb

L'échantillon d'apprentissage comprend 100 exemples.
Quelles sont les performances des réseaux de profondeur fixe (6 couches) pour différentes largeurs ?

largeur	10	20	30	40	50	60	70	80	90	100	200	400
nb. param.	581	2161	4741	8321	12901	18481	25061	32641	41221	50801	201601	803201

La régularisation implicite : pas de sur-apprentissage

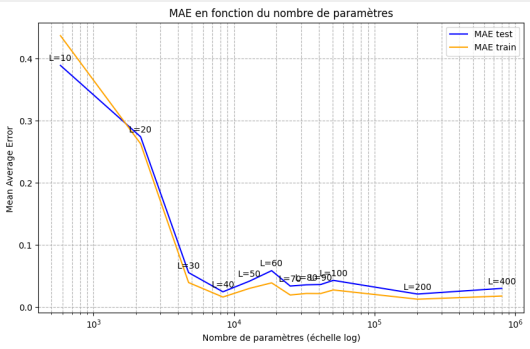


Figure: Source : demo_regularisation_implicite_overfit.ipynb

Même très surparamétré, dans cette expérience, le réseau de neurones ne surapprend pas.
Warning: Cela arrive tout de même.

Le contrôle du risque: En pratique

Pour **w** retourné par un algorithme

$$\begin{array}{rcll} R(g_{\mathbf{w}}) & = & R(g_{\mathbf{w}}) - \widehat{R}(g_{\mathbf{w}}) & \text{(erreur de généralisation)} \\ & + & \widehat{R}(g_{\mathbf{w}}) & \text{observable} \end{array}$$

- N'a d'intérêt que si on est capable de trouver un réseau et un **w** tels que

$$\widehat{R}(g_{\mathbf{w}}) \qquad \text{soit petit.}$$

  - i.e. : On a résolu les problèmes de l'expressivité et de l'optimisation.
- **Erreur de généralisation** : même problématique que précédemment.

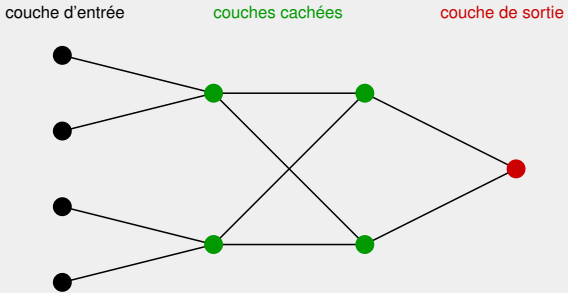
Classification et régression : les principes

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

- Problèmes plus complexes : apprentissage non-supervisé, semi-supervisé, par renforcement, apprentissage de représentations ...
- du **biais** dans les données: $(x_i, y_i)_{i=1..n}$ est rarement i.i.d. selon (X, Y)
  - Les x_i peuvent ne couvrir que partiellement le domaine, être corrompus, etc
  - Les y_i peuvent 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
  - $\mathbf{w}$ est parfois **quantifié** $\Longrightarrow$ **interpretabilité**, faible **coût énergétique/calculatoire/mémoire**
  - $\mathbf{w}$ est tel que $g_{\mathbf{w}}$ est 1**-Lipschitz** $\Longrightarrow$ garantie de **robustesse**
  - **Régularisations explicites** : poids parcimonieux, pénalisation quadratique, dropout

Outline

Multi-Layer Perceptron/fully-connected neural network



- 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

Multi-Layer Perceptron/fully-connected neural network

- H couches (on dit aussi $H-1$ couches cachées)
- les tailles $n_0, n_1, \ldots, n_H \in \mathbb{N}$ 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_h: la fonction d'activation appliquée à chaque couche
  - Typiquement, elle applique la même fonction à chaque entrée d'un vecteur

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

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

Multi-Layer Perceptron/fully-connected neural network : Exemple

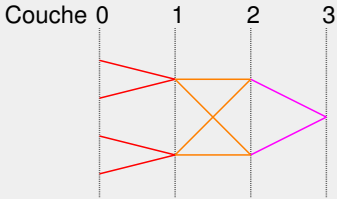


Figure: Réseau Fully-connected

Ci-dessous, chaque $*$ est un nombre

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

$$f_3(x) = \sigma_3\Big(W_3\sigma_2\big(W_2\sigma_1\big(W_1 x + b_1\big) + b_2\big) + b_3\Big)$$

Multi-Layer Perceptron/fully-connected neural network: Vocabulaire et variantes

- **Fully-connected layer** : La matrice est “pleine”
- **Simplifications pour l’étude théorique** :
  - Fully connected **linear** network: $\sigma(t) = t$, $\forall t \in \mathbb{R}$, et $b_h = 0$, pour tout $h = 1..H$.
  - One hidden-layer Neural Network: on prend $H = 2$
- **Fonctions d’activation** :
  - Une zoologie importante : tanh, heavyside, identité, sigmoïde, etc
  - Des non-locales : group-sort, max_pooling
  - La plus utilisée est "Rectified Linear Unit" (ReLU) : $\sigma(t) = \max(0, t)$
  - Souvent, celle associée à la dernière couche est l’identité ou softmax (pour la classification).
- **“Batch-normalisation”** : Au lieu d’optimiser les biais **b**, on règle un couple $(\mathrm{diag}(\gamma), \mathbf{b})$ permettant de centrer les données et fixer leur variance.
- **“Dropout”** : On minimise le risque moyen pour des sous-réseaux aléatoires.
- **L’architecture du réseau:** La donnée des hyperparamètres. Ex : Un réseau ReLU de largeur 100 et de profondeur 10.

Multi-Layer Perceptron/fully-connected neural network

Proposition : Propriétés de f_H, cas linéaire avec biais

Pour tout réseaux linéaire fully-connected, f_H **est affine**:

$$f_H(x) = W_H \cdots W_1 \; x + b'_H$$

où

$$b'_H = W_H \cdots W_2 b_1 + \cdots + W_H b_{H-1} + b_H$$

Preuve:

- Une composition de fonctions affines est affine.
- La formule est facile établir avec une récurrence sur H. (Exercice)

Les réseaux de neurones

Proposition : Propriétés de f_H, cas ReLU

Pour tout réseau de neurones, avec la fonction d'activation ReLU et l'identité sur la dernière couche

1. La fonction f_H est continue
2. La fonction f_H est affine par morceaux
3. Il existe une partition compatible ayant moins de $2^{n_1+\cdots+n_{H-1}}$ morceaux dont les morceaux sont des polyèdres ayant au plus $n_1+\cdots+n_{H-1}$ faces.

- Réciproque dans : Arora, Raman, et al. "Understanding Deep Neural Networks with Rectified Linear Units." ICLR, 2018:
  Toute fonction continue, affine sur un nombre fini de polyhèdres (dont l'union est $\mathbb{R}^{n_0}$) peut être représenté par un réseau de neurone ReLU.

Réseaux convolutifs

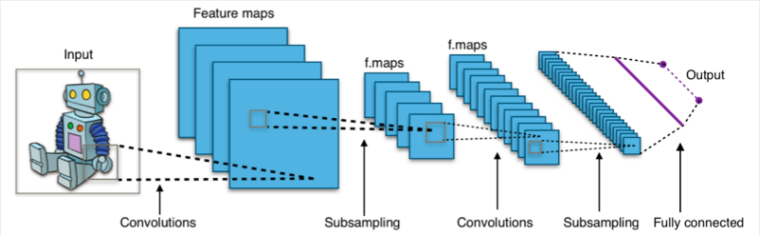
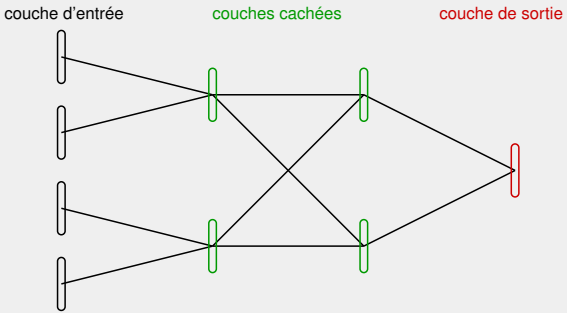


Figure: Source : wikipedia page "apprentissage profond"

Réseaux convolutifs



- Chaque sommet contient un signal/une image. On parle de **canal**.
- Les canaux de la couche d'entrée correspondent aux canaux (ex (r, g, b)) 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 canaux)
- 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_h: la fonction d'activation appliquée à chaque couche
  - Elle applique la même fonction à chaque entrée d'un vecteur

L'action du réseau : la fonction f_H

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

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

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

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

On a:

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

Réseaux convolutifs: Exemple

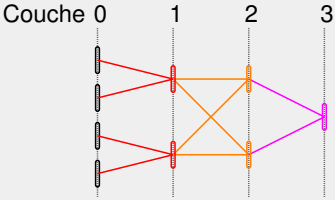


Figure: Réseau convolutif

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)

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

$$f_3(x) = \sigma\Big(W_3\sigma\big(W_2\sigma\big(W_1 x + b_1\big) + b_2\big) + b_3\Big)$$

Réseaux convolutifs: Propriétés

Remarques

- Les formules sont les mêmes que dans le cas fully-connected. Seules les façons de construire les W_h et $f_0(x)$ diffèrent.
- Un réseau convolutif contient des couches convolutives et des couches fully-connected.
- **Un réseau convolutif est un réseau fully-connected dans lequel on impose une structure**.

