Approximate Inference

Zhiwei Han
Faculty of Electrical Engineering and Information Technology
Technical University of Munich
Arcisstr. 21, Munich, 80333
Email: hanzw356255531@icloud.com

***Abstract*—This is a seminar work from Chair of Medientechnik, Technical University of Munich. Most of this work is based an the deep learning book from MIT. In this seminar work, an overview of approximation inference and how this technique is applied to solve intractable problem of posterior in probabilistic model, is given. We begin with the introduction of the intractable problems arised in the inference of a probablistic model and then present a new objective function for the optimization problem. After that, we show the great simplification of the optimization problem with the new objective function instead of the old one with mathematical proof. At the end, a CNN-based variational auco encoder is presented to show that with this technique a deep generative model, whose latent variables are gaussian distributed, can be train in a feasible time.**

***Index Terms*—Approximate Inferience, Variational Inference, Generative Model, Auto Encoder.**

I. Introduction

In many statistical learning problems especially the training of a generative model, inference is considered as the very first step to train the probabilistic models before performing specific optimization method like maximum likelihood learning. For some simple graphical models like Restricted Boltzmann Machines (RBM) and probabilistic PCA, inference can be simply done by computing the posterior and taking the expectation over it [1]. Those computations are critical process and are also basis of training step afterwards. However, with some graphical models have multiple layers like Deep Belief Network or intractable connections between the latent variables, the exact direct computation of inference in a constraint time will cost an exponential amount of time. Consequently, a precise evaluation of inference is infeasible because of the explosion of computational complexity and the limited computational power.

In the context of deep learning, the problem setting can be organized in a more specific way. We assume that we have a set of visible variables $\boldsymbol{v}$, which can be seen as the input of a one layer RBM and a set of latent variables $\boldsymbol{h}$, the corresponding output. The goal of inference on such model is to compute the posterior $p(\boldsymbol{h}|\boldsymbol{v})$ analytically. Unfortunately, the main challenge is usually the result of the intractable inference problems due to several interactions between latent variables in a structured graphical model. In other words, it's definitely inefficient, if we still calculate the posterior in the traditional way when the latent variables are not independent anymore.

One possibility how we deal with these kind of intractable inference problems is variational inference. Instead of trivially integral over the latent variables, we are going to find a distribution to approximate the posterior as much as possible and a lower bound of log likelihood function with respect to this approximate posterior. Finally, maximize this lower bound over model variables. For a perfect approximation q of posterior $p(\boldsymbol{h}|\boldsymbol{v})$, the lower bound is exactly the log likelihood function.

The goal of this seminar work is to present an overview about the approximate inference and effective method to confront these issues in term of statistics. In the second section, we show the basic concept of inference and what is the problem need to handle with through an intuitive example. In the third, fourth and fifth section, we introduce several techniques for solving intractable inference problem and learning with structured probabilistic models, whose latent variables are either discrete or continuous. As the learned approximate posterior inference model can be used in a huge amount of tasks, in the last section, we show that after a neural network is used for recognition model, it's turned out to be a $variational\ auto\text{-}encoder$.

II. Background

A. Inference

In machine learning community, discriminant method and generative method are two main approaches to solve specific learning tasks with large data sets, their models are therefore named as discriminant models (SVM, Logistic Regression) and generative models (GMM, HMM), respectively.

The goal of discriminant models is prediction, in other words the disciminant model learns the **conditional probability distribution** $p(\boldsymbol{c}|\boldsymbol{o})$, which is the conditional probability of class vector $\boldsymbol{c}$ given observation vector $\boldsymbol{o}$, and the model should be able to predict the exact class of a new coming observation according to a predefined criterium (e.g. the conditional probability is higher than a threashold) afterwards. While the generative model does inference, that is to learn the **joint distribution** $p(\boldsymbol{c}, \boldsymbol{o})$ of the given data sets. Since the generative model knowns the joint distribution of the data sets,

so conditional probability can be easily derived by dividing the joint distribution with prior according to bayes rule,

$$p(\boldsymbol{c}|\boldsymbol{o}) = \frac{p(\boldsymbol{c}, \boldsymbol{o})}{p(\boldsymbol{o})} \tag{1}$$

From the above example, we can see that inference is a generalization form of prediction. Therefore, generative model has better representation ability and a faster convergence. The drawbacks is that the training of generative model is more computational complex.

