IFT-7002 Fondements de l'apprentissage machine

Les réseaux de neurones

Shai Shalev-Shwartz
The Hebrew University of Jerusalem

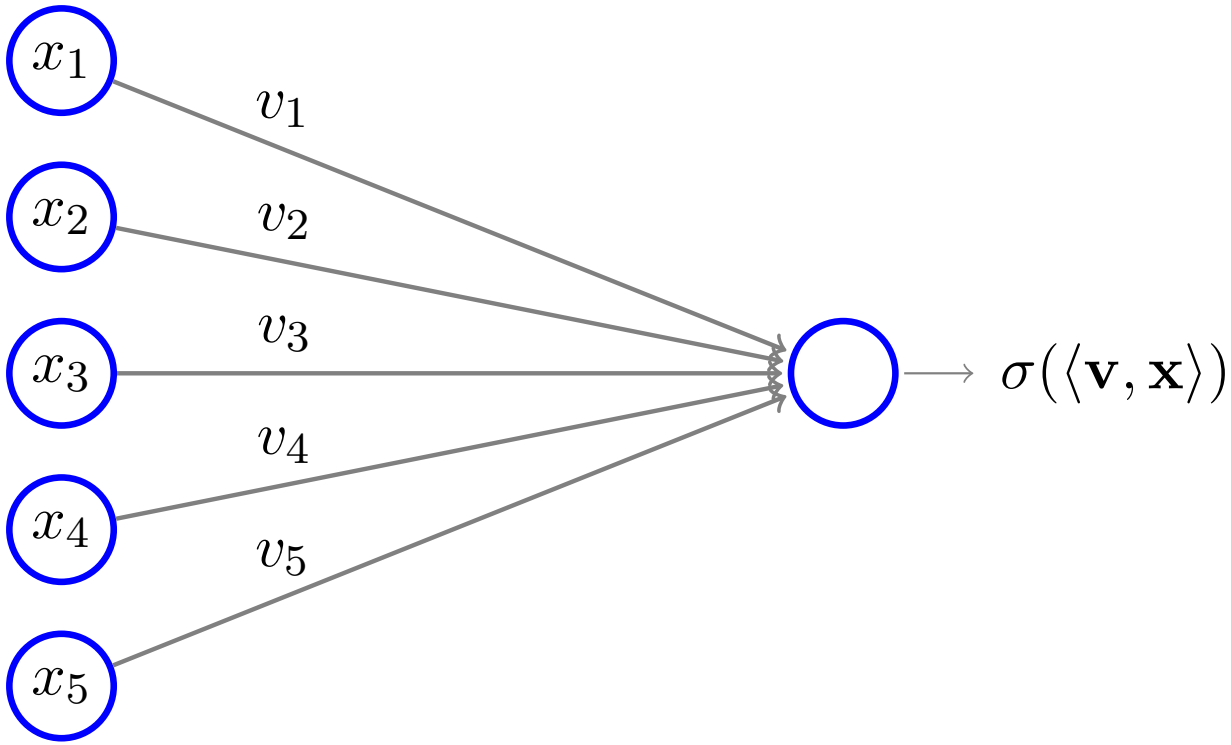
Traduit et adapté par Mario Marchand
Université Laval

Hiver 2021

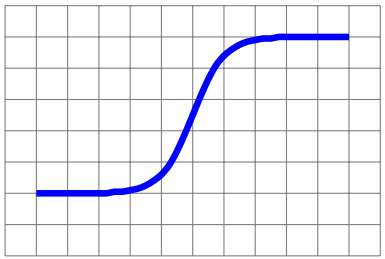
Sommaire

Le neurone artificiel

- un neurone (ou perceptron) est une fonction $\mathbf{x} \mapsto \sigma(\langle \mathbf{v}, \mathbf{x} \rangle)$, t.q. $\sigma : \mathbb{R} \to \mathbb{R}$ est une fonction d'activation.



