Réseaux de Neurones

Contexte : $\hat{f} \in \underset{f \in F}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \Omega(f)$

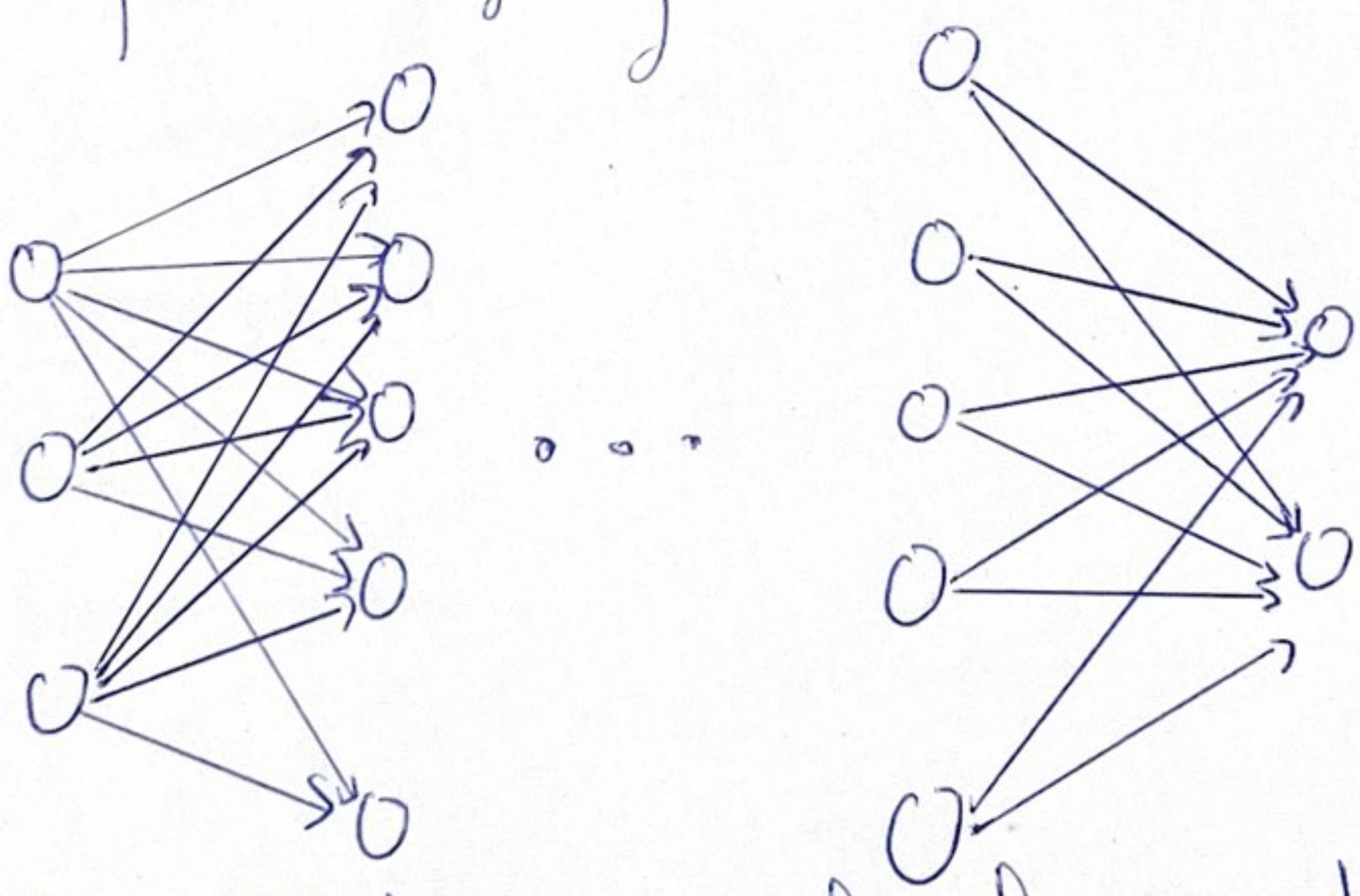
Réseaux de neurones :

Une classe de fonctions F qui est facilement représentable sur un ordinateur et pour laquelle il est facile de faire les calculs nécessaires à l'inférence et à l'optimisation.

