

# M1 info., Ininitiation à la Recherche Success Stories

*Jean-François* COUCHOT  
Université de Franche-Comté



UNIVERSITÉ  
FRANCHE-COMTÉ



# Plan



Intelligence Artificielle

Sécurité

Sciences des données respectueuses de la vie privée



# Plan



## Intelligence Artificielle

De l'entropie de Shannon au gradient boosting en passant par les arbres de décision

## Sécurité

Sciences des données respectueuses de la vie privée



# Plan



## Intelligence Artificielle

De l'entropie de Shannon au gradient boosting en passant par les arbres de décision

## Sécurité

Sciences des données respectueuses de la vie privée



# Arbres de décision

## Exemple<sup>1</sup> de classification

Preg.	Gluc.	BloodP.	SkinThick.	Insul.	BMI	DPF	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
...								

## Un modèle arborescent

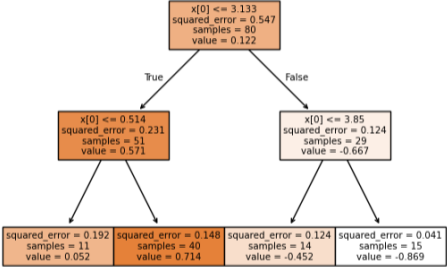
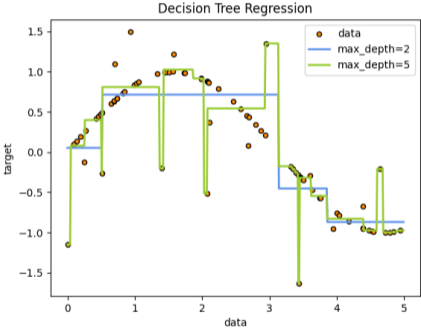
- ▶ Nœud feuille : distribution des valeurs de l'attribut cible
- ▶ Nœud interne : prédicat sur un des attributs entrée
- ▶ Arête sortante d'un nœud : validité ou non du prédicat

- ▶ Connaissant une valeur pour chaque attribut de mesure : prédire Outcome

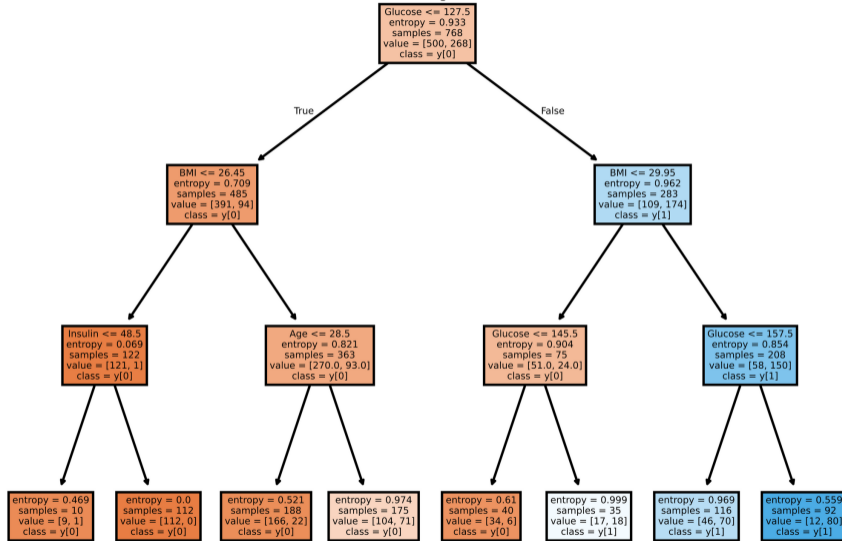


1. <https://www.kaggle.com/uciml/pima-indians-diabetes-database>

# Arbre de régression : prédire une valeur numérique

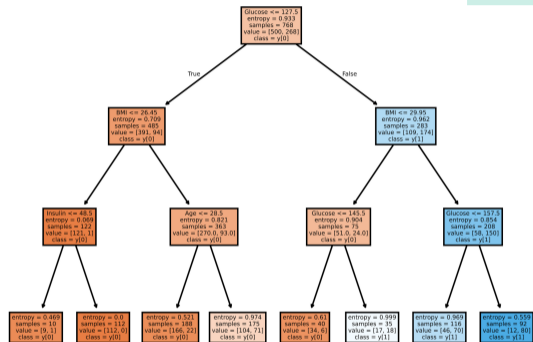


# Les arbre de classification : prédire une classe



# Processus récursif glouton de construction d'un arbres

- ▶ Choix de l'attribut d'entrée  $X_i$  et d'un ensemble  $\delta$  de valeurs qu'il peut prendre réalisant la "meilleure scission du jeu de données"
- ▶ Partition de l'ensemble des données selon  $X_i$  et  $\delta$
- ▶ Jusqu'à un critère de terminaison :
  - ▶ même valeur de la caractéristique-cible pour tous les sous-ensembles
  - ▶ prédiction plus améliorée par une séparation
  - ▶ profondeur maximale atteinte





# Premier arbre de régression<sup>2</sup>

Meilleur découpage : meilleure erreur quadratique moyenne (possible puisque regression)

```
import numpy as np
def choisir_meilleure_partition(dataset)
    b_variable, b_seuil, b_impureté = None, None, np.inf

    for variable in dataset.columns[:-1]: # On exclut la dernière colonne (la cible)
        valeurs_uniques = dataset[variable].unique()
        valeurs_uniques.sort()

        for seuil in valeurs_uniques[:-1]:
            # Division du dataset en deux parties
            grp_gch, grp_dte = dataset[dataset[variable] <= seuil], dataset[dataset[variable] > seuil]

            # Calcul de l'impureté
            impureté_gch = np.mean((grp_gch['target'] - grp_gch['target'].mean())**2)
            impureté_dte = np.mean((grp_dte['target'] - grp_dte['target'].mean())**2)
            impureté_totale = len(grp_gch) * impureté_gch + len(grp_dte) * impureté_dte

            if impureté_totale < b_impureté: # MAJ de la meilleure partition si nécessaire
                b_variable, b_seuil, b_impureté = variable, seuil, impureté_totale

    return b_variable, b_seuil, b_impureté
```

2. Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. Journal of the American statistical association, 58(302), 415-434.

