

Predictive Analytics

Chapter Three

Data Preparation Algorithms

“Before anything else, preparation is the key to success.” Alexander Graham Bell
“In all cases, you end up spending more than 80% of your data analytics lifecycle’ time
preparing your data” A.Bari

For *SVD* and *data reduction sections*, the assigned reading is **Chapter Eleven** from the Book (Ullman’s Book – www.mmds.org)

Anasse Bari, Ph.D.

Copyrights @ Anasse Bari

Learning Outcomes

- Understanding most widely used algorithms for preparing data for analytics
- Learning Data Reduction Algorithms
- Learning the Applications of Singular Value Decomposition
- Acquiring basic hands-on to [R](#) and [Knime](#).

Outline

- Problems with Raw Data
- Introduction to Data Cleaning
- Introduction to Data Preprocessing
- Data Reduction Algorithms
 - Feature Reduction using Missing Values Ratio (MVR)
 - Feature Reduction using Feature' Variance Threshold (FVT)
 - Feature Reduction using Feature Correlation Threshold (FCT)
 - Principal Component Analysis (PCA)
 - Singular Value Decomposition (SVD)
- Data Fusion
 - Joins and Hash Joins
- Data Transformation
- Data Discretization

Outline

Practice One: DataPreparationPractice-1 Handling Missing Values with RapidMiner

[Practice Two: DataPreparationPractice-2 Reducing Data Dimensionality with Knime](#)
[Interview](#)

Practice Three: DataPreparationPractice-3 PCA with RapidMiner

Practice Four: DataPreparationPractice-4 PCA with R

Real Problems with Raw Data

Data can be

- Incomplete
Missing attributes
- Inconsistent
Different representation of attributes
- Noisy
Errors or outliers

Customer Name	Occupation	Zip Code	Customer Type
Jennifer	IT Consultant	20037	Loyal
James	Teacher	DC	Discount
Bari		20052	-10
...

Raw Data

Data can be

- Incomplete data could be generated the time data being collected by human, or due to software or hardware errors
- Inconsistent data could be generated from Different data sources
- Noisy data could be the result of
Errors in Data Entry Process
Errors in Data Collection
Errors in Data Transmission

Customer Name	Occupation	Zip Code	Customer Type
Jennifer	IT Consultant	20037	Loyal
James	Teacher	DC	Discount
Bari		20052	-10
...

Importance of Data Preprocessing

- Low quality data leads to bad and wrong extracted insights.
- Duplicates, errors and noise in data leads to wrong predictions.
- The accuracy of a Predictive or Data Analytics Model depends on the quality of the training and test data.
- From a data warehouse perspective: **“Data extraction, cleaning, and transformation comprise the majority of the work of building a data warehouse”** *Bill Inmon* best known as *The Father of Data Warehousing*.

A data warehouse is a large store of data accumulated from a wide range of sources within a an organization, company or a business.

Data Preprocessing Techniques

- Data Cleaning
- Data Integration (Data Fusion)
- Data Transformation
- Data Reduction
- Data Discretization
- Data Compression

Data Preprocessing Techniques (cont'd)

- Data Cleaning

 - Fill in missing values

 - Smoothing noise data

 - Extract and remove outliers

 - Correct inconsistencies

- Data Integration (Data Fusion)

 - Integration and fusion of data from multiple data sources (heterogeneous data sources)

- Data Transformation

 - Normalization and fusion of heterogeneous data

- Data Reduction

 - Reduce input data in volume while maintaining the same structure and quality.

- Data discretization

 - a process of converting continuous data attribute values into a finite set of intervals with minimal loss of information.

Data Cleaning – (1) Dealing with Missing Data

- Data cleaning is an essential task in data analytics.
- Missing Data:
 - Data has missing values
 - Data might not be always available (many data objects have no recorded value for several attributes – such as zip code as an instance)
 - Data entry errors: some data may not have been considered as important at the time of entry
 - Data is not available for some data objects
- **Handling Missing Data:** (One Solution: FILL IN the missing data but NOT practical)
 - Fill in missing values manually
 - Fill in missing values automatically
 - (More on next slide)

Data Cleaning – (1) Dealing with Missing Data

(Cont'd)

Fill in the missing values manually: Feasibility?

Fill in the missing values automatically possibly with:

- A global constant: “Unknown” or new class.
- The mean values of that attribute across all the data objects
- The most probable value: Inference-based (e.g. Bayesian, Decision trees)

Practicing Data Cleaning

Practice One: DataPreparationPractice-1 RapidMiner

Data Cleaning – (2) Dealing with Noise_(Cont'd)

- Noise: Random Error
- Noise can be caused by:
 - Inconsistency in naming conventions
 - Data transmission problems
 - Data entry problems
- **Handling noise:**
 - Binning method:
 - Sort data and partition into equal depth bins then
 - Smooth by bin means
 - Smooth by bin median
 - Smooth by bin boundaries

Data Cleaning – (2) Dealing with Noise_(Cont'd)

- **Handling noise (cont'd):**

- Data Clustering

- Use the algorithms that detect outliers

- Human Inspection

- Regression: smoothing by fitting the data into regression function

Data Discretization Methods (Cont'd)

- **Binning methods**

- They smooth a sorted data value by consulting its “neighborhood”, that is the values around it.
- The sorted values are partitioned into a number of buckets or bins.
- Smoothing by bin means: *Each value in the bin is replaced by the mean value of the bin.*
- Smoothing by bin medians: *Each value in the bin is replaced by the bin median.*
- Smoothing by boundaries: The min and max values of a bin are identified as the bin boundaries.
Each bin value is replaced by the closest boundary value.

Data Discretization Methods

- **Binning:** Simple discretization method

Data discretization is defined as a process of converting continuous data attribute values into a finite set of intervals with. minimal loss of information)

- Equal-width 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:
$$W = (B - A) / N$$
- This is the most straightforward approach, but outliers may dominate presentation.

Binning Example

- 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 (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

Data Fusion

- Data fusion is also known as Data integration
- Integrate data from different data sources into a coherent data store
- Schema integration, metadata from different sources
- Challenges: Different representation, schemas and representation, data redundancy.
- Joins and Hash Joins.

Handling Data Redundancy:

- Redundant data occur often when integration of multiple databases.
- The same attribute may have different names in different databases.
- One attribute may be a “derived” attribute in another table, e.g., annual revenue.
- Redundant data may be able to be detected by correlational analysis.
- Careful integration of the data from multiple sources may help **reduce/avoid redundancies and inconsistencies and improve mining speed and quality.**

Hash Joins Vs. SQL Joins

- SQL traditionally relies on the use of merge joins
- Joins involve sorting the data in each of the two tables to be joined and then going through the two tables to complete the merge.
 - At each step of the merge algorithm, the key associated with the current record in the first table is compared with the key associated with the current record in the second table and the algorithm makes a determination about whether the join should be completed.
 - If a join is successful or the algorithm determines that no join can be completed, the algorithm advances to the next record in the respective table, and the algorithm continues until end of the tables is reached.
- SQL Joins can be *computationally costly* where the tables to be joined are large.
- **Hash joins** may offer a significant advantage.

Hash Joins Vs. SQL Joins

- Joins involve sorting the data in each of the two tables to be joined and then going through the two tables to complete the merge.
 - At each step of the merge algorithm, the key associated with the current record in the first table is compared with the key associated with the current record in the second table and the algorithm makes a determination about whether the join should be completed.
 - If a join is successful or the algorithm determines that no join can be completed, the algorithm advances to the next record in the respective table, and the algorithm continues until end of the tables is reached.
- SQL Joins can be *computationally costly* where the tables to be joined are large.
- **Hash joins** may offer a significant advantage.

Hash Joins: Data Fusion's Backbone

- Recall that distrusted architectures existed before Google and MapReduce.
- Hash Joins existed before Hadoop and MapReduce. Nowadays, most Relational Database Management Systems (RDBMS) offer Hash Joins instead of SQL Joins. Hash Joins is method for joining large data sets.
- Hash joins are an efficient way of merging two tables (say, A and B).
- Hash joins usually take the smaller of the two tables (say, A, known as the build input) and use a hash function to assign each of the keys of this table to a corresponding bucket in a hash table.
- Then, for each record in the larger table (say, B, known as the probe input) the hash join looks up the record's key in the hash table and retrieves the corresponding record from table A.

More on Hash Joins: https://docs.oracle.com/database/121/TGSQL/tgsql_join.htm#TGSQL242

Hash Joins Example

Employee Table

Employee.Name	Employee.Id
John	125
Lucy	100
May	156
Alice	144
Jack	167

Salary Table

Salary	Employee.Id
5000	125
6000	100
7000	199
5500	144
6500	137
7500	127
9500	198
10000	167

More on Hash Joins from Oracle: https://docs.oracle.com/database/121/TGSQL/tgsql_join.htm#TGSQL242

Hash Joins Example

Employee Table

Employee.Name	Employee.Id
John	125
Lucy	100
May	156
Alice	144
Jack	167

Salary Table

Salary	Employee.Id
5000	125
6000	100
7000	199
5500	144
6500	137
7500	127
9500	198
10000	167

Hash Function: $Hashvalue = keyvalue \bmod 10$

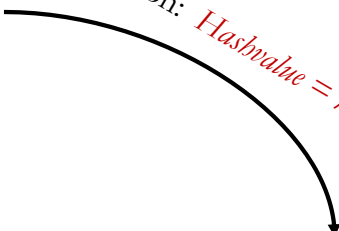
More on Hash Joins from Oracle: https://docs.oracle.com/database/121/TGSQL/tgsql_join.htm#TGSQL242

