2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-7281-8470-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICMLA51294.2020.00088

Dimension Estimation Using Autoencoders with Applications to Financial Market Analysis

Nitish Bahadur
Data Science
Worcester Polytechnic Institute
Worcester, MA
nbahadur@wpi.edu

Randy Paffenroth
Data Science
Worcester Polytechnic Institute
Worcester, MA
rcpaffenroth@wpi.edu

***Abstract*—Dimension Estimation (DE) and Dimension Reduction (DR) are two closely related topics, but with quite different goals. In DR, one attempts to project a random vector, either linearly or non-linearly, to a lower dimensional space that preserves the information contained in the original higher dimensional space. However, in DE, one attempts to estimate the intrinsic dimensionality or number of latent variables in a set of measurements of a random vector. DE and DR are closely linked because reducing the dimension to a smaller value than suggested by DE will likely lead to information loss. In particular, when considering linear methods, such as Principal Component Analysis (PCA), DE and DR are often accomplished simultaneously. However, in this paper we will focus on a particular class of deep neural networks called autoencoders which are used extensively for DR but are less well studied for DE. We show that several important questions arise when using AEs for DE, above and beyond those that arise for more classic DR/DE techniques such as PCA. We address AE architectural choices and regularization techniques that allow one to transform AE latent layer representations into estimates of intrinsic dimension. We demonstrate the effectiveness of our techniques on synthetic, image processing benchmark problems, and, most importantly, the analysis of financial markets.**

***Index Terms*—PCA, Autoencoder, dimension estimation**

I. Introduction

Dimension estimation (DE) is the process of determining the *intrinsic dimensionality* of data (e.g., see Chapter 3 in [1]). Usually, real-world datasets have large numbers of features, often significantly greater than the number of latent factors underlying the data generating process, and DE attempts to quantify the number of latent factors in a dataset. Of course, latent factors are often discussed in relation to linear analysis. However, such ideas can be generalized to a non-linear context, and such generalizations will be our focus here. For example, in an image processing problem each pixel of an image can be thought of as a feature that one measures about the image. However, the measured pixels are not independent of each other since, for example, nearby pixels are likely to have similar colors. Accordingly, it can be quite useful to estimate the underlying latent factors, such as pose and lighting, that effect many pixels simultaneously. The precise definition of a latent factor, and therefore the precise definition of the intrinsic dimensionality, can be quite challenging. For example, Principal Component Analysis (PCA) [2] defines latent factors in terms of linear projections and orthogonality. At the other extreme, one can consider quite complicated scenarios involving fractal dimensions and space filling curves [1]. In this text we take a middle of the road approach, and focus on non-linear, but smooth, manifolds. Such an approach is quite popular, and such DR and DE techniques are used in diverse domains such as engineering, astronomy [3], biology [4], remote sensing [5], economics [6] [7], social media [8], and finance [9] [10]; and this class of techniques has a large extant literature (see, e.g., [1], [11] and references therein).

While PCA uses a linear projection to perform both DR and DE, in this paper our focus is on using non-linear deep autoencoders (AE) for estimating the intrinsic dimensionality of data sets. In particular, while deep AEs, and deep neural networks (NN) in general are used extensively for DR, using AEs for DE is less well studied. While quantifying intrinsic dimension using a linear technique such as PCA is standard, estimating the dimension of real-world time series datasets using AE is more challenging. For example, [12] found that classification accuracy of MNIST [13] and human faces [14] dataset increased with numbers of nodes in AE hidden layer. Our approach is fundamentally different because we refrain from changing the number of hidden nodes for a DE and instead use sparsity inducing penalties to regulate number of non-zero hidden nodes. As another example, [15] used a NN to measure the intrinsic dimension of a NN itself, which is different from our goal of estimating intrinsic dimension of the manifold on which the data lie. Perhaps most importantly, our approach is inspired by [16], where the author's demonstrate that, even when the number of hidden units is large, we can still discover interesting structure by imposing sparsity constraints on the network, without changing the network architecture.

While DR has received greater attention than DE in the NN community, there are several important domains in which DE is actually a more important problem then DR. In particular, a widely studied problem in finance is stock portfolio diversification and such problems are closely connected to DE [17], [18], [19]. The cost to diversify a portfolio is comprised of *the number of shares* of securities and *the number of different securities in the portfolio* [20]. Consider, for example, the following: Portfolio A, with an expected

978-1-7281-8470-8/20/$31.00 ©2020 IEEE
DOI 10.1109/ICMLA51294.2020.00088

return of 14%, a standard deviation of .1231, and containing *20 securities*; and Portfolio B, with an expected return of 14%, a standard deviation of .1238, and containing *15 securities*. A linear covariance-based portfolio analysis would indicate that Portfolio A is superior giving its marginal improvement of a factor of 0.0007 in portfolio standard deviation. However, if the securities had *non-linear dependencies* then the above analysis would not tell the full story. In particular, non-linear interactions can lead linear methods to overestimate the intrinsic dimensionality of data [1], and the addition of 5 securities, with their associated marginal cost, may actually be unnecessary. *Perhaps even more importantly, it is also known that the dimensionality of financial markets can change over time* [21] and detecting such dimensionality changes is one of the key motivators of our approach.

A. Contributions and organization

AEs are designed for DR, but we are interested in using AE for estimating *intrinsic dimension*. Unlike PCA, AEs do not natively support DE in that, for example, there are no ordered equivalents to singular values in the hidden layer that can be used to estimate dimension. Therefore, an AE's innermost hidden layer values must be transformed into quantities that are faithful to the *singular values* that are classically used in PCA, quantities that we call here singular value proxies (SVP), to facilitate estimating dimension. Such SVPs are the focus of our research.

