Faculté des Sciences et Ingénierie - Sorbonne université

Master Informatique parcours - Données Apprentissage et Connaissances

Reconnaissance Des Formes pour l'analyse et l'interprétation d'images

Practicals Report

Section 3
Bayesian Deep Learning

Done by:
Samy NEHLIL - Allaa BOUTALEB
Supervised by:
Nicolas Thome

January 2024

Contents

1 Bayesian linear regression **1**

1.1 PART I: Linear Basis function model . . . 1

1.1.1 ⋆ Q1.2: Recall closed form of the posterior distribution in linear case. Then, code and visualize posterior sampling. What can you observe? . . . 1

1.1.1.1 Closed Form of the Posterior Distribution in Linear Case . . . 1

1.1.1.2 Observations . . . 2

1.1.1.3 Conclusion . . . 3

1.1.2 ⋆ Q1.3: Recall the Closed Form of the Predictive Distribution in Linear Case . . . 3

1.1.3 ⋆ Q1.5: Analyse these results. Why does predictive variance increase when far from training distribution? Prove it analytically in the case where $\alpha = 0$ and $\beta = 1$. . . . 4

1.1.3.1 Analysing the results . . . 4

1.1.3.2 Why does predictive variance increase when far from training distribution? 4

1.1.3.3 Prove it analytically in the case where $\alpha = 0$ and $\beta = 1$ . . . 4

1.1.4 ⋆ Bonus Question: What happens when applying Bayesian Linear Regression on the following dataset? . . . 5

1.1.4.1 Observations . . . 6

1.1.4.2 The Reason Behind Low Predictive Variance in the Hole . . . 7

1.1.4.3 Conclusion . . . 7

1.2 PART II: Non Linear models . . . 7

1.2.1 ⋆ Q2.2 : Code and visualize results on **sinusoidal dataset** using **polynomial basis functions**. What can you say about the predictive variance? . . . 8

1.2.1.1 Observations . . . 9

1.2.1.2 Conclusion . . . 9

1.2.2 ⋆ Q2.4 : Code and visualize results on the **sinusoidal dataset** using **Gaussian basis functions**. What can you say this time about the predictive variance? . . . 9

1.2.2.1 Understanding the Parameter s . . . 9

1.2.2.2 Observations . . . 10

1.2.2.3 Conclusion . . . 11

1.2.3 ⋆ Q2.5: Predictive Variance Behavior with Localized Basis Functions . . . 11

1.2.3.1 Convergence of Predictive Variance . . . 11

1.2.3.2 Behavior Far From Training Data . . . 12

2 Approximate Inference in Classification **13**

2.1 Bayesian Logistic Regression . . . 13

2.1.1 Maximum-A-Posteriori Estimate . . . 13

2.1.1.1 ⋆ Q1.1 Analyze the results provided by Figure 2.1. Looking at $p(y = 1|x, w_{MAP})$, what can you say about points far from train distribution? . 13

2.1.2 Laplace Approximation . . . 15

2.1.2.1 ⋆ Q1.2. Analyze the results provided by Figure 2.2. Compared to previous MAP estimate, how does the predictive distribution behave? 15
2.1.2.2 ⋆ Q1.3. Comment the effect of the regularisation hyper-parameter weight decay. 16
2.1.3 Variational Inference 16
2.1.3.1 ⋆ Q1.4. Comment the code of the VariationalLogisticRegression and LinearVariational classes. 16
2.1.3.2 ⋆ Q1.5. Comment the code of the training loop, especially the loss computation. Analyze the results provided by Figure 2.4. Compared to previous MAP estimate, how does the predictive distribution behave? What is the main difference between the Variational approximation and the Laplace approximation? 18
2.2 Bayesian Neural Networks 19
2.2.1 Variational Inference with Bayesian Neural Networks 19
2.2.1.1 ⋆ Q2.1. Analyze the results showed on Figure 2.5. 19
2.2.2 Monte Carlo Dropout 20
2.2.3 ⋆ 2.2. Again, analyze the results showed on Figure 2.6. What is the benefit of MC Dropout variational inference over Bayesian Logistic Regression with variational inference? 20

3 Uncertainty Applications **22**
3.1 Monte-Carlo Dropout on MNIST 22
3.1.1 LeNet-5 Network with Dropout Layers 22
3.1.2 Investigating Most Uncertain Samples 23
3.1.2.1 ⋆ Q1.1: 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? 24
3.2 Failure Prediction 25
3.2.1 ConfidNet 26
3.2.1.1 LeNet5ConfidNet 26
3.2.2 Evaluating failure prediction performances 27
3.2.2.1 ⋆ Q2.1: Compare the precision-recall curves of each method along with their AUPR values. Why did we use AUPR metric instead of standard AUROC? 27
3.3 Out-of-distribution detection 28
3.3.1 ⋆ Q3.1. Compare the precision-recall curves of each OOD method along with their AUPR values. Which method perform best and why? 29
3.3.2 ⋆ Extra: How do ϵ and *temperature* affect ODIN's performance? 30

List of Figures

1.1 Linear dataset distribution 1
1.2 Evolution of Posterior Distribution with Increasing Data Points 2
1.3 Predictive Mean and Variance Over Linear Dataset Distribution 4
1.4 Hole dataset distribution 5
1.5 Impact of Separative Distance of the gap in the Hole Dataset on Predictive Dynamics in BLR 6
1.6 Non-linear Dataset Distribution 7
1.7 Impact of Polynomial Degree on Predictive Dynamics in BLR 8
1.8 Impact of the Gaussian Basis Function's Spread on Predictive Dynamics in BLR 10

2.1 Depiction of a Bayesian Logistic Regression model utilized in a binary classification task, illustrating uncertainty representation and featuring a weight decay parameter of 5×10^{-2}. Two clearly differentiated clusters of data points — blue and red — symbolize distinct classes. The accompanying shaded regions around these clusters represent the model's predictive uncertainty, with lighter shaded areas denoting regions of lower confidence. . . 14
2.2 Illustration of a Bayesian Logistic Regression with Laplace approximation model applied to a binary classification task, with a weight decay of 5×10^2. 15
2.3 Illustration of a Variational Logistic Regression model applied to a binary classification task 18
2.4 Illustration of a Bayesian Neural Network model applied to a binary classification task. . 20
2.5 (a) 20
2.6 (b) 20
2.7 Bayesian Neural Network model applied to a binary classification task using (a) dropout and (b) Monte-Carlo dropout. 20

3.1 Var-ratios of randomly selected images 24
3.2 Var-ratios of the top-3 most uncertain images 24
3.3 ConfidNet 26
3.4 PR Curves for failure prediction 27
3.5 Kuzushiji-MNIST 28
3.6 PR Curves for various OOD methods along with their AUPR values 29
3.7 Computation time for each OOD method 29
3.8 AUPR for different combinations of ϵ and T (*temperature*) 31

Chapter 1

Bayesian linear regression

1.1 PART I: Linear Basis function model

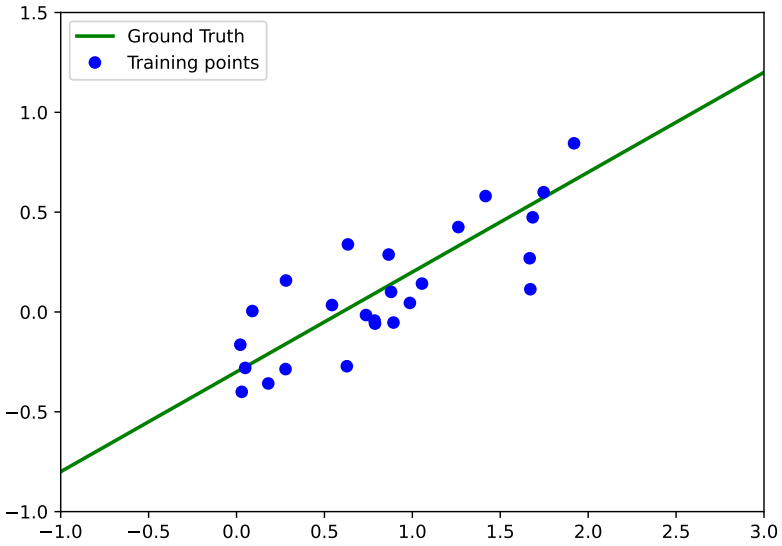


Figure 1.1: Linear dataset distribution

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

1.1.1.1 Closed Form of the Posterior Distribution in Linear Case

In Bayesian Linear Regression, the closed form of the posterior distribution for the linear case can be derived from Bayes' theorem. The goal is to find the posterior distribution $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$ of the weights $\mathbf{w}$, given the training data $\mathbf{X}$ and corresponding targets $\mathbf{y}$, under the assumption of a linear basis function model.

Assumptions

- The prior distribution over the weights $\mathbf{w}$ is Gaussian: $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$, where α is the precision of the prior.
- The likelihood of the data given the weights is also Gaussian: $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\Phi\mathbf{w}, \beta^{-1}\mathbf{I})$, where Φ is the design matrix and β is the noise precision.

Derivation

1. **Bayes' Theorem**: The posterior distribution is proportional to the product of the prior distribution and the likelihood function:

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \propto p(\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w})$$

2. **Substituting the Gaussian Distributions**: Plug in the Gaussian forms for the prior and likelihood:
$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \propto \exp\left(-\frac{\alpha}{2}\mathbf{w}^\top \mathbf{w}\right) \exp\left(-\frac{\beta}{2}(\mathbf{y} - \Phi\mathbf{w})^\top (\mathbf{y} - \Phi\mathbf{w})\right)$$

3. **Expanding and Rearranging**: Expand the exponent and rearrange terms to form a quadratic expression in $\mathbf{w}$:
$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \propto \exp\left(-\frac{1}{2}(\mathbf{w} - \mu)^\top \Sigma^{-1}(\mathbf{w} - \mu)\right)$$
where
   - $\Sigma = (\alpha\mathbf{I} + \beta\Phi^\top\Phi)^{-1}$ is the covariance matrix of the posterior.
   - $\mu = \beta\Sigma\Phi^\top\mathbf{y}$ is the mean of the posterior.

4. **Final Posterior Distribution**: The posterior distribution is a Gaussian with mean μ and covariance Σ:
$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\mathbf{w}|\mu, \Sigma)$$

This derivation shows how the posterior distribution of weights in Bayesian linear regression can be expressed in a closed form as a Gaussian distribution, where the mean μ and covariance matrix Σ are functions of the prior precision α, the noise precision β, the design matrix Φ, and the target values $\mathbf{y}$.

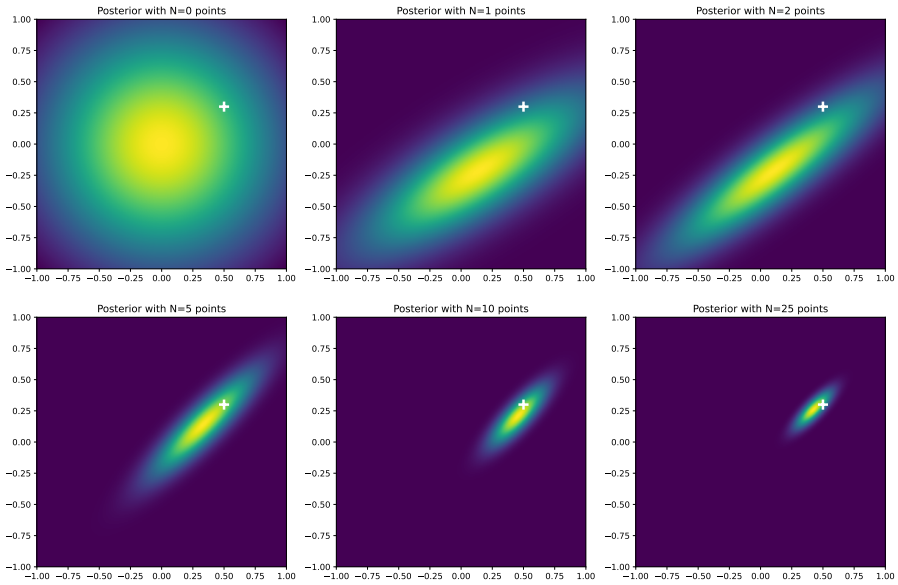


Figure 1.2: Evolution of Posterior Distribution with Increasing Data Points

1.1.1.2 Observations

- With $N = 0$ data points, the posterior distribution is broad and uniform, representing our prior knowledge without any data, centered around the prior mean.
- Introducing data points ($N = 1, 2$), we begin to see the posterior distribution narrow down, indicating increased certainty in the weights. The peak of the posterior, marked by the white plus sign, shows the true optimal values of the weights.
- At $N = 5$ and $N = 10$ data points, the peak of the posterior is more pronounced, demonstrating a tighter concentration around the true values as our belief becomes more confident.