Hash Joins Example

Employee Table

Employee.Name	Employee.Id
John	125
Lucy	100
May	156
Alice	144
Jack	167

Hash Function: $\text{Hashvalue} = \text{keyvalue} \bmod 10$



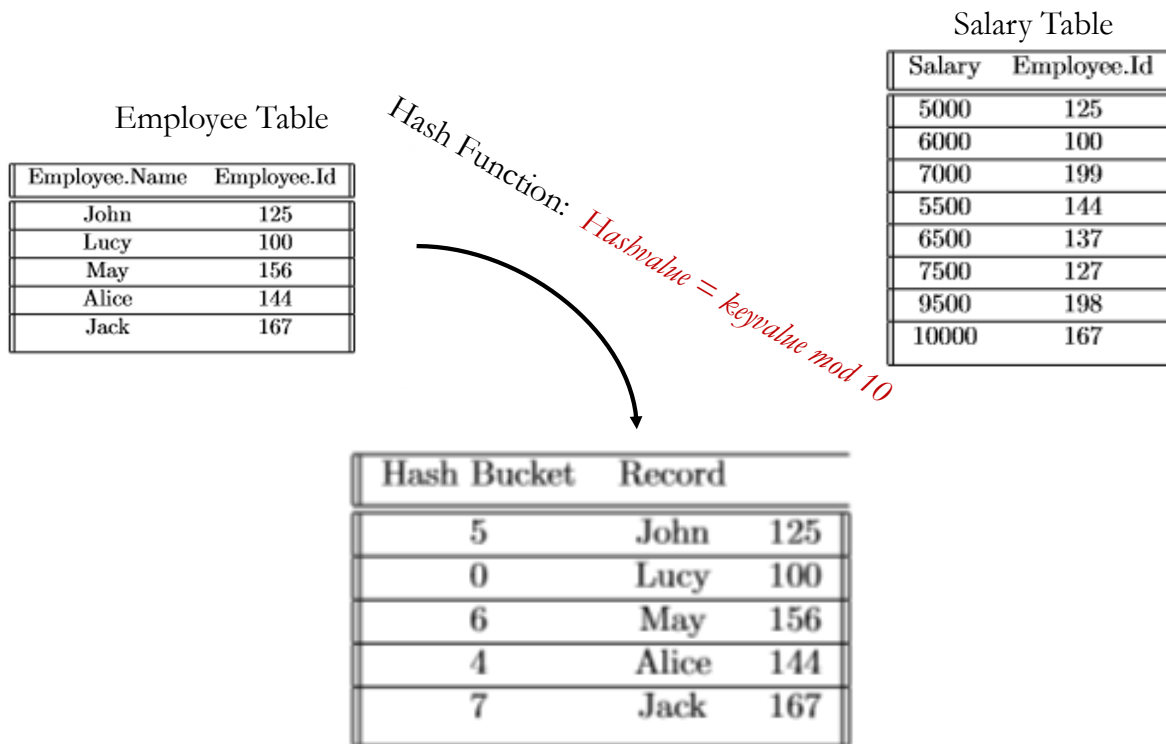
Hash Bucket	Record	
5	John	125
0	Lucy	100
6	May	156
4	Alice	144
7	Jack	167

Salary Table

Salary	Employee.Id
5000	125
6000	100
7000	199
5500	144
6500	137
7500	127
9500	198
10000	167

More on Hash Joins from Oracle: https://docs.oracle.com/database/121/TGSQL/tgsql_join.htm#TGSQL242

Hash Joins Example



The salary table will get scanned one record by one record using the same hash-code function to see if there is same record in regard to employee.id.

It will first calculate employee.id for 125, which returns 5 as hash value.

Then it looks into the hash table to see if there is hash value as 5. If it does, then it checks if there is the same record. For this special case, we found that in the hash table bucket5, there is a same record as 125. Therefore, the Hash join algorithm will add this record in the final output and the scan will continue accordingly... Consider collision handling linear probing chaining... -- See hash table data structure collision handling

More on Hash Joins from Oracle: https://docs.oracle.com/database/121/TGSQL/tgsql_join.htm#TGSQL242

Data Transformation

- Smoothing: remove noise from data
- Aggregation: summarization, fusion of data from multiple data sources
- Normalization: scale data to fall within a small, specified range
- Handling **Normalization**:
 - min-max normalization approach
$$v' = \frac{v - \min A}{\max A - \min A} (\text{new_max}A - \text{new_min}A) + \text{new_min}A$$
 - Z-score normalization
$$v' = \frac{v - \text{mean}A}{\text{stand_dev}A}$$
 - Normalization by decimal scaling
$$v' = \frac{v}{10^j}$$

Where j is the smallest integer such that $\text{Max}(|v'|) < 1$

Data Reduction

“The dilemma of transforming a dataset into a much smaller one in terms of volume, and yet preserve most of the information contained in the original data.” A.Bari

Introducing the Concept of Data Reduction

- **Problem Definition**

- (1) Data may not fit in memory or in storage
- (2) Data mining algorithms take longer computational time when they run on large data versus small data

- **Goal of a Data Reduction Algorithm**

Generate a reduced representation of the dataset that is much smaller in volume yet preserve the structure and the information with the data. Applying the same analytics to the data or the reduced data should produce the same or almost the same insights or forward-insights.

Introducing the Concept of Data Reduction

(continued)

- Dimensionality reduction—remove unimportant features with low prediction capacity
- Related Discipline: Data Compression

Introducing the Concept of Data Reduction

(continued)

- Related Discipline to Data Analytics: **Data Compression**
- Data Compression is a process of deriving more compact (i.e., smaller) representations of data
- Goal of Data Compression is to have a significant reduction in the data size to reduce storage requirement.
- Main Challenges: Perfect or near-perfect reconstruction
 - Lossless compression and Lossy compression
- Strategies for Compression
 - Reducing redundancies

Lossless Compression Algorithms

- **Huffman Coding**
- **Run-length Encoding**
- **Golomb Coding**
- **Arithmetic coding**
- **Lempel-ziv compression**
- **Differential Pulse-code modulation**
- **Bitplane coding**

Dimensionality Reduction: Preliminary Techniques (1)

■ Feature Filtration using Missing Values Ratio (MVR)

Goal

Filter and eliminate columns with too many missing values

Technique

Pick a ratio of observations to missing values.

Eliminate variables whose percentage of missing values falls below MVR' threshold.

The higher the threshold, the more aggressive the reduction.

Note: Intuitively, this is justified by the fact that columns with many missing values may not contain much information, and by the fact that interpolating so many missing values might actually introduce unwanted bias into your dataset and the analytics process.

Dimensionality Reduction: Preliminary Techniques (2)

■ Feature Filtration using Feature' Variance Threshold (FVT)

Goal

Filter and eliminate columns with variance below a certain cutoff (Feature variance threshold - FVT)

Technique

Calculate the variance for each column (feature).

Eliminate Features below a certain threshold (e.g. a value below the mean of variances across the all variances of columns)

The higher the threshold, the more aggressive the reduction.

Note: **Columns with very low variance are not good predictors** because they don't contain a lot of information (their values are all essentially similar). For example, imagine trying to predict whether someone has a cold using a column where "sore throat" is always equal to 1 (zero variance). This column does not provide any information that allows us to differentiate between sick and healthy people.

Dimensionality Reduction: Preliminary Techniques (3)

▪ Feature Reduction using High Feature Correlation Threshold (FCT)

Goal

Filter and eliminate columns and features that are highly correlated with other columns in the dataset ((i.e. pairs of columns that exceed a certain correlation threshold). Since both columns are closely related, we can think of them as columns that stores duplicate information, and hence one column can be removed.

Technique

Calculate the correlation matrix of features per features

Pick a threshold and eliminate features of high correlation threshold

Note: For example, imagine trying to predict whether someone has a cold using columns like “congested” and “runny nose.” These columns both capture the same variation (one usually implies the other) and so one can be removed to reduce the size of the dataset without losing a lot of information.

Practicing Dimensionality Reduction: Preliminary Techniques

DataPreparationPractice-2 Knime (Hands-on – see attached practice under chapter three)

Understanding Data Reduction

- Reducing the dimensionality of the data is about transforming the data and discovering a new axis to represent the data in new coordinates.
- For instance, rather than representing every data observation with two coordinates, a data reduction technique might seek presenting every point with one coordinate.
- *Principle Components Analysis* is one algorithm that is widely used for data reduction.

P rincipal Components Analysis (PCA): a Data Reduction Algorithm

*“a powerful algorithm widely used in preparing data for predictive analytics yet it s an algorithm that is
poorly understood by new the generation data scientists” A.Bari*

Principal Component Analysis (PCA)

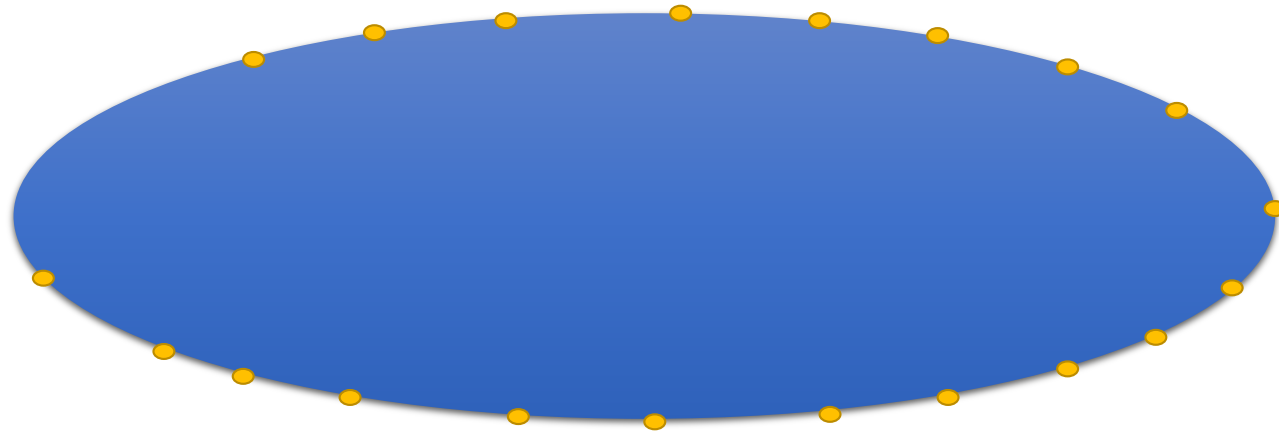
- PCA is a tool to use in Information Theory.
- PCA is a data reduction technique.
- Principal Components represent a new coordinate system for the input data.
- PCA tries to fit the data into a lower dimensional space.
- PCA is considered as both data reduction technique and features construction technique.

