M2 M2A

Report for RDFIA Homework

Section 3
Bayesian deep learning

Students :
Kai MA, Ruyi TANG, Jinyi WU

January 27, 2025

Contents

3-a: Bayesian Linear Regression

Bayesian linear regression provides a probabilistic approach to modeling the relationship between inputs and outputs by incorporating prior beliefs about the parameters. Unlike traditional linear regression, which estimates point values for the model parameters by minimizing a loss function, it treats parameters as random variables with prior distributions. Given observed data, the posterior distribution of the parameters is computed using Bayes' theorem.

This approach offers several advantages, such as capturing uncertainty in predictions and enabling the incorporation of prior knowledge. The posterior distribution can be used to derive credible intervals for predictions, making Bayesian linear regression particularly useful in scenarios where quantifying uncertainty is important.

In this chapter,

1 Probabilistic Setting

In this section, we present the probabilistic framework of Bayesian Linear Regression, focusing on the mathematical formulation and its advantages in capturing uncertainty.

Similar to standard regression, we typically define the probabilistic setting as follows:

Given an input matrix $X \in \mathbb{R}^{N \times d}$, the observed output Y is assumed to follow a linear model with additive Gaussian noise:

$$Y = \Phi w + \varepsilon$$

with:

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_M(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_M(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \dots & \phi_M(x_N) \end{bmatrix} = \begin{bmatrix} \phi(x_1)^T \\ \phi(x_2)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix}$$

where:

- $\Phi \in \mathbb{R}^{N \times M}$ is the design matrix, which transforms input features into a higher-dimensional space to capture more complex relationships.
- $w \in \mathbb{R}^{M \times K}$ represents all the model parameters.
- $\varepsilon \in \mathbb{R}^{N \times K}$ is Gaussian noise, assumed to follow $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$.

Thus, the likelihood function for the observed data point (x, y) follows a normal distribution:

$$p(y|x, w, \beta) \sim \mathcal{N}(y; \phi(x)^T w, \beta^{-1})$$

where $\beta = \frac{1}{\sigma^2}$ represents the precision of the noise in the data, which corresponds to **aleatoric uncertainty** reflecting the inherent randomness in the observations.

In the Bayesian framework, instead of estimating a single optimal w, we place a prior distribution over w to capture uncertainty in the model. Typically, we assume a Gaussian prior:

$$p(w|\alpha) \sim \mathcal{N}(w; 0, \alpha^{-1} I)$$

where α is the precision of the prior distribution,

By applying Bayes' theorem, we can obtain the **posterior distribution over** w:

$$p(w|X, Y, \alpha, \beta) \sim \mathcal{N}(w|\mu, \Sigma) \quad \begin{cases} \Sigma^{-1} = \alpha I + \beta \Phi^T \Phi \\ \mu = \beta \Sigma \Phi^T Y \end{cases} \tag{1}$$

The posterior covariance matrix Σ captures the **epistemic uncertainty** which arises from limited data and reflects uncertainty in the learned parameters w.

To predict for a new input x^*, we integrate over the posterior distribution of w:

$$p(y|x^*, X, Y, \alpha, \beta) = \int p(y|x^*, w, \beta) p(w|X, Y, \alpha, \beta) dw$$

Since both the likelihood and the posterior are Gaussian, their convolution should also be Gaussian. Thus, the **predictive distribution** is written as:

$$p(y|x^*, X, Y, \alpha, \beta) \sim \mathcal{N}(y; \mu^T \phi(x^*), \beta^{-1} + \phi(x^*)^T \Sigma \phi(x^*)) \tag{2}$$

Notably, The first term of the predictive variance β^{-1} represents the aleatoric uncertainty due to inherent noise in the observations, while the second term $\Phi(x^*)^T \Sigma \Phi(x^*)$ represents the epistemic uncertainty due to uncertainty in the learned parameters.

2 Linear basis function model

We start with the simplest case of Bayesian Linear Regression in a 1D setting using linear basis functions $\phi(x) = (1, x)$ and the design matrix Φ is defined as:

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

We set $\alpha = 1$, i.e. we assume that the prior distribution of w: $p(w|\alpha) \sim \mathcal{N}(w; 0, I)$. The data we generate follows a linear relationship and is distributed over a continuous interval with a standard deviation of 0.2.

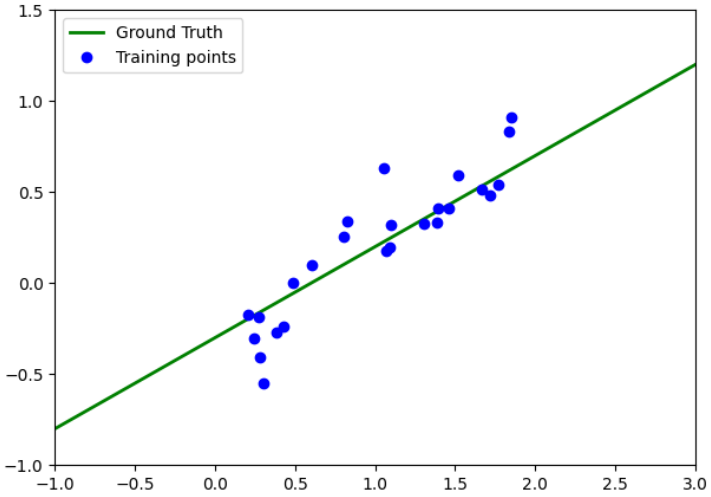


Figure 1: Training Data Visualization - Linear Dataset

Q1.2 Recall closed form of the posterior distribution in linear case. Then, code and visualize posterior sampling. What can you observe?

Using the closed-form solution of the posterior distribution of w from Equation (1):

$$p(w|X, Y, \alpha, \beta) \sim \mathcal{N}(w|\mu, \Sigma) \quad \begin{cases} \Sigma^{-1} = \alpha I + \beta \Phi^T \Phi \\ \mu = \beta \Sigma \Phi^T Y \end{cases}$$

we can now perform posterior sampling.

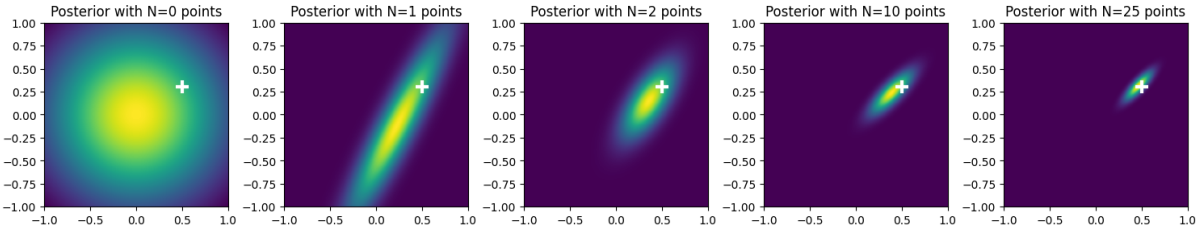


Figure 2: Evolution of Posterior Distribution with Increasing Data Points

From this result of posterior sampling, we observe that, initially, with no data, the posterior is broad and uncertain, covering a wide range of values. As data points are added, the posterior becomes more concentrated, aligning with the true parameter values and reducing uncertainty. We conclude that With a higher number of observations, the model's confidence improves, and the variance in parameter estimates shrinks, reflecting a more precise understanding of the underlying relationship.

Q1.3 Recall and code closed form of the predictive distribution in linear case. The closed form of the predictive distribution can also be calculated by using Equation (2):

$$p(y|x^*, X, Y, \alpha, \beta) \sim \mathcal{N}(y; \mu^T \phi(x^*), \beta^{-1} + \phi(x^*)^T \Sigma \phi(x^*))$$

The visualization of this predictive distribution is shown below:

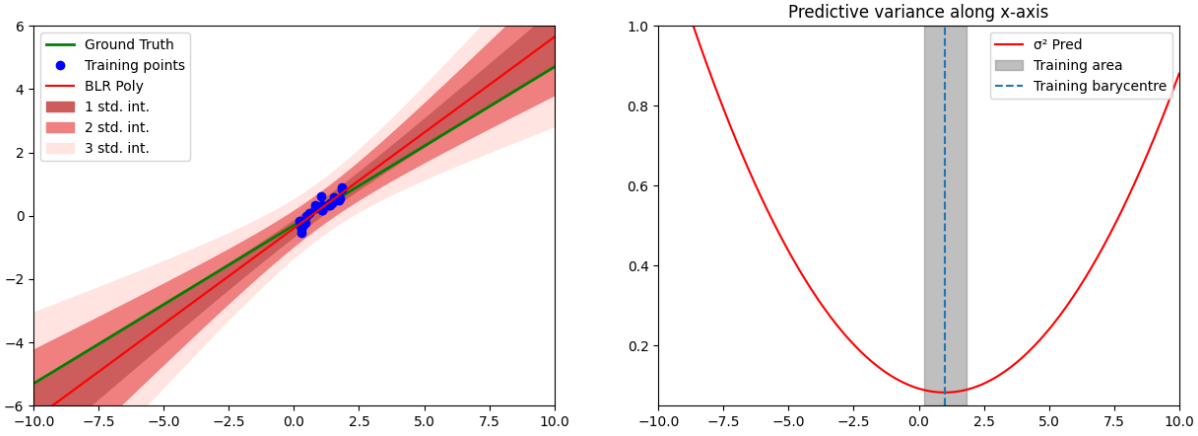


Figure 3: Visualization of predictive distribution in the linear basis function model ($\alpha = 1, \beta \approx 12.5$)

Q1.5 Analyze these results. Why predictive variance increases far from training distribution? Prove it analytically in the case where $\alpha = 0$ and $\beta = 1$.

