Compte-rendu TPs 1-2-3

Bayesian Deep Learning - RDFIA

Célina Hanouti
Rym Kaci

Master 2 DAC
2020-2021

Table des matières

1 TP1 - Bayesian Linear Regression **2**
- 1.1 Analyse des résultats de la distribution prédictive . . . 2
- 1.2 Preuve analytique pour $\alpha = 0$ et $\beta = 1$ . . . 2

2 TP2 - Approximate Inference **4**
- 2.1 Regression logistique bayesienne . . . 4
  - 2.1.1 Inférence variationnelle . . . 4
- 2.2 Réseaux de neurones bayesiens . . . 5

3 TP 3 - Uncertainty Applications **8**
- 3.1 Failure Prediction . . . 8
  - 3.1.1 Maximum Class Probability . . . 8
  - 3.1.2 Monte Carlo Dropout . . . 8
  - 3.1.3 ConfidNet . . . 8
- 3.2 Pourquoi mesure-t-on les performances avec AUPR plutôt que AUC ? . . . 9
- 3.3 Analyse des résultats . . . 9

A Annexe : TP2 - Approximate Inference **10**
  - A.0.1 MAP estimation . . . 10
  - A.0.2 Approximation de Laplace . . . 10

1 TP1 - Bayesian Linear Regression

1.1 Analyse des résultats de la distribution prédictive

On peut observer sur la figure que le degré d'incertitude (variance prédictive) du modèle augmente lorsqu'on s'éloigne des données d'apprentissage et est minimale au barycentre de ces dernières.

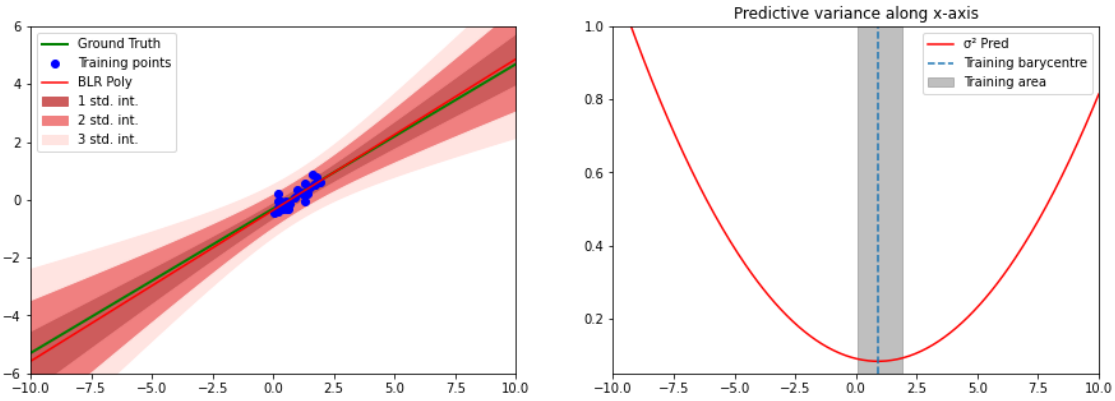

FIGURE 1 – Distribution prédictive

1.2 Preuve analytique pour $\alpha = 0$ et $\beta = 1$

On a :

$$\phi(X) = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$$

et :

$$\Sigma^{-1} = \alpha I + \beta \phi^T \phi$$

Pour $\alpha = 0$ et $\beta = 1$

$$\Sigma^{-1} = \phi^T \phi = \begin{pmatrix} N & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}$$

$$\Sigma = \frac{1}{det(\Sigma^{-1})} \begin{pmatrix} \sum_i x_i^2 & -\sum_i x_i \\ -\sum_i x_i & n \end{pmatrix} = \frac{1}{n\sum_i x_i^2 - (\sum_i x_i)^2} \begin{pmatrix} \sum_i x_i^2 & -\sum_i x_i \\ -\sum_i x_i & n \end{pmatrix}$$

