

SI221 - TP1

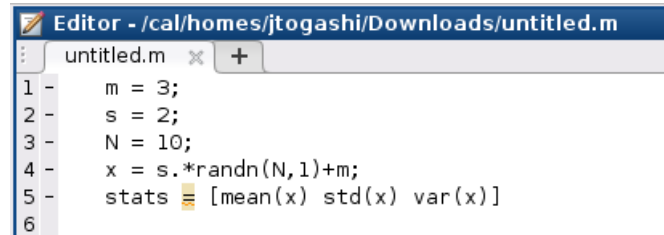
Júlia Togashi de Miranda

February 2021

1 Génération d'une variable aléatoire gaussienne

La fonction `randn` renvoie un échantillon de nombres aléatoires à partir d'une distribution normale avec une moyenne de 0 et une variance 1.

La théorie générale des variables aléatoires stipule que si x est une variable aléatoire dont la moyenne est μ_x et la variance est σ_x^2 , alors la variable aléatoire, y , définie par $y = ax + b$, où a et b sont des constantes, a la moyenne $\mu_y = a\mu_x + b$ et variance $\sigma_y^2 = a\sigma_x^2$.



```
Editor - /cal/homes/jtogashi/Downloads/untitled.m
untitled.m x +
1 - m = 3;
2 - s = 2;
3 - N = 10;
4 - x = s.*randn(N,1)+m;
5 - stats = [mean(x) std(x) var(x)]
6
```

Figure 1: Code pour Générer une variable aléatoire gaussienne

Nous pouvons percevoir qu'en augmentant le nombre d'échantillons N , la valeur de la moyenne et de la variance se rapproche de la valeur attendue. De plus, comme nous n'avons pas défini de random seed, augmentant le N , les valeurs commencent à varier moins entre les exécutions.

N	moyenne	écart-type	variance
10	3.5028	1.8950	3.5909
100	2.8208	2.0434	4.1754
1000	3.0457	1.9638	3.8566
10000	3.0041	1.9712	3.8858

2 Génération de vecteurs aléatoires gaussiens

1. Ensuite, nous pouvons voir le code en utilisant `randn` pour générer nos échantillons. Ce code sera adapté pour les prochaines questions.

```
Editor - /cal/homes/jtogashi/Downloads/untitled.m*
untitled.m* x +
1 - m = [4 9];
2 - c=[1 0 ; 0 1];
3 - N=100;
4 - d=2;
5 - x =randn(N,d)*chol(c)+m;
6 - plot(x(:,1),x(:,2), '*');
7 - stats_mean = mean(x)
8 - stats_cov = cov(x)
9
```

Figure 2: Code pour Générer de vecteurs aléatoires gaussiens

Voici les valeurs réelles de notre estimateur pour la moyenne et la variance, en plus du graphique montrant visuellement l'échantillonnage.

- `stats_mean` = [3.9897 9.1029]
- `stats_cov` = [1.1784 -0.0113; -0.0113 1.1756]

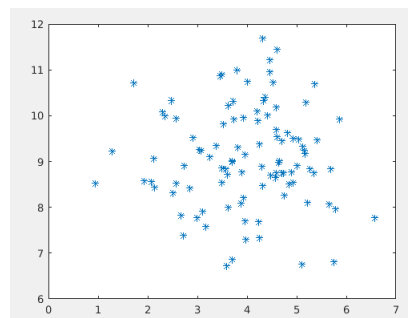


Figure 3: Affichage des échantillons

2. Pour cette question, nous avons simplement changé le code de la question ci-dessus afin que la matrice de covariance `c=[1 0 ; 0 6]`. Cela signifie que la variance de la première variable reste égale à 1, mais de la seconde elle passe à 6. Cela peut être confirmé dans le graphique ci-dessous, dans lequel l'échelle de l'axe y change.

