The Appropriateness of $k$-Sparse Autoencoders in Sparse Coding

Pushkar Bhatkoti
School of Computing and Mathematics, Charles Sturt University, Australia
p@bhatkoti.com

Manoranjan Paul
School of Computing and Mathematics, Charles Sturt University, Australia
mpaul@csu.edu.au

ABSTRACT - Learning of representations usually happens in different ways. Sometimes it persuades sparsity thus enhances performance through the task categorization. The sparse elements entail the learning algorithms that relate to the sparse-coding. Sometimes the algorithms have neural training networks with sparsity penalties and fines. The $k$-sparse autoencoder (KSA) model appears linear. The appropriateness of the model in sparse coding forms the foundation of this paper. Most important, the model appears speedily encoded and easily trained. Given these advantages, the model is suited for solving large-size issues or problems. We used openly available Mixed National Institute of Standard and Technology Database (MNIST) and NYU Object Recognition Benchmark (NORB) dataset in supervisory and unsupervised learning tasks to validate the hypothesis. The result of the paper shows that the traditional algorithms cannot resolve large size problems for sparse coding as the $k$-Sparse autoencoder model.

Keywords—$k$-sparse autoencoder (KSA), Sparsity, algorithms, Sparse-coding

(1) INTRODUCTION

If the learning of representations happens in a way that persuades sparsity, enhanced performance is attained on categorization tasks [68]. The methods entail blends of sampling phases, activation functions, and various penalties. The learning algorithms (LAs) in sparse elements may be approaches related to sparse-coding as explained by Olshausen and Field [54]. In some cases, the algorithms are neural training networks, which have sparsity form of penalties as demonstrated by Nair and Hinton [73]. The methods consist of two phases. The first phase comprises of LAs, which generate a structured dictionary, D. D sparsely stands for the following data.

$$D = \sum_{i=1}^{N} \left( x_i \right)$$

The second phase comprises of encoding algorithms (EAs). Based on D, the EAs characterize mappings from given input form of vectors, $x$, to the corresponding feature factors. Sparsity’s effectiveness may be determined by itself using KSA. As a model, the KSA is linear. The only hidden layers holding activities are defined as being k-highest ($k$). KSAs are easily trained, and speedily encoded. Thus, they are suitable for large-size problems. Such problems are not easily resolved using the traditional algorithms for sparse coding. The traditional encoders map $x$ to hidden representations ($Z$) by applying the following function.

$$Z = f (P x + b)$$

Where $\{P, b\}$ parameterizes the function.

KSA is essentially a linear encoder with weights and activation functions [68]. To validate the appropriateness of KSA, we trained Deep Neural Network (DNN) with KSA classifier to get the optimal results on the MNIST and NORB dataset.

1.1 Statement of Problem

The dictionary learning and sparse coding steps have appeared costly affairs in terms of computation. The computation problem is challenging practically during the coding. A wide-ranging search for research studies does not result into structured studies that examine how KSAs perform in discriminative unsupervised learning, and deep as well as shallow learning tasks. There is a need to determine the performance of KSAs, as methods for sparse encoding to attain precise sparsity within hidden representations.

1.2 Purpose of the study

- Examine the performance of the KSA model in shallow, unsupervised, and deep learning.
- Examine the performance of the model in achieving clear-cut sparsity within the concealed representation.
- Demonstrate how KSAs model can be applied and learned in sparse coding.

1.3 Research question

- At what level does the KSAs model achieve accurate sparsity within concealed representations?

(2) LITERATURE REVIEW

2.1 How does the model perform in shallow, deep, and unsupervised learning tasks?

2.1.1 Performance of the model in unsupervised learning
An autoencoder (AE) play a significant role in unsupervised learning and architectural work [2]. The input size is significant in determining the performance of the model in changing the inputs into appropriate outputs with the insignificant errors [9]. Some researchers showed the performance of the KSAs model in unsupervised learning [48]. The researchers located the activities in the model’s concealed layer. Makhzani and Frey used a MNIST digit handset with over testable 10,000 images and 60,000 images for training [9]. NORB set forms the basis of the research.

