Bayesian Deep Neural Networks

Bayesian Neural Networks

Sungjoon Choi

February 9, 2018

Seoul National University

Table of contents

Introduction

Introduction

- A neural network with a single hidden layer with N_H units:

$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j) \in \mathbb{R}$$

  where $\mathbf{x}$ is an input, $h(\cdot; \mathbf{u}_j)$ is j-th hidden node.
- It is known that a neural network one hidden layer is a **universal approximator** as $N_H \to \infty$ [1].
  - It can approximate any smooth function on a compact support under mild conditions.
  - This property also holds for a polynomial function or Gaussian process regression.

Introduction

- Neural network converges to a Gaussian process:

$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j) \in \mathbb{R}$$

- Suppose that
    - $b \sim \mathcal{N}(0, \sigma_b^2)$ and $v_j \sim \mathcal{N}(0, \sigma_v^2)$
    - $\mathbf{u}_j$ are independently and identically distributed
    - σ_v^2 scales as w^2/N_H then

$$\begin{aligned}\mathbb{E}(f(\mathbf{x})) &= 0 \\ \mathbb{E}(f(\mathbf{x})f(\mathbf{x}')) &= \sigma_b^2 + \sum_{j=1}^{N_H} \sigma_v^2 \mathbb{E}_u(h(\mathbf{x}; \mathbf{u}_j)h(\mathbf{x}'; \mathbf{u}_j)) \\ &= \sigma_b^2 + w^2 \mathbb{E}_u(h(\mathbf{x}; \mathbf{u}_j)h(\mathbf{x}'; \mathbf{u}_j))\end{aligned}$$

- By the central limit theorem, $f(\mathbf{x})$ converges to a Gaussian process as $N_H \to \infty$.

Introduction

- Suppose that
  - $b \sim \mathcal{N}(0, \sigma_b^2)$ and $v_j \sim \mathcal{N}(0, \sigma_v^2)$
  - $\mathbf{u}_j$ are independently and identically distributed
  - σ_v^2 scales as w^2/N_H then

$$\begin{aligned}\mathbb{E}(f(\mathbf{x})) &= 0 \\ \mathbb{E}(f(\mathbf{x})f(\mathbf{x}')) &= \sigma_b^2 + \sum_{j=1}^{N_H} \sigma_v^2 \mathbb{E}_u(h(\mathbf{x}; \mathbf{u}_j)h(\mathbf{x}'; \mathbf{u}_j)) \\ &= \sigma_b^2 + w^2 \mathbb{E}_u(h(\mathbf{x}; \mathbf{u}_j)h(\mathbf{x}'; \mathbf{u}_j))\end{aligned}$$

- By the central limit theorem, $f(\mathbf{x})$ converges to a Gaussian process as $N_H \to \infty$.

Introduction

- If $h(\mathbf{x}; \mathbf{u}) = \text{erf}(u_0 + \sum_{j=1}^{N_H} u_j x_j)$ and $\mathbf{u} \sim \mathcal{N}(0, \Sigma)$, then the covariance function of the neural network is

$$k_{NN}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\bar{\mathbf{x}}^T \Sigma \bar{\mathbf{x}}'}{\sqrt{(1 + 2\bar{\mathbf{x}}^T \Sigma \bar{\mathbf{x}})(1 + 2\bar{\mathbf{x}}'^T \Sigma \bar{\mathbf{x}}'}} \right)$$

where $\bar{\mathbf{x}} = [1, x_1, \ldots, x_d]^T$.

First four papers are optimizing the SAME objectives.

Minimizing the Description Length

Minimizing the Description Length

- Geoffrey Hinton and Drew van Camp, 1993

Minimizing the Description Length

- Minimum description length principle
    - When fitting models to data, it is alway possible to fit the training data better by using a more complex model.
    - The minimum description length principle asserts the the best model is the one that minimizes the combined cost of:
        1. describing the model
        2. describing the misfit between the model and the data
    - One can think in terms of:
        1. a sender who can see both the input and output
        2. a receiver who can only see the input.

Minimizing the Description Length

1. The sender first fits a neural network to the complete set of training data.
2. Then sends the weights of the network to the receiver.
3. For each training data, the sender also sends the discrepancy between the network output and the correct output.
4. By adding the discrepancy to the output of the network, the receiver can generate exactly the correct output.

Minimizing the Description Length

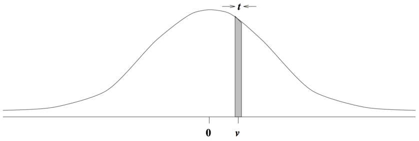
- Total cost $=$ coding the data misfits $+$ coding the weights

Minimizing the Description Length

- **Coding the data misfits**
  - If the data misfits are real numbers, an infinite amount of information is needed.
  - So we assume that they are finely quantized using intervals of fixed width t.
  - The **coding theorem** tells us that:
    - if a sender and a receiver agreed on a probability distribution that assigns a probability mass $p(\Delta y)$ to each possible quantized data misfit,
    - then we can code the misfit using $-\log_2 p(\Delta y)$ bits.
    - Example) If Δy quantizes the misfit into eight parts and the probability mass is assigned equally, then $p(\Delta y) = \frac{1}{8}$ and we need $-\log_2(\frac{1}{8}) = 3$ bits.

Minimizing the Description Length

- **Coding the data misfits**



  - We assume that the data misfits are encoded by assuming that they are drawn from a zeros-mean Gaussian:

$$p(d^c - y^c) = t\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[\frac{-(d^c - y^c)^2}{2\sigma^2}\right].$$

  - Using an optimal code, the **description length** of a data misfit, $d - y$, in units of $log_2(\cdot)$ bits (called 'nats') is:

$$-\log p(d^c - y^c) = -\log t + \log\sqrt{2\pi} + \log\sigma + \frac{(d^c - y^c)^2}{\sigma^2}$$

Minimizing the Description Length

- **Coding the data misfits**
  - Using an optimal code, the **description length** of a data misfit, $d - y$, in units of $log_2(\cdot)$ bits (called 'nats') with N data is:

$$\mathrm{DL} = -N \log t + N \log \sigma + \sum_c \frac{(d^c - y^c)^2}{\sigma^2}$$

  - σ^* that minimizes DL is $\sigma^* = \sqrt{\frac{1}{N} \sum_c \frac{(d^c - y^c)^2}{\sigma^2}}$.
  - Substituting σ^*, we get

$$\mathrm{DL}^* = -N \log t + \frac{N}{2} \log \left[\frac{1}{N} \sum_c (d^c - y^c)^2 \right] + \frac{N}{2}$$

  - The final data misfit cost becomes:

$$\mathrm{C}_{\text{data-misfit}} = kN + \frac{N}{2} \log \left[\frac{1}{N} \sum_c (d^c - y^c)^2 \right]$$

  where k is a constant that depends on t.

