Introduction à l'intelligence artificielle

Romain Giot

2019

IUT de Bordeaux - LP

Introduction

Syllabus i

- 10h de cours
  - 2h ce matin – Un peu de slides, de dessins au tableau et de l'interaction
  - 2h cet après-midi – Lecture de code, début de la pratique
  - 2h mercredi – Pratique
  - 2h vendredi matin – Sujet à définir avec vous en fin de séance (algorithmes évolutionnaires ? algorithmes de partitionnements ? Prédiction dans les systèmes de vélos libre service ?)
  - 2h vendredi après-midi – Exercices dépendants du matin (DS – questions sur ce qu'on a vu)
- But : acquérir quelques notions liées à l'IA (très orienté apprentissage). Durée trop faible pour être exhaustif
- Focus : apprentissage profond / on peut dévier en live si besoin

Syllabus ii

- Enseignant : Romain Giot / Maître de conférence
  - Recherche liée à différentes utilisations de l'apprentissage automatique
    - authentification biométrique : rides du doigt, dynamique de frappe, modèles adaptatifs
    - villes intelligentes : prédiction d'usage des systèmes de vélos libre service, segmentation d'images satelites

Définition i

- Définition du dictionnaire :
  L'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence.

- Définition de Marvin Lee Misky
  La construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique.

- Dans les faits avants :

Définition ii

Systèmes SAT, algorithmes basés sur des heuristiques, algorithmes codés en dur

- Dans les faits aujourd'hui :
  Apprentissage profond (deep learning)

Historique

- **1666** – *calculus ratiocinator* de Gottfried Wilhelm Leibniz. Machine, qui permettrait de démêler le vrai du faux dans toute discussion dont les termes seraient exprimés dans la langue philosophique universelle.
- **1943** – Neurone artificiel qui mime le comportement d'un neurone biologique
- **1950** – Test de Turing
- **1955** – Raisonnement symbolique
- **1957** – Perceptron de Rosenblatt $\sum_{i=1}^{n} x_1 w_1 > \theta$?
- **1959** – General Problem Solver
- **1960** – SAT solver
- $\sim$**1980** – Retropropagation du gradient
- **1997** – Deep blue bat Kasparov
- **2005** – Première conduite autonome sur route inconnue
- **2012** – Apprentissage profond

L'intelligence artificielle dans la vie quotidienne

- **Aujourd'hui**
  - Traduction automatique
  - Planification de trajet
  - Assistant personnel
  - Upscaling jeux ou vidéos
- **Demain**
  - Voiture intelligente
  - Jeux Vidéos
  - Œuvres artistiques
  - Enseignement

L'intelligence artificielle à Bordeaux

- Laboratoire de Recherche en Informatique Bordelais / INRIA / CNRS
- Startups (QUCIT, http://www.bordeauxdatascience.fr/)
- Robocup en 2020 https://www.u-bordeaux.fr/Actualites/De-l-universite/Bordeaux-accueillera-la-RoboCup-20202

Métiers de l'intelligence artificielle

- **Chercheur** : développe de nouveaux modèles
- **Data-scientist** : adapte des modèles existants à ses propres problèmes
- **Intégrateur** : utilise un système intelligent par le bias d'une API spécifique (Azure AI de Microsoft, Watson de IBM, IA Vision de Google, ..)

Apprentissage automatique

Définition

L'apprentissage automatique (en anglais machine learning, littéralement « l'apprentissage machine ») ou apprentissage statistique est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches statistiques pour donner aux ordinateurs la capacité d' « apprendre » à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. Plus largement, cela concerne la conception, l'analyse, le développement et l'implémentation de telles méthodes.

Différentes familles existent

- **Supervisé** : utilisation de données étiquettées pour apprendre
- **Non-supervisé** : utilisation de données non-étiquettées pour apprendre
- **Semi-supervisé** : utilisation de données partiellement étiquettées
- **Par-renforcement**: l'agent apprend avc l'experience en interagissant avec son environement
- **Actif** : un humain guide l'apprentissage
- **En-ligne** : le modèle appris est mis à jour au cours de son utilisation

Nous nous intéresserons à la famille supervisée.

Qu'est-ce que nous pouvons faire en supervisé avec un nouvel exemple ?

