

Bayesian Neural Network: Foundation and Practice

Tianyu Cui, Yi Zhao

Department of Computer Science
Aalto University

May 2, 2019

Outline

Introduction to Bayesian Neural Network

Dropout as Bayesian Approximation

Concrete Dropout

Introduction to Bayesian Neural Network

What's a Neural Network?

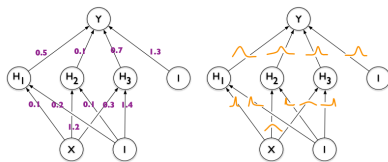


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

Probabilistic interpretation of NN:

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon$, $\epsilon \sim N(0, \sigma^2)$

What's a Neural Network?

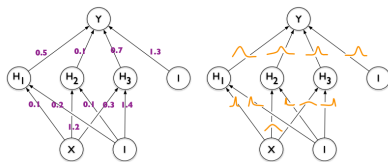


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

Probabilistic interpretation of NN:

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon$, $\epsilon \sim N(0, \sigma^2)$
- ▶ **Likelihood:** $P(y|\mathbf{x}, \mathbf{w}) = N(y; f(\mathbf{x}; \mathbf{w}), \sigma^2)$

What's a Neural Network?

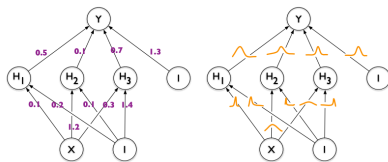


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

Probabilistic interpretation of NN:

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon$, $\epsilon \sim N(0, \sigma^2)$
- ▶ **Likelihood:** $P(y|\mathbf{x}, \mathbf{w}) = N(y; f(\mathbf{x}; \mathbf{w}), \sigma^2)$
- ▶ **Prior:** $P(\mathbf{w}) = N(\mathbf{w}; 0, \sigma_w^2 \mathbf{I})$

What's a Neural Network?

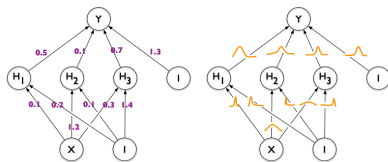


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

Probabilistic interpretation of NN:

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon$, $\epsilon \sim N(0, \sigma^2)$
- ▶ **Likelihood:** $P(y|\mathbf{x}, \mathbf{w}) = N(y; f(\mathbf{x}; \mathbf{w}), \sigma^2)$
- ▶ **Prior:** $P(\mathbf{w}) = N(\mathbf{w}; 0, \sigma_w^2 \mathbf{I})$
- ▶ **Posterior:** $P(\mathbf{w}|y, \mathbf{x}) \propto P(y|\mathbf{x}, \mathbf{w})P(\mathbf{w})$

What's a Neural Network?

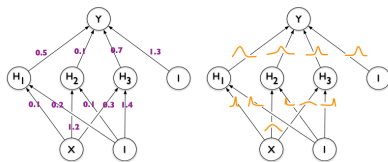


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

Probabilistic interpretation of NN:

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon$, $\epsilon \sim N(0, \sigma^2)$
- ▶ **Likelihood:** $P(y|\mathbf{x}, \mathbf{w}) = N(y; f(\mathbf{x}; \mathbf{w}), \sigma^2)$
- ▶ **Prior:** $P(\mathbf{w}) = N(\mathbf{w}; 0, \sigma_w^2 \mathbf{I})$
- ▶ **Posterior:** $P(\mathbf{w}|y, \mathbf{x}) \propto P(y|\mathbf{x}, \mathbf{w})P(\mathbf{w})$
- ▶ **MAP:** $\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}|y, \mathbf{x})$

What's a Neural Network?

