Dropout as a Bayesian Approximation

appliedAI Seminar — Uncertainty Quantification in Deep Learning

Maternus Herold

20.10.2022

INITIATIVE FOR
APPLIED ARTIFICIAL
INTELLIGENCE

Agenda

1. Motivation

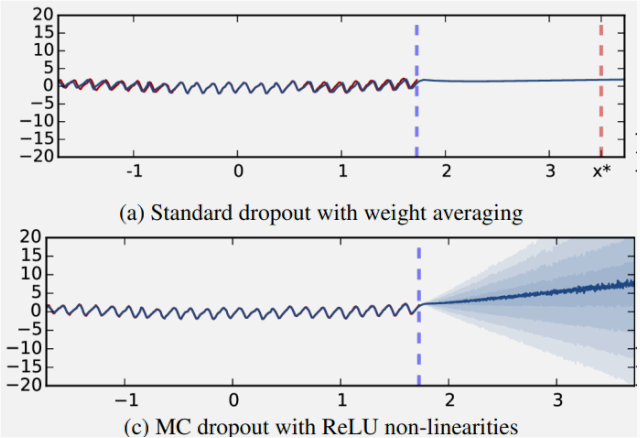
2. Quick intro to Uncertainty Quantification

3. Recap of Dropout in Neural Networks

4. MC Dropout for Uncertainty Quantification in Deep Learning

Motivation

Why do we want to quantify the uncertainty in our models?



Figures showing the true value (red) and the predictive mean (blue) including $\pm 2\sigma$ for a model evaluated on the Mauna Loa CO_2 concentration dataset [4]. The blue shades indicate half a std. deviation each.

Uncertainty Quantification

Uncertainty is devided into aleatoric and epistemic uncrtainty

Uncertainty is devided into aleatoric and epistemic uncrtainty

Aleatoric uncertainty

- Stochastic uncertainty or unknowns, responsible for different outcomes of the same experiment
- Often due to insufficient measurement quality
- Does not preclude the existence of the “unknowns”
- Uncertainty trapped “in the data”
- Methods: Quantile regression, mixture of densities, max. likelihood est.

Uncertainty is devided into aleatoric and epistemic uncrtainty

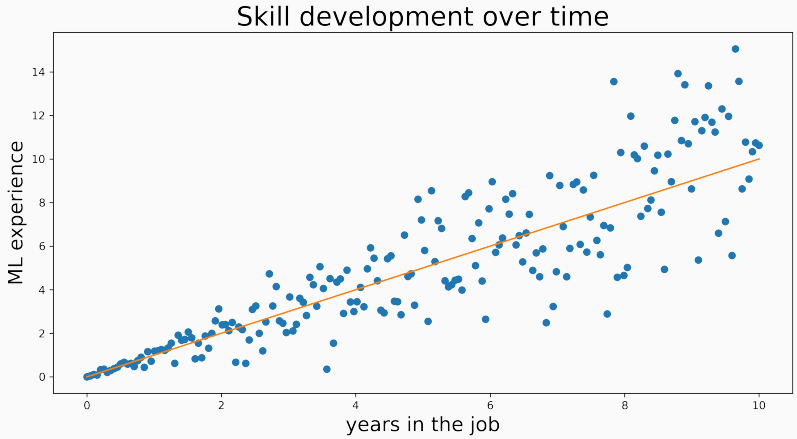
Aleatoric uncertainty

- Stochastic uncertainty or unknowns, responsible for different outcomes of the same experiment
- Often due to insufficient measurement quality
- Does not preclude the existence of the “unknowns”
- Uncertainty trapped “in the data”
- Methods: Quantile regression, mixture of densities, max. likelihood est.

