Probabilistic Graphical Models

Devoir maison 1 :

Othmane SAYEM
Pirashanth RATNAMOGAN
Promotion MVA 2018

Professeurs :
M. Francis Bach
M. Guillaume Obozinski

18 octobre 2017

Table des matières

Introduction

1 Apprendre dans des modèles de graphe discret

Soit M et K des entiers donnés, on cherche ici à trouver l'estimateur de maximum de vraisemblance de $\pi = (\pi_m)_{m\in\{1,..M\}}$ et de $\theta = (\theta_{mk})_{m\in\{1,..M\},k\in\{1,..,K\}}$ des paramètres qui définissent les variations des variables aléatoires discrète z et et x tel que :

$$p(z = m) = \pi_m \quad p(x = k|z = m) = \theta_{mk} \tag{1.1}$$

On suppose disposer de N observations $(x_1, z_1), (x_2, z_2), ...(x_N, z_N)$ qui permettent de calculer nos estimateurs.

1.1 L'estimateur de maximum de vraisemblance de π

Comme on l'a vu dans le cours, on peut définir une nouvelle variable $Z = (Z_1, Z_2, .., Z_K)$ déterministe connaissant les valeurs de z tel que l'on a

$$\{z = m\} \Leftrightarrow \{Z_m = 1, \forall l \in \{1, ..., M\}, tq\ l \neq m\ Z_l = 0\} \tag{1.2}$$

Et l'on a :

$$\forall m \in \ \{1, ..., M\}\ p(z = m) = p(Z_m = 1) = \pi_m \tag{1.3}$$

C'est à dire que Z suit une loi multinomiale de paramètre π. On reprend la démonstration du cours pour calculer l'estimateur de maximum de vraisemblance. L'expression de la log-vraisemblance pour la loi de Z est :

$$\begin{aligned} l_Z(\pi) &= \sum_{i=1}^{N} \log p(Z_i; \pi) \\ &= \sum_{i=1}^{N} \log \prod_{m=1}^{M} \pi_m^{Z_{im}} \\ &= \sum_{i=1}^{N} \sum_{m=1}^{M} Z_{im} \log \pi_m \\ &= \sum_{m=1}^{M} n_m \log \pi_m \end{aligned} \tag{1.4}$$

Où $n_m = \sum_{i=1}^{N} Z_{im}$ (le nombre d'observation de $Z_m = 1$). Le problème de maximisation de la log-vraisemblance se réecrit donc :

$$(\mathcal{P}) \begin{cases} \min\limits_{\pi} -l_Z(\pi) = & -\sum\limits_{m=1}^{M} n_m \log \pi_m \\ & \text{s.t.} \quad \mathbf{1}^T\pi = 1 \end{cases} \tag{1.5}$$

On a $\mathcal{P}$ qui revient à la minimisation du lagrangien définit par :

$$L(\pi, \lambda) = -\sum_{m=1}^{M} n_m \log \pi_m + \lambda(\sum_{m=1}^{M} \pi_m - 1) \tag{1.6}$$

Puisque pour tout $m \in \{1, ..., M\}$, $n_m \geq 0$, on a la fonction objectif $-l_Z$ qui est convexe comme somme de fonctions convexes. L'existence de points admissibles $\pi_1, \pi_2, ..., \pi_M$, $\pi_m \geq 0$ vérifiant la contrainte $\mathbf{1}^T\pi = 1$ est trivial. Les contraintes sont qualifiées.
On a $L(\pi, \lambda)$ qui est convexe par rapport à π, le minimum est donc un point stationnaire de la fonction. Ce qui conduit à

$$\begin{aligned} \frac{\partial L}{\partial \pi_m} &= -\frac{n_m}{\pi_m} + \lambda, \qquad && \forall m \in \{1, ..., M\} \\ \pi_m &= \frac{n_m}{\lambda} && \forall m \in \{1, ..., M\} \end{aligned} \tag{1.7}$$

La contrainte $\sum_{m=1}^{M} \pi_m = 1$ donne immédiatement que $\lambda = \sum_{m=1}^{M} n_m = N$. On en déduit l'estimateur de π_m :

$$\boxed{\hat{\pi}_m = \frac{n_m}{N} \quad \forall m \in \{1, ..., M\}} \tag{1.8}$$

1.2 L'estimateur de maximum de vraisemblance de θ

Pour estimer le paramètre θ, on se donne un échantillon de n observations (x_i, z_i). Connaissant l'estimateur de maximum de vraisemblance de z, on veut maximiser la vraisemblance de x sachant z. On a :

$$\begin{aligned} p(x|z, \theta) &= p(x|z, \theta) \\ &= \prod_{m=1}^{M} \prod_{k=1}^{K} \theta_{m,k}^{x_{ik} z_{im}} \pi_m^{z_{im}} \end{aligned}$$

Où l'on a noté $x_{ik} = 1$ si $x_i = k$ et 0 sinon et $z_{im} = 1$ si $z_i = m$ et 0 sinon. En calculant la log-vraisemblance de l'ensemble des échantillons, on cherche donc à maximiser :

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{N} [\sum_{m=1}^{M} (z_{im} \log(\pi_m) + \sum_{k=1}^{K} x_{ik} z_{im} \log(\theta_{k,m}))] \\ &= \sum_{m=1}^{M} n_m \log(\pi_m) + \sum_{m=1}^{M} \sum_{k=1}^{K} n'_{mk} \log(\theta_{k,m}) \end{aligned}$$

Où on a définit $n_m = \sum_{i=1}^{N} z_{im}$ *ainsi que* $n'_{mk} = \sum_{i=1}^{N} z_{im} x_{ik}$

En utilisant la relation de Bayes $P(x_i = k) = \sum_m P(x_i = k|z_i = m) P(z_i = m)$,

on a donc :

$$\begin{aligned} 1 &= \sum_{k=1}^{K} P(x_i = k) \\ &= \sum_{k=1}^{K} \sum_{m} P(x_i = k | z_i = m) P(z_i = m) \\ &= \sum_{k=1}^{K} \sum_{m=1}^{M} \pi_m \theta_{m,k} \end{aligned}$$

Maximiser la vraisemblance revient donc à résoudre le problème d'optimisation suivant :

$$(\mathcal{P}_2) \begin{cases} \min\limits_{\pi} -l(\theta) = -\sum\limits_{m=1}^{M} n_m \log(\pi_m) - \sum\limits_{m=1}^{M} \sum\limits_{k=1}^{K} n'_{mk} \log(\theta_{k,m}) \\ \text{s.t.} \quad \sum\limits_{k=1}^{K} \sum\limits_{m=1}^{M} \pi_m \theta_{m,k} = 1 \end{cases} \tag{1.9}$$

