Distance Correlation Autoencoder

Rick Wang
Department of Statistics
University of Waterloo
rickwang21@gmail.com

Amir-Hossein Karimi
Cheriton School of Computer Science
University of Waterloo
a6karimi@uwaterloo.ca

Ali Ghodsi
Department of Statistics
University of Waterloo
aghodsib@uwaterloo.ca

***Abstract*—This paper looks at applying a new objective function based on distance correlation for supervised dimensionality reduction using autoencoders. We elaborate on different properties of distance correlation to illustrate that maximizing it would be beneficial for supervised dimensionality reduction. We also described the general structure of the autoencoder used to maximize distance correlation. Lastly, our model was applied to a variety of problem sets from toy examples to regression and image datasets and obtained good results in general.**

I. Introduction

With the rapid increase in high dimensional datasets available for analysis such as images, videos, and text, there arises a need to reduce the dimension of these datasets as most algorithms do not work well with high dimensional data. Supervised dimension reduction is a set of algorithms that seek to reduce the data to low dimensions in a way that is effective for classification and regression algorithms.

There have been multiple recent efforts towards such a goal and a classic example would be fisher linear discriminant analysis for classification. It is an algorithm built on minimizing within class variation while maximizing between class variation with the intended result that data points within same class are clustered together while data points in different classes are far apart. However, it assumes the two classes follow a Gaussian distribution with same covariance structure and could provide sub-optimal results if the data has complex structures needed for classification, i.e. multi-modal data. In the case of regression, a definition of supervised dimension reduction developed by one of the earliest techniques, sliced inverse regression (SIR) [1], would be to find a transformation matrix B such that the response variable Y is independent of X given that $B^T X$ is known. This definition of a p by d matrix B where $d \ll p$ such that the following holds:

$$Y \perp\!\!\!\perp X \mid\mid B^T X \tag{1}$$

reduces the problem to finding a subspace of X and forms the central goal of many later papers such as Likelihood Acquired Directions (LAD) [2] and gradient-based Kernel Dimension Reduction (gKDR) [3]. Unfortunately, this matrix B restricts any transformation of X to be linear and the techniques developed usually cannot handle the large datasets (n>10,000) common in machine learning problems.

In this paper, a new method called Distance Correlation (dCor) autoencoder is proposed. Supervised dimensionality reduction methods based on distance correlation [4] or other independence criterion [5] have been applied previously and were shown to be very effective. However, their method requires iterative optimization which becomes intractable for large datasets. Their method also does not learn an explicit mapping that can be applied to new test points. This paper's contribution is a new autoencoder-based method that seeks to find a low dimension representation Z that has maximal distance correlation with the response variable Y. This allows for an explicit nonlinear transformation of X to a low dimensional representation Z and also scales well to both large and varied datasets such as images, sequences and other unstructured data due to the flexibility of an autoencoder.

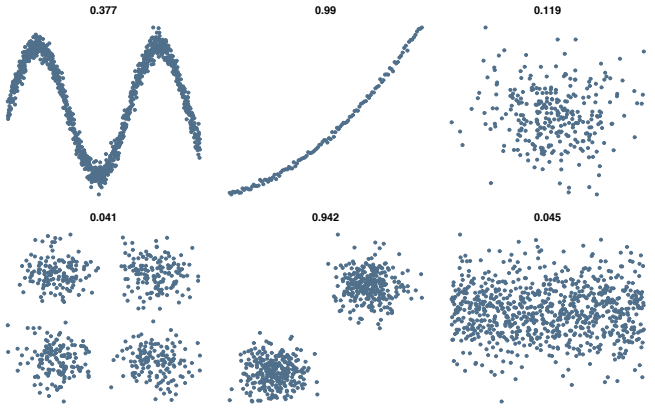


Fig. 1: Various distributions for X and Y random variables and their associated distance correlation value demonstrating the link between distance correlation and statistical dependency.

II. Distance Correlation

Distance correlation (dCor) [6] is a measure for estimating the nonlinear dependency of two random variables $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ where both can take arbitrary dimensions. Distance correlation ranges from 0 to 1 where 0 implies X and Y are independent. In general, dCor works very well as a measure of statistical dependency when compared to linear counterparts such as Pearson Correlation [7]. Székely et al, in the original 2007 paper showed in many cases that a hypothesis test based on distance correlation exhibits a significant increase in power over other tests when testing for the independence of two distributions. Figure 1 depicts plots for different distributions of X and Y and their corresponding sample distance correlation value also demonstrates the statistical dependency between X and Y. These plots essentially indicate that if X has high distance correlation with Y, then if x_i is known, y_i can be predicted with low error due to its high dependence on x_i. As we can see,

978-1-5090-6014-6/18/$31.00 ©2018 IEEE

sample distance correlation increases with higher predictive power of X on Y and vice versa.

