1 Notations

1.1 Model & Data

Let $f(x;\theta)$ a function of x with parameter θ. Let $y=f(x;\theta)$ an observable, thus the observed value obeys a Gaussian distribution. Let D denote a list of observations, $D:=\{(x_i,y_i,\sigma_i)\colon i=1,...,N_D\}$, wherein x_i is the ith input, y_i its observed value, and σ_i the observational error of y_i. We may employ mini-batch technique, thus denote $D_m:=\{(x_i,y_i,\sigma_i)\colon i=1,...,N_m\}\subset D$ as a mini-batch, with batch-size $N_m\leqslant N_D$.

2 Preliminary

2.1 The Bayesian Formula

By Bayes's formula, the posterior can be computed by a valid set of likelihood and prior, as

$$\ln p(\theta|D)=\ln p\left(D|\theta\right)+\ln p(\theta)-\ln p(D).$$

2.2 The General Form of Likelihood

It's very practical to propose that the data are all independent and Gaussian. Indeed, since observations are generally independently taken, this gives the independence; observations with many times of repeatations gives Gaussianity. Then, the likelihood becomes

$$\begin{aligned}\ln p(D|\theta) &= \ln\left(\prod_{i=1}^{N_D}\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left\{-\frac{1}{2}\left(\frac{y_i-f(x_i;\theta)}{\sigma_i}\right)^2\right\}\right)\\ &= \sum_{i=1}^{N_D}\left\{-\frac{1}{2}\ln\left(2\pi\sigma_i^2\right)-\frac{1}{2}\left(\frac{y_i-f(x_i;\theta)}{\sigma_i}\right)^2\right\}.\end{aligned}$$

2.3 Assumption on Prior

Assumption on prior shall take lots of cares and considerations. A bad (or unreasonable) choice of prior ruins all.

2.3.1 Priors in Bayesian Neural Network

As an example, consider Bayesian neural network. Herein, the prior on weights and that on biases are intrinsically different. For prior on weights, we employ Gaussian, and then averaged; and for that on biases, we take uniform instead. This can be illustrated in the section 2.1.1 of Neal (1995) in principle, and in section 2.3.1 ("Regularization") of Nealson (eq. (85) therein) in practice.

2.4 Bayesian Inference

Sample m samples from $p(\theta|D)$, $\{\theta_{(s)}\colon s=1,...,m\}$. Thus, the Bayesian inference gives prediction from x to y as

$$\begin{aligned}\hat{y} &= \mathbb{E}_{\theta\sim p(\theta|D)}[f(x;\theta)]\\ &\approx \left(\frac{1}{m}\sum_{s=1}^{m}\right)f(x;\theta_{(s)}).\end{aligned}$$

2.5 Bayesian as Information Encoder

Comparing with the traditional method, what is the advantage of Bayesian way? The answer is, it encodes more information of data into model. Indeed, it does not encodes the value of peak of the posterior only, as traditional method does, but also much more information on the posterior. XXX

3 Neural Network for Posterior (nn4post)

3.1 The Model

Suppose we have some prior on θ, $p(\theta)$, we gain the unormalized posterior $p(D|\theta)\,p(\theta)$. With D arbitrarily given, this unormalized posterior is a function of θ, denoted by $p(\theta;D)$[1].[2]

We we are going to do is fit this $p(\theta;D)$ by ANN for any given D. To do so, we have to assume that $\mathrm{supp}\{p(\theta;D)\}=\mathbb{R}^d$ for some $d\in\mathbb{N}^+$ (i.e. has no compact support) but decrease exponentially fast as $\|\theta\|\to+\infty$. With this assumption, $\ln p(\theta;D)$ is well-defined. For ANN, we propose using Gaussian function as the activation-function. Thus, we have the fitting function

$$q(\theta;a,\mu,\zeta)=\sum_{i=1}^{N_c} w_i(a)\left\{\prod_{j=1}^{d}\Phi(\theta_j-\mu_{ij},\sigma(\zeta_{ij}))\right\},$$

where

$$\begin{aligned} w_i(a) &= \frac{\exp(a_i)}{\sum_{j=1}^{N}\exp(a_j)}=\mathrm{softmax}(i;a);\\ \sigma(\zeta_{ij}) &= \ln(1+\exp(\zeta_{ij})), \end{aligned}$$

and $a_i,\mu_{ij},\zeta_{ij}\in\mathbb{R}$ for $\forall i,\forall j$ and

$$\Phi(x-\mu,\sigma):=\sqrt{\frac{1}{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

being the Gaussian PDF. The introduction of ζ is for numerical consideration, see below.