2.1.2 How the model performs in a deep learning context
In building DNN the model is exploitable [68]. It is evident, in the pre-training program through a layer by layer demonstration. The pre-training program seems exploitative and discriminative as it can be determined through deep learning [9]. For example, the studies indicate how the shallow KSA becomes trained, then hide the codes. Since the results are fixed and hidden, a new model is trained to produce more hidden code sets. The parameters of the model are critical in initiating the DNN with the hidden layers. In the process of fine-tuning DNNs, the parameter layers are fixed. The underneath layer has a trained soft-max categorizer. The softmax is a mathematical function to calculate the probability distribution in the last layer (output). It is important to fix all the parameters of both the first layer and second layer. The trained softmax classifier needs to be suspended on the second layer. The first layer’s weights need to remain constant while the second layer gets trained jointly using softmax [48]. The previous softmax initialization would be critical in fine-tuning both the layers. The layer-wise-fine-tuning has earned the attention of many stakeholders, and importantly improves the classification performance [48].

In the process, the NORB case, the values of sparsity level (k) and hidden units (c̅) are 150 and 2 while under the MNIST case; the values are 25 and 3 respectively.

2.1.3 How the model performs in shallow learning
Many of the current learning algorithms architectures are characterized as shallow. The internal and external representations learned via such algorithms are essentially plain. They cannot extract various forms of intricate structures found in the input with higher dimensions. They perform poorly in learning such representations as they poorly re-use and combine the related intermediary concepts. Consequently, the models that are hinged on the algorithms are ineffective in generalizing diverse tasks [58]. Makhzani and Frey [48] use back propagation in fine-tuning encoder-weights. They use a regularization approach during the fine-tuning of various algorithms. The approach is similar to the one employed in training the related UL tasks. During the fine-tuning of the KSA utilize the alpha k-largest concealed units within the matching Discriminative-Neural Networks.

The encoder weights are evidently used in the supervised learning. Most of the weights used are learnt through unsupervised learning methods. Multilayer discriminative model is traceable to unsupervised learning methods. Subsequently, the significance of back-propagation becomes evident in adjusting the hidden weights [67]. The discriminative model fine-tunes the hidden weights. The dropout regularization helps in fine-tuning the weights from the dropout encoder. In fact, the addition of noise into the inputs further fine-tunes the discriminative neutral net [73]. Similarly, the KSA gets fine-tuned by using the largest hidden units (k) and corresponding to the discriminative neutral networks.

The poor performance under these tasks can be caused by intermediary concepts, thus making the models ineffective in shallow learning tasks [4, 5].

2.1.4 The performance of the model in achieving sparsity in a concealed representation
The model representations are critical in attaining the best quality categorization outcomes without implementing sparsity in a concealed context [48]. The nonlinearities and regularizations are irrelevant in the process. KSA enhances the learning of these representations supportive of sparsity [61]. Some researchers show algorithms used in sparse coding support complex matrix computations [67]. The sparse coding based KSAs model is effective under the simple matrix multiplications therefore good for to categorizing tasks during learning phase. In a case where the implementation seems parallel and distributed, the operations in the model become substituted using recursive applications until the k-value remains constant [61].

In a low-level sparsity degree schedule, the model uses the algorithm characteristics acquired in the hidden elements [9]. In assigning the units, the k-means becomes comparable to the units [9]. The units appear picked and fixed in the subsequent epochs. However, many units remain constant to avoid the excessive sparsity impeding the inactive unit adjustments in the back-propagation slope. Studies demonstrate how the rescheduling sparsity degree in different epochs is critical in addressing the challenges [9].

2.2 Gaps in the Literature
The approaches used by the previous researchers can rarely control sparsity in their respective samples.

Makhzani and Frey [68] put commendable efforts in evaluating the performance of KSAs in unsupervised learning and discriminative deep and shallow learning tasks. In addition, they out commendable efforts in assessing the performance of KSAs in attaining accurate sparsity within concealed representations. Even then, their approach is unsound as they fail to control sparsity in their samples, giving rise to too many inactive or dead units. Their results are not extrapolated to establish whether they remain unchanged when fewer code-dimensionality values are used. For instance, they do not give an indication of the results expected when 1,000, 5,000, 10,000 units or more are on the MNIST. The findings presented by Makhzani and Frey [68] with respect to
the performance of KSAs have not been confirmed or refuted in succeeding research studies. Coates, Lee and Ng [21] attain higher performances with the different algorithms than those priory published with respect to NORB and CIFAR-10 datasets. A wide-ranging search of the currently available literature does not result into any study that has explored the reasons why the performances vary or the veracity of the findings presented by Coates, Lee and Ng [21].

