TP 4 : Reconnaissance des formes pour l'analyse et l'interprétation d'images

Réalisé par :

Gaël Marec et Léo Hein

Table des matières

1 Bayesian Models

This first practical work aims at taking hand on simple Bayesian models, understand their functioning and gain finer insights on predictive distribution. We will work with Bayesian Linear Regression models using varying basis functions : linear, polynomial and Gaussian in a comparative intend to understand their respective behavior. We use different different 1D toy regression samples ranging from linear to more complex non-linear datasets such as increasing sinusoidal curves.

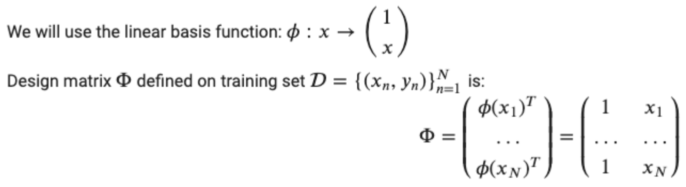
1.1 Linear Basis Function Model

We start with a linear dataset where we will analyze the behavior of linear basis functions in the framework of Bayesian Linear Regression.

1.1) The closed form of the posterior distribution can be written as :

$$\boxed{p(w|x, y) = \mathcal{N}(w, \beta\Sigma\Phi^T Y, (\alpha I + \beta\Phi^T\Phi)^{-1})}$$

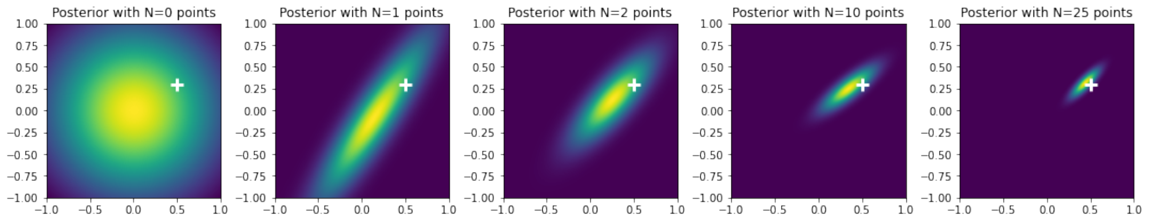
1.2) Using the following linear basis function, we compute the posterior sampling with an increasing number of fed points to discuss its evolution.



We will use the linear basis function: $\phi : x \to \begin{pmatrix} 1 \\ x \end{pmatrix}$

Design matrix Φ defined on training set $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ is:

$$\Phi = \begin{pmatrix} \phi(x_1)^T \\ \dots \\ \phi(x_N)^T \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ \dots & \dots \\ 1 & x_N \end{pmatrix}$$

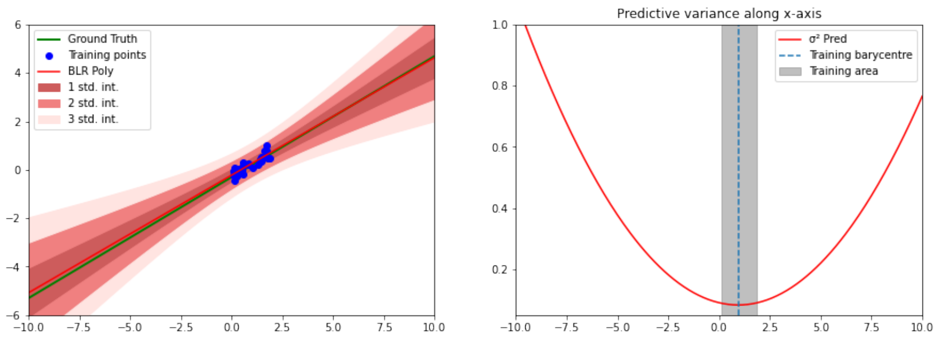


The first picture represents the prior Gaussian, while the other ones are posterior Gaussians. The more we give the model points to work on, the more the resulting distribution converges towards the real parameters, which is an expected behaviour. In particular, one can note that the variance of the posterior distribution quickly decreases with the number of points and already give good results with 25 examples only.

1.3) : The closed form of the predictive distribution can be written as :

$$\boxed{p(y* \,|x*, \mathcal{D}, \alpha, \beta) = \mathcal{N}(y*, \mu^T\Phi(x*), \frac{1}{\beta} + \Phi(x*)^T\Sigma\Phi(x*))}$$

1.4) We now use the last results to compute the predictions on the test dataset, and visualize the obtained results.