- With $N = 25$ data points, the posterior is sharply peaked around the true optimal values, signifying a high level of confidence in the estimation of the weights.

1.1.1.3 Conclusion

- We observe a clear trend of the posterior distribution becoming more focused and peaked around the true optimal values as more data points are considered.
- This behavior underscores the Bayesian learning process, where each additional data point incrementally updates and refines our understanding of the model parameters.
- The progression of the posterior from a state of uncertainty to a state of high confidence illustrates the efficacy of Bayesian inference in assimilating data to improve model predictions.

1.1.2 ⋆ Q1.3: Recall the Closed Form of the Predictive Distribution in Linear Case

The closed form of the predictive distribution in the linear case for Bayesian Linear Regression can be derived from the posterior distribution of the parameters. Given a new point x^*, we wish to predict the corresponding target y^* and quantify our uncertainty about this prediction.
Given the posterior distribution over the weights $\mathbf{w}$ is Gaussian:

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \mathcal{N}(\mathbf{w}|\mu, \Sigma)$$

where μ and Σ are the posterior mean and covariance matrix computed from the training data.
The predictive distribution for a new input x^* is obtained by integrating over all possible parameter values weighted by this posterior:

$$p(y^*|x^*, \mathbf{X}, \mathbf{y}) = \int p(y^*|x^*, \mathbf{w}) p(\mathbf{w}|\mathbf{y}, \mathbf{X}) d\mathbf{w}$$

Assuming a Gaussian likelihood for new predictions given the weights:

$$p(y^*|x^*, \mathbf{w}) = \mathcal{N}(y^*|\mathbf{w}^\top \phi(x^*), \beta^{-1})$$

where $\phi(x^*)$ is the feature vector obtained by applying the basis function to the new input x^*, and β is the noise precision parameter.
The closed form of the predictive distribution is also Gaussian, with the mean and variance given by:

$$\mathbb{E}[y^*|x^*, \mathbf{X}, \mathbf{y}] = \mu^\top \phi(x^*)$$

$$\text{var}[y^*|x^*, \mathbf{X}, \mathbf{y}] = \beta^{-1} + \phi(x^*)^\top \Sigma \phi(x^*)$$

Thus, the function returned by `closed_form` takes a new input x^* and outputs the mean and standard deviation of the predictive distribution:

$$f_{model}(x^*) = (\mu^\top \phi(x^*), \sqrt{\beta^{-1} + \phi(x^*)^\top \Sigma \phi(x^*)})$$

This function encapsulates the predictive distribution for new data points, providing a mean and uncertainty estimate that incorporates both the noise in the observations and the uncertainty in the model

parameters.

1.1.3 ⋆ Q1.5: Analyse these results. Why does predictive variance increase when far from training distribution? Prove it analytically in the case where $\alpha = 0$ and $\beta = 1$.

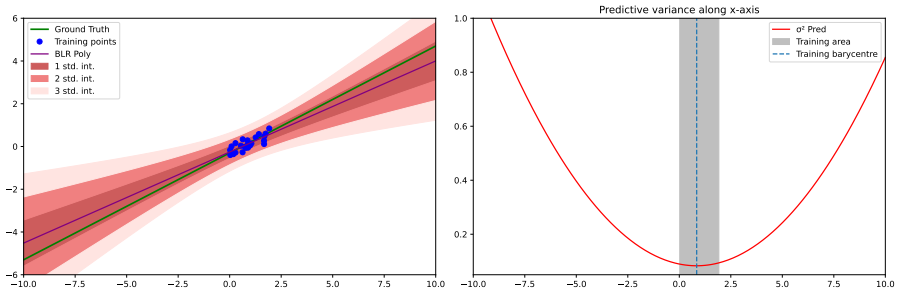


Figure 1.3: Predictive Mean and Variance Over Linear Dataset Distribution

1.1.3.1 Analysing the results

The left subplot illustrates the Bayesian Linear Regression (BLR) polynomial fit to the training data. The green line represents the ground truth, and the blue points are the training data. The purple line is the mean prediction from the BLR model, which aligns well with the ground truth within the range of the training data. The shaded areas represent the confidence intervals, with different colors indicating one, two, and three standard deviations from the mean prediction.
The right subplot depicts the predictive variance of the model along the x-axis. We observe that the variance is low near the training data, indicating higher confidence in the predictions, and it increases as we move away from the range of the training data.

1.1.3.2 Why does predictive variance increase when far from training distribution?

Predictive variance increases away from the training distribution due to the model's uncertainty in regions where it has not been trained. The Bayesian framework quantifies uncertainty in model parameters, which translates into uncertainty in predictions. Since the model has not seen data points far from the training distribution, it is less certain about its predictions in those areas, resulting in higher variance.

1.1.3.3 Prove it analytically in the case where $\alpha = 0$ and $\beta = 1$

When $\alpha = 0$, we are considering a model without a prior on the weights, essentially making the model equivalent to maximum likelihood estimation. The predictive variance for a new input x^* given by:

$$\text{var}[y^*|x^*, \mathbf{X}, \mathbf{y}] = \beta^{-1} + \phi(x^*)^\top \Sigma \phi(x^*)$$

reduces to:

$$\text{var}[y^*|x^*, \mathbf{X}, \mathbf{y}] = 1 + \phi(x^*)^\top (\Phi^\top \Phi)^{-1} \phi(x^*)$$

where Φ is the design matrix constructed from the training inputs.
The term $\phi(x^*)^\top (\Phi^\top \Phi)^{-1} \phi(x^*)$ represents the variance increase due to the uncertainty in the estimate of the weights, which depends on the input x^*. When x^* is close to the training data, $\phi(x^*)$ will be similar

to the rows of Φ, leading to smaller values in the product $\phi(x^*)^\top(\Phi^\top\Phi)^{-1}\phi(x^*)$. Conversely, when x^* is far from the training data, the values become larger, resulting in increased predictive variance.

1.1.4 ⋆ Bonus Question: What happens when applying Bayesian Linear Regression on the following dataset?

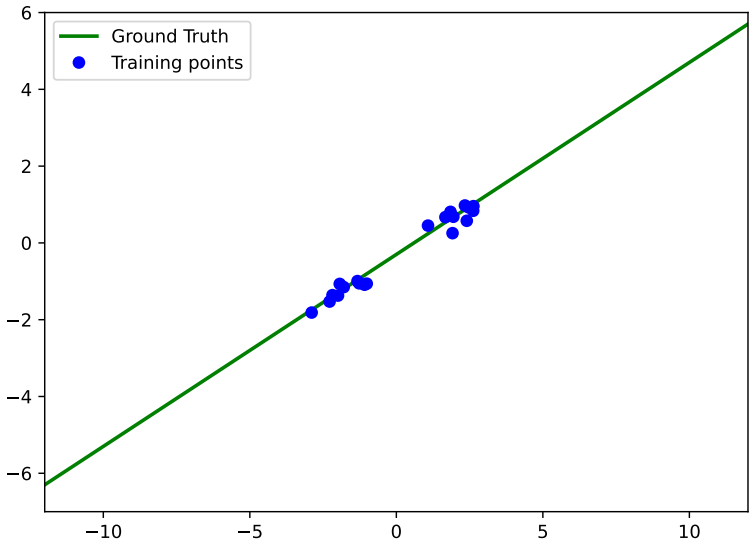
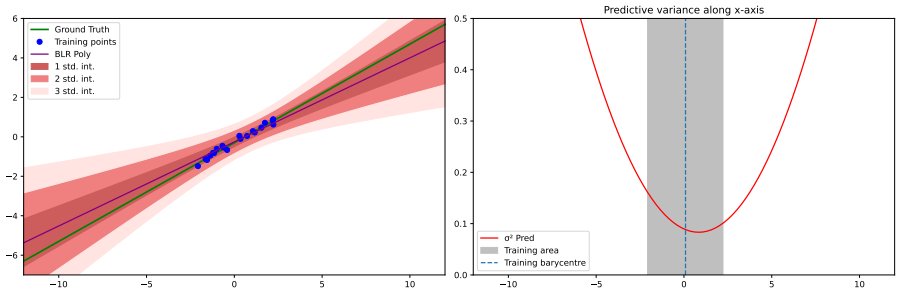
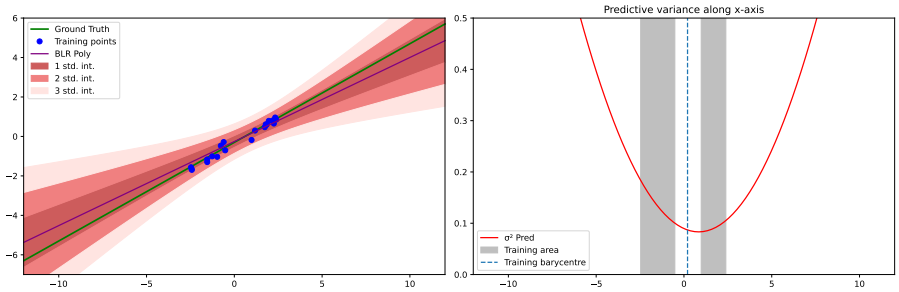


Figure 1.4: Hole dataset distribution

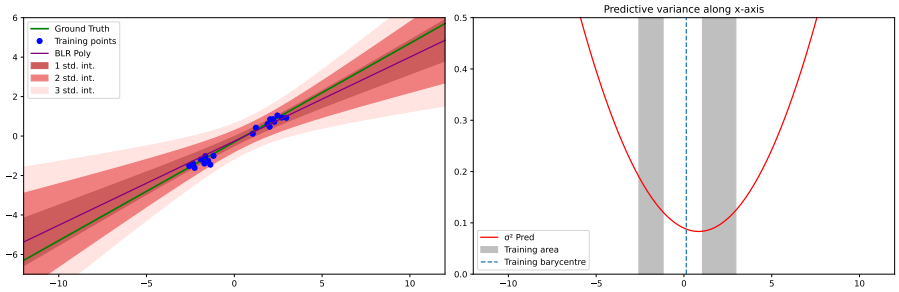
The 'hole' dataset is constructed by generating two clusters of training points with a gap or 'hole' between them. This gap represents a region in the input space where no training data is available. The dataset is defined such that the training points are uniformly distributed in two separate intervals, creating a clear absence of data in the intermediate region.



