Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Uniforming the dimensionality of data with neural networks for materials informatics

Q1 Hiroshi Ohno

Toyota Central R&D Labs., Inc., 41-1 Yokomichi, Nagakute, Aichi 480-1192, Japan

ARTICLE INFO

Article history:
Received 19 May 2015
Received in revised form 17 February 2016
Accepted 17 April 2016
Available online xxx

Keywords:
Denoising autoencoders
Neural networks
Materials informatics
Noise injection
Regularization
Regression

ABSTRACT

Materials informatics is a growing field in materials science. Materials scientists have begun to use soft computing techniques to discover novel materials. In order to apply these techniques, the descriptors (referred to as features in computer science) of a material must be selected, thereby deciding the resulting performance. As a way of describing a material, the properties of each element in the material are used directly as the features of the input variable. Depending on the number of elements in the material, the dimensionality of the input may differ. Hence, it is not possible to apply the same model to materials with different numbers of elements for tasks such as regression or discrimination. In the present paper, we present a novel method of uniforming the dimensionality of the input that allows regression or discriminative tasks to be performed using soft computing techniques. The main contribution of the proposed method is to provide a solution for uniforming the dimensionality among input vectors of different size. The proposed method is a variant of the denoising autoencoder Vincent et al. (2008) [1] using neural networks and gives a latent representation with uniformed dimensionality of the input. In the experiments of the present study, we consider compounds with ionic conductivity and hydrogen storage materials. The results of the experiments indicate that the regression tasks can be performed using the uniformed latent data learned by the proposed method. Moreover, in the clustering task using these latent data, we observed distance preservation in data space, which is also the case for the denoising autoencoder. This result may enable the proposed method to be used in a broad range of applications.

1. Introduction

Q3

The development of materials informatics has resulted in significant progress in the modeling and prediction of material properties, thereby reducing the costs of real-world experiments, and is becoming a promising research field for soft computing (for example, [2–4]). By using soft computing techniques such as neural networks, evolutionary and genetic algorithms, and fuzzy modeling, materials scientists can more effectively search for novel materials. These techniques are used alone and in combination with quantum calculations for materials design. For example, a method combining density functional theory and an evolutionary algorithm was used to predict the crystal structure of $LiBeH_3$ (Hu et al. [5]).

By organizing the data into a material database, researchers can determine the relationships between material properties (for example, conductivity, the critical temperature of superconductors, and melting temperature) and the properties (for example, atomic number, atomic mass, and electron negativity) of the elements in the material. The properties of elements or their combinations are referred to as descriptors (or "features" in computer science). Once the relevant features are obtained, the predictions of properties and the modeling of materials becomes easier. The literature [6] describes five descriptor categories: constitutional, topological, physicochemical, structural, and quantum-chemical. For instance, Seko et al. [7] adopted the sum and product of the element properties, such as atomic number, atomic mass, and number of valence electrons as features involved in the prediction of the melting temperature of single and binary compounds. These are constitutional descriptors. In addition, the use of sum and product operations is based on the domain knowledge of the compounds, and is also found in [8].

From the viewpoint of domain knowledge, we introduce two categories of materials feature representation: expert and naive. Expert representation is preferable for a material property that has a well-known mechanism or theoretical model. As such, many features (descriptors) based on the underlying theory would be expert representations (for example, Table 1 in [3]). In naive representation, we generate features based on the properties of elements in a compound, which are represented by a vector that consists of the

E-mail address: oono-h@mosk.tytlabs.co.jp

http://dx.doi.org/10.1016/j.asoc.2016.04.017
1568-4946/© 2016 Elsevier B.V. All rights reserved.

properties of elements. The advantage of the naive representation is that it is applicable to material properties with poorly understood mechanisms or theoretical models. In addition, it is simple and easy to interpret. As such, we herein adopt the naive representation. While the naive representation has good characteristics, the length of the vectors (namely, the dimensionality of the data) differs depending on the number of elements in the compound. Therefore, we cannot, for example, use the same model for compounds with different numbers of elements in modeling and prediction tasks. Thus, we propose a method of uniforming the dimensionality of input data that allows us to perform tasks using regression and discrimination methods.

Uniforming dimensionality is related to dimensionality reduction methods. Recently, a number of non-linear dimensionality reduction methods have been proposed [9,10]. These methods address the limitations of linear (traditional) methods, such as principle component analysis (PCA) and multidimensional scaling. Kernel PCA [11] and the multi-layer autoencoder [12] are well-known examples. These linear and non-linear methods cannot, however, be adopted as methods of uniforming dimensionality because when these methods are applied to datasets of different dimensionality, the resultant dimensionality of each dataset, even though they may be the same, has a different meaning. In addition, these methods focus primarily on the dimensionality reduction of data. As far as we know, there has been no previous study on non-linear uniforming of the dimensionality of data. Therefore, the present study may be the first attempt to make uniform the dimensionality of data while simultaneously considering both the expansion and reduction of the dimensionality of data. Moreover, if the data size is insufficient for learning, combining this data with data of a different dimensionality will allow the overall data to be learned.

In the neural network literature, the training algorithms of Deep Belief Networks (Hinton et al. [13], Bengio [14]) and stacked autoencoders (Vincent et al. [15]) have brought about great progress. An autoencoder consists of an encoding function, which maps the input data into a latent space, and a decoding function, which reconstructs the input data from the latent space. In the non-linear case, neural networks are often used as the encoding and decoding functions. As a regularized autoencoder, Vincent et al. [1,15,16] have proposed the denoising autoencoder, in which the input data are corrupted by Gaussian noise, whereas the target data used in learning are the original (clean) input data. Noisy inputs are used in a learning neural network to enhance generalization performance (An [17]).

For uniforming the dimensionality of input data, we propose a variant of the denoising autoencoder, in which the input data are corrupted, and an extended part added to make the dimensionality of input uniform is also injected by Gaussian noise. In the latent space formed by the encoding function, we obtain a uniformed representation with inputs of different dimensionality. Thus, we can apply the regression or discriminative tasks to the uniformed input data.

In the experiment, we first compare the proposed method with the multi-layer autoencoder, the denoising autoencoder, and the kernel PCA for synthetic data. Next, we evaluate the proposed method using compounds of four to six elements in ion-conducting bulk materials and hydrogen storage materials composed of two to five elements. We then show that regression can be performed using the uniformed input data, as well as the robustness with respect to data size and number of elements. Moreover, in a clustering task using these data and the k-nearest neighbors (k-nn) method, we find distance preservation, i.e., consistency of class assignment, in the data space, which also holds for the case using the denoising autoencoder. We evaluate the distance preservation using the difference in class assignments between the latent data in the latent representation and the original data in the input space.

The remainder of the present paper is organized as follows. In Section 2, we present background information and define the problem formulation. In Section 3, we describe the learning algorithm of the proposed method in detail. In Section 4, we conduct experiments involving a regression task on synthetic data and for the modeling of ion conductivity and hydrogen storage, and, using the uniformed input data, compare the distance preservation of the proposed method and the denoising autoencoder. In Section 5, we discuss the experimental results, related research, and future studies. Finally, Section 6 concludes the study.