La variance prédictive s'écrit alors de la manière suivante :

$$\sigma^2(X) = \phi^T(X)\Sigma\phi(X) = \begin{pmatrix} 1 & X \end{pmatrix} \Sigma \begin{pmatrix} 1 \\ X \end{pmatrix}$$

$$\sigma^2(X) = \frac{\sum_i x_i^2 - 2x\sum_i x_i + x^2 n}{n\sum_i x_i^2 - (\sum_i x_i)^2}$$

$$\sigma^2(X) = \frac{n(\sum_i \frac{x_i^2}{n} - 2x\bar{x} + x^2)}{n^2(\sum_i \frac{x_i^2}{n} - \bar{x}^2)}$$

$$\sigma^2(X) = \frac{1}{nV[X]}(\sum_i \frac{x_i^2}{n} - \bar{x}^2 + (x - \bar{x})^2)$$

$$\sigma^2(X) = \frac{1}{nV[X]}(V[X] + (x - \bar{x})^2)$$

$$\sigma^2(X) = \frac{1}{n} + \frac{(x - \bar{x})^2}{nV[X]} + \frac{1}{\beta}$$

2 TP2 - Approximate Inference

L'évaluation du posterior $P(\mathbf{w}|X, Y)$ est nécessaire dans plusieurs applications de modèles probabilistes. Cependant, l'évaluation de cette distribution peut s'avérer très compliquée dans certains cas de figure, par exemple, la dimensionalité de l'espace des paramètres peut être trop grande ou bien l'impossibilité d'avoir une forme analytique de cette distribution.

Dans ce TP, nous allons étudier et comparer des approches d'approximation, on considerera particulièrement les méthodes d'approximation Gaussienne (Approximation de Laplace), l'approximation par Inférence Variationnelle ainsi que la technique de Monte Carlo Dropout qui peut être considérée comme une approximation variationnelle.

2.1 Regression logistique bayesienne

2.1.1 Inférence variationnelle

La classe `LinearVariational` implémente une couche linéaire $w^T x+b$ avec inference variationnelle, w et b suivent une loi normale dont les paramètres sont appris : $\mathbf{w}_{\text{mu}}$, $\mathbf{w}_{\text{rho}}$, $\mathbf{b}_{\text{mu}}$ et $\mathbf{b}_{\text{rho}}$.

La méthode `sampling` permet de tirer un w et b selon les deux lois normales définies par $\mathbf{w}_{\text{mu}}$, $\mathbf{w}_{\text{rho}}$, $\mathbf{b}_{\text{mu}}$ et $\mathbf{b}_{\text{rho}}$, qui sont ensuite considérées dans le forward.

La méthode `kl_divergence` calcule la KL entre notre distribution et le prior.

La méthode `forward` calcule la sortie du modèle en appelant la méthode `sampling` pour obtenir un w et un b. Elle appelle également `kl_divergence` afin de calculer la KL entre le prior et la distribution q_θ et accumuler ensuite le résultat dans `accumulated_kl`.

L'inférence variationnelle consiste en des méthodes d'approximation bayesiennes qui nous permettent d'approximer la distribution posterior avec une distribution variationnelle q_θ en posant un problème d'optimisation : q_θ doit être le plus proche possible du posterior sur les données d'entrainement, on doit donc minimiser la divergence KL entre la distribution posterior et q_θ. Minimiser la KL revient à maximiser la borne inférieure de la distribution inconnue appelée "Evidence Lower BOund (ELBO)", les paramètres variationnels optimaux sont definis par :

$$\theta^* = \arg\max_\theta \mathbb{E}_{q_\theta(\boldsymbol{w})}[\log p(\mathcal{D} \mid \boldsymbol{w})] - \mathrm{KL}\left(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w})\right) \tag{1}$$

avec :

le premier terme de 1 correspond à la log vraissemblance et le deuxième terme à la divergence KL.

De plus, on peut ecrire :

$$\mathrm{KL}\left(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w})\right) = \int q_{\boldsymbol{\theta}} \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{w})}{p(\boldsymbol{w})} d\boldsymbol{w} = \int q_{\boldsymbol{\theta}} (\log q_{\boldsymbol{\theta}}(\boldsymbol{w}) - \log p(\boldsymbol{w})) d\boldsymbol{w}$$

Donc :

$$\mathrm{KL}\left(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w})\right) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log q_{\boldsymbol{\theta}}(\boldsymbol{w}) - \log p(\boldsymbol{w})\right]$$

