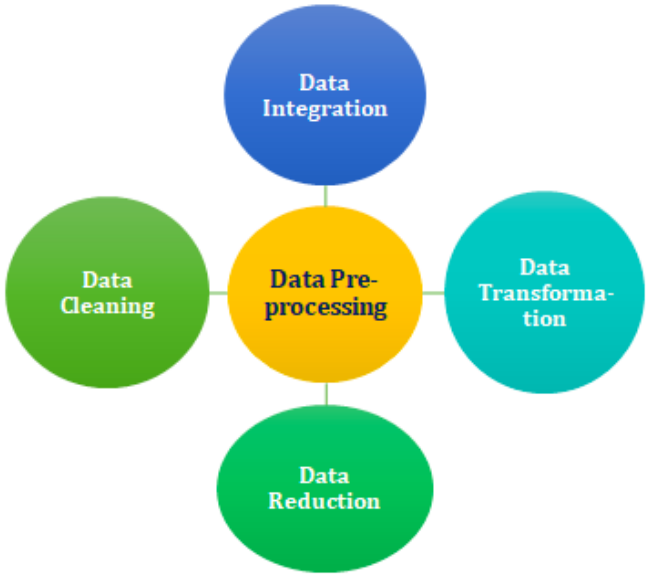
DI501 Introduction to Data Informatics

Lecture 4 – Data Preprocessing – Part IV

Major Tasks in Data Preprocessing

- Data integration
  - Integration of multiple databases, data cubes, or files
- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data reduction
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- Data transformation and data discretization
  - Normalization
  - Concept hierarchy generation



Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values
- Methods
  - Smoothing: Remove noise from data
  - Attribute/feature construction
    - New attributes constructed from the given ones
  - Aggregation: Summarization, data cube construction
  - Normalization: Scaled to fall within a smaller, specified range
    - min-max normalization
    - z-score normalization
    - normalization by decimal scaling
  - Discretization: Concept hierarchy climbing

Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one
- Methods
  - normalization by decimal scaling
  - Discretization: Concept hierarchy climbing

Why do we need data transformation?

Consider the following sample from a dataset:

A1	A2	A3
1.	0.1	1000
2.	0.2	1100
2.	0.1	1700
3.	0.8	2000

Data Transformation

Definitions

- Rescaling a vector means to add or subtract a constant and then multiply or divide by a constant, as you would do to change the units of measurement of the data, for example, to convert a temperature from Celsius to Fahrenheit.
- Normalizing a vector most often means dividing by a norm of the vector. It also often refers to rescaling by the minimum and range of the vector, to make all the elements lie between 0 and 1 thus bringing all the values of numeric columns in the dataset to a common scale.
- Standardizing a vector most often means subtracting a measure of location and dividing by a measure of scale.
  - For example, if the vector contains random values with a Gaussian distribution, you might subtract the mean and divide by the standard deviation, thereby obtaining a “standard normal” random variable with mean 0 and standard deviation 1.

Rescaling

- The goal of applying Feature Scaling is to make sure features are on almost the same scale so that each feature is equally important and make it easier to process by most ML algorithms.
- Converting a feature vector in ”milimeters” into “meters”.
- 2 times scaling:
  - {1,2,3}->{2,4,6}
- Map scales:
  - 'one centimetre to one hundred metres' or 1:10,000 or 1/10,000

Normalization

- Min-max normalization to $[new\_minA,\ new\_maxA]$:
  - $v' = \frac{v - minA}{maxA - minA}\left(new\_max_A - new\_min_A\right) + new\_\text{minA}$
  - Ex: Let income range \$12,000 to \$98,00 normalized to $[0.0,\ 1.0]$. Then \$73,600 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 + 0) + 0 = 0.716$
- z-score normalization – standardization:
  - $v' = \frac{v - \mu}{\sigma}$ (μ: mean, σ: std. dev.)
  - Ex: Let μ=54,000, σ=16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$
- Normalization by decimal scaling
  - $v' = \frac{v}{10^j}$ where j is the smallest integer such that $\max\left(|v'|\right) < 1$