(3) METHODOLOGY

3.1 Selection of datasets
The MNIST dataset of handwritten digits is used in the research study. The dataset contains 50,000 images for training and 5,000 images for test purposes. The dataset of images for training purposes is arbitrarily divided into 40,000 cases for training and 10,000 validation cases. Another NORB dataset is uniform and normalized is also used. The NORB dataset is widely described by Lecun, Huang and Bottou [97]. It has 20,000 examples for training purposes and 20,000 examples for test purposes. It also has 40 images of toys from four generic classes: cars, aircrafts, trains, and two-legged mammals. Each of the 40 images are dual channels. The external size of each of the channels is 96*96 square pixels. The related internal size is 64*64 square pixels. The inner sizes of all the channels are resized to 32*32 square pixels, via the employment of bi-cubic interpolation. From the 32*32 square pixels, vectors are formed. Each of the vectors has an input of 2048 measures, or dimensions.

The mean of the resulting dataset is subtracted from each data point. The result is multiplied by the resulting standard deviation’s reciprocal along all the input dimensions in the entire dataset. That ensures that the attendant contrast is normalized. The 20,000 examples for training purposes are divided into 15,000 cases for training and 5,000 validation cases. The overall method is tested on natural patches of images, which are drawn from a CIFAR-10 form of dataset. A million 8*8 size patches are arbitrarily extracted from 40,000 32*32 CIFAR-10 images. All the patches are thereafter locally normalized for contrast. As well, they are whitened with respect to ZCA [21].

3.2 KSA Training
3.2.1 Sparsity Level Scheduling
The enforcement of low KSA sparsity levels leads to the greedy assignment of individually concealed units to training set cases by the employed algorithm. The assignment happens in the few initial epochs. It occurs in a way that is comparable to the clustering of k-means. In the succeeding epochs, the units are picked as well as reinforced. Other units, which are also hidden, are not adjusted. Overall, the resulting excess sparsity impedes the back propagation of a slope, or gradient, from correcting the other units’ weights [44],[26].

In this research study, that assignment is addressed by planning KSA sparsity levels over the related epochs in an effective way. The study aims at a KSA sparsity of 20 (k=20). A k=200 is started with. At the higher level, the KSA trains each of the concealed units. The level is linearly depressed from k=200 to k=20 over initial 50% of all the epochs, initializing the KSA in a favourable regime, or range. In the regime, the probabilities of picking any of the units are equal. The k=20 level is maintained for the rest of the epochs. Each of the related filters is trained, across the whole KSA sparsity level range (k=200 to k=20).

3.2.2 Hyper-Parameters’ Training
Stochastic slope descent, which has momentum, is used in optimizing the parameters of the adopted model.

$$M \cdot v_c - f(x_c) \cdot \eta_c = v_{c+1}$$
$$x_c + v_{c} = x_{c+1}$$

Where

- \(M\) is momentum
- \(v_c\) is velocity,
- \(\eta_c\) is learning rate
- Standard deviation is \(\sigma\)

In the above optimization equations \(v_c\) is a vector for velocity, \(\eta_c\) represents the rate of learning, and mc represents momentum, all at the iteration at the \(k\)-value. Gaussian distribution is utilized in the weights’ initialization. The distribution’s standard deviation is \(\sigma\). Diverse values for momentum, initializations, and rates of learning are used. Each of them is hinged on a particular training dataset and task. Hyper-parameters are chosen using validation. The value of each of them is determined at each all NORB tasks and MNIST tasks across all the epochs, regardless of whether they are unsupervised or supervised. The reduction of the rates over the epochs is done linearly.

3.2.3 Implementation of Experiment
Notably, the majority of the algorithms for sparse coding necessitate the utilization of intricate matrix computations like matrix decomposition and inversion [24], [30]. On the other hand, KSAs only need simple multiplication of matrixes and sorting-computations in their sparse encoding phases and in the learning of dictionaries. Clearly, any operation that is used in place of the computations should repeatedly apply to the related thresholds until a constant \(k\)-value is obtained [22].