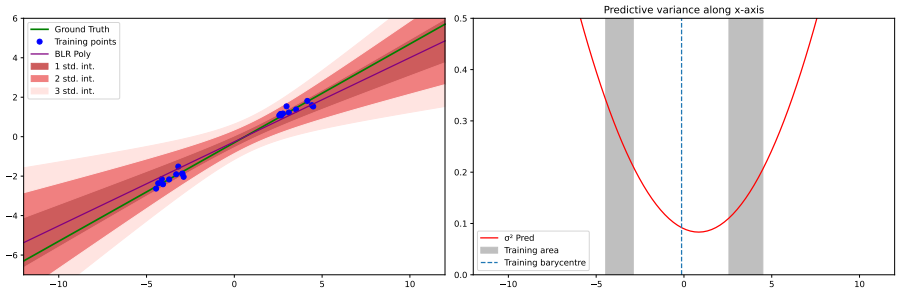
$distance = 0.5$



$distance = 1$



$distance = 2$



$distance = 5$

Figure 1.5: Impact of Separative Distance of the gap in the Hole Dataset on Predictive Dynamics in BLR

1.1.4.1 Observations

With increasing distances between the clusters, the following observations can be made from the figures:

- The BLR model's mean prediction follows the trend of the training data within the clusters and linearly interpolates across the hole where no data is present.
- Surprisingly, the variance in the region between the clusters remains low, which is contrary to what one might expect. Typically, low variance would be anticipated in the areas corresponding to the

training regions of the two clusters.

- As the distance between the clusters increases, the model maintains low uncertainty across the hole.

1.1.4.2 The Reason Behind Low Predictive Variance in the Hole

The low predictive variance within the hole, even as the distance between the clusters increases, can be analytically explained as follows:
When the training data is absent in a region, the BLR model's predictions rely on the prior and the information from the nearest points. For $\alpha > 0$, the model's weights are regularized, which controls the complexity of the model and encourages smoother predictions. Even for $\alpha = 0$, the effect of regularization from the data term $\beta\Phi^\top\Phi$ in the posterior covariance Σ leads to a smooth transition of the mean predictions between the clusters.
This behavior is an artifact of the model's assumptions, particularly the choice of the Gaussian prior and linear basis functions, which do not allow the variance to increase significantly in regions without data, as they do not account for the possibility of more complex underlying functions or discontinuities in the data-generating process.

1.1.4.3 Conclusion

The BLR model assumes a prior that smoothens the weight space, leading to interpolative behavior between clusters. Despite the absence of data in the hole, the model's predictions remain confident due to the influence of the nearby clusters. This phenomenon highlights a potential limitation of Gaussian priors in BLR, where the model may fail to represent uncertainty adequately in regions devoid of data. To address this, one could consider non-parametric models or modify the prior to better capture the uncertainty in regions without training data.

1.2 PART II: Non Linear models

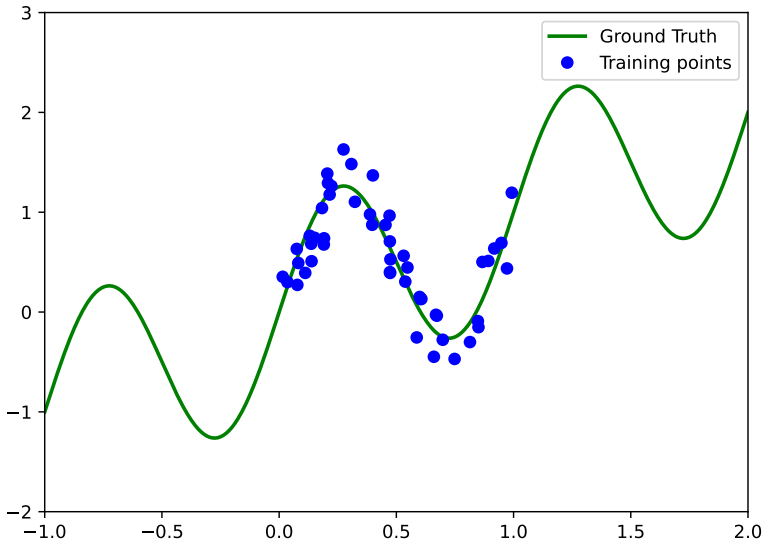
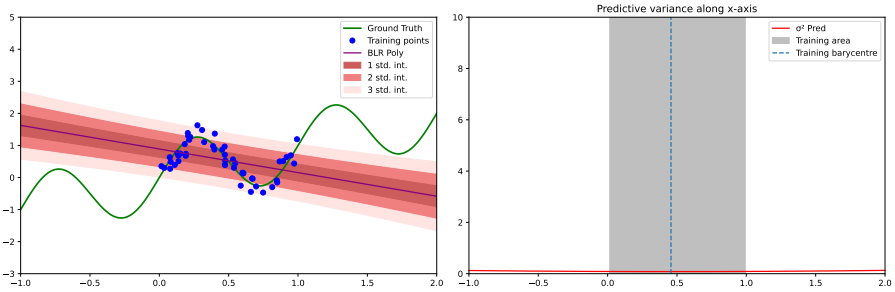


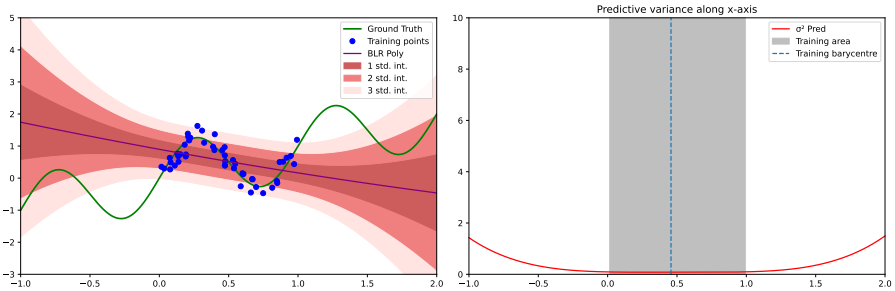
Figure 1.6: Non-linear Dataset Distribution

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

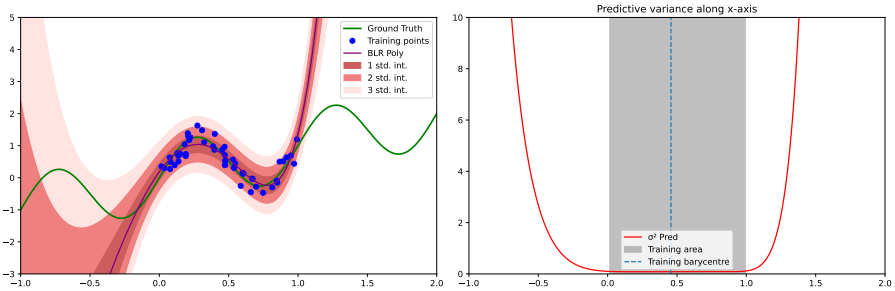
In this part, BLR with polynomial basis functions is applied to the sinusoidal dataset. We vary the degree of the polynomial to investigate its effect on the predictive mean and variance.



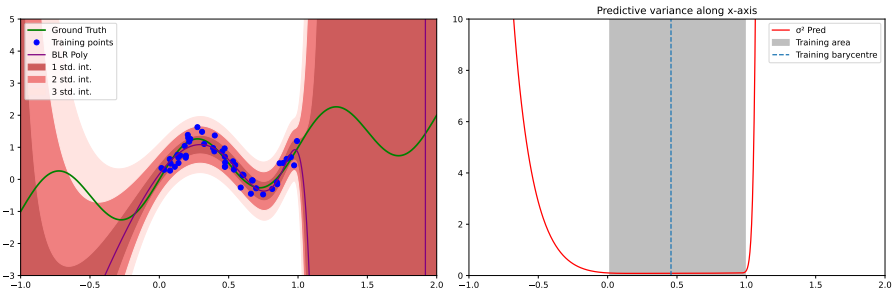
$degree = 1$



$degree = 2$



$degree = 5$



$degree = 20$

Figure 1.7: Impact of Polynomial Degree on Predictive Dynamics in BLR

1.2.1.1 Observations

As the degree of the polynomial basis functions increases, several patterns emerge in the predictive mean and variance:

- With low-degree polynomials, the BLR model cannot capture the sinusoidal pattern well, resulting in a predictive mean that does not match the ground truth closely.
- As the degree increases, the model becomes more flexible and begins to fit the training data more closely. This results in a predictive mean that better captures the sinusoidal shape of the ground truth.
- The predictive variance σ^2_{pred} starts as a relatively flat line for low-degree polynomials, indicating a uniform uncertainty across the domain.
- With higher degrees, the variance develops a 'U' shape, becoming larger away from the center of the training data. This is because high-degree polynomials are more sensitive to the training data, leading to increased confidence where the data is dense and higher uncertainty in regions away from it.

The evolution of σ^2_{pred} from flat to 'U' shaped with increasing polynomial degree is a manifestation of the bias-variance trade-off. Low-degree models have high bias and do not represent the data's structure well, leading to a flat variance. High-degree models have low bias but can overfit, which is reflected in the 'U' shaped variance that indicates high uncertainty in extrapolation regions.

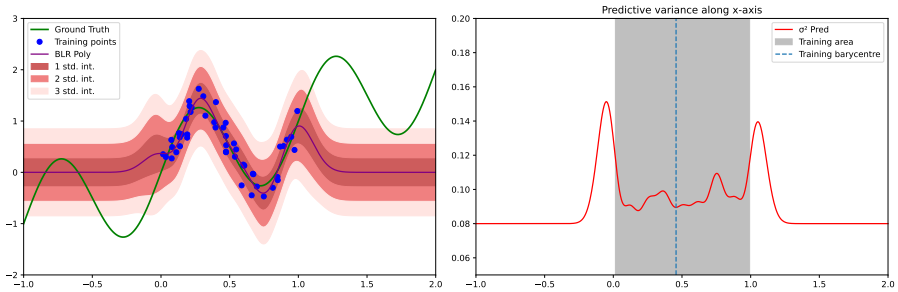
1.2.1.2 Conclusion

In conclusion, the choice of polynomial degree in BLR has a profound effect on the model's predictions. While higher-degree polynomials can capture complex patterns in the data, they also introduce higher variance in the predictions, particularly in regions not well-represented by the training

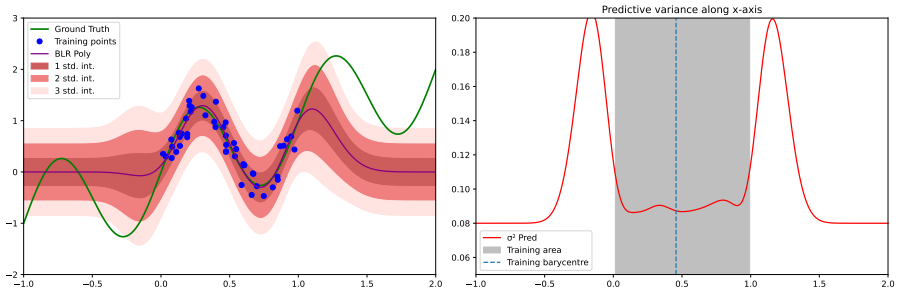
1.2.2 ⋆ Q2.4 : Code and visualize results on the sinusoidal dataset using Gaussian basis functions. What can you say this time about the predictive variance?

1.2.2.1 Understanding the Parameter s

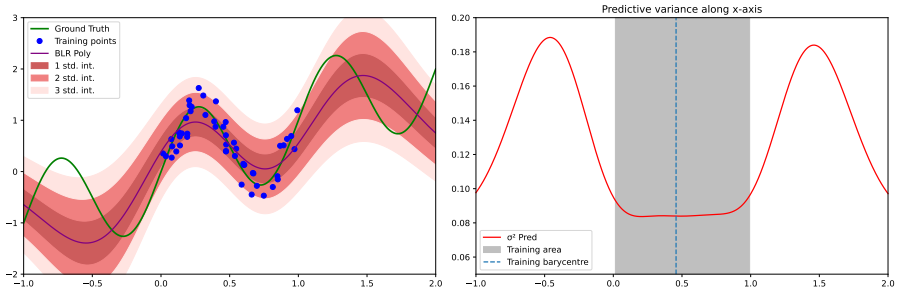
Gaussian basis functions transform the input space into a feature space where each feature is a Gaussian function centered at a specific point with a spread determined by the parameter s.
The parameter s in the Gaussian basis functions determines the width of the Gaussian bell curve. It is essentially the standard deviation of the Gaussian function, controlling the spread of the basis function around its mean μ_j.
A smaller s leads to narrower peaks, causing each basis function to react more to nearby points and less to distant points. Conversely, a larger s results in wider peaks, allowing each basis function to cover more of the input space and respond to points further away from its center.



