MTL782 Assignment 2
A Review of Methods for Dimensionality Reduction

Abhinava Sikdar 2017MT10724
Sumanth Varambally 2017MT60855
Yashank Singh 2017MT10756

September 24, 2020

1 Introduction

Machine Learning algorithms have shown great success in recent years. They are used in a myriad of applications such as facial recognition, machine translation, voice recognition, self-driving cars and stock-price prediction to name a few. It is interesting to note that in most of these applications, the data input to the algorithm is of high-dimensionality. For example, a small 128×128 coloured image, when used as an input to a machine-learning algorithm, would have a dimension of close to 50000. Similarly, an audio recording of 3 seconds (with a sampling rate of 44.1kHz) would produce approximately 130,000 features. Except for a few specialized variants of neural networks (Convolutional Neural Networks for images and Recurrent Neural Networks for audio), most other machine learning algorithms would perform poorly when used with such high number of dimensions. Thus, it is imperative that we obtain a smaller representation of the data that retains important information before being used with a machine-learning algorithm.

Dimensionality reduction refers to mapping the D-dimensional feature space to a subspace of typically lower dimension, say M where $M < D$ such that the mapped or projected data preserves a set of desired intrinsic properties such as variance, class separability, etc. Algorithms for this purpose can be broadly classified as feature selection and feature extraction methods. In Section 2, we provide a brief outline of feature selection methods followed by detailed explanations of feature extraction methods. In Sections 3 upto 9, we elaborate upon various popular feature projection/extraction methods. These include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), t-Distributed Stochastic Neighbour Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP), Non-negative Matrix Factorization (NMF) and Autoencoders.

1.1 The curse of dimensionality

The term "curse of dimensionality" was first coined by Richard E. Bellmann while working on problems in dynamic programming [3]. It refers to a phenomenon encountered while working with problems in machine learning, data mining, vision, bioinformatics, combinatorics, etc. In summary it says that as the number of dimensions increase, the volume of the feature space increases so fast that data becomes increasingly sparse. Hence, for obtaining any statistically significant or reliable result, an exponentially increasing number of data points is required. Additionally, common distance metrics perform poorly or fail in capturing the similarity or closeness of any two data points due to the apparent 'dissimilarity' and 'sparseness' of data points in higher dimensions. This leads to poor and unreliable performance in machine learning, clustering and anomaly detection [31] and highlights the need for methods for dimensionality reduction.

2 Feature Selection

Feature selection refers to the selection of a subset of features available in the dataset. The benefits include easy interpretation and inference, smaller training times, better generalisation and of course, avoidance of the 'curse of dimensionality' through dimensionality reduction. The main intuition behind this is that the dataset may contain irrelevant or redundant features which are not necessary for capturing underlying properties of the data.

2.1 Supervised methods

This class of feature selection algorithms use the target variable for the given classification or regression problem to determine a non-redundant, relevant subset of the features.

2.1.1 Filter methods

They select features without taking into consideration the details of the model. This selection of features is entirely based on the correlation between the features and the target variable. The 'least interesting' variables i.e. the ones with least absolute correlation are suppressed. Information gain, χ^2-test, Fisher score, Pearson's correlation coefficient and variance threshold are some of the examples.

2.1.2 Wrapper methods

These evaluate all the possible subsets of features. High computational time and proneness to overfitting when number of samples are low are their major disadvantages but unlike the filter methods, they are able to capture the relationship between different features. These include algorithms such as recursive feature elimination, sequential feature selection and genetic algorithms.

2.1.3 Embedded methods

These are a class of algorithms which try to combine the previous two so as to harness the merits of both. They are embedded in the models and specific to them. This is often seen as L_1 regularisation in Lasso regression and random forests with regularisation costs.

2.2 Unsupervised methods

These can be again classified as filter, wrapper and embedded methods. While they are similar in almost all attributes to their supervised counterparts, they are not guided by their performance with respect to a target variable. Instead they are guided through performance in clustering algorithms. They also have advantages over their supervised counterparts such as they are unbiased and perform well even in the absence of any prior knowledge and that they can reduce the risk of overfitting [25].

3 Principal Component Analysis

Principal component analysis(PCA), also known as *Karhunen-Loeve* transformation is one of the most popular algorithms used for dimensionality reduction. It is also used for other purposes such as lossy data compression, feature extraction and data visualization. There are broadly two working definitions of the same which lead to the same algorithm.

3.1 Maximizing the variance

As first envisaged in [12], PCA can be seen to be the orthogonal projection of the data onto a lower dimensional space with the objective function of maximising the data variance in this lower dimensional space. This lower dimensional space is known as the *principal subspace*. Consider a set of data points $\mathbf{x}_n$ where $n = 1, \dots, N$ lying in a D dimensional Euclidean space. The aim is to project this data onto $\mathbb{R}^M$ where $M < D$, while maximising variance. Let us first choose $M = 1$. This subspace can be defined using a $D-$dimensional vector, say $\mathbf{u}_1$ which WLOG, can be taken to be a unit vector. Then the projection of each data point $\mathbf{x}_n$ onto this subspace is given by $\mathbf{u}_1^T\mathbf{x}_n$. Hence the variance of this projected data set is given by

$$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 = \frac{1}{N}\sum_{n=1}^{N}\{\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}}\}$$

where $\bar{\mathbf{x}}$ represents the data mean and $\mathbf{S}$ is the data covariance matrix given by

