Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Bayesian Learning for Binary Neural Networks

Bachelor's thesis

Tejas Bhatnagar

Bachelor's programme: Electrical Engineering and Computer Science
Branch of study: Computer Science
Supervisor: Mgr. Alexander Shekhovtsov, Ph.D.

Prague, August 2022

Thesis Supervisor:
Mgr. Alexander Shekhovtsov, Ph.D.
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Bhatnagar Tejas** Personal ID number: **490986**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Electrical Engineering and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Bayesian Learning of Binary Neural Networks

Bachelor’s thesis title in Czech:

Bayesovské učení binárních neuronových sítí

Guidelines:

The project is to study Bayesian learning methods for binary neural networks. Bayesian learning paradigm averages over all models that explain the data well. Binary neural networks are computationally efficient models that can be learned by optimizing their stochastic relaxation [1]. The goal is to obtain binary neural networks with improved generalization capabilities and quantified uncertainty [2]. There is a successful example of Bayesian binary networks [3], which however has a critical flow in large-scale experiments.
1. Perform experiments with simulated data where the optimal classification and Bayesian predictive probability can be computed and visualized. Consider classification and regression problems and compare different ways of training a stochastic binary network: learning a deterministic predictor, a stochastic predictor, maximum likelihood learning of ensemble, variational Bayesian learning in simple and improved forms. Analyze the cold posterior effect [4].
2. Apply the above methods to medium-scale realistic data and analyze improvements and failure modes.

Bibliography / sources:

[1] A. Shekhovtsov, V. Yanush: Reintroducing Straight-Through Estimators as Principled Methods for Stochastic Binary Networks, DAGM (2021)
[2] Eyke Hüllermeier & Willem Waegeman: Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods, Machine Learning (2021)
[3]: Xiangming Meng, Roman Bachmann, Mohammad Emtiyaz Khan: Training Binary Neural Networks using the Bayesian Learning Rule, ICML (2020)
[4] Florian Wenzel et al.: How Good is the Bayes Posterior in Deep Neural Networks Really? ICML (2020)

Name and workplace of bachelor’s thesis supervisor:

Mgr. Oleksandr Shekhovtsov, Ph.D. Visual Recognition Group FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **17.01.2022** Deadline for bachelor thesis submission: **15.08.2022**

Assignment valid until: **30.09.2023**

Mgr. Oleksandr Shekhovtsov, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

I hereby declare that I have written this bachelor's thesis independently and quoted all the sources of information used in accordance with methodological instructions on maintaining ethical principles when working on a university final project in CTU Prague.

In Prague, August 2022

.....
Tejas Bhatnagar

Abstract

Neural Networks with binary weights are of special interest as they are computation friendly and do not demand a lot hardware. However, training is rather challenging as they these binary weights do not have a gradient.

Bayesian learning averages over a range of models that fit the data well and is thus able to provide us with a decent model itself. Another advantage Bayesian learning posses over existing common methods for training binary neural networks is that it is not empirical. However, despite these advantages over their continuous counter parts, Binary Neural Networks are significantly lag behind traditional neural networks in terms of performance. This is because training Binary networks is particularly difficult as it involves a discrete optimization problem. Moreover, traditional methods such Stochastic Gradient Descent can not be used to update the weights as the they are discrete and do not have a gradient.

In this thesis, various Bayesian methods were explored such as Variational Bayesian Learning and Maximum Likelihood. Their performance is analyzed on a toy dataset drawn from generative data model. Existing methodology is also reviewed.

Keywords: Deep Neural Networks, Binary Neural Networks, Bayesian Learning, Variational Bayesian Learning, Maximum Likelihood, Training Binary Networks, Bayesian Inference.

Acknowledgements

I would like to express my gratitude to my supervisor Mgr. Alexander Shekhovtsov, Ph.D. for his patience, guidance and valuable feedback. He has helped me out tremendously through out the course of this thesis.

I would like to thank all the professors who have taught me over the course of my degree. I am grateful to CTU for providing me with the opportunity to study here and further my education.

Lastly, I would also like to thank my family and friends for their unwavering support.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

The idea of Neural networks began with the intention of modelling a machine learning algorithm around the functioning of a biological neuron. This, however, has not been achieved as the functioning of a biological neuron is far more complex than previously imagined. Nonetheless, Artificial Neural Networks have come a long way and have substantially pushed the boundaries of Artificial Intelligence and Machine Learning. They have proved to be the state of the art solution in many fields, including computer vision, speech recognition, statistical machine translation and many more.

Training Deep Neural Networks requires a lot of hardware and often takes a very long time [1]. They are almost exclusively trained on power hungry Graphic Processing Units as a result of which it has become extremely difficult to run on low powered devices. Coming up with an energy efficient Deep Neural Networks is absolutely essential in realising the potential of mobile applications Artificial intelligence. Energy efficient Deep Neural Network models will be absolutely critical in the field of Internet of things.

Various solutions have been proposed to combat this issue, Binary Neural Networks being one of them. The first methodology regarding binary neural networks was introduced by Courbariaux et al. in [2]. Here, both the weights and activation were binary and were used in the subsequent inference and propagation training.

To train Binary Neural Networks, a few of the ideas that have been explored are mentioned below.

