L'essentiel sur l'apprentissage - focus sur le Deep learning

Adrien CHAN-HON-TONG
ONERA
unité image vision apprentissage

Plan

- Exemples d'application du Deep learning
- Formalisation de la classification
    - Séparateur linéaire
    - Réseau de neurones
    - Deep learning
- Quelques points de théorie
- Un peu de pratique

Classification d'images

chat

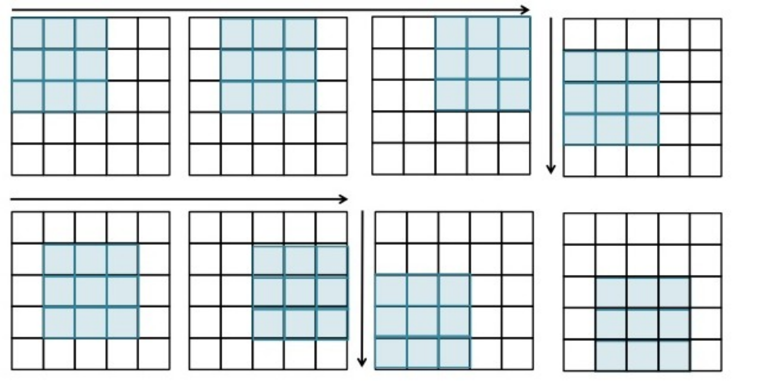
non chat

non chat

Explosion des performances de classification sur ImageNet (2012)

Exemples d'application du Deep learning

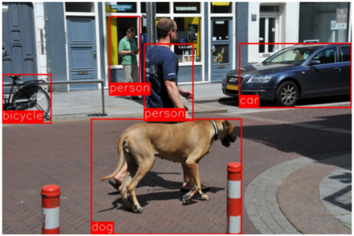
Fenêtre glissante

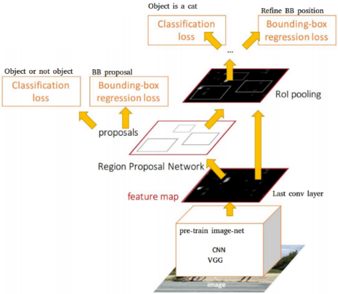


Si on sait classer, on sait faire beaucoup d'autres choses

Exemples d'application du Deep learning

Détection

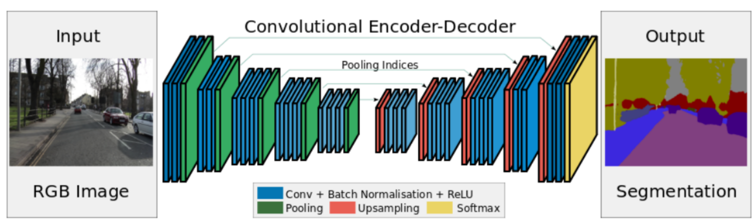




Explosion des performances de classification sur PascalVoc (2014)
Seulement 2 ans pour un énorme saut en terme de puissance disponible.

Exemples d'application du Deep learning

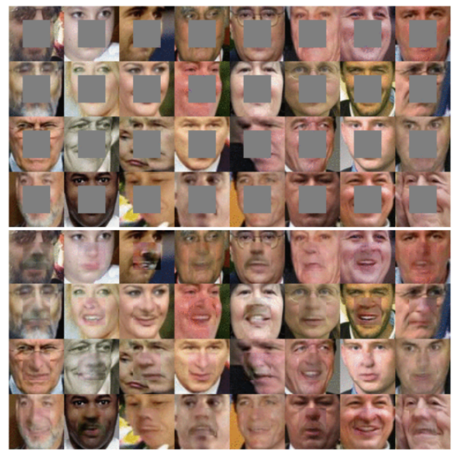
Segmentation sémantique



Explosion des performances en segmentation sémantique (2016)
Seulement 4 ans pour un énorme saut en terme de mémoire disponible.