Epistemic uncertainty

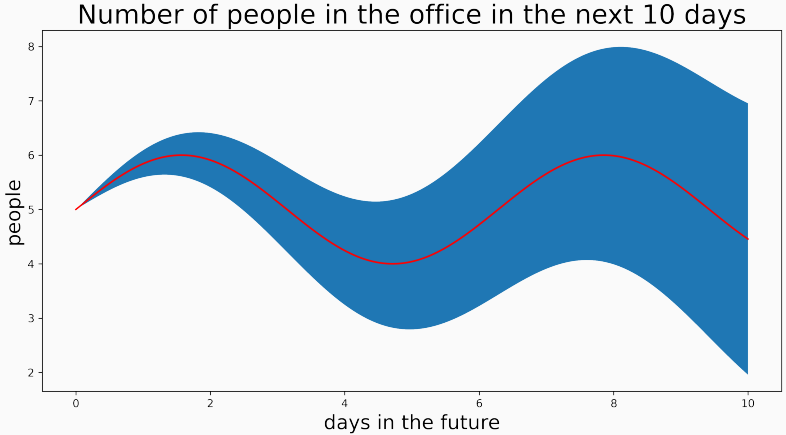
- Systematic uncertainty, due to possibly available information but practically not used
- Inaccuracies in measurements, unused data / features, neglecting effects
- Captured by the model due to design decisions, data used, etc.
- Methods: Bayesian Neural Networks, MC dropout or deep ensembles

Illustration of aleatoric uncertainty



Based on the current information (years in the job), the increasing variance in ML experience cannot be explained.

Illustration of epistemic uncertainty



While the model returns point estimates, it is uncertain about its predictions, especially further in the future.

Recap of Dropout

Dropping our neurons to prevent overfitting

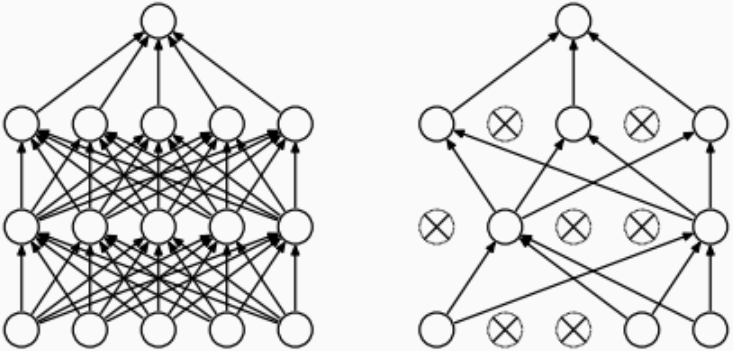


Illustration of a standard neural network vs. one with dropping out nodes [6].

Model definition including dropping out nodes

$$
\begin{aligned}
z_i^{l+1} &= w_i^{l+1} \cdot y^l + b_i^{l+1} \\
y_i^{l+1} &= \sigma(z_i^{l+1})
\end{aligned}
\tag{1}
$$

$$
\begin{aligned}
\eta_j^l &\sim \mathcal{B}(p) \\
\tilde{y}^l &= \eta^l \cdot y^l \\
z_i^{l+1} &= w_i^{l+1} \cdot \tilde{y}^l + b_i^{l+1} \\
y_i^{l+1} &= \sigma(z_i^{l+1})
\end{aligned}
\tag{2}
$$

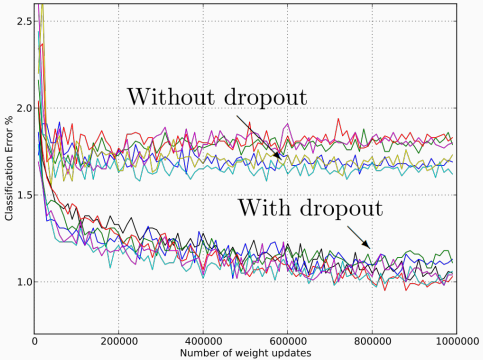
Where the keep probability is p and layers are denoted by $l \in \{0, \ldots, L-1\}$ [6].

Dropout is primarly used to prevent overfitting

- Reduce risk of overfitting
- Model ensemble without optimizing hundreds of models
- Activate all parts of the model

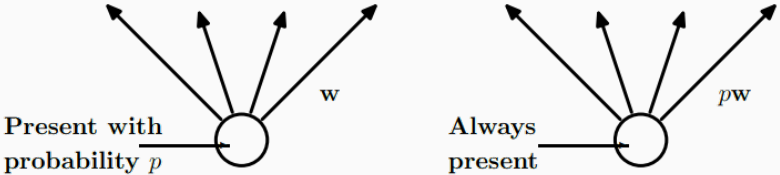
Dropout is primarly used to prevent overfitting

- Reduce risk of overfitting
- Model ensemble without optimizing hundreds of models
- Activate all parts of the model



Dropout improves performance on the test data for different architectures and configurations [6].

