

Vector Quantization as Sparse Least Square Optimization

Chen Wang^{1,2} and Ruisen Luo^{*1}

¹College of Electrical Engineering and Information Technology, Sichuan University,
24 South Section 1, One Ring Road, Chengdu, China, 610065

²Department of Computer Science, University College London, Gower Street,
London, United Kingdom, WC1E 6BT

Abstract

Vector quantization aims to form new vectors/matrices with shared values close to the original. It could compress data with acceptable information loss, and could be of great usefulness in areas like Image Processing, Pattern Recognition and Machine Learning. In recent years, the importance of quantization has been soaring as it has been discovered huge potentials in deploying practical neural networks, which is among one of the most popular research topics. Conventional vector quantization methods usually suffer from their own flaws: hand-coding domain rules quantization could produce poor results when encountering complex data, and clustering-based algorithms have the problem of inexact solution and high time consumption. In this paper, we explored vector quantization problem from a new perspective of sparse least square optimization and designed multiple algorithms with their program implementations. Specifically, deriving from a sparse form of coefficient matrix, three types of sparse least squares, with l_0 , l_1 , and generalized $l_1 + l_2$ penalizations, are designed and implemented respectively. In addition, to produce quantization results with given amount of quantized values (instead of penalization coefficient λ), this paper proposed a cluster-based least square quantization method, which could also be regarded as an improvement of information preservation of conventional clustering algorithm. The algorithms were tested on various data and tasks and their computational properties were analyzed. The paper offers a new perspective to probe the area of vector quantization, while the algorithms proposed could provide more appropriate options for quantization tasks under different circumstances.

Keywords— Vector Quantization, l_0 Least Square Optimization, l_1 Least Square Optimization, Clustering, Algorithm Designing

^{*}Corresponding Author, email: rsluo@scu.edu.cn

1 Introduction

Vector quantization takes vectors/matrices to produce new ones with shared value and acceptable difference from the originals. By reducing the number of values in a vector/matrix, quantization methods could compress information and therefore reduce the storage cost. Quantization has been found great usefulness in areas like image processing[CORG93], speech recognition[SRZ⁺16], and Machine Learning techniques[JDS11]. And recently, with the growing research interests in deploying neural networks on storage-scarce edge devices, vector quantization techniques have grasped considerable attention because of its ability in reducing network (storage) size[FEG01] [HMD15] [SCYE17]. This research also originated from the explorations in the quantization of embedded neural networks, thus it aims to explore 1-d vector quantization algorithms.

Conventional vector quantization methods usually utilize hand-coding domain rules and/or clustering-based methods to quantize the values[SCYE17]. Common approaches include uniform quantization, logarithm quantization and k-means clustering quantization. While these approaches are straightforward and convenient to use, they frequently suffer from the several problems: 1. inadequately-reliable performance. Domain quantization usually produce low-quality outcomes, and it will lead to significant information loss; k-means quantization could usually produce high-quality results for the holistic results, but it could include empty clusters or irrational values (say, out-of-range values) when the number of quantized values are large; 2. inexact results. The results of domain quantization depend on the choice of domain, and k-means clustering is a heuristic method that cannot guarantee the optimal solution and the results are subject to random initialization; and 3. high time-consumption. While K-means clustering method could usually lead to solution with insignificant information loss, the time consumption of this method is significant, especially when the amount of quantized values is large.

In this paper, we intend to research the quantization algorithm from another perspective: sparse least square optimization. We consider the sparse-inducing properties of l_0 and l_1 norm-regularizations and designed algorithms to minimize the difference between the original and the sparsely-constructed vectors. To optimize the performance based on its computational properties, an alternative version with both l_1 and l_2 norm-regularization is also explored and implemented. Furthermore, to design a least square quantization method that could produce results with certain amounts of quantized values (instead of the value of penalization coefficient λ), we combined k-means clustering with least square optimization and produced another novel algorithm with a closed-form solution. Interestingly, this new approach could also be interpreted as an improvement of the conventional k-means clustering quantization method. Notice that our algorithms are very similar to sparse compression in signal processing[EFM10]: the difference is that in our problem the constructed vector should have shared values, while in sparse signal processing it only demands the sparse vector to be able to produce a vector close to the original signal.

To this end, the rest of the paper is arranged as follows: section 2 will be introducing related work in the field, including research outcomes from quantization algorithms and sparse coding processing; section 3 will be introducing our designed algorithms mathematically and analyze their optimization schemes and computational properties; the experimental results of the algorithms are shown, compared and analyzed in section 4 and the properties of the algorithms are examined; and finally, a conclusion is drawn in section 5 and further research related to this paper is discussed.