Feature: an individual measurable property or characteristic of a phenomenon being observed.

Principal Component Analysis (PCA)

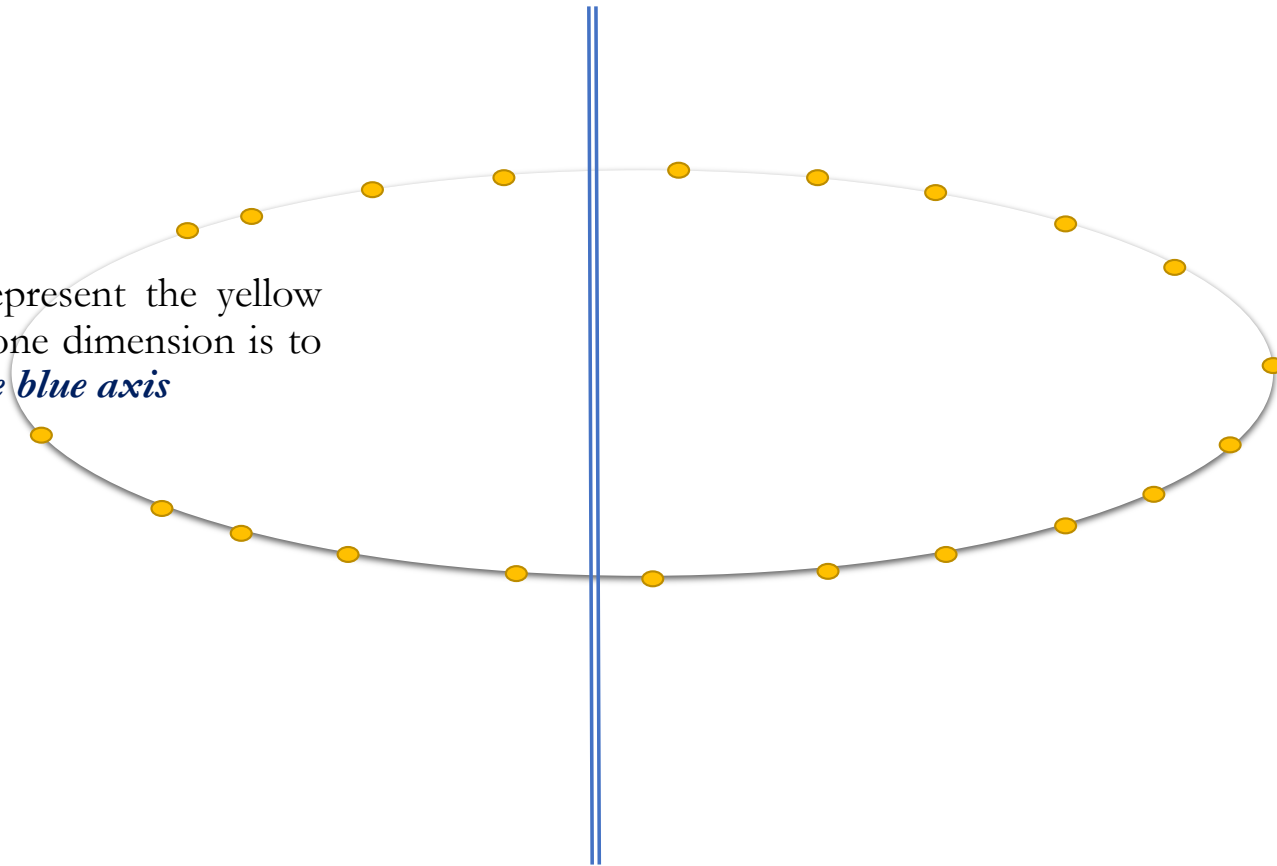
- PCA finds the principal components of data
- PCA allows you to represent your data in terms of its principal components (rather normal representation in x,y,z axis)
- Principal components are the underlying structure of the data.
- Principal components represent the directions (axis) *where there is the most variance* (the axis (dimensions) where the data is MOST spread out) – Consider the example in the next slide.

Principal Component Analysis (PCA)



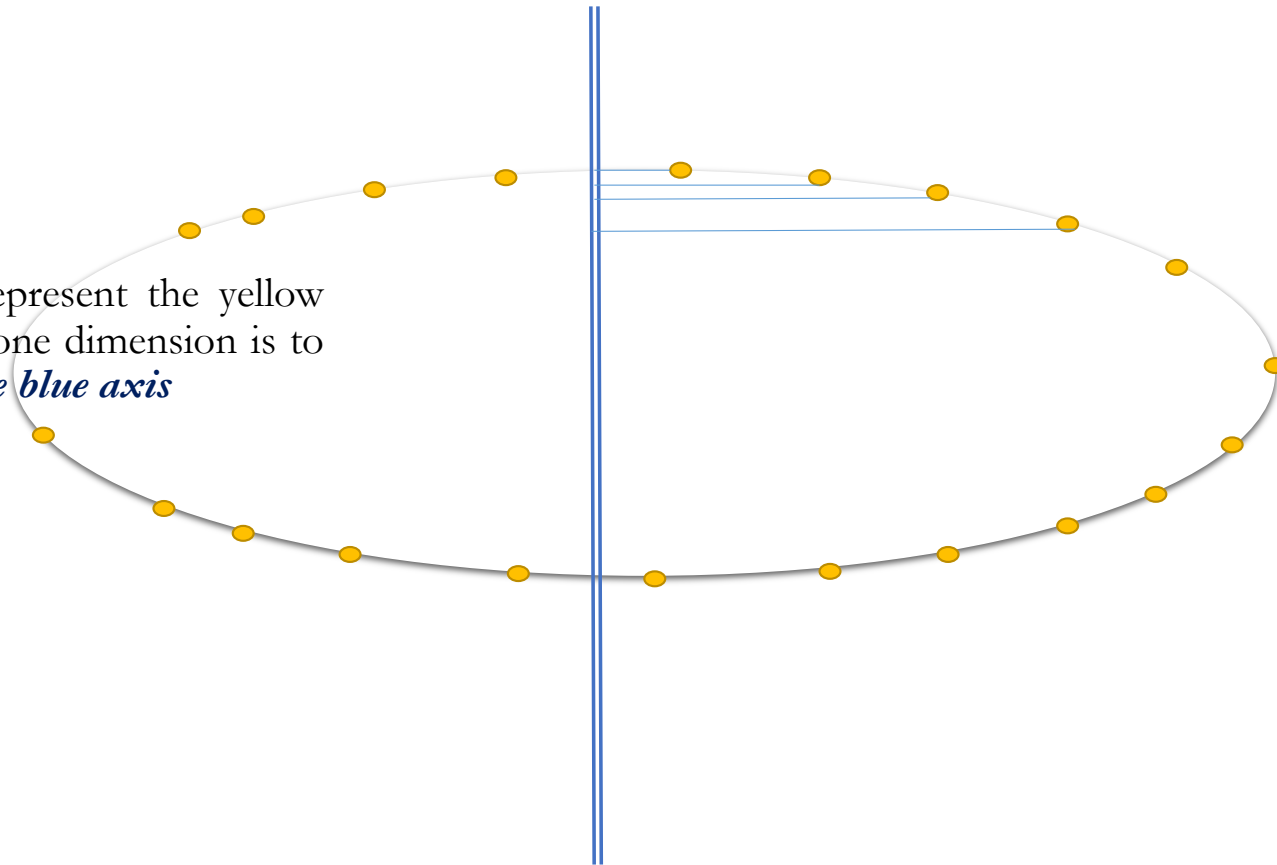
Principal Component Analysis (PCA)

One way to represent the yellow data points in one dimension is to project over *the blue axis*