The results reflect the expected influence of training data on model uncertainty. The left plot shows that the model closely fits the ground truth within the training data range, with low uncertainty, while uncertainty increases significantly as the distance from the training data grows. The right plot quantifies this effect, where the predictive variance is minimized near the training data and rises sharply in regions further away. This indicates that the model exhibits higher confidence in well-sampled areas, while in regions with sparse or no data, predictive uncertainty increases.

This phenomenon is more pronounced when $\alpha = 0$ and $\beta = 1$. In this case, the prior distribution of w approximates a uniform distribution, which is essentially equivalent to having no prior.

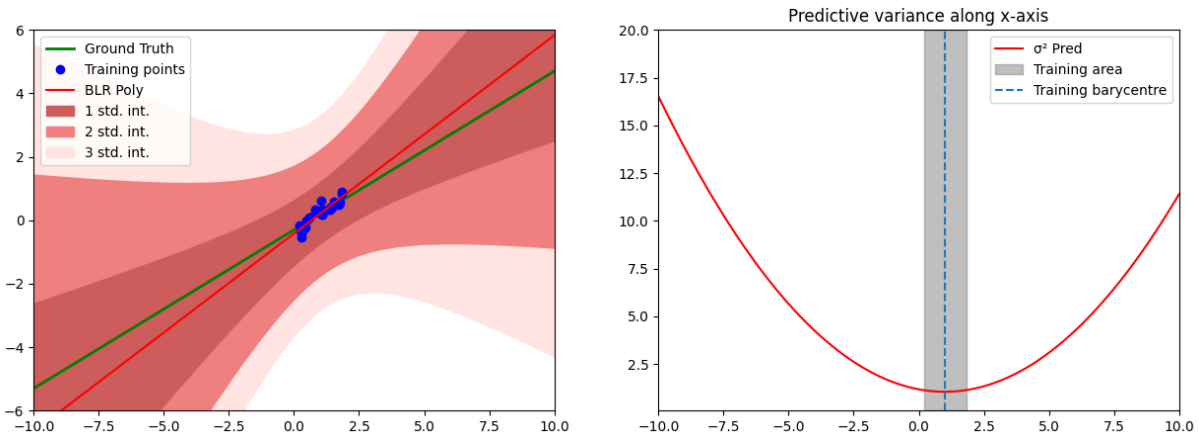


Figure 4: Visualization of predictive distribution in the linear basis function model ($\alpha = 0, \beta = 1$)

Analytically, we calculate $\sigma^2_{\text{pred}}(x^*) = \beta^{-1} + \phi(x^*)^T \Sigma \phi(x^*)$ to see what happens:

$$\Sigma^{-1} = \alpha I + \beta \Phi^T \Phi = \Phi^T \Phi = \begin{bmatrix} 1 & \dots & 1 \\ x_1 & \dots & x_N \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ \dots & \dots \\ 1 & x_N \end{bmatrix} = \begin{bmatrix} N & \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 \end{bmatrix}$$

By simply inverting Σ^{-1}, we have:

$$\Sigma = \frac{1}{|\Sigma^{-1}|} \begin{bmatrix} \sum_{i=1}^N x_i^2 & -\sum_{i=1}^N x_i \\ -\sum_{i=1}^N x_i & N \end{bmatrix} = |\Sigma| \begin{bmatrix} \sum_{i=1}^N x_i^2 & -\sum_{i=1}^N x_i \\ -\sum_{i=1}^N x_i & N \end{bmatrix}$$

Thus, the term $\phi(x^*)^T \Sigma \phi(x^*)$ can be developed as:

$$\begin{aligned} \phi(x^*)^T \Sigma \phi(x^*) &= |\Sigma| \begin{bmatrix} 1 & x^* \end{bmatrix} \begin{bmatrix} \sum_{i=1}^N x_i^2 & -\sum_{i=1}^N x_i \\ -\sum_{i=1}^N x_i & N \end{bmatrix} \begin{bmatrix} 1 \\ x^* \end{bmatrix} \\ &= |\Sigma| (\sum_{i=1}^N x_i^2 - 2x^* \sum_{i=1}^N x_i + N x^{*2}) \\ &= |\Sigma| \sum_{i=1}^N (x_i^2 - 2x^* x_i + x^{*2}) \\ &= |\Sigma| \sum_{i=1}^N |x^* - x_i|^2 \end{aligned}$$

The expression of $\sigma^2_{\text{pred}}(x^*)$ is now written as:

$$\sigma^2_{\text{pred}}(x^*) = 1 + |\Sigma| \sum_{i=1}^N |x^* - x_i|^2$$

This formula indicates that in our specific case, the predictive variance depends on the squared Euclidean distance between each training data point x_i and the prediction point x^*. As x^* moves further from the region with concentrated training data, these squared Euclidean distances increase, leading to a corresponding rise in predictive variance.

More interestingly, we can apply Bayesian Linear Regression to the following dataset, where the training data is separated into two regions:

Q* [*Bonus*] **What happens when applying Bayesian Linear Regression on the following dataset?**

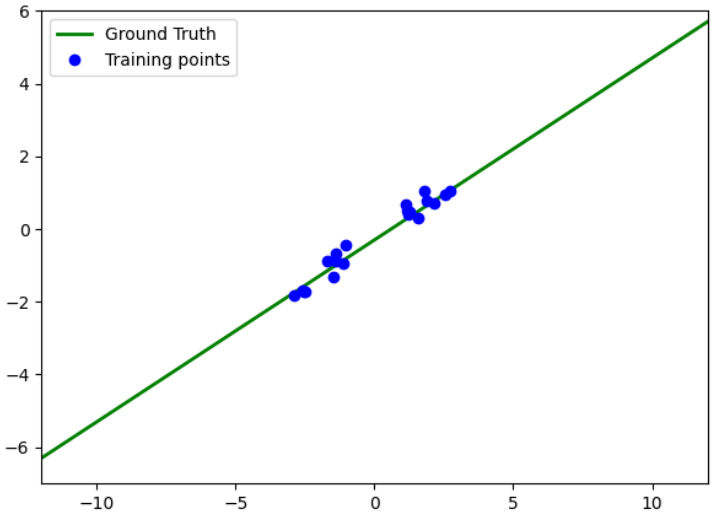


Figure 5: Training Data Visualization - Separate Regions

Our focus now is on analyzing the behavior of the predictive variance in the gap between the two separate regions of the training data. We aim to observe how uncertainty evolves in this region where no data is available and how Bayesian Linear Regression responds to such distribution patterns.

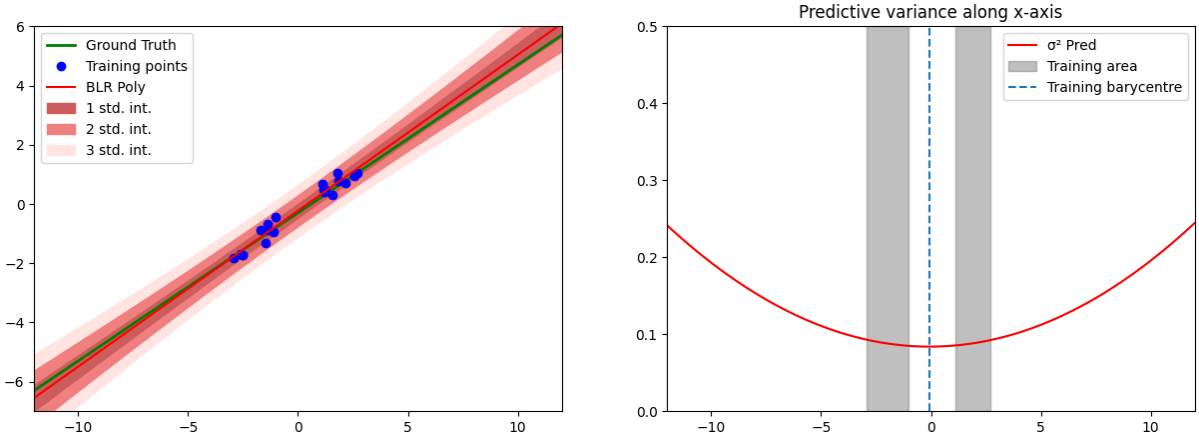


Figure 6: Visualization of predictive distribution in the linear basis function model (Training data in two separate regions)

In the right plot, the predictive variance is observed to decrease and attain the minimum in the central gap between the two data clusters, rather than increasing as might be expected in an area with no data. Analytically, this is because the sum of the squared distances from this point to all training samples is minimized at the barycenter of the training data which is not in the training area.