2 Related Work

The goal of vector quantization is relatively straightforward to achieve in usual situations, thus there are not many complicated methods to address this task. [SCYE17] provides a brief survey for basic methods of vector quantization, which include domain-based uniform and logarithm quantizations and clustering-based techniques such like K-means quantization. The idea of quantizing vectors with clustering methods provides us an open skeleton, in which we could plug novel clustering techniques to produce new quantization algorithms[Wan17]. [NH92] offers an alternative

technique to use Mixture of Gaussian method to perform quantization specifically for neural networks, and a recent paper [UMW17] re-examined this idea and formally designed it to be used for neural network compression and provided a mathematical justification for it. Other techniques to perform vector quantization include [VH11], which utilized divergence instead of distance metric as the measurement and derived an algorithms based on it; [MBS93], which designed a neural network to perform vector quantization; and [HHSZ14], which considered pairwise dis-similarity as the metric for quantization. To the best of our knowledge, hitherto there has not been publications discussing quantization algorithms as sparse least square optimization, thus our paper should be the pioneering work in the area.

There are plenty of academic publication discussing the applications of vector quantization, and recently, academic projects lying in this area have been growingly connected to neural networks, as the ability of quantization in compressing model size is being exploited in implementing edge-device neural networks. [HMD15] proposed a general pipeline to reduce model storage, and an important part of it is quantization. And similar with [UMW17] mentioned above, [LTA15] specifically designed a quantization method for neural networks. [GLYB14] directly utilized existed vector quantization techniques in network compressing and illustrated that the technique could be ideal for modifying neural network precisions. As mentioned in the introductory section, the origin of the algorithms was also the task to solve neural network weight-sharing problem, and a set of NN-based parameters is one of the test datasets in the experimental section.

The algorithm proposed in this work has significant similarity with compressive sensing (sparse signal processing) in terms of regularization idea and optimization target functions [EFM10] [CR07] [WMM⁺10]. Typical approaches to induce sparsity in compressive sensing algorithms are to introduce l_0 norm [TG07], l_1 norm [Tib11][YZ11] and/or $l_{2,1}$ norm [NHCD10] to the target optimization functions. And similarly, our algorithms also utilize these techniques to perform sparsity. Meanwhile, since l_1 norm is not everywhere differentiable and l_0 norm is not even convex, there also exist plenties of algorithms devoted to efficiently solve the optimization problems [SFR07] [GSBF17] [Wri15]. In this paper, we use distinct optimization schemes for different types of target functions, including a newly-proposed algorithm to program l_0 optimization [HM18].

3 Quantization Algorithms

3.1 Problem Setting

The vector quantization task could be described as follows: suppose we have a vector \mathbf{w} that has m distinct values. Now we intend to find a vector \mathbf{w}^* with p distinct values, where $p \leq n$. Alternatively, if we use a more strict constraint with $p \leq l$, which l is a given value we require. Then by denoting the difference between the original vector and the constructed one with l_2 norm, our original target function could be formed as:

$$\begin{aligned} & \underset{\mathbf{w}^*}{\text{minimize}} && ||\mathbf{w} - \mathbf{w}^*||_2^2 \\ & \text{subject to} && \mathbf{o}(\mathbf{w}^*) \leq l \end{aligned} \tag{1}$$

Where $\mathbf{o}(\cdot)$ means the distinct values of a vector. And notice that here we only consider \mathbf{w} in vector form. If the data is coded in a matrix, such like neural network parameters and images, then we could simply 'flatten' the matrix into a vector to perform quantization.

3.2 Original Algorithm with l_1 Norm Regularization

To begin with, we firstly change \mathbf{w} into $\hat{\mathbf{w}} = \mathbf{o}(\mathbf{w})$, which we directly operate on distinct values and recover the full vector by indexing later. And by this operation we will be needing to construct a new vector with length m and p distinct values. Then we could assume there exist a 'base' vector \mathbf{k} with shape $[k \times 1]$, where k is a number given by us. This vector could be generated randomly or by picking from the original vector (we will drop this in the final form, here it is introduced for derivation purpose), and a $[m \times k]$ transformation matrix Ψ , which we form the constructed vector

as:

$$\mathbf{w}^* = \mathbf{\Psi} \mathbf{k}$$

However, optimizing with above representation could not bring the property of sparsity. Here we introduce another matrix \mathbf{A} , with $\mathbf{\Psi}^* = \mathbf{A}\mathbf{\Psi}$, and each entry of $\mathbf{\Psi}^*$ should be:

$$\Psi_{i,j}^* = \sum_{c=0}^i \alpha_c \Psi_{c,j} \quad (2)$$

By designing the matrix with this addition form, we could be able to achieve 'same values' when there exist $\alpha_c = 0$. Equation 2 could be achieved by designing matrix \mathbf{A} as a lower-triangular matrix (with main diagonal on):

$$\mathbf{A} = \begin{bmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ \alpha_1 & \alpha_2 & 0 & \cdots & 0 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix}$$