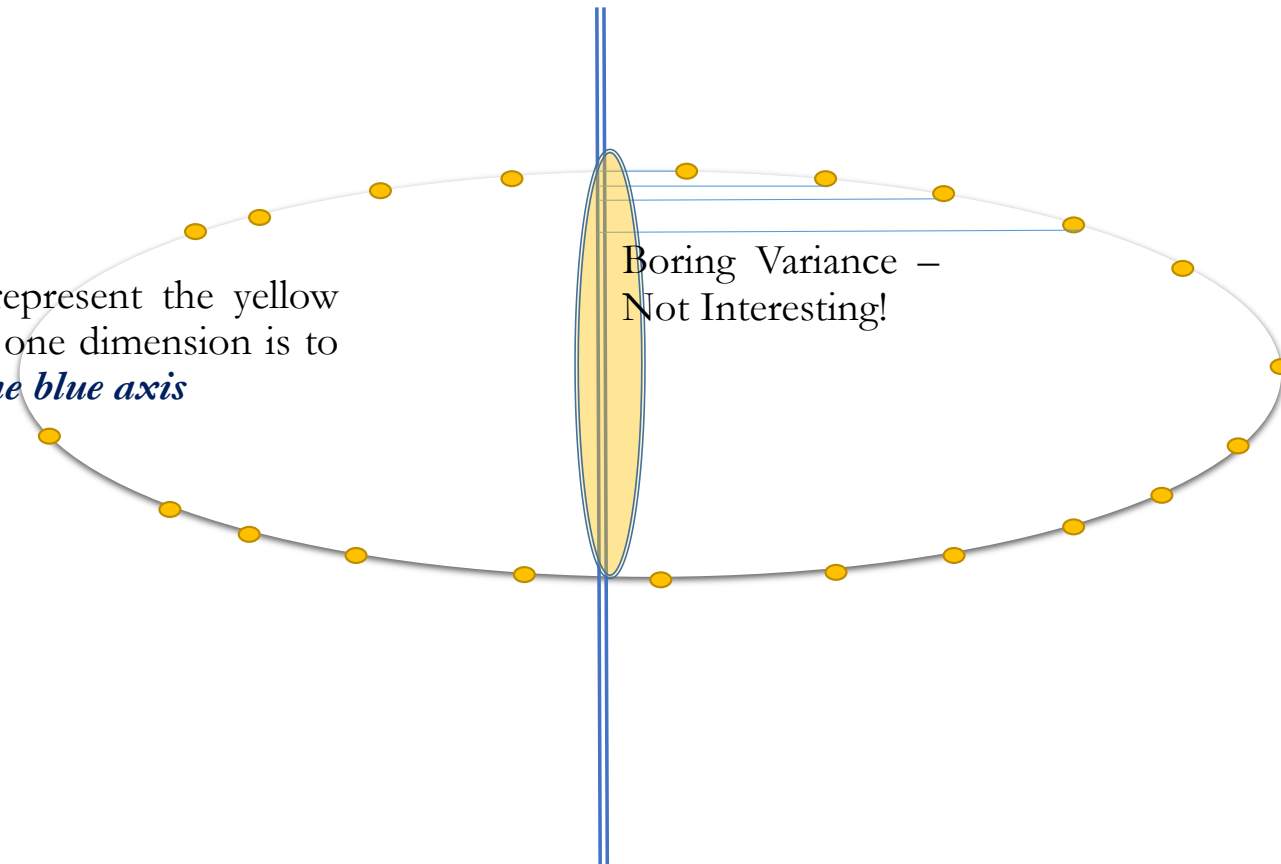
Principal Component Analysis (PCA)

One way to represent the yellow data points in one dimension is to project over *the blue axis*

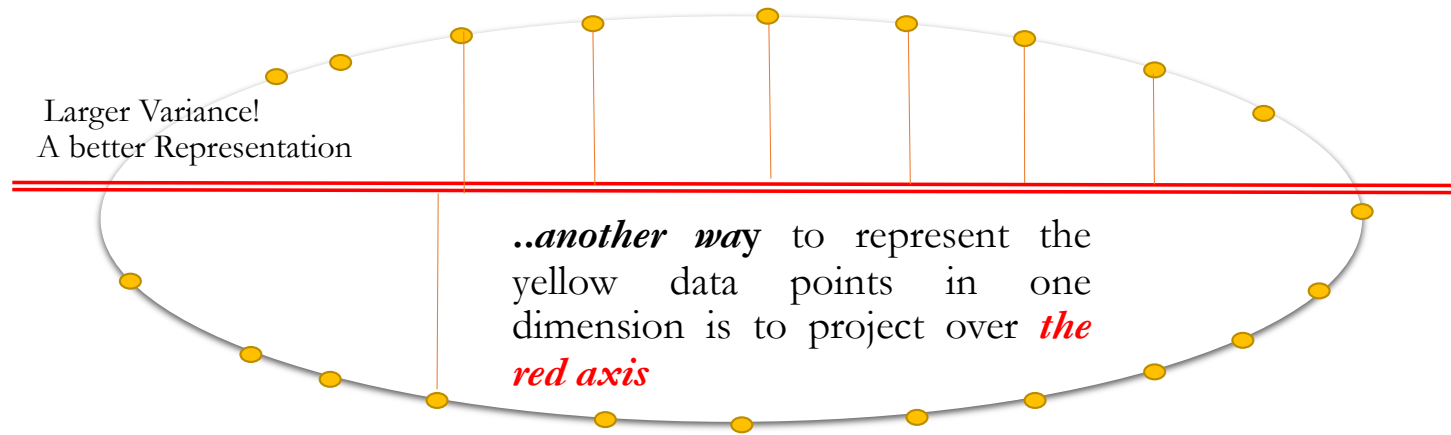


Principal Component Analysis (PCA)

One way to represent the yellow data points in one dimension is to project over *the blue axis*



Principal Component Analysis (PCA)



Principal Component Analysis (PCA)

- PCA finds the principal components of data
- Principal components represent the directions (axis) *where there is the most variance (the axis (dimensions) where the data is MOST spread out)*

Principal Component Analysis (PCA)

- PCA is based on *Eigen Values* and *Eigen Vectors*
- *Each Eigen Value Corresponds to an Eigen Vector*
- **Eigen Values are numbers that represent “how much variance (the strength of variance” is in the data for the corresponding Eigen Vector.**
- An Eigen Value tells you how spread out the data is on the projection of the points on the corresponding Eigen Vector.
- The Eigen Vector with the highest eigenvalue is the first principal component.
- Eigen Vectors transform the data and projects into a **new** dimensions.
- Eigen Vectors give you a useful space of axes to **frame the data in.**
- Eigen Vectors takes you data from one set of axes to another (that is more representative of the data in compact way – from one dimensional space to another)
- PCA gives a better representation on the data and it strips away unnecessary data in your data.

Understanding PCA

- PCA is statistical technique that performs an *orthogonal transformation* of the original n coordinates of the dataset in question into a new set of n coordinates called *principal components*.
- The power of PCA lies into the constraint that every principal component is orthogonal to (*uncorrelated with*) the other components.

From Wikipedia, the free encyclopedia

In linear algebra, an **orthogonal transformation** is a linear transformation $T: V \rightarrow V$ on a real inner product space V , that preserves the inner product. That is, for each pair u, v of elements of V , we have^[1]

$$\langle u, v \rangle = \langle Tu, Tv \rangle.$$

Since the lengths of vectors and the angles between them are defined through the inner product, orthogonal transformations preserve lengths of vectors and angles between them. In particular, orthogonal transformations map **orthonormal bases** to orthonormal bases.

Orthogonal transformations in two- or three-dimensional **Euclidean space** are stiff **rotations**, **reflections**, or combinations of a rotation and a reflection (also known as **improper rotations**). Reflections are transformations that exchange left and right, similar to mirror images. The matrices corresponding to proper rotations (without reflection) have **determinant** $+1$. Transformations with reflection are represented by matrices with determinant -1 . This allows the concept of rotation and reflection to be generalized to higher dimensions.

- The transformation will result on a set of principal components which are vectors that provide a new representation of the data.
- The *first principal component* has the *largest possible variance*, the *second principal component* has the *second largest possible variance*...

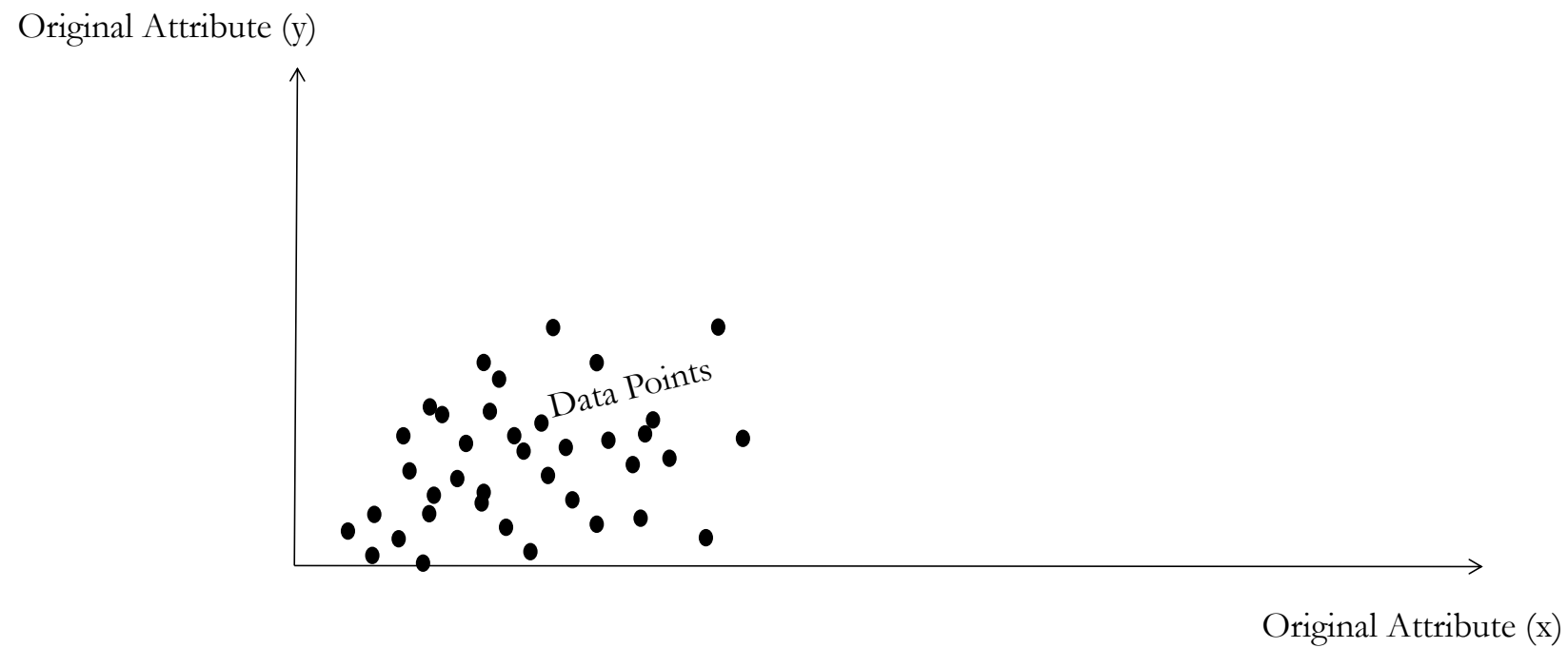
Understanding PCA (continued)

- The power of PCA lies into the constraint that every principal component is orthogonal to (*uncorrelated with*) the other components.
- Reducing an *n-dimensional dataset* to an *m-dimensional dataset* requires keeping only the first $m < n$ components. The components retains most of the data information (the variation) in the data.
- PCA is very sensitive to *the scaling of the attributes' values* in the original dataset.

