

Exercise 1:

$$1) X = R \cos(\theta), Y = R \sin(\theta)$$

$$\text{avec } f_R(r) = r e^{-r^2/2} \mathbb{1}_{[0, \infty)}(r)$$

$$f_\theta(t) = \frac{1}{2\pi} \mathbb{1}_{[0, 2\pi]}(t)$$

Montrer que X et Y sont indépendants

Soit $\lambda: \mathbb{R}^2 \rightarrow \mathbb{R}$ une fonction mesurable, bornée.

$$\mathbb{E}[\lambda(X, Y)] = \mathbb{E}[\lambda(R \cos(\theta), R \sin(\theta))]$$

$$= \iint_{\mathbb{R}^2} \lambda(r \cos(\theta), r \sin(\theta)) f_R(r) f_\theta(\theta) dr d\theta \text{ car } R \perp \theta$$

$$= \iint_{\mathbb{R}^2} \lambda(r \cos(\theta), r \sin(\theta)) \frac{r}{2\pi} e^{-r^2/2} \mathbb{1}_{[0, \infty)}(r) \mathbb{1}_{[0, 2\pi]}(\theta) dr d\theta$$

Soit $\phi: \mathbb{R}^2 \times [-\pi, \pi] \rightarrow \mathbb{R}^2 \setminus (\mathbb{R}_- \times \{0\})$

$$(r, \theta) \rightarrow (r \cos(\theta), r \sin(\theta))$$

On s'est ramené à l'intervalle ouvert $]-\pi, \pi[$ pour avoir une bijection.

Donc ϕ est un C^1 -diffeomorphisme car il y a bijection sur des intervalles ; on peut alors procéder au changement de variable et calculer la jacobienne :

$$|\mathcal{J}_{\phi(r, \theta)}| = \begin{vmatrix} \cos \theta & -r \sin \theta \\ \sin \theta & r \cos \theta \end{vmatrix} = r \cos^2 \theta + r \sin^2 \theta = r \neq 0$$

$$\text{Donc : } \mathbb{E}[\lambda(X, Y)] = \iint_{\mathbb{R}^2} \lambda(x, y) \frac{1}{2\pi} e^{-r^2/2} dx dy$$

Or, $x^2 + y^2 = r^2$ d'où :

$$\mathbb{E}[\lambda(X, Y)] = \iint_{\mathbb{R}^2} \lambda(x, y) \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dx dy$$

$$\text{Par identification : } f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, f_Y(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}$$

Ce résultat montre ainsi que X et Y sont indépendantes et que $\begin{cases} X \sim \text{U}(0,1) \\ Y \sim \text{U}(0,1) \end{cases}$

2) L'objectif est de générer un échantillon de deux lois $\text{U}(0,1)$ indépendantes. Pour cela, on va s'aider de la question 1) en générant X et Y à l'aide de la méthode de la fondation inverse.

Calculons l'inverse générale de R :

$$\begin{aligned} F_R(t) &= \int_{-\infty}^t r e^{-r^2/2} dr \quad r \geq 0 \\ &= \int_0^t r e^{-r^2/2} dr \\ &= [-e^{-r^2/2}]_0^t = 1 - e^{-t^2/2} \quad \text{si } t \geq 0 \end{aligned}$$

$$\text{et } F_R(t) = 0 \quad \text{si } t < 0$$

La fonction de répartition de R étant continue et strictement croissante sur \mathbb{R} , son inverse générale correspond à F_R^{-1} (inverse de sa fonction de répartition) par [RC01]

$$\begin{aligned} \text{On pose } t &= F_R(r) \Leftrightarrow t = 1 - e^{-r^2/2} \\ (\Leftrightarrow) \quad 1-t &= e^{-r^2/2} \\ (\Leftrightarrow) \quad -2\log(1-t) &= r^2 \quad \text{car } 1-t > 0 \\ (\Leftrightarrow) \quad r &= \sqrt{-2\log(1-t)} \quad \text{existe car } 1-t \in]0,1] \\ \text{Donc: } F_R^{-1}(r) &= \boxed{\sqrt{-2\log(1-t)}} \quad \text{pour } t \in]0,1] \end{aligned}$$

D'après la méthode de la fondation inverse, on a:

$$F_R^{-1}(U) \sim R \quad \text{pour } U \sim \text{Unif}(0,1)$$

$$\Leftrightarrow \sqrt{-2\log(U)} \sim R \quad \text{car } 1-U \sim U \sim \text{Unif}(0,1)$$

Pour simuler Θ , on simule une loi uniforme $(0,1)$ U qu'on

multiplicé par 2π ; $\Theta = 2\pi U$ avec $U \sim \text{Unif}(0,1)$

On peut alors écrire $(X, Y) = (R \cos(\Theta), R \sin(\Theta))$.

Ce qui nous donne l'algorithme suivant : (pseudo-code)

Sample U_1, U_2 indépendant r.v with distribution $U([0,1])$;

Set $R = \sqrt{-2 \log(U_1)}$ and $\Theta = 2\pi U_2$;

Set $X = R \cos(\Theta)$ and $Y = R \sin(\Theta)$;

Return (X, Y) .

3) a) A la fin de la boucle while de l'algorithme,

$$\text{on a } V_1^2 + V_2^2 \leq 1$$

Donc V_1 et V_2 sont sur le disque unité

$\mathcal{D}(0,1) = \{V_1^2 + V_2^2 \leq 1\}$ et dès lors que

$$V_1 = 2U_1 - 1 \text{ et } V_2 = 2U_2 - 1 \quad \text{avec } U_1, U_2 \sim U([0,1])$$

on a finalement : $(V_1, V_2) \sim \text{Unif}(\mathcal{D}(0,1))$ (loi uniforme
sur le disque unité)

$$b) T_1 = \frac{V_1}{(V_1^2 + V_2^2)^{1/2}} ; T_2 = \frac{V_2}{(V_1^2 + V_2^2)^{1/2}} ; V = V_1^2 + V_2^2$$

montrons que $T_1 \perp\!\!\!\perp V$:

