Data Preparation and Dimensionality Reduction

Lecture Notes for Big Data Analytics

Faizad Ullah

March 2019

Contents

1 Data Preparation **2**

2 Data Reduction **2**

3 Data Compression **2**
3.1 Lossless Compression vs. Lossy Compression 3
3.2 Graph Summarization 3

4 Low Distortion Embedding **5**

5 t-distributed Stochastic Neighbor Embedding (t-SNE) **6**

6 Multidimensional Scaling (MDS) **6**

7 Isometric mapping (Isomap) **6**

8 Dimensionality Reduction **7**
8.1 Feature Selection 8
8.2 Feature Extraction 9
8.3 Linear Dimensionality Reduction 9
8.3.1 Principal Component Analysis (PCA) 10
8.3.2 Factor Analysis (FA) 10
8.3.3 Truncated Singular Value Decomposition (SVD) 10
8.4 Random Projection and Johnson-Lindenstrauss Lemma 11
8.4.1 The Johnson-Lindenstrauss Lemma 12
8.5 Non-Linear Dimensionality Reduction 13
8.5.1 Linear Discriminant Analysis (LDA) 13
8.5.2 Kernel PCA 13

1 Data Preparation

Real-world data such as images, audio, video and graphs are high-dimensional in nature and susceptible to missing, inconsistent and noisy data, due to heterogeneous sources. Data preparation is composition of tasks to transform raw data before processing and analysis. It is an important step prior to processing and often involves reformatting data, making corrections to data and the combining of data sets to enrich data. Data preparation allows for efficient data analysis (in terms of computational complexity) and it enriches the data and enhance the quality of analytics drawn from the data. It also makes data easier to visualize.

We discussed many elementary steps for data preparation in earlier sessions, like data clean, data de-noising, scaling and normalization, data transformation, de-duplication of data. Most of those steps focused on individual data attributes or pairs of them. Here we discuss more advanced data preparation steps, that address the dataset as a whole.

2 Data Reduction

To perform data analytic tasks on a huge amount of data is impractical, infeasible, and take long time. Data reduction is a technique that can be applied to reduces the quantity/volume of that huge data while preserving the quality of the dataset. Hence mining on such reduced dataset is feasible, efficient yet produced almost the same result as on original data. Data reduction techniques include data compression, dimensionality reduction, and numerosity reduction (we discussed this earlier under de-duplication and Sampling). We will briefly discuss these techniques in this chapter.

3 Data Compression

Data compression is the process of encoding information with fewer bits than the original representation and often deals with large volumes of data. Given a point set $X = \{x_1, x_2, \ldots, x_n\}$, the objective is to find a compression scheme $f : X \mapsto X'$ (encoder) and a decompressor $g : X' \mapsto X$ (decoder) such that $\sum_{i=1}^{n} \|x_i - g(f(x_i))\|^p$, called the called ℓ_p reconstruction error, is minimized.

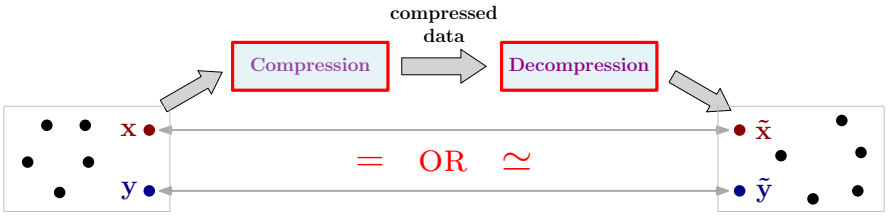

Figure 1 Data compression and decompression

3.1 Lossless Compression vs. Lossy Compression

Note that g is not necessarily f^{-1}. If $g = f^{-1}$, compression is called *lossless* otherwise it is *lossy*. In lossless compression, the original data is recovered completely as it is after decompression, i.e. $X = X'$ but in lossy compression, the recovered data maybe somewhat 'distorted', i.e. $X \simeq X'$ as shown in Figure 2.

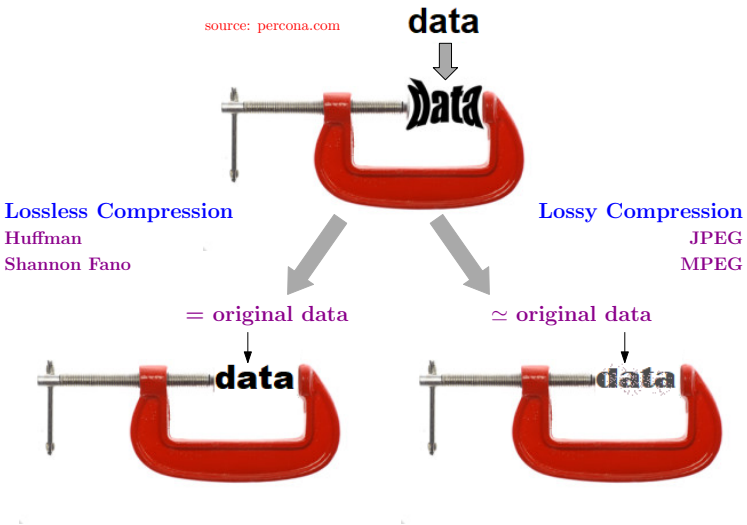

Figure 2 Lossless vs. lossy data compression

3.2 Graph Summarization

We discuss an example application of data compression in the domain of large graph analysis Graphs on millions of nodes and billion of edges are increasingly common. Example