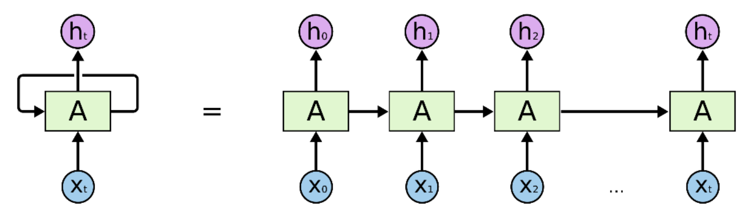
GAN

Mais aussi : de la synthèse d'images



LSTM

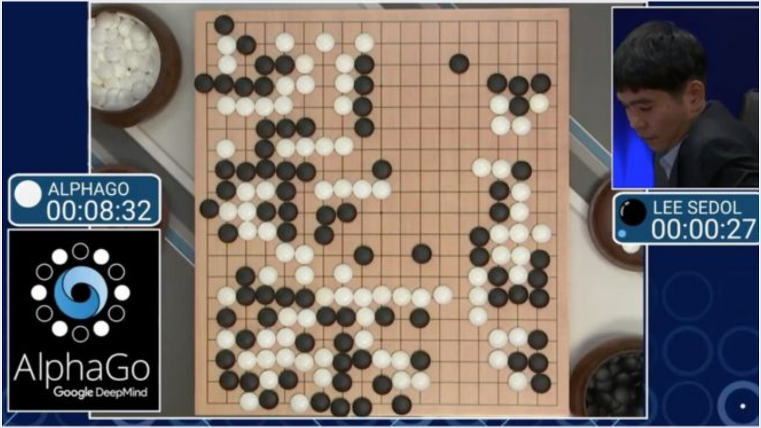
Mais aussi : du traitement de séries temporelles



Alpha Go

Mais aussi : du Deep Reinforcement Learning

Première victoire d'une IA sur le jeu de GO



Alpha Star

Première victoire d'une IA sur un jeu complex

Plan

- Exemples d'application du Deep learning
- Formalisation de la classification
  - Séparateur linéaire
  - Réseau de neurones
  - Deep learning
- Quelques points de théorie
- Un peu de pratique

Formalisation de la classification

On voudrait un système qui prend une observation $x \in \mathbb{R}^D$
En déduit la classe de cette observation $y \in \{-1, 1\}$

Exemple : à partir d'une image $x \in \mathbb{R}^D$, je voudrais savoir si c'est une image de chat $y \in \{-1, 1\}$.

chat

non chat

non chat

Formalisation de la classification

On voudrait un système qui prend une observation $x \in \mathbb{R}^D$
En déduit la classe de cette observation $y \in \{-1, 1\}$

Exemple : à partir d'une image $x \in \mathbb{R}^D$, je voudrais savoir si c'est une image de chat $y \in \{-1, 1\}$.

chat

non chat

non chat

Mais, on ne sait pas définir le lien entre x et y.

Formalisation de la classification

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$

on cherche $\Phi \in \mathbb{R}^{\mathbb{R}^D}$ tel que $\forall i \in \{1, ..., n\}, sign\,(\Phi(x_i)) \approx y_i$

Formalisation de la classification

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$

on cherche $\Phi \in \mathbb{R}^{\mathbb{R}^D}$ tel que $\forall i \in \{1, ..., n\}, sign\left(\Phi(x_i)\right) \approx y_i$

Si $\Phi(x_i) \approx y_i$, est ce qu'on a trouvé le système qu'on voulait ?

