Uncertainty Quantification: SOA

Dimitri MARCHAND

September 12, 2023

Contents

Chapter 1

Introduction

Uncertainty in the context of machine learning refers to the lack of complete knowledge about the true underlying data distribution or model parameters. In practical scenarios, data is often noisy, limited, or contains biases, leading to uncertainties in the predictions generated by ML models. Ignoring or overlooking these uncertainties can have severe consequences, especially in safety-critical applications such as autonomous driving or medical diagnosis.

1.1 what is uncertainty

Uncertainty is usually defined as *the lack of certainty, a state of limited knowledge where it is impossible to exactly describe the existing state, a future outcome, or more than one possible outcome.*

In the context of machine learning, uncertainty refers to the lack of confidence in one's estimates. It is common to separate uncertainty into two categories: aleatoric uncertainty (inherent uncertainty of the system) and epistemic uncertainty (uncertainty about the choice of model). These two classes of uncertainty are described more below.

Uncertainty is an important consideration in many machine learning applications, and in particular, it may be crucial to quantify the amount of uncertainty for any given prediction (i.e. the predictive uncertainty). When high stakes decisions are being made based on predictions of machine learning models, e.g. in health care, it is vital to know how much confidence to have in the prediction. Alternatively, for some sequential decision making tasks high uncertainty may correspond with potentially valuable decisions that should be tested.

1.1.1 Aleatoric uncertainty

Aleatoric uncertainty is also known as statistical uncertainty, and is representative of unknowns that differ each time we run the same experiment. For example, a single arrow shot with a mechanical bow that exactly duplicates each launch (the same acceleration, altitude, direction and final velocity) will not all impact the same point on the target due to random and complicated vibrations of the arrow shaft, the knowledge of which cannot be determined sufficiently to eliminate the resulting scatter of impact points.

In other words, this is the uncertainty that is inherent to the system because of information that cannot be measured (i.e. noise). When this noise is present, aleatoric uncertainty cannot be eliminated even as the number of samples collected tends towards infinity.

Examples of aleatoric uncertainty can be found in physical settings where measurement error may exist (e.g. given the same true temperature, a thermometer may output slightly different values), or where inherent noise exist in the system (e.g. a random roll of a die). Taking more temperature measurements will not reduce the uncertainty stemming from an imprecise thermometer, and likewise, observing more rolls of a die will not help us better guess the outcome of a roll, assuming the die is fair.

1.1.2 Epistemic uncertainty

Epistemic uncertainty is also known as systematic uncertainty, and is due to things one could in principle know but do not in practice. It is a type of uncertainty that arises from limitations in knowledge and information rather than inherent randomness or variability in a system. Epistemic uncertainty is rooted in the idea that there are aspects of a phenomenon or system that could, in theory, be fully understood or known, but due to practical constraints or limitations, remain uncertain.

1.2 how to communicate the quality of UQ prediction

In this part we present some metrics that can be used first to communicate, then to benchmark. Those methods are at first concepts, and then several metrics can be chosen for the same concept. An in-depth study Psaros et al. 2023 of uq metrics can be found and divides metrics in three groups:

- valuation metrics that require a test set
- metrics that require a "gold standard", e.g., for testing novel techniques
- metrics that require neither a test set nor a gold standard

1.2.1 Calibration

the idea Broadly speaking, calibration in the regression setting requires that the probability of observing the target random variable (p^{obs}) below a predicted p^{th} quantile is equal to the expected probability p, for all $p \in [0, 1]$.:

$$p^{obs}(p) = p, \forall p \in [0, 1] \tag{1.1}$$

From this generic statement, we can describe different notions of calibration based on how p^{obs} is defined.

The most common calibration (used in scikit-learn for example) is **average calibration**, defined as the probability of observing the target below the quantile prediction, averaged over F_X(cumulative distribution over the feature space) which must be equal to p:

other calibration are: *group calibration*, *adversarial calibration* The calibration metrics are further detailed in Zhao, Ma, and Ermon 2020.

a typical plot

a typical metric the *Expected Calibration Error* is often used:

1.2.2 Sharpness

1.3 how is uncertainty measured

the following metrics were found. Zhang et al. 2021 made a classification based on the use of test set and/or gold standard.

1.3.1 test set required, no gold standard

calibration Confidence calibration is "the problem of predicting probability estimates representative of the true correctness likelihood." **guo2017**.

