Uncertainty Quantification in Machine Learning

Zhiming Xu
zx2rw@virginia.edu

School of Engineering and Applied Science
University of Virginia

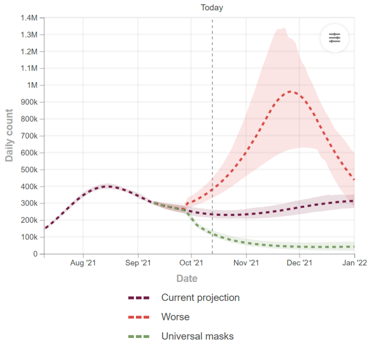
October 22, 2021

Table of Contents

Where Uncertainties Arise

Uncertainties can be observed in many real-world scenarios, such as investment returns, endemic cases, and etc.

The following figure shows an example of uncertainties in COVID case prediction in the U.S., the shaded areas are the 95% uncertainly interval[1].



[1]https://covid19.healthdata.org/united-states-of-america

Types of Uncertainties

By modeling such processes with machine learning techniques, two types of uncertainties inevitably arise.

- *Aleatoric* (data uncertainty). The irreducible uncertainty exists inherently in data, i.e., indeterministic outcomes.
- *Epistemic* (model uncertainty). The uncertainty results from lack of knowledge and informative data.

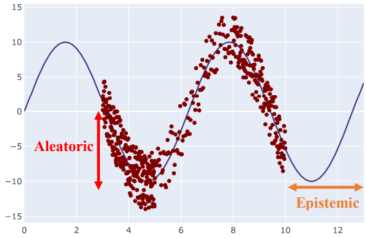


Figure 1: Aleatoric and epistemic uncertainties in a timeseries prediction [1].

Quantify Uncertainties

To account for such uncertainties, Uncertainty Quanlification (UQ) is crucial to make ML/DL models accurate and trustworthy in critical systems or decision-making processes.

The predictive uncertainties (PU) associated with a model is expressed as a direct sum of aleatoric (AU) and epistemic ones (EU).

$$PU = EU + AU$$

Epistemic Uncertainties

Since EU is intrinsic to the model, it can be formulated as a probability distribution over model parameters.

Denote training dataset for classification by $\mathcal{D}_{\text{train}} = \{X, Y\}$ consisting of samples (x_i, y_i). Each $x_i \in \mathbb{R}^f$ is a feature vector and y_i is its label (scalar in regression and category in classification). A model is a function f_Θ with paratermeters Θ that takes x and predicts y. The model likelihood is thus $p\left(y \mid x, \Theta\right)$.

The posterior distribution $p(\Theta \mid x, y)$ for $\mathcal{D}_{\text{train}}$ can be obtained by Bayes' theorem

$$p(\Theta \mid X, Y) = \frac{p(Y \mid X, \Theta)p(\Theta)}{p(Y \mid X)}.$$

Epistemic Uncertainties (cont.)

During testing, a sample x^*'s label w.r.t. $p(\Theta|X,Y)$ can be predicted

$$p(y^* \mid x^*, X, Y) = \int p(y^* \mid x^*, \Theta) p(\Theta \mid X, Y) \, \mathrm{d}\Theta.$$

In fact, $p(\Theta \mid X, Y)$ accepts no analytical solutions, and can only be approximated by variational parameters, i.e., $q_\Phi(\Theta)$. It will thus be approximated to be close to the model's posterior

$$KL\left(q_\Phi\left(\Theta\right) \| p\left(\Theta | X, Y\right)\right) = \int q_\Phi(\Theta) \log \frac{q_\Phi(\Theta)}{p(\Theta \mid X, Y)} \mathrm{d}\Theta.$$

Epistemic Uncertainties (cont.)