Understanding PCA (continued)

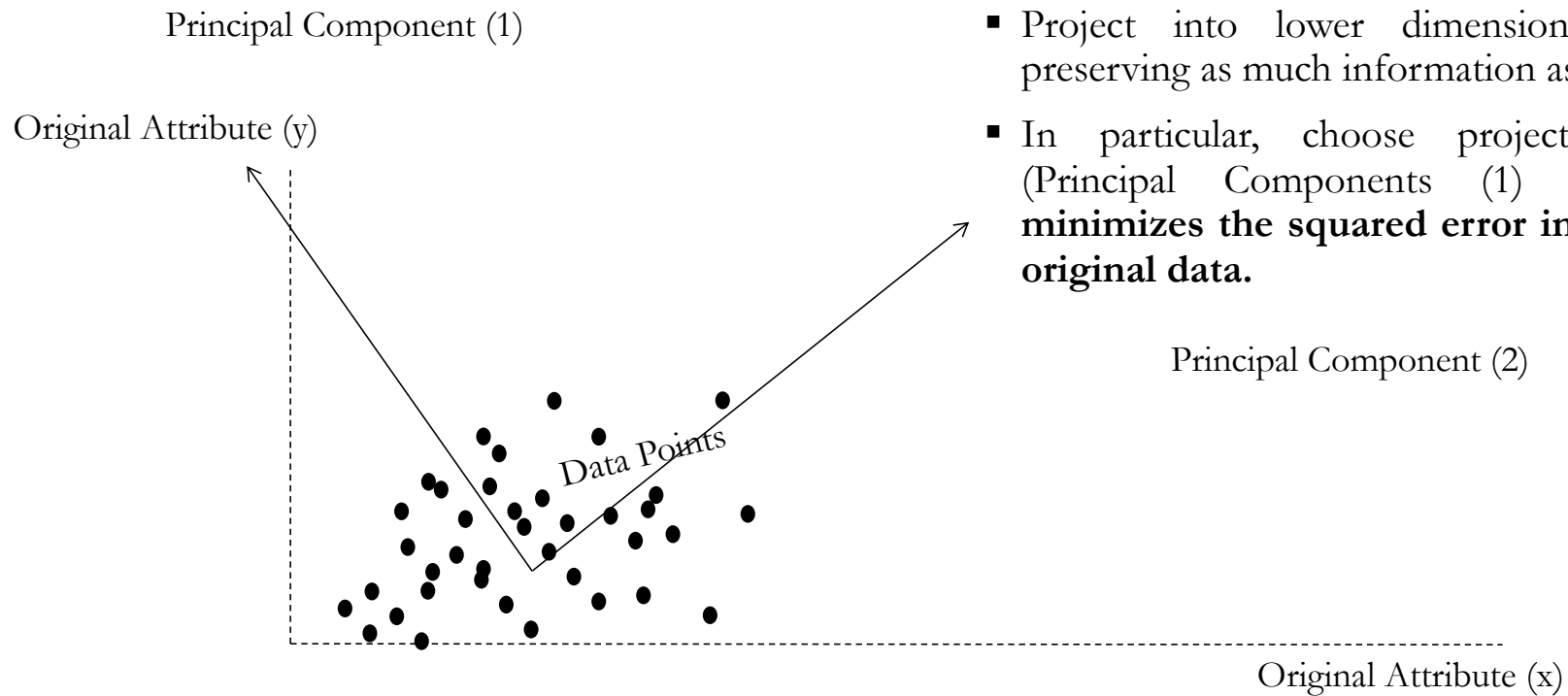
- Variables in the original dataset must be *normalized* before applying PCA.
(Recall normalization from in this chapter, see previous slides)
- Applying PCA to your original dataset results in *a new reduced data representation where you lose **interpretability** of your original data.*
- Meaning that the new coordinates (PCs) are not real variables that have a meaning or measurement unit.
- It is important to note that if *interpretability* in your data analytics project, then PCA may not be the right transformation for your project.

From 2-d space to 1-d space



From 2-d space to 1-d space (continued)

- Given data points in 2-dimensional space,
- Project into lower dimensional space while preserving as much information as possible
- In particular, choose projection over axes (Principal Components (1) and (2)) that **minimizes the squared error in reconstructing original data.**



PCA relies on Eigenvectors and Eigenvalues

- Eigen Values are special set of scalars associated with a linear system (matrix).
- Eigen Vectors are the axis (directions) along which a linear transformation by “stretching/compressing” and flipping Eigen Vectors (Eigen Values are multipliers).

Principal Component Analysis: The Algorithm

- Input: dataset (data vectors) of $m \times n$ -dimensions (numeric data)
- Output: k vectors (principal components) such as $k < n$
- Step#0: **Normalize** the input data (each attribute value should be within the same range) and calculate the covariance matrix
- Step#1: Compute K orthonormal (unit) vectors (principal components)
 - Each input data (vector) is a linear combination of the k principal component vectors
 - The principal components are sorted in order of decreasing “significance” or strength
 - Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Step#3: Generating the reduced dataset $[m \times k\text{-dimensions}]$

Principal Component Analysis: The Algorithm

(continued)

More details:

- Let X be the $n \times d$ data matrix, with one row vector X_n per data point
- Normalize X in X^{norm} *(one way would be subtract the mean of all values from each row vector X_n in X)*
- Calculate the covariance matrix of X denoted as Σ
- Calculate the eigen values then the eigen vectors of Σ
- Output the reduced matrix where the columns are the first M eigen vectors with largest eigen values

Summary on PCA

- PCA is an orthogonal linear transformation
- The input dataset is being transformed to a new coordinate system of lower dimensionality.
- The reduced data is being projected into the coordinate system such that the greatest variance lies on the first coordinate (PCA#1), the second greatest variance lies on the second coordinate and so on.
- The input data need to be normalized before you apply PCA.
- Utilize PCA in your data science project only if **interpretability** is not important in your analysis.

Example: *Numerical example to explain the mathematical mechanics behind Principal Component Analysis (PCA)*

Consider the following Matrix

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 4 & 4 & 1 & 1 \\ 0 & 1 & 4 & 4 \\ 0 & 1 & 2 & 2 \end{bmatrix}$$

The rows represent users and the columns represents their ratings to products. For instance, user 1 rated product 1 with 2 points, product 2 with 3 points, product 3 with 1 point, and product 4 with 1 point.

(continued) Example: *Numerical example to explain the mathematical mechanics behind Principal Component Analysis (PCA)*

Calculate $M^t * M$ (you could also calculate the covariance or the correlation matrix):

$$\begin{bmatrix} 2 & 4 & 0 & 0 \\ 3 & 4 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 4 & 2 \end{bmatrix} \times \begin{bmatrix} 2 & 3 & 1 & 1 \\ 4 & 4 & 1 & 1 \\ 0 & 1 & 4 & 4 \\ 0 & 1 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 20 & 22 & 6 & 6 \\ 22 & 27 & 13 & 13 \\ 6 & 13 & 22 & 22 \\ 6 & 13 & 22 & 22 \end{bmatrix}$$

(continued) Example: *Numerical example to explain the mathematical mechanics behind Principal Component Analysis (PCA)*

Set $\det(M^t M - \lambda I)$ to zero to obtain the eigenvalues.:

$$\begin{vmatrix} 20 - \lambda & 22 & 6 & 6 \\ 22 & 27 - \lambda & 13 & 13 \\ 6 & 13 & 22 - \lambda & 22 \\ 6 & 13 & 22 & 22 - \lambda \end{vmatrix} = 0$$

$$\lambda = (64.63, 26, 0.37, 0)$$

(continued) Example: *Numerical example to explain the mathematical mechanics behind Principal Component Analysis (PCA)*

Use $M^t M * e = \lambda * e$ to obtain the eigenvectors. You might need to normilize them.

$$e_1 = \begin{bmatrix} 0.42 \\ 0.58 \\ 0.49 \\ 0.49 \end{bmatrix}, e_2 = \begin{bmatrix} 0.57 \\ 0.43 \\ -0.5 \\ -0.5 \end{bmatrix}, e_3 = \begin{bmatrix} 0.71 \\ -0.690 \\ 0.11 \\ 0.11 \end{bmatrix}, e_4 = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.71 \\ -0.71 \end{bmatrix}$$

(continued) Example: *Numerical example to explain the mathematical mechanics behind Principal Component Analysis (PCA)*

Construct a matrix of eigenvectors E from the eigenvectors shown in the previous slide, sorted and aligned by the size of the corresponding eigenvalue.

$$E = \begin{bmatrix} 0.42 & 0.57 & 0.71 & 0.00 \\ 0.58 & 0.43 & -0.69 & 0.00 \\ 0.49 & -0.5 & 0.11 & 0.71 \\ 0.49 & -0.5 & 0.11 & -0.71 \end{bmatrix}$$

Construct a matrix of eigenvectors E from the eigenvectors shown in the previous slide, sorted and aligned by the size of the corresponding eigenvalue.