2. Background and problem formulation

Descriptors (features) in materials sciences are crucial for computational materials design. In the case of the underlying theory and empirically known mechanism of material properties, the features are easily identifiable. However, it is necessary to generate the features derived from the properties of elements (for example, electron negativity, atomic number, and atomic mass). With regard to the representation of features, we refer to the former as an expert representation and the latter as a naive representation. The advantage of the naive representation is that it is applicable to the case of material properties with poorly known mechanisms or theoretical models. It is necessary to incorporate the (molecule or crystal) structural features in the representation if two materials with the same composition have different properties. In the case of isomers, the melting temperatures of C_4H_6 are $-125.7\,^\circ C$ for 1-butyne and $-32\,^\circ C$ for 2-butyne, respectively.

In the naive representation, for example, as the features of compound AB, which is composed of elements A and B, the corresponding vector v is composed from the three properties of elements A and B as follows:

$$\boldsymbol{v} = (v_{11} \quad v_{12} \quad v_{21} \quad v_{22} \quad v_{31} \quad v_{32})^T = (v_{ij}), \quad i = 1, 2, 3,\ j = 1, 2,$$

where the index i denotes the property of element, and T denotes transpose.

Index j corresponds to atom A or B. The length of the vector is the product of the number of elements in the compound and the properties of the elements. Therefore, the length of the vector differs depending on the number of elements in the compound. Thus, for compounds with different numbers of elements, we cannot use the input variables vector as a feature of the compound to perform regression or discriminative tasks. Moreover, as shown in the experiments described below, for the data on compounds having various numbers of elements, regression cannot be conducted because of a lack of data. The overall data need to be used for the task. As such, when using the overall data including all number of elements, the input variables as the features of the compounds have to be composed for the task. Therefore, it is necessary for the length of the vector to be made uniform. Note that the physical meaning of the vector changes according to the element (atomic) permutations in the vector. Thus, we sort the elements of the vector by atomic number. For example, if the atomic number A ($j = 1$) is larger than that of B ($j = 2$), then the vector $\boldsymbol{v}$ is sorted in ascending order as follows:

$$(v_{11} \quad v_{12} \quad v_{21} \quad v_{22} \quad v_{31} \quad v_{32})^T \rightarrow (v_{12} \quad v_{11} \quad v_{22} \quad v_{21} \quad v_{32} \quad v_{31})^T.$$

The problem addressed herein is to make uniform the length of the input variables vectors corresponding to compounds, as described below.

Definition 1 *(Uniforming the dimensionality of data).* The problem is defined as constructing $\boldsymbol{x}^{*(i)}$ with dimension d from data $\boldsymbol{x}^{(i)}$ with dimension d_i,

$$\boldsymbol{x}^{*(i)} \in \mathbb{R}^{d} \leftarrow \boldsymbol{x}^{(i)} \in \mathbb{R}^{d_i}, \quad i = 1, \ldots, N,$$

where N is the number of data.

In order to make uniform the length of vectors, we extend the length of each vector to the maximum length of the vectors in the data. We refer to this vector as the extended vector, which is defined as follows:

Definition 2 *(Extended vector).* $\boldsymbol{x} \in \mathbb{R}^m$ is a column vector. $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ is the extended vector corresponding to $\boldsymbol{x}(n > m)$.

$$\hat{\boldsymbol{x}} = (\boldsymbol{x}^T\ \boldsymbol{e}^T)^T,$$

where $\boldsymbol{e} \in \mathbb{R}^{n-m}$ is a vector.

In other words, the extended vector is the concatenation of $\boldsymbol{x}$ and $\boldsymbol{e}$. Then, we obtain the input data as the data of uniformed dimensionality using the encoding function learned by the denoising autoencoder. During the learning of the denoising autoencoder, the extended vector is used as the input vector in the denoising autoencoder, in which elements e_i in $\boldsymbol{e}$ are random numbers, as follows:

$$e_i \sim p(0, \sigma_i^2), \quad i = 1, \ldots, n - m,$$

where $p(0, \sigma_i^2)$ is a probability distribution with mean 0 and variance σ_i^2. Gaussian or uniform distributions can be used as p. In the present paper, we consider a Gaussian distribution, which is a natural choice for practical cases. In the denoising autoencoder, all but the extended part of the input vector is also corrupted by the Gaussian noise with the same variance of p: $\tilde{x}_i = x_i + \epsilon, \epsilon \sim N(0, \sigma_i^2)$. In the experiments described below, for simplicity, all input data are rescaled, and the same variance σ^2 is used. Next, let us define the objective function [16] of learning for a neural network to realize uniform dimensionality:

$$E = \sum_{i=1}^{N} \|\hat{\boldsymbol{x}}^{(i)} - g(f(\tilde{\boldsymbol{x}}^{(i)}))\|^2, \tag{1}$$

where $\tilde{\boldsymbol{x}}$ denotes $(\tilde{\boldsymbol{x}}^T\ \boldsymbol{e}^T)^T$, $g(\cdot)$ denotes the decoding function, and $f(\cdot)$ denotes the encoding function. The decoding and encoding functions are realized using a multi-layer neural network, whereas the decoding function in the denoising autoencoder was realized by a single-layer network [16].

Note that, unlike the denoising autoencoder [1,15,16], the target data in the objective function (1) include random noise, which corresponds to the extended part of the input vector. Due to the extended vector, even with the identity mapping, the proposed method is expected to provide a useful latent representation for regression and discriminative tasks because information from input vectors of different sizes is considered. After the learning of the functions g and f, the outputs of f yield the uniformed input data, which are a latent representation. Then, the problem of learning for the functions g and f is defined as follows:

Definition 3 *(Learning for autoencoder [16]).*

$$\{\boldsymbol{\theta}^*, \boldsymbol{\phi}^*\} = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmin}} E = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\hat{\boldsymbol{x}}^{(i)} - g(f(\tilde{\boldsymbol{x}}^{(i)}; \boldsymbol{\phi}); \boldsymbol{\theta})\|^2,$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are the parameters (weights in the neural network) of the functions g and f, respectively.

