Learned Lossless Compression via an Extension of the Bayes Codes

Yuta Nakahara
Center for Data Science
Waseda University
Tokyo, Japan
y.nakahara@waseda.jp

Shota Saito
Gunma University
Faculty of Informatics
Gunma, Japan
shota.s@gunma-u.ac.jp

Koshi Shimada and Toshiyasu Matsushima
Waseda University
Dept. of Pure and Applied Math.
Tokyo, Japan
i_am_koshi@suou.waseda.jp and toshimat@waseda.jp

***Abstract*—Learned compression is an emerging scheme of data compression where the encoder and decoder are learned from data. In learned compression, various deep neural network-based methods are proposed. However, the idea of data-based code construction does not necessarily require deep neural networks. In this paper, we introduce a framework to incorporate this idea into classical codes via an extension of the Bayes codes.**

I. Introduction

Recently, a new scheme called learned compression has been used for data compression. In contrast to most classical data compression methods, in this scheme, the encoder and decoder are constructed based on a large amount of training data. This scheme is rapidly expanding due to the development of deep generative models. Various deep neural network (DNN) based coding methods have been proposed, e.g., variational auto-encoders [1] and recurrent neural networks [2].

However, the idea of data-based code construction does not necessarily require DNNs. It is also possible to incorporate this idea into classical codes. In particular, lossless compression using the Bayes codes [3] is suitable for incorporating this idea, because the Bayes codes assume a Bayesian statistical model as the data generation model, which can be hierarchically extended. In fact, we have already reported some examples in [4], [5]. In this paper, we generalize and summarize these results as a unified framework of learned compression.

We expect that statistical-model-based methods can be computationally more efficient and more easily avoid overfitting than DNN-based methods. There is also a possibility that statistical-model-based methods may be more suitable than DNN-based methods for lossless compression, where no distortion is allowed and theoretical limits of compression are well studied. In fact, many of the existing learned compression methods have been applied to lossy compression. Thus, we consider the DNN-based methods and the statistical-model-based methods should be studied with careful comparison. We hope this paper encourages the interplay between the two.

II. Preliminaries: the Bayes Codes [3]

We consider lossless compression of a sequence $\boldsymbol{x} = x_1 \dots x_n$, which is generated from a probability distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$. Here, we assume $\boldsymbol{\theta}$ is unknown and we cannot use the true distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ as a coding probability of entropy codes such as arithmetic codes [6]. Therefore, we estimate it by $\hat{p}(\boldsymbol{x})$. In the Bayes codes [3], we assume $\boldsymbol{\theta}$ follows a prior distribution $p(\boldsymbol{\theta}|\boldsymbol{\eta})$, where $\boldsymbol{\eta}$ is a given hyper-parameter. This enables us to adopt an estimation criterion for $p(\boldsymbol{x}|\boldsymbol{\theta})$. That is the Bayes risk function (see, e.g., [7]) based on the Kullback-Leibler (KL) information between the true distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ and an estimated distribution $\hat{p}(\boldsymbol{x})$. It is known that the optimal distribution minimizing this criterion is given as follows.

$$\hat{p}(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\eta}) \mathrm{d}\boldsymbol{\theta}. \tag{1}$$

The code where (1) is used as a coding probability of the arithmetic code is called the Bayes code. The expected code length of the Bayes code converges to the entropy of $p(\boldsymbol{x}|\boldsymbol{\theta})$ with the true $\boldsymbol{\theta}$ for sufficiently large data length, and its convergence speed achieves a theoretical limit [8].

Further, according to [3], we can derive a sequential coding algorithm where the following probability is used as the coding probability of the arithmetic code for the jth symbol x_j.

$$\hat{p}(x_j|x^{j-1}) = \int p(x_j|x^{j-1}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|x^{j-1}, \boldsymbol{\eta}) \mathrm{d}\boldsymbol{\theta}, \tag{2}$$

where x^{j-1} denotes $x_1 \cdots x_{j-1}$. It is known that the code length of the Bayes code using this coding probability coincides with that using (1). If we assume $p(\boldsymbol{\theta})$ is a conjugate prior for $p(\boldsymbol{x}|\boldsymbol{\theta})$, we can analytically solve the integral in (2) and repeat encoding of x_j and updating of $p(\boldsymbol{\theta}|x^j, \boldsymbol{\eta})$.

