femto-st SCIENCES & TECHNOLOGIES

L2 Info., Introduction à la Recherche
2 Success Stories

Jean-François COUCHOT, *Alain* GIORGETTI
Université de Franche-Comté

cnrs · Université Marie & Louis Pasteur · Sup Micro Tech ENSMM · utbm université de technologie Belfort-Montbéliard

Plan

Intelligence Artificielle

Sécurité

Plan

Intelligence Artificielle

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

Sécurité

Plan

Intelligence Artificielle

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

Sécurité

Arbres de décision

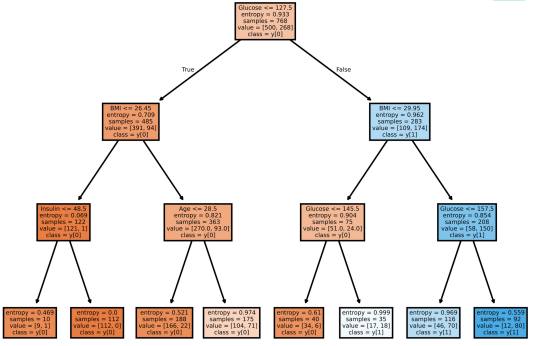
Exemple [1] de classification

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

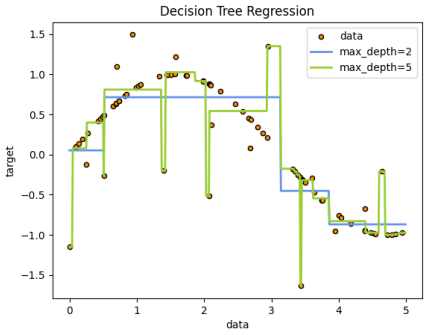
- Nœud feuille : distribution des valeurs de l'attribut cible
- Nœud interne : prédicat sur un des attributs entrée
- Arête sortante d'un nœud : validité ou non du prédicat

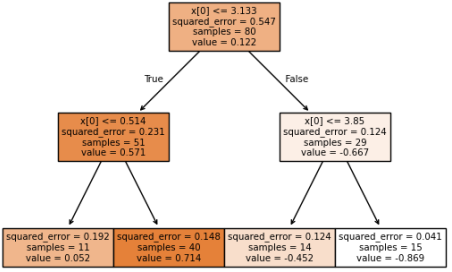
- Connaissant une valeur pour chaque attribut de mesure : predire Outcome



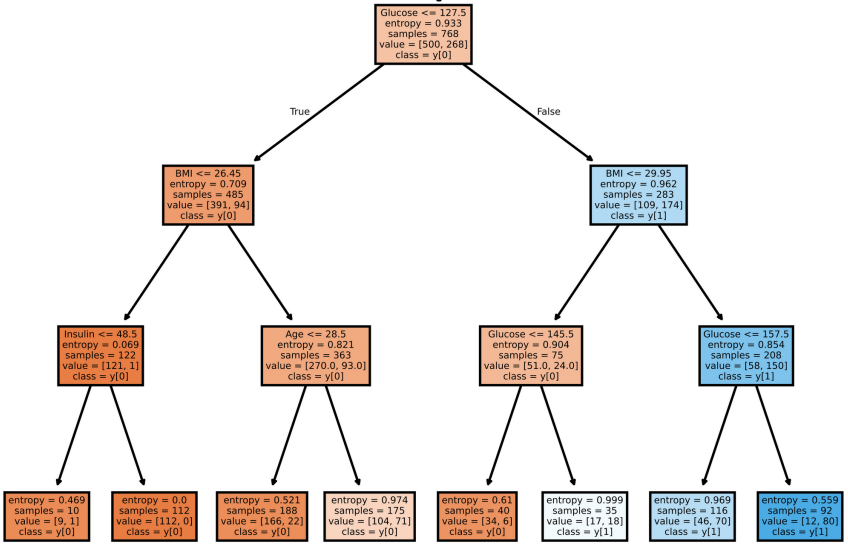
1. https ://www.kaggle.com/uciml/pima-indians-diabetes-database

Arbre de régression : prédire une valeur numérique



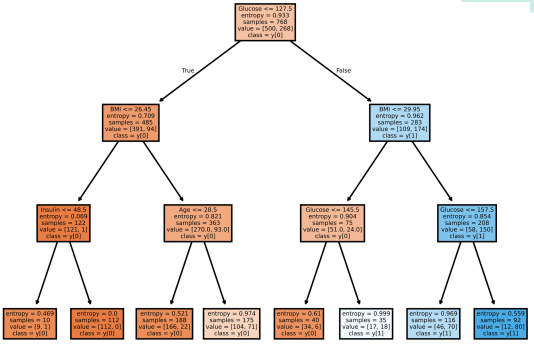


Les arbre de classification : prédire une classe



Processus récursif glouton de contruction d'un arbres

- Choix de l'attribut d'entrée X_i et d'un ensemble δ de valeurs qu'il peut prendre réalisant la "meilleure scission du jeu de données"
- Partition de l'ensemble des données selon X_i et δ
- 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 [2]

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

```python
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 partitº 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) [3]

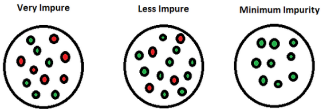
Meilleur découpage : impureté de Gini la plus réduite

```python
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 [5]

Illustrations de l'entropie [4]



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

4. 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 [6]

```python
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

```python
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 [7] : 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 [9]