Le graphe des calculs d'une fonction $f \in F$ doit être acyclique.

Exemples :

- FeedForward : fully connected



$$output = A_{out}\,\sigma\left(A_n \cdots \sigma\left(A_2\,\sigma\left(A_1\,input + b_1\right) + b_2\right) \cdots + b_n\right) + B_{out}$$

- Plus généralement : Un graphe de calcul orienté sans cycle dont les noeuds sont paramétrés.

Exemples de noeuds :

- Fonctions d'activations
  - Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$
  - $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
  - $\text{Relu}(x) = \max(x, 0)$
  - $\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- Fonctions linéaires
  - Transformation affine $(x) = Ax + b$
  - Convolutions $(x) = x * k$ (1D, 2D, ...)
- Fonctions utilitaires
  - Pooling
  - Normalisation
  - Clipping.

I. Optimisation et règle de la chaine

1) SGD

Les réseaux de neurones sont entrainés avec SGD, dont la fonction objectif est

$$f(\theta) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, \underbrace{f_\theta}_{\text{réseau de neurones paramétré par } \theta}(x_i)) + \Omega(\theta)$$

Il est alors possible de constraire un estimateurs du gradient en tirant aléatoirement $B \subset \{1, \dots, n\}$ (de taille fixe ou aléatoire) et d'utiliser l'estimateur

$$\nabla_B f(\theta) = \frac{1}{|B|} \sum_{b \in B} l(y_b, f_\theta(x_b)) + \Omega(\theta).$$

B est appelé le "batch".

L'algorithme d'entrainement est alors :

Commencer à θ_0
Tant que (critère à vérifier pour continuer)
$\theta_t \leftarrow \theta_{t-1} - \gamma_t \nabla_B f(\theta_{t-1})$ (B est tiré à chaque étape)

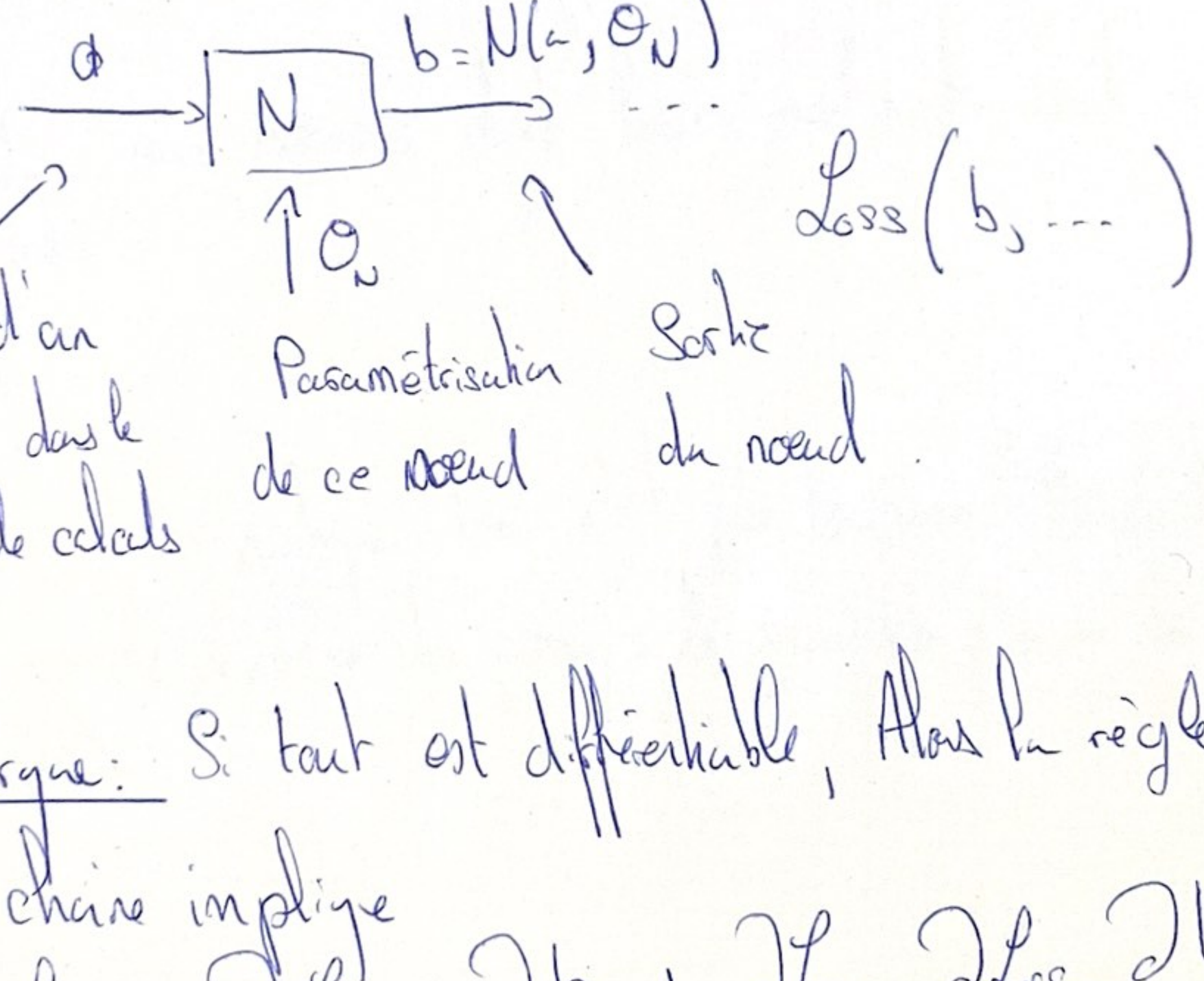
Remarque 1 :