Remark 1: In [9], $p(\boldsymbol{x}|\boldsymbol{\theta})$ is assumed to be a context-tree model, which includes arbitrary (infinite) order Markov models as sub-models. Moreover, [9] has proposed an algorithm to calculate (2) for the context-tree model without any approximation, and its complexity is only $O(n)$. Therefore, the Bayes codes are not only for a trivial model like the Bernoulli model but applicable to a broad range of models.

III. Problem Setting

In this section, we extend the Bayes codes and mathematically formulate the problem of learned compression, i.e., constructing an encoder and decoder from training sequences for a target sequence. Let n_i denote the length of the ith sequence and we define $\boldsymbol{x}_i := x_{i,1} \cdots x_{i,n_i}$. Let $\boldsymbol{x}^m := (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m)$ be the training sequences and $\boldsymbol{x}_{m+1}$ be the target sequence. We assume each sequence $\boldsymbol{x}_i$ is independently generated according to a parameter $\boldsymbol{\theta}_i$. Let $\boldsymbol{\theta}^m := (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_m)$ be

the tuple of them. In addition, we assume the probability distribution of $\boldsymbol{x}_i$ has another parameter $\boldsymbol{\alpha}_i$. For $\boldsymbol{\alpha}_i$, we assume it becomes difficult to calculate integrals in the coding probability (1) or (2) for $\boldsymbol{x}_i$ if we assume any prior distribution on it. Therefore, we treat $\boldsymbol{\alpha}_i$ as an unknown constant. Using such a parameter, the model becomes more flexible. Consequently, $\boldsymbol{x}_i$ independently follows $p(\boldsymbol{x}_i|\boldsymbol{\theta}_i;\boldsymbol{\alpha}_i)$. We also define a tuple $\boldsymbol{\alpha}^m := (\boldsymbol{\alpha}_1,\dots,\boldsymbol{\alpha}_m)$. Moreover, we assume a prior distribution on $\boldsymbol{\theta}_i$. It includes an additional parameter $\boldsymbol{\beta}$ that is assumed to be an unknown constant because of a similar reason to $\boldsymbol{\alpha}_i$, and we assume $\boldsymbol{\theta}_i$ independently follows $p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta})$. Lastly, we assume $\boldsymbol{\eta}$ is unknown and assume a hyper-prior $p(\boldsymbol{\eta})$. An idea of a hyper-prior distribution also plays an important role in the DNN-based methods [1].

Under this condition, we define the problem of learned compression as a problem to construct a coding algorithm for the target sequence $\boldsymbol{x}_{m+1}$ using training sequences $\boldsymbol{x}_1,\dots,\boldsymbol{x}_m$.

Example 1: In our previous study on lossless text compression [4], $p(\boldsymbol{x}_i|\boldsymbol{\theta}_i;\boldsymbol{\alpha}_i)$ was the context-tree model, $p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta})$ was Dirichlet distributions and a tree distribution [10], and $p(\boldsymbol{\eta})$ was beta distributions. In our previous study on lossless image compression [5], $p(\boldsymbol{x}_i|\boldsymbol{\theta}_i;\boldsymbol{\alpha}_i)$ was a two-dimensional autoregressive model, $p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta})$ was a mixture of Gauss-gamma distributions, and $p(\boldsymbol{\eta})$ was a Dirichlet distribution and Gauss-Wishart distributions. Therefore, we could assume a multimodal pixel distribution represented as a mixture of autoregressive models in [5]. Non-linear autoregressive models are also used in the DNN-based methods [2].

IV. Learned Compression Algorithms

If $\boldsymbol{\alpha}_{m+1}$ and $\boldsymbol{\beta}$ are given, in a similar manner to [3], the optimal coding probability for $\boldsymbol{x}_{m+1}$ is given as follows:

$$p(\boldsymbol{x}_{m+1}|\boldsymbol{x}^m;\boldsymbol{\alpha}_{m+1},\boldsymbol{\beta}) = \iint p(\boldsymbol{x}_{m+1}|\boldsymbol{\theta}_{m+1};\boldsymbol{\alpha}_{m+1}) \times p(\boldsymbol{\theta}_{m+1}|\boldsymbol{\eta};\boldsymbol{\beta})p(\boldsymbol{\eta}|\boldsymbol{x}^m;\boldsymbol{\beta})\mathrm{d}\boldsymbol{\theta}_{m+1}\mathrm{d}\boldsymbol{\eta}. \quad (3)$$