# Classification And Regression Tree (CART)<sup>3</sup>

Meilleur découpage : impureté de Gini la plus réduite

```
import numpy as np

def gini_index(y): #Calcule l'indice de Gini pour un ensemble de labels
    _, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    return np.sum(probabilities * (1 - probabilities))

def best_split(X, y):
    best_gini, best_feature, best_threshold = np.inf, None, None

    for feature in range(X.shape[1]):
        thresholds = np.unique(X[:, feature])
        for threshold in thresholds:
            left_idx, right_idx = X[:, feature] <= threshold, X[:, feature] > threshold

            gini_left = gini_index(y[left_idx])
            gini_right = gini_index(y[right_idx])
            gini = (len(left_idx) * gini_left + len(right_idx) * gini_right) / len(y)

            if gini < best_gini:
                best_gini, best_feature, best_threshold = gini, feature, threshold

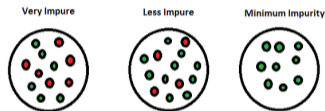
    return best_feature, best_threshold
```

3. Breiman, L. (1984). Classification and regression trees

# Gain d'information exploitant l'entropie<sup>5</sup>



## Illustrations de l'entropie<sup>4</sup>



- ▶  $H(T) = - \sum_i p_i \log_2(p_i)$
- ▶ Très hétérogène :  $H(T_V) = - \left( \frac{6}{12} \log_2\left(\frac{6}{12}\right) + \frac{6}{12} \log_2\left(\frac{6}{12}\right) \right) = 1$
- ▶ Moins hétérogène :  $H(T_L) = - \left( \frac{4}{14} \log_2\left(\frac{4}{14}\right) + \frac{10}{14} \log_2\left(\frac{10}{14}\right) \right) = 0.86$
- ▶ Uniforme :  $H(T_M) = - \left( \frac{9}{9} \log_2\left(\frac{9}{9}\right) \right) = - \log_2(1) = 0$

4. [https://en.wikipedia.org/wiki/Information\\_gain\\_\(decision\\_tree\)](https://en.wikipedia.org/wiki/Information_gain_(decision_tree))

5. Shannon, C. E. (1948). A mathematical theory of communication. The Bell system technical journal, 27(3), 379-423.

# Algorithme ID3<sup>6</sup>

```
import numpy as np

def entropy(y): #Calcule l'entropie d'un ensemble de labels
    _, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    return -np.sum(probabilities * np.log2(probabilities + 1e-9))

def information_gain(parent, left_child, right_child) #Calcule le gain d'information
    weight_left = len(left_child) / len(parent)
    weight_right = len(right_child) / len(parent)
    return entropy(parent) - (weight_left * entropy(left_child) + weight_right * entropy(right_child))

def best_split(X, y):
    best_gain, best_feature, best_threshold = np.inf, None, None

    for feature in range(X.shape[1]):
        thresholds = np.unique(X[:, feature])
        for threshold in thresholds:
            left_idx, right_idx = X[:, feature] <= threshold, X[:, feature] > threshold
            gain = information_gain(y, y[left_idx], y[right_idx])
            if gain > best_gain:
                best_gain, best_feature, best_threshold = gain, feature, threshold

    return best_feature, best_threshold
```

6. Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1, 81-106.

# Arbre de décision dans un code complet

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import cross_val_score, StratifiedShuffleSplit
from sklearn.datasets import load_wine
import matplotlib.pyplot as plt
import numpy as np

data = load_wine()
X, y = data.data , data.target

clf = DecisionTreeClassifier(criterion="entropy",max_depth=5)
cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
scores = cross_val_score(clf, X, y, cv=cv, scoring='accuracy')

mean_score, std_deviation = np.mean(scores), np.std(scores)

print(f"Scores de validation croisée: {scores} et précision moyenne: {mean_score:.4f}")
print(f"Écart type des précisions: {std_deviation:.4f}")

clf = DecisionTreeClassifier(criterion="entropy",max_depth=5)
clf.fit(X, y)

index = 10 # arbitraire
exemple = X[index:index+1]
prediction = clf.predict(exemple)
print("Exemple :", exemple, ", Prédiction :", prediction, "Vraie classe :", y[index])