(continued) Example: *Numerical example to explain the mathematical mechanics behind Principal Component Analysis (PCA)*

How to reduce the dimensionality? Multiply the original matrix M with E_k

Where k is the new desired (reduced) number of dimensions.

$$E_k = \begin{bmatrix} 0.42 & 0.57 & 0.71 \\ 0.58 & 0.43 & -0.69 \\ 0.49 & -0.5 & 0.11 \\ 0.49 & -0.5 & 0.11 \end{bmatrix}$$

$$M \times E_k = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 4 & 4 & 1 & 1 \\ 0 & 1 & 4 & 4 \\ 0 & 1 & 2 & 2 \end{bmatrix} \times \begin{bmatrix} 0.42 & 0.57 & 0.71 \\ 0.58 & 0.43 & -0.69 \\ 0.49 & -0.5 & 0.11 \\ 0.49 & -0.5 & 0.11 \end{bmatrix} = \begin{bmatrix} 3.58 & 1.42 & -0.44 \\ 5.00 & 2.98 & 0.29 \\ 4.51 & -3.55 & 0.18 \\ 2.55 & -1.56 & -0.26 \end{bmatrix}$$

PCA Common Questions (1)

How many Ks to retain in PCA?

To be in the next HW..

PCA Common Questions (2)

covariance-based PCA vs correlation-based PCA

Covariance is one possible measure of correlation

To be in the next HW...

Other applications of PCA

Stylometry?

Others...?

Practicing PCA

DataPreparationPractice-3 PCA RapidMiner

DataPreparationPractice- PCA using Knime

DataPreparationPractice-4 PCA R

Variance (basics for PCA)

- Suppose attributes are A_1 and A_2 , and we have n training examples. x 's denote values of A_1 and y 's denote values of A_2 over the training examples.

- Variance of an attribute:

$$\text{var}(A_1) = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n-1)}$$

The average of the **squared** differences from the Mean.

Covariance Matrix (basics for PCA)

- Covariance of two attributes:

$$\text{cov}(A_1, A_2) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

- If covariance is positive, both dimensions increase together. If negative, as one increases, the other decreases. Zero: independent of each other.

Covariance Matrix (PCA Continued)

- Covariance matrix
 - Suppose we have n attributes, A_1, \dots, A_n .
- Covariance matrix:

$$C^{n \times n} = (c_{i,j}), \text{ where } c_{i,j} = \text{cov}(A_i, A_j)$$

Singular Value Decomposition (SVD)

“a powerful algorithm widely used in preparing data for data analytics yet it s an algorithm that is poorly understood by modern data scientists” A.Bari

Review of Data Reduction

Purpose of data reduction beyond storage and processing of the data:

- Discover hidden correlations
- Remove redundant attributes
- Ease the visibility of the data for the purpose of data exploration

Singular Value Decomposition: The Algorithm

- Consider an input Matrix: $A_{[m \times n]}$
- Matrix A can be decomposed into three matrices as follows:

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

r : is the rank of matrix A The rank of a matrix represents the maximum number of linearly independent column vectors in the matrix or the maximum number of linearly independent row vectors in the matrix. Both definitions are equivalent.

- Matrix A 's SVD decomposition is always possible and it is mathematically proven in the literature (U , Σ and V are unique matrices to A)

Singular Value Decomposition: The Algorithm

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

- A: Input data matrix (m x n matrix (e.g., Microarray data: m patient, n gene-expression))
- U named left singular vectors (m x r matrix e.g. m patients, r learned concepts (types))
U is an orthogonal matrix. $U \times U^T = \text{Identity Matrix}$ (same applies to V)
- Σ named singular values (r x r diagonal matrix (strength of each 'learned concept or cluster') Σ is a diagonal matrix where the diagonals are the *singular values* sorted in decreasing order.
- V named right singular vectors (n x r matrix (n genes, r learned concepts - types))

Singular Value Decomposition: The Algorithm

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

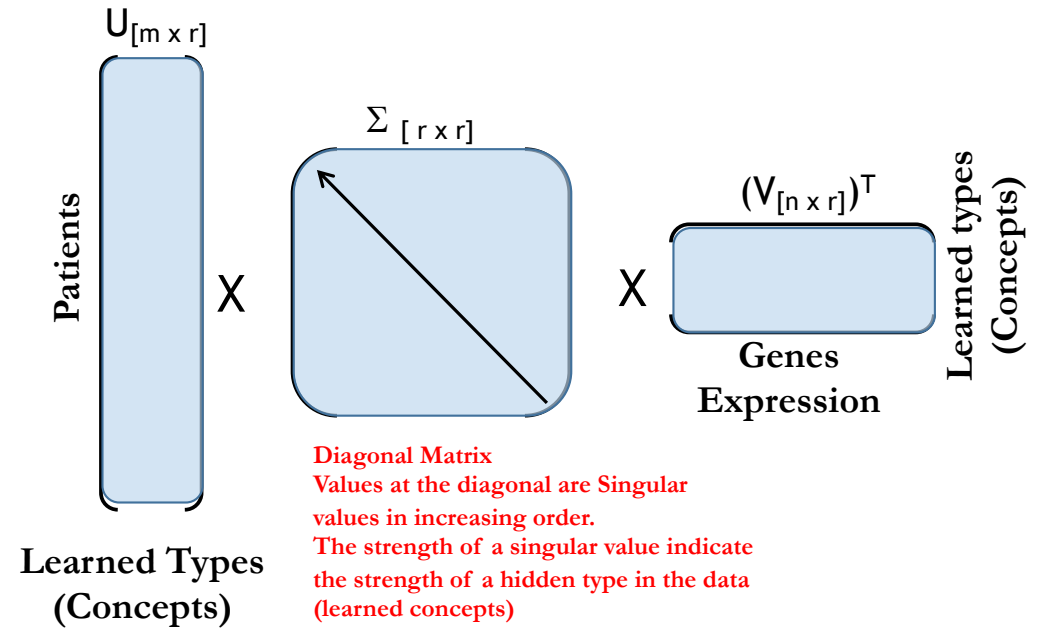
Microarray Sample Dataset

Genes Expression

	G1	G2	G3	G4	G5	G6
P1	2.0	2.0	2.0	1.9	1.9	
P2	2.2	2.2	2.2	1.9	1.0	
P3	2.3	2.3	2.3	1.9	1.9	
P4	2.4	2.4	2.4	1.9	1.0	
P5	1.9	2.1	1.9	2.3	2.3	
P6	1.9	1.9	1.9	2.4	2.4	
P7	1.9	2.0	1.9	2.1	2.1	

Patients'
Samples

=



For more information about mining microarray data please read: A.Bellachia & A.Bari "A Flocking Based Data Mining Algorithm for Detecting Outliers in Cancer Gene Expression Microarray Data", IEEE International Conference on Information Retrieval and Knowledge Management, (IEEE CAMP12), Malaysia, 2012.

Singular Value Decomposition: The Interpretation

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

- A: Input data matrix (m x n matrix (e.g., Microarray data: m patient, n gene-expression))
- U: left singular vectors. It is the patient-to-learned types similarity matrix.
- V: right singular vectors (n x r matrix (n genes, r learned concepts - types)). It is gene to learned types similarity matrix.

Singular Value Decomposition: The Algorithm

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

Patients' Samples

	G1	G2	G3	G4	G5
P1	2	2	2	0.1	0.1
P2	0.5	0.5	0.5	0.1	0.1
P3	1.5	1.5	1.5	0.1	0.1
P4	2.5	2.5	2.5	0.1	0.1
P5	0	0.5	0	1	1
P6	0.1	0.1	0.1	2.5	2.5
P7	0.1	0.5	0.1	0.5	0.5

 \approx

$U_{[m \times r]}$							
-0.5542	0.0730	-0.0336	-0.0424	-0.4140	0.4043	0.5915	
-0.1406	-0.0091	-0.0203	0.4503	0.2821	0.7318	-0.4023	
-0.4163	0.0457	-0.0291	0.0118	0.8167	-0.2225	0.3272	
-0.6920	0.1004	-0.0380	-0.0316	-0.2209	-0.3480	-0.5819	
-0.0740	-0.3613	0.6910	0.5533	-0.1138	-0.2254	0.1286	
-0.0963	-0.9064	-0.3988	-0.0985	-0.0015	0.0112	-0.0167	
-0.0785	-0.1741	0.5997	-0.6917	0.1423	0.2818	-0.1607	

 \times

$\Sigma_{[r \times r]}$				
6.2356				
	3.8429			
		0.4787		
			0.0000	
				0.0000

 \times

$V_{[n \times r]}$				
-0.5694	0.0918	-0.4091	0.7071	0.0000
-0.5804	0.0267	0.8139	-0.0000	0.0000
-0.5694	0.0918	-0.4091	-0.7071	-0.0000
-0.0857	-0.7009	-0.0381	0.0000	-0.7071
-0.0857	-0.7009	-0.0381	0.0000	0.7071

[More examples and Reference to Chapter 11 in Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.](#)

Singular Value Decomposition: The Algorithm

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

Patients' Samples

	G1	G2	G3	G4	G5
P1	2	2	2	0.1	0.1
P2	0.5	0.5	0.5	0.1	0.1
P3	1.5	1.5	1.5	0.1	0.1
P4	2.5	2.5	2.5	0.1	0.1
P5	0	0.5	0	1	1
P6	0.1	0.1	0.1	2.5	2.5
P7	0.1	0.5	0.1	0.5	0.5

 \approx

$U_{[m \times r]}$		
-0.5542	0.0730	-0.0336
-0.1406	-0.0091	-0.0203
-0.4163	0.0457	-0.0291
-0.6920	0.1004	-0.0380
-0.0740	-0.3613	0.6910
-0.0963	-0.9064	-0.3988
-0.0785	-0.1741	0.5997