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$

Hence the problem can be cast as the following optimization problem

$$\max_{\mathbf{u}_1} \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 \text{ s.t. } ||\mathbf{u}_1|| = 1 \tag{1}$$

where the constrain on norm of $\mathbf{u}_1$ is to prevent the objective function from blowing up to ∞ by taking a $\mathbf{u}_1$ such that $||\mathbf{u}_1|| \to \infty$. By introducing the Lagrange relaxation of the problem and performing an unconstrained optimization by taking derivative of

$$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1)$$

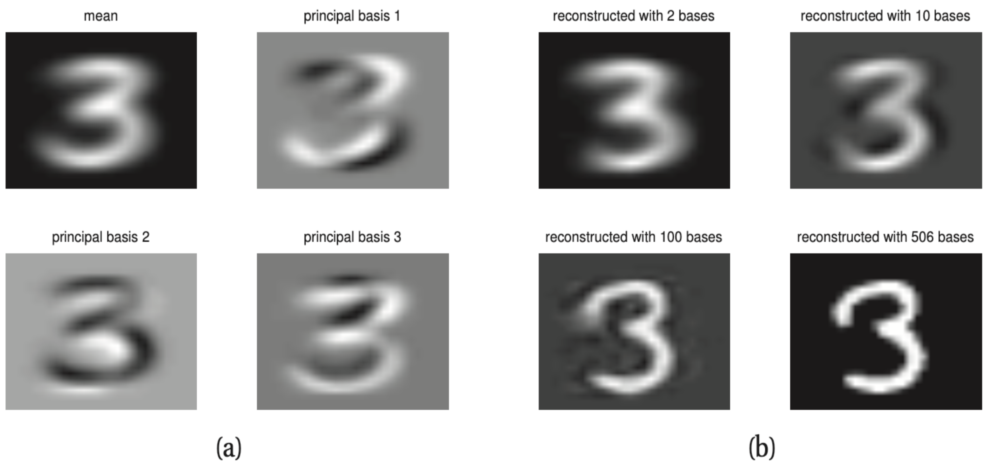
mean
principal basis 1
reconstructed with 2 bases
reconstructed with 10 bases
principal basis 2
principal basis 3
reconstructed with 100 bases
reconstructed with 506 bases
(a)
(b)

Figure 1: (a) shows the mean of the data and the first three eigendigits based on 25 images of digit 3 from MNIST dataset. (b) shows reconstruction of an image based on 2, 10, 100 and all the eigendigits (source: [19])

with respect to $\mathbf{u}_1$ where λ_1 is the Lagrange multiplier, we get that

$$\begin{aligned} \mathbf{S}\mathbf{u}_1 &= \lambda_1 \mathbf{u}_1 \\ \mathbf{u}_1^T \mathbf{S}\mathbf{u}_1 &= \lambda_1 \end{aligned} \tag{2}$$

Clearly, Eqn.(1) is maximised for the largest value of λ_1 which from Eqn,(2), is the largest eigenvalue of the covariance matrix $\mathbf{S}$ and hence, $\mathbf{u}_1$ is the eigenvector corresponding to λ_1. Hence for projection onto a $M-$dimensional subspace, we can project the data onto the subspace formed by $\mathbf{u}_1, \ldots, \mathbf{u}_M$ which are the corresponding eigenvectors to the M largest eigenvalues $\lambda_1, \ldots, \lambda_M$ of the data covariance matrix $\mathbf{S}$.

3.2 Minimizing projection cost

An alternate formulation of the same was given in [8] in which the problem was formulated as the minimization of the mean squared distance between the original and projected data. Consider $\mathbf{u}_1, \ldots, \mathbf{u}_D$ to be an orthonormal basis of the $D-$dimensional space in which the data lies. Then any data point $\mathbf{x}_n$ can be written as

$$\mathbf{x}_n = \sum_{i=1}^{D} (\mathbf{x}_n^T \mathbf{u}_i)\mathbf{u}_i \tag{3}$$

However, our goal is to choose M of the basis vectors and constants z_{ni} and b_i such that for approximation of each data point $\tilde{\mathbf{x}}_n$

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i \mathbf{u}_i \tag{4}$$

the sum of squared distance between the original data point and its approximation over the entire data set

$$J = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \tilde{\mathbf{x}}_n||^2 \tag{5}$$

is minimized. Minimizing wrt z_{ni} by substituting Eqn.(4) in Eqn.(5) and putting the derivative equal to zero and doing the same for b_i, we get that

$$z_{ni} = \mathbf{x}_n^T\mathbf{u}_i \text{ and } b_i = \bar{\mathbf{x}}^T\mathbf{u}_i$$

which when substituted into Eqn.(3) and Eqn.(5), gives us

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D} (\mathbf{x}_n^T\mathbf{u}_i - \bar{\mathbf{x}}^T\mathbf{u}_i)^2 = \sum_{i=M+1}^{D} \mathbf{u}_i^T\mathbf{S}\mathbf{u}_i = \sum_{i=M+1}^{D} \lambda_i \tag{6}$$

The last equality can be derived like in Section 2.2.1. Hence to minimise Eqn.(5), it can be seen from Eqn.(6) that $\mathbf{u}_i, i = M+1, \dots, D$ need to be the eigenvectors corresponding to the smallest eigenvalues $\lambda_i, i = M+1, \dots, D$ of the data covariance matrix $\mathbf{S}$. This brings us to the same conclusion as in the variance maximization formulation that $\mathbf{u}_i, i = 1\dots, M$ need to be the eigenvectors corresponding to the M largest eigenvalues of $\mathbf{S}$.

4 Kernel PCA

Since the objective of the standard PCA is to find the matrix $\mathbf{U} \in \mathbb{R}^{D\times M}$ and transform the data matrix $\mathbf{X}$ as $\tilde{\mathbf{X}} = \mathbf{U}^T\mathbf{X}$, it is a method for linear dimensionality reduction. However it is possible that the data has more complicated structures which cannot be captured well in a linear subspace. To overcome this, the 'kernel trick' was incorporated into the PCA paradigm in [23].
Assume that we have a nonlinear transformation $\phi : \mathbb{R}^D \longrightarrow \mathbb{R}^K$ where typically $K >> D$. Transforming each data point $\mathbf{x}$ through ϕ into such a high dimensional space and then performing PCA would be extremely costly. However we use the calculations in [24] to simplify the task. We assume that the projected data into to K dimensional space is centred. Then, the covariance matrix is given by

$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N} \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T$$

Hence, the eigenvectors and eigenvalues are given as $\mathbf{S}\mathbf{u}_k = \lambda_k\mathbf{u}_k$ where $k = 1, \dots, K$. From these, we can write

$$\frac{1}{N}\sum_{i=1}^{N} \phi(\mathbf{x}_i)\{\phi(\mathbf{x}_i)^T\mathbf{u}_k\} = \lambda_k\mathbf{u}_k$$

and that $\mathbf{u}_k = \sum_{i=1}^{N} a_{ki}\phi(\mathbf{x}_i)$. By defining the kernel function $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x})_i^T\phi(\mathbf{x}_j)$ and simplifying as in [28], we get that

$$\tilde{\mathbf{x}}_k = \phi(\mathbf{x})^T\mathbf{u}_k = \sum_{i=1}^{N} a_{ki}\mathcal{K}(\mathbf{x}, \mathbf{x}_i) \tag{7}$$

