IFT 3700 – Science des données

Université de Montréal/DIRO

Automne 2018

Travail #2

Maxime Daigle

Félix Adam

Mehdi Aqdim

Jonathan Graveline

Question 1

(Voir Question1.ipynb)

Question 2

➢ Trouvez les 1000 paires d'étoiles jumelles les plus proches:

On suppose qu'on veut seulement les étoiles les plus proches deux à deux dans le jeu de données entier.

On suppose également que la densité d'étoile est semblable partout dans le jeu de données.

De calculer la distance entre 6500G d'étoiles est un algorithme quadratique. 6500G ** 2 n'est pas réaliste en temps de calcul même avec 300 serveurs. Une approche plus réaliste serait de partitionner l'espace en 300 régions de dimensions égalent. Il faut également penser à garder une zone critique à la frontière de deux régions dans laquelle les étoiles se retrouvent dans deux serveurs. C'est pour éviter que deux étoiles voisines à la limite de deux régions ne soient pas comparées.

Une fois les régions bien distribuées entre les serveurs il peut encore être nécessaire d'utiliser un algorithme de partition puisqu'il reste encore 22G d'étoiles. « K-d tree » est une bonne option puisque la recherche et l'insertion prennent un temps logarithmique.

Chaque serveur effectue son calcul de distance en parallèle et envoie la liste des 1000 étoiles les plus proches à un serveur qui fera la comparaison finale des étoiles parmi les 300000 candidates (300 serveurs * 1000 étoiles).

➢ Comptez combien d'étoiles il y a dans chaque catégorie:

On suppose qu'avec chaque exemple d'étoile la catégorie est déjà présente dans le jeu de données.

Il faut utiliser quelque chose de semblable à MapReduce avec les données déjà en place sur les serveurs. La procédure « Map » sur chaque serveur devrait aussi compter le nombre d'étoiles de chaque catégorie sur ses propres données. Ensuite puisqu'il n'y a que 10 catégories, un seul serveur peut faire le « Reduce »: le total pour chaque catégorie.

Une boucle aussi simple que de compter le nombre de chaque catégorie sur 22G (6500G d'exemples / 300 serveurs) d'éléments est réaliste en temps de calcul. De plus, avec plusieurs fils d'exécutions ça devrait être rapide.

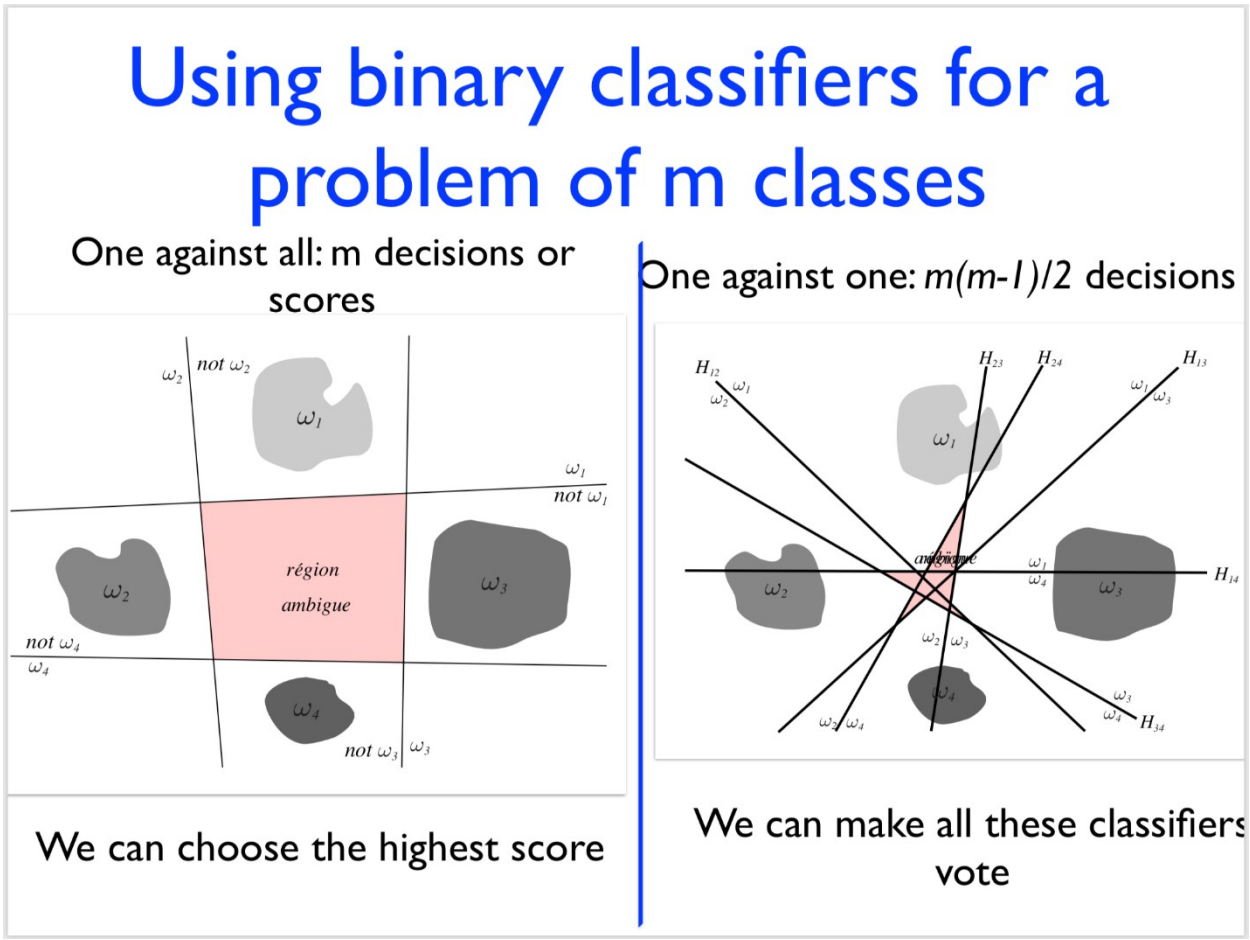
Pour le total ce n'est que 3000 additions (300 serveurs * 10 catégories).

