A Short Tutorial on Bayesian Deep Learning

Franois Darmon and Xu (Shell) Hu

Table of contents

The Basics of Bayesian Neural Networks

Bayesian inference

A brief introduction:

- Bayesians describe data $(\mathbf{x}, \mathbf{y})$ through the *latent variable model*

$$p(\mathbf{x}, \mathbf{y}, w) = p(\mathbf{y}|\mathbf{x}, w)p(\mathbf{x}|w)p(w) = p(w) \prod_i p(y_i|x_i, w)p(x_i|w)$$

  assuming the *likelihood* $p(\mathbf{y}|\mathbf{x}, w)$ and the *prior* $p(w)$ are given.

- Bayesians make predictions according to

$$p(y_{\text{new}}|x_{\text{new}}, \mathbf{x}, \mathbf{y}) = \int p(y_{\text{new}}|x_{\text{new}}, w)p(w|\mathbf{x}, \mathbf{y})dw,$$

  where $p(w|\mathbf{x}, \mathbf{y})$ is the *posterior*.

- Bayesians perform *inference* by obtaining a *variational posterior*

$$q^* = \arg\min_q \ \mathrm{divergence}\big(q(w) \| p(w|\mathbf{x}, \mathbf{y})\big)$$

Mathematical Formulation for Optimization Problem

- Optimization on variational posterior parameters is minimization on KL divergence written as following:

$$\begin{aligned}
\theta^* &= argmin_\theta KL[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathfrak{D})] && (1) \\
&= argmin_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w}P(\mathfrak{D}|\mathbf{w}))} && (2) \\
&= argmin_\theta \mathbf{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathfrak{D}|\mathbf{w})] && (3)
\end{aligned}$$

- The paper proposes gradient descent based optimization on above expression through the methods shown in following slides, without need for computing closed formed KL terms.
- This relaxes restriction on prior and posterior forms of selection.

Mean field approximation

- $q(\boldsymbol{\beta}) = \prod_{i=1}^{W} q_i(\boldsymbol{\beta}_i) \implies L^C(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i}^{W} \mathbb{D}_{KL}(q_i(\boldsymbol{\beta}_i)|p(\boldsymbol{\alpha}))$
- SGD affected by choice of posterior $q(\boldsymbol{\beta})$ and prior $p(\boldsymbol{\alpha})$
- Delta Posterior
    - $L^C(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\log\left(p(\mathbf{w}|\boldsymbol{\alpha})\right) + C$
    - Uniform prior $\implies$ MLE
    - Laplace prior $\implies$ L1 regularisation
    - Gaussian prior $\implies$ L2 regularisation
- Diagonal Gaussian Posterior
    - Uniform prior $\implies$ weight noise
    - Gaussian prior $\implies$ adaptive weight noise

Defining the Objective Function for Optimization

- With optimization addressing minimizing KL divergence as defined previously, would like to reformulate it into a convenient choice of objective function that's easy to optimize
- Recap the optimization as parameters minimization on the KL divergence between variational posterior and actual posterior as our cost function:

$$\theta^* = argmin_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathfrak{D}|\mathbf{w})} d\mathbf{w} \tag{4}$$

- Define the objective function $f((w), \theta)$ as the component being taken expectation of:

$$f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathfrak{D}|\mathbf{w}) \tag{5}$$

- With substituting in this cost function notation, the KL divergence minimization problem becomes:

$$\theta^* = argmin_\theta \mathbb{E}_{q(\mathbf{w}|\theta)} f(\mathbf{w}, \theta) \tag{6}$$

Algorithm Steps for Optimization with Variational Inference on Weights Posteriors

- With the previous problem reformulation, utilizing Monte Carlo estimates, the detailed algorithm steps for optimizing variational posterior parameters are as following:

1. Sample $\epsilon\ \mathcal{N}(0, I)$.
2. Let $\mathbf{w} = \mu + \log(1 + exp(\rho)) \odot \epsilon$ (with $\odot$ denoting element-wise multiplication)
3. Let $\theta = (\mu, \rho)$
4. Let $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathfrak{D}|\mathbf{w})$.