- Using empirical ST methods, the maximum likelihood estimate of the weights is estimated.
- Straight Through estimators [3] and Binary Optimizer [4] are used to overcome the

difficulty of not having a gradient to work with as the weights are binary.

- Ensemble of binary networks is known to give an improvement [5].

1.1 Aim of this thesis

In this thesis, we aim to do the following.

- Compare exact Bayesian learning solutions to Maximum Likelihood and Variational Approximate Bayesian Learning.
- Theoretically derive improved variational Bayesian learning method using mixture of factorized approximations and demonstrate its connection to ensemble methods.
- Compare existing models and methodologies.

Chapter 2

Introduction to Binary Neural Networks

2.1 McCulloch-Pitts Neuron

The first mathematical model of a biological neuron was proposed by McCulloch and Pitts in 1943. This model had binary inputs as well as a binary output.

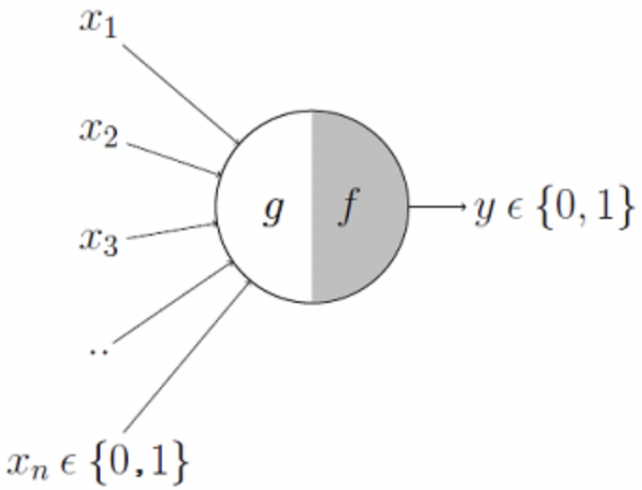


Figure 2.1: McCulloch-Pitt Neuron

All the input values can either be inhibitory, that is, 0 or excitatory, that is 1. g takes in an input and performs an agragate of sorts while f makes the decision. A very simple example of g and f can be is given below.

$$g(x) = \sum_{i=1}^{n} x_i \tag{2.1}$$

$$y = f(g(x)) = 1 \; if \; g(x) > 0 \tag{2.2}$$

$$= 0 \; if \; g(x) \leq 0 \tag{2.3}$$

This model was only binary in nature but however it was not until much later, in 2016 [2], a deep binary neural network was trained.

2.2 Early attempts at binarization

Early binarization had only a single hidden layer [6]. This also introduced us to the unique problems such layers with binary weights bring to the table. Known algorithms such as backpropogation and stochastic gradient descent did not work as the weights can not be incremented by small values. As a result of which, Bayesian inference was used to train these networks and the same would be discussed in this thesis.

The first to claim to have trained a Binary Neural Network was Courbariaux et al. in [2]. However, it can be said that their network was not truly binary as they used real valued weights which were binarized before using the network This method uses latent real valued weights which are binarized in each forward pass during training and testing [7][8]. The gradients needed to update the latent weights are calculated at the binary weights [8].

The first group to bring focus in binary weights in neural networks were Soudry et al. in [9]. They also used Bayesian methods to get around the problems posed by binary weights.

2.3 Review of existing methodology

The work on training binary neural networks can be broadly classified into two different approaches. The first involving designing special architecture which can cater for binary weights, activations and operations [10] [2] [11] [12] [13]. The other being focused on

training methods [8]. Courbariaux et al. developed the methodology related to binary neural networks that is used by most [7] [2]. In this section, we will review the most commonly used methodologies.

2.3.1 Binarization of weights and activation

The primary problem with using binary weights is that they cannot be updated using gradient descent methods such as Stochastic Gradient Descent. In order to tackle this problem, Courbariaux et al. use a set of real valued weights which are binarized within the network to obtain binary weights [2]. Let us denote these real weights with W_r and the binary weights with W_b. W_r is converted to W_b simply as follows.

$$W_b = sign(W_r) \tag{2.4}$$

Thus, W_r can be updated during back-propagation and during inference W_b can be stored and used [2] [7].

Now the next problem that is presented to us is that we cannot calculate the gradient of the sign function as it is undefined. This is solved by using something called a Straight Through Estimator [3] [7]. The weights are binarized in the forward pass and an identity function is performed in the backward pass. Here, since the value of W_r is changed without any change in W_b, the value of W_r is constraint between -1 and +1 [7].

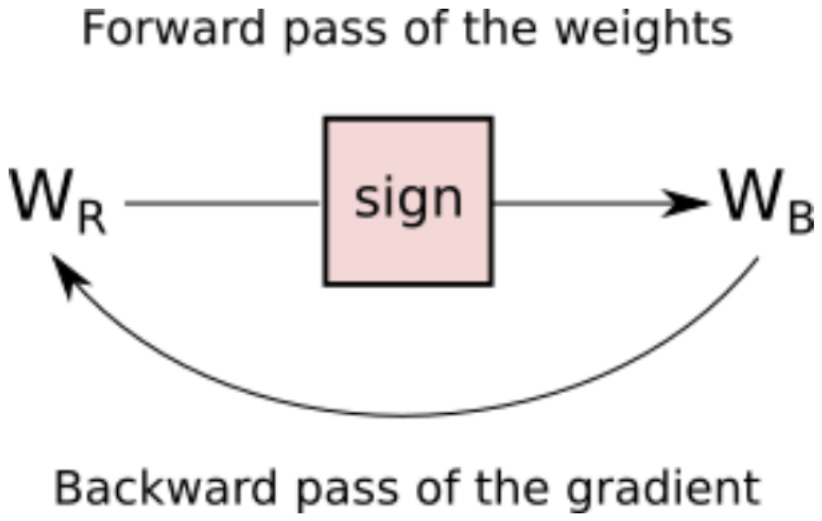


Figure 2.2: Figure visualizing STE [7]

The binarization of activation values was also introduced by Courbariaux et al. in the

same paper [2]. The activation are passed through a sign function as was the case for the weights and a Straight Through Estimator is used in the backward pass.

