Faculté des Sciences et Ingénierie - Sorbonne Université

RDFIA Lab Report

Section 3 : Bayesian deep learning

Yuxuan XI Mojan Omidi

Lab Supervisor : Dr. Alasdair Newson

University year :
2023-2024

Contents

Chapter 8

Bayesian Linear Regression

8.1 Linear Basis function model

We will use the linear basis function: $\phi : x \rightarrow (1, x)$

Design matrix Φ defined on training set is:

$$\Phi = \begin{bmatrix} 1 & x_1 \\ ... & ... \\ 1 & x_n \end{bmatrix}$$

$\rightarrow$ **Question 1.1:** Code linear basis function.

Add a constant term (bias term) to the input data. The code is in the Jupyter notebook file.

$\rightarrow$ **Question 1.2:** Recall closed form of the posterior distribution in linear case. Then, code and visualize posterior sampling. What can you observe?.

The formulas for the mean μ and covariance matrix Σ of the posterior distribution are as follows:

$$p(\omega|D, \alpha, \beta) = \mathcal{N}(\omega, \mu, \Sigma)$$
$$\Sigma^{-1} = \alpha I + \beta \Phi^T \Phi$$
$$\mu = \beta \Sigma \Phi^T Y$$

The code is in the Jupyter notebook file.

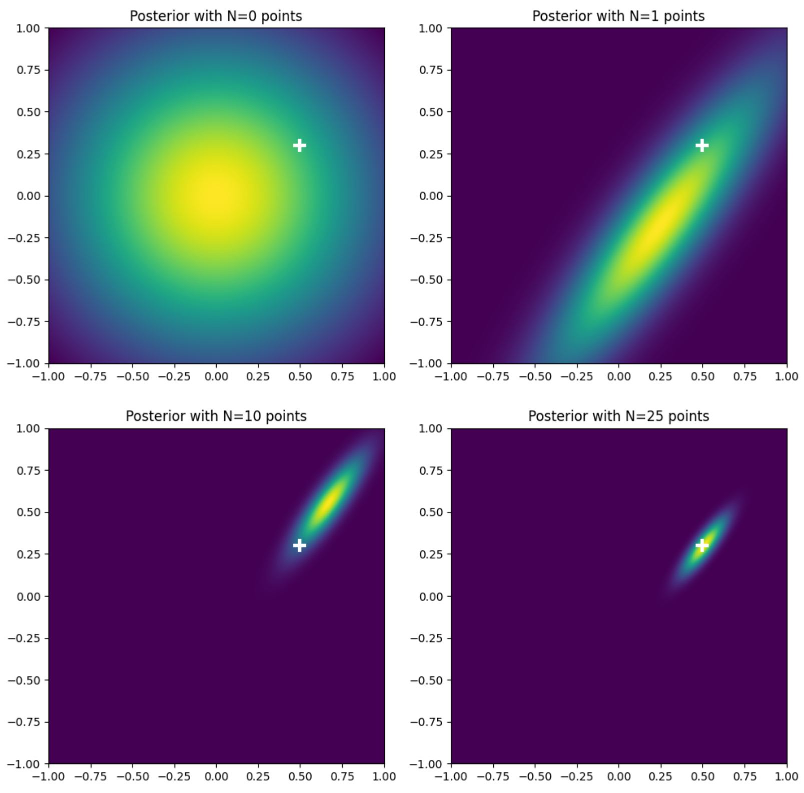


Figure 8.1 – Posterior distribution

These figures show the visualization of the posterior distribution for different numbers of data points, for example $N = 0, 1, 2, 10, 25$. The white cross point is the real data point, which is $(0.5, 0.3)$.

The first figure is for $N = 0$. In this case we have not added any values so it is uniformly distributed and can be said to be the original model. Later, as the number of data points increases, a posterior distribution begins to form, showing preferences for parameter values. The more data points there are, the more concentrated the posterior distribution is toward the true values.

$\rightarrow$ **Question 1.3:** Recall and code closed form of the predictive distribution in linear case.

The predicted mean and predicted variance are as follows:

$$\mu_{pred}(x^*) = \mu^T \Phi(x^*)$$

$$\sigma^2_{pred}(x^*) = \frac{1}{\beta} + \Phi(x^*)^T \Sigma \Phi(x^*)$$

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

The code is in the Jupyter notebook file.

→ **Question 1.4:** Based on previously defined f_pred(), predict on the test dataset. Then visualize results using plot_results() defined at the beginning of the notebook.

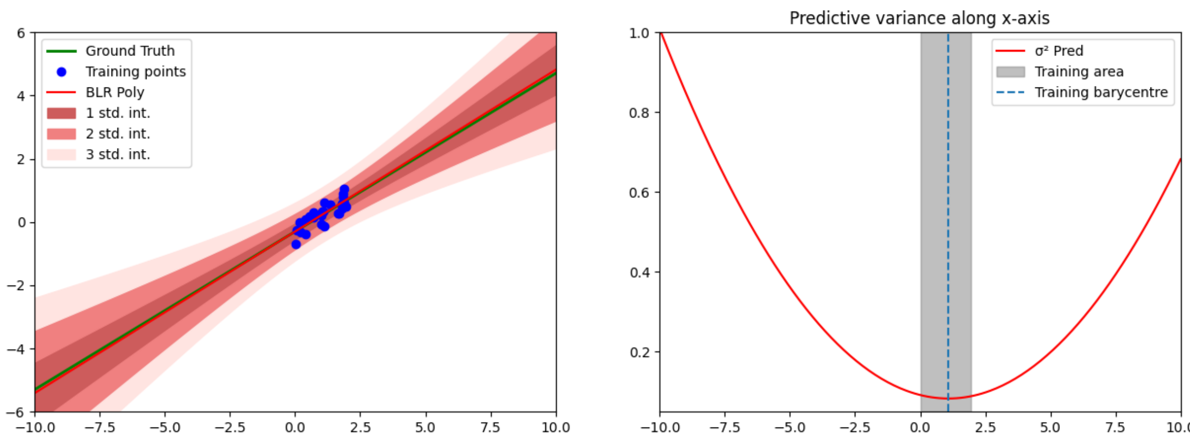


Figure 8.2 – Bayesian linear regression model

The blue points are real training points used to train the Bayesian linear regression model. The green line represents the real output of the test data set, which is used to evaluate the model's predictive performance. The red line represents the predicted mean from the Bayesian linear regression model. Ideally, the red and green lines would be very close. However, in real training, there will be uncertainties, such as noise and the expected value of the model. Therefore, the prediction line usually fluctuates, and an uncertainty interval appears, which is the red area in the figure, with different standard deviation intervals. Uncertainty increases with distance from the training data center.