Accordingly, the main contributions of the paper can be summarized as follows:

1) We developed an algorithm to estimate the non-linear intrinsic dimension using AEs. The values of the innermost hidden layer of classic AEs are not appropriate for DE. Hence, we developed several transforms and normalizations of the innermost layer of AEs to create singular value proxies (SVP) for dimensionality estimation.
2) We demonstrate these techniques on synthetic data sets, and demonstrate that our methods robustly and exactly recover the non-linear dimensionality of complicated low-dimensional manifolds, when competing methods either under-estimate or over-estimate the true intrinsic dimension of the data.
3) Further, we demonstrate how dimension estimation can be applied to the benchmark MNIST dataset and a financial time series dataset comprised of the S&P 500 index constituents.

This paper is structured through six sections. In Section I, we provide the problem definition and a brief introduction of relevant existing literature. We provide an overview of DR and building blocks to estimate dimension in Section II. Section III elaborates architectural choices we make in designing AE for DE that allow for creating SVPs for AE. Section IV details an algorithm to quantify the SVP into dimensionality estimates. Section V presents our experiments with synthetic data, image dataset MNIST, and financial market S&P 500 dataset. Additionally, Section V compares and contrasts the differences between PCA and AE. Finally, Section VI provides a summary and pointers to future work.

II. Dimension Reduction

In this section, we provide an overview of the linear dimension reduction technique such as PCA and the nonlinear DR technique such as AE. DR is the process of reducing a high dimension dataset with N features into a dataset with p features, where $p \ll N$. Perhaps the most classic method in this domain is PCA [22], and a detailed study of PCA will illuminate many of the issues that arise when performing DR and DE with AE. Unfortunately, most DR methods need an estimate of the number of latent variables as a user-defined input to the process of dimensionality reduction. Estimates of the number of latent variables in a particular dataset can come from a variety of principles (e.g., see Chapter 3 in [1]).

A. Linear Dimension Reduction Techniques

Linear techniques such as PCA [23] have seen wide use. The key idea of PCA is to construct low-dimensional subspaces that preserve as much of the variance in the data as possible and thereby preserve data correlation structure.

1) Principal Component Analysis: PCA [23], a linear DR technique, is commonly implemented using the singular-value decomposition (SVD) [2] where a data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ is factored as in

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} \tag{1}$$

where,

- $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ is an ortho-normal (or unitary) matrix.
- $\boldsymbol{\Sigma}$ is a rectangular "diagonal" matrix with the same size as $\boldsymbol{X}$; the m entries $\boldsymbol{\sigma}_m$ on the diagonal are called the singular values of $\boldsymbol{X}$.
- $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ is an ortho-normal (or unitary) matrix.

The linear low-dimensional project of X to k dimension space is $\boldsymbol{U}\hat{\boldsymbol{\Sigma}}$ where $\hat{\boldsymbol{\Sigma}}$ has its rightmost $n-k$ columns removed, corresponding to the $n-k$ smallest singular values σ_i. Note, PCA provides two important features that we will use in the sequel. Namely, the columns of $\boldsymbol{U}$ are ortho-normal and the singular values $\boldsymbol{\sigma_i}$ are ordered from largest to smallest.

Of course, in the presence of finite samples and noise in the data, the choice of an appropriate k is non-trivial. In particular, any σ_i with the property that $\sigma_i = 0$ can be removed without changing the properties of the data (e.g., the Euclidean distances between the points). However, in real-world data it is rarely the case that $\sigma_i = 0$ for any σ_i, and DR using PCA will change the properties of the data in $\boldsymbol{X}$.

B. Example of DE: MNIST hand-written digits

As a running example in this paper, consider images of hand-written digits from the MNIST dataset. For example, using PCA and a user-defined estimate of the number of latent variable $k = 50,\ 100,\ 200,\ 400,\ 784$ we reduce the number of features of MNIST digits and reconstruct digits. Note, already a quite important issue has arisen. How is one to choose k? To assist DE a scree plot that plots the sizes

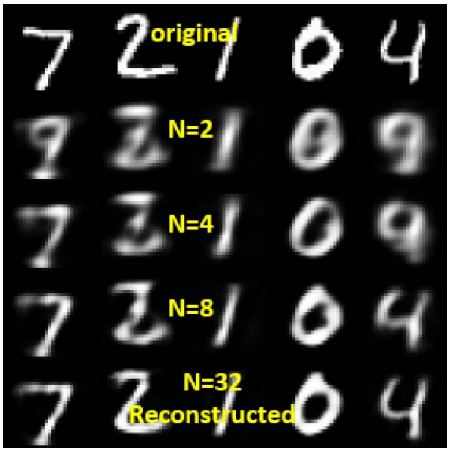


Fig. 1. This figure shows the reconstruction performance as the number of nodes (n) in the AE hidden layer increase from 2 to 32, the reconstructed image visually resembles more and more like the input.

of the singular values σ_i can be used. Moreover, one may expect that reconstruction error will remain low if k is greater than the intrinsic dimensionality of the linear PCA embedding. On the contrary, if k goes below the intrinsic dimensionality, the dimensionality reduction may cause a sudden increase in reconstruction error.

While we use PCA to establish a baseline, we are most interested in moving beyond the linear DR provided by PCA and estimating non-linear dimensionality of datasets. Therefore, we study how AE [24], [25], a classic deep NN used for DR, can also be used for DE.

