
Table of Contents

Projet Partie I A23 ELE6701A	1
Generation des 1024 vecteurs hypotheses	1
Generation du bruit n	2
Generation de paquets s_i a envoyer	2
Vecteur y reçu	2
Borne Union	2
Commentaire I-b	3
Detection ML	3
Commentaire question I-e	4

Projet Partie I A23 ELE6701A

Bouh Abdillahi

Matricule : 1940646

```
clear all;
clc;
H=[1 0 0 0 0 0 0 0 0 0;
   -0.4 1 0 0 0 0 0 0 0 0;
   0 -0.4 1 0 0 0 0 0 0 0;
   0 0 -0.4 1 0 0 0 0 0 0;
   0 0 0 -0.4 1 0 0 0 0 0;
   0 0 0 0 -0.4 1 0 0 0 0;
   0 0 0 0 0 -0.4 1 0 0 0;
   0 0 0 0 0 0 -0.4 1 0 0;
   0 0 0 0 0 0 0 -0.4 1 0;
   0 0 0 0 0 0 0 0 -0.4 1];
```

Generation des 1024 vecteurs hypotheses

```
vecteurs_cell = cell(1, 1024);

for i = 1:1024
    % Convertir l'indice en binaire sur 10 bits
    binaire = dec2bin(i-1, 10) - '0'; % Convertir en vecteur binaire

    % Remplacer les '0' par '-1' dans la representation binaire
    binaire(binaire==0) = -1;

    % Stocker le vecteur dans la cellule
    vecteurs_cell{i} = binaire';
end

% Convertir la cellule en une matrice de taille 10x1024
matrice_vecteurs_hypothese_s = cell2mat(vecteurs_cell);
```

Generation du bruit n

Variance desiree

```
variance_desiree = (0.15);

% Nombre de vecteurs a generer
nombre_paquets = 10000; %% VALEUR A MODIFIER
nombre_vecteurs = nombre_paquets;
taille_vecteur = 10;

% Generation des vecteurs de bruit gaussien blanc avec moyenne nulle
vecteurs_bruit_gaussien = sqrt(variance_desiree) *
    randn(taille_vecteur, nombre_vecteurs);
```

Generation de paquets s_i a envoyer

Nombre de vecteurs aleatoires a generer

```
nombre_vecteurs = nombre_paquets;
taille_vecteur = 10;

% Generation des vecteurs aleatoires
vecteurs_aleatoires_envoyes = randi([1, 2], taille_vecteur,
    nombre_vecteurs);
vecteurs_aleatoires_envoyes(vecteurs_aleatoires_envoyes == 2) = -1;
```

Vecteur y reçu

Matrice des vecteurs y

```
y=vecteurs_aleatoires_envoyes + vecteurs_bruit_gaussien;
```

Borne Union

```
M = 2^10;
pi_i = 1/M;
Borne_union = 0;
N0 = 0.15;
Qmax = 0;
for i=1:1:M
    for j=1:1:M
        if j~=i
            Q = qfunc((1/
(2*sqrt(N0)))*norm(matrice_vecteurs_hypothese_s(:,i)-
matrice_vecteurs_hypothese_s(:,j))));
            Borne_union = Borne_union + pi_i*Q;
            if Q > Qmax
                Qmax = Q;
            end
        end
    end
end
```

```

        end
    end
    Borne_distance_minimale = 10*Qmax;
    disp("La Borne Union vaut : ")
    disp(Borne_union)

    disp("La Borne distance minimale vaut : ")
    disp(Borne_distance_minimale)

    La_Borne_Union_vaut :
        0.0555

    La_Borne_distance_minimale_vaut :
        0.0491

```

Commentaire I-b

Nous remarquons que la borne union est inférieure à la borne distance minimale. La probabilité d'erreur se trouve ainsi mieux estimée avec la borne distance minimale.

Detection ML

```

comparison_matrix=-1000000*ones(1,1024);
indice_guess = -1*ones(1,nombre_paquets);
vecteurs_s_guess = -3434314324*ones(10,nombre_paquets);
error_count_vector = 0;
error_count_symbol = 0;
for i=1:1:nombre_paquets

    for j=1:1:1024
        comparison_matrix(1,j) =
            real(transpose(matrice_vecteurs_hypothese_s(:,j))*Y(:,i) -
                0.5*transpose(matrice_vecteurs_hypothese_s(:,j))*matrice_vecteurs_hypothese_s(:,j))
    end
    [max_value, indice_guess(1,i)] = max(comparison_matrix);
    vecteurs_s_guess(:,i) =
        matrice_vecteurs_hypothese_s(:,indice_guess(1,i));
    comp = isequal(vecteurs_aleatoires_envoyes(:,i),
        vecteurs_s_guess(:,i));
    difference_per_symbol = nnz(vecteurs_s_guess(:,i) ~=
        vecteurs_aleatoires_envoyes(:,i));
    error_count_symbol = error_count_symbol + difference_per_symbol;
    if comp
        test = 0;
    else
        error_count_vector = error_count_vector+1;
    %     disp("error!");
    end

end

taux_erreur = error_count_vector/nombre_paquets

```

```
taux_erreur_par_symbole = error_count_symbol/(nombre_paquets*10)
```

```
taux_erreur =
```

```
0.0479
```

```
taux_erreur_par_symbole =
```

```
0.0050
```

Commentaire question I-e

Nous observons que ce code prend beaucoup de temps à s'exécuter du fait de sa grande complexité. En effet, tester toutes les hypothèses peut s'avérer très fastidieux, surtout pour un nombre important de paquets ou d'observations.

Published with MATLAB® R2019b