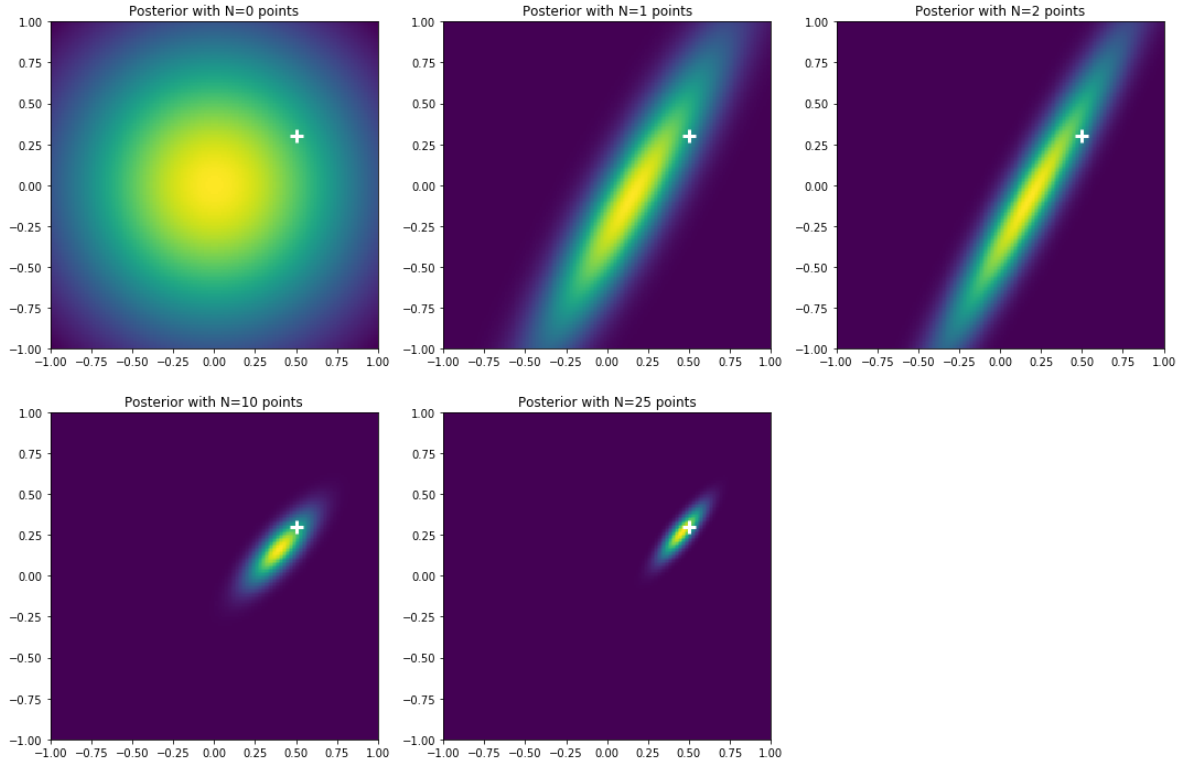
Keyvan BEROUKHIM
Arij RIABI

TP 1 : Distribution Prédictive Postérieure par Régression Linéaire Bayésienne

1.1
Pour faire de la régression linéaire, la fonction de projection non linéaire phi utilisée rajoute une colonne de 1 comme observation, ce qui permet d'utiliser un biais.

1.2
Étant donné les ensembles X et Y, la distribution postérieure **p (w | X, Y)** des poids w d'un modèle est proportionnelle au produit de la vraisemblance p(Y | X, w, β) et du prior p(w | α). α est un hyperparamètre définissant l'écart-type de la gaussienne du prior, il agit comme une régression L2 sur les poids du modèle. β est un paramètre représentant le bruit des données, il est ici connu.

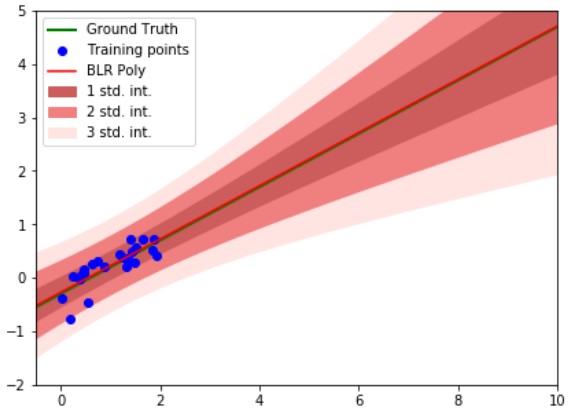
Avec une vraisemblance et un prior gaussiens et un modèle linéaire, la **forme du posterior est aussi gaussienne** et on peut la calculer de **manière analytique**. La gaussienne est très aplatie quand on n'observe aucun point (c'est le prior), elle se concentre et tend vers la vraie valeur quand le nombre de point augmente.

