FE 582

Lecture 2: Data Preparation

Dragos Bozdog

Data Preparation

- Feature Extraction and Portability
- Data Cleaning
- Data Reduction and Transformation

Feature Extraction

- The first phase of feature extraction is crucial, though it is very application specific.

- In some cases, feature extraction is closely related to the concept of data type portability, where low-level features of one type may be transformed to higher-level features of another type.

- In other cases where a heterogeneous mixture of features is available in different forms, an “off-the-shelf” analytical approach is often not available to process such data. In such cases, it may be desirable to transform the data into a uniform representation for processing. This is referred to as *data type porting*.

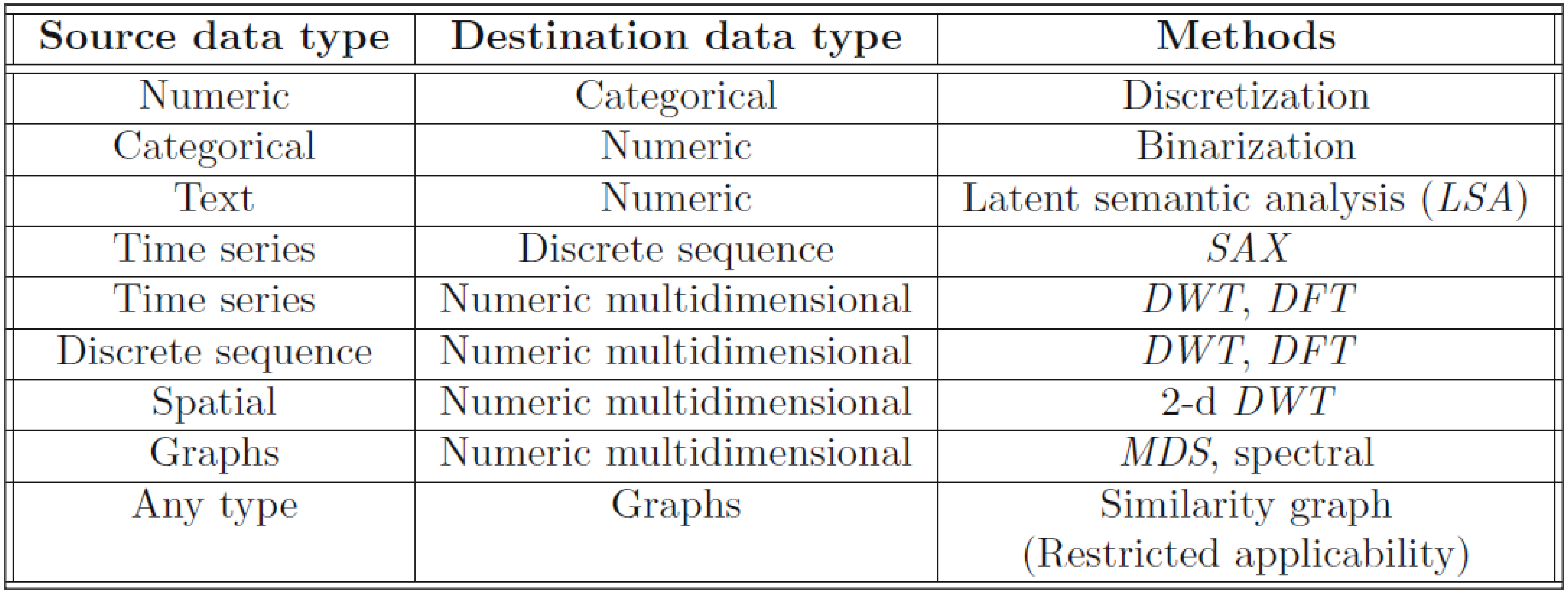
Data Type Portability

- Data type portability is a crucial element of the data mining process because the data is often heterogeneous, and may contain multiple types.

- This section will describe methods for converting between various data types. Because the numeric data type is the simplest and most widely studied one for data mining algorithms, it is particularly useful to focus on how different data types may be converted to it.