Le lagragien associé au problème d'optimisation $(\mathcal{P}_2)$ s'écrit ainsi :

$$L(\theta, \lambda_2) = -\sum_{m=1}^{M} n_m \log(\pi_m) - \sum_{m=1}^{M} \sum_{k=1}^{K} n'_{mk} \log(\theta_{k,m}) + \lambda_2 (\sum_{m=1}^{M} \pi_m \theta_{m,k} - 1)$$

la fonction de vraisemblance $l(\theta)$ étant convexe (somme de fonctions convexes), le lagragien est donc convexe par rapport à θ. Les contraintes sont comme pour le problème $\mathcal{P}$ également qualifiées. Soit $(k, m) \in \{1..K\}\{1..M\}$:

$$\frac{\partial L}{\partial \theta_{mk}} = -\frac{n'_{mk}}{\theta_{mk}} + \lambda_2 \pi_m = 0$$

La solution optimale s'écrit donc :

$$\hat{\theta_{mk}} = \frac{n'_{mk}}{\lambda_2 \pi_m}$$

Or on sait que :

$$\sum_{k=1}^{K} \sum_{m=1}^{M} \pi_m \theta_{m,k} = 1 \qquad \sum_{k=1}^{K} \sum_{m=1}^{M} n'_{mk} = n$$

On en déduit donc que $\boxed{\lambda_2 = n}$. Ainsi, en reprenant l'estimateur de π calculé au chapitre précédent, on déduit l'estimateur de maximum de vraisemblance de θ :

$$\boxed{\hat{\theta_{mk}} = \frac{n'_{mk}}{n\hat{\pi_m}} = \frac{n'_{mk}}{n_m} \quad \forall k \times m \in \{0, ..., K\} \times \{0, ..., M\}}$$

2 Classification Linéaire

2.1 Présentation de la problématique

On dispose d'ensemble de données de couple (x_n, y_n) avec $x_n \in \mathbb{R}^2$ et $y_n \in \{0, 1\}$ répartit dans 3 ensembles de données A,B et C séparés en données pour la phase d'entrainement et données pour la phase de test. Le but de l'exercice est de retrouver grâce à des classifieurs linéaires simples le label $y \in \{0, 1\}$ associé à des échantillons $x \in \mathbb{R}^2$ sachant que pour un label y donnée les x seront répartis selon une même loi.

2.2 Generative model (LDA)

On suppose dans ce modèle que les données (x,y) vérifient :

$$y \sim Bernoulli(\pi) \qquad x|\,\{y = i\} \sim Normal(\mu_i, \Sigma)$$

2.2.1 Estimateur de maximum de vraisemblance

Soit $D = \{(x_1, y_1), (x_1, y_2)..., (x_n, y_n)\}$ l'ensemble des observations. La probabilité d'observation du couple (x_i, y_i) s'écrit alors :

$$p((x_i, y_i); \pi, \mu_0, \mu_1) = [\pi N(x_i; \mu_1, \Sigma)]^{y_i}[(1 - \pi)N(x_i; \mu_0, \Sigma)]^{1-y_i}$$

Ainsi la log-vraissemblance pour l'ensemble des données observées va donc s'écrire sous la forme :

$$\begin{aligned} L(\pi, \mu_0, \mu_1, \Sigma) &= \log \prod_i p(x_i, y_i) \\ &= \sum_{i=1}^{n} y_i \log(\pi) + y_i \log(N(x_i; \mu_1)) + (1 - y_i) \log(1 - \pi) + (1 - y_i) \log(N(x_i; \mu_0)) \end{aligned}$$

En explicitant la densité de la loi normale on sait que :

$$\log(N(x_i; \mu_1)) = -\frac{1}{2} \log(2\pi det(\Sigma)) - \frac{1}{2}(x_i - \mu_1)^T \Sigma^{-1}(x_i - \mu_1)$$

Puisque la fonction de log-vraisemblance est concave comme somme de fonctions concave (que l'on a vu en classe), on sait que le maximum est atteint en un point stationnaire. En dérivant par L rapport au parametre μ_1 on trouve :

$$\frac{\partial L}{\partial \mu_1} = \sum_{i=1}^{n} y_i \Sigma^{-1}(x_i - \mu_1) = 0$$

d'où l'estimateur de maximum de vraisemblance pour μ_1 et μ_0 :

$$\boxed{\hat{\mu_1} = \frac{\sum_i y_i x_i}{\sum_i y_i}}$$

$$\boxed{\hat{\mu_0} = \frac{\sum_i (1 - y_i) x_i}{\sum_i (1 - y_i)}}$$

En dérivant par rapport au paramètre de Bernoulli π, on retrouve le résultat du cours :

$$\boxed{\hat{\pi} = \frac{\sum_i y_i}{N}}$$

On cherche l'estimateur de la matrice de covariance Σ. On note $n_1 = \sum_i y_i$ et $n_0 = n - n_1$. On dérive la partie de L qui dépend de Σ :

$$\begin{aligned} f(\Sigma) &= -\frac{n}{2}\log(det(\Sigma)) - \frac{1}{2}\sum_i y_i(x_i-\mu_1)^T\Sigma^{-1}(x_i-\mu_1) - \frac{1}{2}\sum_i (1-y_i)(x_i-\mu_0)^T\Sigma^{-1}(x_i-\mu_0) \\ &= -\frac{n}{2}\log(det(\Sigma)) - \mathrm{Tr}(Y_1^T\Sigma^{-1}Y_1) - \mathrm{Tr}(Y_0^T\Sigma^{-1}Y_0) \end{aligned}$$

Où l'on a défini $Y_1(i) = \sqrt{y_i}(x_i-\mu_1)$ et $Y_0(i) = \sqrt{(1-y_i)}(x_i-\mu_0)$. On dérive par rapport à $A = \Sigma^{-1}$:

$$\begin{aligned} \frac{\partial L}{\partial A} &= \frac{\partial f}{\partial A} \\ &= n(A^{-1})^T - Y_1Y_1^T - Y_0Y_0^T \\ &= n\Sigma - Y_1Y_1^T - Y_0Y_0^T \end{aligned}$$

D'où l'EMV de la matrice de covariance, qui vérifie la condition d'optimalité $\delta_\Sigma(L) = 0$, est donné par la formule :

$$\boxed{\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}[y_i(x_i-\mu_1).(x_i-\mu_1)^T + (1-y_i).(x_i-\mu_0).(x_i-\mu_0)^T]}$$

2.2.2 La forme de la probabilité conditionnelle

On s'intéresse à la forme de la loi conditionnelle $p(y=1|x)$:

$$\begin{aligned} p(y=1|x) &= \frac{p(y=1,x)}{p(x)} \\ &= \frac{1}{1+\frac{p(y=0,x)}{p(y=1,x)}} \\ &= \frac{1}{1+\frac{1-\pi}{\pi}\frac{N(\mu_0,\Sigma)}{N(\mu_1,\Sigma)}} \end{aligned}$$

On explicite la forme de la densité de loi gaussienne $N(\mu,\Sigma)$:

$$\frac{N(\mu_0,\Sigma)}{N(\mu_1,\Sigma)} = \exp(-0.5(\mu_0^t\Sigma^{-1}\mu_0 - \mu_1^t\Sigma^{-1}\mu_1 + 2\mu_1^t\Sigma^{-1}x - 2\mu_0^t\Sigma^{-1}x)) \qquad (2.1)$$