Proposition : Propriétés de f_H, cas linéaire

Pour tout réseaux convolutif linéaire , f_H **est affine**:

$$f_H(x) = W_H \cdots W_1 \; x + b'_H$$

où

$$b'_H = W_H \cdots W_2 b_1 + \cdots + W_H b_{H-1} + b_H$$

De plus, $x \longmapsto W_H \cdots W_1 x$ est une convolution du signal de départ. (Preuve en exercice)

Réseaux Récurrents

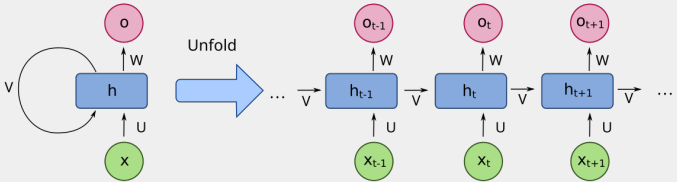


Figure: Source : wikipedia page "recurrent neural networks"

- Modèles de langage : l'entrée $x = (x_i)_{i=1..T}$ est un texte, $x_i = (0,\dots,0,1,0,\dots,0)^T \in \mathbb{R}^{d_{in}}$ est un mot encodé "one-hot" dans un dictionnaire de d_{in} mots. Le RNN prédit le prochain mot.
- Signal : les entrées $x = (x_i)_{i=1..T}$ sont les mesures de capteurs, t est le temps. Le RNN indique si le système fonctionne correctement, ou comment le maintenir dans une configuration, etc

Réseaux Récurrents

- Entrée : La série temporelle $x_1, \ldots, x_T \in \mathbb{R}^{d_{in}}$, pour $d_{in} \in \mathbb{N}$
- État caché : La série temporelle $h_0, \ldots, h_T \in \mathbb{R}^{d_h}$, pour un paramètre $d_h \in \mathbb{N}$, avec $h_0 = 0$ et

$$h_t = \sigma_h\Big(Vh_{t-1} + Ux_t + b_i\Big) \qquad \text{pour tout } t = 1..T$$

  pour
  - La matrice de récurrence : $V \in \mathbb{R}^{d_h \times d_h}$
  - La matrice d'entrée : $U \in \mathbb{R}^{d_h \times d_{in}}$
  - Le biais : $b_i \in \mathbb{R}^{d_h}$
  - l'activation σ_h (souvent l'identité)
- Sortie :
  - Many-to-one : un vecteur

$$V\sigma(h_T) + b_o \quad \in \mathbb{R}^{d_{out}}$$

  - Many-to-many : une série temporelle d'éléments de $\mathbb{R}^{d_{out}}$

$$V\sigma(h_1) + b_o, \cdots, V\sigma(h_T) + b_o$$

  pour $V \in \mathbb{R}^{d_{out} \times d_h}$, $b_o \in \mathbb{R}^{d_{out}}$ et la fonction d'activation σ.

Réseaux Récurrents

Remarques

- On peut déplier les formules du RNN pour obtenir les mêmes même formules que dans le cas fully-connected (exercice). Seules les façons de construire les W_h et $f_0(x)$ diffèrent. Il y a des poids partagés entre les couches.
- **Un réseau récurrent est un réseau fully-connected dans lequel on impose une structure**.
- **Difficultés** : La profondeur du réseaux déplié est T. Quand T est grand (ex: $T = 2000$), cela pose des problèmes de vanishing/exploding gradient (voir chapitre 2).
  On parle de bien gérer les "**dépendances à long terme**", la "**mémorisation**".
- **Principales architectures concurrentes** :
  - Long short-term memory (LSTM), 1997, 114000+ citations google scholar.
  - Gated Recurrent Units (GRU), 2014, 31000+ citations google scholar.
  - State-Space Models (SSM), 2021, 1000 + citations google scholar.

Transformers : Self-attention

- Entrée : La série temporelle $x_1, \dots, x_T \in \mathbb{R}^{d_{in}}$, pour $d_{in} \in \mathbb{N}$, **en lignes**
- On calcule

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{pmatrix} \in \mathbb{R}^{T \times d_{in}} \qquad \text{et} \qquad \begin{cases} Q = XW^q \in \mathbb{R}^{T \times d_h} \\ K = XW^k \in \mathbb{R}^{T \times d_h} \\ V = XW^v \in \mathbb{R}^{T \times d_h} \end{cases}$$

puis le **plongement basé sur l'attention** vaut

$$A = BV \in \mathbb{R}^{T \times d_h},$$

pour le **mécanisme d'attention**

$$B = Soft\_max\left(\frac{1}{\sqrt{d_{in}}} QK^T\right) \in \mathbb{R}^{T \times T}$$

où, pour tout i, j, $\quad Soft\_max(B)_{i,j} = \frac{\exp(B_{i,j})}{\sum_{j'=1}^{T} \exp(B_{i,j'})}$.

- Pour $d_h \in \mathbb{N}$ et les paramètres
  - Une matrice de "query" : $W^q \in \mathbb{R}^{d_{in} \times d_h}$
  - Une matrice de "key" : $W^k \in \mathbb{R}^{d_{in} \times d_h}$
  - Une matrice de "value" : $W^v \in \mathbb{R}^{d_{in} \times d_h}$

Transformers : Multi-headed self-attention

Pour $h \in \mathbb{N}$ (ex : h=8), on a h mécanismes d'attention :

$$(W_i^q, W_i^k, W_i^v)_{i=1..h} \in \left(\mathbb{R}^{d_{in} \times d_h} \times \mathbb{R}^{d_{in} \times d_h} \times \mathbb{R}^{d_{in} \times d_h}\right)^h,$$

pour $d_h = \frac{d_{in}}{h}$. On concatène les résultats

$$A = \left(A_1, A_2, \ldots, A_h\right) \in \mathbb{R}^{T \times d_{in}}.$$

Le mécanisme "Multi-Headed self-attention" est utilisé dans des transformers, GPT, BERT, BART, etc.

Les transformers

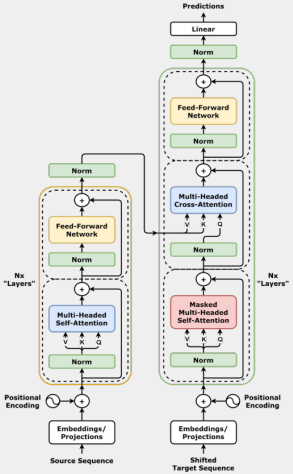


Figure: Source : wikipedia page "Transformer (deep learning architecture)"

Les transformers : Problème

Entreprise	Nom	Année	Nombre de paramètres	temps d'entraînement (en jours)	coût d'entraînement (en dollars)
OpenAI	GPT 1	2018	$117\ 10^6$	quelques	$< 10\ 000$
OpenAI	GPT 2	2019	$1,5\ 10^9$	7-14	50.10^3
OpenAI	GPT 3	2020	$175\ 10^9$	en mois	en millions
MetaAI	BART	2019	$0,4\ 10^9$	7-14	50.10^3

Beaucoup de variantes importantes

- Des architectures pour éviter le “vanishing/exploding gradient”: ex : Couches "Res-net"

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

- Des architectures plus évoluées, dédiés à des applications.
  - En vision : U-Net (segmentation), YOLO (detection), transformer.
  - Séries temporelles et modèles de language : RNN, LSTM, GRU, SSM, transformer.
  - Robotique
  - Graphes
  - Etc
- etc

Modélisation avec des réseaux

- **Autoencoder et VAE** : Un réseau compresse, un réseau décompresse.
- **Création d'embeddings** : Deux réseaux envoient une image et un texte dans un même espace de caractéristiques. (Ex : Visual Query Answering (VQA))
- **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 autre réseau classifiant sur un critère pertinent fonctionne, un troisième réseau classifiant sur un critère non-pertinent échoue.
- etc

Plan

Outline

L'optimisation : Premières propriétés

- On dispose d'un échantillon $(x_i, y_i)_{i=1..n}$.
- On dispose d'un réseau de profondeur H, d'architecture fixée,
  - de paramètre
    $$\theta = (W_H, \ldots, W_1, b_H, \ldots, b_1) \in \Theta$$
  - les fonctions d'activation sont notées σ_h