In this research study, a GPU-implementation is used. The implementation is efficient. It is got using the Gnnumpy library. [69] fully describes the library. The library’s use is based on a lone Nvidia-GTX 680-GPU. The implementation of each of the algorithms is in python. As a language, python is often used in high level and general-purpose programming. It is appropriate for the research study as its design stresses on the readability of codes, and its concise syntax allows for the expression of diverse concepts in a few code lines. It provides clear constructs, which clear give programs on diverse scales. As well, it supports diverse paradigms of programming. These
include procedural, functional, object-oriented and imperative programming. It has a comprehensive and large standard library, a system that is markedly dynamic, and a management of memory that is automated [59] [45].

Table 1 Sample of python script

```python
def __init__(self, visible_size, hidden_size, rho, lamda, beta):
    self.visible_size = visible_size # number of input units
    self.hidden_size = hidden_size # number of hidden units
    self.rho = rho # desired average activation of hidden units
    self.lamda = lamda # weight decay parameter
    self.beta = beta # weight of sparsity penalty term

    # weight of sparsity penalty term
    self.limit0 = 0
    self.limit1 = hidden_size * visible_size
    self.limit2 = 2 * hidden_size * visible_size
    self.limit3 = 2 * hidden_size * visible_size + hidden_size
    self.limit4 = 2 * hidden_size * visible_size + hidden_size + visible_size

    self.__init__ Neural Network weights randomly
    W1, W2 values are chosen in the range [-r, r]
```

GitHub [98], table 1)

3.3 Data Collection
Data were collected for different learnings. The details are given in each of below section.

3.3.1 Data Relating to Unsupervised Learning of Features
Comparison was a key feature of this level of research. The quality of the features of the data sets learnt through the help of the KSA algorithm were compared with those learnt through the use of other means of learning. Features were fixed after their extraction. Each of the researchers was trained on how to handle the collection of the research results. A cross-examination was performed to assess the quality of each of the features. That was done by assessing the rate of errors in the classifier’s work. The reason the errors were used as an indicator of quality is that there is a higher correlation between quality and the resulting errors. The relationship was inverse in nature [76]. A 60% level of dropout regularization was used to assess the quality of Dropout Autoencoders (DOA).

For the Denoising Autoencoders (DA), the quality was assessed through the use of the input pixel (IP) that dropped at a rate of 20%. Non-corrupted IPs was employed in establishing classification features. The k-largest in every KSA was established while every other unit was reduced to zero. This was achieved by sorting each activity in the expected sequence. A linear function of each of the concealed units was made. Non-linearity was achievable only from the selection. The selection was preferred because of its regulative ability [39]. There was a need for regulation since any inclusion of unnecessary sets would have led to misleading results. The study was based on the need to determine the correlations among the many features of the data sets accurately.

3.3.2 Data Related to Supervised Learning of Features
Supervised learning is a highly selective and discriminative process. During this process, the encoders’ weights, which were learnt through the initial phase of unsupervised learning, were used to initialize the first Multilayer Discriminative Model (MDM) layers [31]. The remaining concealed layers were treated via back propagation. Despite that difference, the two layers were used to fine-tune the weights of the preceding layers.

During the research process, the quality of the features that were learnt using the KSA algorithm were compared with those that were learnt through other methods, which had been supervised already. KSA was not the only method used to supervise the quality of the process. The other methods used for this purpose were Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN) and Deep Belief Machine (DBM) [63].

The research process was going to face uncertainty and inaccuracy challenges. Back propagation was introduced to harmonize and orient the weights. Given that there were diverse algorithms being looked at, a regularization approach had to be used to address the problem fully [95]. For instance, weights derived from DAO were fine-tuned by having their drop-out rate regularized. For the DA algorithm, DNN was regularized by bringing in the noise and introducing it to the appropriate input. Finally, for the KSA algorithm, the alpha k-largest units, which were concealed and closely related to the DNN, were used.
3.3.3 Data Related to Deep Learning of Features
KSAs can also be used as the basic building units of DNN [12]. That assertion was fully put into consideration. A shallow KSA was first fine-tuned to obtain the intrinsic concealed codes. Its features were consequently fixed. A repeat of the fine-tuning was done on a second KSA with the aim of obtaining a similar set of concealed codes [65]. The parameters of the resulting KSAs were used to initialize a DNN with both layers.

