An adaptive dimension reduction algorithm for latent variables of variational autoencoder

Yiran Dong and Chuanhou Gao, Senior Member, IEEE

Abstract—This paper proposes a new pattern of inactivating variables to address the dimension reduction issue of latent variables in VAE related models. By introducing $\pi$ vector whose elements are 0 or 1, and further utilizing the element-wise multiplication between $\pi$ and variables, we develop an adaptive dimension reduction algorithm to remove those latent variables inactivated by 0 component of $\pi$ for the purpose of dimension reduction. We make some theoretical analysis about the convergence and local convexity of the algorithm output, i.e., the convergent dimension. These encouraging results are also supported by numerical experiments. Six frequently-used data sets, including two large-sized ones, are illustrated that the proposed algorithm is quite effective both in precision and in efficiency. A potential application of the algorithm is to generate VAE related models with more compact structure and higher precision, except for reducing dimension.

Index Terms—Variational autoencoder, latent variable, dimension reduction, Lagrange loss, convergence

1 Introduction

Variational autoencoder (VAE) has been one of the most popular modeling methods since it was pioneered by Kingma and Welling [1]. Essentially, it is a kind of generative modeling approach, like Generative Adversarial Nets (GAN) [2], that handles models of distribution about datapoints. VAE is built on neural networks and trained with stochastic gradient descent. From the viewpoint of structure, there include encoder, latent variables layer and decoder in a VAE model, and for encoder and decoder there may be multi-layers of hidden layer therein. Fig. 1 exhibits a schematic of VAE. Throughout the working process, the encoder encodes the original samples into latent variables while the decoder reconstruct new samples from latent variables. The whole workflow is implemented through unsupervised learning of neural networks. A more detailed introduction on VAE may be found in the tutorial paper [3]. Benefiting from complete mathematical support, VAE has exhibited strong power in generating many complex data, such as handwritten digits [1], [4], faces [1], [5] and CIFAR images [6] etc.

Following the rapid development of VAE, some improved versions consecutively emerge. A typical one is conditional variational autoencoder (CVAE) [7], [8], which, instead of unsupervised learning in VAE, models complicated distributions about datapoints through supervised learning. Liu and Wang [9] came up with Stein VAE, which uses stein variational gradient descent to maximize the evidence lower bound (ELBO) and update network parameters. To balance the independent constraint of decoder and the information capacity in encoder, Higgins et. al. [10] developed Beta VAE that introduce a hyperparameter $\beta$ to multiply the $KL$ divergence in ELBO. The Structure VAE [11] was proposed with the purpose of using more complicated graph to approach the actual posterior distribution. These improvements further witness the success of VAE.

However, when VAE works, it usually involves many neural nodes. This can be seen from Fig. 1 where either the hidden layers in encoder and decoder or the latent variables layer contain a great deal of neural nodes. Since the number of neural nodes directly determines the number of parameters in the model, too many neural nodes may lead to the extreme complexity of model and serious time-consuming training. It thus needs to remove those redundant neural nodes that may contain useless information even noise to model.

Dimension reduction is a feasible solution to the above issue. Naturally, it is not a novel project in the field of machine learning, and some classical methods, such as principal component analysis [12] etc., have been widely used in many fields. Here, we continue to select this strategy to address the curse of dimensionality in the VAE related models. However, unlike those traditional dimension reduction techniques where various projections usually need to be defined between original features and the reduced ones, the current proposed method (called adaptive dimension reduction algorithm in the context), getting inspiration from sequential minimal optimization [13], works through judging whether each feature contains too much noise. We put forward a new pattern to realize the above process. By introducing $\pi$ vector whose elements are 0 or 1, and further utilizing the element-wise multiplication between $\pi$ and latent variables, the algorithm will remove those latent variables inactivated by 0 component of $\pi$ while keep
those activated by 1 component of $\pi$. The algorithm is also proved to have convergence and local convexity about the output. These properties are further verified by a great deal of numerical experiments. The potential effect of the algorithm is, on the one hand, to obtain higher precision and more efficient VAE related models, one the other hand, to generate a complete new high-efficiency model.

The rest of the paper is organized as follows. Section 2 presents some preliminaries about VAE and CVAE. In Section 3, our main work, the adaptive dimension reduction algorithm as well as its convergence and application, is presented. This is followed by some experimental studies of the proposed algorithms in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries
In this section we will give a short introduction on VAE and CVAE [3], and then present the assumption that is used for algorithms development.

2.1 VAE and CVAE

VAE is a kind of typical latent variable model, whose main purpose is to generate sample points sharing the same distribution with the original samples as far as possible through utilizing latent variables. Let $D = \{x \in \mathbb{R}^N\}$ represent a data set, and the task is to infer the distribution of $x$, $P(x)$, so as to generate new sample points. VAE addresses this issue by introducing latent variables $z \in \mathbb{R}^m$ obeying the standardized normal distribution $N(0, I)$, and further minimizing the $\mathcal{KL}$ divergence between the true posterior distribution $P(z|x)$ and the learned one $Q(z|x)$ on $D$. The graphical model of VAE is exhibited in Fig. 2. Here, the $\mathcal{KL}$ divergence is essentially about the expectation calculation, defined by

$$
\mathcal{KL}[Q(z|x) \parallel P(z|x)] = E_{Q(z|x)}[\log Q(z|x) - \log P(z|x)].
$$

By adding $P(x)$ and applying Bayes rule to $P(z|x)$, the above expression can be rewritten as

$$
\log P(x) - \mathcal{KL}[Q(z|x) \parallel P(z|x)] = E_Q[\log P(x|z)] - \mathcal{KL}[Q(z|x) \parallel P(z)],
$$

which is the core formula in VAE. The left hand side of the formula is just right the maximization objective with the first term to represent the likelihood of $P(x)$ while the second term to express minus error of estimating $P(z|x)$ by $Q(z|x)$. The right hand side takes a form of autoencoder, where $Q$ is “encoding” $x$ into $z$, and $P$ is “decoding” it to reconstruct $x$. It can be also seen as a lower bound of $\log P(x)$ due to the nonnegativity of $\mathcal{KL}[Q(z|x) \parallel P(z|x)]$, usually called ELBO. The inner two terms are both computable, so the whole often acts as the cost function one can optimize via stochastic gradient descent.