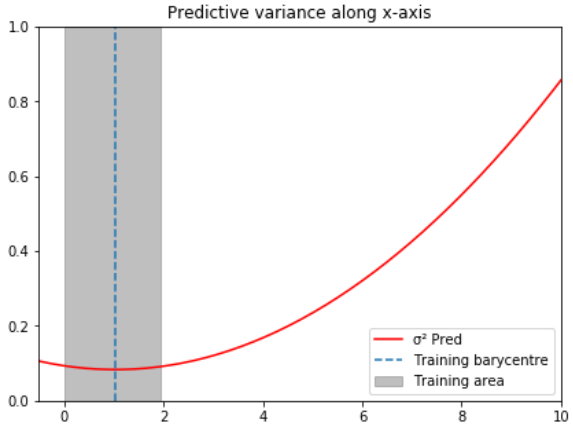


loi de distribution des deux coefficients de la régression linéaire (1D) estimée par MAP en fonction du nombre de points vus

1.3 1.4

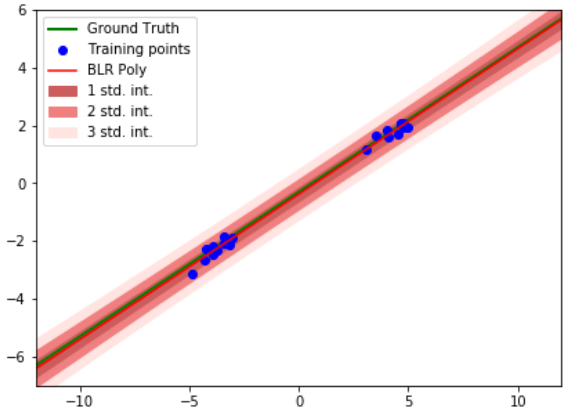
Une fois que la distribution postérieure des poids du modèle est apprise, on l'utilise pour calculer la **distribution postérieure prédictive p(y | x, X, Y)** du label y associé à une observation x. On l'obtient en marginalisant sur les poids en calculant ∫ p(y | w, x, X, Y) p(w | X, Y) dw. Cette distribution est ici aussi gaussienne et calculable de manière analytique.

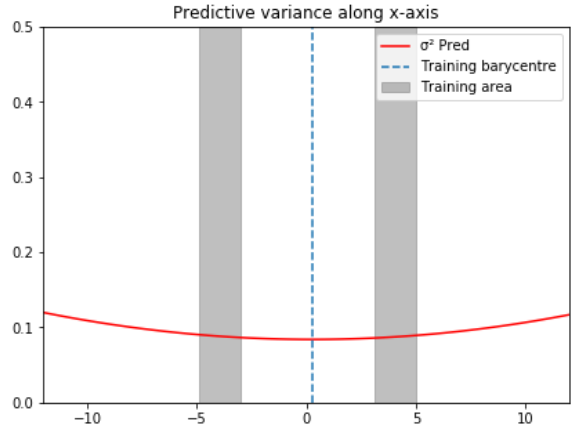




Le mode de la distribution postérieure prédictive correspond parfaitement à la distribution des données non bruitées. Les points du dataset de test se trouvent en général à moins de 2 écart-type de la valeur prédite. **La variance prédite augmente lorsqu'on s'éloigne des points du dataset, elle atteint son minimum pour le barycentre des données** et sa valeur y est 1/β.