Cette KL est estimée de manière stochastique (fonction `kl_divergence`) sur un seul point échantillonné de la distribution des paramètres. Comme ce sampling est stochastique, il est donc pas dérivable (le calcul du gradient est impossible dans ce cas), c'est pour cette raison que nous utilisons le *Parametrization trick* : $\boldsymbol{w}_s = \boldsymbol{\mu}_s + \boldsymbol{\Sigma}_s \odot \varepsilon$ avec

$\varepsilon \sim \mathcal{N}(0,1)$, ce qui permet de calculer le gradient de w_s selon les deux paramètres μ_s et Σ_s. C'est ce que nous faisons dans la fonction `sampling`. Dans le cadre de ce TP, nous posons une contrainte de non-negativité sur la variance, c'est pour cela qu'on considère à la place ρ tel que $\mathbf{\Sigma} = \log(1 + e^{\boldsymbol{\rho}})$.

Dans le `forward`, on échantillonne w_s et b_s à l'aide de la fonction `sampling` tel que $w_s, b_s \sim q_\theta$ (q_θ étant représenté par les paramètres de notre modèle variationnel). On utilise ensuite ces deux estimations pour calculer la sortie de notre modèle :

$\hat{y} = \sigma(w_s^T x + b_s)$.

Cela correspond à minimiser la loss :

$$\mathcal{L}_{\text{VI}}(\boldsymbol{\theta}; \mathcal{D}) = -\sum_{n=1}^{N} \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})} \left[\log p\left(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}\right) + \frac{1}{N} \left(\log q_{\boldsymbol{\theta}}(\boldsymbol{w}) - \log p(\boldsymbol{w}) \right) \right]$$

Etant donné que la loss peut se décomposer par rapport aux exemples d'apprentissage, on peut utiliser des méthodes de descente de gradient stochastique, un exemple étant un estimateur non biaisé du gradient. On considère également une reparametrisation pour pallier au problème de dépendance entre θ et le tirage des échantillons $\boldsymbol{w}_s \sim q_{\boldsymbol{\theta}}$.

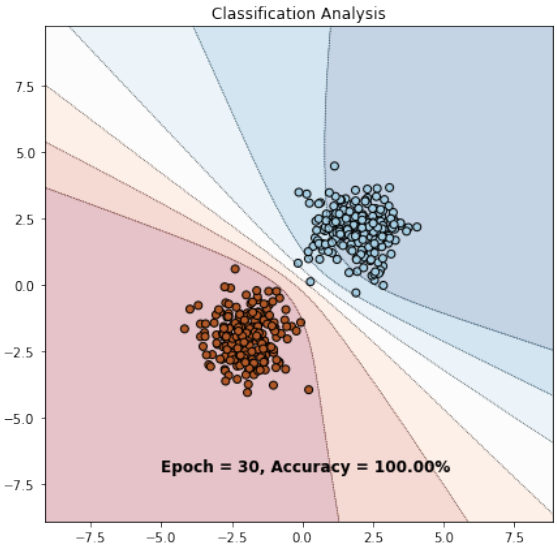

FIGURE 2 – Frontière de decision obtenue en considérant une approximation par inference variationnelle

Pareillement qu'avec l'approximation de Laplace (voir annexe), l'incertitude du modèle est prise en compte, les lignes de niveau se courbent d'avantage lorsqu'on s'éloigne des données d'entrainement. L'incertitude épistémique est donc bien reflétée.

2.2 Réseaux de neurones bayesiens

Dans cette partie, on considère des données non-linéaires. On a donc plus la dépendance linéaire entre $f^{\mathbf{w}}(x)$ et $\mathbf{w}$ qui nous permettait d'exprimer le posterior sous forme Gaussiene. En adaptant la méthode d'inférence variationnelle sur les données non-linéaire, nous obtenons la frontière de décision qu'on observe sur la figure 3.