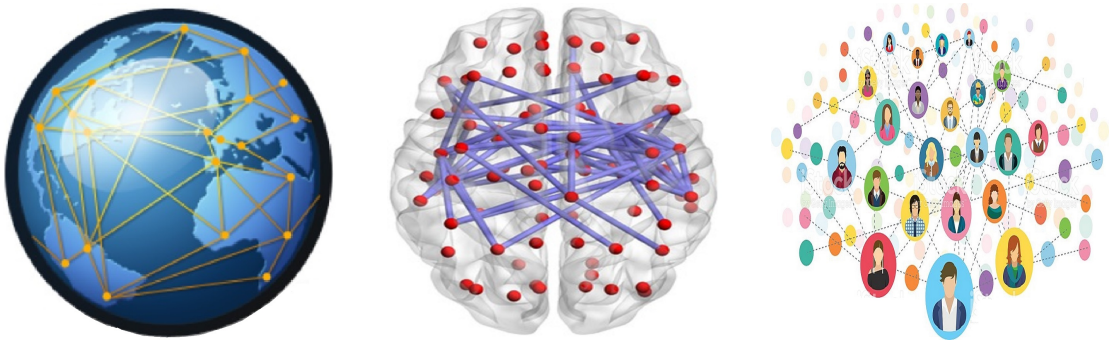

Figure 3 World Wide Web (left) Biological Network (middle) and Social Network (right)

includes, communication networks, social networks and World Wide Web (WWW). The huge magnitudes of graphs (order of a graph is the number of nodes in it and size of a

graph is the number of edges in it) pose serious computational challenges for graph Analysis and Processing, graph visualization, storage and retrieval and transfer over a network (communication bandwidth).

A very successful approach to overcome these challenges is graph compression or graph graph summarization. Where we perform all processing and network analytics instead on a compact version of the graph. The summary of a graph removes certain details yet preserve the essential structure of the graph. More formally, a summary is a partition of vertices of the graph represented by a *supergraph*. Each *super node* represents a subset of vertices of the original graph and Each *super edge* between a pair of supernodes represents edges between the subsets corresponding to the vertices in the supernodes.

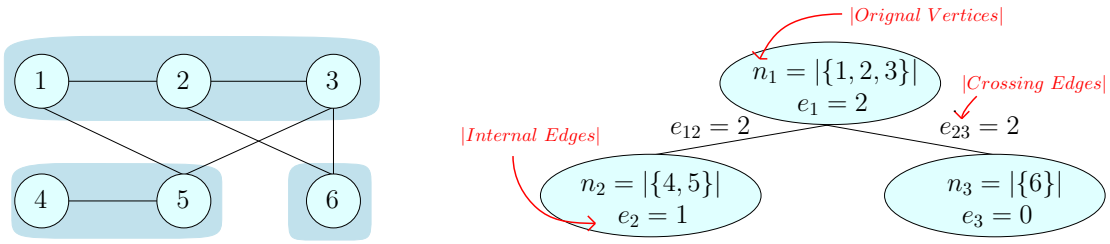

Figure 4 The summary is a graph with weights both on superedges and supernodes.

Given the smaller order and size of the summary graph it is easier to process and visualization. One can perform graph analytics such as get approximate answers to graph structure queries ($(u, v) \sim E$) from the summary only. As pointed above, the summary sometimes is the desired output (with applications in aggregate statistics and privacy preservation). The detailed discussion about graph compression (graph summarization) is given in chapter 13.

Goodness of a Summary is measured by the Reconstruction Error: the ℓ_p norm of error matrix

$$\left\|A - \bar{A}\right\|_p = RE_p(G|S) = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} |A(i,j) - \bar{A}(i,j)|^p\right)^{1/p}$$

where $\overline{A}$, the expected adjacency matrix is defined as: $\bar{A}(u, v)$ is the probability of an edge between u and v

$$\bar{A}(u,v) = \begin{cases} 0 & \text{if } u = v \\ \dfrac{e_i}{\binom{n_i}{2}} & \text{if } u, v \in V_i \\ \dfrac{e_{ij}}{n_i n_j} & \text{if } u \in V_i, v \in V_j \end{cases}$$

This matrix difference can be naively computed in $O(n^2)$
For algorithms to compute the best summary efficiently see [2, 3, 4, 1].

.

$$A = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \qquad \bar{A} = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} \end{array} \\ \begin{bmatrix} 0 & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & 0 & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & 0 & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 1 & 0 & 0 \\ \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 5 Adjacency matrix for original (left) vs. compressed (right) graph

4 Low Distortion Embedding

Most of the analytics are designed for vector spaces with a specific distance metric (called metric-spaces). In many cases the raw data is not in vector format or the distance measure is not a metric. In some cases even if the original data is in metric spaces, we often need to transform the data to a different metric space, where the algorithms are better, more efficient, or the data is easy to visualize. For this purpose we transform the data into a target metric space, while preserving the pairwise distances.