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

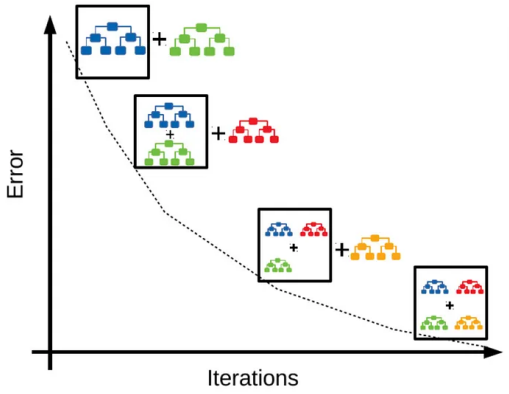


Figure – Illustration de G. Boosting [8]

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

Plan

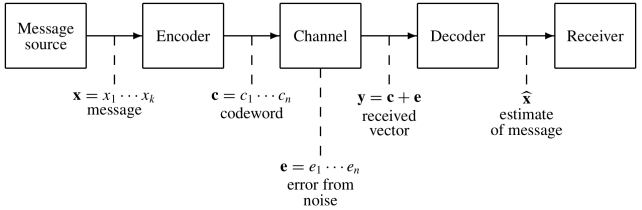
Intelligence Artificielle

Sécurité

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

Communication bruitée

Schéma théorique de communication avec correction [10]



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 [11] 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[12] 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 8ème 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 « ââgmee »)
- 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 ρ

ρ : rapport $\dfrac{\text{nombre de bits de message}}{\text{nombre de bits transmis}}$ 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[13] 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[14]

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[15], dans la DRAM[16], dans les OS,...

13. 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
15. https://fr.wikipedia.org/wiki/Code_QR
16. https://en.wikipedia.org/wiki/ECC_memory

Codes systématiques de Hamming[17] $(2^k - 1, 2^k - k - 1)$

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

$$H_k = \left(\frac{P_k}{I_k}\right) \qquad (1)$$

- 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

$$H_3 = \left(\begin{array}{ccc} 1&1&0\\ 1&0&1\\ 0&1&1\\ \hline 1&1&1\\ \hline 1&0&0\\ 0&1&0\\ 0&0&1 \end{array}\right) \text{ ou bien } H_3' = \left(\begin{array}{ccc} 1&1&1\\ 1&1&0\\ 1&0&1\\ \hline 0&1&1\\ \hline 1&0&0\\ 0&1&0\\ 0&0&1 \end{array}\right) \text{ ou encore } H_3'' = \left(\begin{array}{ccc} 0&1&1\\ 1&1&1\\ 1&0&1\\ \hline 1&1&0\\ \hline 1&0&0\\ 0&1&0\\ 0&0&1 \end{array}\right) \ldots$$

Matrices de génération G_k

Définition

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

- 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

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

Mot de code et syndrome

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

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

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

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

- $v = uG = (1, 0, 1, 1, 0, 1, 0)$
- $\sigma(v) = vH = (0, 0, 0)$
- 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 $w' = (w_1, \ldots, w_{i-1}, \overline{w_i}, \ldots, w_{2^k-1})$ est un mot de code.

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

Rappels

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

- $v = uG = (1, 0, 1, 1, 0, 1, 0)$
- $\sigma(v) = vH = (0, 0, 0)$
- 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 [18] $RS(n, k, m)$

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

$$C(x) = c_O + c_1 x^1 + c_2 x^2 + \dot{c}_2 x^{k+2t}$$

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

- Exemple avec $\mathbb{F}_{2^2}$

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

$+$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
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 $u = (u_1, \ldots, u_k)$, $u_i \in \mathbb{F}_{2^m}$: réécrit en $A(x) = u_1 x^{k-1} + u_2 x^{k-2} + \cdots + u_{k-1} x^1 + u_k$
- Pour α générateur de $\mathbb{F}_{2^m}$: polynome générateur $G(x) = (x + \alpha)(x + \alpha^2) \ldots (x + \alpha^{2t})$
- Polynome de contrôle : $B(x) \equiv A(x) \times x^{2t} \mod (G(x))$
- 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
- $A(x) = 3$
- $G(x) = (x + 2)(x + 2^2) = (x + 2)(x + 3) = x^2 + x + 1$
- $B(x) \equiv 3.x^2 \mod (G(x)) = 3x + 3 \mod (G(x))$ car

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

- 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	α^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 ν 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 ν 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 systèmes $\begin{aligned} S_1\lambda_1 &= S_2 \\ 1\lambda_1 &= 3 \end{aligned}$ admet une solution ($\lambda_1 = 3$). Il y a donc une erreur.

Localisation de l'erreur

Méthode

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

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 :

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

- 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, $1 \leq j \leq 2t$: $E(\alpha^j) = D(\alpha^j) + C(\alpha^j) = D(\alpha^j) + 0 = S_j = \sum_{r=1}^{\nu} e_{i_r} \alpha^{j.i_r}$
- Il suffit de résoudre le système

$$\begin{array}{rcl} e_{i_1}\alpha^{1.i_1} + \ldots e_{i_\nu}\alpha^{1.i_\nu} & = & S_1 \\ e_{i_1}\alpha^{2.i_1} + \ldots e_{i_\nu}\alpha^{2.i_\nu} & = & S_2 \\ & \vdots & \\ e_{i_1}\alpha^{2t.i_1} + \ldots e_{i_\nu}\alpha^{2t.i_\nu} & = & S_{2t} \end{array}$$

- 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 $\begin{array}{rcl} e_2 \times \alpha^2 & = & S_1 \\ e_2 \times \alpha^{2\times 2} & = & S_2 \end{array}$ c.-à-d. $\begin{array}{rcl} e_2 \times 3 & = & 1 \\ e_2 \times 2 & = & 3 \end{array}$ 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 [19]
- les turbo-codes [20]
  - 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) [21] :
  - 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`