Minimizing the Description Length

- **Coding the weights**
  - We assume that the weights of the network are finely quantized and come from a zero-mean Gaussian distribution

$$p(w) = t \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp(-\frac{w^2}{2\sigma_w^2}) :$$

  - Then, the description length of the weights becomes:

$$\mathrm{DL} = \frac{1}{2\sigma_w^2} \sum_j w_j^2$$

Minimizing the Description Length

- **Total description length**
  - The total description length (TDL) is the sum of
    1. **coding the data misfits**
    2. **coding the weights**.
  - Ignoring k which related to the quantization level t, we get

$$TDL = \sum_i \frac{1}{\sigma^2} \sum_c (d_i^c - y_i^c)^2 + \frac{1}{2\sigma_w^2} \sum_j w_j^2 .$$

- Surprisingly, the is just the standard **weight-decay** method.

Noisy weights

- **The ’bits back’ argument**
  - Suppose that the sender and receiver have an agreed Gaussian prior $P(w)$.
  - After learning, the sender has a Gaussian posterior $Q(w)$.
  - We will show that the amount of required bits to communicate the posterior distribution of a weight is equal to the KL-distance from P to Q:

$$G(P, Q) = \int Q(w) \log \frac{Q(w)}{P(w)} dw.$$

  - Note that $D_{KL}(P||Q) = \int p(x) \log \frac{p(x)}{q(x)} dx$.

Noisy weights

- **The 'bits back' argument**
  1. To communicate the noisy weights, the sender first collapses the weights drawn from the posterior $Q(w)$ to pick a precise value within the tolerance t.
  2. Then the sender sends each weight for $Q(w)$ by coding them using the prior $P(w)$ where the communication cost is:
  $$C(w) = \log t - \log P(w)$$
  where $\log t$ is for quantization and $\log P(w)$ is for coding w with the prior. The sender also communicates the data-misfits.
  3. Since the receiver has correct output and misfits, he can also recover the exact same posterior probability $Q(w)$. Hence, following information is redundant:
  $$R(w) = \log t \log Q(w)$$
  4. So the true expected description length is determined by taking an expectation under $Q(w)$:
  $$G(P, Q) = \int Q(w) \log \frac{Q(w)}{P(w)} dw.$$

Noisy weights

- **The 'bits back' argument**
    - **The 'bits back' argument** tells us that when we have a prior $P(w)$ on weights and the sender has a posterior $Q(w)$, the expected description length for a noisy (random) weights is:

      $$G(P, Q) = \int Q(w) \log \frac{Q(w)}{P(w)} dw$$

      where $G(P, Q) = D_{KL}(Q||P)$.
    - Note that the goal of most of variational inference problems for Bayesian neural networks is to find $Q(w)$ that minimizes $D_{KL}(Q||P)$ given some prior $P(w)$.

Ensemble Learning in Bayesian Neural Network

Ensemble Learning

- **Bayesian neural network**
  - Consider a two-layer network with H hidden units:
$$f(x,w) = \sum_{i-1}^{H} v_i \rho(u^T x)$$
where $\rho(a)$ is the 'erf' (cumulative Gaussian) function
$$\rho(a) = \sqrt{\frac{2}{\pi}} \int_0^a \exp(-s^2/2) ds.$$
  - We assume the output has a measurement noise whose variance is β^{-1} and the likelihood of w and β becomes:
$$P(D|w,\beta) = \frac{\exp(-\beta E_D)}{Z_D}$$
where Z_D is a normalizing factor and E_D is the training error:
$$E_D(w) = \frac{1}{2} \sum_i (f(x^i, w) - t^i)^2.$$

Ensemble Learning

- **Bayesian neural network**
  - Likelihood:
$$P(D|w,\beta) = \frac{\exp(-\beta E_D)}{Z_D}$$
    where Z_D is a normalizing factor and E_D is the training error:
$$E_D(w) = \frac{1}{2}\sum_i (f(x^i, w) - t^i)^2.$$
  - Prior:
$$p(w|A) = \frac{\exp(-E_W(w))}{Z_P}$$
    where $E_W(w) = \frac{1}{2} w^T A w$.
  - Posterior:
$$p(w|D,\beta,A) = \frac{1}{Z_F}\exp(\beta E_D(w) - E_W(w)).$$

Ensemble Learning

- **Bayesian neural network**
  - Given a posterior distribution:

$$p(w|D, \beta, A) = \frac{1}{Z_F} \exp(-\beta E_D(w) - E_W(w)),$$

    the prediction for a new input are given by **integration** over the posterior distribution.
  - For example, the predictive mean is given by

$$\langle f(x) \rangle = \int f(x, w) P(w|D, \beta, A) dw.$$

Ensemble Learning

- **Laplace's method**
  - We approximate the posterior distribution with a Gaussian distribution:
    $$p(w|D,\beta,A) = \exp(-\phi(w)).$$
  - Then, expand ϕ around the **mode** of the distribution
    $$w_* = \arg\min \phi(w)$$
    so that
    $$\phi(w) \approx \phi(w_*) + \frac{1}{2}(w - w_*)^T H(w - w_*)$$
    where $H = \nabla\nabla\phi(W)|_{w_*}$.
  - The expected value of f, $\langle f(x)\rangle = \int f(x,w)P(w|D,\beta,A)dw$ can be evaluated by making a further local linearization of $f(x,w)$.