plt.figure()
plot_tree(clf, filled=True)
plt.show()
```

# scikit-learn<sup>7</sup> : Une success story du machine learning



- ▶ Projet de recherche INRIA
- ▶ Implémentation efficace et unifiée d'une multitude d'algorithmes de machine learning : régression, classification, clustering, réduction de dimensionnalité, etc
- ▶ Facilité d'utilisation : conçue pour être intuitive et cohérente
- ▶ Communauté très active ; support important via des forums, des tutoriels et des projets open-source
- ▶ Adoption industrielle : largement utilisé dans l'industrie pour des applications aussi diverses que la finance, la santé, le marketing ou encore la recherche
- ▶ Mon avis : la bibliothèque Python de référence pour le machine learning.

---

7. <https://scikit-learn.org/stable/>

# Pour aller plus loin : le gradient boosting<sup>9</sup>

1. Entraînement d'un modèle simple (souvent un CART) sur les données d'entraînement
2. Calcul des erreurs commises par ce premier modèle
3. Entraînement d'un deuxième arbre qui doit prédire les erreurs du premier modèle
4. Combinaison des prédictions des deux arbres pour obtenir une prédiction plus précise
5. Répète les étapes 2 à 4 plusieurs fois, en ajoutant à chaque itération un nouvel arbre qui corrige les erreurs des arbres précédents

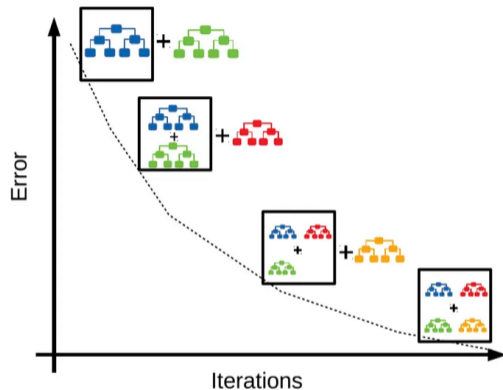


Figure – Illustration de G. Boosting<sup>8</sup>

8. <https://medium.com/@hemashreekilari9/understanding-gradient-boosting-632939b98764>

9. Friedman, J. H. (2002). Stochastic gradient boosting. Computational statistics & data analysis, 38(4), 367-378.

# Plan



Intelligence Artificielle

Sécurité

Les codes détecteurs et correcteurs d'erreur : de Shannon à la 5G

Sciences des données respectueuses de la vie privée





# Plan



Intelligence Artificielle

Sécurité

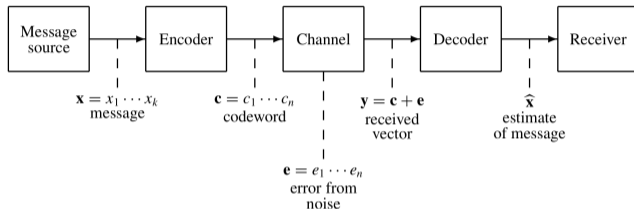
Les codes détecteurs et correcteurs d'erreur : de Shannon à la 5G

Sciences des données respectueuses de la vie privée



# Communication bruitée

## Schéma théorique de communication avec correction<sup>10</sup>



1. A la source : un vecteur  $x$  doit être envoyé
  - ▶ Si  $x$  était transmis tel quel : n'importe quel bruit ajouté rendrait le message irrécupérable
  - ▶ Idée principale : ajouter de la redondance au message
2. Mot de code  $c$  envoyé sur le canal : vecteur contenant  $x$  et redondance
3. Erreur  $e$  : modélisée par un vecteur de bruit ajouté au mot de code
4. Estimation  $\hat{x}$  de  $x$  : construite à partir du vecteur reçu  $y$

10. Shannon, C. E. (1948). A mathematical theory of communication. The Bell system technical journal, 27(3), 379-423.

# Des erreurs partout



## Taux d'erreur

Le taux d'erreur (BER pour Bit Error Ratio) est le rapport entre le nombre de bits erronés reçus par rapport au nombre total de bits transmis.

## Ordres de grandeur<sup>11</sup> de BER

- ▶ Disques optique :  $10^{-5}$
- ▶ Ligne téléphonique :  $10^{-6}$
- ▶ Communication par fibres optiques :  $10^{-9}$

---

11. <https://ljk.imag.fr/membres/Jean-Guillaume.Dumas/Publications/IntroductionAuxCodesCorrecteurs.pdf>



# Détecter, corriger : des exemples de codes

## Au temps des transmissions analogiques<sup>12</sup> de mauvaise qualité

Remplacer chaque lettre par un mot dont la première lettre coïncide avec la lettre à épeler. Sens de « Écho, Roméo, Roméo, Écho, Uniforme, Roméo » ?

### Code détecteur : bit de parité

- ▶ Codage d'un caractère : nombre décimal entre 0 et 127 ( ? bits)
- ▶ Ajout d'un 8<sup>ème</sup> bit de parité : la somme des 8 bits est paire
- ▶ Corriger 1 erreur ? Détecter 2 erreurs ?

Car.	Dec.	Bin.	Bin+Parité
B	65	1000001	10000010
C	66	1000010	10000100
:		:	
a	97	1100001	11000011

### Code correcteur : duplication

- ▶ Répéter la totalité du message suffisamment ( $\geq 3\times$ ) de fois (penser à « ââgmeee »)
- ▶ 011010  $\rightarrow$  000.111.111.000.111.000
- ▶ Si au plus une erreur, message original associé à 100.111.110.001.110.000 ?

12. Rousseau, C., & Saint-Aubin, Y. (2009). Mathématiques et technologie. Springer Science & Business Media.