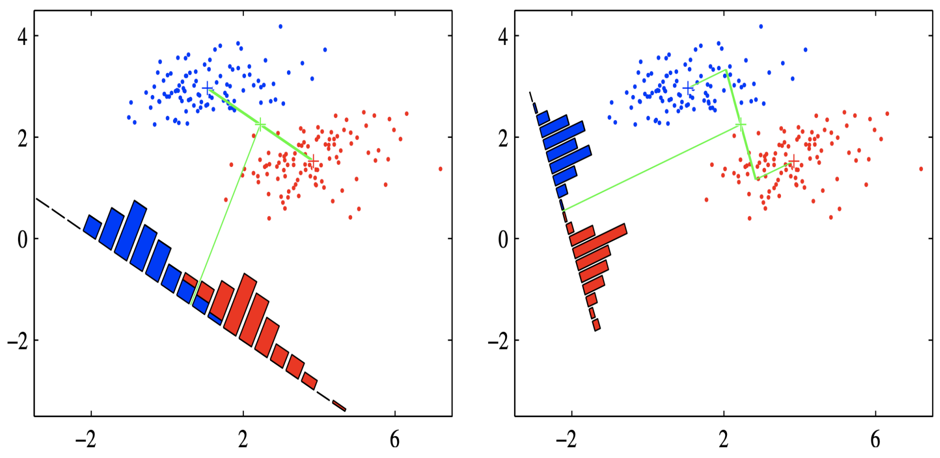

Figure 2: On left there is considerable class overlap when projecting along the difference of means. On right is the projection based on Fisher's LDA (source: [4])

where $\mathbf{K}\mathbf{a}_k = \lambda_k N \mathbf{a}_k$ and $\mathbf{K}_{i,j} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$. Note that if the projected data is not centred contrary to our assumption, we can substitute $\mathbf{K}$ with the gram matrix $\tilde{\mathbf{K}}$ [4]. This gram matrix is given by $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$ where $\mathbf{1}_N$ is a $N \times N$ matrix with all elements equal to $1/N$. We can choose to project our data onto a subspace of lower dimension by using Eqn.(7) and limiting the number of kernel principal components by taking only the ones corresponding to the largest eigenvalues λ_k.
The utility of kernel PCA is that without explicitly projecting out data onto a higher dimensional space, we were able to obtain the projections onto the principal components in the $\phi-$space. This when restricted by choosing only M components out of K corresponding to the largest eigenvalues λ_k gives us a non linear projection of our data onto a M dimensional subspace while preserving maximum variance in the $\phi-$space.

5 Linear Discriminant Analysis

Linear discriminant analysis(LDA) or Fisher's linear discriminant is a classification technique based on dimensionality reduction. It uses a linear transformation to project the high dimensional data into a lower dimensional subspace. Projection of 2D data onto a line is done through $y = \mathbf{w}^T\mathbf{x}$, whereas in general, for projection onto $D > 1$ dimensions, D linear "features" $y_k = \mathbf{w}_k^T\mathbf{x}$ are introduced and combinely written as $\mathbf{y} = \mathbf{W}^T\mathbf{x}$ where $\{\mathbf{w}_k\}$ are the columns of $\mathbf{W}$.

LDA determines $\mathbf{W}$ such that the degree of separability of different classes in higher dimensions is preserved as well as possible after the projection. This is done by determining the objective function, the optimization of which would lead to *i)* minimization of *within-class* variance and *ii)* maximization of *between-class* variation. Consider K classes with their mean and variances given by

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \text{ and } \mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T$$

Hence, the total *within-class* variance is given by $\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k$. From [6], we can write the total covariance matrix $\mathbf{S}_T = \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m})(\mathbf{x}_n - \mathbf{m})^T$ as the sum $\mathbf{S}_W + \mathbf{S}_B$ where $\mathbf{S}_B$ is a measure of *between-class* variance. This gives us that

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$

Defining the corresponding $\mathbf{s}_W$ and $\mathbf{s}_B$ in the projection space of D dimensions, we can use a number of objective functions as defined in [9] which minimize $\mathbf{s}_W$ and maximize $\mathbf{s}_B$. One example is that of

$$J(\mathbf{W}) = \text{Tr}\{\mathbf{s}_W^{-1} \mathbf{s}_B\}$$

which can be written explicitly in terms of $\mathbf{W}$ as

$$J(\mathbf{W}) = \text{Tr}\{(\mathbf{W}\mathbf{S}_W\mathbf{W}^T)^{-1}(\mathbf{W}\mathbf{S}_B\mathbf{W}^T)\}$$

which on maximization, as done in [9] give that $\mathbf{w}_k, k = 1, \ldots, D$ are the eigenvectors corresponding to the D largest eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B$.

6 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) first proposed in [26] is an unsupervised, non-linear dimensionality reduction technique primarily used for data exploration and visualizing high-dimensional data into a low dimensional space of 2 or 3 dimensions.While PCA focuses on preserving large distances or overall variance, t-SNE is concerned with preserving pair-wise local similarities that allows it to capture non-linear structure. The algorithm can be broken down into three main stages explained below.

6.1 Stage I

In the first stage, t-SNE constructs a probability distribution over pairs of high-dimensional data points that dictates similarity between neighbouring points. The distribution is defined in such a way that higher the similarity between a pair of points, higher the probability given by the distribution. In simple terms, suppose we pick a data point $\mathbf{x}_i$, now to define the probability of picking a neighbour $\mathbf{x}_j$ we assume a Gaussian distribution centred at $\mathbf{x}_i$ with parameters local to $\mathbf{x}_i$. The similarity is then defined as the density of the Gaussian distribution at the point $\mathbf{x}_j$. These similarities are then summed over all data points for normalization to define a probability distribution.