Stochastic Optimisation

Common gradient problem

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int q_\phi(\mathbf{z}) f_\theta(\mathbf{z}) d\mathbf{z}$$

- Don't know this expectation in general.
- Gradient is of the parameters of the distribution w.r.t. which the expectation is taken.

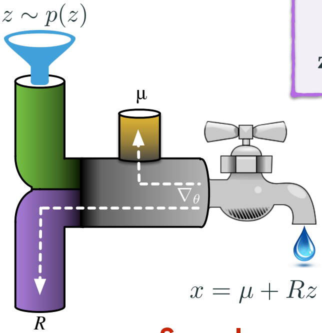
1. **Pathwise estimator**: Differentiate the function **f(z)**
2. **Score-function estimator**: Differentiate the density **q(z|x)**

Typical problem areas

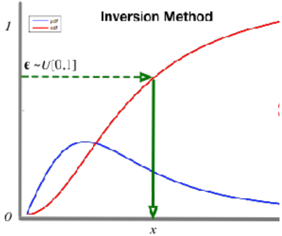
- Sensitivity analysis
- Generative models and inference
- Reinforcement learning and control
- Operations research and inventory control
- Monte Carlo simulation
- Finance and asset pricing

Reparameterisation Tricks

Distributions can be expressed as a transformations of other distributions.



$$z \sim q_\phi(\mathbf{z})$$
$$\mathbf{z} = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon)$$



Samplers, one-liners and change-of-variables

$$p(z) = \left|\frac{d\epsilon}{dz}\right| p(\epsilon) \implies |p(z)dz| = |p(\epsilon)d\epsilon|$$

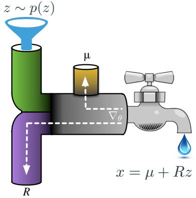
Pathwise Estimator

(Non-rigorous) Derivation

$$\nabla_\phi \mathbb{E}_{q(z)}[f(\mathbf{z})] = \nabla_\phi \int q_\phi(\mathbf{z}) f(\mathbf{z}) d\mathbf{z}$$

Known transformation

$z = g(\epsilon, \phi); \quad \epsilon \sim p(\epsilon)$



$$= \nabla_\phi \int p(\boldsymbol{\epsilon}) \frac{d\boldsymbol{\epsilon}}{d\mathbf{z}} f(g(\boldsymbol{\epsilon}, \phi)) g'(\epsilon, \phi) d\boldsymbol{\epsilon}$$

Change of variables

$$= \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(g(\phi, \epsilon)] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(g(\phi, \epsilon)]$$

Inv fn Thm

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f_\theta(g(\epsilon, \phi))]$$

Other names

- Unconscious statistician
- Stochastic backpropagation
- Perturbation analysis
- Reparameterisation trick
- Affine-independent inference

When to use

- Function *f* is differentiable
- Density *q* is known with a suitable transform of a simpler base distribution: inverse CDF, location-scale transform, or other co-ordinate transform.
- Easy to sample from base distribution.

Log-derivative Trick

Score function is the derivative of a log-likelihood function.

$$\nabla_\phi \log q_\phi(\mathbf{z}) = \frac{\nabla_\phi q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})}$$

Several useful properties

Expected score

$$\mathbb{E}_{q(z)}\left[\nabla_\phi \log q_\phi(\mathbf{z})\right] = 0$$

↞Show this

$$\mathbb{E}_{q(z)}[\nabla_\phi \log q_\phi(\mathbf{z})] = \int q(z) \frac{\nabla_\phi q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})} = \nabla \int q_\phi(\mathbf{z}) = \nabla 1 = 0$$

Fisher Information

$$\mathbb{V}[\nabla_\theta \log p(\mathbf{x};\theta)] = \mathcal{I}(\theta) = \mathbb{E}_{p(x;\theta)}[\nabla_\theta \log p(\mathbf{x};\theta) \nabla_\theta \log p(\mathbf{x};\theta)^\top]$$

