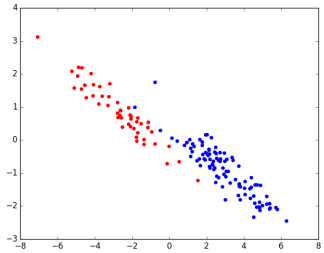
TD 7 : Analyse discriminante linéaire

Jules Kozolinsky

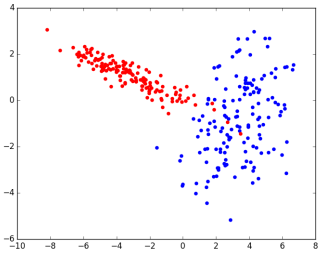
Modèles de régression

0.1

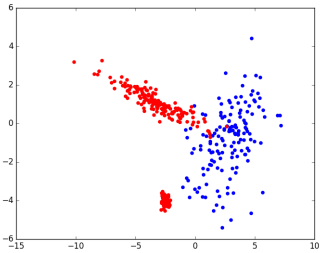
On représente les données avec la couleur bleu pour la classe 0 et rouge pour les points de la classe 1.



Ensemble d'apprentissage A



Ensemble d'apprentissage B

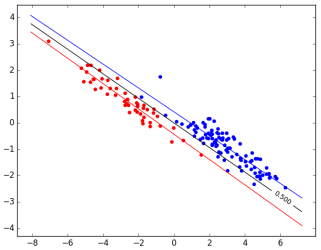


Ensemble d'apprentissage C

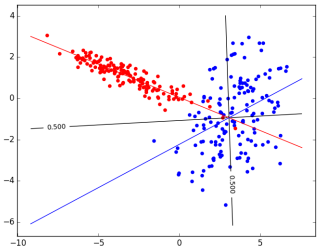
On constate que pour les points de l'ensemble d'apprentissage A la covariance semble être la même alors que pour les ensembles d'apprentissage B et C elle semble clairement différente. Enfin, on remarque que la classe 1 (en rouge) sur l'ensemble d'apprentissage C est bien le mélange de deux Gaussiennes.

0.2

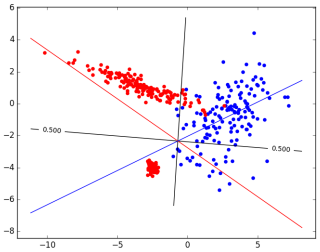
Pour chaque classe de l'ensemble d'entrainement, on effectue une régression linéaire. Ensuite pour chaque point considéré, on regarde quelle droite est plus proche de notre point et on classifie de la sorte.
On obtient les résultats visibles sur la figure ci-dessous (régressions linéaires de la couleur des classes et frontière de classification en noire).
On remarque que quand la covariance est faible alors, les résultats sont plutôt bons (cf données A) alors que quand la covariance est de grande, les données sont éparses et la régression linéaire est beaucoup moins précise (cf classe 0 pour données B et C). De plus la classe 1 des données C est un mélange de deux Gaussiennes, donc les résultats de la régression linéaire pour cette classe sont mauvais.
Ainsi, pour de faibles covariance et des distributions gaussiennes simples, la méthode de classification par plug-in de régression linéaire peut s'avérer efficace.



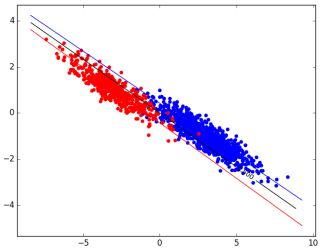
Ensemble d'apprentissage A
(erreur : 6.0%)



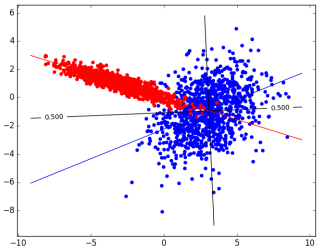
Ensemble d'apprentissage B
(erreur : 16.67%)



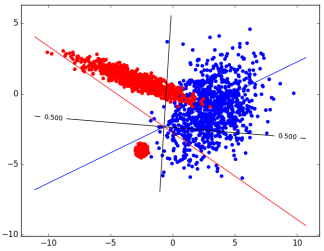
Ensemble d'apprentissage C
(erreur : 35.0%)