Additionally, this also relates to the fact that our basis function is linear, as we have assumed a linear relationship between the input points and their corresponding labels. In a linear model, two points uniquely determine a straight line, which allows the model to confidently interpolate between the clusters. As a result, the

predictive variance is minimized in the gap region because the model assumes that the underlying relationship remains linear, leading to a lower perceived uncertainty in that region. However in the non linear case, we can still observe a increasing predictive variance in the gap area.

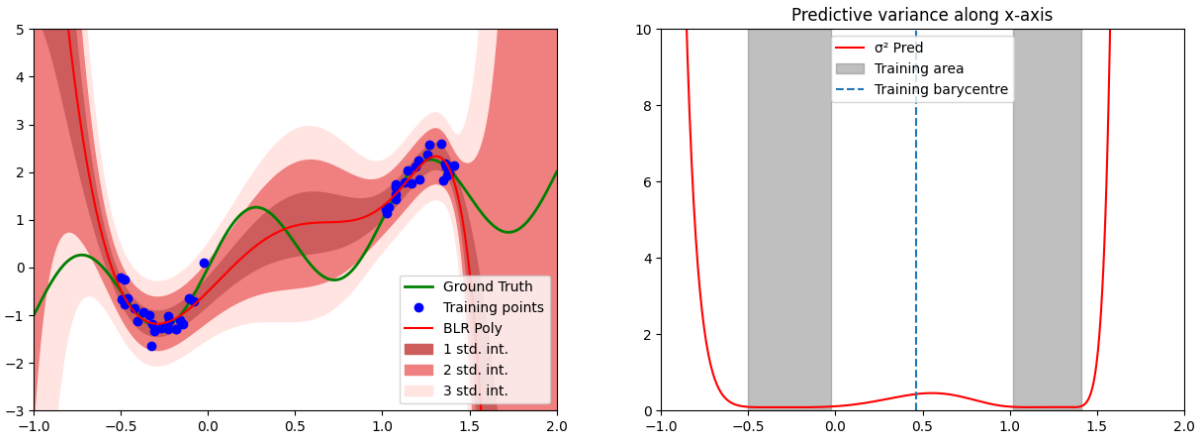


Figure 7: Visualization of predictive distribution in the polynomial basis function model (Training data in two separate regions)

Moreover, the left plot appears more compact compared to the previous ones because $\frac{1}{\beta}$ has decreased. A lower value of $\frac{1}{\beta}$ indicates reduced observation noise, leading to tighter confidence intervals around the predicted mean. This implies that the model assigns greater confidence to its predictions, resulting in a narrower spread of uncertainty.

3 Non linear model

Linear models assume a simple relationship between inputs and outputs, which may not hold in real-world scenarios. Non-linear models address this limitation by using non-linear basis functions, such as polynomial and radial basis functions (RBFs), to capture complex patterns in the data. In this section, we extend Bayesian methods to non-linear settings, enabling more flexible modeling while preserving uncertainty quantification.

In the upcoming experiments, we will train on this sinusoidal dataset below to explore how non-linear models adapt to complex data patterns:

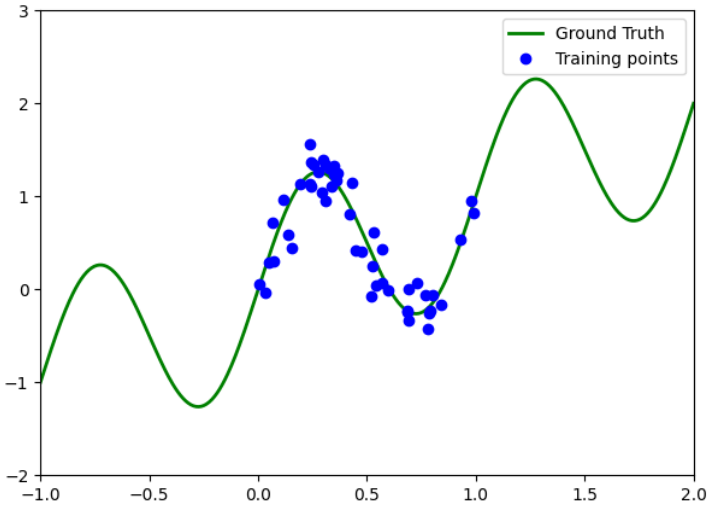


Figure 8: Training Data Visualization - Sinusoidal Dataset

3.1 Polynomial Basis Function

We now consider a polynomial basis function:

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

where $\phi_j = x^j$ for $j \geq 0$ and $D = 10$ in our case.

The design matrix Φ is defined as:

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

Q2.2 Code and visualize results on sinusoidal dataset using polynomial basis functions. What can you say about the predictive variance?

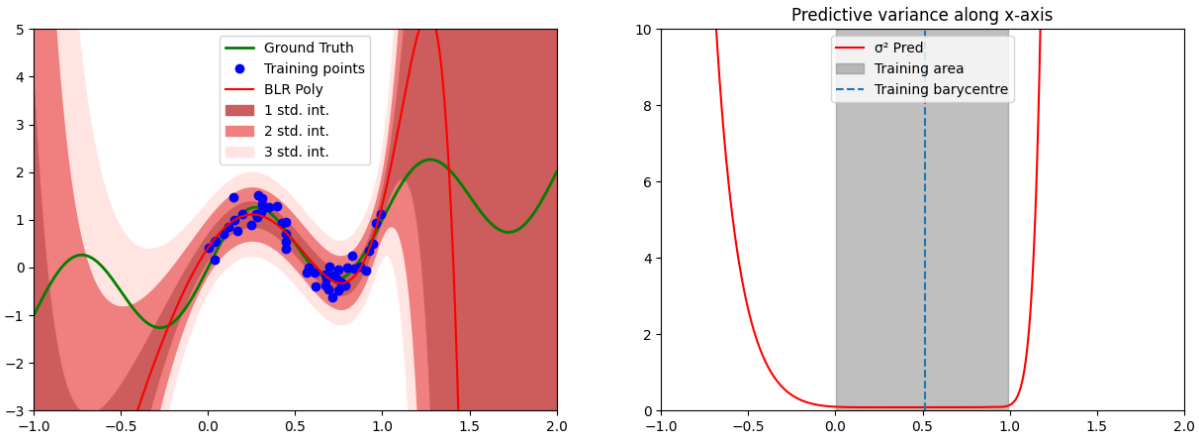


Figure 9: Visualization of predictive distribution in the polynomial basis function model

The figure shows that the polynomial basis function model fits well within the training data range but struggles outside this region. In the left plot, the model's

predictions closely follow the ground truth within the training area; however, outside this region, the predictions deviate significantly, indicating poor generalization. Additionally, the prediction intervals expand drastically at the boundaries, reflecting a high level of uncertainty. The right plot further confirms this behavior, where the predictive variance is low within the training region but increases sharply beyond it, emphasizing both the model's growing uncertainty and its decreasing accuracy in unseen regions.

3.2 Gaussian Basis Function

In the same way, we can apply a Gaussian basis function (aka. radial basis functions). We define:

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

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

In our case $M = 9$ and the values of μ_j are uniformly spaced between 0 and 1, which correspond to the boundaries of the training data.

The design matrix Φ is defined as:

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

Q2.4 Code and visualize results on sinusoidal dataset using Gaussian basis functions. What can you say this time about the predictive variance?