The figure on the right is a prediction variance diagram, which represents the distribution of the prediction variance along the x-axis, and the uncertainty of the model prediction changes as x changes. The gray area represents the range of training data. In this range, the model has information learned directly from the data, so the predictions here are usually more accurate. The blue dotted line is the average of the positions of all training data points, which can also be understood as the center point of the training data.

We can observe that near the training center, the red curve is lower, which shows that the uncertainty of the model prediction is smaller, because there is more training data in this area to support the model prediction. As the input values deviate from the training center, the red curve rises, that is, the uncertainty of the prediction increases, because the model is less confident in predicting areas not covered by the training data.

→ **Question 1.5:** Analyse these results. Why predictive variance increases far from training distribution? Prove it analytically in the case where $\alpha = 0$ and $\beta = 1$.

$$\Sigma^{-1} = \alpha I + \beta \Phi^T \Phi = \alpha \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \beta \begin{pmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}$$

So we have:

$$\Sigma^{-1} = \begin{pmatrix} \alpha + \beta N & \beta \sum x_i \\ \beta \sum x_i & \alpha + \beta \sum x_i^2 \end{pmatrix}$$

When $\alpha = 0$ and $\beta = 1$, we have

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

So here,

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

And then,

$$\begin{aligned} \sigma^2_{pred}(x^*) &= \frac{1}{\beta} + \Phi(x^*)^T \Sigma \Phi(x^*) \\ &= \Phi(x^*)^T \Sigma \Phi(x^*) \\ &= \begin{pmatrix} 1 & x^* \end{pmatrix} \Sigma \begin{pmatrix} 1 \\ x^* \end{pmatrix} \\ &= \frac{1}{det(\Sigma^{-1})} (\begin{pmatrix} 1 & x^* \end{pmatrix} \begin{pmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix} \begin{pmatrix} 1 \\ x^* \end{pmatrix}) \\ &= \frac{1}{det(\Sigma^{-1})} (\sum x_i^2 - 2x^* \sum x_i + N x^{*2}) \\ &= \frac{1}{det(\Sigma^{-1})} \sum (x_i - x^*)^2 \end{aligned}$$

With

$$det(\Sigma^{-1}) = N \sum x_i^2 - (\sum x_i)^2$$

The prediction variance is related to x^* and also depends on the distribution of x_i. When the value of x^* deviates from the mean of the training data set, the prediction variance increases. In areas with less training data, prediction uncertainty is higher.

→ **Bonus Question :** What happens when applying Bayesian Linear Regression on the following dataset?

Training data points and real data generating function are as follows:

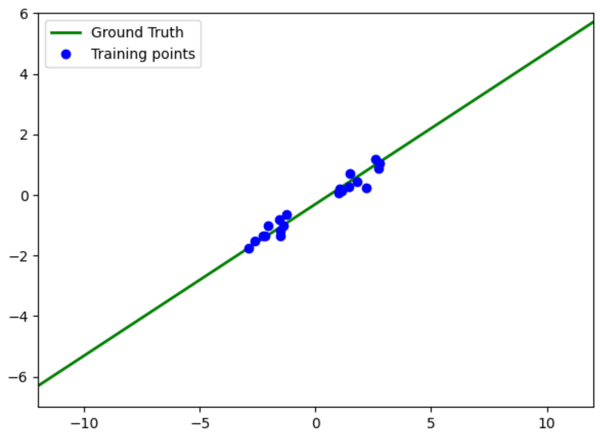


Figure 8.3 – Training data

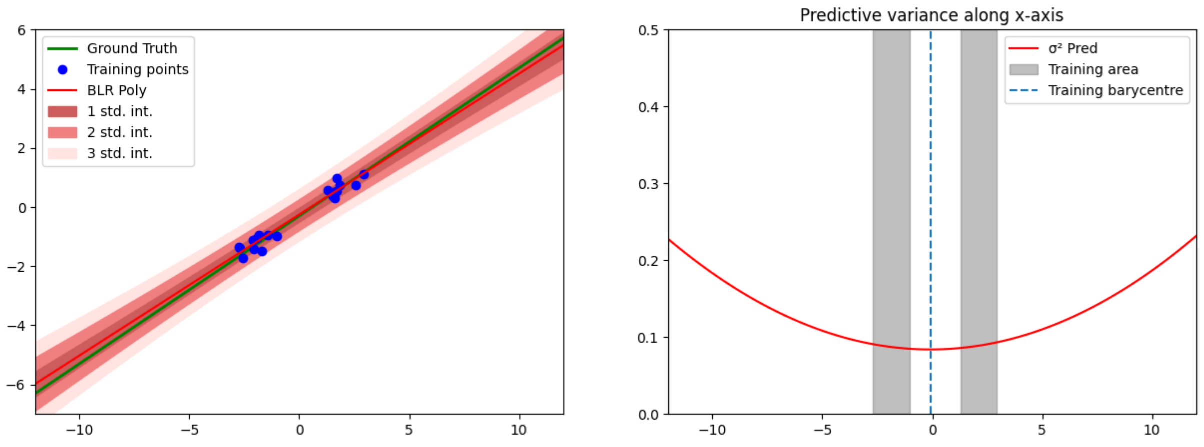


Figure 8.4 – Bayesian linear regression model

From the provided data set, we can see that what is different from the previous data set is that there is a gap in the middle of the data and there are insufficient training data points. Usually in this case, the prediction variance of the expected model will increase, but from the result plot, the prediction variance becomes smaller in the absence of data. This is because when a data point is located close between two clusters of training data points, this proximity leads to a lower squared difference, that is, increased uncertainty, and thus a lower prediction variance.

8.2 Non Linear models

8.2.1 Polynomial basis functions

We will first use polynomial basis functions:

$$\phi : x \to (\phi_0, \phi_1, ..., \phi_{D-1})$$

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

Design matrix Φ defined on training set is:

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{D-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^{D-1} \end{bmatrix}$$

$\rightarrow$ **Question 2.1:** Code polynomial basis function.