. En pratique, B n'est pas toujours aléatoire et il est possible de voir les données dans un ordre déterministe.

Remarque 2 :

En pratique, il existe des algorithmes plus avancés (Adagrad, RMSProp, Adam, ...), mais ils se basent tous sur la construction de gradients stochastiques.

2) Backpropagation : Comment calculer les gradients ?



Remarque : Si tout est différentiable, Alors la règle de la chaine implique

$$\frac{\partial Loss}{\partial a}(a) = \frac{\partial Loss}{\partial b}(b)\frac{\partial b}{\partial a}(a) \text{ et } \frac{\partial Loss}{\partial \theta_N}(a) = \frac{\partial Loss}{\partial b}(b)\frac{\partial b}{\partial \theta_N}(a)$$

Algorithme de Backpropagation :

- (Forward) : Calculer Loss normalement en gardant la valeur de N(a) pour chaque noeud (les activations)
- (Backward) : Remonter le graphe de calculs en partant de la fin.

Pour chaque noeud, on calcule $\frac{\partial Loss}{\partial inputs}$ et $\frac{\partial Loss}{\partial params}$ avec les formules précédentes, puis on pusce $\frac{\partial Loss}{\partial inputs}$ aux noeuds dont leurs outputs sont nos inputs.

Remarque : En pratique, tout n'est pas toujours différentiable, il est alors possible d'utiliser des "proxy's de gradients (comme les sous-gradients).

Remarque: Le problème étant non convexe, SGD n'a pas vraiment de garanties fortes de convergence. Cependant, il a été observé que le réseau converge vers l'optimum avec beaucoup de paramètres, et il est possible d'obtenir des garanties dans les régimes avec une infinité de neurones.

II. Réseaux reliés à une couche cachée

$$f(x) = \sum_{j=1}^{m} \eta_j \left(\omega_j^T x + b_j\right)_+$$

Modèle déjà très riche qui peut être étudié "simplement".

1) Propriété d'homogénéité

Si $\alpha_j > 0$, le réseau est invariant par la transformation

$$\begin{cases} \eta_j \leftarrow \eta_j \alpha_j \\ \omega_j \leftarrow \omega_j / \alpha_j \\ b_j \leftarrow b_j / \alpha_j \end{cases}$$

Il s'agit de la propriété d'homogénéité des

réseaux Relu.

Remarque : Il existe donc plusieurs paramétrisations donnant les mêmes fonctions

Remarque : Il est donc possible d'envisager l'étude théorique comme restreinte à un sous-ensemble de poids.

