Uncertainty in Deep Learning

Kevin Barkevich

RIT


TacoTime
SPORTSMAN'S
Country Store
TacoTime


OLD TOWN
KEEP CLEAR
KEEP CLEAR

Uncertainty: What is it?

Aleatoric:

- *“Alea”* - Latin for “dice”
- Represents the variability/randomness in the outcome of an experiment
- High uncertainty indicates the presence of meaningless “random noise”
- **Not** reducible with more training data

Epistemic:

- Represents a lack of knowledge in what the experiment’s outcome should be
- High uncertainty indicates a knowledge gap in the model
- Reducible with more training data

Animation: What is it?

↓ **Raised on cartoons**

That's animation!

Spongebob (Cartoon Animation)

That's animation!

Rick & Morty (Cartoon Animation)

I don't know what that is!

Coraline (Stop-motion Animation)

This is **Epistemic Uncertainty.**

Lack of Knowledge:

- The man cannot tell if a stop-motion film is animation.

Knowledge Gap:

- The man was only raised (trained) on cartoon animation.
- The man could only identify cartoons as animation.

If the man had been raised (trained) on more types of animation (a broader dataset), he could've identified Coraline as animation.

Animation: What is it?

↓**Raised on a broad variety of animation**

That's animation!

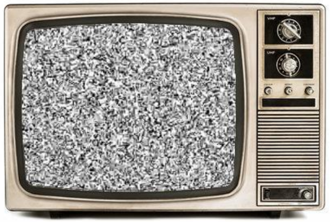
That's animation!

I don't know what that is!

Spongebob (Cartoon Animation)

Coraline (Stop-motion Animation)

Someone knocked over the satellite dish



This is **Aleatoric Uncertainty.**

<u>Variability/Randomness</u>

- The satellite dish being knocked over is unrelated to the type of media being presented.

<u>Meaningless/Random Noise</u>:

- The TV static does not indicate the animation type.

Even though the woman has been raised (trained) on more types of animation (a broader dataset), she <u>still cannot tell</u> if the TV program was animation or not.

Aleatoric Uncertainty: Variance of the Input

X, ***Y*** = Input, Target We assume ***Y*** is conditionally independent given ***X***

*(Common assumption that **Y** is normally distributed given **X:**)*

$P(\boldsymbol{Y} \mid \boldsymbol{X}) = \mathcal{N}(\mu(\boldsymbol{X}), \sigma^2(\boldsymbol{X}))$ μ and σ^2 are the true mean and variance function

<u>Equivalently…</u>

$\boldsymbol{Y} = \mu(\boldsymbol{X}) + \epsilon(\boldsymbol{X})$ with $\epsilon(\boldsymbol{X}) \sim \mathcal{N}(0, \sigma^2(\boldsymbol{X}))$ ***Y*** is generated from ***X*** by $\mu(\boldsymbol{X})$ plus a

zero-mean Gaussi $\hat{\mu}, \hat{\sigma}^2$ ise with variance $\sigma^2(\boldsymbol{X})$. This quantifies the <u>input-dependent (heteroscedastic) aleatoric uncertainty.</u>

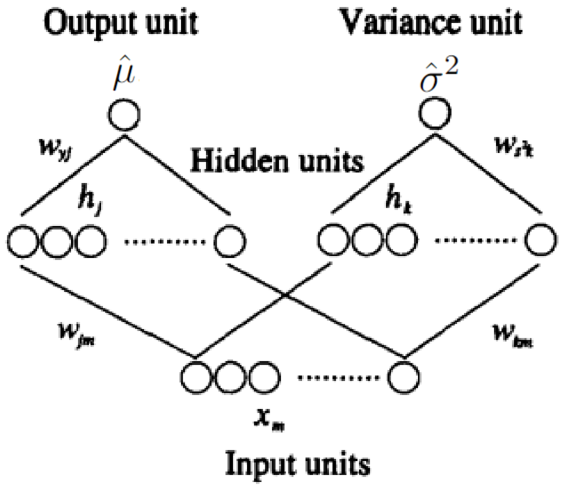
Seitzer, M., Tavakoli, A., Antić, D., & Martius, G. (2022). On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks. *ICLR 2022 - 10th International Conference on Learning Representations*. https://arxiv.org/abs/2203.09168v2

How do we get estimates (μ, σ^2) of the true mean/variance functions?