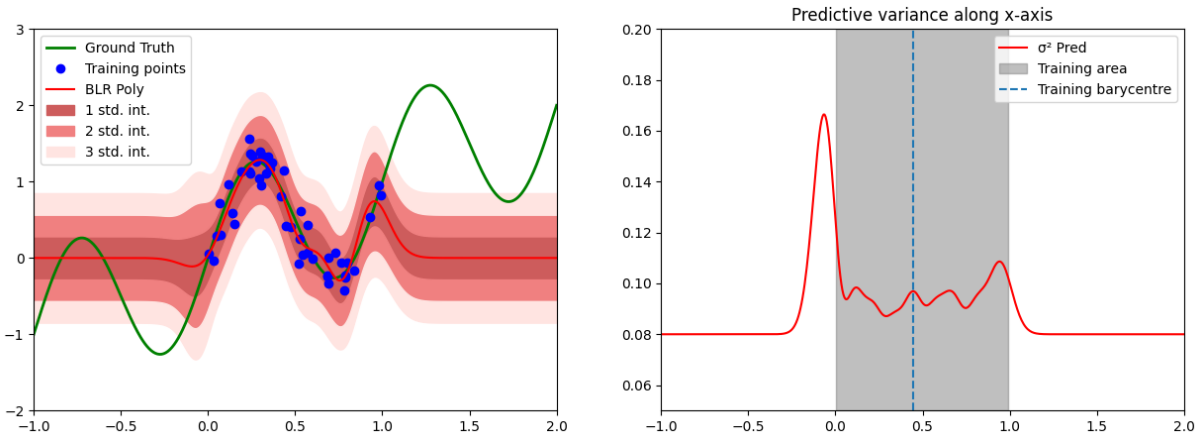


Figure 10: Visualization of predictive distribution in the Gaussian basis function model

In the Gaussian basis function model, the predictive variance rapidly increases near the training data boundaries but approaches nearly zero in distant regions. This indicates that the model exhibits overconfidence beyond the training data range, as it fails to align with the ground truth while maintaining an unrealistically low predictive variance. Consequently, the model's uncertainty estimation proves inadequate in areas lacking sufficient training data, leading to overly confident yet inaccurate predictions.

This is because the basis functions we use here produce only local features. In other words, when choosing a point x^* far from μ_j, we have $\phi(x^*) \to (0, 0, ..., 0)$, causing our prediction to also approach zero, which is not well generalized. This locality prevents the model from properly quantifying uncertainty and leads to a near-zero predictive variance.

Q2.5 Explain why in regions far from training distribution, the predictive variance converges to this value when using localized basis functions such as Gaussians.

This is because the basis functions we use here produce only local features. In other words, when choosing a point x^* far from μ_j, we have $\phi(x^*) \to (0, 0, ..., 0)$, causing our prediction to also approach zero. This behavior results in poor generalization in data-sparse regions.

Also, let's recall that when $\phi(x^*) \to 0$:

$$\sigma^2_{pred} = \beta^{-1} + \phi(x^*)^T \Sigma \phi(x^*) \to \beta^{-1} \approx 0.08$$

This is coherent with the previous results.

To conclude, the locality of the basis function could prevent the model from properly quantifying uncertainty and lead to a near-zero predictive variance.

However, if we try to make it look a bit more global by extending the value of μ_j, we may observe a more familiar result to us:

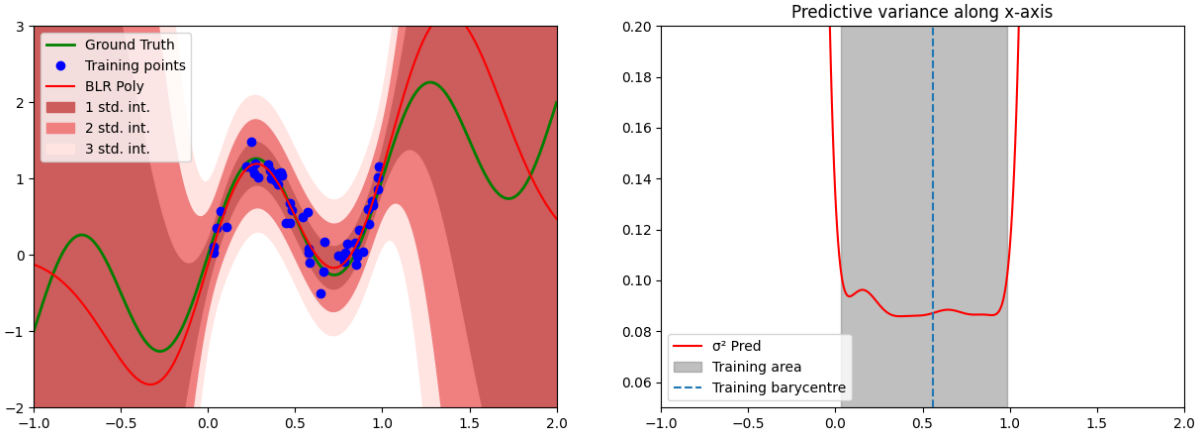


Figure 11: Visualization of predictive distribution in the Gaussian basis function model ($\mu_j \in [-2, 2], M = 15$)

4 Conclusion

In this report, we explored Bayesian linear regression as a probabilistic framework for modeling data relationships while quantifying uncertainty. We examined both linear and nonlinear models, utilizing basis functions such as polynomial and Gaussian functions to extend the model's capabilities. Through theoretical analysis and practical experiments, we observed how Bayesian methods improve prediction reliability by incorporating prior knowledge and refining uncertainty estimates as more data becomes available.

Overall, Bayesian approaches provide a robust and interpretable framework for regression tasks, offering valuable insights into both parameter estimation and predictive confidence.

***A little confusion...**

In the course, the likelihood is defined in two different ways:

$$p(y|x, w, \sigma) \sim \mathcal{N}(y; \phi(x)^T w, \sigma^2)$$

And also defined as:

$$p(y|x, w, \beta) \sim \mathcal{N}(y; \phi(x)^T w, \beta^{-1})$$

Both definitions appear in the slides. However, it is also stated in the slides and applied in the practical sessions that $\beta = \frac{1}{2\sigma^2}$ which is clearly inconsistent with the previous two equations.

3-b: Approximate inference

In Bayesian regression, given the prior distribution $p(w)$, we aim to compute the following two distributions:

- The posterior distribution:

$$p(w \mid D) \propto p(y \mid X, w)p(w)$$

- The predictive distribution:

$$p(y \mid x^*, D) = \int p(y \mid x^*, w)p(w \mid D)\, dw$$

For Bayesian linear regression, the posterior distribution is often Gaussian, allowing us to compute the predictive value y^* analytically.

However, for Bayesian logistic regression, $p(Y \mid X, w)$ is no longer Gaussian, so the posterior distribution $p(w \mid D)$ does not have an explicit solution. Approximate inference methods, such as Maximum A Posteriori (MAP), Laplacian approximation, and variational inference, are therefore required.

In this chapter, we will explore various approximate inference methods for binary classification problems and compare their performance. Additionally, we will implement Bayesian Neural Networks (BNNs) using variational inference and Monte Carlo dropout.

1 Bayesian Logistic Regression

In this section, we use the binary classification clusters generated by the `make_blob` function as the training dataset. Subsequently, we will train the decision boundary for the binary classification problem using three approximate inference methods: Maximum A Posteriori (MAP), Laplacian approximation, and variational inference.

1.1 Maximum-A-Posteriori Estimate

We approximate the posterior distribution by taking only the point with the highest probability, i.e.,

$$p(\boldsymbol{w}|\mathcal{D}) = \delta(\boldsymbol{w} - \boldsymbol{w}_{\text{MAP}}),$$

such that the predictive distribution can be approximated as:

$$p(\mathbf{y} = 1 \mid \boldsymbol{x}^*, \mathcal{D}) = \int p(\mathbf{y} = 1 \mid \boldsymbol{x}, \boldsymbol{w})p(\boldsymbol{w} \mid \mathcal{D})d\boldsymbol{w} \approx p(y = 1 \mid \boldsymbol{x}, \boldsymbol{w}_{\text{MAP}}).$$

The value of $\boldsymbol{w}_{\text{MAP}}$ can be determined by the following optimization problem:

$$\begin{aligned}
\boldsymbol{w}_{\text{MAP}} &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) \\
&= \arg\max_{\boldsymbol{w}} p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \\
&= \arg\max_{\boldsymbol{w}} \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w})p(\boldsymbol{w}) \\
&= \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} -log(p(y_n|\boldsymbol{x}_n, \boldsymbol{w})) - log(p(\boldsymbol{w})) \\
&= \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} \big(- y_n \log \sigma(\boldsymbol{w}^T \boldsymbol{x}_n + b) \\
&\quad - (1 - y_n) \log(1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_n + b))\big) + \frac{1}{2\sigma_0^2} \|\boldsymbol{w}\|_2^2.
\end{aligned}$$

where $\sigma(\cdot)$ represents the sigmoid function.

This approach is also known as the plug-in approximation and is generally used as a baseline method. The result is as follows:

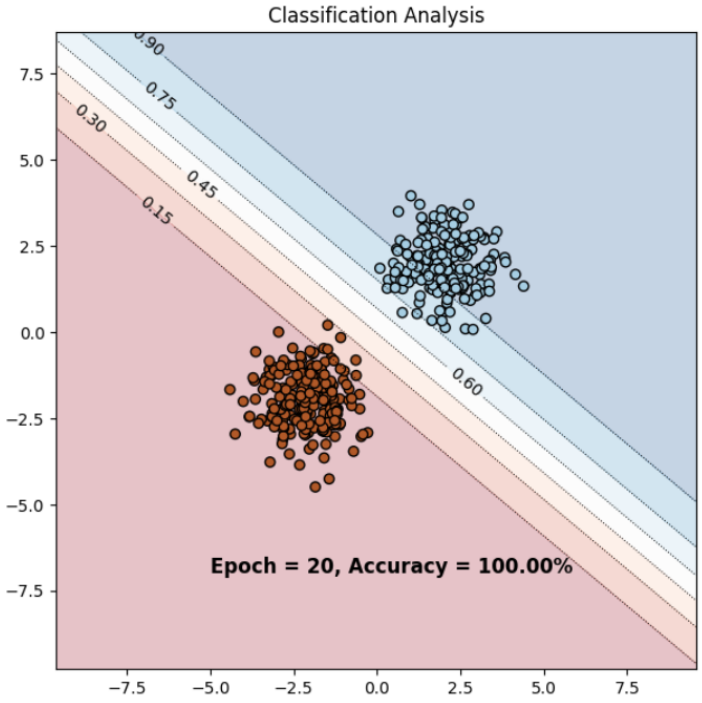


Figure 12: Classification by MAP

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?