This property that increasing distance correlation increases the dependency and therefore predictive power between X and Y is critical for formulating a method for supervised dimensionality reduction. If a function $z_i = f(x_i, \theta)$ can be found such that $Z = (z_1, \ldots, z_n)$ has high distance correlation with $Y = (y_1, \ldots, y_n)$, then Z can effectively be used instead of X since Z is highly predictive for Y. All that remains is to restrict the dimensions of z_i to be less than x_i.

To form the proper objective function, we can use the sample distance correlation measure developed by Székely et al. Defining the following terms where $|\cdot|$ indicates L2 norm:

$$a_{kl} = |x_k - x_l| \qquad a_{..} = \frac{1}{n^2}\sum_{k,l} a_{kl}$$

$$a_{.l} = \frac{1}{n}\sum_{k} a_{kl} \qquad a_{k.} = \frac{1}{n}\sum_{l} a_{kl}$$

$$b_{kl} = |x_k - x_l| \qquad b_{..} = \frac{1}{n^2}\sum_{k,l} b_{kl}$$

$$b_{.l} = \frac{1}{n}\sum_{k} b_{kl} \qquad b_{k.} = \frac{1}{n}\sum_{l} b_{kl}$$

$$A_{kl} = a_{kl} - a_{.l} - a_{k.} + a_{..}$$
$$B_{kl} = b_{kl} - b_{.l} - b_{k.} + b_{..}$$

Székely et al. proved that an empirical estimate for *distance covariance* is:

$$V_n^2(X, Y) = \sum_{k,l=1} A_{kl} B_{kl}$$

The empirical estimate for *distance variance* is:

$$V_n^2(X) = \sum_{k,l=1} A_{kl}^2$$

Finally the empirical estimate for *distance correlation* is:

$$R_n^2(X, Y) = \begin{cases} \frac{V_n^2(X,Y)}{\sqrt{V_n^2(X)V_n^2(Y)}}, & \text{if } V_n^2(X)V_n^2(Y) > 0 \\ 0, & \text{if } V_n^2(X)V_n^2(Y) = 0 \end{cases}$$

Since it can be incredibly hard to find the optimal $z_i = f(x_i, \theta)$ both in terms of the form of the function and its parameters θ for the objective $R_n^2(f(X, \theta), Y)$, autoencoders from the field of deep learning will be proposed as a solution in the next section.

III. Distance Correlation Autoencoder

Typical autoencoders, whether deep or shallow, are neural networks with relatively simple structures and used for unsupervised dimension reduction. These neural networks seek to encode data into a lower dimensional representation and reproduce the original data from this low dimension representation with the least amount of distortion. There are two main components to an autoencoder. The first is an encoder to reduce data and denoted in this paper by X to a lower dimension representation denoted by Z. Another component called decoder then reproduces X with $\hat{X}$ as closely as possible from Z. A simple autoencoder example would be a shallow linear autoencoder and the transformation the encoder imposes on the input data X can be written as:

$$z_i = W_0 x_i$$

Similarly for the decoder we have:

$$\hat{x_i} = W_0^* z_i$$

and the overall objective function to minimize is:

$$\min_{W_0, W_0^*} |x_i - \hat{x_i}|^2 = \min_{W_0, W_0^*} |x_i - W_0^* W_0 x_i|^2$$

A locally optimal solution can be then be found using back-propagation with stochastic gradient descent.

However this model can be extended considerably with multiple "layers" to represent $W_0, W_1, \ldots, W_l$ and $W_0^*, \ldots, W_{l^*}^*$ and nonlinear functions can be applied element-wise at each matrix multiplication. This significantly increases the capacity of the autoencoder to approximate a wide variety of possible functions and at the same time, provide fast optimization of parameters using back-propagation with stochastic gradient descent. Additionally, LSTM neural networks [8] and convolutional networks can be also be used for both the decoder and encoder when applied to sequence-based or image data, respectively.

Drawing from the experiences researchers had with training autoencoders that have additional objective functions while reducing the reproduction error of the decoding section, it seemed appropriate to attempt a like effort for the distance correlation objective function. The high capacity of autoencoders should let us sidestep the problem of directly finding an optimal function $f(x_i, \theta)$ by approximating it using a sufficiently high number of parameters and complicated nonlinear functions. Additionally, autoencoders as neural networks have an extremely high number of parameters and would unlikely be stuck in a poor local minima for distance correlation objective function (also discussed here [9]). This was later observed during testing where repeated experimentation obtained very close distance correlation values despite different initial parameter values. A variety of neural network specific techniques can also be used for better results such as Adadelta [10] and regularization methods such as Dropout [11].