  La prédiction est la fonction
  $$f_\theta(x) = \sigma_H\Big(W_H \cdots \sigma_2\big(W_2\sigma_1(W_1 x + b_1) + b_2\big) \cdots + b_H\Big)$$
- Les réseaux ReLU:
  - $\sigma_h = ReLU$, pour $h = 1, \ldots, H-1$
  - Cas de la **régression** : $\sigma_H = Id$
  - Cas de la **classification** : σ_H est softmax.
- On considère une fonction coût définie par
  $$\begin{array}{rcl} E : \Theta & \longrightarrow & \mathbb{R} \\ \theta & \longmapsto & E(\theta) = \sum_{i=1}^n L(f_\theta(x_i), y_i) \end{array}$$

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

L'optimisation : Premières propriétés

Proposition: Non-coercivité dans le cas ReLU

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

Preuve utilisant l'homogénéité:
On considère θ avec des biais nuls et, pour tout $\lambda > 0$, on définit θ^λ par

$$W_1^\lambda = \lambda^{H-1} W_1 \qquad \text{et} \qquad W_h^\lambda = \lambda^{-1} W_h, \forall h = 2, \dots, H$$

et des biais nuls.
Comme pour tout $\lambda > 0$, $\sigma(\lambda t) = \lambda \sigma(t)$, on a pour tout x et tout $\lambda > 0$,

$$\begin{aligned} f_{\theta^\lambda}(x) &= \lambda^{-1} W_H \cdots \sigma\left(\lambda^{-1} W_2 \sigma\left(\lambda^{H-1} W_1 x\right)\right) \cdots \\ &= f_\theta(x) \end{aligned}$$

Donc $E(\theta) = E(\theta^\lambda)$, pour tout $\lambda > 0$, mais $\lim_{\lambda \to +\infty} \|\theta^\lambda\| = +\infty$.
Donc E n'est pas coercive.

□

L'optimisation : Premières propriétés

Proposition: Non-convexité

Pour la loss quadratique $L(y', y) = (y' - y)^2$, le réseaux ReLU fully connnected d'architecture $(1, 1, 1)$ et l'échantillon constitué de l'exemple $(x, y) = (1, 0)$, le risque empirique est non convexe.

Preuve: On a pour tout $(w_1, w_2) \in \mathbb{R}^2$ avec $w_1 \geq 0$, et pour $(b_1, b_2) = (0, 0)$

$$E(w_1, w_2, b_1, b_2) = (w_2 \sigma(w_1 x + b_1) + b_2 \ - \ y)^2 = (w_1 w_2)^2.$$

Cette fonction n'est pas convexe car:

- En $(w_1, w_2) = (0, 1)$, $\quad E(0, 1, 0, 0) = 0$
- En $(w_1, w_2) = (1, 0)$, $\quad E(1, 0, 0, 0) = 0$
- En $(w_1, w_2) = (0.5, 0.5) = 0.5 \times (1, 0) + 0.5 \times (0, 1)$,

$$E(0.5, 0.5, 0, 0) = 0.25^2 \ > \ 0 = 0.5 \times E(1, 0, 0, 0) + 0.5 \times E(0, 1, 0, 0).$$

□

Outline

La Rétropropagation

On suppose:

- La fonction coût est de la forme :

$$\theta \longmapsto E(\theta) = \sum_{i=1}^{n} L\big(f_\theta(x_i), y_i\big)$$

  où L est C^1.
- On suppose que, pour tout $h \in \{1, \dots, H\}$, σ_h est C^1
  - On peut régulariser σ_h
  - Il existe un cadre formel pour faire du calcul différentiel et de l'optimisation avec des activations comme ReLU
- Sous ces hypothèses: E **est différentiable**.
- Les frameworks de deep learning utilisent la **différentiation automatique**.
- Ci-dessous, on ne considère qu'un unique exemple (x, y):
  - On somme si besoin plusieurs gradients $\nabla L\big(f_\theta(x_i), y_i\big)$

La Rétropropagation

- $\mathbf{W}_h$ la matrice pour passer de la couche $h-1$ à h,
- $f_\theta^h(x)$, le contenu de la couche h,
- $d_\theta^h(x) = \sigma'_h(\mathbf{W}_h f_\theta^{h-1}(x) + \mathbf{b}_h)$.

Proposition: Gradient pour un réseau fully-connected

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

$$\frac{\partial E}{\partial (\mathbf{W}_h)_{i,j}}(\theta) = \left[f_\theta^{h-1}(x) \right]_j \left[d_\theta^h(x) \right]_i \Delta_i^h(x)$$

$$\frac{\partial E}{\partial (\mathbf{b}_h)_i}(\theta) = \left[d_\theta^h(x) \right]_i \Delta_i^h(x)$$

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

$$\Delta^h(x) = \begin{cases} \nabla_{y_1} L(f_\theta(x), y) & \text{, si } h = H \\ \mathbf{W}_{h+1}^T . \mathrm{diag}\left(d_\theta^{h+1}(x)\right) . \Delta^{h+1}(x) & \text{, sinon} \end{cases}$$

Rq: $\mathbf{W}_{h+1}^T$ remonte d'un niveau dans le réseau: **Rétropropagation**.

Rétropropagation: Problèmes connus

- **“Vanishing/Exploding gradient”** :
  - Si $\mathbf{W}_{h+1}^T.\text{diag}\left(d_\theta^{h+1}(x)\right)$ est systématiquement une contraction $\Longrightarrow$ “Vanishing gradient”
  - Si $\mathbf{W}_{h+1}^T.\text{diag}\left(d_\theta^{h+1}(x)\right)$ augmente systématiquement la norme $\Longrightarrow$ “Exploding gradient”

Rétropropagation: Problèmes connus

Dans certaines régions de l'espace, la fonction objectif est très irrégulière :

Pour une fonction d'activation positivement homogène[1] (notamment ReLU):

Pour $\lambda > 0$, $\lambda \sim 0$

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

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

On a, pour tout x, $f_{\theta^\lambda}(x) = f_\theta(x)$, pour $h > 1$

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{W}_{h,i,j}}(\theta^\lambda) &= \left[f_{\theta^\lambda}^{h-1}(x)\right]_j \left[d_{\theta^\lambda}^h(x)\right]_i (\Delta_\lambda^h)_i(x) \\
&= \lambda^{H-(h-1)} \left[f_\theta^{h-1}(x)\right]_j \left[d_\theta^h(x)\right]_i \lambda^{-(H-h)} \Delta_i^h(x) \\
&= \lambda \frac{\partial E}{\partial \mathbf{W}_{h,i,j}}(\theta)
\end{aligned}$$

mais $\frac{\partial E}{\partial \mathbf{b}_{h,i}}(\theta^\lambda) = \lambda^{-(H-h)} \frac{\partial E}{\partial \mathbf{b}_{h,i}}(\theta)$ et $\frac{\partial E}{\partial \mathbf{W}_{1,i,j}}(\theta^\lambda) = \lambda^{-(H-1)} \frac{\partial E}{\partial \mathbf{W}_{1,i,j}}(\theta)$.

Les petits b_h^λ ont de forts gradients et seront multipliés par des forts W_h^λ.

[1] Attention à la non-différentiabilité en 0

Pour gagner en profondeur, faciliter l'apprentissage

- Couches ResNet

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

- Batch normalisation : On centre (au mieux) les données après chaque couche, à l'aide d'un opérateur diagonal.
- Contraindre les matrices $\mathbf{W}_h$ à être orthogonales ou presque. Bonus: Gain de robustesse.
- Augmenter la largeur. Pb: Complexité = largeur^2.

Plan

Outline

Paysage de la fonction objectif : Introduction

Rappel, pour tout θ:

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

On utilise un algorithme d'optimisation pour trouver un θ, on veut que

$$\widehat{R}(f_\theta) - \inf_\theta \widehat{R}(f_\theta)$$

soit le plus faible possible.

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

Paysage de la fonction objectif : Introduction

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

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

On distingue:

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

$$\forall \theta \in \mathcal{O}, \qquad \widehat{R}(f_{\theta^*}) \leq \widehat{R}(f_\theta)$$

- θ^* **est un point critique du second ordre:**
  - On a

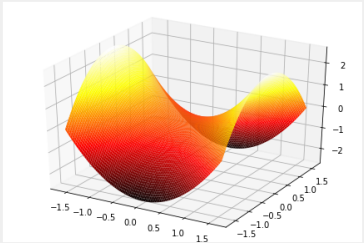
$$\nabla \widehat{R}(f_{\theta^*}) = 0 \qquad \text{et} \qquad \nabla^2 \widehat{R}(f_{\theta^*}) \succeq 0$$

- θ^* **est un point critique du premier ordre:**
  - On a

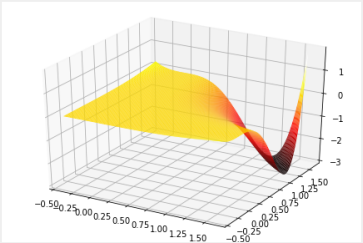
$$\nabla \widehat{R}(f_{\theta^*}) = 0$$

Paysage de la fonction objectif : Introduction

- **θ^* est un point selle** si c'est un point critique qui n'est ni un minimiseur local, ni un maximiseur local
  - **Un point selle θ^* 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 θ^* 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**
- Pour le gradient stochastic, **sans vitesse de convergence**, que les itérés convergent vers un minimiseur local.