# Efficacité de la détection/correction ?



## Définition du rendement $\rho$

$\rho$  : rapport  $\frac{\text{nombre de bits de message}}{\text{nombre de bits transmis}}$  toujours inférieur ou égal à 1.

## Rendements des codes précédents

- ▶ Parité :  $\rho_{\text{parite}} = 7/8 = 87.5\%$ , mais détection seulement
- ▶ Duplication :  $\rho_{\text{dupl}_3} = 1/3 \approx 33.3\%$ , avec correction



# Autres exemples

Formule de Luhn<sup>13</sup> pour les CB, les SIRET

1. Doubler un chiffre sur deux mod. 9, depuis l'avant dernier, de dr. à gche.
2. Somme de tous les chiffres multiple de 10  $\rightsquigarrow$  validité du nombre original.
3. Exemples : validité de "972-487-086" et "927-487-086" ?
4. Rendement  $\rho_{\text{Luhn}_{\text{CB}}} = 15/16 \approx 93.8\%$ , mais détection uniquement

Numéro de SS<sup>14</sup>

1. NIR : numéro de 13 chiffres (Genre, deux derniers chiffres de l'année de naissance, mois de naiss...)
2. Clef sur 2 chiffres : NIR mod. 97 (après gestion ev. des corses)
3. Rendement  $\rho_{\text{NSS}} = 13/15 \approx 86.7\%$ , mais détection uniquement

En vrac

Dans les QR codes<sup>15</sup>, dans la DRAM<sup>16</sup>, dans les OS,...

13. [https://fr.wikipedia.org/wiki/Formule\\_de\\_Luhn](https://fr.wikipedia.org/wiki/Formule_de_Luhn)

14. [https://fr.wikipedia.org/wiki/Num%C3%A9ro\\_de\\_s%C3%A9curit%C3%A9\\_sociale\\_en\\_France#ancrage\\_C](https://fr.wikipedia.org/wiki/Num%C3%A9ro_de_s%C3%A9curit%C3%A9_sociale_en_France#ancrage_C)

15. [https://fr.wikipedia.org/wiki/Code\\_QR](https://fr.wikipedia.org/wiki/Code_QR)

16. [https://en.wikipedia.org/wiki/ECC\\_memory](https://en.wikipedia.org/wiki/ECC_memory)

# Codes systématiques de Hamming<sup>17</sup> ( $2^k - 1, 2^k - k - 1$ )



## Remarques

- ▶  $2^k - 1$  : taille du mot de code
- ▶  $2^k - k - 1$  : taille du mot  $u$  à transmettre
- ▶ Ne corrige au plus qu'une seule erreur par mot de  $2^k - 1$  bits
- ▶ Code systématique : « le mot  $u$  est complété avec des bits de redondance »

## Points clés

- ▶ Deux matrices  $G_k$  et  $H_k$  à valeur dans  $\mathbb{B}$  :
  - ▶  $G_k$  : pour générer le mot de code  $v$  à partir du mot à transmettre  $u$
  - ▶  $H_k$  : pour contrôler  $w$  et le corriger éventuellement
- ▶ Le tout avec des produits matriciels

---

17. Hamming RW. Error detecting and error correcting codes. Bell Syst Tech J. 1950 ;29(2) :147-60.

# Matrices de contrôle $H_k$

## Définition

$$H_k = \begin{pmatrix} P_k \\ I_k \end{pmatrix} \quad (1)$$

- ▶  $I_k$  : matrice identité  $k \times k$
- ▶  $P_k$  : en ligne, tous les vecteurs non nuls de  $\mathbb{B}^k$  qui ne sont pas dans  $I_k$
- ▶  $\rightsquigarrow H_k$  : contient tous les vecteurs non nuls de  $\mathbb{B}^k$ .
- ▶  $\rightsquigarrow H_k$  :  $2^k - 1$  lignes et  $k$  colonnes

## Exemple avec $H_3$

$$H_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{ou bien } H'_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{ou encore } H''_3 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ \hline 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \dots$$



# Matrices de génération $G_k$

## Définition

$$G_k = (I_{2^k - k - 1} \mid P_k) \quad (2)$$

- ▶  $P_k$  : même matrice que dans  $H_k$
- ▶  $I_{2^k - k - 1}$  : matrice identité  $(2^k - k - 1) \times (2^k - k - 1)$
- ▶  $\rightsquigarrow G_k$  :  $2^k - k - 1$  lignes et  $2^k - 1$  colonnes

## Exemple avec $G_3$

$$\text{si } H_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ alors } G_3 = \left( \begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right)$$

# Mot de code et syndrome

Mot de code pour  $(u_1, \dots, u_{2^k-k-1}) \in \mathbb{B}^{2^k-k-1}$

- ▶ Mot de code associé :  $v = uG_k \in \mathbb{B}^{2^k-1}$
- ▶  $G_k = (I_{2^k-k-1} \mid P_k)$  : les  $2^k - k - 1$  premiers bits de  $v$  sont ceux de  $u$
- ▶ Rendement  $\rho_{H_k} = (2^k - k - 1)/(2^k - 1)$  : croissant pour  $k > 3$

Syndrome de  $w \in \mathbb{B}^{2^k-1}$

- ▶ Soit  $w = (w_1, \dots, w_{2^k-1})$  le mot à décoder.