And now we have two matrices, \mathbf{A} and $\mathbf{\Psi}$, to control the constructed vector. Intuitively, if we add l_1 and/or l_0 norm regularization on the target function, it will be possible for us to produce vector with shared values. Here we consider l_1 in the first place because it is continuous and convex. Our optimization target would become:

$$\begin{aligned} & \min_{\alpha, \psi} \|\hat{\mathbf{w}} - \mathbf{w}^*\|_2^2 + \lambda \|\alpha\|_1 \\ & = \min_{\alpha, \psi} \|\hat{\mathbf{w}} - \mathbf{A}\mathbf{\Psi} \mathbf{k}\|_2^2 + \lambda \|\alpha\|_1 \end{aligned} \quad (3)$$

Equation 3 is hard to optimize. Nevertheless, to determine the system we could fix $\mathbf{\Psi}$ and only optimize α . We now consider to fix the $\mathbf{\Psi}$ matrix as a 'ones' matrix:

$$\mathbf{\Psi}^* = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

And we will get a $[m \times 1]$ vector \mathbf{k}^* , with each of its components has identical values. We use N_p to denote this value, and in practice, we could use the summation of the original vector $\hat{\mathbf{w}}$ to represent this value. And now, our optimization target becomes:

$$\min_{\alpha, \psi} \|\hat{\mathbf{w}} - \mathbf{A} \mathbf{k}^*\|_2^2 + \lambda \|\alpha\|_1 \quad (4)$$

And this is equivalent to form a vector of α and a lower-triangular matrix with N_p values:

$$\mathbf{N}_p^* = \begin{bmatrix} N_p & 0 & 0 & \cdots & 0 \\ N_p & N_p & 0 & \cdots & 0 \\ N_p & N_p & N_p & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ N_p & N_p & N_p & \cdots & N_p \end{bmatrix}$$

And in practice, for computational convenience, we could first sort the original vector to form $\hat{\mathbf{w}}_* = \text{sorted}(\hat{\mathbf{w}})$ and add a scaling hyper-parameter δ to the original summation value for computational stability. The final optimization target with l_1 regularization will be as follows:

$$\min_{\alpha} \|\hat{\mathbf{w}}_* - \delta \mathbf{N}_p^* \alpha\|_2^2 + \lambda \|\alpha\|_1 \quad (5)$$

Equation 5 is very similar to the optimization target in compressive sensing. Nevertheless, there are two significant differences: firstly, the root of the target function and the derivations are different from those in compressive sensing; and secondly, the produced vector \mathbf{w}^* will be a quantized vector

instead of simply a sparse vector close to the original as in compressive sensing. The optimization of this formula is not very hard: the target function is a typical LASSO problem, and it is solved by **Coordinate Descent** with LASSO solvers in SK-learn in our program [PVG⁺11].

3.3 Revised l_1 Regularization Algorithm and l_0 Regularization Algorithm

One problem with the algorithm in equation 5 is that the optimization tends to concentrate the values around one center and only make very small differences between values (see section 4 for details), thus it will performance poorly when encountering dispersedly-distributed data. To avoid the problem, in this paper we consider to add **negative l_2 norm penalization** to encourage large and dispersed non-zero values. The revised form could be denoted with the following formula:

$$\min_{\alpha} \|\hat{w}_* - \delta N_p^* \alpha\|_2^2 + \lambda_1 \|\alpha\|_1 - \lambda_2 \|\alpha\|_2^2 \quad (6)$$

Equation 6 is like a **generalized Elastic Net**, since conventional E-net only permits positive co-efficients as parameters. There are rare, if any, integrated Lasso optimization packages that permits the parameter setting like equation 6. Thus, this algorithm is implemented via tensorflow with **Proximal Adagrad** optimizer in our program.

Another variation of the algorithm based on 5 could be to replace the l_1 norm with l_0 norm. In this algorithm, instead of directly add a penalization term, we explicitly set limitations of the number of distinct values:

$$\begin{aligned} \min_{w^*} \quad & \|\hat{w}_* - \delta N_p^* \alpha\|_2^2 \\ \text{subject to} \quad & \|\alpha\|_0 \leq p \end{aligned} \quad (7)$$

Where p is a number that we manually set, which indicate the upper bound of the amount of distinct values. The optimization of l_0 norm is NP-hard [AK98], thus we could only be solve by heuristic-based algorithms. In this paper we utilize the recent-proposed *L0Learn* package [HM18], and support the number of k up to 100. However, one should notice that the l_0 optimization method is not universal, which means, it could not reach arbitrary required number of values under our settings.

3.4 Clustering-based Least Square Sparse Optimization

One drawback of 5 and the revised 6 is that they could not explicitly indicate the number of demanded clusters/distinct values. And for the algorithm with equation 7, we could only set the upper bound of the amount of distinct values and there are no guarantees for how many values will finally be produced. To tackle this indefinite-amount problem, here we discuss a general target that could produce definite amount of values with least square form, and design a basic method based on the combination of K-means clustering and least square optimization.