Ensemble Learning

- **Markov chain Monte Carlo method**
  - The central idea is to replace integrals weighted by the posterior by finite sums, so that
    $$\int p(w|D,\beta,A)g(w)dw \approx \frac{1}{m}\sum_{i=1}^{m} g(w_i)$$
    where w_i are drawn from the posterior.
  - Examples of MCMC:
    - Metropolis-Hastings algorithm: generates a random walk using a function that is **proportional** to the proposal density.
    - Gibbs sampling: Sample from the conditional distributions. For example, if we want to sample from $p(x_1, x_2)$, use $p(x_1|x_2)$ and $p(x_2|x_1)$.

Ensemble Learning

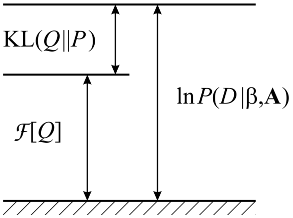
- **Ensemble learning**
  - This method is called a variational inference, nowadays.
  - We introduce a variational distribution $Q(w)$ that approximates the posterior.
  - We start with a marginal likelihood:

$$\begin{aligned}\ln p(D|\beta, A) &= \ln \int p(D|w,\beta)p(w|A)dw \\ &= \ln \int \frac{p(D|w,\beta)p(w|A)}{Q(w)} Q(w)dw \\ &\geq \int \ln \left\{ \frac{p(D|w,\beta)p(w|A)}{Q(w)} \right\} Q(w)dw \\ &= \mathcal{F}[Q].\end{aligned}$$

  - $\mathcal{F}[Q]$ only requires likelihood $p(D|w,\beta)$ and prior $p(w|A)$.

Ensemble Learning

- **Ensemble learning**



  - $\mathcal{F}[Q]$ is a lower bound of log marginal likelihood $\ln p(D|\beta, A)$.

Ensemble Learning

- **Ensemble learning**
  - Show that $\ln p(D|\beta, A) = KL(Q||P) + \mathcal{F}[Q]$:
    1. $\ln p(D|\beta, A) = KL(Q||P) + \int \ln \left\{ \frac{p(D|w,\beta)p(w|A)}{Q(w)} \right\} Q(w)dw$
    2. $\quad = -\int \ln \frac{p(w|D,\beta,A)}{Q(w)} Q(w)dw + \int \ln \left\{ \frac{p(D|w,\beta)p(w|A)}{Q(w)} \right\} Q(w)dw$
    3. $\quad = \int Q(w) \left[\ln \frac{p(D|w,\beta)p(w|A)}{p(w|D,\beta,A)} \right] dw$
    4. $\quad = \int Q(w) \left[\ln \frac{p(D,w|\beta,A)}{p(w|D,\beta,A)} \right] dw$
    5. $\quad = \int Q(w) \left[\ln \frac{p(w|D,\beta,A)p(D|\beta,A)}{p(w|D,\beta,A)} \right] dw$
    6. $\quad = \int Q(w) \ln p(D|\beta, A) dw$
    7. $\quad = p(D|\beta, A)$
  - More fancy if we go the reverse direction.

Ensemble Learning

- **Ensemble learning**
  - What we have seen so far is that if we maximize $\mathcal{F}[Q]$ then it is equivalent to saying that we are maximizing the lower bound of the marginal log likelihood $\ln p(D|\beta, A)$.
  - $\mathcal{F}[Q]$ is often calls evidence lower bound (ELBO) and heavily used in formulating variational auto-encoders (VAEs).
  - Furthermore,

$$\begin{aligned}
\mathcal{F}[Q] &= \int \ln \left\{ \frac{p(D|w,\beta)p(w|A)}{Q(w)} \right\} Q(w)dw \\
&= \int \left(\ln p(D|w,\beta) + \ln \frac{p(w|A)}{Q(w)} \right) Q(w)dw \\
&= \int \ln p(D|w,\beta)Q(w)dw - D_{KL}(Q(w)||p(w|A)
\end{aligned}$$

  - As our goal is to maximize $\mathcal{F}[Q]$, we are trying to
    1. maximize the marginal likelihood$p(D)$
    2. reduce the gap between $p(w|D)$ and $q(w)$.
    3. keep the variational distribution $Q(w)$ close to our prior $p(w|A)$.

Ensemble Learning

- **Ensemble Learning**
  - In this paper, $Q(w)$ is a Gaussian with mean $\bar{w}$ and covariance C.
  - Main contribution of this paper is to find an efficient update rule for optimizing $Q(w)$ w.r.t. $\bar{w}$ and C using matrix calculus and the fact that $\rho(a)$ is the 'erf' function.
  - Note that the previous paper (Hinton and Camp, 1993) assumes that C is a diagonal matrix.

Practical variational inference

Practical variational inference

- **Introduction**
  - "In the eighteen years since **variational inference** was first proposed for neural networks (Hinton and Camp, 1993) it has not seen widespread use.
  - We believe this is largely due to the difficulty of deriving analytical solutions to the required integrals over the variational posteriors."
  - "The approach taken here is to forget about analytical solutions and search instead for variational distributions whose expectation values (and derivatives thereof) can be efficiently approximated with numerical integration."

Practical variational inference

- **Variational inference**
  - The goal of the variational inference is to fit $Q(w)$ by minimizing **variational free energy** $\mathcal{F}]Q]$ w.r.t. parameter β:

$$\mathcal{F} = -\left\langle \ln\left[\frac{p(D|w)p(w|\alpha)}{Q(w|\beta)}\right]\right\rangle_{w\sim Q(\beta)}$$

Practical variational inference

- **Minimum description length**
  - The variational free energy (to be minimized) can be reinterpreted as:

$$\begin{aligned}\mathcal{F} &= -\left\langle \ln \left[\frac{p(D|w)p(w|\alpha)}{Q(w|\beta)} \right] \right\rangle_{w \sim Q(\beta)} \\ &= \langle L^N(w, D) \rangle_{w \sim Q(\beta)} + D_{KL}(Q(\beta) || P(\alpha))\end{aligned}$$

    where $L^N = -\ln p(D|w) = -\sum_{(x,y) \in D} \ln p(y|x, w)$ is the negative log likelihood.
  - The first term in $\mathcal{F}$ is called the error loss:

$$L^E(\beta, D) = \langle L^N(w, D) \rangle_{w \sim Q(\beta)}.$$

  - The second term in $\mathcal{F}$ is called the complexity loss:

$$L^C(\alpha, \beta) = D_{KL}(Q(\beta) || P(\alpha)).$$

    which can be realized with **bits-back coding**.