- 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.

Outline

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 fully-connected de paramètre $\theta = (\mathbf{W}_1, \ldots, \mathbf{W}_H, \mathbf{b}_1, \ldots, \mathbf{b}_H)$
- des observations $(x^i, y^i)_{i=1..n}$:

$$E(\theta) = \widehat{R}(f_\theta) = \sum_{i=1}^{n} L(f_\theta(x^i) - y^i)$$

Paysage pour les réseaux larges

Théoreme (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 $\theta = (\mathbf{W}_1, \ldots, \mathbf{W}_H, \mathbf{b}_1, \ldots, \mathbf{b}_H)$ de E.
On suppose que $\operatorname{rang}(A) = n$ et que, pour tout $h \in \{1, \ldots, H\}$, $\operatorname{rang}(\mathbf{W}_h) = n_h$.
Alors, on a

$$\widehat{R}(f_\theta) = 0$$

et θ est un minimiseur 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.

Plan

Outline

Geometry-induced implicit regularization: Deep learning context

- Implicit regularization:
  - Neural networks often have far more parameters than training examples.
  - Their performance is not explained by classical arguments.
  - Some global minimizers generalize well, some don't. The optimization finds a solution that does.
- Puzzling empirical observation:
  - Neuron alignment
  - Saddle-to-saddle dynamics
  - Flat-minimas generalize well
- NN are successful independently of the learning objective:
  - In classification, regression, generative models...
- For many unrelated data types:
  - Vectors, images, audio, natural langage, time-series,...

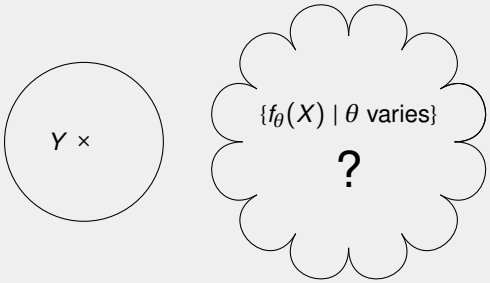
$\longrightarrow$ **Need to study Neural Networks**
independently of the learning objective, algorithm, data type etc

Geometry of deep ReLU neural networks

- Given a deep ReLU architecture
- $X \in \mathbb{R}^{n_0 \times n}$ (resp $Y \in \mathbb{R}^{n_L \times n}$) contains n input (resp output) examples in $\mathbb{R}^{n_0}$
- Denoting $f_\theta(X) \in \mathbb{R}^{n_L \times n}$ the prediction of X by the neural network of parameter $\theta \in \mathbb{R}^p$
- Learning performs a local search solving (for instance)

$$\operatorname{argmin}_\theta \|f_\theta(X) - Y\|_F^2 \quad \Longleftrightarrow \quad \operatorname{argmin}_{Y' \in \{f_\theta(X) \mid \theta \text{ varies}\}} \|Y' - Y\|_F^2$$

- We study the local geometry of two sets:
  - the *image set* $\{f_\theta(X) \mid \theta \text{ varies}\}$;
  - the *pre-image set* $\{\theta' \mid f_{\theta'}(X) = f_\theta(X)\}$.



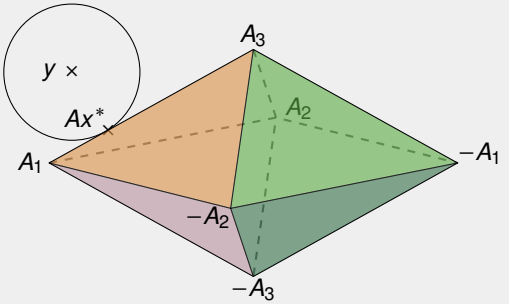
Analogy with ℓ^1 regularization

- Given $A \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$
- Write ℓ^1 regularization under the form

$$\left| \begin{array}{l} \operatorname{argmin}_x \ \|Ax - y\|^2 \\ \|x\|_1 \leq \tau \end{array} \right.$$

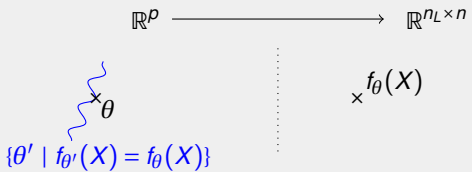
- The geometrical object is:

$$\{Ax \mid \|x\|_1 \leq \tau\} = \tau \operatorname{conv}(A_1, -A_1, \cdots, A_p, -A_p).$$



Geometry of Neural Networks: In parameter space

- We study the local geometry of two sets:
  - the *image set* $\{f_\theta(X) \mid \theta \text{ varies}\}$;
  - the *pre-image set* $\{\theta' \mid f_{\theta'}(X) = f_\theta(X)\}$.



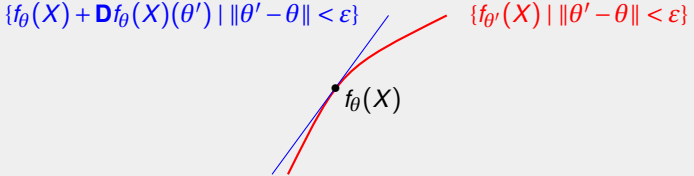
Notation for Deep ReLU architecture

- Depth L, widths $(n_0, \dots, n_L)$,
- $\sigma_l = ReLU$, $\forall l \neq L$, and σ_L is analytic (e.g. the identity of softmax)
- Parameter $\theta = (W_L, \dots, W_1, b_L, \dots, b_1) \in \mathbb{R}^p$
- For all $l = 0, \dots, L$, and all $x \in \mathbb{R}^{n_0}$

$$f_l(x) = \begin{cases} x & \text{if } l = 0 \\ \sigma_l\big(W_l f_{l-1}(x) + b_l\big) & \text{otherwise.} \end{cases}$$

- For all $x \in \mathbb{R}^{n_0}$, $f_\theta(x) = f_L(x)$.
- When differentiable, we denote $\mathbf{D}f_\theta(X) : \mathbb{R}^p \longrightarrow \mathbb{R}^{n_L \times n}$ such that

$$f_{\theta+\theta'}(X) = f_\theta(X) + \mathbf{D}f_\theta(X)(\theta') + o(\|\theta'\|)$$

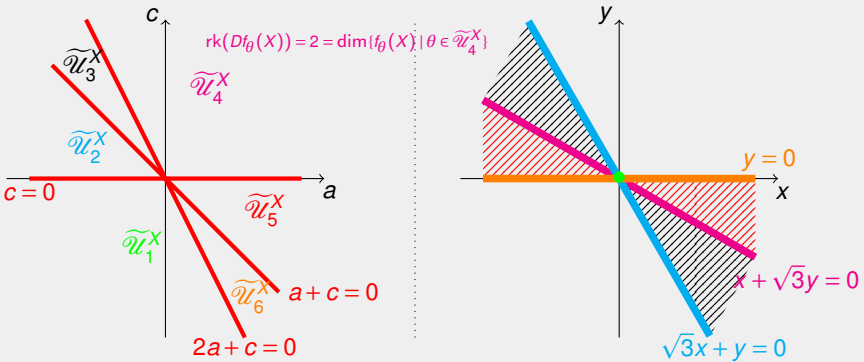


- A key quantity : $\mathrm{rk}\big(\mathbf{D}f_\theta(X)\big) = \dim\{f_{\theta'}(X) \mid \|\theta' - \theta\| < \varepsilon\}$

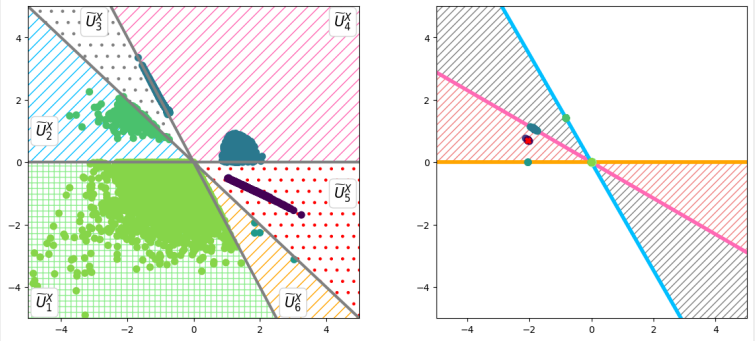
Why does geometry induce regularization? (1/2)