- *e.g.*, σ est la fonction sigmoïde



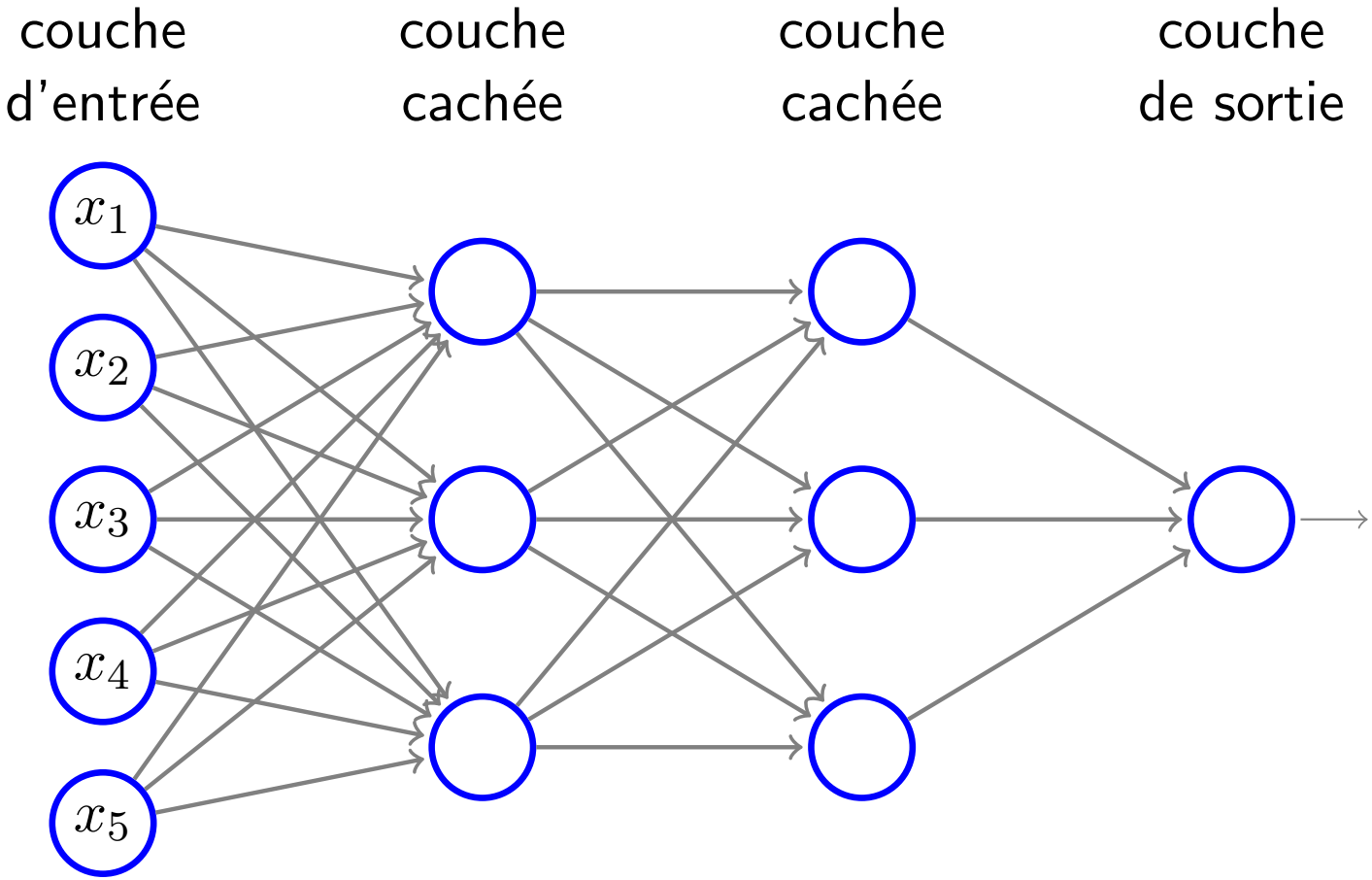
Réseaux de neurones

- En connectant plusieurs neurones ensemble nous obtenons un réseau de neurones.
- Nous nous intéressons aux réseaux "feedforward" dont le réseau forme un graphe $G = (V, E)$ orienté acyclique et pondéré.
- Noeuds d'entrée : les noeuds n'ayant pas d'arcs entrants.
- Noeuds de sortie : les noeuds n'ayant pas d'arcs sortants.
- Noeuds cachés : ceux ayant des arcs entrants et des arcs sortants.
- Fonction de poids : $w : E \to \mathbb{R}$ (chaque arc de E possède un poids).
- Pour tout noeud $v \in V$, $a[v]$ dénote l'entrée au noeud v et $o[v]$ désigne la sortie du noeud v. Nous avons :

$$\begin{aligned} a[v] &= \sum_{u:u\to v\in E} w[u \to v]o[u] \\ o[v] &= \sigma(a[v])\,. \end{aligned}$$

Réseaux de neurones multi-couches

- Organisation en couches : $V = \dot{\cup}_{t=0}^{T} V_t$.
  - $V_t =$ l'ensemble des noeuds de la couche t.
- Présence d'arcs seulement entre couches adjacentes.
- Exemple d'un réseau de profondeur 3 :



La classe d'hypothèses d'un réseau à architecture fixe

- Un réseau de neuronnes (V, E, σ, w) nous donne une seule hypothèse (prédicteur) $h_{V,E,\sigma,w} : \mathbb{R}^{|V_0|-1} \to \mathbb{R}^{|V_T|}$
- (V, E, σ) désigne une architecture. Cela donne la classe d'hypothèses :

$$\mathcal{H}_{V,E,\sigma} \stackrel{\text{def}}{=} \{h_{V,E,\sigma,w} \ : \ w \text{ est une assignation de } E \text{ vers } \mathbb{R}\} \ .$$

- L'architecture reflète notre connaissance a priori et l'apprentissage a pour tâche de trouver la fonction de poids w.
- Examinons :
  - l'erreur d'estimation (la complexité d'échantillon)
  - l'erreur d'approximation (l'expressivité de l'architecture)
  - La complexité algorithmique de la tâche d'apprentissage

Sommaire

Complexité d'échantillon des réseaux de neurones

Théorème

$\mathrm{VCdim}(\mathcal{H}_{V,E,sign}) \in O(|E| \log(|E|))$.

Preuve:

- Soit $\mathcal{H} \stackrel{\text{def}}{=} \mathcal{H}_{V,E,\text{sign}}$ et $V = V_0 \cup V_1 \cup \cdots \cup V_T$.
- En assignant différents poids entre la couche V_{t-1} et V_t nous obtenons la classe $\mathcal{H}_t$ des fonctions $\{\pm 1\}^{|V_{t-1}|} \to \{\pm 1\}^{|V_t|}$.
- On a alors $\mathcal{H} = \mathcal{H}_T \circ \cdots \circ \mathcal{H}_1$. Le nombre maximum $\tau_{\mathcal{H}}(m)$ de dichotomies que l'on peut effectuer sur un ensemble de m points à l'aide des fonctions de $\mathcal{H}$ doit donc satisfaire

$$\tau_{\mathcal{H}}(m) \leq \prod_{t=1}^{T} \tau_{\mathcal{H}_t}(m)\,.$$

Complexité d'échantillon des réseaux de neurones

- Puisque $\mathcal{H}_t = \mathcal{H}_{t,1} \times \cdots \times \mathcal{H}_{t,|V_t|}$, où chaque $\mathcal{H}_{t,i}$ est une fonction $\{\pm 1\}^{|V_{t-1}|} \to \{\pm 1\}$, on a alors que

$$\tau_{\mathcal{H}_t}(m) \;=\; \prod_{i=1}^{|V_t|} \tau_{\mathcal{H}_{t,i}}(m)\,.$$