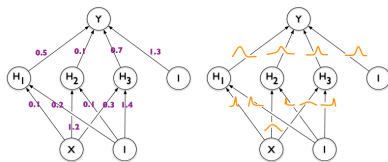


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

Probabilistic interpretation of NN:

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon$, $\epsilon \sim N(0, \sigma^2)$
- ▶ **Likelihood:** $P(y|\mathbf{x}, \mathbf{w}) = N(y; f(\mathbf{x}; \mathbf{w}), \sigma^2)$
- ▶ **Prior:** $P(\mathbf{w}) = N(\mathbf{w}; 0, \sigma_w^2 \mathbf{I})$
- ▶ **Posterior:** $P(\mathbf{w}|y, \mathbf{x}) \propto P(y|\mathbf{x}, \mathbf{w})P(\mathbf{w})$
- ▶ **MAP:** $\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}|y, \mathbf{x})$
- ▶ **Prediction:** $y' = f(\mathbf{x}'; \mathbf{w}^*)$

What's a Bayesian Neural Network?

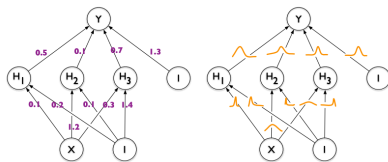


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

What do I mean by being Bayesian?

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon, \epsilon \sim N(0, \sigma^2)$
- ▶ **Likelihood:** $P(y|\mathbf{x}, \mathbf{w}) = N(y; f(\mathbf{x}; \mathbf{w}), \sigma^2)$
- ▶ **Prior:** $P(\mathbf{w}) = N(\mathbf{w}; 0, \sigma_w^2 \mathbf{I})$
- ▶ **Posterior:** $P(\mathbf{w}|y, \mathbf{x}) \propto P(y|\mathbf{x}, \mathbf{w})P(\mathbf{w})$
- ▶ **MAP:** $\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}|y, \mathbf{x})$

What's a Bayesian Neural Network?

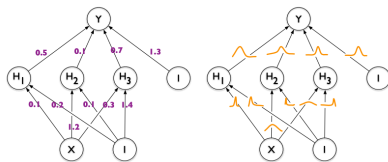


Figure: A simple NN (left) and a BNN (right)[Blundell, 2015].

What do I mean by being Bayesian?

- ▶ **Model:** $y = f(\mathbf{x}; \mathbf{w}) + \epsilon, \epsilon \sim N(0, \sigma^2)$
- ▶ **Likelihood:** $P(y|\mathbf{x}, \mathbf{w}) = N(y; f(\mathbf{x}; \mathbf{w}), \sigma^2)$
- ▶ **Prior:** $P(\mathbf{w}) = N(\mathbf{w}; 0, \sigma_w^2 \mathbf{I})$
- ▶ **Posterior:** $P(\mathbf{w}|y, \mathbf{x}) \propto P(y|\mathbf{x}, \mathbf{w})P(\mathbf{w})$
- ▶ **MAP:** $\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}|y, \mathbf{x})$
- ▶ **Prediction:** $y' = f(\mathbf{x}'; \mathbf{w}), \mathbf{w} \sim P(\mathbf{w}|y, \mathbf{x})$

Why Should We Care?

Calibrated prediction uncertainty:

The models should know what they don't know.

One Example: [Gal, 2017]

- ▶ We train a model to recognise dog breeds.



Why Should We Care?

Calibrated prediction uncertainty:

The models should know what they don't know.

One Example: [Gal, 2017]

- ▶ We train a model to recognise dog breeds.
- ▶ What would you want your model to do when a cat are given?



Why Should We Care?

Calibrated prediction uncertainty:

The models should know what they don't know.

One Example: [Gal, 2017]

- ▶ We train a model to recognise dog breeds.
- ▶ What would you want your model to do when a cat are given?
- ▶ A prediction with high uncertainty.



Why Should We Care?

Calibrated prediction uncertainty:

The models should know what they don't know.

One Example: [Gal, 2017]

- ▶ We train a model to recognise dog breeds.
- ▶ What would you want your model to do when a cat are given?
- ▶ A prediction with high uncertainty.

Successful Applications:

- ▶ Identify adversarial examples [Smith, 2018].
- ▶ Adapted exploration rate in RL [Gal, 2016].
- ▶ Self-driving car [McAllister, 2017, Michelmore, 2018] and medical analysis [Gal, 2017].

How To Learn a Bayesian Neural Network?

What's the difficult part?

- ▶ $P(\mathbf{w}|y, \mathbf{x})$ is generally intractable

How To Learn a Bayesian Neural Network?