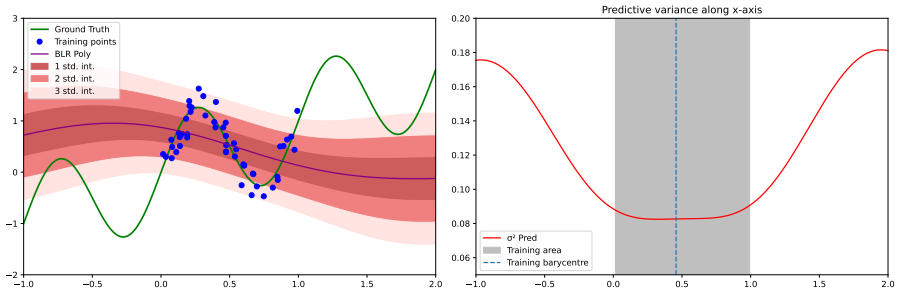
$s = 0.11$



$s = 0.2$



$s = 0.5$



$s = 1$

Figure 1.8: Impact of the Gaussian Basis Function's Spread on Predictive Dynamics in BLR

1.2.2.2 Observations

Upon examining the figures for different spacings s, the following observations can be made:

- For $s = 0.11$, the predictive variance is not uniform across the training area, indicating that the model is uncertain about its predictions even where training data is present. This fluctuation in variance reflects the model's sensitivity to the exact placement of the training points due to the narrow width of the Gaussian functions.

- As s increases to 0.2, the predictive variance within the training area becomes smoother, suggesting that the model is less sensitive to the training data and that the Gaussian functions are starting to generalize better. However, the variance still increases sharply outside the training area, showing a clear boundary of model confidence.
- With further increases in s to 0.5 and 1.0, the predictive variance within the training area becomes almost flat, implying a consistent confidence across this region. The transition of variance from the training area to the points outside is no longer abrupt but becomes gradual. The model's confidence decreases steadily as we move away from the training data, showing a more realistic representation of uncertainty.

These observations indicate that a smaller s makes the model highly flexible but also very sensitive to the training data, causing overfitting-like behavior. As s is increased, the model's flexibility is reduced, leading to predictions that generalize better across the training area but may underfit the data if s is too large. The behavior of the predictive variance with respect to s highlights the importance of choosing an appropriate scale for the Gaussian basis functions to balance the trade-off between fitting the data and generalizing well to new observations.

1.2.2.3 Conclusion

The spacing parameter s in Gaussian basis functions is crucial in controlling the trade-off between bias and variance in Bayesian Linear Regression. While small s values lead to a model with high variance and potentially overfitted predictions, larger s values smooth out the variance, providing a more consistent and generalizable model. The gradual increase in variance outside the training area for larger s values is a desirable property, indicating increasing uncertainty away from the observed data, which is more reflective of true model confidence.

1.2.3 ⋆ Q2.5: Predictive Variance Behavior with Localized Basis Functions

In the context of Bayesian Linear Regression, localized basis functions like Gaussians have a significant impact on the model's predictions and their associated uncertainty. One notable behavior is the convergence of predictive variance to a particular value in regions far from the training data.

1.2.3.1 Convergence of Predictive Variance

Localized basis functions such as Gaussian basis functions have limited influence or *local support*. In regions close to a Gaussian's mean μ_j, the basis function contributes significantly to the model's predictions. However, as we move away from the mean, the Gaussian's contribution quickly diminishes due to its exponential decay property.
The predictive variance in Bayesian Linear Regression is composed of two terms: the variance due to the noise in the data, denoted by β^{-1}, and the variance due to the uncertainty in the weights, which is influenced by the design matrix Φ and the prior on the weights. Mathematically, it is given by:

$$\text{var}[y^*|x^*, \mathbf{X}, \mathbf{y}] = \beta^{-1} + \phi(x^*)^\top \Sigma \phi(x^*)$$

Here, Σ represents the posterior covariance matrix of the weights, and $\phi(x^*)$ is the vector of basis functions evaluated at a new input x^*.

1.2.3.2 Behavior Far From Training Data

Far from the training distribution, the Gaussian functions centered around the training points approach zero. Consequently, the design matrix Φ corresponding to those distant points becomes sparse, filled with values close to zero. The product $\phi(x^*)^\top \Sigma \phi(x^*)$ therefore converges to a small value, leaving β^{-1} as the dominant term in the predictive variance.
Since β^{-1} is a constant representing the noise level in the data, the predictive variance converges to this noise level in regions far from the training distribution. This implies that, regardless of the model's complexity or the number of basis functions used, the model's confidence in its

Chapter 2

Approximate Inference in Classification

This chapter expands the exploration of Bayesian Deep Learning, building upon previous discussions to delve deeper into the nuances of classification tasks. It specifically addresses the challenges associated with these tasks, most notably the lack of a closed-form solution for the posterior distribution in Logistic Regression models. Through a detailed comparative analysis, the chapter assesses various approximate inference methods, such as Laplacian approximation and variational inference, in the context of binary classification problems. This examination is crucial for gaining deep insights into the practical implementation of Bayesian methods in complex machine learning scenarios. This represents a significant progression from elementary linear regression models previously explored, providing a thorough understanding of Bayesian principles within the machine learning landscape.

2.1 Bayesian Logistic Regression

In this section, the focus is primarily on Bayesian Logistic Regression. We begin by illustrating the difference between the continuous predictions typical in linear regression models and the discrete class label predictions fundamental to classification models. We specifically explore binary classification through logistic regression, examining different methods for approximating the posterior distribution. This examination is essential due to the intractability that arises from the absence of conjugacy between the likelihood and prior distributions in these models.

2.1.1 Maximum-A-Posteriori Estimate

2.1.1.1 ⋆ Q1.1 Analyze the results provided by Figure 2.1. Looking at $p(y = 1|x, w_{MAP})$, what can you say about points far from train distribution?

The examination of the results from the provided plot, with a specific focus on $p(y = 1|x, w_{MAP})$, unveils key insights into the model's behavior, particularly in areas distant from the training data distribution:

- **Linear Separation and Confidence Assessment:** The model's linear separation between classes suggests it has identified a linear decision boundary, typical of logistic regression or linear SVMs. Observing $p(y = 1|x, w_{MAP})$ reveals that the uncertainty (or model confidence) remains unchanged for points far from the training data. This is evidenced by the consistent coloration at the plot's periphery, indicating that the model maintains high confidence in its predictions, irrespective of data absence in those regions.
- **Constant Confidence Implications:** The unvarying high confidence away from the training data is a significant observation. Ideally, the model's confidence should diminish as it ventures beyond the data-experienced regions. This unchanging high confidence throughout suggests the model's overconfidence in its predictions, a serious concern in practical applications where such overconfidence can lead to erroneous decisions.
- **Characteristics of the MAP Baseline:** The Maximum A Posteriori (MAP) baseline, approximating the output distribution as a Dirac delta function for each training example, results in constant confidence contours. The model exhibits near-certainty in its predictions across the space,

except for a narrow zone around the decision boundary. This demonstrates a high certainty level in most areas, which may not accurately reflect the true uncertainty or variability of the model, especially in untrained regions.

- **Ideal Predictive Uncertainty Behavior:** A well-calibrated model should show rising uncertainty in predictions when moving away from training data-covered areas. This heightened uncertainty is crucial for acknowledging the model's limitations and averting overconfident predictions in unexplored feature space regions. The absence of this behavior in the current model suggests a potential issue in the model's uncertainty estimation or a limitation in the model architecture to capture the true data variability.

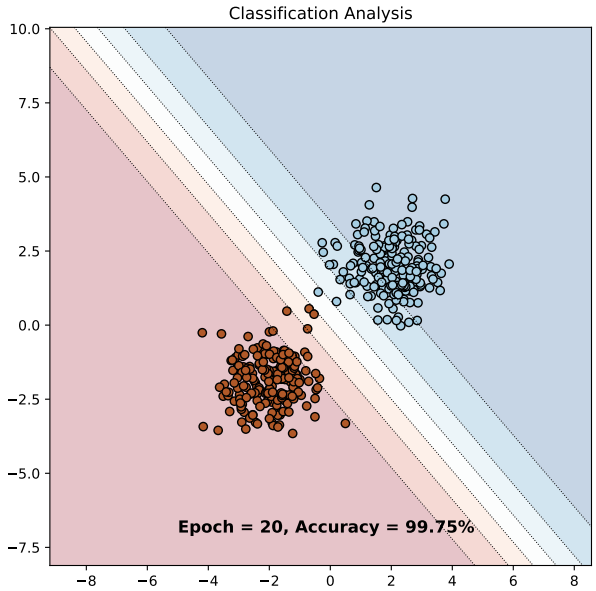


Figure 2.1: Depiction of a Bayesian Logistic Regression model utilized in a binary classification task, illustrating uncertainty representation and featuring a weight decay parameter of 5×10^{-2}. Two clearly differentiated clusters of data points — blue and red — symbolize distinct classes. The accompanying shaded regions around these clusters represent the model's predictive uncertainty, with lighter shaded areas denoting regions of lower confidence.

In summary, while the model achieves a clear linear separation and appears to be highly confident across the feature space, this uniform confidence, especially in regions far from training data, is a sign of overconfidence. This overconfidence can be misleading and is not representative of a well-calibrated model. It highlights the need for models that can better represent uncertainty and acknowledge their limitations in areas of the feature space where they lack training data.

2.1.2 Laplace Approximation

2.1.2.1 ⋆ Q1.2. Analyze the results provided by Figure 2.2. Compared to previous MAP estimate, how does the predictive distribution behave?

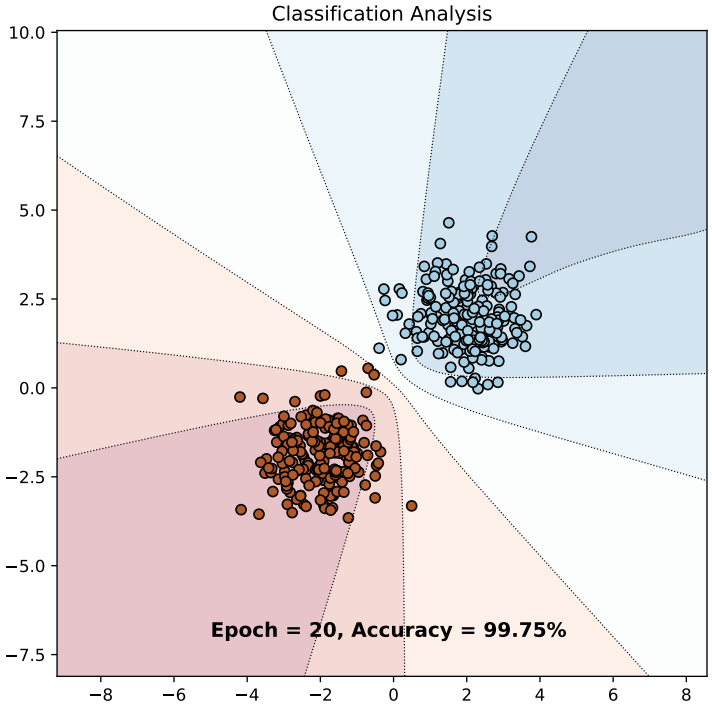


Figure 2.2: Illustration of a Bayesian Logistic Regression with Laplace approximation model applied to a binary classification task, with a weight decay of 5×10^2.