Intuitively, calibration refers to the degree to which a predicted uncertainty matches the true underlying uncertainty in the data. For example, suppose you make a series of 10 predictions about the winning odds of a horse race, and for each race you predict that a certain horse will win with 70% chance. If you called the correct winning horse in roughly 7 of the 10 races (70% of the races), then your uncertainty predictions are said to be calibrated. The above example is an example of calibrated classification with binary outcomes (win or lose).

We can further consider calibration in the regression setting. If you make predictions about the amount of rainfall each day for a whole month, and for each day, you make a predictive statement, "the amount of rainfall today will not be more than x inches, with 50% chance", and if your prediction was correct for roughly 15 out of 30 days (50% of the days), then your predictions are said to be calibrated.

several variants can be found:

average calibration plot this is a plot where

RMS calibration error the equation is:

$$RMSCE = \sqrt{\mathbb{E}_{p\in[0,1]}[p - \hat{p}(p)]^2} \tag{1.2}$$

where

adversarial group calibration

1.3.2 no test set nor gold standard required

1.3.3 gold standard required, no test set

1.3.4 to sort

Chapter 2

Methods in detail

2.1 Probabilistic methods

Probabilistic methods form the bedrock of uncertainty quantification, utilizing statistical and probabilistic principles to comprehend and quantify uncertainty in various contexts. In this group, we delve into Bayesian methods, including Bayesian inference, updating, and model averaging, which empower us to merge prior knowledge with observed data for accurate uncertainty estimation. Additionally, we explore Monte Carlo methods, encompassing Markov Chain Monte Carlo (MCMC), importance sampling, and sequential Monte Carlo (SMC), which furnish us with invaluable tools for sampling from intricate distributions and estimating integrals, thereby enabling the comprehensive quantification of uncertainties.

2.1.1 Monte-Carlo methods

Monte Carlo methods are stochastic simulation techniques widely used for uncertainty quantification in complex systems. These methods leverage random sampling to estimate statistical properties of models that involve uncertain inputs and complex relationships. In uncertainty quantification, the goal is to assess the impact of uncertainties on model predictions, making Monte Carlo methods an indispensable tool for decision-making and risk assessment.

Main Idea

Monte Carlo methods approximate complex systems by generating random samples from input parameter distributions and propagating them through the model. The uncertainty is computed by analyzing the statistical distribution of the model outputs resulting from these samples. As the number of samples increases, the approximation becomes more accurate, allowing for the quantification of uncertainty.

Computing Uncertainty

Uncertainty in Monte Carlo methods is computed by generating a large number of samples from the input parameter distributions. These samples are used to produce a distribution of model outputs. Various statistical measures such as mean, variance, quantiles, and confidence intervals can be computed from this output distribution to quantify uncertainty.

Characteristics

White and Black Box Monte Carlo methods can work with both white-box (analytical) and black-box (simulated or real-world) models, making them versatile for various applications.

Inline and Offline Monte Carlo simulations can be performed in both inline and offline modes. Inline simulations involve generating samples on-the-fly during model execution, while offline simulations precompute a set of samples before running the model.

Strengths Monte Carlo methods handle complex models, propagate uncertainties through intricate relationships, and provide rich insights into the behavior of uncertain systems.

Weaknesses High computational cost, especially for high-dimensional problems, and convergence issues can limit the practicality of Monte Carlo methods.

Variants and Recent Papers

Variants

Markov Chain Monte Carlo (MCMC) An advanced variant that efficiently samples complex distributions by constructing a Markov chain.

Sequential Monte Carlo (SMC) Used for filtering and smoothing problems in state-space models.

Importance Sampling A technique to improve Monte Carlo efficiency by sampling more frequently from regions that contribute significantly to the output distribution.

Libraries and Repositories

PyMC3: A Python library for Bayesian modeling and probabilistic machine learning using Monte Carlo methods. Stan: A probabilistic programming language for Bayesian inference, using MCMC techniques. Emcee: A Python library for MCMC sampling, especially suited for complex parameter spaces. UQpy: A general-purpose Python library for uncertainty quantification, providing various sampling and propagation techniques.

2.1.2 Sampling/Dropout-based Methods

2.1.3 Presentation

main idea

Sampling/Dropout-based Methods are a class of techniques used in machine learning for estimating uncertainty in deep learning models. The main idea is to leverage stochasticity during model training and inference to capture uncertainty information. This is achieved by introducing randomness in the form of dropout layers or using Monte Carlo sampling during the forward pass. These techniques enable the model to estimate uncertainty in predictions, which is essential for making reliable decisions, especially in critical applications such as medical diagnosis, autonomous vehicles, and financial risk analysis.

uncertainty computation

Model Training with Dropout During training, dropout layers are introduced in the neural network architecture. Dropout randomly deactivates neurons with a certain probability, creating an ensemble of diverse models within a single model. This encourages robustness and aids in uncertainty estimation.

