

Apprentissage machine & deep learning

Réseaux de neurones

A. Boulch, A. Chan Hon Tong, S. Herbin, B. Le Saux



Plan de la séance

Introduction

Les neurones

Multi-couche

Exemple : classification d'images



Labels:

Voiture
Chien
Maison
Avion
Jouet
Jeu
Balle
Chat
Route
Arbre
Lego
Ski
Burger
Tasse
Soleil
Fleur
...

Images + Labels

Exemple : classification d'images

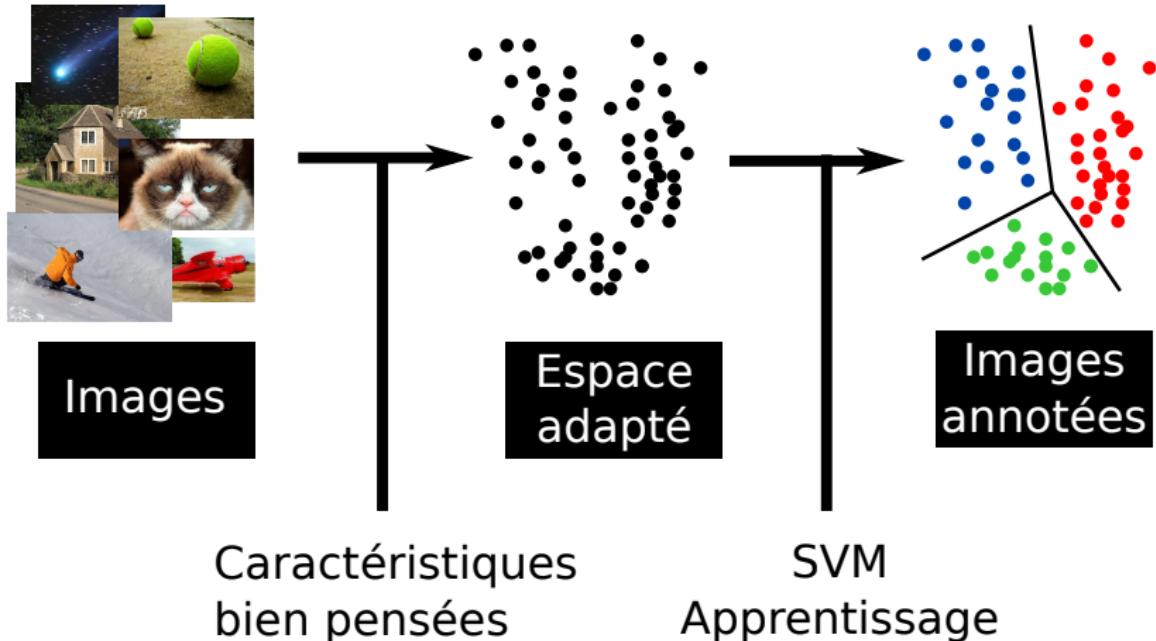


Labels:

Voiture
Chien
Maison
Avion
Jouet
Jeu
Balle
Chat
Route
Arbre
Lego
Ski
Burger
Tasse
Soleil
Fleur
...

Images + Labels → Images annotées

Caractéristique + SVM



Les limites



Classification binaire



Qu'est ce qu'un bon descripteur pour des oranges ?



Les limites



Classification binaire



Qu'est ce qu'un bon descripteur pour des oranges ?



Les limites



Classification binaire



Qu'est ce qu'un bon descripteur pour des oranges ?



Rond
Couleur: orange



Les limites



Classification binaire



Qu'est ce qu'un bon descripteur pour des oranges ?



Rond
Couleur: orange



Les limites



Classification binaire



Qu'est ce qu'un bon descripteur pour des oranges ?



Rond
Couleur: orange
Pas de ligne



Les limites



Classification binaire



Qu'est ce qu'un bon descripteur pour des oranges ?



Rond
Couleur: orange
Pas de ligne



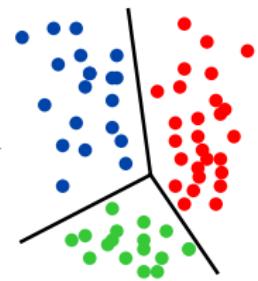
Tout apprendre ?