3. Learning algorithm

In order to realize decoding and encoding functions that have vector-valued outputs, we use a multi-layer neural network with three hidden layers, which is formulated as follows (for example, [18], Section 5):

$$o_k = h\left(\sum_{j=1}^{l_1+1} w_{kj}^{H_1} \cdot h\left(\sum_{i=1}^{I+1} w_{ji}^{I} \cdot I_i^{(t)}\right)\right),$$

$$O_l^{(t)} = \sum_{j=1}^{l_1+1} w_{lj}^{O} \cdot h\left(\sum_{k=1}^{l_2+1} w_{jk}^{H_2} \cdot o_k\right), \quad l = 1, \ldots, I, t = 1, \ldots, N,$$

where o_k denotes the kth output of the encoding function $f(\cdot)$, $O_l^{(t)}$ denotes the lth output of the network for the tth data, I denotes the number of inputs, $I_i^{(t)}$ denotes the ith input data for the tth data, $h(\cdot)$ is the sigmoid function, $h(x) = 1/(1 + exp(-x))$. Thus, we have $\boldsymbol{\theta} = (\{w_{lj}^{O}\}, \{w_{jk}^{H_2}\})$ and $\boldsymbol{\phi} = (\{w_{kj}^{H_1}\}, \{w_{ji}^{I}\})$. The outputs o_k in the second layer correspond to the uniformed input data. According to Definition 3, the objective function (1) is optimized by the stochastic gradient descent [19,15] with the backpropagation algorithm [1].

Next, we present the overall algorithm for uniforming the dimensionality of data (Algorithm 1). In Algorithm 1, ε is the parameter for deciding the termination of learning and σ^2 controls the convergence and stability of the algorithm. The larger the value of σ^2 becomes, the worse the convergence and stability of the algorithm becomes. Thus, in the experiments, we determine the value of σ^2 by trial and error.

Algorithm 1. Learning algorithm for uniforming the dimensionality of data.

```
Input: max iterations $T$, error threshold $\varepsilon$, noise variance $\sigma^2$, training data
    $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$, test data $\{\boldsymbol{x}'^{(i)}\}_{i=1}^{N'}$.
 1: Initialize the parameters (weights), $\boldsymbol{W} = (\boldsymbol{\theta}, \boldsymbol{\phi})$.
 2: for $k = 1$ to $T$ do
 3:     Select $\boldsymbol{x}$ from $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ in a uniformly random manner.
 4:     Generate a random vector $\boldsymbol{v} \sim \mathcal{N}(0, \sigma^2)$.
 5:                     ▷ $|\boldsymbol{v}| = \max\{|\boldsymbol{x}^{(i)}|, i = 1, \cdots, N\} - |\boldsymbol{x}|$.
 6:                              ▷ if $|\boldsymbol{v}| = 0$ then $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}$, go to 9.
 7:     Set $\boldsymbol{v}$ to the extended part of input vector $\hat{\boldsymbol{x}}$.
 8:                              ▷ $\hat{\boldsymbol{x}}$ is the extended input vector of $\boldsymbol{x}$.
 9:     $\boldsymbol{t} \leftarrow \hat{\boldsymbol{x}}$                  ▷ $\boldsymbol{t}$ denotes the target (output) vector.
10:     Generate a random vector $\boldsymbol{w} \sim \mathcal{N}(0, \sigma^2)$.
11:     $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x} + \boldsymbol{w}$        ▷ The unextended part of $\hat{\boldsymbol{x}}$ is corrupted by noise $\boldsymbol{w}$.
12:     Update $\boldsymbol{W}$.                    ▷ Use the backpropagation algorithm.
13:     Calculate RMSE $e$ for $\{\boldsymbol{x}'^{(i)}\}_{i=1}^{N'}$.
14:                           ▷ RMSE denotes the root mean squared error.
15:     Check convergence using $e$ and $\varepsilon$.          ▷ Terminate the learning.
16: end for
17: return $\boldsymbol{W}$
```

After the learning of the neural network, we generate data with uniformed dimensionality through the encoding function f. After a certain number of iterations, while injecting noise into the extended part of the data, the outputs of the encoding function, which are the outputs in the second hidden layer of the neural network, are averaged. This procedure is described in Algorithm 2.

[1] The computational cost was very large for the five-layer neural network used in the experiments. Thus, we used the stochastic gradient descent and the momentum term in the backpropagation algorithm because the convergence instability was small in the pretraining experiments and the algorithm was simple. We used 0.9 for the value of the momentum term throughout the experiments.

Algorithm 2. Generating uniformed data by uniforming the dimensionality of data.

```
Input: max iterations S, noise variance $\sigma^2$, data $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$.
1:  Initialize the parameters (weights) of the encoding function using $\boldsymbol{\phi}^*$
    obtained from Algorithm 1.
2:  $\boldsymbol{s}=\boldsymbol{0}$
3:  for $i=1$ to $S$ do
4:      Generate a random vector $\boldsymbol{w}\sim\mathcal{N}(0,\sigma^2)$.
5:      $\hat{\boldsymbol{x}}\leftarrow\boldsymbol{x}+\boldsymbol{w}$
6:      $\boldsymbol{s}\leftarrow\boldsymbol{s}+f(\hat{\boldsymbol{x}};\boldsymbol{\phi}^*)$
7:  end for
8:  $\boldsymbol{s}\leftarrow\boldsymbol{s}/S$
9:  return $\boldsymbol{s}$
```

4. Experiments and results

Here, we evaluate the proposed method, which is presented in Algorithms 1 and 2, using a linear regression task with synthetic data. First, we compare the proposed method with the conventional methods of the multi-layer autoencoder, the denoising autoencoder, and kernel PCA. As a specific application, we then compare the proposed method with the denoising autoencoder using data on the ion conductivity of bulk material. In addition, to evaluate the robustness of the proposed method, we considered the dataset of the hydrogen storage materials. In this case, the data size was 586 and the number of constituent elements was varied from two to five, while for the ion conductivity of bulk material, the size of the dataset was 76 and the number of constituent elements was varied from four to six.

4.1. Synthetic data

Data preparation: In order to generate synthetic data, we drew random samples from a multivariate normal distribution $N(\boldsymbol{0}, \Omega)$, which consisted of twenty elements. The diagonal elements in Ω were set to uniform random numbers $\in[0, 1]$, and the non-diagonal elements were set to uniform random numbers $\in[0, 0.1]$. We collected three hundred 10-dimensional vectors (the first half of 20-dimensional vectors) and three hundred 20-dimensional vectors, and prepared five datasets of 10- and 20-dimensional vectors. In order to generate the output data, we prepared the linear model as follows. The coefficients of the linear model were set to uniform random numbers $\in[0, 1]$ from the identity distribution. The outputs were calculated by multiplying the coefficients by the 10- and 20-dimensional vectors (as the input vector) with injected random noise $N(0, 1)$.

For the 10- and 20-dimensional vector data, we separated the data evenly into training data and test data.