Dropout is switched offduring inference — all nodes are active

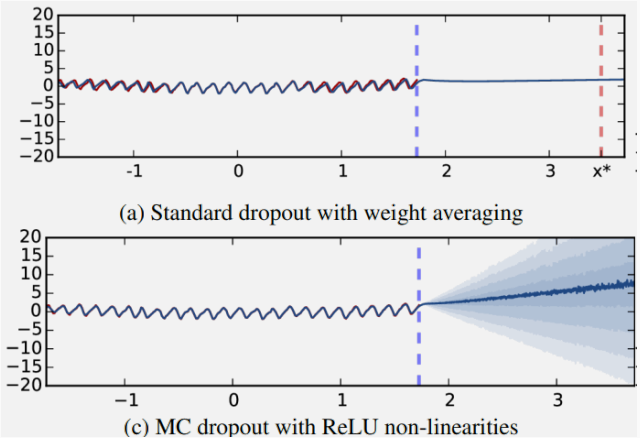


(a) At training time (b) At test time

As dropping nodes is turned off during inference, the weights need to be scaled to not get larger than expected [6].

Monte Carlo Dropout

Q: How do we get the model's uncertainty?



Figures showing the true value (red) and the predictive mean (blue) including $\pm 2\sigma$ for a model evaluated on the Mauna Loa CO_2 concentration dataset [4]. The blue shades indicate half a std. deviation each.

Retrieving the predictive mean and variance

Given a model $\mathcal{M}$ which was trained with dropout and a *keep probability* of p. Generate $\{(\hat{y}_i, \mathbf{x})\}_{i=1}^N$ by **keeping dropout on**.

Estimators for $\mathbb{E}(\mathbf{y}), \mathrm{Var}(\mathbf{y})$ read

$$\begin{aligned} \mathbb{E}(\mathbf{y}) &\approx \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i(\mathbf{x}) \\ \mathrm{Var}(\mathbf{y}) &\approx \tau^{-1}\mathbf{I} + \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i(\mathbf{x})\hat{y}_i(\mathbf{x})^T - \mathbb{E}(\mathbf{y})\mathbb{E}(\mathbf{y})^T \end{aligned} \tag{3}$$

Where $\tau = \frac{l^2 p}{2N\lambda}$ the GP precision; τ, p obtained via hyperparameter search [5].

Implementation MC Dropout

Algorithm 1: Implementation of Monte Carlo Dropout by using dropout during inference as well and computing statistics of the samples from the predictive distribution.

1: $\mathcal{M} \leftarrow \mathcal{M}$`.train()`
2: $\hat{y}_i \leftarrow \mathcal{M}(\mathbf{x}) \; \forall i \in [1, \ldots, N]$
3:
4: $\hat{\mathbb{E}}(\mathbf{y}) \leftarrow$ `ŷ.mean()`
5: $\hat{\sigma}(\mathbf{y}) \leftarrow$ `ŷ.std()`
6:
7: **return** $\hat{\mathbb{E}}(\mathbf{y}), \hat{\sigma}(\mathbf{y})$

Why does it work?

💡 *"Averaging over forward passes is equivalent to Monte Carlo integration over a Gaussian Process posterior approximation."*

→ We'll familiarize with the idea **but** I refer to [3, 4] for a detailed discussion.

Why does it work?

💡 *"Averaging over forward passes is equivalent to Monte Carlo integration over a Gaussian Process posterior approximation."*

→ We'll familiarize with the idea **but** I refer to [3, 4] for a detailed discussion.

Target: predictive distribution for a new observation $\mathbf{x}^*$

$$
\begin{aligned}
p(y^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) &= \int p(y^*|\mathbf{x}^*, \omega) p(\omega|\mathbf{X}, \mathbf{Y}) d\omega \\
&\approx \int p(y^*|\mathbf{x}^*, \omega) q_{\theta^*}(\omega) d\omega, \quad \theta^* = \underset{\theta}{\operatorname{argmin}}\ \mathrm{KL}(q_\theta(\omega)|p(\omega|\mathbf{X}, \mathbf{Y}))
\end{aligned}
\tag{4}
$$