Images

Réseaux de
neurones profonds

Apprendre à la fois
les caractéristiques
et le classifieur



Images
annotées

Plan de la séance

Introduction

Les neurones

Décision linéaire

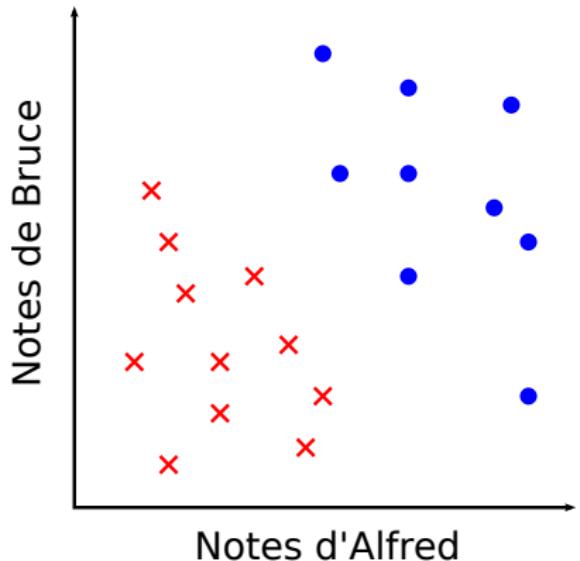
Descente de gradient stochastique

Multi-couche

Problème de recommandation

Un exemple simple

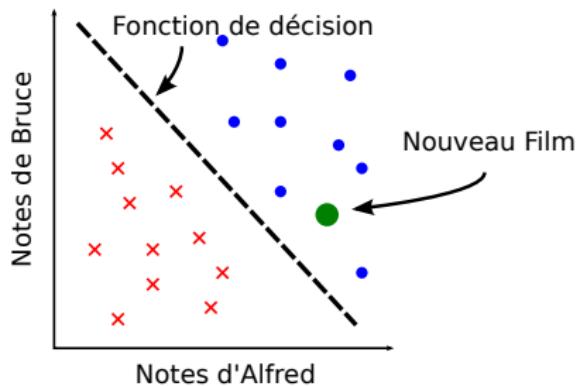
Étant données les notes de deux personnes et mon appréciation, sur un ensemble de films, prédire mon avis pour un nouveau film (noté par Alfred et Bruce).



Décision linéaire

Objectif

Une fonction linéaire de $\mathbb{R}^2 \rightarrow \{0, 1\}$ qui vaut 1 pour les films appréciés, 0 sinon.



Expression

Notant x_1 et x_2 les notes d'Alfred et Bruce. La séparation :

$$w_1x_1 + w_2x_2 + b = \mathbf{w}^T \mathbf{x} + b = 0$$

Notons h la fonction :

$$h : \mathbb{R}^2 \longrightarrow \mathbb{R}$$

$$\mathbf{x} \mapsto \mathbf{w}^T \mathbf{x} + b$$

h vaut 0 sur la frontière et est de signe différent de part et d'autres.

Expression

En utilisant la fonction signe, on a un classifieur linéaire :

$$s(x) = \begin{cases} 1, & \text{if } x \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Fonction constante par morceaux, à dérivée nulle \implies optimisation difficile. On utilise une approximation de la fonction signe, la fonction sigmoïde :

$$h : \mathbb{R}^2 \longrightarrow \{0, 1\}$$

$$\mathbf{x} \mapsto g(\mathbf{w}^T \mathbf{x} + \mathbf{b}) \text{ where } g(z) = \frac{1}{1 + \exp(-z)}$$

→ Perceptron !