2.3.2 Bitwise operations

The reason why binary weights reduce computation costs is that the dot product is reduced to just bitwise operation. Using a logical XNOR operation is the equivalent of multiplying these binary values. These bitwise operation are much simpler than computing floating point multiplications [7]. It is claimed by Courbariaux et al. that these operation sped up optimization by 23 folds [2]. In another paper, it was shown that XNOR operations could replace multiply-accumulate operations and thus resulting in a 58 fold speedup on CPUs [11].

2.4 Other Existing Methodologies

2.4.1 Courbariaux et al.

Courbariaux et al. tested their methodology on the MNIST, SVHN and CIFAR-10 datasets and reported their results in [2]. They have also discussed using stochastic binarization which can lead to better results with their binary neural network model [2] [7].

2.4.2 XNOR-Net

Soon after [2], Rastegari et al. introduced their model called XNOR-Net. XNOR-Net uses most of the methodology from [2] while incorporating a a new term. This new term is called 'gain term' and is introduced to compensate for the lost information during binarization [7]. The addition of this gain term showed improvement in the performance of their binary neural network [11]. Rastegari et al. also reported a 64x speed up from traditional neural networks [11].

2.4.3 DoReFa-Net

Zhou et al. introduced a model which allowed variable size for weights and activations during backpropogation in [14]. They also put an emphasis on speeding up training time

instead of just inference [14] [7]

2.4.4 Bayesian Binary Neural Network

Khan et al. proposed a Bayesian algorithm to train binary neural networks. This approach of theirs claims to justify the use of Straight Through Estimator [3] and Binary Optimizer [4], which are claimed to be derived based on intuition and are not theoretically justified [8].

2.4.5 Why is training Binary Neural Network hard?

Despite the advantages Binary networks posses over traditional neural networks, the former still lags behind the latter in terms of performance. Since the weights are binary, binary neural networks can only represent a subset of discrete function which makes them weaker than continuous function approximators [5] [15]. Their training is challenging since in theory it involves a discrete optimization problem [8]. Zhu et al. state that the two other major issues are the issues with robustness with respect to input perturbations, and the stability issue with respect to network parameters [5].

2.5 Motivation for Bayesian Learning

Most of the common methodologies that have been discussed in this section are based on empirical learning. Even if we manage to find the maximum likelihood estimate of the weights, there is little guarantee that it will perform well as it is based on a finite data sample.

Bayesian Binary Neural Networks can also help us by providing uncertainty estimates [8] which can further help us in decision making. This uncertainty can also enable us to perform continual learning [8] [16]. We also wish to see if a Bayesian Model can average over multiple good models.

Chapter 3

Bayesian Learning for Binary Neural Networks

3.1 Basic Bayesian Models

The reason for choosing a basic Bayesian model over an exact Bayesian learning model that exact model are often intractable. Here, we intend to inspect the model where we can compute it precisely. Thereafter, we intend to visualize the predictive distributions and run experiments so as to statistically compare these methods.

3.1.1 Network architecture and toy data

We will consider a simple one layered neural network with two inputs and a single output. We will also consider a sample test data which comprises of multiple Gaussian mixtures labelled as positive or negative. The points will drawn from a generative model which is capable of providing us with as many points as needed for training and testing. The points will be divided into two classes based on which Gaussian they belong to.

For this experiment, we will consider 4 different models which will have $n = 10$, 12, 14 and 16 nodes or neurons in the hidden layer. Another thing to note is that W_1 and b_1 are real valued and W_2 is binary. W_1 and b_1 are random and this is because we wish to able to track the exact Bayesian learning and thus implement a randomized embedding of the data. Thus, we will only be concerned with learning W_2.

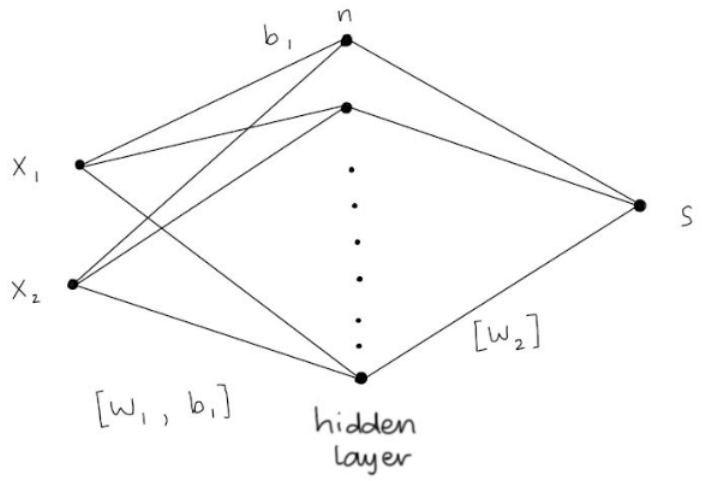


Figure 3.1: Figure visualizing the single layer neural network

$$W_1, b_1 \in \mathbb{R} \tag{3.1}$$

$$W_2 \in \pm 1 \tag{3.2}$$

Now, for any given set of W_1 and b_1, we can have 2^n different possible networks. Our model can be defined as follows.

$$model : p(y \mid x; W_2, \frac{1}{n}) \tag{3.3}$$

In short, our model can be defined only by W_2.