Ensemble de test A
(erreur : 12.34%)



Ensemble de test B
(erreur : 19.45%)



Ensemble de test C
(erreur : 43.4%)

0.3

Passons à la régression logistique. On maximise la log-vraisemblance, ce qui revient à minimiser l'opposé du risque empirique pour la perte logistique :

$$R_n(\omega) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\sigma(\omega^T x_i)) + (1-y_i)\log(1-\sigma(\omega^T x_i))$$

Ainsi, en dérivant :

$$\nabla_\omega R_n(\omega) = -\frac{1}{n}\sum_{i=1}^{n} x_i(y_i - \sigma(\omega^T x_i)) = -\frac{1}{n}X^T(y-\mu(\omega))$$

où $\mu_i(\omega) = \sigma(\omega^T x_i)$.
On cherche alors à minimiser la fonction $J(\omega) = R_n(\omega) + \lambda||\omega||_2^2$

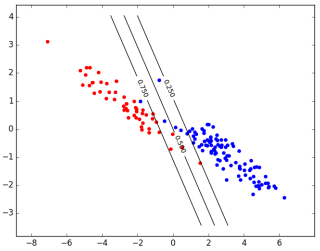
Résolution par méthode de gradient

On effectue une méthode de descente de gradient de pas γ.

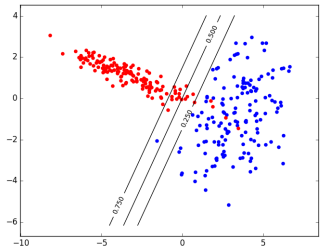
$$\omega_{t+1} = \omega_t - \gamma \nabla J(\omega_t)$$

Résultats :

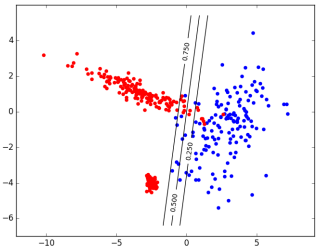
Trois algorithmes d'optimisation ont été implémentés : descente de gradient avec pas constant, avec pas variable (suivant la règle d'Armino) et l'algorithme des moindres carrés pondérés.
Pour les données suivantes, on a utilisé l'algorithme des moindres carrés pour la régression logistique avec les données A, et la descente de gradient pour les données B et C.



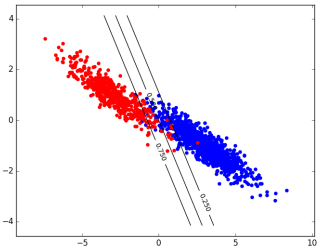
Ensemble d'apprentissage A
(erreur : 2.7%)



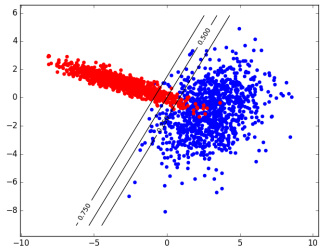
Ensemble d'apprentissage B
(erreur : 3.0%)



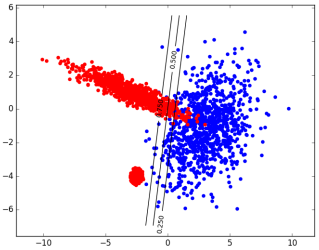
Ensemble d'apprentissage C
(erreur : 4.25%)



Ensemble de test A
(erreur : 2.27%)



Ensemble de test B
(erreur : 4.15%)



Ensemble de test C
(erreur : 2.77%)

Note : ici on représente la frontière de séparation (probabilité 0.5) ainsi que les lignes de niveau pour $P(Y = 1|X = x) = 0.25$ et $P(Y = 1|X = x) = 0.75$.

Commentaires :

Les résultats sont bien meilleurs que pour la méthode de classification par plug-in de la régression linéaire. Toutefois on remarque lorsque les données se "mélange", le modèle linéaire de f fait défaut. On pourrait penser aux méthodes à noyaux pour se débarrasser du modèle de f.