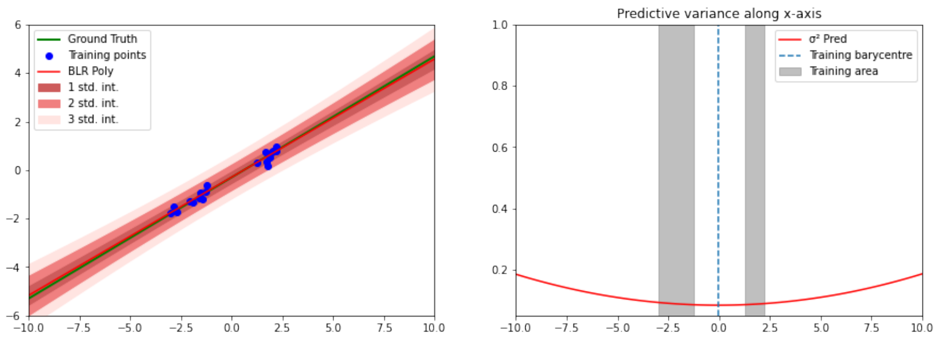


The uncertainty as well as the variance increase with the distance from the training points barycenter, in which the minimal value is reached. Intuitively, this fits well to the model as the only true information the model has are the training points, giving certainty in a local space. Analytically one has :

$$\boxed{\sigma^2_{\text{pred}} = \beta^{-1} + \Phi(x*)^T \Sigma \Phi(x*) \text{ with } \Sigma^{-1} = \alpha I + \beta \Phi^T \Phi.}$$

So, when $\alpha = 0$ and $\beta = 1$,one has $\Sigma = \Phi^T \Phi$. After computing, one has $\sigma^2 - \dfrac{1}{N} \propto (x - \bar{x})^2$ so σ^2 increases quadratically when x* is far from training data, as we observe in the results.

bonus question) We now compute the predictions on a different test dataset, presenting a "hole" in the distribution in order to compare the results with the first dataset.



Because the training points are more discarded, the influence of x* on σ^2_{pred} is weaker. Indeed, more values of σ are made plausible by the data. Moreover, we observe that the variance reaches a minimum where we don't have any training points (training data barycenter) and increases slowly when deviating from the data. As a matter of fact, our model is not able to perform well with this particular dataset as the data "positions" doesn't seem to be taken into account.

1.2 Non Linear Models

We now introduce a more complex toy dataset, which is an increasing sinusoidal curve. The goal of this part is to get insight on the importance of the chosen basis function on the predictive variance behavior.

1.2.1 Polynomial basis functions

We now use the following polynomial basis function with the sinusoidal dataset.

We will first use polynomial basis functions:

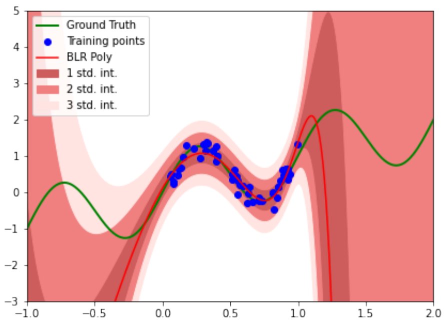
$$\phi : x \to \begin{pmatrix} \phi_0 \\ \dots \\ \phi_{D-1} \end{pmatrix}$$

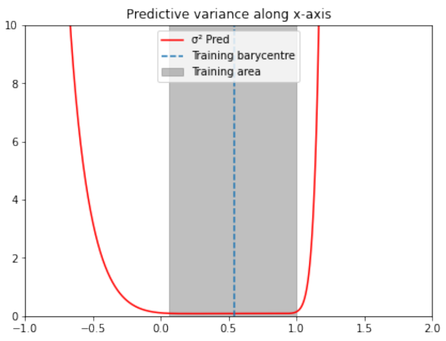
where $\phi_j = x^j$ for $j \geq 0$ and $D \geq 0$

Design matrix Φ defined on training set $\mathcal{D}$ is:

$$\Phi = \begin{pmatrix} \phi(x_1)^T \\ \dots \\ \phi(x_n)^T \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{D-1} \\ \dots & \dots & \dots & \dots & \\ 1 & x_N & x_N^2 & \dots & x_N^{D-1} \end{pmatrix}$$

We implement the closed form with polynomial features and visualize the results.