The figure above shows the classification results after 20 training iterations, with an accuracy of 100%. The background in different colors represents the predicted probability $p(y = 1 \mid x, w_{\text{MAP}})$ at any point, and the dashed lines denote iso-probability contours.

The model's results resemble linear classification, which is clearly inconsistent with the actual situation. For points far from the training distribution, even though the model provides a predicted probability for these points, we know that the confidence in such predictions is relatively low. However, this uncertainty is not reflected in the figure. This demonstrates that MAP is a rather crude approximation method, and other more advanced approaches can be used to better capture the prediction uncertainty of the model in unseen regions.

1.2 Laplace Approximation

From the results above, we observe that using MAP as a replacement for the posterior probability $p(\boldsymbol{w} \mid \mathcal{D})$ is rather crude. Now, we attempt to approximate the posterior probability $p(\boldsymbol{w} \mid \mathcal{D})$ with a Gaussian distribution $q(\boldsymbol{w})$.

$$p(\boldsymbol{w} \mid \mathcal{D}) \approx q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- The mean of the Gaussian distribution is determined by $p(\boldsymbol{w} \mid \mathcal{D})$:

$$\boldsymbol{\mu}_{\text{lap}} = \boldsymbol{w}_{\text{MAP}} = \arg\min_{\boldsymbol{w}} -\log p(\boldsymbol{w} \mid \mathcal{D})$$

- The variance of the Gaussian distribution is given by the inverse of the Hessian matrix of $p(\boldsymbol{w} \mid \mathcal{D})$ at $\boldsymbol{w} = \boldsymbol{w}_{\text{MAP}}$:

$$(\boldsymbol{\Sigma}_{\text{lap}})^{-1} = \nabla\nabla_{\boldsymbol{w}}\big[-\log p(\boldsymbol{w} \mid \mathcal{D})\big]_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}}$$

The result is as follows:

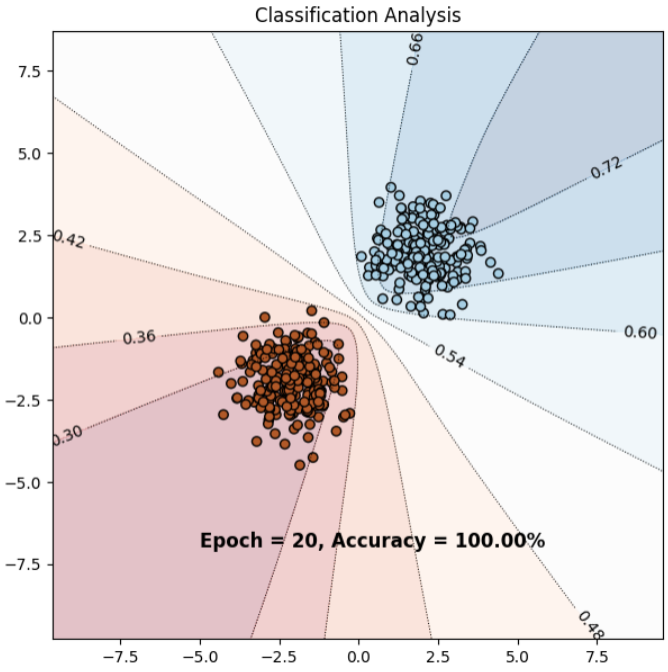


Figure 13: Classification by Laplace Approximation

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

After 20 iterations, the Laplace Approximation also achieved an accuracy of 100%. The background color represents the probability at each point $p(y = 1 \mid x, \mathcal{D})$, and the dashed lines indicate iso-probability contours.

Compared to the MAP estimate, the iso-probability contours of the Laplace Approximation are no longer linear but resemble an umbrella-like structure, with denser contours near the decision boundary. This distribution better reflects the uncertainty for points far from the training distribution and aligns more closely with real-world scenarios, especially as the model behaves more cautiously in regions not well-covered by the training data. However, there are still certain limitations, such as for points in the bottom-left and top-right corners of the figure.

1.3 Comment the effect of the regularization hyper-parameter WEIGHT_DECAY.

The regularization hyper-parameter `WEIGHT_DECAY` represents the L2 regularization method, which directly acts on the model's weight parameters. Its purpose is to add a weight penalty term to the loss function:

$$L_{\text{total}} = L_{\text{loss}} + \texttt{weight\_decay} \cdot \|\boldsymbol{w}\|_2^2$$

By imposing a weight penalty during the optimization process, `WEIGHT_DECAY` can limit excessively large weights, thus preventing the model from overfitting or underfitting and enhancing the generalization ability of the model.

1.3 Variational Inference

In variational inference, we aim to define an approximate variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$, parametrized by $\boldsymbol{\theta}$, such that the approximate predictive distribution can be expressed as:

$$p(y \mid \boldsymbol{x}^*, \mathcal{D}) \approx \int p(y \mid \boldsymbol{x}^*, \boldsymbol{w}) q_{\boldsymbol{\theta}^*}(\boldsymbol{w})\, d\boldsymbol{w}$$

where

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} KL(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w} \mid \mathcal{D})) \\
&= \arg\min_{\boldsymbol{\theta}} -\mathcal{L}_{\text{VI}}(\boldsymbol{X}, Y, \boldsymbol{\theta}) + \log(p(Y \mid \boldsymbol{X})) \\
&= \arg\min_{\boldsymbol{\theta}} -\mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log(p(Y \mid \boldsymbol{X}, \boldsymbol{w}))] + KL(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w})) \\
&= \arg\min_{\boldsymbol{\theta}} -\sum_{i=1}^{N} \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}[\log(p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}))] + KL(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w})) \\
&\approx \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} -\log(p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}_i)) + KL(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w}))
\end{aligned}$$

where $\boldsymbol{w}_i \sim q_{\boldsymbol{\theta}}(\boldsymbol{w}_i) = \mathcal{N}(\boldsymbol{w}_i \mid \mu_i, \sigma_i^2) = \mu_i + \sigma_i \odot \mathcal{N}(\epsilon_i \mid 0, 1)$.

For univariate Gaussian distribution, the KL term between the approximate variational distribution $q_{\boldsymbol{\theta}}(\mathbf{w}) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and the prior $p(\mathbf{w}) \sim \mathcal{N}(0, \sigma_p^2)$ rewrites:

$$\mathrm{KL}\left[q_{\boldsymbol{\theta}}(w))||p(w)\right] = \sum_{i=1}^{d}\left[log\left(\frac{\sigma_p}{\sigma_i}\right) + \frac{\sigma_i^2 + \mu_i^2}{2\sigma_p^2} - \frac{1}{2}\right]$$

1.4 Comment on the code of the VariationalLogisticRegression and LinearVariational classes.

Using the method described above, we can now define the `LinearVariational` class and implement Variational Logistic Regression using its variational layer.

- In the `LinearVariational` class, the inputs are `input_size`, `output_size`, and `prior_std`. Three tensors, `w_mu`, `w_rho`, and `b_mu`, are initialized as parameters to be optimized. Here, `w_mu` and `w_rho` determine the mean and variance of the Gaussian distribution from which $\boldsymbol{w}$ is sampled, and `b_mu` determines the bias term in the output, i.e., $y = \boldsymbol{x} \cdot \boldsymbol{w} + b$. This class also defines the `kl_divergence` method to conveniently collect the divergence loss for parameter optimization.
- In the `VariationalLogisticRegression` class, the inputs are `input_size` and `prior_std` (defaulting to 4.0). A variational fully connected layer is defined using `LinearVariational`, and the value of `kl_divergence` is collected using this fully connected layer.

The detailed code commentary can be found in the `3-b_Approximate_Inference.ipynb` file.

