Devoir n°2

Pierre Snell

November 14, 2018

Sommaire

Liste des figures

Liste des tableaux

1 Algorithme de rétropropagation

Les calculs ci-dessous se font avec une fonction d'activation tangeante hyperbolique, dont la dérivée selon le résultat de l'activation vaut :

$$\frac{\partial y_j^t}{\partial a_j^t} = \frac{\partial tanh(a_j^t)}{\partial a_j^t} = 1 - tanh(a_j^t)^2$$

En effet :

$$\frac{\partial sinh(x)}{\partial x} = cosh(x) \qquad \frac{\partial cosh(x)}{\partial x} = sinh(x) \qquad tanh(x) = \frac{sinh(x)}{cosh(x)}$$

$$\frac{\partial tanh(x)}{\partial x} = \frac{1}{\partial x}\frac{sinh(x)}{cosh(x)} = \frac{cosh(x)\frac{\partial sinh(x)}{\partial x} - sinh(x)\frac{\partial cosh(x)}{\partial x}}{cosh^2(x)}$$

$$= \frac{cosh(x)cosh(x) - sinh(x)sinh(x)}{cosh^2(x)} = \frac{cosh^2(x) - sinh^2(x)}{cosh^2(x)}$$

$$= 1 - \frac{sinh^2(x)}{cosh^2(x)} = 1 - tanh^2(x) \qquad \square$$

L'erreur observée est donnée par : $e_j^t = r_j^t - y_j^t$ avec r_j^t les label des données mis sous forme de one-hot vector $\begin{cases} 1 & \text{en j si label de classe j} \\ 0 & \text{autrement} \end{cases}$ et y_j^t les étiquettes prédites par le réseau et l'erreur quadratique : $E^t = \frac{1}{2}\sum_{j=1}^{K}(e_j^t)^2$

1.a Couche de sortie

$$\Delta w_{j,i} = -\eta\frac{\partial E}{\partial w_{j,i}} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial w_{j,i}}$$

On peut calculer $\frac{\partial E^t}{\partial w_{i,j/0}}$ grâce à la règle de chaînage des dérivées partielles :

$$\frac{\partial E^t}{\partial w_{i,j/0}} = \frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{i,j/0}}$$

1.a.i Calcul de la correction pour $w_{j,i}$

$$\begin{aligned}
\frac{\partial E^t}{\partial e_j^t} &= \frac{\partial}{\partial e_j^t}\frac{1}{2}\sum_{l=1}^{K}(e_l^t)^2 = e_j^t \\
\frac{\partial e_j^t}{\partial y_j^t} &= \frac{\partial}{\partial y_j^t}(r_j^t - y_j^t) = -1 \\
\frac{\partial y_j^t}{\partial a_j^t} &= \frac{\partial tanh(a_j^t)}{\partial a_j^t} = 1 - tanh(a_j^t)^2 = 1 - (y_j^t)^2 \\
\frac{\partial a_j^t}{\partial w_{j,i}} &= \frac{\partial}{\partial w_{j,i}}\sum_{l=1}^{R} w_{j,l}y_l^t + w_{j,0} = y_i^t
\end{aligned}$$

On a donc la correction suivante :

$$\begin{aligned}\Delta w_{j,i} &= -\eta \frac{\partial E}{\partial w_{j,i}} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial w_{j,i}} = \frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{j,i}} \\ &= -\frac{\eta}{N}\sum_{t=1}^{N} e_j^t \cdot (-1) \cdot (1-(y_j^t)^2) \cdot y_i^t \\ &= \frac{\eta}{N}\sum_{t=1}^{N} e_j^t \cdot y_i^t \cdot (1-(y_j^t)^2)\end{aligned}$$

On peut poser $\delta_j^t = e_j^t \cdot (1-(y_j^t)^2)$
Ce qui donne

$$\Delta w_{j,i} = \frac{\eta}{N}\sum_{t=1}^{N}\delta_j^t y_j^t$$

1.a.ii Calcul de la correction pour w_0

On reprend les résultats de $w_{j,i}$ seul la dérivée $\frac{\partial a_j^t}{\partial w_{j,0}}$ change :

$$\frac{\partial a_j^t}{\partial w_0} = \frac{\partial}{\partial w_0}\sum_{l=1}^{R} w_{j,l} y_l^t + w_{j,0} = 1$$