2.1) Predictive variance drastically increases as we move away from training data. In addition, the minimum is not unique anymore as it is reached on the entire [0,1] segment, i.e. the training area, meaning that there is no randomness anymore near the data. Our model performs very well around the training area.

1.2.2 Gaussian basis functions

We now use the following Gaussian basis function with the sinusoidal dataset.

Now, let's consider gaussian basis functions:

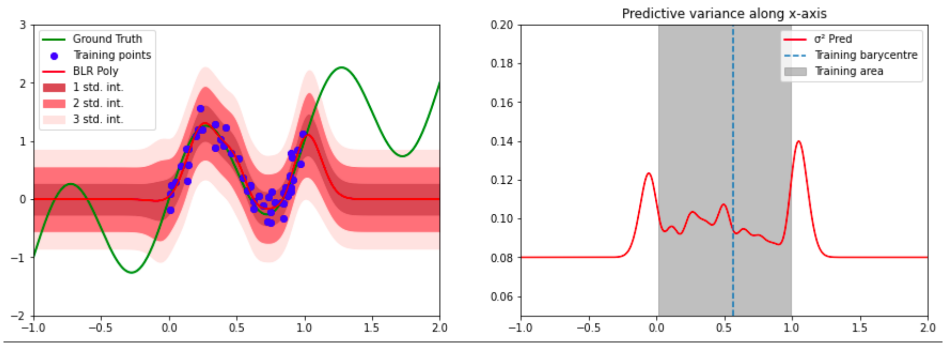
$$\phi : x \to \begin{pmatrix} \phi_0 \\ \dots \\ \phi_{D-1} \end{pmatrix}$$

where $\phi_j = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$ for $j \geq 0$

Design matrix Φ defined on training set $\mathcal{D}$ is:

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_{D-1}(x_1) \\ \dots & \dots & \dots & \dots \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_{D-1}(x_N) \end{pmatrix} = \begin{pmatrix} e^{-\frac{(x_1-\mu_0)^2}{2s^2}} & e^{-\frac{(x_1-\mu_1)^2}{2s^2}} & \dots & e^{-\frac{(x_1-\mu_{D-1})^2}{2s^2}} \\ \dots & \dots & \dots & \dots \\ e^{-\frac{(x_N-\mu_0)^2}{2s^2}} & e^{-\frac{(x_N-\mu_1)^2}{2s^2}} & \dots & e^{-\frac{(x_N-\mu_{D-1})^2}{2s^2}} \end{pmatrix}$$

We implement the closed form with Gaussian features and visualize the results.



2.2) The minimal variance does not correspond with the training points barycentre anymore. What strikes first is that the variance minimum is obtained when one is far away from the training region. Indeed the variance converges to a value of 0.08 on both sides of the training points, which is inconsistent to the previous results and the intuitive sense of the model. One can note two main pics of variance at the border of the training region. While the increase of variance is expected, the following decrease is quite unintended. Inside the training region, the variance is quite chaotic, varying a lot, but the values remain quite low. More qualitatively, on the left graph, we can see the issue as the model apprehend quite well the data in the training region but behave as if the dataset was linear outside.

2.3) The variance converges to the value of 0.08 because it corresponds to two times the variance of the added noise (centered normal distribution) which has a standard deviation of $= 0.2$. $(2 \times \sigma^2)$

2 Variational inference

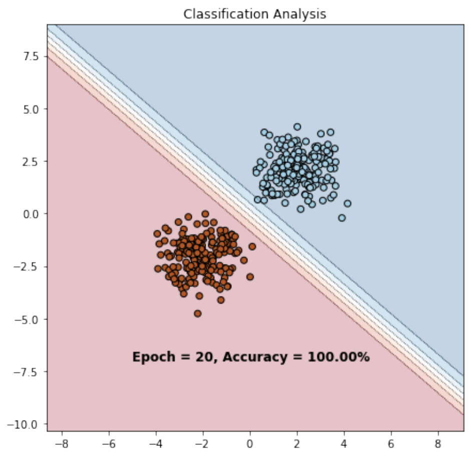
In classification tasks, even for a mere Logistic Regression, we don't have access to a closed form of the posterior $p(wD)$. Unlike in Linear regression, the likelihood isn't conjugated to the Gaussian prior anymore. We will need to approximate this posterior.

In this practical work, we aim at taking hand on approximate inference methods and understand how they work on linear and non-linear 2D datasets. To do so, we will explore and compare approximate inference approaches on 2D binary classification datasets. Studied approaches include Laplacian approximation, variational inference with mean-field approximation and Monte Carlo dropout.