Discriminant de Fisher

On modélise les données par des nuages gaussiens sur $\mathbb{R}^p$ de distribution $\mathcal{N}(\mu_0, \Sigma_0)$ et $\mathcal{N}(\mu_1, \Sigma_1)$, respectivement pour les classes 0 et 1. Soit f_{μ_0,Σ_0} et f_{μ_1,Σ_1} les fonctions de densité des nuages

gaussiens, tels que :

$$f_{\mu_i,\Sigma_i} = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma_i)}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}$$

On considèrera seulement dans un second temps que $\Sigma_0 = \Sigma_1 = \Sigma$.

0.4 a) b)

Soit $A \in \mathbb{R}^n, i \in \{0,1\}$, on a :

$$\begin{aligned} P(X \in A \wedge Y = 1) = P(X \in A | Y = 1)P(Y = 1) &= \int_A f_{\mu_1,\Sigma_1}(x) P(Y = 1) d\mu(x) \\ &= \int_A \frac{f_{\mu_1,\Sigma_1}(x)}{f_X(x)} \pi dP(x) \end{aligned}$$

où f_X est la fonction de densité de la variable aléatoire X :

$$\begin{aligned} P(X \in A) &= P(Y = 0)P(X \in A | Y = 0) + P(Y = 1)P(X \in A | Y = 1) \\ &= (1-\pi) \int_A f_{\mu_0,\Sigma_0} + \pi \int_A f_{\mu_1,\Sigma_1} = \int_A (1-\pi) f_{\mu_0,\Sigma_0} + \pi f_{\mu_1,\Sigma_1} \\ &= \int_A f_X \end{aligned}$$

Ainsi, en définissant la probabilité conditionnelle comme un noyau de transition par rapport à X :

$$\begin{aligned} P(Y = 1 | X = x) = \frac{P(Y = 1 \wedge X = x)}{P(X = x)} &= \frac{\pi f_{\mu_1,\Sigma_1}}{(1-\pi) f_{\mu_0,\Sigma_0} + \pi f_{\mu_1,\Sigma_1}} \\ &= \frac{1}{1 + \frac{(1-\pi)}{\pi} \frac{f_{\mu_0,\Sigma_0}}{f_{\mu_1,\Sigma_1}}} \\ &= \sigma(f(x)) \end{aligned}$$

où $f(x) = \log(\frac{f_{\mu_1,\Sigma_1}}{f_{\mu_0,\Sigma_0}}) + \log(\frac{\pi}{1-\pi})$.
Le modèle conditionnel de $P(Y|X = x)$ induit par le modèle génératif choisi corresponds donc bien à un modèle probabiliste de classification linéaire de même forme que la régression logistique.

c)

On a $\pi = \frac{1}{n}\sum_{i=1}^{n} y_i$ et posant $\theta = (\mu_1, \mu_0, \Sigma_1, \Sigma_0) \in (\mathbb{R}^p, \mathbb{R}^p, \mathcal{S}_p(\mathbb{R}), \mathcal{S}_p(\mathbb{R}))$,

$$\begin{aligned} f_\theta(x) &= \log(\frac{f_{\mu_1,\Sigma_1}}{f_{\mu_0,\Sigma_0}}) + \log(\frac{\pi}{1-\pi}) \\ &= \log(\frac{\frac{1}{\sqrt{(2\pi)^p \det(\Sigma_1)}} e^{-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}}{\frac{1}{\sqrt{(2\pi)^p \det(\Sigma_0)}} e^{-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)}}) + \log(\frac{\pi}{1-\pi}) \\ &= -\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + \frac{1}{2}\log(\frac{\det(\Sigma_0)}{\det(\Sigma_1)}) + \log(\frac{\pi}{1-\pi}) \end{aligned}$$

Ainsi,

$$\begin{aligned} \nabla_{\mu_1} f_\theta(x) &= \Sigma_1^{-1}(x-\mu_1) \\ \nabla_{\mu_0} f_\theta(x) &= -\Sigma_0^{-1}(x-\mu_0) \end{aligned}$$

