

ÉLÉMENTS LOGICIELS POUR LE TRAITEMENT DE DONNÉES MASSIVES  
RAPPORT FINAL  
5 février 2017

---

# Theano : kmeans & color quantization

---

Mehdi MIAH  
Duc-Vinh TRAN

ENSEIGNANTS :  
Xavier DUPRÉ  
Mathieu DURUT

---

Résumé :

Mots clés : Theano, kmeans, color quantization, big data

---

## 1 Introduction

Le clustering est un apprentissage non supervisé : ceci consiste en la création de groupes d'objets ayant une similarité. Ce problème a une complexité exponentielle : il n'est pas possible de le résoudre de manière optimale en un temps raisonnable. C'est pour cela qu'il est d'usage d'avoir recours à des heuristiques permettant l'obtention d'un résultat approché.

Un des algorithmes les plus connus pour effectuer cela est le k-means.

Appliqué à une image (chaque pixel de l'image est représenté par un vecteur de dimension 3, correspondant aux 3 couleurs RGB), cela permet de construire une nouvelle image contenant uniquement  $K$  couleurs. L'obtention de ces  $K$  couleurs se fera automatiquement et différera d'une image à l'autre.

## 2 Application

Notre objectif étant d'appliquer un color quantization sur une image, et donc sur possiblement des millions de points, il est important d'utiliser des implémentations compatibles avec de telles données. Pour cela, nous avons premièrement travaillé sur un petit dataset en ayant recours à un algorithme combinatoire sous `numpy`, puis à la version distribuée par `scikit-learn`. Et enfin, dans le contexte des big data, nous avons implémenté une version à travers le framework `Theano` qui sollicite aussi bien le processeur (CPU) que la carte graphique (GPU).

### 2.1 Application à de petits datasets

Il s'est avéré rapidement que la version combinatoire, et donc naïve, implémentée sous `numpy` était bien plus lente par rapport à la version de `scikit-learn`. Ces résultats étaient, à vrai dire, attendus.

Ensuite, l'implémentation sous `Theano`, sous CPU et sous GPU nous ont donnée les résultats suivants :

### 2.2 Application à de grands datasets

Une fois ces premiers résultats obtenus, l'adaptation à une image a été implémentée : une image possédant évidemment plus de points et se "décrivant" par bien plus de couleurs, cela a complexifié la tâche par :

- un plus grand nombre de points ;
- un plus grand nombre de clusters

### 2.3 Bonus : application à des images avec un nombre de clusters croissants

## 3 Difficultés rencontrés

Dans le cadre du cours *Éléments logiciels pour le traitement de données massives*, nous avons tout d'abord souhaité utiliser le langage `Cuda` pour implémenter une version sous GPU.

Puis, nous avons fait le choix de `Theano`. En effet, ce langage permet une implémentation intuitive de la descente de gradient. Toutefois, disposant d'un ordinateur sous Linux et d'un autre sous Windows, les installations ne furent pas aussi simple l'une que l'autre. Tandis que l'installation de `Theano` et de `Cuda` pour l'implémentation sous GPU fut rapide sous Linux, le groupe a éprouvé des difficultés à utiliser le GPU avec `Theano` sous Windows. En fin de compte, le code a été développé sur les deux terminaux, dont l'un disposait d'un accès à la GPU et l'autre non.

## 4 Conclusion