 \times

$\Sigma_{[r \times r]}$		
6.2356		
	3.8429	
		0.4787

 \times

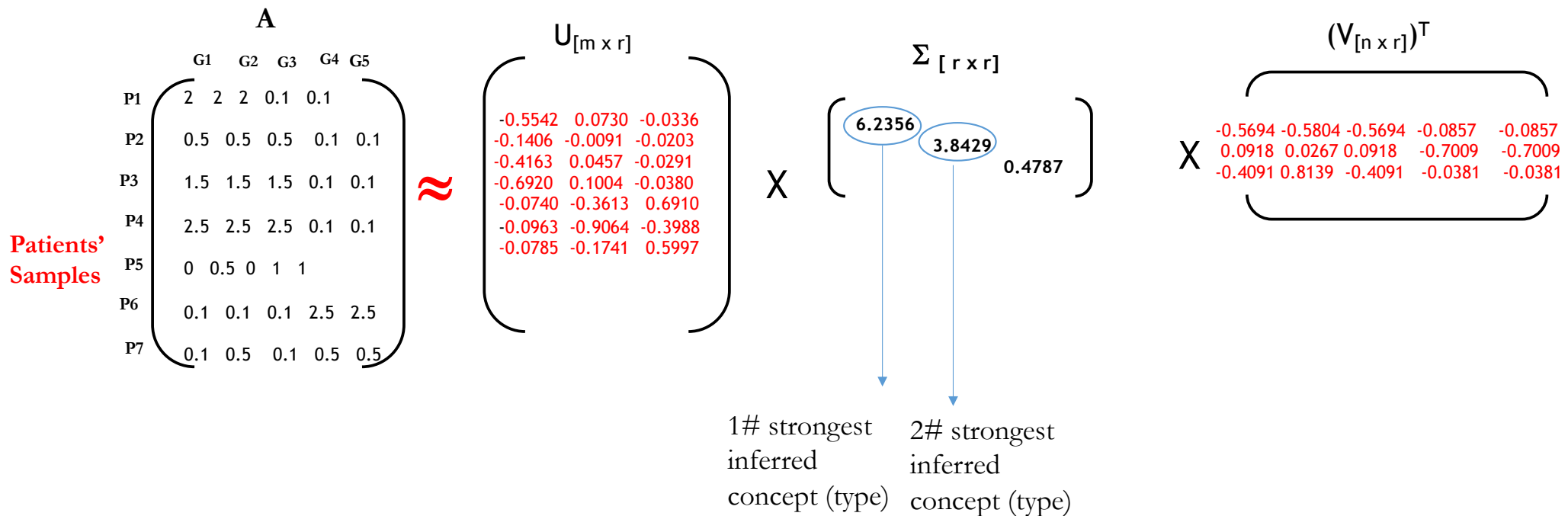
$(V_{[n \times r]})^T$				
-0.5694	-0.5804	-0.5694	-0.0857	-0.0857
0.0918	0.0267	0.0918	-0.7009	-0.7009
-0.4091	0.8139	-0.4091	-0.0381	-0.0381

[More examples and Reference to Chapter 11 in Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.](#)

Singular Value Decomposition: The Interpretation

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

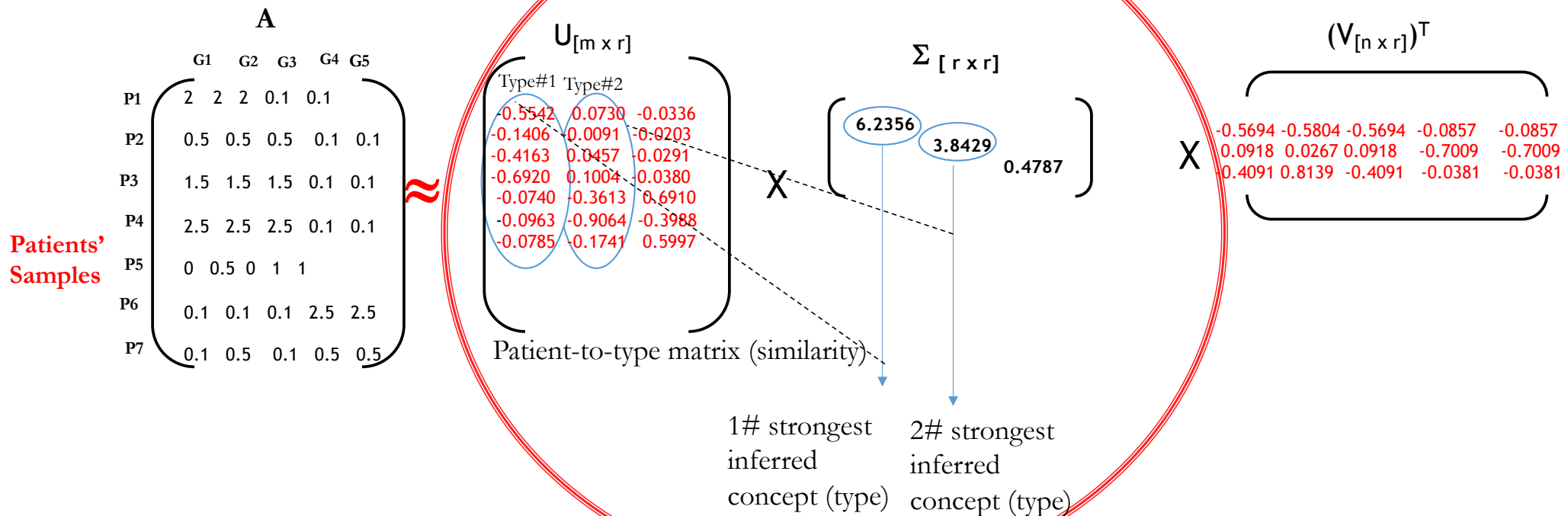


[More examples and Reference to Chapter 11 in Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.](#)

Singular Value Decomposition: The Interpretation

(continued)

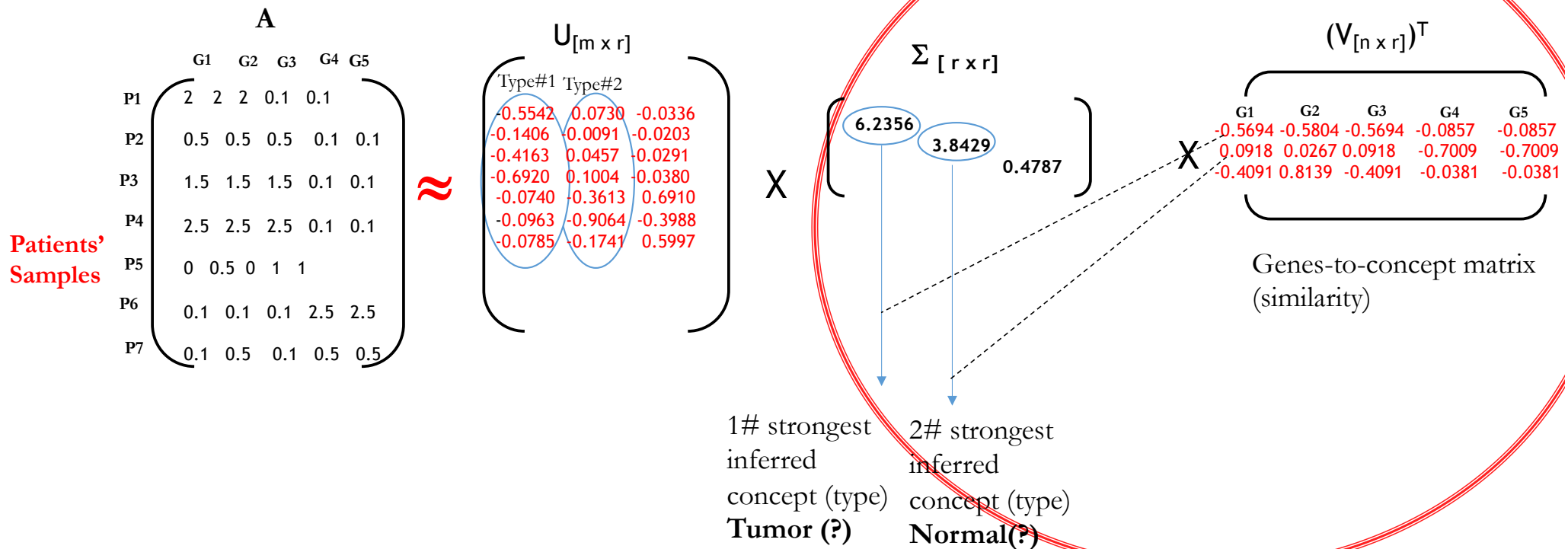
$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$



Singular Value Decomposition: The Interpretation

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$



[More examples and Reference to Chapter 11 in Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.](#)

Singular Value Decomposition: The Interpretation

(continued)

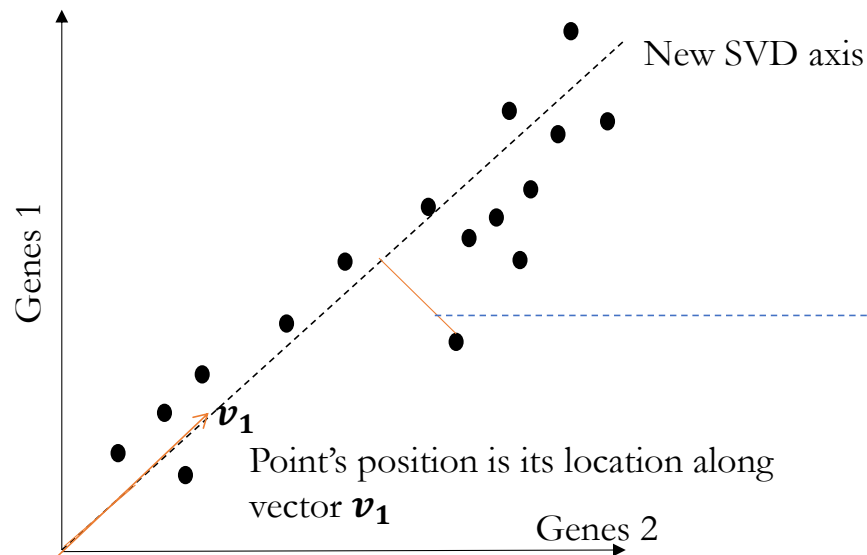
$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

- A: Input data matrix (m x n matrix (e.g., Microarray data: m patient, n gene-expression))
- U: left singular vectors. It is the patient-to-learned types similarity matrix.
- V: right singular vectors (n x r matrix (n genes, r learned concepts - types)). It is gene to learned types similarity matrix.