Using similar notations as before, let $x_i \in R^p$ with corresponding response variable $y_i \in R^q$ for $i = 1, \ldots, n$ be vectors from a dataset X, Y. Each pair of y_i, y_j encode the desired distance between the corresponding x_i, x_j. The goal of the distance correlation autoencoder is to find a low dimension representation Z that has maximal distance correlation with Y with the intended purpose of increasing the statistical dependency of Y on Z. A sufficiently powerful classifier should then be able to classify more accurately using Z than an alternative Z^* that has less distance correlation. Alternatively,

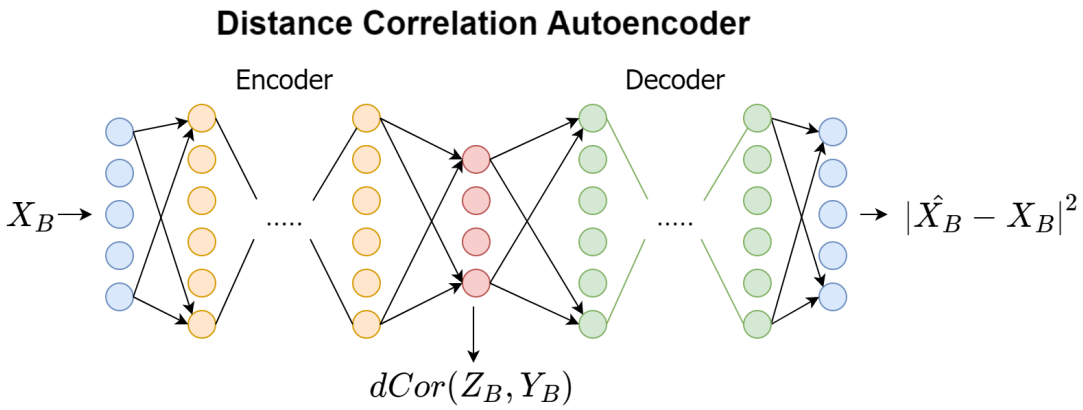


Fig. 2: Setup for Deep dCor Autoencoder.

from a regression perspective, the predictive power of Z for Y should be high assuming a sufficiently well-performing regression model due to high dependency. Lastly, it would be highly preferable to find a transformation $Z = f(X;\theta)$ such that the distance correlation between Z and Y is close to maximal value at 1. This implies $Y \approx a + bZC$ and means that y_i can easily be predicted from z_i due to their almost direct linear relationship.

A. Simple Autoencoder

A simple shallow autoencoder will be used as the first model with both the encoder and decoder having only one hidden layer each. The encoder layer learns a set of parameters for the function $z_i = f(x_i;\theta)$. Assuming $x_i \in R^m$ and hyperbolic tan activation function for encoding layer and linear activation for the latent layer, we have:

$$\begin{aligned} h_i &= \tanh(W_0 x_i + b_0) \\ z_i &= W_1 h_i + b_1 \\ f(x_i, W_0, W_1, b_0, b_1) &= W_1 \tanh(W_0 x_i + b_0) + b_1 \end{aligned}$$

where $\tanh$ is applied element-wise to $Wx_i + b$. Since the z_i's are the latent representations encoded by the autoencoder, we want it to have high distance correlation with the labels Y and increase the statistical dependency of Y on Z (and vice versa). Therefore, we want to find W_0, W_1, and b_0 such that Z has maximal distance correlation with Y.

To calculate sample distance correlation objective function, a batch of sample points X_B, Y_B is needed. The autoencoder then encodes X_B into Z_B and a modified distance correlation formula is used to calculate the distance correlation between Z_B and Y_B. The squared L_2 norm was used instead to avoid repeatedly calculating square roots. Also, the negative log was used to turn the whole objective function into a minimization of the distance correlation objective. Note that a bias, b_1 is not needed and may be omitted for z_i since moving all points by a fixed translation does not change any distance values and hence distance correlation, but we will include it for consistency. The result is the following distance correlation objective function for some batch X_B, Y_B:

$$\begin{aligned} &= \min_{\substack{W_0,W_1,\\ b_0,b_1}} \; -\log(R^2_{n_b}(Z_B, Y_B)) \\ &= \min_{\substack{W_0,W_1,\\ b_0,b_1}} \; -\log\Big(\sum_{i,j\in B} Z_{ij}Y_{ij}\Big) + \log\Big(\sum_{i,j\in B} Z^2_{ij}\Big) + \log\Big(\sum_{i,j\in B} Y^2_{ij}\Big) \\ &= \min_{\substack{W_0,W_1,\\ b_0,b_1}} \; dCor(Z_B, Y_B) \end{aligned}$$

where n_b denotes the number of data points in the batches X_B, Y_B. Z_{kl} and Y_{kl} are calculated similar to A_{kl} and B_{kl} in Section II.