Soit $h: \mathbb{R}^2 \rightarrow \mathbb{R}$ une fonction mesurable, bornée.

$$\mathbb{E}[h(T_1, V)] = \mathbb{E}\left[h\left(\frac{V_1}{(V_1^2 + V_2^2)^{1/2}}, V_1^2 + V_2^2\right)\right]$$

$$= \frac{1}{\pi} \iint_{V_1^2 + V_2^2 \leq 1} h\left(\frac{V_1}{(V_1^2 + V_2^2)^{1/2}}, V_1^2 + V_2^2\right) dV_1 dV_2$$

car l'aire du disque unité (de rayon 1) vaut π

$$= \frac{1}{\pi} \iint_{\{V_1^2 + V_2^2 \leq 1\} \cap \{V_2 \geq 0\}} h\left(\frac{V_1}{(V_1^2 + V_2^2)^{1/2}}, V_1^2 + V_2^2\right) dV_1 dV_2$$

car on a une symétrie pour V_2 en \mathbb{R}_+ et \mathbb{R}_- et

il nous faut une bijection pour avoir le changement de variable

Sur ϕ : $D(0,1) \cap \{v_2 > 0\} \rightarrow [-1,1] \times [0,1]$

$$(v_1, v_2) \rightarrow \left(\frac{v_1}{(v_1^2 + v_2^2)^{1/2}}, v_1^2 + v_2^2 \right)$$

ϕ est un C^1 -diffeomorphisme de $D(0,1) \cap \{v_2 > 0\}$

dans $(-1,1] \times [0,1]$ car :

- ϕ est C^1 sur $D(0,1) \cap \{v_2 > 0\}$ car chacune de ses composantes est C^1

- ϕ est injective sauf $\forall (v_1, v_2), \forall (v'_1, v'_2) \in D(0,1) \cap \{v_2 > 0\}$ tel que $(v_1, v_2) \neq (v'_1, v'_2)$
 $\phi(v_1, v_2) \neq \phi(v'_1, v'_2)$

car supposons $\phi(v_1, v_2) = \phi(v'_1, v'_2)$ alors

$$\frac{v_1}{(v_1^2 + v_2^2)^{1/2}} = \frac{v'_1}{(v'_1^2 + v'_2^2)^{1/2}} \quad \text{et } v_1^2 + v_2^2 = v'_1^2 + v'_2^2$$

$$(2) \quad v_1 = v'_1 \quad \text{et} \quad v_2^2 = v'_2^2$$

$$\Rightarrow v_2 = v'_2 \quad \text{car } v_2, v'_2 > 0$$

absurde donc ϕ injective (donc bijective)

Donc pour la jacobienne de ϕ :

$$|\mathcal{J}\phi(v_1, v_2)| = \begin{vmatrix} \frac{v_2^2}{(v_1^2 + v_2^2)^{3/2}} & -\frac{v_1 v_2}{(v_1^2 + v_2^2)^{3/2}} \\ 2 v_1 & 2 v_2 \end{vmatrix} \\ = \frac{2 v_2^3 + 2 v_1^2 v_2}{(v_1^2 + v_2^2)^{3/2}} = \frac{2 v_2 (v_2^2 + v_1^2)}{(v_1^2 + v_2^2)^{3/2}}$$

$$|\mathcal{J}\phi(v_1, v_2)| = \frac{2 v_2}{(v_1^2 + v_2^2)^{1/2}} = 2 \left(1 - \frac{v_1^2}{v_1^2 + v_2^2}\right)^{1/2} > 0$$

D'où : $\#(\lambda(T_1, V)) = \frac{1}{\pi} \iint_{(-1,1] \times [0,1]} \lambda(t, v) \frac{1}{\sqrt{1-t^2}} dt dv$

ce résultat montre que $T_1 \perp\!\!\!\perp V$.

On utilise le même raisonnement pour montrer l'indépendance entre T_2 et V :

$$\mathbb{E}[h(T_2, V)] = \frac{1}{\pi} \iint_{\{(v_1^2 + v_2^2) \leq 1\} \cap \{v_1 > 0\}} h(v_2/(v_1^2 + v_2^2)^{1/2}, v_1^2 + v_2^2) dv_1 dv_2$$

on a $\{v_1 > 0\}$ car on a une parité en v_1 .

De même que précédemment, soit:

$$\phi : D(0, 1) \cap \{v_1 > 0\} \rightarrow [-1, 1] \times [0, 1]$$

$$(v_1, v_2) \rightarrow \left(\frac{v_2}{(v_1^2 + v_2^2)^{1/2}}, v_1^2 + v_2^2 \right)$$

est un C^1 -diffeomorphisme par les mêmes arguments que précédemment avec:

$$|\mathcal{J}_{\phi(v_1, v_2)}| = \begin{vmatrix} -v_1 v_2 & \frac{v_1^2}{(v_1^2 + v_2^2)^{3/2}} \\ (v_1^2 + v_2^2)^{3/2} & \frac{v_1^2}{(v_1^2 + v_2^2)^{3/2}} \\ 2v_1 & 2v_2 \end{vmatrix}$$

$$|\mathcal{J}_{\phi(v_1, v_2)}| = 2 \left(1 - \frac{v_2^2}{v_1^2 + v_2^2} \right)^{1/2} > 0$$

D'où: $\mathbb{E}[h(T_2, V)] = \frac{1}{\pi} \iint_{[-1, 1] \times [0, 1]} h(\tau, v) \frac{1}{\sqrt{1-\tau^2}} d\tau dv$

Donc $T_2 \perp\!\!\!\perp V$ d'où: $(T_1, T_2) \perp\!\!\!\perp V$

Montrons que $(T_1, T_2) \sim (\cos(\Theta), \sin(\Theta))$

Soit $h : \mathbb{R} \rightarrow \mathbb{R}$ une fonction mesurable, bornée.

$$\mathbb{E}[h(\cos(\Theta))] = \frac{1}{2\pi} \int_0^{2\pi} h(\cos(\theta)) d\theta$$

$$= \frac{1}{2\pi} \int_0^\pi h(\cos(\theta)) d\theta + \frac{1}{2\pi} \int_\pi^{2\pi} h(\cos(\theta)) d\theta$$