The predictive distribution $p_\Theta(y^* \mid x^*, X, Y)$ can be approximated by minimizing the KL divergence

$$p_\Theta(y^* \mid x^*, X, Y) \approx \int p_\Theta(y^* \mid x^*, \Theta) q^*_\Phi(\Theta)\ \mathrm{d}\Theta.$$

Or equivalently, maximize the *evidence lower bound* (ELBO)

$$\mathcal{L}_{VI}(\Phi) := \int q_\Phi(\Theta) \log p(Y \mid X, \Theta)\ \mathrm{d}\Theta - KL(q_\Phi(\Theta) \| p(\Theta)).$$

The process above is called *variational inference* (VI).

Epistemic Uncertainties (cont.)

To obtain data-dependent uncertainty, the precision (in a regression model) can be formulated as a function of data.

A possible way to quantify epistemic uncertainties is mix two functions, predictive mean $f_\Theta(x)$ and model precision $g_\Theta(x)$ in a likelihood function $y_i = \mathcal{N}\left(f_\Theta(x), g_\Theta(x)^{-1}\right)$.

Table of Contents

Monte Carlo Dropout

Monte Carlo (MC) can be used to approximate the aforementioned posterior, but it can be extremely costly to be integrated into DL frameworks. MC dropout aims to reduce the computational cost.

Dropout is a regularization technique used in training neural networks. It randomly zeros out a neuron's output with a preset probability p

$$Dropout(\mathbf{Z}) = \mathbf{d} \circ \mathbf{Z} = \mathbf{d} \circ (\mathbf{W} \cdot \mathbf{x}), \mathbf{d} \in \mathbb{R}^h, \mathbf{d}_i \sim B(1-p).$$

Monte Carlo Dropout (cont.)

MC dropout introduces dropout in testing. Instead of serving as a regularization, dropout here is a Monte Carlo sampling from all possible models. Then, the approximation is made by running dropout several times and then averaging results.

Other types of probability distributions and drop schemes can also be used, like Gaussian dropout ($\mathbf{d}_i \sim N(0, 1)$) and spatial Bernoulli dropout (on feature maps).

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is also effective in approximating the posterior. It works by first sampling a random variable z_0 from distribution $q(z_0)$ or $q(z_0|x)$, then it applies a stochastic transition to the series of z_i

$$Z_t \sim q(z_t|z_{t-1}, x).$$

This transition will repeat many times until its outcome converges to the posterior distribution. However, this process requires a large amount of computational time to converge.

Markov Chain Monte Carlo (cont.)

The stochastic gradient MCMC (SG-MCMC) [2], [3] is proposed for training deep neural networks. It only tries to estimate the gradient of each mini-batch, thus largely reducing computational costs.

Table of Contents

Variatinal Inference

Variational inference (VI) is an approximation method that learns the posterior distribution over Bayes neural network weights.

VI-based methods consider the posterior approximation as an optimization problem besides the task's objective, therefore, it is optimized during the training with gradient descend as well.

Variational Inference (cont.)

Approximating posterior distribution with VI in training, the loss becomes

$$\mathcal{L}(\Theta) = \frac{1}{2|D|}\sum_{i=1}^{|D|}\mathcal{L}_R(y_i, x_i) + \frac{1}{|D|}KL(q_\theta(w)\|p(w)).$$

$$\mathcal{L}_R(y, x) = -\log(\hat{\tau}_x)^\top \mathbf{1} + \left\|\sqrt{\hat{\tau}_x \circ (y - \hat{\mu}_x)}\right\|^2$$

$$\hat{\mu}_x = \hat{\mu}(x, w_\mu),\ w \sim q_\theta(w),\ \hat{\tau}_x = \hat{\tau}(x, w_r).$$

D is the training set and $|D|$ is the number of samples.

Variational Autoencoder

Variational autoencoder (VAE) is a deep learning model that uses VI in an encoder-decoder structure. The encoder maps high-dimensional data to a low-dimensional latent variable, and the decoder reproduces the original data from the latent variable that should conform to a prior distribution.

Variational Autoencoder (cont.)

