Algorithme des k-moyennes pour la quantification d'images

Guillaume Wisniewski
guillaume.wisniewski@limsi.fr

février 2013

1 Algorithme des k-moyennes

L'objectif de cette partie est d'implémenter l'algorithme des k-moyennes sur un exemple jouet. Dans toute cette partie les exemples sont représentés par des listes de `float` (ce sont des points de $\mathbb{R}^n$). Le corpus sera donc constitué par une liste de listes, par exemple :

```
data = [[2.0, 3.0],  [3.0, 1.0],  [4.0, 2.0], [11.0, 5.0],
        [12.0, 4.0], [12.0, 6.0], [7.0, 5.0], [8.0, 4.0],
        [8.0, 6.0]]
```

1. Donnez le code d'une fonction `choose_initial(data, k)` qui choisit aléatoirement k points *différents* de la liste de points `data`. On pourra utiliser la fonction `choice` de la bibliothèque `random` ; pour supprimer les doublons d'une liste, il suffit de la transformer en `set` avec l'instruction `set(data)`.
2. Écrivez une fonction qui prend en paramètres deux points et retourne la distance euclidienne de ces deux points.
3. Rappelez le principe des k-moyennes. Quelle est la complexité de cet algorithme ?
4. Donnez deux exemples d'applications de cet algorithme.
5. Écrivez une fonction `kmeans` qui renvoie (dans cet ordre) une liste donnant pour chaque point le numéro de la partition à laquelle il appartient et la liste des centres des partitions. Cette méthode prend en paramètres :
   — `data` la liste des points ;

— `k` le nombre de partitions à trouver ;
— `t` un critère d'arrêt dont la signification sera expliquée plus loin.
— `maxiter` le nombre maximales d'itérations de l'algorithme.
On utilisera la distance euclidienne pour mesure la similarité entre deux points.

Lors de chaque itération il est possible de calculer la valeur d'un critère d'erreur. Celui-ci est défini par :

$$\mathcal{E} = \sum_{\mathbf{x} \in \mathcal{D}} d(\mathbf{x}, c(\mathbf{x}))$$

où $\mathcal{D}$ est l'ensemble des exemples, $c(\mathbf{x})$ est le centre de la partition à laquelle est rattaché $\mathbf{x}$ et d la distance utilisée. Cette erreur permet de définir un critère d'arrêt à l'algorithme des k-moyennes : si $\mathcal{E}_i$ est l'erreur à la i^{e} itération, les mises à jour sont arrêtées dès que :

$$|\mathcal{E}_i - \mathcal{E}_{i-1}| \leq t$$

6. Implémentez cette stratégie d'arrêt dans la fonction `kmeans`
7. Quel est le résultat du clustering des points données au début de la section ?
8. Le résultat de la fonction k-moyenne est-il déterministe (sera-t-il identique si on appelle plusieurs fois la méthode avec les mêmes donnés) ?

2 Application à la quantification d'images

La quantification d'image est une méthode permettant de compresser une image en réduisant le nombre de couleurs utilisées. Il suffit, pour cela, de « fusionner » les couleurs qui sont « proches ». La similarité entre couleurs se définit naturellement comme la distance entre les représentations de ces couleurs par des vecteurs indiquant la part des composantes rouge, verte et bleue. Il suffit alors d'utiliser l'algorithme des k-moyennes pour déterminer quelles sont les couleurs qui doivent être fusionnées et vers quelles valeurs elles doivent être fusionnées.

Vous trouverez sur le site du cours un squelette de programme utilisant la méthode `kmeans` définie dans la partie précédente pour réaliser la quantification d'une image.

9. Expliquez comment l'algorithme des k-moyennes peut être utilisé pour réaliser la quantification d'une image.

10. Dans le mode *true color*[1] chaque couleur est codée par un vecteur indiquant les nuances de rouge, vert et bleu et chaque nuance est représentée par un entier compris entre 0 et 255.
    — Combien de couleurs peuvent-être représentées dans le mode *true color* ?
    — Quelle est la taille d'une image de 100×100 pixels dans ce mode ?
    — Quelle est la taille de cette même image si on n'autorise que 32 couleurs différentes ?
11. Appliquer la méthode à une image de votre choix pour différentes valeurs de k. Qu'observez-vous ?
12. Répétez l'expérience précédente en utilisant comme mesure de similarité la fonction suivante[2] :

```
def perceptualColorDistance(color1, color2):
"""return the distance between two given colors,
computed perceptually. Colors must be sequences
of 3 integer values in [0,255]."""
  rmean = (color1[0] - color2[0]) // 2
  dr = color1[0] - color2[0]
  dg = color1[1] - color2[1]
  db = color1[2] - color2[2]
  return (((512+rmean)*dr*dr)>>8) + 4*dg**2 + (((767-rmean)*db**2)>>8)
```

    Que remarquez-vous ?
13. L'évaluation des méthodes non supervisées est toujours problématique. Pour évaluer la qualité de l'approche proposée, on propose de comparer le résultat de celle-ci à une quantification aléatoire dans laquelle les représentants de chaque classe et l'appartenance à chaque classe sont choisis aléatoirement. Implémentez cette méthode.
14. Le programme fournit ne considère qu'un échantillon des points de l'image. Pourquoi ? Que se passe-t-il si on augmente ou diminue cette valeur ?

1. qui permet de représenter toutes les nuances de couleurs qu'un œil peut distinguer
2. Cette mesure de similarité est fondée sur la perception des couleurs, cf. `http://www.compuphase.com/cmetric.htm`