Monte Carlo Dropout During model inference, Monte Carlo sampling is applied by repeatedly running the forward pass with dropout enabled. This process generates multiple stochastic predictions for each input, resulting in a probability distribution over the model's output.

Uncertainty Estimation The uncertainty can be quantified using various metrics, such as variance, entropy, or predictive entropy. Variance measures the dispersion of predictions, while entropy indicates the model's confidence or lack thereof in its predictions.

loss metrics

The main loss functions used in Sampling/Dropout-based Methods are generally the same as those in traditional deep learning models: MSE, cross-entropy, NLL

Charateristics

white box model

Sampling/Dropout-based Methods can be considered white box models as they retain interpretability to some extent. While dropout introduces stochasticity, the underlying neural network architecture remains transparent, allowing for analysis and understanding of the model's behavior.

offline model

Sampling/Dropout-based Methods are typically offline models since they require multiple forward passes during inference to obtain uncertainty estimates. This can be computationally expensive compared to inline methods, but it offers more accurate uncertainty quantification.

Weaknesses

Increased Computation The repeated forward passes during inference can be computationally expensive, slowing down the prediction process.

Limited Uncertainty Modeling Dropout-based methods may struggle to capture more complex types of uncertainty, such as epistemic and aleatoric uncertainty, as effectively as more advanced methods

Strengths

Simplicity Implementing dropout-based uncertainty estimation is relatively straightforward and can be applied to various neural network architectures.

Improved Generalization Dropout during training acts as a regularization technique, leading to improved generalization on unseen data.

Quantifiable Uncertainty Provides a measure of uncertainty for each prediction, enabling better decision-making in critical applications.

variants and recent papers

Dropout with Bayesian Approximation Introduces Bayesian principles to dropout-based methods for better uncertainty modeling.

Concrete Dropout: Improves dropout-based uncertainty estimation using a concrete distribution to approximate dropout probabilities.

What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? Maddox and Hartley 2020

Variational Dropout Sparsifies Deep Neural Networks Molchanov et al. 2017

Libraries and Repositories

2.2 Model-based methods

Model-based techniques are centered around constructing and refining mathematical models to better comprehend and capture uncertainties:

- **sensitivity analysis** offers insights into the impact of input uncertainties on model outputs, paving the way for informed decision-making
- **Surrogate modeling** including Kriging and response surface methods, facilitates approximation of complex simulations by simpler functions, serving as a powerful tool for uncertainty assessment
- **polynomial chaos expansion**
- **model discrepancy calibration**

Surrogate models and approximations offer efficient strategies for representing complex systems while accounting for uncertainties. Non-intrusive methods like polynomial chaos expansion and Gaussian process regression provide means to approximate functions with uncertainty quantification, yielding reliable predictive distributions. Additionally, radial basis function networks and interval analysis present alternate avenues to capture uncertainties within models, allowing for a comprehensive exploration of the potential range of outcomes.

2.2.1 Calibration method

Main idea

The main idea behind calibration is to ensure that the predicted uncertainties match the empirical probabilities of outcomes. A well-calibrated model should provide predictions such that, for example, if it predicts a 70% chance of rain, it should rain approximately 70% of the time in those instances.

Computation of uncertainty

Uncertainty is computed in calibration methods by comparing the predicted probability scores with the observed outcomes. Commonly used metrics include reliability diagrams, probability integral transforms, and expected calibration error (ECE). These metrics measure the alignment of predicted probabilities with actual event frequencies.

Characteristics

White box and black box Calibration methods can be applied to both white-box and black-box models. White-box models, such as linear regression, allow for a more interpretable calibration process. However, black-box models, like deep neural networks, can also benefit from calibration techniques.

Online and Inline Calibration can be performed both inline (during model training) and offline (after model training). Inline calibration methods aim to optimize model calibration directly during the training process, while offline methods adjust the model's predictions post-training.

Strengths

- Improves the reliability of uncertainty estimates.
- Can be applied to various machine learning models.
- Enhances model interpretability.
- Reduces the risk of overconfidence.