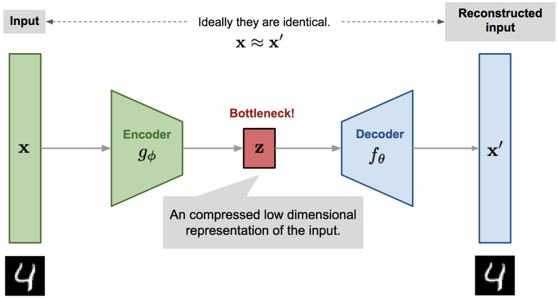
A probabilistic model $P_\Theta(x)$ that produces sample x from a latent variable z is

$$p_\Theta(x) = \int_z p_\Theta(x|z)p(z)\mathrm{d}z.$$

VI is used to model the evidence lower bound (ELBO)

$$\log p_\Theta(x) = \mathbb{E}_{q_\Phi(z|x)}\left[\log p_\Theta(x|z)\right] - D_{KL}\left(q_\Phi\left(z|x\right)\|p(x)\right).$$

Variational Autoencoder (cont.)



Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 2: A VAE's structure. Encoder is colored in green and decoder in blue. $z = \mu + \hat{z} \cdot \sigma$. $\hat{z}$ is sampled from a prior distribution. μ and σ are both NN outputs.

Table of Contents

Deep Gaussian Processes

Gaussian processes (GPs) is a non-parametric type of Bayesian models. It first encodes the similarity between samples with kernel function $k(\cdot, \cdot)$, and represents distributions over the latent variables w.r.t. the input samples as a Gaussian distribution $f_x \sim \mathcal{N}(m(x), k(x, x'))$.

The output y then becomes a distribution based on a likelihood function $y|f_x \sim h(f_x)$.

Deep Gaussian Processes (cont.)

To solve GPs analytically requires to take matrix inverses, which is of $\mathcal{O}(N^3)$ complexity w.r.t. the scale of data. Therefore, a variational lower bound is optimized instead

$$\log p(Y) \geq \sum_{y,x \in Y,X} \mathbb{E}_{q(f_x)} \left[\log p\left(y|f_x\right)\right] - \mathrm{KL}\left(q(f_Z) \| p(f_Z)\right)$$

$q(f_x)$ is the variational approximation to the distribution of f_x and Z is the locations of the inducing points.

Deep Gaussian Processes (cont.)

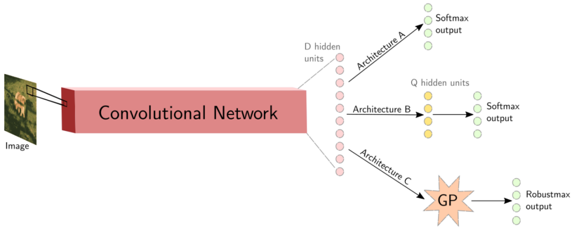


Figure 3: A Gaussian Process Deep Neural Network framework to increase robustness in image classification [4]. Architecture A and B are both ordinary CNNs for image classification, while C utilizes GP to ensure robust performances over adversarial samples.

Table of Contents

Maximum a Posterior Estimation

A *maximum a posterior probability* (MAP) estimate is an estimate of an unknown quantity on the basis of observable data. It is similar to maximum likelihood estimation (MLE). But its optimization objective utilizes an additional prior distribution over the quantity to be estimated.

Maximum a Posterior Estimation (cont.)

Suppose an unobserved parameter θ and its observation x. The likelihood function is $\theta \mapsto f(x \mid \theta)$ with its MLE $\hat{\theta}_{\text{MLE}}(x) = \arg\max_{\theta} f(x|\theta)$.

Assume an additional prior g over θ exists. Then θ can be seen as a random variable and its posterior distribution can be calculated from Bayes' theorem.

$$\theta \mapsto f(\theta \mid x) = \frac{f(x \mid \theta)\, g(\theta)}{\displaystyle\int_{\Theta} f(x \mid \vartheta)\, g(\vartheta)\, d\vartheta}$$

Maximum a Posterior Estimation (cont.)