Calcul de $\nabla_{\Sigma_1} f_\theta(x)$

1) On pose $\varphi(M) = A^T M^{-1} A$ où $M \in \mathcal{M}_p(\mathbb{R})$ et $A \in \mathbb{R}^n$.
Ainsi $\varphi = h \circ g$ où $h(M) = A^T M A$ et $g(M) = M^{-1}$.
On a : $dh_M(H) = A^T H A$ et $dg_M(H) = -M^{-1}HM^{-1}$ (on factorise par M quand on calcule $g(M+H)$).
D'où $d\varphi_M(H) = dh_{g(M)}(dg_M(H)) = dh_{M^{-1}}(-M^{-1}HM^{-1}) = -A^T M^{-1} H M^{-1} A$
Ainsi, $\nabla_M \varphi(M) = -(M^{-1})^T A A^T (M^{-1})^T$.

2) On pose $\psi(M) = \log(\det(M))$ où $M \in \mathcal{M}_p(\mathbb{R})$
On a : $\frac{\partial}{\partial M_{ij}} \log\det(M_{ij}) = \frac{1}{det(M)} \frac{\partial \det(M)}{\partial M_{ij}} = \frac{1}{det(M)} Cof_{ij}(M) = ((M^{-1})^T)_{ij}$.
Ainsi, $\nabla_M \psi(M) = (M^{-1})^T$.

Finalement,

$$\nabla_{\Sigma_1} f_\theta(x) = \frac{1}{2}(\Sigma_1^{-1})^T (x-\mu_1)(x-\mu_1)^T (\Sigma_1^{-1})^T - \frac{1}{2}(\Sigma_1^{-1})^T$$

De même,

$$\nabla_{\Sigma_0} f_\theta(x) = -\frac{1}{2}(\Sigma_0^{-1})^T (x-\mu_0)(x-\mu_0)^T (\Sigma_0^{-1})^T + \frac{1}{2}(\Sigma_0^{-1})^T$$

Puis, si $\Sigma_0 = \Sigma_1 = \Sigma$, on a :

$$\nabla_{\Sigma} f_\theta(x) = \frac{1}{2}(\Sigma^{-1})^T ((x-\mu_1)(x-\mu_1)^T - (x-\mu_0)(x-\mu_0)^T)(\Sigma^{-1})^T$$

Principe du maximum de vraisemblance

On souhaite minimiser $-\mathbb{E}(\log(P_\theta(Y|X))$, soit, d'après la question b), le risque :

$$R_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\sigma(f_\theta(x_i))) + (1-y_i)\log(1-\sigma(f_\theta(x_i)))$$

On pose $\varphi(x) = y\log(\sigma(x)) + (1-y)\log(1-\sigma(x))$ et on a $\varphi'(x) = y - \sigma(x)$.
Ainsi,

$$\begin{aligned}
\nabla_{\mu_1} R_n(\theta) &= -\frac{1}{n}\sum_{i=1}^{n} \nabla_{\mu_1} f_\theta(x_i)(y_i - \sigma(f_\theta(x_i))) \\
&= -\frac{1}{n}\sum_{i=1}^{n} \Sigma_1^{-1}(x_i - \mu_1)(y_i - \sigma(f_\theta(x_i))) \\
&= -\frac{1}{n}\Sigma_1^{-1} X_{\mu_1}^T Y_\nu
\end{aligned}$$

où $\nu_i(\theta) = \sigma(f_\theta(x_i))$.
De même,

$$\nabla_{\mu_0} R_n(\theta) = \frac{1}{n}\Sigma_0^{-1} X_{\mu_1}^T Y_\nu$$

Puis,

$$\begin{aligned}
\nabla_{\Sigma_1} R_n(\theta) &= -\frac{1}{n}\sum_{i=1}^{n} \nabla_{\Sigma_1} f_\theta(x_i)(y_i - \sigma(f_\theta(x_i))) \\
&= -\frac{1}{2n}(\Sigma_1^{-1})^T \sum_{i=1}^{n}((x_i-\mu_1)(x_i-\mu_1)^T(\Sigma_1^{-1})^T - I)(y_i - \sigma(f_\theta(x_i))) \\
\nabla_{\Sigma_0} R_n(\theta) &= -\frac{1}{n}\sum_{i=1}^{n} \nabla_{\Sigma_0} f_\theta(x_i)(y_i - \sigma(f_\theta(x_i))) \\
&= \frac{1}{2n}(\Sigma_0^{-1})^T \sum_{i=1}^{n}((x_i-\mu_0)(x_i-\mu_0)^T(\Sigma_0^{-1})^T - I)(y_i - \sigma(f_\theta(x_i)))
\end{aligned}$$