- A ReLU network of parameter $\theta = (a,b,c,d) \in \mathbb{R}^4$: $f_\theta(x) = b\sigma(ax+c)+d$
- A sample $X = (0,1,2)$
- We have $f_\theta(X) = (\ b\sigma(c)+d\ ,\ b\sigma(a+c)+d\ ,\ b\sigma(2a+c)+d\ ) \in \mathbb{R}^{1\times 3}$

$$\theta = (a,b,c,d) \in \mathbb{R}^4 \xrightarrow{f_\theta(X)} \mathbb{R}^{1\times 3} \longrightarrow \mathbb{R}^2 \sim (1,1,1)^\perp$$



Why does geometry induce regularization? (2/2)



Denote

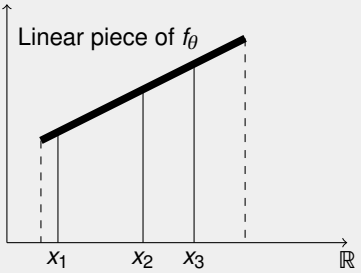
$$d(\theta): \text{ lower-semicontinuous envelop of } \mathrm{rk}(Df_\theta(X))$$

For a learning objective L

$$\underset{\theta}{\text{Minimise}}\ L(f_\theta(X)) \quad \Longleftrightarrow \quad \underset{\theta: d(\theta)\leq k}{\text{Minimise}}\ L(f_\theta(X)), \text{ for } k \in \mathbb{N}$$

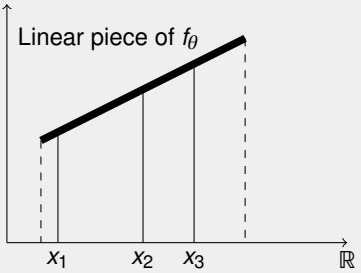
$\Rightarrow$ **Implicit regularization with** $d(\theta)$

What is the effect of this regularization? (1/1)



$Df_\theta(x_1)$, $Df_\theta(x_2)$, $Df_\theta(x_3)$ are linearly dependent
$\Rightarrow \mathrm{rk}(Df_\theta(X))$ is reduced
$\Rightarrow$ Regularization reduces the number of linear pieces 'perceived' by X, fills holes linearly

What is the effect of this regularization? (1/1)

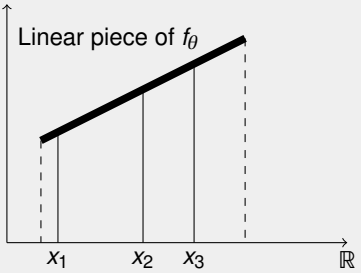


$Df_\theta(x_1)$, $Df_\theta(x_2)$, $Df_\theta(x_3)$ are linearly dependent
$\Rightarrow \mathrm{rk}(Df_\theta(X))$ is reduced
$\Rightarrow$ Regularization reduces the number of linear pieces 'perceived' by X, fills holes linearly

Experiment: Recovery of $y = |x|$

X contains 10 random numbers
ReLU MLP of Width = 20 and depth = 6

What is the effect of this regularization? (1/1)

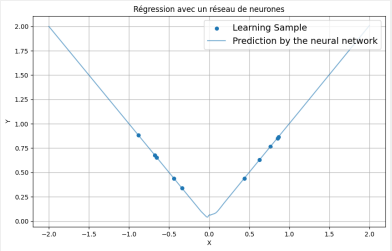


$Df_\theta(x_1)$, $Df_\theta(x_2)$, $Df_\theta(x_3)$ are linearly dependent
$\Rightarrow \mathrm{rk}(Df_\theta(X))$ is reduced
$\Rightarrow$ Regularization reduces the number of linear pieces 'perceived' by X, fills holes linearly

Experiment: Recovery of $y = |x|$

X contains 10 random numbers
ReLU MLP of Width = 20 and depth = 6
init = he_normal, Optimizer: Adam

Learning rate= 0.002, batch size= 5 , 500 epochs

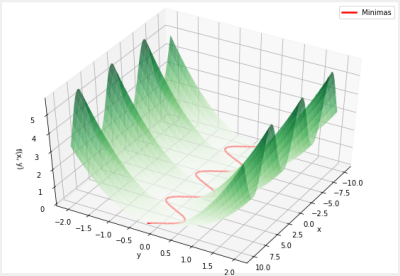


Regularization and minima flatness

Define $L(\theta) \triangleq R(f_\theta(X))$

If θ is a local minimizer of L, and $\theta' \longmapsto f_{\theta'}(X)$ is smooth with constant local dimension in the vicinity of θ,

$\Rightarrow$ there exists $\varepsilon > 0$ s.t. $\{\theta' \mid f_{\theta'}(X) = f_\theta(X), \|\theta' - \theta\| < \varepsilon\}$ are local minimizers of L.



Constant rank theorem: $\dim\{\theta' \mid f_{\theta'}(X) = f_\theta(X), \|\theta' - \theta\| < \varepsilon\} = nb_{param} - d(\theta)$

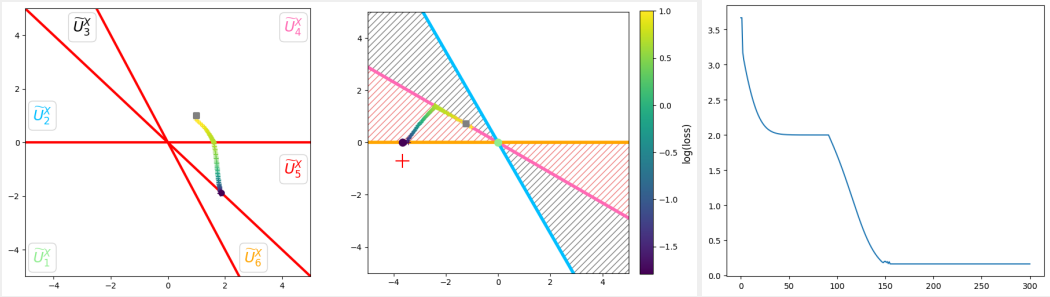
$\Rightarrow$ More regularization

$\Rightarrow$ 'valleys' of larger dimension

$\Rightarrow$ escape is difficult for stoch. algo.

Near a critical point $\theta^{k+1} = \theta^k - \eta(\underbrace{\nabla L(\theta^k)}_{\simeq 0} + noise)$.

Saddle-to-saddle dynamic on the example



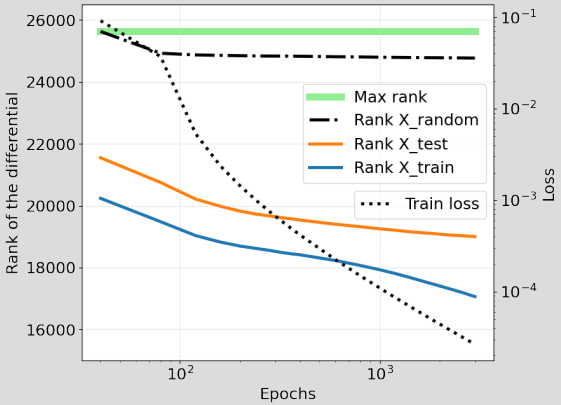
The saddle-to-saddle phenomenon occurs when the sequence $(d(\theta^k))_{k\in\mathbb{N}}$ is increasing.

Experience 1: Comportement pendant l'apprentissage

Description de l'expérience

- architecture $(784, 30, 30, 30, 10)$;
- nombre of parametres: 25720;
- Données d'apprentissage MNIST X_{train} de taille 4000;
- Données de test MNIST X_{test} de taille 20000;
- Données aléatoires X_{random} (Bruit Gaussien) de taille 20000.

Dimension locale pendant l'apprentissage

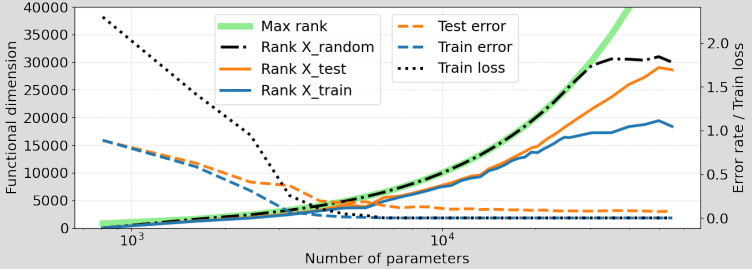


Experience 2: Comportement quand la largeur varie

Description de l'expérience

- architecture $(784, w, w, w, 10)$, pour plusieurs w;
- Données d'apprentissage MNIST X_{train} de taille 4000;
- Données de test MNIST X_{test} de taille 10000;
- Données aléatoires X_{random} (Bruit Gaussien) de taille 40000.

Functional dimensions as the width increases



Outline

Géométrie locale des réseaux ReLU fully-connected.

