計測自動制御学会東北支部 第 305 回研究集会 (2016.11.25)
資料番号 305-4

カーネル法に基づく自己符号化器

Autoencoder Using Kernel Method

裴 岩

Yan Pei

会津大学 コンピュータ理工学部

School of Computer Science and Engineering, University of Aizu

キーワード： カーネル法 (kernel method), 深層学習 (deep learning), 自己符号化器 (autoencoder), カーネル法に基づく主成分分析 (kernel-based principal component analysis), カーネル法に基づく線形回帰 (kernel-based linear regression)

連絡先： 〒 965-8580 福島県会津若松市一箕町鶴賀 会津大学 コンピュータ理工学部 コンピュータサイエンス部門 裴岩 E-mail: peiyan@u-aizu.ac.jp

1. Introduction

The discovery on visual information processing of brain in neuroscience [6, 13] leads to developments on the subject that how to simulate high level data abstractions in artificial intelligence. Deep learning is a such way to implement this objective. The multiple levels and abstractions of learning model establish a basic framework of deep learning. In each level, it is implemented by multiple linear and non-linear transforms. From low level feature extraction to high level feature extraction, each level is a process of feature extraction and selection. Neural network (NN) is an universal approximator [5], therefore, it is used as a primary method to implement models and algorithms of deep learning [8]. Autoencoder [4], sparse coding [10], restricted Boltzmann machine, deep belief networks [3], and convolutional neural networks [9] are primary models and algorithms in deep learning. Most models and algorithms of deep learning use NN as their basic implementation structure.

Autoencoder is a representative algorithm in NN-based deep learning [3]. The objective of autoencoder is to learn a representation function of a set of data, which can generate or learn a generative models of the data. There are two parts in conventional NN-based autoencoder, i.e. an encoder and a decoder. Encoder part transfers data into another representation space, this process is called coder or latent presentation. The objective of encoder tries to present data in form of meaningful information, i.e. feature extraction. Decoder part transfers data proceed by the encoder back to their original representation space aiming to minimize the error between the original data and the data transferred back. If the error is as small as possible, the algorithm or method of autoencoder is considered as a great solution.

Kernel method is one of machine learning algorithms that tries to establish a linear model with structural risk minimization in feature space to solve the problem that cannot be dealt with linear model in its original space. It uses kernel function to establish a relation between the inner products of original space and projected high dimensional space. For example, $a \in R^d$ and $b \in R^d$ ($d \in Z^+$) are two real number vectors in inner product space, and there is a feature map φ can transfer these two vectors into a high dimensional space, i.e. $\varphi(a) \in R^h$ and $\varphi(b) \in R^h$ ($h \in Z^+$ and $d <= h$), the kernel function f implements the relation $< \varphi(a), \varphi(b) >= f(< a, b >)$. If a matrix composed by training samples with transformation function, i.e. kernel matrix, is semi-positive definite matrix, the function can be used as a kernel function. The kernel method is established with solid theoretical fundamental, and its methods or paradigms try to minimize structural risk, i.e. shrinkage estimators [7].

This work attempts to use algorithms of kernel method for data transformations in autoencoder. The NN and kernel method share some same characteristics. First, both of them can be used in classification and regression tasks in machine learning. Second, NN and kernel method can transfer data representation form and dimension, and find a model in transferred space to deal with the tasks. NN origins from the neuroscience that mimics biological neural networks, and it has few theoretical fundamental to prove the effectiveness of its algorithms and models. However, the kernel method is established with inner product simplification (kernel trick) and statistical learning theory. We, therefore, believe that kernel method is a perspective methodology to implement deep structural machine learning algorithms. In this paper, we use kernel-based principal component analysis as an encoder and kernel-based linear regression as decoder in an autoencoder model. This presents the originality of this work. The proposed method and structure are evaluated, analysed and discussed using some image data. If the proposed method is duplicated in the form of deep structure, it is prospected as a promising approach in deep learning field.