Pour (1), on fait le changement de variable bijectif sur $[0, \pi]$: $t = \cos(\theta)$ et $dt = -\sin(\theta) d\theta$

$$\textcircled{1} = - \int_{-1}^{-1} h(t) \sin(\theta) dt = \int_{-1}^1 h(t) \frac{1}{\sqrt{1-t^2}} dt$$

$$\text{car } t^2 = \cos^2(\theta) \Leftrightarrow t^2 = 1 - \sin^2(\theta) \\ \Leftrightarrow \sin(\theta) = \sqrt{1-t^2}$$

Pour \textcircled{2}, on fait le changement de variable $u = \theta - \pi$
 $du = d\theta$

$$\textcircled{2} = \int_0^\pi h(\cos(u+\pi)) du = \int_0^\pi h(\cos(u)) du$$

Puis on fait le changement de variable bijectif sur $[0; \pi]$:

$$t = -\cos(u)$$

$$dt = \sin(u) du = \sqrt{1-t^2} du$$

$$\textcircled{2} = \int_{-1}^1 h(t) \frac{1}{\sqrt{1-t^2}} dt$$

$$\text{Donc: } E[h(\cos(\theta))] = \frac{1}{2\pi} \left[2 \int_{-1}^1 h(t) \frac{1}{\sqrt{1-t^2}} dt \right]$$

$$E[h(\cos(\theta))] = \int_{-1}^1 h(t) \frac{1}{\pi \sqrt{1-t^2}} dt$$

On reconnaît la densité qui est apparue lors de la démonstration de l'indépendance entre T_1 et V .

$$f_{T_1}(t) = \frac{1}{\pi \sqrt{1-t^2}} \quad t \in [-1, 1] \quad \text{donc } T_1 \sim \cos(\theta)$$

$$\text{et } V \sim U([0, 1])$$

par identification avec l'intégrale qui montre que $T_1 + V$

De même pour $h: \mathbb{R} \rightarrow \mathbb{R}$ mesurable et bornée!

$$E[h(\sin(\theta))] = \frac{1}{2\pi} \int_0^{2\pi} h(\sin(\theta)) d\theta$$

$$= \frac{1}{2\pi} \int_{-\pi/2}^{\pi/2} h(\sin(\theta)) d\theta$$

Si on pose le changement de variable $T = \sin(\Theta)$

$$dT = \cos(\Theta) d\Theta$$

$$\Leftrightarrow \frac{1}{\sqrt{1-T^2}} dT = d\Theta$$

$$\text{Donc: } E[h(\cos(\Theta))] = \frac{1}{\pi} \int_{-1}^1 h(T) \frac{1}{\sqrt{1-T^2}} dT$$

Donc $T_2 \sim \sin(\Theta)$ et finalement:

$$(T_1, T_2) \sim (\cos(\Theta), \sin(\Theta))$$

Une façon un peu plus intuitive de le montrer est

que: $(V_1, V_2) \sim (\sqrt{V} \cos(\Theta), \sqrt{V} \sin(\Theta))$

avec $\sqrt{V} \sim U([0, 1])$ car (V_1, V_2) est un point
d'un disque unité, donc on a:

$$\left(\frac{V_1}{\sqrt{V}}, \frac{V_2}{\sqrt{V}} \right) \sim (T_1, T_2) \sim (\cos(\Theta), \sin(\Theta))$$

c) On remarque que: $X = ST_1$ et $Y = ST_2$

Gr; $S = \sqrt{-2 \log(V)}$ avec $V \sim U([0, 1]) \rightarrow$ question 3.b

S fait référence à l'inverse généralisé de R qu'en
a démontré à la question 2.

Donc: $S \sim R$ (Rayleigh distribution)

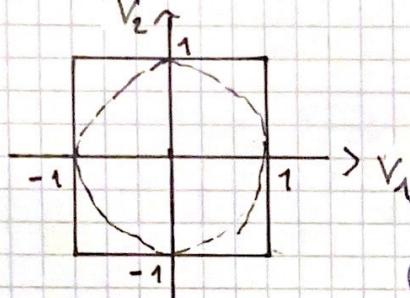
Etant donné que $(T_1, T_2) \sim (\cos(\Theta), \sin(\Theta))$

avec $\Theta \sim U([0, 2\pi])$, on a finalement:

$$(X, Y) \sim (R \cos(\Theta), R \sin(\Theta)) \quad \text{question 1}$$

Donc: $X, Y \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1)$

d) Faisons un schéma pour la réalisation de (V_1, V_2)



Tous que (V_1, V_2) est en dehors du disque white, on continue de tirer un autre point. Ainsi que

(V_1, V_2) est obligatoirement dans le carré car $V_1 = 2U_1 - 1$ et $V_2 = 2U_2 - 1$ avec $U \in [0, 1]$.

Donc la probabilité que (V_1, V_2) soit dans le cercle correspond à : $P(V_1, V_2) \in \mathcal{D}(0, 1)) = \frac{\text{Aire}(\mathcal{D}(0, 1))}{\text{Aire}(\text{carré})} = \frac{\pi}{4}$

Le nombre de fois où la boucle sera répétée c'est-à-dire le nombre de fois où on doit tirer un point jusqu'à qu'il se trouve dans le cercle peut être modélisé par une loi géométrique avec comme paramètre de succès $p = \frac{\pi}{4}$ donc $N \sim G(\frac{\pi}{4})$ avec N

une variable aléatoire qui représente le nombre de fois où la boucle a été exécutée.

Donc $E(N) = \frac{1}{p} = \frac{4}{\pi} \in]1, 2[$

Donc le nombre de fois attendu où on parcourt la boucle white est de 2 fois.