Formalisation de la classification

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$

on cherche $\Phi \in \mathbb{R}^{\mathbb{R}^D}$ tel que $\forall i \in \{1, ..., n\}, sign(\Phi(x_i)) \approx y_i$

Si $\Phi(x_i) \approx y_i$, est ce qu'on a trouvé le système qu'on voulait ?
$\rightarrow$ **NON**

$$\Phi(x) = \begin{cases} \text{si } \exists i / x = x_i & \text{retourner} \quad y_i \\ \text{sinon} & \text{retourner} \quad -1 \end{cases}$$

Formalisation de la classification

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$

base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on cherche $\Phi \in \mathbb{R}^{\mathbb{R}^D}$ tel que $\forall i \in \{1, ..., n\}, sign\left(\Phi(x_i)\right) \approx y_i$

dans l'espoir que $\forall i \in \{1, ..., n\}, sign\left(\Phi(x'_i)\right) \approx y'_i$

problème mathématiquement mal posé

Formalisation de la classification

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on se donne une famille de fonction $x \rightarrow \Phi(x, \theta)$, θ est un ensemble de poids, paramètres, coefficients

on optimise θ avec l'objectif que $\forall i \in \{1, ..., n\}, sign\left(\Phi(x_i, \theta)\right) \approx y_i$

dans l'espoir que $\forall i \in \{1, ..., n\}, sign\left(\Phi(x'_i, \theta)\right) \approx y'_i$

Formalisation de la classification

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

sélection : on se donne une famille de fonction $x \to \Phi(x, \theta)$

optimisation : on optimise θ avec l'objectif que $sign\left(\Phi(x_i, \theta)\right) \approx y_i$

évaluation : dans l'espoir que $\forall i \in \{1, ..., n\}, sign\left(\Phi(x'_i, \theta)\right) \approx y'_i$

classification supervisée

étant donnée un problème réel, on cherche une famille Φ tel qu'après optimisation, on ait une bonne évaluation

Mais (*no free lunch*)

La performance sur l'ensemble des problèmes est constante

Séparateur linéaire

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on optimise w, b avec l'objectif que $sign(w^T x_i + b) \approx y_i$

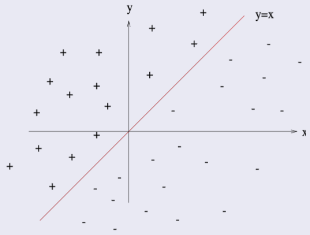
dans l'espoir que $\forall i \in \{1, ..., n\}, sign(w^T x'_i + b) \approx y'_i$

Séparateur linéaire

base d'apprentissage $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), \ldots, (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on optimise w, b avec l'objectif que $sign(w^T x_i + b) \approx y_i$

dans l'espoir que $\forall i \in \{1, \ldots, n\}, sign(w^T x'_i + b) \approx y'_i$



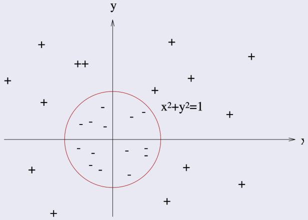
Avantage : optimisation bien étudiée, simplicité (théorie de la généralisation)
Inconvénient : modèle trop simpliste pour capturer des distributions complexes

Séparateur linéaire

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on optimise w, b avec l'objectif que $sign(w^T x_i + b) \approx y_i$

dans l'espoir que $\forall i \in \{1, ..., n\}, sign(w^T x'_i + b) \approx y'_i$



Avantage : optimisation bien étudiée, simplicité (théorie de la généralisation)
Inconvénient : modèle trop simpliste pour capturer des distributions complexes

Réseau de neurones et Deep learning ?

La même chose avec une famille de fonction plus compliquée.

Réseau de neurones et Deep learning ?

1 filtre linéaire

$x \in \mathbb{R}^D \to x|w \in \mathbb{R}$

Réseau de neurones et Deep learning ?

1 filtre linéaire

$x \in \mathbb{R}^D \to x|w \in \mathbb{R}$

K filtres linéaires ?

$x \in \mathbb{R}^D \to (x|w_k)_{k\in\{1,\dots,K\}} \in \mathbb{R}^K$

comment les recombiner pour produire une décision ? ?

Réseau de neurones et Deep learning ?