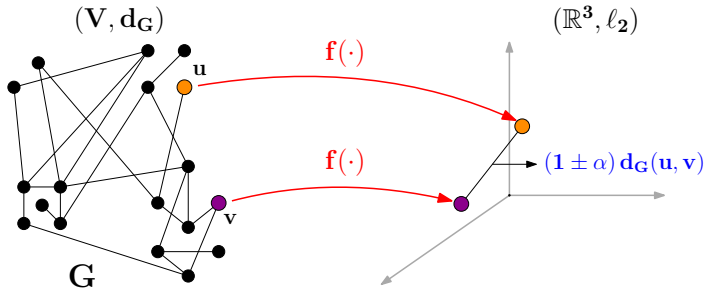

Figure 6 Graph vertices embedded to $(\mathbb{R}^3, \ell_2)$

Given two metric spaces (X, d) and (Y, d') and a real $\alpha > 0$, the low distortion embedding problem is to find an embedding function $f : X \mapsto Y$ such that

$$\forall\, x_i, x_j \in X \quad \frac{1}{\alpha} d(x_i, x_j) \;\leq\; d'(f(x_i), f(x_j)) \;\leq\; d(x_i, x_j)$$

Points in X are embedded into Y, almost preserving pairwise distances, because the space Y may be easy to work with or the distance metric d' may be computationally nicer.

Examples include Graph vertices with shortest paths distances embedded to $(\mathbb{R}^k, \ell_2)$, as shown in Figure 6 or sequences with edit distance embedded into Euclidean space.

5 t-distributed Stochastic Neighbor Embedding (t-SNE)

6 Multidimensional Scaling (MDS)

Multidimensional scaling is another non-linear dimensionality reduction technique. The detailed discussion about this technique is given in chapter 13. Given $X = \{x_1, \ldots, x_n\}$ and pairwise distance matrix $D = \{d_{ij}\}$, the multi-dimensional scaling problem is to find a k-dimensional representation $\{x'_1, x'_2, \ldots, x'_n\}$ for points in X such that $\forall\ x_i, x_j \in X \quad d(x'_i, x'_j) \sim D(i, j)$. Metric and non-metric are the two types of MDS algorithms.

Given $X = \{x_1, \ldots, x_n\}$ and pairwise distance matrix $D = \{d_{ij}\}$, the multi-dimensional scaling problem is to find a k-dimensional representation $\{x'_1, x'_2, \ldots, x'_n\}$ for points in X such that $\forall\ x_i, x_j \in X \quad d(x'_i, x'_j) \sim D(i, j)$. Figure 7 shows an example for $k = 2$.

source: statisticshowto.com

CITY	Clearwater	Daytona Beach	Ft. Lauderdale	Gainesville	Jacksonville
Clearwater	0	159	247	131	197
Daytona Beach	159	0	230	97	89
Ft. Lauderdale	247	230	0	309	317
Gainesville	131	97	309	0	68
Jacksonville	197	89	317	68	0

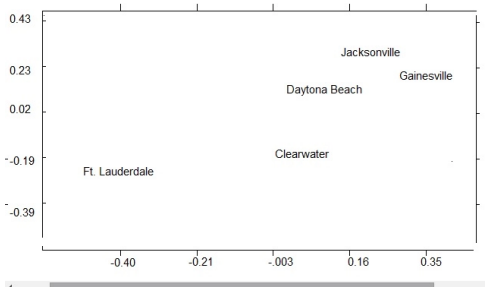

Figure 7 Cities in Florida and their distances (left) and the graph produced after scaling (right) for $k = 2$

Many methods depending on whether or not the given and required distance measure is metric or Euclidean. A distance measure is metric if it satisfies the triangle inequality, i.e. $d(x, z) \leq d(x, y) + d(y, z)$. Squared Euclidean distances are not metric.

7 Isometric mapping (Isomap)

Isomap is an extension of kernel PCA or MDS. This technique tries to preserve the geodesic distance (to its nearest neighbors) and is based on spectral theory. Isomap creates neighborhood netwrok and uses graph distance to approximate the graph distance (geodesic) between all pairs of points. Finally, it finds the low dimensional embedding (of dataset) through eignevalue decomposition of geodesic distance.

Therefore, there is need for more sophisticated feature extraction methods. Before we discuss such a dimensionality reduction technique, i.e. let's recall the key linear algebra concept of projection.

8 Dimensionality Reduction

Having cursed dimensionality enough, now we finally start to do something about it. While we discussed many issues with large dimensions, we focus on computational aspect of the curse such as processing time, storage capacity and communication bandwidth. Since it is hard to work with high dimensional vectors (perhaps in tens or hundreds of thousands), for computational efficiency and quality of analytics, we would like to reduce the number of dimensions to a manageable number (in hundreds perhaps) without "distorting" the data too much.