Practical variational inference

- **Total loss**
  - The total loss is:

$$L(\alpha, \beta, D) = L^E(\beta, D) + L^C(\alpha, \beta)$$

    where $L^E(\beta, D) = \langle L^N(w, D) \rangle_{w \sim Q(\beta)}$ and
    $L^C(\alpha, \beta) = D_{KL}(Q(\beta) || P(\alpha))$.
  - The negative log likelihood L^N is:

$$L^N(w, D) = -\ln p(D|w) = -\sum_{(x,y) \in D} \ln p(y|x, w)$$

Practical variational inference

- **Choice of distributions**
  - Delta posterior
    - The simplest posterior $Q(\beta)$ that assigns probability 1 to a particular set of weights w.
    - We recover ordinary maximum likelihood training.
    - If the prior is a Laplace distribution with $\mu = 0$, then we have L1 regularization.
    - If the prior is a Gaussian distribution with $\mu = 0$, then we have L2 regularization.

Practical variational inference

- **Choice of distributions**
  - Gaussian posterior
    - Suppose $Q(\beta) \sim \mathcal{N}(\mu, \sigma)$.
    - We approximate $L^E(\beta, D)$ as:

$$L^E(\beta, D) \approx \frac{1}{S} \sum_{k=1}^{S} L^N(w^k, D)$$

      where w^k drawn independently from $Q(\beta)$.
    - We also utilize following identities of Gaussian differentiations:

$$\nabla_\mu \langle V(a) \rangle_{a \sim \mathcal{N}} = \langle \nabla_a V(a) \rangle_{a \sim \mathcal{N}} \text{ and } \nabla_\Sigma \langle V(a) \rangle_{a \sim \mathcal{N}} = \frac{1}{2} \langle \nabla_a \nabla_a V(a) \rangle_{a \sim \mathcal{N}}$$

Practical variational inference

- This paper is the first paper to formally use a variational inference on Bayesian neural networks.
- This paper assumes that the posterior distribution is a Gaussian distribution where the covariance matrix is diagonal and presented efficient gradients for optimizing the mean and variance.

Bayes by backprop

Bayes by backprop

- **Weight uncertainty** in neural networks
  - All weights in the neural networks are represented by probability distributions over passible values, rather than having a single fixed value.
  - Instead of training a single network, the propose method trains an ensemble of networks, where each network has its weights drawn from a shared, learnt probability distribution.

Bayes by backprop

- **Being Bayesian by backpropagation**
  - The goal is to minimize the variational free energy:

$$\begin{aligned}\mathcal{F}(D,\theta) &= D_{KL}(q(w|\theta)||p(w)) - \mathbb{E}_{q(w|\theta)}[\log p(D|w)] \\ &= \int q(w|\theta)\log\frac{q(w|\theta)}{p(w)}dw - \int q(w|\theta)\log p(D|w)dw \\ &= \int q(w|\theta)\left[\log q(w|\theta) - \log p(W) - \log p(D|w)\right]dw\end{aligned}$$

  - We can approximate $\mathcal{F}(D,\theta)$ as

$$\mathcal{F}(D,\theta) \approx \log q(w|\theta) - \log p(W) - \log p(D|w)$$

    where w is drawn from $q(w|\theta)$.

Bayes by backprop

- **Being Bayesian by backpropagation**
    - Use $f(w, \theta) = \log q(w|\theta) - \log p(W) - \log p(D|w)$ for training $q(w|\theta)$.
    - This is it.

Summary of Variational Inference

Summary of Variational Inference

- Log marginal likelihood
  $= \text{ELBO (variational free energy)} + D_{KL}(q(w|\theta)||p(w|D))$
- Note that $p(w|D)$ can hardly be computed analytically.

Summary of Variational Inference

- Derivation:

$$
\begin{aligned}
\ln p(D) &= \int \ln p(D) q(w|\theta) dw \\
&= \int q(w|\theta) \ln \frac{p(D)p(w|D)}{p(w|D)} dw \\
&= \int q(w|\theta) \ln \frac{p(w, D)}{p(w|D)} dw \\
&= \int q(w|\theta) \ln \frac{p(D|w)p(w)}{p(w|D)} dw \\
&= \int q(w|\theta) \ln \frac{q(w|\theta)p(D|w)p(w)}{q(w|\theta)p(w|D)} dw \\
&= \int q(w|\theta) \ln \frac{q(w|\theta)}{p(w|D)} dw + \int q(w|\theta) \ln \frac{p(D|w)p(w)}{q(w|\theta)} dw \\
&= D_{KL}(q(w|\theta)||p(w|D)) + \mathcal{F}[q]
\end{aligned}
$$

where $\mathcal{F}[q]$ is the variational free energy or ELBO.

Summary of Variational Inference

- The variational free energy or ELBO can further experessed as:
- Derivation:

$$
\begin{aligned}
\mathcal{F}[q] &= \int q(w|\theta) \ln \frac{p(D|w)p(w)}{q(w|\theta)} dw \\
&= \int q(w|\theta) \ln p(D|w) dw + \int q(w|\theta) \ln \frac{p(w)}{q(w|\theta)} dw \\
&= \mathbb{E}_{q(w|\theta)}[\ln p(D|w)] - D_{KL}(q(w|\theta) || p(w)) \\
&= \text{likelihood under } q - \text{prior fitting term}
\end{aligned}
$$

- We try to maximize $\mathcal{F}[q]$ to

  1. ~~maximize the marginal likelihood $p(D)$~~
  2. ~~reduce the gap between $p(w|D)$ and $q(w)$.~~
  3. keep the variational distribution $Q(w)$ close to our prior $p(w|A)$.