- ▶ Le mot  $\sigma(w) = wH_k \in \mathbb{B}^k$  : appelé syndrome de  $w$

Exemple avec  $u = (1, 0, 1, 1)$

$$G = \left( \begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 & \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & \end{array} \right), H = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

- ▶  $v = uG = (1, 0, 1, 1, 0, 1, 0)$
- ▶  $\sigma(v) = vH = (0, 0, 0)$
- ▶ si  $w = (1, 0, 0, 1, 0, 1, 0)$
- ▶  $\sigma(w) = wH = (0, 1, 1)$

# Détection puis correction d'une erreur



## Mot de code et Syndrome nul

Pour un code de Hamming( $2^k - 1, 2^k - k - 1$ ), un mot  $w$  est un mot de code si et seulement si son syndrome  $\sigma(w) = wH$  est nul

## Correction lorsque le syndrome $\sigma(w)$ n'est pas nul

Soit un code de Hamming( $2^k - 1, 2^k - k - 1$ ) et  $w$  tel que  $\sigma(w)$  n'est pas nul :

- ▶ il existe une ligne  $i$  de  $H$  qui lui est égale et
- ▶ le vecteur  $w' = (w_1, \dots, w_{i-1}, \overline{w_i}, \dots, w_{2^k-1})$  est un mot de code.



# Retour à l'exemple avec $u = (1, 0, 1, 1)$

## Rappels

$$G = \left( \begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 & \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & \end{array} \right), H = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- ▶  $v = uG = (1, 0, 1, 1, 0, 1, 0)$
- ▶  $\sigma(v) = vH = (0, 0, 0)$
- ▶ si  $w = (1, 0, 0, 1, 0, 1, 0)$
- ▶  $\sigma(w) = wH = (0, 1, 1)$

## Détection de l'erreur et correction

- ▶  $v = (1, 0, 1, 1, 0, 1, 0)$  et  $\sigma(v) = (0, 0, 0)$  :  $v$  est un mot de code et le mot original était donc  $(1, 0, 1, 1)$
- ▶  $w = (1, 0, 0, 1, 0, 1, 0)$  et  $\sigma(w) = wH = (0, 1, 1) = H_3$  : le mot de code associé à  $w$  est  $w' = (1, 0, 1, 1, 0, 1, 0)$  ; le mot original était donc  $(1, 0, 1, 1)$

# Les codes de Reed-Solomon<sup>18</sup> $RS(n, k, m)$



## Paramètres

- ▶  $m, n, k, t$  naturels contraints par  $n = 2^m - 1 = k + 2t$
- ▶  $m$  : nombre de bits utilisés par symbole (pour des caractères,  $m = 8$  p.ex)
- ▶  $k$  : nombre de symboles dans le message original
- ▶  $2t$  : nombre de symboles de redondance
- ▶  $n$  : nombre de symboles transmis

## Erreurs prises en compte

- ▶ 2 types d'erreurs : changement de données et perte de données
- ▶ Majoration du nombre d'erreurs :  $2E + S \leq n - k$  avec  $E$  le nombre de changements et  $S$  le nombre de suppressions
- ▶ Si pas de suppression : au plus  $t$  changements

---

18. Reed, I. S., & Solomon, G. (1960). Polynomial codes over certain finite fields. Journal of the society for industrial and applied mathematics, 8(2), 300-304.

# Calculs sur des polynômes dans le corps finis $\mathbb{F}_{2^m}$

## Polynômes

Tous les calculs vont se faire sur des polynômes

$$C(x) = c_0 + c_1x^1 + c_2x^2 + \dots + c_{k+2t}x^{k+2t}$$

de degré  $k + 2t$  où les coefficients sont des entiers  $0 \leq c_i \leq 2^m$  de  $\mathbb{F}_{2^m}$ .

## Le corps finis $\mathbb{F}_{2^m}$

- ▶ Ensemble  $\{0, \dots, 2^m - 1\}$
- ▶ Muni d'une somme (+) et d'un produit ( $\times$ ) aux "bonnes" propriétés
- ▶ Avec un élément générateur  $\alpha : \mathbb{F}_{2^m} \setminus \{0\} = \{\alpha, \alpha^2, \dots, \alpha^{2^m-1}\}$

- ▶ Exemple avec  $\mathbb{F}_{2^2}$

$\times$	0	1	2	3	$+$	0	1	2	3
0	0	0	0	0	0	0	1	2	3
1	0	1	2	3	1	1	0	3	2
2	0	2	3	1	2	2	3	0	1
3	0	3	1	2	3	3	2	1	0

- ▶ de générateur  $\alpha = 2 : \{1, 2, 3\} = \{2, 2^2, 2^3\}$

# Construction du mot de code

## Polynômes à construire

- ▶ Mot  $u = (u_1, \dots, u_k)$ ,  $u_i \in \mathbb{F}_{2^m}$  : réécrit en  $A(x) = u_1x^{k-1} + u_2x^{k-2} + \dots + u_{k-1}x^1 + u_k$
- ▶ Pour  $\alpha$  générateur de  $\mathbb{F}_{2^m}$  : polynôme générateur  $G(x) = (x + \alpha)(x + \alpha^2) \dots (x + \alpha^{2^t})$
- ▶ Polynôme de contrôle :  $B(x) \equiv A(x) \times x^{2t} \pmod{G(x)}$
- ▶ Mot de code  $C(x) = A(x) \times x^{2t} + B(x)$

## Exemple avec mot "3" à transmettre

- ▶ Si  $m = 2$  bits suffisent, alors pour un mot de longueur  $k = 1$ ,  $t = 1$  erreur possible
- ▶  $A(x) = 3$
- ▶  $G(x) = (x + 2)(x + 2^2) = (x + 2)(x + 3) = x^2 + x + 1$