1 filtre linéaire

$x \in \mathbb{R}^D \to x|w \in \mathbb{R}$

K filtres linéaires + 1 filtre linéaire !

$x \in \mathbb{R}^D \to (x|w_k)_{k \in \{1,\dots,K\}} \in \mathbb{R}^K \to (x|w_k)\,|w' \in \mathbb{R}$

1 filtre linéaire

$x \in \mathbb{R}^D \to x|w \in \mathbb{R}$

K filtres linéaires + 1 filtre linéaire = 1 filtre linéaire :-(

$$x \in \mathbb{R}^D \to (x|w_k)_{k\in\{1,\ldots,K\}} \in \mathbb{R}^K \to (x|w_k)\,|w' = x|\sum_k w'_k w_k \in \mathbb{R}$$

Réseau de neurones

avec un petit peu de non linéarité ?

$relu\,(x) = \max(x, 0)$
sigmoide $sigm\,(x) = \frac{1}{1+\exp(-x)}$
tangente hyperbolique $tanh\,(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$

Réseau de neurones

1 filtre

$x \in \mathbb{R}^D \rightarrow x|w \in \mathbb{R}$

K filtres + relu + 1 filtre = réseau de neurones 1950

$x \in \mathbb{R}^D \rightarrow (relu\,(x|w_k))_{k \in \{1,\ldots,K\}} \in \mathbb{R}^K \rightarrow (relu\,(x|w_k))\,|w' \in \mathbb{R}$

Réseau de neurones

1 filtre linéaire

$x \in \mathbb{R}^D \to x|w \in \mathbb{R}$

K filtres + relu + 1 filtre = réseau de neurones 1950

$x \in \mathbb{R}^D \to (relu\,(x|w_k))_{k\in\{1,\dots,K\}} \in \mathbb{R}^K \to (relu\,(x|w_k))\,|w' \in \mathbb{R}$

avec K grand on peut approximer n'importe quelle fonction
Mais ce n'est pas le but : ce qui compte c'est la performance en test

Or expérimentalement, les réseaux sont meilleurs pour l'apprentissage quand ils sont profonds et pas épais

Deep learning

- 1 filtre
- K filtres + relu + 1 filtre = réseau de neurones 1950
- k_1 filtres + relu + k_2 filtre + relu + ... + k_R filtre + relu + 1 filtre = deep learning 2012

Maintenant vous savez ce qu'est le deep learning !

Beaucoup de questions

Comment on choisit la structure du réseau ? Les non linéarité ?
Comment est ce qu'on optimise θ ? Quelle fonction de perte ?
Quelle stratégie de mise à jour ? ...
Multiclasses ? entrée non vectorielle ? ...
Quelles garanties ?

Beaucoup de questions...
Peu de réponses !

L'essentiel

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on choisit une structure $DL(x, \theta)$

on optimise θ avec l'objectif que $sign(DL(x_i, \theta)) \approx y_i$

dans l'espoir que $\forall i \in \{1, ..., n\}$, $sign(DL(x'_i, \theta)) \approx y'_i$

Plan

- Exemples d'application du Deep learning
- Formalisation de la classification
  - Séparateur linéaire
  - Réseau de neurones
  - Deep learning
- Quelques points de théorie
- Un peu de pratique

Quelques points de théorie

Comment évaluer un système deep learning ?

base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on regarde si $\forall i \in \{1, ..., n\}, sign(DL(x'_i, \theta)) \approx y'_i$

Quelles garanties ?

Probabilité d'erreurs

On a f (apprentissage ou pas) et on voudrait que $f(x) = y(x)$.
Supposons que $f(x) \neq y(x)$ avec probabilité p.
Si on tire uniformément $x_1, y_1, \ldots, x_N, y_N$, alors :

$$P\left(f(x_n) = y_n\right) = \left(1 - p\right)^N$$

C'est une justification fondammentale de l'évaluation empirique : *si l'algorithme fait des erreurs, on devrait en trouver en l'évaluant.*

Quelles garanties ?