C. Autoencoders

An AE is a type of artificial NN used to learn an approximation to the identity function, so that the output $\hat{\boldsymbol{X}}$ is similar to n-dimensional input $\boldsymbol{X}$. Because the hidden layer acts a bottleneck, when the AE compresses the input to a latent space representation and then reconstructs the output from the latent space representation, the hidden units encode significant features in input data $\boldsymbol{X}$.

Just like k, which defines the number of principal components in PCA, n (Fig. 1), which defines the number of nodes in the innermost hidden layer, needs to be estimated before building the AE. Again, how do we estimate the appropriate number of nodes in the innermost hidden layer? PCA and AE are closely related [26] and an AE with linear activation function and squared loss function will compute the same subspace as PCA [26]. However, the parameterization of the subspace can be quite different. For example, the SVD guarantees that the basis is orthogonal and the singular values are ordered. *Running gradient descents, a common optimization routine for NN, on an AE makes no such guarantees. Restoring such guarantees for an AE is the focus of our work.* AEs are, in many ways, quite similar to PCA. In particular, at their simplest, an AE [16] with one layer hidden takes input data $\boldsymbol{x}_i \in \mathbb{R}^{n\times 1}$ as transforms that vector into a new $\boldsymbol{y}_i \in \mathbb{R}^{k\times 1}$, often called the hidden layer, according to the mapping

$$\boldsymbol{y}_i = \theta(\mathbf{W}_1\boldsymbol{x_i} + \boldsymbol{b_1}) \tag{2}$$

where $\mathbf{W}_1$ is the weight matrix of the first layer, and $k < n$. The function θ is typically a non-linear activation function such as a sigmoid [27] or a rectified linear unit (ReLU) [28] [1]. Another layer then maps $\boldsymbol{y_i}$ to $\hat{\boldsymbol{x_i}} \in \mathbb{R}^{n\times 1}$ according to

$$\hat{\boldsymbol{x_i}} = \theta(\mathbf{W}_2\boldsymbol{y_i} + \boldsymbol{b_2}) = \theta(\mathbf{W}_2\theta(\mathbf{W}_1\boldsymbol{x_i} + \boldsymbol{b_1}) + \boldsymbol{b_2}) \tag{3}$$

where $\mathbf{W}_2 \in \mathbb{R}^{n\times k}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{n\times 1}$ are the weight matrix and bias vector of the second layer. Deep NN with many layers can be defined in an analogous fashion. The parameters $\mathbf{W}_1$, $\boldsymbol{b_1}$, $\mathbf{W}_2$, $\boldsymbol{b_2}$ are found by minimizing some cost function (4) that quantifies the difference between the output $\hat{\boldsymbol{x_i}}$ and input as in

$$J(\mathbf{W}, \mathbf{b}; \mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2, \tag{4}$$

with $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2\}$ and $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2\}$, leading to the optimization problem

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W}, \mathbf{b}; \mathbf{x}). \tag{5}$$

At this point, we can begin to understand the differences between PCA and AE. In particular, $\boldsymbol{y_i}$ is a projection of $\boldsymbol{x_i}$ into a k dimensional space, just as $\boldsymbol{U}\hat{\boldsymbol{\Sigma}}$ is a projection of the rows of a data matrix $\mathbf{X}$ into a k dimensional space. However, that is where the analogy ends. While the basis in $\boldsymbol{U}$ is explicitly ortho-normal and the diagonal entries in $\hat{\boldsymbol{\Sigma}}$ are ordered by size, the structure of the AE is hidden inside the nonlinear function $\theta(\mathbf{W}_1\boldsymbol{x_i} + \boldsymbol{b_1})$. *So, as opposed to the singular values in $\boldsymbol{\Sigma}$, the size of the entries in $\boldsymbol{y_i}$ do not provide information about the intrinsic dimensionality of the data set.*

Fortunately, the ideas in [16] provide a path forward. When a NN is constrained by k being small, then the network is forced to learn a compressed representation of input. However, when k is not known beforehand, which is the entire point of DE, one can penalize the hidden layer entries $\boldsymbol{y_i}$ so that they are encouraged to be small, or even 0. To achieve sparsity in the innermost hidden layer we penalize the innermost hidden layer with the ℓ_1 norm denoted by $\|.\|_1$. The penalty forces small hidden layer entries to zero and prevents over estimation of dimension. In that way, the number of large entries in $\boldsymbol{y_i}$ can be interpreted as a measure of the intrinsic dimension. Additionally, to achieve consistency in DE and make the DE process independent of range of numeric values we use $\|.\|_2$, which normalizes the input values into the innermost hidden layer.

DE requires both sparsity and consistency, which is shown in Fig. 3a and Fig. 3b. Accordingly, we follow the ideas in [16] and start by normalizing the hidden layer as in (6)

$$\boldsymbol{y}^i_{norm} = \frac{\boldsymbol{y_i}}{\sqrt{\sum_{i=1}^{k} \boldsymbol{y_i}^2}} \tag{6}$$

[1]We use the notation θ for the non-linear activation function instead of the more standard σ, as σ is also used to denote singular values.

TABLE I
AE ARCHITECTURE

Layer Type	Nodes	Regularizer	Activation
input layer	784 or 494	-	-
encoder layer 1	392	-	relu
dropout layer 1	drop 30%	-	
encoder layer 2	128	-	relu
dropout layer 2	drop 30%	-	
lambda layer	128	-	-
hidden layer	64 or 100	$l1$, as in (7)	sigmoid
decoder layer 1	128	-	relu
decoder layer 2	392	-	relu
output layer	784 or 494	-	sigmoid or tanh

$$J_{sparse}(\mathbf{W}, \mathbf{b}; \mathbf{x}) = J(\mathbf{W}, \mathbf{b}; \mathbf{x}) + \lambda \sum_{i=1}^{k} \|\boldsymbol{y}^i_{norm}\|_1 \qquad (7)$$

to force some entries of $\boldsymbol{y_i}$ to be small or even 0. The number of large entries in $\boldsymbol{y_i}$, when appropriately interpreted, as described in the sequel, provides an estimate of the dimensionality of $\mathbf{X}$.