By approximating the posterior distribution with a Gaussian (as opposed to retaining just a point as with the MAP technique), we are able to curve the contour lines. This allows us to visually isolate the predicted boundary, which maintains a similar angle to the previous approach, and the accuracy remains unchanged. However, in regions distant from the training data, epistemic uncertainty becomes evident. Areas with weaker coloration, indicating low confidence, expand. Therefore, we observe a visible blur, which aligns with our objective this time.

The Gaussian approximation of the posterior distribution offers several advantages over the MAP technique. Firstly, it provides a more nuanced representation of uncertainty by modeling the distribution's shape rather than relying solely on a single point estimate. This allows for the curvature of contour lines, which better captures the complexity of the decision boundary.

Visually isolating the predicted boundary allows for a clearer understanding of the model's behavior, particularly in regions where the training data is sparse or absent. The expansion of regions with low confidence highlights the model's awareness of its uncertainty, which is crucial for making informed decisions.

In summary, by embracing uncertainty through the Gaussian approximation of the posterior distribution, we gain valuable insights into the model's behavior and its confidence in predictions. This approach not only enhances interpretability but also fosters trust in the model's outputs, especially in scenarios where uncertainty is inherent.

2.1.2.2 ⋆ Q1.3. Comment the effect of the regularisation hyper-parameter weight decay.

The weight decay hyper-parameter **regulates the complexity** of the model by imposing a penalty on the loss function for large weights. This penalty encourages the model to maintain smaller weight values, typically mitigating overfitting risks.
From a Bayesian perspective, weight decay corresponds to the precision (inverse variance) of the prior distribution over the weights. A higher weight decay value reflects a tighter prior, which pulls the weights closer to zero unless compelling evidence from the data suggests otherwise. Consequently, this adjustment can influence the predictive distribution, potentially making it more conservative. Consequently, the decision boundary may exhibit reduced flexibility, and the model may manifest increased uncertainty, particularly in regions distant from the training data.
Excessive weight decay leads to heightened predictive uncertainty, depicted by wider shaded areas, whereas insufficient weight decay results in overly confident predictions, indicated by narrower shaded areas, potentially unwarranted for unseen data.

2.1.3 Variational Inference

2.1.3.1 ⋆ Q1.4. Comment the code of the VariationalLogisticRegression and LinearVariational classes.

LinearVariational

- **Description:** Represents a single linear layer with variational inference applied. It approximates the weights and biases of the layer with distributions rather than fixed values.
- **Initialization:** The class is initialized with the variational parameters for the weights (w_μ, w_ρ) and biases b_μ of the layer. *prio_var* represents the variance of the prior distribution (σ_p^2), specifying our prior belief about the distribution of the weights.
- **Sampling Method:** Uses the reparametrization trick to sample from the variational posterior distribution for the weights $w_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. We sample from a centered isotropic multivariate Gaussian where $\sigma^2 = \log(1 + e^\rho)$ to avoid numerical issues. Thus, $w_i = \mu_i + \sigma_i \odot \epsilon_s$, where $\epsilon_s \sim \mathcal{N}(0, 1)$ is Gaussian noise. The reparametrization trick enables gradient backpropagation through the sampling process.
- **KL Divergence Method:** Calculates the Kullback-Leibler divergence between the variational posterior and the prior distribution for the weights:

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

  where σ_p^2 is the variance of the prior distribution $p(w)$ and (μ_i, σ_i^2) represent the mean and variance of the variational distribution $q_\theta(w)$.
- **Forward Method:** Defines the forward pass by performing a linear transformation, i.e., $w^T x + b$. We sample the weights and compute the output of the layer using the sampled weights and the mean of the biases.

VariationalLogisticRegression

- **Description:** Represents a logistic regression model using variational inference.

- **Initialization:** The class is initialized with one linear variational layer used to perform the linear transformation in logistic regression.
- **Forward Method:** Defines the forward pass for the logistic regression model by returning $f(x) = \sigma(w^T x + b)$ where σ is the sigmoid function.
- **KL Divergence Method:** Simply calls the same method of the LinearVariational layer to obtain the KL divergence term for the loss computation.

Implementation details of Variational Logistic Regression Model

1. Gaussian Distribution for Weights and Biases: The LinearVariational layer in this model treats weights and biases as random variables, following Gaussian distributions. Unlike standard linear layers with deterministic weights and biases, here the mean (w_{mu} for weights and b_{mu} for biases) and the standard deviation (expressed through w_{rho} for weights) are learned during training. This approach provides a probabilistic interpretation of model parameters, diverging from the conventional deterministic framework.

2. Sampling Using Reparameterization Trick: Sampling in this model leverages the reparameterization trick, a cornerstone for variational inference. This technique enables the model to backpropagate through stochastic nodes by reconfiguring the random variable in a manner that disentangles the randomness from the parameters. Such a setup is critical for maintaining the differentiability of the model while incorporating stochastic elements.

3. KL Divergence Computation: The $kl_divergence$ method is responsible for computing the Kullback-Leibler (KL) divergence between the variational posterior (distribution of weights and biases) and a predefined prior distribution. This divergence is integral to variational inference, assessing the deviation of the learned distribution from the prior, thereby quantifying the model complexity.

4. Forward Pass: The forward method dictates the processing of input data through the layer. It involves sampling weights and biases, followed by executing a linear transformation (matrix multiplication and bias addition). This step is crucial for generating predictions based on the current state of the model.

5. VariationalLogisticRegression Class: This class encapsulates the Logistic Regression model, employing variational methods with the LinearVariational layer at its core (fc_{var}). It distinguishes itself from standard logistic regression by its capability to express uncertainty in predictions, attributing to the variational approach.

6. Sigmoid Activation: Post-processing through the variational linear layer, the forward method applies a sigmoid activation function. This is a hallmark of logistic regression, converting the output into a probabilistic format (ranging between 0 and 1), suitable for binary classification tasks.

7. KL Divergence for the Whole Model: The VariationalLogisticRegression class's $kl_divergence$ method invokes the corresponding method in its LinearVariational layer. This divergence plays a pivotal role in the model's loss function, balancing data likelihood against model complexity (measured by its divergence from the prior).

In essence, these classes collectively offer a Bayesian perspective on logistic regression. By considering model parameters as distributions rather than fixed values, the model acquires the ability to not only predict but also quantify the uncertainty in its predictions, marking a significant advancement over traditional logistic regression approaches.

2.1.3.2 ⋆ Q1.5. Comment the code of the training loop, especially the loss computation. Analyze the results provided by Figure 2.4. Compared to previous MAP estimate, how does the predictive distribution behave? What is the main difference between the Variational approximation and the Laplace approximation?

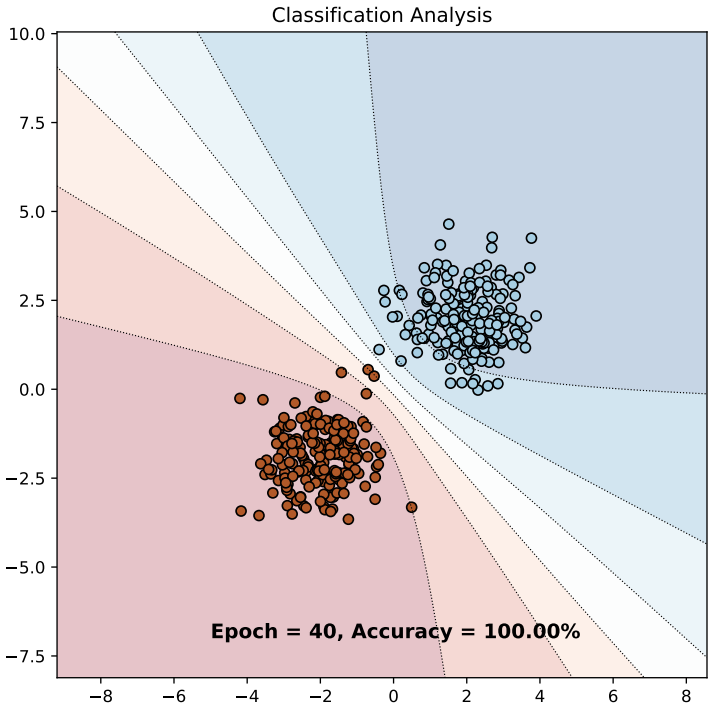


Figure 2.3: Illustration of a Variational Logistic Regression model applied to a binary classification task

Training Procedure

The training loop adheres to a standard PyTorch format. The loss function computes the Evidence Lower Bound (ELBO), which we aim to maximize. Ideally, we seek to maximize the likelihood of the data directly, but this often proves intractable due to the integral over the weights. Hence, we compute the Kullback-Leibler divergence $KL(q_\theta(w)||p(w))$ between the variational distribution $q_\theta(w)$ and the prior distribution $p(w)$. This divergence serves as a regularization term, encouraging the variational distribution to closely resemble the prior distribution. It quantifies the information lost when using $q_\theta(w)$ to approximate $p(w)$, which we endeavor to minimize.

To ensure effective model fitting to the data, we compute the negative log-likelihood $NLL(\theta; D)$ of the data under the model parameterized by the weights sampled from $q_\theta(w)$. This is accomplished using a binary cross-entropy loss. Consequently, as maximizing the ELBO is equivalent to minimizing:

$$LVI(\theta; D) = NLL(\theta; D) + KL(q_\theta(w)||p(w)),$$

we utilize gradient descent to update the parameters of the variational distribution to better approximate the true posterior.

The results of this variational approximation yield decision boundaries that are less rounded compared to Maximum a Posteriori and Laplace approximation methods, as depicted in Figure 2.4. This approach offers improved coverage of noisy data points, with all training points falling within regions of high confidence. Consequently, it yields a distinct decision frontier in the central area while effectively addressing aleatoric uncertainty.

Comparison with Maximum a Posteriori and Laplace Approximation

Compared to Maximum a Posteriori estimation, the variational approach does not merely identify the most probable weights (as MAP does) but instead approximates the entire posterior distribution over the weights. In the variational approach, the predictive distribution encapsulates the model's uncertainty about its predictions.

Compared to the Laplace approximation, the variational approximation actively optimizes a parameterized distribution to closely resemble the true posterior, with resemblance quantified by the KL divergence. This optimization typically involves a more complex objective function. On the other hand, Laplace approximation passively fits a Gaussian distribution around the MAP estimate, relying on the curvature of the log-posterior at that point. It assumes that the posterior is locally Gaussian and primarily focuses on finding the MAP estimate and computing the Hessian at that location.

2.2 Bayesian Neural Networks

In this section, we illustrate the expansion of Bayesian methodologies to neural network architectures, highlighting the application of variational inference to a Multi-Layer Perceptron (MLP). Our focus centers on implementing the concepts introduced in the preceding section to demonstrate their efficacy in handling intricate, non-linear datasets. Through this practical application, we aim to showcase the adaptability and robustness of Bayesian approaches within the domain of neural network modeling.

2.2.1 Variational Inference with Bayesian Neural Networks

2.2.1.1 ⋆ Q2.1. Analyze the results showed on Figure 2.5.

Bayesian Neural Network with Two Hidden Layers

By integrating Bayesian principles into a neural network with two hidden layers, we construct a model adept at discerning intricate patterns within the dataset. In this framework, each neuron's weight is conceptualized as a random variable, embodying our uncertainty regarding its true value. Consequently, the resulting decision boundary becomes complex and non-linear. Notably, the shaded regions, denoting the model's predictive uncertainty, indicate high confidence in proximity to the training data points, with uncertainty escalating as distance from the data points increases. These regions are delineated through the aggregation of outcomes from numerous stochastic passes.

Remarkably, this behavior engenders the formation of "clusters" reminiscent of the moon-shaped patterns inherent in the dataset. These formations are a consequence of our optimization of the Evidence Lower Bound (ELBO), facilitating the attainment of a model that closely approximates the dataset and consequently gives rise to these clusters. Furthermore, this approach enhances interpretability by furnishing outputs that convey not merely binary outcomes but rather the degree of confidence one might possess when predicting the locations of new data points.

