Big Data Mining - TD 1

L'objectif de ce TD est :
- apprendre à manipuler le calcul parallèle sous R, comprendre quand il peut être utile ou non;
- apprendre à manipuler des algorithmes de clustering (kmeans et mélanges gaussiens) sur des données réelles.
- s'initier aux fonctionnalités de MapReduce via R.

1. Comparer les temps d'exécution pour effectuer des calculs à l'aide :
- d'une boucle for classique,
- d'une boucle utilisant du calcul parallèle (foreach),
Pour cela, créer deux matrices M et V à l'aide du code suivant

```
n=1000
m=30000
M = matrix(rnorm(n*m), ncol=m)
V = matrix(rnorm(10*m), ncol=m)
```

Créer ensuite une fonction qui calcule la distance euclidienne entre une ligne de M et chaque ligne de V, à l'aide d'une fonction apply.

```
eucd <- function(u){
    euc.distu = apply(V, 1, function(w) sqrt(sum((u - w)^2)))
}
```

Utiliser cette fonction pour chaque ligne de M, en séquentiel puis en parallèle à l'aide du package foreach (regardez l'exemple dans le cours).
Vérifier que votre machine utilise bien plusieurs coeurs lorsque vous lui demandez d'exécuter des calculs en parallèle (Gestionnaire des tâches sous Windows).
Comparer les temps de calcul.

2. Récupérer les données MNIST sur http://yann.lecun.com/exdb/mnist/ ainsi que le fichier pour les manipuler mnist.r (en ligne sur le site de J.Jacques).
Commencer par tirer aléatoirement un échantillon de ces données (taille 1000)
Réaliser un clustering de ces données à l'aide de l'algorithme des kmeans en 1 à 20 classes, en séquentiel et en parallèle.
Comparer les temps de calculs.
Tracer les inerties intra-classes obtenues et essayer de choisir un nombre de clusters intéressant.
Analyser le clustering obtenu en représentant l'image moyenne de chaque cluster, pour le nombre de clusters que vous aurez retenu (et pour 10 clusters également si ce n'est pas ce nombre que vous avez retenu).

3. Nous allons maintenant utiliser un clustering à l'aide d'un modèle de mélange gaussien spécifique à la grande dimension (fonction hddc du package HDclassif). Dans un premier temps utilisons le modèle par défaut de la fonction hddc.
Faites comme précédemment et choisissez un nombre de clusters à l'aide du critère BIC (avec toujours un sous-échantillon de 1000 images parmi les 60000).
Analyser le clustering obtenu en représentant l'image moyenne de chaque cluster, pour le nombre de clusters que vous aurez retenu (et pour 10 clusters également si ce n'est pas ce nombre que vous avez retenu).

4. Comparer les partitions de 1000 images (en 10 clusters) obtenus par kmeans et hddc (en utilisant cette fois tous les modèles disponibles pour hddc, et en sélectionnant le meilleur par BIC) avec la *vraie* partition définie par les chiffres représentés sur les images.
Ces comparaisons seront faites à l'aide du critère ARI (Adjusted Rand Index, disponible via la fonction AdjustedRandIndex du package mclust).

5. Recommencer la question 4 sur les 60000 images. Attention, cela peut prendre beaucoup de temps de calcul…

6. En vous inspirant du tutoriel « MapReduce pour Statisticien », implémenter l'algorithme des k-means dans un formalisme Map Reduce. Il sera certainement préférable de commencer par implémenter une version classique de l'algorithme des k-means avant de passer à la version Map Reduce. Trois articles proposant des implémentations Map Reduce des k-means vous seront envoyés par email pour vous donner des idées d'implémentation.
Tester votre implémentation sur les données iris, en comparant votre partition obtenue à la variable espèce. Comparer vos résultats à ceux obtenus par la fonction kmeans de R.