1) Autoencoder Model: To estimate dimension of MNIST digits the AE model uses a 9 layer AE. The encoder layer also includes 2 dropout layers to reduce over-fitting. Table I defines the layers, activation function and regularizers. The MNIST input and output layer has 784 nodes; the output layer uses the sigmoid activation function. Similarly, in the sequel when estimating dimension of financial time series we use the S&P 500 index constituents. While the AE model is similar, the input and output layers have 494 nodes and $tanh$ activation function is used in the output layer, since returns of stocks cannot be negative.

III. Autoencoder - Singular Value Proxies

As we are primarily interested in DE using the AE innermost hidden layer, which does not natively provide singular value equivalent that can be quantified into intrinsic dimension of the dataset, we create *singular value proxies* using the innermost hidden layer values.

A. L1 Hidden Layer Regularizer

AE faithfully reproduces the input without worrying about the transformations the hidden layers go through. Our goal is to adapt the AE innermost hidden layer entries to estimate dimension of the dataset. We leverage [16] approach by using large number of hidden units and still discover interesting structure by imposing sparsity constraints on the innermost hidden layer. To make DE consistent and small, we impose $\|.\|_1$ penalty on the innermost hidden layer entries. While the penalty enhances sparsity in the innermost layer it diminishes the reproducing capability of AE, as the loss worsens. As illustrated in Fig. 2, we experiment with several λ values to tune the AE. For our synthetic data, we know the true dimension. Accordingly, we can choose out optimal λ as the one that achieves this dimension. However, in our real-world examples, we generate different SVP profiles in a similar fashion, but choose the optimal λ value as the one for which the neural network loss stops changing as we increase λ further. In other words, we choose the largest penalty such that the λ penalty does not adversely affect the loss of the autoencoder.

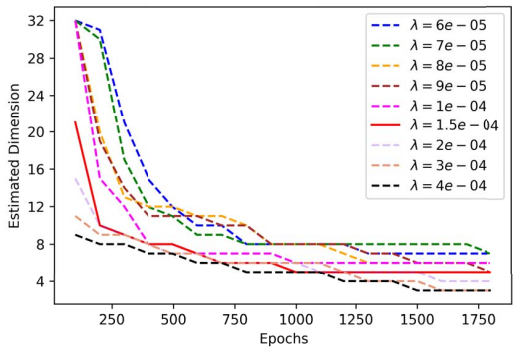


Fig. 2. The innermost hidden layer values for synthetic data $\boldsymbol{D}$ for different λ shows how regularization parameter affects sparsity of the hidden layer. For $\lambda = 1.5e-4$, the red line, DE is 5. In our real-world examples, we generate different SVP profiles in a similar fashion, but choose the optimal λ value as the one for which the neural network loss stops changing as we increase λ.

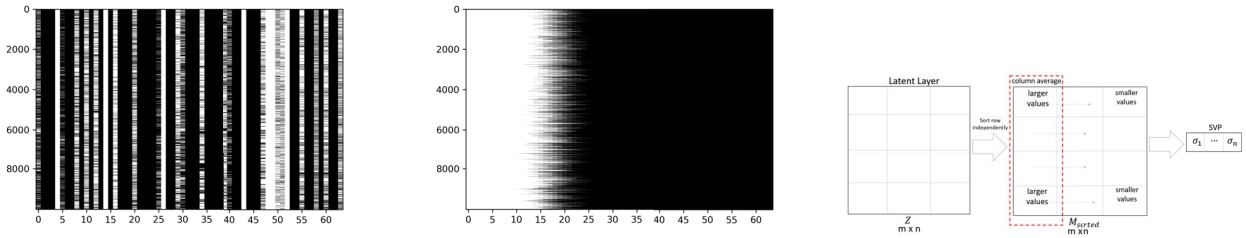
B. Latent Space Sorting

Given a sparse latent layer, as shown in Fig. 3a, our next step is to refashion the hidden layer values into SVP to facilitate DE of the dataset. To refashion the hidden layer, 2 approaches were evaluated: first to take the average value of features (columns of hidden layer)(Fig. 3) and create SVP for DE and second, sort each row hidden layer independently such that the larger values are on the left and smaller values are on the right (Fig. 3b and then take the column average to create SVP for DE. We preferred the second approach because this is equivalent to singular value matrix, where the larger values are on the left and smaller values are on the right. Additionally, sorting first and then taking averages moderately amplifies the larger SVP and gives provides slightly lower estimate of intrinsic dimension. While we calculate SVP using both methods, we only report intrinsic dimension where we sort first and then take averages. Algorithm 1 provides the steps we follow.

Algorithm 1 Transforming Latent Representation to Singular Value Proxies

Input: The hidden layer $\boldsymbol{Z} \in \mathbb{R}^{m \times n}$ in
Output: σ_i, singular value proxies for DE. out

1: Sort each row of $\boldsymbol{Z}$ independently, so that for each row the largest value is on the left and smallest value is on right. $\boldsymbol{M}_{sorted} = row\_sort(\boldsymbol{Z})$
2: Take average of each column of $\boldsymbol{M}_{sorted}$ to calculate SVP.
3: SVP: $svp_i = \frac{\sum_{i=1}^{m} \boldsymbol{Z}_{i,n}}{m}$
4: We get SVPs $\{svp_1, svp_2, svp_3, ..., svp_n\}$ for n features.