On note E les arcs du réseaux et B les neurones des couches 1 jusqu'à H.
On note $\theta \in \mathbb{R}^E \times \mathbb{R}^B$ que l'on identifie, si besoin, à
$(W_1, \dots, W_H, b^1, \dots, b^H) \in \left(\mathbb{R}^{n_1 \times n_0} \times \cdots \times \mathbb{R}^{n_H \times n_{H-1}}\right) \times \left(\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_H}\right)$.
On rappelle

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

Pour $n \in \mathbb{N}^*$, on définit la fonction **pattern d'activation**

$$\begin{array}{rcl} a : \mathbb{R}^{n_0 \times n} \times \left(\mathbb{R}^E \times \mathbb{R}^B\right) & \longrightarrow & \{0,1\}^{(n_1 + \cdots + n_{H-1}) \times n} \\ (X, \theta) & \longmapsto & a(X, \theta) \end{array}$$

où

$$a(X,\theta)_{i,j} = \begin{cases} 1 & \text{si } \left[W_h f_{h-1}(x^j) + b_h\right]_v \geq 0 \\ 0 & \text{sinon,} \end{cases}$$

pour v le neurone correspondant à i, $h \in \{1, \dots, H-1\}$ la couche de v, et la j^{ieme} colonne x^j de X.

Pour $X \in \mathbb{R}^{n_0 \times n}$ fixé, elle atteint un nombre fini de valeur que l'on note $\Delta_1^X, \dots, \Delta_{q_X}^X$.

Géométrie locale des réseaux ReLU fully-connected.

On note, pour $X \in \mathbb{R}^{n_0 \times n}$ fixé et pour $j \in \{1, \dots, q_X\}$,

$$\widetilde{\mathcal{U}}_j^X = \mathrm{Int}\left\{\theta \in \mathbb{R}^E \times \mathbb{R}^B \mid a(X,\theta) = \Delta_j^X\right\},$$

et $m_X = \left|\left\{j \in \{1,\dots,q\} \mid \widetilde{\mathcal{U}}_j^X \neq \emptyset\right\}\right|$.
Quitte à changer l'ordre, on suppose que les $\widetilde{\mathcal{U}}_1^X, \dots, \widetilde{\mathcal{U}}_{m_X}^X$ sont non-vides.

Lemme (Prédiction polynomiale par morceaux en θ)

Pour tout $n \in \mathbb{N}^*$, $X \in \mathbb{R}^{n_0 \times n}$, les ensembles $\widetilde{\mathcal{U}}_1^X, \dots, \widetilde{\mathcal{U}}_{m_X}^X$ sont non-vides, ouverts et disjoints

- *Pour tout $j = 1, \dots, m_X$, $\theta \longmapsto f_\theta(X)$ est une fonction polynomiale de degré inférieur à H sur $\widetilde{\mathcal{U}}_j^X$.*
- *Le complémentaire $\left(\cup_{j=1}^{m_X} \widetilde{\mathcal{U}}_j^X\right)^c$ est fermé et de mesure de Lebesgue nulle.*

Rq : La fonction $\theta \longmapsto f_\theta(X)$ est une composition de fonctions continues, elle est continue.

Géométrie locale des réseaux ReLU fully-connected.

Pour $n \in \mathbb{N}^*$, $X \in \mathbb{R}^{n_0 \times n}$, et $j \in \{1, \dots, m_X\}$, on note

$$r_j^X = \max_{\theta \in \widetilde{\mathcal{U}}_j^X} \mathrm{rk}(\mathbf{D}f_\theta(X)) \qquad \text{et} \qquad \mathcal{U}_j^X = \{\theta \in \widetilde{\mathcal{U}}_j^X \mid \mathrm{rk}(\mathbf{D}f_\theta(X)) = r_j^X\}.$$

Théoreme (J. Bona-Pellissier, F. Bachoc, F. Malgouyres 2024)

Pour tout réseau ReLU, $n \in \mathbb{N}^*$, $X \in \mathbb{R}^{n_0 \times n}$, par définition

- *$\mathcal{U}_1^X, \dots, \mathcal{U}_{m_X}^X$ sont non-vides et disjoints;*
- *Pour tout $j \in \{1, \dots, m_X\}$, la fonction $\theta \mapsto a(X, \theta)$ est constante sur $\mathcal{U}_j^X$ et prend des valeurs distinctes pour $j' \neq j$;*
- *Pour tout $j \in \{1, \dots, m_X\}$, $\theta \mapsto \mathrm{rk}(\mathbf{D}f_\theta(X))$ est constante sur $\mathcal{U}_j^n$ et vaut r_j^X.*

De plus,

- *Les ensembles $\mathcal{U}_1^X, \dots, \mathcal{U}_{m_X}^X$ sont ouverts;*
- *$\left(\cup_{j=1}^{m_X} \mathcal{U}_j^X\right)^c$ est fermé et de mesure de Lebesgue nulle;*
- *Pour tout $j \in \{1, \dots, m_X\}$, $\theta \mapsto f_\theta(X)$ est une fonction polynomiale de degré inférieur à H sur $\mathcal{U}_j^X$.*

Géométrie locale des réseaux ReLU fully-connected.

Corollaire

Pour tout réseaux ReLU fully-connected, profond.
Pour tout $n \in \mathbb{N}^*$, $X \in \mathbb{R}^{n_0 \times n}$, $j \in \{1, \dots, m_X\}$ *et* $\theta \in \mathscr{U}_j^X$, *il existe* $\varepsilon_{X,\theta} > 0$ *tel que*

- *L'image locale*

$$\left\{ f_{\theta'}(X) \in \mathbb{R}^{n_L \times n} \mid \|\theta' - \theta\| < \varepsilon_{X,\theta} \right\}$$

  est une variété régulière de dimension $\mathrm{rk}\left(\mathbf{D} f_\theta(X)\right)$*;*

- *la pre-image*

$$\left\{ \theta' \in \mathbb{R}^E \times \mathbb{R}^B \mid f_{\theta'}(X) = f_\theta(X) \quad \text{and} \quad \|\theta' - \theta\| < \varepsilon_{X,\theta} \right\}$$

  est une variété régulière de dimension $|E| + |B| - \mathrm{rk}\left(\mathbf{D} f_\theta(X)\right)$.

C'est une conséquence directe du théorème précédent et du théorème du rang constant.

Théorème du rang constant.

Theorem (Théorème du rang constant)

Soient $U \subset \mathbb{R}^n$ un ouvert, $a \in U$, et $f : U \to \mathbb{R}^p$ une application de classe $\mathscr{C}^1$. Si la différentielle de f est de rang constant r sur U, alors il existe :

- *un $\mathscr{C}^1$-difféomorphisme φ d'un ouvert $V \subset \mathbb{R}^n$ contenant 0 sur un ouvert de U, avec $\varphi(0) = a$, et*
- *un $\mathscr{C}^1$-difféomorphisme ψ d'un ouvert de $\mathbb{R}^p$ contenant $f(\varphi(V))$ sur un ouvert de $\mathbb{R}^p$, avec $\psi(f(a)) = 0$,*

tels que :

$$\forall x = (x_1, \dots, x_n) \in V, \quad (\psi \circ f \circ \varphi)(x) = (x_1, \dots, x_r, 0, \dots, 0).$$

On note $r = \mathrm{rk}\left(\mathbf{D}f_\theta(X)\right)$, $n = |E| + |B|$, $p = nn_L$, $a = \theta$, $f : \theta' \mapsto f_{\theta'}(X)$.
On prend $\varepsilon_{X,\theta}$ tel que $B(\theta, \varepsilon_{X,\theta}) \subset \varphi(V)$ et on note $V' = \varphi^{-1}\left(B(\theta, \varepsilon_{X,\theta})\right)$.

$$\varphi \quad \text{est une carte de} \quad V' \cap \left(\{0\} \times \mathbb{R}^{n-r}\right) \quad \text{sur} \quad \left\{\theta' \in \mathbb{R}^E \times \mathbb{R}^B \mid f_{\theta'}(X) = f_\theta(X) \quad \text{and} \quad \|\theta' - \theta\| < \varepsilon_{X,\theta}\right\},$$

et

$$\psi^{-1} \quad \text{est une carte de} \quad \psi \circ f \circ \varphi\left(V' \cap \left(\mathbb{R}^r \times \{0\}\right)\right) \quad \text{sur} \quad \left\{f_{\theta'}(X) \in \mathbb{R}^{n_L \times n} \mid \|\theta' - \theta\| < \varepsilon_{X,\theta}\right\}.$$