In addition to maximizing distance correlation between X and Y, it is also important that the autoencoder can reconstruct the data point from the latent code z_i with low error as a measure of regularization. Assuming hyperbolic tan activation functions once again, we have:

$$\begin{aligned} d_i &= \tanh(W_2 z_i + b_2) \\ \hat{x_i} &= \tanh(W_3 d_i + b_3) \end{aligned}$$

A suitable objective function could be mean-squared error and letting λ denote the weight of the reconstruction loss ($\lambda = 0.1$ was used for all experiments), we have:

$$\begin{aligned} &= \min_{\substack{W_0,W_1,W_2,W_3,\\ b_0,b_1,b_2,b_3}} \lambda \frac{1}{n_b} \sum_{i\in B} |\hat{x_i} - x_i|^2 \\ &= \min_{\substack{W_0,W_1,W_2,W_3,\\ b_0,b_1,b_2,b_3}} \lambda L(\hat{X_b}, X_b) \end{aligned}$$

Combined, the overall objective function is:

$$= \min_{\substack{W_0,W_1,W_2,W_3,\\ b_0,b_1,b_2,b_3}} dCor(Z_b, Y_b) + \lambda L(\hat{X_b}, X_b)$$

And for each batch the parameters are updated with stochastic gradient descent for some learning rate μ:

$$W_0 = W_0 - \mu\Big(\frac{\partial dCor(Z_b, Y_b)}{\partial W_0} + \lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial W_0}\Big)$$
$$W_1 = W_1 - \mu\Big(\frac{\partial dCor(Z_b, Y_b)}{\partial W_1} + \lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial W_1}\Big)$$
$$W_2 = W_2 - \mu\Big(\lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial W_2}\Big)$$
$$W_3 = W_3 - \mu\Big(\lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial W_3}\Big)$$
$$b_0 = b_0 - \mu\Big(\frac{\partial dCor(Z_b, Y_b)}{\partial b_0} + \lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial b_0}\Big)$$
$$b_1 = b_1 - \mu\Big(\frac{\partial dCor(Z_b, Y_b)}{\partial b_1} + \lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial b_1}\Big)$$
$$b_2 = b_2 - \mu\Big(\lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial b_2}\Big)$$
$$b_3 = b_3 - \mu\Big(\lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial b_3}\Big)$$

For any new points x_t encountered during the test, the parameters of the trained distance correlation autoencoder are fixed and the following equations are used to estimate z_t without the corresponding y_t.

$$h_t = \tanh(W_0 x_t + b_0)$$
$$z_t = W_1 h_t + b_1$$

B. Deep Autoencoder

This model is easily extended to deep encoders and decoders with other activation functions and multiple layers. Using the general notation defined below:

- z_i be the latent representation of an input vector x_i generated by the encoder
- $\hat{x}_i$ be the reconstructed output generated by the decoder from z_i
- θ and ϕ be the parameters of the encoder and decoder respectively
- $f(x_i; \theta)$ and $g(z_i; \phi)$ represent the transformation the encoder and decoder impose on input x_i and latent representation z_i respectively
- $L(\hat{x}_i, x_i)$ represent a suitable reconstruction error such as mean squared error

We have the following equations:

$$z_i = f(x_i; \theta)$$
$$\hat{x_i} = g(z_i; \phi)$$

The objective function is then:

$$= \min_{\theta,\phi} dCor(Z_b, Y_b) + \lambda L(\hat{X}_b, X_b)$$

And the gradient descent updates are:

$$\theta = \theta - \mu\Big(\frac{\partial dCor(Z_b, Y_b)}{\partial \theta} + \lambda\frac{\partial L(\hat{X}_b, X_b)}{\partial \theta}\Big)$$
$$\phi = \phi - \mu\lambda\Big(\frac{\partial L(\hat{X}_b, X_b)}{\partial \phi}\Big)$$

In general during testing, it was found that stochastic gradient descent converged very slowly towards any local minima and using more modern techniques like ADAM [12] or Adadelta were very effective at reducing the number of epochs necessary for convergence. Additionally, regularization techniques like Dropout or noisy layers can be employed since deep neural networks can easily overfit. Lastly, any new test points x_t could be estimated with $z_t = f(x_t; \theta)$ similar to the simple shallow autoencoder case.

IV. Results

A. Toy Examples

Initial testing with toy examples was done on concentric circles and two-moons dataset. Since these are relatively small dataset and visualization was important, only 4 layers were chosen for both encoder and decoder and the number of nodes in each layer was restricted to 6. The activation function used for every layer was hyperbolic tan. The dimension of Z was set at 2.

In Figure 3 we observe the latent representation of the concentric circle and two-moons datasets produced by the distance correlation autoencoder, showing that it works very well. The two classes of points are nicely separated with only minor intermix of points in the middle. A simple linear classifier would work very well to classify the two set of points. Additionally, there were no overfitting issues as the test points were properly placed with the training points. This was also accompanied by much higher sample distance correlation to match the proposition that higher distance correlation increases the predictive power of Z on Y.

Also, the shape of the overall distribution of points indicates by what process the autoencoder works through to properly separate points and maximize distance correlation. The general shape is of a cone in three-dimensional space and it can be seen that the original circle distribution of points was transformed into a three-dimensional cone such that one class of points in the middle formed the center of the cone. A projection of this from the side would suffice to separate the two class of points. To demonstrate this, PCA was used on the last encoder layer and very similar results to distance correlation autoencoder were obtained. The orientation and width of the cone were somewhat different but as these do not affect sample distance correlation, it is understandable if the autoencoder settled on a different and better orientation for its results. Another plot of the representation generated by the last layer of the encoder shows the curved cone shape of the higher dimension representation.