Following this introductionary section, we review the fundamentals of autoencoder, kernel method, kernel-based principal component analysis, and kernel-based linear regression in section 2. The proposed method that uses kernel method to construct an autoencoder model is described and explained in section 3. The encoder and decoder are implemented using kernel-based principal component analysis and kernel-based linear regression, respectively. In section 4., we use some images as training samples to evaluate the performance of proposed method. Because the image data are a visual signals, it is easy to evaluate and compare the performance with our visual perception. Finally, we conclude the whole work, and present some open topics and opportunities in the field of kernel method-based deep learning in section 5.

2. An Brief Overview on Autoencoder, Kernel Method, Kernel-based Principal Component Analysis, and Kernel-based Linear Regression

2.1 Autoencoder

The term, autoencoder, is presented as a method of learning efficient coding using NN [4, 17]. The autoencoder is implemented in the form of NN, it is used to reconstruct its input signals, i.e. coding presentation. Because the training process of autoencoder does not need the label of signal, it is a unsupervised learning model and method. The autoencoder usually includes two parts in its structure. The one is an encoder that transfers input signals into another space with another presentation form. The other is a decoder that transfers the transferred signals back to their original space with the objective to minimize the error between the input signals and the transferred back signals.

We describe autoencoder with a formal form. There are two Hilbert spaces, $PP \in R^d$ and $YY \in R^h$ $(d, h \in Z^+)$, and two transformations, $\Delta : PP \to YY$, and $\Theta : YY \to PP$ $(\Delta \in R^{h^{R^d}}, \Theta \in R^{d^{R^h}}; h, d \in Z^+)$. The objective of training an autoencoder is an optimization problem of Δ and Θ (an optimization problem in a function space), that minimizes the error between input signals and output signals, e.g. $py \in PP$ is an input signal, the optimization problem is shown in Eq. (1).

$$(\Theta, \Delta) = \arg \min_{\Delta \in R^{h^{R^d}}, \Theta \in R^{d^{R^h}}} ||py - \Theta(\Delta(py))||. \quad (1)$$

From the viewpoint of mathematical transition, the transformations Δ and Θ can be implemented not only by NN, but also by other forms of linear and non-linear transformations. The autoencoder implementation form is not limited within the field of NN. Kernel method is also an effective way to implement such transformation and copes with the optimization problem of Eq. (1).

2.2 Structural Risk Minimization and Kernel Method

The fundamental of kernel method lies in statistical learning theory and principle of structural risk minimization. There are $X \in R^d$ and $Y \in R$ $(d \in Z^+)$ that follow a certain and unknown distribution $P(X, Y)$. Machine learning algorithm tries to find the a function $Y = f(X, \gamma)$ can predict the relation between X and Y, where γ presents a set of parameters that defines f. To evaluate performance of prediction function f, we define a loss function $L(Y, f(X, \gamma))$ to penalize the errors [2]. The expected risk (Eq.(2)) was defined as a criterion to select a function f because convex function can be used as a loss function if it has arity two, positive range, and $L(x, x) = 0$ [14].

$$risk_{expected}(\gamma) = \int L(y, f(x, \gamma))dP(x, y). \quad (2)$$

We do not know the distribution of $P(X, Y)$, so we cannot know the expected risk. However, we can use training samples to estimate the expected risk, i.e. empirical risk (Eq.(3)).

$$risk_{empirical}(\gamma) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i, \gamma)). \quad (3)$$

In the limitation condition, if the number of training samples are infinity $(n \to \infty)$, the empirical risk can approximate expected risk, i.e. in Eq. (4).

$$\lim_{n\to\infty} risk_{expected}(\gamma) = risk_{empirical}(\gamma). \quad (4)$$

However, the over fitting problem often happens because minimization of $risk_{empirical}(\gamma)$ with finite training sample cannot guarantee the optimal selection of γ. We introduce structural risk into the optimization of $risk_{empirical}(\gamma)$ as in Eq. (5), where Ω is a function that measures the capacity of a set of functions f with parameter γ, α is a parameter used to manage trade off between training error and capacity.

$$risk_{structural}(\gamma) = risk_{empirical}(\gamma) + \alpha\Omega(\gamma). \quad (5)$$

Kernel method tries to minimize structural risk using kernel trick, i.e. finding optimal model in projected high dimensional feature space. For example, support vector machine tries to find the maximal margin to achieve this objective.