- However, other forms of conversion are also useful in many scenarios. For example, for similarity-based algorithms, it is possible to convert virtually any data type to a graph and apply graph-based algorithms to this representation.

Portability of different data types



Source data type	Destination data type	Methods
Numeric	Categorical	Discretization
Categorical	Numeric	Binarization
Text	Numeric	Latent semantic analysis (*LSA*)
Time series	Discrete sequence	*SAX*
Time series	Numeric multidimensional	*DWT*, *DFT*
Discrete sequence	Numeric multidimensional	*DWT*, *DFT*
Spatial	Numeric multidimensional	2-d *DWT*
Graphs	Numeric multidimensional	*MDS*, spectral
Any type	Graphs	Similarity graph (Restricted applicability)

Numeric to Categorical Data: Discretization

- The most commonly used conversion is from the numeric to the categorical data type. This process is known as *discretization*. The process of discretization divides the ranges of the numeric attribute into φ ranges. Then, the attribute is assumed to contain φ different categorical labeled values from 1 to φ, depending on the range in which the original attribute lies.

- Ex: Consider the age attribute. One could create ranges [0, 10], [11, 20], [21, 30], and so on. The *symbolic* value for any record in the range [11, 20] is “2” and the symbolic value for a record in the range [21, 30] is “3”. Because these are symbolic values, no ordering is assumed between the values “2” and “3”.

- Furthermore, variations within a range are not distinguishable after discretization. Thus, the discretization process does lose some information for the mining process.

Numeric to Categorical Data: Discretization

The discretization process can be performed in a variety of ways depending on application specific goals:

- *Equi-width ranges:* In this case, each range [*a, b*] is chosen in such a way that $b - a$ is the same for each range. This approach has the drawback that it will not work for data sets that are distributed nonuniformly across the different ranges. The range [*min, max*] is then divided into φ ranges of equal length.

- *Equi-log ranges:* Each range [*a, b*] is chosen in such a way that $\log(b) - \log(a)$ has the same value. This kinds of range selection has the effect of geometrically increasing ranges $[a, a \cdot \alpha]$, $[a \cdot \alpha, a \cdot \alpha 2]$, and so on, for some $\alpha > 1$. This kind of range may be useful when the attribute shows an exponential distribution across a range.

- *Equi-depth ranges:* In this case, the ranges are selected so that each range has an equal number of records

Categorical to Numeric Data: Binarization

- In some cases, it is desirable to use numeric data mining algorithms on categorical data. Because binary data is a special form of both numeric and categorical data, it is possible to convert the categorical attributes to binary form and then use numeric algorithms on the binarized data.

- If a categorical attribute has φ different values, then φ different binary attributes are created. Each binary attribute corresponds to one possible value of the categorical attribute.

- Therefore, exactly one of the φ attributes takes on the value of 1, and the remaining take on the value of 0.

Text to Numeric Data

- It is possible to convert a text collection into a form that is more amenable to the use of mining algorithms for numeric data. The first step is to use *latent semantic analysis (LSA)* to transform the text collection to a nonsparse representation with lower dimensionality.

- Traditional text mining algorithms are directly applied to the reduced representation obtained from *LSA.*

- Alternatively, after scaling, traditional numeric measures, such as the Euclidean distance, can be used.

Time Series to Discrete Sequence Data

Time-series data can be converted to discrete sequence data using an approach known as *symbolic aggregate approximation (SAX)*. This method comprises two steps:

- *Window-based averaging:* The series is divided into windows of length *w*, and the average time-series value over each window is computed.
- *Value-based discretization:* The (already averaged) time-series values are discretized into a smaller number of approximately *equi-depth* intervals. The idea is to ensure that each symbol has an approximately equal frequency in the time series.

Thus, *SAX* might be viewed as an equi-depth discretization approach after window-based averaging.