The MAP then estimates θ as the mode of the posterior distribution of this r.v.

$$\begin{aligned}\hat{\theta}_{\mathrm{MAP}}(x) &= \arg\max_{\theta} \ f(\theta \mid x) \\ &= \arg\max_{\theta} \ \frac{f(x \mid \theta)\, g(\theta)}{\displaystyle\int_{\Theta} f(x \mid \vartheta)\, g(\vartheta)\, d\vartheta} \\ &= \arg\max_{\theta} \ f(x \mid \theta)\, g(\theta).\end{aligned}$$

The denominator is eliminated as it is positive and irrelevant to θ.

Laplace Approximations

Laplace approximations (LAs) can be used to estimate the posterior distribution $f(x \mid \theta)$. For general LAs, $\int_a^b g(x)\mathrm{d}x$ is approximated in the following way

$$\text{Let } h(x) = \log g(x),\ \int_a^b g(x)\mathrm{d}x = \int_a^b \exp(h(x))\mathrm{d}x$$

$$\int_a^b \exp(h(x))\mathrm{d}x \approx \int_a^b \exp\left(h(x_0) + h'(x_0)(x - x_0) + \frac{1}{2}h''(x_0)(x-x_0)^2\right)$$

(Taylor series approximation)

$$= \int_a^b \exp\left(h(x_0) + \frac{1}{2}h''(x_0)(x-x_0)^2\right)\mathrm{d}x$$

(x_0 is optimal, thus $h'(x_0) = 0$)

$$= \exp(h(x_0)) \int_a^b \exp\left(-\frac{1}{2}\frac{(x-x_0)^2}{-h''(x_0)^{-1}}\right)$$

($\exp(h(x_0))$ is constant, the integral becomes a Gaussian).

Laplace Approximations (cont.)

For a neural network, if we approximate the log posterior over the parameters θ of a network given some dataset $\mathcal{D}$ around a MAP estimate [5]

$$\log p(\theta \mid \mathcal{D}) \approx \log p(\theta^* \mid \mathcal{D}) - \frac{1}{2}(\theta - \theta^*)^\top \hat{H}(\theta - \theta^*).$$

$\bar{H} = \mathbb{E}[H]$ is the average Hessian of the negative log posterior. If exponentiated, the last term is a Gaussian functional form for θ. The posterior over θ is then approximated as Gaussian

$$\hat{\theta} \sim \mathcal{N}\left(\theta^*, \bar{H}^{-1}\right).$$

Laplace Approximations (cont.)

The posterior mean when predicting on unseen dataset $\mathcal{D}^*$ can then be approximated by averaging the predictions of T Monte Carlo samples $\theta^{(t)}$ from the approximate posterior

$$p\left(\mathcal{D}^* \mid \mathcal{D}\right) = \int p\left(\mathcal{D}^* \mid \theta\right) p\left(\theta \mid \mathcal{D}\right) \mathrm{d}\theta \approx \frac{1}{T} \sum_{t=1}^{T} p\left(\mathcal{D}^* \mid \theta^{(t)}\right).$$

Thank You for Your Attention

Q & A

References

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. W. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *CoRR*, vol. abs/2011.06225, 2020. arXiv: 2011.06225. [Online]. Available: https://arxiv.org/abs/2011.06225.

[2] T. Chen, E. B. Fox, and C. Guestrin, "Stochastic gradient hamiltonian monte carlo," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, ser. JMLR Workshop and Conference Proceedings, vol. 32, JMLR.org, 2014, pp. 1683–1691. [Online]. Available: http://proceedings.mlr.press/v32/cheni14.html.

References (cont.)

[3] N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven, "Bayesian sampling using stochastic gradient thermostats," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 3203–3211. [Online]. Available: https://proceedings.neurips.cc/paper/2014/hash/21fe5b8ba755eeaece7a450849876228-Abstract.html.

[4] J. Bradshaw, A. G. de G. Matthews, and Z. Ghahramani, *Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks*, 2017. arXiv: 1707.02476 [stat.ML].

[5] H. Ritter, A. Botev, and D. Barber, "A scalable laplace approximation for neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Skdvd2xAZ.