However, we have some difficulties in calculating this. First, $\boldsymbol{\alpha}_{m+1}$ and $\boldsymbol{\beta}$ are unknown. Second, the integral for $\boldsymbol{\eta}$ is often computationally expensive. Therefore, we build in pre-determined point estimators $\hat{\boldsymbol{\alpha}}_{m+1}$, $\hat{\boldsymbol{\beta}}$, and $\hat{\boldsymbol{\eta}}$. $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\eta}}$ are estimated from $\boldsymbol{x}^m$ and shared between the encoder and decoder beforehand. $\hat{\boldsymbol{\alpha}}_{m+1}$ is estimated from $\boldsymbol{x}_{m+1}$ and sent to the decoder as a header information. Therefore, we use the following approximate coding probability.

$$\hat{p}(\boldsymbol{x}_{m+1}) = \int p(\boldsymbol{x}_{m+1}|\boldsymbol{\theta}_{m+1};\hat{\boldsymbol{\alpha}}_{m+1})p(\boldsymbol{\theta}_{m+1}|\hat{\boldsymbol{\eta}};\hat{\boldsymbol{\beta}})\mathrm{d}\boldsymbol{\theta}_{m+1}. \quad (4)$$

This method can be considered a kind of learned compression algorithm. It is because estimating $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ from $\boldsymbol{x}^m$ corresponds to constructing a coding algorithm for $\boldsymbol{x}_{m+1}$. Hereafter, we call an estimation phase for $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ a learning phase and we call a phase to estimate $\boldsymbol{\alpha}_{m+1}$ and coding $\boldsymbol{x}_{m+1}$ a compression phase. We will describe them in order.

A. Learning Phase

To estimate $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$, we can use any Bayesian estimation methods such as variational Bayesian (VB) methods and Markov chain Monte Carlo (MCMC) methods. Here, we use the VB methods because it is usually more scalable than the MCMC methods. In particular, the VB method simultaneously estimating a deterministic parameter such as $\boldsymbol{\alpha}^m$ and $\boldsymbol{\beta}$ is called the empirical variational Bayesian (EVB) method. For more scalability, we can also use the stochastic variational inference (SVI) method [11].

In EVB method, we approximate the joint posterior distribution $p(\boldsymbol{\theta}^m,\boldsymbol{\eta}|\boldsymbol{x}^m;\boldsymbol{\alpha}^m,\boldsymbol{\beta})$ by a *variational distribution* $q(\boldsymbol{\theta}^m,\boldsymbol{\eta})$, which satisfies the factorization property: $q(\boldsymbol{\theta}^m,\boldsymbol{\eta}) = q(\boldsymbol{\theta}^m)q(\boldsymbol{\eta})$.

It is known (see, e.g., [12]) that minimizing the KL information $\mathrm{KL}(q(\boldsymbol{\theta}^m,\boldsymbol{\eta})\|p(\boldsymbol{\theta}^m,\boldsymbol{\eta}|\boldsymbol{x}^m;\boldsymbol{\alpha}^m,\boldsymbol{\beta}))$ is equivalent to maximizing a function called *variational lower bound* $\mathrm{VL}(q;\boldsymbol{\alpha}^m,\boldsymbol{\beta})$. Further, it is also known (see, e.g., [12]) that the optimal variational distribution, which maximize $\mathrm{VL}(q;\boldsymbol{\alpha}^m,\boldsymbol{\beta})$, fulfills the following equations:

$$\ln q^*(\boldsymbol{\theta}^m) = \mathbb{E}_{q^*(\boldsymbol{\eta})}\left[\ln p(\boldsymbol{\theta}^m,\boldsymbol{\eta},\boldsymbol{x}^m;\boldsymbol{\alpha}^m,\boldsymbol{\beta})\right] + \mathrm{const.}, \quad (5)$$

$$\ln q^*(\boldsymbol{\eta}) = \mathbb{E}_{q^*(\boldsymbol{\theta}^m)}\left[\ln p(\boldsymbol{\theta}^m,\boldsymbol{\eta},\boldsymbol{x}^m;\boldsymbol{\alpha}^m,\boldsymbol{\beta})\right] + \mathrm{const.} \quad (6)$$