- `stats_mean` = [3.9095 8.9347]
- `stats_cov` = [0.8476 -0.0543; -0.0543 5.6298]

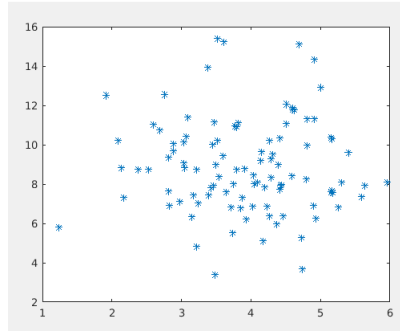


Figure 4: Affichage des échantillons

3. En étendant l'explication donnée à la question précédente, maintenant pour le cas de dimension supérieure à 1, nous pouvons obtenir le vecteur aléatoire centré et covariance souhaité en utilisant le même code déjà spécifié.

Un fait utile est que si nous avons un vecteur aléatoire x avec la matrice de covariance Σ , alors le vecteur aléatoire Ax a la moyenne $AE(x)$ et la matrice de covariance $\Omega = A\Sigma A^T$. Donc, si nous commençons avec des données qui ont une moyenne de zéro, la multiplication par A ne changera pas cela, donc nos données continueront à être centrées.

En commençant par zéro moyen et $\Sigma = I$. Nous pouvons transformer cela en données avec une matrice de covariance donnée en choisissant A comme racine carrée cholesky de Ω - alors Ax aurait la matrice de covariance souhaitée Ω .

4. Nous pouvons observer que, contrairement aux exemples précédents, où les échantillons étaient bien dispersés, en ajoutant maintenant la covariance, nous pouvons observer un axe de plus grande variance.

- stats_mean = [3.7445 8.9113]
- stats_cov = [1.7633 1.6055; 1.6055 4.7211]

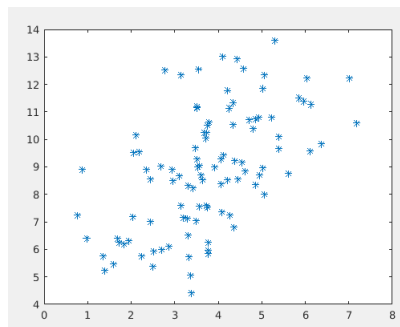


Figure 5: Affichage des échantillons

Utilisation de la fonction suggérée dans le texte du TP: $[V, S] = \text{eig}(s)$ retourne dans la matrice V les vecteurs propres de s , et dans S la matrice diagonalisée, nous avons:

- $V = [-0.8944 \ 0.4472; 0.4472 \ 0.8944]$
- $S = [1 \ 0; 0 \ 6]$

En utilisant la formule contenue dans la diapositive du cours, nous pouvons calculer l'ellipsoïde de Mahalanobis associé à s comme: $\tan(2\alpha) = \frac{2s_{12}}{s_{11}-s_{22}}$, donc $\alpha = -0.4636$

Nous pouvons vérifier cela si nous faisons $VS V^T$ nous avons correctement récupéré la matrice de covariance s .

5. En faisant le même processus, nous générerons maintenant 3 classes différentes:

```

Editor - /cal/homes/togashi/Downloads/untitled.m*
1 - m1 = [4 9];
2 - m2 = [8.5 7.5];
3 - m3 = [6 3.5];
4 - s1 = [2 2; 2 5];
5 - s2 = [2 -2; -2 5];
6 - s3 = [7 -4; -4 7];
7
8 - N=100;
9 - d=2;
10 - x1 = randn(N,d)*chol(s1)+m1;
11 - x2 = randn(N,d)*chol(s2)+m2;
12 - x3 = randn(N,d)*chol(s3)+m3;
13
14 - plot(x1(:,1),x1(:,2),'r*');
15 - hold on;
16 - plot(x2(:,1),x2(:,2),'g*');
17 - hold on;
18 - plot(x3(:,1),x3(:,2),'b*');
19 - stats_mean1 = mean(x1);
20 - stats_cov1 = cov(x1);
21 - stats_mean2 = mean(x2);
22 - stats_cov2 = cov(x2);
23 - stats_mean3 = mean(x3);
24 - stats_cov3 = cov(x3);
25