Formally, given a set $\mathbf{x}_i, \ldots, \mathbf{x}_N$ of N data points lying in D dimensions, t-SNE first computes the conditional probability density of choosing $\mathbf{x}_j$ given $\mathbf{x}_i$ as

$$P_{j|i} = \frac{\exp(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(\frac{||\mathbf{x}_i - \mathbf{x}_k||^2}{2\sigma_i^2})} \tag{8}$$

The similarity between data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is then defined as

$$P_{ij} = \frac{P_{j|i} + P_{i|j}}{2N} \tag{9}$$

Here we can see that in Eqn.(9),$P_{ji} = P_{ij}$, $P_{ii} = 0$ and $\sum_{i,j} P_{ij} = 1$. However, note that in Eqn.(8), $P_{j|i} \neq P_{i|j}$ as σ_i is not necessarily equal to σ_j. The spread of the Gaussian kernels σ_i for a point $\mathbf{x}_i$ is set in such a way that the perplexity(a measure of how well a probability model predicts a sample) of the conditional distribution equals a predefined perplexity calculated using the bisection method.This is done to ensure that no single point wields a disproportionate influence. As a result, σ is tuned to the density of the data i.e. smaller values of σ in denser parts of the manifold and larger values of σ in the sparser parts of the manifold.

6.2 Stage II

The second stage is similar to the first stage in constructing a probability distribution on a pair of points, however in this stage the points are the low-dimensional projections $\mathbf{y}_i$ of the high dimensional data points $\mathbf{x}_i$. However, in defining the conditional probability distribution in the lower dimensional space, the Gaussian kernel is not used. This is because of the "crowding problem" addressed in [26] as:"the area of the two-dimensional map that is available to accommodate moderately distant data points will not be nearly large enough compared with the area available to accommodate nearby data points". The Gaussian distribution has a short tail which causes points to be squashed together. To ensure even spreading of points in the lower dimensional space, t-SNE uses the student-t distribution with one degree of freedom(a.k.a Cauchy Distribution) that has a heavy tail which allows dissimilar objects to be modeled far apart in the map.
Formally, let $\mathbf{y}_1, \ldots, \mathbf{y}_N$ be the embeddings of $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in the lower L dimensional space, the similarity between the projected points $\mathbf{y}_i$ and $\mathbf{y}_j$ is then defined as

$$Q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathbf{y}_i - \mathbf{y}_k||^2)^{-1}} \tag{10}$$

We set $Q_{ii} = 0$, also note that in Eqn.(10), $Q_{ji} = Q_{ij}$ and $\sum_{i,j} Q_{ij} = 1$.

6.3 Stage III

In the last step the low dimensional embeddings are computed by minimizing the KL divergence [15] of the distribution P from the distribution Q with respect to the embeddings $\mathbf{y}_i$ $\forall i = 1, \ldots, N$ i.e.

$$J = \text{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$\min_{\mathbf{y}_1,\dots,\mathbf{y}_N} \mathrm{KL}(P||Q)$$

This can be solved using gradient descent, differentiating the cost function w.r.t $\mathbf{y}_i$ gives

$$\frac{\delta J}{\delta \mathbf{y}_i} = 4\sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}$$

The optimization process has it's own challenges. If the lower dimensional embeddings are unconstrained to move around freely, they may get stuck in a local minima in the early stages. To prevent this t-SNE uses *early compression* which adds L_2 regularizer in early stages. t-SNE also uses *early exaggeration* where all the P_{ij}'s are multiplied at the early stages that makes Q_{ij}'s to focus more on larger P_{ij}'s (i.e. closer points), making early clusters more tightly knit, allowing them to move around more easily without getting in each other's ways.

6.4 Limitations

While t-SNE has been much celebrated for data visualisation, it has not triumphed in dimensionality reduction. t-SNE has quadratic space and time complexity on the number of data points due to pairwise calculations of probabilities, which makes is unsuitable for data sets with over 10K data points. Moreover, the cost function of t-SNE is non-convex, hence when the intrinsic dimension is higher, the optimization often gets stuck in local minima and different runs of the algorithm produce different results which makes it non-deterministic in terms of optimality of projections. Although t-SNE is a non-linear projection algorithm it still makes a strong assumption about the local manifold structure being linear. This is a significant assumption as the distance between neighbouring points is measured in Euclidean space which assumes linearity. This makes t-SNE infeasible for manifolds where even local structure in non-linear. In such cases Auto-encoders may come out to be a better choice.

7 UMAP

UMAP or Uniform Manifold Approximation and Projection for Dimension Reduction is another algorithm for dimensionality reduction that was proposed in [18]. It gained popularity for being faster than t-SNE and accurate in finding lower dimensional embeddings. Moreover, the paper [18] provided sound theoretical justification leveraging results from topology, Riemannian geometry and category theory. The algorithm first aims to learn the approximate topology of the higher dimensional space or manifold by representing it as a finite open cover. It does so by taking a combination of simplical complexes as the finite open cover, oversimplified to our use case which are essentially k-nearest neighborhood graphs with nodes and edges. It then finds a lower dimensional representation with a similar a topological representation as the higher dimensional manifold minimizing the cross entropy given in Eqn.(11) over the weights of the graph constructed in the lower dimension. Formally let $\mathbf{x}_1, \dots, \mathbf{x}_N$ be our data points in the high dimensional space. Then for each point, we find it's k nearest neighbors. Let σ_i be the diameter of the neighborhood of $\mathbf{x}_i$ and r_i be the distance to the nearest neighbor. Then a weighted graph is constructed as follows

$$w_i(\mathbf{x}_i, \mathbf{x}_j) = \exp(\frac{-d(\mathbf{x}_i, \mathbf{x}_j) - r_i}{\sigma_i})$$

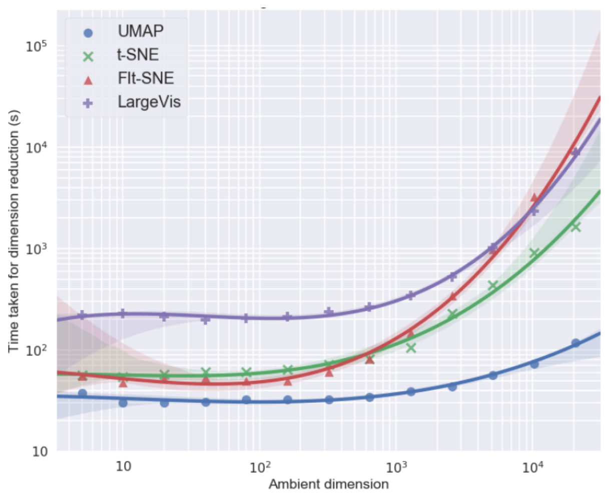

Figure 3: Runtime performance scaling of UMAP, t-SNE, FIt-SNE and Largevis w.r.t. ambient dimension of the data.