Variants and recent papers

Calibration methods have seen significant development in recent years. Variants include temperature scaling, Platt scaling, and beta calibration. Recent research explores Bayesian calibration techniques, ensemble-based calibration, and adversarial calibration methods.

Libraries and repositories

2.2.2 Sensitivity analysis

Main idea

Sensitivity analysis is a systematic approach to assess the influence of input parameter variations on the output of a model, thereby quantifying the impact of uncertainties on predictions. By understanding the sensitivity of model outputs to different inputs, decision-makers can make more informed choices and manage risks effectively.

Computation of uncertainty

Uncertainty in sensitivity analysis is often computed through variance-based methods, such as Sobol indices, which decompose the total variance of the output into contributions from individual input parameters and their interactions. Sobol indices provide insights into the relative importance of each parameter, both individually and in combination, contributing to the overall uncertainty.

Characteristics

White and Black Box Model Sensitivity analysis can be applied to both white box (transparent) and black box (opaque) models. White box models provide insights into the relationships between inputs and outputs, while black box models focus on input-output behavior without revealing internal mechanisms.

Inline and Offline Model: Sensitivity analysis can be performed either inline during model execution or offline after data collection. Inline methods facilitate real-time analysis, while offline methods are more suitable for post-processing and complex models.

Strengths Strengths and Weaknesses: Sensitivity analysis offers a structured approach to uncertainty quantification. It aids in identifying critical input parameters and interactions, enhancing decision-making.

Weaknesses However, it might oversimplify complex systems and fail to capture nonlinear relationships.

Variants and Recent Papers

some variants

Global Sensitivity Analysis Extends sensitivity analysis to consider interactions across the entire parameter space.

Local Sensitivity Analysis Focuses on sensitivity at specific points in the parameter space.

Time-dependent Sensitivity Analysis Analyzes how sensitivities evolve over time.

Libraries and Repositories

Librairies

SALib A Python library for sensitivity analysis, offering various methods for both global and local sensitivity analysis.

UQLab A versatile uncertainty quantification toolbox with built-in sensitivity analysis functionalities.

2.2.3 Surrogate modeling

Main idea

Surrogate Modeling, also known as metamodeling or response surface modeling, involves constructing an approximation (surrogate) of a complex and often computationally expensive simulation or evaluation process. The primary goal is to reduce the computational burden associated with direct evaluations while maintaining acceptable accuracy. This technique is particularly useful in scenarios where repeated evaluations of the target process are required, such as optimization, sensitivity analysis, and uncertainty quantification.

Uncertainty computation

Uncertainty quantification in Surrogate Modeling is achieved by integrating probabilistic approaches within the surrogate model framework. Techniques like Bayesian inference and Gaussian processes allow for quantifying uncertainty by estimating the posterior distribution over the surrogate model parameters. This distribution captures the inherent uncertainty due to limited data and provides a basis for constructing predictive intervals and uncertainty bounds.

Characteristics

White or Black Box Model Surrogate Modeling can encompass both white-box models (e.g., polynomial regression) and black-box models (e.g., Gaussian processes). The choice depends on the interpretability of the surrogate and the available information about the underlying target process.

Inline or Offline Model Surrogate models can be used either inline (within the optimization loop) or offline (precomputed). Inline models provide real-time predictions but require computational resources, while offline models reduce the computational load during optimization but may lack flexibility.

Strengths Surrogate Modeling reduces computational cost, accelerates optimization, provides insight into the underlying process, and quantifies uncertainty in predictions.

Weaknesses The surrogate model's accuracy is influenced by the choice of model type, parameterization, and data quality. The quantified uncertainty may not always accurately capture complex dependencies.

Variants and Recent Papers

Variants

- **Kriging** Gaussian processes with spatial correlation
- **Radial Basis Functions**
- **Neural Network Surrogates**
- Polynomial Regression

2.2.4 polynomial chaos expansion

Main idea

Polynomial Chaos Expansion aims to represent uncertain quantities as polynomial functions of random variables. The main idea is to construct an orthogonal polynomial basis tailored to the distribution of the uncertain parameters. By projecting the uncertain function onto this basis, the uncertainty quantification problem is transformed into solving deterministic polynomial equations.