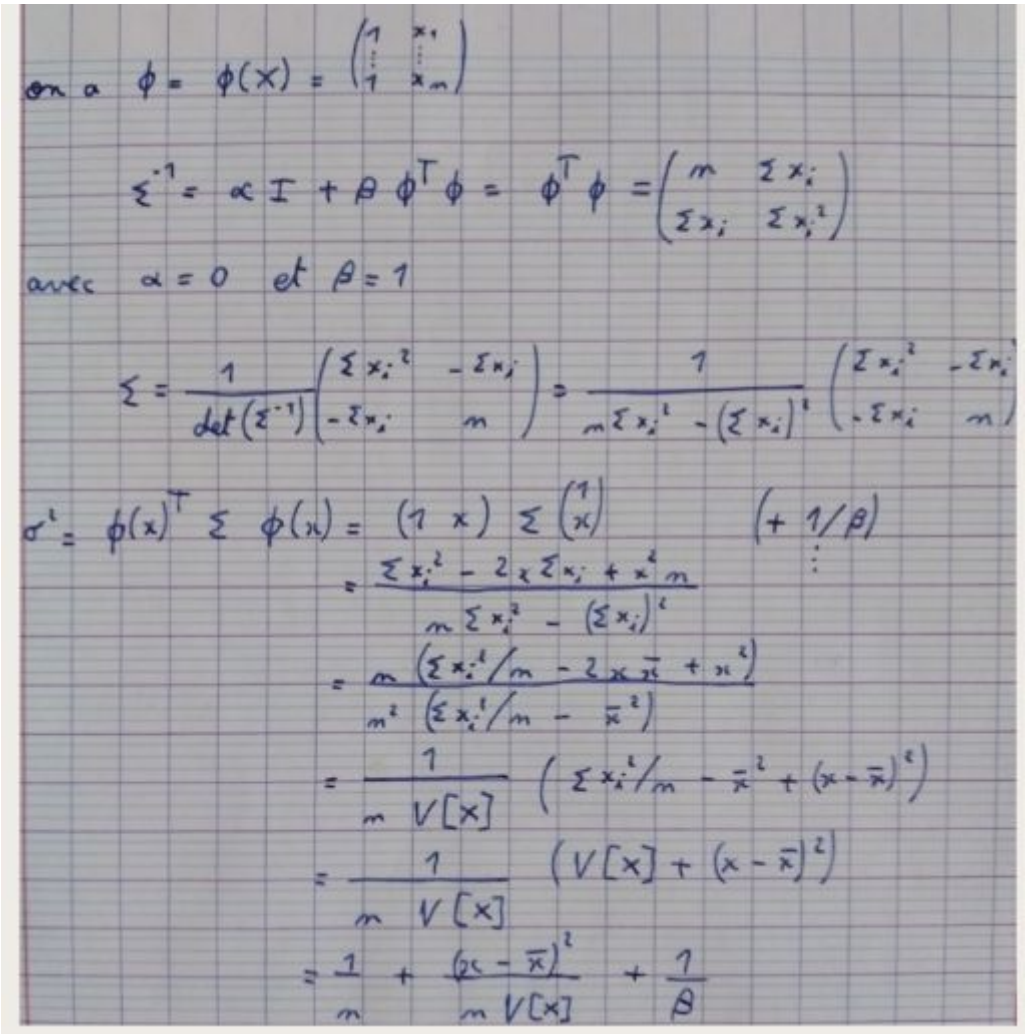
1.4 b





Le minimum de la variance prédite est encore atteint pour le barycentre des données, alors qu'on souhaiterait qu'elle y soit élevée car il n'y a pas de point dans cette zone. La variance du dataset étant élevée, quand on s'éloigne du centre l'incertitude augmente de manière très lente. En résumant le dataset par une moyenne et une variance, le modèle **ne prend pas en compte la position des points** comme l'aurait fait un SVM. Une autre manière de voir les choses est de dire que **le modèle n'est pas fait pour ce type de dataset**.

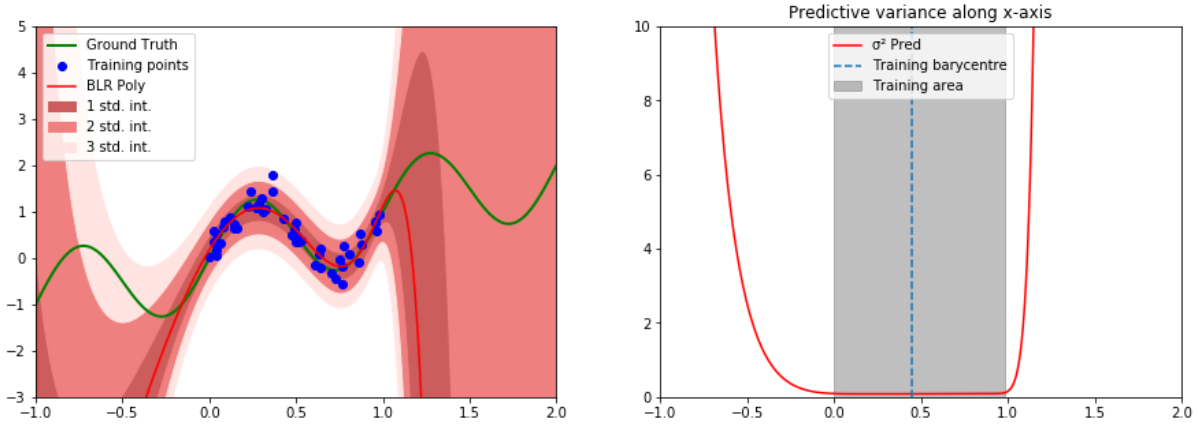
1.5



La variance prédite est proportionnelle à la **distance entre x et le barycentre** des données et est inversement proportionnelle à la **taille du dataset** et à la **variance du dataset**.