Et

$$\begin{aligned}
\nabla_{\Sigma} R_n(\theta) &= -\frac{1}{n}\sum_{i=1}^{n} \nabla_{\Sigma} f_\theta(x_i)(y_i - \sigma(f_\theta(x_i))) \\
&= \frac{1}{2n}(\Sigma^{-1})^T \sum_{i=1}^{n}((x-\mu_0)(x-\mu_0)^T - (x-\mu_1)(x-\mu_1)^T)(\Sigma^{-1})^T(y_i - \sigma(f_\theta(x_i)))
\end{aligned}$$

Aparté : Pourquoi ne pas apprendre Σ_i^{-1} ?

Remarquons que f_θ ne dépend que de l'inverse de Σ_i, on pourrait donc être tenté d'apprendre directement Σ_i^{-1} et non Σ_i (car $\forall M \in \mathcal{M}_p(\mathbb{R}), \det(M^{-1}) = \dfrac{1}{\det(M)}$).

On note $\alpha = (\mu_1, \mu_0, \Sigma_1^{-1}, \Sigma_0^{-1}) = (\mu_1, \mu_0, \Omega_1, \Omega_0) \in (\mathbb{R}^p, \mathbb{R}^p, \mathcal{M}_p(\mathbb{R}), \mathcal{M}_p(\mathbb{R}))$ (on perds peut-être la symétrie).
Donc :

$$f_\alpha(x) = -\frac{1}{2}(x-\mu_1)^T\Omega_1(x-\mu_1) + \frac{1}{2}(x-\mu_0)^T\Omega_0(x-\mu_0) + \frac{1}{2}\log(\frac{\det(\Omega_1)}{\det(\Omega_0)}) + \log(\frac{\pi}{1-\pi})$$
$$\nabla_{\mu_1} f_\alpha(x) = \Omega_1(x-\mu_1)$$
$$\nabla_{\mu_0} f_\alpha(x) = -\Omega_0(x-\mu_0)$$
$$\nabla_{\Omega_1} f_\alpha(x) = -\frac{1}{2}(x-\mu_1)(x-\mu_1)^T + \frac{1}{2}(\Omega_1^{-1})^T$$
$$\nabla_{\Omega_0} f_\alpha(x) = \frac{1}{2}(x-\mu_0)(x-\mu_0)^T - \frac{1}{2}(\Omega_0^{-1})^T$$

Et

$$R_n(\alpha) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\sigma(f_\alpha(x_i))) + (1-y_i)\log(1-\sigma(f_\alpha(x_i)))$$
$$\nabla_{\mu_1} R_n(\alpha) = -\frac{1}{n}\Omega_1 X_{\mu_1}^T Y_\nu$$
$$\nabla_{\mu_0} R_n(\alpha) = \frac{1}{n}\Omega_0 X_{\mu_1}^T Y_\nu$$
$$\nabla_{\Omega_1} R_n(\theta) = \frac{1}{2n}\sum_{i=1}^{n}((x-\mu_1)(x-\mu_1)^T - (\Omega_1^{-1})^T)(y_i - \sigma(f_\theta(x_i)))$$
$$\nabla_{\Omega_0} R_n(\theta) = -\frac{1}{2n}\sum_{i=1}^{n}((x-\mu_0)(x-\mu_0)^T - (\Omega_0^{-1})^T)(y_i - \sigma(f_\theta(x_i)))$$
$$\nabla_{\Omega} R_n(\theta) = \frac{1}{2n}\sum_{i=1}^{n}((x-\mu_1)(x-\mu_1)^T - (x-\mu_0)(x-\mu_0)^T)(y_i - \sigma(f_\theta(x_i)))$$

On remarque que dans l'hypothèse $\Sigma_1 = \Sigma_0$ que l'ont fait pour la méthode LDA, l'inverse de Ω n'apparaît pas. On n'a à calculer aucune inverse de matrice. Cela pourrait permettre de gagner du temps de calcul.