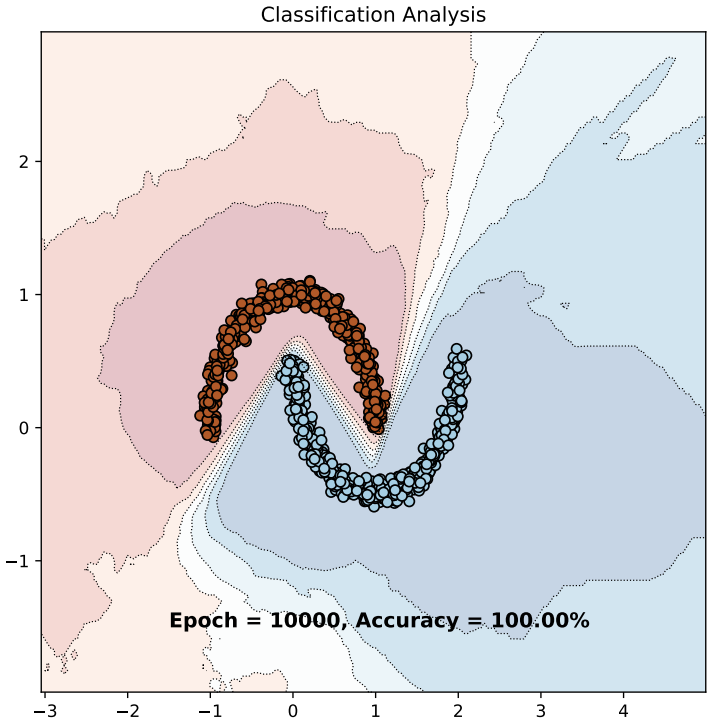


Figure 2.4: Illustration of a Bayesian Neural Network model applied to a binary classification task.

2.2.2 Monte Carlo Dropout

2.2.3 ⋆ 2.2. Again, analyze the results showed on Figure 2.6. What is the benefit of MC Dropout variational inference over Bayesian Logistic Regression with variational inference?

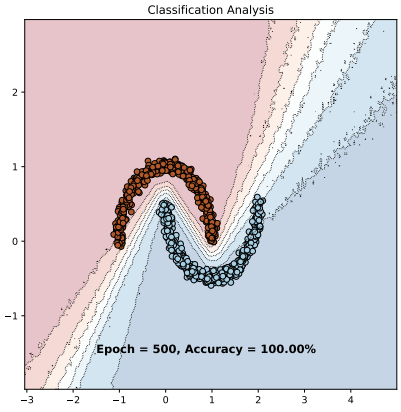


Figure 2.5: (a)

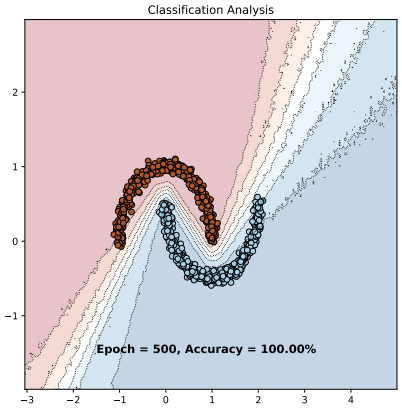


Figure 2.6: (b)

Figure 2.7: Bayesian Neural Network model applied to a binary classification task using (a) dropout and (b) Monte-Carlo dropout.

Monte Carlo dropout is a technique that corresponds with variational inference in Bayesian neural networks, where the dropout mechanism serves as a variational distribution for the network weights. Essentially, dropout introduces Bernoulli random variables, resulting in a posterior predictive distribution that incorporates weight uncertainty. The formula for the predictive distribution of an output y for a new

input x^* is given by:

$$p(y|x^*, X, Y) \approx \frac{1}{S} \sum_{s \in S} p(y^*|x^*, w_s),$$

where w_s represents the network weights after dropout, and S is the number of Monte Carlo samples or dropout iterations. However, compared to the variational approach, Monte Carlo dropout necessitates significantly fewer computational resources, as it eliminates the need to approximate a distribution with additional parameters. Consequently, it facilitates faster training with fewer epochs while still maintaining an interpretable model.

The results, as depicted in Figure 2.6b, showcase a decision boundary with adjacent bands indicating the model's confidence levels. These bands are formed by applying dropout during inference and averaging results from multiple stochastic passes. This approach illustrates the model's uncertainty in predictions, which contrasts with the smooth gradients of deterministic neural networks (as shown in Figure 2.6a). The speckled appearance of uncertainty regions in the plot reflects variations in confidence across different input regions, attributed to the randomness introduced by Monte Carlo Dropout. Integrating Monte Carlo Dropout in a standard neural network effectively transforms it into a Bayesian-like model, capable of expressing uncertainty in predictions. This approach not only enables the network to learn complex decision boundaries but also provides estimates of uncertainty, a feature often absent in regular neural networks. This probabilistic interpretation can lead to more informed decisions by indicating the reliability of the network's predictions across different input areas.

Chapter 3

Uncertainty Applications

3.1 Monte-Carlo Dropout on MNIST

In this section, we explore the application of Monte-Carlo (MC) Dropout for uncertainty estimation in deep neural networks, specifically applied to the MNIST dataset. The MNIST dataset, a collection of handwritten digits, serves as a standard benchmark in the field of machine learning. Our focus is to utilize MC Dropout, a technique for approximate Bayesian inference, to quantify the model's uncertainty in its predictions. This method involves enabling dropout during inference, allowing us to generate a distribution of outputs for each input. By analyzing these distributions, we can gain insights into the model's confidence and identify instances where the model is most uncertain. Such uncertainty estimation is crucial in critical applications where decisions based on model predictions require a high degree of reliability.

3.1.1 LeNet-5 Network with Dropout Layers

In our pursuit to implement Monte-Carlo Dropout variational inference, we adopt a network architecture inspired by LeNet-5. This architecture is renowned for its simplicity and effectiveness in handling image classification tasks, particularly with datasets like MNIST. The model's structure is modified to integrate dropout layers, essential for Monte-Carlo Dropout implementation.
The network comprises the following layers:

- Two convolutional layers, with the first having 6 channels, a kernel size of 5, padding of 2, and ReLU activation, and the second having 16 channels, a kernel size of 5, and ReLU activation.
- Max pooling layers following each convolutional layer, both with a kernel size of 2.
- Flattening the output from the convolutional layers before feeding it to the fully connected layers.
- A dropout layer with a probability $p = 0.25$ followed by a fully-connected layer of size 120 with ReLU activation.
- Another dropout layer with a probability $p = 0.5$, followed by a final fully-connected layer of size 10, mapping to the number of classes.

The implementation of the LeNet5-style neural network is augmented with dropout layers to enable Monte-Carlo Dropout during both training and inference. This is crucial for evaluating the model's uncertainty in predictions.
To train this model, we use a cross-entropy loss function, common in classification tasks, and optimize the network using Stochastic Gradient Descent (SGD) with momentum and weight decay. The training process spans 20 epochs, with each epoch providing insights into the loss and accuracy performance on the training dataset. Post-training, the model's state is saved, allowing for future inference without the need to retrain.
The inclusion of dropout layers and the choice of architecture make this model an appropriate candidate for exploring Monte-Carlo Dropout in uncertainty quantification, setting the stage for subsequent evaluations of model confidence and prediction reliability.

3.1.2 Investigating Most Uncertain Samples

In the realm of classification tasks, the quantification of uncertainty is pivotal for understanding the reliability of model predictions. In this subsection, we delve into various measures for estimating uncertainty in the context of classification using the Monte-Carlo (MC) Dropout approach on the MNIST dataset. These measures include variation ratios, entropy, and mutual information, each providing unique insights into the model's confidence.

- **Variation Ratios**: This measure involves collecting the most frequently predicted label across multiple stochastic forward passes. It is defined as variation-ratio$[x] = 1 - \frac{f_x}{T}$, where f_x is the frequency of the most common label, and T is the total number of passes.
- **Entropy**: Entropy captures the average amount of information contained in the predictive distribution, calculated as $\mathcal{H}[y|x,\mathcal{D}] = -\sum_c \left(\frac{1}{T}\sum_t p(y=c|x,w_t)\right) \log\left(\frac{1}{T}\sum_t p(y=c|x,w_t)\right)$.
- **Mutual Information**: This metric highlights points where the model's average uncertainty is maximized, defined as $\mathcal{I}[y,w|x,\mathcal{D}] = \mathcal{H}[y|x,\mathcal{D}] - \frac{1}{T}\sum_{c,t} p(y=c|x,w_t)\log p(y=c|x,w_t)$.

To compute these uncertainty estimates, we utilize a modified LeNet-5 model, integrating dropout layers to facilitate MC Dropout during inference. The model structure incorporates convolutional layers, max pooling, dropout layers, and fully connected layers, designed to capture the nuances of the MNIST dataset effectively.
Through a detailed implementation in PyTorch, we train the model over 20 epochs using cross-entropy loss and Stochastic Gradient Descent (SGD) with momentum and weight decay. The training process is essential to prepare the model for accurately evaluating uncertainty in predictions. Post-training, the model's performance is analyzed using the described uncertainty measures, allowing us to identify and scrutinize the most uncertain predictions. This investigation is crucial for applications where the cost of incorrect predictions is high, underscoring the need for robust uncertainty estimation methods in deep learning models.

3.1.2.1 ⋆ Q1.1: 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?

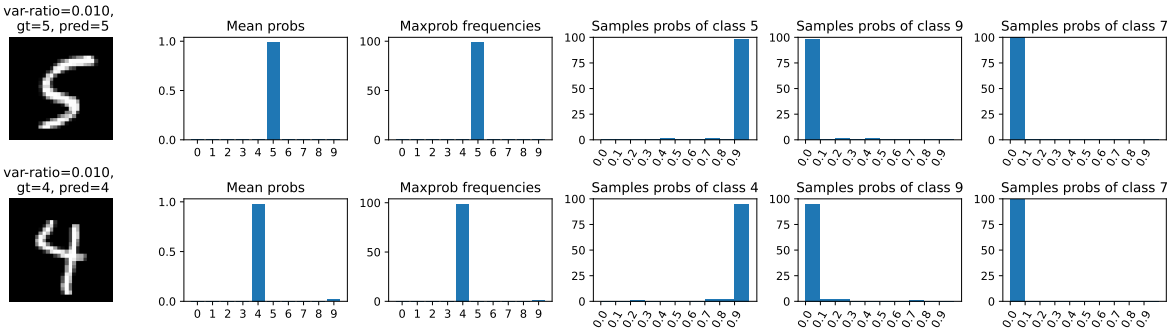


Figure 3.1: Var-ratios of randomly selected images

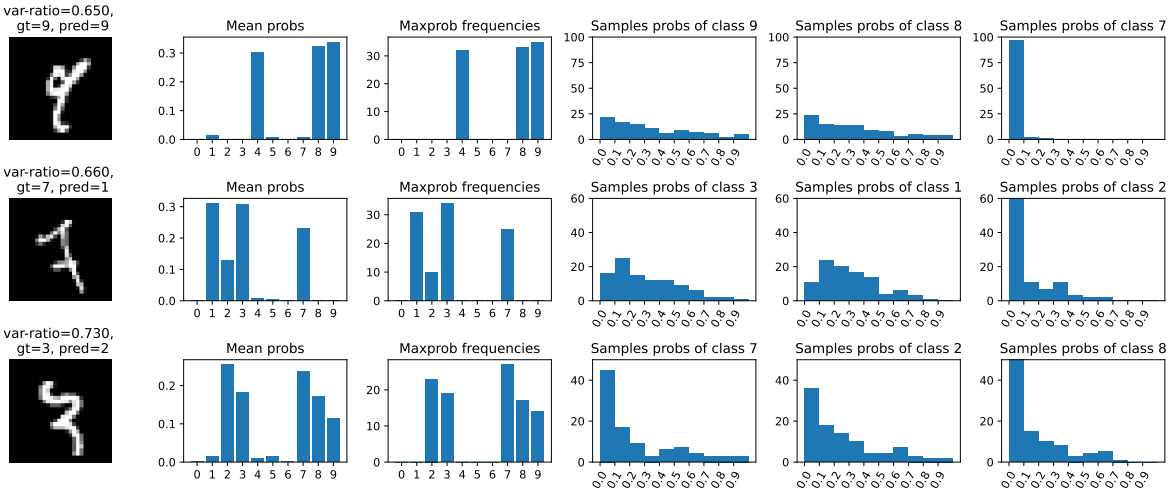


Figure 3.2: Var-ratios of the top-3 most uncertain images

- **Certainty in Predictions:**
  - When the model exhibits certainty (with a variance ratio approaching 0), the mean probabilities show a strong concentration on the true class.
  - The maxprob frequencies indicate a marked preference for a single class, signifying high confidence in the prediction.
- **Uncertainty in Predictions:**
  - In cases of heightened uncertainty (where the variance ratio is substantial), the mean probabilities are distributed across various classes.
  - This uncertainty is reflected in the maxprob frequencies, which show a more equitable distribution across classes.
  - The histograms of sample probabilities for specific classes demonstrate variability among the stochastic forward passes, leading to increased uncertainty.
- **Comparison of Distributions:** A comparison between the distributions for random images and those for the top uncertain images reveals a significant difference

– The most uncertain images generally display a uniform probability distribution across classes, indicating the model's perplexity and reduced confidence.