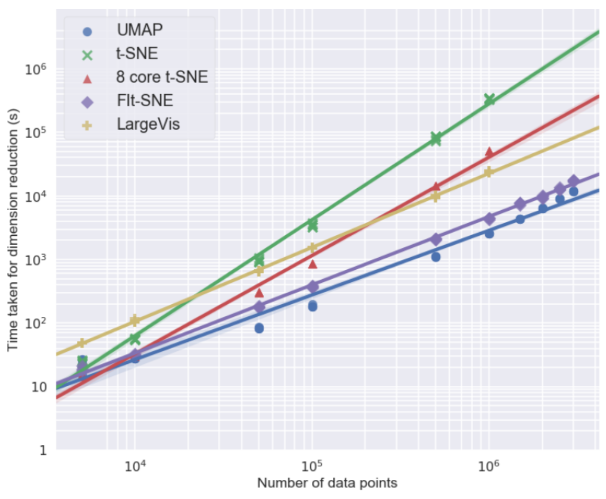

Figure 4: Runtime performance scaling of t-SNE and UMAP on various sized subsamples of the full Google News dataset.

This is not symmetric and hence is symmetrized as follows

$$w(\mathbf{x}_i, \mathbf{x}_j) = w_i(\mathbf{x}_i, \mathbf{x}_j) + w_j(\mathbf{x}_j, \mathbf{x}_i) - w_i(\mathbf{x}_i, \mathbf{x}_j) w_j(\mathbf{x}_j, \mathbf{x}_i)$$

Then the cross entropy to be minimized is given by

$$C(w, w') = \sum_{\tilde{i}j} (w(i,j) \log(\frac{w(i,j)}{w'(i,j)} + (1 - w(i,j)) \log(\frac{w(i,j)}{w'(i,j)}))) \tag{11}$$

In Eqn.(11) w are the weights computed from the data points and w' are the weights computed from our low-dimensional embedding that are being optimised.

7.1 Comparison with t-SNE

UMAP has become widely popular because of it's capability to embed very complex manifolds and being faster than other methods with such embedding capacity. UMAP usually gives better results than t-SNE as t-SNE constructs the lower dimensional embeddings only giving importance to the local structure of the manifold, however UMAP preserves the global structure as well. Another shortcoming of t-SNE is that it is slow and doesn't scale well, however UMAP is is faster and scalable to millions of data points. UMAP also does not require PCA pre-processing unlike t-SNE and can be run on raw data. The following results(Fig. 3, 4, 5) were shown in [18].

8 Non-negative Matrix Factorization

Non-negative Matrix Factorization [29] continues on the theme of obtaining a simpler representation for the data in terms of a basis. This method is one in the class of matrix factor

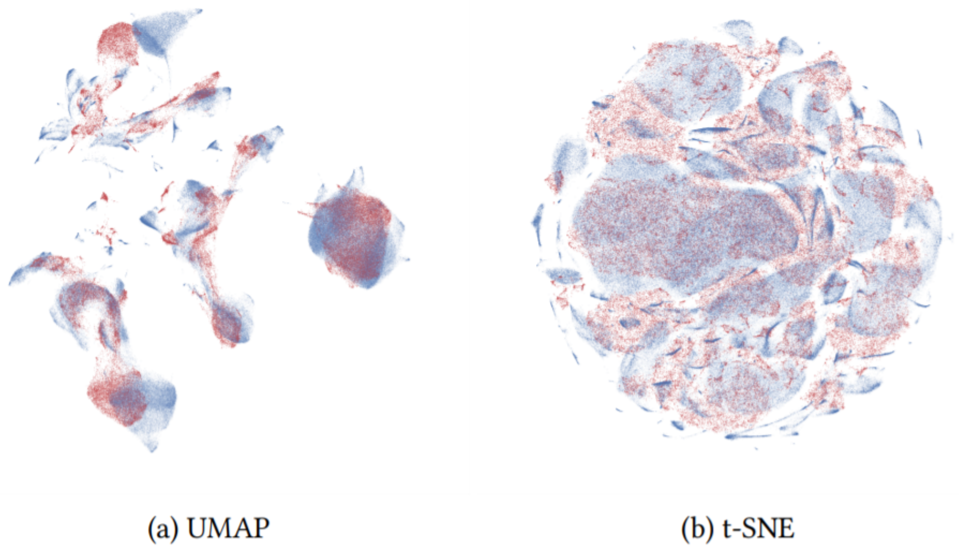

Figure 5: Procrustes based alignment of a 10% subsample (red) against the full dataset (blue) for the flow cytometry dataset for both UMAP and t-SNE. (source: [18])

methods which aim to decompose the data matrix into two factor matrices such that one matrix denotes a collection of basis vectors, while the other matrix contains coefficient vectors for each data example in terms of the obtained basis. Mathematically, given the data matrix X of dimensions $p \times n$, where p is the dimensionality of the data and n is the number of data points, we write

$$X \approx WH$$

where W is a basis matrix of size $p \times r$ and H is the coefficient matrix of size $r \times n$. We note that even the previously discussed methods of PCA and LDA can be interpreted as matrix factor methods. Some other matrix factor methods include K-SVD [1] and ICA [13]. The main difference between all the five above mentioned methods boils down to the constraints imposed upon the matrices U and V. While PCA imposes orthogonality between vectors of W, LDA aims to promote seperability amongst the classes. K-SVD promotes sparsity in the coefficient vectors, while ICA aims to learn statistically independent components from the data. The main distinguishing difference of NMF from the above methods is the non-negativity constraint applied to the matrices. Since the other methods are not widely used for dimensionality reduction, we restrict this discussion to NMF.

As noted in [10], the reason why NMF is preferred and has become popular is due to its capability to automatically extract sparse and easily interpretable factors. Due to the non-negative constaint, the factors obtained are more suitable for many real-world applications as we discuss in Section 8.2.