(a) Latent Layer (b) Re-fashioned latent layer (c) SVP creation process

Fig. 3. (a) Without any transformation, the innermost hidden layer, which has 64 nodes, for all the $10,000$ MNIST digits in the test dataset is shown here. (b) After sorting 64 innermost hidden layers (rows) independently we get an image where the larger hidden layer values are on the left and smaller values are on the right. (c) Steps needed to convert a latent representation into singular value proxies that can be used to estimate dimension.

IV. Dimensionality Estimation Algorithm

The linear dimension of data is usually higher than the intrinsic dimension. The noise in real data prevents the singular values from being exactly 0. To prevent smaller SVP from inflating the DE we use 2 different analytic:

1) Greater Than Equal To 1%: Count the singular values or SVP that are larger than 1% to estimate intrinsic dimension. The threshold 1% is configurable. (Algorithm: 2)
2) Up to 90%: Using the largest singular values or SVP, count the number of values required such that the cumulative value is larger than 90% of the sum of the values. Again, 90% threshold is configurable. (Algorithm: 3)

Algorithm 2 Dimensionality using Greater Than Equal To $\alpha = 1\%$

Input: $(\sigma_1, \sigma_2, ...\sigma_n)$, singular values, $\alpha = 1\%$ in
Output: p, the number of values greater than equal 1%. out

1: Calculate $\sigma_{sum} = \sum_{i=1}^{n} \sigma_i$
2: Dimensionality $p = \sum I(\sigma_{i\%})$, where

$$I(\sigma_{i\%}) = \begin{cases} 1 & : \text{if } \frac{\sigma_i}{\sigma_{sum}} \geq 1\% \\ 0 & : \text{otherwise,} \end{cases} \tag{8}$$

Algorithm 3 Dimensionality using up to $\alpha = 90\%$

Input: $(\sigma_1, \sigma_2, ...\sigma_n)$, singular values, $\alpha = 90\%$ in
Output: p, the number of values greater than equal 1% out

1: Sort $(\sigma_1, \sigma_2, ...\sigma_n)$ in descending order, where $\boldsymbol{\sigma}_i$ are singular values.
2: Calculate $\sigma_{sum} = \sum_{i=1}^{n} {\sigma_i}^2$
3: Calculate $\sigma_{i\%}$, where $\sigma_{i\%} = \frac{{\sigma_i}^2}{\sigma_{sum}}$
4: Dimensionality p, is the value of l where $\sum_{l=1}^{w} \sigma_{i\%} \geq \alpha(90\%)$

V. Experiments

Our experiments comprise of 2 parts. In the first part we estimate dimension of each digit in the MNIST dataset. The second part estimates financial market dimension using S&P 500 constituents. While the MNIST data is static in nature, the S&P 500 dataset has varying correlation among index constituents.

A. Estimate Dimension of Synthetic Data

We begin out experiments by considering synthetic data set $\boldsymbol{C} \in \mathbb{R}^{5000\times784}$, where $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$, $\boldsymbol{A} \in \mathbb{R}^{5000\times5}$ and $\boldsymbol{B} \in \mathbb{R}^{5\times784}$. As this is a linear problem, the singular value decomposition (SVD) [2] of $\boldsymbol{C}$ correctly gives us 5 (Fig. 5) non-zero singular values. However, to test our non-linear DE method, we apply non-linear transformation to columns of $\boldsymbol{C}$ to create $\boldsymbol{D} \in \mathbb{R}^{5000\times784}$. The steps are listed in Algorithm 4. We use several highly non-linear maps such as x^y, e^y, $1/y$, and 2^y. However, as these non-linear maps are all *parameterized by the same low-dimensional* $\boldsymbol{C}$, *we can guaranteed that all of the synthetic data points lay on a non-linear manifold on intrinsic dimension* 5. The first 3000 of $\boldsymbol{D}$ forms the training set and the remaining 2000 instances form the test set. Using SVD to estimate the dimension of $\boldsymbol{D}$ gives us 20 (Fig. 5), a number significantly larger than 5. However, the DE provided by our AE model, as illustrated in Table I, robustly and exactly recovers the correct dimension of 5. In particular, the dimension is estimated by taking the mean values of the innermost hidden layer features, column wise, as shown in Fig. 4. In addition, to mitigate over-fitting, we use 2 dropout layers where we drop 30% of nodes in each dropout layer; the dropout layers make the DE more stable as shown in Fig. 5. This stylized example demonstrates how AE can be used to estimate dimension for non-linear data sets, and the various parameter settings used for this synthetic data are used in all experiments, unless otherwise noted.