To optimize Eq. (2), it needs to know a specific form about $Q(z|x)$. The usual choice for it, and also for $P(x|z)$ is the Gaussian distribution, i.e.,

$$
Q(z|x) = N(\mu(x), \Sigma(x)) \text{ and } P(x|z) = N(f(z), \sigma^2),
$$

where $\mu$, $\Sigma$ and $f$ are arbitrary deterministic functions that can be learned via neural networks, $\Sigma$ constrained to be a diagonal matrix, and $\sigma^2$ the constant covariance. With these distributions, the “reparameterization trick” is further adopted, then backpropagation can compute a gradient used for stochastic gradient descent, which completes the learning process.

2.2 Assumption

In order to facilitate the design of dimension reduction algorithms, the following assumption is made for VAE and CVAE. We denote the training precision loss function by $h_z$, where $z$ is identified as an $m$-dimensional latent variable vector in the related model $M$. Further, we introduce a special $m$-dimensional vector $\pi$, whose elements are ones or zeros, to build a new model with latent variables $z \otimes \pi$ where the symbol $\otimes$ represents the element-wise multiplication. Then the assumption states:

Assumption: $\forall \pi^1, \pi^2$ satisfying $\pi^1_i \leq \pi^2_i$ for any $i = 1, ..., m$ and furthermore $\sum_i \pi^1_i < \sum_i \pi^2_i$, if

$$h_{z|\pi^1} < h_z \text{ and } h_{z|\pi^2} < h_z,$$

then $h_{z|\pi^1} < h_{z|\pi^2}$.
The physical meaning of the above assumption is that if continue two-time dimension reduction of the latent variables can both generate higher-precision models than $\mathcal{M}$, then the second time dimension reduction can obtain higher-precision model than that by the first time dimension reduction.

It should be noted that the assumption is reasonable to a certain extent. Essentially, the purpose of dimension reduction is to remove redundant neural nodes that carry useless information in the model, also thought as noise. If the continue dimension reduction both generate better models than the original one, it is reasonable to think that the dimension reduction truly deletes the redundant neural, so the more the redundant neural nodes are removed, the higher precision the model has. Another point should be noted that the assumption can work not only for VAE and CVAE, but for other autoencoder models.

3 Adaptive Dimension Reduction Algorithm

In this section, we will develop the adaptive dimension reduction algorithm for latent variables in VAE and CVAE, and also give the corresponding theoretical analysis.

3.1 Lagrange Loss

In the process of algorithm design, we expect to find a model with latent variables $\mathbf{z}$ that has the minimum precision loss compared to the original model with latent variables $\mathbf{z}$. Mathematically, it can be modelled as, for enough small $\varepsilon \geq 0$,

$$h_2 - h_x \leq \varepsilon. \tag{4}$$

Utilizing $\pi$ vector, we rebuild a new model to be

$$h_{\pi\varepsilon\mathbf{x}} - h_x \leq \varepsilon. \tag{5}$$

From the expression $\pi \otimes \mathbf{z}$, the latent variable $z_i$ is activated if the corresponding $\pi_i = 1$; otherwise, $z_i$ is inactivated, and the information about $z_i$ cannot pass to the decoder and is seen as noise. Fig. 4 exhibits the generation process of new latent variables utilizing $\pi$. For the new model of (5), we have

**Lemma 1.** If $\sum_{i=1}^m \pi_i = \hat{m}$ in (5), then (4) holds true from (5).

**Proof.** If $\sum_{i=1}^m \pi_i = \hat{m}$, $\pi \otimes \mathbf{z}$ will define a group of new latent variables, which corresponds to a new model including $\hat{m}$ latent variables, but shares the same parameters with $\mathcal{M}$. Note that the model $\tilde{\mathcal{M}}$ having training precision loss function $h_2$ has been trained to possess better training precision in all models with $\hat{m}$ latent variables, so $h_2 \leq h_{\pi\varepsilon\mathbf{x}}$, i.e., the result is true.

The following task for the algorithm development is to minimize $\sum_{i=1}^m \pi_i$ satisfying (5). Based on (2), we take the loss function as the minus of ELBO, i.e.,

$$h_{\pi\varepsilon\mathbf{x}} = -E_Q[\log P(\pi \otimes \mathbf{z})] + KL[Q(\pi \otimes \mathbf{z}) \parallel P(\pi \otimes \mathbf{z})], \tag{6}$$

where

$$Q = Q(\pi \otimes \mathbf{z}) = \prod_i N(z_i; \mu_i(x), \Sigma_i(x))^\pi_i,$$

$$P(\pi \otimes \mathbf{z}) = [\prod_i N(0, 1)^{1 - \pi_i}]. \tag{7}$$

Similarly, we can express $h_2$, by combining which with (6) and further inserting them into (5) we can get the following convex optimization model

$$\min_{\pi} \sum_{i=1}^m \pi_i,$$

$$\int_\mathbf{z} Q(\mathbf{z}) \log \frac{P(\mathbf{z})}{Q(\pi \otimes \mathbf{z})} d\mathbf{z} - KL[Q(1 - \pi) \otimes \mathbf{z}] \parallel P((1 - \pi) \otimes \mathbf{z}) \leq \varepsilon, \tag{8}$$

where $\mathbf{1}$ is a $m$-dimensional vector with every element to be 1. Note that in the above model, the first term of the left side of the inequality constraint is closely related to the distance $\|f(\pi \otimes \mathbf{z}) - f(\mathbf{z})\|$ while the second term measures the $KL$ divergence between the latent variables one discards and the Gaussian noise. The former needs to be small enough so that the expectation values, $f(\pi \otimes \mathbf{z})$ and $f(\mathbf{z})$, can approach the original data $\mathbf{x}$, which means that the discarded units from $\mathbf{z}$ do not contain too much information of data set. However, the latter should be large enough, which needs to discard units from $\mathbf{z}$ as many as possible since the latent variables are i.i.d. We thus introduce a new parameter $\eta$, named gap hyperparameter, to split the constraint in (8) into two inequalities and develop a relatively relaxed optimization model

$$\min_{\pi} \sum_{i=1}^m \pi_i,$$

$$\int_\mathbf{z} Q(\mathbf{z}) \log \frac{P(\mathbf{z})}{Q(\pi \otimes \mathbf{z})} d\mathbf{z} \leq (1 + \eta)\varepsilon,$$

$$KL[Q((1 - \pi) \otimes \mathbf{z}) \parallel P((1 - \pi) \otimes \mathbf{z})] \geq \eta\varepsilon. \tag{9}$$