2.3 Kernel-based Principal Component Analysis

Principal component analysis (PCA) is a data transformation method that pursues to establish a coordinate system where the training samples have a maximal total variance [11]. It uses orthogonal transformation to keep the relation of training samples, i.e. inner product, as the same before and after the transformation. The total variance of the data that is projected to a direction v (new constructed coordinate system) can be expressed by Eq.s (6) and (7), where C is the co-variance matrix of the data. The objective of the data projection is to find a new coordinate system where the total variance has the maximum value. The aim of this optimization problem is to obtain a solution of Eq. (8). We can solve this optimization problem by using Lagrangian multiplier method or matrix calculus method.

$$\sigma^2 = v^T C v \tag{6}$$

$$C = \frac{1}{n}\sum_{i=1}^{n} x_i {x_i}^T = \frac{1}{n} X^T X. \tag{7}$$

$$v = \arg\max_{v} v^T C v. \tag{8}$$

Kernel-based PCA is the generic form of PCA, which constructs new coordinate system in a high dimensional space where the projected data have maximal total variance [12]. It establishes a relation of co-variance matrix and kernel matrix (Eq.s (9), κ's pronunciation is kappa, $\tilde{X}^T = [\varphi(x_1), \varphi(x_2), ..., \varphi(x_N)]$) of projected data, and tries to calculate the eigenvalue problem of kernel matrix to indirectly find the eigenvalue problem solution of co-variance matrix (Eq.s (10)-(14)) . Because we have an assumption that the projected data is centred, we need to use kernel matrix to express a matrix with centred data, i.e. Gram matrix (Eq. (15), 1_N presents a $N \times N$ matrix with $1_N(i,j) = \frac{1}{N}$).

$$\kappa = \tilde{X}\tilde{X}^T. \tag{9}$$

$$\begin{aligned} (\kappa)u &= \lambda u. & (10)\\ (\tilde{X}\tilde{X}^T)u &= \lambda u. & (11)\\ \tilde{X}^T(\tilde{X}\tilde{X}^T)u &= \tilde{X}^T\lambda u. & (12)\\ (\tilde{X}^T\tilde{X})(\tilde{X}^T u) &= \lambda(\tilde{X}^T u). & (13)\\ (\tilde{C})(\tilde{X}^T u) &= \lambda(\tilde{X}^T u). & (14) \end{aligned}$$

$$\tilde{\kappa} = \kappa - 1_N\kappa - \kappa 1_N + 1_N\kappa 1_N. \tag{15}$$

2.4 Kernel-based Linear Regression

Linear regression is a method that establishes the relation between independent variables $x_i \in R^d, i = 1, 2, ..., N$ $(d \in Z^+)$ and $y \in R$ $(Y^T = [y_1, y_2, ..., y_N])$ (Eq. (16)). Many methods are used to establish the linear regression models, such as least square method, interpretation, maximum-likelihood estimation, etc. If we use least square method to find the regression model, Eq. (18) is an optimization problem that considers w as an optimization target. We establish the normal equations (Eq. (19)) to solve this problem and try to obtain a proper w (Eq. (20)).

$$Y = Xw + \varepsilon. \tag{16}$$

$$X = \begin{bmatrix} 1 & x_1^T \\ \vdots & \vdots \\ 1 & x_N^T \end{bmatrix}. \tag{17}$$

$$w = \arg\min_{w} ||Y - Xw||^2. \tag{18}$$

$$\begin{aligned} X^T X w &= X^T Y. & (19)\\ w &= (X^T X)^{-1} X^T Y. & (20) \end{aligned}$$

With the primary motivation of kernel method, we can project data X into high dimensional space ($\bar{X}$, Eq. (21)) and try to find a linear relation between projected $\bar{X}$ and Y. The normal equations can be re-written as in Eq. (24). We note the $\alpha = \bar{X}(\bar{X}^T\bar{X})^{-2}\bar{X}^T y$, and use kernel trick to express α with kernel matrix (Eq.s (26)-(30), $\bar{\kappa} = 1_{N\times N} + \kappa$). Finally, the expression of kernel-based linear regression is in Eq. (31).