Suppose we want to construct a vector with p distinct values now, and here we directly set the parameter vector α to a vector with p entries ($[p \times 1]$ shape). And now we need to use a transformation matrix to transform the p -value vector into a $[m \times 1]$ vector while maintaining the values constructed. One possible scheme could be to use a $n \times p$ transformation matrix E with one-hot encoded at each row. Under this scheme, the optimization target will be as following:

$$\begin{aligned} \min_{E, \alpha} \quad & \|\hat{w}_* - EN_p^* \alpha\|_2^2 \\ \text{subject to} \quad & \|E_i\|_0 = 1, \forall i = 1, 2, \dots, m \\ & \|E_i\|_1 = 1, \forall i = 1, 2, \dots, m \end{aligned} \quad (8)$$

The constraint of E in equation 8 means that for each row in the matrix, it will be 1 if we want the corresponding value to be belonging to this cluster; otherwise, it will 0. An alternative expression of the matrix will be:

$$E_{ij} = \begin{cases} 1, & I(\hat{\mathbf{w}}_*) = j \\ 0, & \text{else} \end{cases}$$

here $I(\hat{W}_i)$ means the group(cluster) which the i th value belongs to. The optimization would be difficult to perform with two optimization variables changing simultaneously. Here, we propose one simple method to deal with this problem: we firstly use clustering methods (e.g. K-means) to obtain $I(\hat{\mathbf{w}}_*) = K(\hat{\mathbf{w}}_*)$, and then we could get matrix \mathbf{E} . Then the target function could be further transferred into the following expression:

$$\begin{aligned} & \min_{\alpha} \|\hat{\mathbf{w}}_* - \mathbf{E} \mathbf{N}_p^* \alpha\|_2^2 \\ & = \min_{\alpha} \|\hat{\mathbf{w}}_* - \hat{\mathbf{N}}_p^* \alpha\|_2^2 \end{aligned} \quad (9)$$

Where the matrix $\hat{\mathbf{N}}_p^*$ would be a matrix like follows:

$$\hat{\mathbf{N}}_p^* = \begin{bmatrix} N_p & 0 & 0 & \cdots & 0 \\ N_p & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ N_p & 0 & 0 & \cdots & 0 \\ N_p & N_p & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ N_p & N_p & N_p & \cdots & N_p \end{bmatrix}$$

The optimization of equation 9 is a typical linear regression problem and could be solved in closed form with $O(p^2m + p^3)$ time complexity. By taking derivatives and set it to 0, we could obtain the solution:

$$\alpha = (\hat{\mathbf{N}}_p^{*T} \hat{\mathbf{N}}_p^*)^{-1} \hat{\mathbf{N}}_p^{*T} \hat{\mathbf{w}}_* \quad (10)$$

One interesting point is, from the perspective of clustering methods, equation 10 could be viewed as an improvement of K-means clustering quantization. In conventional clustering-based quantization algorithm we simply use the center points as the quantization results. Here, we alternatively use the point that produce the least distance with the original.

Notice that there should exist multiple schemes to solve the optimization problem proposed by equation 8, and the method proposed here is only a basic solution. The exploration of solving this task could be our future research concentrations.

4 Experimental Results

To verify the rationality and effectiveness of the proposed methods and make a comparison with existed methods, we test the algorithms with three kind of data: DNN weight matrix, MNIST Image, and artificial data sampled from different distributions. In the experiments, we focus on the **information loss, time-consumption and the characteristic of the results**, and analyze the reason behind the outcomes and the conditions in which our method(s) will be preferable. The information loss is denoted by the l_2 difference between the original vector and the quantized vector. Notice that in addition to loss value comparisons, we also use relative loss comparison to compare the trend of the loss. The relative loss comparison is computed by:

$$loss_r = \frac{loss - loss_0}{(loss - loss_0)_n} \quad (11)$$

Where $loss$ is a vector that collects l_2 differences w.r.t. clusters/ λ values. $loss_0$ denotes the first loss of the loss vector, and $loss_n$ means the last component of the vector. Notice that a high l_2 loss might not be as bad as it seems. For example, for data some distribution, the large amounts of quantized values away from original will produce high l_2 loss, but the performance might be only depended on the key quantized values, which are not far away from ground truth (see the

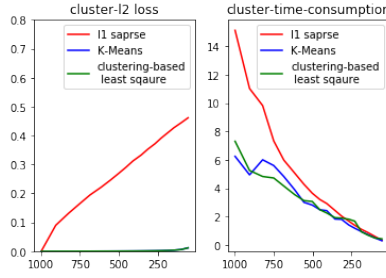


Figure 1: loss and time comparison for Neural Network weight Matrix

comparison in figure 7 as an example). Nevertheless, in this paper we choose it as the metric to evaluate information loss.