The code is in the Jupyter notebook file.

$\rightarrow$ **Question 2.2:** Code and visualize results on sinusoidal dataset using polynomial basis functions. What can you say about the predictive variance?

The data is as follow:

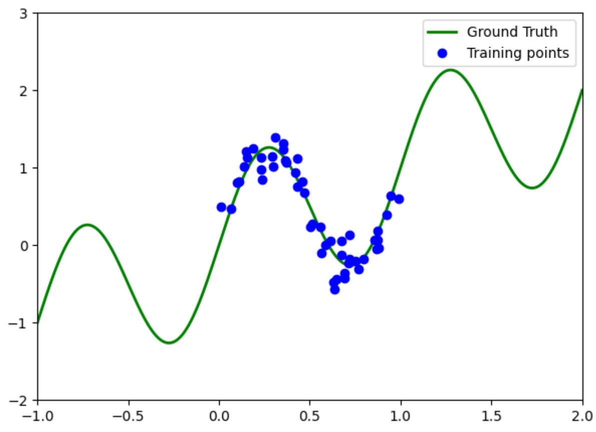


Figure 8.5 – Training dataset

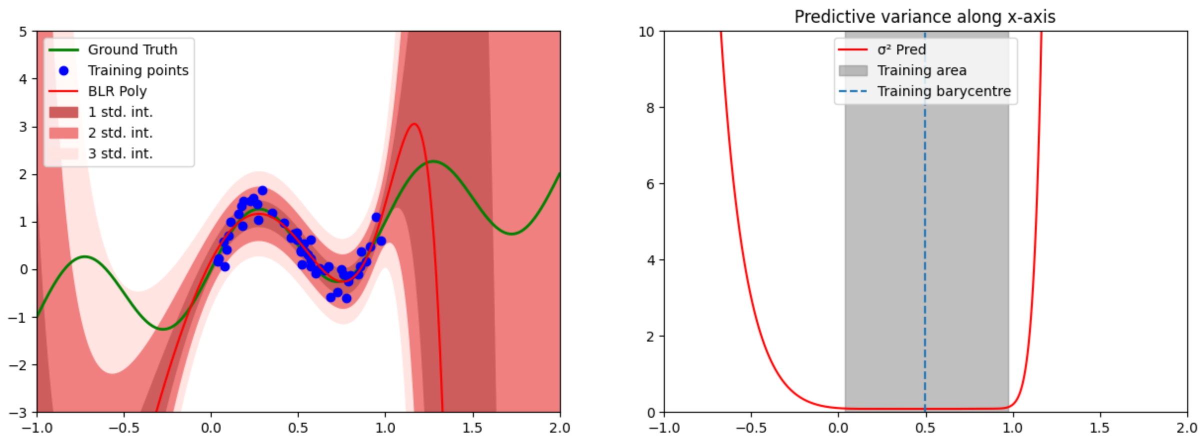


Figure 8.6 – Bayesian non-linear regression model

We can observe from the image that for the left image, in the data point area, the model's prediction is close to the true value, while in other areas the deviation is larger. At the same time, in areas with dense data points, the uncertainty is smaller, while in areas with sparse data points, the uncertainty is larger. In the right plot, the prediction variance increases significantly in areas where training data points are sparse or not covered.

The minimum variance is no longer solely at the centre of the data (we can see that the minimum is between 0 and 1 on the x-axis), and the farther away we move from the data, the more the variance increases, meaning the model is losing certainty when it comes to making predictions.

This Bayesian regression model is able to capture the pattern of sinusoidal data relatively accurately and provides a measure of uncertainty in the predictions. In areas with dense data, the prediction variance is very low, indicating that the model prediction is more accurate.

8.2.2 Gaussian basis functions

Now, let's consider gaussian basis functions:

$$\phi : x \to (\phi_0, \phi_1, ..., \phi_M)$$

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

Design matrix Φ defined on training set is:

$$\Phi = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & ... & \phi_M(x_1) \\ ... & ... & ... & ... \\ \phi_0(x_n) & \phi_1(x_n) & ... & \phi_M(x_n) \end{bmatrix}$$

$\rightarrow$ **Question 2.3:** Code gaussian basis function.

The code is in Jupyter notebook file.

$\rightarrow$ **Question 2.4:** Code and visualize results on sinusoidal dataset using Gaussian basis functions. What can you say this time about the predictive variance?

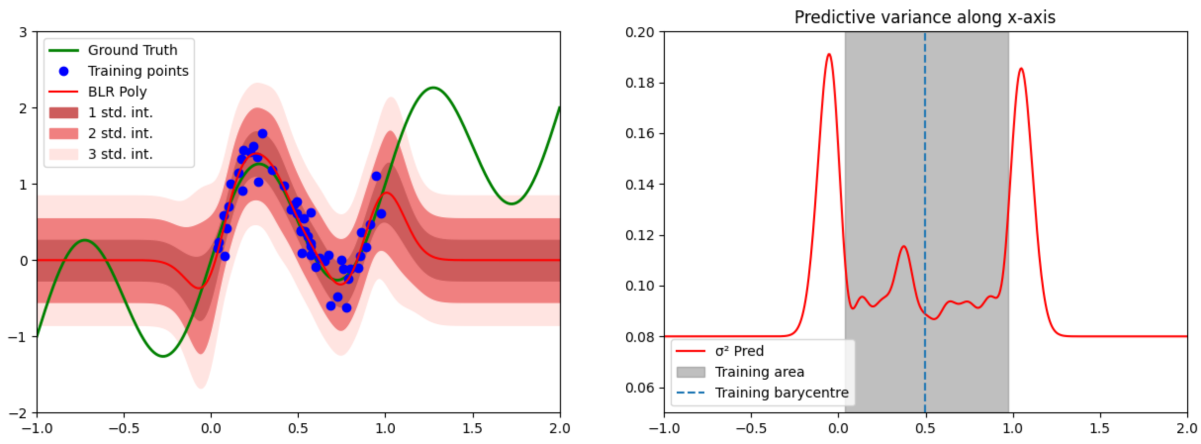


Figure 8.7 – Bayesian non-linear regression model

The code is in Jupyter notebook file.

We observe from the plot that the variance values are almost the same for the region where the data lies. The variance increases as we move away from the data region, but continues to decrease to a value close to 0.08 and levels off as we move further away from the data. But in general, the farther away the data is, the greater the variance.