The fine-tuning exercise was targeted at the DNN features. The process began with the fixing of the parameters in the first two layers. A softmax multiplier was applied on the latter layer of the parameters. The two layers were trained using softmax software and initialization. The two unit layers were jointly initialized at the preceding stage of initialization. The initialization process was carried out in phases owing to the fact that such sequential initialization results in the improvement of the process as opposed to simultaneous initialization [45]. Despite the lengthy procedure of the sequential initialization, care had been taken to ensure that the $k$-largest concealed codes were not altered. In the MINST data set, $k=100$ and $\alpha=3$ in both the unit layers. The second dataset, NORB had its values being $\alpha=3$ and, $k=100$ in the two layers.

3.4 Data Validity and Reliability
Quantitative and qualitative measures help in measuring the validity of any assumption. The collected variables became modified, thus changing them into numerical figures. With a mathematical formula, the predictability of data was possible [73]. The sequential experiments made the data reliable. As a result, the normalization of errors was accomplished through the series of experiments.

(4) RESULTS

4.1 Sparsity Level Effect
The number of concealed datasets on which each of the KSA filters visualized was 900 MNIST units at different $k$-values. Learning from these units was from arbitrary image patches. A localized Gabor form of filters on which the KSAs learnt was determined.

<table>
<thead>
<tr>
<th>Methods</th>
<th>$k$-value</th>
<th>Optimum rate of error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>1.80</td>
<td></td>
</tr>
<tr>
<td>DOA</td>
<td>1.95</td>
<td></td>
</tr>
<tr>
<td>RBM</td>
<td>1.81</td>
<td></td>
</tr>
<tr>
<td>KSA</td>
<td>200</td>
<td>1.32</td>
</tr>
<tr>
<td>KSA</td>
<td>150</td>
<td>1.45</td>
</tr>
<tr>
<td>KSA</td>
<td>80</td>
<td>1.37</td>
</tr>
<tr>
<td>KSA</td>
<td>20</td>
<td>1.32</td>
</tr>
</tbody>
</table>

Figure 1: Filters of the KSA for diverse sparsity $k$-levels, learnt from MNIST with 900 hidden units. In this example $K$ value of 40 is the better than $K$ value of 60.

The figure above shows the model filter visualization using the concealed units of 900. It also illustrates the diverse sparsity levels using the MNIST approach.
The figure 2 above indicates the use of NORB in filtering the hidden units of 900 images. It also indicates the diverse sparsity levels.

<table>
<thead>
<tr>
<th>Methods</th>
<th>k-value</th>
<th>Optimum rate of error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>10.2</td>
<td></td>
</tr>
<tr>
<td>DOA</td>
<td>10.1</td>
<td></td>
</tr>
<tr>
<td>RBM</td>
<td>8.6</td>
<td></td>
</tr>
<tr>
<td>KSA</td>
<td>200</td>
<td>9.1</td>
</tr>
<tr>
<td>KSA</td>
<td>150</td>
<td>8.3</td>
</tr>
<tr>
<td>KSA</td>
<td>80</td>
<td>8.6</td>
</tr>
<tr>
<td>KSA</td>
<td>20</td>
<td>8.7</td>
</tr>
</tbody>
</table>

Table 2 – Unsupervised learning methods with 3500 NORB images

From Table 1 and Table 2, it is clear that KSA performed better than DA, DOA, and RBM. The best KSA result was obtained by \( k=200 \) and \( \alpha = 3 \) with 900 MNIST units and \( k=150 \) and \( \alpha =2 \) with 3500 NORB units.

(5) CONCLUSION

Sparse coding is one of the latest technologies in software development. The use of KSAs in this endeavor is invaluable. KSAs are the latest development in the market of encoders. In unsupervised learning of features, KSA performs better than DA, DOA, and RBM. In supervised learning of features, deep KSA performs better than DA, DOA, and RBM. In deep supervised learning of features, KSA performs better than DA, DOA, and RBM.

(6) Future Work

Future research should confirm or refute the findings presented by Makhzani and Frey (2013) with respect to the performance of KSAs in unsupervised, discriminatively deep as well as shallow learning tasks. In addition, future research should confirm or refute the findings presented by Makhzani and Frey [48] with respect to the performance of KSAs in attaining precise sparsity within hidden representations. In addition, further research to establish why the performance of KSAs and related algorithms may differ in different settings as shown by Coates, Lee and Ng [20].

(6) REFERENCES

[94] Neural Computation, 23(7), 1661-74.