Dropout as a Bayesian Approximation

Dropout as a Bayesian Approximation

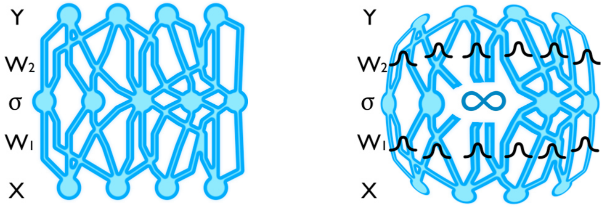


Figure 1: A neural network and Bayesian neural network. (Copyright to Yarin Gal's blog.)

Dropout as a Bayesian Approximation

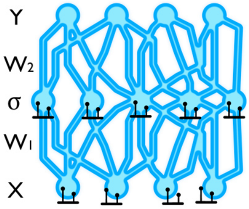


Figure 2: A Bayesian neural network with dropout. (Copyright to Yarin Gal's blog.)

Dropout as a Bayesian Approximation

- Goal of a Bayesian neural network is to find a posterior distribution:

$$p(W|X,Y) \propto P(Y|X,W)P(W)$$

  where X and Y are the input and output training data and W is a set of parameters of our interest.
- Once we have $p(W|X,Y)$, the output y_* at unseen $\mathbf{x}_*$ is predicted as:
$$p(y_*|\mathbf{x}_*) = \int_W p(y|\mathbf{x}_*,W)p(W|X,Y)dW.$$
- In practice, none of them is tractable. ☹

Dropout as a Bayesian Approximation

- Variational inference is often used to handle this issue.
- Instead of finding $p(W|X,Y)$, optimize a variational distribution $q_\theta(W)$ by minimizing

$$\mathrm{KL}(q_\theta(W)\|p(W|X,Y)) = \int q_\theta(W)\log\frac{q_\theta(W)}{p(W|X,Y)}dW.$$

- Of course computing $\mathrm{KL}(q_\theta(W)\|p(W|X,Y))$ is not tractable as well. ☹
- However, we can compute the lower-bound called evidence lower-bound (ELBO) instead! ☺

$$\mathrm{ELBO} = \int q_\theta(W)\log p(Y|X,W)dW - \mathrm{KL}(q_\theta(W)\|p(W))$$

Dropout as a Bayesian Approximation

- In computing ELBO, we do not need to know $p(W|X,Y)$. ☺

$$\text{ELBO} = \int q_\theta(W) \log p(Y|X,W) dW - \text{KL}(q_\theta(W) \| p(W))$$

- Instead, all we need is
  - prior: $p(W)$
  - variational distribution: $q_\theta(W)$
  - likelihood: $p(Y|X,W)$

Dropout as a Bayesian Approximation

- How can we define a likelihood function?

$$p(Y|X, W)$$

- Let's use a Gaussian process.
- We need a kernel function.

Dropout as a Bayesian Approximation

- We will use following kernel function:

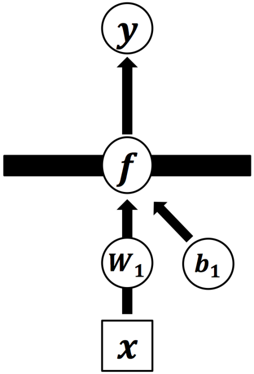
$$K(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{w})p(b)\sigma(\mathbf{w}^T\mathbf{x} + b)\sigma(\mathbf{w}^T\mathbf{y} + b)d\mathbf{w}db.$$

- Then we use Monte Carlo integration with K terms:

$$\hat{K}(\mathbf{x}, \mathbf{y}) = \frac{1}{K}\sum_{k=1}^{K}\sigma(\mathbf{w}_k^T\mathbf{x} + b_k)\sigma(\mathbf{w}_k^T\mathbf{y} + b_k)$$

  where $\mathbf{w}_k \sim p(\mathbf{w})$ and $b_k \sim p(b)$.
- Note that K will be the number hidden units in the hidden layer.

Dropout as a Bayesian Approximation

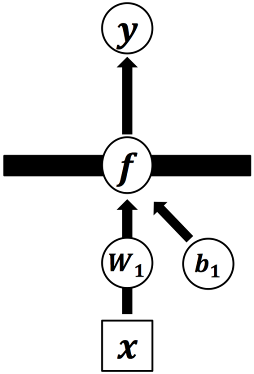


- Using $\hat{K}$ as the covariance function of the Gaussian process, we have the followings:

$$\mathbf{w}_k \sim p(\mathbf{w}), b_k \sim p(b)$$
$$W_1 = [\mathbf{W}_k]_{k=1}^K, \mathbf{b} = [b_k]_{k=1}^K$$
$$\hat{K}(\mathbf{x}, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^K \sigma(\mathbf{w}_k^T \mathbf{x} + b_k)\sigma(\mathbf{w}_k^T \mathbf{y} + b_k)$$
$$F|X, W_1, b \sim \mathcal{N}(0, \hat{K}(X, X))$$
$$Y|F \sim \mathcal{N}(F, \tau^{-1} I_N)$$

where $W_1 \in \mathbb{R}^{Q \times K}$.

Dropout as a Bayesian Approximation

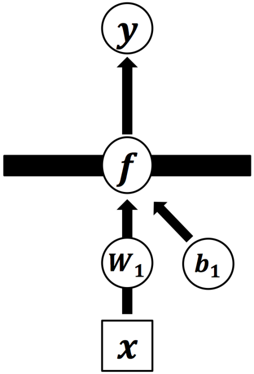


- The marginal likelihood becomes:

$$p(Y|X) = \int p(Y|F)p(F|W_1, b, X)p(W_1)p(b)$$

  where the integration is w.rt. F, W_1, and b.
- What we have so far is a marginal likelihood where the W_1 and b are **auxiliary random variables**.
- Where do they came from?
  - $\hat{K}(\mathbf{x}, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \sigma(\mathbf{w}_k^T \mathbf{x} + b_k)\sigma(\mathbf{w}_k^T \mathbf{y} + b_k)$.