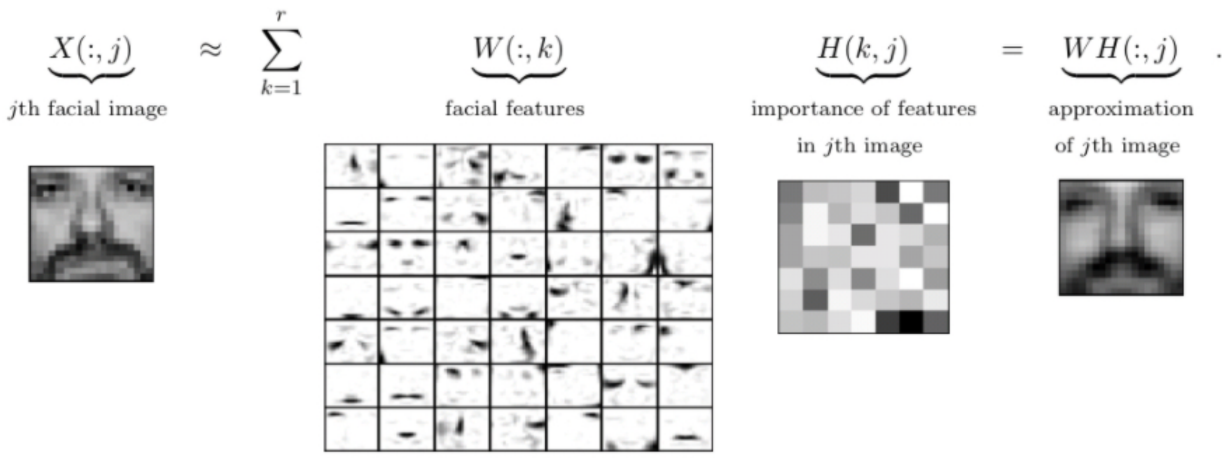
$\underbrace{X(:,j)}_{j\text{th facial image}} \approx \sum_{k=1}^{r} \underbrace{W(:,k)}_{\text{facial features}} \underbrace{H(k,j)}_{\text{importance of features in } j\text{th image}} = \underbrace{WH(:,j)}_{\text{approximation of } j\text{th image}}.$

Figure 6: Decomposition of the CBCL face database (source: [10])

8.1 Method

The optimization objective of NMF can be stated formally as below:

$$\begin{aligned} &\min_{W,H} \|X - WH\|_F^2 \\ \text{s.t.} \quad & W \geq 0 \\ & H \geq 0 \end{aligned}$$

The constraints $W \geq 0$ and $H \geq 0$ indicate that every element in the matrix is non-negative. However, as noted by [10] this optimization procedure is NP-Hard to compute exactly. Therefore, several heuristical approaches are employed to solve this problem approximately in practice. Most algorithms use a two-phase coordinate descent scheme; this means that they optimize for one of the two factors W or H while keeping the other fixed. This is because the optimization procedure then becomes convex and can hence be easily solved.

8.2 Interpretability

Due to the non-negativity constraint applied to the factor matrices, the obtained representations are more interpretable compared to other similar methods.

[10] illustrates this on a data-set of gray scale images of faces of size $p \times n$. Using NMF, a basis matrix W of size $p \times r$ is obtained. Since W is non-negative, the basis vectors can actually be interpreted as images themselves, using which every image in the data-set is reconstructed. Moreover, since $r \ll n$, only important features that are present in many images are learnt. For the facial images, these are eyes, noses, moustaches and lips. The columns of H indicate which features are important for which images. Thus, H can be viewed as a reduced dimensionality representation of the original facial images. This is illustrated in Figure 6.

Another use-case tackled in [10] is the use of NMF in text mining. When NMF is applied to non-negative data matrix corresponding to a document specified in bag-of-words setting, the each column vector obtained from matrix W corresponds to a topic, i.e. set of words found simultaneously in several different documents, while the matrix H can be interpreted as the weights assigned to different topics in each document.

Thus, due to their interpretability, we note that NMF can be used in a variety of applications for dimensionality reduction.

9 Autoencoders

Autoencoders are neural networks that are trained to replicate the input data at the output layer. They are used to learn a lower-dimensional representation or encoding from the data in a unsupervised manner. An autoencoder consists of an encoder, which "encodes" or reduces the dimensionality of the high-dimensional data into a reduced dimension encoding, and a decoder which reconstructs the original data back from the encoded representation. Mathematically, the action of a simple autoencoder with an encoder g_θ and a decoder h_ϕ can be represented as:

$$\begin{aligned} z &= g_\theta(x) \\ \hat{x} &= h_\phi(z) \end{aligned} \tag{12}$$

Due to the non-negativity constaint, [10] the representations learnt Autoencoders have various applications such as generative models like variational auto-encoders [14], super-resolution [30], anomaly detection [22] and machine translation [5]. However, they are most widely used for dimensionality reduction. The lower-dimensional encoding learnt by autoencoders are further used as input for classification or other tasks that benefit from simpler representations.

It is expected that these representations retain essential information about the input while discarding the irrelevant details and noise. One cause for concern associated with replicating the input data at the output layer is that the network could simply learn the identity function. Hence it is imperative to ensure that uncorrelated noise is discarded in the learnt representation. This is implicitly taken care of through the reduced dimension of the bottleneck layer and explicitly tackled through various regularization techniques applied at the bottleneck layer. We shall discuss some of these regularization techniques in the following sections.

9.1 Vanilla autoencoder

The simplest implementation of an autoencoder consists of a feedforward neural network that replicates the same input data at the output layer. There is no regularization applied at the bottleneck layer. Assuming that we have input dimensionality n and latent code dimensionality k, the encoder g_θ takes an n-dimensional input and produces a k-dimensional latent code. The decoder h_ϕ takes a k-dimensional latent code as the input and produces an n-dimensional reconstruction of the input data.