W_1 and b_1 are generated as follows.

$$S \coloneqq \{x \mid 0 \leq x < 1\} \tag{3.4}$$

$$w \in_R S \tag{3.5}$$

$$b_1 \in_R S \tag{3.6}$$

$$W_1 = w \ / \ norm(w) \tag{3.7}$$

The data points for the toy data are drawn from a known Gaussian mixture model.

The following figure 3.2, shows us the contour plot of a Gaussian mixture model. From here, each point will be classified into one of two classes based on which Gaussian it belongs to.

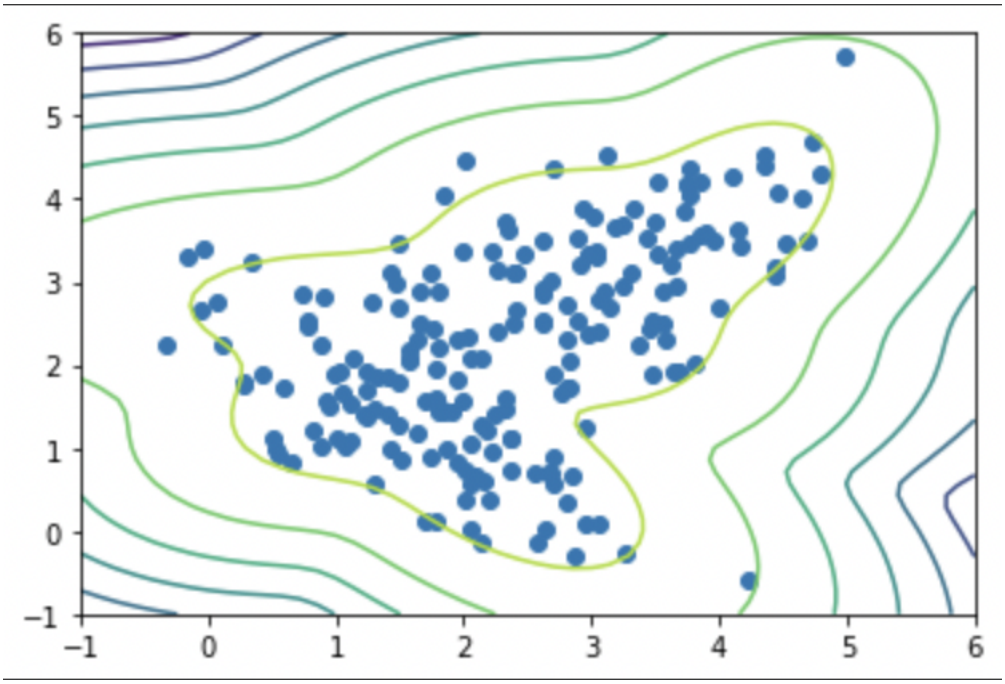


Figure 3.2: Figure visualizing a Gaussian mixture model with 6 Gaussians

3.1.2 Predictive Distribution

Now, the probability of a point belonging to the positive class is given by

$$p(y = 1|s) = \frac{1}{1 + \exp(-s)} \tag{3.8}$$

Here, s = output of our network.

Now, to find the optimal combination, W_2^*, we will maximise the log likelihood of this probability. This is the Maximum Likelihood estimate of W_2.

$$W_2^* = \arg\min \big(\sum_{i=1}^{n} - \log p(y = y_i|x_i; W_2 i) \big) \tag{3.9}$$

This would give us the predictive distribution,

$$p(y = 1|x; W_2^*) \tag{3.10}$$

3.1.3 Bayesian Predictive Probability

For this part, we would make use of the same architecture as in the previous section.

Here, the bayesian posterior is defined as follows.

$$p(W_2|D) = \frac{\prod_{i=1}^{n} p(y = y_i|x_i); W_2}{\sum_{W_2} \prod_{i=1}^{n} p(y = y_i|x_i; W_2)} \tag{3.11}$$

Now, the predictive probability is given as follows.

$$p(y = 1|D) = \sum_{W_2} p(y = y_i|x_i)p(W_2|D) \tag{3.12}$$

A thing to note here is that is that the computation for this toy model is tractable as we can exhaustively sum over 2^n possible combinations of the binary weights.