2.1 2.2

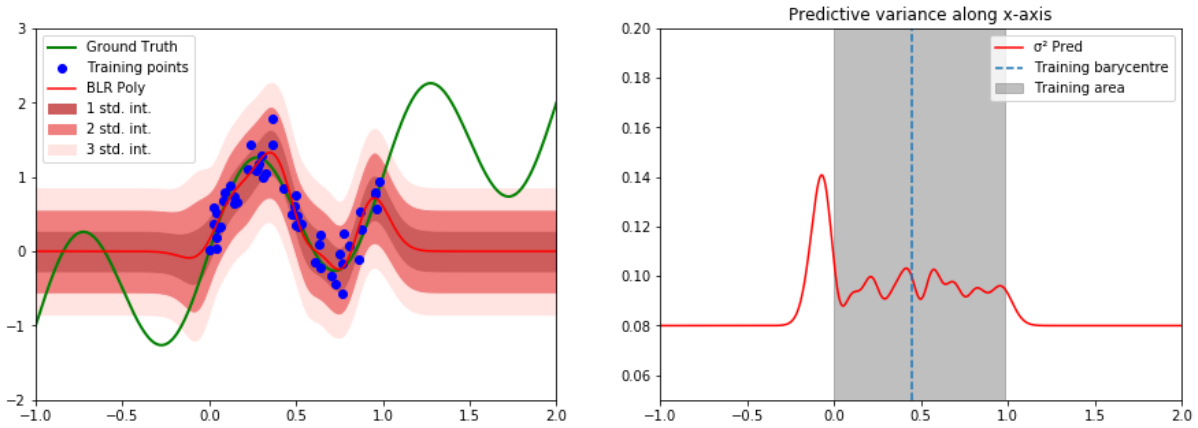
Pour faire de la régression **polynomiale**, la fonction phi rajoute une colonne de 1 ainsi que les x_i^d pour différents d.



Le modèle appris colle bien à la fonction à approximer au niveau des points du dataset. La **variance prédite augmente très rapidement** au delà des points vus.

2.3 2.4 2.5

Pour faire de la régression "**radius-based**", après qu'on ait maillé l'espace, la fonction phi représente x par son vecteur des 'distances' aux noeuds de la grille.



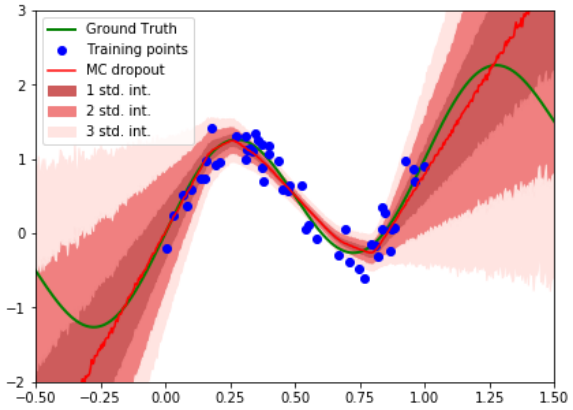
La variance n'est plus convexe. Quand x s'éloigne des noeuds du maillage, phi(x) tend vers le vecteur nul et donc la **variance prédite tend vers une petite valeur** (1/beta), on souhaiterait cependant qu'elle tende vers l'infini.

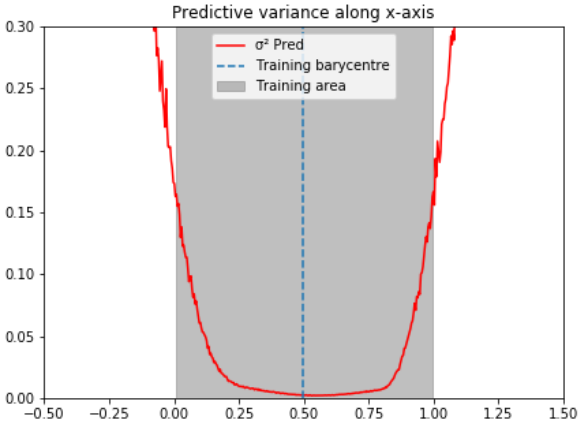
TP 2 : Approximation par Inférence Variationnelle et Monte Carlo Dropout

1.1

Avec des réseaux non linéaires, il n'existe pas de solution analytique pour calculer la variance prédictive d'un réseau. La technique de **Monte Carlo Dropout** consiste à utiliser du dropout dans le réseau en train et en test. Pour une entrée donnée, on obtient une variance de prédiction en réalisant plusieurs fois l'inférence.

1.2 1.3





Comme souhaité, la variance obtenue est faible au niveau des points du dataset et augmente quand on s'en éloigne. La courbe prédite a une allure différente de précédemment car on est passé d'un modèle polynomial à un MLP.