- En combinant les deux (in)égalités, nous avons

$$\tau_{\mathcal{H}}(m) \leq \prod_{t=1}^{T} \prod_{i=1}^{|V_t|} \tau_{\mathcal{H}_{t,i}}(m)\,.$$

- Or, $\mathcal{H}_{t,i}$ coincide avec la classe des demi-espaces homogènes effectuées à l'aide de $d_{t,i}$ poids. Donc $\mathrm{VCdim}(\mathcal{H}_{t,i}) = d_{t,i}$. Alors

$$\tau_{\mathcal{H}_{t,i}}(m) \leq \left(\frac{em}{d_{t,i}}\right)^{d_{t,i}} \leq (em)^{d_{t,i}}\,.$$

Complexité d'échantillon des réseaux de neurones

- En combinant les deux inégalités, nous avons

$$\tau_{\mathcal{H}}(m) \leq (em)^{\sum_{t,i} d_{t,i}} = (em)^{|E|} .$$

- Maintenant, considérons que ces m points sont pulvérisés par $\mathcal{H}$.
- On a alors $\tau_{\mathcal{H}}(m) = 2^m$. Donc

$$2^m \leq (em)^{|E|} \Rightarrow m \leq \frac{|E|}{\log(2)} \log(em) .$$

- Donc $m' \leq a \log(m')$, avec $m' = em$ et $a = \frac{e|E|}{\log(2)}$.
- Selon le Lemme A.1 : $m' \leq a \log(m') \Rightarrow m' \leq 2a \log(a)$. Donc, $m = \mathrm{VCdim}(\mathcal{H})$ doit satisfaire

$$em \leq \frac{2e|E|}{\log(2)} \log(\frac{e|E|}{\log(2)}) \Rightarrow m \in O(|E| \log |E|) .$$

Complexité d'échantillon des réseaux de neurones

Remarques :

- En fait, la $\mathrm{VCdim}(\mathcal{H}_{V,E,\sigma})$ dépend de la fonction d'activation utilisée (c'est $\sigma = \mathrm{sign}$ pour le résultat précédent). Voir exercice 5, chap 20.
- Par contre si on utilise $b \in O(1)$ bits pour chaque poids, $\mathrm{VCdim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|) \leq \log_2\left(2^{b|E|}\right) = b|E|$ peu importe la fonction d'activation utilisée.
- Donc, selon le théorème fondamental, la complexité d'échantillon satisfait

$$m_{\mathcal{H}_{V,E,\sigma}}(\epsilon, \delta) \in O((|E| + \log(1/\delta))/\epsilon^2)\,.$$

- Cependant, en pratique, il arrive que les réseaux de neurones généralisent bien même lorsqu'entraînés avec $m \ll |E|$ exemples.
- Certains résultats théoriques (*e.g.*, Peter Bartlett, 1997) démontrent également que la complexité d'échantillon est plus affectée par la magnitude des poids que par leur nombre $|E|$ (lorsque $\sigma =$ sigmoïde).
- D'où l'utilisation d'heuristiques pénalisant la présence de poids de grandes magnitudes comme le "weight decay" et le "early stopping".

Sommaire

Expressivité des réseaux de neurones

- Par simplicité, examinons le cas des entrées Booléennes et de la fonction d'activation sign.
- Quels types de fonctions de $\{\pm 1\}^n$ vers $\{\pm 1\}$ est-il possible de réaliser avec $\mathcal{H}_{V,E,\mathrm{sign}}$?

Théorème

Pour tout n, il existe $G(V,E)$ de profondeur 2, tel que $\mathcal{H}_{V,E,sign}$ contient toutes les fonctions de $\{\pm 1\}^n$ vers $\{\pm 1\}$.

Preuve:

- Soit $f : \{\pm 1\}^n \to \{\pm 1\}$.
- Soit $\mathbf{u}_1, \ldots, \mathbf{u}_k$ tous les vecteurs de $\{\pm 1\}^n$ tels que $f(\mathbf{u}_i) = +1$.
- On a que $g_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathrm{sign}(\langle \mathbf{u}_i, \mathbf{x} \rangle - n + 1) = +1$ ssi $\mathbf{x} = \mathbf{u}_i$.

Expressivité des réseaux de neurones

- Utilisons alors chaque noeud $v_{1,i} \in V_1$ comme neurone pour implémenter $g_i(\mathbf{x})$.
- Or, $f(\mathbf{x})$ est la disjonction des $g_i(\mathbf{x})$ et peut donc s'écrire comme

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{k} g_i(\mathbf{x}) + k - 1\right),$$

- et peut donc peut être réalisée par le neurone de sortie $v_{2,1}$. ■

- Avec cette méthodologie, le nombre k de neurones dans V_1 doit être 2^n si l'on désire pouvoir implémenter toutes les fonctions Booléennes avec des réseaux de profondeur 2.
- Le prochain théorème démontre que toute architecture doit contenir un nombre exponentiel en n de neurones pour qu'elle puisse réaliser toutes les fonctions Booléennes sur n entrées.