3.1.1 Numerical Consideration

If, in q, we regard w, μ, and σ as independent variables, then the only singularity appears at $\sigma=0$. Indeed, σ appears in Φ (as well as the derivatives of Φ) as denominator only, while others as numerators. However, once doing numerical iterations with a finite step-length of σ, the probability of reaching or even crossing 0 point cannot be surely absent. This is how we may encounter this singularity in practice.

Introducing the ζ is our trick of avoiding this singularity. Precisely, using a singular map that pushes the singularity to infinity solves the singularity. In this case, using softplus(.) that pushes $\sigma=0$ to $\zeta\to-\infty$, so that, with finite steps of iteration, singularity (at $-\infty$) cannot be reached.

This trick (i.e. pushing a singularity to infinity) is the same as in avoiding the horizon-singularity of Schwarzschild solution of black hole.

3.2 Interpretation

3.2.1 As a Mixture Distribution

$q(\theta;\,a,\,\mu,\,\zeta)$ has a probablitic interpretation. $\prod_{j=1}^{d}\Phi(\theta_j-\mu_{ij},\,\sigma(\zeta_{ij}))$ corresponds to multi-dimensional Gaussian distribution (denote $\mathcal{N}$), with all dimensions independent with each other. The $\{w_i(a)\}$ is a categorical distribution, randomly choosing the Gaussian distributions. Thus $q(\theta;a,\mu,\zeta)$ is a composition: categorical $\to$ Gaussian. This is the *mixture distribution*.

1. This is why we use " ; " instead of " , ", indicating that D has been (arbitrarily) given and fixed.
2. The normalized posterior $p(\theta|D)=p(D|\theta)\,p(\theta)/p(D)=p(\theta;D)/p(D)$, by Bayes's rule.

3.2.2 As a Generalization

This model can also be intrepreted as a direct generalization of mean-field variational inference. Indeed, let $N_c = 1$, this model reduces to mean-field variational inference. Remark that mean-field variational inference is a mature algorithm and has been sucessfully established on many practical applications.

3.3 Loss-Function

We use "evidence of lower bound" (ELBO) as loss. It is ensured to have a unique global minimal, at which $p(\theta; D) = q(\theta; a, \mu, \zeta)$.

$$\begin{aligned} \mathrm{ELBO}(a, \mu, \zeta) &:= \mathbb{E}_{\theta \sim q(\theta; w, b)}[\ln p(\theta; D) - \ln q(\theta; a, \mu, \zeta)] \\ &\approx \left(\frac{1}{n} \sum_{\theta^{(s)}} \right) \{\ln p(\theta_{(s)}; D) - \ln q(\theta_{(s)}; a, \mu, \zeta)\}, \end{aligned}$$

where $\{\theta_{(s)} : s = 1, ..., n\}$ is sampled from $q(\theta; a, \mu, \zeta)$ as a distribution. Since there's no compact support for both $p(\theta; D)$ and $q(\theta; a, \mu, \zeta)$, ELBO is well-defined, as the loss-function (or say loss-function, performance, etc) of the fitting.

4 Stochastic Optimization

Suppose, instead of use the whole dataset, we employ mini-batch technique. Let D_m denotes the mini-batch with batch-size N_m. With this, we have approximation

$$\ln p(\theta; D) \approx \frac{N_D}{N_m} \ln p(\theta; D_m).$$

Let $q_m(\theta; a, \mu, \zeta)$ the function that fits the $p(\theta; D_m)$, thus we would expect

$$q(\theta; a, \mu, \zeta) \approx [q_m(\theta; a, \mu, \zeta)]^{N_D/N_m}.$$

XXX

5 Computational Resource of Training

Recall that d denotes the dimension of θ, the parameter of model $f(x; \theta)$; N_c denotes the number of categories in the mixture distribution; N_D the number of data.