What's the difficult part?

- ▶ $P(\mathbf{w}|y, \mathbf{x})$ is generally intractable
 - ▶ Standard **approximate inference** (difficult):

How To Learn a Bayesian Neural Network?

What's the difficult part?

- ▶ $P(\mathbf{w}|y, \mathbf{x})$ is generally intractable
 - ▶ Standard **approximate inference** (difficult):
 - ▶ Laplace Approximation [MacKay, 1992];

How To Learn a Bayesian Neural Network?

What's the difficult part?

- ▶ $P(\mathbf{w}|y, \mathbf{x})$ is generally intractable
 - ▶ Standard **approximate inference** (difficult):
 - ▶ Laplace Approximation [MacKay, 1992];
 - ▶ Hamiltonian Monte Carlo [Neal, 1995];

How To Learn a Bayesian Neural Network?

What's the difficult part?

- ▶ $P(\mathbf{w}|y, \mathbf{x})$ is generally intractable
 - ▶ Standard **approximate inference** (difficult):
 - ▶ Laplace Approximation [MacKay, 1992];
 - ▶ Hamiltonian Monte Carlo [Neal, 1995];
 - ▶ (Stochastic) Variational Inference [Blundell, 2015].

How To Learn a Bayesian Neural Network?

What's the difficult part?

- ▶ $P(\mathbf{w}|y, \mathbf{x})$ is generally intractable
 - ▶ Standard **approximate inference** (difficult):
 - ▶ Laplace Approximation [MacKay, 1992];
 - ▶ Hamiltonian Monte Carlo [Neal, 1995];
 - ▶ (Stochastic) Variational Inference [Blundell, 2015].
- ▶ Most of the algorithms above are complicated both in theory and in practice.

How To Learn a Bayesian Neural Network?

What's the difficult part?

- ▶ $P(\mathbf{w}|y, \mathbf{x})$ is generally intractable
 - ▶ Standard **approximate inference** (difficult):
 - ▶ Laplace Approximation [MacKay, 1992];
 - ▶ Hamiltonian Monte Carlo [Neal, 1995];
 - ▶ (Stochastic) Variational Inference [Blundell, 2015].
- ▶ Most of the algorithms above are complicated both in theory and in practice.
- ▶ A simple and practical Bayesian neural network: dropout [Gal, 2016].

Dropout as Bayesian Approximation

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

- ▶ In classical neural network (**without** prediction **uncertainty**):

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

- ▶ In classical neural network (**without** prediction **uncertainty**):
 - ▶ During training: turn on dropout,

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

- ▶ In classical neural network (**without** prediction **uncertainty**):
 - ▶ During training: turn on dropout,
 - ▶ During prediction: **turn off** dropout.

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

- ▶ In classical neural network (**without** prediction **uncertainty**):
 - ▶ During training: turn on dropout,
 - ▶ During prediction: **turn off** dropout.
- ▶ In Bayesian neural network (**with** prediction **uncertainty**):

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

- ▶ In classical neural network (**without** prediction **uncertainty**):
 - ▶ During training: turn on dropout,
 - ▶ During prediction: **turn off** dropout.
- ▶ In Bayesian neural network (**with** prediction **uncertainty**):
 - ▶ During training: turn on dropout,

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

- ▶ In classical neural network (**without** prediction **uncertainty**):
 - ▶ During training: turn on dropout,
 - ▶ During prediction: **turn off** dropout.
- ▶ In Bayesian neural network (**with** prediction **uncertainty**):
 - ▶ During training: turn on dropout,
 - ▶ During prediction: **turn on** dropout.

Dropout as Bayesian Approximation

Dropout works by **randomly** setting network units to zero.

- ▶ In classical neural network (**without** prediction **uncertainty**):
 - ▶ During training: turn on dropout,
 - ▶ During prediction: **turn off** dropout.
- ▶ In Bayesian neural network (**with** prediction **uncertainty**):
 - ▶ During training: turn on dropout,
 - ▶ During prediction: **turn on** dropout.

We can obtain the distribution of prediction by repeating forward passing several times.

That's it!

Why Is That?