3.1.4 Results

The ground truth for the data used is shown in the Figure 3.3

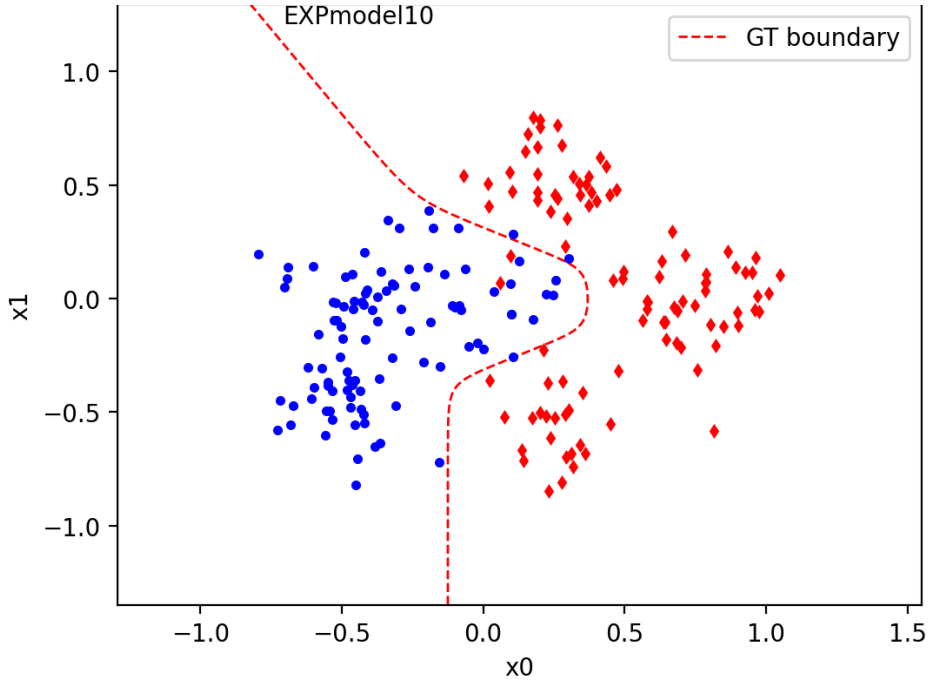
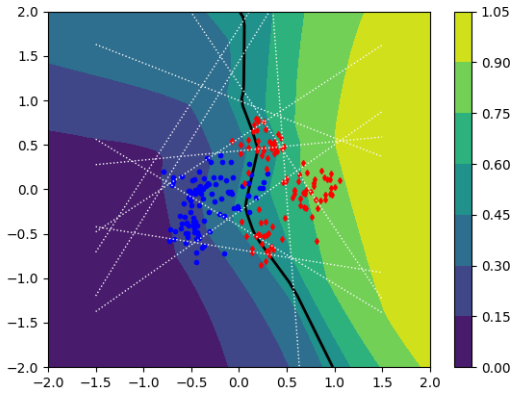
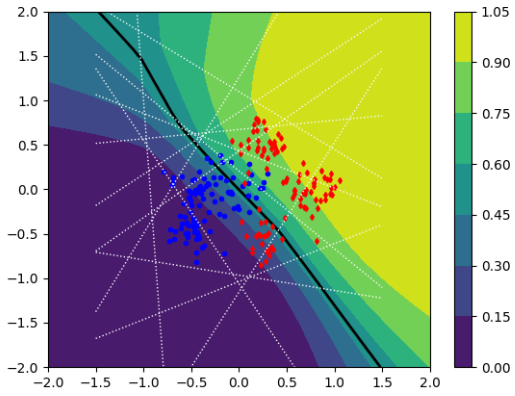


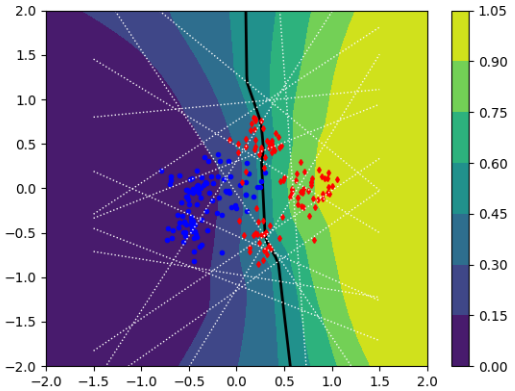
Figure 3.3: Figure showing the ground truth for the experiment. The blue points belong to first set of Gaussians and the red points belong to the second set of Gaussians. The dotted red line the line separating the two classes of points and is considered to be the ground truth for this model.



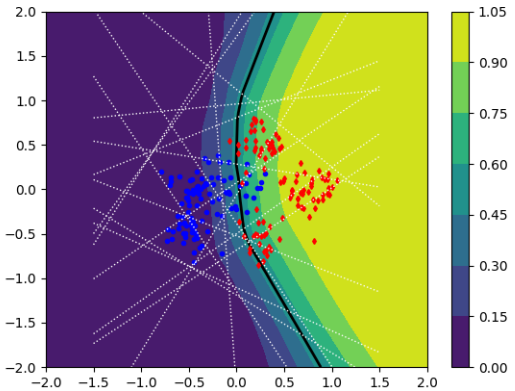
(a) Contour Plot for n = 10



(b) Contour Plot for n = 12



(c) Contour Plot for n = 14



(d) Contour Plot for n = 16

Figure 3.4: Figures visualizing the contour of the predictive probability. The black line represents where the p(y = 1 — x, D) = p(y = 2— x, D) = 0.5. The white dotted lines represent the random initializing of W_1 and b_1

Now, the Bayesian predictive distribution that comes up is shown in 3.4.

As seen from Figure 3.4, the resulting Bayesian distribution is not very accurate. The following reasons can be given to explain this.

