~~Valentyn Geruchte — 3702995~~
~~Léane Salais — 16003610~~
Rapport TP partie 2 – 10 février

# rdfia

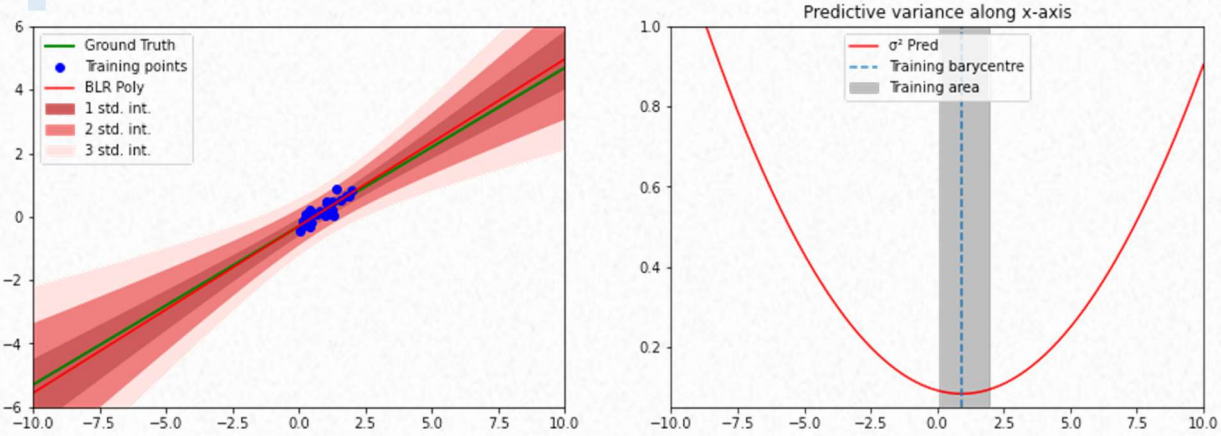
bayesian deep learning

Ce rapport traite les questions isolées. On trouvera toutes les autres réponses dans les Notebooks.

## TP 1 : Bayesian Linear Regression

On implémente la forme close de la régression bayésienne sur trois bases, linéaire, polynomiale et gaussienne. Entraîner le modèle sur des données 2D simples va permettre de visualiser la variance prédictive. Par elle, la régression bayésienne nous renseigne sur le potentiel d'erreur du modèle dans diverses régions de l'espace des données ; partout où elle augmente, elle indique une forte incertitude épistémique et nous incite à tempérer notre confiance dans les prédictions.

### 1 Affichage de la distribution prédictive dans le cas linéaire avec la BLR




**À gauche**, une visualisation de la prédiction du modèle (ligne rouge) *versus* la *ground truth* (ligne verte). Le modèle approxime plutôt bien la droite originale avec un léger décalage sur l'angle exact.
Les aires en nuances de rouge représentent son incertitude : leur largeur est proportionnelle à la variance prédictive, or elles sont resserrées près des points d'entraînement et s'étendent plus loin, là où l'écart avec la *ground truth* s'accroît. En mesurant cela, on peut donc comprendre même sans voir la *ground truth* que le modèle a des chances de se tromper là, et c'est tout l'intérêt de la régression bayésienne.

**À droite**, la projection de la variance prédictive sur une seule dimension pour plus de lisibilité. Dans la *training area*, l'algorithme est sûr de lui, la variance est faible et même minimale au niveau du barycentre des points d'entraînement ; elle augmente dès qu'on s'en éloigne et explose en l'absence totale de points.

### 2 Analyse théorique de sa forme avec $\alpha = 0$ et $\beta = 1$ : explication du comportement observé

Pour la calculer, on utilise la matrice de représentation des données $\Phi$ et la matrice de covariance $\Sigma$.

On a $\Phi = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$, $\Sigma^{-1} = \alpha + \beta\Phi^T\Phi$ et $\sigma^2(x^*) = \frac{1}{\beta} + \Phi(x^*)^T\Sigma\,\Phi(x^*)$, où les $\begin{matrix} x_1 \\ \vdots \\ x_n \end{matrix}$ sont les datapoints.

Avec $\alpha = 0$ et $\beta = 1$, on écrit donc $\Sigma^{-1} = \Phi^T\Phi = \begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum {x_i}^2 \end{pmatrix}$ et $\sigma^2(x^*) = \Phi(x^*)^T\Sigma\,\Phi(x^*)$.