2) Régularisation

En ajoutant pour chaque j un terme de pénalisation de la forme $\alpha \eta_j^2 + \|w_j\|_2^2 + b_j^2/M^2$ pour homogénéiser, ce qui est la même chose que de pénaliser par

$$\alpha \alpha_j^2 \eta_j^2 + \left(\|w_j\|_2^2 + b_j^2/M^2\right)/\alpha_j^2 \quad , \forall \alpha_j > 0$$

par homogénéité, on voit que c'est équivalent (en optimisant en α_j) à pénaliser par

$$2|\eta_j| \left(\|w_j\|_2^2 + b_j^2/M^2\right)^{1/2}.$$

Ainsi, par homogénéité, on peut se restreindre au cas $\|w_j\|_2^2 + b_j^2/M^2 = 1$ avec une pénalisation $\alpha |\eta_j|$.

Pour simplicité, nous allons nous restreindre au cas où les paramètres sont ~~restreints~~ contraints. Pour le cas général, merci de lire (Bach 2024).

III. Erreur de généralisation - Cas contraint

Le paragraphe précédent a servi d'heuristique pour justifier pourquoi nous pouvons nous restreindre au cas

$$\begin{cases} \forall j, \|w_j\|_2^2 + b_j^2 / R^2 = 1 \\ \|A\|_1 \leq D \end{cases}$$

~~3. de plus~~

Notons (H) l'ensemble des paramètres vérifiant ces contraintes.

De plus, faisons les hypothèses suivantes :

- $\|x\| \leq R$ p.s.
- ℓ est G-Lipschitz en son second argument.

Analyse par Rademacher :

D'après le principe de symétrisation :

$$\mathbb{E}\left(R(\hat{f}) - \inf_{\theta} R(f_\theta)\right) \leq 2R_n(\mathcal{H}) \quad (4)$$

où $R_n(\mathcal{H}) = \mathbb{E}_{\substack{x,y \\ \varepsilon_1,\dots,\varepsilon_n \overset{iid}{\sim} \mathcal{R}(\frac{1}{2})}} \left(\sup_{\theta} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i l(y_i, f_\theta(x_i)) \right)$.

Par principe de contraction lipschitz,

$$R_n(\mathcal{H}) \leq G\, \mathbb{E}\left(\sup_{\theta} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i f_\theta(x_i) \right)$$

$$= G\, \mathbb{E}\left(\sup_{\theta} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \eta_j \left(\omega_j^T x_i + b_j\right)_+ \right)$$

En réinjectant les contraintes et en utilisant le résultat de dualité $\sup_{\|\eta\|_1 \leq D} \zeta^T \eta = D\|\zeta\|_\infty$:

$$R_n(\mathcal{H}) \leq G.\mathbb{E}\left(\sup_{j} \sup_{\|\omega_j\|^2 + b_j^2/n^2 = 1} D \left| \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i \left(\omega_j^T x_i + b_j\right)_+ \right| \right)$$

De plus, comme l'information est redondante en j :

$$R_n(w) \leq GD \, \mathbb{E}\left(\sup_{\|w\|_2^2 + \|b\|^2/R^2} \left| \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i (w_j^T x_i + b_j)_+ \right| \right)$$

Nous aimerions utiliser le fait que $(\cdot)_+$ est 1-Lipschitz, mais les 1-1 font que nous ne sommes pas dans le cas vu en cours.

À la place, nous avons besoin de la version plus générale suivante :

Proposition : Si $\forall i$, ϕ_i est L-Lipschitz, et $\phi_i(0) = 0$, alors

$$\mathbb{E}_\varepsilon \left(\sup_\theta \left| \sum_{i=1}^{n} \varepsilon_i \phi_i(a_i(\theta)) \right| \right) \leq 2 \, \mathbb{E}_\varepsilon \left(\sup_\theta \left| \sum_{i=1}^{n} \varepsilon_i a_i(\theta) \right| \right)$$

Démonstration : Admis.

