

TD 7 — k -means : décomposition de l'inertie et choix de K

Séance 7

Christophe Ambroise

2026-09-04

Objectifs

Démontrer la décomposition $I_T = I_W + I_B$ et l'identifier au théorème de la variance totale, puis programmer k -means à la main sur Iris : convergence de Lloyd, optima locaux, choix de K , et confrontation avec la décomposition de l'ACP qui partage la **même** inertie totale.

L'exercice 1 se traite **sans ordinateur**. Les exercices 2 à 4 portent sur Iris, quatre mesures en centimètres, **sans la colonne species** — elle ne servira qu'à interpréter, jamais à construire.

Partie théorique

Exercice 1 — La décomposition de l'inertie

Soit n points $x_1, \dots, x_n$ de $\mathbb{R}^p$ et une partition en K groupes $C_1, \dots, C_K$ de tailles $n_1, \dots, n_K$. On note $\hat{\mu}_k$ le centre de gravité du groupe k et $\hat{\mu}$ celui du nuage entier.

L'objectif est d'établir

$$\underbrace{\sum_{i=1}^n \|x_i - \hat{\mu}\|^2}_{I_T} = \underbrace{\sum_k \sum_{i \in C_k} \|x_i - \hat{\mu}_k\|^2}_{I_W} + \underbrace{\sum_k n_k \|\hat{\mu}_k - \hat{\mu}\|^2}_{I_B}.$$

1. Montrer que $I_W = \sum_i \|x_i\|^2 - \sum_k n_k \|\hat{\mu}_k\|^2$. *Indication* : $\sum_{i \in C_k} x_i = n_k \hat{\mu}_k$.
2. Montrer que $I_B = \sum_k n_k \|\hat{\mu}_k\|^2 - n \|\hat{\mu}\|^2$.
3. Conclure.
4. Soit Z la variable aléatoire « numéro du groupe », de loi $P(Z = k) = n_k/n$, et X un point tiré uniformément dans le nuage. Écrire le **théorème de la variance totale** $\text{Var}(X) = \mathbb{E}[\text{Var}(X | Z)] + \text{Var}(\mathbb{E}[X | Z])$ et identifier chaque terme à I_T/n , I_W/n ou I_B/n .
5. En déduire que minimiser I_W équivaut à maximiser I_B . Pourquoi ne peut-on pas choisir K en minimisant I_W ?

Partie empirique

Exercice 2 — Lloyd écrit à la main

```
import numpy as np, pandas as pd

iris = pd.read_csv("../data/iris.csv")
X = iris.drop(columns="species").to_numpy(float)

def matrice_appartenance(etiquettes):
    valeurs = np.unique(etiquettes)
    return (np.asarray(etiquettes)[:, None] == valeurs[None, :]).astype(float)

def distances2(X, C):
    """||x - c||2 = ||x||2 + ||c||2 - 2 x·c, sans aucune boucle."""
    return (np.sum(X**2, axis=1)[:, None]
            + np.sum(C**2, axis=1)[None, :] - 2 * X @ C.T)
```

```
iris_df <- read.csv("../data/iris.csv")
X <- as.matrix(iris_df[, setdiff(names(iris_df), "species")])

matrice_appartenance <- function(etiquettes) {
  f <- factor(etiquettes)
  outer(as.integer(f), seq_along(levels(f)), "==") * 1
}

distances2 <- function(X, C) outer(rowSums(X^2), rowSums(C^2), "+") - 2 * X %*% t(C)
```

1. Vérifier que les centres se calculent en une ligne : $\hat{\mu} = (C^T X) / \text{diag}(C^T C)$. Comparer à `KMeans.cluster_centers_` (respectivement `kmeans()$centers`).
2. Écrire l'algorithme de Lloyd : initialisation par K observations tirées au hasard, puis alternance affectation / recentrage jusqu'à stabilisation. Enregistrer I_W **après chaque demi-étape**.
3. Vérifier numériquement que la suite obtenue est décroissante. Pourquoi l'est-elle nécessairement ? Traiter les deux demi-étapes séparément.
4. Combien d'itérations pour converger ? Comparer la partition obtenue à celle de la bibliothèque.
5. Vérifier sur votre partition que $I_W + I_B = I_T$ et que I_W coïncide avec `inertia_` (respectivement `tot.withinss`).

Exercice 3 — Optima locaux et choix de K

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# une seule initialisation, aléatoire : la variabilité devient visible
valeurs = [KMeans(3, n_init=1, init="random", random_state=g).fit(X).inertia_
            for g in range(30)]

valeurs <- sapply(1:30, function(g) {
  set.seed(g)
  kmeans(X, centers = 3, nstart = 1, algorithm = "Lloyd", iter.max = 100)$tot.withinss
})
```

1. Lancer 30 initialisations aléatoires avec **une seule** relance. Combien d'optima distincts ? Combien de fois le minimum est-il atteint ? De combien la pire solution s'en écarte-t-elle ?
2. Quelle valeur de `n_init` / `nstart` retenir, et pourquoi n'est-ce pas une option de confort ?
3. Tracer I_W pour K de 1 à 10, puis les **différences secondes**. Que suggère le coude ?
4. Calculer la **silhouette** moyenne pour K de 2 à 7. Quel K désigne-t-elle ? Comment arbitrer entre les deux critères ?
5. Standardiser les quatre variables et relancer. Combien de fleurs changent de groupe ? Faut-il standardiser Iris ?

Exercice 4 — Huygens contre Pythagore

L'ACP décompose la **même** inertie totale, mais selon un sous-espace :

$$I_T = \underbrace{\sum_{j \leq q} d_j^2}_{\text{projetée}} + \underbrace{\sum_{j > q} d_j^2}_{\text{résiduelle}},$$

où les d_j sont les valeurs singulières de la matrice **centrée**.

```
from sklearn.decomposition import PCA

d = np.linalg.svd(X - X.mean(axis=0), compute_uv=False)
Z = PCA(n_components=2).fit_transform(X)

acp <- prcomp(X, center = TRUE, scale. = FALSE)
d2 <- acp$sdev^2 * (n - 1)      # prcomp normalise par n - 1
```

1. Vérifier numériquement que $\sum_j d_j^2 = I_T$. Pourquoi ?
2. Donner la part d'inertie projetée sur les deux premiers axes, et la comparer au $R^2 = I_B/I_T$ de k -means en trois groupes. Que compare-t-on exactement ?
3. Relancer k -means **dans le plan factoriel**. La partition change-t-elle ? Mesurer son I_W en dimension 4 et comparer.
4. Ajuster un mélange gaussien à covariances sphériques, puis à covariances libres. Comparer chaque partition à celle de k -means et aux espèces. Qu'est-ce que cela confirme du cours ?
5. Conclure : les deux décompositions sont-elles concurrentes ?

Remarque finale

! À retenir

$I_T = I_W + I_B$ vaut pour **toute** partition : c'est le théorème de la variance totale, avec la partition pour conditionnement. Comme I_T ne dépend pas de la partition, minimiser l'intra revient à maximiser l'inter — mais I_W décroît toujours avec K , si bien que le critère ne peut jamais choisir K tout seul. L'ACP découpe la même inertie selon Pythagore plutôt que selon Huygens.

Les deux scripts exécutables correspondants sont `python/td07_kmeans.py` et `r/td07_kmeans.R`.