The solution of (9) will discard the latent variables that contain little information of data set and are also contaminated by noise heavily.

Further by introducing the Lagrange multipliers $\alpha$ and $\beta$, we rewrite (9) to be

$$\min_{\pi} L(Q, P, \pi) = \sum_{i=1}^m \pi_i + \beta \left( \int_\mathbf{z} Q(\mathbf{z}) \log \frac{P(\mathbf{z})}{Q(\pi \otimes \mathbf{z})} d\mathbf{z} - (1 + \eta)\varepsilon \right) - \alpha (KL[Q((1 - \pi) \otimes \mathbf{z}) \parallel P((1 - \pi) \otimes \mathbf{z})] - \eta\varepsilon).$$

We name $L(Q, P, \pi)$ by Lagrange loss in the context. Unlike the traditional pattern, here the Lagrange multipliers $\alpha$ and $\beta$ are set to be hyperparameters that represent a preference of decreasing dimension or keeping information. To make the constraints of (9) be “hard” [13], a feasible selection is to set them as functions of sample $\mathbf{x}$, given by

$$\alpha(x) = \begin{cases} 0, & x \geq 0; \\ -\infty, & x < 0. \end{cases}, \quad \beta(x) = \begin{cases} +\infty, & x > 0; \\ 0, & x \leq 0. \end{cases} \tag{11}$$
3.2 Algorithm design and theoretical analysis

The main purpose of designing adaptive dimension reduction algorithm is to minimize the Lagrange loss. Note that there is only a decision variable $z$ in $L(\pi)$, and other parameters $\alpha, \beta, \epsilon$ and $\eta$ are set to be hyperparameters. We firstly develop algorithm to be able to pick out the optimal combination units in $\pi$ for any given number of units, shown in Algorithm 1. In this algorithm, we consider the fixed encoder and decoder which have been trained on training set ahead of being input, and decide to keep or discard the units in latent variable vector based on the Lagrange loss.

**Algorithm 1** Optimization of $L(\pi)$ regarding $n$ combination units in $\pi$

| Input: $n \in [1, m]$, $L(\pi)$ |
|-----------------------------------|
| 1: $\pi = \text{ones}(m)$, a function to produce a vector with all elements to be 1; |
| 2: while not convergent do |
| 3: randomly generate $n$ non-repetitive indexes, denoted by $i_1, ..., i_n$; |
| 4: $\forall j \in [1, n]$ set $\pi_{i_j} = 0$ or 1, and enumerably generate $2^n$ vectors, labeled by $\Pi = \{\pi_1, ..., \pi_{2^n}\}$; |
| 5: for $\xi$ in $\Pi$ do |
| 6: if $L(\xi) < L(\pi)$ then |
| 7: $\pi = \xi$; |
| 8: end if |
| 9: end for |
| 10: end while |

Output: $\pi$

| Theorem 1. For any given integer $n \in [1, m]$, the Lagrange loss $L(\pi)$ in Algorithm 1 is convergent. |

Proof. Since $\pi$ is $m$-dimensional, which is finite, and its component is 0 or 1, Algorithm 1 will create a monotone sequence $L(\xi_1) > L(\xi_2) > ...$ Also, since the function values of Lagrange loss are finite, there must exist the lowest bound, which completes the proof. □

**Remark 1.** Algorithm 1 suggests that for any $n$, $L(\pi)$ or $\pi$ is convergent. However, the convergent results may be quite different even though the same inputs are fed into, since in Line 3 of the algorithm those $n$ location indexes of $\pi$ are generated randomly.

**Remark 2.** Theoretically, the appearance of the minimum Lagrange loss in Algorithm 1 may require to run it at a large $n$, which will be very time-consuming since each element in $\Pi$ should be covered to compute all $L(\pi)$'s. A practical strategy is to implement an early stopping rule that calls the algorithm off if the Lagrange loss or the sum of $\pi$ keeps unchanged for several epochs. However, the rule needs to make sure every element in $\pi$ to be visited in a large probability.

The output of Algorithm 1 essentially suggests a convergent dimension of latent variables in the model with respect to the Lagrange loss. The following theorem indicates that this result also induces the local minimum of the training precision loss.

**Theorem 2.** Assume $\pi_{\tilde{m}} \in \Pi_{\tilde{m}}$ is the output of Algorithm 1, where $\Pi_{\tilde{m}} = \{\pi \mid \sum \pi_i = \tilde{m}\}$, then the infimum of the training loss function $\inf_{\pi_{\tilde{m}}} h_{\tilde{m}}$ is achieved by the result of Algorithm 1 $\pi_{\tilde{m}}$, and $\inf_{\pi_{\tilde{m}}} h_{\tilde{m}}$ will attain the strict local minimum at $\alpha = \tilde{m}$ for all integers $\alpha \in [\tilde{m} - \tilde{n}, \tilde{m} + \tilde{n}]$ with $\tilde{n} = \min[n, \tilde{m}, m - \tilde{m}]$.

Proof. See Appendix A for the detailed proof. □

For convenience, we use the expression $\text{Opt}(L(Q, P, \pi), n)$ to represent the optimization strategy of $\pi$ through Algorithm 1. Then the adaptive dimension reduction algorithm, labeled by Algorithm 2, can be described as follows.

