Atelier SD1: Reconnaissance de caractères manuscrits (MNIST)

Résumé

Présentation du problème de reconnaissance de caractères manuscrits (MNIST DataBase) à partir d'images numérisées. L'objectif n'est pas la recherche de la meilleure prévision mais une comparaison des performances de différentes technologies ou librairies (R, H20, Scikit-learn, Spark MLlib).

1 Introduction

1.1 Objectifs

L'objectif général est la construction d'un meilleur modèle de reconnaissance de chiffres manuscrits. Ce problème est ancien (zipcodes) et sert souvent de base pour la comparaison de méthodes et d'algorithmes d'apprentissage. Le site de Yann Le Cun : MNIST DataBase, est à la source des données étudiées, il décrit précisément le problème et les modes d'acquisition. Il tient également à jour la liste des publications proposant des solutions avec la qualité de prévision obtenue. Ce problème a également été proposé comme sujet d'un concours Kaggle mais sur un sous-ensemble des données.

De façon très schématique, plusieurs stratégies sont développées dans une vaste littérature sur ces données.

- Utiliser une méthode classique (k-nn, random forest...) sans trop raffiner mais avec des temps d'apprentissage rapide conduit à un taux d'erreur autour de 3%.
- Ajouter ou intégrer un pré-traitement des données permettant de recaler les images par des distorsions plus ou moins complexes.
- Construire une mesure de distance adaptée au problème, par exemple invariante par rotation, translation, puis l'intégrer dans une technique d'apprentissage classique comme les k plus proches voisins.
- Utiliser une méthode plus flexibles (réseau de neurones "profond") avec une optimisation fine des paramètres.
- ...

L'objectif de cet atelier n'est pas de concurrencer les meilleurs prévisions mais de comparer sur des données relativement volumineuses les performances de différents environnements technologiques et librairies. Une dernière question est abordée, elle concerne l'influence de la taille de l'échantillon d'apprentissage sur le temps d'exécution ainsi que sur la qualité des prévisions.

Principalement 4 environnements sont testés.

- R classique disponible sur le site du CRAN.
- Python en utilisant la librairie `scikit-learn`
- H2O mais seulement sous la forme du module R qui est disponible.
- Enfin la librairie MLlib de Spark à l'aide de l'API `pyspark` et donc finalement en langage python.

Remarque importante, les meilleures résultats actuellement obtenus font appel à des modèles invariants par transformation : *scattering* de ou apprentissage profond avec réseaux de neurones "convolutionnels" dont de nombreuses librairies proposent des implémentations : tensorflow, torch... amis nécessitant des moyens techniques (cluster, cartes GPU) plus conséquents ; seul H20 est testé.

2 Déroulement de l'atelier

2.1 Ressources

Plusieurs tutoriels d'initiation sont disponibles afin d'acquérir les compétences techniques nécessaires en plus de celles du cours d'apprentissage statistique.

Pédagogiques

- Compléments sur R avec RHadoop pour introduire les fonctionnalités MapReduce.
- Compléments sur Python langage fonctionnel et orienté objet.
- Introduction à la technologie H2O qui implémente quelques méthodes d'apprentissage pour données massives dont une version simplifiée de *deep learning*.
- Introduction à pySpark et à la manipulation de bases de données résí-

lientes.
- Introduction à MLlib qui implémente quelques algorithmes classiques d'apprentissage pour données massives car *scalable* (passe à l'échelle volume).

2.1.1 Matériel

En fonction du contexte : ordinateur individuel multi-cœurs, cluster...

3 Solutions élémentaires sans distorsion

3.1 Les données

Les données sont constituées d'un échantillon d'apprentissage : 60 000 images de 754 pixels à niveau de gris et d'un échantillon test de 10 000 images.

3.2 Avec R

La solution en R est décliné dans un calepin. Le télécharger et l'exécuter si le noyau `IRkernel` est bien installé pour permettre d'utiliser R dans un *notebook Jupyter*.