En introduisant la fonction Sigmoid $w \to \sigma(w)$ présentée en cours, et utilisée pour la régression logistique, on peut réecrire la probabilité conditionnelle $p(y=1|x)$:

$$\begin{aligned} p(y=1|x) &= \sigma(c*(\mu_0^t\Sigma\mu_0 - \mu_1^t\Sigma\mu_1 + 2\mu_1^t\Sigma x - 2\mu_0^t\Sigma x)) \\ &= \sigma(a^T x + b) \end{aligned}$$

Où a, b et c sont des constantes qui dépendent de π, μ_1, μ_0 *and* Σ

$$\boxed{c = \exp[\tfrac{1}{2} + \log(\tfrac{1-\pi}{\pi})]}$$

$$\boxed{a = 2c(\mu_1^t\Sigma^{-1} - \mu_0^t\Sigma^{-1}) \qquad b = c(\mu_0^t\Sigma^{-1}\mu_0 - \mu_1^t\Sigma^{-1}\mu_1)}$$

Ainsi grâce aux hypothèses sur l'égalité des matrices de covariance pour les distrbutions données par $y = 0$ et $y = 1$, on a que dans le cadre de notre modèle les x données pour $p(y = 1|x) = const$ sont sur une même droite donnée en inversant l'équation donnée par 2.1.

2.3 Résultats numériques :

On trouve pour le jeu A :

$$\mu_0 = \begin{pmatrix} 2.90 \\ -0.90 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -2.69 \\ 0.87 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 2.44 & -1.13 \\ -1.13 & 0.61 \end{pmatrix} \quad \pi = 0.33$$

On trouve pour le jeu B :

$$\mu_0 = \begin{pmatrix} 3.34 \\ -0.83 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -3.21 \\ 1.1 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 3.35 & -0.14 \\ -0.14 & 1.74 \end{pmatrix} \quad \pi = 0.5$$

On trouve pour le jeu C :

$$\mu_0 = \begin{pmatrix} 2.79 \\ -0.84 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -2.94 \\ -0.96 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 2.88 & -0.63 \\ -0.63 & 5.2 \end{pmatrix} \quad \pi = 0.625$$

2.3.1 Représentation graphique et analyse des résultats :

On trace ci-dessous les différents jeux de données d'entrainement ainsi que la frontière $p(y = 1|x) = 0.5$ de notre classifieur.

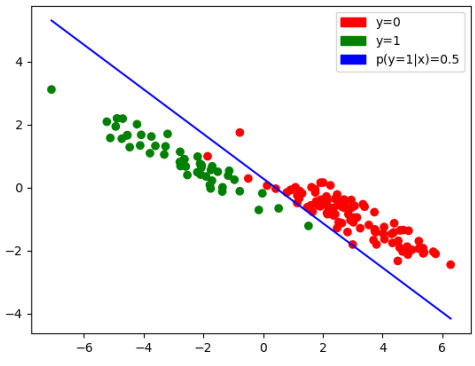

Figure 1: A Train, Generative Model

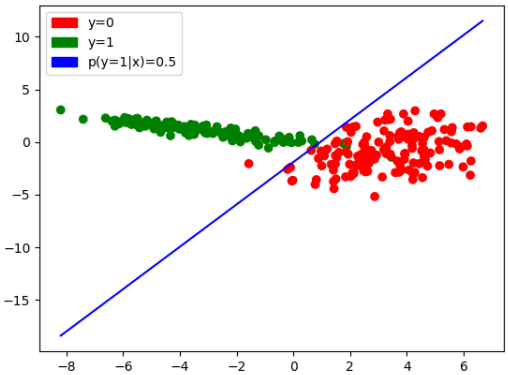

Figure 2: B Train, Generative Model

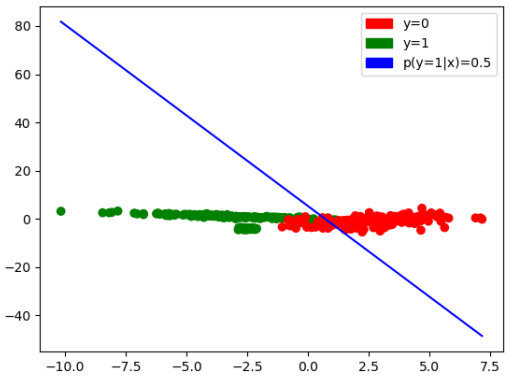

Figure 3: C Train,Generative Model

- Dans le jeu A, les hypothèses du modèle LDA semblent vérifiées après analyse des données : la distribution des données pour les labels 0 et 1 semblent être de moyenne différente mais de covariance similaire. Le modèle choisi semblent donc bien adapté ici et la frontière calculée ici sépare bien les deux distributions de label différent.

- Pour les données B également, les deux groupes ($y = 0$ et $y = 1$) apparaissent bien comme deux gaussiennes , mais dont les covariances sont différentes selon la distribution. Le modèle est donc peu judicieux ici et la frontière tracée ne semble pas optimale.

- En ce qui concerne les données C, les deux groupes ($y = 0$ et $y = 1$) ne semblent pas suivre une loi gaussienne. Ici aussi les hypothèses du modèle ne sont pas vérifiés et la frontière calculée n'est pas optimale.

2.4 Régression logistique

2.4.1 Le modèle

On implémente le modèle de régression logistique à la fonction affine $w^T x + b$. On suppose que, pour une observation $X = x$, $Y = 1|X = x$ suit une loi bernoulli de paramètre $\theta = \sigma(w^T x + b)$. Pour simplifier le problème, on considère la variable $x_1 = (x, 1)$ et $u = (w, b)$ qui nous permettent de se place dans le cadre du cours. On cherchera donc à estimer $\theta = \sigma(u^T x_1)$.

2.4.2 Maximum de vraisemblance :

On suppose qu'on a n échantillons $(x_i, y_i)_{i\in[n]}$. En reprenant les résultats du cours, on a montré que le calcul de (w, b) s'effectue de manière itérative, en appliquant jusqu'à convergence l'algorithme de Newton-Ralphson :

$$u^{n+1} \leftarrow u^n + (X^T DX)^{-1} X^T (Y - \eta)$$

Avec : $\eta_i = \sigma(u_n^T x_i)$ et $D = diag(\eta_i(1 - \eta_i))$

Au cas où la matrice hessienne $X^T DX$ n'est pas inversible, l'algorithme de Newton diverge. Pour éviter ce cas, l'ajout d'un terme de régularisation λ s'impose pour avoir la converge. On calculera donc la hessienne et le gradient de la fonction objective : $w \to l(w) + \lambda||w||^2$ où l'on a noté l la log-vraisemblance pour le modèle et les échantillons données et λ un paramètre déterminé. Le calcul du gradient et de la Hessienne et du gradient ne change que très peu en rajoutant ce terme de régularisation et on aura l'algorithme qui devient :

$$u^{n+1} \leftarrow u^n + (X^T DX + \lambda I)^{-1} X^T (Y - \eta - \lambda u^n)$$

Lors du calcul on fera également attention au choix du point initial qui s'avère être déterminant pour la convergence de l'algorithme de Newton.