Computating the uncertainty

The uncertainty associated with a quantity of interest is computed by approximating its probability distribution using the coefficients obtained from the orthogonal polynomial projection. The higher the order of the polynomial expansion, the more accurately the distribution is approximated.

some loss functions used

Loss functions in PCE methods are used to quantify the discrepancy between the true behavior of the system and the polynomial approximation. Mean squared error, Kullback-Leibler divergence, and moment matching are some common loss functions used to ensure accurate representation of the uncertain quantities.

characteristics

white bo model PCE methods are typically considered white box models as they provide insight into the effect of input uncertainties on the system's output.

offline PCE methods are primarily offline models, requiring the computation of coefficients beforehand. However, they can be used inline by employing surrogate models.

Strengths PCE offers high accuracy in capturing complex relationships between inputs and outputs, handles both aleatory and epistemic uncertainties, and is versatile across various domains.

Weaknesses PCE can become computationally expensive for high-dimensional problems, requires careful selection of polynomial basis and truncation levels, and struggles with capturing sharp changes or discontinuities.

Variants and recent papers Several variants of PCE have emerged, including Sparse PCE, Multi-element PCE, and Stochastic Galerkin methods. Recent papers explore advancements in adaptive basis selection, efficient computation techniques, and hybrid approaches integrating PCE with other UQ methods.

Libraries and repos

UQpy A Python library for uncertainty quantification that supports various methods including PCE.

chaospy A Python library specifically focused on Polynomial Chaos Expansion.

OpenTurns An open-source UQ library that includes PCE methods.

Quadpy A library for numerical integration that can be useful in PCE computation.

2.2.5 Quantile-Regression/Predicted-Intervals

Main idea

Quantile Regression/Predicted Intervals focus on estimating different quantiles of the target variable, providing a richer understanding of the underlying uncertainty. Unlike traditional point estimates, quantile regression allows us to capture the entire distribution of possible outcomes.

Computation of uncertainty

Uncertainty in Quantile Regression/Predicted Intervals is computed by estimating a range of quantiles of the target variable. This is achieved by optimizing a loss function that penalizes the model for underestimating or overestimating specific quantiles. The result is a set of prediction intervals, each corresponding to a desired quantile level.

Loss function used

in particular, tquantile regression uses **the pinball loss**, which measures the weighted absolute deviation between predicted and observed quantiles. It encourages the model to produce accurate quantile estimates.

Characteristics

White box and black box model Some implementations, like linear quantile regression, offer transparency, making them white-box models.

Offline model Quantile Regression/Predicted Intervals models are typically trained offline, but they can be adapted for online learning scenarios. The choice depends on the application and the frequency of updates required.

Strengths

- **Rich Uncertainty Representation**: Provides a comprehensive view of uncertainty by estimating multiple quantiles.
- **Robust to Outliers**: Resistant to extreme values due to its quantile-based nature.
- **Interpretability**: Transparent models like linear quantile regression offer insight into relationships between variables.

Weaknesses

- **Computational Complexity**: Training models for multiple quantiles can be computationally intensive.
- **Data Requirements**: Requires a significant amount of data, especially for extreme quantiles.
- **Interpolation Limitation**: May struggle to provide accurate predictions between quantiles.

2.3 Machine learning and neural network methods

This group uses machine learning and neural networks, showcasing an array of techniques designed to bolster uncertainty quantification in data-driven contexts.

Within the framework of machine learning, we explore ensemble methods, variational autoencoders (VAEs), and conformal predictions, which contribute to heightened prediction accuracy and augmented confidence by harnessing the synergy of multiple models, latent variable modeling, and well-calibrated uncertainty intervals. Furthermore, Dirichlet networks and evidential deep learning offer innovative avenues to embrace the inherent uncertainties of neural network predictions. By bridging the gap between conventional deep learning and uncertainty estimation, these methods yield a comprehensive toolkit for enhancing both predictive power and uncertainty awareness.

2.3.1 Ensemble Methods

main idea

Ensemble methods are a powerful approach in machine learning that combine multiple models to make more accurate predictions and quantify uncertainty. This document explores the main idea of ensemble methods, how uncertainty is computed, the main loss functions, characteristics, strengths, and weaknesses. Additionally, it provides a practical example, recent variants, and relevant libraries for implementing and studying ensemble methods for uncertainty quantification.

Computation of Uncertainty

Ensemble methods utilize multiple models, often referred to as base models or learners, to make predictions. The idea is to combine these models to achieve better generalization and robustness. Uncertainty quantification in ensemble methods can be achieved through various techniques such as:

Bootstrap Aggregating (Bagging) Bagging involves training multiple models on different bootstrap samples of the training data. Uncertainty is computed by aggregating the predictions of these models, such as variance, confidence intervals, or entropy.