- ▶ **High-level idea:** Implement **variance inference** with a specific class of distributions $q_M(\omega)$ is **equivalent** to implement **dropout training**.

Why Is That?

- ▶ **High-level idea:** Implement **variance inference** with a specific class of distributions $q_M(\omega)$ is **equivalent** to implement **dropout training**.
- ▶ Optimizing **ELBO** in variance inference is the **same** as optimizing the **cost function** in dropout training.

Why Is That?

- ▶ **High-level idea:** Implement **variance inference** with a specific class of distributions $q_M(\omega)$ is **equivalent** to implement **dropout training**.
- ▶ Optimizing **ELBO** in variance inference is the **same** as optimizing the **cost function** in dropout training.
- ▶ The optimal **variational parameters** in variance inference is the **same** as the optimal **parameters** in dropout training.

Variational Inference

- ▶ We use a simple distribution $q_M(\omega)$ to approximate the true posterior distribution $p(\omega|y, X)$: $q_M(\omega) \approx p(\omega|y, X)$.

Variational Inference

- ▶ We use a simple distribution $q_M(\omega)$ to approximate the true posterior distribution $p(\omega|y, X)$: $q_M(\omega) \approx p(\omega|y, X)$.
- ▶ Minimize the $KL(q_M(\omega)|p(\omega|y, X))$ is equivalent to minimize the **negative** ELBO.

Variational Inference

- ▶ We use a simple distribution $q_M(\omega)$ to approximate the true posterior distribution $p(\omega|y, X)$: $q_M(\omega) \approx p(\omega|y, X)$.
- ▶ Minimize the $KL(q_M(\omega)|p(\omega|y, X))$ is equivalent to minimize the **negative** ELBO.

- ▶ **negative** ELBO:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega)|p(\omega)).$$

Variational Inference

- ▶ We use a simple distribution $q_M(\omega)$ to approximate the true posterior distribution $p(\omega|y, X): q_M(\omega) \approx p(\omega|y, X)$.
- ▶ Minimize the $KL(q_M(\omega)|p(\omega|y, X))$ is equivalent to minimize the **negative** ELBO.

- ▶ **negative** ELBO:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega)|p(\omega)).$$

- ▶ After optimization, prediction can be estimated by:

$$y' = f(\mathbf{x}'; \mathbf{w}), \mathbf{w} \sim q_M(\omega)$$

Compare Two Objective Functions

- ▶ negative ELBO:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega) | p(\omega)).$$

- ▶ Coss function:

$$L(W) = \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n, \mathbf{W}, \mathbf{z}_n))^2 + \lambda \sum_{i,j}^L (\|w_{i,j}\|^2).$$

Compare Two Objective Functions

- ▶ negative ELBO:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega)|p(\omega)).$$

- ▶ $q_M(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$

- ▶ Coss function:

$$L(W) = \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n, \mathbf{W}, \mathbf{z}_n))^2 + \lambda \sum_{i,j}^L (\|w_{i,j}\|^2).$$

Compare Two Objective Functions

- ▶ negative ELBO:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega)|p(\omega)).$$

- ▶ $q_M(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$

- ▶ The **loss functions** will be the same if we use Monte Carlo to simulate the integral. (reparameterization)

- ▶ Coss function:

$$L(W) = \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n, \mathbf{W}, \mathbf{z}_n))^2 + \lambda \sum_{i,j}^L (\|w_{i,j}\|^2).$$

Compare Two Objective Functions

- ▶ negative ELBO:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega)|p(\omega)).$$

- ▶ $q_M(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$

- ▶ The **loss functions** will be the same if we use Monte Carlo to simulate the integral. (reparameterization)

- ▶ $p(\omega) = N(\omega; 0, \mathbf{I})$

- ▶ Coss function:

$$L(W) = \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n, \mathbf{W}, \mathbf{z}_n))^2 + \lambda \sum_{i,j}^L (\|w_{i,j}\|^2).$$

Compare Two Objective Functions

- ▶ negative ELBO:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega) | p(\omega)).$$

- ▶ $q_M(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$

- ▶ The **loss functions** will be the same if we use Monte Carlo to simulate the integral. (reparameterization)

- ▶ $p(\omega) = N(\omega; 0, \mathbf{I})$

- ▶ The **regularizations** will be the same by using further approximation.