- In this class of networks, there are no good models so a Bayesian model cannot do much in this scenario.
- A good set of models are needed to average over.

3.2 Variational Bayesian learning for Binary Neural Networks

In the model described in the previous section, we were not trying to learn the binary weights but rather which combination of weights gave us the best results. Here, we intend to learn the posterior distribution.

We consider a layer of binary weights w with $w_i \in \{0, 1\}$. Other weights and biases will be real-valued and fixed or their point estimates will be learned (not Bayesian).

For Bayesian learning we need a prior model of the weights (what we know without any data). We assume a factorized uniform prior: $p(w) = \prod_i p(w_i)$; $p(w_i{=}1) = p(w_i{=}0) = \frac{1}{2}$.

Let D denote our training data. The goal is to approximate the posterior distribution $p(w \mid D) = \frac{p(D|w)p(w)}{p(D)}$ with the variational distribution $q(w)$ that we choose to also factorize over the weights:

$$q(w) = \prod_i q(w_i), \tag{3.13}$$

where $q(w_i)$ is a Bernoulli distribution with outcomes $\{0, 1\}$ which can described by specifying either of:

- probability of state 1: $\theta_i = \mathbb{P}_q(w_i{=}1)$;
- logit: $\eta_i = \log \frac{\theta_i}{1-\theta_i}$.

It is trivial to convert one parameterization into other and is a matter of convenience which parameterization to use when.

The variational Bayesian problem is to minimize the KL divergence

$$\begin{aligned}\mathrm{KL}(q(w)\|p(w \mid D)) &= \sum_w q(w) \log \frac{q(w)}{p(w \mid D)} = \sum_w q(w) \log \frac{q(w)p(D)}{p(D \mid w)p(w)} \\ &= -\sum_w q(w) \log p(D \mid w) + \mathrm{KL}(q(w)\|p(w)) + \log p(D).\end{aligned} \tag{3.14}$$

The term $\log p(D)$ does not depend on q and thus can be omitted from minimization. Expanding data log-likelihood as the sum over all data points, we get

$$\log p(D \mid w) = \sum_j \log p(x_j \mid w) = \sum_j l_j(w). \tag{3.15}$$

When multiplying with $\frac{1}{N}$, the first term becomes the usual expected data likelihood, where the expectation is in training data and parameters $w \sim q(w)$. The VB problem reads

$$\min_\theta -\mathbb{E}_{w\sim \mathrm{Ber}(\theta)} \frac{1}{N} \sum_j l_j(w) + \frac{1}{N}\mathrm{KL}(q(w)\|p(w)) + const, \tag{3.16}$$

where $\mathrm{Ber}(\theta_i)$ shall denote Bernoulli variable with probability θ_i and $\mathrm{Ber}(\theta)$ a vector of independent such variables.

We employ mirror descent to handle constraints $\theta \in [0, 1]$ in this composite function, linearizing only the data part and keeping the prior KL part non-linear. Let

$$g = \frac{1}{N} \sum_j \nabla_\theta \mathbb{E}_{w\sim \mathrm{Ber}(\theta)} l_j(w)$$

be the stochastic gradient of the data term in the weight mean parameters θ. The (stochastic) mirror descent step solves the following proximal step problem:

$$\min_\theta \Big(g^\intercal \theta + \frac{1}{\varepsilon}\mathrm{KL}(\mathrm{Ber}(\theta)\|\mathrm{Ber}(\theta^t)) + \frac{1}{N}\mathrm{KL}(\mathrm{Ber}(\theta)\|\mathrm{Ber}(0)) \Big). \tag{3.17}$$

Notice that $\mathrm{KL}(\mathrm{Ber}(\theta)\|\mathrm{Ber}(0)) = -H(\mathrm{Ber}(\theta))$ – the entropy of Bernoulli distribution with probabilities θ. For brevity, we introduce the prior scaling coefficient $\lambda = \frac{1}{N}$ in front of the entropy (may be optionally be lowered to decrease the regularization effect). Let

θ^t be the current parameters. The composite proximal problem becomes

$$\min_{\theta} \Big(g^{\mathsf{T}}\theta + \frac{1}{\varepsilon}\mathrm{KL}(\mathrm{Ber}(\theta)\|\mathrm{Ber}(\theta^t)) - \lambda H(\mathrm{Ber}(\theta)) \Big). \tag{3.18}$$

To minimize this equation and solve for θ, we simply to differentiate equation 6.

$$\frac{\partial}{\partial\theta}\Big(g^{\mathsf{T}}\theta + \frac{1}{\varepsilon}\mathrm{KL}(\mathrm{Ber}(\theta)\|\mathrm{Ber}(\theta^t)) - \lambda H(\mathrm{Ber}(\theta)) \Big) = 0 \tag{3.19}$$

$$\frac{\partial}{\partial\theta}(g^{\mathsf{T}}\theta) + \frac{\partial}{\partial\theta}(\frac{1}{\varepsilon}\mathrm{KL}(\mathrm{Ber}(\theta)\|\mathrm{Ber}(\theta^t)) + \frac{\partial}{\partial\theta}(\lambda H(\mathrm{Ber}(\theta))) = 0 \tag{3.20}$$