Boosting Boosting builds an ensemble by sequentially training models that correct the mistakes of their predecessors. Uncertainty can be measured by considering the weights of the individual models or their prediction distributions.

Bayesian Model Averaging (BMA) BMA utilizes Bayesian methods to estimate model parameters and combines predictions using model weights as probabilities. Uncertainty is obtained through Bayesian inference.

Characteristics

Black box model Ensemble methods can be considered black box models, as the predictions are based on complex combinations of individual models' outputs. However, some ensembles, like BMA, can be seen as white box models, offering interpretable uncertainty estimates.s

Inline and offline model Ensemble methods can be used both inline during the training process to improve model performance or offline for uncertainty quantification on pre-trained models.

Strengths

Improved Prediction Accuracy Ensemble methods often outperform single models, especially when trained on diverse datasets or using different algorithms.

Uncertainty Quantification Ensemble methods offer valuable uncertainty estimates, which is crucial in safety-critical applications.

Robustness Ensembles are more resistant to overfitting and noise, making them suitable for noisy or sparse data.

Weaknesses

Increased Computational Complexity Training and maintaining multiple models require more computational resources.

Interpretability As ensemble methods combine various models, the interpretability of individual predictions can be challenging.

Variants and recent paper

Deep ensembles were among first techniques to be implemented Fort and Hu 2021, Roijers and Van De Kamp 2022, Yoder and Bilmes 2020.

libraries and repositories

There are several libraries and repositories that speed up the implementation of Ensemble-Methods and allow studying use cases:

TensorFlow Probability This library by Google provides tools for Bayesian deep learning, enabling uncertainty estimation using Bayesian approaches.

PyMC3 A Python library for probabilistic programming that can be used to implement Bayesian Ensemble-Methods.

2.3.2 Conformal prediction

Main idea

Unlike traditional methods that produce point predictions, CP provides prediction sets with associated confidence levels. The main idea behind CP is to create prediction regions such that, with a given confidence level, they will contain the true target value for future data points. A worth noting tutorial can be found at Linusson 2017

Computing the uncertainty

The computation of uncertainty in CP involves two fundamental steps:

Training Stage Given a labeled dataset, a base machine learning algorithm (e.g., random forests, support vector machines, neural networks) is trained to make point predictions. For each data point, the base model's prediction is recorded, forming the nonconformity scores. The nonconformity measures the model's deviation from making an accurate prediction.

Conformal Inference Stage The nonconformity scores from the training stage are utilized to construct prediction regions for new, unseen data points in the test set. These prediction regions are formed by selecting a predefined number of the most similar training examples based on their nonconformity scores. The width of the prediction region i

Loss metrics used

Conformal Prediction employs loss functions that evaluate the quality of prediction regions. The main loss functions include:

Inductive Loss Measures the proportion of training instances outside their respective prediction regions.

Transductive Loss Evaluates the fraction of instances in the test set that fall within their prediction regions.

Characteristics

white box model Conformal Prediction is a model-agnostic technique, making it a white box model in the sense that it can work with any base predictor. It does not rely on the internal workings of the base model and is transparent in its approach to uncertainty quantification.

offline model Conformal Prediction can be categorized as an offline model, as it requires a calibration stage that uses a separate calibration dataset. During calibration, the model learns the relationship between the model's output and the actual prediction errors, which allows it to produce reliable confidence levels for the prediction intervals. Once the calibration is completed, the model can make predictions with uncertainty for new data points without requiring further access to the calibration dataset.

Strengths

Validity and reliability Conformal prediction provides valid confidence measures, ensuring that the true target falls within the prediction regions with a given probability.

model agnostic Conformal prediction can be applied to any base model, allowing flexibility in the choice of algorithms for different tasks.

Calibration The significance level ϵ allows practitioners to control the trade-off between prediction accuracy and confidence.

Incremental Learning Conformal Prediction supports incremental and online learning setups.

Weaknesses It can be computationally intensive, and its performance might degrade with high-dimensional or non-stationary data.

Variants and recent papers

Variants Over the years, various variants of Conformal Prediction have been proposed, each with its own specific advantages and use cases. Some of the notable variants and recent papers include:

Nonconformist-based Conformal Prediction The paper Ek 2010 introduces the Nonconformist framework, which extends Conformal Prediction to handle multiple conformity measures, allowing users to choose the most suitable one for their specific problem.