Résultats : cf Annexe

d)

Résultats : cf Annexe

Commentaires :
Comme le nom de l'algorithme l'indique, les frontières de séparation ne sont plus linéaire, mais quadratique. On s'est en effet donné un degré de liberté en plus.

0.5

Plutôt que de représenter les quatre classifieurs sur le même schéma, j'ai choisi d'afficher les résultats pour chaque question.

0.6 Tableau des taux d'erreurs de classifications

	A_train	A_test	B_train	B_test	C_train	C_test
Régression linéaire	6%	12.34%	16.67%	19.45%	35%	43.4%
Régression logistique	2.7%	2.27%	3%	4.15%	4.25%	2.77%
LDA	2.67%	2%	2%	4.75%	4%	2.23%
QDA	1.34%	2.07%	1.76%	2.20%	1.25%	1.93%

0.7 Commentaires

La méthode QDA est très efficace sur l'ensemble des données là où les méthodes linéaires (LDA et régression logistique) sont meilleures sur les données de faibles covariance et de Gaussiennes simples. On remarque une très faible différence entre LDA et la régression logistique, compréhensible d'après la question 4*b*) (le modèle de f ne semble pas importer tant que ça pour ces exemples). La première méthode par plug-in est inefficace.

0.8 Méthode des k-plus proches voisins

Pour la méthode des k plus proches voisins, on ajuste k par validation croisée. On choisit donc aléatoirement des indices puis on découpe notre intervalle pour lancer localement un $k - PPV$. C'est pourquoi lors de chaque exécution de l'algorithme, on peut avoir un k optimum différent.
Ici un tableau récapitulatif avec des k possibles (en pratique les plus souvent donnés par l'algorithme de validation croisée).

	A_train	A_test	B_train	B_test	C_train	C_test
$k = 1$	0%	2.47%	0%	4.0%	0%	2.3%
$k = 2$	2%	2.67%	1%	2.85%	0.5%	2.13%
$k = 3$	3.33%	2.06%	1%	3.25%	3.5%	2.1%

La méthodes des k plus proches voisins est remarquablement efficace étant donné qu'elle ne considère pas de modèle pour f. Elle l'est d'autant plus que le modèle des fonctions se complexifient (comme sur les données C). Cette efficacité du taux d'erreur est compensé par un temps de calcul plus important que les autres méthodes.

0.9

On considère les données MNIST. On en sélectionne aléatoirement $D = 6000$ qu'on partitionne selon $x = \frac{1}{3}$, i.e. un tiers de données d'entrainement, deux tiers de données de test.

LDA et QDA à K classes

Soit K le nombre de classe de notre problème.

Modèle

On modélise les données de chaque classe comme un nuage Gaussien de distribution $\mathcal{N}(\mu_k, \Sigma_k)$ pour chaque classe k.
On note $\forall k \in \{0, K-1\}$,

$$f^k(x) \doteq f_{\mu_k,\Sigma_k}(x) = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma_k)}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

Alors :

$$\nabla_{\mu_k} f^k(x) = \Sigma_k^{-1}(x-\mu_k) f^k(x)$$

Et

$$\nabla_{\Sigma_k} f^k(x) = f^k(x)(\frac{1}{2}(\Sigma_k^{-1})^T (x-\mu_k)(x-\mu_k)^T (\Sigma_k^{-1})^T - \frac{1}{2}(\Sigma_k^{-1})^T))$$

Maximisation de la vraisemblance On a, notant $\pi_j = P(Y=j)$ et $h_k = \pi_k f^k$,

$$P(Y=k|X=x) = \frac{P(Y=k \wedge X=x)}{P(X=x)} = \frac{\pi_k f_{\mu_k,\Sigma_k}}{\sum_{j=0}^{K-1} \pi_j f_{\mu_j,\Sigma_j}} = \frac{h_k}{\sum_{j=0}^{K-1} h_j}$$