Experimental setup: We used a five-layer neural network as a multi-layer neural network. The number of outputs in the first hidden layer is equal to that in the third hidden layer, which is denoted l_1. Since the network realizes an autoencoder, the number of outputs in the network is equal to the number of inputs in the network (i.e., the dimensionality of the input data). The number of outputs in the second hidden layer corresponds to the uniformed dimensionality of the input data (i.e., the length of the input vector in the linear model), which is denoted l_2. All input data were rescaled to the range [0, 1]. The number of inputs I of the network was 20, and l_1 was set to 40. The parameter values in Algorithm 1 were set as

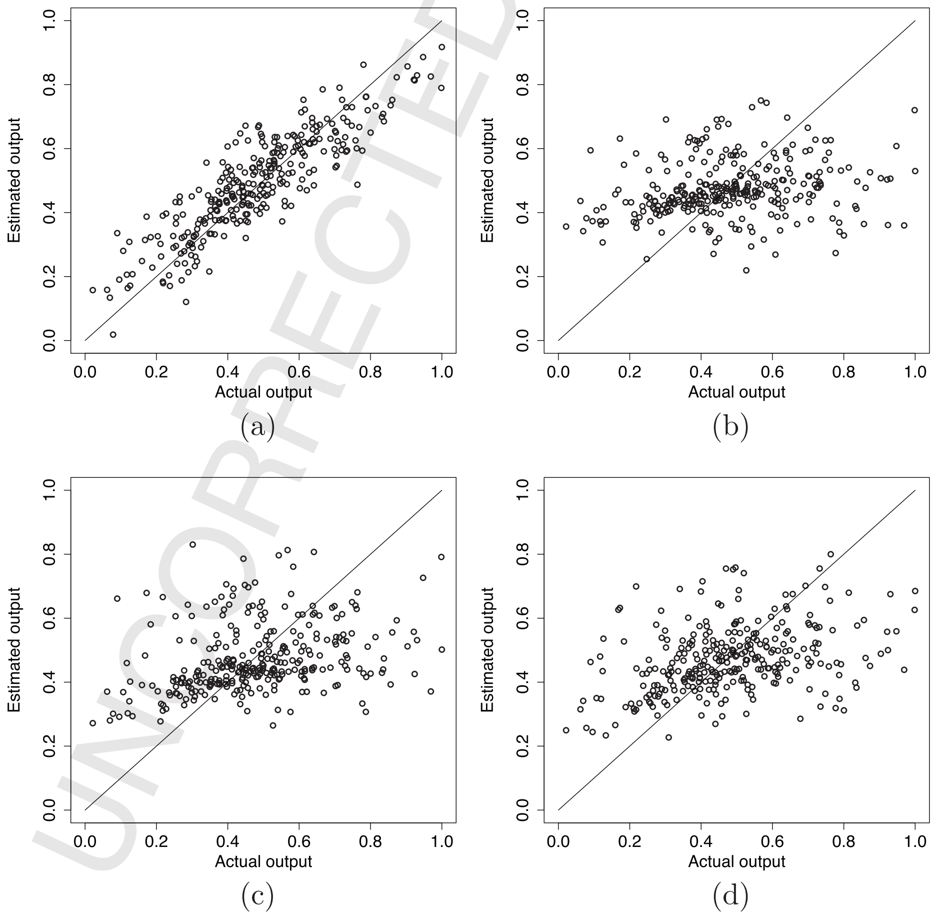

Fig. 1. Results of the linear regression task of trial 5 using (a) the proposed method, (b) the multi-layer autoencoder, (c) the denoising autoencoder, and (d) kernel PCA. The *X*-axis denotes the actual output. The *Y*-axis denotes the estimated output of the linear model.

Table 1
Comparison of generalization results for the test datasets on the linear regression task.

Method	RMSE	Correlation
Proposed method	0.0872 ± 0.0045	0.862 ± 0.0142
Multi-layer autoencoder	0.168 ± 0.0128	0.251 ± 0.146
Denoising autoencoder	0.162 ± 0.0124	0.344 ± 0.120
Kernel PCA	0.148 ± 0.0241	0.476 ± 0.173

follows: $T = 5{,}000{,}000$, $\epsilon = 10^{-5}$, $\sigma^2 = 0.01$. In the learning with the training data, we varied l_2 in $\{5, 10, 15, 20\}$, and l_2 was 20 at the minimum root mean squared error (RMSE) for the test data. After the learning, we generated the uniformed (latent) data using Algorithm 2. Here, S was set to 100, and σ^2 was set to 0.01, which was the value used in the learning for the network. Thus, we obtained the uniformed data for the training and test data, where the length of the vector was 20.

A multi-layer autoencoder and denoising autoencoder were also realized using a five-layer neural network. The parameters were set to the values used in the proposed method. In kernel PCA, the Gaussian radial basis function kernel was used. The kernel width was set to the median distance between data points [20].

The training data with the 10-dimensional vectors were used for the learning of the multi-layer autoencoder, the denoising autoencoder, and kernel PCA. Then, for the each method, the transformed 20-dimensional vector data were obtained using both the training data and the test data with 10-dimensional vectors. Note that these methods were used as the expansion of the dimensionality of data.

Next, we applied the linear regression task in order to evaluate the generalization performance of the linear model. In order to compare the generalization performance of the proposed method with that of conventional methods, the test data (300 data points) were evaluated using a linear model. In the learning for the linear model, we used the uniformed data corresponding to the training data (300 data points). The test data on the evaluation were the uniformed data corresponding to the test data. For the conventional methods, the training data on the linear regression task consisted of the transformed data corresponding to the training data with 10-dimensional vectors and the training data with 20-dimensional vectors. The test data were also prepared in the same manner.

Results: For the test data, we evaluated the generalization performance five times in order to remove sampling bias. The generalization performance measured by RMSE and the correlation is summarized in Table 5. The RMSE and the correlation were averaged over five trials. The table entries show the mean and the standard deviation. The proposed method outperformed the conventional methods because the expansion and reduction of the dimensions of data were performed simultaneously under the learning process of uniforming the dimensionality. In addition, the results of the linear regression task of trial 5 are shown in Fig. 1 (Table 1).

4.2. Ion conductivity data

In order to confirm the usefulness of the proposed method for a regression task and for comparison with the denoising autoencoder regarding distance preservation, we used ion conductivity data for bulk materials from the literature. The dataset of ion conductivity is shown in Table 2. The number of data was 76. The number of data for each number of elements in the materials is also shown in Table 3.

Data representation: For the naive representation, we used the following properties of the elements in the compounds as features

Table 2
Dataset of ion conductivity.