Note that the entropy for a Bernoulli distribution is as follows,

$$H(\mathrm{Ber}(\theta)) = -\theta\ln(\theta) - (1-\theta)\ln((1-\theta) \tag{3.21}$$

$$g^{\mathsf{T}} + \frac{1}{\varepsilon}\frac{\partial}{\partial\theta}(\sum_i \theta_i \ln(\frac{\theta_i}{\theta_i^{\mathsf{T}}}) + (1-\theta_i)\ln(\frac{1-\theta_i}{1-\theta_i^{\mathsf{T}}}) + \frac{\partial}{\partial\theta}(-\theta_i\ln(\theta_i) - (1-\theta_i)\ln((1-\theta_i))) = 0 \tag{3.22}$$

$$g^{\mathsf{T}} + \frac{1}{\varepsilon}(\ln(\frac{\theta_i}{\theta_i^{\mathsf{T}}}) + \frac{\theta_i}{\theta_i} - \ln(\frac{1-\theta_i}{1-\theta_i^{\mathsf{T}}}) - \frac{1-\theta_i}{1-\theta_i}) + \lambda(-\ln(\theta_i) - \frac{\theta_i}{\theta_i} + \ln(1-\theta_i) + \frac{1-\theta_i}{1-\theta_i}) = 0 \tag{3.23}$$

$$g^{\mathsf{T}} + \frac{1}{\varepsilon}(\ln(\frac{\theta_i}{\theta_i^{\mathsf{T}}}) - \ln(\frac{1-\theta_i}{1-\theta_i^{\mathsf{T}}})) + \lambda(\ln(1-\theta_i) - \ln(\theta_i)) = 0 \tag{3.24}$$

Recall that the logit parameter is described as follows,

$$\eta_i = \log\frac{\theta_i}{1-\theta_i}. \tag{3.25}$$

Therefore, the following substitutions can be made.

$$g^{\mathsf{T}} + \frac{1}{\varepsilon}(\eta_i - \eta_i^{\mathsf{T}}) + \lambda(-\eta_i) = 0 \tag{3.26}$$

Now, solving for η_i, we get the following.

$$\eta_i = \frac{\eta_i^{\mathsf{T}} - \varepsilon g^{\mathsf{T}}}{1 - \lambda\varepsilon} \tag{3.27}$$

Algorithm 1:
Variational Bayesian Learning Using Mirror Descent Update

Inp. : training data $(x_i, y_i)_{i=1}^N$, predictive model $p(y|x; w)$ with binary weights w
Outp.: Factorized Bernoulli distribution over weights with probabilities θ
Initialize: $\eta = 0$ `/* natural parameters of Bernoulli over weights */`
for *iterations* $t = 0 \dots T$ **do**
  Compute weight probabilities $\theta = \text{sigmoid}(\eta)$;
  Sample (x, y) from the dataset `/* or a mini-batch */`
  Define $L(w) = -\log p(y_i|x_i, w)$;
  Use ARM estimator [17] to obtain g, gradient of $\mathbb{E}_{w\sim\text{Bernoulli}(\theta)} L(w)$ in θ;
  Update $\eta = \frac{\eta - \varepsilon g}{1 - \lambda\varepsilon}$

3.3 Finite Mixture Variational Baysian Learning

Let $\bar{\theta}_i^{(s)} \in [0, 1]$ be chosen at random from $\text{Beta}(\alpha, \alpha)$ and for the purpose of this section considered fixed for $s = 1 \dots S$, where S is the number of such samples.

Let the prior be given by the mixture model:

$$p(w) = \frac{1}{S} \sum_{s=1}^{S} \prod_i \text{Ber}(w_i; \bar{\theta}_i^{(s)}), \tag{3.28}$$

where S is the number of mixture components for each weight. It can be equivalently stated in terms of the mixture hidden component variable s with uniform categorical

distribution and the conditional distribution:

$$p(s) = \frac{1}{S}; \tag{3.29}$$

$$p(w|s) = \prod_i \mathrm{Ber}(w_i; \bar{\theta}_i^{(s)}); \tag{3.30}$$

$$p(w) = \sum_s p(s)p(w|s). \tag{3.31}$$

This prior is not uniform any more, however it can be made not too far away from the uniform distribution on average. This is affected by both: the choice of α and the choice of the number of samples. In particular, in the limit of the number of samples going to the infinity, for any α we have

$$p(w_i{=}1) \to \int_{-\infty}^{\infty} \mathrm{Beta}(\theta; \alpha, \alpha)\mathrm{d}\theta = \frac{1}{2}. \tag{3.32}$$

On the other hand, for α approaching infinity, $\mathrm{Beta}(\alpha, \alpha)$ approaches the delta distribution at $\frac{1}{2}$ and we can have a uniform prior with one sample only. However, small values of α will be beneficial for the variational approximation below. In practice we will choose the number of samples, guided by computation constraints and can chose the value of α as the trade-off between prioring the solution by a specific probability distribution $p(w)$ and the accuracy of the subsequent variational inference.