Le calcul de l'inverse permet de déduire que $\Sigma = \frac{1}{n\sum {x_i}^2 - (\sum x_i)^2}\begin{pmatrix} \sum {x_i}^2 & -\sum x_i \\ -\sum x_i & n \end{pmatrix}$ en utilisant le déterminant.

**On prouve maintenant que la variance prédictive atteint un minimum au niveau du barycentre des points. Soit $x^*$ une nouvelle entrée, on peut déduire de tout cela les développements suivants :**

$$\sigma^2(x^*) = \Phi(x^*)^T \Sigma \Phi(x^*) = (1 \quad x^*)\Sigma\begin{pmatrix}1\\x^*\end{pmatrix}$$

$$= \frac{(\sum x_i^2 - x^*\sum x_i \quad -\sum x_i + nx^*)\begin{pmatrix}1\\x^*\end{pmatrix}}{n\sum x_i^2 - (\sum x_i)^2}$$

$$= \frac{\sum x_i^2 - x^*\sum x_i + x^*(-\sum x_i + nx^*)}{n\sum x_i^2 - (\sum x_i)^2}$$

$$= \frac{\sum x_i^2 - 2x^*\sum x_i + n(x^*)^2}{n\sum x_i^2 - (\sum x_i)^2}$$

$$= \frac{n\left(\frac{\sum x_i^2}{n} - \frac{2x^*\sum x_i}{n} + (x^*)^2\right)}{n^2\left(\frac{\sum x_i^2}{n} - \frac{(\sum x_i)^2}{n^2}\right)}$$

$$= \frac{\frac{\sum x_i^2}{n} - 2x^*\bar{x} + (x^*)^2}{n\left(\frac{\sum x_i^2}{n} - \bar{x}^2\right)}$$

$$= \frac{\frac{\sum x_i^2}{n} - 2x^*\bar{x} + (x^*)^2 + \bar{x}^2 - \bar{x}^2}{n\,Var(X)}$$

$$= \frac{\frac{\sum x_i^2}{n} - \bar{x}^2 + (x^*)^2 - 2x^*\bar{x} + \bar{x}^2}{n\,Var(X)}$$

$$= \frac{\frac{\sum x_i^2}{n} - \bar{x}^2 + (x^* - \bar{x})^2}{n\,Var(X)}$$

$$= \frac{Var(X) + (x^* - \bar{x})^2}{n\,Var(X)}$$

$$= \frac{1}{n} + \frac{1}{n\,Var(X)}(x^* - \bar{x})^2$$

**Sous cette forme, on voit que $\sigma^2(x^*)$ atteint son minimum $1/n$ lorsque $x^* = \bar{x}$ . On a bien le comportement attendu – $\bar{x}$ est bien le barycentre des points, à l'intérieur de la zone d'entraînement ; et du fait du carré, elle a une représentation parabolique et augmente de part et d'autre de ce barycentre.**

Généralement, on a vu dans ce TP que la variance prédictive augmentait en s'éloignant des points de *train*. Cela dit, avec les régressions non-linéaires, on a remarqué l'importance de l'*encoding* pour avoir ce comportement (cas des *features* gaussiennes). Une fois le bon *encoding* trouvé, l'avantage de la régression bayésienne par rapport aux méthodes déterministes sera de pouvoir pondérer les segments de prédiction par une notion d'incertitude, ce qui permet une utilisation critique du modèle.

# TP 2 : Approximate Inference

Une méthode semblable devrait pouvoir être utilisée en classification. Mais la mesure de la variance prédictive requerra des approximations, car la distribution postérieure ne sera plus analytiquement calculable comme en régression (plus de forme close). Un moyen de s'en sortir est d'utiliser l'inférence variationnelle.

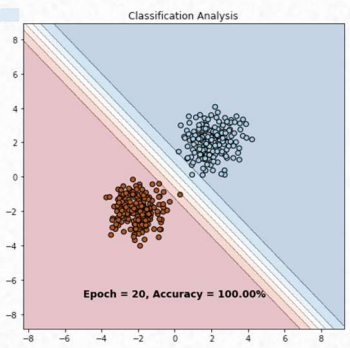
## 1 Principe de l'inférence variationnelle : classe `LinearVariational`

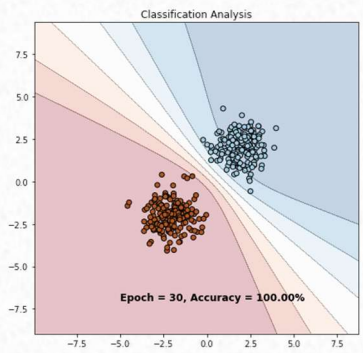
**L'inférence variationnelle permet d'estimer la sortie du modèle en fonction d'un échantillonnage sur ses paramètres. On a l'habitude d'avoir des paramètres fixes pour chaque couche ; ici, ils vont plutôt suivre une distribution. Grâce à cette notion de distribution, on pourra considérer plusieurs possibilités tout à fait *envisageables* pour les paramètres et obtenir une visualisation de leur effet. Détails techniques :**

