**Abstract**

We propose an alternative training framework for Bayesian neural networks (BNNs), which is motivated by viewing the Bayesian model for supervised learning as an autoencoder for data transmission. Then, a natural objective can be invoked from the rate-distortion theory. Specifically, we end up minimizing the mutual information between the weights and the dataset with a constraint that the negative log-likelihood is smaller than a certain value. The classical Blahut-Arimoto algorithm for solving this kind of optimization is infeasible due to the intractable expectations over the weights and the dataset, so we develop a new approximation to the steps of the Blahut-Arimoto algorithm. Our method exhibits some attractive properties over the conventional KL-regularized training of BNNs with fixed Gaussian prior: firstly, improved stability during optimization; secondly, a more flexible prior which can be understood from an empirical Bayes viewpoint.

1 Rate-distortion Perspective on BNNs

Given a sample \( S := X \times Y \) with inputs \( X := \{ (x_i) \}_{i=1}^n \) and labels \( Y := \{ (y_i) \}_{i=1}^n \), consider a sender who would like to send \( Y \) to a receiver. Instead of transmitting the raw data, the sender may compress the data with an encoder \( q(w | S) \) and send the code \( w \) or the full encoder distribution. The receiver can then reconstruct the data by decoding the code \( w \) with the predefined decoder \( p(y | x, w) \):

\[
\hat{y}_i \sim q(y | x_i, S) := \int p(y | x_i, w)q(w | S)dw.
\]  

(1)

This data-transmission view motivates a new learning objective for Bayesian models based on information theory. Notice that Bayesian inference is a particular case where the true posterior is used as the encoder. However, the encoder \( q(w | S) \) is not necessarily the posterior \( p(w | S) \). It comes from an empirical view of the joint distribution \( p(S, w) \):

\[
p(S, w) = q(w | S)p^*(S) \quad \text{with} \quad p^*(S) = \prod_{i=1}^n p^*(x_i, y_i),
\]  

(2)

where we denote the “true” distribution of data as \( p^*(x, y) \), which is unknown and model independent. The data-transmission analogy of Bayesian inference offers us a direct way to relate code/weight complexity with model generalization. To see this, we analyze the limits of the encoder \( q(w | S) \). If we force \( w \) to memorize everything in \( S \) including the noise, e.g. to have an identity map, we have to pay a high price for the communication cost. However, the encoding may not be even useful for another sample since it is too specific for the sample \( S \). On the other hand, if \( w \) is independent of \( S \), then the receiver has no way to decode the message no matter how powerful the decoder is.

*Contributed during an internship at Amazon.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.
The trade-off between the complexity and the generalization can be formulated as a rate-distortion trade-off \[ I(w; S) \] [Cover and Thomas 2012]. The compression rate is measured by the mutual information \( I(w; S) \). The distortion function, denoted by \( d(w, S) \), is defined naturally as the negative log-likelihood. This argument is in line with the Minimum Description Length principle, which says that the best model among all equally good models is the one that leads to the best compression of the data. Specifically, the rate-distortion trade-off is expressed as the following optimization problem:

\[
\min_{q(w|S) \in \mathcal{Q}(S)} \left[ I(w; S) \equiv I(q(w|S)) \right] \quad \text{s.t.} \quad \mathbb{E}_{p^*(S)}[\mathbb{E}_{q(w|S)}d(w, S)] \leq D \tag{3}
\]

\[
I(q(w|S)) := \mathbb{E}_{p^*(S)}[\mathbb{E}_{q(w|S)}[\log \frac{q(w|S)}{q(w)}]], \quad d(w, S) := -\sum_{i=1}^n \log p(y_i|x_i, w), \tag{4}
\]

where \( \mathcal{Q}(S) \) is the set of properly normalized pdfs. A similar rate-distortion analysis for unsupervised representation learning has been conducted by [Alemi et al. 2017].

