

TD 3 — Les k plus proches voisins : distance, choix de k , bruit

Séance 3

Christophe Ambroise

2026-10-05

Objectifs

Démontrer le lemme qui ramène la classification à l'estimation de η — la première brique du théorème de Stone — puis explorer les k -PPV sur Iris : mesurer ce que la distance doit aux unités, choisir k par validation croisée, et constater ce que coûtent des variables inutiles.

Les exercices 2 à 4 portent tous sur Iris : 150 fleurs, quatre mesures en centimètres, trois espèces équilibrées. L'exercice 1 reprend le modèle à deux gaussiennes du TD 1, dont le risque de Bayes vaut $R^* = 0,0990$.

Partie théorique

Exercice 1 — Le lemme de la règle par substitution

Une règle **par substitution** estime d'abord $\eta(x) = P(Y = 1 \mid X = x)$ par un $\hat{\eta}_n$, puis décide

$$g_n(x) = \mathbb{1}_{\{\hat{\eta}_n(x) > 1/2\}}.$$

Les k -PPV en sont un cas : $\hat{\eta}_n(x)$ y est la fraction de voisins de classe 1. On veut montrer que **bien estimer η suffit à bien décider**.

1. Rappeler la règle de Bayes g^* et montrer que, pour une décision quelconque g , l'excès de risque conditionnel s'écrit

$$P(g(X) \neq Y \mid X = x) - P(g^*(X) \neq Y \mid X = x) = |2\eta(x) - 1| \cdot \mathbb{1}_{\{g(x) \neq g^*(x)\}}.$$

2. Montrer que si $g_n(x) \neq g^*(x)$, alors $|\eta(x) - \frac{1}{2}| \leq |\hat{\eta}_n(x) - \eta(x)|$. *Indication : où se trouvent $\eta(x)$ et $\hat{\eta}_n(x)$ par rapport à $\frac{1}{2}$?*
3. En déduire le **lemme** :

$$R(g_n) - R^* \leq 2 \mathbb{E}[|\hat{\eta}_n(X) - \eta(X)|].$$

4. À quoi sert ce lemme dans la preuve du théorème de Stone ? Que resterait-il à démontrer ensuite ?

Partie empirique

Exercice 2 — La distance dépend des unités

Le code de chargement et de calcul du plus proche voisin est fourni.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler

iris = pd.read_csv("../data/iris.csv")
X, y = iris.drop(columns="species"), iris["species"]
Xa = X.to_numpy(float)
Z = StandardScaler().fit_transform(Xa)

def plus_proche_voisin(M):
    """Indice du plus proche voisin de chaque ligne, elle-même exclue."""
    D = ((M[:, None, :] - M[None, :, :]) ** 2).sum(-1)
    np.fill_diagonal(D, np.inf)
    return D.argmax(1)
```

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

plus_proche_voisin <- function(M) {
  D <- as.matrix(dist(M))
  diag(D) <- Inf
  apply(D, 1, which.min)
}
```

1. Donner l'écart-type de chaque variable. Laquelle domine la distance euclidienne, et dans quel rapport ?
2. Combien de fleurs changent d'**espèce de plus proche voisin** entre les données brutes et les données standardisées ? Les identifier.
3. Estimer par validation croisée à dix plis l'erreur du 7-PPV, avec et sans standardisation. Conclure — la standardisation améliore-t-elle le modèle ?
4. Reprendre la question 3 après avoir exprimé la **largeur de sépale en millimètres**. Que devient chaque chiffre ? Reformuler alors ce que la standardisation apporte réellement.
5. Où doivent être estimées $\hat{\mu}_j$ et $\hat{\sigma}_j$? Écrire le code qui garantit cette contrainte.