Finding $q_{\theta^*} \Rightarrow$ Minimizing the KL-Divergence

Minimizing KL-Div. $\Leftrightarrow$ maximizing log evidence lower bound [1].

$$
\begin{aligned}
\log p(\mathbf{Y}|\mathbf{X}) &= \log \int p(\mathbf{Y}|\mathbf{X}, \omega) p(\omega) d\omega \\
&= \log \int p(\mathbf{Y}|\mathbf{X}, \omega) p(\omega) \frac{q_\theta(\omega)}{q_\theta(\omega)} d\omega \\
&= \log \left(\mathbb{E}_{q_\theta} \frac{p(\mathbf{Y}|\mathbf{X}, \omega) p(\omega)}{q_\theta(\omega)} \right) \\
&\overset{\text{Jensen's ineq.}}{\geq} \mathbb{E}_{q_\theta} \log \left(\frac{p(\mathbf{Y}|\mathbf{X}, \omega) p(\omega)}{q_\theta(\omega)} \right) \\
L_{\text{VI}} &= \int q_\theta(\omega) \log(p(\mathbf{Y}|\mathbf{X}, \omega)) d\omega - \text{KL}(q_\theta | p(\omega))
\end{aligned}
\tag{5}
$$

Maximizing the log evidence lower bound

Using MC integration to approximate the integral with one sample $\omega \sim q_\theta$ and optimizing w.r.t. θ:

$$\hat{L}(\theta) = \log p(\mathbf{Y}|\mathbf{X}, \hat{\omega}) - \mathrm{KL}(q_\theta | p(\omega)) \tag{6}$$

Connecting the VI approach with dropout

$\Rightarrow q_\theta(\omega)$ is a distribution over weights of a network. Thus, define $q_\theta(\omega)$ s.t. weights can be turned off.

$$
\begin{aligned}
q_{\theta^*} &= \prod_i q_{\mathbf{M}_i}(\mathbf{W}_i), \quad i \in [1, \ldots, L] \\
q_{\mathbf{M}_i}(\mathbf{W}_i) &= \mathbf{M}_i \text{diag}(z_{i,j}) \\
z_{i,j} &\sim \mathcal{B}(p_i) \\
\mathbf{W}_i &\sim q_{\mathbf{M}_i}(\mathbf{W}_i)
\end{aligned} \tag{7}
$$

Drwabacks of Monte Carlo Dropout

- Requires large number of samples making inference slow ($> 10^3$ in [4])
- Inference time is crucial in a lot of applications

Single-shot Monte Carlo Dropout

- “Succession” of the original Monte Carlo Dropout
- Preserves advantages of Bayesian NN without being slower than non-Bayesian NN [2]
- Idea: approximate the expected value and variance of the MC Dropout per layer

Approximation via Moment Propagation

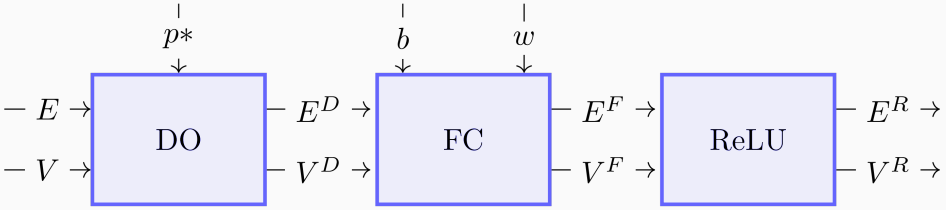


Illustration of Moment Propagation through dropout, fully-connected and ReLU layers [2]. The expectation and variance *flow* through the network during a single forward pass

Approximation via Moment Propagation ctd.

Dropout Layer per layer with nodes i

$$E^i = E^i_{\text{in}} \cdot p$$
$$V^i \overset{X \perp\!\!\!\perp Y}{=} V^i_{\text{in}} \cdot p(1-p) + V^i_{\text{in}} p^2 + (E^i_{\text{in}})^2 \cdot p(1-p) \tag{8}$$

Approximation via Moment Propagation ctd.