Note that \( q(w) = \sum_{S} p^*(S)q(w|S) \) is the aggregated posterior [Makhzani et al. 2015, Tomczak and Welling 2017]. This coupling term makes the optimization difficult to solve. We show in the following lemma how to convert equation (3) to an equivalent but more convenient problem.

**Lemma 1** (10.8.1 [Cover and Thomas 2012]). The mutual information has a variational form:

\[
I(X; Y) = \min_{m(y)} D_{KL}(p(x, y)||p(x)m(y)), \quad \text{where} \quad m^*(y) = p(y) = \int p(x, y)dx.
\]

By applying Lemma 1 to equation (3), we rewrite the rate-distortion trade-off as:

\[
\min_{m(w)} \min_{q(w|S) \in \mathcal{Q}(S)} I(q(w|S), m(w)) \quad \text{s.t.} \quad \mathbb{E}_{p^*(S)}[\mathbb{E}_{q(w|S)}d(w, S)] \leq D \tag{5}
\]

\[
I(q(w|S), m(w)) := \mathbb{E}_{p^*(S)}[\mathbb{E}_{q(w|S)}[\log \frac{q(w|S)}{m(w)}]]. \tag{6}
\]

One may see the connection to the empirical Bayes [Robbins 1985, Kucukelbir and Blei 2014], since equation (5) involves optimizing the “posterior” \( q(w|S) \) and the “prior” \( m(w) \) at the same time. This perspective links information theory and (empirical) Bayes at a model level rather than at an inference level.

### 2 Approximate Blahut-Arimoto Algorithm

The direct optimization of equation (5) is cumbersome, since we need to parameterize a high dimensional mapping \( q(w|S) \). Alternatively, we resort to the classical Blahut-Arimoto algorithm [Arimoto 1972, Blahut 1972]. We first write the Lagrange dual function of equation (5):

\[
F(\beta) := \min_{m(w)} \min_{q(w|S) \in \mathcal{Q}(S)} I(q(w|S), m(w)) + \beta \mathbb{E}_{p^*(S)}[\mathbb{E}_{q(w|S)}d(w, S) - D]. \tag{7}
\]

The Blahut & Arimoto algorithm for computing \( F(\beta) \) is simply an alternating minimization whose solution is characterized by the following fixed point equations:

\[
q(w|S) = \frac{m(w) \exp(-\beta d(w, S))}{\int m(v) \exp(-\beta d(v, S))dv} \quad \text{and} \quad m(w) = \sum_{S} p^*(S)q(w|S). \tag{8}
\]

Next, we make two approximations to the above equations:

1. We use a variational approximation \( q(w|\theta) \) for \( q(w|S) \) by solving

\[
\theta(S) = \arg \min_{\theta} D_{KL}(q(w|\theta)||q(w|S)) \tag{9}
\]

\[
= \arg \min_{\theta} D_{KL}(q(w|\theta)||m(w)) + \beta \mathbb{E}_{q(w|\theta)}[d(w, S)]. \tag{10}
\]

Following [Blundell et al. 2015], we parameterize \( q(w|\theta) \) as a Gaussian distribution.
2. We approximate $m(w) \simeq \sum_{S} p^*(S) q(w|\theta(S)) \simeq \frac{1}{K} \sum_{k=1}^{K} q(w|\theta(B_k)) =: \hat{m}(w)$, where $B_k$ is a bootstrap sample of size $n_b$ drawn from the empirical distribution $p_S(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x_i = x) \delta(y_i = y)$. Note that we only resample data to create $B_k$ if $n_b > n$.

We call the resulting approach $\beta$-BNN. The detailed algorithm is shown in Algorithm 1. Note that the step for updating $q(w|\theta)$ resembles the ELBO derived by Blundell et al. [2015] for vanilla BNNs, except that the coefficient $\beta$ is now formally introduced, and instead of setting $m(w)$ to be $N(0, I)$, $m(w)$ is approximated by a mixture of variational posteriors. It is clear that $n_b$ and $K$ determine how close we follow the classical Blahut-Arimoto steps. However, we do not want to take very large $n_b$ and $K$, since both steps are approximated. Inspired by this argument, we also consider an online version, where $\hat{m}(w)$ is updated whenever a new variational posterior is produced.