Exercice 2:

On définit l'ensemble $Q = \{ \frac{1}{m} \mid m \in \mathbb{N}^* \}$

1) $P(x_n, A) = \mathbb{P}(X_{n+1} \in A \mid X_n = x_n)$

Si $x \notin Q$: on a alors $X_{n+1} \sim U([0, 1])$ donc:

$$P(x_n, A) = \int_A \mathbb{1}_{[0,1]}(t) dt$$

$$P(x_n, A) = \int_{A \cap [0,1]} dt$$

Si $x \in Q$: on pose $E_1 = \{ X_{n+1} = \frac{1}{m+1} \}$ et $E_2 = \{ X_{n+1} \sim U([0, 1]) \}$ \rightarrow forment une partition

D'après la loi des probas totales:

$$\mathbb{P}(X_{n+1} \in A) = \mathbb{P}(E_1) \mathbb{P}(X_{n+1} \in A \mid E_1) + \mathbb{P}(E_2) \mathbb{P}(X_{n+1} \in A \mid E_2)$$

$$= (1 - x_n^2) \underbrace{\int_{\frac{1}{m+1}} \frac{1}{m+1} dt}_{\text{ Sachant } X_{n+1} = \frac{1}{m+1}, \text{ voir précédemment}} + x_n^2 \underbrace{\int_{A \cap [0,1]} dt}_{\mathbb{P}(X_{n+1} \in A \mid E_2) = 0}$$

$$\text{ Sachant } X_{n+1} = \frac{1}{m+1}, \text{ voir précédemment}$$

$$\mathbb{P}(X_{n+1} \in A \mid E_1) = 0$$

$$\text{ Si } \frac{1}{m+1} \notin A$$

Donc en remplaçant X_n par x on obtient finalement:

$$P(x_n, A) = (1 - x^2) \int_{A \cap [0,1]} dt + x^2 \int_{A \cap [0,1]} dt \quad \text{ si } x = \frac{1}{m} \in Q$$

$$\int_{A \cap [0,1]} dt \quad \text{sinon}$$

2) Montrons que: $\forall A \in \mathcal{B}([0,1])$, $\Pi P(A) = \Pi(A)$

$$\Pi P(A) = \int_{\mathbb{R}} \Pi(\omega) P(\omega, A) d\omega$$

$$= \int_{\mathbb{Q}} \Pi(\omega) P(\omega, A) d\omega + \int_{\mathbb{Q}^c} \Pi(\omega) P(\omega, A) d\omega$$

Or: $\mathbb{Q} = \left\{ \frac{1}{m} ; m \in \mathbb{N}^* \right\} = \bigcup_{m \in \mathbb{N}^*} \left\{ \frac{1}{m} \right\}$ c'est une union de singletons dénombrable

\rightarrow donc \mathbb{Q} est de mesure nulle

$$\Rightarrow \int_{\mathbb{Q}} \Pi(\omega) P(\omega, A) d\omega = 0$$

Donc: $\Pi P(A) = \int_{\mathbb{Q}^c} \Pi(\omega) P(\omega, A) d\omega$) car $\omega \notin \mathbb{Q}$

$$= \int_{\mathbb{Q}^c} \Pi(\omega) \underbrace{\int_{A \cap [0,1]} d\Gamma}_{\substack{= \mathbb{E}[f(X_1)] \\ = 1}} d\omega$$

$$\Pi P(A) = 1 \times \Pi(A)$$

$$\boxed{\Pi P(A) = \Pi(A)} \quad \forall A \in \mathcal{B}([0,1])$$

Donc Π est invariant pour P ,

3) Soit $x \notin \left\{ \frac{1}{m} ; m \in \mathbb{N}^* \right\} = \mathbb{Q}$

$$Pf(x) = \mathbb{E}[f(X_1) | X_0 = x]$$

$$= \int_{\mathbb{R}} f(y) P(x, y) dy \quad) \text{ car } x \notin \mathbb{Q}$$

$$= \int_{[0,1]} f(y) dx dy$$

$$\boxed{Pf(x) = \int_{[0,1]} f(y) dy = \Pi(f)}$$

Par récurrence, montrons que: $P^n f(x) = \Pi(f) \quad \forall n \geq 1$

Initialisation: $P^1 f(x) = Pf(x) = \Pi(f)$ (voir précédemment)

Hérédité: Supposons que $\forall n \in \mathbb{N}^*$, $P^n f(x) = \Pi(f)$

montrons au rang $n+1$ soit: $P^{n+1} f(x) = \Pi(f)$

$$\begin{aligned}
 P^{n+1}(f(x)) &= P(P^n(f(x))) = \int_{\mathbb{R}} P(x, dy) P^n(f(y)) \quad \text{car } x \notin Q \\
 &= \int_{\mathbb{R}} P^n(f(y)) dy \quad \text{d'après l'hypothèse de récurrence} \\
 &= \int_{\mathbb{R}} \int_{\mathbb{R}} f(y) \pi_{G^n}(dy) dy \\
 &= \int_{\mathbb{R}} f(x) \pi_{G^n}(dx) \\
 P^{n+1}(f(x)) &= \pi(f(x))
 \end{aligned}$$

Donc : $P^n(f(x)) = \pi(f(x)) \quad \forall n \geq 1$

et $\lim_{n \rightarrow +\infty} P^n(f(x)) = \pi(f(x)) = \int_{\mathbb{R}} f(x) \pi(dx)$

4) Soit $x = \frac{1}{m}$, $m \geq 2$

a) Par récurrence, montrons que : $P^n(x, \frac{1}{m+n}) = \prod_{k=0}^{n-1} \left(1 - \frac{1}{(m+k)^2}\right)$

Initialisation : pour $n=1$: $P(x, \frac{1}{m+1}) = x^2 \int_{\frac{1}{m+1}}^1 f_{\pi_{G^1}}(y) dy + (1-x^2) f_{\pi_{G^1}}(\frac{1}{m+1})$