The dependence of computational resource on N_D is intactable, since this dependence is determined by the inner complexity of $f(x; \theta)$. Thus, we shall fix this N_D or just omit it by introducing mini-batch technique.

5.1 At Each Iteration

5.1.1 Overview

At each step of iteration of optimizer (e.g. `GradientDescentOptimizer`), the computational resources spent on time, for traditional maxima a posterior, variational inference with mean-field approximation, and neural network for posterior respectively:

$$\begin{aligned} \text{MAP} &= \Theta(d); \\ \text{Mean-Field VI} &= \Theta(d); \\ \text{nn4post} &= \Theta(N_c\, d). \end{aligned}$$

5.1.2 Traditional MAP

At each step of iteration of optimizer (e.g. `GradientDescentOptimizer`), the computational resource spent on time is of $\Theta(d)$, i.e. computing the partial derivative values of loss-function by model paramters $\{\theta_j : j = 1, 2, ..., d\}$.

5.1.3 Variational Inference with Mean-Field Approximation

At each step of iteration of optimizer (e.g. `GradientDescentOptimizer`), the computational resource spent on time is of $\Theta(2d) = \Theta(d)$, i.e. computing the partial derivative values of loss-function by each paramter of mean-field approximation $\{(\mu_j, \sigma_j) : j = 1, 2, ..., d\}$.

5.1.4 Neural Network for Posterior

At each step of iteration of optimizer (e.g. `GradientDescentOptimizer`), the computational resource spent on time is of $\Theta(N_c + 2N_c\ d) = \Theta(N_c\ d)$[3], i.e. computing the partial derivative values of loss-function by each paramter of mean-field approximation

$$\{(a_i, \mu_{ij}, \zeta_{ij}) : i = 1, 2, ..., N_c; j = 1, 2, ..., d\}.$$

5.2 Essential Number of Iterations

The essential number of iterations of optimizer depends both on N_c and d, and increasing either N_c or d will also increase it.

Indeed, when increasing d, the searching path of peaks in the paramter-space can oscillate along more dimensions, this makes the path longer.

And, when increasing N_c, the optimizer needs more steps of iterations for tuning the relative ratios between the a_is, while in the case of mean-field approximation where $N_c = 1$, there's no need of such tuning. This effect can be visualized by the figure 1, wherein notice that, since the two loss are closed in the tail, it hints that $N_c = 1$ is the intrinsic number of peaks of the posterior.