Normalization

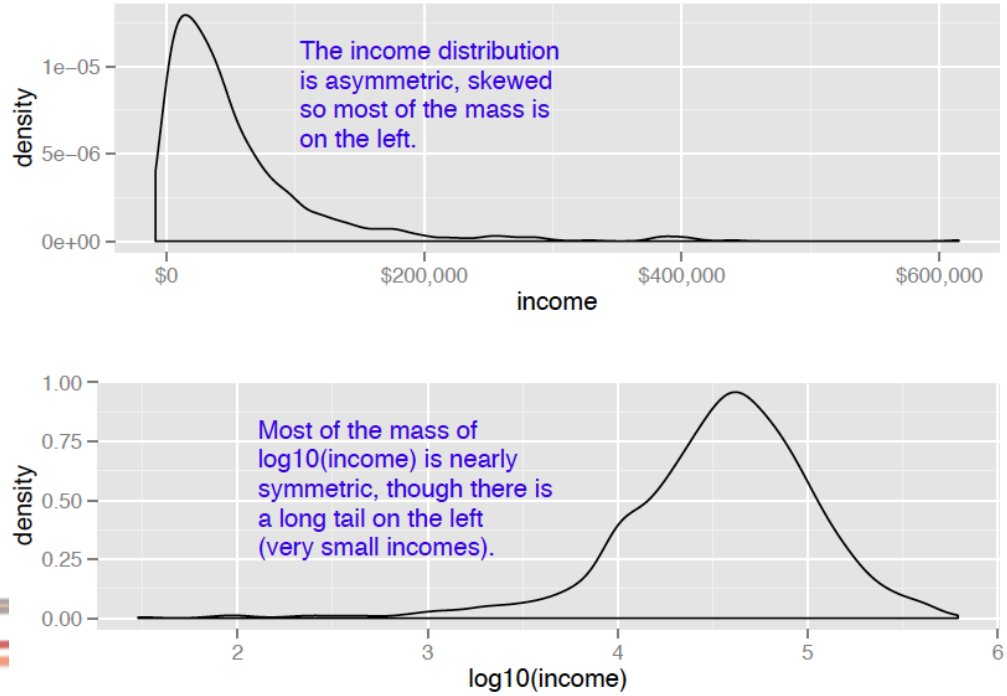
- Min-max normalization to $[new\_minA,\ new\_maxA]$:
  - $v' = \frac{v - minA}{maxA - minA}\left(new\_max_A - new\_min_A\right) + new\_\text{minA}$
  - Ex: Let income range \$12,000 to \$98,000 normalized to $[0.0,\ 1.0]$. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 + 0) + 0 = 0.716$
- z-score normalization – standardization:
  - $v' = \frac{v - \mu}{\sigma}$
  - Ex: Let μ=54,000

Normalizing data by mean and standard deviation is most meaningful when the data distribution is roughly symmetric otherwise not.

- Normalization by decimal scaling
  - $v' = \frac{v}{10^j}$ where j is the smallest integer such that $\max\left(|v'|\right) < 1$

Normalization

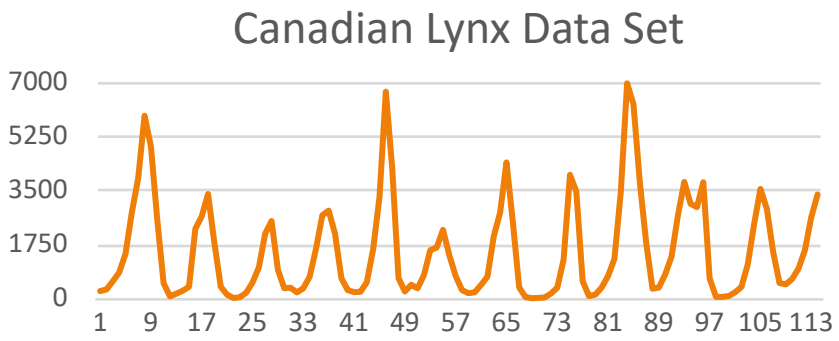
- Log transformation: Monetary amounts—incomes, customer value, account or purchase sizes—are some of the most commonly encountered sources of skewed distributions (such as lognormal), so taking the log of the data can restore symmetry.

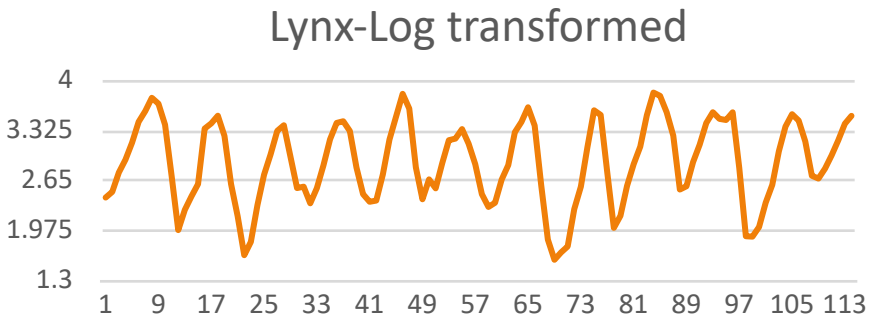


Normalization

- *A log transformation example:* Canadian Lynx data set is normalized using $\log_{10}$ and the normalized values are utilized for further processing.