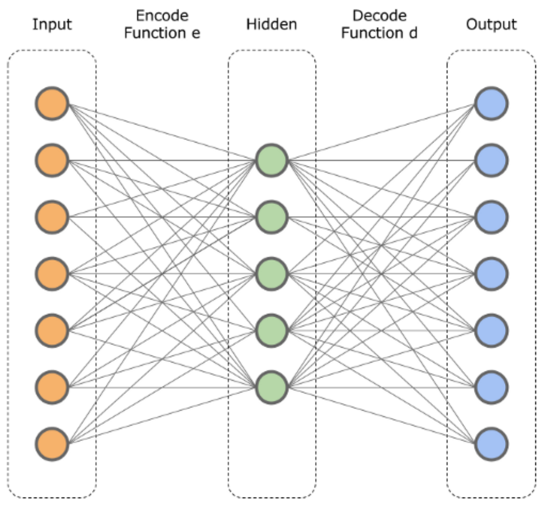

Figure 7: General architecture of an autoencoder[1]

Using the notations from Eqn.(12), we can write the learning objective of the auto-encoder as

$$\mathcal{L} = \mathbb{E}[\ell(x, \hat{x})] = \mathbb{E}[\ell(x, h_\phi(g_\theta(x)))]$$

Most commonly used loss functions used include L2 or Mean Squared Error Loss ($\ell(x, \hat{x}) = (x - \hat{x})^2$) and L1 or Mean Absolute Error Loss ($\ell(x, \hat{x}) = |x - \hat{x}|$). The learning objective $\mathcal{L}$ can be minimized as in standard neural networks through backpropagation. The encoder g_θ is then seperated and the learnt representation $z = g_\theta(x)$ is used as a condensed representation for the data point x. The first published application of autoencoders to the problem of dimensionality reduction was by [11]. They described autoencoders as a non-linear generalization of PCA (an aspect we explore in Section 9.4) and used restricted Boltzmann machine (RBM) neurons to build a 7-layer deep feedforward neural network. They then visualized the latent codes produced on the MNIST [16] dataset. They showed the superiority of these representations in terms of both seperability of different clusters and reconstruction quality compared to PCA.

9.2 Denoising autoencoder

As previously mentioned, it is imperative to ensure that autoencoders only learn useful features from the input data, while discarding noise and other unimportant details. Denoising autoencoders (DAE) [27] explicitly ensure this by artificially inducing noise into the input data while training the output layer to produce noiseless images. The claim is that by doing so, the autoencoder learns to denoise images by retaining informative dimensions in the bottleneck layer while discarding the high-frequency noise. [27] argue that the DAE learning objective can be seen as a way to define and learn a lower-dimension manifold onto which

[1] https://towardsdatascience.com/deep-autoencoders-for-collaborative-filtering-6cf8d25bbf1d

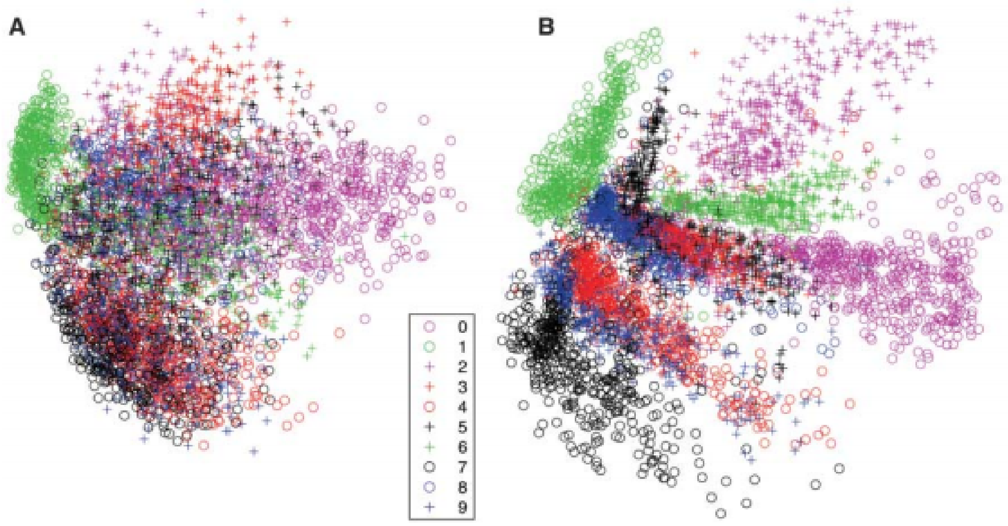

Figure 8: Figure showing the clusters generated from the representations produced by (A) PCA and (B) Autoencoders on the MNIST dataset. Note that the different digits are more clearly seperable with autoencoders (source: [11])

the data is projected. The latent representation z can be viewed as a low-dimensional coordinate system on this manifold and thus well-suited to capture the main variations in the data.

We "corrupt" the data through a function $\Gamma(x)$, followed by reconstructing the data through the encoder g_θ and decoder h_ϕ. We then aim to minimize the error between the original data point x and the reconstruction $\hat{x}$. We describe the learning objective as follows:

$$\widetilde{x} = \Gamma(x)$$
$$\mathcal{L} = \mathbb{E}[\ell(x, \hat{x})] = \mathbb{E}[\ell(x, h_\phi(g_\theta(\widetilde{x})))]$$

As before, commonly used loss functions $\ell(x, \hat{x})$ include Mean Squared Error and Mean Absolute Error. The learnt encoder g_θ is used to produce the low-dimensional representations from the data. Various methods of corrupting the data can be used. These include addition of isotropic Gaussian noise, Masking noise (a fraction of the input is destroyed or set to 0) or Salt Pepper noise (a fraction of the input is randomly selected and set to the minimum or maximum value).

9.3 Sparse autoencoders

Sparse feature learning algorithms aim to learn sparse representations for high-dimensionality data. These methods are motivated by the assumption that most high-dimensional data can represented by low-dimensional feature vectors. They typically employ a dictionary learning algorithm producing an overcomplete dictionary and an encoding algorithm to map the input vector x to a feature vector z. Some of these methods include MOD [7] and K-SVD [1]. However, these algorithms are computationally expensive; both the dictionary learning step and the sparse encoding steps involve solving iterative optimization problems at each step.

Inspired by these methods, sparse autoencoders aim to learn a sparse latent representation z for the input data x.