Expressivité des réseaux de neurones

Théorème

Soit $G(V, E)$ t.q. $\mathcal{H}_{V,E,sign}$ contient toutes les fonctions de $\{\pm 1\}^n$ vers $\{\pm 1\}$. Alors, $|V|$ est exponentiel en n.

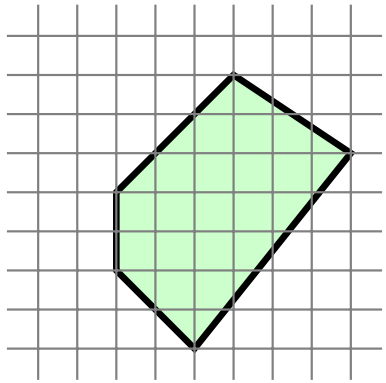
Preuve:

- Si pour $G(V, E)$, $\mathcal{H}_{V,E,\text{sign}}$ contient toutes les fonctions de $\{\pm 1\}^n$ vers $\{\pm 1\}$, on a que $\{\pm 1\}^n$ est pulvérisé par $\mathcal{H}_{V,E,\text{sign}}$,
- Donc $\text{VCdim}(\mathcal{H}_{V,E,\text{sign}}) \geq |\{\pm 1\}^n| = 2^n$.
- Or, nous savons que $\text{VCdim}(\mathcal{H}_{V,E,\text{sign}}) \in O(|E| \log |E|) \subseteq O(|V|^3)$.
- Donc $2^n \leq \text{cst}|V|^3$. Alors $|V| \in \Omega(2^{n/3})$. ■

Remarque : Un théorème similaire existe pour les fonctions à valeurs $\in \mathbb{R}$. En particulier, pour tout $\epsilon > 0$ et pour toute fonction 1-Lipschitzienne $f : [-1, +1]^n \to [-1, +1]$, il existe $h \in \mathcal{H}_{V,E,\tanh}$ t.q. $|h(\mathbf{x}) - f(\mathbf{x})| \leq \epsilon$ pour tout $\mathbf{x} \in [-1, +1]^n$ en autant que $|V|$ soit exponentiel en n.

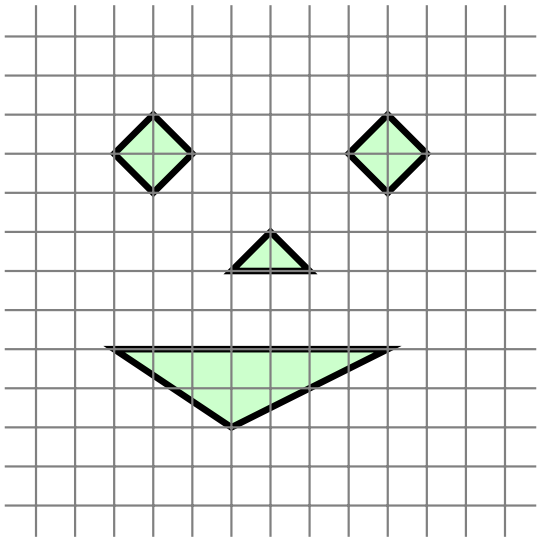
Expressivité des réseaux : intuition géométrique

- Chaque neurone exprime un demi-espace.
- Un réseau à deux couches peut donc exprimer une intersection de demi-espaces.



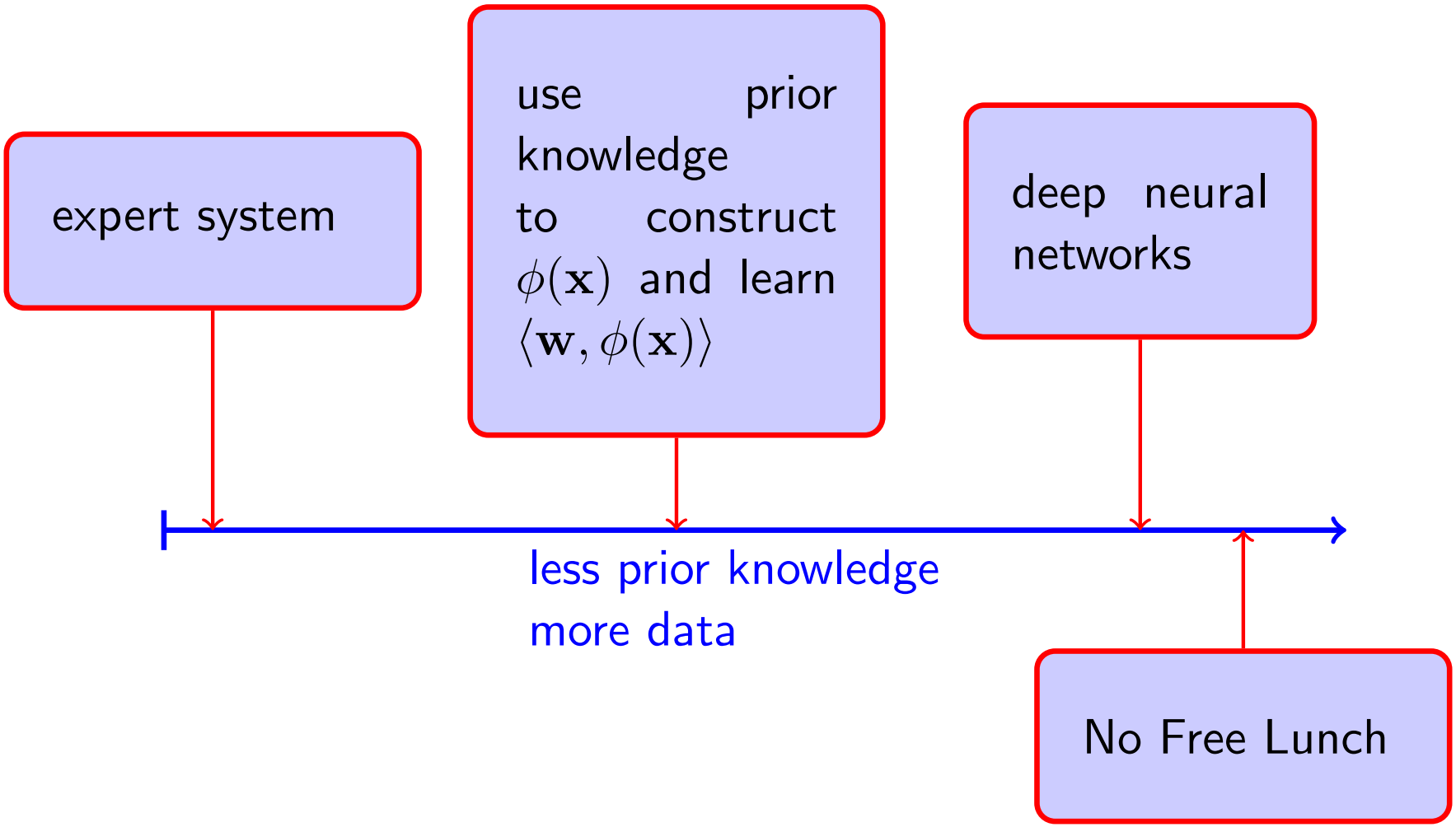
Expressivité des réseaux : intuition géométrique