2.1 Pour faire de l'**inférence variationnelle**, on apprend ici deux paramètres (moyenne et écart-type) pour chaque poids du réseau et on approxime p(w | X, Y) par :

$$q_\theta(\mathbf{w}) = \prod \mathcal{N}(w_j|\mu_j, \sigma_j)$$

On minimise ensuite :

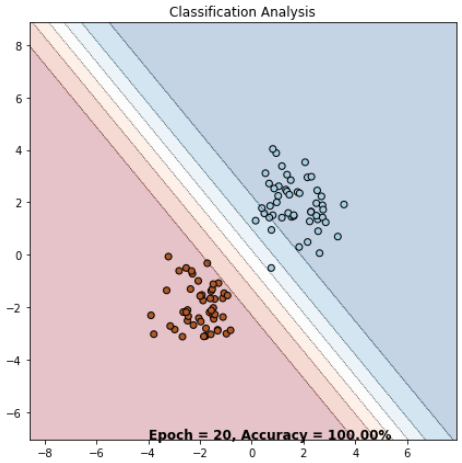
$$KL(q_\theta(\mathbf{w})||p(\mathbf{w}|\mathbf{X},\mathbf{Y})) = -\mathcal{L}_{VI}(\mathbf{X},\mathbf{Y},\theta) + \log p(\mathbf{Y}|\mathbf{X})$$

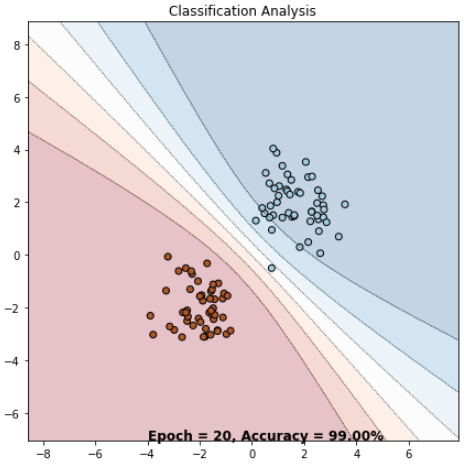
La "Variational Inference" loss est une "Evidence Lower BOund" ou "**ELBO**". Elle est définie par:

$$\mathcal{L}_{VI}(\theta) = \int q_\theta(\mathbf{w}) \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) dw - KL(q_\theta(\mathbf{w})||p(\mathbf{w})) \leq \log p(\mathbf{Y}|\mathbf{X})$$

Afin de minimiser cette loss par descente de gradient, on approxime l'intégrale par Monte Carlo Sampling en échantillonant des poids selon $q_\theta(\mathbf{w})$ et on utilise le "Reparametrization trick" pour échantillonner selon une gaussienne tout en restant dérivable.

2.2

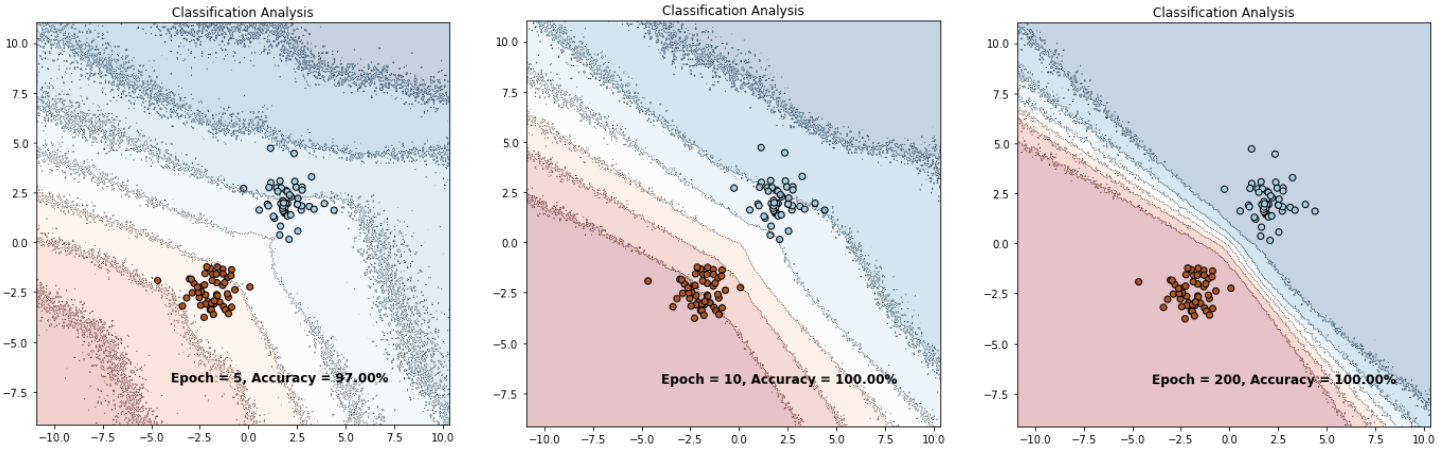




En faisant une régression logistique classique, le modèle associe des valeurs très fortes à des zones de l'espace loin de tout point du dataset. Avec le modèle d'inférence variationnelle, la certitude décroît quand on s'éloigne. Pour entraîner le modèle on se sert de la librairie d'inférence variationnelle "Pyro" basée sur PyTorch .

