Dropout as a Bayesian Approximation: An interpretation of Gal and Ghahramani (2016)

Anna Lithell
Advanced Statistical Inference
EURECOM
Anna.Lithell@eurecom.fr

June 3, 2025

Abstract

This project is part of the EURECOM course *Advanced Statistical Inference* and aims to provide an interpretation of the work described in *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*, published in 2016 by Gal and Ghahramani [2]. The project consists of two parts. The first part is to reproduce some of the results described in the paper. The second part is describing the findings and conclusions drawn, both from the experiment but also from reading the paper. This report concludes the second part of the project.

1 The Paper

The uncertainty of a neural network (NN) model is an important quantitative metric that tells us how confident a model is in its predictions. However, classic deep learning tools do not capture predictive uncertainty of a model. Instead, Bayesian models and frameworks can be used to achieve this. The paper authors Gal and Ghahramani begin introducing the reader to the fact that while Bayesian models offer a grounded mathematical framework to reason about a model's uncertainty, they are computationally heavy and difficult to implement in real-life.

Their findings presented in the paper show a new way of reasoning about Bayesian approximation. By doing so, they prove that dropout as an example can be interpreted as a Bayesian Approximation of an existing model, estimating the model's predictive uncertainty without changing the architecture of the model itself. This allows us to implement NNs utilizing Bayesian models in order to predict the model uncertainty, while maintaining a comparably low computational cost.

1.1 Dropout as a form of Bayesian Approximation

Dropout is a well-known method utilized frequently in order to prevent over-fitted models in the real of machine learning. In traditional applications during training, using dropout will randomly set units in a layer to zero. In the paper, Gal and Ghahramani present the following standard equation for dropout. [1]

$$\mathcal{L}_{dropout} = \frac{1}{N}\sum_{i=1}^{N} E(y_i, \hat{y}_i) + \lambda \sum_{l=1}^{L} \left(\|W_l\|_2^2 + \|b_l\|_2^2 \right)$$

Where the first term is the standard loss function and the second term represents L2 weight decay, used in regularization. A unit is dropped with a probability of p_i for layer i. In the third section of the paper, Gal and Ghahramani show how a NN with dropout is mathematically equivalent to a variational approximation of a deep Gaussian process (DGP).

The goal of Bayesian modeling is that we want to predict new labels y given a test input x and training data (X,Y) by marginalizing over the weights, giving us a true posterior predictive.

$$p(y \mid x, X, Y) = \int p(y \mid x, \omega)\, p(\omega \mid X, Y)\, d\omega$$

However, the integral of the posterior distribution $p(\omega \mid X, Y)$ is intractable to compute. To approximate the intractable posterior, the paper authors define $p(\omega)$ as a dropout distribution.

$$W_i = M_i \cdot diag(z_{i,j}), \quad z_{i,j} \sim Bernoulli(p_i)$$

As a result, this means that during training each forward pass of a NN samples a different dropout mask. Using this simplified and computationally efficient $p(\omega)$ allows us to define an optimization goal. The goal is to minimize the KL divergence between the approximate posterior $p(\omega)$ and the posterior of the DGP $p(\omega \mid X, Y)$.

$$-\int q(\omega) \log p(Y \mid X, \omega)\, d\omega + KL(q(\omega) \,||\, p(\omega))$$

Meaning, training a NN with dropout is equivalent to minimizing the objective defined above. This integral can be further simplified by rewriting the first term of the equation as a sum.

$$-\sum_{n=1}^{N} \int q(\omega) \log p(y_n \mid x_n, \omega)\, d\omega$$

Each term can then be approximated by Monte Carlo integration with a single sample to get an unbiased estimate $-\log p(y_n \mid x_n, \hat{\omega}_n)$.

The second term in the optimization objective can be further approximated to obtain the final objective given a model precision τ.

$$\mathcal{L}_{GP-MC} \propto \frac{1}{N}\sum_{n=1}^{N}\left(-\frac{1}{\tau}\log p(y_n \mid x_n, \hat{\omega}_n)\right) + \sum_{i=1}^{L}\left(\frac{p_i l^2}{2\tau N}\|\mathbf{M}_i\|_2^2 + \frac{l^2}{2\tau N}\|\mathbf{m}_i\|_2^2\right)$$

where the unbiased estimate is defined as

$$E(y_n, \hat{y}(x_n, \hat{\omega}_n)) = -\frac{1}{\tau}\log p(y_n \mid x_n, \hat{\omega}_n)$$

1.2 How to obtain model uncertainty

Once we have trained a NN with dropout, we can start making uncertainty-aware predictions using Monte Carlo (MC) Dropout. An approximate predictive distribution is given by the following equation.

$$p(y^* \mid x^*) = \int p(y^* \mid x^*, \omega)\, q(\omega)\, d\omega$$

In a practical implementation, the NN model needs to enable dropout during testing for the following scheme to work. We sample T sets of vectors of realization from the Bernoulli distribution $\{z_1^t, ..., z_L^t\}_{t=1}^T$ with $z_i^t = [z_{i,j}^t]_{j=1}^{K_i}$, giving $\{W_t^1, \ldots, W_L^t\}_{t=1}^T$, equivalent to enabling dropout. The following estimate is defined as MC dropout.

$$\mathbb{E}_{q(y^*|x^*)}(y^*) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{y}^*(x^*, W_1^t, ..., W_L^t)$$

In practice, the MC dropout is according to the authors of the paper equivalent to performing T stochastic forward passes through the network and then averaging the results. After performing this operation, the model's predictive mean and variance can be calculated from the results of the T forward passes. This is key to what makes this paper's contribution monumental, as this method can be applied to existing NN models trained with dropout.