2.5 Résultats numériques :

On trouve pour le jeu A (régularisation avec lambda=0.1) :

$$w = \begin{pmatrix} -4.1 \\ -5.72 \end{pmatrix} \quad b = -0.54$$

On trouve pour le jeu B :

$$w = \begin{pmatrix} -1.7 \\ 1.023 \end{pmatrix} \quad b = 1.34$$

On trouve pour le jeu C :

$$w = \begin{pmatrix} -2.20 \\ 0.71 \end{pmatrix} \quad b = 0.96$$

2.5.1 Représentation graphique et analyse des résultats :

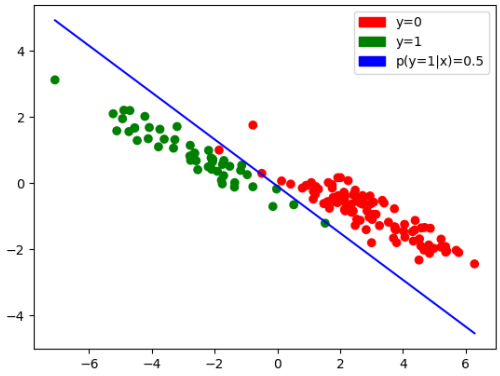

Figure 4: A Train, Logistic regression

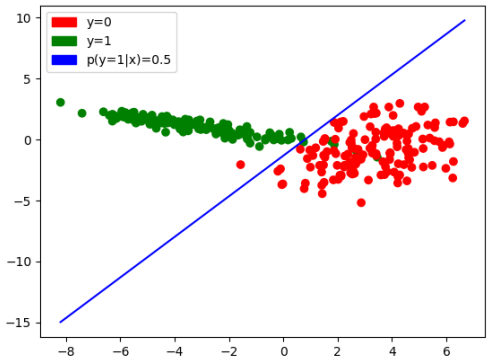

Figure 5: B Train, Logistic regression

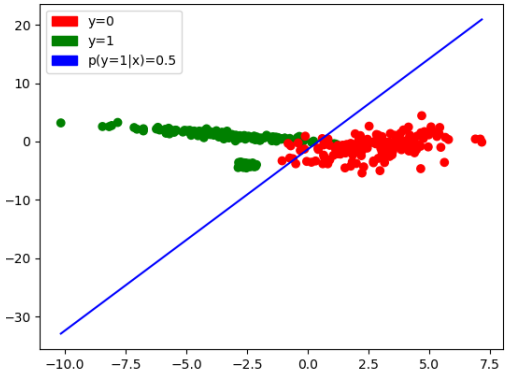

Figure 6: C Train, Logistic regression

On a vu que le modèle LDA est un modèle qui dérive de la régression logistique où l'on suppose par avance que les données sont gaussiennes, donnant ainsi une forme particulière aux coefficients que l'on peut apprendre. La régression logistique parvient donc à de bons résultat sur la phase d'apprentissage du jeu A, mais les résultats sur le B et C semblent bonnes aussi vu que le modèle de la régression logistique est assez générale.

2.6 La régression linéaire

2.6.1 Le modèle

La régression linéaire est un modèle simple qui dans notre cadre permet de modéliser simplement la probabilité pour un échantillon x d'appartenance à la classe 1 par

$$p(y = 1|x) = w^T x + b \quad w \in \mathbb{R}^2 \quad b \in \mathbb{R} \tag{2.2}$$

C'est l'expression la plus commune de la régression linéaire que nous avons implémenté. Néanmoins en cours nous avons également vu la version probabilistique donnée par :

$$y|x \sim \mathcal{N}(w^T x + b, \sigma^2) \tag{2.3}$$

Et l'on a donc :

$$p(y_i|x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-1}{2} \frac{(y_i - w^T x_i - b)^2}{\sigma^2}) \tag{2.4}$$

Evidémment, évaluer des fonctions de distribution continues en des points données ainsi qu'approximer des probabilités par des fonctions affines n'a que peu de sens, cela est dû à la non parfaite adéquation du modèle de régression linéaire au clustering. Nous travaillerons tout de même avec ces hypothèses.

2.6.2 La détermination des paramètres maximisant la vraisemblance

La méthode pour déterminer les coefficients dans la régression linéaire est classique. On suppose comme d'habitude que l'on a n échantillons $(x_1, ...x_n)$ ainsi que les labels associés $(y_1, y_2, ...y_n)$. On considère la matrice X tel que la ligne i contient la description normalisée de l'échantillon x_i, et le vecteur $y = (y_i)_{i \in \{1..n\}}$. On est dans le cadre exact du cours (ou l'on a rajouté la ligne de 1), où l'on a montré que le paramètre maximisant la vraisemblance $\hat{w}$ (lorsque $X^T X$ est inversible) est donné par l'équation.

$$\boxed{\hat{w} = (X^T X)^{-1} X^T y \quad \hat{\sigma}^2 = \frac{(y_i - w^T x - b)^2}{n}} \tag{2.5}$$

2.7 Résultats numériques :

On trouve pour le jeu A :

$$w = \begin{pmatrix} -0.26 \\ -0.37 \end{pmatrix} \quad b = 0.49 \quad (\sigma = 0.20)$$

On trouve pour le jeu B :

$$w = \begin{pmatrix} -0.10 \\ 0.05 \end{pmatrix} \quad b = 0.5 \quad (\sigma = 0.23)$$

On trouve pour le jeu C :

$$w = \begin{pmatrix} -0.13 \\ -0.02 \end{pmatrix} \quad b = 0.51 \quad (\sigma = 0.24)$$