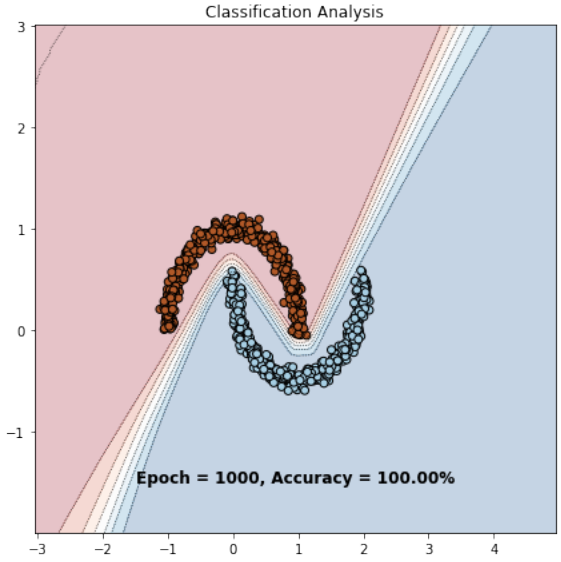

FIGURE 3 – Frontière de décision obtenue en considérant un réseau de neurones avec une couche cachée variationnelle

Comme avec les données linéaires, on obtient également une accuracy de 100% sur le dataset non linéaire. Cependant, on observe que l'incertitude est moins "marquée" lorsqu'on s'éloigne des données d'entrainement, bien que celle-ci soit présente. Notre intuition est que le passage au non linéaire avec ce dataset limite particulièrement le nombre de frontières de décision possible. [GAL et al.] [1] a montré que si l'on considère un réseau de neurones avec une profondeur quelconque et des activations non-linéaires, en appliquant du dropout, cela serait mathématiquement équivalent à une approximation variationnelle. Le dropout est donc un cas particulier d'une approximation variationnelle sur un Réseau de neurones bayésien.

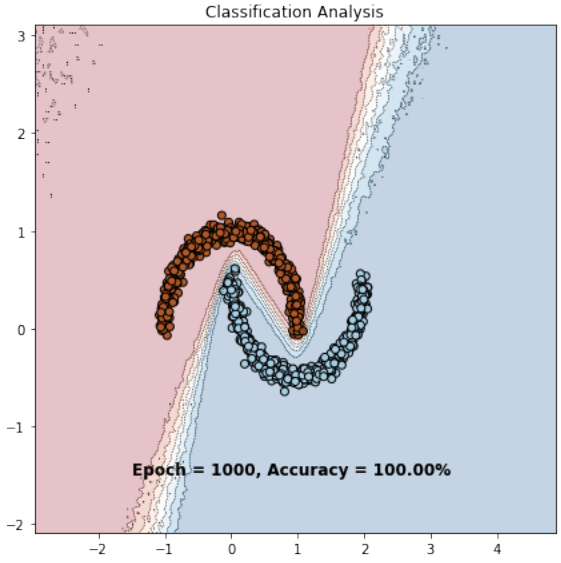

FIGURE 4 – Frontière de décision obtenue en considérant un réseau de neurones à une couche cachée avec du MC dropout

Sur la figure 4, on a bien une incertitude épistémique sur le modèle. Si on compare la méthode MC Dropout avec l'inférence variationnelle avec la régression logistique, cette dernière a un cout computationnel beaucoup plus grand que le MC Dropout. En effet, le nombre de paramètres du modèle est doublé et nécessite plus de temps à converger. Tandis, que l'estimation de l'incertitude peut être obtenu facilement et à un coût moindre avec le Dropout.

3 TP 3 - Uncertainty Applications

3.1 Failure Prediction

Généralement, lorsqu'on apprend un modèle on estime ses performances par des métriques (accuracy, précision, rappel ...). Cependant, en inférence on aimerait également être capables d'associer un degré d'incertitude à la prédiction et de ce fait être capables de prédire quand le modèle se trompe. Sur des tâches à haut risque (on peut penser à la conduite autonome par exemple) celà nous permettrait de penser à d'autres solutions (donner la main au conducteur par exemple) . Les processus bayesiens arrivent à modéliser cette incertitude sur des modèles simples, mais ils restent très peu tractables en deep learning. Il existe donc d'autres méthodes (un domaine encore assez exploratoire) pour estimer cette incertitude. Dans le cadre de ce tp on s'intéresse aux méthodes suivantes :

3.1.1 Maximum Class Probability

Pour les problèmes de classification, lorsqu'on utilise des réseaux de neurones, la façon la plus commune de faire est d'opter pour un coût d'entropie croisée à minimiser. L'entropie croisée permet de comparer 2 distributions de probabilités, les sorties de notre réseau doivent donc représenter une distribution de probabilité sur les différentes classes à prédire. Pour ce faire on applique un softmax sur les logits de sortie et la classe prédite sera la classe avec la plus grande probabilité. **La méthode MCP propose de retourner la probabilité associée à la classe prédite et de l'interpréter comme mesure de certitude**. Néanmoins, en pratique celà ne marche pas très bien car **les modèles de deep learning sont souvent "surconfiants"**.