Major experiments includes the comparison between k-means and proposed l_1 (5 under coordinate descent) and clustering-based least square with clustering methods (9); analysis of computational properties of l_1 optimization and comparison between original l_1 (5) and revised $l_1 + l_2$ regularization (6) methods (under Proximal Adagrad); cluster number-difference analysis and cluster number-time consumption analysis; and the results produced by l_0 -optimization method (7).

Experimental results shows that in general, 1. the clustering-based least square method enjoys the best performance in terms of information loss, and the additional time consumption comparing to K-means method is not significant; 2. K-means method remains better than l_1 and $l_1 + l_2$ in terms of information preservation, as l_1 -based method (under coordinate descent) tends to concentrate weights to the central. However, for quantization with large number of clusters (distinct values), the information loss of l_1 is acceptable, and if the amount of values in the original vector/matrix is in relatively small magnitude, l_1 -based method could considerably save computation time; 3. the revised method with $l_1 + (\text{negative})l_2$ optimization could outperform that with l_1 solely (under the setting of Proximal Adagrad); 4. l_1 and $l_1 + l_2$ revised methods perform better on Gaussian-distributed data and are less competitive on uniformly-distributed ones; and finally, 5. l_0 -based quantization method (under the setting of *L0 Learner*) could provide good performance within acceptable running time, it could not universally produce quantization results (some amounts of quantization amounts are irretrievable) and the optimization could fail under some circumstances (especially when the demanded quantization amount is large).

4.1 DNN Weight Matrix

As stated in the previous sections, the research was originated from the task of neural network weights quantization problem. To test the effectiveness of our methods on neural network weights, we pick a 100×10 weight matrix of a layer (1000 parameters) of a fully-connected network, and performed quantization with the methods described above. The results of loss and time-consumption comparisons between l_1 method, K-means clustering and clustering-based least square method are shown as figure 1.

From the figure we could find out that in this case l_1 method doesn't show any advantages over clustering method: this is because of the number of parameters, as the optimization time of l_1 will increase as the number of data raise. However, we could find the l_2 loss of sparse least square method is still acceptable. And from the time consumption figure we could find out that clustering-based least square method doesn't spend significantly much time than K-means method. Considering the variation in the time of K-means algorithm itself, sometime the l_2 -based least square method even cost less time than K-means algorithm does. And if we take $l_1 + l_2$ method into consideration, and normalize the loss with equation 11, then we could get the result of figure 2. The left side of this figure shows the trend of losses under different algorithms, and the right hand of the figure shows the comparison between number of clusters of l_1 (coordinate descent) and $l_1 + l_2$ (proximal Adagrad). Notice that to produce the similar number of clusters