2.1 Bayesian Logistic Regression

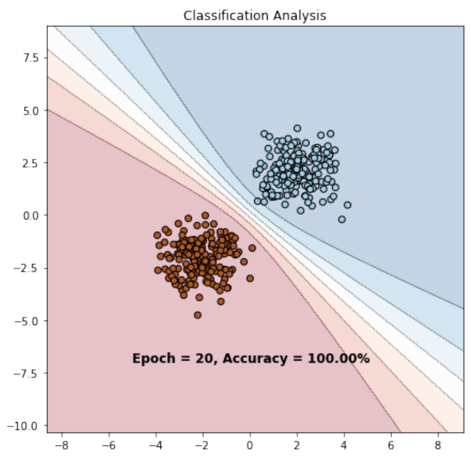
2.1.1 Maximum-A-Posteriori Estimate

1.1) We obtained a linear separation of the two classes. By looking $p(y = 1|x, w_{\text{MAP}})$ we see that the uncertainty does not increase when the data is far from the training points (the color is uniform across the border). The model remains confident far from the border.



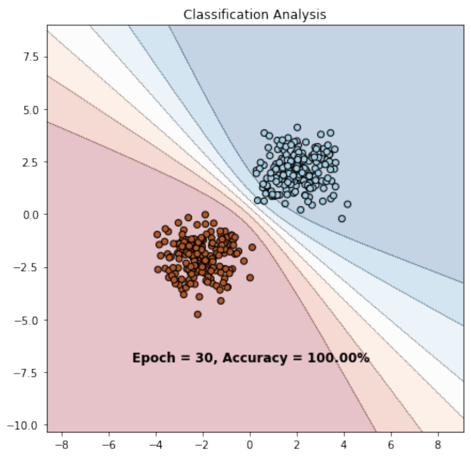
2.1.2 Laplace Approximation

1.2) The separation is not linear anymore, we observe a curved separating boundary. The certainty now decreases along the border when moving away from the data. Moreover, in order to maintain the same level of confidence along the border, one must at the same time move away perpendicular to the border.



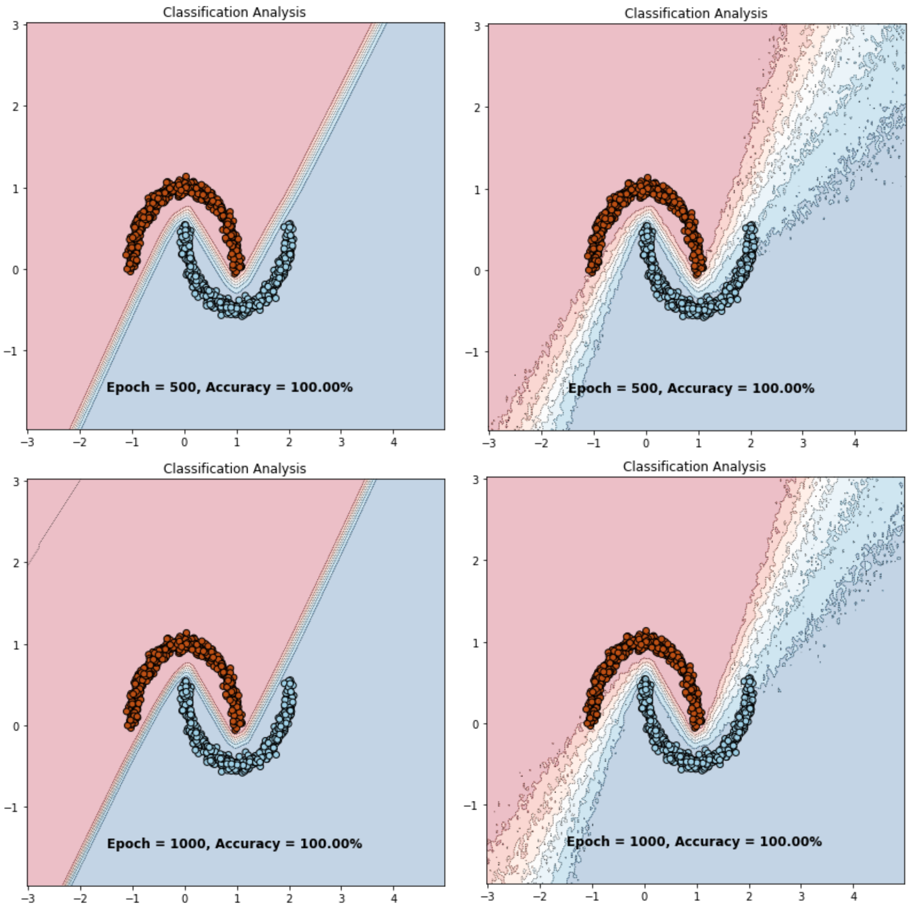
2.1.3 Variational Inference

1.3) The data still is well separated, the variance seems to have increased around the decision border, which is more and more striking as one get away from the data. The results with the variational layer still remain quite similar to the previous one obtained with Laplace approximation. In addition, in all three methods, the regions behind the data doesn't present any uncertainty, the classification is certain.