Fonction objectif

Objectif

Trouver \mathbf{w} et b correspondants à nos données existantes (les couples $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathbb{R} \times \{0, 1\}$) :

$$\forall i \quad h(\mathbf{x}^{(i)} | \mathbf{w}, b) \approx y^{(i)}$$

Critère

$$J_{\mathcal{M}}(\mathbf{X}, \mathbf{Y} | \mathbf{w}, b) = \sum_{i \in \mathcal{M}} (h(\mathbf{x}^{(i)} | \mathbf{w}, b) - y^{(i)})^2$$

Le minimum de $J_{\mathcal{M}}$ est atteint lorsque $h(\mathbf{x}^{(i)} | \mathbf{w}, b) = y^{(i)}$:

$$\arg \min_{\mathbf{w}, b} J_{\mathcal{M}}(\mathbf{X}, \mathbf{Y} | \mathbf{w}, b)$$

Descente de gradient stochastique

Toutes les fonctions f lisses vérifient

$$\forall x, \nabla f_x \neq 0 \Rightarrow \exists \epsilon > 0, f(x) > f(x - \epsilon \nabla f_x)$$

descente de gradient en apprentissage :

$$f(\mathbf{w}, b) = J_{\mathcal{M}}(\mathbf{X}, \mathbf{Y} | \mathbf{w}, b)$$

descente de gradient stochastique :

$$f(\mathbf{w}, b) = J_{\mathcal{M}' \subset \mathcal{M}'}(\mathbf{X}, \mathbf{Y} | \mathbf{w}, b)$$

\mathcal{M}' est tiré uniformément à chaque itération

Descente de gradient stochastique

$$\begin{aligned}\mathbf{w} &= \mathbf{w} - \alpha * \Delta \mathbf{w}^{(i)} \\ b &= b - \alpha * \Delta \mathbf{b}^{(i)}\end{aligned}$$

Algorithme

1. **Initialisation** : \mathbf{w} et b aléatoires
2. Boucle
 - 2.1 Choisir un couple $(x^{(i)}, y^{(i)})$ au hasard
 - 2.2 Calculer Δw_1 , Δw_2 et Δb
 - 2.3 Mettre à jour \mathbf{w} et b

Perceptron en python

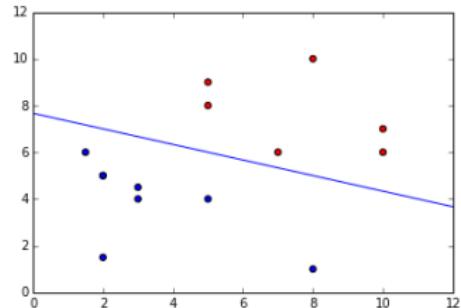
Code

```
from sklearn.linear_model
import perceptron

dataset = pd.DataFrame.from_items([
    ('Alfred', [10.0, ... ,1.5]),
    ('Bruce', [6.0, ... ,6.0]),
    ('ClassId',[1, ... ,0])])

m_perceptron = perceptron.Perceptron(
    n_iter=1000,
    verbose=0, random_state=None,
    fit_intercept=True, eta0=0.001)

m_perceptron.fit(dataset[['Alfred', 'Bruce']],
                 dataset['ClassId'])
```



Perceptron en python

Code

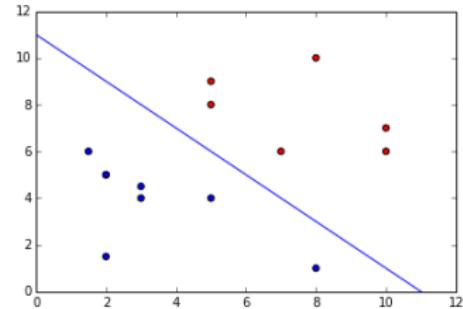
À comparer avec le résultat d'une SVM sur le même problème :

```
from sklearn import svm

dataset = pd.DataFrame.from_items([
    ('Alfred', [10.0, ... ,1.5]),
    ('Bruce', [6.0, ... ,6.0]),
    ('ClassId',[1, ... ,0])])

m_svm_clf = svm.SVC(C=1.0,
    kernel='linear')

m_svm_clf.fit(dataset[['Alfred', 'Bruce']],
    dataset['ClassId'])
```



→ Perceptron = modèle de neurone unitaire, à combiner dans des architectures plus complexes