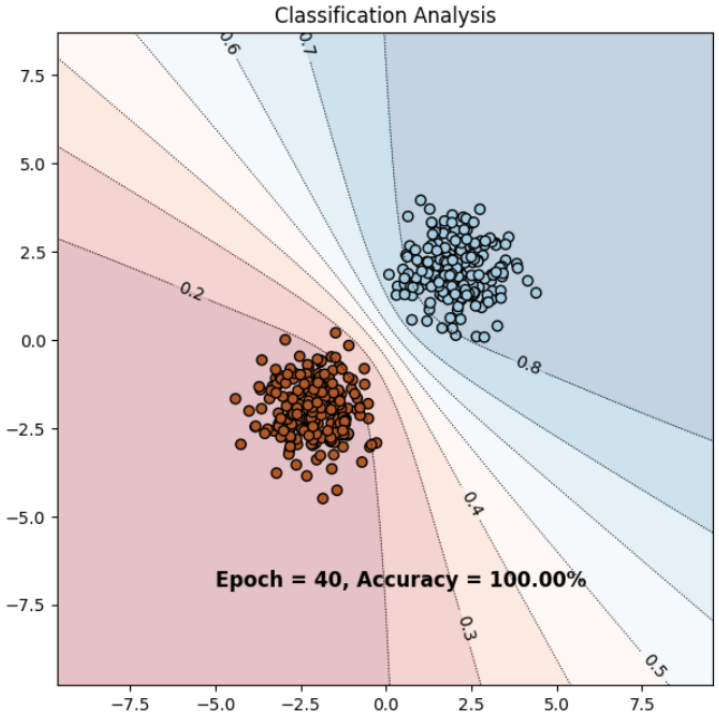


Figure 14: Classification by Variational Inference

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?

From the previous calculations, we know that minimizing the KL divergence is equivalent to minimizing the Evidence Lower Bound (ELBO), expressed as:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} -\log(p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}_i)) + KL(q_{\boldsymbol{\theta}}(\boldsymbol{w}) \| p(\boldsymbol{w}))$$

The first term in the equation above can be further transformed into $\sum_{i=1}^{N} -\log(p(y_i \mid f^{\boldsymbol{w}_i}(\boldsymbol{x}_i)))$, which corresponds to the binary cross-entropy loss function between $(y, \hat{y})$. Therefore, ELBO is the sum of the KL divergence and the binary cross-entropy.The detailed code commentary can be found in the `3-b_Approximate_Inference.ipynb` file.

Figure3 is the result of classification by Variational Inference. After 40 iterations, the Variational Inference also achieved an accuracy of 100%. The background color represents the probability at each point $p(y = 1 \mid x, \mathcal{D})$, and the dashed lines indicate iso-probability contours.

Compared to the MAP estimate, variational inference also captures the uncertainty of points far from the training data effectively and shows smoother probability transitions near the decision boundary. However, it exhibits overconfidence for points in the bottom-left and top-right corners of the figure, indicating certain limitations.

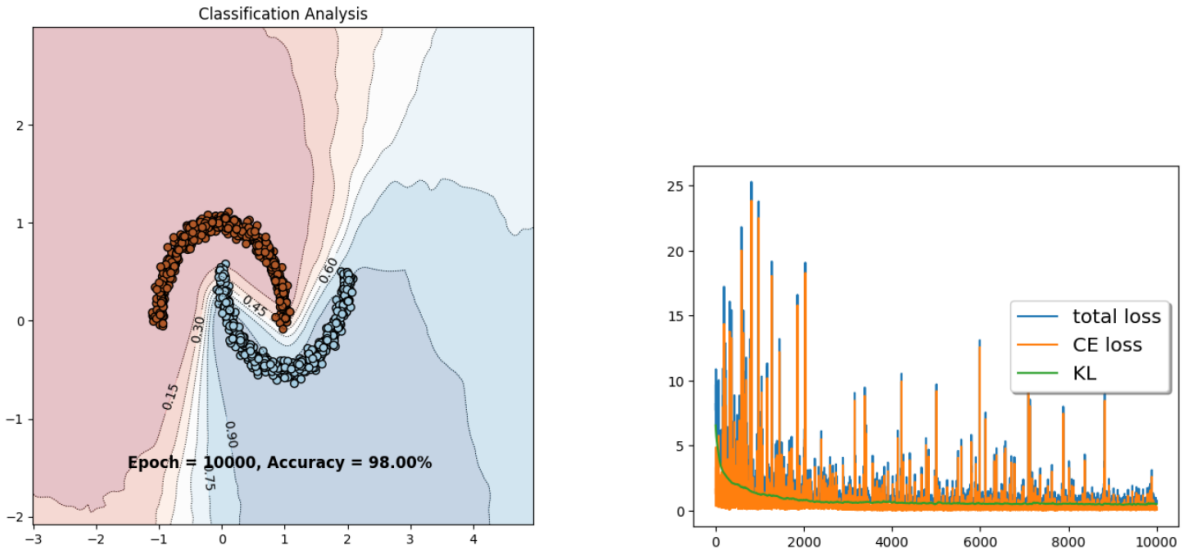
Although the results of the variational approximation and the Laplace approximation are very similar, their approaches are fundamentally different:

- **Variational approximation:** Variational approximation seeks a variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ to approximate the posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$. The optimization objective is to minimize the KL divergence. This method is more flexible and can adapt to complex distributions.
- **Laplace approximation:** Laplace approximation assumes that the posterior distribution $p(\boldsymbol{w} \mid \mathcal{D})$ is Gaussian around the MAP point. This method is relatively simple but may perform poorly for complex distributions.

2 Bayesian Neural Networks

2.1 Variational Inference with Bayesian Neural Networks

Now we will use Variational Inference to build a Bayesian Neural Networks. We define a variational MLP with 1 hidden layer and ReLU activation. The result is as follows:



(a) Classifcation by BNN (b) Loss of BNN

Figure 15: Result on The moons dataset

2.1 Analyze the results showed on plot.

In the left image, The background color and contour lines have the same meaning as in the previous experiments. After 10,000 epochs, the classification accuracy reaches 98%. For the moons dataset, we observe that the decision boundary is no longer linear but shows a clear curve. Additionally, the uncertainty for points far from the training data is significantly increased. This demonstrates that variational inference remains applicable to complex posterior distributions, and increasing the number of hidden layers can enhance the performance to some extent.

The right image shows the trend of CE loss, KL loss, and total loss during training. The CE loss and KL loss exhibit a fluctuating downward trend, with the KL loss steadily decreasing and stabilizing after 4,000 iterations.

2.2 Monte Carlo Dropout

2.2 Again, analyze the results showed on plot. What is the benefit of MC Dropout variational inference over Bayesian Logistic Regression with variational inference?

The figure below shows the classification results of an MLP using Monte Carlo Dropout. The background and contour lines in the plot are the same as those in the previous experiments. From the figure, we can see that after 500 epochs, the accuracy reaches 98.1%. Additionally, there are discrete circles along the contour lines, indicating that the model exhibits higher uncertainty for points far from the training data, which aligns better with real-world scenarios.

Compared to Bayesian Logistic Regression with variational inference, the model achieves 98.1% accuracy in just 500 epochs, even surpassing the results of variational inference after 10,000 epochs. MC Dropout simulates the weight distribution

by randomly dropping neurons, which is more efficient than approximating the posterior distribution through variational inference. Furthermore, it performs better in handling complex nonlinear problems.

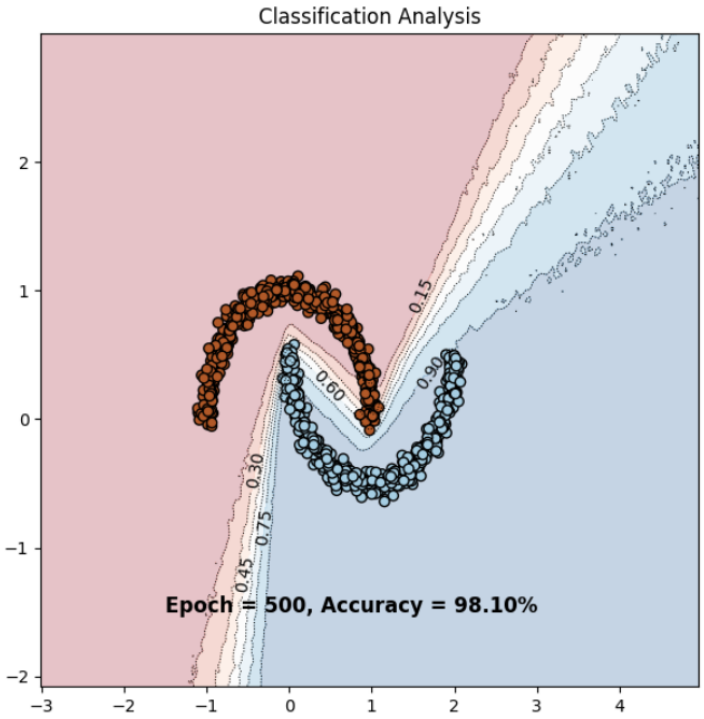


Figure 16: Classification by Monte Carlo Dropout

3-c: Uncertainty Applications

In this chapter, we focus on applications based on uncertainty estimation which is a great technique to tackle the tendencies for neural networks to exhibit overconfident predictions. We will address the capabilities of MC Dropout variational inference to qualitatively evaluate the most uncertain images according to the mode. Then, we'll move to 2 scenarios of failure prediction and out-of-distribution detection where good uncertainty estimation is crucial.