– Conversely, random images confidently identified by the model exhibit an asymmetrical distribution with a pronounced peak at the true class, indicating higher certainty.

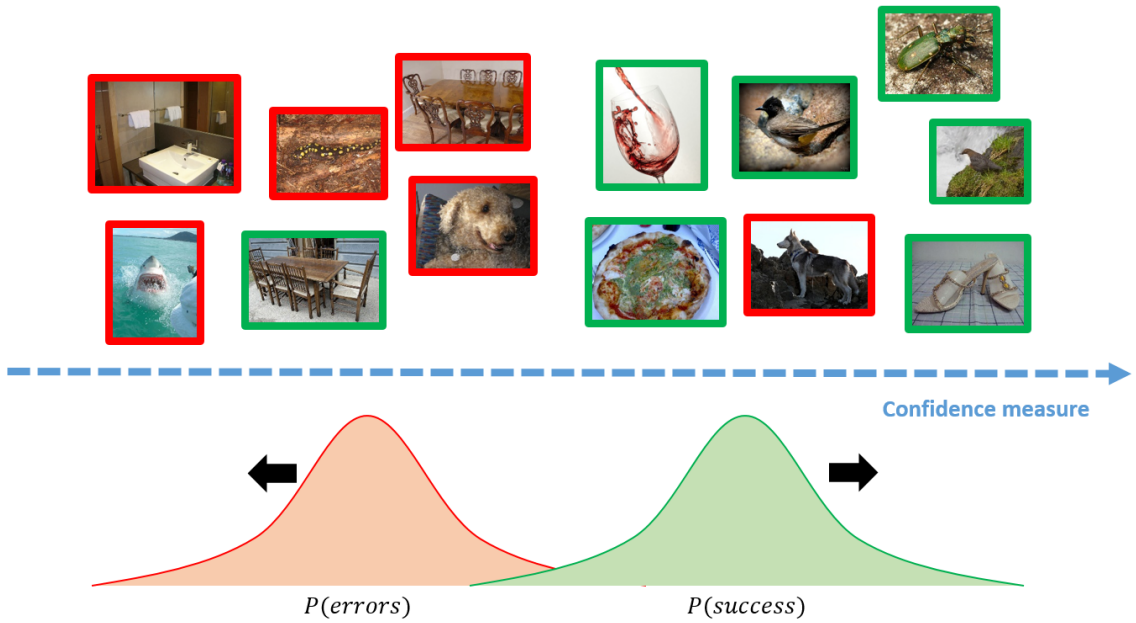
Conclusion: The histograms and probability distributions provide insights into the model's confidence levels. They are instrumental in understanding why certain images are classified with assurance, while others face misclassifications or elevated uncertainty in predictions.

3.2 Failure Prediction

In the realm of machine learning, the robustness of a model is not solely determined by its accuracy but also by its ability to gauge the confidence in its predictions. This aspect becomes crucial when deploying models in real-world scenarios where decisions based on incorrect predictions can have significant consequences. Failure prediction aims to endow models with the capability to discern the reliability of their outputs, thereby enhancing their utility and safety in practical applications.

The overarching goal of failure prediction is to develop confidence measures that accurately reflect the veracity of the model's predictions. Such measures enable a model to not only provide its best guess but also an associated confidence level, which can inform whether the prediction should be acted upon or further reviewed. In contexts where the stakes are high, such as autonomous driving or medical diagnosis, a well-calibrated confidence measure allows for a strategic handover to human oversight or supplementary systems whenever the model's certainty falls below a predefined threshold.

This section delves into the methodologies and techniques designed to predict failures, comparing traditional approaches with the proposed ConfidNet method, and exploring how these can be leveraged to make informed decisions about when to trust the model's judgment and when to seek alternative recourse.



3.2.1 ConfidNet

ConfidNet proposes a novel approach to uncertainty estimation by focusing on the True Class Probability (TCP) instead of the more common Maximum Class Probability (MCP). It introduces a secondary neural network, trained to predict the TCP using a mean squared error loss function, which measures the discrepancy between the model's confidence and the actual probability of the true class. This auxiliary network, built as a multilayer perceptron, is trained separately from the main classification model to ensure that the original predictive performance is not compromised.
During training, only the ConfidNet-specific layers are updated, which are tailored to generate a confidence score between 0 and 1. The model's effectiveness is evaluated using the Area Under the Precision-Recall curve (AUPR), a suitable metric for performance in scenarios with imbalanced classes. ConfidNet's method allows systems to make informed decisions based on the model's confidence in its predictions, potentially deferring to human judgment or alternative systems when uncertainty is high.

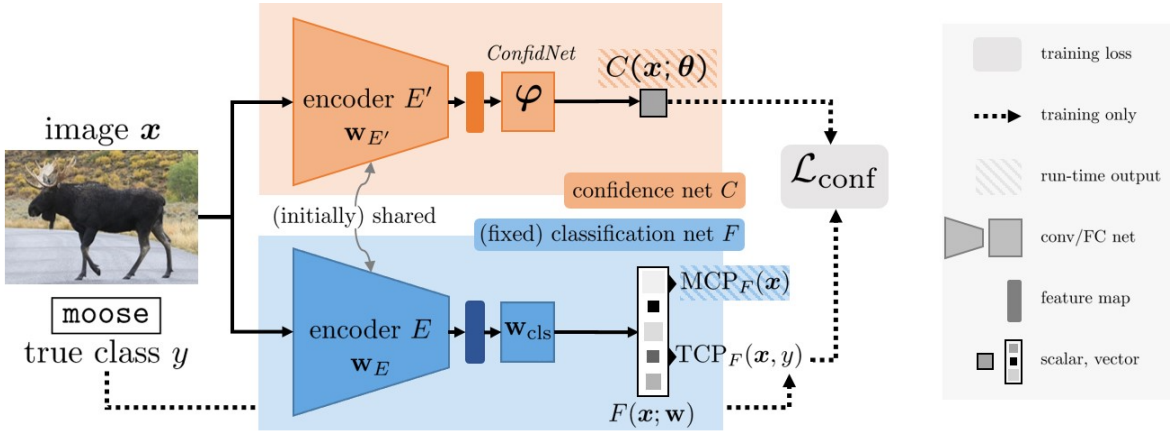


Figure 3.3: ConfidNet

3.2.1.1 LeNet5ConfidNet

In this section, the ConfidNet approach is applied to enhance the LeNet-5 architecture with an auxiliary branch for uncertainty estimation. The LeNet-5ConfidNet class extends the conventional LeNet-5 model by incorporating additional layers specifically designed to predict the True Class Probability (TCP). This probability reflects the model's confidence that its prediction matches the true class label.
The additional layers form a secondary neural network and consist of several fully connected layers followed by a sigmoid activation to produce a confidence score. The training process involves optimizing a mean squared error loss function, which minimizes the difference between the model's confidence and the actual TCP. To ensure the primary classification task remains unaffected, only the parameters of the auxiliary network are updated during the training of ConfidNet.
The model is trained for 30 epochs, with a special emphasis on fine-tuning the uncertainty estimation after 20 epochs. This includes freezing the original classification layers to prevent changes in their learned weights and disabling dropout to maintain consistency in the predictions.
The performance of ConfidNet is quantified using the Area Under the Precision-Recall curve (AUPR) metric, which is particularly suited for evaluating models on imbalanced datasets. The higher the AUPR, the better the model is at distinguishing between correct and incorrect predictions.

3.2.2 Evaluating failure prediction performances

In this part, we delve into the performance evaluation of failure prediction by ConfidNet, contrasting it with other established methods such as Maximum Class Probability (MCP) and Monte Carlo Dropout with entropy. The key metric for comparison is the Area under the Precision-Recall curve (AUPR), a robust measure for models operating on imbalanced datasets. Here, the focus is on the model's ability to identify classification errors, treating them as the positive class for the purpose of precision-recall analysis. The AUPR thus serves as a statistical tool to gauge the efficacy of each method in distinguishing between correct and incorrect predictions, crucial for applications requiring high-reliability predictions.

3.2.2.1 ⋆ Q2.1: Compare the precision-recall curves of each method along with their AUPR values. Why did we use AUPR metric instead of standard AUROC?

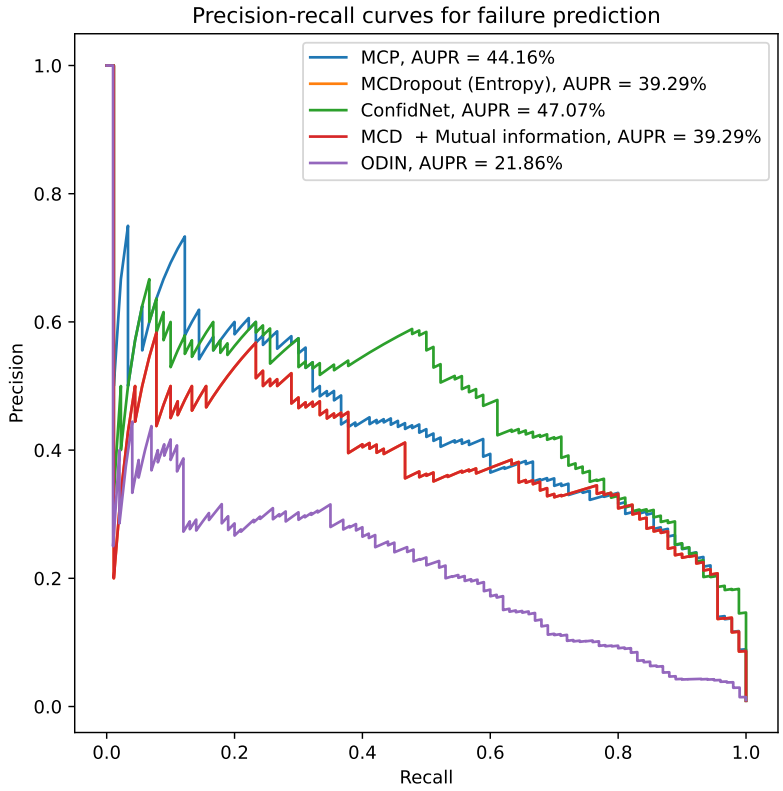


Figure 3.4: PR Curves for failure prediction

Method Performance Overview:

- **ConfidNet:** Achieves the highest AUPR at 47.07%, indicating superior capability in distinguishing between correct and incorrect predictions.
- **Maximum Class Probability (MCP):** Records an AUPR of 44.16%, showing strong performance despite its simplicity.
- **Monte Carlo Dropout:** Both mutual information and entropy approaches yield an equal AUPR of 39.29%, suggesting effective but lesser performance than ConfidNet and MCP.