Simultaneously, we maximize the variational lower bound $\mathrm{VL}(q;\boldsymbol{\alpha}^m,\boldsymbol{\beta})$ with respect to $\boldsymbol{\alpha}^m$ and $\boldsymbol{\beta}$, and define

$$(\boldsymbol{\alpha}^*)^m,\boldsymbol{\beta}^* := \arg\max_{\boldsymbol{\alpha}^m,\boldsymbol{\beta}} \mathrm{VL}(q;\boldsymbol{\alpha}^m,\boldsymbol{\beta}). \quad (7)$$

However, $q^*(\boldsymbol{\theta}^m)$, $q^*(\boldsymbol{\eta})$, $(\boldsymbol{\alpha}^*)^m$, and $\boldsymbol{\beta}^*$ depend on each other. Therefore, we update them in turn from any initial values until the convergence. Let $q^{(t)}(\boldsymbol{\theta}^m)$, $q^{(t)}(\boldsymbol{\eta})$, $(\boldsymbol{\alpha}^{(t)})^m$ and $\boldsymbol{\beta}^{(t)}$ denote approximate solutions at the tth iteration. After convergence, we use the expectation or the mode of $q^{(\infty)}(\boldsymbol{\eta})$ and $\boldsymbol{\beta}^{(\infty)}$ as $\hat{\boldsymbol{\eta}}$ and $\hat{\boldsymbol{\beta}}$. In the following, we describe some discussions for calculating (5), (6), and (7).

1) Update of $q^{(t)}(\boldsymbol{\theta}^m)$ and $q^{(t)}(\boldsymbol{\eta})$: By calculating (5), an additional factorization is induced, and $q^{(t)}(\boldsymbol{\theta}^m) = \prod_{i=1}^m q^{(t)}(\boldsymbol{\theta}_i)$ holds. Each factor is represented as follows:

$$\ln q^{(t)}(\boldsymbol{\theta}_i) = \ln p(\boldsymbol{x}_i|\boldsymbol{\theta}_i;\boldsymbol{\alpha}_i^{(t-1)}) + \mathbb{E}_{q^{(t-1)}(\boldsymbol{\eta})}[\ln p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta}^{(t-1)})] + \mathrm{const.} \quad (8)$$

By calculating (6), $q^{(t)}(\boldsymbol{\eta})$ is represented as follows:

$$\ln q^{(t)}(\boldsymbol{\eta}) = \textstyle\sum_{i=1}^m \mathbb{E}_{q^{(t)}(\boldsymbol{\theta}_i)}[\ln p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta}^{(t-1)})] + \ln p(\boldsymbol{\eta}) + \mathrm{const.} \quad (9)$$

Remark 2: When the components of the assumed model are exponential families and they are locally conjugate each other, in most cases, (8) and (9) have closed-form parametric representations, and their updates are represented as the updates of their parameters. So, we can efficiently calculate them.

2) Update of $(\boldsymbol{\alpha}^{(t)})^m$ and $\boldsymbol{\beta}^{(t)}$: In a similar manner to general cases (e.g., [12]), the variational lower bound $\mathrm{VL}(q^{(t)};(\boldsymbol{\alpha}^{(t-1)})^m,\boldsymbol{\beta}^{(t-1)})$ is described as follows.

$$\begin{aligned}\mathrm{VL}(q^{(t)};(\boldsymbol{\alpha}^{(t-1)})^m,\boldsymbol{\beta}^{(t-1)}) &= \textstyle\sum_{i=1}^m \mathbb{E}_{q^{(t)}}[\ln p(\boldsymbol{x}_i|\boldsymbol{\theta}_i;\boldsymbol{\alpha}_i^{(t-1)})] \\ &+ \textstyle\sum_{i=1}^m \mathbb{E}_{q^{(t)}}[\ln p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta}^{(t-1)})] + \mathbb{E}_{q^{(t)}}[\ln p(\boldsymbol{\eta})] \\ &- \textstyle\sum_{i=1}^m \mathbb{E}_{q^{(t)}}[\ln q(\boldsymbol{\theta}_i)] - \mathbb{E}_{q^{(t)}}[\ln q(\boldsymbol{\eta})]. \end{aligned} \quad (10)$$