Dimensionality reduction is the technique used for data reduction, in which we reduce the number of attributes or variables from the data. The specific definition and measure of distortion depends on the given data set and the target application. While there could be other objectives for dimensionality reduction, our goal is to reduce dimensionality of the dataset, while preserving pairwise distance.

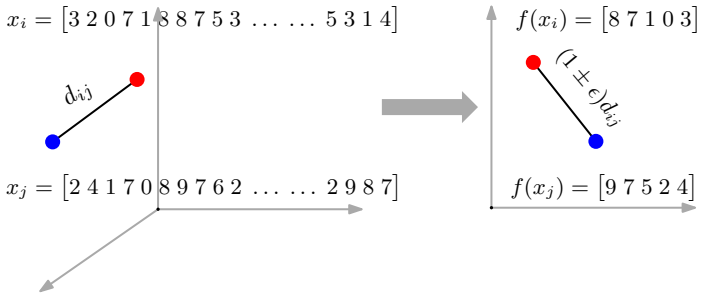

Figure 8 Two points in a high dimensional space reduced to a 5 dimensional space

In this course we will study many techniques for dimensionality reduction, namely, the Johnson-Lindenstrauss transform and (it's variations), the AMS transform (that is originally meant for something different), Locality Sensitive Hashing, Singular Value Decompostion, and Principal Component Analysis.

Problem 1. *Given a point set $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$, Find a dimensionality reduction function $f : \mathbb{R}^m \mapsto \mathbb{R}^k$, $k \ll m$ such that $d(f(x_i), f(x_j)) \simeq d(x_i, x_j)$, i.e.*

$$\forall\, x_i, x_j \in \mathcal{X} \quad (1-\epsilon)d(x_i, x_j) \;\leq\; d(f(x_i), f(x_j)) \;\leq\; (1+\epsilon)d(x, y)$$

Note that dimensionality reduction is a special case of low distortion embedding since the distance measure d is the same in both domain and co-domain, i.e. the given and required distance measure is the same. Dimensionality reduction is, however, different from data compression since it does not require that $x \simeq f(x)$, but only that $d\big(f(x_i), f(x_j)\big) \simeq d(x_i, x_j)$ as shown in Figure 9.

While the specific method of dimensionality reduction depends on the objective, all dimensionality reduction methods can be divided into two broad categories (as shown in Figure 11)

Broadly there are two types of dimensionality reduction i) linear dimensionality reduction and ii) non-linear dimensionality reduction. In some texts it is categorized as feature selection and feature extraction.

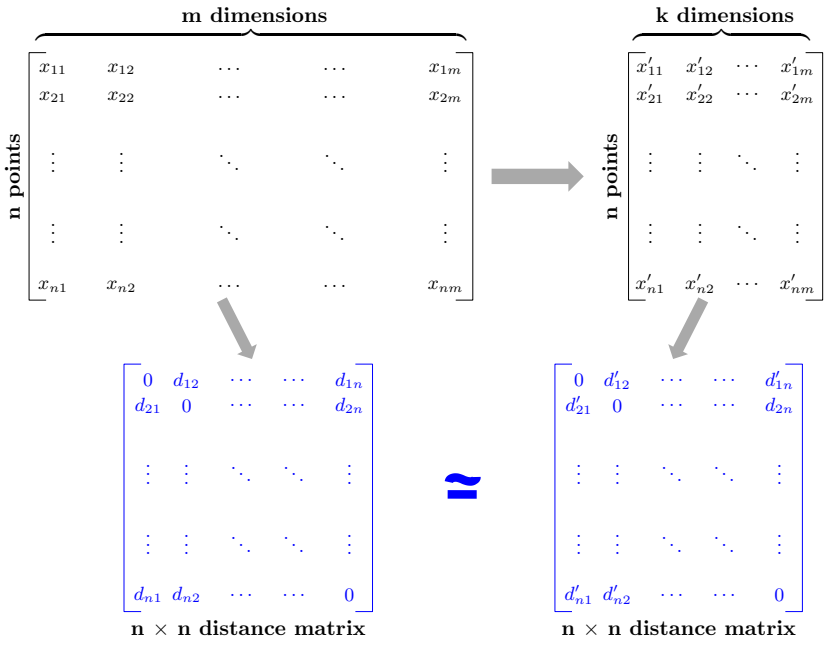

Figure 9 Dimensionality reduction

1. Feature Selection: Select a few variables that are the *most relevant* and discard the rest.

2. Feature Extraction - Create new features from data, which are usually are linear or non-linear combination of old ones. The objective is least reconstruction error or maximum inter-class separation.

8.1 Feature Selection

An immediate idea (that does not work) for dimensionality reduction is to select a fixed subset of coordinates. Simply consider the first k coordinates for each vector and ignore the rest. In the worst case all the information could be in the remaining coordinates (suppose all vectors are almost 0 in the all of the first k coordinates). Similarly considering the last k coordinates or any fixed subset of coordinates for that matter would not work since all meaningful information about at least some of the classes of points could remain in the non-selected features.