Score-function Estimator

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int q_\phi(\mathbf{z}) f_\theta(\mathbf{z}) d\mathbf{z}$$

Leibnitz integral rule

$$= \int \frac{q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})} \nabla_\phi q_\phi(\mathbf{z}) f(\mathbf{z}) d\mathbf{z}$$

Identity

$$= \int q_\phi(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z}) f(\mathbf{z}) d\mathbf{z}$$

Log-deriv

$$= \mathbb{E}_{q_\phi(\mathbf{z})} [f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})]$$

Gradient

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \mathbb{E}_{q_\phi(\mathbf{z})} [(f(\mathbf{z}) - c) \nabla_\phi \log q_\phi(\mathbf{z})]$$

Control Variate

Other names
- Likelihood ratio method
- REINFORCE and policy gradients
- Automated & Black-box inference

When to use
- Function is not differentiable, not analytical.
- Distribution *q* is easy to sample from.
- Density *q* is known and differentiable.

Variational Dropout and the Local Reparametrization trick [2]

Variational inference (remainder)

- Each weight of the NN is a random variable with a prior distribution $p(w)$
- Posterior distribution of the weights $p(w|\mathcal{D})$ cannot be computed, we need to approximate it with an easier distribution $q_\phi(w)$
- Finding the best ϕ so that $D_{KL}(q_\phi(w)||p(w|\mathcal{D}))$ is minimized is equivalent to maximize

$$\mathcal{L}(\phi) = D_{KL}(q_\phi(w)||p(w)) + \sum_{x,y \in \mathcal{D}} \mathbb{E}_{q_\phi(w)}[\log p(y|x,w)] \quad (1)$$

Local reparametrization trick

- Linear layer: $B = AW$, with A a $(M \times K)$ input matrix, B a $(M \times L)$ output matrix and W a $(K \times L)$ weight matrix.
- Posterior approximation on the weights: each w_{ij} is independent and $q_\phi(w_{ij}) = \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$
- Minibatch gradient descent, sample M *independent* weight matrices W^i from q_ϕ and $b_i^T = W^i a_i^T$, **not efficient**
- Sample directly from B with

$$q_\phi(b_{mj}|A) = \mathcal{N}((A\mu)_{mj}, (A^2\sigma^2)_{mj}) \tag{2}$$

$\longrightarrow$ Sample $(M \times L)$ values instead of $(M \times K \times L)$ and easy parallel implementation

Dropout

- Method used for **regularization** of deep networks
- At train time: Randomly select with probability p inputs of each layer and set them to 0

$$B = (A \circ \xi)\theta$$

  with ξ random noise matrix and θ weight matrix
- At test time: Use all inputs ($p = 1$)

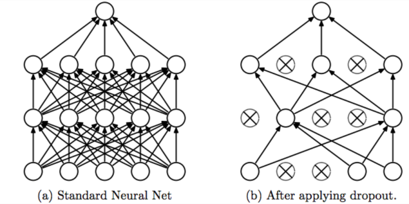


Figure 1: Dropout principle

Dropout as a variational method

Dropout with continuous noise $\mathcal{N}(1, \alpha)$ can be seen as a variational method with local reparametrization trick:

- Posterior approximation of weights:

$$W = \theta \text{diag}(s)$$

  with $q_\phi(s_i) = \mathcal{N}(1, \alpha)$.
- Variational parameter ϕ can be decomposed in two terms: $\phi = (\theta, \alpha)$ with θ mean of the weights and α variance of dropout.
- Variational lower bound:

$$\mathcal{L}(\phi) = \underbrace{-D_{KL}(q_\phi(w)||p(w))}_{\text{Can be made independent of } \theta} + \underbrace{\sum_{x,y \in \mathcal{D}} \mathbb{E}_{q_\phi(w)}[\log p(y|x, w)]}_{\text{Training objective of usual dropout neural network}} \tag{3}$$

Examples in Computer Vision [1]

Motivations

Classical tasks of compputer vision: semantic segmentation, monocular depth estimation.