No.	Compound	σ (S/cm)
1	Li0.34La0.51TiO2.94	1.00E−03
2	Li0.27La0.59TiO3	6.80E−04
3	Li0.10La0.63TiO3	7.90E−05
4	(Li0.1La0.5)0.9Sr0.1TiO3	1.50E−03
5	Li0.15La0.51Sr0.15TiO3	5.30E−05
6	Li0.25La0.41Sr0.25TiO3	7.60E−05
7	(Li0.1La0.63)(Mg0.5W0.5)O3	1.00E−06
8	Li0.38La0.5Na0.13TiO3	2.00E−05
9	Li0.5(La0.4Nd0.1)TiO3	1.00E−03
10	Li0.245La0.592Ti0.98Mn0.02O3	1.00E−03
11	La0.58Li0.36Ti0.95Mg0.05O3	2.10E−04
12	La0.56Li0.36Ti0.95Al0.05O3	6.40E−04
13	La0.55Li0.36Ti0.95Mn0.05O3	1.90E−04
14	La0.55Li0.36Ti0.95Ge0.05O3	3.60E−04
15	La0.55Li0.36Ti0.95Ru0.05O3	5.20E−05
16	La0.51Li0.36Ti0.95W0.05O3	7.30E−04
17	La0.54Li0.36TiO3	8.90E−04
18	La0.55Li0.36Ti0.995Al0.005O3	1.10E−03
19	Li0.067La0.64TiO2.99	7.90E−05
20	Li0.06La0.66Ti0.93Al0.06O3	1.70E−06
21	Li0.10La0.66Ti0.90Al0.10O3	7.30E−06
22	Li0.15La0.66Ti0.85Al0.15O3	9.60E−06
23	Li0.20La0.66Ti0.80Al0.20O3	4.30E−05
24	Li0.25La0.66Ti0.75Al0.25O3	7.70E−05
25	Li0.30La0.66Ti0.70Al0.30O3	1.70E−05
26	La0.32Li0.03NbO3	4.06E−06
27	La0.31Li0.06NbO3	2.33E−05
28	La0.3Li0.09NbO3	3.52E−05
29	La0.29Li0.12NbO3	4.25E−05
30	La0.28Li0.15NbO3	3.85E−05
31	La0.27Li0.18NbO3	3.82E−05
32	La0.59Na0.12TiO3	1.00E−07
33	La0.53Na0.21TiO3	1.00E−07
34	Li0.25La0.25TaO3	1.40E−03
35	La0.25Na0.2Li0.05NbO3	2.10E−05
36	La0.2Na0.25Li0.15NbO3	5.30E−06
37	La0.15Na0.3Li0.25NbO3	2.90E−07
38	La0.25Ag0.2Li0.05NbO3	3.90E−05
39	La0.2Ag0.25Li0.15NbO3	2.00E−05
40	La0.15Ag0.3Li0.25NbO3	2.90E−07
41	Li0.34Pr0.56TiO3	1.00E−06
42	Sm0.52Li0.34TiO3	1.00E−07
43	Nd0.55Li0.34TiO3	1.00E−07
44	Nd0.25Li0.25TaO3	4.00E−06
45	Sm0.25Li0.25TaO3	3.90E−07
46	Y0.25Li0.25TaO3	5.00E−09
47	Li0.5Sr0.56Fe0.25Ta0.75O3	1.00E−04
48	Li0.5Sr0.56Cr0.25Ta0.75O3	6.00E−05
49	Li0.33Sr0.56Cr0.225Ta0.775O3	1.00E−04
50	Li0.33Sr0.56Co0.225Ta0.775O3	5.10E−06
51	Li0.33Sr0.56Ga0.225Ta0.775O3	7.70E−06
52	Li0.33Ca0.56Fe0.225Ta0.775O3	1.50E−06
53	Li0.33(Ca0.8Sr0.2)0.56Fe0.225Ta0.775O3	5.80E−07
54	Li0.33(Ca0.5Sr0.5)0.56Fe0.225Ta0.775O3	4.10E−05
55	Li0.33(Ca0.2Sr0.8)0.56Fe0.225Ta0.775O3	9.80E−05
56	Li0.33(Ca0.1Sr0.9)0.56Fe0.225Ta0.775O3	1.30E−04
57	Li0.33Sr0.56Fe0.225Ta0.775O3	8.50E−05
58	LiCaTiNbO6	1.00E−07
59	LiSrTiNbO6	1.00E−06
60	LiSrTiTaO6	5.50E−04
61	LiSr2Ti2NbO9	1.00E−06
62	LiBa2Ti2NbO9	1.00E−07
63	LiSr2Ti2TaO9	3.20E−05
64	LiCa1.65Ti1.3Nb1.7O9	1.00E−07
65	LiCa1.65Ti1.3Ta1.7O9	1.00E−07
66	LiSr1.65Ti2.15W0.85O9	1.00E−06
67	LiSr1.65Ti1.3Nb1.7O9	2.00E−05
68	LiSr1.65Ti1.3Ta1.7O9	4.90E−05
69	LiSr1.65Zr1.3Ta1.7O9	1.30E−05
70	Li0.1Sr0.8Ti0.7Nb0.3O3	1.00E−07
71	Li0.3Sr0.6Ti0.5Nb0.5O3	5.40E−06
72	Li0.3Sr0.6Ti0.5Ta0.5O3	1.70E−04
73	Li0.3Sr0.6Ti0.45Fe0.05Ta0.5O3	6.00E−05
74	Li0.3Sr0.6Ti0.40Fe0.10Ta0.5O3	3.60E−05
75	Li0.3Sr0.6Ti0.35Fe0.15Ta0.5O3	2.80E−05
76	Li0.3Sr0.6Ti0.20Fe0.30Ta0.5O3	1.00E−07

(No. 1 ∼ 34 and No. 41 ∼ 76 from Tables 2 and 5 in [21], and No. 35 ∼ 40 from Table 1 in [22].)

Table 3
Number of data for each number of elements in the dataset of ion conductivity.

Number of elements	Number of data
4	20
5	48
6	8

Table 4
Evaluation results for validation data of ion conductivity. The root mean squared error (RMSE) was averaged over five trials. The minimum RMSE was given by $l_1 = 120$ and $l_2 = 60$.

Combination of values for the parameters	RMSE
$l_1 = 30, l_2 = 15$	0.04791 ± 0.003416
$l_1 = 30, l_2 = 30$	0.04755 ± 0.002844
$l_1 = 30, l_2 = 60$	0.04692 ± 0.002667
$l_1 = 30, l_2 = 90$	0.04697 ± 0.003382
$l_1 = 60, l_2 = 15$	0.04782 ± 0.003336
$l_1 = 60, l_2 = 30$	0.04551 ± 0.002903
$l_1 = 60, l_2 = 60$	0.04499 ± 0.003266
$l_1 = 60, l_2 = 90$	0.04553 ± 0.003039
$l_1 = 120, l_2 = 15$	0.04804 ± 0.003185
$l_1 = 120, l_2 = 30$	0.04498 ± 0.003510
$l_1 = 120, l_2 = 60$	0.04342 ± 0.003403
$l_1 = 120, l_2 = 90$	0.04441 ± 0.004237

of the input data for the linear regression task: electron negativity, atomic number, first ionization potential, atomic mass, group of the periodic table, period of the periodic table, single covalent radius, double covalent radius, electron affinity, and composition ratio of the compound. Ten features were used. Thus, the input data are represented by the vector $\boldsymbol{x}$, as follows:

$$\boldsymbol{x} = (x_{1,1}, x_{1,2}, \ldots, x_{10,1}, x_{10,2}, \cdots)^T = (x_{i,j}), \quad i = 1, \ldots, 10,$$

$$j = 1, \ldots, \{4, 5, 6\},$$

where the indices i and j denote the feature number and the element number, respectively. In other words, the length of the vector is varied from 40 to 60. Hence, we cannot apply the linear regression task for all compounds with different numbers of elements. In addition, it is necessary to use the overall data on the linear regression task because the number of data for each element is smaller than the number of regressor variables, as shown in Table 3.