On a donc la correction suivante :

$$\begin{aligned}\Delta w_0 &= -\eta \frac{\partial E}{\partial w_0} = -\frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial w_0} \\ &= \frac{\eta}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial e_j^t}\ \frac{\partial e_j^t}{\partial y_j^t}\ \frac{\partial y_j^t}{\partial a_j^t}\ \frac{\partial a_j^t}{\partial w_0} \\ &= -\frac{\eta}{N}\sum_{t=1}^{N} e_j^t \cdot (-1) \cdot (1-(y_j^t)^2) \cdot 1 \\ &= \frac{\eta}{N}\sum_{t=1}^{N} e_j^t \cdot (1-(y_j^t)^2)\end{aligned}$$

On pose toujours $\delta_j^t = e_j^t \cdot (1-(y_j^t)^2)$
Ce qui donne :

$$\Delta w_0 = \frac{\eta}{N}\sum_{t=1}^{N}\delta_j^t$$

1.b Couches cachées

On peut calculer $\frac{\partial E^t}{\partial w_{j,i}}$ grâce à la règle de chaînage des dérivées partielles :

$$\frac{\partial E^t}{\partial w_{i,j/0}} = \frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{i,j/0}}$$

1.b.i Calcul de la correction pour $w_{i,j}$

En reprennant les résultats des couches de sortie, seul $\frac{\partial E^t}{\partial y_j^t}$ change :

$$\begin{aligned}\frac{\partial E^t}{\partial y_j^t} &= \frac{\partial}{\partial y_j^t}\frac{1}{2}\sum_k (e_k^t)^2 = \sum_k e_k^t \frac{\partial e_k^t}{\partial y_j^t} \\ &= \sum_k e_k^t \frac{\partial e_k^t}{\partial a_k^t}\frac{\partial a_k^t}{\partial y_k^t} \\ &= \sum_k e_k^t \frac{\partial (r_k^t - y_k^t)}{\partial a_k^t}\frac{\partial (\sum_l w_{k,l} y_l^t + w_{k,0})}{\partial y_k^t} \\ &= \sum_k e_k^t (-(1-(y_k^t)^2)) w_{k,j}\end{aligned}$$

On pose toujours le même $\delta_k^t = e_k^t \cdot (1-(y_k^t)^2)$

$$\frac{\partial E^t}{\partial y_j^t} = -\sum_k \delta_k^t w_{k,j}$$

Ce qui donne la correction suivante :

$$\Delta w_{j,i} = -\eta \frac{\partial E}{\partial w_{j,i}} = -\frac{\eta}{N}\sum_{t=1}^N \frac{\partial E^t}{\partial w_{i,j}} = \frac{\eta}{N}\sum_{t=1}^N \frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{j,i}}$$

$$= \frac{\eta}{N}\sum_{t=1}^N \big[-\sum_k \delta_k^t w_{k,j}\big] \cdot (1-(y_j^t)^2) y_i^t$$

On pose donc un nouveau delta pour les couches cachées :

$$\delta_{j_c}^t = (1-(y_j^t)^2)\sum_k \delta_k^t w_{k,j}$$

Ce qui donne :

$$\Delta w_{j,i} = \frac{\eta}{N}\sum_{t=1}^N \delta_{j_c}^t y_i^t$$

1.b.ii Calcul de la correction pour w_0

De même avec tout les résultats précédents :

$$\Delta w_{j,i} = \frac{\eta}{N}\sum_{t=1}^N \frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_0} \qquad \frac{\partial a_j^t}{\partial w_{j,i}} = 1 \qquad \frac{\partial y_j^t}{\partial a_j^t} = 1-(y_j^t)^2 \qquad \frac{\partial E^t}{\partial y_j^t} = -\sum_k \delta_k^t w_{k,j}$$

$$\delta_{j_c}^t = (1-(y_j^t)^2)\sum_k \delta_k^t w_{k,j}$$

$$\Delta w_0 = \frac{\eta}{N}\sum_{t=1}^N \delta_{j_c}^t$$

2 Discriminants non-linéraires

La méthode choisie pour optimiser les hyper-paramètres est une recherche en grille, on choisit des valeurs pour chacun des paramètres puis on teste les classificateurs pour toutes les combinaisons de paramètres possibles. En faisant cela, on parcourt une grille de paramètres et on regarde les zones ou on obtient un meilleur score afin d'affiner la recherche.