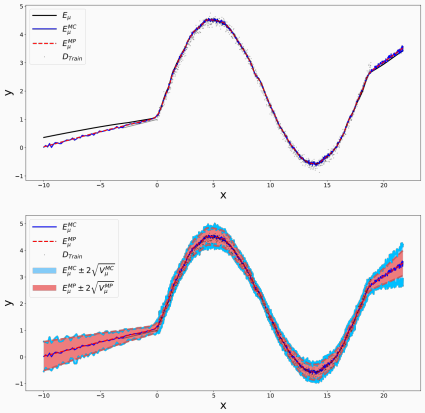
Dropout Layer per layer with nodes i

$$E^i = E^i_{\text{in}} \cdot p$$
$$V^i \overset{X \perp\!\!\!\perp Y}{=} V^i_{\text{in}} \cdot p(1-p) + V^i_{\text{in}} p^2 + (E^i_{\text{in}})^2 \cdot p(1-p) \tag{8}$$

Dense Layer

$$E^i = \sum_j^{N_{\text{in}}} w_{ji} E^j_{\text{in}} + b^i$$
$$V^i = \sum_j^{N_{\text{in}}} w^2_{ji} V^j_{\text{in}} \tag{9}$$

Comparison of Single Shot MC and MC Dropout



Visual comparison of the two methods using MC Dropout. The models were trained on data in the range of $[-3, 19]$. In the upper figure, the predictions are shown. In the lower panel, the uncertainty is included, showing similar results [2].

Comparison of Single Shot MC and MC Dropout ctd.

Dataset	N	Q	Test RMSE		Test NLL		Test RT [s]	
			MC	MP	MC	MP	MC	MP
Boston	506	13	3.14 ±0.20	3.10 ±0.20	2.57 ±0.07	2.56 ±0.08	2.51 ±0.03	0.04 ±0.00
Concrete	1,030	8	5.46 ±0.12	5.40 ±0.12	3.12 ±0.02	3.13 ±0.03	3.37 ±0.04	0.04 ±0.00
Energy	768	8	1.65 ±0.05	1.61 ±0.05	1.95 ±0.04	2.01 ±0.04	2.84 ±0.03	0.04 ±0.00
Kin8nm	8,192	8	0.08 ±0.00	0.08 ±0.00	-1.10 ±0.01	-1.11 ±0.01	7.37 ±0.06	0.04 ±0.00
Naval	11,934	16	0.00 ±0.00	0.00 ±0.00	-4.36 ±0.01	-3.60 ±0.01	9.69 ±0.11	0.04 ±0.00
Power	9,568	4	4.05 ±0.04	4.04 ±0.04	2.82 ±0.01	2.84 ±0.01	6.85 ±0.07	0.04 ±0.00
Protein	45,730	9	4.42 ±0.03	4.41 ±0.02	2.90 ±0.00	2.91 ±0.00	31.38 ±0.09	0.05 ±0.00
Wine	1,599	11	0.63 ±0.01	0.63 ±0.01	0.95 ±0.01	0.95 ±0.01	4.78 ±0.01	0.04 ±0.00
Yacht	308	6	2.93 ±0.22	2.91 ±0.26	2.35 ±0.07	2.11 ±0.07	2.01 ±0.01	0.04 ±0.00

Benchmarking both approaches on the UCI benchmarking datasets where $N = 10^4$ forward passes were used for the MC Dropout model. Both methods show similar results on the test data while the approach of Moment Propagation has a significant advantage w.r.t. inference time [2].

Dropout as a Bayesian Approximation

appliedAI Seminar — Uncertainty Quantification in Deep Learning

Maternus Herold

20.10.2022

INITIATIVE FOR
APPLIED ARTIFICIAL
INTELLIGENCE

Literatur

[1] Christopher M Bishop und Nasser M Nasrabadi. *Pattern recognition and machine learning*. Bd. 4. 4. Springer, 2006.

[2] Kai Brach, Beate Sick und Oliver Dürr. *Single Shot MC Dropout Approximation*. 2020. URL: https://arxiv.org/abs/2007.03293.

[3] Yarin Gal und Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Appendix*. 2015. URL: https://arxiv.org/abs/1506.02157.

References ii

[4] Yarin Gal und Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR. 2016, S. 1050–1059.

[5] Yarin Gal und Jishnu Mukhoti. *Code Examples for Dropout as Bayesian Approximation*. `https://github.com/yaringal/DropoutUncertaintyExps`. [Online; accessed 19-Oct-2022]. 2016.

[6] Nitish Srivastava u. a. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), S. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.