2.7.1 Représentation graphique et analyse des résultats :

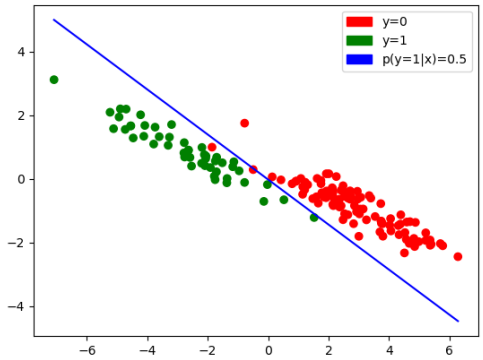

Figure 7: A Train, linear regression

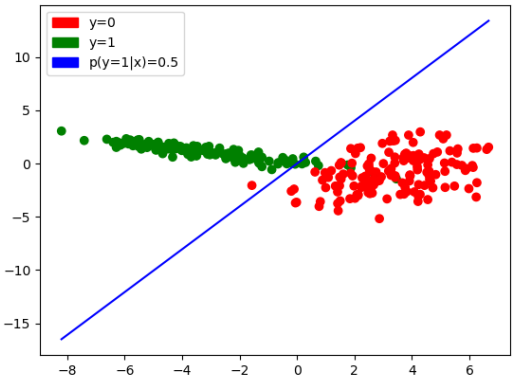

Figure 8: B Train, Linear regression

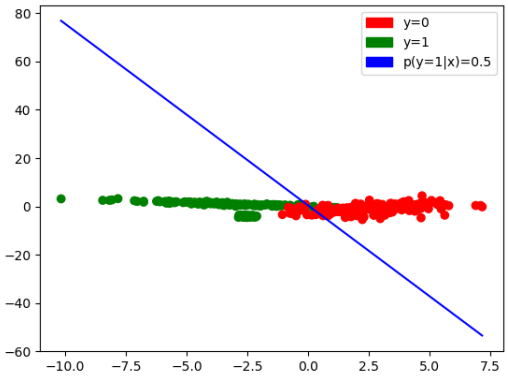

Figure 9: C Train, Linear regression

Dans cette partie nous avons effectué deux modélisations par régression linéaire (voir plus haut). Afin de ne pas surcharger le rapport nous n'affichons que les résultats pour le modèle avec le moins de taux d'erreurs (voir la comparaison en **2.12**). Ainsi nous ne traçons ici les graphes que pour la régression linéaire classique non probabilisé qui fournit les meilleurs résultats. Le modèle linéaire fournit des résultats honorables sur les trois jeux de données mais il n'est pas adapté à ce type de classification comparativement aux autres modèles étudiés.

2.8 QDA Model

On reprend le Generative Model, mais en relaxant cette fois-ci les hypothèses. On ne suppose plus que la matrice de covariance des deux classes est similaire :

$$y \sim Bernoulli(\pi) \qquad x|\{y=i\} \sim Normal(\mu_i, \Sigma_i)$$

2.8.1 Estimateurs du maximum de vraisemblance

Comme démontré dans le chapitre du Generative Model, l'estimateur du maximum de vraissemblance pour les paramètres π, μ_0, μ_1 restent les mêmes et s'écrivent sous la forme :

$$\boxed{\hat{\pi} = \frac{\sum_i y_i}{N}}$$

$$\boxed{\hat{\mu_1} = \frac{\sum_i y_i x_i}{\sum_i y_i}}$$

$$\boxed{\hat{\mu_0} = \frac{\sum_i (1-y_i) x_i}{\sum_i (1-y_i)}}$$

Pour le calcul de l'EMV des matrices de covariance Σ_1 et Σ_0, on le déduit du calcul de gradient déjà effectué dans le chapitre de Generative Model également :

$$\boxed{\hat{\Sigma_1} = \frac{1}{n_1} \sum_{i=1}^{n} [y_i (x_i - \mu_1).(x_i - \mu_1)^T]}$$

$$\boxed{\hat{\Sigma_0} = \frac{1}{n_0} \sum_{i=1}^{n} [(1-y_i)(x_i - \mu_0).(x_i - \mu_0)^T]}$$

- Concernant la représentation graphique de la probabilité conditionnelle $p(y = 1|x)$, les matrices de covariances étants différentes, la probabilité s'écrira sous la forme :

$$p(y = 1|x) = \sigma(\alpha x^T x + \beta x + \gamma)$$

Où α, β et γ sont des constantes non nulles dépendants des paramètres estimés $\hat{\pi}, \hat{\Sigma_0}, \hat{\Sigma_1}, \hat{\mu_0}$ et $\hat{\mu_1}$ La courbe prendra ainsi la forme d'une conique, qui dépendra du jeu de données choisi.

2.9 Résultats numériques :

On trouve pour le jeu A :

$$\mu_0 = \begin{pmatrix} 2.90 \\ -0.90 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -2.69 \\ 0.87 \end{pmatrix} \quad \Sigma_0 = \begin{pmatrix} 2.31 & -1.05 \\ -1.05 & 0.58 \end{pmatrix} \quad \Sigma_1 = \begin{pmatrix} 2.70 & -1.3 \\ -1.3 & 0.70 \end{pmatrix} \quad \pi = 0.33$$

On trouve pour le jeu B :

$$\mu_0 = \begin{pmatrix} 3.34 \\ -0.83 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -3.21 \\ 1.1 \end{pmatrix} \quad \Sigma_0 = \begin{pmatrix} 2.54 & 1.06 \\ 1.06 & 2.96 \end{pmatrix} \quad \Sigma_1 = \begin{pmatrix} 4.15 & -1.33 \\ -1.33 & 0.52 \end{pmatrix} \quad \pi = 0.5$$

On trouve pour le jeu C :

$$\mu_0 = \begin{pmatrix} 2.79 \\ -0.84 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -2.94 \\ -0.96 \end{pmatrix} \quad \Sigma_0 = \begin{pmatrix} 2.90 & 1.25 \\ 1.25 & 2.92 \end{pmatrix} \quad \Sigma_1 = \begin{pmatrix} 2.87 & -1.76 \\ -1.76 & 6.56 \end{pmatrix} \quad \pi = 0.625$$

2.9.1 Représentation graphique et analyse des résultats :

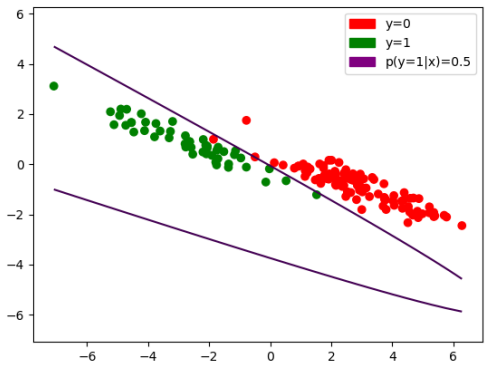

Figure 10: A Train, QDA

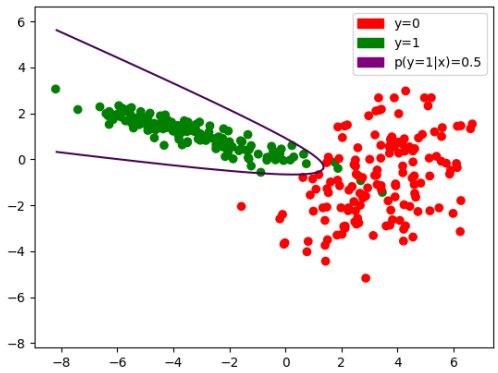

Figure 11: B Train, QDA

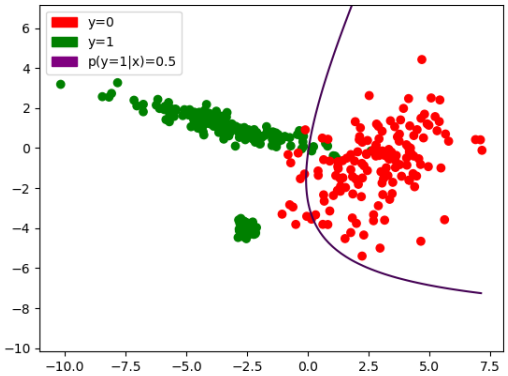

Figure 12: C Train, QDA

2.10 L'implémentation des différents classifieurs

On crée des classes permettant de représenter chacun des classifieurs. Chaque classe représentant un classifieur est contenu dans un fichier à son nom. Chaque classe possède des attributs simples `coef` (représentant les paramètres estimées) ainsi que des méthodes simples dont notamment `fit` qui permet d'estimer les coefficient de la régression ou `predict` qui permet d'estimer $p(y = 1|x)$ sachant le modèle que l'on a pu estimer grâce aux données d'apprentissage.
On trouve dans le fichier **Utils.py** les fonctionalités nécessaire à plusieurs regresseur. On y trouve notamment la fonction permettant de tracer les échantillons ainsi que la frontière $p(y = 1|x) = 0.5$.
On trouve la comparaison finale de tous les modèles crées dans le fichier **Compare-all-models.py**.