3.1

On peut aussi utiliser la technique du Monte Carlo dropout sur cette tâche de classification. En entraînant le modèle, celui-ci devient de plus en plus certain de ses décisions. On pourrait envisager de stopper l'entraînement à l'epoch 10 par exemple afin d'obtenir des scores de confiance plus interprétables tout en classifiant bien le dataset.

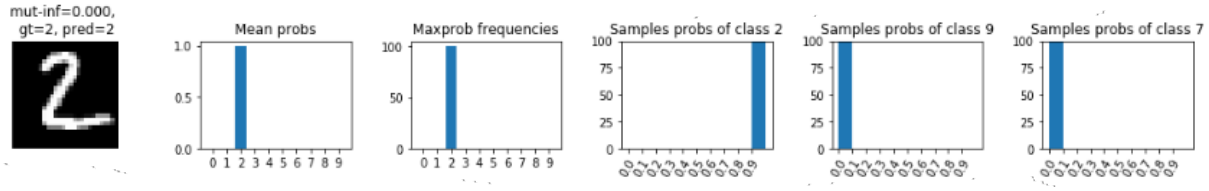


Pour obtenir ces courbes, on exécute le réseau (plusieurs fois) sur chaque pixel de l'image puis on regroupe les prédictions en quelques catégories arbitraires dont trace les contours. Les limites des catégories ne sont pas lisses car les prédictions du réseau ne le sont pas.

TP 3 : Application des Mesures d'Incertitudes

1.1
En entraînant un CNN simple avec du dropout sur MNIST, on obtient une précision en train et en test de 99%.

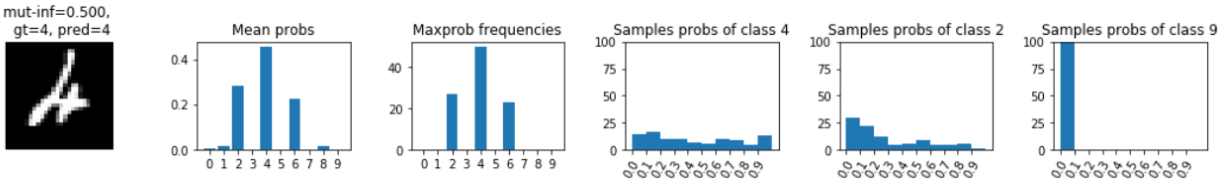
1.2
En utilisant le dropout en test aussi, le réseau classifie en général bien les images, quels que soient les poids annulés par le dropout :



Pour la plupart des images, le réseau prédit la bonne classe
le réseau prédit ici la classe 2 dans 100% des cas

1.3
Pour certaines images, le réseau ne **prédit pas tout le temps la même classe** selon les poids annulés par le dropout. Il existe dans ce cas plusieurs manières de définir une **incertitude à partir des différentes prédictions** :