Alors $R_n(\theta) \leq 2 \, GD \, \mathbb{E}\left(\sup_{\|w\|_2^2 + b^2/R^2 = 1} \left| w^T \left(\frac{1}{n} \sum_{i=1}^{n} \varepsilon_i x_i \right) + b \left(\frac{1}{n} \sum_{i=1}^{n} \varepsilon_i \right) \right| \right)$

$$\left(\text{dualité} \right) = 2 \, GD \, \mathbb{E}\left(\sqrt{ \left\| \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i x_i \right\|_2^2 + R^2 \left(\frac{1}{n} \sum_{i=1}^{n} \varepsilon_i \right)^2 } \right)$$

$$\leq 2GD\sqrt{\mathbb{E}\left(\quad " \quad \right)}$$

(Jensen)

$$= 2GD\sqrt{\left(\frac{1}{n}\mathbb{E}\left(\|x\|_2^2\right) + \frac{h^2}{n}\right)}$$

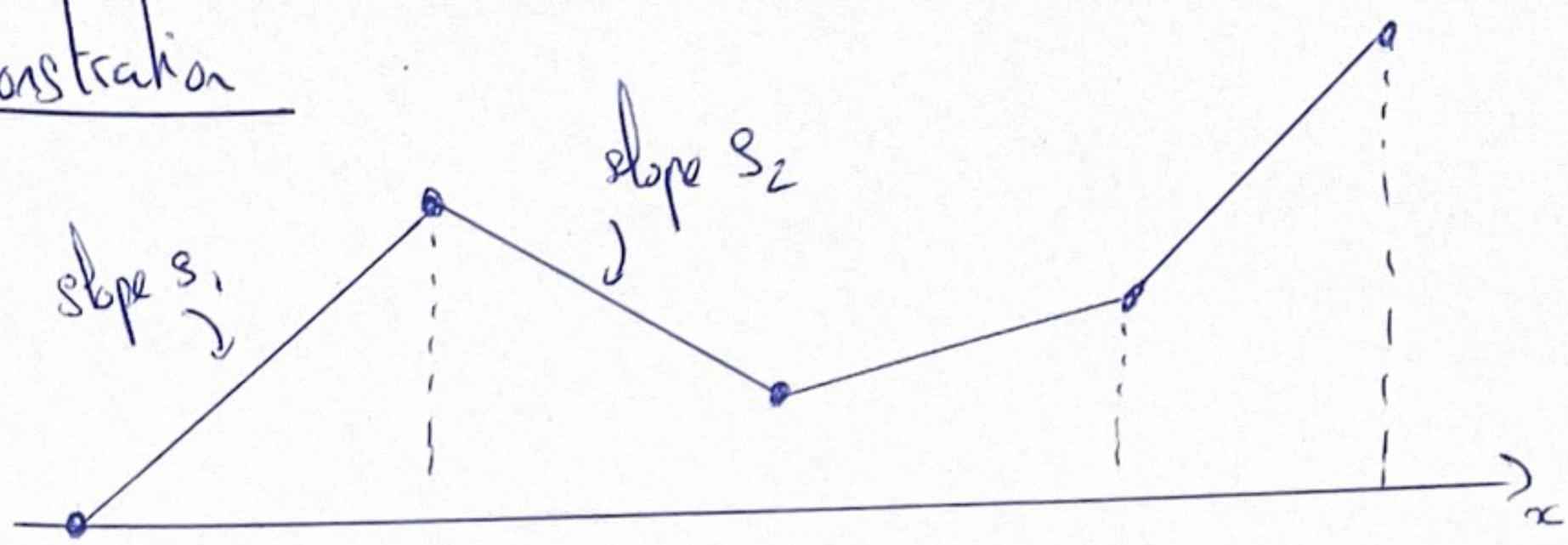
$$\leq \frac{2GDh\sqrt{2}}{\sqrt{n}}$$

IV. Approximation Universelle

Analyser l'erreur d'approximation est ~~trop~~ en dehors des objectifs de ce cours. À la place, nous allons regarder la propriété d'approximation universelle des réseaux à une couche cachée. On se place dans le cas $d = 1$

Théorème : Toute fonction affine par morceaux sur $(-R, R)$, continue, et avec au plus $m-2$ morceaux sur $(-R, R)$ peut être approximée par un réseau Relu à une couche cachée avec au plus m neurones.

Démonstration



$-M$ a_1 ... a_{m-3} M

Commençons par le cas $f(-M) = 0$. Le cas général est obtenu en rajoutant un neurone en observant que

$$(x+M)_+ + (-x+M)_+ = 2M \text{ sur } [-M, M].$$

Alors $f(x) = s_1 (x - (-M))_+$ sur $[-M, a_1]$.

$$f(x) = s_1 (x-(-M))_+ + (s_2 - s_1)(x - (a_1))_+$$

sur $[a_1, a_2]$

...

□