**Algorithm 1 Approximate Blahut-Arimoto Algorithm**

1: **Input**: $S$ (dataset), $\beta$ (coefficient), $K$ (# mixture components), $n_b$ (size of a bootstrap sample).
2: **Initialize**: $\Theta = \{\theta_k^{(0)} = (0, I)\}_{k=1}^{K}$; $\hat{m}(w) = \frac{1}{K} \sum_{\theta \in \Theta} q(w|\theta)$.
3: for all $t = 1, \ldots, T$ do
4: Draw $K$ bootstrap samples $\{B_k\}_{k=1}^{K}$ of size $n$ from $p_S(x, y)$.
5: for all $k = 1, \ldots, K$ do
6: $\theta_k^{(t)} \leftarrow n_b$ SGD steps on the loss of (10) initialized at $\theta_k^{(t-1)}$.
7: $\Theta = \Theta \cup \{\theta_k^{(t)}\} \setminus \{\theta_k^{(t-1)}\}$.
8: if do online update or $k = K$ then
9: $\hat{m}(w) = \frac{1}{K} \sum_{\theta \in \Theta} q(w|\theta)$.
10: **Output**: $\Theta$.

3 Experiments

We test $\beta$-BNN, online $\beta$-BNN and vanilla BNN [Blundell et al. 2015] on the colorful MNIST dataset [Bullen 2017], where each image is converted to RGB space and blended with a random background. We also test a fixed-prior $\beta$-BNN, which is a special case: $\hat{m}(w) \equiv N(0, I)$.

For this experiment, $T = 100$ is sufficient to converge: $q(w|\theta)$ and $\hat{m}(w)$ are specified in Section 2, $p(y|x, w)$ is implemented by a multilayer perceptron (Linear400-ReLU-Linear400-ReLU-Linear10-Softmax); $\theta_k^{(t)}$ is obtained by running SGD over $B_k$ once with batch size 128 and learning rate $10^{-3}$. The comparison is shown in Table 1, where the prediction is $\arg \max_x p(y|x, E[w])$. We allocate 10000 points as the validation set, and choose $\beta$ from 10 candidates ranging uniformly from $10^{-11}$ to $10^{-2}$. We set $K = 5$ for the comparison as the performance only increases marginally for $K \geq 5$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\beta^*$</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla BNN</td>
<td>$\frac{1}{n}$</td>
<td>90.05</td>
</tr>
<tr>
<td>Fixed-prior $\beta$-BNN</td>
<td>$10^{-10}$</td>
<td>95.86</td>
</tr>
<tr>
<td>$\beta$-BNN</td>
<td>$10^{-5}$</td>
<td>96.08</td>
</tr>
<tr>
<td>Online $\beta$-BNN</td>
<td>$10^{-3}$</td>
<td>97.12</td>
</tr>
</tbody>
</table>

Table 1: The comparison on colorful MNIST. We choose $n_b = 10000$, $K = 5$ for $\beta$-BNN. Then, at each iteration, it goes through the training set once. We choose $n_b = 128$, $K = 5$ for online $\beta$-BNN, and increase $T$ to visit the same amount of data. Note that vanilla BNN corresponds to fixed-prior $\beta$-BNN with $\beta = \frac{1}{n}$ and $K = 1$.

We empirically observe that fixed-prior BNNs suffer from slow convergence due to very large KL terms (about $10^6$). Thus, we need to choose very small $\beta$ (about $10^{-9}$) to compensate, which will not be scalable for much larger networks. The convergence comparison is shown in Figure 1, where we can see that online $\beta$-BNN converges to a better local minimum and enjoys a faster convergence.

![Figure 1: Testing accuracy over training epochs.](image-url)
References