- **ODIN:** Exhibits the lowest AUPR at 21.86%, indicating reduced reliability for failure detection in this context.

Implications and Insights:

- The preference for AUPR over AUROC is due to its effectiveness in representing model performance on the minority class, particularly in scenarios with high costs associated with false negatives.
- The high performance of ConfidNet suggests its approach in estimating true class probability aligns well with the inherent uncertainty in the task.

3.3 Out-of-distribution detection

In this section, we confront our MNIST-trained models with the challenge of Kuzushiji-MNIST—an out-of-distribution (OOD) dataset composed of Japanese Kanji characters—testing their ability to recognize when they encounter data that diverges from their training. This exercise is pivotal in assessing model robustness and reliability in real-world scenarios, where data often strays from the neatly curated conditions of a laboratory. By benchmarking against ODIN, a state-of-the-art OOD detection method, we aim to glean insights into the efficacy of our uncertainty estimation techniques and enhance our understanding of their performance boundaries when faced with the unfamiliar.

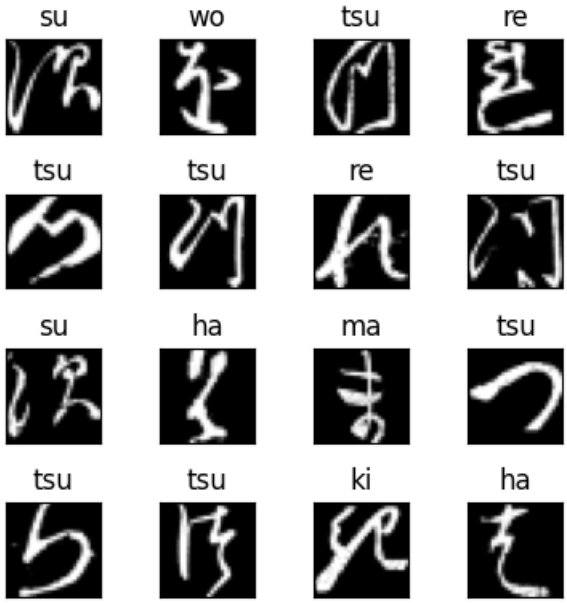


Figure 3.5: Kuzushiji-MNIST

Precision, recall, and the Area Under the Precision-Recall Curve (AUPR) were computed for both the Maximum Class Probability (MCP) method and MCDropout with mutual information on MNIST. Subsequently, the same metrics were evaluated on KMNIST, with samples from KMNIST considered as OOD instances. This process involved concatenating the uncertainties from both datasets and computing the precision-recall curve, along with AUPR, to obtain a quantifiable measure of the model's OOD detection capabilities.

3.3.1 ⋆ Q3.1. Compare the precision-recall curves of each OOD method along with their AUPR values. Which method perform best and why?

The precision-recall curves for out-of-distribution (OOD) detection highlight the nuances of each method's ability to discern between in-distribution and OOD samples, with AUPR providing a single-figure measure of performance across all levels of recall.

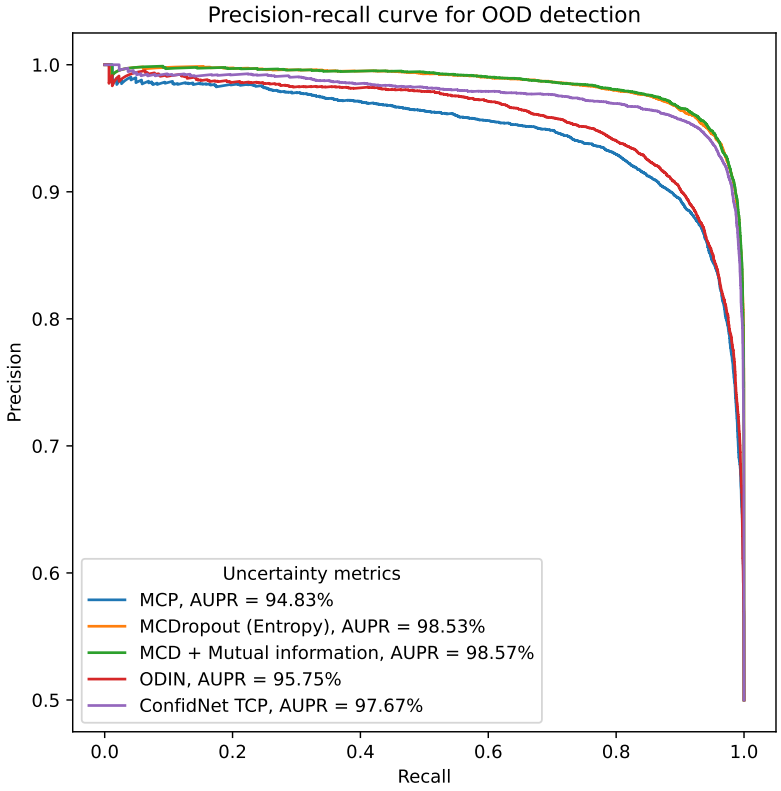


Figure 3.6: PR Curves for various OOD methods along with their AUPR values

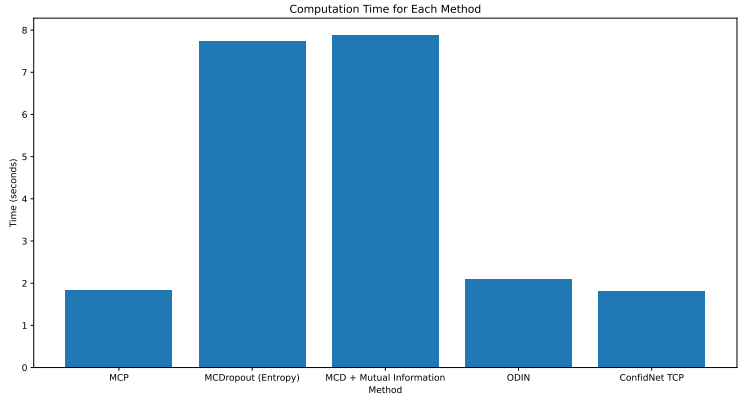


Figure 3.7: Computation time for each OOD method

- **MCP (Maximum Class Probability):** Shows an AUPR of 94.83%, indicating a solid baseline performance. However, it's slightly lower performance compared to other metrics is understandable

given its lack of direct uncertainty consideration.

- **MCDropout (Entropy):** With an AUPR of 98.53%, this metric reflects the model's ability to effectively estimate uncertainty through entropy. The dropout mechanism introduces randomness that entropy captures well, aiding in efficient distinction between in-distribution and OOD samples.
- **MCD + Mutual Information:** Reports a slightly higher AUPR at 98.57%. This approach considers both predictive and data uncertainty, providing a slight edge over entropy alone. The added consideration of the information gained about model parameters upon observing data contributes to more refined OOD detection.
- **ODIN:** Achieves an AUPR of 95.75%, a commendable performance considering it enhances MCP. Temperature scaling and adversarial perturbation improve OOD sensitivity, but not as effectively as methods leveraging dropout's stochastic nature.
- **ConfidNet TCP (True Class Probability):** Scores an AUPR of 97.67%, suggesting the effectiveness of learning to predict the true class probability directly from data. However, it slightly underperforms compared to methods utilizing MC Dropout, likely due to its focus on individual predictions rather than overall network uncertainty.

The two top-performing methods—MCD + Mutual Information and MCDropout (Entropy)—demonstrate the effective use of Monte Carlo Dropout for OOD detection. These methods achieve the highest AUPR by assessing predictions and considering the underlying uncertainty in both the data and the model. This comprehensive approach to uncertainty quantification is especially beneficial for OOD detection.

Additionally, there is a significant variance in the computational cost for OOD detection across methods, with more complex approaches (MCDropout and MCD + Mutual Information) taking 4 times as long as ODIN (2 seconds).

In conclusion, while ODIN offers a reasonable balance between performance and computational speed, the combination of MC Dropout with predictive entropy or mutual information yields the best results in detecting OOD samples in this experiment. However, this slight performance advantage incurs increased computational time, which may be a constraint in real-time applications.

3.3.2 ⋆ Extra: How do ϵ and *temperature* affect ODIN's performance?

In the pursuit of optimizing out-of-distribution (OOD) detection capabilities, we scrutinize the influence of two pivotal hyperparameters in the ODIN method: the perturbation magnitude ϵ and temperature scaling T. Their precise calibration is crucial since they govern the model's sensitivity to distributional shifts, directly impacting the reliability of uncertainty estimates.

To this end, we employ a systematic grid search approach, exhaustively exploring a range of values for both ϵ and T. The resultant heatmap encapsulates the performance of each parameter pairing, offering a visual narrative of their interdependent effects on OOD detection as quantified by the Area Under the Precision-Recall (AUPR) metric. This meticulous analysis not only aims to identify the most effective combination but also to shed light on the underlying mechanics of ODIN's response to varied degrees of input perturbation and softmax temperature adjustments. The insights garnered here are instrumental in refining ODIN for robust OOD detection in diverse operational environments.

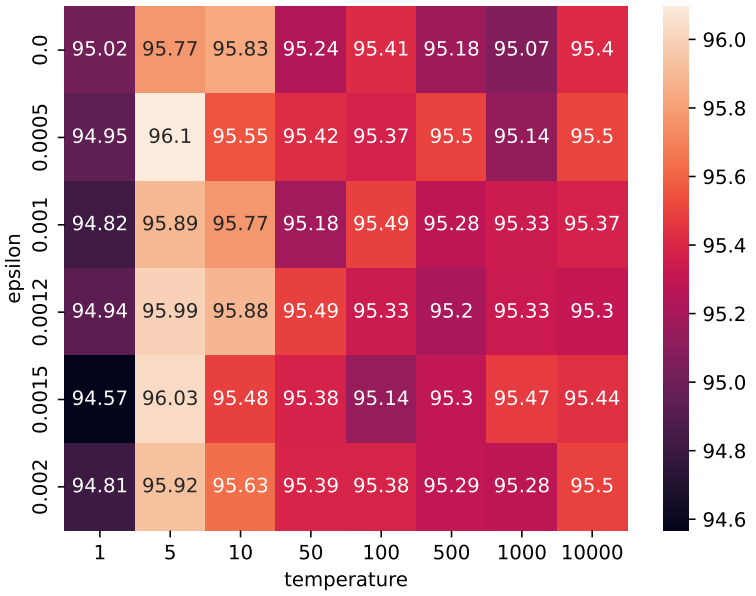


Figure 3.8: AUPR for different combinations of ϵ and T (*temperature*)

Observations from the Heatmap:

1. There is a trend of increasing AUPR with rising temperature values up to a point, beyond which AUPR plateaus or decreases.

2. Epsilon shows a nuanced effect on AUPR, suggesting that perturbation impact is secondary to

temperature scaling within the explored range.

3. Highest AUPR value is observed at the temperature of 5 and epsilon = 0.0005 (96.1), indicating an optimal balance for ODIN's OOD detection.

4. AUPR values remain relatively consistent across different epsilons for a fixed temperature, especially at higher temperatures, highlighting temperature scaling's dominant role.

Conclusion:

The heatmap encapsulates the performance landscape of the ODIN method for OOD detection, highlighting an optimal hyperparameter region. Temperature scaling emerges as a more influential factor compared to adversarial perturbation in maximizing AUPR for OOD detection.