Probabilité d'accidents

Connaissant f, un accident est une situation où $f = y$ sur une base d'évaluation $x_1, y_1, \ldots, x_N, y_N$ mais où $f \neq y$ sur un échantillon χ.
Si $f(x) \neq y(x)$ avec probabilité p, alors la probabilité d'un accident est : $P(A) = p(1-p)^N = \phi(p)$.
Or, cette probabilité est bornée indépendamment de p :
$\phi'(p) = (1-p)^{N-1}(1-(N+1)p)$, donc, ϕ a un extremum en $\frac{1}{N+1}$ (qui est un maximum vu que $\phi(0) = \phi(1) = 0$).
Donc, $P(A) \leq \frac{1}{N+1}(1 - \frac{1}{N+1})^N$

Des garanties statistiques

Probabilité d'accidents

Connaissant f, un accident est une situation où $f = y$ sur une base d'évaluation $x_1, y_1, \ldots, x_N, y_N$ mais où $f \neq y$ sur un échantillon χ.
La probabilité d'un accident vérifie :

$$P(A) \leq \frac{2}{e} \frac{1}{N+1}$$

Attention

000	echec à l'évaluation
001	echec à l'évaluation
010	echec à l'évaluation
100	echec à l'évaluation
110	echec à l'évaluation p=2/3, accident p=1/3
101	echec à l'évaluation p=2/3, accident p=1/3
011	echec à l'évaluation p=2/3, accident p=1/3
111	bon fonctionnement

Dans le cas uniforme, la probabilité d'accident est la même que la probabilité de bon fonctionnement (1/8) !

La borne de la probabilité d'accident est principalement une borne de non passage de l'évaluation !

Est ce qu'une probabilité d'accidents faibles est suffisante sur un système critique ? ?

Oui, Non, En fait ça dépend...

Oui : la probabilité d'un crash d'avion doit être inférieur à 10^{-9} (par heure de vol) mais pas nécessairement nulle.

Non : aujourd'hui, la norme DO-178 ne considère que des systèmes logiciels infaibles. Un crash ne peut résulter que d'une panne matériel ou d'une sortie du domaine d'emploi.

En fait ça dépend : on accepte un crash provenant d'une sortie du domaine d'emploi probabilisé - ne peut on pas voir le module ia comme infaible mais incompatible avec certain éléments extérieurs faiblement probables ?

Vérification

Qu'est ce que la vérification formelle ?

La vérification formelle consiste à transformer des propriétés des programmes informatiques en assertion mathématique, puis à démontrer ces assertions.

Un programme dont les spécifications sont vérifiées formellement est un programme **infaible** (vis à vis des spécifications).

Vérification formelle et Deep learning

Reluplex : il est possible de formaliser les spécifications d'un système ACAS. Et, il est possible de formaliser la satisfaction de ces spécifications par un réseau de neurones.
Si un réseau de neurones satisfait cette assertion, il est un système ACAS infaible.

Vérification

Pourquoi ne pas tout vérifier ?

Pour vérifier il faut pouvoir spécifier !
Comme on ne sait définir l'ensemble *des images de chat*, on ne peut spécifier que le réseau doit classer comme *chat* l'ensemble *des images de chat*.

Donc évaluation empirique

base de test $(x'_1, y'_1), \ldots, (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$
on regarde si $\forall i \in \{1, \ldots, n\}, sign(DL(x'_i, \theta)) \approx y'_i$
donc risque d'erreurs non négligeable

Certification ?

Est ce qu'une probabilité d'accidents faibles est suffisante sur un système critique ? ?

Aujourd'hui, on accepte des accidents probabilisés - parce qu'on les comprends : vents, radar, ...

Acceptera t on des accidents probabilisés (qu'on ne comprends pas) ?