Bayesian DL can be a way to:

- Learn from noisy data
- Learn from small datasets
- **Have a confidence score of the predictions**

Sources of uncertainties

One can define two sources of uncertainties in computer vision.

- Epistemic:
  Uncertainty on which model has generated the data. Given enough training data, it could be reduced to 0
- Aleatoric:
  Noise inherent in the observations (sensor noise, ambiguous class ...). It cannot be explained away with more training data.

Epistemic

It can be modelled with Bayesian Networks

$$p(y|x, \mathcal{D}) = \mathbb{E}_{p(w|\mathcal{D})} p(x|y, w) \tag{4}$$

Sample $w_1..w_N$ from $q_\phi(w)$ and compute $p_k = p(x|y, w_k)$. The mean and variance of the prediction can be estimated with the mean and variance of this sequence.

Aleatoric

For regression tasks:

- Predict parameters of a distribution instead of single value
- Use minus log likelihood as loss function
- Examples:
  - Predict **Gaussian distribution** with mean f and variance σ

$$l((f, \sigma), \hat{f}) = -\frac{||f - \hat{f}||^2}{2\sigma^2} - \log(\sigma)$$

  - Predict **Laplacian distribution** with mean f and variance σ

$$l((f, \sigma), \hat{f}) = -\frac{|f - \hat{f}|}{\sigma} - \log(\sigma)$$

Example semantic segmentation

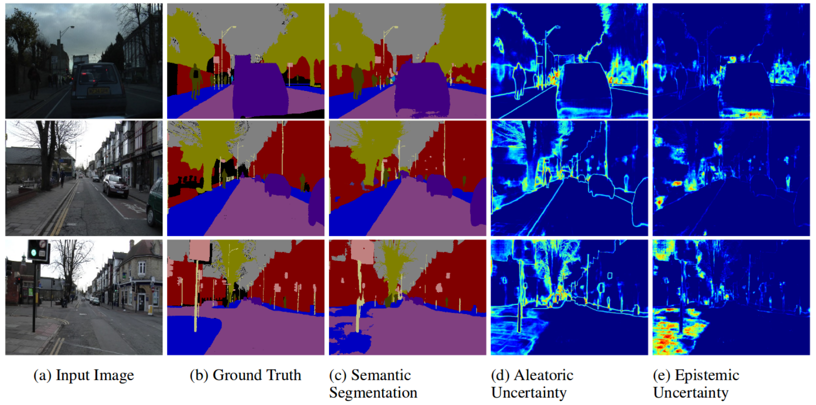


Figure 2: Example of semantic segmentation

Aleatoric vs Epistemic uncertainties

Train dataset	Test dataset	RMS	Aleatoric variance	Epistemic variance
Make3D / 4	Make3D	5.76	0.506	7.73
Make3D / 2	Make3D	4.62	0.521	4.38
Make3D	Make3D	3.87	0.485	2.78
Make3D / 4	NYUv2	-	0.388	15.0
Make3D	NYUv2	-	0.461	4.87

(a) Regression

Train dataset	Test dataset	IoU	Aleatoric entropy	Epistemic logit variance ($\times 10^{-3}$)
CamVid / 4	CamVid	57.2	0.106	1.96
CamVid / 2	CamVid	62.9	0.156	1.66
CamVid	CamVid	67.5	0.111	1.36
CamVid / 4	NYUv2	-	0.247	10.9
CamVid	NYUv2	-	0.264	11.8

(b) Classification

Figure 3: Influence of the dataset size and of new dataset on the two categories of uncertainty measure

Questions?

A. Kendall and Y. Gal.
What uncertainties do we need in bayesian deep learning for computer vision?
In *Advances in neural information processing systems*, pages 5574–5584, 2017.

D. P. Kingma, T. Salimans, and M. Welling.
Variational dropout and the local reparameterization trick.
In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.