On souhaite maximiser la log-vraisemblance. On note le risque :

$$R_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log(\frac{h_{y_i}(x_i)}{\sum_{j=0}^{K-1} h_j(x_i)}) = -\frac{1}{n}\sum_{i=1}^{n}(\log(h_{y_i}(x_i)) - \log(\sum_{j=0}^{K-1} h_j(x_i)))$$

On pose alors $\varphi_k(x) = \log(h_k(x)) - \log(\sum_{j=0}^{K-1} h_j(x))$

D'où

$$\nabla_{\mu_k} \varphi_i(x) = \delta_{ik} \frac{\nabla_{\mu_k} f^k(x)}{f^k(x)} - \pi_k \frac{\nabla_{\mu_k} f^k(x)}{\sum_{j=0}^{K-1} h_j(x)}$$

Ainsi,

$$\nabla_{\mu_k} R_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n}(\delta_{y_i k}\frac{\nabla_{\mu_k} f^k(x_i)}{f^k(x_i)} - \pi_k \frac{\nabla_{\mu_k} f^k(x_i)}{\sum_{j=0}^{K-1}\pi_j f^j(x_i)})$$
$$= -\frac{1}{n}\sum_{i=1}^{n}(\delta_{y_i k}\Sigma_k^{-1}(x_i-\mu_k) - \Sigma_k^{-1}(x-\mu_k)\frac{\pi_k f^k(x_i)}{\sum_{j=0}^{K-1}\pi_j f^j(x_i)})$$

De même :

$$\nabla_{\Sigma_k} R_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n}(\delta_{y_i k}\frac{\nabla_{\Sigma_k} f^k(x_i)}{f^k(x_i)} - \pi_k \frac{\nabla_{\Sigma_k} f^k(x_i)}{\sum_{j=0}^{K-1}\pi_j f^j(x_i)})$$
$$= -\frac{1}{2n}\sum_{i=1}^{n}(\delta_{y_i k}(\Sigma_k^{-1})^T(x_i-\mu_k)(x_i-\mu_k)^T(\Sigma_k^{-1})^T - (\Sigma_k^{-1})^T))$$
$$-((\Sigma_k^{-1})^T(x_i-\mu_k)(x_i-\mu_k)^T(\Sigma_k^{-1})^T - (\Sigma_k^{-1})^T))\frac{\pi_k f^k(x_i)}{\sum_{j=0}^{K-1}\pi_j f^j(x_i)})$$

Résume

On note :

$$H_k(x) = \frac{\pi_k f^k(x)}{\sum_{j=0}^{K-1}\pi_j f^j(x)}$$
$$A_k(x) = \Sigma_k^{-1}(x-\mu_k)$$
$$B_k(x) = (\Sigma_k^{-1})^T(x-\mu_k)(x-\mu_k)^T(\Sigma_k^{-1})^T - (\Sigma_k^{-1})^T$$

On a finalement :

$$R_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\log(H_{y_i}(x_i))$$
$$\nabla_{\mu_k} R_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}A_k(x_i)(H_k(x_i)-\delta_{y_i k})$$
$$\nabla_{\Sigma_k} R_n(\theta) = \frac{1}{2n}\sum_{i=1}^{n}B_k(x_i)(H_k(x_i)-\delta_{y_i k})$$

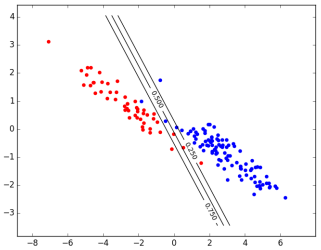
Résultats Cette méthode est implémentée (pour QDA) et tourne bien, mais elle est très longue à s'exécuter. En effet, des matrices de taille $784, 784$ à inverse, 10 classes, même en prenant une

centaine de données d'entrainement, le tout est très lent. Ça fonctionne mais je n'ai malheureusement pas de tableaux avec des résultats concluants.
Ci-dessous les résultats (rapides) des k plus proches voisins.