Le fait qu'on comprenne une erreur ne modifie pas sa dangerosité, mais modifie son acceptabilité...
La réponse est autant sociale que scientique et technique !

L'essentiel à retenir

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on choisit une structure $DL(x, \theta)$

on optimise θ avec l'objectif que $sign(DL(x_i, \theta)) \approx y_i$

dans l'espoir que $\forall i \in \{1, ..., n\}, sign(DL(x'_i, \theta)) \approx y'_i$

cette qualification est considérée, aujourd'hui, comme insuffisante tout en étant pourtant présente dans les systèmes critiques

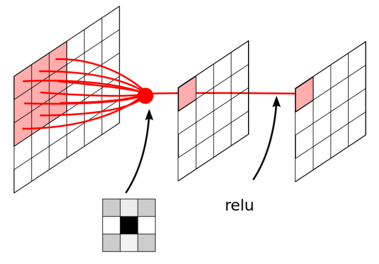
Classification d'images

1 image RGB de 480x340 = 1 vecteur dans $[0, 255]^{163200}$

Il est impossible de faire de la classification d'images sans tenir compte des spécificitées des images

Petits éléments pratiques

Convolution



$$\text{l'image } x \in \mathbb{R}^{H \times W \times Ch}$$
$$+ \text{ le filtre de convolution } w \in \mathbb{R}^{\Delta H \times \Delta W \times Ch}$$
$$= \text{ une } \textit{image} \text{ en sortie } y \in \mathbb{R}^{H - \Delta H \times W - \Delta W}$$

$$y_{h,w} = \sum_{dh,dw,ch \in [0,\Delta H] \times [0,\Delta W] \times [0,Ch]} x_{h+dh,w+dw,ch} \times w_{dh,dw,ch}$$

Petits éléments pratiques

Convolution

$$y_{h,w} = \sum_{dh,dw,ch \in [0,\Delta H]\times[0,\Delta W]\times[0,Ch]} x_{h+dh,w+dw,ch} \times w_{dh,dw,ch}$$

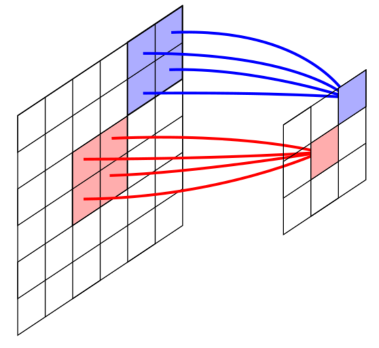
comme un filtre linéaire mais prenant en compte l'aspect spatial des images

permet d'extraire de l'image des informations locales (contrastes, texture, bords...) avec un nombre de poids très inférieur

à compléter avec un non linéarité (ex : relu)

Petits éléments pratiques

Pooling



l'image en entrée $x \in \mathbb{R}^{H \times W \times Ch}$

l'image en sortie $y \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times Ch}$

$$y_{h,w,ch} = \max_{dh,dw \in [0,1]} x_{2h+dh,2w+dw,ch}$$

Petits éléments pratiques

Pooling

$$y_{h,w,ch} = \max_{dh,dw \in [0,1]} x_{2h+dh,2w+dw,ch}$$

augmentation du champs récepteur de la convolution
invariance à de petites transformations spatiales

L'essentiel à retenir

base d'apprentissage $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), \ldots, (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on choisit une structure $DL(x, \theta)$

on optimise θ avec l'objectif que $sign(DL(x_i, \theta)) \approx y_i$

dans l'espoir que $\forall i \in \{1, \ldots, n\}, sign(DL(x'_i, \theta)) \approx y'_i$

cette qualification est considérée, aujourd'hui, comme insuffisante
tout en étant pourtant présente dans les systèmes critiques

Plan

- Exemples d'application du Deep learning
- Formalisation de la classification
  - Séparateur linéaire
  - Réseau de neurones
  - Deep learning
- Quelques points de théorie
- Un peu de pratique

TP !