2.a Hyperparamètres

2.a.i Knn

Pour les knn le paramètre évident est le nombre de voisins. La distance ensuite peut jouer un effet, mais entre le paramètre "uniforme" et "distance", "distance" l'emporte quasiment à chaque fois sur tout les shuffles du dataset possibles.
On fixe donc le paramètre de mesure de la distance à "distance" (poid de voisins inversement proportionnels à leur distance euclidienne).
Pour optimiser K, on choisit de faire un K-fold sur le dataset d'entraînement. Ce qui nous permet d'entraîner un paramètre puis de le tester avant de passer au suivant.
Pour trouver K, on parcourt la grille suivante : $K \in [\![1, ..., 10]\!] \cup [\![10, 20, ..., 90, 100]\!]$.
Cette première recherche rapide montre que les meilleurs valeurs possibles pour K se trouvent dans l'intervalle $[\![1, 4]\!]$ en fonction du découpage du dataset.
On explore donc l'intervalle de 1 à 10 par pas de 1.
Puisque le temps alloué pour l'exercice est long, et que les résultats varient en focntion du découpage, on moyenne le K-fold sur 20 résultats. On trouve les valeurs suivantes (pour la dernière boucle):

K	1	2	3	4	5	6	7	8	9
Boucle 20	0.01050	0.01050	0.01011	0.01016	0.01354	0.012955	0.01571	0.01464	0.01628

Table 1: Comparaison des erreurs pour différents paramètres de K

K moyen	2.1
K optimisé	2

En moyennant on trouve un K moyen très souvent autour de 2. ($K_{moy} \in [1.8,\ 2.9]$) Le meilleur paramètre pour bien généraliser sur n'importe qu'elle division des données est donc k = 2.
On prend cette donnée pour ensuite entraîner le classificateur par K plus proches voisins et on trouve une erreur de 0.01435.
On remarque que en moyenne l'erreur pour 1, 2 parfois 3 voisins est similaire et qu'elle augmente ensuite. En moyenne on se rapproche plus de K = 2 qui est le paramètre optimal. En revanche une zone de classification "correcte" serait $K \in [\![1, 4]\!]$ car c'est dans cette zone que la performance est optimale.
On peut donc comprendre que le jeu de données a des données hétérogènes sans trop de groupes éloignés. En effet avec plusieurs groupes de plusieurs données bien distincts et séparés, on pourrais trouver un K plus grand. Or dès que l'on dépasse les 3 plus proches voisins la performance baisse. On peut donc penser que en moyenne, les données forment des groupes de quelques valeurs seulement plutôt disparates ce qui explique un K faible

2.a.ii SVM

Pour les svm on cherche à optimiser σ la taille du noyau gaussien et la marge C. Pour sigma on peut affiner avec nos connaissances en prenant des multiples de $\sigma_{min} = \min_{\forall \boldsymbol{x^i} \neq \boldsymbol{x^j}} ||\boldsymbol{x^i} - \boldsymbol{x^j}||$

On prend ensuite $\sigma \in \{\ 2^i \cdot \sigma_{min} \quad i \in [\![1, 6]\!]\ \}$

Pour C, on commence par une recherche large $C \in \{\ 10^i \quad i \in [\![-5, 5]\!]\ \}$ On affine ensuite sur quelques valeurs qui semblent intéressantes.