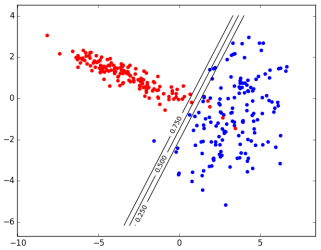
	train	test
$k = 1$	0%	9.55%
$k = 2$	5.4%	10.3%
$k = 3$	4.7%	8.75%
$k = 4$	5.3%	9.775%
$k = 5$	6.5%	9.225%
$k = 6$	7.15%	9.225%

Annexe

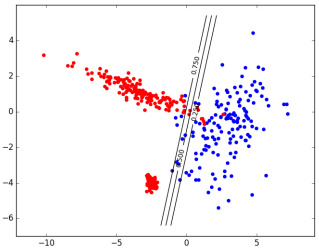
LDA



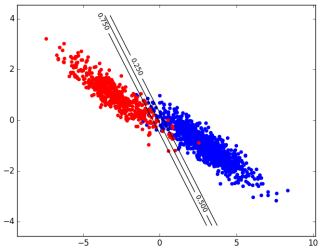
Ensemble d'apprentissage A
(erreur : 2.67%)



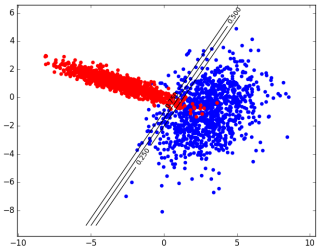
Ensemble d'apprentissage B
(erreur : 2.0%)



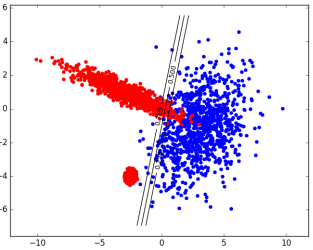
Ensemble d'apprentissage C
(erreur : 4.0%)



Ensemble de test A
(erreur : 2.0%)

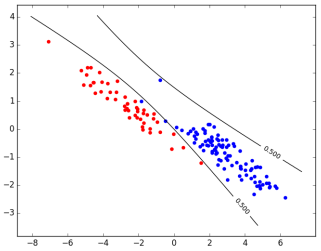


Ensemble de test B
(erreur : 4.75%)

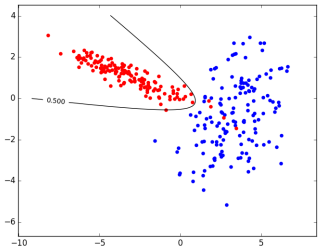


Ensemble de test C
(erreur : 2.23%)

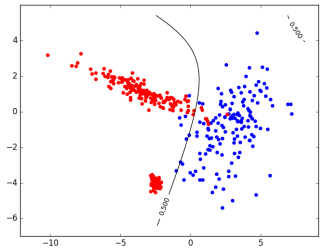
QDA



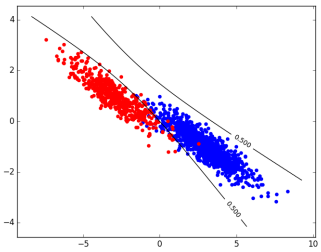
Ensemble d'apprentissage A
(erreur : 1.34%)



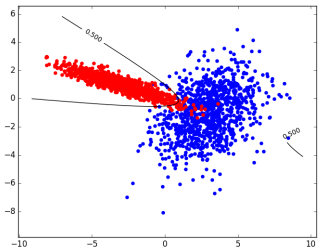
Ensemble d'apprentissage B
(erreur : 1.67%)



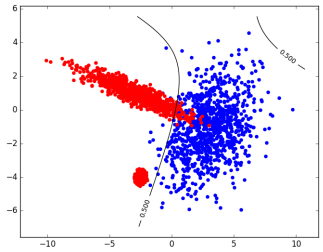
Ensemble d'apprentissage C
(erreur : 3.5%)



Ensemble de test A
(erreur : 2.07%)



Ensemble de test B
(erreur : 2.20%)



Ensemble de test C
(erreur : 2.1%)