Experimental setup: We used a five-layer neural network as a multi-layer neural network. The network architecture was the same as that used for the synthetic data. The number of inputs I of the network was 60. In the learning, we varied l_1 in {30, 60, 120} and l_2 in {15, 30, 60, 90}. For deciding these parameter values, we used five-fold cross-validation with five trials in order to remove the random effects of fold assignments because of the small data size. All input data were rescaled to the range [0, 1]. The parameter values in Algorithm 1 were set as follows: $T = 5,000,000$, $\epsilon = 10^{-5}$, $\sigma^2 = 0.01$. After the learning, we generated the uniformed data using Algorithm 2. Here, S was set to 100, and σ^2 was set to 0.01, which is the value used in the learning.

Results: Table 4 shows the evaluation results for the validation data. The first column lists the combination of values for l_1 and l_2. The second column lists the root mean square error (RMSE) averaged over five trials (with different random seeds) and its standard deviation. As shown in the table, we adopted $l_1 = 120$ and $l_2 = 60$, which were the combinations of values at the minimum RMSE.

Next, we applied the linear regression task to the ion conductivity data as the output and the uniformed data as the input variables, in which the length of the vector was 60. The linear model described the natural logarithm of the ion conductivity in terms of 60 input variables. The result of the linear regression task is shown in Fig. 2. The linear model had an r^2 value of 0.918.

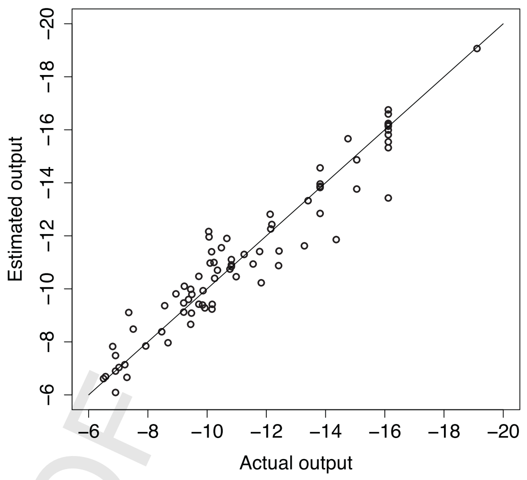

Fig. 2. Results of a regression task performed using the uniformed data as the input for ion conductivity. The X-axis denotes the actual output, and the Y-axis denotes the estimated output of the linear model.

In order to compare the quality of the obtained latent representation by the proposed method with that of the denoising autoencoder, the distance preservation was evaluated. Using the k-nn method to evaluate the distance preservation, we quantified the difference of class assignments between the original data on the input space and the latent data on the latent representation. In general, each dataset could be assigned a different name (or number) as a class derived by the k-nn method. Therefore, we needed to assign the same class name (or number) for two class assignment data. Thus, we used permutation to reassign the data class. The evaluation value E of the distance preservation with n classes was defined as follows:

$$E(D_x, D) = \max_{\sigma \in S_n} \frac{1}{|D|} \sum_{i=1}^{|D|} I((\sigma \circ D_x)_i, D_i), \tag{420}$$

where σ denotes the permutation in the permutation group S_n of degree n, n is the number of classes, D_i denotes the ith data element of D, and $I(\cdot)$ is the indicator function. The operator $\circ$ means that the permutation σ applies to the data D_x, D and D_x denote the class assigned datasets of the original dataset and the latent dataset, respectively, and E denotes the same class assignment ratio (henceforth, matching rate) between the original dataset and the latent dataset.

In the evaluation, we used the data of five elements having the maximum number of data (see Table 3). The denoising autoencoder was applied to these data, where the latent data, in which the length of the vector was equal to that of the proposed method, was derived. Fig. 3 shows the results of the matching rate E, which was averaged over five trials. The error bars correspond to one standard deviation.

The figure indicates that the proposed method (solid line) and the denoising autoencoder (dashed line) do not have remarkably different performances.

4.3. Hydrogen storage data

In this section, we evaluate the robustness of the proposed method. Thus, we consider a dataset in which both the data size and the number of elements are larger than those of the ion conductivity data. The dataset of hydrogen storage materials [2] we prepared had a size of 586. The number of data for each number of the elements

[2] The dataset was obtained from http://hydrogenmaterialssearch.govtools.us.

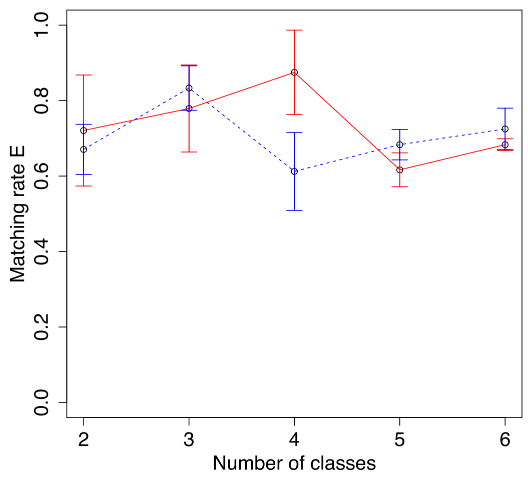

Fig. 3. Results of evaluation of distance preservation for the proposed method (solid) and the denoising autoencoder (dashed). The matching rate *E* denotes the same class assignment ratio between the original dataset and the latent dataset. The *X*-axis denotes the number of classes. The error bars show the standard deviation.

Table 5
Number of data for each number of elements in the dataset of hydrogen storage materials.

Number of elements	Number of data
2	245
3	250
4	65
5	26

in the materials is also shown in Table 5. The output in regression is the hydrogen weight percentage.

Data representation: We also used the naive representation. The number of the features was ten. Thus, the input data are represented by the vector $\boldsymbol{x}$, as follows:

$$\boldsymbol{x} = (x_{1,1}, x_{1,2}, \ldots, x_{10,1}, x_{10,2}, \cdots)^T = (x_{i,j}), \quad i = 1, \ldots, 10,\ j = 1, \cdots, \{2, 3, 4, 5\},$$

where the indices *i* and *j* denote the feature number and the element number, respectively. The length of the vector was varied from 20 to 50. The number of data for each number of elements is shown in Table 5.

Experimental setup: We also used a five-layer neural network as a multi-layer neural network. The number of inputs *I* of the network was 50. In the learning, we varied l_1 in {25, 50, 100} and l_2 in {25, 50, 75, 100}. We used five-fold cross-validation with five trials. All input data were rescaled to the range [0, 1]. The parameter values in Algorithm 1 were set as follows: T = 5, 000, 000, $\epsilon = 10^{-5}$, $\sigma^2 = 0.01$. In Algorithm 2, *S* was set to 100, and σ^2 was set to 0.01.