$\rightarrow$ **Question 2.5:** Explain why in regions far from training distribution, the predictive variance converges to this value when using localized basis functions such as Gaussians.

When using localized basis functions such as Gaussian, the prediction variance converges to a specific value in regions far away from the training data distribution. That is to say, when x^* is far away from the training data, $\Phi(x^*)$ will be very small and tends to zero, so $\Phi(x^*)^T \Sigma \Phi(x^*)$ will also be very small and tends to zero. So the variance converges to $\frac{1}{\beta}$.

Chapter 9

Approximate Inference in Classification

9.1 Bayesian Logistic Regression

And the dataset is as follow:

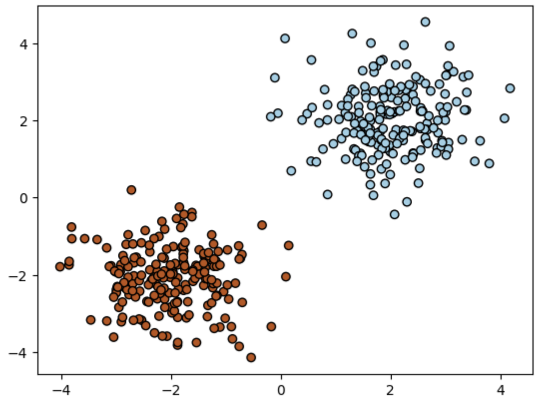


Figure 9.1 – Dataset

9.1.1 Maximum-A-Posteriori Estimate

→ **Question 1.1:** Analyze the results provided by previous plot. Looking at $p(\mathbf{y} = 1|\boldsymbol{x}, \boldsymbol{w}_{\text{MAP}})$, what can you say about points far from train distribution?

First, let's analyze the resulting figure. The white line is the decision boundary. It can be observed from the figure that the model completely separates the two datasets. The blue and red line areas represent the model's confidence in the corresponding classification. The darker the color, the higher the confidence in its classification. The logistic regression model outputs a probability value for each data point, which represents the model's confidence in classifying the point as a positive class based on the current weight. When this probability area is 1, the model is very confident in the prediction that the point belongs to the positive class, which is represented by dark red. On the contrary, when the probability value is close to 0, the model believes that the point belongs to another class, which is the dark blue part of the figure. In this model, we can see that the two types of data are perfectly separated, and the accuracy is 100%.

But it also highlights the problem that the model shows the same level of confidence far away from the decision boundary, without exhibiting any uncertainty. This means that the model does not handle uncertainty well, especially in regions far from the decision boundary.

Therefore, the model is good at classifying data into their respective categories, but is not suitable for estimating the uncertainty of data points that are outside the training data set.

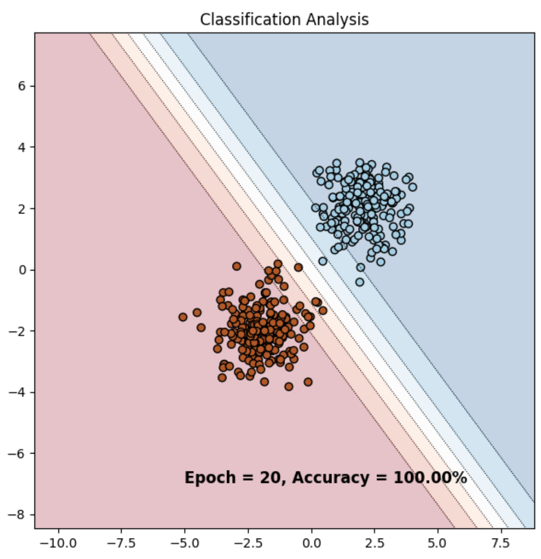


Figure 9.2 – Plot decision boundary

9.1.2 Laplace Approximation

→ **Question 1.2:** Analyze the results provided by previous plot. Compared to previous MAP estimate, how does the predictive distribution behave?

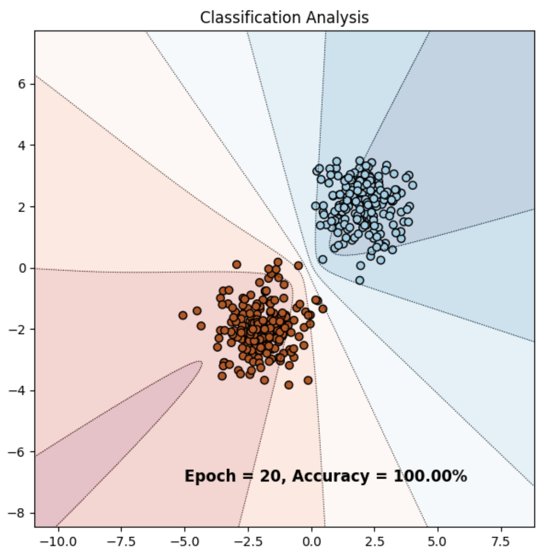


Figure 9.3 – Plot decision boundary

Compared with the previous MAP, the difference is that the Laplace approximation can reflect uncertainty. As shown in the figure, regional changes in contours represent uncertainty. In detail, contours represent changes in predicted probabilities, with lines becoming wider away from data-dense regions, indicating higher uncertainty. As can be seen from the figure, the accuracy reaches 100%, but through the distribution of contours, we observe that the model's prediction uncertainty is high, thus indicating that the model may perform poorly on unknown data.

$\rightarrow$ **Question 1.3:** Comment the effect of the regularisation hyper-parameter WEIGHT_DECAY.

WEIGHT_DECAY controls the complexity of the model and imposes penalties on large weights in the loss function, which helps the model maintain smaller weight values and prevent overfitting.

Higher WEIGHT_DECAY values mean tighter priors, which pull the weights towards zero. Making the prediction distribution more conservative results in less flexible decision boundaries and the model exhibiting higher uncertainty in regions further away from the training data. A WEIGHT_DECAY that is too high will increase forecast uncertainty and the shaded area will be wider, while a WEIGHT_DECAY that is too low may lead to overconfidence.

9.1.3 Variational Inference

$\rightarrow$ **Question 1.4:** Comment the code of the VariationalLogisticRegression and LinearVariational classes.

LinearVariational implements a fully connected layer.