- Un réseau à trois couches peut donc exprimer l'union de l'intersection de demi-espaces.



- L'expressivité des réseaux augmente rapidement avec le nombre de couches !

Compromis données vs. connaissance a priori



Sommaire

Complexité d'apprendre les réseaux de neurones.

Considérez la minimisation du risque empirique :

$$\mathrm{ERM}_{\mathcal{H}_{V,E,\sigma}}(S) = \operatorname*{argmin}_{h \in \mathcal{H}_{V,E,\sigma}} L_S(h) = \operatorname*{argmin}_{w} L_S(h_{V,E,\sigma,w})$$

- Théorème : Ce problème est $\mathcal{NP}$-difficile pour $\sigma = \mathrm{sign}$ même s'il s'agit d'un réseau à 2 couches avec seulement 4 unités cachés.
  - Ici on suppose que les étiquettes de S sont générés par un $f \in \mathcal{H}_{V,E,\sigma}$.
- Est-il alors possible de trouver (en temps raisonnable) une solution proche de l'optimale ?
- Non ! selon Bartlett et Ben-David (2002).
- De plus, certains résultats tendent à démontrer qu'il est $\mathcal{NP}$-difficile d'apprendre un réseau à 2 couches contenant $\omega(\log(n))$ neurones à l'aide d'un réseau de plus grande taille.
  - Ici, n est le nombre d'unités d'entrées.

Algorithme pour apprendre avec les réseaux de neurones ?

- La classe des réseaux de neurones est très riche, mais c'est impossible d'y effectuer la minimisation du risque empirique.
- La technique principalement utilisée est la descente de gradient stochastique (DGS).
- Non convexe, aucune guarantie, peut nécessiter un temps excessif.
- Mais, en pratique, la DGS parvient souvent à trouver une bonne solution.

DGS pour réseaux de neurones

Pseudo-code ($\mathbf{w}$ = l'ensemble des poids et $|\mathbf{w}|$ = le nombre) :

- Initialisation : chaque $w_i \sim U[-c, c]$ avec $c = \sqrt{3/|\mathbf{w}|}$ et $\boldsymbol{\theta}_1 = \mathbf{0}$.
- Mise à jour selon le momentum de Nesterov (Sutskever et al., 2013) :

$$\boldsymbol{\theta}_{t+1} = \mu_t \boldsymbol{\theta}_t - \eta_t \nabla L_B(\mathbf{w}_t + \mu_t \boldsymbol{\theta}_t)$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \boldsymbol{\theta}_{t+1}$$

  - μ_t est le paramètre de momentum (e.g. $\mu_t = 0.9$ pour tout t)
  - η_t est le taux d'apprentissage (e.g. $\eta_t = 0.01$ pour tout t)
  - ∇L_B est un estimé du gradient $L_{\mathcal{D}}$ effectué à l'aide d'un petit ensemble d'exemples (appelé un "minibatch").
- On retourne le $\mathbf{w}_t$ ayant le mieux performé sur un ensemble de validation.
- Montrons comment calculer ∇L_B par rétro-propagation du gradient.

Rétro-propagation du gradient de l'erreur

- On considère un réseau feedforward avec T couches de neurones effectuant un traitement.
  - Les neurones de la couches 0 n'effectuent pas de traitement et se contentent de reproduire l'entrée (instance) $\mathbf{x}$.
- On a donc $V = V_0 \cup V_1 \cup \cdots \cup V_T$ avec $k_t \stackrel{\text{def}}{=} |V_t|$ pour $t \in \{0, \ldots, T\}$.
- W_{t-1} désigne la matrice $k_t \times k_{t-1}$ des poids entre les neurones de la couche V_{t-1} et ceux de la couche V_t. Donc, $W_{t-1,i,j}$ désigne le poids de l'arc allant du neurone j de V_{t-1} au neurone i de V_t.
- L'entrée totale $a_{t,i}$ au neurone i de V_t est donnée par la somme pondérée des sorties $o_{t-1,j}$ des neurones j de V_{t-1}, *i.e.*,

$$a_{t,i} = \sum_{j=1}^{k_{t-1}} W_{t-1,i,j} o_{t-1,j} \,.$$

- Utilisons une fonction d'activation σ différentiable telle que la sigmoïde $\sigma(x) = 1/(1 + e^{-x})$. Alors $o_{t,i} = \sigma(a_{t,i})$.