B. Estimate MNIST digit dimension

MNIST is a publicly available dataset comprising digits 0 to 9. There are 60,000 training images and 10,000 testing images. Each image is 28×28, which gives us 784 features. Each feature is the pixel intensity, which is normalized before dimension is estimated. We use a 7-layer AE (Table. I) with $l1$ penalty on innermost hidden layer with $\lambda = 1e-5$. DE is 9, which is illustrated in Fig. 3a and Fig. 3b. To study the impact of outliers to estimated dimension by PCA and AE we

Algorithm 4 Create $\boldsymbol{D}$ by applying non-linear transformation to columns of $\boldsymbol{C}$

Input: $\boldsymbol{C} \in \mathbb{R}^{5000\times784}$ in
Output: $\boldsymbol{D} \in \mathbb{R}^{5000\times784}$ out

1: **repeat**
2: $i \leftarrow 0$ {the first column of $\boldsymbol{D}$}
3: Select 2 random columns from $\boldsymbol{C}$ where $col_1 \leftarrow [1, 784]$ and $col_2 \leftarrow [1, 784]$
4: Select 2 non-linear functions f_1 and f_2 from
(i) $\{f \mid f\ is\ x^y$ (power)$\}$,
(ii) $\{f \mid f\ is\ e^y$ (exp)$\}$,
(iii) $\{f \mid f\ is\ \frac{1}{y}$ (reciprocal)$\}$,
(iv) $\{f \mid f\ is\ 2^y$ (exp2)$\}$
5: $\boldsymbol{D}_{col_i} \leftarrow f_1(\boldsymbol{C}_{col_1}) + f_2(\boldsymbol{C}_{col_2})$
6: **until** i = 784

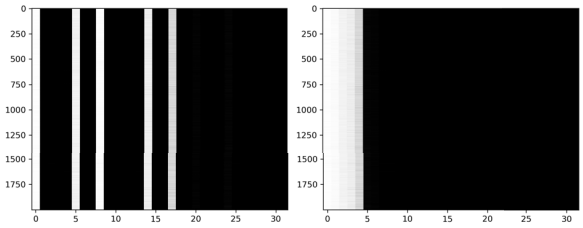


Fig. 4. AE correctly estimating the dimension of synthetic dataset $\boldsymbol{D}$, a non-linear transformation of $\boldsymbol{C}$, as 5. The white bands in the innermost hidden layer plot represents the features of the synthetic dataset. The y-axis indicates the 2000 test instances and the x-axis represents the 32 nodes of the innermost hidden layer. The image on the right indicates independently sorted rows to facilitate SVP creation for DE.

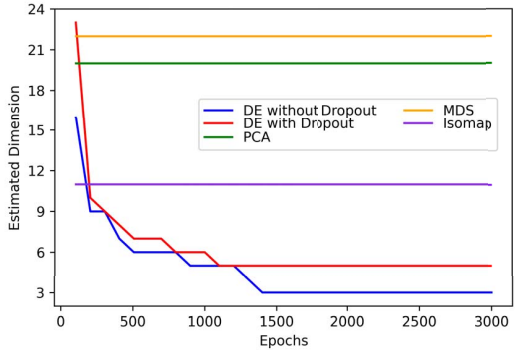


Fig. 5. The plot compares synthetic data, $\boldsymbol{D}$, DE using PCA, MDS, Isomap, without dropout in AE and with 2 30% dropout layers sandwiched between encoder layers. With dropouts in encoders, DE stabilizes at 5 around 1100 epochs and remains stable.

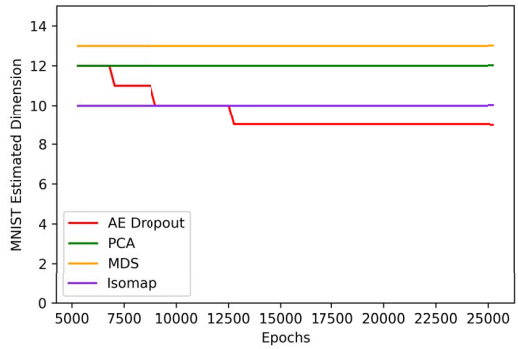


Fig. 6. MNIST DE plot compares DE by PCA, MDS, Isomap and AE with 2 30% dropout layers in the encoder. As represented by the red line, AE estimates MNIST dimension as 9. The dimension stabilizes after 14000 epochs. The plot indicates the snapshot of estimated dimension every 1000 epochs and we start at epoch 5000 to emphasize the differences between the trained AE and the competing methods.

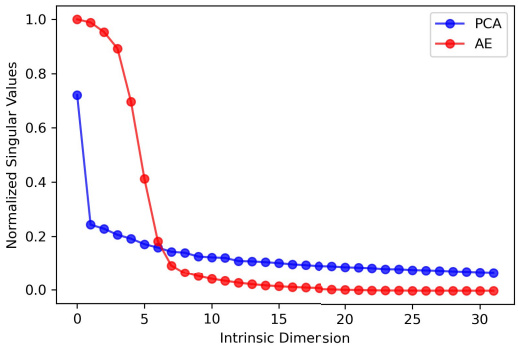


Fig. 7. The MNIST scree plot compares the normalized singular values of PCA to SVP of AE used for DE. The elbow shaped red line in the scree plot shows that MNIST has 9 independent dimensions.

only select images whose mean square error (MSE) is below a threshold varying between 0.002 and 0.09. For example, when we use a MSE threshold of 0.002 we have 137 instance of digit 1 from the test set, because the digit 1 image has no curves compared to digit 8. We estimate both PCA (linear) and AE (non-linear) dimension and show in Fig. 8. We find that as we increase the MSE threshold more and more outliers images get added to our dataset, and this results in larger singular values muting the impact of other singular values, and consequently decreasing the linear estimated dimension. However, the non-linear dimension estimated by AE is stable (Fig. 8).