Sinon, exécuter les instructions de celui-ci dans un environnement comme RStudio. Noter les temps d'exécution, la précision estimée sur l'échantillon test.

3.3 Python `scikit-learn`

Le scénario propose une analyse classique de ce type de données pour illustrer les fonctionnalités de la librairie `scikit-learn`. Les données utilisées sont alors celles fournies avec la librairie et ne comporte que 1787 images digitalisées de chiffres manuscrits.

Comme pour R, un calepin décline la séquence des instructions à exécuter sur l'ensemble des données.

3.4 R et H2O

Le scénario d'introduction à H2O est relativement succinct alors que le tutoriel en ligne d'apprentissage de ce logiciel (page 40) propose lui aussi une étude de reconnaissances des caractères avec notamment une version simplifiée d'apprentissage profond ou *deep learning*.

Tester cet environnement sur un poste avec R, éventuellement sur un cluster ? Comparer les différentes stratégies de modélisation dont celle par réseaux de neurones qui nécessite une procédure complexe d'optimisation des nombreux paramètres alors qu'elle est présentée comme la "meilleure" approche sur les données de reconnaissance de caractères sans prétraitement de distorsion des images.

3.5 Avec MLlib

Un dernier calepin décline les instructions de l'API pyspark permettant d'opérer les fonctions de la librairies MLlib de Spark sur des bases de données résilientes et donc susceptibles de passer à l'échelle volume en s'adaptant aux capacités (nombre de machines ou de nœuds) d'un cluster.

4 Autres solutions

Est-il simple, possible, de tester d'autres solutions ?

4.1 Des articles

Un pré-traitement des données pour recaler les images ou la définition d'une distance spécifique ou encore l'utilisation d'un algorithme insensible à certaines transformations des images devient indispensable pour améliorer la prévision. Le site MNIST DataBase donne de nombreuses pistes.

Le scénario (en Matlab avec une fonction C) développe une solution proposée par Simard et al. (1998). Elle consiste à prendre en compte une distance dite tangentielle entre les images de caractères, distance invariante par un ensemble de transformations. Une implémentation efficace nécessiterait d'inclure le code C du calcul de cette distance dans des algorithmes d'apprentissage (comme k-nn) en python et des exécutions sur un cluster...

D'autres pistes peuvent être explorées, notamment celles des machines à noyaux par Müller et al. 2001 qui illustre ce type d'apprentissage par les données de MNIST sans semble-t-il faire mieux que la distance tangentielle.

Une des dernières approches en date est en connexion avec le *deep learning* en utilisant une décomposition, invariante par transformations, des images

par *scattering* sur bases d'ondelettes suivant les travaux initiés par Stéphane Mallat. L'application aux données MINST est explicitée par Bruna et Mallat (2013). L'effet de la taille de l'échantillon d'apprentissage y est également étudié.

4.2 Des librairies

L'apprentissage profond, sujet très à la mode et à gros enjeux en reconnaissance faciale sur les images, donne lieu à de très nombreux développements en relation, ou non, avec Python. En voici quelques uns :

- à partir de python : pylearn2, sklearn-theano intègre la librairie dédiée *Théano* pour, peut-être, en faciliter l'utilisation ;
- la librairie torch en pleine expansion ;
- le tutoriel de tensorFlow (Google) traite justement ces données.
- ...

5 Conclusion

L'un des objectifs de cet atelier est d'apporter des éléments de réponse à une question qui se pose naturellement face à des données massives. Quelle est la meilleure stratégie pour un temps de calcul contraint : apprendre une méthode "sommaire" sur toutes les données ou une méthode sophistiquée (optimisation des paramètres" sur un échantillon de celles-ci ? La comparaison entre les différentes implémentations de Random Forest : R, Python SciKit-learn, MLlib de Spark, contribue à la discussion.

Que dire des implémentations du *deep learning* et du coût de développement que représente la traque des dernières erreurs de classification ?