Table 6
Evaluation results for validation data of hydrogen storage materials. The root mean squared error (RMSE) was averaged over five trials.

Combination of values for the parameters	RMSE
$l_1 = 25, l_2 = 25$	0.06256 ± 0.001449
$l_1 = 25, l_2 = 50$	0.06130 ± 0.001389
$l_1 = 25, l_2 = 75$	0.06126 ± 0.001512
$l_1 = 25, l_2 = 100$	0.06001 ± 0.001599
$l_1 = 50, l_2 = 25$	0.05818 ± 0.001721
$l_1 = 50, l_2 = 50$	0.05641 ± 0.001493
$l_1 = 50, l_2 = 75$	0.05583 ± 0.001441
$l_1 = 50, l_2 = 100$	0.05590 ± 0.002180
$l_1 = 100, l_2 = 25$	0.05436 ± 0.002135
$l_1 = 100, l_2 = 50$	0.05255 ± 0.001914
$l_1 = 100, l_2 = 75$	0.05196 ± 0.001659
$l_1 = 100, l_2 = 100$	0.05195 ± 0.001987

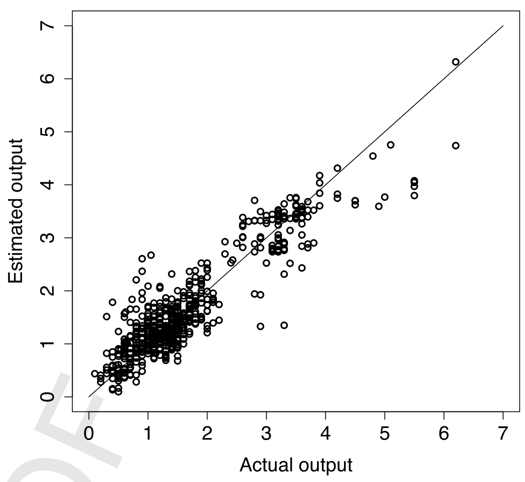

Fig. 4. Results of a regression task performed using the uniformed data as the input for hydrogen storage materials. The *X*-axis denotes the actual output, and the *Y*-axis denotes the estimated output of the linear model.

Results: Table 6 shows the evaluation results for the validation data. The first column lists the combination of values for l_1 and l_2. The second column lists the root mean square error (RMSE) averaged over five trials and it's standard deviation. As shown in the table, we adopted $l_1 = 100$ and $l_2 = 75$, which were the combinations of values with the lower RMSE and the smaller network size. The value of the RMSE was not so large compared to that of the ion conductivity.

Next, we applied the linear regression task to the hydrogen weight percentage as the output and the uniformed data as the input variables, where the length of the vector was 75. The result of the linear regression task is shown in Fig. 4. The linear model had an r^2 value of 0.838. Thus, a reasonably good result was obtained, which was also the case for the ion conductivity. The proposed method worked well for the larger data size and the larger range of the number of elements, demonstrating its robustness.

Based on the experiments on the synthetic data and the materials data, we conclude that the proposed method is useful for uniforming the dimensionality of data.

5. Discussion

In this section, we discuss the experimental results, related research, including, for example, overload learning [23] and the denoising autoencoder [1,15,16], and areas for future study.

In the case of the synthetic data, the proposed method outperformed the conventional methods. This implies the feasibility of performing the expansion and reduction simultaneously. The advantage of the proposed method can be exploited by using the information contained in input vectors of different sizes, whereas conventional methods use only input vectors of the same size.

In both the materials datasets, i.e., the ion conductivity and hydrogen storage datasets, we confirmed the robustness of the proposed method. Regardless of the data size, the difference in RMSE between the ion conductivity and hydrogen storage datasets was small (see Tables 4 and 6). Furthermore, in Tables 4 and 6, a slight increase in the standard deviation was observed according to the increase in the network size. However, the learning of the network was still stable while varying the number of elements in the materials because there was no extremely high RMSE with respect to either the mean or the standard deviation. This could be because of the regularization induced by noise injection. Although the r^2 values of regression were high, there was some spread of points and outliers in the scatter plots, as shown in Figs. 2 and 4. With regard to this matter, we propose a method for improving the performance of regression.

The key point of the proposed method is that the noise is injected into the extended part of the vector in the input variables. The effect of the injected noise appears to be similar to the overload task in learning for a neural network [23]. The overload task is used to add an additional unit to the input layer of a neural network. In the learning, the additional unit is set to a random number. The injected noise corresponds to the additional task in overload learning. According to [23], the generalization performance was improved as compared to the overload task in learning. Therefore, the injected noise could lead to the acquisition of an available latent representation. The effectiveness of the overload was shown to work well empirically [23]. The theoretical analysis has, however, been left as an open problem.

Moreover, during learning, noise is added to a part of the vector other than the extended part, as in the denoising autoencoder. This leads to the regularization avoiding over-fitting [24]. In the experiments, when the number of parameters (weights) was larger than the number of data, the generalization performance did not deteriorate remarkably (see Table 4). For the theoretical analysis, Bishop [25] showed that the regularization term in the error function when learning with noise belongs to the class of generalized Tikhonov regularization. Grandvalet et al. [26] demonstrated that when the injected noise is Gaussian, the noise injection can be a stochastic alternative to regularization of the error function.

Hence, the proposed method has the advantages of both the overload learning and the denoising autoencoder. However, the noise injection generates numerous distinguishable training patterns in the learning. Therefore, the proposed method has a low convergence speed [27], which is caused by not only the backpropagation, but also the noise injection. Thus, in the future, we hope to investigate an efficient learning algorithm for complicated networks, such as a five-layer neural network.

For high-dimensional data, it is important for data analysis to reduce the dimensionality for visualization on the 2- or 3-D map. This is also important for learning in order to reduce computational complexity. In the experiments, we used the matching rate of class assignment using the k-nn method to evaluate the distance preservation. Then, the matching rate was somewhat low for both the proposed method and the denoising autoencoder. Distance preservation is also related to the preservation of topological information of data. Therefore, the matching rate should be even higher. As pointed out in [28], the main issue for the latent representation with dimensionality reduction is to consider how the low-dimensionality latent representation preserves the distances between the original data. Accordingly, in order to improve the distance preservation, it would be necessary to introduce a stress function that evaluates the quality of the matching between the distances in the latent representation and the data spaces [28].