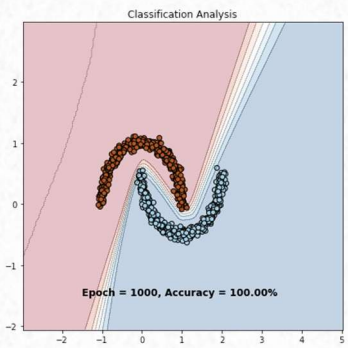
On voudrait apprendre une distribution variationnelle sur les poids du modèle dont la distance (divergence de Kullback-Leibler) à la distribution optimale doit être minimale. Comme on n'y a pas accès pour estimer exactement cette distance, **on passe par l'ELBO (Evidence Lower Bound)**. Maximiser l'ELBO, c'est maximiser la vraisemblance des données selon les poids, en régularisant par la distance à un prior sur ces poids.

Pour chacune des couches du modèle, on apprend en fait non pas directement les poids et biais, mais des moyennes et variances gaussiennes qui permettront de les représenter (**mean field**). Pour lancer un forward sur un exemple $x$, **on échantillonnera des paramètres pour chaque couche** selon ces moyennes et variances apprises ; et une approche de Monte-Carlo permettra finalement, à force de répétitions, d'obtenir une estimation empirique de la distribution associée à $x$ en sortie.

## 2 *Variational inference* sur classes linéairement séparables… …ou non



**Tout à gauche**, baseline du cas linéaire : via le *maximum a posteriori, on* approxime la distribution des classes en sortie par un Dirac (classe la plus probable) pour chaque exemple d'entraînement. De ce fait, les distributions n'ont aucune « amplitude », il n'y a qu'une seule possibilité. La certitude du modèle est de 1 en tout point de l'espace (à l'exception d'une aire très restreinte qui longe la frontière de classification) ; et même si la classification est parfaite, ce n'est pas ce qu'on veut.

**Toujours dans le cas linéaire, à côté**, l'approche *Variational Inference* détaillée plus haut permet de courber les lignes de niveau et met en lumière une incertitude épistémique. La classification est toujours parfaite, mais on sait maintenant que les prédictions du modèle ne sont pas systématiquement les mêmes dans certaines régions de l'espace. L'incertitude est désormais plus évidente près de la frontière entre les classes, et s'étend aussi à des aires éloignées du jeu d'entraînement – la couleur blanche étant signe d'une confusion totale sur ces zones. Cela rappelle le TP1 et c'était l'objectif.
Pendant l'entraînement, la distribution prédictive varie de façon spectaculaire avant de se stabiliser sur la fin. C'est lié au fait que les poids soient en cours d'apprentissage. Cela dit, les dernières epochs ne réduisent pas l'incertitude sur les zones claires : telle quelle, elle est donc indépendante de la quantité d'entraînement.

**A droite sur le cas non-linéaire**, la même approche VI montre une moins grande incertitude épistémique, venant d'une moins grande variance prédictive finale. On accuse la structure des données. Notre hypothèse est qu'il y a peu de formes de frontières possibles pour avoir une bonne classification, donc peu d'options pour les poids (là où dans le cas linéaire, la frontière peut avoir plusieurs angles…). On a expérimenté sans constater beaucoup de différence en élargissant le modèle à trois couches (cf. Notebook).

Dans le Notebook se trouvent les résultats d'une approche Monte-Carlo Dropout qui est plus rapide à coder – même si la preuve de son efficacité est plus complexe. Elle est plus satisfaisante sur le dataset non-linéaire, pour un modèle de même taille que le précédent : l'incertitude épistémique finale reste très évidente, avec des lignes de confiance troublées.

# TP 3 : Uncertainty applications

Observer la valeur de la Maximum Class Probability (MCP) ne suffit pas à montrer l'incertitude d'un modèle sur un exemple, car elle peut rester très élevée même pour une erreur. On explore ici d'autres techniques pour sonner l'alarme quand un classifieur est forcé de faire une prédiction peu fiable. C'est l'occasion d'utiliser de vrais datasets : on montrera que l'approche fonctionne sur MNIST.