If a logarithmic transformation is applied to a distribution, the differences between smaller values will be expanded (because the slope of the algorithmic function is steeper when values are small) whereas the differences between larger values will be reduced (because of the very moderate slope of the log distribution for larger values).





Normalization

- Scaling to median and quantiles:

$$v' = \frac{(v - medianA)}{IQRA}$$

  - i.e., subtract the median from all the observations and then divide by the interquartile difference.
  - This transformation scales features using statistics that are robust to outliers.

Normalization

- Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve).
  - Normalization is useful when your data has varying scales and the algorithm you are using does not make any assumptions about the distribution of your data, such as k-nearest neighbors and artificial neural networks.
- Standardization assumes that your data has a Gaussian distribution. This does not strictly have to be true, but the technique is more effective if your attribute distribution is Gaussian.
  - Standardization is useful when your data has varying scales and the algorithm you are using does make assumptions about your data having a Gaussian distribution, such as linear regression, logistic regression and linear discriminant analysis.

Discretization

- Often data are given in the form of continuous values.
- If their number is huge, model building for such data can be difficult.
- Many data mining algorithms operate only in discrete search or variable space.
- For instance, decision trees typically divide the values of a variable into two parts according to an appropriate threshold value.
- The goal of discretization is to reduce the number of values by grouping them into a number, b, of intervals or bins.

Discretization

- Discretization: Divide the range of a continuous attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Reduce data size by discretization
  - Supervised vs. unsupervised
  - Split (top-down) vs. merge (bottom-up)
  - Discretization can be performed recursively on an attribute
  - Prepare for further analysis, e.g., classification

Data Discretization Methods

- Unsupervised discretization: does not require the class information to discretize continuous attributes.
  - Binning
    - Top-down split, unsupervised
  - Histogram analysis
    - Top-down split, unsupervised
  - Clustering analysis (unsupervised, top-down split or bottom-up merge)
  - The best number of bins is determined experimentally.

Data Discretization Methods

- Supervised discretization (bivariate grouping): makes use of the class label when partitioning the continuous features.
  - Decision-tree analysis (supervised, top-down split)
  - Chi-merge analysis
  - Entropy based discretization
  - It tries to maximize the “purity” of the intervals (i.e. to contain as less as possible mixture of class labels)

Simple Discretization:

Binning

- Equal-width (distance) partitioning
  - Divides the range into N intervals of equal size: uniform grid
  - if A and B are the lowest and highest values of the attribute, the width of intervals will be: W = (B –A)/N.
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well
- Equal-depth (frequency) partitioning
  - Divides the range into N intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky

Binning Methods for Data Smoothing

- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
- Partition into equal-frequency (equi-depth) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
- Smoothing by bin means:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
- Smoothing by bin boundaries:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Equi-width:
34-4=30
30/3=10 for each bin the width is
Bin 1: 4-13
Bin 2: 14-23
Bin 3: 24-34

Binning

Jenks Natural Break

- The Jenks optimization method, also called the Jenks natural breaks classification method, is a data clustering method designed to determine the best arrangement of values into different classes.
- This is done by seeking to minimize each class's average deviation from the class mean, while maximizing each class's deviation from the means of the other classes.
- In other words, the method seeks to reduce the variance within classes and maximize the variance between classes.

Ref: https://en.wikipedia.org/wiki/Jenks_natural_breaks_optimization

Discretization by Classification & Correlation Analysis

- Classification (e.g., decision tree analysis)
  - Supervised: Given class labels, e.g., malignant vs. benign
  - Using entropy to determine split point (discretization point)
  - Top-down, recursive split
- Correlation analysis (e.g., Chi-merge: χ2-based discretization)
  - Supervised: use class information
  - Bottom-up merge: find the best neighboring intervals (those having similar distributions of classes, i.e., low χ2 values) to merge
  - Merge performed recursively, until a predefined stopping condition

Discretization by Classification & Correlation Analysis

- Chi-merge: first sort the training examples according to their value for the attribute being discretized and then constructing the initial discretization, in which each example is put into its own interval.
- Example: Consider iris data set and we will discretize on sepal length attribute. We have three classes: setosa, versicolor, virginica

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa

Discretization by Classification & Correlation Analysis

- Chi-merge: first sort the training examples according to their value for the attribute being discretized and then constructing the initial discretization, in which each example is put into its own interval

The formula for computing the χ^2 value is:

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

Where:

$m = 2$ (the 2 intervals being compared)
$k =$ number of classes
$A_{ij} =$ number of examples in ith interval, jth class.
$R_i =$ number of examples in ith interval $= \sum_{j=1}^{k} A_{ij}$
$C_j =$ number of examples in jth class $= \sum_{i=1}^{m} A_{ij}$
$N =$ total number of examples $= \sum_{j=1}^{k} C_j$
$E_{ij} =$ expected frequency of $A_{ij} = \frac{R_i * C_j}{N}$

Discretization by Classification & Correlation Analysis

- Example: Consider iris data set and we will discretize on sepal length attribute. The first column is the lower bound of that interval. The next three numbers in each row comprise the class frequency vector, which represents how many examples of each class are in that interval (the order is setosa, versicolor, virginica).

Int	Class frequency			χ^2
4.3	16	0	0	4.1
4.9	4	1	1	2.4
5.0	25	5	0	8.6
5.5	2	5	0	2.9
5.6	0	5	1	1.7
5.7	2	5	1	1.8
5.8	1	3	3	2.2
5.9	0	12	7	4.8
6.3	0	6	15	4.1
6.6	0	2	0	3.2
6.7	0	5	10	1.5
7.0	0	1	0	3.6
7.1	0	0	12	

Int	Class frequency			χ^2
4.3	45	6	1	30.9
5.5	4	15	2	6.7
5.8	1	15	10	4.9
6.3	0	14	25	5.9
7.1	0	0	12	

ChiMerge discretizations for sepal-length at the .50 and .90 significance levels (x2 = 1.4 and 4.6)

Concept Hierarchy Generation

- Concept hierarchy organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies facilitate drilling and rolling in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for age) by higher level concepts (such as youth, adult, or senior)
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers
- Concept hierarchy can be automatically formed for both numeric and nominal data—For numeric data, use discretization methods shown

Automatic Concept Hierarchy Generation

- More examples:

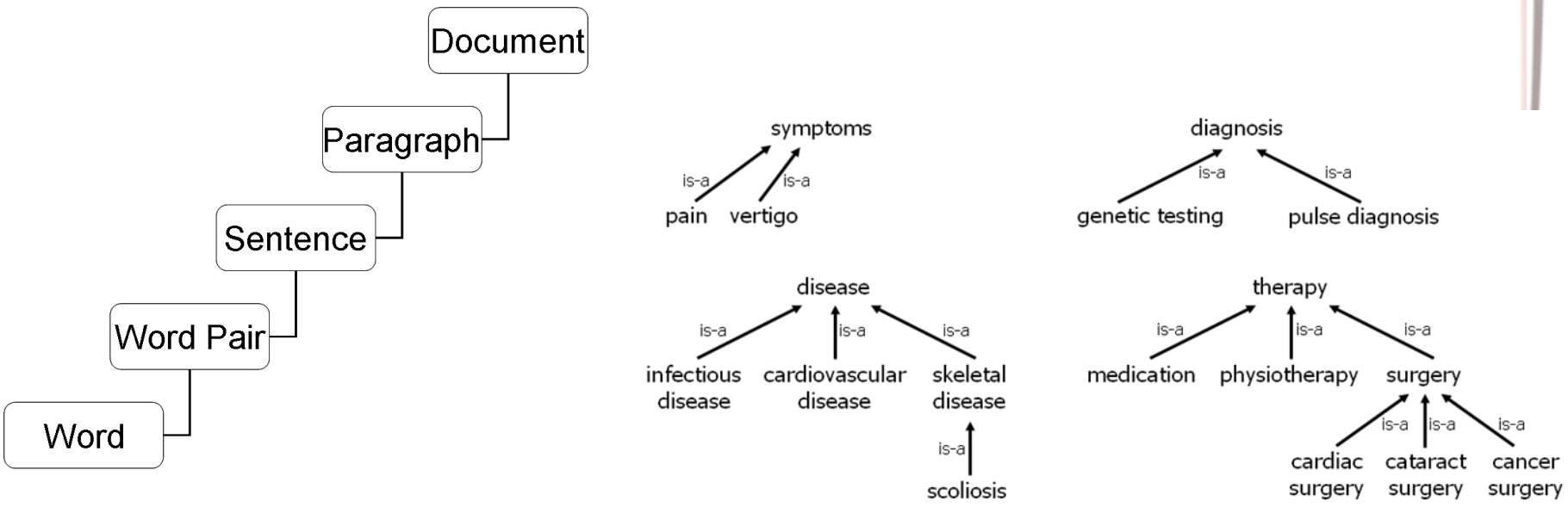


Figure 1: Concept hierarchy (taxonomy)