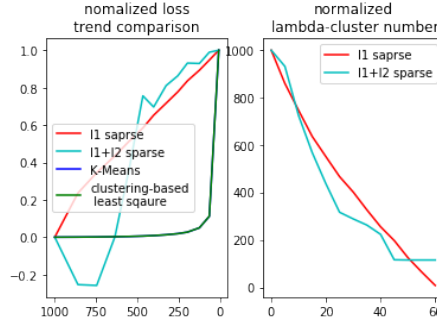


Figure 2: relative loss trend and lambda-cluster comparison

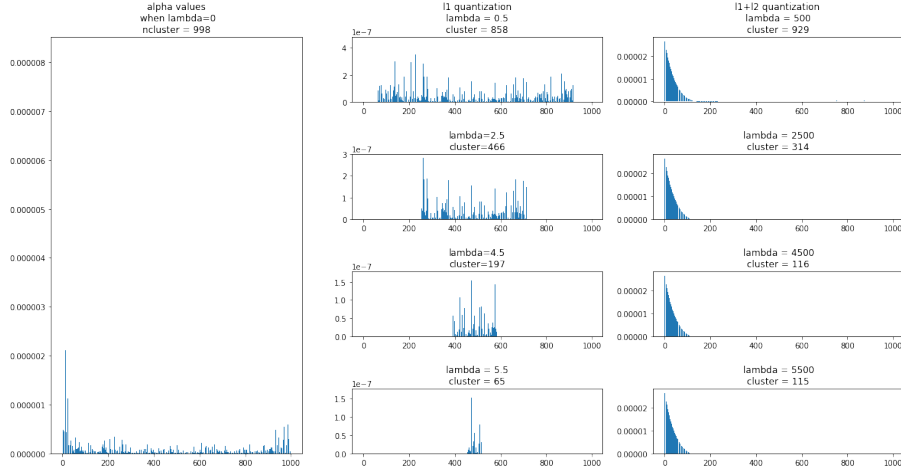
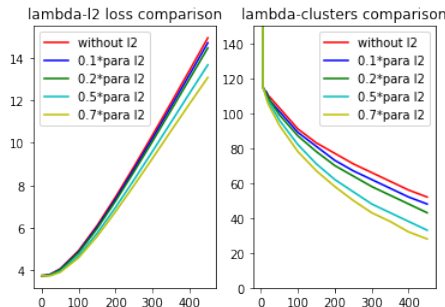
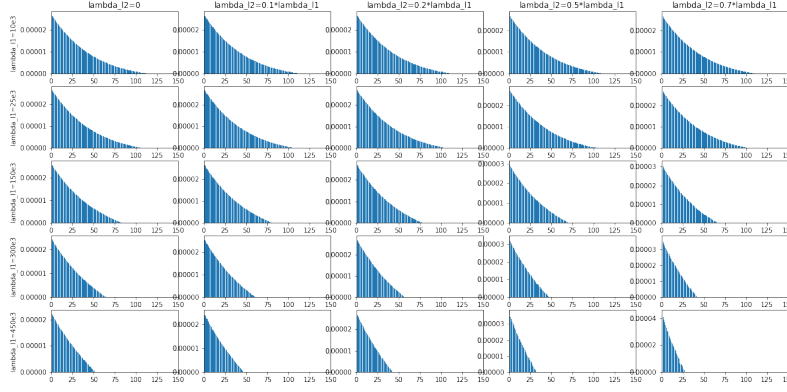
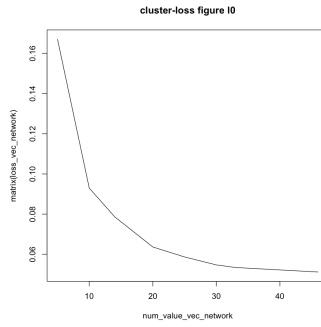


Figure 3: the distribution of weights for neural network under two kind of sparse least square optimization

at the beginning, we add a rescaling factor $c = 1000$ to compute $\lambda_{l_1-l_2} = c \times \lambda_{l_1}$. And we set λ_2 in equation 6 as $\lambda_2 = 0.2 * \lambda_{l_1-l_2}$. Future insight of the characteristics of the result could be observed from figure 3. From this figure we could find out one possible reason for l_1 method to enduring larger l_2 loss could be that it tends to concentrate the α values to the central, and thus will lose some valuable information. On the contrary, l_1+l_2 results demonstrate a strong trend to focusing the α values on the beginning places. While this is not a preferable characteristic, the phenomenon could be attributed to the properties of the optimizer since it also happen when $\lambda = 0$.

To show the rationality of the using l_1+l_2 over l_1 solely, we performed the same quantization with different setting of λ_2 values. In figure 4. From this figure we could find out that for the same λ_1 values, a larger λ_2 value could induce less cluster numbers simultaneously with less l_2 loss. And if we inspect the figure 5 of the illustration of α values, we could find out that with larger λ_2

Figure 4: quantization number of clusters and loss with different λ_2 with respect to same λ_1 values

Figure 5: α values under different settings of λ_2 parameterFigure 6: loss with respect to clusters for l_0 least square optimization

values, the number of non-zeros entries will decrease while the magnitude of single α value would increment. This property reflects our purpose of our designing in equation 6.

And finally, the cluster-loss curve of l_0 -based optimization could be shown as figure 6. While the figure shows a nice property in terms of l_2 loss, it is noticeable that the method is not universal (not every amount of values is retrievable) and it could not support number of clusters (distinct values) more than 50 in this case.

4.2 MNIST Image Data

Vector quantization could be used in image processing to quantize the values of the image and reduce storage cost. Here we choose to use MNIST digits as example to show the performance of image quantization of our proposed methods. Again to compare the performance we at the same time used ordinary K-means clustering method. The result could be shown as figure 7. From the figure we could find out that K-means and the clustering-based least square optimization could provide best performances in general, and there is no significant differences in execution time. $l_1 + l_2$ could provide less norm difference loss than using l_1 solely, but the quality of image might subject to individual's opinion. In terms of running time, here l_1 optimization approach could provide significant time-cost advantages over other methods. Another remark of the MNIST quantization is that the K-means methods sometime provide out-of-range values (not in the interval $[0, 1]$) when the number of clusters are large. However, for the least-square optimization methods, this problem does not happen at least in our circumstance.

The optimization result of l_0 method is shown separately in figure 8. From the information among the images could see this method could not exactly reach every of the number of amounts we require. However, the quality of images could be generally high, although there is a strange behavior that the loss is not guaranteed to be smaller when the number of clusters increases.