Dropout as a Bayesian Approximation

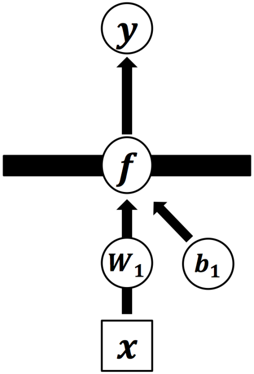


- Suppose $\phi(\mathbf{x}) \in \mathbb{R}^K$ is a hidden layer corresponding to $\mathbf{x} \in \mathbb{R}^Q$:

$$\phi(\mathbf{x}, W_1, b) = \sqrt{\frac{1}{K}}\sigma(W_1^T\mathbf{x} + b).$$

- Let $\Phi = [\phi(\mathbf{x}_n, W_1, b)]_{n=1}^N \in \mathbb{R}^{N\times K}$ be a concatenated feature matrix.
- Then we have $\hat{K}(X, X) = \Phi\Phi^T$.

Dropout as a Bayesian Approximation

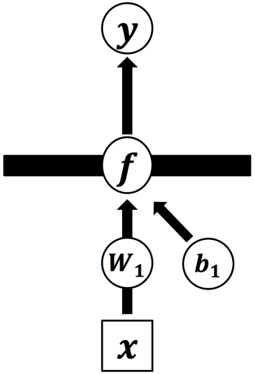


- Finally we have the following likelihood:

$$p(Y|X) = \int \mathcal{N}(Y; 0, \Phi\Phi^T + \tau^{-1}) p(W_1) p(b)$$

where the integration is w.r.t. W_1 and b.

Dropout as a Bayesian Approximation



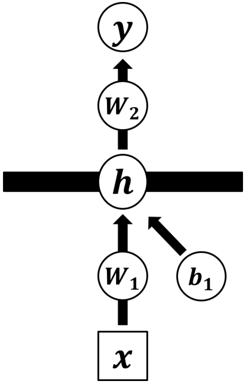
- Here comes a tricky part.
- The probability distribution of $Y \in \mathbb{R}^{N \times D}$ can be written as joint distribution of column vectors $\mathbf{y}_d \in \mathbb{R}^N$.
- Then for each $\mathbf{y}_d$ has the following distribution:

$$\begin{aligned}&\mathcal{N}(\mathbf{y}_d; 0, \Phi\Phi^T + \tau I)\\ &\quad = \int \mathcal{N}(\mathbf{y}_d; \Phi\mathbf{w}_d, \tau^{-1} I)\mathbf{N}(\mathbf{w}_d; 0, I) d\mathbf{w}_d\end{aligned}$$

where $\mathbf{w}_d \sim \mathcal{N}(0, I)$ and Φ is a concatenated feature matrix of

$$\phi(\mathbf{x}, W_1, b) = \sqrt{\frac{1}{K}}\sigma(W_1^T\mathbf{x} + b).$$

Dropout as a Bayesian Approximation



- Now we know that $\mathbf{y}_d$ has the following distribution:

$$\mathcal{N}(\mathbf{y}_d; 0, \Phi\Phi^T + \tau I) = \int \mathcal{N}(\mathbf{y}_d; \Phi\mathbf{w}_d, \tau^{-1} I)\mathbf{N}(\mathbf{w}_d; 0, I)d\mathbf{w}_d$$

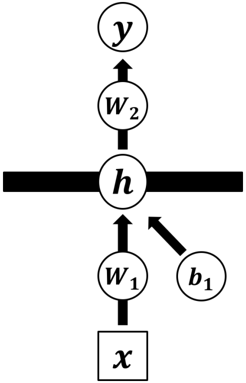
where $\mathbf{w}_d \sim \mathcal{N}(0, I)$ and Φ is a concatenated feature matrix of

$$\phi(\mathbf{x}, W_1, b) = \sqrt{\frac{1}{K}}\sigma(W_1^T\mathbf{x} + b).$$

- Writing $W_2 = [\mathbf{w}_d]_{d=1}^D \in \mathbb{R}^{K\times D}$ then we have the following likelihood:

$$p(Y|X) = \int p(Y|X, W_1, W_2, b)p(W_1)p(W_2)p(b)$$

Dropout as a Bayesian Approximation



- Finally, we have:

$$p(Y|X) \\ = \int p(Y|X, W_1, W_2, b)p(W_1)p(W_2)p(b)$$

where $X \in \mathbf{R}^{N\times Q}$, $\Phi \in \mathbf{R}^{N\times K}$, $Y \in \mathbf{R}^{N\times D}$, $W_1 \in \mathbf{R}^{Q\times K}$, and $W_2 \in \mathbf{R}^{K\times D}$.

Dropout as a Bayesian Approximation

- Recall ELBO:

$$\text{ELBO} = \int q_\theta(W) \log p(Y|X, W) dW - \text{KL}(q_\theta(W) || p(W))$$

where $q_\theta(W) = q(W_1, W_2, b_1)$ and
$\log p(Y|X, W) = p(Y|X, W_1, W_2, b_1)$.

- To sum up, we have

$$\int_{W_1 W_2 b_1} q_\theta(W_1, W_2, b_1) \log p(Y|X, W_1, W_2, b_1) dW_1 dW_2 db_1$$
$$- \text{KL}(q_\theta(W_1, W_2, b_1) || p(W_1, W_2, b_1)).$$

Dropout as a Bayesian Approximation

- However, integration with respect to $\{W_1, W_2, b_1\}$ is still problematic.

$$\int_{W_1 W_2 b_1} q_\theta(W_1, W_2, b_1) \log p(Y|X, W_1, W_2, b_1) dW_1 dW_2 db_1 \\ - \mathrm{KL}(q_\theta(W_1, W_2, b_1) || p(W_1, W_2, b_1)).$$

- Use re-parametrization trick $\tilde{W} = g(W, \epsilon_W)$ where $\epsilon_W \sim p(\epsilon)$ is a random variable.

$$\int_{\epsilon_{W_1}, \epsilon_{W_2}, \epsilon_{b_1}} q_\theta(\epsilon_{W_1}, \epsilon_{W_2}, \epsilon_{b_1}) \log p(Y|X, \tilde{W}_1, \tilde{W}_2, \tilde{b}_1) d\epsilon_{W_1} d\epsilon_{W_2} d\epsilon_{b_1} \\ - \mathrm{KL}(q_\theta(W_1, W_2, b_1) || p(W_1, W_2, b_1)).$$

Dropout as a Bayesian Approximation

- Approximate the integral with Mote Carlo integration:

$$\mathcal{L}_{GP-MC} \triangleq -\sum_{n=1}^{N} \log p(y_n|x_n, \hat{W}_1^n, \hat{W}_2^n, \hat{b}_1^n) + \mathrm{KL}(q(W_1, W_2, b_1) \| p(W_1, W_2, b_1))$$

  with realizations $\hat{W}_1^n, \hat{W}_2^n, \hat{b}_1^n$ from dropout.