1 Monte-Carlo Dropout on MNIST

In this section we look into training a model in the style of LeNet-5 to implement Monte-Carlo dropout variational inference on the MNIST dataset. And we're interested in analyzing the output probabilities for uncertain samples. To spot the most uncertain images, we have implemented three methods to compute uncertainty estimations: **var-ratios**, **entropy**, and **mutual information**. And in this section we chooses the variational ratios to measure uncertainty.

The variation ratio,which gives a quantitative measure of uncertainty for an image, is calculated as follows:

$$\text{variation-ratio}[x] = 1 - \frac{f_x}{T}$$

where x denotes the given image, f_x is the frequency of the chosen label and T the number of pass.

1.1 What can you say about the images themselves. 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?

In the experiment, we apply a LeNet-5 style model with Monte-Carlo dropout for variational inference to calculate variation ratios for each image. In the visualization, we demonstrate each image with its variation ratio value, ground truth ("gt"), predicted class ("pred"), along with 5 histograms with distribution of mean output probability for each class, distribution of the predicted class across T forward passes, and distribution of output probabilities for specific classes (with the highest, the second and the fourth highest frequencies).

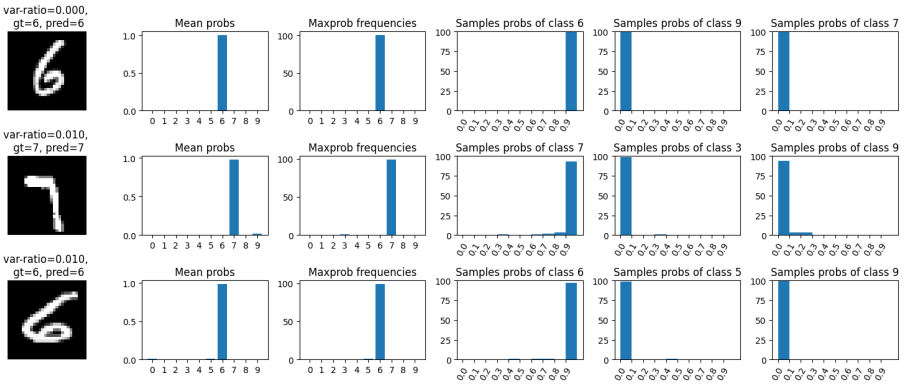


Figure 17: Random samples with variational ratio value and output probability distributions

Figure 17 shows random samples with clear images and small var-ratio values, meaning that the images are deemed certain by the model. Their predicted classes correspond with the ground truth. We can see that the histograms mainly concentrate on one single value for the prediction, meaning the model is very confident in its prediction and the sample was consistently classified as the predicted class.

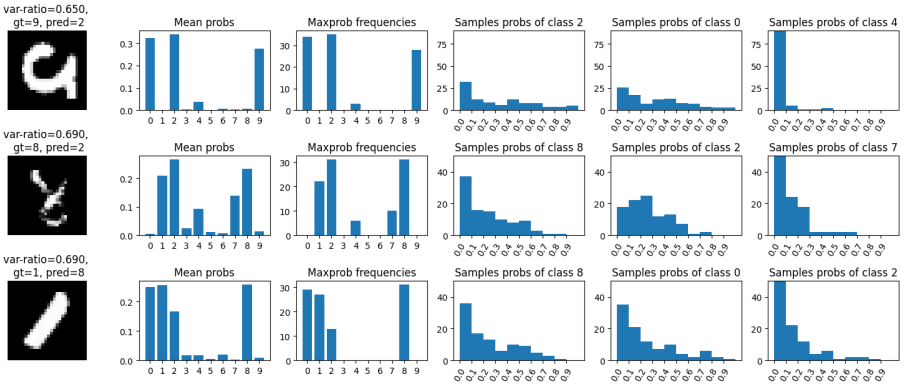


Figure 18: Top-3 most uncertain images along with their var-ratios value and output probability distributions

Figure 18 shows top-3 most uncertain images that are less easy-to-read, making it more challenging to identify the actual numbers.These three images are also shown with higher var-ratio values. Their histograms are more spread-out, indicating that the model's predictions were distributed across multiple classes, meaning that the model is uncertain about which class the sample belongs to.

We also observe the similarities between the histograms of mean probabilities and predicted class. suggesting that the model may choose a certain class as prediction even with low average probabilities. For example, the last row from Figure 18 shows that the class 2 with low mean probability (around 0.17), but from the last histogram we see it occasionally reaches high probability values around 0.8 - 0.9.

To conclude, comparing with the uncertain images, Random samples may exhibit more "peaked" probability distributions, resulting in higher confidence, while high-

uncertainty samples often have multi-peaked distributions and typically do not concentrate on a single class but rather spread across multiple classes, reflecting the model's "confusion."

And as for the failure cases, they can show that the model is overly confident when misclassifying samples, while high-uncertainty samples demonstrate the model's struggle to distinguish ambiguous features, leading to close probabilities for multiple classes.

2 Failure Prediction

In this section, we make comparisons of different methods to obtain a reliable confidence measure for model predictions, allowing the system to distinguish between correct and incorrect predictions and so identify cases where the model does not perform as expected or fails to make accurate predictions. When the confidence is low, the model may signal for alternative actions like human intervention, backup systems, etc.

In this section, we implement ConfidNet that learns true class probabilities (TCP) using a separate confidence network to address failure predictions. And we compare it with two other methods: MCP (Maximum Class Probability) where confidence is estimated using the highest softmax probability and MCDropout with entropy that uses entropy-based uncertainty obtained through multiple stochastic forward passes.

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?

To assess different methods, we need a suitable metric. Failure detection is treated as a binary classification task, with errors as the positive class, while correct predictions serve as the negative detection class. And since the model shows high test accuracies, we anticipate the occur of imbalanced data where failures (positives) are rare compared to correct predictions (negatives).

And AUPR (Area Under the Precision-Recall Curve) is more suitable for this case with imbalanced data compared with AUROC (Area Under the Receiver Operating Characteristic Curve). AUROC is less sensitive to class imbalance because it considers both positive (errors) and negative (correct predictions) classes equally, and therefore can yield misleading results, especially affected by the large number of true negatives.

Conversely, for AUPR,the precision-recall (PR) curve is the graph of the precision $= TP/(TP + FP)$ as a function of the recall $= TP/(TP + FN)$ where TP, TN, FP and FN are the numbers of true positives, true negatives, false positives and false negatives respectively. AUPR focuses on how well the model balances precision (avoiding false positives) and recall (identifying true positives), and thus provides more meaningful insights for failure detection, where errors (positives) are rare but critical.

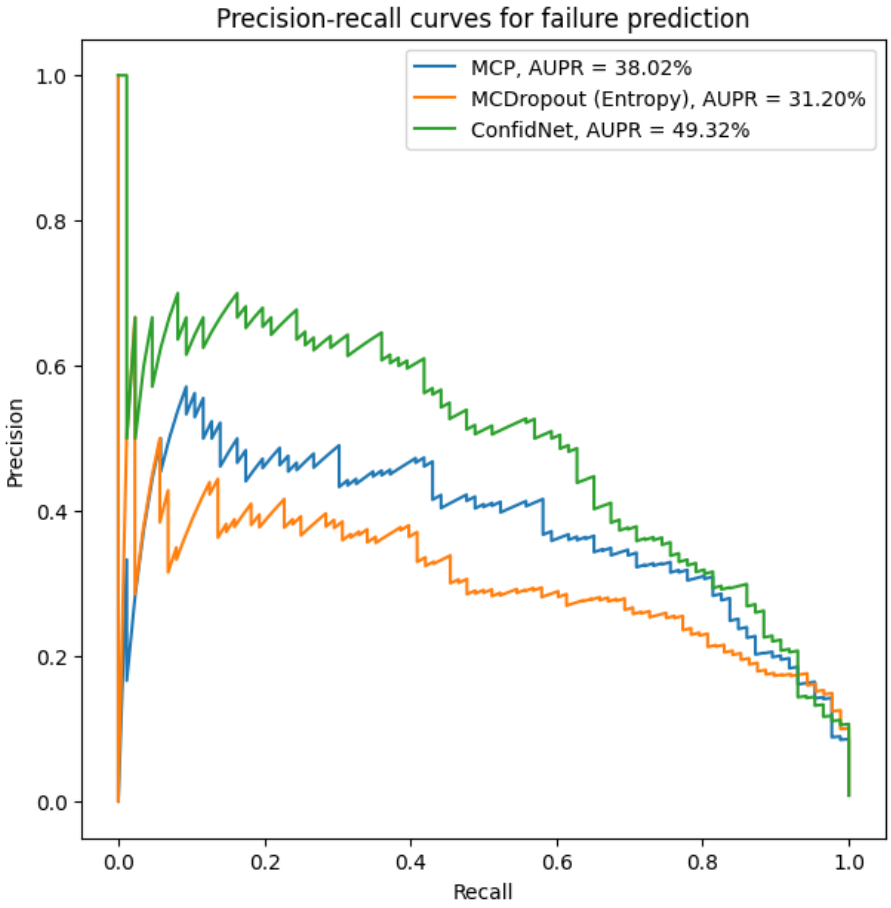


Figure 19: Precision-recall curves for tested methods including MCP, MCDropout, ConfidNet with corresponding AUPR values.