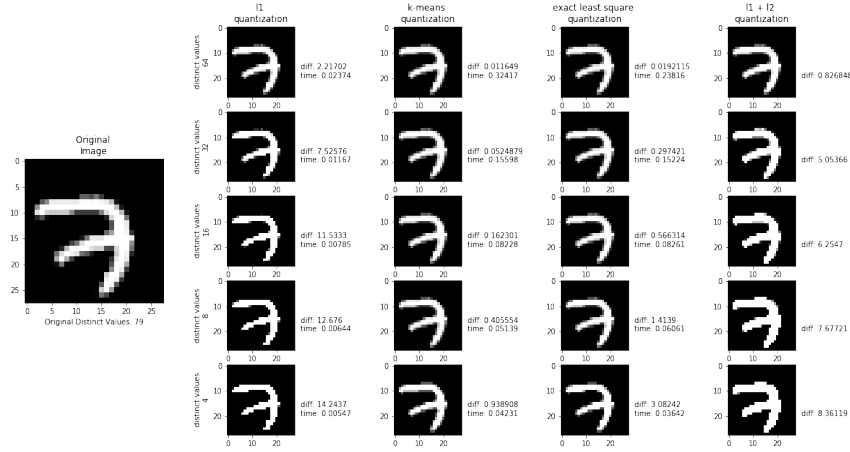


Figure 7: MNIST quantization comparison for l_1 , K-means, clustering-based least square and l_1+l_2 methods

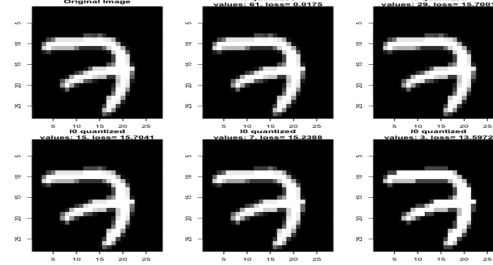


Figure 8: MNIST quantization results for l_0

4.3 Artificially-Generated Data

To test the performance of our algorithm on data with different distributions, we generated 500 hundred data in Mixture of Gaussian, Uniform and Single Gaussian respectively. The distribution of the data we used could be shown as figure 9. According to the experimental results of figure 3, we could naturally think the l_1 based algorithms would perform better on Gaussian and Mixture of Gaussian data. And indeed, our experimental result of figure 10 could verify our assumption and indicate that the l_1 based algorithms would be better to be used when the distribution of data is Gaussian-like. Meanwhile, for the setting of 500 artificial data, l_1 sparse least square quantization method appears significant time-saving property. And for the Mixture of Gaussian- and Gaussian-distributed data, the absolute value of loss is still acceptable if the number of distinct values are large, and it thus could be used in the corresponding circumstances.

The behaviors between different λ_2 values on artificial data are also examined in our experiments. The result could be shown in figure 11, and again, we could obtain lower difference loss with smaller amount of clusters/distinct values, which would be a favorable characteristic in quantization.

As for the l_0 quantization method, in our experiments l_0 optimization of the artificial data could

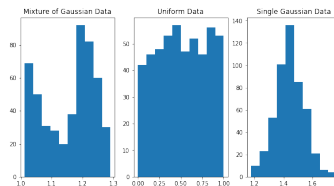


Figure 9: The distribution of the artificially-generated data

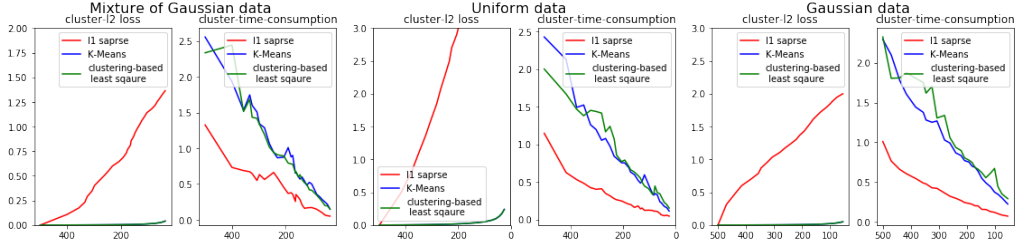


Figure 10: The comparison between l_1 optimization method, K-means quantization and clustering-based least square

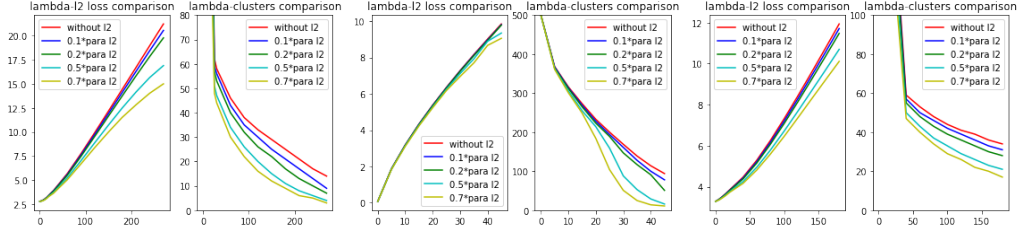


Figure 11: The comparison between l_1 and l_1+l_2 quantization. **left to right: Loss of MoG, Num of clusters of MoG, Loss of Uniform, Num of clusters of Uniform, Loss of Gaussian, Num of clusters of Gaussian**

not provide meaningful results, indicating that this inexact method has the risk of failure if the optimization is difficult for it to solve, despite its better performance than that of l_1 -based method.