One may suggest that, in order to not let the adversary know what coordinates you would choose, choose a random subset of k coordinates and ignore the remaining. This may not work either as, again, all meaningful information about at least some of the classes of points could remain in the non-sampled features. For example, suppose that vectors have non zero entries only in the first k coordinates. Then, a randomly chosen subset will select the mostly zero coordinates and distort most pairs of points.

Both the above methods are *data oblivious*. A *data dependent* approach would be to

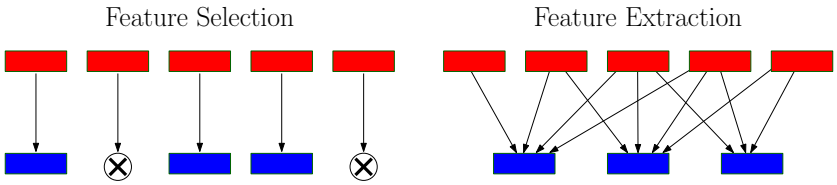

Figure 10 Feature selection and feature extraction example

eliminate/select feature based on a goodness measure or (ir)relevance score of the data. For example:

- Feature variance - eliminate coordinate with close to 0 variance
- Eliminate one in a pair with close to ± 1 correlation
- Eliminate features "independent" of class variable (ρ or χ^2)
- For each feature find training accuracy of classifier based on that feature only - eliminate those with low accuracy
- Score based on normalized mutual information, information gain, conditional entropy
- We discussed a domain specific criterion of eliminating feature - stop word removal for text analysis

8.2 Feature Extraction

The objective of feature extraction is to create new (fewer) features using some (linear or non-linear) combinations of the old ones such that reconstruction error is minimized or inter-class separation is maximized. A form of feature extraction is *feature aggregation* which aggregates groups of coordinates. For example, means of k groups of m/k coordinates. One can construct examples where this would not work as well. It depends on how groups are made and a deterministic strategy can be countered by adversary. Randomized groups may also have problems. Therefore, there is need for more sophisticated feature extraction methods.

8.3 Linear Dimensionality Reduction

Problem 2 (Dimensionality Reduction Problem)**.** *Given a point set* $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$*, Find a linear dimensionality reduction function* $f : \mathbb{R}^m \mapsto \mathbb{R}^k$*,* $k \ll m$ *such that* $d(f(\mathbf{x}_i), f(\mathbf{x}_j)) \simeq d(\mathbf{x}_i, \mathbf{x}_j)$*. More precisely,*

$$\forall\, \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X} \quad (1-\epsilon)d(\mathbf{x}_i, \mathbf{x}_j) \;\leq\; d(f(\mathbf{x}_i), f(\mathbf{x}_j)) \;\leq\; (1+\epsilon)d(\mathbf{x}_i, \mathbf{x}_j)$$

If f is a linear function, it is called linear diensionality reduction and f can be represented by a linear transformation A, i.e. $f(\mathcal{X}) = A\mathcal{X}$, $\mathcal{X}$: the $n \times m$ data matrix with each $\mathbf{x}_i \in \mathcal{X}$ as a row.

8.3.1 Principal Component Analysis (PCA)

In this section, we will briefly describe the PCA as dimensionality reduction technique. Later we discuss the details algebra, algorithm and geometry of the PCA. This technique is data dependent technique, which identifies the relationship in data, transform the data based on these relations and then quantify the importance of these relationships. Relationship are identified through covariance matrix. Eigenvectors and eigenvalues are identified through eigen-decomposition or linear transformation of the covariance matrix. The data is then transformed to principal components using eigenvectors and finally we only keep the importance principal components, quantified by the relationships using eigenvalues.

8.3.2 Factor Analysis (FA)

Factor analysis is also linear dimensionality reduction technique and one of the unsupervised machine learning algorithm. This technique creates factors which represents the common variance in the data, i.e., variance in the data due to correlation among the observed variables.

8.3.3 Truncated Singular Value Decomposition (SVD)

Truncated-SVD is another liner dimensionality reduction technique. The detailed discussion about SVD will be given later. Unlike PCA it usually works better in case of sparse data (data with many zero values). Truncated-SVD factorized the matrix (data matrix) where the number of columns is equal to the truncation.