Predicting Aleatoric Uncertainty

The NN outputs *two* values:

- predicted mean $\hat{\mu}(\boldsymbol{X})$
- predicted variance $\hat{\sigma}^2(\boldsymbol{X}) > 0$.



These observed values are **treated as a sample from a Gaussian distribution** with the predicted mean and variance.

The new predicted value (variance) must be accounted for in the loss, so that the optimizer (i.e. stochastic gradient descent) can optimize for it.

Nix, D. A., & Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. *IEEE International Conference on Neural Networks - Conference Proceedings*, *1*, 55–60. https://doi.org/10.1109/ICNN.1994.374138

Predicting Aleatoric Uncertainty

Assuming the errors are normally distributed around the target:

$$p\left(y|x\right) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2\left(x\right)}} e^{\frac{-(y-\hat{\mu}(x))^2}{2\hat{\sigma}^2(x)}}$$

Take the natural log of both sides to get the log-likelihood we want to maximize:

$$\ln p\left(y|x\right) = -\frac{1}{2}\ln(2\pi) - \frac{\log\hat{\sigma}^2\left(x\right)}{2} - \frac{\left(y - \hat{\mu}\left(x\right)\right)^2}{2\hat{\sigma}^2\left(x\right)}$$

Negate both sides to produce the negative log-likelihood we want to minimize:

$$\mathcal{L}_{\mathrm{NLL}} = -\ln p\left(y|x\right) = \frac{\log\hat{\sigma}^2\left(x\right)}{2} + \frac{\left(y - \hat{\mu}\left(x\right)\right)^2}{2\hat{\sigma}^2\left(x\right)} + \text{constant}$$

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2016). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems*, *2017-December*, 6403–6414. https://arxiv.org/abs/1612.01474v3
Seitzer, M., Tavakoli, A., Antić, D., & Martius, G. (2022). On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks. *ICLR 2022 - 10th International Conference on Learning Representations*. https://arxiv.org/abs/2203.09168v2

Predicting Aleatoric Uncertainty

The optimal parameters of the model θ can be found using maximum likelihood estimation by minimizing the negative log-likelihood criterion $\mathcal{L}_{\mathrm{NLL}}$:

$$\arg_\theta \min \mathcal{L}_{\mathrm{NLL}}(\theta) = \arg_\theta \min \mathop{\mathbb{E}}_{x,y}\left[\frac{\log \hat{\sigma}^2(x)}{2} + \frac{(y-\hat{\mu}(x))^2}{2\hat{\sigma}^2(x)} + \text{constant}\right]$$

The gradients of $\mathcal{L}_{\mathrm{NLL}}$ with respect to $\hat{\mu}(x), \hat{\sigma}^2(x)$ are given by:

$$\nabla_{\hat{\mu}}\mathcal{L}_{\mathrm{NLL}}(\theta) = \mathop{\mathbb{E}}_{x,y}\left[\frac{\hat{\mu}(x)-y}{\hat{\sigma}^2(x)}\right], \quad \nabla_{\hat{\sigma}^2}\mathcal{L}_{\mathrm{NLL}}(\theta) = \mathop{\mathbb{E}}_{x,y}\left[\frac{\hat{\sigma}^2(x)-(y-\hat{\mu}(x))^2}{2\left(\hat{\sigma}^2(x)\right)^2}\right]$$

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2016). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems*, *2017-December*, 6403–6414. https://arxiv.org/abs/1612.01474v3

Seitzer, M., Tavakoli, A., Antić, D., & Martius, G. (2022). On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks. *ICLR 2022 - 10th International Conference on Learning Representations*. https://arxiv.org/abs/2203.09168v2

Estimating Aleatoric Uncertainty (Semantic Seg.)