$P(x, \frac{1}{m+1}) = 1 - x^2 = 1 - \frac{1}{m^2}$ OK pour $n=1$

Hérédité : Supposons qu'il existe un rang $n \geq 1$ tel que

$$P^n(x, \frac{1}{m+n}) = \prod_{k=0}^{n-1} \left(1 - \frac{1}{(m+k)^2}\right)$$

montrons-le au rang $n+1$:

$$\begin{aligned}
 P^{n+1}(x, \frac{1}{m+n+1}) &= P(P^n(x, \frac{1}{m+n})) \\
 &= \int P^n(y, \frac{1}{m+n+1}) P(x, dy) \quad \text{car } x = \frac{1}{m} \\
 &= \int P^n(y, \frac{1}{m+n+1}) (1-y^2) f_{\pi_{G^{n+1}}}(dy) \quad \text{on intègre sauf la mesure de Dirac} \\
 &= (1-x^2) P^n(\frac{1}{m+n}, \frac{1}{m+n+1})
 \end{aligned}$$

$$\begin{aligned}
 P^{n+1}(x, \frac{1}{m+n+1}) &= \left(1 - \frac{1}{m^2}\right) P^n\left(\frac{1}{m+1}, \frac{1}{m+n+1}\right) \quad \text{hypothèse de récurrence et on pose} \\
 &= \left(1 - \frac{1}{m^2}\right) \prod_{k=0}^{n-1} \left(1 - \frac{1}{(m+k+1)^2}\right) \\
 &= \left(1 - \frac{1}{m^2}\right) \prod_{k=1}^n \left(1 - \frac{1}{(m+k)^2}\right)
 \end{aligned}$$

$$P^{n+1}(x, \frac{1}{m+n+1}) = \prod_{k=0}^n \left(1 - \frac{1}{(m+k)^2}\right)$$

Donc l'hypothèse de récurrence est vérifiée, on a alors :

$$\forall n \in \mathbb{N}^*, P^n(x, \frac{1}{n+m}) = \prod_{k=0}^{n-1} \left(1 - \frac{1}{(m+k)^2}\right)$$

b) Pour $A = \bigcup_{q \in \mathbb{N}} \left\{ \frac{1}{m+1+q} \right\}$, on a :

$$\begin{aligned}
 P^n(x, A) &= P^n(x, \bigcup_{q \in \mathbb{N}} \left\{ \frac{1}{m+1+q} \right\}) \\
 &= \sum_{q \in \mathbb{N}} P^n(x, \frac{1}{m+1+q}) \rightarrow \text{somme de termes positifs} \\
 &\geq P^n(x, \frac{1}{m+1+(n-1)}) \rightarrow \text{pour } q=n-1 \\
 &= P^n(x, \frac{1}{m+n}) \quad \text{d'après 4.a)} \\
 &= \prod_{k=0}^{n-1} \left(1 - \frac{1}{(m+k)^2}\right) \quad \text{bien}
 \end{aligned}$$

$$\begin{aligned}
 \text{Dès } P^n(x, \frac{1}{m+n}) &= \prod_{k=m}^{m+n-1} \left(1 - \frac{1}{k^2}\right) \quad \text{car } 1 - \frac{1}{k^2} < 1 \text{ et } m \geq 2 \\
 &\geq \prod_{k=2}^{m+n-1} \left(1 - \frac{1}{k^2}\right) \\
 &= \prod_{k=2}^{m+n-1} \frac{(k+1)(k-1)}{k^2} \\
 &= \prod_{k=2}^{m+n-1} \left(\frac{k+1}{k}\right) \prod_{k=2}^{m+n-1} \frac{k-1}{k}
 \end{aligned}$$

$$\Rightarrow P^n(x, \frac{1}{m+n}) \geq \frac{m+n}{2} \times \frac{1}{m+n-1}$$

$$= \frac{1}{2} \times \frac{m+n}{m+n-1}$$

$$= \frac{1}{2} \times \frac{1 + \frac{m}{n}}{\frac{m}{n} + 1 - \frac{1}{n}}$$

Par passage des inégalités large de la limite, on obtient:

$$\lim_{n \rightarrow \infty} P^n(x, A) \geq \lim_{n \rightarrow \infty} P^n(y, \frac{1}{m+n}) \geq \frac{1}{2} > 0$$

Gr: $A = \bigcup_{q \in \mathbb{N}} \left\{ \frac{1}{m+1+q} \right\}$ est une union dénombrable de singletons, donc il est de mesure nulle

$$\Rightarrow \pi(A) = \int_{A \cap [0,1]} dt = 0$$

Finallement: $\lim_{n \rightarrow \infty} P^n(x, A) > \pi(A)$ car $\pi(A) = 0 < \frac{1}{2}$

donc $\boxed{\lim_{n \rightarrow \infty} P^n(y, A) \neq \pi(A)}$

Exercice 3: Stochastic Gradient Learning in Neural Networks

- 1) L'objectif ici est de résoudre le problème:

$$\min_{w \in \mathbb{R}^n} R_n(w) = \min_w \frac{1}{n} \sum_{i=1}^n (y_i - w^T x_i)^2$$

On va pour cela se servir de l'algorithme de descente du gradient stochastique qui a pour avantage par rapport à la descente du gradient standard de ne pas calculer à chaque itération le gradient sur l'ensemble des données mais uniquement sur un échantillon aléatoire d'où le nom stochastique. Calculons le gradient concerné:

$$L(x_i; w, y_i) = (y_i - w^T x_i)^2 = f(x_i w, y_i)$$

$$\nabla_w f(x_i w, y_i) = 2(y_i - w^T x_i)x_i$$

Condition de convergence: (conv.)