Transductive Conformal Predictors: Addressing Accuracy-Adaptivity Dilemma The paper Gammerman et al. 2013 addresses the trade-off between accuracy and adaptivity in Conformal Prediction by introducing the concept of transductive conformal predictors, which adapt better to new data points.

Inductive Conformal Prediction The paper Vovk, Gammerman, and Hähnel 2021 explores inductive conformal prediction for neural networks, enabling uncertainty estimation in deep learning models.

Libraries and Repositories

To speed up implementation or study use cases of Conformal Prediction, several libraries and repositories are available:

conformal A Python library for Conformal Prediction available at `https://github.com/donlnz/conformal`. This library provides various Conformal Prediction algorithms and utilities, making it easy to apply CP to different machine learning models.

pyCP Another Python library for Conformal Prediction accessible at `https://github.com/donlnz/pycp`. It provides an implementation of several Conformal Prediction algorithms, along with visualization tools and examples to aid in understanding and using the technique effectively.

conformal inference A repository on GitHub at `https://github.com/donlnz/conformal_inference`, offering implementations of conformal inference algorithms, allowing researchers and practitioners to explore and experiment with different variants of Conformal Prediction.

2.3.3 Dirichlet-networks/Evidential-deep-learning

Main idea

The core idea behind Dirichlet Networks is to treat prediction outputs as Dirichlet distributions over classes instead of point estimates. This allows the network to capture not only the most likely class but also the uncertainty associated with each class. The Dirichlet distribution parameters are learned during training, reflecting the network's confidence in its predictions.

Computing uncertainty

Uncertainty is computed in Dirichlet Networks using the parameters of the Dirichlet distribution associated with each class. The parameters include concentration parameters, which control the spread of the distribution, and a normalization constant. Higher concentration parameters indicate lower uncertainty, while lower values indicate higher uncertainty. The entropy of the distribution serves as a measure of uncertainty for a particular prediction.

Loss used

The main loss function used in Dirichlet Networks is the evidence lower bound (ELBO) loss. It combines the negative log-likelihood of the observed data and the Kullback-Leibler (KL) divergence between the predicted Dirichlet distribution and a prior distribution. This encourages the network to produce well-calibrated predictions with accurate uncertainty estimates.

Characteristics

White box Dirichlet Networks can be considered more of a white-box model, as the uncertainty quantification process is explicitly built into the architecture. This transparency aids in understanding and validating uncertainty estimates.

Inline model Dirichlet Networks are typically inline models, meaning they provide uncertainty estimates alongside predictions in real-time. This makes them suitable for applications where uncertainty-aware decision-making is crucial.

Strengths

Accurate Uncertainty Estimation Dirichlet Networks offer improved uncertainty estimates, making them useful for safety-critical applications.

Transparency The incorporation of uncertainty estimation into the architecture enhances model interpretability.

Flexible DNs can handle both aleatoric and epistemic uncertainties, providing a more comprehensive uncertainty quantification.

Weaknesses

Computational Complexity The Dirichlet distribution adds complexity to the model, which might lead to increased computational requirements during training and inference.

Data Dependency Like other deep learning models, DNs heavily rely on data quality and quantity.

Variants and recent papers

Recent papers

Libraries and repositories

2.3.4 Auxiliary methods / learning Loss distributions

main idea

The main idea of Auxiliary Methods/Learning Loss Distributions revolves around incorporating auxiliary tasks or learning loss distributions to enhance the estimation of model uncertainty. Traditional approaches to uncertainty quantification often focus on measuring epistemic and aleatoric uncertainties separately. However, the use of auxiliary tasks or learning loss distributions enables capturing complex dependencies between input data and model predictions, leading to more accurate and robust uncertainty estimates. This section is mostly about the article **Post-hoc Uncertainty Learning using a Dirichlet Meta-Model**Shen et al. 2022, whose idea is to propose a novel Bayesian meta-model that can augment pre-trained models with better uncertainty quantification abilities. The Dirichlet meta-model is used to capture different uncertainties, including total uncertainty and epistemic uncertainty.

Computation of Uncertainty

In Auxiliary Methods/Learning Loss Distributions, uncertainty is computed by leveraging the additional information provided by auxiliary tasks or learning loss distributions. These auxiliary tasks are designed to capture different aspects of the underlying data distribution and can be integrated into the model architecture in various ways, such as multi-task learning or utilizing auxiliary loss terms during training.