First, the variational parameters are initialized, w_mu and b_mu are initialized to 0, w_rho and b_rho are initialized to 1, and the variational posterior of each parameter is a lognormal distribution centered at zero. The sampling method uses a reparameterization technique to sample weights, which allows us to backpropagate the probability distribution. The formula is as follows:

$$\sigma = \log 1 + e^{p}$$

We use the kl_divergence method to calculate the KL divergence between the variational posterior distribution and the prior distribution of the parameters as a regularization term. At the end, we create the forward method, which first samples the weights and biases, then performs matrix multiplication and addition of biases, and finally accumulates the calculated KL divergence into the accumulated_kl_div attribute of the parent module so that it can be used during the entire model training process. Optimize the KL divergence.

VariationalLogisticRegression defines a logistic regression model using variational stratification.

Use LinearVariational as its fully connected layer, obtain the accumulated KL divergence through accumulated_kl_div, and start accumulating KL divergence again at each epoch. The forward propagation function calls the variable layer and applies the sigmoid function to get the output.

→ **Question 1.5:** Comment the code of the training loop, especially the loss computation. Analyze the results provided by previous plot. Compared to previous MAP estimate, how does the predictive distribution behave? What is the main difference between the Variational approximation and the Laplace approximation?

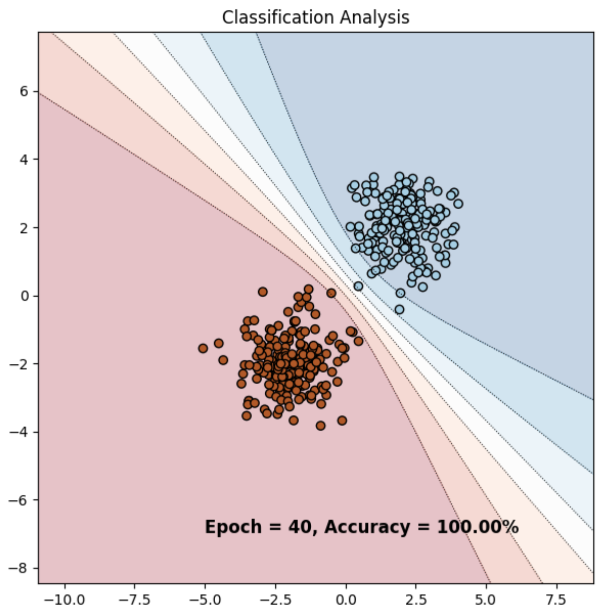


Figure 9.4 – Plot decision boundary

At the beginning of each training epoch, we reset the optimizer's gradient and accumulated KL divergence. Then propagate forward and use elbo to calculate the loss. We need to maximize ELBO, which consists of the negative log-likelihood of the data and the KL divergence, where the negative log-likelihood measures how well the model fits the data, and the KL divergence serves as the regularization term. Then perform backward propagation, calculate the gradient, and update the model.

First we analyze the resulting image. After 40 times of training, the accuracy rate is 99.75%, which is lower than the previous method. The decision boundary clearly separates the two types of data, showing the model's good fit to the data and taking into account the uncertainty in the prediction. All training points fall within the high confidence region.

Variational inference provides predictive distributions that exhibit more uncertainty than the previous maximum a posteriori estimate (MAP).

The Laplace approximation approximates the posterior distribution by fitting a Gaussian distribution around the maximum posterior estimate point. It cannot fully capture the complex characteristics of the posterior distribution and relies more on local information. It mainly focuses on finding the optimal point (MAP) and calculating the Hessian matrix (Hessian) of the point. Variational inference approximates the posterior distribution by optimizing the full parameter space of the distribution, rather than a single value, and is quantified by KL divergence.

9.2 Bayesian Neural Networks

The dataset is as follow:

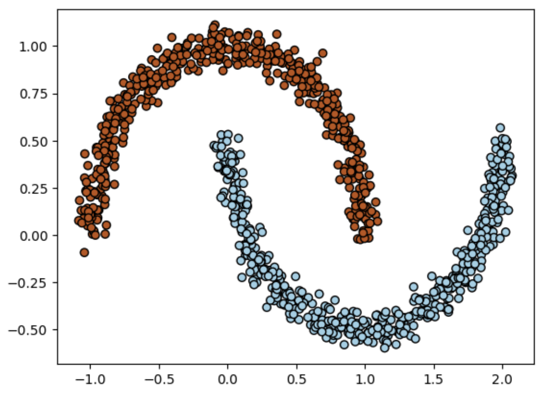


Figure 9.5 – Dataset

9.2.1 Variational Inference with Bayesian Neural Networks

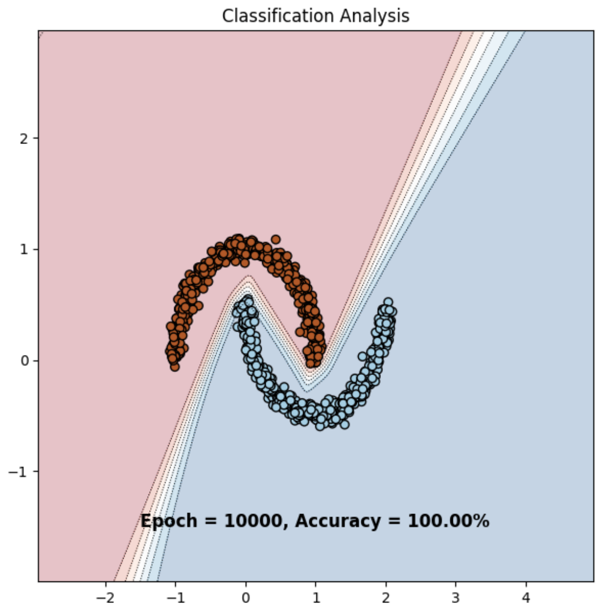


Figure 9.6 – Plot decision boundary

After 10,000 epochs, the accuracy reached 99.90%. We can see that the data is non-linear and the decision boundaries clearly separate the data. We know that the gradient shading near the decision boundary represents the uncertainty of the forecast. At the edge of the data, the shaded area is wider, indicating that the confidence level is lower and difficult to determine.