Time Series to Numeric Data

- This particular transformation is very useful because it enables the use of multidimensional algorithms for time-series data. A common method used for this conversion is the discrete wavelet transform (*DWT*). The wavelet transform converts the time series data to multidimensional data, as a set of coefficients that represent averaged differences between different portions of the series.
- If desired, a subset of the largest coefficients may be used to reduce the data size.
- An alternative method is the discrete Fourier transform (*DFT*).
- The common property of these transforms is that the various coefficients are no longer as dependency oriented as the original time-series values.

Discrete Sequence to Numeric Data

This transformation can be performed in two steps:

- Convert the discrete sequence to a *set* of (binary) time series, where the number of time series in this set is equal to the number of distinct symbols.
- Map each of these time series into a multidimensional vector using the wavelet transform.

Finally, the features from the different series are combined to create a single multidimensional record.

Discrete Sequence to Numeric Data

Ex: Consider the following sequence, which is drawn on four symbols:
ACACACTGTGACTG

- This series can be converted into the following set of four binary time series corresponding to the symbols A, C, T, and G, respectively:

  10101000001000
  01010100000100
  00000010100010
  00000001010001

- A wavelet transformation can be applied to each of these series to create a multidimensional set of features. The features from the four different series can be appended to create a single numeric multidimensional record.

Spatial to Numeric Data

- Spatial data can be converted to numeric data by using the same approach that was used for time-series data. The main difference is that there are now two contextual attributes (instead of one).

- This requires modification of the wavelet transformation method.

- The approach is fairly general and can be used for any number of contextual attributes.

Graphs to Numeric Data

Graphs can be converted to numeric data with the use of methods such as multidimensional scaling (*MDS*) and spectral transformations.

- This approach works for those applications where the edges are weighted, and represent similarity or distance relationships between nodes.
- A spectral approach can also be used to convert a graph into a multidimensional representation.
- This is also a dimensionality reduction scheme that converts the structural information into a multidimensional representation.

Any Type to Graphs

Many applications are based on the notion of similarity.

- For example: the clustering problem is defined as the creation of groups of similar objects, whereas the outlier detection problem is defined as one in which a subset of objects differing significantly from the remaining objects are identified.

- Many forms of classification models, such as nearest neighbor classifiers, are also dependent on the notion of similarity. The notion of pairwise similarity can be best captured with the use of a *neighborhood graph*.

Any Type to Graphs

For a given set of data objects $O = \{O_1 \cdots O_n\}$, a neighborhood graph is defined as follows:

- A single node is defined for each object in O. This is defined by the node set *N*, containing *n* nodes where the node *i* corresponds to the object O_i.

- An edge exists between O_i and O_j, if the distance *d*(O_i, O_j) is less than a particular threshold. The weight $w_{ij}$of the edge $\{i, j\}$ is equal to a kernelized function of the distance between the objects O_i and O_j, so that larger weights indicate greater similarity. An example is the *heat kernel*:

$$w_{ij} = e^{-d(O_i, O_j)^2/t^2}$$

- Here, *t* is a user-defined parameter.

Data Cleaning

There are several important aspects of data cleaning:

- *Handling missing entries:* Many entries in the data may remain unspecified because of weaknesses in data collection or the inherent nature of the data. The process of estimating missing entries is also referred to as *imputation*.

- *Handling incorrect entries:* In cases where the same information is available from multiple sources, *inconsistencies* may be detected. Such inconsistencies can be removed as a part of the analytical process.

- *Scaling and normalization:* The data may often be expressed in very different scales. This may result in some features being inadvertently weighted too much so that the other features are implicitly ignored.

Handling Missing Entries

Three classes of techniques are used to handle missing entries:

- Any data record containing a missing entry may be eliminated entirely. However, this approach may not be practical when most of the records contain missing entries.
- The missing values may be estimated or imputed. However, errors created by the imputation process may affect the results of the data mining algorithm.
- The analytical phase is designed in such a way that it can work with missing values. Many data mining methods are inherently designed to work robustly with missing values. This approach is usually the most desirable because it avoids the additional biases inherent in the imputation process.