5 Conclusion

This paper discussed vector quantization problem and proposed several least square optimization-based algorithms to better accomplish the task. The characteristics and computational properties of the proposed algorithms are examined, and the advantages and drawbacks of them are analyzed. The algorithms are implemented with different optimization packages and tested with Neural Network weights, MNIST image and artificially-generated data respectively, and the results are demonstrated and analyzed.

The paper made following major contributions: Firstly, it proposed several novel quantization methods with different levels of precision and time-consumption, which could provide more options for quantization tasks under different situations; Secondly, the paper innovated the pioneering work of using least square optimization to solve, and it could bring huge research potentials in the area; And thirdly, the paper demonstrated the experimental results of the implementations of the algorithms proposed in the paper, and tested them on different kinds of data with various requirements and goals.

In the future, the authors intend to continue to explore quantization algorithms with the idea demonstrated in the paper. We intend to research on finding better optimization methods, revisions of target functions and extend the quantization method into high-dimensional situations.

6 Acknowledgement

The authors would like to thank Mr. Feng Chen and Mr. Shixiong Wang from Northwestern Polytechnical University, China. Mr. Chen and Mr. Wang provided the authors many useful comments for the algorithms and programs of the research.

References

- [AK98] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237 – 260, 1998.
- [CORG93] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray. Using vector quantization for image processing. *Proceedings of the IEEE*, 81(9):1326–1341, Sep 1993.
- [CR07] E. Candès and J. Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23:969–985, June 2007.
- [EFM10] M. Elad, M. A. T. Figueiredo, and Y. Ma. On the role of sparse and redundant representations in image processing. *Proceedings of the IEEE*, 98(6):972–982, June 2010.
- [FEG01] Köksal F., Alpaydyn E., and Dündar G. Weight quantization for multi-layer perceptrons using soft weight sharing. *Artificial Neural Networks — ICANN*, 2130, 2001.
- [GLYB14] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014.
- [GSBF17] S. Gazagnes, E. Soubies, and L. Blanc-Féraud. High density molecule localization for super-resolution microscopy using cel0 based sparse approximation. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 28–31, April 2017.
- [HHSZ14] Barbara Hammer, Daniela Hofmann, Frank-Michael Schleif, and Xibin Zhu. Learning vector quantization for (dis-)similarities. *Neurocomputing*, 131:43 – 51, 2014.
- [HM18] Hussein Hazimeh and Rahul Mazumder. L0learn: Fast algorithms for l0-regularized learning. <https://github.com/hazimehh/L0Learn>, 2018.
- [HMD15] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.
- [JDS11] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, Jan 2011.
- [LTA15] Darryl Dexu Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. *CoRR*, abs/1511.06393, 2015.
- [MBS93] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. ‘neural-gas’ network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, Jul 1993.
- [NH92] S.J. Nowlan and G.E. Hinton. simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493, 1992.
- [NHCD10] Feiping Nie, Heng Huang, Xiao Cai, and Chris H. Ding. Efficient and robust feature selection via joint l2,l1-norms minimization. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1813–1821. Curran Associates, Inc., 2010.
- [PVG⁺11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

- [SCYE17] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *CoRR*, abs/1703.09039, 2017.
- [SFR07] Mark Schmidt, Glenn Fung, and Rómer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, pages 286–297, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [SRZ⁺16] M. Saleem, Z. U. Rehman, U. Zahoor, A. Mazhar, and M. R. Anjum. Self learning speech recognition model using vector quantization. In *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, pages 199–203, Aug 2016.
- [TG07] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, Dec 2007.
- [Tib11] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011.
- [UMW17] K. Ullrich, E. Meeds, and M. Welling. Soft Weight-Sharing for Neural Network Compression. *ArXiv e-prints*, February 2017.
- [VH11] Thomas Villmann and Sven Haase. Divergence-based vector quantization. *Neural Computation*, 23(5):1343–1392, 2011. PMID: 21299418.
- [Wan17] Chen Wang. The practical implementation and algorithm exploration of embedded deep learning (unpublished). Master’s thesis, Dept. of Computer Science, University College London, London, United Kingdom, Sep 2017.
- [WMM⁺10] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, June 2010.
- [Wri15] S. J. Wright. Coordinate Descent Algorithms. *ArXiv e-prints*, February 2015.
- [YZ11] Junfeng Yang and Yin Zhang. Alternating direction algorithms for l1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.