2.11 Résultats

2.11.1 Calcul de l'erreur

Pour ce premier TP on utilise une métrique simple mesurant le taux d'erreur dans notre prédiction sur l'ensemble test de N éléments $(x_i)_{i\in\{1..N\}}$. On suppose avoir notre vecteur de vrais résultats donnés par y^{true} et notre vecteur de prédiction donné par y^{pred} tq :

$$y_i^{pred} = \mathbb{1}_{f(x_i)\geq 0.5} \quad \forall i \in \{1..N\} \tag{2.6}$$

Où f correspond au classifieur que l'on aura préalablement appris sur une base d'apprentissage.

$$E = \sum_{i=1}^{N} \frac{\mathbb{1}_{y_i^{pred} \neq y_i^{true}}}{N} \tag{2.7}$$

2.11.2 Pour la base A

On représente ci-dessous nos échantillons pour la base de test avec des couleurs associées avec leur label qui est connu. On affiche de plus la frontière $p(y = 1|x) = 0.5$ donné par le modèle. Idéalement la frontière doit parfaitement séparer les échantillons qui ont un label 1 de ceux qui ont un label 0.

Analyse : Comme on l'a dit précédemment es échantillons pour la base semblent distribuées selon deux lois normales de moyenne dépendant du label de l'échantillon mais de matrice de covariance identique. On est dans le cadre où le Generative Model peut performer puisque les hypothèse du modèle sont respectés. Il apparait que l'erreur finale du Generative Model est assez proche de celle de tous les autres modèles. Sur la partie entrainement le Generative Model se fait distancer par d'autres modèles (QDA et Régression logistique) mais au final les 4 modèles sont globalement équivalents.

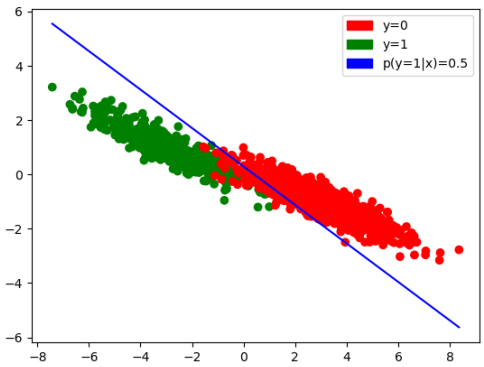

Figure 13: A Test, Generative Model

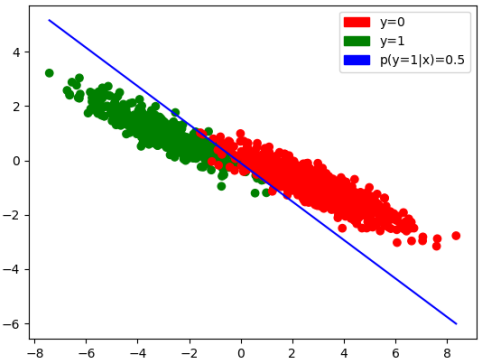

Figure 14: A Test, Logistic Regression

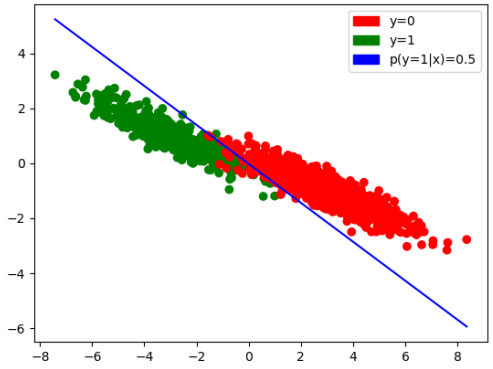

Figure 15: A Test, Linear Regression

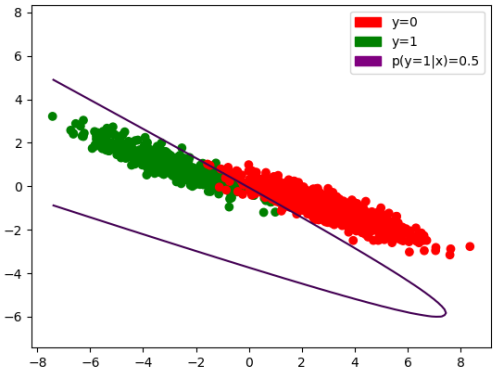

Figure 16: A Test, QDA Model

2.11.3 Pour la base B

On représente ci-dessous comme pour la base A nos échantillons pour la base de test avec des couleurs associées avec leur label qui est connu. On affiche de plus la frontière $p(y = 1|x) = 0.5$ donné par le modèle. Idéalement la frontière doit parfaitement séparer les échantillons qui ont un label 1 de ceux qui ont un label 0.

Analyse sur la base de test : Comme on a pu le dire précédemment Les échantillons pour cette base semblent distribuées selon deux lois normales de moyenne et de matrice de covariance dépendant du label de l'échantillon. Le modèle le plus adapté semble donc être le QDA qui est un modèle supposant justement ces hypothèses. On voit qu'il fournit les meilleurs résultats sur ce jeu de données, en particulier bien meilleurs que ceux du Generative Model ou ceux de la Régréssion Linéire. On voit que la régression logistique fournit des résultats semblable au QDA Model sur la phase d'apprentissage. Néanmoins puisque celle-ci a des hypothèses et des contraintes en moins elle ne performe pas aussi bien que QDA dans la phase de test (overfitting avec la base d'apprentissage).
Ici le QDA Model est clairement le modèle qu'il faut utiliser.