Handling Incorrect and Inconsistent Entries

Key methods:

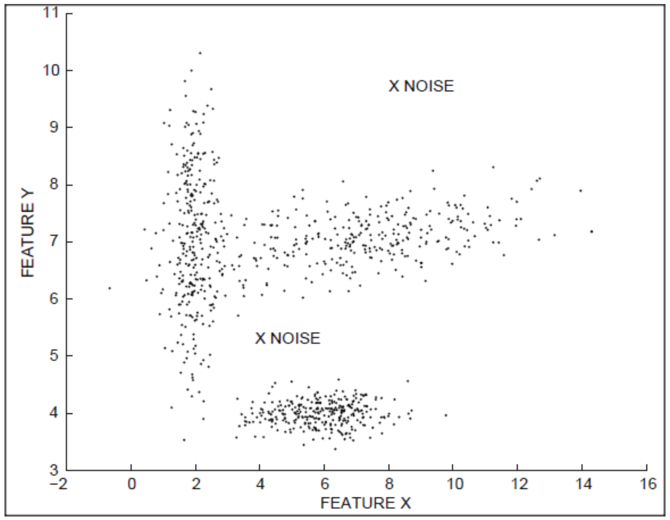
- *Inconsistency detection:* This is typically done when the data is available from different sources in different formats. These topics are studied under the general umbrella of data integration within the database field.

- *Domain knowledge:* A significant amount of domain knowledge is often available in terms of the ranges of the attributes or rules that specify the relationships across different attributes. For example, if the country field is “United States,” then the city field cannot be “Shanghai”. Many *data scrubbing* and *data auditing* tools have been developed that use such domain knowledge and constraints to detect incorrect entries.

Handling Incorrect and Inconsistent Entries

Key methods:

- *Data-centric methods:* In these cases, the *statistical behavior* of the data is used to detect outliers.



- Ex: Isolated points may be caused by errors in the data
- Need to used outlier detection methods

Scaling and Normalization

- Consider the case where the j^{th} attribute has mean μ_j and standard deviation σ_j. Then, the j^{th} attribute value x_i^j of the i^{th} record $\overline{X_i}$ may be normalized as follows:

$$z_i^j = \frac{x_i^j - \mu_j}{\sigma_j}$$

- A second approach uses *min-max scaling* to map all attributes to the range [0, 1]. Let min_j and max_j represent the minimum and maximum values of attribute *j*. Then, the j^{th}attribute value $x_i^j$of the i^{th} record $\overline{X_i}$ may be scaled as follows:

$$y_i^j = \frac{x_i^j - min_j}{max_j - min_j}$$

Data Reduction and Transformation

The reduction of the data may be in terms of the number of rows (records) or in terms of the number of columns (dimensions). Types of data reduction:

- *Data sampling:* The records from the underlying data are sampled to create a much smaller database.
- *Feature selection:* Only a subset of features from the underlying data is used in the analytical process. Typically, these subsets are chosen in an application-specific way.
- *Data reduction with axis rotation:* The correlations in the data are leveraged to represent it in a smaller number of dimensions. Examples of such data reduction methods include principal component analysis (*PCA*), singular value decomposition (*SVD*), or latent semantic analysis (*LSA*) for the text domain.
- *Data reduction with type transformation:* This form of data reduction is closely related to data type portability.

Sampling for Static Data

- *Sampling without replacement*

- *Sampling with replacement*

- *Biased sampling*

  - some parts of the data are intentionally emphasized because of their greater importance to the analysis.

- Ex: temporal-decay bias where more recent records have a larger chance of being included in the sample, and stale records have a lower chance of being included. In exponential-decay bias, the probability *p*(*X*) of sampling a data record *X*, which was generated *δt* time units ago, is proportional to an exponential decay function value regulated by the decay parameter *λ*:

$$p(\bar{X}) \sim e^{-\lambda \cdot \delta t}$$

Sampling for Static Data

Stratified sampling: In some data sets, important parts of the data may not be sufficiently represented by sampling because of their rarity.