▶  $B(x) \equiv 3 \cdot x^2 \pmod{G(x)} = 3x + 3 \pmod{G(x)}$  car

$$\begin{array}{r|l} 3x^2 & x^2 + x + 1 \\ 3x^2 + 3x + 3 & \hline 3x + 3 & \end{array}$$

- ▶ Mot de code  $C(x) = 3x^2 + 3x + 3$

# Détection d'erreur

## A réception du polynome $D(x)$

- ▶ Calcul des syndromes  $S_j = D(\alpha^j)$  pour chaque  $j$ ,  $1 \leq j \leq 2t$ .
- ▶ Si  $S_j = 0$  pour  $1 \leq j \leq 2t$  : pas d'erreur (preuve admise) et  $A$  est les  $k$  coefs de degré le plus élevé de  $D(x)$
- ▶ Sinon : il y a une erreur à corriger

## Exemple avec une réception de $D(x) = x^2 + 3x + 3$ .

- ▶ Pour  $\alpha = 2$  : calcul de  $D(\alpha^j)$

$j$	$\alpha^j$	$D(\alpha^j)$	$S_j$
1	2	$2 \times 2 + 3 \times 2 + 3 = 3 + 1 + 3 = 1 =$	$S_1$
2	3	$3 \times 3 + 3 \times 3 + 3 = 2 + 2 + 3 = 3 =$	$S_2$

- ▶  $S_1$  et  $S_2$  ne sont pas nuls : présence d'une erreur



# Recherche des erreurs



Formalisation du polynome des erreurs  $E(x)$  avec  $C(x) = D(x) + E(x)$

- ▶ Recherché :  $E(x) = \sum_{r=1}^{\nu} e_{i_r} x^{i_r}$
- ▶  $E(x)$  : contient  $\nu$  coefficients non nuls (nombre d'erreurs) inférieur ou égal à  $t$
- ▶  $i_r$  : les indices (entre 0 et  $n - 1$ ) où il y a une erreur
- ▶  $e_{i_r}$  : la valeur pour corriger l'erreur
- ▶ Nombre d'erreurs : la plus grande valeur  $\nu$  t.q. le système suivant a une solution (admis) :

$$\begin{aligned} S_{\nu} \lambda_1 + S_{\nu-1} \lambda_2 + \cdots + S_1 \lambda_{\nu} &= S_{\nu+1} \\ S_{\nu+1} \lambda_1 + S_{\nu} \lambda_2 + \cdots + S_2 \lambda_{\nu} &= S_{\nu+2} \\ &\vdots \\ S_{2\nu-1} \lambda_1 + S_{2\nu-2} \lambda_2 + \cdots + S_{\nu} \lambda_{\nu} &= S_{2\nu} \end{aligned}$$

Poursuite de l'exemple

Le système  $\begin{matrix} S_1 \lambda_1 & = & S_2 \\ 1 \lambda_1 & = & 3 \end{matrix}$  admet une solution ( $\lambda_1 = 3$ ). Il y a donc une erreur.

# Localisation de l'erreur

## Méthode

- ▶ De  $\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2 + \dots + \lambda_\nu x^\nu$  : recherche des racines des racines  $r_1, \dots, r_\nu$
- ▶ Pour chaque racine  $r$  : il existe (admis)  $i_r$  tq  $r \cdot \alpha^{i_r} = 1$
- ▶ Les erreurs sont les coefficients de  $x^{i_r}$ ,  $i_r \in \{i_1, \dots, i_\nu\}$

## Poursuite de l'exemple

- ▶ On déduit d'abord que  $\Lambda(x) = 1 + 3x$
- ▶ Dans  $\{0, 1, 2, 3\}$ , l'unique racine est  $r = 2$
- ▶ A résoudre :

$$\begin{aligned}r \times \alpha^{i_r} &= 1 \\2 \times 2^{i_1} &= 1 (r = 2 \text{ et } \alpha = 2) \\2^{i_1} &= 3 (\text{mult. à gche. et à drte. par } 3) \\i_1 &= 2\end{aligned}$$

- ▶ C'est le coefficient de  $x^2$  qui est erroné

# Reconstruction du mot de code

## Méthode

- ▶  $C(x) = D(x) + E(x)$  :  $E(x) = D(x) + C(x)$  et donc pour chaque  $j$ ,  $1 \leq j \leq 2t$  :  
 $E(\alpha^j) = D(\alpha^j) + C(\alpha^j) = D(\alpha^j) + 0 = S_j = \sum_{r=1}^{\nu} e_{i_r} \alpha^{j \cdot i_r}$

$$e_{i_1} \alpha^{1 \cdot i_1} + \dots + e_{i_\nu} \alpha^{1 \cdot i_\nu} = S_1$$

$$e_{i_1} \alpha^{2 \cdot i_1} + \dots + e_{i_\nu} \alpha^{2 \cdot i_\nu} = S_2$$

- ▶ Il suffit de résoudre le système

⋮

$$e_{i_1} \alpha^{2t \cdot i_1} + \dots + e_{i_\nu} \alpha^{2t \cdot i_\nu} = S_{2t}$$

- ▶ On construit  $E(x) = \sum_{r=1}^{\nu} e_{i_r} x^{i_r}$ , puis  $C(x) = D(x) + E(x)$

## Poursuite de l'exemple

- ▶ A résoudre 
$$\begin{array}{l} e_2 \times \alpha^2 = S_1 \\ e_2 \times \alpha^{2 \times 2} = S_2 \end{array} \quad \text{c.-à-d.} \quad \begin{array}{l} e_2 \times 3 = 1 \\ e_2 \times 2 = 3 \end{array} \quad \text{de solution } e_2 = 2$$

- ▶  $E(x) = 2x^2$  et finalement  $C(x) = x^2 + 3x + 3 + 2x^2 = 3x^2 + 3x + 3$

# Pour aller plus loin



- ▶ les codes convolutifs<sup>19</sup>
- ▶ les turbo-codes<sup>20</sup>
  - ▶ standards 3G et 4G
  - ▶ la NASA lors de Mars reconnaissance Orbiter (1999 ... 2017)
  - ▶ la norme IEEE 802.16 (WiMAX)
- ▶ les codes de parité à faible densité (LDPC)<sup>21</sup> :
  - ▶ standard 5G
  - ▶ télévision numérique terrestre
  - ▶ Norme Wifi 802.11
  - ▶ disques SSD

---

19. Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE transactions on Information Theory, 13(2), 260-269.