C. Estimate S&P 500 dimension

When stocks are randomly selected and combined in equal proportions into a portfolio the risk of a portfolio declines as the number of different stocks in the portfolio increases. In [20], the authors found that the economic benefits of diversification are virtually exhausted when a portfolio contains ten stocks. However, [21] studied financial markets from

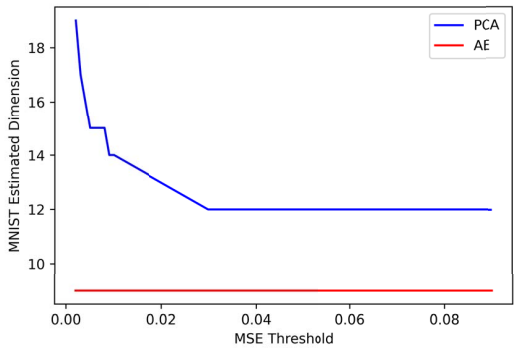


Fig. 8. We compare the change in estimated dimension between PCA and AE. As we increase the MSE threshold more outlier images get added to our test dataset and consequently result in larger σ_i, which is used for DE. Larger σ_i dominates and mutes the effect of other σ_i and leads to lower dimension. DE by AE is not affected.

the perspective of low-dimensional manifolds and found that the dimension is time varying and not static. To estimate dimension using AE we use S&P 500[2] index constituents. The dataset is comprised of 585 days (from 01/02/2018 to 04/30/2020) of end-of-day prices. We normalize end of day prices by calculating daily return $r_{i,t}$ (Eqn. 9), where daily return is defined as

$$r_{i,t} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}} \tag{9}$$

where, $P_{i,t}$ is end of day adjusted close price of security i on day t. We can then form a data matrix $\boldsymbol{X}$ with $X_{i,t} = r_{i,t}$. In particular, given d days and k instruments we have that

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_j \\ \vdots \\ \boldsymbol{x}_d \end{bmatrix} \in \mathbb{R}^{d \times k} \tag{10}$$

468 days (from 01/02/2018 to 11/11/2019) are used for training and 117 days (from 11/12/2019 to 04/30/2020) are used for testing. S&P 500 AE model layers, nodes and activation functions are illustrated in Table. I. We find that the intrinsic dimension of S&P 500 is 30 as shown in Fig. 9. DE can also be used to detect anomalies in time series. As illustrated in Fig. 10, when we use trained AE to estimate dimension from November 12, 2019 to April 30, 2020 we find large reconstruction error during March and April 2020. Because of pandemic in 2020, global economy slowed down and financial markets suffered huge losses. The extreme market dislocation is exhibited by large reconstruction error. However, we believe this is not a limitation of the AE DE process but rather a facet of dataset that will be addressed in our future work by using

[2]S&P 500 is an equity index that measures the stock performance of 500 large companies listed on stock exchanges in the United States.

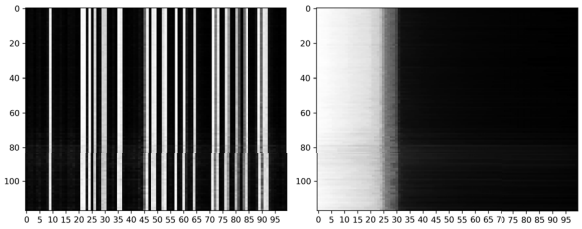


Fig. 9. AE estimates that intrinsic dimension of S&P 500 is 30. The white bands in the left image shows the 30 intrinsic dimensions. The y-axis are the 117 days(Nov 2019 - April 2020) and the x-axis is the 100 units in the innermost hidden layer. The image on the right shows the innermost hidden layer after rows are independently sorted. We use the independently sorted rows to create SVP and subsequently estimate dimension. $\lambda = 5e-5$ and 2 dropout layers where 30% units are dropped is embedded between encoding layers.

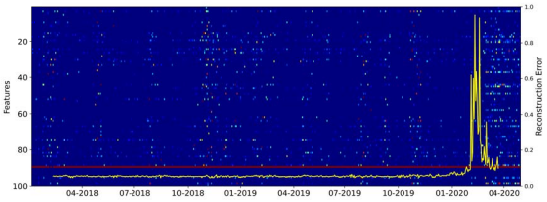


Fig. 10. The S&P 500 DE shows a delicate balance between DE and AE loss, especially in financial markets dataset. When AE is trained using data from Jan 2018 to Nov 11, 2019 and tested using data from Nov 12, 2019 to April 30, 2020 we find large reconstruction error between March and April 2020, as indicated by spike in reconstruction error. Contemporaneous DE of time series is challenging.

intraday prices from NYSE TAQ database and robust deep AE [29], where outliers are removed before DE. So instead of $DE(\boldsymbol{X})$, we will $DE(\boldsymbol{L})$, the low-dimensional manifold.

a) Estimate S&P 500 Industry Sectors dimension: S&P 500 index constituents can be divided into industry sectors such as Energy, Finance, Technology, Healthcare etc. that respond differently to economic climate. To study the effect of 2020 pandemic on different industry sectors we use SPDR Exchange Traded Funds (ETF) constituents. Because ETF's have different number of constituents we create a Euclidean distance matrix $\boldsymbol{S} \in \mathbb{R}^{60 \times 60}$, as defined in [21], where each entry is pairwise distance between returns of ETF constituents. For example, $\boldsymbol{S_{XLK,02/07/20}}$ for technology sector (XLK) consists of distances between returns of 69 XLK constituent returns between 11/12/2019 and 02/07/2020 (past 60 days) and similarly for Energy sector (XLE) $\boldsymbol{S_{XLE,02/07/20}}$ consists of 26 XLE constituent returns between 11/12/2019 and 02/07/2020.