9.2.2 Monte Carlo Dropout

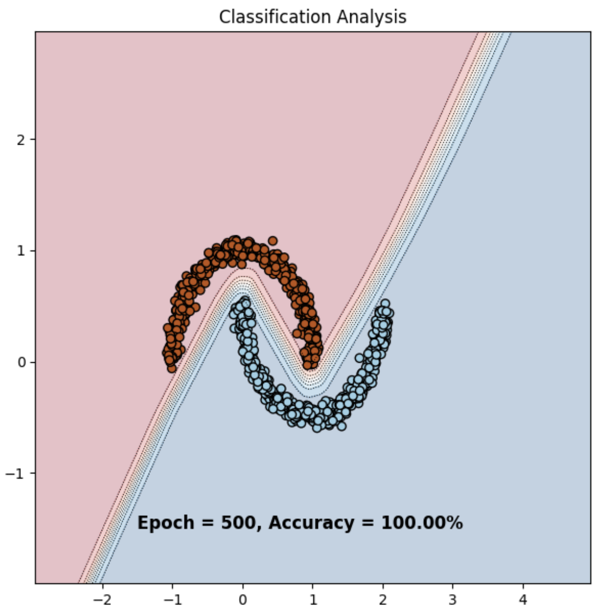


Figure 9.7 – Plot decision boundary

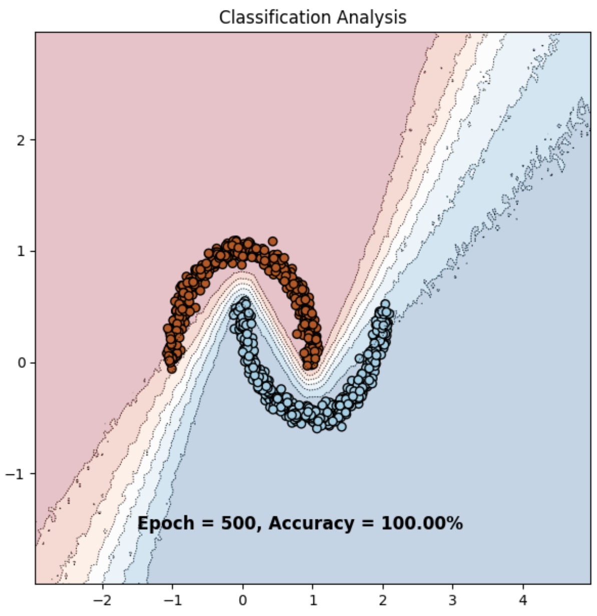


Figure 9.8 – Plot decision boundary

→ **Question 2.1:** Again, analyze the results showed on plot. What is the benefit of MC Dropout variational inference over Bayesian Logistic Regression with variational inference?

In the first image, without MC Dropout, the model achieved 100% accuracy after 500 epochs. It becomes evident that our model if fully capable of distinguishing between the two classes. The variance steadily grows as we move farther from our data, serving as a representation of our epistemic uncertainty.

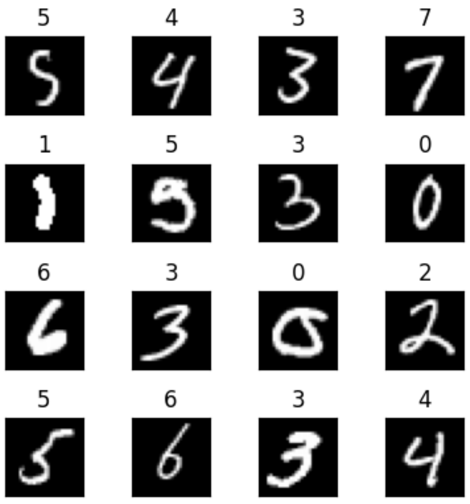
In the second image, with MC Dropout, the accuracy is still 100%, but the decision boundary contains more uncertainty areas.

In MC Dropout, dropout is a variational distribution used as network weights, randomly discarding some neurons in each forward pass, thus simulating the uncertainty of the weights. For a new input, by executing the forward pass with dropout multiple times and calculating the average output, the predicted distribution of the output corresponding to the input can be approximately obtained. MC Dropout can show how the model's confidence changes with the input space. Not only does it allow the network to learn complex decision boundaries, it also provides uncertainty estimates that ordinary neural networks often lack, so it can better capture the uncertainty of predictions.

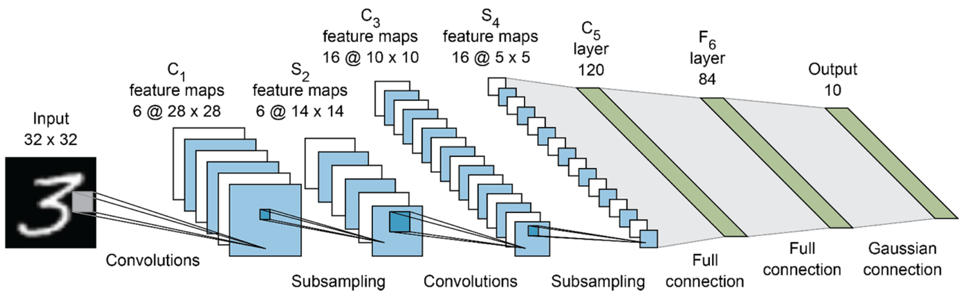
Chapter 10

Uncertainty Applications

10.1 Monte-Carlo Dropout on MNIST



10.1.1 LeNet-5 network with dropout layers



All the implentations are in the colab file.

10.1.2 Investigating most uncertain samples

→ **Question 1.1:** What can you say about the images themselfs. How do the histograms along them helps to explain failure cases? Finally, how do probabilities distribution of random images compare to the previous top uncertain images?

Use the lenet model to predict the test set test_loader. Predictive uncertainty was calculated using the variation ratios (var-ratios) method.