2.2 Variational Inference with Bayesian Neural Networks

2.1) With the two methods, BNN and MC dropout, the estimated classifications perfectly fit the topology of the moon-shaped data after at most 500 epochs. As expected, the variance and thus the uncertainty increase when going away from the data. Moreover the shape of prediction are the same, so we experimentally showed that training BNN with variational posterior approximation is equivalent to MC with dropout. We can reduce the variance by increasing the number of epochs, as shown in the two bottom figures, in which the "cone" shape on each side of the data is thinner. Some of the benefits of MC Dropout variational inference over Bayesian Logistic Regression with variational inference is that there is a possibility to change the dropout rate, the complexity is smaller and it is easier to implement.


Classification Analysis
Epoch = 500, Accuracy = 100.00%
Classification Analysis
Epoch = 500, Accuracy = 100.00%
Classification Analysis
Epoch = 1000, Accuracy = 100.00%
Classification Analysis
Epoch = 1000, Accuracy = 100.00%

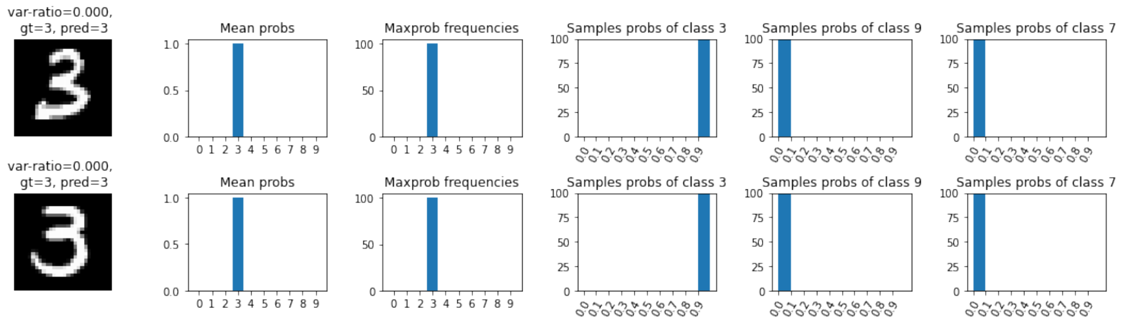
3 Uncertainty application

This last practical work aims at taking hand on applying uncertainty estimation for failure prediction and out-of-distribution detection. To do so, we will focus on applications based on uncertainty estimation. We will first use MC Dropout variational inference to qualitatively evaluate the most uncertain images according to the mode. Then, we'll move to 2 examples where good uncertainty estimation is crucial : failure prediction and out-of-distribution detection.