Rétro-propagation du gradient de l'erreur

- Pour tout exemple $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, on a que $\mathbf{x} \in \mathbb{R}^n$ et $\mathbf{y} \in \mathbb{R}^r$.
- Si $\mathbf{w}$ désigne l'ensemble des poids du réseau de neurones et que $\mathbf{h_w}(\mathbf{x})$ désigne la sortie du réseau sur l'instance $\mathbf{x}$ et que ℓ désigne notre fonction de perte, l'erreur $L_B(\mathbf{w})$ de $\mathbf{h_w}$ sur un ensemble "minibatch" B d'exemples est donnée par

$$L_B(\mathbf{w}) = \frac{1}{|B|} \sum_{(\mathbf{x},\mathbf{y}) \in B} \ell(\mathbf{h_w}(\mathbf{x}), \mathbf{y}) \,.$$

- Le gradient $\nabla L_B(\mathbf{h_w})$ de cette erreur est donc obtenu en moyennant le gradient de la perte $\ell(\mathbf{h_w}(\mathbf{x}), \mathbf{y})$ sur chaque exemple de B

$$\nabla L_B(\mathbf{h_w}) = \frac{1}{|B|} \sum_{(\mathbf{x},\mathbf{y}) \in B} \nabla \ell(\mathbf{h_w}(\mathbf{x}), \mathbf{y}) \,.$$

- $\nabla \ell(\mathbf{h_w}(\mathbf{x}), \mathbf{y})$ est le vecteur constitué de toutes les dérivées partielles

$$\frac{\partial \ell(\mathbf{h_w}(\mathbf{x}), \mathbf{y})}{\partial W_{t-1,i,j}} \stackrel{\text{def}}{=} G_{t-1,i,j} \,.$$

Rétro-propagation du gradient de l'erreur

- Puisque la variation de $W_{t-1,i,j}$ affecte la sortie $o_{t,i}$ et que, selon la règle d'enchaînement des dérivées, nous avons

$$\frac{\partial o_{t,i}}{\partial W_{t-1,i,j}} = \sigma'(a_{t,i}) \frac{\partial a_{t,i}}{\partial W_{t-1,i,j}} = \sigma'(a_{t,i}) o_{t-1,j} \,,$$

en définissant $\delta_{t,i} \stackrel{\text{def}}{=} \frac{\partial \ell(\mathbf{h_w}(\mathbf{x}),\mathbf{y})}{\partial o_{t,i}}$, nous obtenons

$$\begin{aligned} \frac{\partial \ell(\mathbf{h_w}(\mathbf{x}),\mathbf{y})}{\partial W_{t-1,i,j}} &= \frac{\partial \ell(\mathbf{h_w}(\mathbf{x}),\mathbf{y})}{\partial o_{t,i}} \frac{\partial o_{t,i}}{\partial W_{t-1,i,j}} \\ &= \delta_{t,i} \sigma'(a_{t,i}) o_{t-1,j} \,. \end{aligned}$$

- Pour obtenir la valeur de $\frac{\partial \ell(\mathbf{h_w}(\mathbf{x}),\mathbf{y})}{\partial W_{t-1,i,j}}$, il faut donc connaître $\delta_{t,i}$.

Rétro-propagation du gradient de l'erreur

- Mais $\boldsymbol{\delta}_t \stackrel{\text{def}}{=} (\delta_{t,1}, \ldots, \delta_{t,k_t})$ s'obtient de $\boldsymbol{\delta}_{t+1}$ à l'aide de la règle d'enchaînement des dérivées

$$\delta_{t,i} = \frac{\partial \ell(\mathbf{h}_{\mathbf{w}}(\mathbf{x}), \mathbf{y})}{\partial o_{t,i}} = \sum_{j=1}^{k_{t+1}} \frac{\partial \ell(\mathbf{h}_{\mathbf{w}}(\mathbf{x}), \mathbf{y})}{\partial o_{t+1,j}} \frac{\partial o_{t+1,j}}{\partial o_{t,i}}$$

$$= \sum_{j=1}^{k_{t+1}} \delta_{t+1,j} \frac{\partial o_{t+1,j}}{\partial a_{t+1,j}} \frac{\partial a_{t+1,j}}{\partial o_{t,i}} = \sum_{j=1}^{k_{t+1}} \delta_{t+1,j} \sigma'(a_{t+1,j}) W_{t,j,i}$$

- Après avoir présenté l'instance $\mathbf{x}$ au réseau, on calcul tous les vecteurs $\mathbf{a}_t \stackrel{\text{def}}{=} (a_{t,1}, \ldots, a_{t,k_t})$ et $\mathbf{o}_t \stackrel{\text{def}}{=} (o_{t,1}, \ldots, o_{t,k_t})$ pour $t = 1, \ldots, T$. Ensuite, on calcul $\boldsymbol{\delta}_T$ à l'aide de

$$\delta_{T,i} = \frac{\partial \ell(\mathbf{h}_{\mathbf{w}}(\mathbf{x}), \mathbf{y})}{\partial o_{T,i}} = \frac{\partial \ell(\mathbf{o}_T, \mathbf{y})}{\partial o_{T,i}} .$$

  - Notez que $\boldsymbol{\delta}_T = \mathbf{o}_T - \mathbf{y}$ lorsque $\ell(\mathbf{o}_T, \mathbf{y}) = (1/2)||\mathbf{o}_T - \mathbf{y}||^2$.