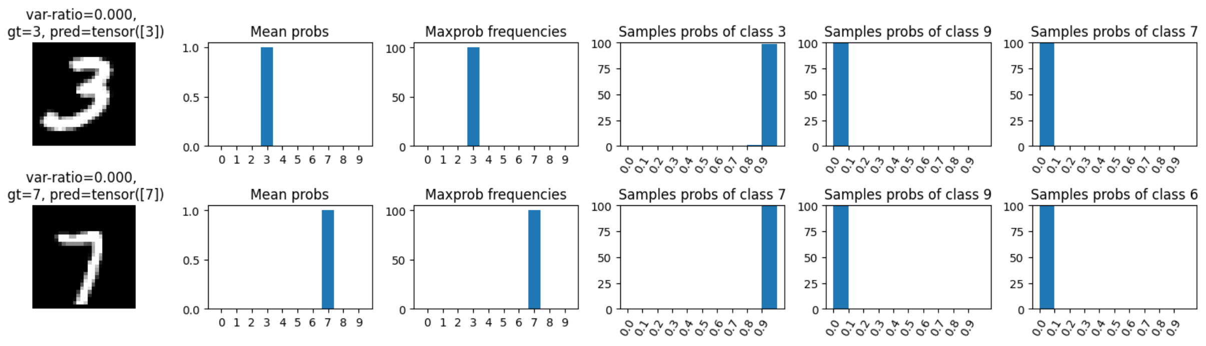
The function returns multiple results:

— pred_var: Predicted category for each sample on the test set.
— labels: The true label of each sample on the test set.
— uncertainty_var: Uncertainty estimate of the variation ratio corresponding to each sample.
— errors_var: A 0/1 vector indicating whether the model prediction is wrong (1 means wrong, 0 means correct).
— hists: The class histogram for each sample on the test set, showing the number of times each class occurs in all random forward passes.
— mc_samples: Predicted probabilities for all forward propagations, these can be used to further analyze the uncertainty of the predictions.

And then, we draw the predicted image by calling the plot_predicted_images function for visual observation.

- Plotting random images with their var-ratios value

Randomly select the indexes of two samples from the test set, predict them, and visualize the results. The results are as follows:



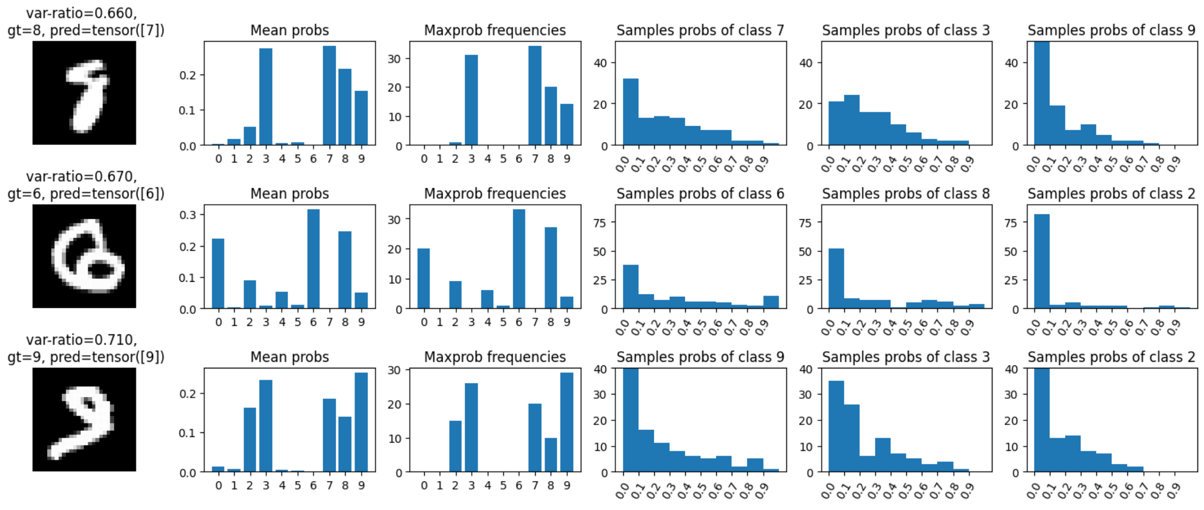
On the left of the image are images of handwritten numbers "3" and "7". The model's prediction result is the same as the real label, and the mutation ratio of both is 0, which means the model is very certain about this prediction result and there is no uncertainty.

In the middle of the image, Mean probs plot shows the average probability of the model predicting each category. For the first number 3 and the second number 7, the corresponding category bar height is very high, and it is zero for other categories.

The Maxprob frequencies plot shows the frequency of the most commonly predicted class across all MC Dropout samples, and this means that the model tends to choose this category.

Finally, on the right side of the image is the Samples probs of class X plot, which shows the probability distribution of a specific class. We can see that for the number 3, when the specific category is 3, there is a very large probability, and when the specific categories are 9 and 7, the probability is 0.

- Visualize the top-3 most uncertain images along with their var-ratios value



We analyze the results line by line.

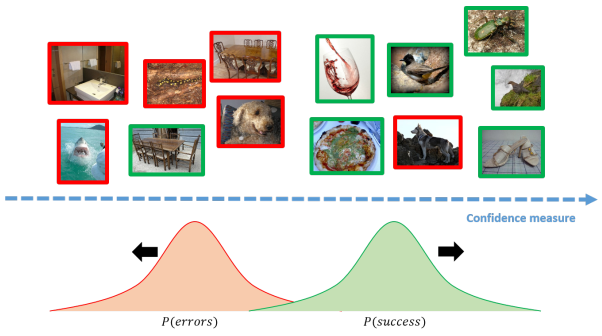
For the first line, the image shows a handwritten "8", but the model predicts a "7". The mutation ratio is 0.660, and the model is quite uncertain about this prediction.

In the Mean probs plot, we can see that the probabilities of several categories are relatively average, such as 3, 7, 8 and 9, so the uncertainty of the results is high. Maxprob frequencies plot can already be seen that there are multiple choices for categories during the training process.

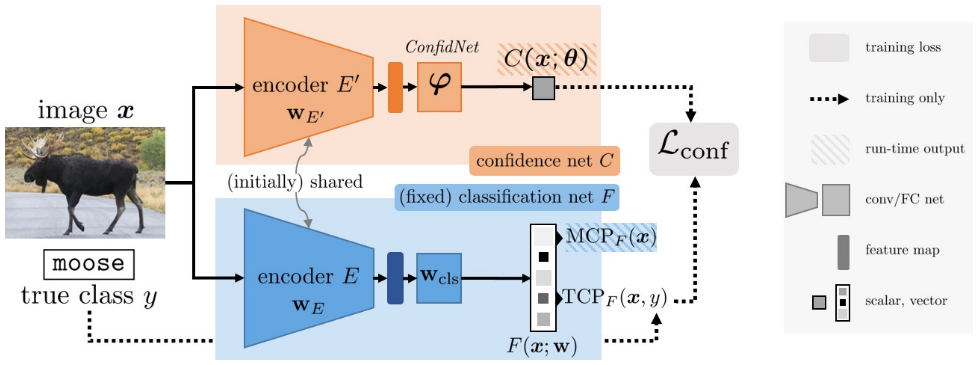
Performing the same analysis on the second sample number and the third sample number, we can obtain similar analysis results to the first sample number.