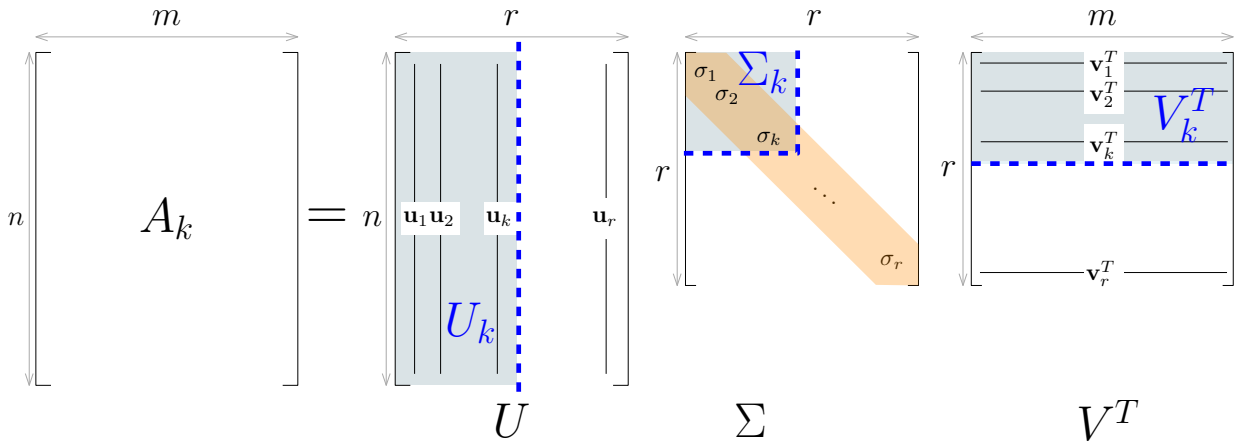

Figure 11 Optimal Low rank factorization of the data matrix using the top k singular vectors

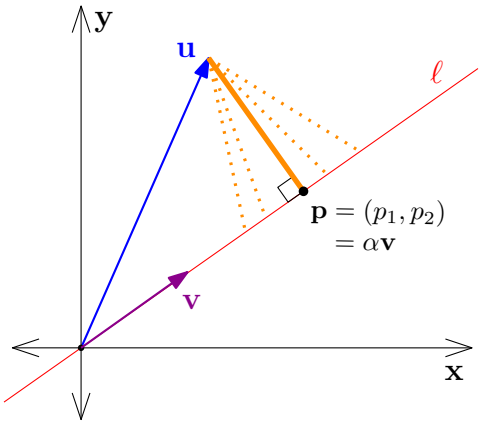

Figure 12 The projection of vector $\mathbf{u}$ on $\mathbf{v}$

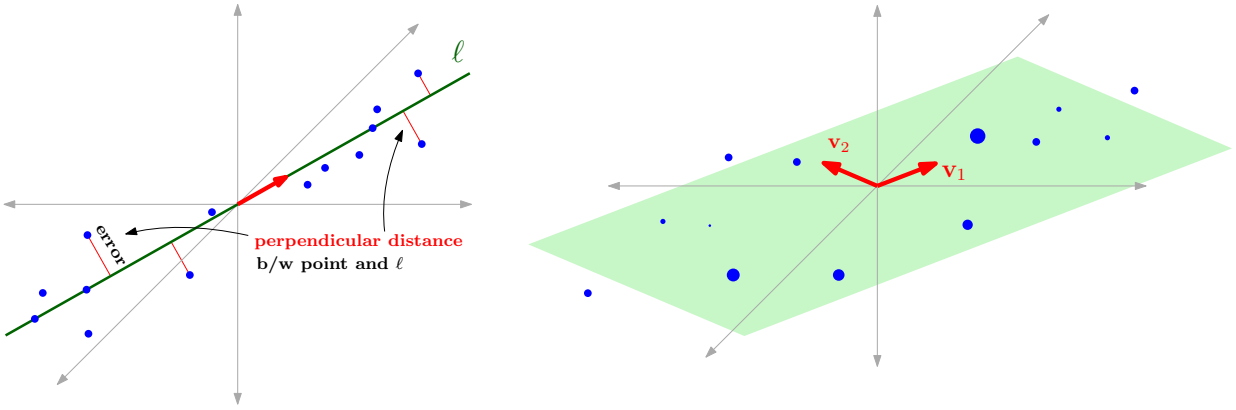

Figure 13 The m-d data lies in a subspace

8.4 Random Projection and Johnson-Lindenstrauss Lemma

In this session, we discuss a linear dimensionality reduction method by Johnson and Lindenstrauss based on random projection. This is a data oblivious dimensionality reduction method. We give a brief review of projection from our linear algebra lecture.

Let $\mathbf{v}$ be a unit vector, let l be a line in the direction of $\mathbf{v}$ and let p be the point on l that is closest to a vector (point). The line connecting $\mathbf{u}$ to p is perpendicular to $\mathbf{v}$, otherwise p will not be the closest point (by the Pythagoras theorem). The point (vector) p is called the *projection* of $\mathbf{u}$ on $\mathbf{v}$. Finding projection of $\mathbf{u}$ on the standard basis vectors is easy as shown in Figure 12.

For general vectors we derive it from dot product. Note that p is just scaled vector $\mathbf{v}$, i.e. $p = a\,\mathbf{v}$. We need to find the scalar a. Since $\mathbf{u} - p = \mathbf{u} - a\mathbf{v}$ is perpendicular on $\mathbf{v}$, $\mathbf{v} \cdot (\mathbf{u} - a\,\mathbf{v}) = 0$. Hence, $\mathbf{v} \cdot \mathbf{u} - \mathbf{v} \cdot a\,\mathbf{v} = \mathbf{v} \cdot \mathbf{u} - a\,\mathbf{v} \cdot \mathbf{v} = 0$ which means $a\,\mathbf{v} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$ and therefore $a = \dfrac{\mathbf{v} \cdot \mathbf{u}}{\mathbf{v} \cdot \mathbf{v}} = \dfrac{\mathbf{v} \cdot \mathbf{u}}{\|\mathbf{v}\|}$