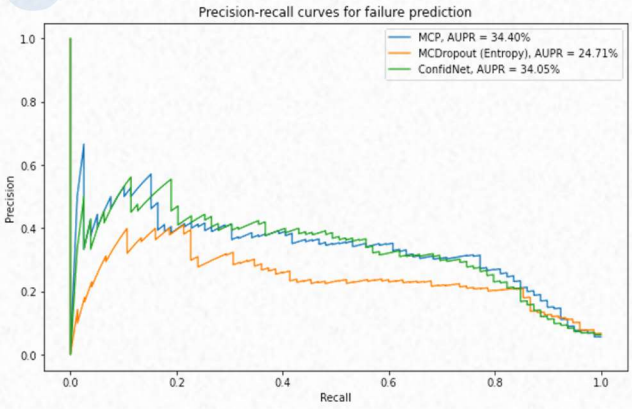
## 1 Principe de la *failure prediction*

Plutôt que la MCP, pour savoir si le modèle a raison ou tort, il faudrait plutôt regarder la probabilité de la classe qu'on aurait dû prédire (*True Class Probability*, TCP). Cette vraie classe n'étant pas disponible sur les données de test, on va entraîner un réseau parallèle (ConfidNet) qui apprend à estimer les TCP sur l'ensemble de train et qui va pouvoir extrapoler pour mesurer cela sur les ensembles de test.
**Là où la TCP approchée par ConfidNet est supérieure à $1/2$, on est à peu près sûr que le classifieur a raison. Là où la TCP est inférieure à $1/K$, où K est le nombre total de classes, il y a de grandes chances qu'il ait tort : on va donc déclencher une alarme de *failure risk*.**

## 2 Usage d'AUPR vs. AUROC pour l'évaluation

Pour évaluer les détecteurs d'erreurs, on réduit le problème à deux classes, *erreur* et *non-erreur*. *Erreur* étant la classe positive, on cherche une métrique qui permet de séparer vraies et fausses alarmes.
La courbe ROC permet de visualiser un équilibre entre le taux de vrais positifs et le taux de faux positifs *sur toutes les données* dans une classification. Or, si le dataset est déséquilibré (comme dans notre cas où l'on espère que les erreurs du classifieur initial sont rares), ces taux seront biaisés du fait du grand nombre de vrais négatifs. **Avec le score AUROC, on aura notamment l'impression que la performance du modèle de détection est exceptionnelle même s'il ignore en fait toutes les erreurs.**
**En revanche, le rapport PR (precision-recall) ne dépend que des prédictions positives (alarmes, légitimes ou pas) et non de toutes les données : l'AUPR évite le biais dénoncé, c'est donc une métrique plus adaptée.**

### 3 *Résultats obtenus*




Les courbes ci-contre permettent de quantifier la performance de ConfidNet vs. les baselines MCP et MCDropout en *failure prediction*. Même si les résultats se ressemblent pour un haut seuil de recall (quand il faut rappeler toutes les erreurs, ce qui impacte fortement la précision), **ConfidNet tient le haut du classement** : il a une meilleure précision quand le rappel requis est faible. L'AUPR finale élevée de MCP est due à la stochasticité des situations : on a énoncé les torts de MCP plus haut et on sait que c'est ConfidNet qui gagne en moyenne.

**On notera que dans tous les cas, les modèles n'ont pas des performances remarquables. Leur précision est quasi-nulle quand le recall s'élève, elle descend déjà sous 40% pour tous dès lors qu'il faut repérer plus de 20% des erreurs. On aura donc systématiquement un grand nombre de fausses alarmes, la technique est largement perfectible.**

Une dernière partie du TP a consisté à essayer de détecter des exemples hors-distribution avec la méthode ODIN (voir Notebook).

## Conclusion générale

Les explorations effectuées ont permis de mieux comprendre comment les méthodes bayésiennes pouvaient nous renseigner sur le potentiel d'erreur d'un modèle, quel que soit le dataset utilisé. Le cas de la régression était simple, la classification requérait plus de perspectives théoriques.

Pour la classification, certaines méthodes permettaient de relativiser la certitude sur la frontière apprise, en se fondant sur la variabilité des prédictions ; parmi elles, le MCDropout était le plus évident à mettre en œuvre. Facile à coder, il donnait aussi des résultats interprétables et des visualisations parlantes. Quant à la détection d'erreurs, on a constaté que tous les modèles étaient encore peu performants, qu'ils impliquent ou non le Dropout, et nous savons qu'il s'agit d'un axe de recherche actif.

Toutes ces méthodes permettent de créer des modèles plus fiables, dotés d'une forme de réflexivité, auxquels on accorderait plus facilement notre confiance une fois qu'ils seraient déployés (médecine, conduite autonome…).