- With Gaussian prior on the parameters W_1, W_2, and b_1:

$$\mathcal{L}_{GP-MC} = -\sum_{n=1}^{N} \log p(y_n|x_n, \hat{W}_1^n, \hat{W}_2^n, \hat{b}_1^n) + \lambda_1 \| W_1 \|^2 + \lambda_2 \| W_2 \|^2 + \lambda_3 \| b_1 \|^2$$

  with realizations $\hat{W}_1^n, \hat{W}_2^n, \hat{b}_1^n$ from dropout.

Dropout as a Bayesian Approximation

- To sum up, we have:

$$\mathcal{L}_{GP-MC} = -\sum_{n=1}^{N} \log p(y_n | x_n, \hat{W}_1^n, \hat{W}_2^n, \hat{b}_1^n) \\ + \lambda_1 \| W_1 \|^2 + \lambda_2 \| W_2 \|^2 + \lambda_3 \| b_1 \|^2$$

  with realizations $\hat{W}_1^n, \hat{W}_2^n, \hat{b}_1^n$ from dropout.

Dropout as a Bayesian Approximation

- Once we have W_1, W_2, and b_1, predictive mean and variance can be approximated as:

$$\begin{aligned}\mathbb{E}_q(y_*) &= \int y_* q(y_*|x_*) dy_* \\ &\approx \frac{1}{T}\sum_{t=1}^{T} \hat{y}_*(x_*, \hat{W}_{1,t}, \hat{W}_{w,t}, \hat{b}_{1,t})\end{aligned}$$

$$\mathbb{E}_q(y_* y_*) \approx \tau^{-1} + \frac{1}{T}\sum_{t=1}^{T} \hat{y}_*(x_*, \hat{W}_{1,t}, \hat{W}_{w,t}, \hat{b}_{1,t}) \hat{y}_*(x_*, \hat{W}_{1,t}, \hat{W}_{w,t}, \hat{b}_{1,t})$$

- It can be naturally extended to multi-layer neural networks.

Stein Variational Gradient Descent

Stein Variational Gradient Descent

- Proposed a general purpose variational inference algorithm by
    - iteratively transports a set of particles to match the target distribution
    - by applying a form of functional gradient descent that minimizes the KL divergence.

Stein Variational Gradient Descent

- **Background**
  - Let x be a continuous random variable of interest in $\mathcal{X} \subset \mathbb{R}^d$ and $\{D_k\}$ is a set of i.i.d. observation
  - $p_0(x)$ is prior, $p(x) = \bar{p}(x)/Z$ is posterior where $\bar{p}(x) \triangleq p_0(x) \prod_{i=1}^{N} p(D_k|x)$ where the conditioning on data D is omitted.
  - Let $k(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel and $\mathcal{H}$ is the reproducing kernel Hilbert space corresponding to $k(\cdot, \cdot)$, i.e., $\mathcal{H} = \{f : f(x) = \sum_{i=1}^{m} a_i k(x, x_i), a_i \in \mathbb{R}, m \in \mathbb{N}, x_i \in \mathcal{X}\}$, equipped with an inner product $\langle f, g \rangle_{\mathcal{H}} = \sum_{ij} a_i b_j k(x_i, x_j)$.
  - $\phi(x) = [\phi_1(x), \cdots, \phi_d(x)]^T$ is a smooth vector function.

Stein Variational Gradient Descent

- **Stein's identity (for 1-D case)**

$$
\begin{aligned}
\mathbb{E}_{x\sim p}[\mathcal{A}_p\phi(x)] &= \int_{x\in\mathcal{X}} \nabla_x p(x)\phi(x) + \nabla_x\phi(x)p(x)dx \\
&= \int_{x\in\mathcal{X}} \nabla_x \left(p(x)\phi(x)\right) dx \\
&= 0
\end{aligned}
$$

- This holds when $p(x)\phi(x) = 0, \forall x \in \partial\mathcal{X}$ for compact $\mathcal{X}$.
- For $\mathcal{X} = \mathbb{R}^d$, it is sufficient to show that $p(x)\phi(x)$ goes to zero for $x = \infty$ using Green's identity.
    - Suppose $g(x) = p(x)\phi(x)$. Then, very loosely speaking,

$$
\int_{x\in\mathcal{X}} \nabla_x g(x)dx = \oint g(x)dx.
$$

Stein Variational Gradient Descent

- **Stein discrepancy**
  - Intuitively speaking, it is the maximum violation of Stein's identity for some ϕ in a set $\mathcal{F}$:

$$\mathbb{D}(q,p) = \max_{\phi \in \mathcal{F}} \{\mathbb{E}_{x \sim q}[\text{trace}(\mathcal{A}_p \phi(x))]\}.$$

  - Here, the choice of the function set $\mathcal{F}$ is critical as it decides the discriminative power.