However, further training the autoencoder to minimize does tend to produce increasingly extreme visualizations and efforts to separate the two classes of points. However, it does seem easier to classify which is matched by an increase in sample distance correlation for the test dataset (0.832 to 0.873).

A further example using the two moon dataset but same autoencoder configuration similarly shows how effective maximizing distance correlation can result in easily separable representations. The sample distance correlation value is a very

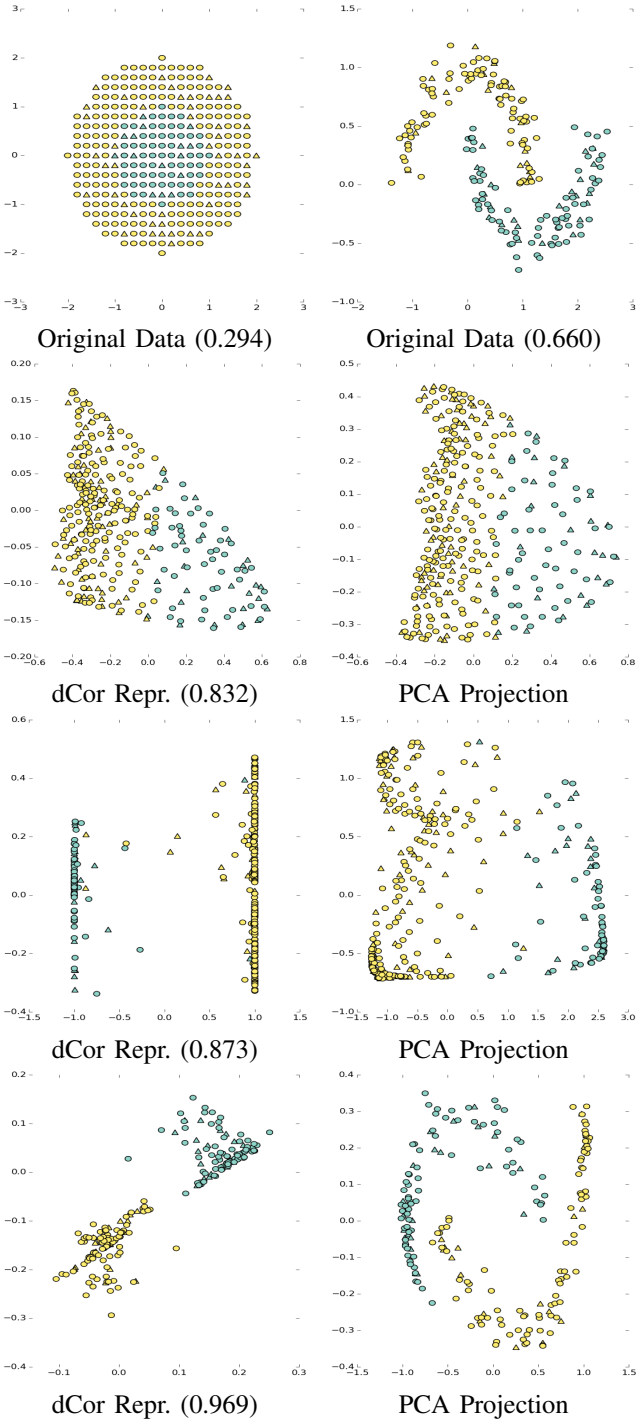


Fig. 3: **top row:** concentric circle and two-moons datasets, **second row:** dCor representation after training concentric circles for 2500 epochs, along with sample distance correlation in brackets, and PCA projected representation of final encoder layer, **third row:** identical setup for concentric circles after 4000 epochs, here we observe more extreme separation of latent space and higher sample distance correlation, **fourth row:** identical setup for two-moons after 2000 epochs. The two different classes are colored yellow and blue. Triangles indicate test examples that were not included in the initial training process while circles represent training data points.

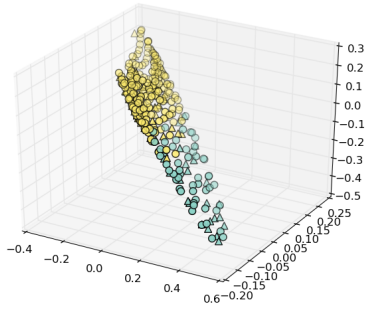


Fig. 4: Plot of last encoder layer before projection.

high 0.969 and this is matched in visualization by two clusters of easily separable points.

B. Regression Problems

Aside from classification datasets where Y encodes fixed distances between different classes of points, it is also possible to apply distance correlation autoencoder to regression datasets where Y can be any real number. The first dataset used is the Boston housing dataset composed of 506 sample and the main focus is to predict the median housing value given 13 covariates. The dataset was randomly split into 400/106 for training and validation purposes and this process was repeated five times. A linear regression model was then fitted to the low dimension representation generated by each model and the averaged root mean squared error was reported in the tables below.

