(Vector) Quantization

Link: https://github.com/HengchaoChen/Quantization

1 Formulation

In this section, we formulate the problem of quantization. Let $\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K\} \subseteq \mathbb{R}^d$ be the quantization grid, which is also referred to as the codebook. Given any vector $\boldsymbol{a} \in \mathbb{R}^d$, the quantization procedure replaces $\boldsymbol{a}$ by the closest codebook vector in $\mathcal{S}$. Specifically, we replace $\boldsymbol{a}$ by

$$\boldsymbol{s}^{\boldsymbol{a}} = \operatorname*{argmin}_{\boldsymbol{s} \in \mathcal{S}} \|\boldsymbol{s} - \boldsymbol{a}\|, \tag{1.1}$$

where $\|\cdot\|$ indicates the L_2 norm. One brute force method to find $\boldsymbol{s}^{\boldsymbol{a}}$ is to compute the distance between $\boldsymbol{a}$ and every $\boldsymbol{s} \in \mathcal{S}$, and then determine the closest codebook vector. But, the computational complexity of this method is $\mathcal{O}(Kd)$, thus inefficient in the high dimensional case.

Example 1.1 (3-bit quantization)**.** In the toy example, the dimension $d = 1$ and the quantization grid is

$$\mathcal{S} = \{-4 + \frac{8i}{7} \mid i = 0, \ldots, 7\} \subseteq \mathbb{R}.$$

Given a vector $\boldsymbol{v} = (3.2, -1.4, 2.5, -0.9, 1.8, -3.7, 0.0, 4.0, 2.2, -1.3)$, the quantization problem is to apply (1.1) to each entry of $\boldsymbol{v}$. We can use arithmetic math to simplify the quantization procedure. Specifically, given $a \in \mathbb{R}$, we can calculate $z = (a + 4) * 7/8$ and find its closest integer in $\{0, 1, \ldots, 7\}$. When $z \leq 0$, the closest integer is 0. When $z \geq 7$, the closest integer is 7. In other cases, if $z - \mathrm{round}(z)$ is larger than 0.5, the closest integer is $\mathrm{round}(z) + 1$. Else, the closest integer is $\mathrm{round}(z)$.

2 Python Realization

In Quantization.ipynb, I provide a function quantization() that finds the quantized vector. Notably, I use vectorization to enhance the computational efficiency. The outcome of the result is

$$w = (6, 2, 5, 2, 5, 0, 3, 7, 5, 2).$$

3 Large Scale Vector Consideration

The computational cost is heavy when the sample size is large. However, as the quantization problem is parallelizable, we can use parallel computation to significantly enhance the computational efficiency. In particular, we can divide the large dataset into chunks and assign them to multiple devices, where the computation will be done in parallel.

4 Non-Uniform Quantization Grid and Vector Quantization

In contrast to Example 1.1, when the quantization grid is non-uniform, we cannot reduce the problem to each dimension separately. Instead, the quantization problem is exactly given by (1.1). Unlike Section 1, our current objective is to determine the optimal quantization grid. This topic is vector quantization [1].

The selection of the quantization grid depends on the feature vector $\boldsymbol{v}$ and and the codebook size K. Once the feature vector $\boldsymbol{v}$ and the size K are fixed, the optimal quantization grid is given as follows

$$\mathcal{S}^* = \underset{\mathcal{S}=\{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_K\}\subseteq\mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} e_i^2(\mathcal{S}), \tag{4.1}$$

where $e_i(\mathcal{S}) = \min_{\boldsymbol{s}\in\mathcal{S}} \|\boldsymbol{s} - \boldsymbol{v}_i\|$ is the quantization error of the ith entry of the feature vector and d is the dimension of the feature space. The objective function in (4.1) represents the total quantization error once the codebook $\mathcal{S}$ is fixed. Thus, the optimal codebook is to minimize this objective function over all possible codebooks.

This is a discrete optimization problem. In fact, we can view problem (4.1) as a K-means problem, where we find K groups and use the group mean as the codebook vector such that the total quantization error is minimized. So we can use any K-means algorithm to find the optimal codebook. For example, we can use multiple random initialization and for each initialization, we iteratively estimate the group mean and assign the data to a given group. This is the logic behind the K-means algorithm.

However, there exist two limitations in this formulation (4.1) and the K-means algorithm.

- First, it only focuses on minimizing the quantization error for a given dataset $\boldsymbol{v}$, the result of which may not be able to generalize. In fact, the dataset $\boldsymbol{v}$ can be a small sample of a large dataset, and the goal is to determine a codebook that works well for the large dataset.
- Second, the K-means algorithm requires multiple random initialization, which brings heavy computational cost.

4.1 One Dimensional Non-Uniform Quantization

In this section, we discuss one-dimensional non-uniform quantization. [2] suggests the use of NF (normal float) in quantizing neural network weights. Following Figure 7 in [2], we are able to create 3 bit NF as follows

3-bit NF = [-1.0000, -0.4457, -0.2008, 0.0000, 0.1487, 0.3133, 0.5260, 1.0000].

Then we rescale feature vector $\boldsymbol{v}$ to the interval $[-1, 1]$ and then perform the quantization. If we still use the feature vector $\boldsymbol{v}$ in Example 1.1, we will obtain the quantized vector as

$$w = (6, 1, 6, 1, 5, 0, 3, 0, 6, 1).$$

This quantized vector is different from the one using the uniform quantization grid.

Remark 4.1. In practice, one can further improve the quantization procedure by using sensitivity based non-uniform quantization [3] and using outlier isolation [3, 4]. This can improve the model performance in terms of the perplexity.

References

[1] Siegfried Graf and Harald Luschgy. *Foundations of quantization for probability distributions*. Springer Science & Business Media, 2000.

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc., 2023.

[3] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

[4] Hanlin Tang, Yifu Sun, Decheng Wu, Kai Liu, Jianchen Zhu, and Zhanhui Kang. Easyquant: An efficient data-free quantization algorithm for llms. *arXiv preprint arXiv:2403.02775*, 2024.