Stein Variational Gradient Descent

- **Kernelized Stein discrepancy (KSD)**
  - Kernelized Stein discrepancy (KSD) assumes that the set is in the unit ball of RKHS:

$$\mathbb{D}(q,p) = \max_{\phi in \mathcal{F}} \{\mathbb{E}_{x\sim q}[\text{trace}(\mathcal{A}_p \phi(x))], \text{ s.t. } \|\phi\|_{\mathcal{H}^d \le 1}\}.$$

  - The optimal solution of KSD is:

$$\phi(x) = \phi^*_{q,p}(x)/\|\phi^*_{q,p}\|_{\mathcal{G}^d}$$

    where

$$\phi^*_{p,q}(\cdot) = \mathbb{E}_{x\sim q}[\mathcal{A}_p k(x,\cdot)]$$

    and

$$\mathbb{D}(q,p) = \|\phi^*_{q,p}\|_{\mathcal{H}^d}.$$

  - If we rewrite the KSD, we have:

$$\mathbb{D}(q,p) = \left\| \int_{x\in\mathcal{X}} [\nabla_x \log p(x) k(x,\cdot) + \nabla_x k(x,\cdot)] q(x) dx \right\|_{\mathcal{H}^d}$$

    and since $p(x)$ is considered through the score function $\nabla_x \log p(x) = \nabla_x \log \bar{p}(x)$, unnormalized posterior $\bar{p}(x)$ can be used without loss of generality.

Stein Variational Gradient Descent

- **Variational inference using smooth transforms**
  - The goal of variational inference is to find a simpler distribution $q^*(x) \in \mathcal{Q}$ that minimizes the KL divergence between $q(x)$ and the posterior $p(x)$.
  - This paper focusses on the sets $\mathcal{Q}$ obtained by **smooth transforms from a tractable reference distribution** $z = \mathbf{T}(x)$ where $\mathbb{T} : \mathcal{X} \to \mathcal{X}$ is a smooth one-to-one transform and x is drawn from a tractable distribution $q_0(x)$.
  - By the change of variables formula, the density of z is:

$$q_{\mathbf{T}}(z) = q(\mathbf{T}^{-1}(z)) |\det(\nabla_z \mathbf{T}^{-1}(z))|.$$

  - Now, our problem is to find $\mathbf{T}$ that maps $\mathcal{X}$ $\mathcal{X}$ where $\mathcal{X} \subset \mathbb{R}^d$ and we will perform **steepest descent** on $\mathbf{T}$ in RKHS.

Stein Variational Gradient Descent

- **Stein operator as the derivative of KL divergence**
  - We will consider $T(x) = x + \epsilon\phi(x)$ where $\phi(x) \in \mathbb{R}^d$ and $x \in \mathcal{X}$ (note that x is a continuous random variable of our interest).
  - Now, we will connect the Stein operator and the derivative of KL divergence w.r.t. the perturbation magnitude ϵ:
  - Let $T(x) = x + \epsilon\phi(x)$ and $q_{\mathbf{T}}(z)$ the density of $z = \mathbf{T}(x)$ when $x \sim q(x)$, we have

$$\nabla_\epsilon D_{KL}(q_{\mathbf{T}}||p)_{\epsilon=0} = -\mathbb{E}_{x\sim q}[\text{trace}(\mathcal{A}_p\phi(x))]$$

    where $\mathcal{A}_p\phi(x)$ is the Stein operator and the optimal direction $\phi^*_{q,p}$ is

$$\phi^*_{q,p}(\cdot) = \mathbb{E}_{x\sim q}[\nabla_x \log p(x)k(x,\cdot) + \nabla_x k(x,\cdot)]].$$

  - $\phi^*_{q,p}(\cdot)$ tells us the optimal direction of the random variable of our interest x if we were to minimize the KL divergence between p and $q_{\mathbf{T}}$.

Stein Variational Gradient Descent

- **Stein Variational Gradient Descent**
  - To implement the iterative procedure, (unnormalized) posterior $p(x)$ is given.
  - Then, we draw a set of particles $\{x_i^0\}_{i=1}^n$ for the initial distribution q_0.
  - Each particle x_i is updated as follows:

$$x_i^{l+1} \leftarrow x_i^l + \epsilon_l \hat{\phi}^*(x_i^l)$$

where

$$\hat{\phi}^*(x) = \frac{1}{n}\sum_{j=1}^{n}\left[k(x_j^l, x)\nabla_{x_j^l}\log p(x_j^l) + \nabla_{x_j^l} k(x_j^l, x)\right].$$

Stein Variational Gradient Descent

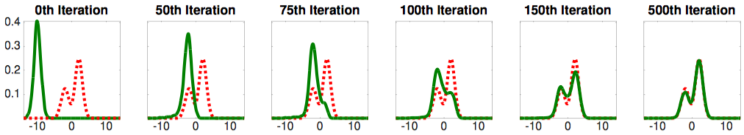
- The update rule has nice interpretations:

$$\hat{\phi}^*(x) = \frac{1}{n}\sum_{j=1}^{n}\Big[k(x_j^l, x)\nabla_{x_j^l}\log p(x_j^l) + \nabla_{x_j^l} k(x_j^l, x)\Big].$$

1. $k(x_j^l, x)$: Similarity between current particle to update x and j-th particle x_j^l.
2. $\nabla_{x_j^l}\log p(x_j^l)$: Particle update direction of current particle to update x where it is computed from j-th particle x_j^l.
    - Note that as we are using the score function, i.e., $\nabla_x \log p(x)$, unnormalized $p(x)$ can be used!
    - This is also used in policy gradient methods such as REINFORCE.
3. $\nabla_{x_j^l} k(x_j^l, x)$: This term can be interpreted as running a gradient ascent method on $k(\cdot,\cdot)$. As the value of a kernel function usually increases as the distance between two inputs decreases, it can be interpreted as an attractive force between particles.

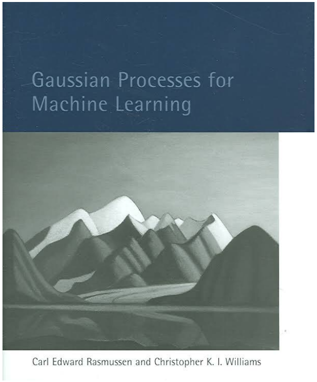
Stein Variational Gradient Descent

- How it works: a toy example with 1D Gaussian mixture.



Questions?

Backup slides


Gaussian Processes for Machine Learning

Carl Edward Rasmussen and Christopher K. I. Williams

Gaussian process for machine learning [2]

References i

K. Hornik.
Some new results on neural network approximation.
Neural networks, 6(8):1069–1072, 1993.

C. E. Rasmussen and C. K. Williams.
Gaussian processes for machine learning, volume 1.
MIT press Cambridge, 2006.