TABLE I: Boston Housing Dataset Comparison. dCor and gKDR parameters were decided using cross validation and selecting best. Regression errors reported as averaged RMS.

	Dimensions			
Methods	**3**	**5**	**7**	**9**
dCor	4.957	4.564	4.438	4.354
SPCA (RBF) [5]	7.642	7.953	5.690	5.161
gKDR	5.626	5.960	5.365	5.153
PCA	7.595	7.875	5.891	5.170
Original Data	5.002			

The second regression dataset used was the music year prediction dataset [13] to demonstrate the capacity of the distance correlation encoder to handle datasets of very large scale. The dataset contained 90 covariates and 463,715 training samples and the goal was to predict the year of song release. All features were scaled to 0 mean and unit variance to prevent reconstruction error from dominating overall objective function. Linear regression was used to predict each song's year of release and the RMS errors are presented below. The autoencoder comprised of 4 layers for encoder and decoder with 120 nodes using hyperbolic tan activation function.

V. Alternative Configurations

The flexibility of autoencoders also allows different neural networks to be used for encoder and decoder. This is valuable in the scenarios where the input data points are not fixed

TABLE II: Music Year Dataset. Regression errors reported as averaged RMS.

	Dimensions		
Methods[1]	**10**	**20**	**40**
dCor	8.875	8.86	8.86
PCA	10.50	10.30	10.15
Original Data	———— 9.51 ————		

length vectors and represent a great majority of present challenges involving these datasets. For example, imaging datasets pose common problems for machine learning researchers in image recognition, classification, segmentation and they are commonly represented as 3D tensors. Similarly, text data is very common given the vast amount of text data generated by the Internet but processing and using text data presents a whole field of problems on its own. Numerous advances have been made to produce reasonable vectors from variable length texts such as document vectors, word vectors and other methods. Hence two additional autoencoder structures were proposed and tested in this paper.

A. Stanford IMDB Dataset

The Stanford IMDB dataset was used for testing purposes. In this case, there are 100,000 movie reviews and 50,000 are labeled as positive or negative according to their sentiment; half of the labeled 50,000 samples were used as testing set. Each review is composed of one or more unprocessed sentence and the task at hand is to predict the sentiment of a movie review given its text. Several different approaches were attempted. The first approach is *document vectors*.

1) Document Vectors: The idea behind document vector [14] is very similar to the original word2vec implementation to encode words into fixed length vectors. Word2vec had useful geometric properties such the Euclidean distance between word vectors being mostly dependent on their semantic meaning; similar words would have low Euclidean distance and vice versa for dissimilar words. A useful property arising from this is the ability to add and subtract word vectors in a semantic fashion to approximate existing word vectors. For example, $v(\text{king}) - v(\text{man}) + v(\text{women}) \approx v(\text{queen})$. This also implies subtle relationships between word vectors such as Beijing is to China what Washington is to the USA. These properties are highly useful for various text-related machine learning problems such as semantic analysis.

There were two main methods for training these word vectors, Skip-gram [15] and Continuous Bag-of-Words Model (CBOW) [16]. The first approach builds a model where word vectors are trained to predict the surrounding words. The second builds a model where multiple word vectors are summed or concatenated to predict a word. Typically in both cases, the word or words to predict is within a certain window of context in existing sentences and the word vectors are updated to best predict the correct word or surrounding words.

[1]We do not compare with supervised dimension techniques given the large sample size ($n > 10,000$). Most techniques (e.g., gKDR) had exponentially increasing memory requirements and required $>$ 100GB of memory to run.

Document vectors extend these two methods in the following way. Instead of using word vectors to predict the surrounding words, each document has a unique vector associated that is used as input to predict random words in the document. This unique vector is updated to provide better predictions during the training process. Alternatively, for the other method, a unique document vector and several word vectors are combined to predict random words in the document. The document vector provides the context in addition to existing words that increase the accuracy of the prediction and is updated through training. This should increase the predictive power of these document vectors and therefore encourage them to provide better "context" that encapsulates the overall meaning of the document. Any new document vectors, in either case, are added to the original trained and now fixed classifier for prediction and trained accordingly.

Using these document vectors, we can directly apply the distance correlation autoencoder. The document vectors were trained on the entire text dataset, training, test and unlabeled, using the distributed bag of words approach. The hidden dimensions selected was 200. The structure of the autoencoder was 4 layers, 500 nodes per layer for both the encoder and decoder and all activation functions were hyperbolic tan.

As we can see, distance correlation autoencoder achieved remarkably close results to using the full 200 dimensions with just 5 dimensions. In comparison, PCA required consistently more dimensions to achieve similar accuracy and indicates dCor was capable of extracting more relevant features for classification.