Suppose the m-D data lies on the line ℓ and let $\mathbf{v}$ be the unit vector in direction of ℓ as

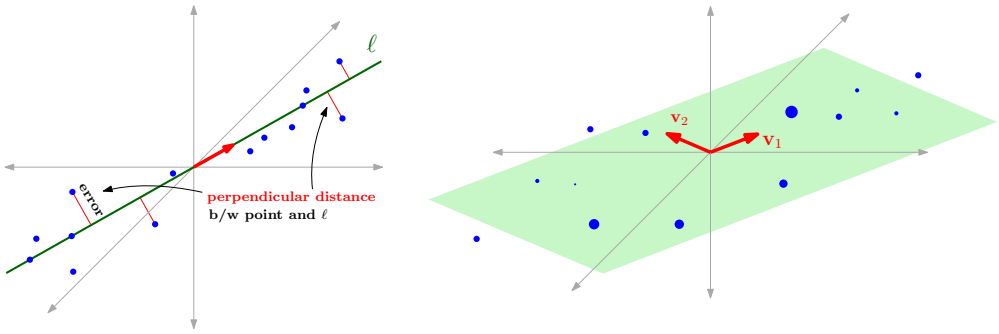

Figure 14 Finding the low dimensional space to which the data is close by, then the error would be perpendicular distance not a vertical distance

shown in Figure **??**. For $\mathbf{x_i} \in \mathcal{X}$, let $f(\mathbf{x_i}) := \mathbf{v} \cdot \mathbf{x_i}$.

In this case, since $\mathbf{v} \cdot \mathbf{x}_i = \mathbf{x}_i$ (as $\mathbf{x}_i$ lies on ℓ), we get

$$\forall i,\, j \quad \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| = \|\mathbf{v} \cdot \mathbf{x}_i - \mathbf{v} \cdot \mathbf{x_j}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$$

Now suppose the m-d data lies on a plane with orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ as in Figure 13. Let $\mathbf{V}$ be the matrix with $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ as columns. For $\mathbf{x_i} \in X$, let $f(\mathbf{x_i}) := \mathbf{x_i}\, \mathbf{V}$. Then,

$$\forall i,\, j \quad \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| = \|\mathbf{x}_i\mathbf{V} - \mathbf{x_j}\mathbf{V}\| = \|\mathbf{x}_i - \mathbf{x}_j\|$$

In both these cases,there is 0 error dimensionality reduction, i.e. without any distortion. However, the problem is that we do not know $\mathbf{V}$.

It is possible to find the low dimensional space to which the data is close by, with an approach similar to (multiple) linear regression. However, the error here would be perpendicular distance not vertical distance as shown in Figure 14. Secondly, whereas the goal in linear regression is to minimize SSE, in dimensionality reduction it would be to minimize pairwise distances .

This approach can be taken with modified goals and is called *Principal Component Analysis* (PCA) which is a form of feature extraction and is data dependent dimensionality reduction.

8.4.1 The Johnson-Lindenstrauss Lemma

The correct answer is a classic result in functional analysis - the Johnson-Lindenstrauss Lemma. Its statement is as follows:

Theorem 1. *Given* $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$. *For* $\epsilon \in (0, 1/2)$, *there exists a linear map* $f : \mathbb{R}^m \to \mathbb{R}^k$, $k = c \log n/\epsilon^2$ *such that for any* $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$

$$(1 - \epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2 \leq (1 + \epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2$$

Note that this lemma works only for ℓ_2 distance. This lemma has found many applications in algorithms. For example, both of our generic proximity computation problems have reduced time complexity. The distance matrix can now be computed in $O(n^2 \frac{\log n}{\epsilon^2})$ instead of $O(n^2 \frac{\log n}{\epsilon^2})$ and the nearest neighbor computation can be done in $O(n \frac{\log n}{\epsilon^2})$ instead of $O(nm)$.

With more and more interest by theoretical computer science community, there is a lot of work towards proving this lemma.

8.5 Non-Linear Dimensionality Reduction

In real-world, non-linear data is found frequently. And if we are dealing with such data, we need some non-linear techniques to handle it. Generally, linear methods do not perform well on non-linear data. So in this section, we will discuss some important non-linear dimensionality reduction techniques.

8.5.1 Linear Discriminant Analysis (LDA)

LDA is data dependent, supervised dimensionality reduction technique. It is very similar to PCA but unlike PCA, LDA finds some additional information, i.e., interested in the axes that maximizes the separation between multiple classes by computing the directions called linear discriminant.

8.5.2 Kernel PCA

Kernel PCA can also be considered as the non-linear form of PCA. It usually works well for non-linear datasets where PCA cannot be applied efficiently. When using kernel PCA, we initially passed the data through a kernel function that projects the data (temporarily into new high-dimensional feature space) such that the classes becomes linearly separable. The data is then transformed to lower dimensional space by using the normal PCA. In this way, Kernel PCA transform non-linear data into a lower-dimensional space of data which can be used with linear classifiers. Kernel PCA with appropriately chosen kernel in certain cases significantly outperforms PCA and other data transformation methods.