2 An interpretation in code

After reading the paper, I decided to design a NN model following the provided recipe. The model consists of 4 hidden layers with 1024 hidden units in each layer. The model uses ReLU nonlinearities, the mean squared error (MSE) loss function and a dropout probability of 0.1 as default in the network. The model aims to predict the uncertainty in a regression task. The task I've chosen is to predict the median housing price in different Californian districts considering 8 different parameters, such as the median income given a district and the average number of rooms in a house [3]. To speed up the training phase, I used mini-batches of size 64.

A key difference compared to traditional NNs is the fact that dropout is enabled during both training and testing. This allows each forward pass to generate a different output, even though the input is the same [4]. According to the paper, the practical implementation of this is what is equivalent to Monte-Carlo sampling and key to how approximations of Bayesian Models become feasible in real-life. In my implementation, the following operation is performed.

```
def predict_mc(model, data_loader, T=100):
    model.train() # Important: keep dropout active
    device = next(model.parameters()).device
    all_predictions = []
    with torch.no_grad():
        for X_batch, _ in data_loader:
            X_batch = X_batch.to(device)
            # make multiple forward passes for each input sample
            batch_predictions = torch.stack([model(X_batch) for _ in range
                (T)], dim=0)
            all_predictions.append(batch_predictions)

    # Concatenate predictions from all batches
    all_predictions = torch.cat(all_predictions, dim=1) # Concatenate
        along the batch dimension (dim=1)

    mean = all_predictions.mean(dim=0)
    std = all_predictions.std(dim=0)
    return mean, std

# Create test_loader
batch_size = 64 # Use same batch size as during training
test_dataset = TensorDataset(X_test, y_test)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=
    False)

# Call the modified predict_mc function with the test_loader
mean, std = predict_mc(model, test_loader, T=100)
```

The function enables dropout during testing. It then concatenates the predictions made for all batches after making T forward passes for each input sample. The function returns the estimated model mean as the average predicted output for each input sample over T stochastic forward passes, while the estimated variance measures the uncertainty of the model's predictions across T forward passes. I defined T = 100 as the default number of forward passes.

3 Results

In Figure 3, three different plots illustrate the model's predictive uncertainty. The left subplot shows the residuals between predicted and true median house

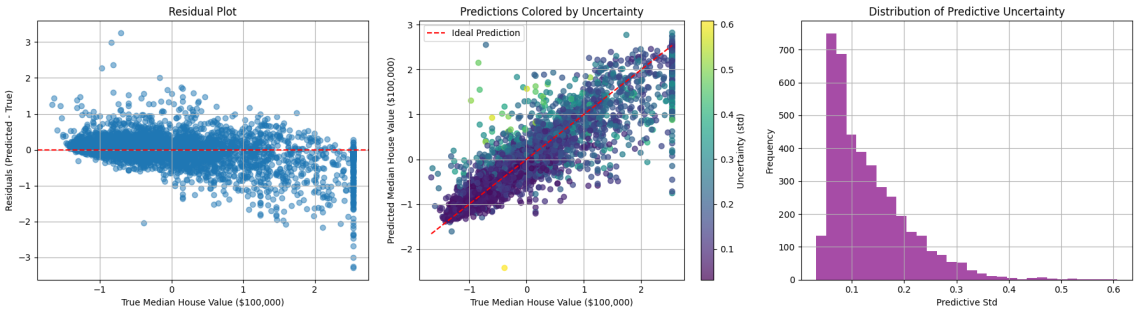

Figure 1: Residuals vs. true values (left), predictions colored by model certainty (center) and distribution of predictive uncertainty (right).

values. As expected, the residuals are centered around zero, but for higher true values there is an increased dispersion, indicating underfitting or higher difficulty to determine the correct median price.

The center graph highlights the uncertainty in each prediction made by the model by using different colors. Brighter (yellow) points correspond to higher uncertainty, while cooler (blue) points indicate more confident predictions. The graph shows that the model is more confident for lower and frequent values, while a higher median price often results in a less confident prediction.

Finally, the right-most graph shows the distribution of predictive uncertainty across the test set. A low variance in this graph corresponds to a high certainty in the prediction made. The model is in general confident, as a majority of predictions have low variance. However, the long slope indicates some predictions are significantly uncertain.

4 How AI tools have been used in this project

Chat-GPT have been used to provide a simplified explanation of the paper and the presented concepts. This explanation have then been modified to become more complex, explaining how all equations presented in the paper are related to one another. In addition, Chat-GPT have been used when designing the architecture of the NN used in the project. It has also been used to debug code, generate graphs, write references in BibTeX format and equations in LaTeX syntax.

References

[1] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Appendix, 2016.

[2] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[3] Inria. California housing dataset — scikit-learn mooc. `https://inria.github.io/scikit-learn-mooc/python_scripts/datasets_california_housing.html`, 2025. Accessed: 2025-05-30.

[4] Ahmed Taha. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. https://ahmdtaha.medium.com/dropout-as-a-bayesian-approximation-representing-model-uncertainty-in-deep-learning-7a2e49e64a15, 2018. Accessed: 2025-05-30.