Images with high uncertainty have more dispersed probability distributions, indicating that the model gives multiple predictions in different forward propagations with high uncertainty. However, uniquely randomly selected images will show a more concentrated probability distribution, the model will be more consistent in multiple forward propagations, and it will be more confident in its prediction results.

10.2 Failure prediction

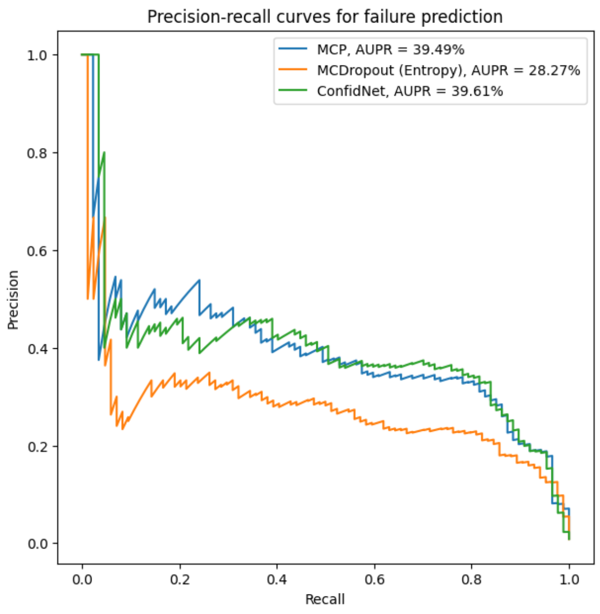


10.2.1 ConfidNet



10.2.2 Evaluate failure prediction performances

→ **Question 2.1:** Compare the precision-recall curves of each method along with their AUPR values. Why did we use AUPR metric instead of standard AUROC?



The image shows the precision-recall curves of three uncertainty estimation methods: ConfidNet, MCP and MCDropout. And the corresponding average precision-recall value, which is AUPR. Detailed analysis is as follows:

For the precision-recall curve:

— Precision is the ratio between the number of samples correctly predicted as failure cases by the model and the total number of samples predicted by the model as positive classes.

$$Precision = \frac{TP}{TP + FP}$$

— Recall is the ratio of the number of samples predicted as failure cases to the total number of samples that are actually positive classes ratio between the total number of samples.

$$Recall = \frac{TP}{TP + FN}$$

Therefore, the closer the curve is to the upper right corner of the graph, the higher the accuracy and recall rate, which means that the model has better performance in predicting failure.

For AUPR, it is a measure of the area under the precision-recall curve, which reflects the overall performance of the model's prediction failure.

— ConfidNet (blue curve): has the highest AUPR, close to 39.61%, indicating that it has the best performance in predicting model failure.

— MCP (green curve): AUPR is close to 39.49%, which is similar to the ConfidNet result.

— MCDropout (orange curve): AUPR is 28.27%, the effect is worse than the first two methods.

And AUROC is the area under the ROC curve. So for the ROC curve:

— True Positive Rate is the same as Recall.

$$TPR = Recall = \frac{TP}{TP + FN}$$

— False Positive Rate is the ratio between the number of samples incorrectly labeled as failure cases and the total number of samples that are actually negative cases.

$$FPR = \frac{FP}{FP + TN}$$

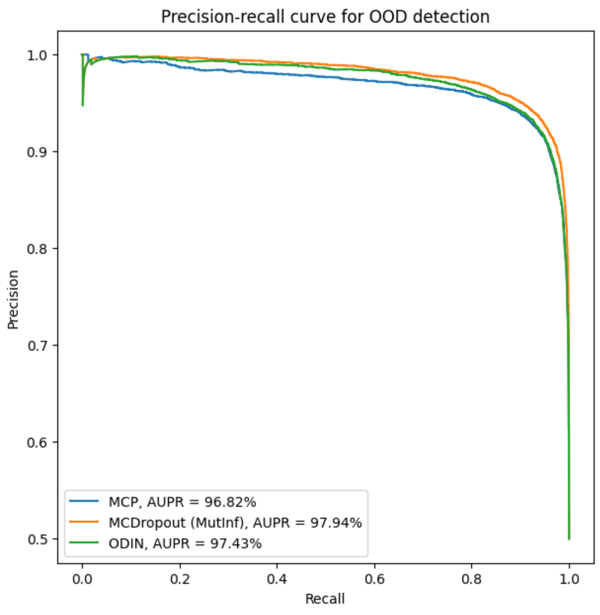
So the ROC curve curves toward the upper left corner, which means there is high TPR and low FPR, indicating that the model can correctly identify most positive samples while maintaining a low false positive rate.

Our model is good at making accurate predictions, so there are fewer positive classes than negative classes, that is, there are usually fewer failed cases than successful ones, which results in class imbalance. In this case, AUPR is a better performance metric because AUPR focuses on the prediction performance of the positive minority class. AUROC will produce misleading results in the case of class imbalance.

AUPR is more inclined to focus on the performance of the model in the positive class, because both Precision and Recall calculate the proportion of the positive class. In AUROC, FPR calculates the ratio of FP.

10.3 Out-of-distribution detection

→ **Question 3.1:** Compare the precision-recall curves of each OOD method along with their AUPR values. Which method perform best and why?



A concept is introduced here called OOD, which is Out-of-distribution detection. By combining the uncertainty of MNIST predictions with the uncertainty of KMNIST predictions, KMNIST samples are considered to be OOD samples, and these data are used to evaluate the model's performance on OOD detection. On this basis, the odin_preprocessing equation is used for preprocessing. Adding small perturbations to the input of the model makes it easier for the model to distinguish between in-distribution samples and out-of-distribution samples.

We evaluate by plotting the precision-recall curves of the three methods and calculating the AUPR value. The three methods are: MCP, MCDrpoout and ODIN preprocessing.

The MCDropout method has the highest AUPR value, and the AUPR value of ODIN is similar, but the MCP is slightly lower. All three methods have good results.

The closer the curve is to the upper right corner of the graph, the higher the precision and recall, which means that the model performs better in OOD detection. Therefore, the MC-Dropout method has the best results.