Plan de la séance

Introduction

Les neurones

Multi-couche

Décision non linéaire

Couches

Algorithme de rétro-propagation

Algorithme de rétro-propagation

1 filtre linéaire :

$$x \in \mathbb{R}^D \rightarrow wx + b \in \mathbb{R}$$

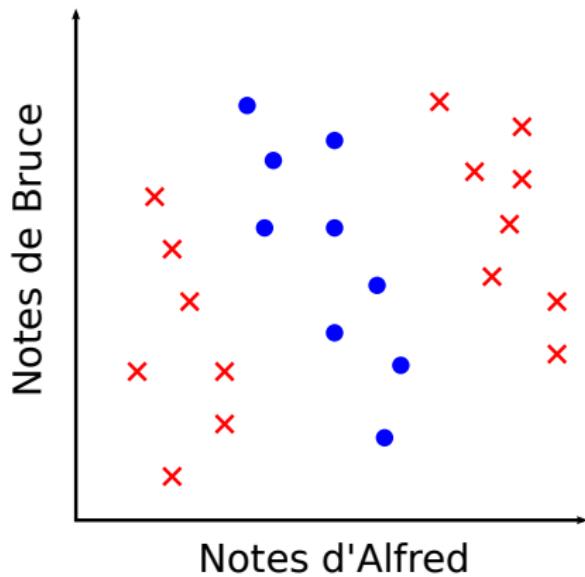
K filtres linéaires :

$$x \in \mathbb{R}^D \rightarrow (w_k x + b_k) \in \mathbb{R}^K$$

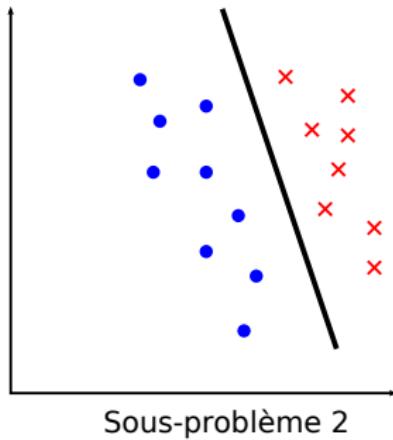
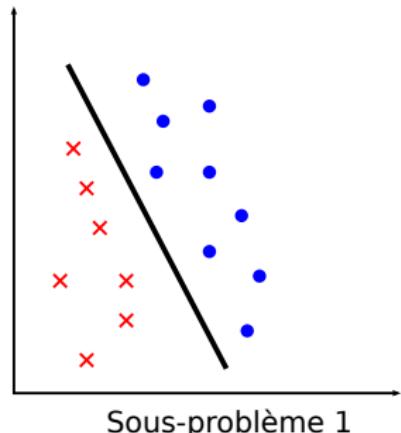
K filtres linéaires puis 1 filtre :

$$x \in \mathbb{R}^D \rightarrow (w_k x + b_k) \in \mathbb{R}^K \rightarrow w' (w_k x + b_k) + b' \in \mathbb{R}$$