$$\boldsymbol{Y} = \mu(\boldsymbol{X}) + \epsilon(\boldsymbol{X}) \quad \text{with } \epsilon(\boldsymbol{X}) \sim \mathcal{N}(0, \sigma^2(\boldsymbol{X}))$$

The objective can be approximated through Monte Carlo integration:

$$\hat{x}_{i,t} = \mu_i + \sigma_i^2 \epsilon_t \text{ with } \epsilon_t \sim \mathcal{N}\left(0, \sigma^2(x)\right)$$

$$\mathcal{L}_x = \sum_i \log \frac{1}{T} \sum_t \exp\left(\hat{x}_{i,t,c} - \log \sum_{c'} \exp \hat{x}_{i,t,c'}\right)$$

In the following code example, we use cross-entropy loss (which implicitly applies a softmax activation) to create a semantic segmentation aleatoric uncertainty-aware loss function.

```
# Fetch training data and pass it through the model
T = 2
criterion = torch.nn.CrossEntropyLoss()
data, target = ...
mu, sigma_squared = model(data)

# Obtain the loss
p_hat = []
for t in range(T):
    epsilon = torch.normal(mean=torch.zeros(mu.shape),
                           std=torch.ones(mu.shape))
    noisy_output = mu + sigma_squared * epsilon
    p_hat.append(criterion(noisy_output, target))
loss = torch.mean(torch.stack(p_hat, dim=-1))
loss.backward()
```

Kendall, A., & Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *Advances in Neural Information Processing Systems*, *2017-December*, 5575–5585. https://arxiv.org/abs/1703.04977v2

Epistemic Uncertainty: Uncertainty of the Model

Model uncertainty indicates a knowledge gap in the model.

By repeatedly sampling a model with different dropout parameters, this uncertainty can be represented.

Monte Carlo (MC) Dropout

- Does not require a “special” loss function
- Can be used with existing NN models trained with dropout
- The multiple passes can (with good enough hardware) be done concurrently

Gal, Y., & Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *33rd International Conference on Machine Learning, ICML 2016*, *3*, 1651–1660. https://arxiv.org/abs/1506.02142v6

Epistemic Uncertainty Using MC Dropout

With $\mathcal{W} = \{\mathbf{W}_i\}_{i=1}^{L}$ as the set of dropout-enabled weights for a model with L layers and $q(\mathcal{W})$ defined as an approximating variational distribution:

$$q(\mathbf{y}^*|x^*) = \int p(\mathbf{y}^*|x^*, \mathcal{W})\, q(\mathcal{W})\, d\mathcal{W}$$

The estimate can be acquired through a sampling of T groups of dropout-enabled weights:

$$\mathbb{E}(\mathbf{y}^*) \approx \frac{1}{T} \sum_{t=1}^{T} \hat{\mathbf{y}}^* \left(x^*, \mathbf{W}_1^t, ..., \mathbf{W}_L^t\right)$$

This is equivalent to performing T stochastic forward passes through the network with dropouts enabled and averaging the results.

The sample variance across T stochastic forward passes through the network with dropouts enabled can also be calculated to determine the model uncertainty.

Gal, Y., & Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *33rd International Conference on Machine Learning, ICML 2016*, *3*, 1651–1660. https://arxiv.org/abs/1506.02142v6

Seitzer, M., Tavakoli, A., Antić, D., & Martius, G. (2022). On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks. *ICLR 2022 - 10th International Conference on Learning Representations*. https://arxiv.org/abs/2203.09168v2

Nix, D. A., & Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. *IEEE International Conference on Neural Networks - Conference Proceedings*, *1*, 55–60. https://doi.org/10.1109/ICNN.1994.374138

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2016). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems*, *2017-December*, 6403–6414. https://arxiv.org/abs/1612.01474v3

Kendall, A., & Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *Advances in Neural Information Processing Systems*, *2017-December*, 5575–5585. https://arxiv.org/abs/1703.04977v2

Gal, Y., & Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. 33rd International Conference on Machine Learning, ICML 2016, 3, 1651–1660. https://arxiv.org/abs/1506.02142v6