One reason why the generalization performance in the linear regression task was somewhat low is the lack of output directivity of the latent data. In other words, the latent data are learned regardless of the output data in the linear regression task. Bengio et al. [29] demonstrated that the autoencoder trained to minimize reconstruction cross-entropy has become a latent representation, which is useful for the output in discriminative tasks. Snoek et al. [30] proposed a nonparametrically guided autoencoder using a Gaussian process to guide the latent representation. In these studies, the objective function includes a guided term with respect to the output in the data. Therefore, we propose the objective function considering a linear regression task as follows (*cf.* [29,31,32]):

$$E = \sum_{i=1}^{N} \|\hat{\boldsymbol{x}}^{(i)} - g(f(\tilde{\boldsymbol{x}}^{(i)}))\|^2 + \sum_{i=1}^{N} (y^i - \beta^T f(\tilde{\boldsymbol{x}}^{(i)}))^2,$$

where β denotes the regression coefficients that are estimated simultaneously during learning of the functions g and f, and y^i denotes the ith output data of the data. Here, we refer to the autoencoder with this objective function as a linear guided autoencoder, or LGAE. The latent data in the LGAE provides a suitable representation with respect to the linear regression task. The latent data obtained by the proposed method using this objective function could improve the performance of linear regression tasks.

Finally, the proposed method is application-independent. By modifying the data representation in the input vector according to the application, the proposed method can be extended to other problems.

6. Conclusion

In the present paper, we proposed a uniforming method for the dimensionality of data using a neural network for materials informatics, in which different numbers of elements in compounds are often dealt with, in order to predict the properties of materials or to classify compounds. In addition, we considered the case of material properties for which the underlying mechanisms or theoretical models are poorly understood. Therefore, we introduced a naive representation for the compound. The length of the vector in the naive representation was varied depending on the number of elements. Hence, uniformed dimensionality of the input data was necessary. Unlike conventional methods, such as the multilayer autoencoder, the denoising autoencoder, and kernel PCA, the proposed method can make uniform the dimensionality of data while simultaneously considering the expansion and reduction of the dimensionality.

In the proposed method, uniforming of dimensionality is realized by noise injection into the extended part of the input vector during learning for the autoencoder, which is a variant of the denoising autoencoder [1,15,16]. The latent representation in the neural network becomes a uniformed representation of the input data.

Experiments on synthetic data, ion conductivity data, and hydrogen storage materials data revealed that the proposed method works well for the linear regression task, as compared to the conventional methods, and exhibits distance preservation and robustness. The results may enable us to apply the proposed method to a broad range of applications.

Further study is necessary in order to investigate the statistical properties of the uniformed representation of data. Furthermore, in order to improve the performance of the linear regression task, we proposed the linear guided autoencoder, which has a linear guided term in the objective function.

In the future, we hope to confirm the effectiveness of the linear guided autoencoder for use in materials informatics.

Acknowledgments

The author would like to thank Satoshi Koide for useful comments on an earlier version of the manuscript and for engaging in useful discussions of the present study.

References

[1] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ICML'08, ACM, 2008, pp. 1096–1103.
[2] P. Ghanshyam, W. Chenchen, J. Xun, R. Sanguthevar, R. Ramamurthy, Accelerating materials property predictions using machine learning, Sci. Rep. 3 (2013) 2810+.
[3] S. Curtarolo, G.L.W. Hart, M.B. Nardelli, N. Mingo, S. Sanvito, O. Levy, The high-throughput highway to computational materials design, Nat. Mater. 12 (3) (2013) 191–201.

[4] M. Fornari, Computational materials discovery goes platinum, Physics 6 (2013) 140.
[5] C.-H. Hu, A.R. Oganov, Y.M. Wang, H.Y. Zhou, A. Lyakhov, J. Hafner, Crystal structure prediction of $LiBeH_3$ using *ab initio* total-energy calculations and evolutionary simulations, J. Chem. Phys. 129 (23) (2008) 234105.
[6] T. Le, V.C. Epa, F.R. Burden, D.A. Winkler, Quantitative structure–property relationship modeling of diverse materials properties, Chem. Rev. 112 (2012) 2889–2919.
[7] A. Seko, T. Maekawa, K. Tsuda, I. Tanaka, Machine learning with systematic density-functional theory calculations: application to melting temperatures of single- and binary-component solids, Phys. Rev. B 89 (2014) 054303.
[8] K.T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K.R. Müller, E.K.U. Gross, How to represent crystal structures for machine learning: towards fast prediction of electronic properties, Phys. Rev. B 89 (2014) 205118.
[9] C.J.C. Burges, Dimension reduction: a guided tour, Found. Trends Mach. Learn. 2 (4) (2010) 275–365.
[10] L. Van der Maaten, E. Postma, H. Van den Herik, Dimensionality reduction: a comparative review, Technical Report TiCC TR 2009-005, 2009.
[11] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[12] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[13] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[14] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[15] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408.
[16] P. Vincent, A connection between score matching and denoising autoencoders, Neural Comput. 23 (7) (2011) 1661–1674.
[17] G. An, The effects of adding noise during backpropagation training on a generalization performance, Neural Comput. 8 (3) (1996) 643–674.
[18] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[19] L. Bottou, Stochastic gradient learning in neural networks, in: Proceedings of Neuro-Nîmes 91, EC2, Nimes, France, 1991.
[20] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, J. Mach. Learn. Res. 13 (2012) 723–773.
[21] S. Stramare, V. Thangadurai, W. Weppner, Lithium lanthanum titanates: a review, Chem. Mater. 15 (21) (2003) 3974–3990.
[22] Y.I. Tetsuhiro Katsumata, M. Itoh, New perovskite-type lithium ion conductors, LaxMyLi1-3x-yNbO3 (M = Ag and Na), Solid State Ionics 113–115 (1998) 465–469.
[23] I. Noda, Learning method by overload, in: IJCNN'93-Nagoya, 1993, pp. 1357–1360.
[24] R. Salah, G. Xavier, B. Yoshua, V. Pascal, Adding noise to the input of a model trained with a regularized objective, CoRR abs/1104.3250, 2011.
[25] C.M. Bishop, Training with noise is equivalent to Tikhonov regularization, Neural Comput. 7 (1) (1995) 108–116.
[26] Y. Grandvalet, S. Canu, S. Boucheron, Noise injection: theoretical prospects, Neural Comput. 9 (5) (1997) 1093–1108.
[27] M. Chen, K.Q. Winberger, Z.E. Xu, F. Sha, Marginalizing stacked linear denoising autoencoders, J. Mach. Learn. Res. 16 (2015) 3849–3875.
[28] N.D. Lawrence, J. Quiñonero Candela, Local distance preservation in the GP-LVM through back constraints, in: Proceedings of the 23rd International Conference on Machine Learning, ICML'06, ACM, 2006, pp. 513–520.
[29] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in Neural Information Processing Systems, vol. 19, 2007, pp. 153–160.
[30] J. Snoek, R.P. Adams, H. Larochelle, Nonparametric guidance of autoencoder representations using label information, J. Mach. Learn. Res. 13 (1) (2012) 2567–2588.
[31] M. Ranzato, M. Szummer, Semi-supervised learning of compact document representations with deep networks, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 792–799.
[32] E.C. Malthouse, A.C. Tamhane, R.S.H. Mah, Nonlinear partial least squares, Comput. Chem. Eng. 21 (8) (1997) 875–890.