- ▶ Coss function:

$$L(W) = \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n, \mathbf{W}, \mathbf{z}_n))^2 + \lambda \sum_{i,j}^L (\|w_{i,j}\|^2).$$

I Know You Want Some Code

- ▶ Train one neural network (`network`) with dropout;

```
1  y = []
2  for _ in xrange(10):
3      network.train()
4      y.append(network(x))
5  y_mean = numpy.mean(y)
6  y_var = numpy.var(y)
```

I Know You Want Some Code

- ▶ Train one neural network (`network`) with dropout;
- ▶ Dropout units at prediction time;

```
1  y = []
2  for _ in xrange(10):
3      network.train()
4      y.append(network(x))
5  y_mean = numpy.mean(y)
6  y_var = numpy.var(y)
```

I Know You Want Some Code

- ▶ Train one neural network (`network`) with dropout;
- ▶ Dropout units at prediction time;
- ▶ Repeat several (10) times;

```
1  y = []
2  for _ in xrange(10):
3      network.train()
4      y.append(network(x))
5  y_mean = numpy.mean(y)
6  y_var = numpy.var(y)
```

I Know You Want Some Code

- ▶ Train one neural network (`network`) with dropout;
- ▶ Dropout units at prediction time;
- ▶ Repeat several (10) times;
- ▶ Look at the `mean` and `sample variance` of prediction.

```
1  y = []
2  for _ in xrange(10):
3      network.train()
4      y.append(network(x))
5  y_mean = numpy.mean(y)
6  y_var = numpy.var(y)
```

Results

Dataset	N	Q	Avg. Test RMSE and Std. Errors			Avg. Test LL and Std. Errors		
			VI	PBP	Dropout	VI	PBP	Dropout
Boston Housing	506	13	4.32 \pm 0.29	3.01 \pm 0.18	2.97 \pm 0.19	-2.90 \pm 0.07	-2.57 \pm 0.09	-2.46 \pm 0.06
Concrete Strength	1,030	8	7.19 \pm 0.12	5.67 \pm 0.09	5.23 \pm 0.12	-3.39 \pm 0.02	-3.16 \pm 0.02	-3.04 \pm 0.02
Energy Efficiency	768	8	2.65 \pm 0.08	1.80 \pm 0.05	1.66 \pm 0.04	-2.39 \pm 0.03	-2.04 \pm 0.02	-1.99 \pm 0.02
Kin8nm	8,192	8	0.10 \pm 0.00	0.10 \pm 0.00	0.10 \pm 0.00	0.90 \pm 0.01	0.90 \pm 0.01	0.95 \pm 0.01
Naval Propulsion	11,934	16	0.01 \pm 0.00	0.01 \pm 0.00	0.01 \pm 0.00	3.73 \pm 0.12	3.73 \pm 0.01	3.80 \pm 0.01
Power Plant	9,568	4	4.33 \pm 0.04	4.12 \pm 0.03	4.02 \pm 0.04	-2.89 \pm 0.01	-2.84 \pm 0.01	-2.80 \pm 0.01
Protein Structure	45,730	9	4.84 \pm 0.03	4.73 \pm 0.01	4.36 \pm 0.01	-2.99 \pm 0.01	-2.97 \pm 0.00	-2.89 \pm 0.00
Wine Quality Red	1,599	11	0.65 \pm 0.01	0.64 \pm 0.01	0.62 \pm 0.01	-0.98 \pm 0.01	-0.97 \pm 0.01	-0.93 \pm 0.01
Yacht Hydrodynamics	308	6	6.89 \pm 0.67	1.02 \pm 0.05	1.11 \pm 0.09	-3.43 \pm 0.16	-1.63 \pm 0.02	-1.55 \pm 0.03
Year Prediction MSD	515,345	90	9.034 \pm NA	8.879 \pm NA	8.849 \pm NA	-3.622 \pm NA	-3.603 \pm NA	-3.588 \pm NA

Table 1. Average test performance in RMSE and predictive log likelihood for a popular variational inference method (VI, Graves (2011)), Probabilistic back-propagation (PBP, Hernández-Lobato & Adams (2015)), and dropout uncertainty (Dropout). Dataset size (N) and input dimensionality (Q) are also given.