Rétro-propagation du gradient de l'erreur

- Ayant $\boldsymbol{\delta}_T$, on obtient alors le gradient de $\ell(\mathbf{h}_\mathbf{w}(\mathbf{x}), \mathbf{y})$ par rapport aux poids de la dernière couche à partir de

$$\frac{\partial \ell(\mathbf{h}_\mathbf{w}(\mathbf{x}), \mathbf{y})}{\partial W_{T-1,i,j}} = \delta_{T,i} \sigma'(a_{T,i}) o_{T-1,j} \, .$$

- et $\boldsymbol{\delta}_{T-1}$ à l'aide de

$$\delta_{T-1,i} = \sum_{j=1}^{k_T} \delta_{T,j} \sigma'(a_{T,j}) W_{T-1,j,i}$$

- Et ainsi de suite jusqu'à ce que l'on obtienne le gradient de $\ell(\mathbf{h}_\mathbf{w}(\mathbf{x}), \mathbf{y})$ par rapport aux poids de la première couche.
- Le gradient $\nabla \ell(\mathbf{h}_\mathbf{w}(\mathbf{x}), \mathbf{y})$ s'obtient donc de la sortie vers l'entrée, *i.e.*, par rétro-propagation du gradient de l'erreur.

Algorithme de la rétro-propagation du gradient de l'erreur

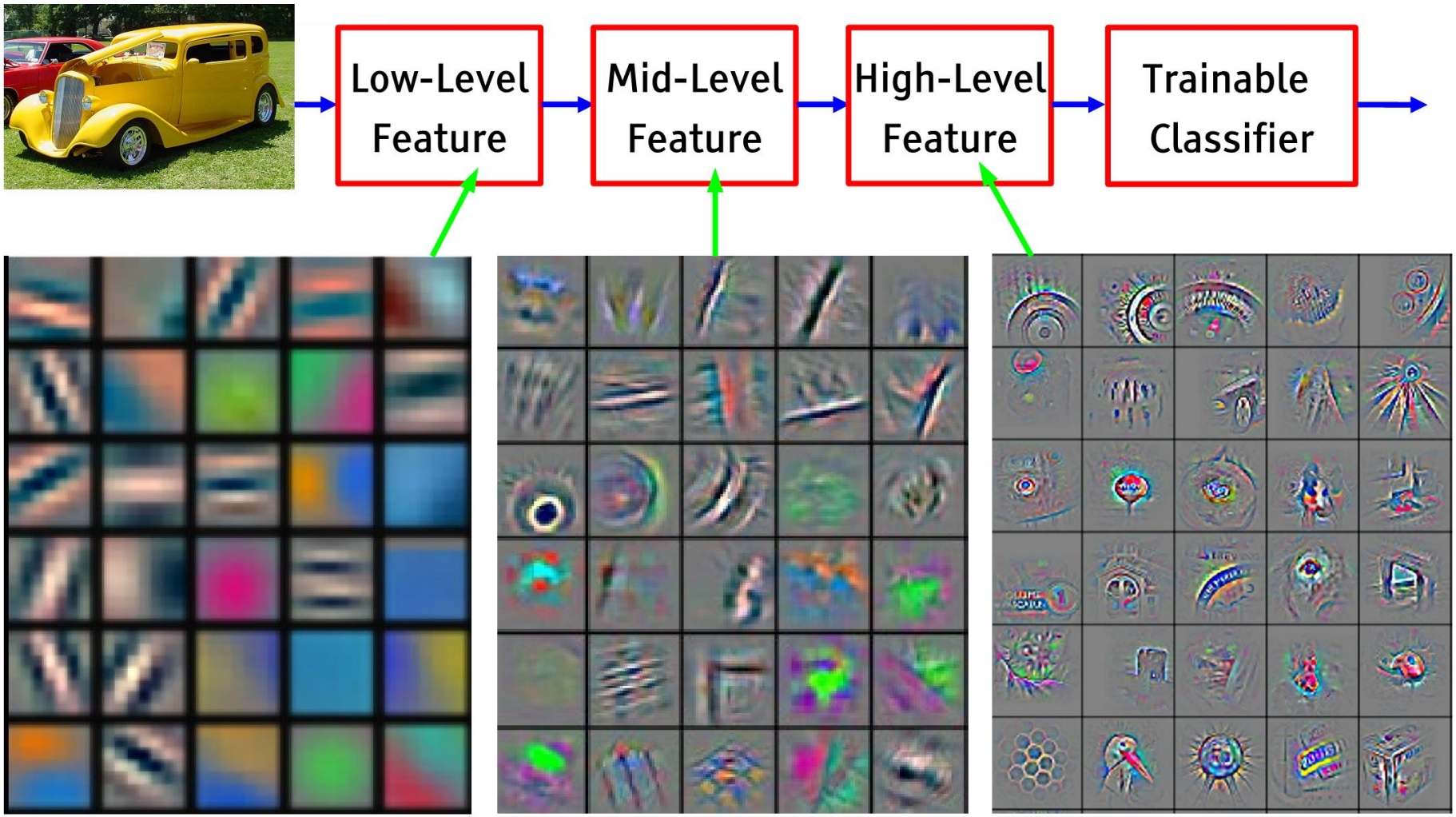
- Initialisation du gradient : $G_{t-1,i,j} = 0 \, \forall (i,j) \in V_t \times V_{t-1} \, \forall t \in [T]$.
- Pour tous les exemples $(\mathbf{x}, \mathbf{y})$ de B faire :
  - (Propagation de $\mathbf{x}$ de l'entrée vers la sortie)
  - $\mathbf{o}_0 = \mathbf{x}$
  - Pour $t = 1, \ldots, T$ et pour $i = 1, \ldots, k_t$:
    - $a_{t,i} = \sum_{j=1}^{k_{t-1}} W_{t-1,i,j} o_{t-1,j}$
    - $o_{t,i} = \sigma(a_{t,i})$
  - (Propagation de $\boldsymbol{\delta}_T$ et du gradient de l'erreur de la sortie vers l'entrée)
  - $\boldsymbol{\delta}_T = \nabla \ell(\mathbf{o}_T, \mathbf{y})$
  - Pour $t = T-1, \ldots, 1$ et pour $i = 1, \ldots, k_t$:
    - $\delta_{t,i} = \sum_{j=1}^{k_{t+1}} W_{t,j,i} \delta_{t+1,j} \sigma'(a_{t+1,j})$
    - $g_{t-1,i,j} = \delta_{t,i} \sigma'(a_{t,i}) o_{t-1,j} \, \forall j \in V_{t-1}$
  - Cumuler : $G_{t-1,i,j} = G_{t-1,i,j} + g_{t-1,i,j} \, \forall (i,j) \in V_t \times V_{t-1} \, \forall t \in [T]$.
- Sortie : $G_{t-1,i,j} = G_{t-1,i,j}/|B| \, \forall (i,j) \in V_t \times V_{t-1} \, \forall t \in [T]$.