$$\bar{X} = \begin{bmatrix} 1 & \varphi(x_1)^T \\ \vdots & \vdots \\ 1 & \varphi(x_N)^T \end{bmatrix} \tag{21}$$

$$\bar{X}^T\bar{X}w = \bar{X}^TY. \quad (22)$$
$$w = (\bar{X}^T\bar{X})^{-1}\bar{X}^TY. \quad (23)$$
$$w = (\bar{X}^T\bar{X})(\bar{X}^T\bar{X})^{-2}\bar{X}^TY. \quad (24)$$
$$w = \bar{X}^T\alpha. \quad (25)$$

$$\bar{X}^T\bar{X}w = \bar{X}^TY. \quad (26)$$
$$\bar{X}^T\bar{X}\bar{X}^T\alpha = \bar{X}^TY. \quad (27)$$
$$\bar{X}\bar{X}^T\bar{X}\bar{X}^T\alpha = \bar{X}\bar{X}^TY. \quad (28)$$
$$\bar{\kappa}^2\alpha = \bar{\kappa}Y. \quad (29)$$
$$\alpha = \bar{\kappa}^{-1}Y. \quad (30)$$

$$Y = \bar{X}w = \bar{\kappa}^{-1}\alpha = (1_{N\times N} + \kappa)\alpha \quad (31)$$

3. Autoencoder Using Kernel Method: towards Model and Algorithm of Kernel Method-based Deep Learning

3.1 Kernel Method-based Deep Learning and Autoencoder

The primary function of autoencoder is implemented by linear and non-linear transformations, which is carried out by NN [4]. NN is an universal approximator that can approximate any linear and non-linear transformations with any arbitrary accuracy [5]. For simulating the high level data abstractions, deep learning model constructs the multiple transformations to implement this process. Data abstraction in multiple levels and data transformation with multiple linear or non-linear transformations are two characteristics of deep learning. From this viewpoint, any meaningful data transformation method with multiple data transformations can implement deep learning model and its objective, i.e. high level data abstraction.

Kernel method can implement linear and non-linear transformations that transfer data from their original space into a high dimensional space to find linear learning model. In the view of data transformation, there is not any difference between kernel method and NN. Between each transformation, kernel method can construct linear learning model to obtaining meaningful data abstraction with the principle of structural risk minimization. The kernel method can be used in the structure of data abstraction with multiple levels to implement a deep learning model and algorithm. However, the transformation in kernel method have not been proved as a universal approximator, this subject needs to further study and investigate in our future work. From the viewpoint of data transformation, fuzzy system is also a universal approximator [15], so that fussy system can be as well as considered as another implementation of deep learning.

As a summary, kernel method-based deep learning has two characteristics. The one is multiple linear and non-linear transformations to implement high level dat abstraction. The other is that linear and non-linear transformations are constructed by kernel method that uses kernel trick to find linear learning models and select data features. In this work, we investigate an autoencoder implementation by kernel-based principal component analysis and kernel-based linear regression (Fig. 1). If the basic structure is implemented with multiple levels, it can construct a deep learning model for classification and regression to implement high level data abstraction.

3.2 Encoder Using Kernel-based Principal Component Analysis

Encoder is a data abstraction process that tries to transfer data into another representative space for feature selection and extraction. As explanation above, the encoder part implements the transformation $\Delta : PP \to YY$, PP

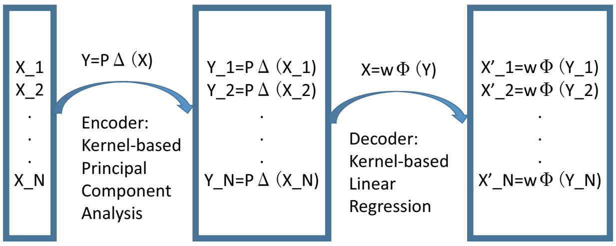


Fig. 1 concept of kernel method-based autoencoder, the encoder part is implemented by kernel-based principal component analysis, and the decoder part is implemented by kernel-based linear regression.

is the original space of the data, and YY is the projected space by kernel trick. We use kernel-based PCA to transfer data from space PP to space YY, and implement transformation Δ. If polynomial kernel and Gaussian kernel are used in the kernel-based PCA, the dimension of YY is $\frac{(r+d-1)!}{d!(r-1)!}$ (d is dimension of training sample, r is the degree of polynomial kernel function, $d \in Z^+, r \in Z^+$) and infinite, respectively. We can reduce the dimension of original data, and use only first m ($m \in Z^+, m <= d$) principal components to present the original data, i.e. feature extraction.