TABLE III: IMDB Movie Review Comparison

	Dimensions			
Methods	**5**	**10**	**20**	**40**
dCor	88.62%	88.67%	88.79%	88.66%
PCA	76.08%	86.06%	87.77%	88.29%
Original Data	———— 88.92% ————			

2) LSTM Autoencoder: An alternative structure to directly inputting sequence of words for each review is LSTM neural networks [17]. To save on computational requirements, 300 dimension word vectors from the Glove dataset [18] were used for input and the output of the LSTM decoder was similarly set to be 300 dimension word vectors. For a movie review, its words were converted to the corresponding word vector from the Glove dataset and inputted into the LSTM encoder up to a maximum of 300 words. Note that stop words and words not found in the Glove dataset were removed. After inputting the words of the movie review, the final output would be sent to a latent layer whose nodes corresponded to the desired dimensions. This latent representation would then repeatedly be sent to the decoder to reproduce the inputted word vectors. TSNE plot [19] for 20 dimension representation for random 2000 data points in test dataset is presented in Figure 5 along with the sample distance correlation calculated in Table IV.

TABLE IV: Accuracy performance using K-Nearest Neighbour classifier ($k = 5$)

	Dimensions		
Methods	**5**	**10**	**20**
LSTM dCor	82.21%	86.82%	87.26%

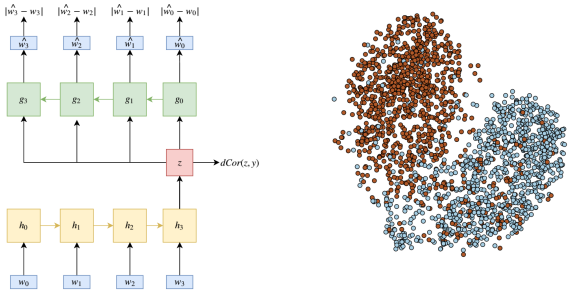


Fig. 5: **left:** 100 nodes, 1 layer for encoder and decoder, **right:** TSNE Plot (0.6122)

Results in general look good and could be improved further through more optimization. The TSNE plot also indicates that the two classes of points were properly separated.

B. Convolutional Autoencoders

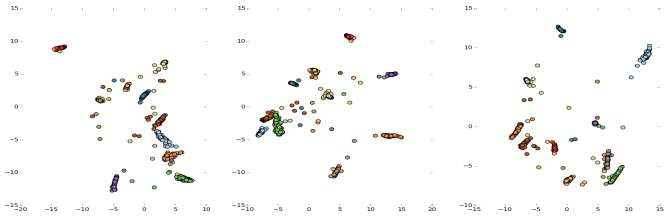
Another possible network structure for distance correlation autoencoders is to use convolutional layers. This provides a natural representation for images with multiple colour channels but also leverages the powerful feature extracting capabilities of convolutional networks. Previous experiments [20] showed that convolutional networks learn to extract different important features for classification or other machine learning problems during training and it should similarly hold here.

1) MNIST: The MNIST dataset [21] was chosen and the first configuration uses two deep feed-forward neural networks for the encoder and decoder. In this case, rectified linear activation [22] was found to perform well while linear activation functions from encoder to latent space yielded poor results. There were 4 layers for both the encoder and the decoder with 1000 nodes each since previous testing indicated that the autoencoder performs better if data is transformed to higher dimensions first. Dropout was also used to minimize overfitting. The low dimension representation was set at 40 features to represent approximately 20 fold reduction in dimensions. Only minimal preprocessing to scale vector values to 0-1 was done and the entire dataset of 60,000 training samples was used. The remaining 10,000 test samples were used for validation purposes. The results for different dimensions of the latent representation were plotted below for test points along with TSNE [19] for all 40 dimensions. A side by side comparison of results without dCor maximization on the same autoencoder structure was also shown.

Additionally, a convolutional architecture was also used with 4 convolutional layers and one max pooling layer for the encoder. Linear activations were used to connect the encoder to the latent layer and 3 feed-forward layers were used for the decoder. This design decision was inspired by the fact that reconstruction was not the main purpose and feature extraction from encoder to latent was much more important. The decoder was trained to minimize mean squared error with the flattened original images in 1 dimension. TSNE plot for 40 dimension representation was also plotted and accuracy results were compared with PCA and feed-forward dCor autoencoder using k-nearest neighbour ($k = 5$) as the classifier. Another model was also used as comparison where the same convolutional structure is used but the latent representation is trained to classify digits rather than minimize distance correlation.

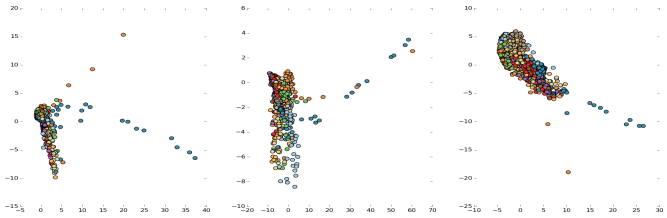
The results based on the plots in Figure 6 were quite good and it is clear that the points representing different classes are neatly separated and tightly clustered. Furthermore, overfitting was not an issue and test points were mostly correctly placed in the same clusters. There were also significant differences from the base autoencoder in terms of latent representation generated and it indicates distance correlation maximization indeed has a significant impact on the dimension reduction training process.