- A stratified sample:

  - first partitions the data into a set of desired strata, and then

  - independently samples from each of these strata based on predefined proportions in an application-specific way.

Reservoir Sampling for Data Streams

- A particularly interesting form of sampling is that of *reservoir sampling* for data streams.

- In reservoir sampling, a sample of *k* points is *dynamically* maintained from a data stream.

- For each incoming data point in the stream, one must use a set of efficiently implementable operations to *maintain* the sample.

- In the static case, the probability of including a data point in the sample is *k/n* where *k* is the sample size, and *n* is the number of points in the “data set.”

Reservoir Sampling for Data Streams

- In this case, the “data set” is not static and cannot be stored on disk. Furthermore, the value of *n* is constantly increasing as more points arrive and previous data points (outside the sample) have already been discarded.

- The sampling approach works with *incomplete knowledge* about the previous history of the stream at any given moment in time.

- For each incoming data point in the stream, we need to *dynamically* make two simple *admission control* decisions:

- What sampling rule should be used to decide whether to include the newly incoming data point in the sample?

- What rule should be used to decide how to eject a data point from the sample to “make room” for the newly inserted data point?

Reservoir Sampling for Data Streams

- Sample algorithm for reservoir sampling in data streams

- For a reservoir of size *k*, the first *k* data points in the stream are used to initialize the reservoir. Subsequently, for the *n*th incoming stream data point, the following two admission:

    - Insert the *n*th incoming stream data point into the reservoir with probability *k/n*.

    - If the newly incoming data point was inserted, then eject one of the old *k* data points at random to make room for the newly arriving point.

Feature Subset Selection

There are two primary types of feature selection:

- *Unsupervised feature selection:* This corresponds to the removal of noisy and redundant attributes from the data. Unsupervised feature selection is best defined in terms of its impact on clustering applications, though the applicability is much broader.

- *Supervised feature selection:* This type of feature selection is relevant to the problem of data classification. In this case, only the features that can predict the class attribute effectively are the most relevant. Such feature selection methods are often closely integrated with analytical methods for classification.

Dimensionality Reduction with Axis Rotation

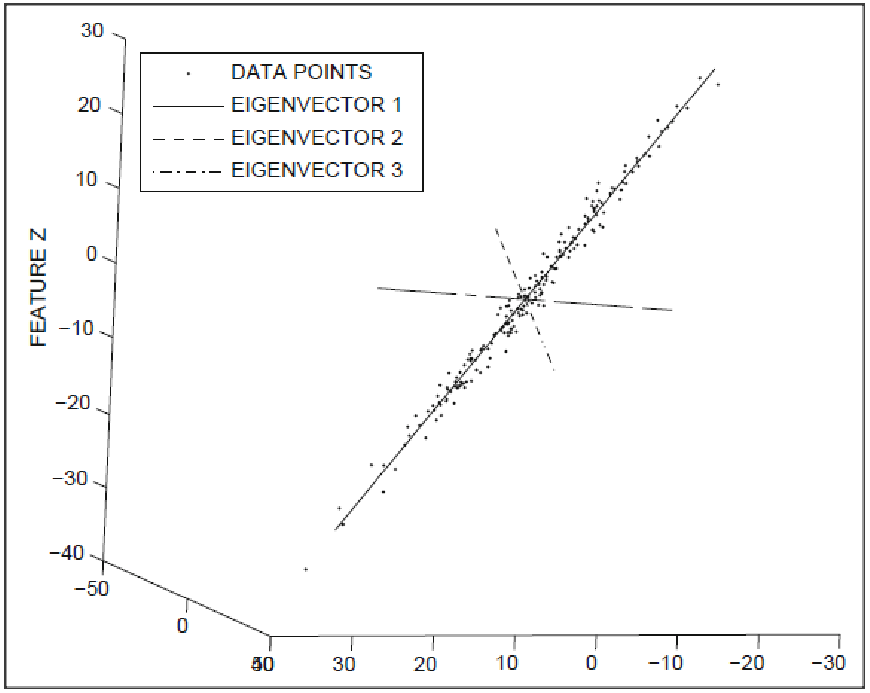
- In real data sets, a significant number of correlations exist among different attributes. In some cases, hard constraints or rules between attributes may uniquely define some attributes in terms of others.