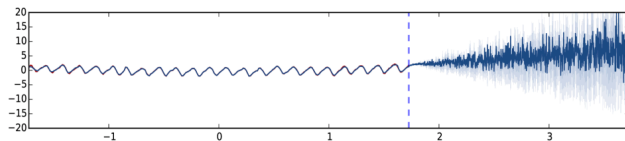


Figure 3. Predictive mean and uncertainties on the Mauna Loa CO₂ concentrations dataset for the MC dropout model with ReLU non-linearities, approximated with 10 samples.

Concrete Dropout

How To Choose Dropout Probability?

The simplest way is **Grid Search** (used in original paper)

- ▶ Problems:
 - ▶ Immense waste of computational resources
 - ▶ The number of possible per-layer dropout configurations grow exponentially as the number of the model layers increases.

How To Choose Dropout Probability?

The simplest way is **Grid Search** (used in original paper)

- ▶ Problems:
 - ▶ Immense waste of computational resources
 - ▶ The number of possible per-layer dropout configurations grow exponentially as the number of the model layers increases.
- ▶ One solution: Restrict the grid-search to a small number of possible dropout values
 - ▶ Might hurt uncertainty calibration.

More Elegant Method

Concrete Dropout

More Elegant Method

Concrete Dropout

- ▶ Tune dropout probability p_i using **gradient method**.

What Is The Optimisation Objective?

- ▶ The negative ELBO we used before:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega)|p(\omega)).$$

What Is The Optimisation Objective?

- ▶ The negative ELBO we used before:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + \text{KL}(q_M(\omega) | p(\omega)).$$

- ▶ Now, we almost use the same objective:

$$L(\theta) = - \frac{1}{M} \sum_{i \in S} \log p(y_i | X_i, \omega) + \text{KL}(q_\theta(\omega) | p(\omega)).$$

- ▶ $q_\theta(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$
- ▶ S a random set of M data points

What Is The Optimisation Objective?

- ▶ The negative ELBO we used before:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + \text{KL}(q_M(\omega) | p(\omega)).$$

- ▶ Now, we almost use the same objective:

$$L(\theta) = -\frac{1}{M} \sum_{i \in S} \log p(y_i | X_i, \omega) + \text{KL}(q_\theta(\omega) | p(\omega)).$$

- ▶ $q_\theta(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$
- ▶ S a random set of M data points
- ▶ $-\frac{1}{M} \sum_{i \in S} \log p(y_i | X_i, \omega)$ is the model's likelihood

What Is The Optimisation Objective?

- ▶ The negative ELBO we used before:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + KL(q_M(\omega)|p(\omega)).$$

- ▶ Now, we almost use the same objective:

$$L(\theta) = -\frac{1}{M} \sum_{i \in S} \log p(y_i|X_i, \omega) + KL(q_\theta(\omega)|p(\omega)).$$

- ▶ $q_\theta(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$
- ▶ S a random set of M data points
- ▶ $-\frac{1}{M} \sum_{i \in S} \log p(y_i|X_i, \omega)$ is the model's likelihood
- ▶ $KL(q_\theta(\omega)|p(\omega))$ is a "regularisation" term which ensure that the approximate posterior $q_\theta(\omega)$ does not deviate too far from the prior $p(\omega)$

What Is The Optimisation Objective?