In the Kernel PCA, we need to specify 3 important hyperparameters — the number of components we want to keep, the type of kernel and the kernel coefficient (also known as the gamma).

References

[1] M. Beg, M. Ahmad, A. Zaman, and I. Khan. Scalable approximation algorithm for graph summarization. In *Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 502–514, 2018.

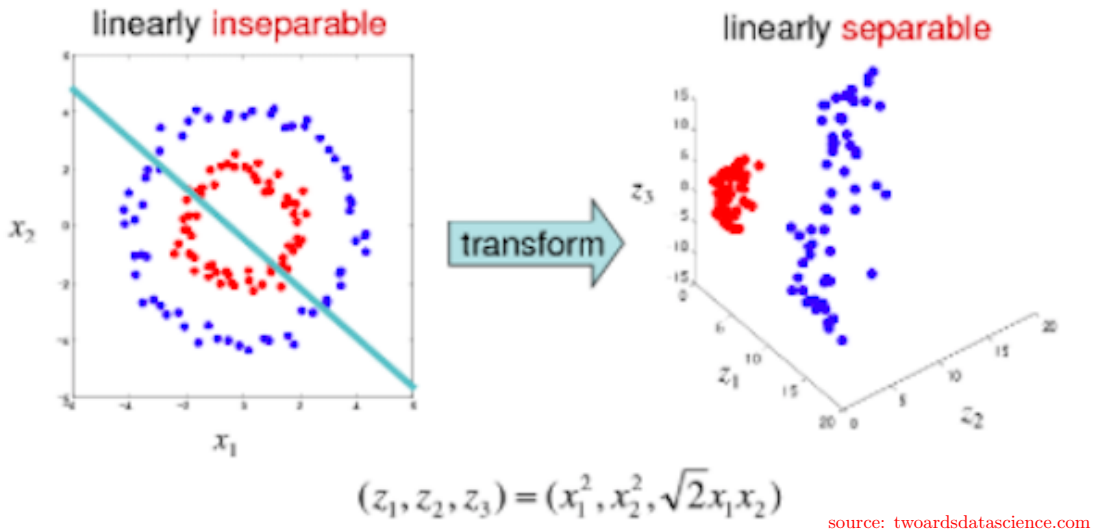

Figure 15 Transforming the data with Kernel PCA helps separate linearly inseparable data

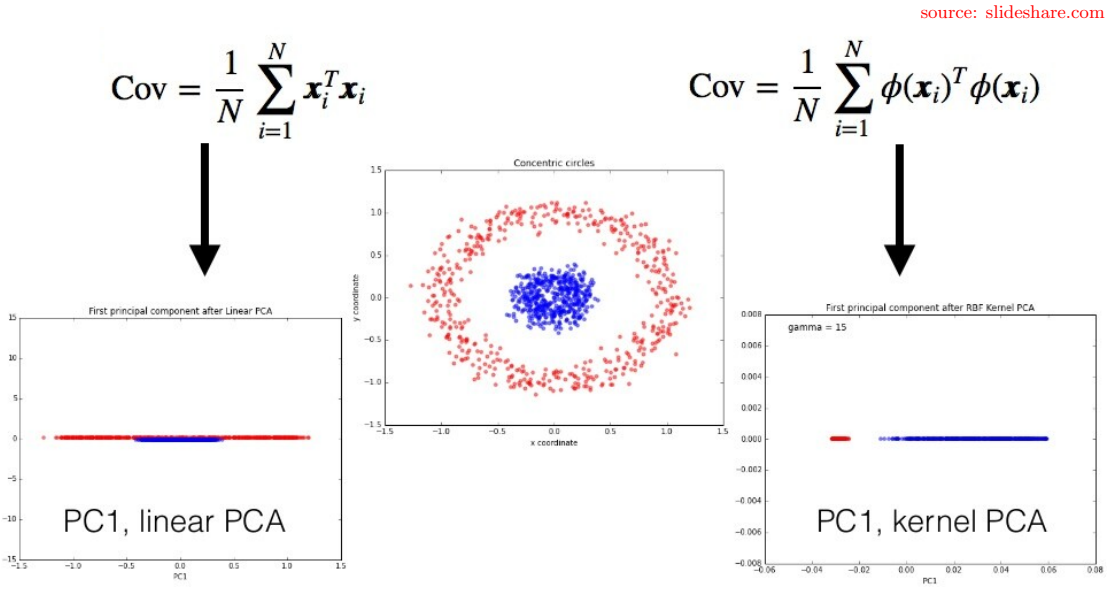

Figure 16 Applying PCA (left) and kernel PCA (right) to same data

[2] Kristen LeFevre and Evimaria Terzi. GraSS: Graph Structure Summarization. In *International Conference on Data Mining, SDM*, pages 454–465, 2010.

[3] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph summarization with bounded error. In *International Conference on Management of Data, SIGMOD*, pages 419–432, 2008.

[4] Matteo Riondato, David García-Soriano, and Francesco Bonchi. Graph Summarization with Quality Guarantees. In *International Conference on Data Mining, ICDM*, pages 947–952, 2014.