- These correlations and constraints correspond to implicit redundancies because they imply that knowledge of some subsets of the dimensions can be used to predict the values of the other dimensions.

Dimensionality Reduction with Axis Rotation

- Ex: consider the 3-dimensional data set. In this case,
if the axis is rotated to the orientation illustrated, the correlations and redundancies in the newly transformed feature values are removed.

- As a result of this redundancy removal, the entire data can be (approximately) represented along a 1-dimensional line. Thus, the *intrinsic dimensionality* of this 3-dimensional data set is 1.



Dimensionality Reduction with Axis Rotation

A natural question arises as to how the correlation-removing axis system such as that in previous example may be determined in an automated way.

Two natural methods to achieve this goal:

- *principal component analysis (PCA)*
- *singular value decomposition (SVD)*.

These two methods, while not exactly identical at the definition level, are closely related. Although the notion of principal component analysis is intuitively easier to understand, *SVD* is a more general framework and can be used to perform *PCA* as a special case.

Principal Component Analysis

- *PCA* is generally applied after subtracting the mean of the data set from each data point. This operation is referred to as *mean centering*, and it results in a data set centered at the origin.

- The goal of *PCA* is to rotate the data into an axis-system where the greatest amount of variance is captured in a small number of dimensions.

- An important observation is that the variance of a data set along a particular direction can be expressed directly in terms of its covariance matrix.

Principal Component Analysis

- Let C be the $d \times d$ symmetric covariance matrix of the $\mathrm{n} \times d$ data matrix D. Thus, the $(i,j)^{th}$ entry c_{ij} of C denotes the covariance between the *i*th and *j*th columns (dimensions) of the data matrix *D*. Let *μ*μ_i represent the mean along the *i*th dimension. Specifically, if x_k^m be the *m*th dimension of the *k*th record, then the value of the covariance entry c_{ij} is as follows:

$$c_{ij} = \frac{\sum_{k=1}^{n} x_k^i x_k^j}{n} - \mu_i \mu_j, \qquad i, j \in \{1 \cdots d\}$$

- Let $\bar{\mu} = (\mu_1 \cdots \mu_d)$ means vector, then the covariance matrix;

$$C = \frac{D^T D}{n} - \bar{\mu}^T \bar{\mu}$$

Principal Component Analysis

- The covariance matrix C is positive semi-definite, because it can be shown that for any *d*-dimensional column vector $\bar{v}$, the value of $\bar{v}^T C \bar{v}$ is equal to the variance of the 1-dimensional projection $D\bar{v}$ of the data set D on $\bar{v}$.

$\bar{v}^T C \bar{v} = \frac{(D\bar{v})^T D\bar{v}}{n} - (\bar{\mu}\bar{v})^2$ = Variance of 1-dimensional points in $D\bar{v} \geq 0$

- How can one determine such directions?

- Because the covariance matrix is symmetric and positive semidefinite, it can be diagonalized as follows:

$$C = P\Lambda P^T$$

Principal Component Analysis

- The columns of the matrix P contain the orthonormal eigenvectors of C, and Λ is a diagonal matrix containing the nonnegative eigenvalues.

- The entry Λ_{ii} is the eigenvalue corresponding to the *i*th eigenvector (or column) of the matrix P.

- These eigenvectors represent successive orthogonal solutions to the previous optimization model maximizing the variance $\bar{v}^T C \bar{v}$ along the unit direction $\bar{v}$.