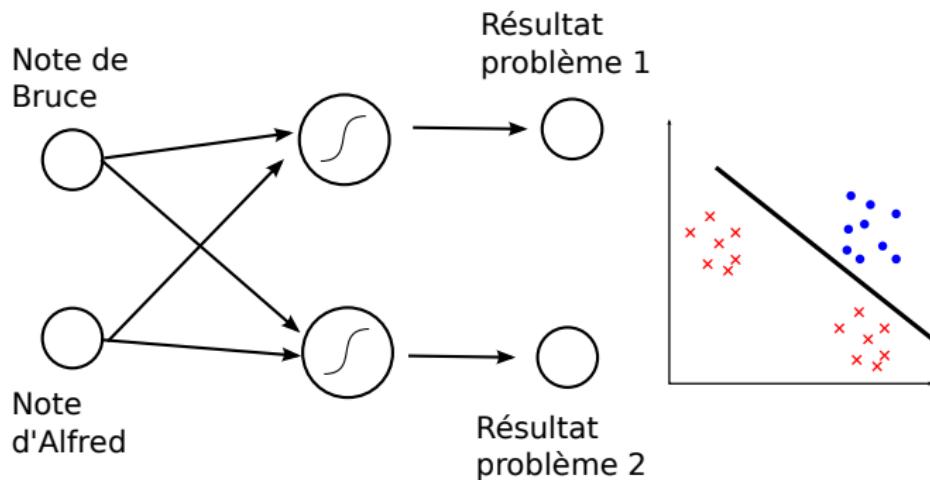
Exemple



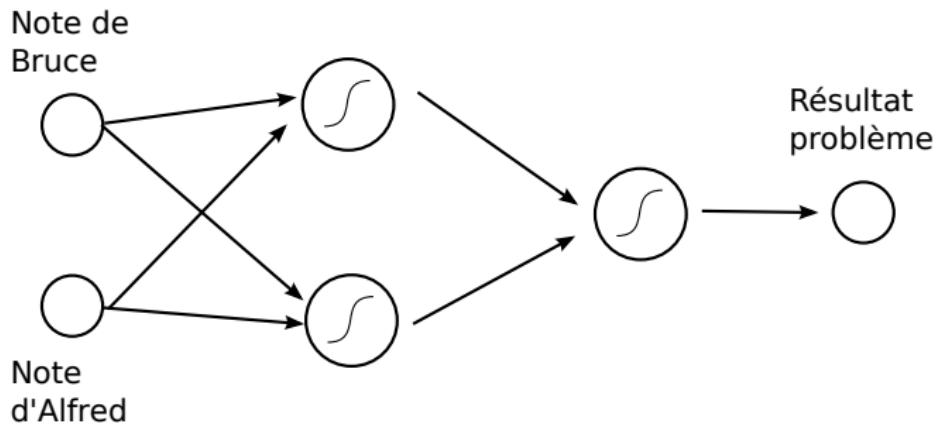
Exemple



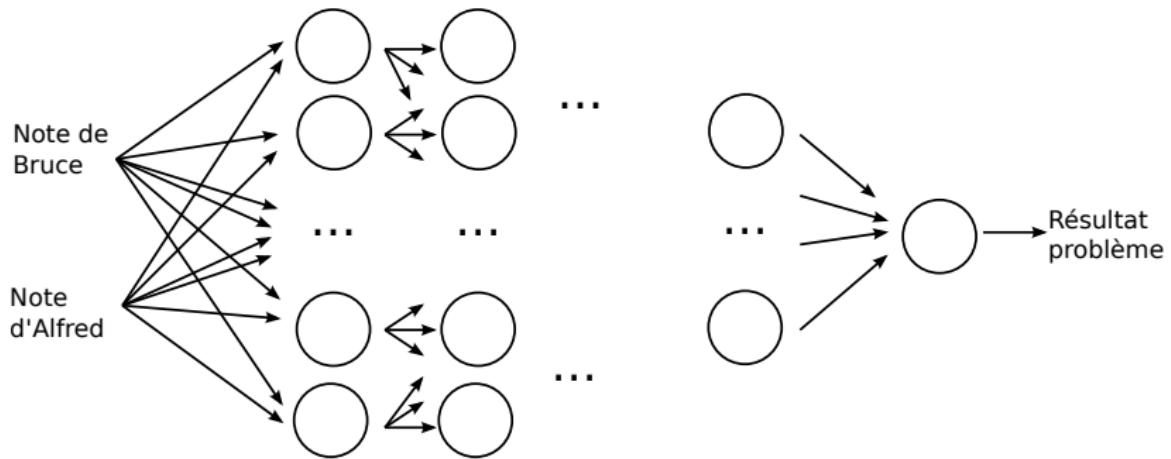
Exemple



Exemple



Plusieurs couches de neurones



Algorithme de rétro-propagation

Le nombre de variables devient gigantesque et la fonction f complète devient compliquée **comment calculer le gradient ?**

heureusement si $h(x) = f(g(x))$ alors

$$h'(x) = g'(x) f'(g(x))$$

chaque couche peut être dérivée vis à vis de la suivante seulement

Algorithme de rétro-propagation

Phase 1 : Propagation

- ▶ Prop. “forward” de l'échantillon d'apprentissage pour générer des activations
- ▶ Prop “backward” de l'erreur de prédiction par rapport à la cible, pour générer les deltas des poids

Phase 2 : Mise-à-jour des poids

- ▶ $\text{delta} * \text{activation d'entrée} \Rightarrow \text{gradient du poids}$
- ▶ une fraction du gradient est soustraite du poids