Sommaire

Caractéristiques émergentes des réseaux de neurones

- L'approche "d'ingénierie des caractéristiques" : un expert propose une fonction de projection $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^N$. Il utilise ensuite un algorithme d'apprentissage (*e.g.*, régularisation de Tikhonov) pour construire un prédicteur linéaire sur $\boldsymbol{\phi}(\mathbf{x})$.
- L'approche "Deep learning" : Les neurones des couches cachées sont en fait des caractéristiques apprises automatiquement durant l'apprentissage du réseau à partir des données.
- Les neurones près de l'entrée du réseau constituent des caractéristiques de bas niveau (*e.g.*, filtres) et les neurones près de la sortie constituent des caractéristiques de haut niveau qui sont nécessaires à l'accomplissement de tâches difficiles.

Caractéristiques émergentes des réseaux de neurones



Taken from Yan LeCun's deep learning tutorial

Apprentissage de tâches multiples et apprentissage de représentations

- L'idée : les neurones des premières couches et des couches intermédiaires sont utilisés pour différentes tâches similaires.
- Seule la dernière couche est spécifique à la tâche.
- "transfer learning" : après avoir appris une certaine tâche, il arrive parfois que les neurones intermédiaires sont utilisés comme caractéristiques pour apprendre une nouvelle tâche similaire à la première. La dernière couche intermédiaire constitue une représentation des instances.
  - ex : les couches intermédiaires d'un réseau qui traduit du français vers l'anglais pourraient être utilisées pour la traduction du français à l'allemand.

Quelques découvertes du "Deep"

- Fonction d'activation ReLU : $\sigma(a) = \max\{0, a\}$. Élimine le problème du gradient presque nul sans sacrifié l'expressivité.
- Les réseaux à convolution : le "weight sharing" des couches cachées se trouvent à faire la convolution du signal d'entrée, ce qui, combiné au max-pooling, procure une quasi-invariance par translation : très utilisé en vision.
- Utilisation de réseaux gigantesques : Il arrive souvent que le nombre de poids ajustables excède de beaucoup le nombre d'exemples.
  - L'overfitting est évité en empêchant les poids de grandes magnitudes d'apparaître (régularisation et "early stopping").
- Le "Dropout" : une autre forme de régularisation qui met la sortie à zéro de certains neurones durant l'apprentissage.
- Trucs pour la DGS : le momentum de Nesterov et autres formes d'approximation du second degré.
- Utilisation de GPUs.

L'historique

- Années 1940–79 :
  - Apprentissage simple inspiré du cerveau (Pitts, Hebb, et autres).
  - Le perceptron (Rosenblatt).
  - La "backprop" (Werbos 1975).
- Années 1980–1994 :
  - Popularisation de la "backprop" (Rumelhart, Hinton et Williams 1986, LeCun 1985) et de la DGS (Bottou). Succès empirique initial.
- Années 1995–2005 :
  - Perte d'intérêt en faveur des méthodes convexes (SVM, Boosting...).
- 2006 – :
  - Regain de popularité grâce à l'utilisation (avant l'apprentissage) de méthodes non supervisées comme les RBM et les auto-encodueurs réducteurs de bruits (Hinton, Bengio, LeCun, Ng, ...)
  - Plusieurs trucs permettant l'apprentissage de réseaux gigantesques à l'aide de données massives.
  - 2012 : Krizhevsky, Sustkever, Hinton : ammélioration significative de l'état de l'art sur les données d'imagenet (reconnaissance de 1000 classes d'objets), sans utiliser le pré-traitement non-supervisé.

L'historique

- 2019 : Bengio, LeCun, et Hinton reçoivent le prix Turing pour leurs travaux sur les réseaux de neurones.

Résumé

- Les réseaux de neurones procurent un niveau d'expressivité très élevé.
- C'est $\mathcal{NP}$-difficile de faire leur apprentissage.
- Empiriquement, ça fonctionne souvent très bien.
- Nous donne l'état de l'art pour plusieurs tâches pratiques.
- Question : quand est-ce que ça fonctionne et pourquoi ?