Un logiciel comme Hadoop peut être utilisé pour ce genre de calcul distribué avec MapReduce.

- **Produisez un classifieur qui, étant donné le vecteur de caractéristiques, prédit la catégorie de l'étoile :**

Deux classifieurs possibles pour la classification sont SVM et KNN. SVM est un algorithme d'apprentissage supervisé. Il est donc nécessaire d'avoir un jeu d'entraînement. Une approche possible est d'entrainer un classifieur différent sur chaque serveur en parallèle avec leur propre jeu de données. Ensuite, chaque serveur envoie ses résultats à un serveur qui choisit la classe la plus fréquemment prédite par les 300 serveurs. (Voir question 3 pour la classification de SVM avec plusieurs classes.) Le SVM serait plus approprié pour catégoriser un gros ensemble de données puisque la classification prend moins de temps que l'entrainement. Avec KNN il n'y a pas d'entrainement mais la classification prend plus de temps. KNN serait donc plus approprié si on prévoit classifier peu de nouvelles étoiles. Il est quand même possible avec KNN d'utiliser la même approche distribuée qu'avec SVM (un classifieur par serveur).

- **Question 3**



Source : 10_classifieurs_lineaires_en-3395_with-extra-slides.pdf fait par Pascal Vincent et présenté en IFT3395

Il y a deux façons d’utiliser un classifieur binaire pour réaliser la classification de m classes.

La méthode « one against all » (OAA) ou « one vs rest » consiste à entraîner un classifieur pour chaque classe. L’ensemble de données d’entrainement fournis au classifieur w_i est constitué d’exemples avec une étiquette positive pour les exemples de la classe i et d’exemples avec une étiquette négative pour tous les autres exemples. Ainsi, le classificateur w_i sert à déterminer si l’exemple est dans la classe i ou non. La décision sur un exemple est prise en regardant le score de confiance de chaque w_i (i=0, 1, …, m) sur cet exemple. La classe prédite est la classe correspondant au classifieur donnant le plus grand score.

La méthode one vs one (OvO) consiste à entrainer un classifieur pour chaque pair de classe possible. L’ensemble de données d’entrainement fournis au classifieur w_ij est constitué des exemples de la classe i et des exemples de la classe j. Ainsi, chaque classifieur w_ij apprend à différencier la classe i et j. La décision sur un exemple est prise en prenant la classe prédite le plus souvent à travers tous les classifieurs.

Pour m classes, OOA a besoins de m classifieurs tandis que OvO a besoins de m(m-1)/2. Cela veut dire que pour m = 3, OOA a 3 classifieurs et OvO a 3 classifieurs. Pour m = 25, OOA a 25 classifieurs et OvO a 300 classifieurs. Pour m = 12500, OOA a 12500 classifieurs et OvO a 78 118 750 classifieurs.

- ## Question 4

(Voir Question4.ipynb)