C \ γ	$1.78 \cdot 10^2$	$4.46 \cdot 10^1$	$1.11 \cdot 10^1$	$2.79 \cdot 10^0$	$6.97 \cdot 10^{-1}$	$1.74 \cdot 10^{-2}$
0.001	0.9060	0.9060	0.9060	0.9060	0.9060	0.9060
0.1	0.9060	0.9050	0.3012	0.02799	0.0422	0.10239
1	0.7748	0.2706	0.0344	0.0064	0.0072	0.0162
$20 \rightarrow 45$	0.752 ± 0.01	0.250 ± 0.01	0.032	0.0048 ± 0.001	0.0046 ± 0.0002	0.0066 ± 0.002
100	0.7412	0.2494	0.0320	0.0060	0.0048	0.0064
1000	0.7916	0.26299	0.03719	0.00639	0.00579	0.00679

Table 2: Comparaison des erreurs pour différents paramètres de C et γ

Puisque un Svm est bien plus long à entraîner on réalise juste un K-fold pour chercher les meilleurs paramètres.
Ici, γ doit sans hésiter être égal à 0.697 car c'est la que se trouvent les meilleures valeurs pour tout les C confondus.
Or $\gamma = \frac{1}{2 \cdot \sigma^2}$ donc le paramètre optimal est $\sigma = \frac{1}{\sqrt{2\gamma}} = \frac{1}{\sqrt{2 \cdot 0.697}} = 0.847$

On pourrait ensuite chercher plus précisément (à l'ordre de grandeur inférieur) mais les performances sont déjà suffisantes.
En revanche, pour C les valeurs comprises entre 10 et 50 sont acceptables, l'erreur varie en fonction du découpage du dataset mais les performances restent relativement correctes vers ces valeurs. Tant que l'ordre de grandeur est entre 10 et 100 les résultats sont corrects.
Un σ relativement élevé montre que les données peuvent êtres séparés par des plans plutôt lisses et moins stricte au bruit. Pour le paradigme biais/variance, un plus petit σ représente un peu moins de biais et un peu plus de variance.
Pour C, un C élevé signifie que la pénalisation des variables dans la marge est forte et donc que peu de variables vont se retrouver dans la marge. Il est donc normal d'avoir un C au alentours de 10 à 100 en effet, trop peu et toutes les données sont dans la marge, trop haut et la frontière est trop "stricte".

2.a.iii MLP

Pour le Mlp, les paramètres importants sont la taille et le nombre des couches.
On peut les trouver avec une règle du pouce : la première couche vaut 2 * taille des features.
On essaye donc avec des multiples de la couche d'entrée. On essaye une seule couche en faisant varier sa taille puis deux à taille variable puis trois (en prennant la couche intermedaire avec une taille (Csortie + Centrée)/2)

Ce \ Cs	0	20	30	40	50	60	70
32	0.0185	0.0181	0.0166	0.0169	0.0159	0.0139	0.0153
48	0.0185	0.0171	0.0155	0.0125	0.0153	0.0159	0.0135
64	0.0176	0.0185	0.0151	0.0120	0.0139	0.0143	0.0151
80	0.0173	0.0198	0.0158	0.0129	0.0133	0.0139	0.0141
96	0.0140	0.0145	0.0139	0.0137	0.0151	0.0143	0.0143

Table 3: Comparaison des erreurs pour différentes tailles de couches à 2 niveaux

La table suivante montre les couches d'entrée, de sortie et dans chaque cellules, la couche intermédiaire et l'erreur.

Ce \ Cs	20	30	40	50	60	70	80
32	26 : 0.0235	31 : 0.0210	36 : 0.0172	41 : 0.0175	46 : 0.0147	51 : 0.0171	56 : 0.0193
48	34 : 0.0177	39 : 0.0197	44 : 0.0138	49 : 0.0129	54 : 0.0159	59 : 0.0148	64 : 0.0146
64	42 : 0.0175	47 : 0.0176	52 : 0.0139	57 : 0.0163	62 : 0.0143	67 : 0.0138	72 : 0.0149
80	50 : 0.0169	55 : 0.0165	60 : 0.0141	65 : 0.0114	70 : 0.0117	75 : 0.0147	80 : 0.0127
96	58 : 0.0117	63 : 0.0143	68 : 0.0161	73 : 0.0155	78 : 0.0141	83 : 0.0144	88 : 0.0123