We define here the problem setting for the rest of this seminar work. Our inference problems are built so that, the models are consisting of visible variables $\boldsymbol{v}$ and latent variables $\boldsymbol{h}$. We would like to maximize likelihood of the given dataset $\boldsymbol{x}$.

Since for discriminant model there are already lots of efficient computational algorithms and this seminar work is mainly about approximate inference, in the rest of this seminar work we mainly focus on the application of approximate inference in generative models. Consider the standard training procedure of a generative model, which has visible variables $\boldsymbol{v}$ and latent variables $\boldsymbol{h}$, as first step we need to compute the likelihood by marginalizethe its visible variables over latent variable as follows.

$$L(\boldsymbol{v} \mid \boldsymbol{\theta}) = \int_{\boldsymbol{h}} p(\boldsymbol{v} \mid \boldsymbol{h}, \boldsymbol{\theta}) p(\boldsymbol{h} \mid \boldsymbol{\theta}) \mathrm{d}\boldsymbol{h}. \tag{2}$$

Simple graphical models remain the computation of posterior $p(\boldsymbol{h} \mid \boldsymbol{\theta})$ still solvable e.g. RBM (see fig. 1). Unfortunately, most applicable graphical model usually have interactions between their latent variables and thus also have intractable posterior distribution (see fig. 2), it means that the v-structure and the intractable edge between the latent varialbes$p(\boldsymbol{h} \mid \boldsymbol{\theta})$ make the posterior distribution intractable. Consequently, the computational expense rise dramatically and it is almost impossible to finish the computations of posterior in a feasible training time. However, with approximation inference, we are given a powerful weapon and then able to solve this problem.

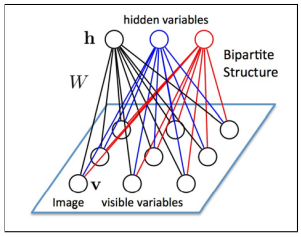

Fig. 1. RBM: every latent variable is independent to each other since there is no connection between them. The posteriors are through factorization solvable.

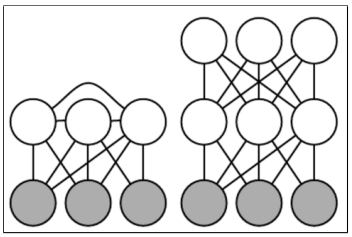

Fig. 2. Models with v-structure and edge between latent variables: the posterior distribution are intractable because of the interaction between latent variables

B. MAP Inference

As shown in (2), when training a probablistic model e.g. a generative model, we are always interested in computing the data distribution by integral over a set latent variables, namely inference. However, integral over all latent variables could be computationally expensive and should be therefore avoided in the real implementation. A solution to this problem is to computer the most likely latent variable $\boldsymbol{h}^*$, rather than integral over all possible latent variables. Because in practice, for most $\boldsymbol{h}$, $P(\boldsymbol{v}|\boldsymbol{h})$ will be nearly zero, and hence contribute almost nothing to the calculation of likelihood $p(\boldsymbol{v})$.[1][2]

Mathmatically, it is equal to an optimization problem as follows,

$$\boldsymbol{h}^* = \arg\max_{\boldsymbol{h}} p(\boldsymbol{h} \mid \theta), \tag{3}$$

and approximate (2) as

$$\begin{aligned} L(\boldsymbol{v} \mid \boldsymbol{\theta}) &= \int_{\boldsymbol{h}} p(\boldsymbol{v} \mid \boldsymbol{h}, \boldsymbol{\theta}) p(\boldsymbol{h} \mid \boldsymbol{\theta}) \mathrm{d}\boldsymbol{h} \\ &= p(\boldsymbol{v} \mid \boldsymbol{h}^*, \theta) p(\boldsymbol{h}^* \mid \theta) \end{aligned} \tag{4}$$

This method is known as maximum a posteriori inference (MAP).

C. EM Algorithm

Expectation Maximization (EM) [3] algorithm is a standard iterative learning algorithm, which is based on maximum likelihood estimation and especially designed for models with latent varialbes.