There are several issues need to be considered in this process. First, the kernel function and its parameter setting selection is a problem for obtaining better feature extraction. This is, in a function space $\Delta \in R^{h^{R^d}}$, we need to find a proper or optimal Δ to implement encoder. The kernel function decides the topological structure of projected high dimensional space and feature construction. Second, the principal component selection is also an issue for feature extraction. We need to analyse and investigate these two subjects when applying kernel-based PCA as encoder in implementation of an autoencoder.

3.3 Decoder Using Kernel-based Linear Regression

Decoder is used to restore the data back to their original representation space. It implements, selects, and optimizes the transformation $\Theta \in R^{d^{R^h}}$. In this work, we use kernel-based linear regression to implement this part. Because the kernel-based linear regression establishes the relation between multiple variables x_i and one variable y. The x_i in Eq. (16) presents $Y_1, Y_2, ..., Y_N$ in Figure 1, Y has m dimension decided by selection of principal components of kernel-based PCA. The y in Eq. (16) is one of dimension of $X'_1, X'_2, ..., X'_N$ in Figure 1, X' has d dimension as the same as their original input data (X). So we need to establish the number of d kernel-based linear regression models to transfer $\Delta(X)$ back to X, i.e. as in Eq. (32), and $\Theta = \sum_{i=1}^{d} \Theta_i$, ($i = 1, 2, ..., d$). We implement the decoder transformations (Θ) in this form.