3.1.2 Monte Carlo Dropout

La méthode MCDropout consiste à réaliser **plusieurs prédictions pour un même exemple**, en appliquant un dropout différent à chaque passe. **Les variations dans les prédictions sont ensuite mesurées pour établir une valeur d'incertitude à la prédiction**. Lors de ce tp, trois mesures sur ces prédictions ont été vues : **le var-ratio**, **l'entropie** et **l'information mutuelle**.

3.1.3 ConfidNet

La méthode ConfidNet se base sur les points faibles de la méthode MCP. Comme dit précedemment, le principal problème de la méthode MCP est que le modèle est surconfiant et celà principalement lorqu'il se trompe. Lorsque le modèle se trompe la probabilité associé à la classe eronnée est supérieure à celle associée à la véritable classe. L'intuition est donc que plutôt que de se baser sur la probabilité de la classe prédite, le modèle se base sur la probabilité de la vraie classe. Cependant, ceci est uniquement possible en phase d'entrainement (la vraie classe n'étant pas disponible en test). La méthode propose donc d'apprendre un modèle en parallèle du modèle de prédiction, ce modèle a pour objectif d'apprendre à prédire la probabilité de la vraie classe (comme un problème de régression).

3.2 Pourquoi mesure-t-on les performances avec AUPR plutôt que AUC ?

Sur ce type de modèle qui réalise un taux de bonne classification quasi parfait. Les exemples mal classés sont très peu nombreux, et de ce fait les ratio vrais positifs et faux positifs ne sont pas très significatifs. En revanche, la métrique AUPR prend en compte ce côté unbalanced entre les exemples bien classés et ceux mal classés.

3.3 Analyse des résultats

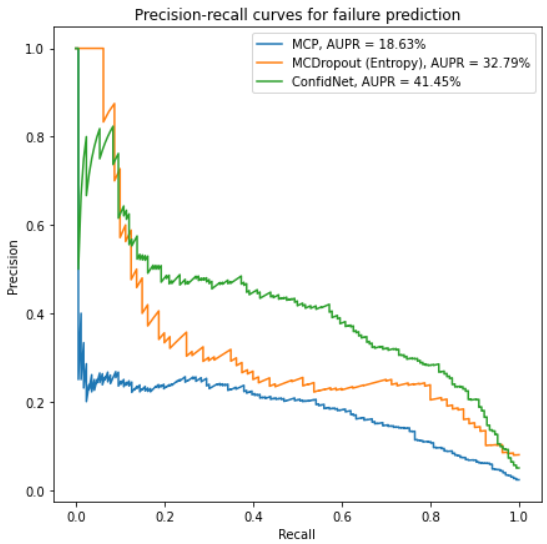

FIGURE 5 – Courbes précision rappel pour la prédiction d'échec

— Les méthodes MCDropout et ConfidNet ont toutes les deux de meilleurs performances globales et poinctuelles par rapport à la la méthode MCP. En effet, on peut voir que dès qu'on commence à considérer les exemples erronés (rappel > 0) la précision du modèle MCP s'effondre ce qui justifie bien d'un point de vue expérimentale ce qui a été dit précedemment (la méthode MCP est surconfiante quant à l'incertitude des exemples erronés)

— Pour un rappel entre 0 et 0.1 la précision du modèle MCDropout est supérieure à celle du ConfidNet, celà est justifié par l'aspect stochastique des executions.

— Globalement, la méthode ConfidNet a de bien meilleures performances (ponctuelles et moyenne) que les deux autres méthodes

A Annexe : TP2 - Approximate Inference

Dans cette partie, on considère un problème de classification binaire à l'aide d'une regression logistique : $f(\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)$ avec σ la fonction sigmoide.

Comparé à ce qu'on a vu avec la regression linéaire, on ne peut plus integrer dans l'espace des paramètres $\mathbf{w}$ étant donné que la distribution posterior n'est plus Gaussienne.