**Algorithm 2** Adaptive dimension reduction algorithm

| Input: $Q(\tilde{z} | x), P(\tilde{z} | x)$ (after training on training set), $\pi = \text{ones}(m)$, $L(Q, P, \pi)$ and $n$ |
|-----------------------------------|
| 1: while not convergent do |
| 2: $\pi = \text{Opt}(L(Q, P, \pi), n)$; |
| 3: set new latent variables $\tilde{z}$ with $\tilde{m} = \sum_{i=1}^m \pi_i$; |
| 4: retrain VAE or CVAE on training set to get new $Q(\tilde{z} | x)$ and $P(\tilde{z} | x)$; |
| 5: $Q = Q(\tilde{z} | x), P = P(\tilde{z} | x)$; |
| 6: $\pi = \text{ones}(\tilde{m})$; |
| 7: end while |

Output: $\tilde{m}$

Here we use “adaptive” in the name of Algorithm 2. The main reason is that the algorithm, unlike other traditional dimension reduction algorithms, such as principal component analysis etc., that usually work in need of setting the result beforehand, can compute the reduced dimension adaptively according to the data sets. A further look at the above algorithm reveals that Line 2 implements Algorithm 1. As said in Remark 1, the result of Algorithm 1 is random, which may subconsciously affect the result of Algorithm 2. However, the following theorem suggests Algorithm 2 to be convergent.

**Theorem 3.** Given a group of specific $m$ and $n$, Algorithm 2 is convergent, and the invariance of convergent dimension under it is independent of the randomness of the optimizer in Line 2.

Proof. The detailed proof can be found in Appendix B. □

**Theorem 4.** Every iteration in Algorithm 2 returns a VAE with smaller loss on training set.

Proof. Consider a random epoch in Algorithm 2, let $m$ be the dimension of the latent variables vector $z$ at the beginning of this epoch and $\tilde{m}$ be the dimension of the latent variables vector $\tilde{z}$ we get after this epoch, then there is a $\pi_{\tilde{m}} \in \Pi_{\tilde{m}}$ such that $\pi_{\tilde{m}} = \text{Opt}(L(Q^m, p^m, \text{ones}(m), n)$, thus $h_{\pi_{\tilde{m}}} - h_{\tilde{z}} \leq \epsilon$

Without losing generality, we let $\epsilon = 0$, we have $h_{\pi_{\tilde{m}}} \leq h_{\pi}$, and by the Lemma 1, $h_{\pi} \leq h_\pi$. □

**Remark 3.** The output of Algorithm 2 is the dimension of latent variables in $\mathcal{M}$, which is actually sensitive to the initial $m$ and $n$ due to running Algorithm 1. However, the invariance of convergent dimension do not rely on the initial dimension and the randomness. The VAE model after every epoch in Algorithm 2 is guaranteed to be better than the last one according to training loss.
3.3 Algorithms application to generate better VAE

Except for reducing the dimension of latent variables, the algorithms can be also used to generate the hidden layers of encoder and decoder in VAE, and get better VAE.

Consider an initial VAE only with a single layer of encoder (input layer), a middle layer of latent variables and a single layer of decoder (output layer). We use Algorithm 2 to generate the hidden layers in encoder and decoder in the form of symmetry. The first time of application of the algorithms will yield better dimension of the latent variables. Then we use this “better dimension” as the number of neural units to construct the first layer of hidden layers in encoder and decoder. Based on the new VAE (with new encode, latent variables and decoder), we make the second time of applications of Algorithm 2. The same procedures are implemented repeatedly until the number of hidden layers reach the desired value. The detailed process to generate better VAE is exhibited in Fig. 5.

Noticeably, the above “better VAE” is generated in a greedy way. During every time of applying Algorithm 2, the neural units in the hidden layers of encoder and decoder are generated as few as possible. A potential advantage of this generation pattern is that the generated VAE has fast forwarding speed and strong robustness due to reduction of neural units.

4 Experimental Studies and Discussions

In this section, we use 6 data sets to evaluate our algorithms, including MNIST, FashionMNIST, ATTfaces, Covid-19, CACD and CAT. The basic information about them is listed in Table 1. In terms of the numbers of pictures and of pixels in one picture, we sort the first two data sets as small-sized data sets, the middle two data sets as middle-sized data sets while the last two as large-sized data sets. All data sets are used to test the algorithms on VAE while only the data sets of MNIST and Covid-19 are employed on CVAE. The initial dimensions of latent variables for every data set depend on the number of pixels on one image, which are given in Table 2. We also provide the values of all hyperparameters used in the models, and the setting of the training set and testing set in this table. Here, the hyperparameters $\alpha$ and $\beta$ are set around the initial dimension of latent variables for the purpose of making a balance between dimension reduction and information reservation.

| Name of data set | Number of pictures | Number of pixels on one picture |
|------------------|--------------------|--------------------------------|
| MNIST            | 70000              | 28x28                          |
| FashionMNIST     | 70000              | 28x28                          |
| ATTfaces         | 400                | 112x92                         |
| Covid-19         | 306                | 1000x1000                      |
| CACD             | 163446             | 250x250x3                      |
| CAT              | 9000               | 500x375x3                      |

MNIST: handwritten digits; FashionMNIST: images of clothes; ATTfaces and CACD: human face pictures; Covid-19: images of x-ray in different condition; CAT: cat face pictures.

