1 Preliminary

1.1 Assumptions on Posterior

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. Thus, for a list of observations $D:=\{(x_i,y_i,\sigma_i)\colon i=1,...,N_D\}$ (σ_i is the observational error of y_i), we can construct a (logrithmic) likelihood, as

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

If in addition assume a Gaussian prior, for some hyper-parameter σ,

$$p(\theta)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\theta^2}{2\sigma^2}\right),$$

then we have posterior $p(\theta|D)$

$$\begin{aligned}\ln p(\theta|D) &= -\frac{1}{2}\left\{\sum_{i=1}^{n}\left(\frac{y_i-f(x_i;\theta)}{\sigma_i}\right)^2+\left(\frac{\theta}{\sigma}\right)^2\right\}\\ &- \frac{1}{2}\left\{\sum_{i=1}^{n}\ln(2\pi\sigma_i^2)+\ln(2\pi\sigma^2)\right\},\end{aligned}$$

where the second line is θ-independent.

1.2 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 Neural Network for Posterior (nn4post)

2.1 The Model

Suppose we have a model, $f(x,\theta)$, where x is the input and θ the set of parameters of this model. Let D denotes an arbitrarily given dataset, i.e. $D=\{(x_i,y_i)\colon i=1,2,...,N_D\}$ wherein, for $\forall i$, x_i is the input and y_i the target (observed). With some assumption of the dataset, e.g. independency and Gaussianity, we can gain a likelihood $L(\theta;D):=p(D|\theta)$. Suppose we have some prior on θ, $p(\theta)$, we gain the unormalized posterior $L(D,\theta)\,p(\theta)$. With D arbitrarily given, this unormalized posterior is a function of θ, denoted by $p(\theta;D)$[1].[2]

1. This is why we use " ; " instead of " , ", indicating that D has been (arbitrarily) given and fixed.

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 $\text{supp}\{p(\theta; D)\} = \mathbb{R}^d$ 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

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

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.

2.1.1 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.

2.2 Interpretation

2.2.1 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*.

2.2.2 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.

2. The normalized posterior $p(\theta|D) = p(D|\theta)\, p(\theta) / p(D) = p(\theta; D) / p(D)$, by Bayes's rule.

2.3 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.

3 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.

3.1 At Each Iteration

3.1.1 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:

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

3.1.2 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 $\{\theta_j : j = 1, 2, ..., d\}$.

3.1.3 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 $\{(\mu_j, \sigma_j) : j = 1, 2, ..., d\}$.

3.1.4 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

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

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

3.2 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.

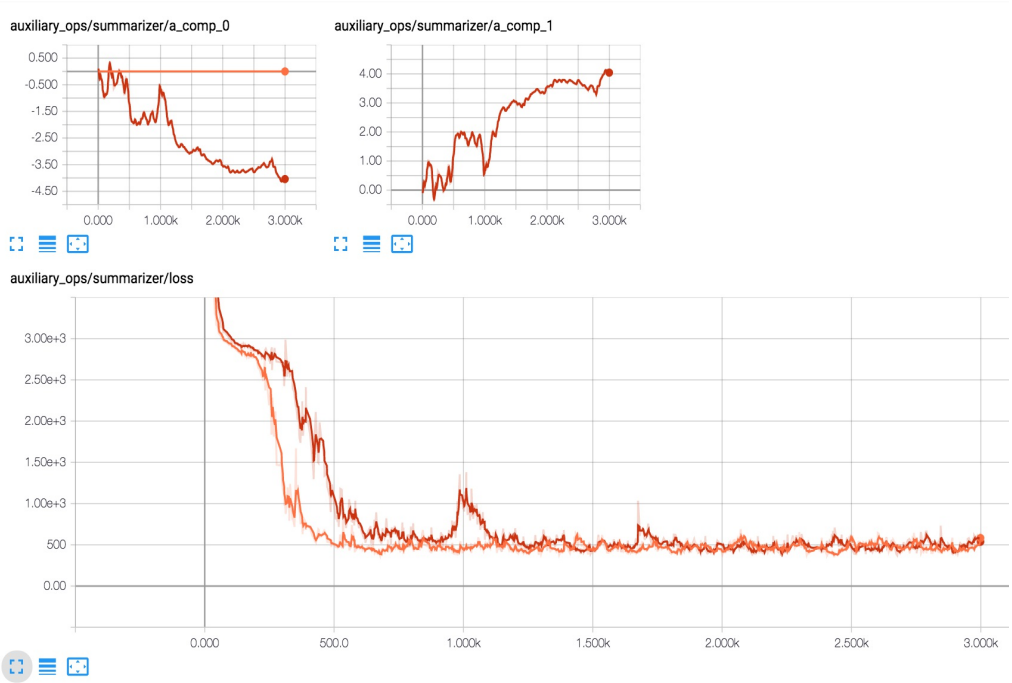

Figure 1. The orange line represents $N_c = 1$ and the red $N_c = 2$ ("`a_comp_i`" represents a_i). The first converges faster than the later. And precisely as it shows, the case $N_c = 2$ needs 500 steps of iterations to tune the a_1 and a_2 so that only one peak is essentially left, and it is just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

4 When & How to Use?

As the figure 1 hints,

XXX: try $N_c = 1, 2, 3, \ldots$ 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 posterior, and we shall stop increasing N_c any more, wasting the computational resource.

XXX: 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 $a_{\max}$?)

5 XXX: Deep Learning?

XXX: It cannot solve the vanishing gradient problem of deep neural network, since this problem is intrinsic to the posterior of deep neural network.