20. Berrou, C., Glavieux, A., & Thitimajshima, P. (1993, May). Near Shannon limit error-correcting coding and decoding : Turbo-codes. 1. In Proceedings of ICC'93-IEEE International Conference on Communications (Vol. 2, pp. 1064-1070). IEEE.

21. [https://en.wikipedia.org/wiki/Low-density\\_parity-check\\_code](https://en.wikipedia.org/wiki/Low-density_parity-check_code)

# Plan



Intelligence Artificielle

Sécurité

Sciences des données respectueuses de la vie privée

Bruiter et exploiter des données : de Warner aux GAFAM



# Plan



Intelligence Artificielle

Sécurité

Sciences des données respectueuses de la vie privée

Bruiter et exploiter des données : de Warner aux GAFAM



# Motivation ; données embarrassante à nettoyer<sup>23</sup>

Table avec 1 seul attribut binaire :  $Q_1$ ="avez-vous triché au moins une fois?"

- ▶ Embarras : tentation pour un étudiant de ne pas répondre honnêtement.

Bruiter selon Warner<sup>22</sup>

- ▶ Chaque étudiant·e lance 2 fois une pièce de monnaie {Pile, Face} sans montrer les 2 résultats successifs  $t_1$  et  $t_2$ .
- ▶ Ajout de la question  $Q_2$  : « Est-ce que  $t_2$  est égal à Pile ? ».
  - ▶ Si  $t_1$  vaut Pile, l'étudiant·e répond honnêtement à la question  $Q_1$ .
  - ▶ Sinon ( $t_1 = \text{Face}$ ), l'étudiant·e répond honnêtement à la question  $Q_2$ .

Analyse de l'extension

- ▶ Réponse partiellement aléatoire : on ne sait pas si une réponse OUI d'un·e étudiant·e provient d'une tricherie ou d'un Pile au second tirage.
- ▶ Honnêteté de l'étudiant·e renforcée : c'est lui·elle qui modifie ses données.

---

22. Warner, S. L. (1965). Randomized response : A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association, 60(309), 63-69.

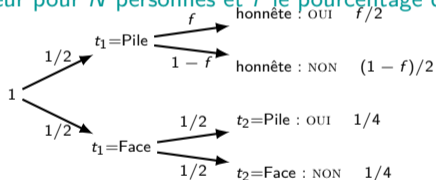
23. <https://fr.coursera.org/lecture/stanford-statistics/warners-randomized-response-model-ck65q>

# Motivation : estimation du pourcentage de tricheurs

## Point clef

- ▶ Un OUI individuel : on ne connaît pas exactement son origine.
- ▶ Après calcul du pourcentage global des OUI : on doit pouvoir estimer le pourcentage des étudiants ayant triché au moins une fois.

## Estimateur pour $N$ personnes et $f$ le pourcentage de tricheur·euse·s



		y	
		OUI	NON
x	OUI	3/4	1/4
	NON	1/4	3/4

- ▶ Pourcentage de OUI observés :  $r \approx 1/4 + f/2$
- ▶ Estimation  $\hat{f}$  du nbre. original de OUI :  $\hat{f} = 2r - 1/2$
- ▶ Xpl. : #OUI= , #NON=  $\rightsquigarrow$   
 $\hat{f} =$  %.

$$\begin{aligned} \frac{\Pr[\mathcal{M}(x_1)=y]}{\Pr[\mathcal{M}(x_2)=y]} &\leq \frac{\Pr[\mathcal{M}(\text{OUI})=\text{OUI}]}{\Pr[\mathcal{M}(\text{OUI})=\text{OUI}]} \\ &\leq 3 \end{aligned}$$



# Idée principale : gestion du bruit

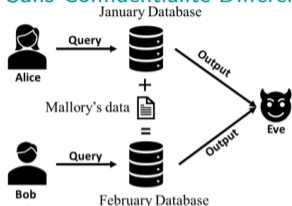


- ▶ Intuition : on peut ajouter du bruit ppur protéger si on sait l'enlever dans l'analyse
- ▶ Questions :
  - ▶ Quel type de bruit ajouter ? Quelle quantité ?
  - ▶ Quelle garanties dispose-t-on pour tel ou tel bruit ?
  - ▶ Quels sont les conséquences sur l'utilité ?



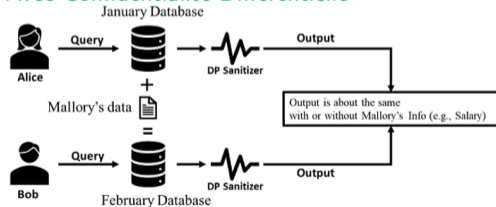
# Ex. de requêtes sur des bases voisines<sup>24</sup>

## Sans Confidentialité Différentielle



- ▶ Requête mensuelle : (nb empl., salaire moyen).
- ▶ Res. :  
{jan. : (100, \$55000), fev. : (101, \$56000)}.
- ▶ Connaiss. suppl. : 0 sortie + Mallory en fev..
- ▶  $\rightsquigarrow$  salaire de Mallory : \$156000.

## Avec Confidentialité Différentielle



- ▶ Même requêtes, mêmes conaiss. suppl..
- ▶ Res. nettoyés :  
{jan : (102, \$55551), fev : (97, \$55975)}.
- ▶ Salaire de Mallory ?

24. Privacy-Preserving Machine Learning. Manning Early Access Program Publications, 2021.



## Intuition pour 2 bases $D_1$ et $D_2$ voisines l'une de l'autre

- ▶ Résultats (agrégés, statistiques, ...) proches.
- ▶  $\Leftrightarrow$  Probabilités sur  $\mathcal{M}(D_1)$  et  $\mathcal{M}(D_2)$  égales (à  $\epsilon$  près).

## Pourquoi une confidentialité différentielle ?

- ▶ Les données privées : souhait qu'elles affectent peu les résultats.