Table 4: Comparaison des erreurs pour différentes tailles de couches à trois niveaux

En fonction des découpages du dataset, un paramètre ressort, la première couche de 80 neuronnes donne toujours les meilleurs résultats.
Ensuite, deux ou trois couches donnent des résultats très proches, toutes les configurations donnent des résultats acceptables.
On retiens en moyenne ± 55 neurones pour la couche de sortie.
Puisque une troisième couche cachée n'apporte pas grand chose en termes de résultats mais qu'elle implique beaucoup

plus de calculs, le meilleurs modèle est donc un modèle à deux couches, (80,55).

2.b Performance des modèles optimisés

On effectue de nouveau un K-fold mais cette fois ci sur le jeu de test pour voir les performances en généralisation de chaques modèles.

	Mlp	Svm	Knn
Erreur entrainement	0.040	0.005	0.015
Erreur test	0.015	0.005	0.011
Temps	$2.15 \pm 0.15s$	$0.09 \pm 0.01s$	$0.003 \pm 0.0001s$

2.c Comparaisons des modèles

Ces trois modèles donnent de bonnes performances sur le dataset, et peuvent êtres utilisés en pratique. En revanche, nous pouvons quand même les comparer sur la même tâche qu'ils ont dû effectuer.

2.c.i Bilan

Même si ces performances sont moindres, le meilleur rapport, coût/facilité/embarquabilité est le Knn (pour cette tâche).
Si l'on vise la précision, les Svms sont plus adaptés mais légèrement plus long et plus coûteux à entraîner.
Les Mlp sont des modèles complexes qui généralisent très bien et sont adaptables partout en revanche, ici pour le même résultat, les coûts de calculs sont trop élevés.
On pourrait arriver à une précision optimales avec bien plus de couches mais, même pour égaler le Svm cela demanderais trop de calculs et le modèle serait plus simple que d'autres qui perfomment autant avec moins de ressources.

3 Discrimination avec noyau et descente de gradient

3.a Equations des gradients de α^t et de w_0

$$E(\boldsymbol{\alpha}, w_0|\boldsymbol{\mathcal{X}}) = \sum_{x^t \in \mathcal{Y}} [1 - r^t h(\boldsymbol{x}^t|\boldsymbol{\alpha}, w_0)] + \lambda \sum_{\alpha^t \in \boldsymbol{\alpha}} \alpha^t$$

$$h(\boldsymbol{x^t}|\boldsymbol{\alpha}, w_0) = \sum_{\boldsymbol{x}^s \in \mathcal{X}} \alpha^s r^s K(\boldsymbol{x}^s, \boldsymbol{x}^t) + w_0$$

$$\mathcal{Y} = \{\boldsymbol{x}^t \in \mathcal{X} | r^t h(\boldsymbol{x}^t|\boldsymbol{\alpha}, w_0) < 1\}$$

$$\alpha^t > 0 \ \forall t$$

3.a.i Mise à jour du gradient de α^i

$$\nabla\alpha^i = \frac{\partial E(\boldsymbol{\alpha}, w_0|\boldsymbol{\mathcal{X}})}{\partial \alpha^i}$$

$$\frac{\partial E(\boldsymbol{\alpha}, w_0|\boldsymbol{\mathcal{X}})}{\partial \alpha^i} = \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} 1}{\partial \alpha^i} - \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} r^t h(\boldsymbol{x}^t|\boldsymbol{\alpha}, w_0)}{\partial \alpha^i} + \frac{\partial \lambda \sum\limits_{\alpha^s \in \boldsymbol{\alpha}} \alpha^s}{\partial \alpha^i}$$

$$= 0 - \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} r^t \big(\sum\limits_{\boldsymbol{x}^s \in \mathcal{X}} \alpha^s r^s K(\boldsymbol{x}^s, \boldsymbol{x}^t) + w_0 \big)}{\partial \alpha^i} + \lambda \cdot 1$$

$$= - \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} r^t \big(\sum\limits_{\boldsymbol{x}^s \in \mathcal{X}} \alpha^s r^s K(\boldsymbol{x}^s, \boldsymbol{x}^t) + w_0 \big)}{\partial \alpha^i} + \lambda$$

$$\nabla\alpha^i = - \sum_{x^t \in \mathcal{Y}} r^t \big(r^i K(\boldsymbol{x}^i, \boldsymbol{x}^t) \big) + \lambda$$