→ f est convexe (car linéaire en w) et différentiable par rapport à w

→ $\|\nabla_w f\| \leq \gamma$ si $\exists C > 0$, il y a $\|w\| \leq C$ car:
 $\|\nabla_w f\| \leq 2(1 + C\|w\|)\|x_i\|$

x_i est borné car notre ensemble d'observations est fini

→ il faut imposer une suite $(\varepsilon_k)_{k \geq 0}$ décroissante, $\varepsilon_k > 0 \forall k$
 $\sum_{k \in \mathbb{N}} \varepsilon_k = +\infty$ et $\sum_{k \in \mathbb{N}} \varepsilon_k^2 < +\infty$

Voici le déroulé de notre algorithme en pseudo-code:

- Choisir un vecteur $w_0 \in \mathbb{R}^n$ et une suite $(\varepsilon_k)_{k \geq 0}$, positive

- Pour $k = 0, 1, 2, \dots, n$

- > Tirer aléatoirement une observation (x_{k+1}, y_{k+1}) indépendante

- des observations précédentes $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$

- > $w_{k+1} = w_k - \varepsilon_k \nabla_w f(x_{k+1}; w_k, y_{k+1})$

La condition d'arrêt ici est l'exécution de n fois l'intérieur de la boucle

TP1_MARCIANO_MATHIAS

November 3, 2022

1 Computational Statistics

2 TP 1 : Reminder on Markov Chains – Stochastic gradient descent

Question 1

```
[4]: import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import array

[5]: def stochastic_gradient(X, Y, w0, N, alpha=1):
    """
    X, Y : liste des observations
    w0 : notre vecteur initial
    N : nombre d'iteration de l'algorithme
    On choisit notre suite epsilon = 1/k**alpha avec alpha=1 par defaut
    """
    w_bis = w0
    n = len(Y)
    for k in range(1, N+1):
        i = np.random.randint(0, n) # Tire un nombre aleatoirement entre 0 et n-1
        gradient = -2 * (Y[i] - dot(X[i,:], w_bis)) * X[i,:]
        wk = w_bis - (1/k**alpha) * gradient
        w_bis = wk
        k+=1
    return wk
```

Question 2

```
[6]: def sample_x(n):
    #Genere un echantillon des xi de taille n sur la boule unite (qui implique une convergence car le gradient est borne)
    return np.array(np.random.uniform(-1,1, size=(n,2)))

def sample_w():
```

```

#Genere un vecteur normal aleatoirement pour un hyperplan
w = np.array(np.random.uniform(-1,1,size=2))
return w/np.linalg.norm(w)

def sample_z(n, w):
    #Genere un echantillon de z = (x, y) de taille n
    Z = []
    x = sample_x(n)
    for k in range(n):
        y = np.dot(x[k,:], w)
        Z.append([x[k,:], y])
    return Z

def graph(Z,w, xlim1=-1.1, xlim2=1.1, ylim1=-1.1, ylim2=1.1, bruit=False, ↴
legend=True):
    #Permet de tracer notre resultat
    n = len(Z)
    x=[]
    y=[]
    for k in range(n):
        x.append(Z[k][0][0])
        y.append(Z[k][0][1])

    plt.figure(figsize=(10,7))

    colors = ['red' if Z[k][1] > 0 else 'green' for k in range(n)] # Associe une ↴
couleur pour chaque la
    plt.scatter(x, y, c=colors) ## The observations

    if bruit==True: #Utile pour question 4
        a=np.linspace(-2,2,500)
        plt.plot(a,-(w[0]/w[1])*a, color='black', lw=3, linestyle='--')
    else:
        droite = (1.2 / (w[0]**2 + w[1]**2))**0.5 * np.array((-w[1], w[0]))
        plt.plot([-droite[0], droite[0]], [-droite[1], droite[1]], color='black', ↴
linestyle='--', lw=3) # On trace la droite separant les differents labels

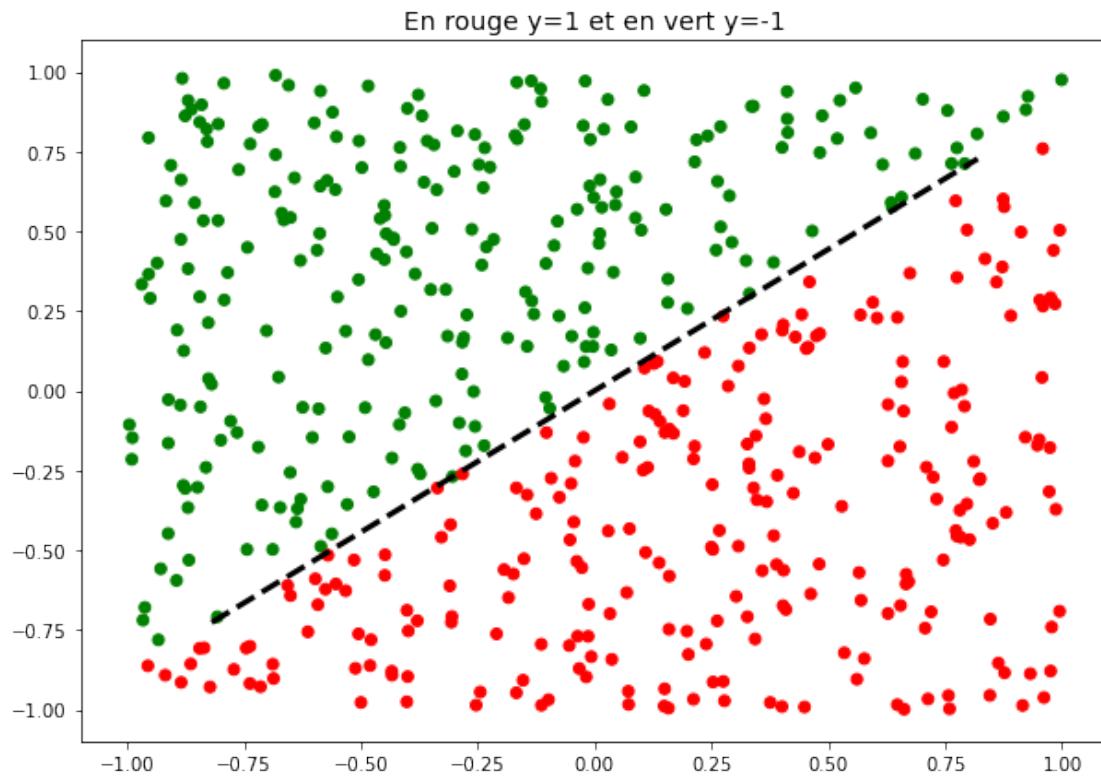
    if legend==True:
        plt.title("En rouge y=1 et en vert y=-1" , fontsize = 14)

    plt.xlim((xlim1,xlim2))
    plt.ylim((ylim1,ylim2))
    plt.show()