VI. Conclusion

This paper proposes an algorithm for estimating dimension using AE. AEs are designed to reproduce input without controlling the transformations of the hidden layer. Since we were interested in using AE for estimating intrinsic dimension we refashioned innermost hidden layer values into quanti-

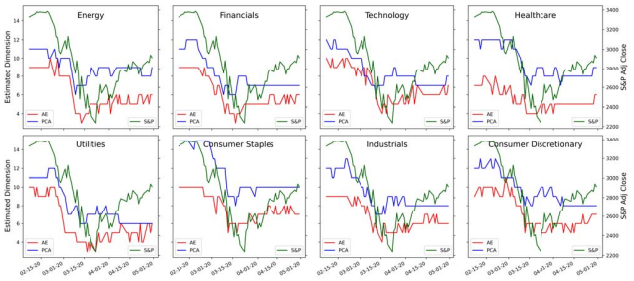


Fig. 11. AE DE of different Industrial sectors are performed using euclidean distances between SPDR ETF constituents between Feb and Apr 2020. DE uses $\boldsymbol{S} \in \mathbb{R}^{60\times60}$, which comprises distances between each day's daily returns of ETF constituents for past 60 days. While dimension of all industries slid due to pandemic, dimension of Technology and Consumer Staples appears to be recovering.

ties that are faithful to the singular values and we refer to these as singular value proxies (SVP). To facilitate DE we use large number of nodes in innermost layer and impose sparsity constraint through $\|.\|_1$ penalty to estimate intrinsic dimension. The delicate balance between sparsity and AE loss is established by using different λ values, where higher λ values enhanced sparsity but compromised AE loss. Further, to achieve consistency in DE and make the DE process independent of range of numeric values in the hidden layers we use $\|.\|_2$, which normalizes the input values into the innermost hidden layer. We test our approach on 3 different datasets. First, we successfully apply AE DE approach to a synthetic dataset that correctly detects the intrinsic dimension of 5. Unfortunately, PCA significantly overestimated(20) the dimension of the synthetic dataset. Second, we applied AE DE algorithm to MNIST digit dataset and compared the DE of AE to DE by PCA, MDS, and Isomap. Finally, we apply the algorithm to estimate dimension of financial markets using S&P 500 index constituents. Moreover, we analyze the effect of pandemic on different industrial sectors by using industry's dimension as a metric. We used the change in dimension to gauge the impact to industry/sector. While dimension slid for all industry, dimension recovery varied among industrial sectors. In the next iteration we plan to use intraday (higher frequency) trades and quotes data from NYSE. To make DE robust we hope to leverage Robust Deep Autoencoders. Subsequently, we want to apply industry/sector DE to asset allocation models.

REFERENCES

[1] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[2] G. Golub and C. Reinsch, "Singular value decomposition and least squares solutions. numerische mathematik," *Springer*, p. 18, 1970.

[3] B. Hoyle, M. M. Rau, K. Paech, C. Bonnett, S. Seitz, and J. Weller, "Anomaly detection for machine learning redshifts applied to sdss galaxies," *Monthly Notices of the Royal Astronomical Society*, vol. 452, no. 4, pp. 4183–4194, 2015.

[4] T. L. Parsons and T. Rogers, "Dimension reduction for stochastic dynamical systems forced onto a manifold by large drift: a constructive approach with examples from theoretical biology," *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 41, p. 415601, 2017.

[5] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, 2016.

[6] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo, "Text mining of news-headlines for forex market prediction: A multi-layer dimension reduction algorithm with semantics and sentiment," *Expert Systems with Applications*, vol. 42, no. 1, pp. 306–324, 2015.

[7] R. Rathnayaka, Z. Wang, D. Seneviratna, and S. Nagahawatta, "An econometric evaluation of colombo stock exchange: Evidence from arma & pca approach," in *Proceedings of the 2nd International Conference on Management and Economics*. Proceedings of the 2nd International Conference on Management and Economics, 2013, 2013, p. 10.

[8] I. Lee, "Big data: Dimensions, evolution, impacts, and challenges," *Business Horizons*, vol. 60, no. 3, pp. 293–303, 2017.

[9] X. Wang, "On the effects of dimension reduction techniques on some high-dimensional problems in finance," *Operations Research*, vol. 54, no. 6, pp. 1063–1078, 2006.

[10] C. Albanese, K. Jackson, and P. Wiberg, "Dimension reduction in the computation of value-at-risk," *The Journal of Risk Finance*, vol. 3, no. 4, pp. 41–53, 2002.

[11] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, no. 66-71, p. 13, 2009.

[12] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[13] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[14] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*. IEEE, 1994, pp. 138–142.

[15] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, "Measuring the intrinsic dimension of objective landscapes," *arXiv preprint arXiv:1804.08838*, 2018.

[16] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[17] V. Alexeev and F. Tapon, "Equity portfolio diversification: how many stocks are enough? evidence from five developed markets," *Evidence from Five Developed Markets (November 28, 2012). FIRN Research Paper*, 2012.

[18] M. Statman, "How many stocks make a diversified portfolio?" *Journal of financial and quantitative analysis*, vol. 22, no. 3, pp. 353–363, 1987.

[19] G. Y. Tang, "How efficient is naive portfolio diversification? an educational note," *Omega*, vol. 32, no. 2, pp. 155–160, 2004.

[20] J. L. Evans and S. H. Archer, "Diversification and the reduction of dispersion: An empirical analysis," *The Journal of Finance*, vol. 23, no. 5, pp. 761–767, 1968.

[21] N. Bahadur, R. Paffenroth, and K. Gajamannage, "Dimenslon estlmatlon of equlty markets," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5491–5498.

[22] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[23] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[25] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[26] E. Plaut, "From principal subspaces to principal components with linear autoencoders," *arXiv preprint arXiv:1804.10253*, 2018.

[27] J. Mira and F. Sandoval, *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7-9, 1995: Proceedings*. Springer Science & Business Media, 1995, vol. 930.

[28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[29] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 665–674.