By training the model jointly on the main task and auxiliary tasks, the model learns to extract meaningful features from the data, leading to better uncertainty quantification. The uncertainty estimates can be obtained using various methods, including Bayesian inference, Monte Carlo sampling, or ensembling techniques.

The uncertainty is computed by training the meta-model on the output probabilities of the pre-trained model. The meta-model learns to capture the uncertainty in the pre-trained model's predictions and provides a more accurate measure of uncertainty. The method is computationally efficient and requires no additional training data.

Characteristics

White and Black Box model AM/LLD methods can be both white and black box models, as they can be integrated with various machine learning architectures.

Inline and Offline Model These methods can be used both inline, during model training, and offline, during prediction or post-training analysis.

Weaknesses AM/LLD methods may require careful design of auxiliary tasks and balancing their influence on the overall model training. Selecting appropriate auxiliary tasks can be challenging.

Strengths These methods offer a principled approach to capturing various sources of uncertainty and improving model confidence estimation.

Variants end recent papers

Several variants of AM/LLD methods have been proposed, including variations in how auxiliary tasks are chosen, loss distribution modeling, and incorporation of external uncertainty estimates.

Characteristics

Black box model The proposed method is a black-box model that requires no additional data other than the training dataset.

Offline model It can be trained separately from the pre-trained model.

Strengths

- flexibility
- computational efficiency
- ability to capture different sources of uncertainty

Weaknesses

- Data intensive

Variants and recent papers

: Several variants of this method have been proposed, including WhiteboxChen et al. 2019 and LULAKristiadi, Hein, and Hennig 2021. Whitebox requires an additional validation set to train the meta-model, while LULA needs an additional OOD dataset during training to distinguish the in-distribution samples and outliers.

Recent papers in this field include "Deep Ensembles: A Loss Landscape Perspective"Fort and Hu 2021 by Wilson et al. and "Uncertainty Quantification in Deep Learning with Application to Autonomous Systems" by Kottas et al.

Libraries and repos

2.4 Data-driven approaches

Incorporating the rich potential of available data, this group explores data-driven strategies to estimate uncertainties and refine predictive models. By augmenting training data through data augmentation and generation methods, models can gain resilience to uncertainties. Resampling techniques like bootstrapping, cross-validation, leave-one-out cross-validation, and jackknife resampling provide insights into model performance and predictive variability, enabling robust uncertainty assessment and decision-making in the face of limited data.

2.4.1 Data augmentation/generation-based methods

Main idea

The main idea behind data augmentation and generation-based methods is to enhance the training dataset by creating new samples that are plausible within the data distribution. This approach introduces diversity into the training set, allowing the model to learn more robust and accurate representations of the underlying data distribution. These methods can be broadly categorized as follows:

Data Augmentation In data augmentation, existing data samples are perturbed through various transformations, such as rotation, translation, scaling, and noise addition. These perturbed samples are then included in the training set, expanding its diversity.

Generation-Based Methods Generation-based methods involve the creation of new synthetic data samples using generative models, such as variational autoencoders (VAEs) or generative adversarial networks (GANs). These synthetic samples, resembling the original data distribution, contribute to a more comprehensive dataset for model training.

Computation of uncertainty

Uncertainty computation in data augmentation and generation-based methods is typically achieved through the use of probabilistic models. Bayesian neural networks (BNNs) are often employed, where the weights of the model are treated as random variables, allowing for uncertainty quantification in predictions. Monte Carlo dropout and ensembling techniques are used to obtain multiple predictions from the model, which are then used to compute various uncertainty metrics, such as epistemic and aleatoric uncertainties.

Characteristics

White or Black Box Model The choice between a white or black box model depends on the underlying generative model. VAEs tend to be more transparent (white box), while GANs might be less interpretable (black box).

Inline or Offline Model These methods are typically inline models, meaning that they are integrated into the model architecture during training and inference. However, they can also be used offline to generate additional training data before training.

Weaknesses Weaknesses include potential overfitting to augmented/generated data and increased computational requirements.

Strengths Strengths encompass improved uncertainty estimates, enhanced generalization, and better model performance.

Variants and recent papers

Libraries and repos

Chapter 3

Frameworks

3.1 MAPIE

3.2 uncertainty toolbox

3.3 OpenOOD: Benchmarking Generalized OOD Detection

3.4 Mixture Density Networks (MDN) for distribution and uncertainty estimation

3.5 Bayesian Torch

3.6 Blitz: A Bayesian Neural Network library for PyTorch

3.7 Fortuna

3.8 TensorFlow Probability

UQ