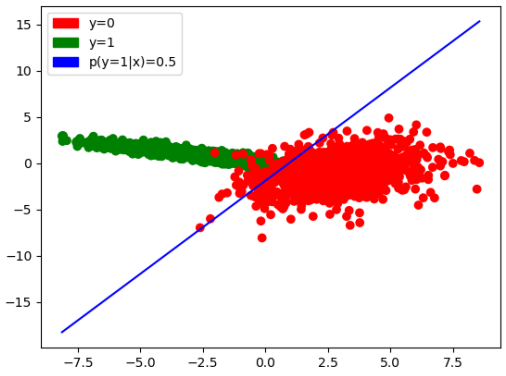

Figure 17: B Test, Generative Model

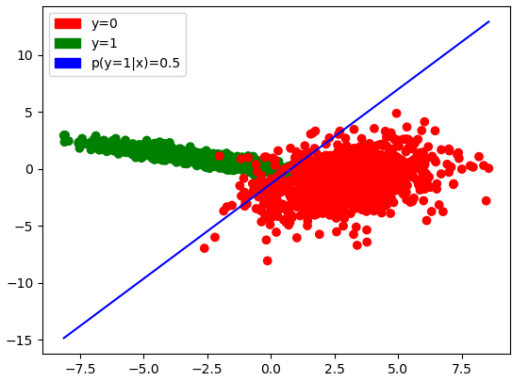

Figure 18: B Test, Logistic Regression

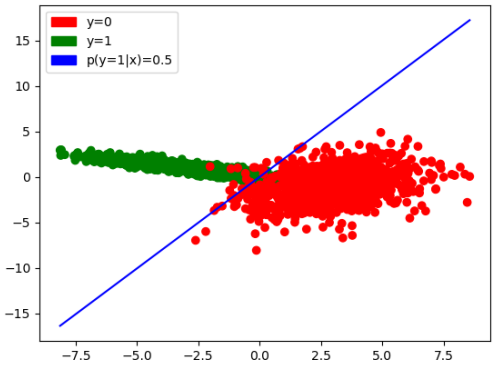

Figure 19: B Test, Linear Regression

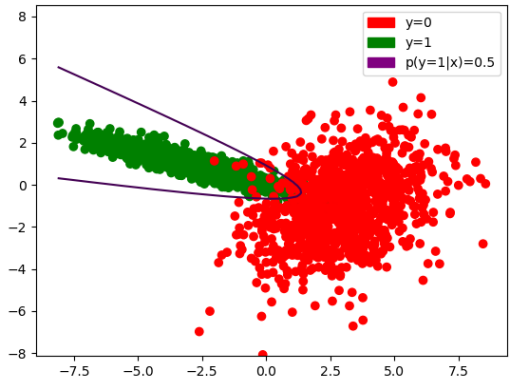

Figure 20: B Test, QDA Model

2.11.4 Pour la base C

On représente ci-dessous comme pour les bases A et B nos échantillons pour la base d'apprentissage et pour la base de test avec des couleurs associées avec leur label qui est connu. On affiche de plus la frontière $p(y = 1|x) = 0.5$ donné par le modèle. Idéalement la frontière doit parfaitement séparer les échantillons qui ont un label 1 de ceux qui ont un label 0.

Analyse : Dans la configuration de la base C on a une intersection entre les deux distributions d'échantillons qui est assez importante et on a une présence non négligeable d'outliers. On n'est pas dans un cadre gaussien parfait et les modèles le supposont auront donc nécessairement une erreur importante. Dans ce cadre il est plus judicieux d'utiliser la régression logistique qui fournit effectivement les meilleurs résultats à la fois dans la phase d'apprentissage et la phase de test. On voit que le Generative Model et le QDA Model ont des performances similaires dans ce cadre.

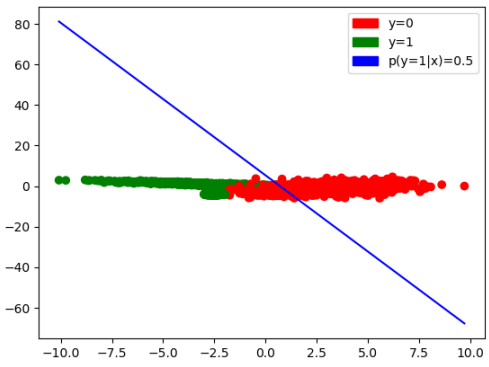

Figure 21: C Test, Generative Model

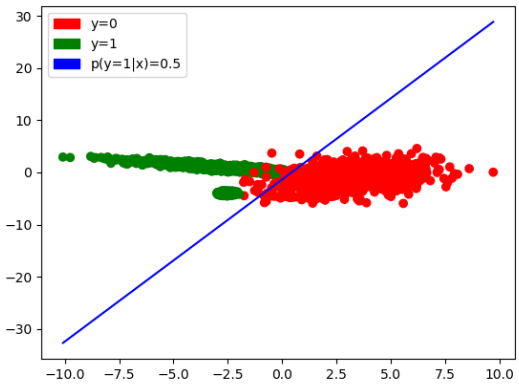

Figure 22: C Test, Logistic Regression

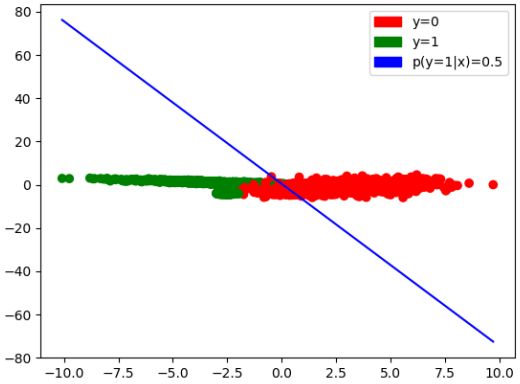

Figure 23: C Test, Linear Regression

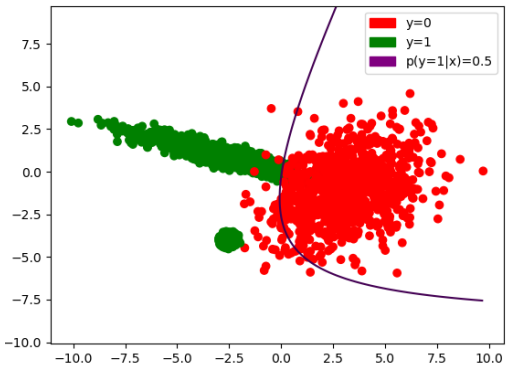

Figure 24: C Test, QDA Model

2.11.5 Comparaison des taux d'erreurs des modèles sur les différents jeu de données

On affiche ci-dessos les résultats des taux d'erreurs des différents modèles sur les jeux d'entrainements A,B et C et sur leurs jeux de test respectif A,B et C.

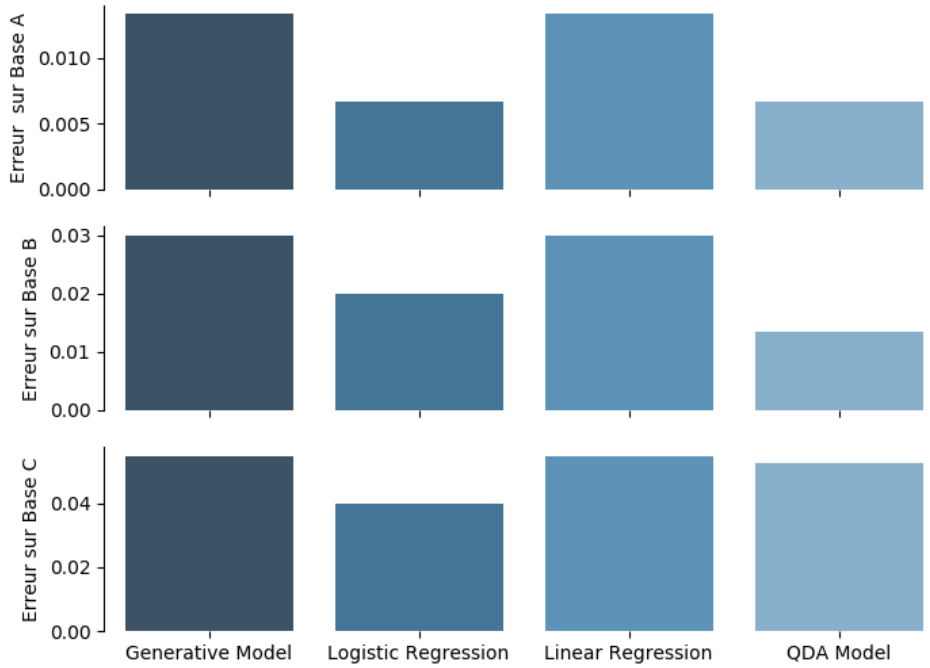

Figure 25: Comparaison des taux d'erreurs des modèles sur la base d'apprentissage