EM algorithm includes two steps and runs untill the predined convergent criterium is satisfied,

1) Initialization: Initilize the model parameters $\boldsymbol{\theta}_0$

2) Expection Step: Compute the objective function (sum of ELBO on all data index) according to (3),

$$\sum \mathcal{L}(\boldsymbol{v}^{(i)}, \boldsymbol{\theta}, q). \tag{5}$$

Note, set $q(\boldsymbol{h}^{(i)} \mid \boldsymbol{v})$ for all the index of the data set we need to train on and remain distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ always equal to $p(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{\theta}_0)$ while updating $p(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{\theta}_t)$ with θ_t, where t is the number of current iteration.

3) Maximization Step: Maximize the objective function (sum of ELBO on all data index) over model parameters $\boldsymbol{\theta}_t$ with arbitary opimization algorithm.

4) Repeat 2), 3) untill converge:

III. Variational Inference

A. Objective function

Many difficult sample based inference problems which make use of observations can be reconstructed as optimization problems and they maximize the log-likelihood function of the given datasets.[2][4] Approximate Inference algorithm will then simplify the underlying optimization problems by

using the approximation of posteriors.

While the intractation between latent variables make the likelihood computation much more difficult (because of integral), instead of directly calculating and optimizing the log-likelihood, we introduce a new objective function here, which it is easy to compute and optimize if a distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ could be found. This means that we need to find a new distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ which can make a good approximation of the posterior. Or in other words, it is a function, which takes a vetor $\boldsymbol{v}$ as visible variable and give us a distribution over the latent variable $\boldsymbol{h}$ that are likely to produce the visible variable $\boldsymbol{v}$. Ideally, the distribution of $\boldsymbol{h}$ under $q(\boldsymbol{h} \mid \boldsymbol{v})$ is much simpler than the one under the posterior $p(\boldsymbol{h} \mid \boldsymbol{v})$. This trick makes the computation of $\mathbb{E}_{\boldsymbol{h}\sim q}\Big[p(\boldsymbol{v} \mid \boldsymbol{h})\Big]$ significant much easier.

We begin with the deriviation of the new objectvie function from the definition of KL Divergence between the distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ and the posterior $p(\boldsymbol{h} \mid \boldsymbol{v})$ for some arbitary q, then we get the $\log p(\boldsymbol{v} \mid \boldsymbol{h})$ and $\log p(\boldsymbol{h})$ term after applying Bayes rule to $\log p(\boldsymbol{h} \mid \boldsymbol{v})$. Here we can take the $\log p(\boldsymbol{v})$ term out of the expectation since it has no dependency with $\boldsymbol{h}$.

$$
\begin{aligned}
&D_{KL}\Big[q(\boldsymbol{h}) \mid\mid p(\boldsymbol{h} \mid \boldsymbol{v})\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log \frac{q(\boldsymbol{h})}{p(\boldsymbol{h} \mid \boldsymbol{v})}\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h}) - \log p(\boldsymbol{h} \mid \boldsymbol{v})\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h}) - \log \frac{p(\boldsymbol{v}, \boldsymbol{h})}{p(\boldsymbol{v})}\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h}) - \log \frac{p(\boldsymbol{v}, \boldsymbol{h})}{p(\boldsymbol{v})}\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h}) - \log \frac{p(\boldsymbol{v} \mid \boldsymbol{h})p(\boldsymbol{h})}{p(\boldsymbol{v})}\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h}) - \log p(\boldsymbol{v} \mid \boldsymbol{h}) - \log p(\boldsymbol{h})\Big] + \log p(\boldsymbol{v})
\end{aligned}
\tag{6}
$$

After reforming the expectation term into KL divergence and reranging the formula, it yields the new objective function,

$$
\begin{aligned}
J(\theta) &= \log p(\boldsymbol{v}) - D_{KL}\Big[q(\boldsymbol{h} \mid \boldsymbol{v}) \mid\mid p(\boldsymbol{h} \mid \boldsymbol{v})\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log p(\boldsymbol{v} \mid \boldsymbol{h})\Big] - D_{KL}\Big[q(\boldsymbol{h} \mid \boldsymbol{v}) \mid\mid p(\boldsymbol{h})\Big]
\end{aligned}
\tag{7}
$$

B. Core Idea

C. Variational Auto-Encoder

IV. Experiment

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[3] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[4] Y. Anzai, *Pattern recognition and machine learning*. Elsevier, 2012.