4.1 Experimental results of VAE

In this subsection, the reduction dimension algorithm is applied to the VAE model. For the small-sized data sets, we use DNN to construct the encoder and decoder of VAE. With those experimental setups reported in Table 2, Algorithm 2 is implemented on their training sets, respectively, and outputs the optimal dimension of latent variables to be 22 for MNIST and 15 for FashionMNIST. The comparisons between the initial dimension and the optimal one are displayed in Table 3. It is clear that the proposed algorithms can reduce the dimension of latent variables greatly, both from the initial 100 to the current low values. These results imply a large possibility that there will be serious overfitting phenomena in the models with 100-dimensional latent variables. To observe these phenomena, we further make experiments on the testing sets of these two data sets using the VAE models with latent variables of initial dimension, convergent dimension and others. The dimensions selection in the experiments is in large gaps at the beginning stage of initial dimension while in small gaps whilst being close to the convergent dimension. The reason is that the change of dimension will have a little effect on the model if it is far away from the convergent dimension. Shown in Table 4 are the results, where the “Loss” items are computed according to (6) and (7). As can be seen from this table, the “Loss” values at the initial dimensions are apparently lower than those at the convergent dimensions. Moreover, the latter reach the
To observe the effect of Algorithm 2 more visually, we use the trained VAE model to generate new images at some different dimensions of latent variables, especially at the convergent dimension, with the results shown in Figs. 6 and 7 for MNIST and FashionMNIST, respectively. As can be seen from these two figures, at the convergent dimensions the generated images look very clear (sub-figures (a)). However, at other dimensions, the generated images look not so clear (sub-figures (c) and (d)), and there exist some blurry parts in the images, identified by the red frames. These observations confirm again that Algorithm 2 is quite valid in improving the VAE model precision through reducing the dimension of latent variables.

Following the same procedures, we continue to test Algorithm 2 on the middle-sized (ATTfaces and Covid-19) and large-sized (CACD and CAT) data sets given in Table 1. The experimental setups are also presented in Table 2. Since the pixels on one picture is rather high for these sample sets, to save space and time we use convolutional neural network (CNN) and ReLU activation function to construct the encoder, and transposed convolution neural network (TCNN) to construct the decoder in the VAE model. Table 3 reports the convergent results for these four data sets, being 100, 4, 40 and 19, respectively. It is amazing to notice that there is significant decrease from the initial dimension to the convergent dimension for ATTfaces, Covid-19 and CACD, especially for Covid-19, the decrease from 100 to 4 attaining 25 times. This will bring a good result that the training time of the VAE model will reduce dramatically. We also evaluate the results of Algorithm 2 using the testing sets of these four data sets. The results are displayed in Tables 4 and 5. As one may expect, the VAE models still have the best performance at the convergent dimension of latent variables for middle-sized and large-sized data sets. Similarly, we use the trained VAE models to generate new ATTfaces and Covid-19 images at different dimensions of latent variables, exhibited in Figs. 8 and 9. The results show again that the images generated at the convergent dimensions approach the original images most (sub-figures (b)) while other cases contains too many unclear parts, especially like Fig. 8(c), Fig. 9(c)(d), nearly the whole pictures being unclear. Hence, Algorithm 2 is still quite strong to handle middle-sized and large-sized data sets under consideration.

| Model | Data set | Hyperparameters | Initial dimension of latent variables | Training set size vs. Testing set size |
|-------|----------|-----------------|---------------------------------------|--------------------------------------|
|       |          | α   β  | η | ε |                          |                                      |
| VAE   | MNIST    | 100 | 75  | 3  | 0.1 | 100 | 6:1 |
|       | FashionMNIST | 100 | 75  | 3  | 0.1 | 100 | 6:1 |
|       | ATTfaces | 2000 | 1800 | 1 | 1 | 2000 | 9:1 |
|       | Covid-19 | 100 | 80  | 1 | 1 | 100 | 4:1 |
|       | CACD     | 3000 | 2800 | 1 | 0.1 | 800 | 24:1 |
|       | CAT      | 50 | 40  | 1 | 0.1 | 100 | 8:1 |
| CVAE  | MNIST    | 100 | 80  | 3  | 0.1 | 100 | 6:1 |
|       | Covid-19 | 1000 | 800 | 1 | 0.1 | 100 | 4:1 |

| Model | Dimension | Data set | MNIST | FashionMNIST | VAE | CVAE |
|-------|-----------|----------|--------|--------------|-----|------|
|       | Initial   | 100      | 100    | 2000         | 100 | 22  |
|       | Convergent| 22       | 15     | 4            | 40  | 19  |

| Data set | VAE | CVAE |
|----------|-----|------|
| MNIST    |     |      |
| FashionMNIST |     |      |

To observe the effect of Algorithm 2 more visually, we use the trained VAE model to generate new images at some different dimensions of latent variables, especially at the convergent dimension, with the results shown in Figs. 6 and 7 for MNIST and FashionMNIST, respectively. As can be seen from these two figures, at the convergent dimensions the generated images look very clear (sub-figures (a)). However, at other dimensions, the generated images look not so clear (sub-figures (c) and (d)), and there exist some blurry parts in the images, identified by the red frames. These observations confirm again that Algorithm 2 is quite valid in improving the VAE model precision through reducing the dimension of latent variables.

Following the same procedures, we continue to test Algorithm 2 on the middle-sized (ATTfaces and Covid-19) and large-sized (CACD and CAT) data sets given in Table 1. The experimental setups are also presented in Table 2. Since the pixels on one picture is rather high for these sample sets, to save space and time we use convolutional neural network (CNN) and ReLU activation function to construct the encoder, and transposed convolution neural network (TCNN) to construct the decoder in the VAE model. Table 3 reports the convergent results for these four data sets, being 100, 4, 40 and 19, respectively. It is amazing to notice that there is significant decrease from the initial dimension to the convergent dimension for ATTfaces, Covid-19 and CACD, especially for Covid-19, the decrease from 100 to 4 attaining 25 times. This will bring a good result that the training time of the VAE model will reduce dramatically. We also evaluate the results of Algorithm 2 using the testing sets of these four data sets. The results are displayed in Tables 4 and 5. As one may expect, the VAE models still have the best performance at the convergent dimension of latent variables for middle-sized and large-sized data sets. Similarly, we use the trained VAE models to generate new ATTfaces and Covid-19 images at different dimensions of latent variables, exhibited in Figs. 8 and 9. The results show again that the images generated at the convergent dimensions approach the original images most (sub-figures (b)) while other cases contains too many unclear parts, especially like Fig. 8(c), Fig. 9(c)(d), nearly the whole pictures being unclear. Hence, Algorithm 2 is still quite strong to handle middle-sized and large-sized data sets under consideration.
Fig. 6: MNIST images generated by the VAE models with different latent variables: (a) original images; (b) 22 dimensions (convergent); (c) 50 dimensions; (d) 100 dimensions. The red frames identify some unclear parts of images. The same mark is used in Figs. 7, 8, 9 and 11 with the same meaning.