- **Classification** : attribuer une (ou plusieurs) classes (l'image contient du cheval et de l' humain)
- **Segmentation** : attribuer une classe à chaque pixel (ses pixels contiennent un cheval, ceux-là un humain)
- **Segmentation d'instances** : attribuer une classe et l'individu concerné par la classe à chaque pixel (il y a un cheval dans ses pixels, un autre dans ceux-ci, . . .)
- **Régression** : attribuer une valeur (le nombre de calories est X)
- **Encodage** : convertir la donnée (transformer la photo comme si elle est peinte par Van Gogh)
- **Génération** : transformer un ensemble de valeurs aléatoires en visage artificiel réaliste

Nous nous intéressons à la classification.

Apprendre un modèle supervisé

- **Objectif** : *apprendre* (i.e., *optimiser*) les paramètres θ d'un modèle f en utilisant un jeu de données étiquetté $(x_n, y_n)_{1 \leq n \leq N}$, avec $x_n \in X$ et $y_n \in Y$.
  - $Y = \mathbb{R}$: régression
  - $Y = \{0, 1\}$: classification binaire
  - $Y = \{1, ..., I\}$: classification avec I classes
- L'algorithme d'optimisation dépend du type de modèle
- En général, il faut minimiser en fonction de coût L qui dépend du problème
  - $argmin_\theta \sum_i L(f(x_i, \theta), y_i)$

Utiliser un modèle

Une fois les paramètres θ du modèle f appris, il est trivial de l'utiliser sur la nouvelle donnée x:

- $y = f_\theta(x)$

Évaluer un modèle

L'évaluation d'un modèle consiste tout simplement à :

- l'utiliser sur une base d'exemples non utilisés lors de l'apprentissage
- calculer une métrique d'évaluation (taux de reconnaissance par exemple)

Gérer le manque de données pour apprendre/tester

La **validation croisée** permet de gérer le manque de données pour apprendre et valider en :

- partageant le jeu de données en N ensembles
- utiliser N-1 ensembles pour apprendre
- utiliser 1 ensemble pour tester
- répéter l'opération en changeant l'ensemble de test

Différents stratégies existent : K-fold, aléatoire, LeaveOneOut, ...
+ version stratifiée

Optimiser les hyper-paramètres du modèle

- La réalité est un peu plus compliqué : les modèles peuvent être paramétrés par des hyper-paramètres qui ne sont pas appris.
- Les configurer n'est pas une tâche aisée.
- Différentes approches existent : sélection aléatoire, recherche en grille, algorithmes d'optimisation (descente de gradient, évolutionnaires, ...)
- 3 jeu sont nécessaires : apprentissage, validation, test

Algorithmes

Différents algorithmes existent et ont des performances différentes.

Les plus communs sont :

- K-plus proches voisins
- Arbres de décisions
- Machines à vecteurs supports
- Réseaux de neurones
- Réseaux de neurones profonds (mais on verra après)

Écosystème python

- numpy : Bibliothèque de manipulation de tenseurs. Ultra-optimisée, écrite en C
- pandas : Bibliothèque de manipulation de tables avec des colonnes nommées. Basée sur numpy
- matplotlib : Bibliothèque de visualisation de graphiques
- scikit-learn : Bibliothèque d'apprentissage. Assez optimisée

Exemples d'utilisation de scikit learn

```
from sklearn import datasets
from sklearn import svm

iris = datasets.load_iris()
X = iris.data
Y = iris.target



clf = svm.SVC(gamma=0.001, C=100.)
clf.fit(X[:-1], Y[:-1])
print(clf.predict(X[-1:]))
print(Y[-1])
```