Principal Component Analysis

- *The greatest variance-preserving directions are also the correlation-removing directions*. The eigenvalues represent the variances of the data along the corresponding eigenvectors. In fact, the diagonal matrix Λ is the new covariance matrix after axis rotation.

- Therefore, eigenvectors with large eigenvalues preserve greater variance, and are also referred to as *principal components*.

- Because of the nature of the optimization formulation used to derive this transformation, a new axis system containing only the eigenvectors with the largest eigenvalues is optimized to *retaining the maximum variance in a fixed number of dimensions*.

Principal Component Analysis

- Without loss of generality, it can be assumed that the columns of *P* (and corresponding diagonal matrix Λ) are arranged from left to right in such a way that they correspond to decreasing eigenvalues.

- The transformed data matrix *D* in the new coordinate system after axis rotation to the orthonormal columns of *P* can be algebraically computed as the following linear transformation:

$$D' = DP$$

Singular Value Decomposition

- Singular value decomposition (*SVD*) is closely related to principal component analysis (*PCA*).

- *SVD* is more general than *PCA* because it provides *two* sets of basis vectors instead of one. *SVD* provides basis vectors of both the rows and columns of the data matrix, whereas *PCA* only provides basis vectors of the rows of the data matrix.

- Furthermore, *SVD* provides the same basis as *PCA* for the rows of the data matrix in certain special cases:

- SVD provides the same basis vectors and data transformation as PCA for data sets in which the mean of each attribute is 0.

Singular Value Decomposition

- A formal way of defining *SVD* is as a decomposable product of (or *factorization* into) three matrices:

$$D = Q\Sigma P^T$$

- where, *Q* is an *n* × *n* matrix with orthonormal columns, which are the *left singular vectors*.

- Σ is an *n*×*d* diagonal matrix containing the *singular values*, which are always nonnegative and, by convention, arranged in nonincreasing order.

- *P* is a *d*×*d* matrix with orthonormal columns

Singular Value Decomposition

Let P_k and Q_k be the truncated *d* × *k* and *n* × *k* matrices obtained by selecting the first *k* columns of *P* and *Q*, respectively. Let Σ_k be the $k \times k$ square matrix containing the top *k* singular values.

- Then, the *SVD* factorization yields an *approximate d*-dimensional data representation of the original data set *D*:

$$D \approx Q_k \Sigma_k P_k$$

- The columns of P_k represent a *k*-dimensional basis system for a reduced representation of the data set.

Latent Semantic Analysis

- Latent semantic analysis (*LSA*) is an application of the *SVD* method to the text domain.

- In this case, the data matrix *D* is an $n \times d$ document-term matrix containing normalized word frequencies in the *n* documents, where *d* is the size of the lexicon.

- No mean centering is used, but the results are approximately the same as *PCA* because of the sparsity of *D*. The sparsity of *D* implies that most of the entries in *D* are 0, and the mean values of each column are much smaller than the nonzero values

- In the text domain, the reduction in dimensionality from *LSA* is rather drastic.

  - For example, it is not uncommon to be able to represent a corpus drawn on a lexicon of 100,000 dimensions in fewer than 300 dimensions.

Dimensionality Reduction with Type Transformation

In these methods, dimensionality reduction is coupled with type transformation. In most cases, the data is *transformed* from a more complex type to a less complex type, such as multidimensional data.

- *Time series to multidimensional:* A number of methods, such as the discrete Fourier transform and discrete wavelet transform are used. While these methods can also be viewed as a rotation of an axis system defined by the various time stamps of the contextual attribute, the data are no longer dependency oriented after the rotation. Therefore, the resulting data set can be processed in a similar way to multidimensional data.

- *Weighted graphs to multidimensional:* Multidimensional scaling and spectral methods are used to embed weighted graphs in multidimensional spaces, so that the similarity or distance values on the edges are captured by a multidimensional embedding.

STEVENS
INSTITUTE of TECHNOLOGY
School of Business

Thank You

Dragos Bozdog