Fig. 7: FashionMNIST images generated by the VAE models with different latent variables: (a) original images; (b) 15 dimensions (convergent); (c) 50 dimensions; (d) 100 dimensions.

TABLE 5: Testing results of VAE on ATTfaces and Covid-19 at different dimensions of latent variables

| ATTfaces | Covid-19 |
|----------|----------|
| **Dimension** | **Loss**<sup>†</sup> | **Dimension** | **Loss**<sup>†</sup> |
| 2000 | 4646.1689 | 100 | 440538.0312 |
| 800 | 5250.0884 | 80 | 446455.8438 |
| 700 | 5260.8203 | 70 | 455817.3125 |
| 600 | 4687.7856 | 60 | 530468.6875 |
| 500 | 5556.7773 | 50 | 503117.2812 |
| 400 | 6118.9741 | 40 | 466038.4375 |
| 300 | 5914.4434 | 30 | 379977.5625 |
| 200 | 5717.3677 | 20 | 424810.0312 |
| 150 | 5324.4092 | 15 | 447669.2188 |
| 105 | 5187.8164 | 10 | 458102.1562 |
| **100**<sup>§</sup> | **4322.5444**<sup>§</sup> | **4**<sup>§</sup> | **339562.6562**<sup>§</sup> |
| 50 | 4500.3042 | 2 | 436385.90621 |

TABLE 6: Testing results of VAE on CACD and CAT at different dimensions of latent variables

| ATTfaces | Covid-19 |
|----------|----------|
| **Dimension** | **Loss**<sup>†</sup> | **Dimension** | **Loss**<sup>†</sup> |
| 800 | 107977.4453 | 100 | 310707.5000 |
| 700 | 117547.9688 | 70 | 327680.2188 |
| 600 | 100867.4531 | 60 | 324024.2188 |
| 500 | 111773.3125 | 50 | 334639.5312 |
| 400 | 109225.4609 | 40 | 324773.5312 |
| 300 | 105882.0703 | 30 | 308162.6250 |
| 200 | 109955.7734 | 20 | 304568.1250 |
| 100 | 95690.7578 | 19<sup>§</sup> | **268931.7500**<sup>§</sup> |
| 50 | 112495.2422 | 15 | 294182.7812 |
| **40**<sup>§</sup> | **94860.6016**<sup>§</sup> | 10 | 348134.4688 |
| 20 | 97928.5234 | 6 | 305552.1875 |
Finally, we use Algorithm 2 to generate better VAE through learning the data set of MNIST according to the procedures described in Section 3.3. We begin the application experiments with the initial dimension of latent variables to be 450. After the first round of running Algorithm 2, we get the convergent dimension of latent variables to be 272. We put 272 neural nodes both into the encoder and decoder to construct the first layer of hidden layer therein. Then we use the new VAE model with new encoder, 272-dimensional latent variables and new decoder to begin the second round of applying Algorithm 2. The resulting convergent dimension is 24, at which we further get the “Loss” on the testing set to be 69.664. This value is a little less than the best result “69.6778” reported in Table 4, which corresponds to the VAE model with a layer of 400-dimensional hidden layer in the encoder and decoder, and 22-dimensional latent variables. We also provide the image generated by the new better VAE model in Fig. 9, from which the image generated by this new model (subfigure (b)) looks a little more clear than other one (subfigure (c)). Superficially, the “Loss” criterion seems not a strong evidence to say the new VAE better than the previous convergent VAE. To make a farther comparison, we provide the number of parameters contained in these two models in Table 7. Clearly, there is a great difference between the total parameters numbers of two models. For the new better VAE model, the total parameters number is 447456, which decreases about 32%, compared to the result 655228 that the convergent VAE model has. Hence, with so few parameters, the new better VAE will have dramatic improvement on the speed of training and forwarding.

Note that from the viewpoint of technique, this application just uses the adaptive dimension reduction algorithm based on a greedy strategy, so it is still possible to have a better combination of different layers in the model.

Fig. 8: ATTfaces images generated the VAE models with different latent variables: (a) original images; (b) 100 dimensions (convergent); (c) 150 dimensions; (d) 400 dimensions.
Fig. 9: MNIST images: (a) original image, the same to Fig. 6(a); (b) generated by the new better VAE model; (c) the same to Fig. 6(b); (d) generated by GAN from [2]. The yellow frames in (b) serves for identifying the most clear digits for comparisons with the same most clear ones in (d).

4.2 Experimental results of CVAE

In this subsection, the CVAE model works for validating the reduction dimension algorithm, which, unlike VAE, is supervised learning. Every image is thus assigned a label beforehand, and the loss function, the optimization model, and the Lagrange loss are quite similar to those for VAE, just adding the label information \( y \) in the corresponding conditions of probability expressions. Here, we use one-hot encoding to represent the label of each image, so there are two vectors in the latent layer, i.e., the latent variables and the labels. In this case, only two data sets, a small-sized MNIST and a middle-sized Covid-19, are considered.

For the MNIST data set, each image is assigned a number, ranging from 0 to 9, as the label, i.e., the dimension of one-hot encoding is 10, which is fixed in the latent layer. And for Covid-19, a three classification task is designed, i.e., each image is labeled by a patient with Covid-19, a patient with other disease or a healthy patient. We use the parameters given in Table 2 and the initial dimension presented in Table 3, then we get the convergent dimension to be 4 and 5 for these two data sets, also exhibited in Table 3 after applying Algorithm 2 to CVAE. Further, we make some comparison experiments through testing sets between the CVAE models at the convergent dimensions and at others. The results are reported in Table 8. Also, as one might expect, the optimal results appear at the convergent dimension. In addition, the generated MNIST and Covid-19 images, shown in Figs. 11 and 12, respectively, support this point likewise. Hence, Algorithm 2 witnesses success again for CVAE working on the used data sets.