Apprentissage profond

Apprentissage vs Apprentissage profond

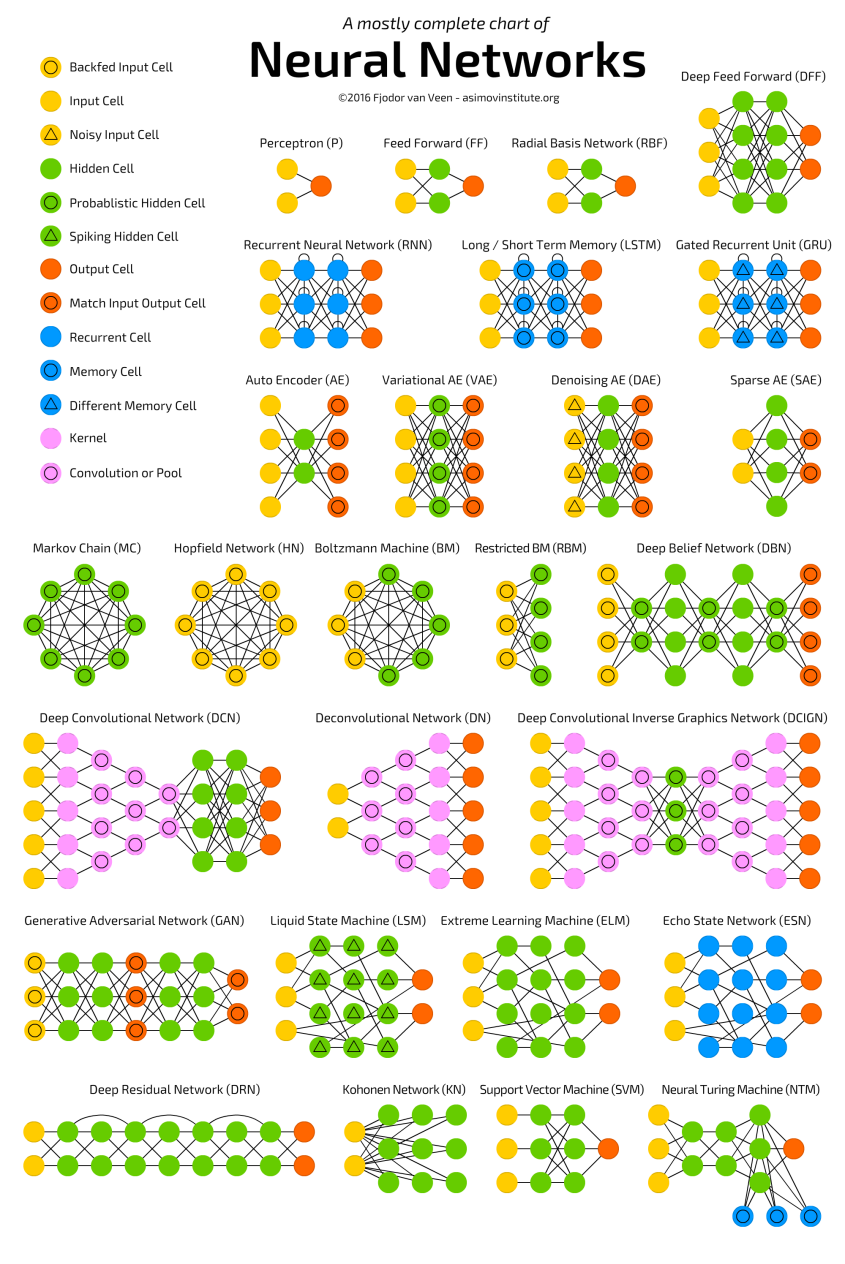
- Petit rappel :
  - pour apprendre un modèle, il faut lui donner des données...
  - Mais à quoi ça correspond au juste ? Des informations extraites (un histogramme de couleurs) depuis les données brutes (une image) par un humain ou des algorithmes codés par des humains
  - Et alors ? Hé bien c'est compliqué !
- Qu'est-ce qui change ici ?
  - Plus besoin de faire cette extraction des caractéristiques
  - On fourni directement la donnée brute a traiter
- Comment on peut faire ?
  - On ajoute plus de traitements dans le modèle
  - Les traitements du début extraient l'information et la façon d'extraire est apprise lors de l'apprentissage
  - Les traitements de fin répondent au problème

Conséquences

- Les modèles sont plus compliqués
  - le nombre de paramètres à apprendre explose
  - les méthodes d'optimisation des paramètres fonctionnent mal (gradient vanishing)
- L'apprentissage change
  - les algorithmes d'optimisation ont du être améliorés
  - il faut plus (BEAUCOUP) plus de données pour apprendre (big data à la rescousse)
  - if faut plus (BEAUCOUP) plus de puissance de calcul pour apprendre (GPU à la rescousse)

Si le sujet vous intéresse : http://d2l.ai/index.html.

Les différents modèles



Écosystème Python

- `tensorflow` bibliothèque de calcul de tenseurs de Google
- `cntk` bibliothèque bas niveau de deep-learning de Microsoft
- `keras` bibliothèque de deep-learning utilisant `tensorflow` ou `cntk`
- `pytorch` bibliothèque de deep-learning de Facebook
- tous les projets opensource d'autres compagnies (Adobe, Netflix, Yahoo, Baidu, Zalando, ...)

Les mains dans le camboui

Lecture de code

- Hello world de scikit-learn
- Hello world de keras

Travail à faire

- À partir de ces 2 hello worlds + lecture de la doc des différentes bibliothèques
- Classification de la base FashionMNIST (10 catégories de vêtements de Zalando)
- Validation croisée
- Optimisation des paramètres du modèle ou du choix du modèle

Ne pas hésiter à travailler avec un sous-ensemble de la base pour aller plus vite