```

Figure 6: Code pour Générer de vecteurs aléatoires gaussiens

- Vecteurs aléatoires 1:
 - stats_mean1 = [3.9459 8.7993]
 - stats_cov1 = [2.0205 1.7505; 1.7505 3.7879]
- Vecteurs aléatoires 2:
 - stats_mean2 = [8.6673 7.3463]
 - stats_cov2 = [1.8925 -1.6944; -1.6944 4.7570]
- Vecteurs aléatoires 3
 - stats_mean3 = [6.1232 3.3856]
 - stats_cov3 = [6.3215 -3.9349; -3.9349 6.8949]

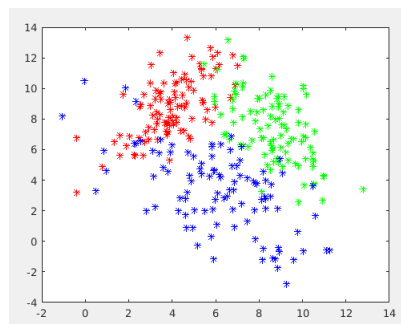


Figure 7: Affichage des échantillons

3 Courbes d'équidensité

1. Ensuite, nous avons le code pour générer la grille, en utilisant la fonction `mesh` et `linspace` avec les intervalles demandés.

```
Editor - /cal/homes/jtogashi/Downloads/untitled2.m
untitled.m  untitled2.m  +
1 - X = linspace(0.27,12.5,57);
2 - Y = linspace(-2,15,57);
3 - F = zeros(57);
4 - mesh(X,Y,F);
5 -
```

Figure 8: Code pour générer une grille

Ensuite, pour observer la grille, nous traçons n'importe quelle fonction, avec toutes les valeurs égales à zéro. On voit ainsi que les plages pour X et Y sont correctes.

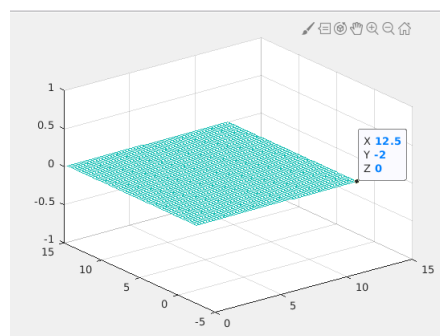


Figure 9: Exemple de grille utilisant des zéros pour pouvoir visualiser

2. Ci-dessous, nous utilisons certaines fonctions disponibles par Matlab pour générer la fonction de probabilité conditionnelle pour la classe 1.

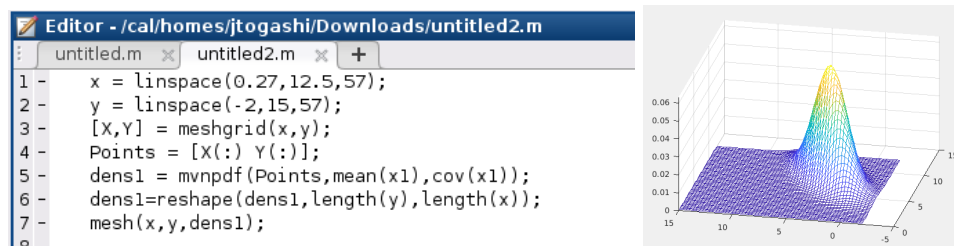


Figure 10: Code pour la densité de probabilité conditionnelle en chaque point grille et grille résultant

Comme on peut le voir, comme prévu, la densité de probabilité est plus élevée pour la moyenne de classe 1.

3. Le graphique ci-dessous a été obtenu en utilisant la fonction de contour, comme décrit dans les instructions TP. Les courbes résultantes sont des ellipses concentriques, ce qui est attendu, car la matrice de covariance n'est pas égale à l'identité.