A.0.1 MAP estimation

On considère une baseline qui reduit la distribution posterior en un point $\mathbf{w}_{\mathbf{MAP}}$ tel que :

$$\arg\max_{\boldsymbol{w}} \prod_{n=1}^{N} p\left(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}\right) p(\boldsymbol{w})$$

.

Sur un dataset joué qui consiste en deux gaussiennes, observons la frontière de decision induite par $\mathbf{w}_{\mathbf{MAP}}$:

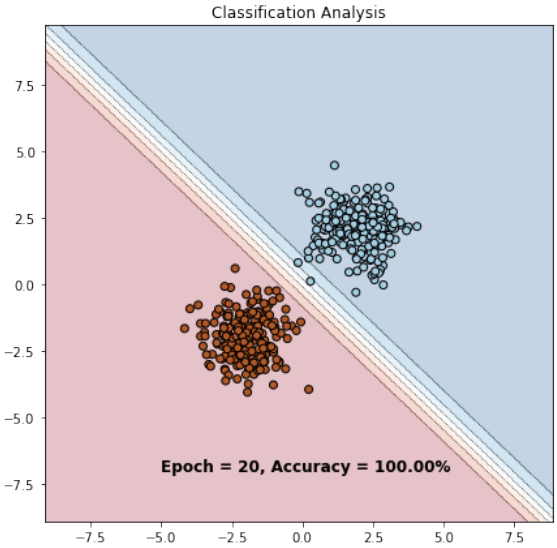

FIGURE 6 – Frontière de decision obtenue en considérant une estimation MAP

Le modèle considéré ici est un modèle déterministe dégénéré où on a uniquement considéré un point pour effectuer l'estimation, c'est à dire qu'on approxime la distribution prédictive avec une distribution de dirac. En utilisant cette méthode "plug-in" pour approximer $p(\mathbf{y} = 1|\boldsymbol{x}^*, \mathcal{D}) \approx p(y = 1|\boldsymbol{x}, \boldsymbol{w}_{\text{MAP}})$, au bout de 20 epochs, nous obtenons une frontière de décision satisfaisante, lorsqu'on s'éloigne de celle-ci, le modèle devient plus confiant, cependant, on remarque que l'on arrive pas à capturer l'incertitude épistémique, c'est à dire, faire en sorte que l'incertitude augmente lorsqu'on s'éloigne des données d'entrainement, ce qui n'est pas le cas ici.

A.0.2 Approximation de Laplace

On considère à présent une autre méthode simple qui vise à trouver une approximation gaussienne afin d'approximer la distribution posterior $P(\mathbf{w}|X, Y)$ en estimant la moyenne et la co-variance et cela en considérant l'optimum de l'estimation MAP.

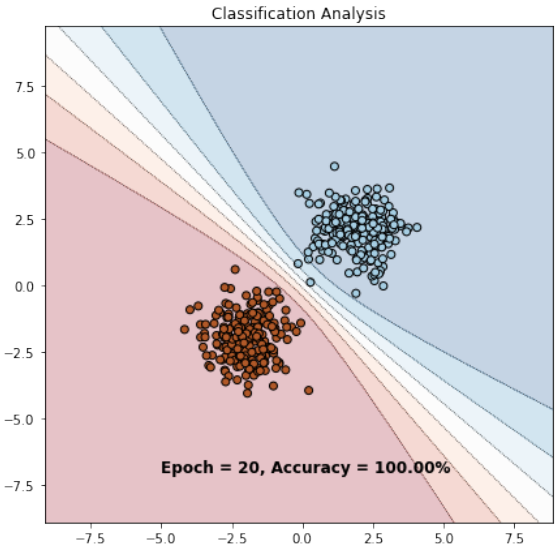

FIGURE 7 – Frontière de decision obtenue en considérant une approximation de Laplace

Avec l'approximation de Laplace, on prend en compte l'incertitude du modèle, en effet, on observe sur la figure ci-dessus que les lignes se courbent lorsqu'on s'eloigne des données d'entrainement : l'incertitude augmente bien lorsqu'on s'éloigne de ces dernières. Les limitations de cette méthode réside dans le fait qu'on effectue une approximation locale de la distribution au point $\mathbf{w}_{\text{MAP}}$ et on ignore les propriétés globales de la distribution.

Références

[1] Yarin GAL et Zoubin GHAHRAMANI. *Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning*. 2016. arXiv : 1506.02142 [stat.ML].