Therefore, regarding $(\boldsymbol{\alpha}^{(t-1)})^m$, we only have to maximize $\mathbb{E}_{q^{(t)}}[\ln p(\boldsymbol{x}_i|\boldsymbol{\theta}_i;\boldsymbol{\alpha}_i^{(t-1)})]$ for each $\boldsymbol{\alpha}_i^{(t-1)}$, and for $\boldsymbol{\beta}^{(t-1)}$, we

only have to maximize $\sum_{i=1}^m \mathbb{E}_{q^{(t)}}[\ln p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta}^{(t-1)})]$, i.e., we use the following updating formulas.

$$\boldsymbol{\alpha}_i^{(t)} = \arg\max_{\boldsymbol{\alpha}_i} \mathbb{E}_{q^{(t)}}[\ln p(\boldsymbol{x}_i|\boldsymbol{\theta}_i;\boldsymbol{\alpha}_i)] \tag{11}$$

$$\boldsymbol{\beta}^{(t)} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^m \mathbb{E}_{q^{(t)}}[\ln p(\boldsymbol{\theta}_i|\boldsymbol{\eta};\boldsymbol{\beta})] \tag{12}$$

Remark 3: In most cases, the objective functions in (11) and (12) have similar forms to log-likelihood functions of $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}$. Therefore, we can apply various methods for maximum likelihood estimation.

B. Compression Phase

1) Calculating Header Information: Before calculating (4), we have to estimate $\boldsymbol{\alpha}_{m+1}$ by $\hat{\boldsymbol{\alpha}}_{m+1}$, and send it to the decoder. We maximize the following formula. This is equivalent to minimizing the code length of the Bayes code for $\boldsymbol{x}_{m+1}$.

$$\ln \int p(\boldsymbol{x}_{m+1}|\boldsymbol{\theta}_{m+1};\boldsymbol{\alpha}_{m+1}) p(\boldsymbol{\theta}_{m+1};\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}}) \mathrm{d}\boldsymbol{\theta}_{m+1} \tag{13}$$

We may use any optimization algorithm, but the expectation maximization (EM) algorithm works well for this problem. For any probability distribution $r(\boldsymbol{\theta}_{m+1})$ such that the right-hand side of (14) can be defined, it is known that the following inequality holds (see e.g., [12]).

$$\begin{aligned}&\ln \int p(\boldsymbol{x}_{m+1}|\boldsymbol{\theta}_{m+1};\boldsymbol{\alpha}_{m+1}) p(\boldsymbol{\theta}_{m+1};\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}}) \mathrm{d}\boldsymbol{\theta}_{m+1} \\ &\geq \int r(\boldsymbol{\theta}_{m+1}) \ln \frac{p(\boldsymbol{x}_{m+1},\boldsymbol{\theta}_{m+1};\boldsymbol{\alpha}_{m+1},\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}})}{r(\boldsymbol{\theta}_{m+1})} \mathrm{d}\boldsymbol{\theta}_{m+1}\end{aligned} \tag{14}$$

We maximize this lower bound by repeating the following E-step and M-step from an initial value.

E-step: Fix a tentative value $\boldsymbol{\alpha}_{m+1}^{\mathrm{old}}$, then (14) holds as an equality when the following holds.

$$r(\boldsymbol{\theta}_{m+1}) = p(\boldsymbol{\theta}_{m+1}|\boldsymbol{x}_{m+1};\boldsymbol{\alpha}_{m+1}^{\mathrm{old}},\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}}) \tag{15}$$

Therefore, we update $r(\boldsymbol{\theta}_{m+1})$ by this formula. If we assume $p(\boldsymbol{\theta}_{m+1}|\boldsymbol{\eta};\boldsymbol{\beta})$ is a conjugate prior for $p(\boldsymbol{x}_{m+1}|\boldsymbol{\theta}_{m+1};\boldsymbol{\alpha}_{m+1})$, this posterior distribution has a closed-form parametric representation, and we can efficiently calculate it in a similar manner to Remark 2.