Singular Value Decomposition: Choice of Axis

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$



(Analogous to PCA)

**Minimizing the sum
of reconstruction errors:**

$$\sum_{i=1}^N \sum_{j=1}^D \|x_{ij} - z_{ij}\|^2$$

where \mathbf{x}_{ij} are the “old” and \mathbf{z}_{ij} are the
“new” coordinates

SVD gives ‘best’ axis to project on:

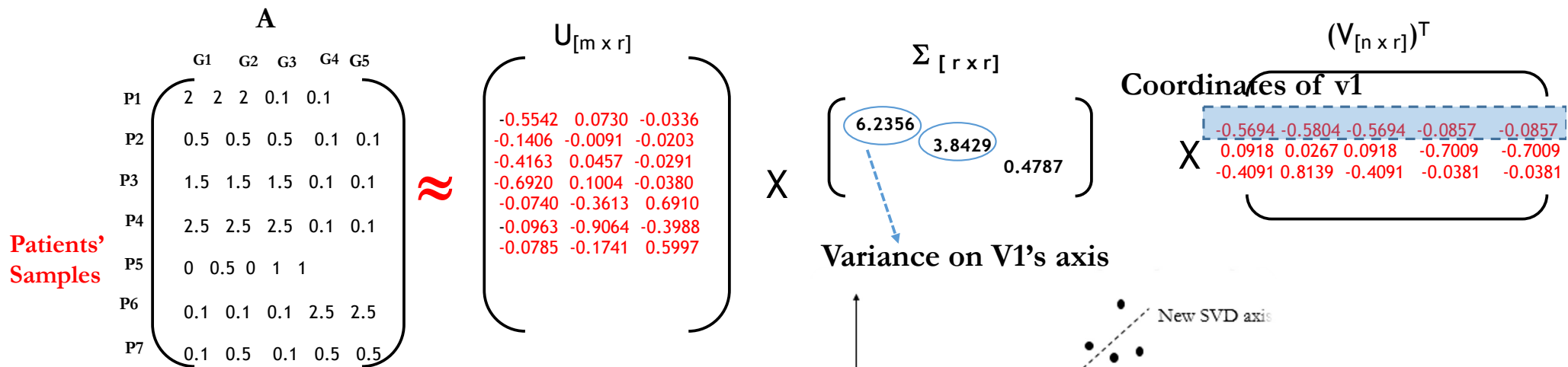
‘best’ = minimizing the reconstruction errors

In other words, minimum reconstruction error

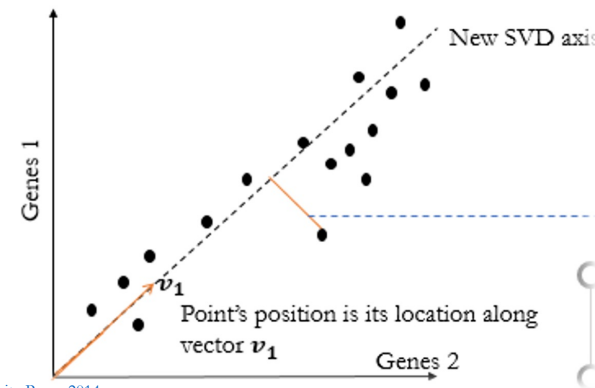
Singular Value Decomposition: **Another** Interpretation

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$



Variance on V_1 's axis



- U: left singular vectors. It is the patient-to-learned types similarity matrix.
- V: right singular vectors ($n \times r$ matrix (n genes, r learned concepts - types)). It is gene to learned types similarity matrix.

Singular Value Decomposition: **Another** Interpretation

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

Patients' Samples

	G1	G2	G3	G4	G5
P1	2	2	2	0.1	0.1
P2	0.5	0.5	0.5	0.1	0.1
P3	1.5	1.5	1.5	0.1	0.1
P4	2.5	2.5	2.5	0.1	0.1
P5	0	0.5	0	1	1
P6	0.1	0.1	0.1	2.5	2.5
P7	0.1	0.5	0.1	0.5	0.5

 \approx

$U_{[m \times r]}$			$\Sigma_{[r \times r]}$			$(V_{[n \times r]})^T$				
-0.5542	0.0730	-0.0336	6.2356			-0.5694	-0.5804	-0.5694	-0.0857	-0.0857
-0.1406	-0.0091	-0.0203		3.8429		0.0918	0.0267	0.0918	-0.7009	-0.7009
-0.4163	0.0457	-0.0291			0.4787	-0.4091	0.8139	-0.4091	-0.0381	-0.0381
-0.6920	0.1004	-0.0380								
-0.0740	-0.3613	0.6910								
-0.0963	-0.9064	-0.3988								
-0.0785	-0.1741	0.5997								

 \times

Multiplying matrices U and Σ and then apply transpose to the resulted matrix will result in projecting patients on the new SVD's axis.

[More examples and Reference to Chapter 11 in Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.](#)

Singular Value Decomposition: **Another** Interpretation

(continued)

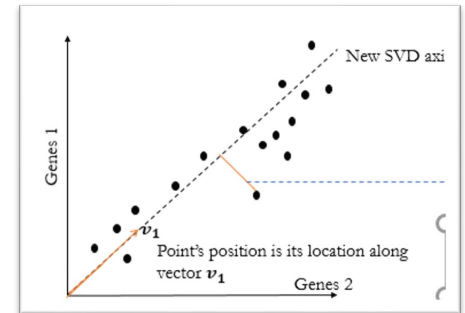
$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

$$\begin{matrix}
 U_{[m \times r]} \\
 \begin{pmatrix}
 -0.5542 & 0.0730 & -0.0336 \\
 -0.1406 & -0.0091 & -0.0203 \\
 -0.4163 & 0.0457 & -0.0291 \\
 -0.6920 & 0.1004 & -0.0380 \\
 -0.0740 & -0.3613 & 0.6910 \\
 -0.0963 & -0.9064 & -0.3988 \\
 -0.0785 & -0.1741 & 0.5997
 \end{pmatrix}
 \end{matrix}
 \times
 \begin{matrix}
 \Sigma_{[r \times r]} \\
 \begin{pmatrix}
 6.2356 & & \\
 & 3.8429 & \\
 & & 0.4787
 \end{pmatrix}
 \end{matrix}$$

Multiplying matrices U and Σ will result in projecting patients on the new SVD's axis.

$$(U \Sigma)
 \begin{pmatrix}
 -3.455 & 0.281 & -0.016 \\
 -0.879 & -0.035 & -0.010 \\
 -2.594 & 0.177 & -0.014 \\
 -4.315 & 0.384 & -0.018 \\
 -0.461 & -1.387 & 0.331 \\
 -0.599 & -3.482 & -0.191 \\
 -0.493 & -0.669 & 0.287
 \end{pmatrix}$$

The first column represents the projection of patients on the first learned concept (Tumor (?))



Singular Value Decomposition: Querying Data

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

- Finding similar patients according to the discovered concept space (type tumor vs. cancer).
- Model the patient in question as query vector (e.g. $Q [3.5 \ 2.3 \ 0 \ 0 \ 0..]$)
- Mapping the Query to the concept space by multiplying Q with V .
 V : Genes-to-concept matrix (similarity)
- You can have multiple Queries to see if the Queries (patients) will be fall into the tumor or the normal concepts.

Singular Value Decomposition: Querying Data

(continued)

$$A_{[m \times n]} = U_{[m \times r]} \times \Sigma_{[r \times r]} \times (V_{[n \times r]})^T$$

- Example:
- Q1 [0 3.5 2.3 0 0]
- Mapping the Query into the concept space:

vector Q1 \times matrix V = [0 3.5 2.3 0 0] \times

-0.5694	0.0918	-0.4091
-0.5804	0.0267	0.8139
-0.5694	0.0918	-0.4091
-0.0857	-0.7009	-0.0381
-0.0857	-0.7009	-0.0381

- Assume another patient of: Q2 [6,0,0,0,0]
- Mapping the Query to the concept space by multiplying Q with V.
V: Genes-to-concept matrix (similarity)
- Q1 and Q2 projected on the concept space would give you a clear similarity in the concept space (even though they do not share same measures in the genes expression they might end up on the same concept – tumor vs normal)

[More examples and Reference to Chapter 11 in Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.](#)

Practical Applications of SVD

Required Reading:

Chapter 11 from Mining Massive Datasets, www.mmds.org

[Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge University Press, 2014.](#)

Additional Readings on SVD' applications:

<http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf>

<http://alias-i.com/lingpipe/demos/tutorial/sgd/read-me.html>

Additional Readings Data Reduction:

http://www.iasri.res.in/ebook/EB_SMAR/e-book_pdf%20files/Manual%20II/9-data_reduction.pdf

Resources

Data Compression' Lecture notes : <https://www.seas.gwu.edu/~ayoussef/cs6351/>

Data Reduction: http://www.iasri.res.in/ebook/EB_SMAR/e-book_pdf%20files/Manual%20II/9-data_reduction.pdf

Knime and Data Reduction Methods: <http://www.kdnuggets.com/2015/05/7-methods-data-dimensionality-reduction.html>

More on Data Reduction: http://web.stanford.edu/~thairu/07_184.Guest.1sts.pdf

Predictive Analytics

End of Chapter Three

Data Preparation Algorithms

“Before anything else, preparation is the key to success.” Alexander Graham Bell
“In all cases, you end up spending more than 80% of your data analytics lifecycle’ time
preparing your data” A.Bari

For *SVD* and *Data Reduction* sections, the assigned reading is **Chapter Eleven** from the Book (Ullman’s Book – www.mmids.org)

Anasse Bari, Ph.D.

Copyrights @ Anasse Bari