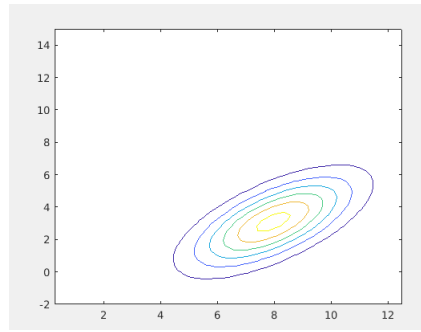


Figure 11: Les courbes d'équidensités pour la classe 1

4. En faisant de même pour les deux autres classes, nous obtenons:

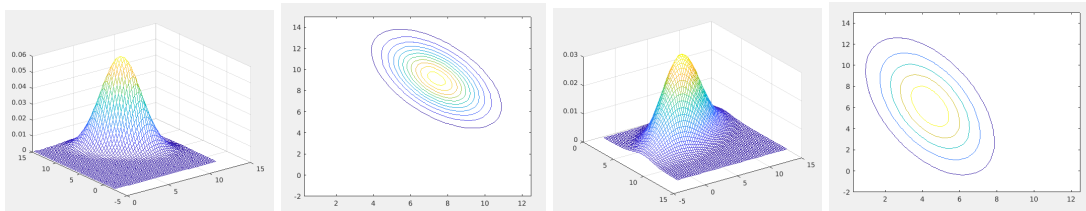


Figure 12: Classe 2 et Classe 3 respectivement

Comme on peut le voir sur la figure ci-dessous, les pics correspondent à la moyenne de chaque classe. La classe 3, qui a une matrice de covariance avec des valeurs plus élevées, a une courbe plus dispersée, donc un pic plus petit par rapport à la classe 1 par exemple.

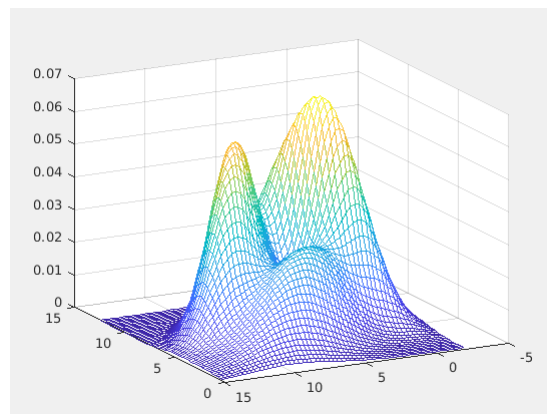


Figure 13: Sur la même figure les trois lois de densités conditionnelles

4 Visualisation des frontières

1. On peut observer le classement effectué compte tenu de la densité de probabilité de chaque classe, si à un point donné la densité de classe 1 est supérieure aux autres, ce point sera classé 1, et il sera au niveau 1 dans le graphique 3D ci-dessous . De même pour les autres classes.

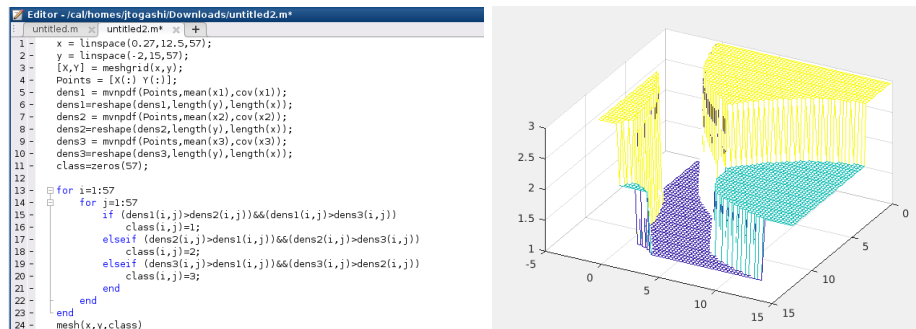


Figure 14: Classifier les points (X(i), Y(j)) de la grille dont les éléments contiendront l'étiquette (1, 2 ou 3)