3.a.ii Mise à jour des w_0

$$\nabla w_0 = \frac{\partial E(\alpha^i, w_0|\boldsymbol{\mathcal{X}})}{\partial w_0}$$

$$= \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} 1}{\partial w_0} - \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} r^t h(\boldsymbol{x}^t|\alpha^i, w_0)}{\partial w_0} + \frac{\partial \lambda \sum\limits_{\alpha^s \in \boldsymbol{\alpha}} \alpha^s}{\partial w_0}$$

$$= 0 - \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} r^t \big(\sum\limits_{\boldsymbol{x}^s \in \mathcal{X}} \alpha^s r^s K(\boldsymbol{x}^s, \boldsymbol{x}^t) + w_0 \big)}{\partial w_0}$$

$$= - \frac{\partial \sum\limits_{x^t \in \mathcal{Y}} r^t \big(\sum\limits_{\boldsymbol{x}^s \in \mathcal{X}} \alpha^s r^s K(\boldsymbol{x}^s, \boldsymbol{x}^t) + w_0 \big)}{\partial w_0}$$

$$= - \sum_{x^t \in \mathcal{Y}} r^t \cdot (1)$$

$$\Delta w_0 = \eta \sum_{x^t \in \mathcal{Y}} r^t$$

3.b Expérimentation

3.b.i paramètre entraînement

Comme indiqué on prend $\sigma \in \{0,\ 1\}$ et pour λ on effectue une recherche en grille.
$\lambda \in [\![0.01, ..., 0.09]\!] \cup [\![0.1, ..., 0.9]\!] \cup \{1, 5, 10, 25, 50, 100\}$ La recherche de paramètres optimaux prend $\pm$180s pour 120 classificateurs.

3.b.ii paramètres finaux

Après cette recherche, on trouve que les meilleurs paramètres possibles sont $\lambda \in [\![0.06, 0.1]\!]$ et $\sigma = 0.6$ En revanche, beaucoup de paramètres permettent un erreur de moins de 10%, allant de $\sigma = 0.1 \to 0.8$ et $\lambda = 0.1 \to 0.9$ On voit donc que les paramètres optimaux se trouvent dans la zone des points violets foncés. Tout paramètres