$$x_i = \Theta_i(\Delta(X)). \tag{32}$$

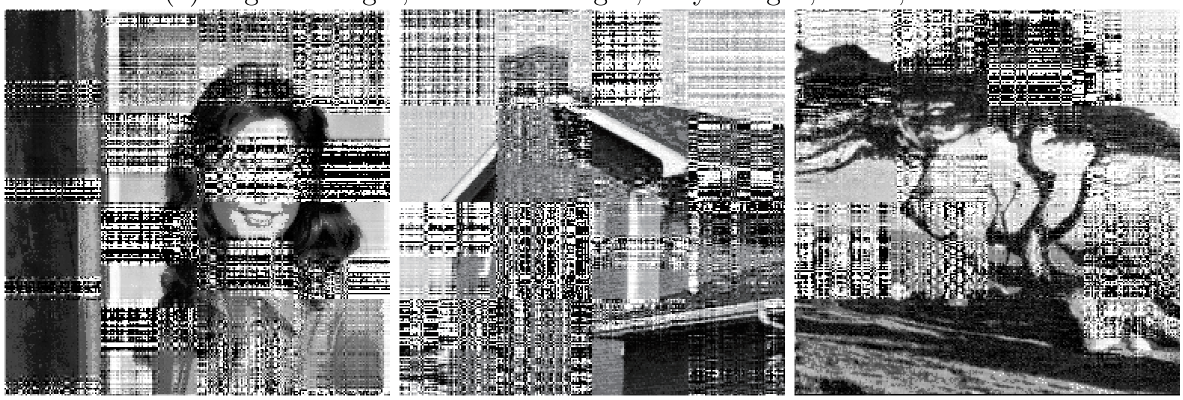
4. Evaluations and Discussions

4.1 Performance of Proposed Autoencoder

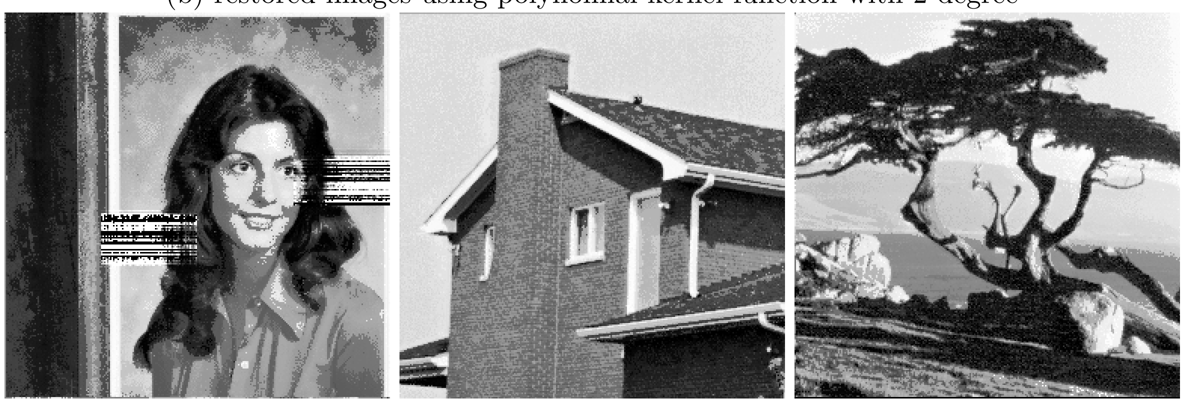
Because abstract digital data or classification problem can not be visually evaluated, we use visual image data to evaluate our proposal. We first encoder, and then decoder the images in our evaluation, the designed autoencoder can be considered as an image filter in this process. We use three images to evaluate our proposed autoencoder. These images are grayscale format with 256×256 pixels (Figure 2-(a)). We use polynomial kernel function with 2 and 3 degree in the kernel-based PCA and linear regression. The definition of polynomial kernel function is in Eq. (33), and $r = 2$ and $r = 3$ in the evaluations. The restored images using our proposed autoencoder are shown in Figure 2-(b) and 2-(c), which uses polynomial kernel functions with 2 and 3 degree, respectively.

$$\kappa(x, y) = (< x, y > +1)^r. \tag{33}$$

(a) original images, from left to right, they are girl, house, and tree



(b) restored images using polynomial kernel function with 2 degree



(c) restored images using polynomial kernel function with 3 degree

Fig. 2 Three original test images, and restored images using proposed autoencoder with polynomial kernel functions with 2 and 3 degree, they are all greyscale format images, and the size of all the images are 256×256. We can observe that restored images using polynomial kernel function with 3 degree are clearer than those using polynomial kernel function with 2 degree. The structural similarities of three images with polynomial kernel function with 2 and 3 degree are 0.3723, 0.2802, 0.4633 and 0.9292, 0.9998, 0.9998, respectively

Kernel function and its parameter selection decide the topological structure of high dimension Hilbert space. The new constructed coordinate system using kernel-based PCA and kernel-based linear regression model also depend on the selections of kernel function and its parameter. In this evaluation, we use polynomial kernel function in both encoder and decoder with 2 and 3 degree. We can visually observe that the restored images using polynomial kernel function with 3 degree is better than the that using the one with 2 degree. Some of the results using polynomial kernel with 2 degree faults in obtaining a proper inverse matrix of kernel matrix and solution of quadratic programming, this is the reason that the images from polynomial kernel function with 2 degree are unclear. The first image of Figure 2-(c) fault in restore with the same reason.

We use structural similarity [16] to quantitatively evaluate and analyse the restored images. The μ and σ are the mean value and variance value in Eq. (34), where $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator; L is the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$); and $k_1 = 0.01$ and $k_2 = 0.03$. The structural similarities of three images with polynomial kernel function with 2 and 3 degree are 0.3723, 0.2802, 0.4633 and 0.9292, 0.9998, 0.9998, respectively. From this evaluation, we quantitatively investigate the data transform performance of our proposed autoencoder. Because the SSIM value is near 1, it means the original image and restored image have more similarity, the restored images obtained by polynomial kernel function with 3 degree are better.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{34}$$

The encoder and decoder of proposed autoencoder have two kernel-based data transformation algorithms, kernel-based PCA and kernel-based linear regression. The performance of both algorithms depends on the kernel function's selection and their parameter setting. In this evaluation, we use the polynomial kernel function with 2 and 3 degree in both of them, but it is not necessary to use the same kernel function in kernel-based PCA and kernel-based linear regression. The proposed autoencoder has a variety of implementation ways to construct feature abstraction and extraction algorithm by considering training data and kernel function characteristics. The subject will be involved in our future work.