Specifically, sparse autoencoders enforce a sparsity constraint $\Lambda(z)$ on the latent representation z, so that the new learning objective becomes

$$\mathcal{L} = \mathbb{E}[\ell(x, \hat{x}) + \Lambda(g_\theta(x))]$$

By doing so, the network forces only some of the neurons to be active at once. Different sparsity constraints encourage different types of regularization in the latent representation. Some different sparsity constrains commonly used are

- L1 regularization in the sparse layer [2]

$$\Lambda(z) = \lambda \sum_i |z_i|$$

- KL divergence minimization between the average activation $\hat{\rho}_j$ at the hidden unit j of the bottleneck layer and a sparsity hyperparameter ρ usually close to 0 [20]. For a batch of m examples, with latent dimensionality k

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} z_j(x_i)$$
$$\Lambda(z) = \sum_{j=1}^{k} \mathrm{KL}(\rho \,||\, \hat{\rho}_j)$$

- Manually zeroing all the neurons except those with the k largest activations in the bottleneck layer. The reconstruction loss is then backpropagated only through the k largest activations. During the sparse encoding step, $z = g_\theta(x)$ is computed as before and all but the k largest activations in the bottleneck layer are zeroed out. [17]

9.4 Autoencoders and PCA

The representations learnt by an autoencoder are, in fact, closely related to the principal components discussed in Section 3 when a linear activation function is used in the autoencoder. Suppose a single hidden layer with k neurons is used in an autoencoder, with k less than the number of features d. Let W_1 and W_2 represent the weight matrices in the hidden and output layer respectively. Then, the optimization objective solved by the autoenconder is:

$$\min_{W_1, W_2} \|x - \hat{x}\|^2 = \min_{W_1, W_2} \|x - W_2 W_1 x\|^2$$

We note that the objective is similar to the PCA objective in Eqn.(5). The main difference is that the vectors obtained from PCA are naturally orthogonal to each other since they are the eigenvectors of a symmetric matrix. As argued by [21], the optimal weights of the autoencoder span the same vector subspace as the subspace spanned by the first k principal components. The principal components can even be retrieved by performing a singular value decomposition on the weight matrices of the autoencoders.

We note that the main advantage of an autoencoder over PCA is its ability to capture non-linear relationships using various different activation functions. Therefore, one can view PCA as a special case of autoencoder with an orthogonality constraint added to the columns of the weight matrices. An autoencoder is capable of capturing better patterns and reconstructing the input with a lower rate of information loss.

10 Conclusion

In this work, we have examined different techniques for dimensionality reduction. This is motivated by the phenomenon commonly referred to as the "curse of dimensionality", exponentially more number of examples are required to capture the behaviour of the distribution from which data samples are drawn. In the absence of a large number of examples, the likelihood of over-fitting on the training set increases dramatically. We examine the following methods for dimensionality reduction:

- One straightforward solution to the problem of large dimensionality is to remove irrelevant features, either by manual inspection or by using methods like discarding features uncorrelated with the label, pruning unimportant features by building a decision tree, etc. These suite of methods are referred to as **Feature Selection**.
- **Principal Component Analysis** projects the data into a lower dimensional subspace in such a manner that the projected variance is preserved as much as possible. Kernel PCA aims to extend PCA to capture non-linear dependencies through the kernel trick. It is often the most commonly used method due to it's simplicity.
- **Linear Discriminant Analysis** project the data into a lower dimensional subspace while preserving separability between the classes. It is important to note that class information is required to perform LDA.
- **t-Distributed Stochastic Neighbour Embedding** learns a lower dimensional representation that aims to preserve pair-wise local similarities. Due to its computational complexity and non-deterministic nature, it is often used only for data visualisation purposes rather than dimensionality reduction.
- **Uniform Manifold Approximation and Projection** is also a visualisation technique like t-SNE that tries to learn lower-dimensional embeddings for manifolds in higher-dimensional space. It improves over t-SNE in terms of scalability and the quality of the lower dimensional embeddings obtained.
- **Non-negative Matrix Factorization** learns a non-negative basis for the data space and yields a lower-dimensional representation of the data in terms of the learnt basis. It is best suited for applications where interpretability of the lower-dimensional representations is important.
- **Autoencoders** leverage the recent developments in deep learning to learn a lower-dimensional representation that is capable of capturing non-linear dependencies. They

can be used with image, audio and textual data using the various different variants of neural networks (Convolutional Neural Networks, Recurrent Neural Networks, etc.) Their wide-spread applicability, ease of training and good performance make them preferable for most applications of dimensionality reduction.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[2] D. Arpit, Y. Zhou, H. Ngo, and V. Govindaraju. Why regularized auto-encoders learn sparse representation? In *International Conference on Machine Learning*, pages 136–144, 2016.

[3] R. Bellman. *Dynamic Programming.* Dover Publications, 1957.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[5] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[6] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Willey & Sons, New Yotk, 1973.

[7] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, volume 5, pages 2443–2446 vol.5, 1999.

[8] K. P. F.R.S. Liii. on lines and planes of closest fit to systems of points in space. 1901.

[9] K. Fukunaga. *Introduction to Statistical Pattern Recognition (2nd Ed.).* Academic Press Professional, Inc., USA, 1990.

[10] N. Gillis. The why and how of nonnegative matrix factorization. *Regularization, optimization, kernels, and support vector machines*, 12(257):257–291, 2014.

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

[12] H. Hotelling. Analysis of a complex of statistical variables into principal components. 1993.

[13] A. Hyvärinen. Survey on independent component analysis. 1999.

[14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[15] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22, 1951.

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

[17] A. Makhzani and B. Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[18] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[19] K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[20] V. Nair and G. E. Hinton. 3d object recognition with deep belief nets. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, page 1339–1347, Red Hook, NY, USA, 2009. Curran Associates Inc.

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

[22] M. Sakurada and T. Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014.

[23] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks — ICANN'97*, pages 583–588, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[24] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[25] S. Solorio-Fernández, J. Carrasco-Ochoa, and J. F. Martínez-Trinidad. A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 01 2019.

[26] L. van der Maaten and G. E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

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

[28] Q. Wang. Kernel principal component analysis and its applications in face recognition and active shape models, 2012.

[29] Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353, 2012.

[30] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao. Coupled deep autoencoder for single image super-resolution. *IEEE transactions on cybernetics*, 47(1):27–37, 2015.

[31] A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5:363–387, 10 2012.