Regularization induite par la géométrie

$$\dim^-(\theta, X) = \lim_{\epsilon \to 0} \min_{j : B(\theta,\epsilon) \cap \mathscr{U}_j^X \neq \emptyset} r_j^X,$$
$$\dim^+(\theta, X) = \lim_{\epsilon \to 0} \max_{j : B(\theta,\epsilon) \cap \mathscr{U}_j^X \neq \emptyset} r_j^X.$$

$$(P) : \operatorname*{Minimise}_{\theta} R(f_\theta(X)).$$
$$(P_k) : \operatorname*{Minimise}_{\theta : \dim^-(\theta,X) \leq k} R(f_\theta(X)), \qquad \text{pour } k \in \mathbb{N} \text{ donné.}$$

Corollaire

Pour tout $n \in \mathbb{N}^*$, $X \in \mathbb{R}^{n_0 \times n}$, et toute fonction objectif régulière $R : \mathbb{R}^{n_L \times n} \longrightarrow \mathbb{R}$,

- *Pour tout $\theta^* \in \bigcup_{j=1}^{p_X} \mathscr{U}_j^X$. On note $k = \operatorname{rk}(\mathbf{D} f_{\theta^*}(X))$.*

$$\theta^* \text{ est un point critique de } (P) \iff (\theta^*, 1) \text{ satisfait les conditions KKT de } (P_k).$$

- *Pour tout $\theta^* \in \mathbb{R}^E \times \mathbb{R}^B$. On note $k = \dim^+(\theta^*, X)$. Nous avons*

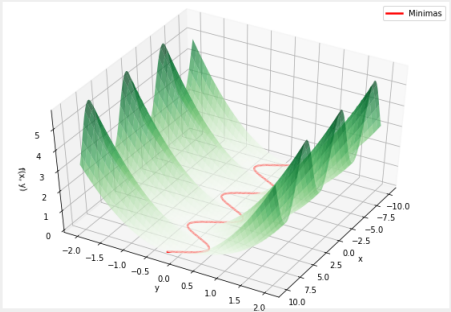
$$\theta^* \text{ est un minimiseur local de } (P) \iff \theta^* \text{ est un minimiseur local de } (P_k).$$
$$\theta^* \text{ est un point selle de } (P) \iff \theta^* \text{ est un point selle de } (P_k).$$

Minima flatness

Any local minimizer $\theta \in \cup_{j=1}^{p_X} \mathscr{U}_j^X$ of (P) is dimension $\left(|E| + |B| - \mathrm{rk}\left(\mathbf{D} f_\theta(X)\right)\right)$ flat.

Définition

A local minimizer θ of (P) is said to be **dimension** k **flat**, *for $k \in \mathbb{N}$, if and only if there exist $\varepsilon > 0$ and a smooth manifold $\mathscr{M} \subset \mathbb{R}^R \times \mathbb{R}^B$ of dimension k such that $\theta \in \mathscr{M}$, and every $\theta' \in \mathscr{M} \cap B(\theta, \varepsilon)$ is also a local minimizer of (P).*



Lien avec le nombre de zones affines perçues par X: Cas une couche cachée

On considère un réseau ReLU de largeurs $(1, n_1, 1)$, avec $\sigma_2 = Id$.
On rappelle que $a(X,\theta) \in \{0,1\}^{n_1 \times n}$, et pour $k \in [\![1, n_1]\!]$, $i \in [\![1, n]\!]$

$$a(X,\theta)_{k,i} = \begin{cases} 1 & \text{if } [W_1 x^{(i)} + b_1]_k \geq 0, \\ 0 & \text{sinon.} \end{cases}$$

On note, pour tout X et θ,

$$\mathscr{A}(X,\theta) = \{\delta \in \{0,1\}^{n_1} \mid \text{il existe } i \in [\![1,n]\!] \text{ satisfaisant } a(x^{(i)},\theta) = \delta\}.$$

Pour $i \in [\![1, n]\!]$,

$$\begin{cases} \mathbf{e}_i = (0,\dots,0,\underset{\uparrow}{0}_{i},x^{(i+1)} - x^{(i)},\dots,x^{(n)} - x^{(i)}), \\ \mathbf{e}_{n+i} = (x^{(i)} - x^{(1)},\dots,x^{(i)} - x^{(i-1)},\underset{\uparrow}{0}_{i},0,\dots,0). \end{cases}$$

Ainsi que $\mathbf{e}_0 = \mathbf{e}_{2n}$. On définit aussi, pour toutl $i \in [\![1, n]\!]$,

$$\mathbb{1}_i = (0,\dots,0,\underset{\uparrow}{1}_{i},\dots,1) \in \mathbb{R}^{1\times n} \quad \text{and} \quad \mathbb{1}_{n+i} = (1,\dots,1,\underset{\uparrow}{0}_{i},\dots,0) \in \mathbb{R}^{1\times n}.$$

Lien avec le nombre de zones affines perçues par X: Cas une couche cachée

Théoreme

On considère une architecture ReLU (E, V, Id), avec $L = 2$ et $n_0 = n_2 = 1$. On considère $n \in \mathbb{N}^*$, et un échantillon $X = (x^{(1)}, x^{(2)}, \dots, x^{(n)}) \in \mathbb{R}^{1\times n}$ tel que

$$x^{(1)} < x^{(2)} < \cdots < x^{(n)}.$$

Pour tout $j \in [\![1, p_X]\!]$, il existe $\alpha \in \{1, \dots, 2n\}^{n_1}$ tel que pour tout $\theta \in \mathscr{U}_j^X$ et tout $k \in [\![1, n_1]\!]$, $a(X,\theta)_{k,:} = \mathbb{1}_{\alpha_k}$, et

$$\mathrm{rk}\left(\mathbf{D}f_\theta(X)\right) = \mathrm{rk}\left(\mathbb{1}, \mathbf{e}_{\alpha_1 - 1}, \mathbf{e}_{\alpha_1}, \dots, \mathbf{e}_{\alpha_{n_1} - 1}, \mathbf{e}_{\alpha_{n_1}}\right).$$

On a donc $\mathscr{U}_j^X = \widetilde{\mathscr{U}_j^X}$ et, pour tout $\theta \in \mathscr{U}_j^X$,

$$\frac{1}{2}|\mathscr{A}(X,\theta)| \leq \mathrm{rk}\left(\mathbf{D}f_\theta(X)\right) \leq 2|\mathscr{A}(X,\theta)|.$$

Plan

Outline

Paysage de la fonction objectif : Introduction

Rappel, pour tout θ:

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

On utilise un algorithme d'optimisation pour trouver un θ, on veut que

$$\widehat{R}(f_\theta) - \inf_\theta \widehat{R}(f_\theta)$$

soit le plus faible possible.

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

Paysage de la fonction objectif : Introduction

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

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

On distingue:

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

$$\forall \theta \in \mathcal{O}, \qquad \widehat{R}(f_{\theta^*}) \leq \widehat{R}(f_\theta)$$

- θ^* **est un point critique du second ordre:**
  - On a

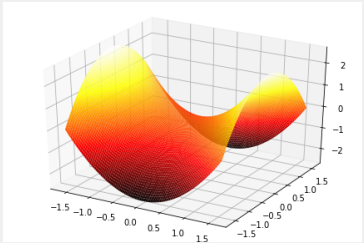
$$\nabla \widehat{R}(f_{\theta^*}) = 0 \qquad \text{et} \qquad \nabla^2 \widehat{R}(f_{\theta^*}) \succeq 0$$

- θ^* **est un point critique du premier ordre:**
  - On a

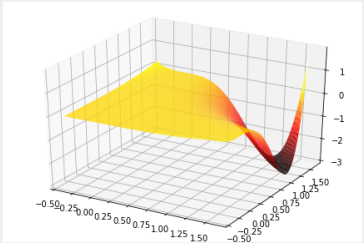
$$\nabla \widehat{R}(f_{\theta^*}) = 0$$

Paysage de la fonction objectif : Introduction

- **θ^* est un point selle** si c'est un point critique qui n'est ni un minimiseur local, ni un maximiseur local
  - **Un point selle θ^* 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 θ^* 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**
- Pour le gradient stochastic, **sans vitesse de convergence**, que les itérés convergent vers un minimiseur local.

- 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.

Outline

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", NeurIPS 2016
- .... (Notamment dans les groupes de S. Arora à Princeton, de F. Bach à l'ENS)
- E. Achour, S. Gerchinovitz, F. Malgouyres, "The loss landscape of deep linear neural networks: a second-order analysis", Journal of Machine Learning Research, 2024.

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 $\mathbf{W} = (W_1, \ldots, W_H)$ *un point critique du premier ordre de* $\widehat{R}$ *et* $r = rk(W_H \cdots W_1)$.
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* $\mathscr{S}$.