```

Exemple pour $n = 200$

```
[139]: #On génère une collection de 500 observations ainsi qu'un vecteur normal w d'un
       ↪hyperplan
n = 500
w = sample_w()
Z = sample_z(n,w)
graph(Z,w)
```



Question 3

L'idée maintenant est d'approcher le vecteur normal w grâce à la méthode de la descente du gradient stochastique et de pouvoir comparer si l'algorithme a été performant ou non.

```
[123]: def performance(X,Y,w):
    #Cette fonction performance nous permet d'évaluer le pourcentage de réussite
    ↪de l'algorithme
    Y = sign(Y)
    n = len(Y)
    succes = 0
    for i in range(n):
        if sign(Y[i]) == sign(dot(X[i,:],w)):
            succes += 1
    return (succes/n*100)
```

```
[140]: #On creer un tableau numpy pour les observations xi et yi
X = np.array((Z[0][0][0], Z[0][0][1]))

for k in range(1, n):
    X = np.vstack([X, [Z[k][0][0], Z[0][0][1]]])

Y = np.dot(X, w)

#On prend aleatoirement notre vecteur initial pour la descente du gradient
#stochastique
w0 = sample_w()

#On initialise à 700 iterations
N = 700

w_gradient = stochastic_gradient(X, Y, w0, N)
#On normalise le vecteur pour pouvoir évaluer la performance de l'algorithme
w_gradient = w_gradient / np.linalg.norm(w_gradient)

print("Le vecteur normal issu des donnees est : ", w)
print("Le vecteur normal estimé w* = ", w_gradient)
print("L'ecart ou l'erreur entre les deux vecteurs est de : ", np.linalg.
      norm(w-w_gradient))
print("Le taux de reussite de l'algorithme est de : ", performance(X,Y,w_gradient), "%")
```

Le vecteur normal issu des donnees est : [0.66369382 -0.74800436]

Le vecteur normal estimé w* = [0.62982745 -0.77673508]

L'ecart ou l'erreur entre les deux vecteurs est de : 0.04441153809260878

Le taux de reussite de l'algorithme est de : 100.0 %

L'algorithme de descente du gradient stochastique est performant dans notre cas car il est très proche du vecteur normal à estimer (l'erreur est faible) et a un bon taux de réussite pour estimer les labels.

Question 4

```
[141]: #On initialise un bruit gaussien totalement aleatoire
bruit = 0.4 * np.random.normal(0,1,size=(n,2))
X_bruit = X + bruit

w0 = np.array(np.random.uniform(-1,1,size=2))
w0 = w0/np.linalg.norm(w0)

w_gradient_bruit = stochastic_gradient(X_bruit, Y, w0, N)
w_gradient_bruit = w_gradient_bruit/np.linalg.norm(w_gradient_bruit)
```

```

Y_bruit = np.dot(X_bruit, w_gradient_bruit)

print("Le vecteur normal issu des donnees est : ", w)
print("Le vecteur normal estimé w* = ", w_gradient_bruit)
print("L'ecart ou l'erreur entre les deux vecteurs est de : ", np.linalg.norm(w
    - w_gradient_bruit))
print("Le taux de reussite de l'algorithme est de : ", performance(X_bruit, Y, w
    - w_gradient_bruit), "%")

```

Le vecteur normal issu des donnees est : [0.66369382 -0.74800436]

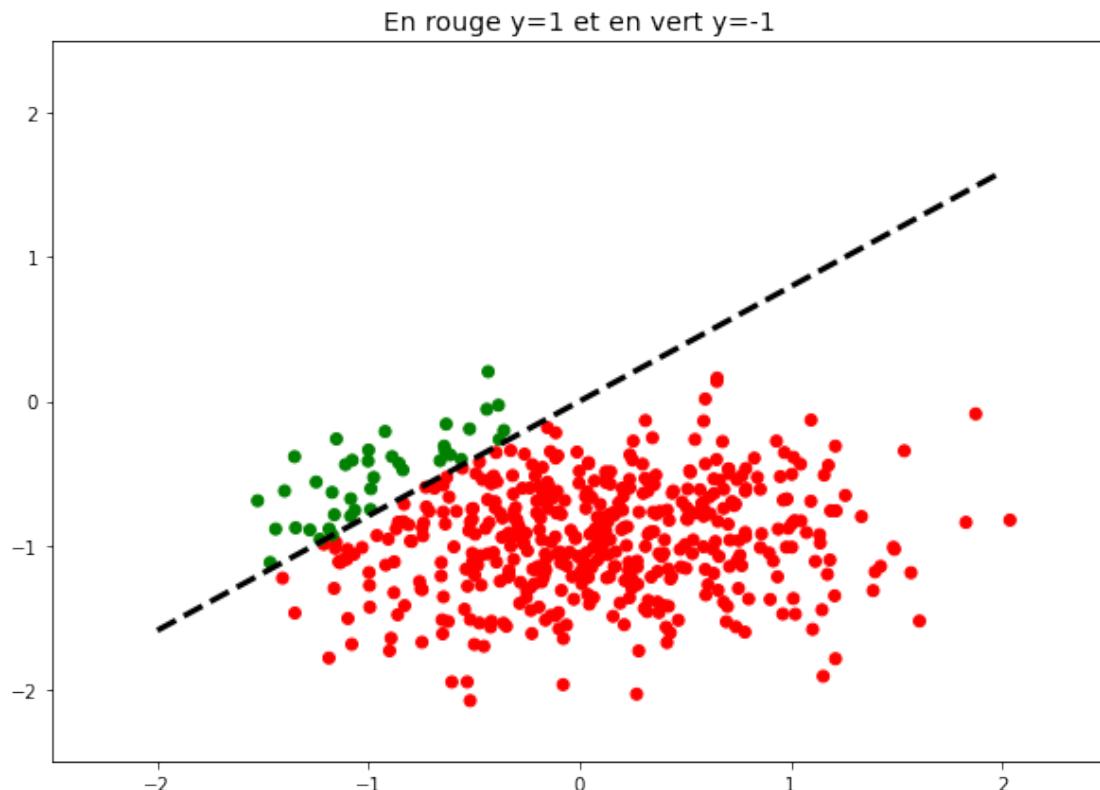
Le vecteur normal estimé w* = [0.62275017 -0.78242074]