Exercice 3 — Choisir k par validation croisée

```
from sklearn.model_selection import StratifiedKFold, cross_validate

plis = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
ks = list(range(1, 52, 2))

erreurs = []
for k in ks:
```

```

modele = make_pipeline(StandardScaler(), KNeighborsClassifier(k))
erreurs.append(1 - cross_validate(modele, X, y, cv=plis)["test_score"].mean())

```

```

library(class)

# Plis stratifiés : chaque classe est répartie également entre les K plis.
plis_stratifies <- function(y, K = 10, graine = 0) {
  set.seed(graine)
  pli <- integer(length(y))
  for (cls in levels(y)) {
    idx <- sample(which(y == cls))
    pli[idx] <- rep_len(seq_len(K), length(idx))
  }
  pli
}

erreur_vc_knn <- function(X, y, k = 7, standardise = TRUE, K = 10, graine = 0) {
  pli <- plis_stratifies(y, K, graine)
  mean(sapply(seq_len(K), function(p) {
    app <- X[pli != p, , drop = FALSE]; val <- X[pli == p, , drop = FALSE]
    if (standardise) {
      centre <- colMeans(app); echelle <- apply(app, 2, sd)
      app <- scale(app, centre, echelle); val <- scale(val, centre, echelle)
    }
    mean(knn(app, val, y[pli != p], k = k) != y[pli == p])
  })))
}

KS <- seq(1, 51, by = 2)
erreurs <- sapply(KS, function(k) erreur_vc_knn(X, y, k = k))
k_etoile <- KS[which.min(erreurs)]

```

1. Tracer l'erreur de validation croisée en fonction de k . Quel est k^* ?
2. Ajouter l'erreur d'**apprentissage** sur la même figure. Pourquoi ne peut-elle pas servir à choisir k ? Que vaut-elle en $k = 1$, et pourquoi ?
3. Relancer avec vingt graines différentes. Entre quelles valeurs k^* varie-t-il ? Et l'erreur correspondante ? Qu'en conclure sur la phrase « la valeur optimale est $k^* = 7$ » ?
4. Quel k retiendriez-vous pour un rapport, et comment le justifieriez-vous ?
5. Le théorème de Stone suggère $k_n = \lceil \sqrt{n} \rceil$, soit $k = 13$ pour $n = 150$. Est-ce un argument recevable pour fixer k sans validation ?

Exercice 4 — Variables de bruit et sélection

```

import numpy as np

def ajoute_bruit(X, m, graine=0, reserve=100):
    """m colonnes gaussiennes, prélevées dans une réserve tirée une seule fois.

    Les jeux successifs sont ainsi emboîtés : la dégradation observée ne doit
    rien au changement de tirage.
    """

```

```
bruit = np.random.default_rng(graine).normal(size=(len(X), reserve))
return np.hstack([np.asarray(X), bruit[:, :m]])
```

```
set.seed(0)
reserve <- matrix(rnorm(nrow(X) * 100), nrow = nrow(X)) # tirée une seule fois
ajoute_bruit <- function(X, m) cbind(X, reserve[, seq_len(m), drop = FALSE])
```

1. Ajouter $m \in \{0, 2, 5, 10, 20, 50, 100\}$ variables de bruit gaussien aux quatre variables d'Iris. Réestimer l'erreur du 7-PPV standardisé à chaque étape.
2. Faire la même chose avec LDA. Comparer les deux dégradations et expliquer l'écart à partir de ce que chaque méthode fait de ses variables.
3. Le CM affirme qu'ajouter une variable ne peut pas augmenter l'erreur de **Bayes**. Est-ce contredit ? Quelle quantité augmente ici ?
4. Ajouter une sélection de variables (`SelectKBest`, $k = 4$) **dans** le pipeline. Que devient la courbe ?
5. Faire maintenant la sélection **avant** la validation croisée, sur le jeu complet. Pour rendre l'effet visible, le faire sur 500 variables de bruit pur et **aucune** variable informative. Comparer les deux erreurs et conclure.

Remarque finale

! À retenir

Les k -PPV n'ont ni paramètre ni hypothèse de loi : toute leur modélisation tient dans la distance. D'où leur consistance universelle, et leur sensibilité aux unités et aux variables inutiles.

Les deux scripts exécutables correspondants sont `python/td03_knn.py` et `r/td03_knn.R`. La démonstration complète du théorème de Stone est dans `proofs/stone-1977.md`.