4.2 Kernel-based Deep Learning Implementation

In the mathematical viewpoint, autoencoder concept implements two data transformations. The first transformation pursues to find the feature of the training data, and the second one transfers (restores) back the data into their original form. If the restored data are as the same as the original data, the autoencoder can be considered as a great algorithm. If we apply this autoencoder with multiple levels, the whole structure can implement high level data abstraction so that the algorithm can simulate the perception as our human brain. The conventional autoencoder uses NN as a basic unit for data transformation, the primary idea of kernel method also can be considered as a tool of data transformation. In this paper, we initially implemented a kernel-based autoencoder and evaluated its performance. This is one of the originality of this work.

The primary philosophy of kernel method lies on statistical learning theory and principle of structural risk minimization. However, it is also a method of data transform that has potential possibility to implement high level data abstraction in deep learning algorithms. The direct way to implement kernel-based deep learning is duplicately applying kernel transformation $\varphi(\varphi(...\varphi(x)))$ [1]. The obvious issue of this implementation ignores the objective of deep learning that pursues to obtain the data feature between transformations and implements high level abstraction rather than simply simulates deep structure. Designing a proper kernel function can simulate data transformation, but cannot obtain the data feature so that implements high level data abstraction. In each level of data transformation, designing a proper kernel function, setting efficient parameter, and finding data feature to implement high level data abstraction are three subjects to improve this method.

The second way to implement kernel-based deep learning does not only simulate the deep structure of kernel transformation, but also apply kernel-based machine learning algorithm to construct high level data abstraction. This work uses kernel-based PCA and kernel-based linear regression for feature extraction, but it is not limited in these two algorithms, other kernel-based algorithms also can be involved in the proposed algorithm structure. The algorithm selection of autoencoder, kernel function's selection and design, construction of high level data abstraction are potential research subjects in our future work.

5. Conclusion and Future Works

We proposed to use kernel-based algorithms in a deep structure to implement high level data abstraction. A kernel method based autoencoder is initially designed, implemented and evaluated in this work. The encoder and decoder parts are implemented by kernel-based PCA and kernel-based linear regression, respectively. Kernel-based PCA transfers the data into a high dimensional Hilbert space for obtaining constructed data feature, kernel-based linear regression transfers the data back to their original form for evaluating the performance of autoencoder. We found that selection of kernel function decides the deigned autoencoder, and fault in matrix inverse influences the algorithm applications of kernel method. If we duplicately apply proposed autoencoder in a deep structure, we can implement deep learning algorithm with kernel method.

The algorithm selection, kernel function and its parameter selection, and deep learning algorithm evaluation are three remaining subjects in our future work. First, the kernel-based PCA and kernel-based linear regression were initially evaluated in our designed autoencoder, but other kernel method-based algorithms also can be used in this framework. We hope this study subject can lead to more implementations in kernel method based deep learning algorithm, e.g. kernel-based discriminant analysis, support vector machine, etc. Second, from our evaluations, performance of designed autoencoder depends on the selection of kernel function. But, we have not known the optimal selection and its conclusion on selection issue of kernel function and its parameters. This leads to the further study on the optimal selection subject of kernel function in kernel-based deep learning. Third, in this work, we only implemented one level autoencoder, and simply evaluated its performance. We need to apply this structure in multiple levels to implement high level data abstraction. These and other study subjects will be involved in our future works.

参考文献

1) Y. Cho and L. K. Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009.

2) T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2009.

3) G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

4) G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

5) K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are univer-

sal approximators. *Neural networks*, 2(5):359–366, 1989.

6) D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

7) W. James and C. Stein. Estimation with quadratic loss. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 361–379, 1961.

8) Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

9) Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

10) B. A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

11) K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

12) B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

13) C. J. Shatz. David hunter hubel (1926-2013). *Nature*, 502(7473):625–625, 2013.

14) A. J. Smola, B. Schölkopf, and K.-R. Müller. General cost functions for support vector regression. In *IN Proceedings of the 8th International Conference on Artificial Neural Networks*. Citeseer, 1998.

15) L.-X. Wang and J. M. Mendel. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE transactions on Neural Networks*, 3(5):807–814, 1992.

16) Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

17) D. Williams and G. Hinton. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.