L'ecart ou l'erreur entre les deux vecteurs est de : 0.05348709624520736

Le taux de reussite de l'algorithme est de : 92.0 %

On remarque que notre estimation est moins bonne avec du bruit que précédemment avec un écart du taux de réussite notable d'environ 8 à 10%. On peut tracer le graphique similaire à la question 2 pour comparer les performances :

```
[142]: Z_bruit = []
for k in range(n):
    Z_bruit.append([X_bruit[k,:], Y_bruit[k]])
graph(Z_bruit,w_gradient_bruit, -2.5, 2.5, -2.5, 2.5, bruit=True)
```



Question 5

```
[2]: from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import pandas as pd

[92]: #On importe l'ensemble de données au format csv
df = pd.read_csv('sample_data/breast-cancer-wisconsin.data', sep = ',', header=None, na_values='?')

#On remplace les valeurs ? par 0
df = df.fillna(0)

X = np.array(df[df.columns[:10]].values)
Y = np.array(np.where(df[10]==4,1,-1))

#On normalise nos données
s = preprocessing.StandardScaler()
X = s.fit_transform(X)

#on ne peut pas utiliser notre fonction sample_w car on est plus dans R^2
w = np.array(np.random.uniform(-1,1,size=X.shape[1]))
w = w/np.linalg.norm(w)

w_gradient = stochastic_gradient(X, Y, w, 500)
```

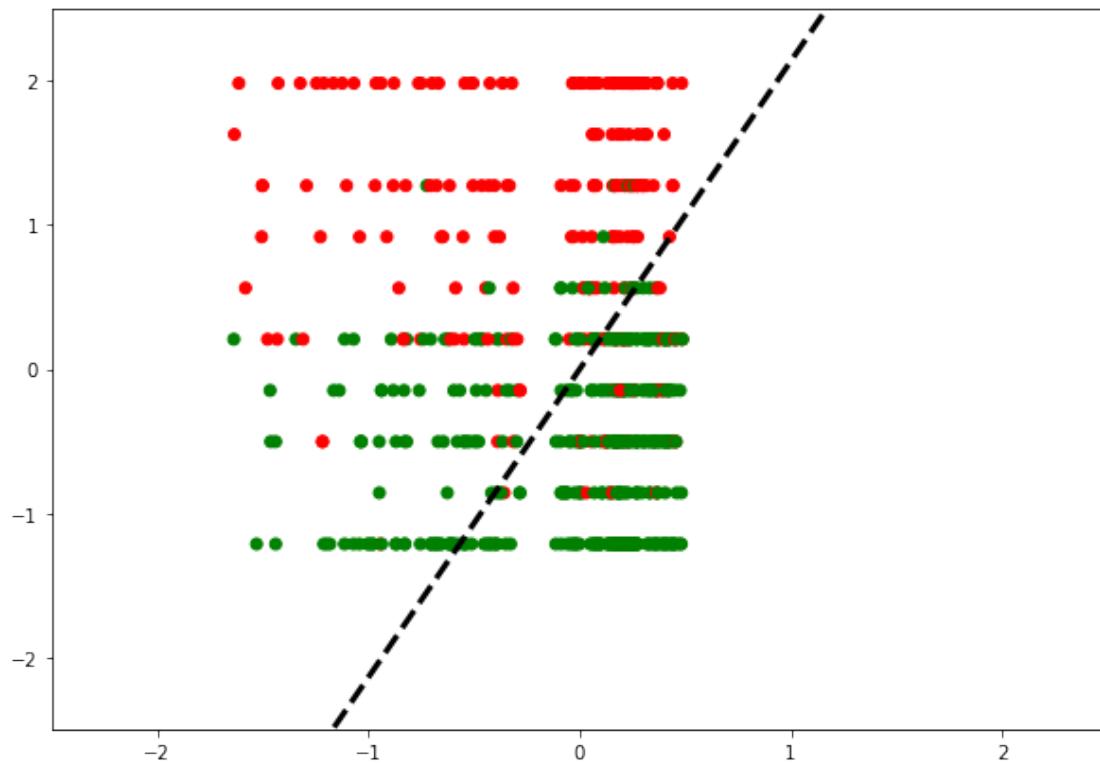
```
[93]: print("Le vecteur normal issu des données est : w = ", w)
print("Le vecteur normal estimé est w* = ", w_gradient)
print("Le taux de réussite de l'algorithme est de : ", performance(X, Y, w_gradient), "%")
```

```
Le vecteur normal issu des données est : w = [-0.26660721  0.4205747
-0.02876423  0.10686799 -0.39213633  0.39008646
 0.35500701  0.34010562  0.20464162  0.38764807]
Le vecteur normal estimé est w* = [ 1.57118512 -0.73484727  0.63322379
1.54405241 -0.83981118 -1.11291829
 1.21761006 -0.12960775 -0.15797188  0.18896266]
Le taux de réussite de l'algorithme est de : 71.67381974248927 %
```

Si on trace un graphique comme précédemment, on voit nettement la baisse d'efficacité de l'algorithme mais ceci est normal car les données ne sont jamais "parfaites". Cependant, 71% reste un bon taux de réussite.

```
[94]: Z_data = []
for k in range(len(Y)):
    Z_data.append([X[k,:], Y[k]])
```

```
graph(Z_data,w_gradient, -2.5, 2.5, -2.5, 2.5, bruit=True, legend=False)
```



```
[143]: !apt-get install texlive texlive-xetex texlive-latex-extra pandoc  
!pip install pypandoc
```

```
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
pandoc is already the newest version (1.19.2.4~dfsg-1build4).  
pandoc set to manually installed.  
The following package was automatically installed and is no longer required:  
  libnvidia-common-460  
Use 'apt autoremove' to remove it.  
The following additional packages will be installed:  
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre  
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common  
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1  
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern  
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest  
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5  
  rubygems-integration t1utils tex-common tex-gyre texlive-base  
  texlive-binaries texlive-fonts-recommended texlive-latex-base  
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
```