- ▶  $\rightsquigarrow$  Difficile de distinguer si une personne particulière participe ou non.
- ▶  $\rightsquigarrow$  Propriétaire des données : moins inquiet·e de partager ses données.



## Définition ( $\epsilon$ -confidentialité différentielle (DP))

Soit  $\epsilon \in \mathbb{R}^+$ . L'algorithme probabiliste non déterministe  $\mathcal{M}$  respecte la  $\epsilon$ -confidentialité différentielle si

$$\begin{aligned} \forall D_1, D_2 \in \mathbb{N}^{|\mathcal{X}|} \text{ t.q. } \|D_1 - D_2\|_1 = 1, & \quad (D_1 \text{ et } D_2 \text{ voisines}) \\ \forall R \text{ t.q. } R \subseteq \mathcal{M}(\mathbb{N}^{|\mathcal{X}|}), & \quad (\text{pour tte image de l'algo.}) \\ \Pr[\mathcal{M}(D_1) \in R] \leq e^\epsilon \Pr[\mathcal{M}(D_2) \in R] & \quad (\text{si } \epsilon \text{ petit, } e^\epsilon \approx 1 + \epsilon) \end{aligned}$$

## Budget de fuite $\epsilon \in \mathbb{R}^+$ : déviation permise, fuite autorisée

- ▶  $\Pr[\mathcal{M}(D_1) \in R] \leq e^\epsilon \Pr[\mathcal{M}(D_2) \in R]$  : résultats approximativement égaux (mais pas nécessairement) avec/sans la donnée d'1 personne.
- ▶  $\epsilon = 0$  : aucune déviation permise (sorties toutes égales avec/sans la donnée d'1 personne), données parfaitement protégées (mais inutiles).
- ▶  $\epsilon$  petit... grand : tout dépend de la fuite qu'on s'autorise

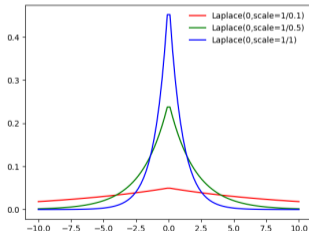
25. Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006, March). Calibrating noise to sensitivity in private data analysis. In Theory of cryptography conference (pp. 265-284). Springer, Berlin, Heidelberg.

# Requête $Q_1$ : nombre d'employé·e·s dans la base

## Objectifs, données, idée

- ▶ Publier un nombre d'employé·e·s avec un mécanisme  $\epsilon$ -DP
- ▶  $Q_1(D_{\text{jan}}) = 100$ ,  $Q_1(D_{\text{fev}}) = 101$ . ...
- ▶ Ajouter un bruit centré en 0 dépendant d' $\epsilon$  :

MEO : bruit laplacien centré en 0,  $\mathcal{M}_L(D) = Q_1(D) + v$ ,  $v \sim \text{Lap}(0, \epsilon^{-1})$

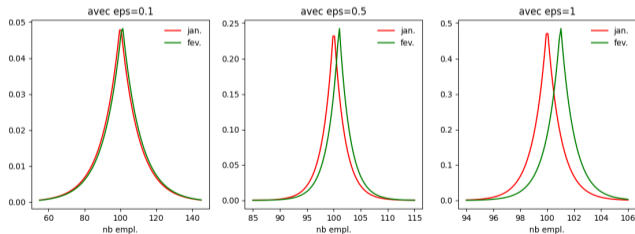


# Requête $Q_1$ : nombre d'employé·e·s dans la base

## Objectifs, données, idée

- ▶ Publier un nombre d'employé·e·s avec un mécanisme  $\epsilon$ -DP
- ▶  $Q_1(D_{\text{jan}}) = 100$ ,  $Q_1(D_{\text{fev}}) = 101$ . ...
- ▶ Ajouter un bruit centré en 0 dépendant d' $\epsilon$  :

MEO : bruit laplacien centré en 0,  $\mathcal{M}_L(D) = Q_1(D) + v$ ,  $v \sim \text{Lap}(0, \epsilon^{-1})$





## Définition (Mécanisme de réponses randomisées généralisées)

- ▶ Domaine :  $\{v_1, \dots, v_k\}$

- ▶  $\Pr[\mathcal{M}_{GRR}(x) = v_i] = \begin{cases} p = \frac{e^\epsilon}{k - 1 + e^\epsilon} & \text{pour } v_i = x \\ q = \frac{1}{k - 1 + e^\epsilon} & \text{sinon} \end{cases}$

## Estimation de la fréquence d'apparition de $v_i$

- ▶  $N$  personnes,  $f_i$  (resp.  $r_i$ ) la fréquence d'apparition initiale de la valeur  $v_i$  (resp. après application de  $\mathcal{M}_{GRR}$  à chaque réponse indiv.)

- ▶ Estimateur non biaisé de  $f_i$ ,  $\hat{f}_i = \frac{r_i - Nq}{N(p - q)}$

$$\text{Var}[\hat{f}_i] = \frac{q(1 - q)}{N(p - q)^2} + \frac{f_i(1 - p - q)}{N^2(p - q)}$$

26. Kairouz, P., Bonawitz, K., & Ramage, D. (2016). Discrete distribution estimation under local privacy. arXiv preprint arXiv :1602.07387.

# Analyses multidimensionnelles respectueuses : exemples dans les GAFAM

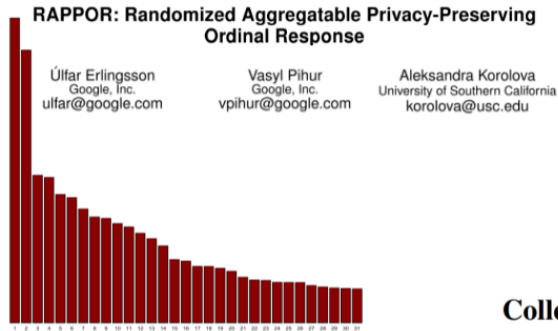


Figure 6: Relative frequencies of the top 31 unexpected Chrome homepage domains found by analyzing ~14 million RAPPOR reports, excluding expected domains (the homepage “google.com”, etc.).



The Count Mean Sketch technique allows Apple to determine the most popular emoji to help design better ways to find and use our favorite emoji. The top emoji for US English speakers contained some surprising favorites.

## Collecting Telemetry Data Privately

**Bolin Ding, Janardhan Kulkarni, Sergey Yekhanin**  
Microsoft Research  
{bolind, jakul, yekhanin}@microsoft.com

Windows Insiders in Windows 10 Fall Creators Update to protect users' privacy while collecting application usage statistics.