### TABLE 7: Comparison between the number of parameters in previous convergent VAE and in new better VAE

| Model Parameters  | VAE (Weights, Biases) | Better VAE (Weights, Biases) |
|-------------------|-----------------------|------------------------------|
| Hidden layer in encoder | 313600 400, 213248 272 | 213248 272, 6528 24 |
| Latent variables \( \mu \) | 8800 22, 6528 24 | 6528 24, 22 |
| Latent variables \( \Sigma \) | 8800 22, 6528 24 | 6528 24, 22 |
| Hidden layer in decoder | 8800 400, 213248 272 | 213248 272, 6528 24 |
| Output layer | 313600 784, 213248 784 | 213248 784, 6528 24 |
| Total Loss | 655228, 447456 | 447456, 69.664 |
| Loss | 69.6778 | 69.664 |

### TABLE 8: Testing results of CVAE on MNIST and Covid-19 at different dimensions of latent variables

| Dimension | MNIST Loss | Covid-19 Loss |
|-----------|------------|---------------|
| 100       | 28.6498    | 460397.3438   |
| 80        | 28.2854    | 531166.5625   |
| 70        | 28.0323    | 461905.8125   |
| 60        | 27.3108    | 521487.5938   |
| 50        | 27.0910    | 428283.6875   |
| 40        | 26.7257    | 493133.9062   |
| 30        | 26.8649    | 555563.6250   |
| 20        | 26.4686    | 565356.6250   |
| 10        | 26.3910    | 442531.8438   |
| 5         | 26.2822    | 442047.6875   |
| 4         | 25.9880    | 301410.6875   |
| 2         | 28.9239    | 540075.1250   |

For the MNIST data set, each image is assigned a number, ranging from 0 to 9, as the label, i.e., the dimension of one-hot encoding is 10, which is fixed in the latent layer. And for Covid-19, a three classification task is designed, i.e., each image is labeled by a patient with Covid-19, a patient with other disease or a healthy patient. We use the parameters given in Table 2 and the initial dimension presented in Table 3, then we get the convergent dimension to be 4 and 5 for these two data sets, also exhibited in Table 3 after applying Algorithm 2 to CVAE. Further, we make some comparison experiments through testing sets between the CVAE models at the convergent dimensions and at others. The results are reported in Table 8. Also, as one might expect, the optimal results appear at the convergent dimension. In addition, the generated MNIST and Covid-19 images, shown in Figs. 11 and 12, respectively, support this point likewise. Hence, Algorithm 2 witnesses success again for CVAE working on the used data sets.
4.3 Comparisons with other start-of-the-art methods

As a kind of popular generative models, VAE may be compared favorably with GAN. The proposed dimension reduction strategy further enhances the performance of VAE in generating clearer images. However, compared with GAN, the current model is still within the framework of VAE, and possesses the intrinsic weaknesses that VAE has. A key one is that only the lower bound, i.e., ELBO, of log \( P(x) \), but not itself, is optimized in the VAE model. Another weakness is the strong assumption to the distribution of fake images that says every pixel is independent of each other. These may lead to less clear fake images than those generated by GAN. We display their comparisons in Fig. 9, where MNIST image in subfigure (d) is generated by GAN \(^2\) and the yellow frames point out the most clear digits. By comparing them to the same most clear digits in subfigure (b), we can find digits “9” and “8” look more clear in (d).

However, VAE has its own advantages. This first one is that it can generate specific images we want, but GAN takes the Gaussian noise as an input and generates the images randomly, with the whole process totally uncontrollable. In addition, the images generative process of GAN is unstable and needs more computation resource \(^2\). Finally, instead of training a generator directly, VAE combines the encoder and decoder as a generator which gives more interpretability to the model \(^3\). This can be seen from the right hand side of the Eq. (3), where the second term measures the information from data \( x \) with the Gaussian noise i.e. how much information is encoded from data set, and the first term measures the information we need to reconstruct \( x \) from \( z \) sampled from encoder.

5 Conclusion

This paper has proposed an adaptive dimension reduction algorithm to reduce the dimension of latent variables contained in VAE related models. By introducing \( \pi \) and Lagrange loss, we have proved that the algorithm is convergent and has the property of invariance of convergent result, although there are many random factors during the operating steps. We have also proved the convergent dimension of every time experiment is the local minimum point of the infimum of the training loss function around its finite range. We use six frequently-used data sets, including two large-sized data sets, to validate that the algorithm is rather effective both in precision and in efficiency, where the latter is reflected by a great reduction of the model parameters after implementing the algorithm. In addition, the algorithm, with the help of \( \pi \), provides another point of interpretability to neural networks that which units contain noise (inactivate) and which units contain useful information (activate) in the latent variables layer. As an extra finding, the algorithm can be applied again and again to generate more compact and higher-precision VAE by symmetrically constructing hidden layers in encoder and decoder.

Note that in this paper the proposed algorithm works within the framework of VAE or CVAE. Actually, it can work for any model with the similar structure of VAE. The trick of using element-wise multiplication of \( \pi \) is also extended to other dimension reduction techniques.