We then consider the variational approximate posterior in a form of a mixture of factorized distributions:

$$q(w) = \frac{1}{S}\sum_{s=1}^{S}\prod_i \mathrm{Ber}(w_i; \theta_i^{(s)}), \tag{3.33}$$

Again, the distribution $q(w)$ can be given as the marginal distribution of

$$q(w, s) = p(s)q(w|s), \tag{3.34}$$

$$\text{where } q(w|s) = \prod_i \mathrm{Ber}(w_i; \theta_i^{(s)}). \tag{3.35}$$

Let us consider the following variational learning of the mixture:

$$\min_q \mathrm{KL}(q(w, s)\|p(w, s|D)). \tag{3.36}$$

$$\mathrm{KL}(q(w,s)\|p(w,s|D)) = \sum_{s,w} q(w,s)\frac{\log q(w,s)}{\log p(w,s|D)} \tag{3.37a}$$

$$= \sum_{s} p(s)\mathrm{KL}(q(w|s)\|p(w|s,D)) \tag{3.37b}$$

Minimization in the collection of θ therefore decouples into independent minimizations for each s. The difference from 3.2 is only that for each s the prior is not uniform but is given by $p(w|s)$, which is a Bernoulli distribution with mean parameters $\bar{\theta}^{(s)}$. It can be derived similar to 3.2 that the resulting mirror descent update for η^s is:

$$\eta^{(s)} = \eta^{(s)}\frac{1}{1-\lambda\varepsilon} + \bar{\eta}^{(s)}\frac{1}{1-\lambda\varepsilon} - \varepsilon g^s \frac{1}{1-\lambda\varepsilon}, \tag{3.38}$$

where εg^s is the gradient of the expectation of the likelihood with $w \sim \text{Bernoulli}(\theta^s)$.

The resulting algorithm for finding the optimal mixture model is to apply Algorithm 1 for each mixture component independently using the update (3.38) which has a term keeping each $\eta^{(s)}$ close to its initialization $\bar{\eta}^{(s)}$. Therefore it is also reasonable to initialize $\eta^{(s)}$ to $\bar{\eta}^{(s)}$. The learning is thus equivalent to S instances of Variational Bayesian learning with the weight decay towards different random intitializations. In the end the mixture model is composed out of learned $\eta^{(s)}$ and we can sample from it a finite ensemble for testing.

Chapter 4

Conclusion

4.1 Comparison of existing methodologies

Table 4.1 shows the accuracy of each methodology discussed in 2 as claimed by the respective authors.

The CIFAR dataset was used as the primary source of validation only in [8].Courbariaux et al. [2] shows best results on CIFAR dataset. The rest of the work has been more focused on improving accuracy on the ImageNet dataset.

Methodology	Accuracy(%)
Courbariaux et al.	89.95
XNOR-Net	89.83
DoReFa-Net	83.92
Khan et al.	93.72

Table 4.1: Table comparing different methodologies on CIFAR-10 dataset

Here, clearly the model proposed by Khan et al. works the best and coincidentally happens to be based on Bayesian Learning.

4.2 Review of the derived Bayesian models

The basic Bayesian models that we used in this thesis did not provide us with satisfactory results. This was mainly because it did not have a good set of models to average over.

A Varitational Bayesian Learning using mirror descent method was also derived. Algorithm 1 described how how factorized distribution over weights with probability θ is learnt. In section 3.3, we saw how Algorithm 1 can be used to optimal mixture model for each independent mixture component.

To understand how these models perform on real world data, we would first to evaluate it's performance on the toy data used in this thesis and then, if the results are satisfactory, we would need to extrapolate this learning to a more complex binary neural network.

Bibliography

[1] Y Chen, T.-J. Yang, J. Emer, and V. Sze, "Understanding the limitations of existing energy-efficient design approaches for deep neural networks", *Energy*, vol. 2, no. L1, p. L3, 2018.

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1", *arXiv preprint arXiv:1602.02830*, 2016.

[3] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation", *arXiv preprint arXiv:1308.3432*, 2013.

[4] K. Helwegen, J. Widdicombe, L. Geiger, Z. Liu, K.-T. Cheng, and R. Nusselder, "Latent weights do not exist: Rethinking binarized neural network optimization", *Advances in neural information processing systems*, vol. 32, 2019.

[5] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4923–4932.

[6] D. Saad and E. Marom, "Training feed forward nets with binary weights via a modified chir algorithm", *Complex Systems*, vol. 4, no. 5, 1990.

[7] T. Simons and D.-J. Lee, "A review of binarized neural networks", *Electronics*, vol. 8, no. 6, p. 661, 2019.

[8] X. Meng, R. Bachmann, and M. E. Khan, "Training binary neural networks using the bayesian learning rule", in *International conference on machine learning*, PMLR, 2020, pp. 6852–6861.

[9] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights", *Advances in neural information processing systems*, vol. 27, 2014.

[10] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations", *Advances in neural information processing systems*, vol. 28, 2015.

[11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks", in *European conference on computer vision*, Springer, 2016, pp. 525–542.

[12] J. Bethge, C. Bartz, H. Yang, Y. Chen, and C. Meinel, "Meliusnet: Can binary neural networks achieve mobilenet-level accuracy?", *arXiv preprint arXiv:2001.05936*, 2020.

[13] M. Khan and W. Lin, "Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models", in *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 878–887.

[14] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients", *arXiv preprint arXiv:1606.06160*, 2016.

[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[16] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning", *arXiv preprint arXiv:1710.10628*, 2017.

[17] M. Yin and M. Zhou, "Arm: Augment-reinforce-merge gradient for stochastic binary networks", *arXiv preprint arXiv:1807.11143*, 2018.