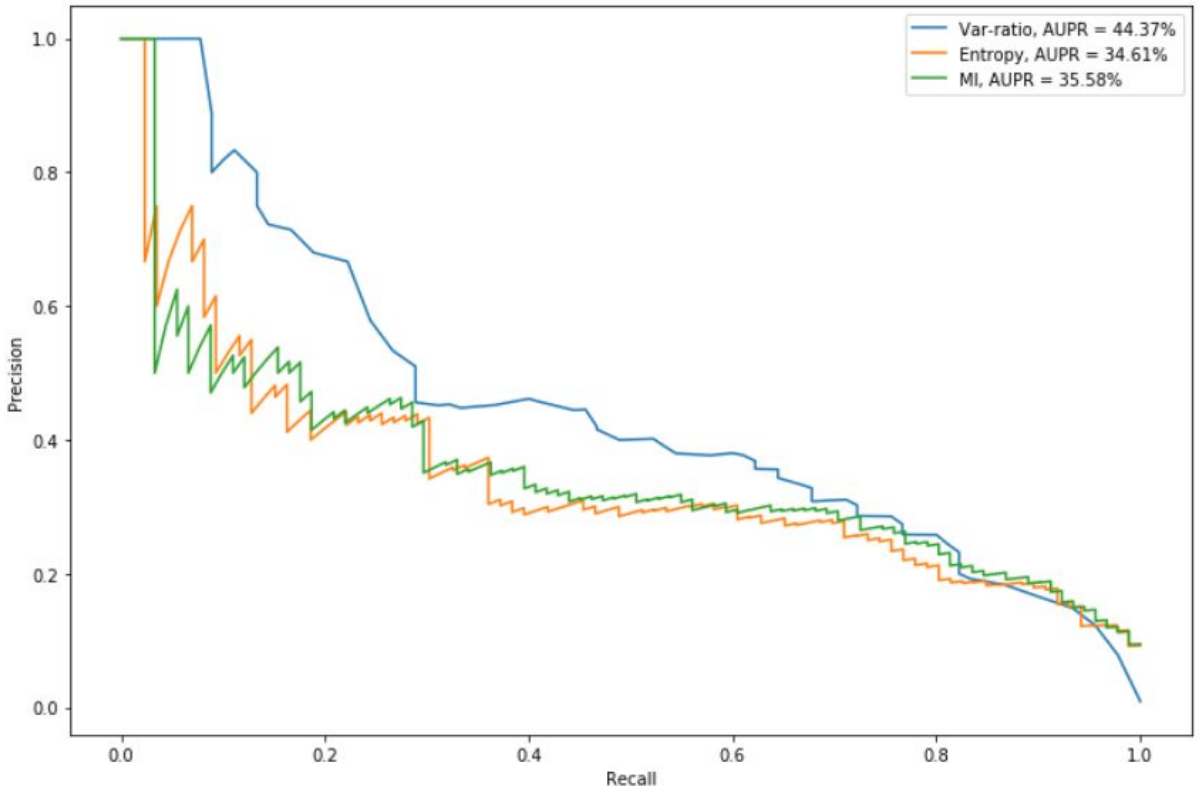
- **ratio de variation**: proportion des cas n'étant pas affectés à la classe majoritaire.
- **entropie**: quantité d'information contenue dans la distribution prédictive.
- **information mutuelle**: mesure à quel point connaître la sortie ou les poids du modèle fournit des informations sur l'autre.



Une des images pour laquelle le réseau est le moins sûr de lui (quelle que soit la métrique). Le réseau prédit en général un '4' un '2' ou un '6'. Les probabilités associées à la classe '4' sont à peu près équiréparties dans [0; 1], celles pour la classe '2' sont plus faibles.

2
Afin d'avoir une idée de la confiance que l'on peut accorder au réseau, on souhaiterait que les performances du réseau soient proportionnelles à sa confiance: on voudrait que le **réseau soit bien calibré**.

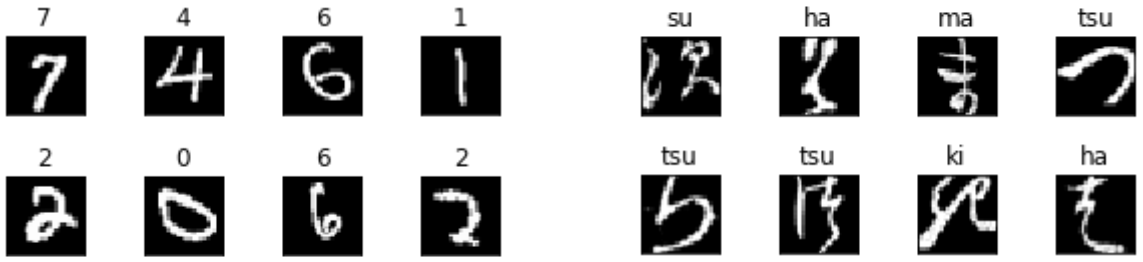
En triant les sorties du réseau par certitude pour **prédire ses erreurs** sur le jeu de test (les sorties les moins certaines devraient être celles ayant le plus de chance d'être erronées), on peut tracer une courbe de précision/rappel. Quelle que soit la manière de calculer l'incertitude utilisée, le réseau **n'arrive pas bien à prédire ses erreurs: pour prédire 20% des erreurs, le réseau se trompe une fois sur deux**.