Appendix: Proofs of some results

A. Proof of Theorem \(^2\)

Proof. Let \( \pi_{lh} = \text{Opt}(\text{ones}(m), n) \) where \( \pi_{lh} \in \Pi_{lh} \). Since \( \pi_{lh} \) is unchanged after enough epochs, then \( \inf_{\pi_{lh} \in \Pi_{lh}} h_{\pi_{lh} \pi_{lh}} = h_{\pi_{lh} \pi_{lh}} \) for \( i=1,2,...,\tilde{h} \)

\[
\begin{align*}
\pi_{lh} \pi_{lh} &= h \leq \epsilon \\
\pi_{lh} \pi_{lh} - h &= \epsilon \text{ or } h_{\pi_{lh} \pi_{lh}} - h > \epsilon 
\end{align*}
\]

where \( \pi_{lh} \in \Pi_{lh} \). By the assumption, we get \( h_{\pi_{lh} \pi_{lh}} < h_{\pi_{lh} \pi_{lh}} \). The units of latent variables which is related to zeros are considered as...
noise and the units which related to ones are considered contain information, thus the equality is achieved for any $\pi_{\tilde{\eta}}$, then

$$h_{\epsilon\xi\tilde{\eta}} \leq \inf_{\pi_{\tilde{\eta}}} h_{\epsilon\xi\pi_{\tilde{\eta}}},$$

(12)

In another hand, for any $\pi_{\tilde{\eta}} \in \Pi_{\tilde{\eta}}$, 

$$h_{\epsilon\xi\pi_{\tilde{\eta}}} - h_{\pi} > \epsilon$$

Therefore, $h_{\epsilon\xi\tilde{\eta}} < h_{\epsilon\xi\pi_{\tilde{\eta}}}$, by the arbitrariness of $\pi_{\tilde{\eta}}$, we have

$$h_{\epsilon\xi\tilde{\eta}} \leq \inf_{\pi_{\tilde{\eta}}} h_{\epsilon\xi\pi_{\tilde{\eta}}}.$$  

(13)

Combining the equation (13),(14) and the fact that $\inf_{\pi_{\tilde{\eta}}} h_{\epsilon\xi\pi_{\tilde{\eta}}} \leq h_{\epsilon\xi\pi_{\tilde{\eta}}}$ , we have that $\inf_{\pi_{\tilde{\eta}}} h_{\epsilon\xi\pi_{\tilde{\eta}}}$ is a minimum, since the number of elements in $\Pi_{\tilde{\eta}}$ and $\Pi_{\tilde{\eta}}$ is finite which means $\inf_{\pi_{\tilde{\eta}}} h_{\epsilon\xi\tilde{\eta}}$, and $\inf_{\pi_{\tilde{\eta}}} h_{\epsilon\xi\pi_{\tilde{\eta}}}$ are achievable, so we can replace the $\leq$ into $<$, then we have the strictness of this local minimum. □

B. Proof of Theorem 3

Proof. First of all, we assume that the $Q(\pi|x),P(x|\pi)$ have been trained on the training set and have the same parameters as $Q(\pi \otimes z|x),P(x|\pi \otimes z)$ do.

We use $z'$ to represent the latent variable whose dimension is $i$

It is easy to prove that $m = 0$ can not be the convergent dimension since $m = 0$ means no data are passed to the decoder, the fake data generated by the decoder is all noise and the equation (11) can not be achieved, so if we set $z'$ as the initial latent variable, it will not decrease anymore. Since the $z'$ is finite, there is a integer $s$ such that dim $z'^{s}$ as the initial latent variable is changeable and $z'^{s}$ is unchangeable. That is to say the Algorithm 2 is convergent

We assume that we run the Algorithm 2 at the initial dimension $m$ and return $\tilde{\eta}$, then by the definition of the algorithm, $\tilde{\eta}$ is convergent dimension if and only if for $i=1,2...\text{min}(\tilde{\eta},\tilde{m})$, 

$$h_{\epsilon\xi\pi_{\tilde{\eta}}} - h_{\pi} \geq \epsilon.$$  

Since $\tilde{\eta}$ is unchanged after enough epoch, by the arbitrariness of $\pi_{\tilde{\eta}}$, we have

$$\inf_{\pi_{\tilde{\eta}}} h_{\epsilon\xi\pi_{\tilde{\eta}}} - h_{\pi} \geq \epsilon,$$

which is independent with the choice of $\pi_{\tilde{\eta}}$, thus is independent with the randomness of Algorithm 1.

Acknowledgments

This work was funded by the National Nature Science Foundation of China under Grant 12071428 and 11671418, and Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ20A010002.

References

[1] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” CoRR, vol. abs/1312.6114, 2014.
[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in NIPS, 2014.
[3] C. Doersch, “Tutorial on variational autoencoders,” ArXiv, vol. abs/1606.05908, 2016.
[4] T. Salimans, D. Kingma, and M. Welling, “Markov chain monte carlo and variational inference: Bridging the gap,” in International Conference on Machine Learning, pp. 1218–1226, PMLR, 2015.
[5] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in International conference on machine learning, pp. 1278–1286, PMLR, 2014.
[6] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, “Draw: A recurrent neural network for image generation,” in International Conference on Machine Learning, pp. 1462–1471, PMLR, 2015.
[7] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” in NIPS, 2015.
[8] J. Walker, C. Doersch, A. Gupta, and M. Hebert, “An uncertain future: Forecasting from static images using variational autoencoders,” ArXiv, vol. abs/1606.07873, 2016.
[9] Q. Liu and D. Wang, “Stein variational gradient descent: A general purpose bayesian inference algorithm,” in NIPS, 2016.
[10] I. Higgins, A. LM, X. CB, and M. MB, “Beta-vae: Learning basic visual concepts with a constrained variational framework,” Proc. ICLR, Toulon, France, 2017.
[11] M. J. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta, “Composing graphical models with neural networks for structured representations and fast inference,” Advances in neural information processing systems, vol. 29, pp. 2946–2954, 2016.
[12] I. T. Jolliffe and J. Cadima, “Principal component analysis: a review and recent developments,” Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2065, p. 20150202, 2016.
[13] J. Platt, “Sequential minimal optimization : A fast algorithm for training support vector machines,” Microsoft Research Technical Report, 1998.
[14] S. Boyd, S. P. Boyd, and L. Vandenberghe, Convex optimization. 2004.
[15] B.-C. Chen, C.-S. Chen, and W. H. Hsu, “Cross-age reference coding for age-invariant face recognition and retrieval,” in Proceedings of the European Conference on Computer Vision (ECCV), 2014.