GT05

"Discussions sur les thèmes de l'apprentissage automatique et les réseaux de neurones"

A. Boucaud, P. Navaro, J.L Parouty, L. Risser

Structuration des discussions du GT

- Partie 1: Modèles d'apprentissage et architectures de réseaux de Neurones (15 minutes)
- Partie 2: Langages, packages et usages (15 minutes)
- Partie 3: Hardware et grandes infrastructures (15 minutes)
- Partie 4: Julia et outils émergents (15 minutes)

Partie 1

Modèles d’apprentissage et architectures de réseaux de Neurones

15 minutes

Modèles classiques pour l'apprentissage

Grande variété d'outils classiques pour l'apprentissage :

- Regression linéaire et logistique
- Méthodes par arbres (arbres de décisions, forêts aléatoires)
- Méthodes à Vecteurs de Support (SVM)
- Clustering
- Réduction de dimension

Architectures de réseaux de neurones

Partie 2

Langages, packages et usages

15 minutes

Langages pour faire du ML

Interpreté (wrapper)	**Compilé**
Python	C++
R	Java
Javascript	Scala
	Clojure
	Swift
	Matlab

Principales librairies (Python)

Machine learning	Deep learning
Scikit-learn	Keras / TensorFlow
XGBoost	Fast.ai / PyTorch
LightGBM	spaCy

Exploration et mise en forme des données

Python	R	Julia
Scikit-learn	Dataframes	DataFrames.jl
Pandas	Tidyverse	CSV.jl
Datasette		

Partie 3

Hardware et grandes infrastructures

15 minutes

Usages

- Sur machine perso
- En ligne avec Google Colab
- Serveurs HPC
- AWS, Google Cloud, MS Azure
- Cluster Hadoop

Architectures de calcul

- Diversité d'architectures de calcul pour l'apprentissage en dimension raisonable
  - Processeurs multi-coeurs
  - GPU, TPU, eGPU
  - GraphCore, Optical Processing Units (OPU)
- Passage à l'échelle de l'apprentissage
  - Serveurs intra-Laboratoires
  - Mésocentres
  - Grandes infrastructures de calcul
- Transfert de données massives vers un serveur ?

Partie 4

Julia et outils émergents

15 minutes

Le langage Julia

- Facile à utiliser car interactif et dynamique
- On peut utiliser les notebooks Jupyter
- Performant car tout est compilé
- Syntaxe simple proche des mathématiques
- Dispatch multiple et macros
- Accès très facile vers les GPU
- Création de package très simple favorisant le partage de code et la reproductibilité

Les défauts (de jeunesse)

- Pas de package incontournable en ML.
- Communauté encore réduite, pas de gros acteur.
- Pas d'environnement de programmation convivial et le développement en Julia peut être pénible à cause de la lattence de la compilation.

Quand apprendre Julia?

- Maintenant! Si vous voulez développer un package performant
- Dans 1-2 ans avec une meilleure expérience de programmation.
- Dans 3-5 ans ce sera aussi facile qu'avec Python ou R, la performance en plus.

Quelques outils émergents

- ONNX - open standard for machine learning interoperability
- TensorFlow Probability - bring uncertainty to ML models
- JAX - compile and run your NumPy programs on GPUs and TPUs
- hummingbird - convert a ML model to PyTorch