2. Ci-dessous nous pouvons observer en 2D la division de l'espace et les frontières entre les 3 classes.

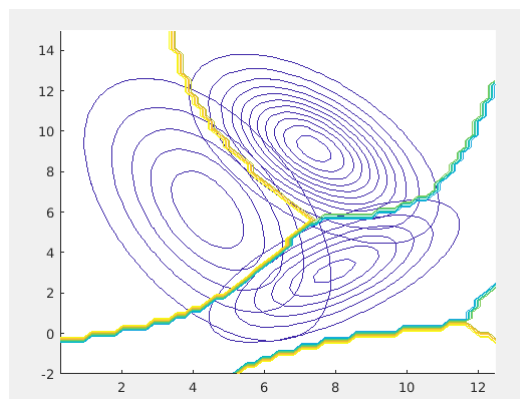


Figure 15: Les frontières entre classes

5 Application

On peut voir cela dans ce cas pratique, car les classes sont bien mais mixtes qu'auparavant. Une variable qui correspond au poids a une moyenne et une variance bien au-dessus du MPG.

Voici les moyennes et les variances pour chaque classe:

- Classe 1:
 - stats_mean1 = [20.0 3372.4]
 - stats_cov1 = [41.5 -4335.6; -4335.6 632576.3]

```

Editor - /cal/homes/jtogashi/Downloads/untitled4.m*
untitled.m  untitled2.m  untitled4.m*  +
1 - format long g
2 - s = load('voitures.mat');
3 - cars=s.cars;
4 - c1=find(cars(:,8)==1);
5 - c2=find(cars(:,8)==2);
6 - c3=find(cars(:,8)==3);
7 -
8 - class1=cars(c1,[1 5])
9 - class2=cars(c2,[1 5]);
10 - class3=cars(c3,[1 5]);
11 - m1=mean(class1)
12 - s1=cov(class1)
13 - m2=mean(class2)
14 - s2=cov(class2)
15 - m3=mean(class3)
16 - s3=cov(class3)
17 -
18 - x = linspace(-100,100,100);
19 - y = linspace(500,6000,100);
20 - [X,Y] = meshgrid(x,y);
21 - Points = [X(:) Y(:)];
22 - dens1 = mvnpdf(Points,m1,s1);
23 - dens1=reshape(dens1,length(y),length(x));
24 - dens2 = mvnpdf(Points,m2,s2);
25 - dens2=reshape(dens2,length(y),length(x));
26 - dens3 = mvnpdf(Points,m3,s3);
27 - dens3=reshape(dens3,length(y),length(x));
28 - class=zeros(100);

```

Figure 16: Code pour afficher voitures.mat

- Classe 2:
 - stats_mean2 = [27.6 2433.4]
 - stats_cov2 = [43.2 -1656.9; -1656.9 241880.8]
- Classe 3
 - stats_mean3 = [30.4 2221.2]
 - stats_cov3 = [37.0 -1101.0; -1101.0 102718.4]

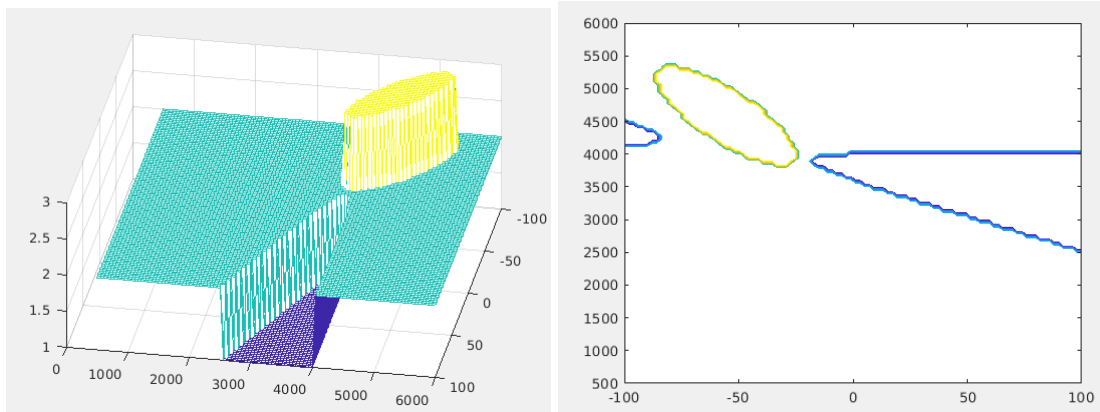


Figure 17: Les frontières entre classes