- ▶ The negative ELBO we used before:

$$L(M) = - \int q_M(\omega) \log p(y|X, \omega) d\omega + \text{KL}(q_M(\omega)|p(\omega)).$$

- ▶ Now, we almost use the same objective:

$$L(\theta) = -\frac{1}{M} \sum_{i \in S} \log p(y_i|X_i, \omega) + \text{KL}(q_\theta(\omega)|p(\omega)).$$

- ▶ $q_\theta(\omega) = \prod_{i,j} q_{m_{i,j}}(\omega_{i,j}) = \prod_{i,j} m_{i,j} z_i$,
where $z_i \sim \text{Bernoulli}(1 - p_i)$
- ▶ S a random set of M data points
- ▶ $-\frac{1}{M} \sum_{i \in S} \log p(y_i|X_i, \omega)$ is the model's likelihood
- ▶ $\text{KL}(q_\theta(\omega)|p(\omega))$ is a "regularisation" term which ensure that the approximate posterior $q_\theta(\omega)$ does not deviate too far from the prior $p(\omega)$
- ▶ Except: $\theta = \{m_{i,j}, p_i\}$
 - ▶ This time, we try to optimize both weight $m_{i,j}$ and dropout probability p_i

How To Find The Optimal Parameter?

- ▶ Two methods are often adopted.

How To Find The Optimal Parameter?

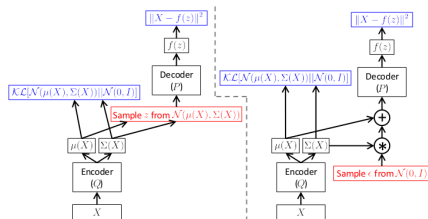
- ▶ Two methods are often adopted.
 - ▶ Score function estimator
the variance of gradient can be very high

How To Find The Optimal Parameter?

- ▶ Two methods are often adopted.
 - ▶ Score function estimator
the variance of gradient can be very high
 - ▶ Pathwise derivative estimator (also refer to reparameterization trick)

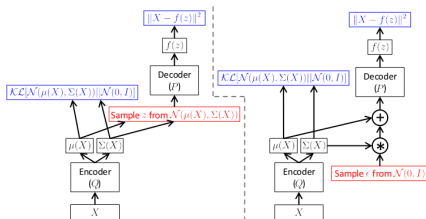
How To Find The Optimal Parameter?

- ▶ Two methods are often adopted.
 - ▶ Score function estimator
the variance of gradient can be very high
 - ▶ Pathwise derivative estimator (also refer to reparameterization trick)
- ▶ Recall the "reparameterization trick" used in VAE.



How To Find The Optimal Parameter?

- ▶ Two methods are often adopted.
 - ▶ Score function estimator
the variance of gradient can be very high
 - ▶ Pathwise derivative estimator (also refer to reparameterization trick)
- ▶ Recall the "reparameterization trick" used in VAE.



- ▶ Similarly, in order to train p_i , instead of sample from Bernoulli($1-p_i$), we sample from another distribution.

Reparameterization For Bernoulli Distribution

- ▶ When using **reparameterization trick**, we assume that the distribution at hand can be re-parametrised in the form $g(\theta, \epsilon)$
 - ▶ θ is the distribution's parameters
 - ▶ ϵ is a random variable which does not depend on θ

Reparameterization For Bernoulli Distribution

- ▶ When using **reparameterization trick**, we assume that the distribution at hand can be re-parametrised in the form $g(\theta, \epsilon)$
 - ▶ θ is the distribution's parameters
 - ▶ ϵ is a random variable which does not depend on θ
- ▶ But this cannot be simply done with the **discrete** Bernoulli distribution.

Reparameterization For Bernoulli Distribution

- ▶ When using **reparameterization trick**, we assume that the distribution at hand can be re-parametrised in the form $g(\theta, \epsilon)$
 - ▶ θ is the distribution's parameters
 - ▶ ϵ is a random variable which does not depend on θ
- ▶ But this cannot be simply done with the **discrete** Bernoulli distribution.
- ▶ **Concrete Distribution**
 - ▶ A continuous distribution used to approximate discrete random variables.

Reparameterization For Bernoulli Distribution

- ▶ When using **reparameterization trick**, we assume that the distribution at hand can be re-parametrised in the form $g(\theta, \epsilon)$
 - ▶ θ is the distribution's parameters
 - ▶ ϵ is a random variable which does not depend on θ
- ▶ But this cannot be simply done with the **discrete** Bernoulli distribution.
- ▶ **Concrete Distribution**
 - ▶ A continuous distribution used to approximate discrete random variables.
- ▶ Replace dropout's discrete Bernoulli distribution with its **continuous** relaxation.