Autoencoder with dCor maximization



1st and 2nd dim. 3rd and 4th dim. 5th and 6th dim.

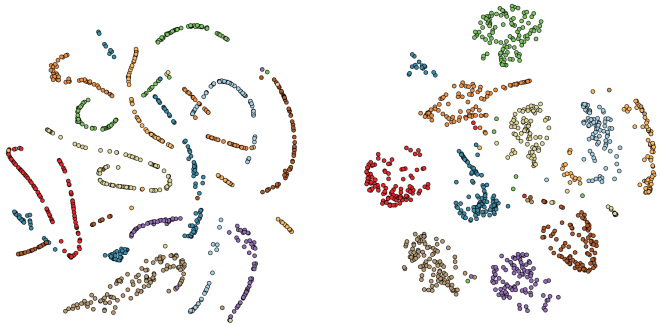
Autoencoder without dCor maximization



1st and 2nd dim. 3rd and 4th dim. 5th and 6th dim.

Fig. 6: Only 1,000 points from the test set were plotted but clear clustering can be observed. There is a significant difference if dCor was not used as an objective function.

A summary of results comparing different variants of distance correlation is presented in Table V. All values reported are accuracy for the entire test set using k-nearest neighbor ($k = 5$). As shown, dCor autoencoder in both cases obtained high accuracy even without directly training with a classification objective. The convolutional dCor results excluding the extreme case of 5 dimensions, had less than 1% error rate even with very high compression in the case of 10 dimensions. It also steadily achieved better results than using a normal convolutional network trained to classify as a feature extractor.



Feed-forward Convolutional (0.986)

Fig. 7: The two TSNE plots. Only 1,000 plots were plotted in total from the test set. The two plots differed greatly but it is clearly seen that different classes are separated. The convolutional version exhibits better clustering and visualization though and matches its increased accuracy with the k-nearest neighbor. Sample distance correlation for test dataset in brackets for convolutional version.

TABLE V: MNIST Accuracy

	Dimensions				
Methods	**5**	**10**	**20**	**40**	**100**
dCor Convolutional	96.27%	99.25%	99.24%	99.27%	99.18%
dCor Feed-forward	83.50%	98.15%	98.29%	97.89%	98.08%
Conv Feature Extractor	99.03%	99.05%	99.12%	99.29%	99.11%

VI. Conclusion

In this essay, we analyzed the results of an autoencoder built on maximizing the distance correlation of the latent representation with the label or regressor Y. We showed that this was both feasible and performed excellently compared to alternative algorithms but also that it was easily scalable to high dimension and large datasets such as MNIST or CIFAR. This structure was capable of leveraging distance correlation as an effective measure of statistical dependency that results in a good representation and the high capacity of nonlinear autoencoders to find this representation. We showed that this representation was better than many other algorithms and could handle datasets of an entirely different magnitude than what was previously tested for supervised dimension reduction techniques while achieving excellent performance even with simple classifiers.

It was also possible to use convolutional layers to increase the quality of the representation generated by the distance correlation autoencoder and in the future, results could be further improved by building better feature extracting encoders. Different optimization or regularization methods could also be attempted to find even better mappings.

Finally, by developing a successful model for supervised dimension reduction that could scale to large datasets in an effective manner, it would be interesting to see if these representations could be used with non-neural network models to compete with neural network models in many tasks where previously a large gap in performance existed. There also exists the possibility of extending this framework to semi-supervised scenarios to leverage large amounts of unlabeled data to improve performance.

References

[1] K.-C. Li, "Sliced inverse regression for dimension reduction," *Journal of the American Statistical Association*, vol. 86, no. 414, pp. 316–327, 1991.

[2] R. D. Cook and L. Forzani, "Likelihood-based sufficient dimension reduction," *Journal of the American Statistical Association*, vol. 104, no. 485, pp. 197–208, 2009.

[3] K. Fukumizu and C. Leng, "Gradient-based kernel dimension reduction for regression," *Journal of the American Statistical Association*, vol. 109, no. 505, pp. 359–370, 2014.

[4] P. Vepakomma, C. Tonde, and A. Elgammal, "Supervised Dimensionality Reduction via Distance Correlation Maximization," *ArXiv e-prints*, Jan. 2016.

[5] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Z. Jahromi, "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds," *Pattern Recognition*, vol. 44, no. 7, pp. 1357–1371, 2011.

[6] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *The Annals of Statistics*, vol. 35, no. 6, p. 27692794, 2007.

[7] M. Clark, "A comparison of correlation measures."

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Artificial Intelligence and Statistics*, 2015, pp. 192–204.

[10] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[11] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[12] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *ArXiv e-prints*, Dec. 2013.

[14] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents." in *ICML*, vol. 14, 2014, pp. 1188–1196.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[18] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[19] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[20] L. Hertel, E. Barth, T. Käster, and T. Martinetz, "Deep convolutional neural networks as generic feature extractors," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–4.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, p. 22782324, 1998.

[22] R. H. Hahnloser, R. Sarpeshkar, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000.