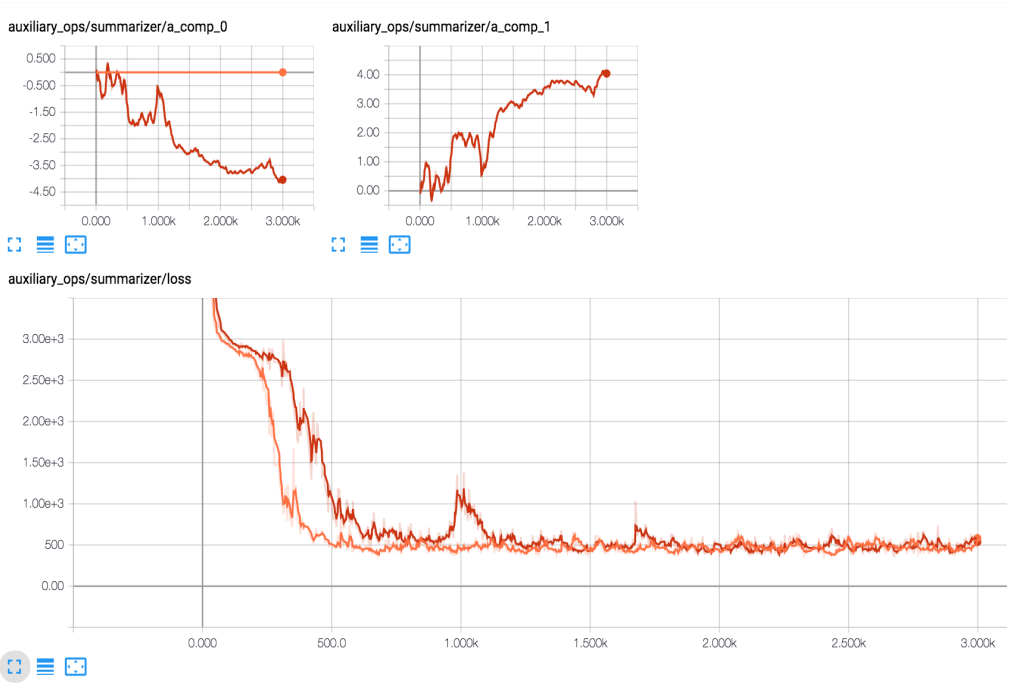

Figure 1. The orange line represents $N_c = 1$ and the red $N_c = 2$ ("`a_comp_i`" represents a_i). The first converges faster than the later. And precisely as it shows, the case $N_c = 2$ needs 500 steps of iterations to tune the a_1 and a_2 so that only one peak is essentially left, and it is just around 500 steps of iterations that the two losses get together. (For the source code, see `'nn4post/tests/shadow_neural_network.py'`.)

3. Herein we have supposed that $d \gg 1$, which is quite practical.

5.3 Batch-Size

The batch-size needed for variational inference (including both this model and the mean-field approximiation) is generally greater than that for non-Bayesian. This is an experimental result (on MNIST dataset), but what is the reason?

6 When & How to Use?

As the figure 1 hints, employing a large N_c will unnecessarily waste computational resource. So, instead we'd better try $N_c = 1, 2, 3, ...$ one by one, until increasing N_c cannot reduce the loss apparently. At this situation, e.g. $N_c = n$ for some n, it hints that the posterior we are fitting has only n apparent peaks. This reveals the intrinsic nature of the posterior, and we shall stop increasing N_c any more, stop wasting the computational resource.[4]

7 Deep Learning

It cannot solve the vanishing gradient problem of deep neural network, since this problem is intrinsic to the posterior of deep neural network. Indeed, the posterior has the shape like $\exp(-x^2/\sigma^2)$ with $\sigma \to 0$, where x is the variable (argument) of the posterior. It has a sharp peak, located at a tiny area, with all other region extremely flat. The problem of find this peak, or equivalently, findng its tiny area, is intrinsically intactable.

So, even for Bayesian neural network, a layer by layer abstraction along depth cannot be absent.

8 Problems

8.1 Generalization Problems

8.1.1 From Mini-Batch to the Whole Dataset

XXX $q(\theta; a, \mu, \zeta) \to [q(\theta; a, \mu, \zeta)]^{N_D/N_m}$, this will be intactable if $q(\theta; a, \mu, \zeta)$ involves summation.

8.1.2 Model Transfer

Since all components of θ in $q(\theta; a, \mu, \zeta)$ are not independent, transfering the model by copying its subgraph (e.g. several based layers) and then attaching directly to other model's graph cannot be taken. XXX

8.1.3 A Solution: Mean-Field Approximation

I mean, not the Gaussian $q(\theta_i, \lambda_i)$ where λs denotes arbitrary parameters, but a general form of it. However, the $q(\theta; \lambda)$ can be decomposited as $q(\theta; \lambda) = \prod_i q(\theta_i; \lambda_i)$. This solves the provious two problems in one go. XXX

4. (A proposal:) Or iteratively? That is, first training by $N_c = 1$; when loss becomes stable after a period of training, add a new peak, so that $N_c = 1 \to 2$; then, when loss becomes stable again after a new period of training, add a new peak, so that $N_c = 2 \to 3$; repeating. Question: if so, then what is the initial value of a of the newly added peak? (Being $a_{\max}$?)