Figure 19 illustrates the precision-recall curves for each method (MCP, MCDropout, ConfidNet) with corresponding AUPR values. We can see that ConfidNet outperforms the other two methods with the highest AUPR value of 49.32%. This is due to its training that learns TCP directly, providing a more reliable confidence estimate. And ConfidNet tends to achieve higher precision with high recall. The learned confidence of ConfidNet enables better separation of correct and incorrect predictions.

With AUPR value of 38.02%, MCP struggles due to overconfident predictions for incorrect samples, leading to poor precision. As for MCDropout, it is computationally intensive. This method's reliance on multiple forward passes may introduce variability, leading to less consistent results with the lowest AUPR value of 31.20%.

So to conclude:

- **ConfidNet** maintains higher precision across a broader range of recall values, indicating its superior ability to identify errors without increasing false positives. And therefore is the most effective method for failure prediction due to its ability to learn and approximate the true class probability (TCP).
- **AUPR** is the preferred metric for evaluating failure prediction because it focuses on the critical positive class (errors), providing a more relevant evaluation

for this imbalanced problem.

3 Out-of-distribution detection

Neural networks generalize well when the training and testing data come from the same distribution. However, in real-world applications, testing data often deviate from the training distribution. Detecting out-of-distribution (OOD) examples is critical for ensuring the robustness of classifiers.

In this section, our objective is to accurately detect OOD samples, using the Kuzushiji-MNIST dataset (Japanese Kanji characters) as OOD samples for a model trained on the MNIST dataset. And we compare the results of 3 OOD detection methods: MCP (Maximum Class Probability), MC Dropout with Mutual Information and ODIN.

In this section, we implement the ODIN method that improves OOD detection by enhancing maximum softmax probabilities with:

1. Temperature Scaling

$$p(y = c \mid \mathbf{x}, \mathbf{w}, T) = \frac{\exp(f_c(\mathbf{x}, \mathbf{w})/T)}{\sum_{k=1}^{K} \exp(f_k(\mathbf{x}, \mathbf{w})/T)} \tag{3}$$

- $p(y = c \mid \mathbf{x}, \mathbf{w}, T)$: The probability of input $\mathbf{x}$ being classified as class c, under model parameters $\mathbf{w}$ and temperature T. This is computed via the softmax function.
- $f_c(\mathbf{x}, \mathbf{w})$: The logit value (pre-softmax output) of the model for class c, given input $\mathbf{x}$ and model parameters $\mathbf{w}$.
- T: The temperature parameter used in scaling. When $T > 1$, the softmax probabilities are smoothed, increasing separability between in-distribution and OOD scores.
- $\sum_{k=1}^{K} \exp(f_k(\mathbf{x}, \mathbf{w})/T)$: The normalization factor ensuring the probabilities sum to 1 across all K classes.

2. Inverse Adversarial Perturbation

$$\tilde{\mathbf{x}} = \mathbf{x} - \epsilon \cdot \text{sign}\big(- \nabla_{\mathbf{x}} \log(p(y = \hat{y} \mid \mathbf{x}, \mathbf{w}, T))\big) \tag{4}$$

- $\tilde{\mathbf{x}}$: The perturbed input, modified to enhance OOD detection.
- $\mathbf{x}$: The original input sample.
- ϵ: The perturbation magnitude, a small positive constant (e.g., 0.0014), controlling the intensity of the perturbation.
- $\text{sign}(\cdot)$: The sign function, which extracts the sign (positive or negative) of the gradient.
- $-\nabla_{\mathbf{x}} \log(p(y = \hat{y} \mid \mathbf{x}, \mathbf{w}, T))$: The gradient of the log-likelihood of the predicted class $\hat{y}$ with respect to the input $\mathbf{x}$. This represents how much a small change in $\mathbf{x}$ affects the model's confidence.

- $p(y = \hat{y} \mid \mathbf{x}, \mathbf{w}, T)$: The softmax probability of the predicted class $\hat{y}$, given input $\mathbf{x}$ and model parameters $\mathbf{w}$, with temperature scaling applied.
- $\mathbf{w}$: The model's weight parameters.
- T: The temperature parameter (same as in temperature scaling).

And we compare the ODIN method with MCDropout and MCP in OOD detection.

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

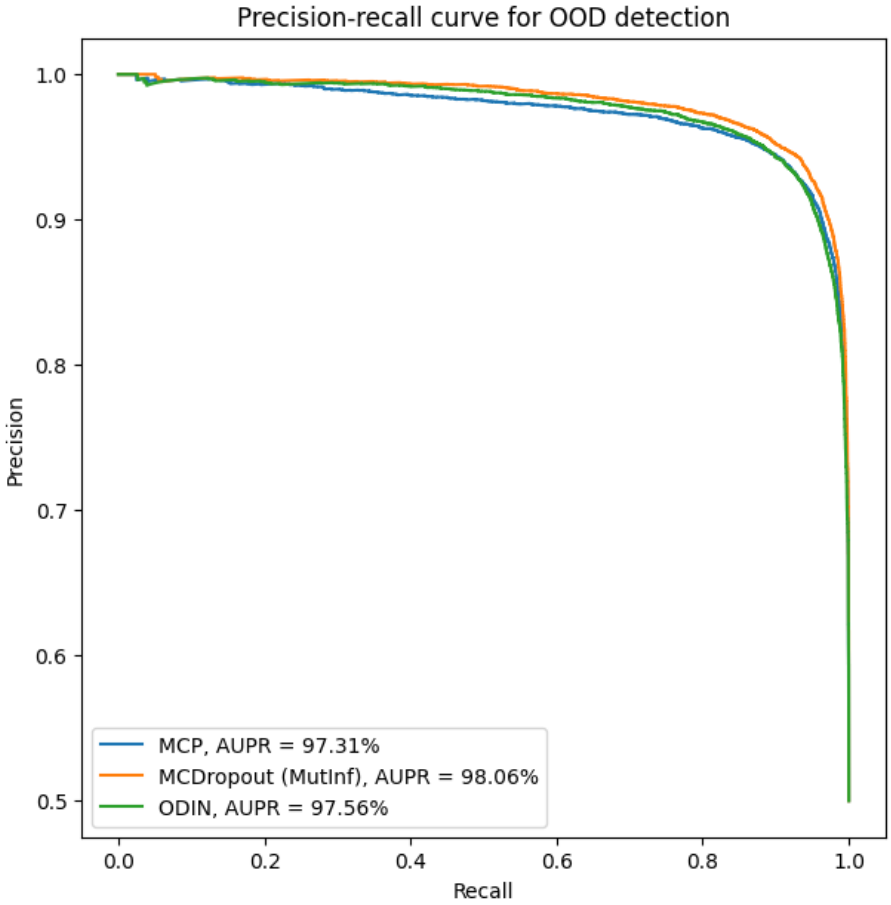


Figure 20: Precision-recall curves and AUPR values for 3 uncertainty metrics: MCP,

Figure 20 displays the precision-recall curves for 3 uncertainty metrics (MCP, ODIN, MCDropout mutual information) with their corresponding AUPR values. Although the three methods achieves similar AUPR values, we can see that ODIN performs slightly better than MCP with higher AUPR values, since ODIN combines temperature scaling and adversarial perturbation, enhancing in-distribution confidence relative to OOD confidence, while MCP relies on maximum softmax probability but lacks robustness for separating OOD from in-distribution samples. And we are also recieving a surprising result with MCDropout that has best performance among 3 OOD methods with highest AUPR value at 98.06%.This suggests that MCDropout captures model uncertainty more effectively by introducing randomness into the

network during multiple forward passes.It provides a richer estimate of uncertainty compared to single-pass methods like MCP or ODIN. We also observe from the experiment that the MCDropout took longer time in computation compared with ODIN and MCP.

To conclude, MCDropout outperforms the other two methods due to its robust estimation of uncertainty through multiple forward passes, at the expense of computational resources. ODIN improves upon MCP by using temperature scaling and adversarial perturbation, making it a practical choice for scenarios where computational resources are limited.And in this section, we are also choosing AUPR values over AUROC for its focus on the precision and recall of OOD detection, which is more relevant in imbalanced settings where OOD samples are rare but critical to detect.