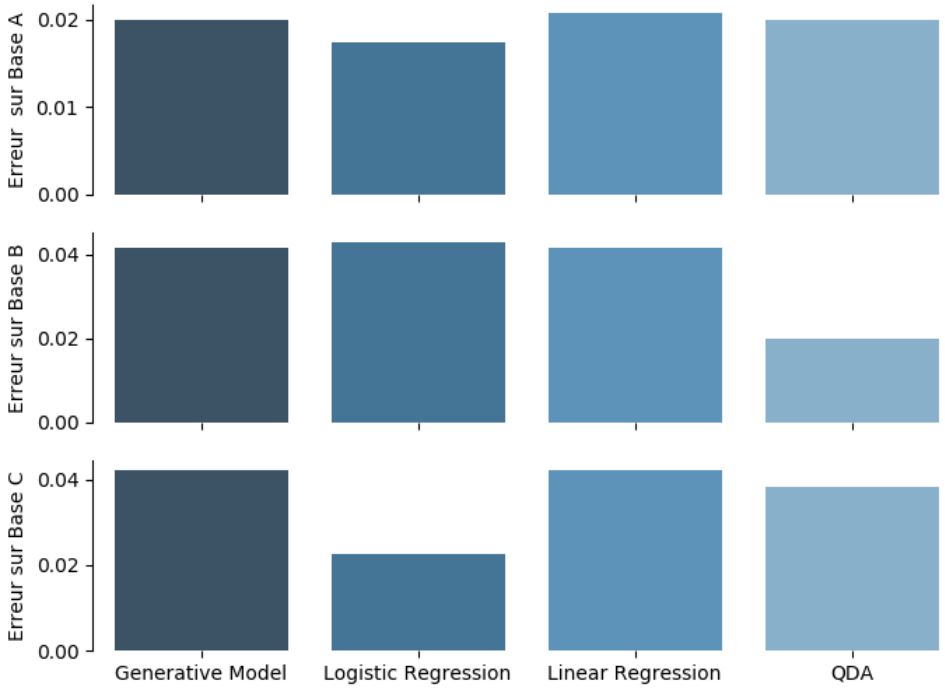

Figure 26: Comparaison des taux d'erreurs des modèles sur la base de test

Conformément à l'analyse faite dans les parties précédentes. Dans le cadre de l'exemple apprentissage/test A le Generative Model est le plus adapté puisqu'il est bien adapté à la distribution des données (2 distributions gaussiennes de moyennes différentes mais de covariance identique) ce qui lui permet de ne pas overfitter les données d'apprentissage contrairement au QDA Model et à la régression logistique (mais tous les classifieurs produisent des résultats équivalent). Dans le cadre de l'exemple apprentissage/test B le QDA Model est le plus adapté puisqu'il est bien adapté à la distribution des données (2 distributions gaussiennes différentes). Dans le cadre de la phase d'apprentissage/test C la regression logistique est la plus adaptée au vue de la présence d'outlier et de la non distribution lisse des données.

2.11.6 Conclusion sur les test

On a vu dans la partie théorique que le Generative Model est un cas particulier du QDA Model présupposant que les covariances des deux distributions normales utilisées dans le QDA ont la même covariance. De la même manière on a vu que le QDA Model est un cas qui se rapproche de la forme d'une régression logistique (mais avec un terme quadratique en plus).
Alors que naivement on pourrait penser qu'il faut toujours utiliser la régression logistique qui est le modèle le plus générale étudié ici, il apparait que le TP nous fait comprendre que ce n'est pas ce qu'il faut faire. En effet, puisqu'on est dans une logique d'estimation des paramètres il est judicieux d'utiliser un modèle présupposant justement la bonne distribution des échantillons d'entrée pour éviter l'overfitting (on le voit la régression logistique produit les meilleurs résultats sur

la base d'apprentissage mais pas sur la base de test).
Ainsi si l'on ne voulait utiliser qu'un modèle par jeu de données. Il faut observer les échantillons, comprendre quelle est la distribution (qqplot, ...) et utiliser dans nos exemples :

- Le Generative Model pour le jeu de données A (car les échantillons sont distribuées selon leur label par deux distributions normale avec covariance assez similaire et moyenne différente)
- Le QDA Model pour le jeu de données B car les échantillons sont distribuées selon leur label par deux distributions normale avec covariance et moyenne différente)
- La Régression Logistique pour le jeu de données C car les échantillons sont distribuées non homogènement et aucun de nos autres modèles n'est adapté.

2.12 Annexe : Comparaison de la régression linéaire probabilisée à la "classique"

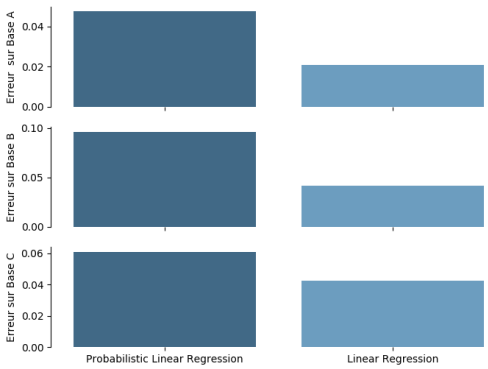

Figure 27: Comparaison des taux d'erreurs des modèles de régression linéaire sur la base d'apprentissage

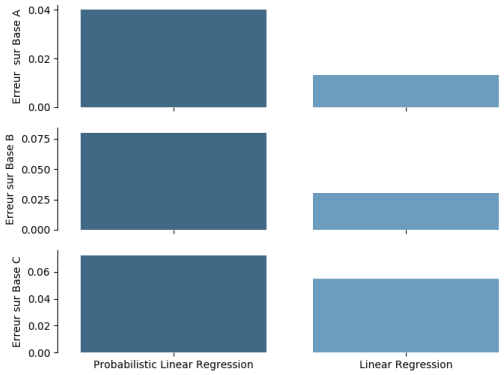

Figure 28: Comparaison des taux d'erreurs des modèles de régression linéaire sur la base de test

On voit que la régression linéaire "classique" performe mieux mais les deux modèles ne sont pas adaptés au clustering.