La courbe optimale passerait par le coin supérieur droit.
Ici, le "var-ratio" est préférable aux autres métriques car il permet d'obtenir de meilleures performances

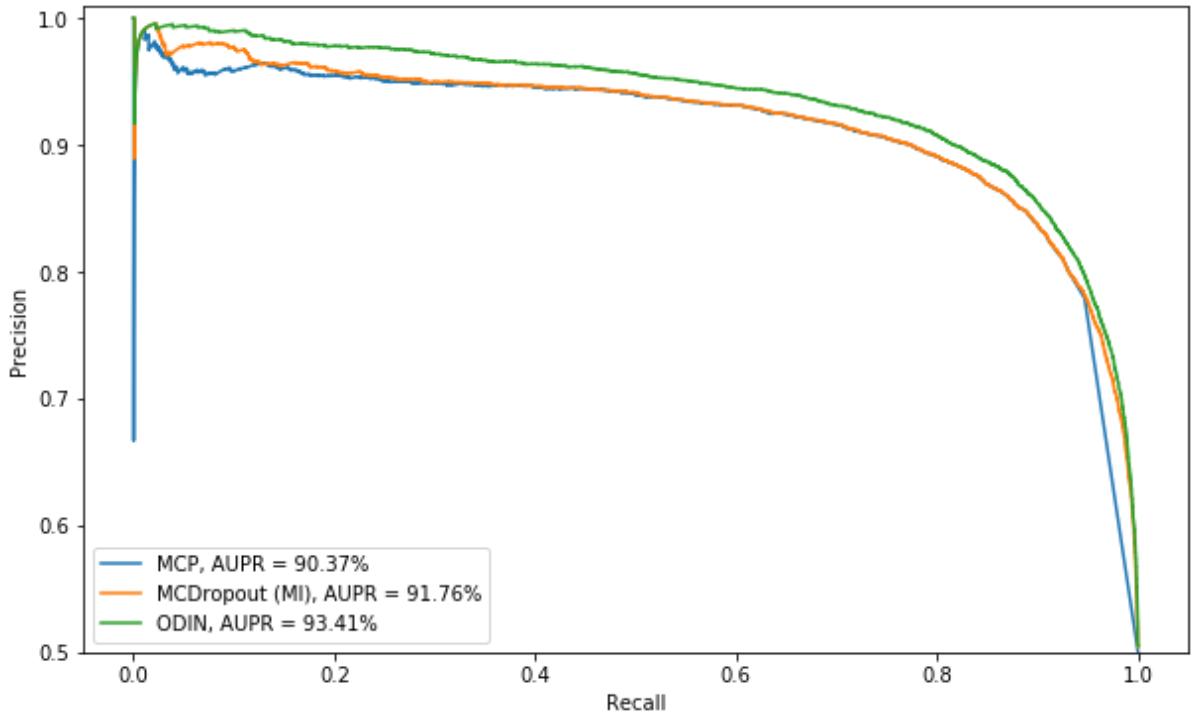
3

Après avoir entraîné notre réseau sur MNIST, nous lui présentons des kanjis sous le même format. En se servant de l'incertitude, on essaye de détecter les objets "**Out Of Distribution**".



datasets MNIST et KMNIST (Kanji)

En traçant les courbes précision/rappel, nous remarquons que le réseau a beaucoup plus de facilité pour cette tâche. **Pour détecter 90% des kanjis, le réseau a 10% de faux-positifs**, pour en détecter 100%, le réseau a 50% de faux-positifs.



CONCLUSION

La **robustesse** d'un modèle de machine learning peut être définie par plusieurs notions qui se recoupent : garantie de performances, stabilité des prédictions, capacité de généralisation, calibration de la certitude. Seuls des modèles suffisamment **performants, robustes et interprétables** peuvent être utilisés dans l'industrie. Nous avons ici pris en main certaines techniques en rapport avec ces notions.

Pour des tâches de **régression** ou de classification binaire, la **distribution postérieure prédictive** fournit plus d'information que le simple mode, ce qui rend le modèle **plus interprétable**. La variance de la prédiction est une **forme d'incertitude** du modèle.

La distribution postérieure prédictive n'est calculable de manière analytique que dans des cas simples et avec des modèles linéaires, mais on peut l'approximer de différentes manières. Par **inférence variationnelle**, en changeant la modélisation des poids du réseau, on peut maximiser une ELBO mais l'architecture du modèle et son entraînement perdent en interprétabilité. Une solution plus simple et interprétable consiste à faire du **Monte Carlo Dropout**, c'est à dire à utiliser du **dropout** en inférence sur les poids du réseau et à moyenner les résultats obtenus.

En plus du gain en interprétabilité du modèle, la variance prédite peut par exemple être utile face au dilemme exploration/exploitation en reinforcement learning.

Pour des tâches de **classification** multi-classe, on peut mesurer l'**incertitude du modèle** en utilisant du Monte Carlo Dropout. Il existe différentes métriques définissant une incertitude à partir de l'ensemble des prédictions. Connaître l'incertitude du modèle permet de rendre le modèle plus interprétable, mais l'incertitude peut aussi être utilisée par exemple pour la **prédiction d'erreurs** ainsi que la détection d'objets "**Out Of Distribution**".