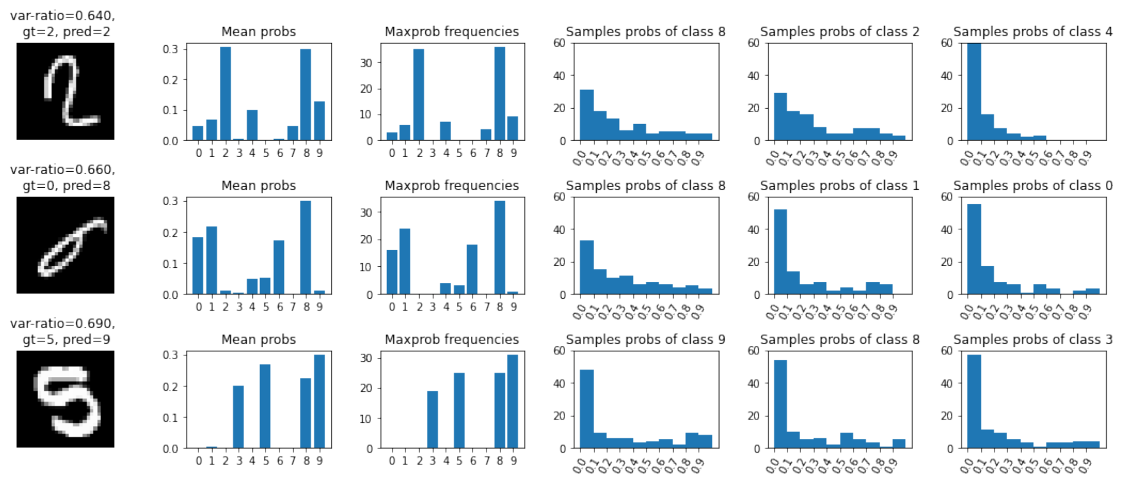
3.1 Monte-Carlo Dropout on MNIST

By applying MC Dropout variational inference method, we're interested to obtain an uncertainty measure which can be use to spot the most uncertain images in our dataset.

1.1) In the random images just below, the prediction is made with certainty which is illustrated by the unique pic in the histograms, corresponding to a probability of 1 to the associated class.



In the worst images (considering the var-ratio) a human eye cannot discern with certainty the value of the represented number. For example, the last number seems to be a five, but it could also be a 9 as well. No one can say with certainty. This behavior is also illustrated in the predictions of our model. Indeed, the "mean probs" histograms show that the model is uncertain as several classes present a non-zero probability. With our example, the model seems to hesitate between 5 classes : 3,5,8 and 9 having high and similar probabilities and a very low probability for the class 1. So one can see a failed prediction if no particular class probability seems to override the others, which is what we can see for each of the following pictures.



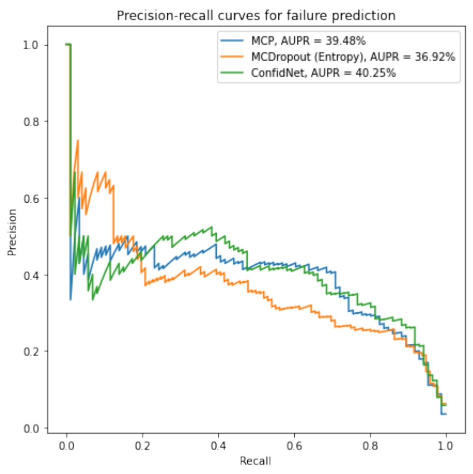
3.2 Failure prediction

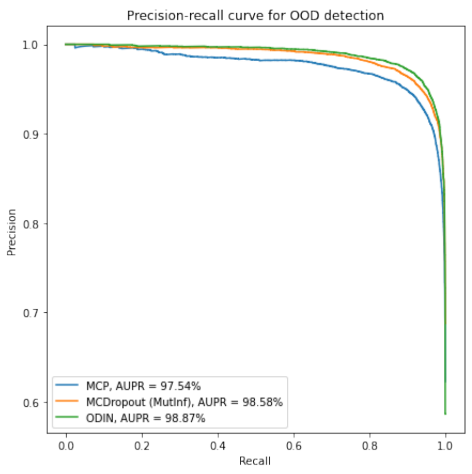
The objective of failure prediction is to provide confidence measures for model's predictions that are reliable and whoseranking among samples enables to distinguish correct from incorrect predictions. Equipped with sucha confidence measure, a system could decide to stick to the prediction or, on the contrary, to handover to a human or a back-up system with, e.g.other sensors, or simply to trigger an alarm.

2.1) We obtain the following ranking : ConfidNet > MCP > MCD.

However, one can see that for each of the three methods, the network has difficulties predicting its errors : 39.48% for MCP, 36.92 % for MCDropout entropy and 40.25 % for Confidnet. One would like the curves to reach the top right part of the graph. As we can see, the precision to detect at least 10% of errors doesn't exceed 50%, which are quite bad results.

AUPR is more sensitive to improvements for the positive class, which is our classification error. AUPR is based on the positive predictive value (PPV) and the true positive rate (TPR) whereas AUROC is based on the true positive rate (TPR) and the false positive rate (FPR). As a matter of fact, AUROC is more appropriate when one wants to evenly take into account positive and negative classes (when observations are balanced between the two classes). However, in this example, we are interested in the positive class only, AUPR is more relevant. (precision-recall curves are appropriate for imbalanced datasets.)





3.1) We obtain the following ranking : ODIN > MCDropout > MCP.

For this task, the different methods present very good results as the curves nearly reach the top-right corner, which is the optimum. For example, the recall for 80% is higher than 0.95% for all methods. The ODIN is particularly efficient.