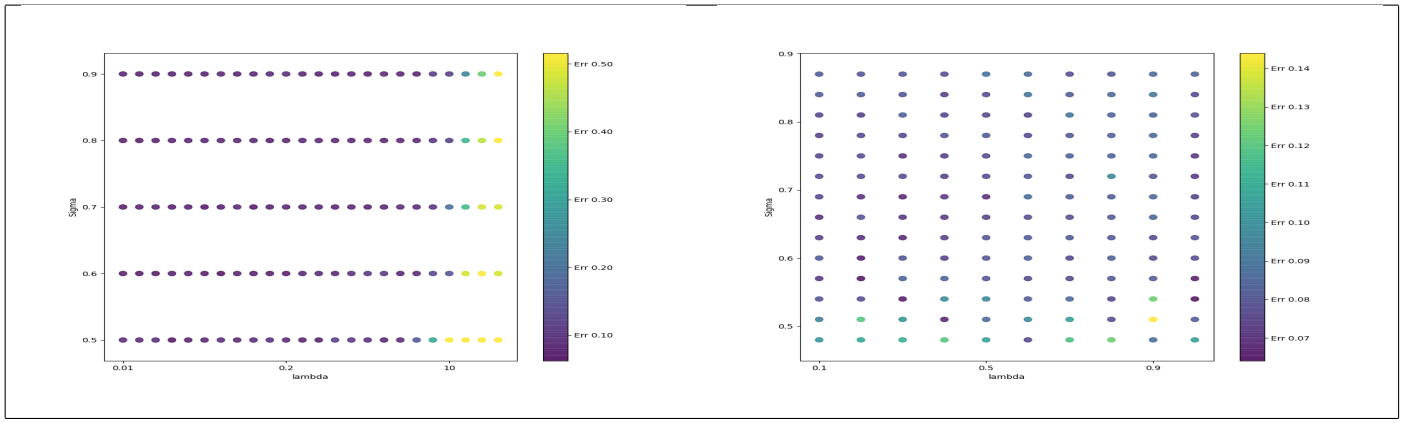

Figure 1: Grille de recherche des paramètres
Abscisse : λ. Ordonnées : σ
À gauche la grille de recherche pour un pas grand
À droite, un affinage autour de $\lambda \in [\![0.5, ..0.9]\!]$ et $\sigma \in [\![0.5, ...0.9]\!]$

$\lambda = 0.5 \pm 0.2$ et $\sigma = 0.7 \pm 0.15$ permettent des résultats avec moins de 0.07% d'erreur. En comptant tout les paramètres qui donnent moins de 10% d'erreur, 127 combinaisons sont possibles.
Le *lambda* faible vient du fait que les données sont bruitées et que nous avons besoin de plus de données dans la marge pour classifier correctement expliqué en (2.a.ii). λ trop haut et la frontière serait trop "stricte", les données bruitées ne serait pas prises en compte.
Trop bas et les données de l'autre classe viendrait mal se classer dû à une frontière trop "souple".
σ relativement haut, signifie que l'on souhaite une frontière plutôt "smooth" pour pouvoir capturer les données bruitées. Si σ était trop faible, la frontière de décision serait beaucoup plus "en dent de scie" et le bruit gênerait la bonne classification.

3.b.iii Graphique

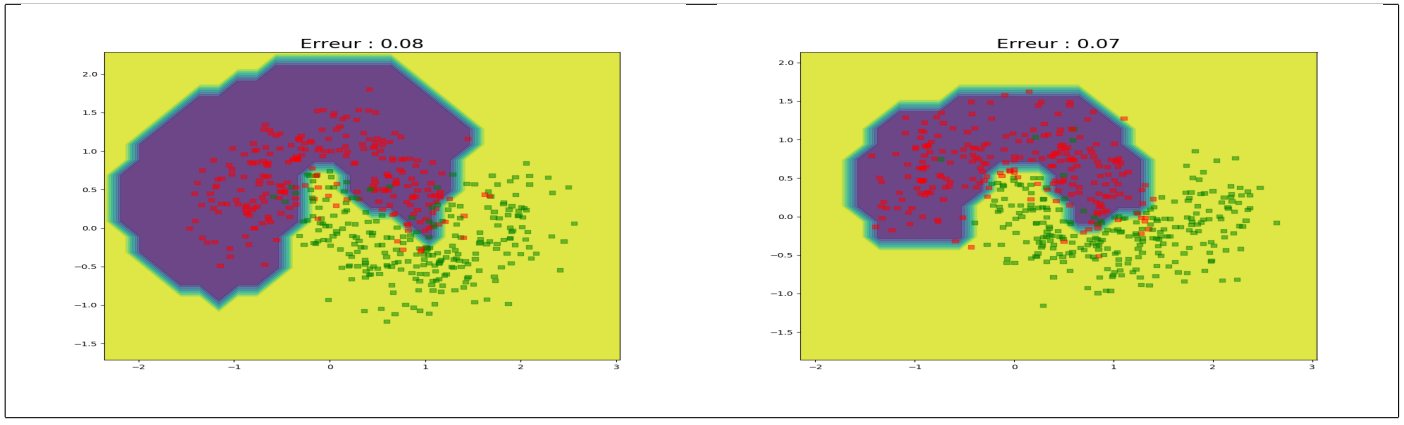

Figure 2: Frontières de décision selon 2 classes avec bruit gaussien $\sigma = 0.3$
Taux moyen d'erreur 7%