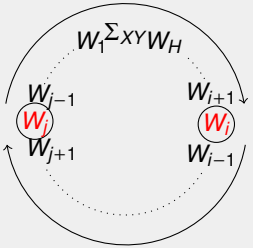
Proposition

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

Paysage pour les réseaux linéaires

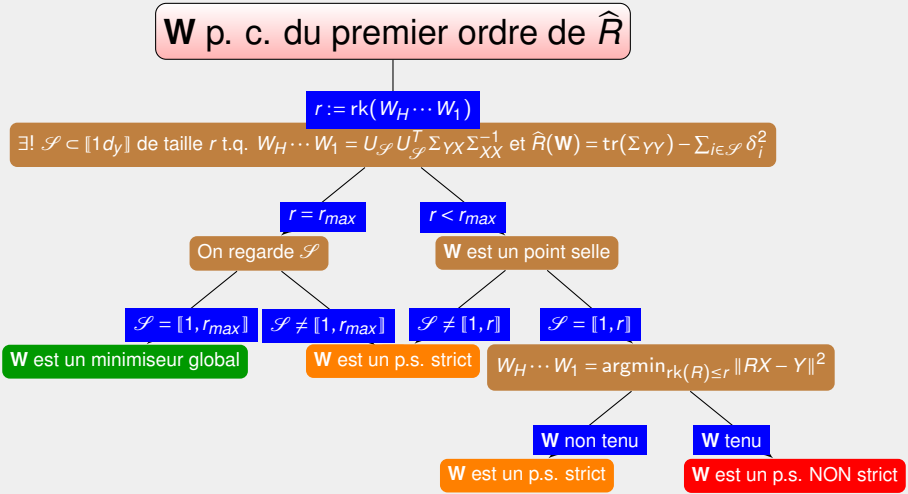


Figure: Theorem [E. Achour et al.]

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 $[\![1, r]\!]$ tenus et non-tenus.*

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:*

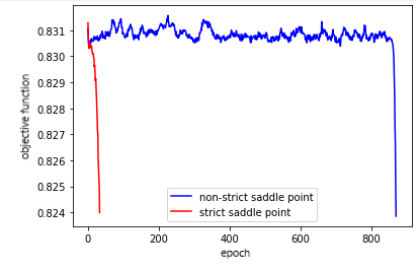
$$W_H = [U_{\mathscr{S}_{max}}, U_{Q_{max}} A_R] D_{H-1}^{-1}$$

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

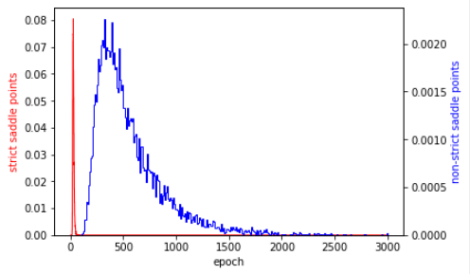
$$W_1 = D_1 \begin{bmatrix} U_{\mathscr{S}_{max}}^T \Sigma_{YX} \Sigma_{XX}^{-1} \\ M_D \end{bmatrix}.$$

Paysage pour les réseaux linéaires

Empiriquement, on trouve:



(a) Initialisation au voisinage d'un point selle strict vs non-strict



(b) Histogramme des epoch d'échappement

Paysage pour les réseaux linéaires

Conclusion [E. Achour et al.]

- **Classification** de l'ensemble des points critiques en: 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**

Outline

Paysage pour les réseaux linéaires (cas à 1 couche cachée)

On simplifie

- Cas $H = 2$
- On note pour $A \in \mathbb{R}^{n_2 \times n_1}$ et $B \in \mathbb{R}^{n_1 \times n_0}$

$$E(A, B) = \sum_{i=1}^{L} \|y_i - ABx_i\|^2$$

- On note pour $D \in \mathbb{R}^{n_2 \times n_0}$

$$F(D) = \sum_{i=1}^{L} \|y_i - Dx_i\|^2$$

- On note

$$\Sigma_{XX} = \sum_{i=1}^{L} x_i x_i^T \in \mathbb{R}^{n_0 \times n_0} \quad , \quad \Sigma_{XY} = \sum_{i=1}^{L} x_i y_i^T \in \mathbb{R}^{n_0 \times n_2}$$

$$\Sigma_{YX} = \sum_{i=1}^{L} y_i x_i^T \in \mathbb{R}^{n_2 \times n_0} \quad , \quad \Sigma_{YY} = \sum_{i=1}^{L} y_i y_i^T \in \mathbb{R}^{n_2 \times n_2}$$

Paysage pour les réseaux linéaires (cas à 1 couche cachée)

Remarques

On a

1. Pour tout $C \in \mathbb{R}^{n_1 \times n_1}$ inversible, $AB = (AC)(C^{-1}B) = A'B'$
2. Si Σ_{XX} est inversible alors
$$\Sigma_{YX}\Sigma_{XX}^{-1} \in \operatorname{argmin}_D F(D)$$
3. Soit $M \in \mathbb{R}^{n \times p}$ avec $p \leq n$ de rang p. Pour tout $x \in \mathbb{R}^n$, la projection $P_M(x)$ de x sur l'espace vectoriel généré par les colonnes de M vaut
$$P_M(x) = M(M^T M)^{-1} M^T x$$

Paysage pour les réseaux linéaires (cas à 1 couche cachée)

Lemme 1

Si Σ_{XX} est inversible. Soient $A \in \mathbb{R}^{n_2 \times n_1}$ telle que $\text{rang}(A) = n_1$ et $B \in \mathbb{R}^{n_1 \times n_0}$. Alors, (A, B) est un point critique du premier ordre de E si et seulement si

$$AB\Sigma_{XX}B^T = \Sigma_{YX}B^T \qquad \text{et} \qquad B = (A^TA)^{-1}A^T\Sigma_{YX}\Sigma_{XX}^{-1}$$

Lemme 2

Si Σ_{XX} est inversible. Soient $A \in \mathbb{R}^{n_2 \times n_1}$ telle que $\text{rang}(A) = n_1$ et $B \in \mathbb{R}^{n_1 \times n_0}$.
Les deux assertions suivantes sont équivalentes:

1. (A, B) est un point critique du premier ordre de E
2. $$AB = P_A\Sigma_{YX}\Sigma_{XX}^{-1}$$

   et

   $$P_A\Sigma = P_A\Sigma P_A = \Sigma P_A$$

   pour

   $$\Sigma = \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY} \quad \in \mathbb{R}^{n_2 \times n_2}$$

Paysage pour les réseaux linéaires (cas à 1 couche cachée)

On diagonalise (Σ est symétrique)

$$\Sigma = U \Lambda U^T$$

avec $\Lambda \in \mathbb{R}^{n_2 \times n_2}$ diagonale et $U \in \mathbb{R}^{n_2 \times n_2}$ unitaire.
Pour $\mathscr{S} \subset \{1, \cdots, n_2\}$, on note $U_{\mathscr{S}}$ la matrice extraite de U en prenant les colonnes d'indice dans $\mathscr{S}$.

Proposition 1: Paramétrisation des points critiques

Si Σ_{XX} est inversible. Soient $A \in \mathbb{R}^{n_2 \times n_1}$ et $B \in \mathbb{R}^{n_1 \times n_0}$ avec A telle que $\mathrm{rang}(A) = n_1$.

On suppose que les valeurs propres de Σ sont distinctes.

Alors, (A, B) est un point critique du premier ordre de E si et seulement si il existe $C \in \mathbb{R}^{n_1 \times n_1}$ inversible et $\mathscr{S} \subset \{1, \cdots, n_2\}$ de taille n_1 tels que

$$A = U_{\mathscr{S}} C \qquad \text{et} \qquad B = C^{-1} U_{\mathscr{S}}^T \Sigma_{YX} \Sigma_{XX}^{-1}$$

Paysage pour les réseaux linéaires (cas à 1 couche cachée)

Pour simplifier les notations, on suppose les valeurs propres de Σ ordonnées:

$$\lambda_1 > \lambda_2 > \cdots > \lambda_{n_2}$$

Proposition 2: minimiseur global $\Leftrightarrow$ minimiseur local

Si Σ_{XX} est inversible. Soient $A \in \mathbb{R}^{n_2 \times n_1}$ et $B \in \mathbb{R}^{n_1 \times n_0}$ avec A telle que $\text{rang}(A) = n_1$.

On suppose que les valeurs propres de Σ sont distinctes.

1. pour tout point critique du premier ordre (A, B) de E et pour $\mathscr{S}$ définissant A et B (voir la proposition précédente), on a
   1. $$AB = P_{U_{\mathscr{S}}} \Sigma_{YX} \Sigma_{XX}^{-1}$$
   2. $$E(A, B) = \text{trace}(\Sigma_{YY}) - \sum_{i \in \mathscr{S}} \lambda_i$$
2. Si (A, B) est un minimiseur global alors c'est un point critique du premier ordre associé à $\mathscr{S} = \{1, \cdots, n_1\}$.
3. (A, B) **est minimiseur local si et seulement si c'est un minimiseur global**.

Merci pour votre attention !