M-step: Fix $r(\boldsymbol{\theta}_{m+1}) = p(\boldsymbol{\theta}_{m+1}|\boldsymbol{x}_{m+1};\boldsymbol{\alpha}_{m+1}^{\mathrm{old}},\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}})$, then maximization of (14) is achieved when $\boldsymbol{\alpha}_{m+1}^{\mathrm{new}}$ is as follows.

$$\begin{aligned}\boldsymbol{\alpha}_{m+1}^{\mathrm{new}} = \arg\max_{\boldsymbol{\alpha}_{m+1}} &\int p(\boldsymbol{\theta}_{m+1}|\boldsymbol{x}_{m+1};\boldsymbol{\alpha}_{m+1}^{\mathrm{old}},\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}}) \\ &\times \ln p(\boldsymbol{x}_{m+1}|\boldsymbol{\theta}_{m+1};\boldsymbol{\alpha}_{m+1}) \mathrm{d}\boldsymbol{\theta}_{m+1}\end{aligned} \tag{16}$$

If $q^{(t)}(\boldsymbol{\theta}_i)$ and $p(\boldsymbol{\theta}_{m+1}|\boldsymbol{x}_{m+1};\boldsymbol{\alpha}_{m+1}^{\mathrm{old}},\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}})$ has the same form, which is often holds when we assume a conjugate prior, then (16) is equivalent to (11), and we can easily solve it.

2) Entropy Coding: Finally, we sequentially encode $\boldsymbol{x}_{m+1}$ in a similar manner to Sec. II. More specifically, we use the following coding probability for $x_{m+1,j}$ in the arithmetic code.

$$\begin{aligned}&\hat{p}(x_{m+1,j}|(x_{m+1})^{j-1}) \\ &= \int p(x_{m+1,j}|(x_{m+1})^{j-1},\boldsymbol{\theta}_{m+1};\hat{\boldsymbol{\alpha}}_{m+1}) \\ &\quad \times p(\boldsymbol{\theta}_{m+1}|(x_{m+1})^{j-1};\hat{\boldsymbol{\alpha}}_{m+1},\hat{\boldsymbol{\eta}},\hat{\boldsymbol{\beta}}) \mathrm{d}\boldsymbol{\theta}_{m+1},\end{aligned} \tag{17}$$

where $(x_{m+1})^{j-1}$ denote $x_{m+1,1}\cdots x_{m+1,j-1}$.

V. Numerical Results in Previous Papers

Here, we introduce numerical results obtained by applying our learned compression framework. Those results have been reported in our previous papers. In [4], we performed experiments on synthetic discrete sequences and real genome data. On synthetic data, the average code length was reduced by 2.08%. On real data, the average code length was reduced by 0.89%. In [5], we performed experiments on benchmark images, and the average code length was reduced by 5.02%.

VI. Conclusion

We introduced a framework to incorporate the idea of learned compression, i.e., the idea to construct the codes from data, into classical data compression methods via an extension of the Bayes codes. We generalized some specific examples reported in our previous papers [4], [5] and summarized them as a unified framework of learned compression. We hope this framework encourages the interplay between the DNN-based methods and the statistical-model-based methods.

Acknowledgment

This work was supported in part by JSPS KAKENHI Grant Number JP22K02811, JP22K14254, JP23K03863, JP23K04293, and by JST SPRING Grant Number JPMJSP2128.

References

[1] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rkcQFMZRb

[2] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5435–5443.

[3] T. Matsushima, H. Inazumi, and S. Hirasawa, "A class of distortionless codes designed by Bayes decision theory," *IEEE Transactions on Information Theory*, vol. 37, no. 5, pp. 1288–1293, 1991.

[4] Y. Nakahara, S. Saito, K. Shimada, and T. Matsushima, "Hyperparameter learning of bayesian context tree models," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 537–542.

[5] Y. Nakahara and T. Matsushima, "Hyperparameter learning of stochastic image generative models with bayesian hierarchical modeling and its effect on lossless image coding," in *2021 IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–6.

[6] J. Rissanen and G. Langdon, "Universal modeling and coding," *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 12–23, 1981.

[7] J. O. Berger, *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[8] B. S. Clarke and A. R. Barron, "Information-theoretic asymptotics of Bayes methods," *IEEE Transactions on Information Theory*, vol. 36, no. 3, pp. 453–471, May 1990.

[9] T. Matsushima and S. Hirasawa, "Reducing the space complexity of a Bayes coding algorithm using an expanded context tree," in *2009 IEEE International Symposium on Information Theory*, June 2009, pp. 719–723.

[10] Y. Nakahara, S. Saito, A. Kamatsuka, and T. Matsushima, "Probability distribution on full rooted trees," *Entropy*, vol. 24, no. 3, 2022. [Online]. Available: https://www.mdpi.com/1099-4300/24/3/328

[11] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 40, pp. 1303–1347, 2013. [Online]. Available: http://jmlr.org/papers/v14/hoffman13a.html

[12] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, January 2006. [Online]. Available: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/