Concrete Dropout

- ▶ Using the following function, we approximate Bernoulli distribution as concrete distribution:

$$z = \text{sigmoid}\left(\frac{1}{t} \cdot (\log p - \log(1 - p))\right) + \log u - \log(1 - u)$$

Concrete Dropout

- ▶ Using the following function, we approximate Bernoulli distribution as concrete distribution:

$$z = \text{sigmoid}\left(\frac{1}{t} \cdot (\log p - \log(1 - p))\right) + \log u - \log(1 - u)$$

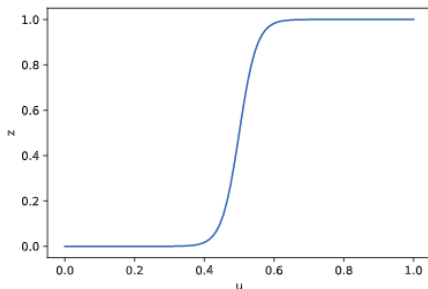
- ▶ Compared with Gaussian case:
 - ▶ Sample from $\epsilon \sim N(0, 1)$, $z \sim N(u, \sigma)$

Concrete Dropout

- ▶ Using the following function, we approximate Bernoulli distribution as concrete distribution:

$$z = \text{sigmoid}\left(\frac{1}{t} \cdot (\log p - \log(1 - p))\right) + \log u - \log(1 - u)$$

- ▶ Sample from $u \sim \text{Unif}(0, 1)$, $z \sim \text{Bern}(1 - p)$ (approximately)



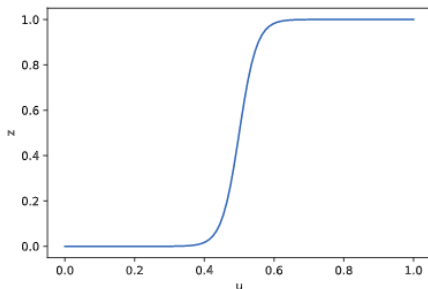
- ▶ Compared with Gaussian case:
 - ▶ Sample from $\epsilon \sim N(0, 1)$, $z \sim N(u, \sigma)$

Concrete Dropout

- ▶ Using the following function, we approximate Bernoulli distribution as concrete distribution:

$$z = \text{sigmoid}\left(\frac{1}{t} \cdot (\log p - \log(1 - p))\right) + \log u - \log(1 - u)$$

- ▶ Sample from $u \sim \text{Unif}(0, 1)$, $z \sim \text{Bern}(1 - p)$ (approximately)



- ▶ Compared with Gaussian case:
 - ▶ Sample from $\epsilon \sim N(0, 1)$, $z \sim N(u, \sigma)$
- ▶ Now, we have everything needed to train the model.

Result








Using concrete dropout, we can choose the dropout probability effectively, and also get a better performance.

DenseNet Model Variant	MC Sampling	IoU
No Dropout	-	65.8
Dropout (manually-tuned $p = 0.2$)	✗	67.1
Dropout (manually-tuned $p = 0.2$)	✓	67.2
Concrete Dropout	✗	67.2
Concrete Dropout	✓	67.4

The performance of Concrete dropout against base-line models with DenseNet on the CamVid road scene semantic segmentation dataset

Thanks for listening

References

-  MacKay, David JC. "A practical Bayesian framework for backpropagation networks." *Neural computation* 4.3 (1992): 448-472.
-  Neal, Radford M. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.
-  Blundell, Charles, et al. "Weight uncertainty in neural networks." *arXiv preprint arXiv:1505.05424* (2015).
-  Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." *international conference on machine learning*. 2016.
-  McAllister, Rowan, et al. "Concrete problems for autonomous vehicle safety: advantages of Bayesian deep learning." *International Joint Conferences on Artificial Intelligence, Inc.*, 2017.
-  Gal, Yarin, Riashat Islam, and Zoubin Ghahramani. "Deep bayesian active learning with image data." *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017.
-  Micheltore, Rhiannon, Marta Kwiatkowska, and Yarin Gal. "Evaluating Uncertainty Quantification in End-to-End Autonomous Driving Control." *arXiv preprint arXiv:1811.06817* (2018).