Deep Learning-I

by

Dr. Monika Goyal

Assistant Professor, Department of AI and ML

Dayananda Sagar University

Bengaluru

MODULE-2

Mathematical background for Deep learning:

- Data Manipulation
- Data Preprocessing
- Linear Algebra
- Calculus
- Probability

Data Manipulation

- In order to get anything done, we need some way to store and manipulate data.

There are two important things we need to do with data:

(i)Acquire them

(ii)Process them once they are inside the computer.

Python for Data manipulation

- Let us consider synthetic data,we introduce the n-dimensional array, which is also called the *tensor*.
- We import the np (numpy)npx (numpy_extension) modules from MXNet(MXNet is an open-source deep learning software framework, used to train, and deploy deep neural networks.)
- Here,the np module includes functions supported by NumPy, while the **npx module contains a set of extensions developed to empower deep learning within a NumPy-like environment.**
- When using tensors,we almost always invoke the set_np function: this is for compatibility of tensor processing by other components of MXNet.

Python for Data manipulation...

```
# Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
x=[1,2,3,4,5,6,7,8]
X
o/p[1, 2, 3, 4, 5, 6, 7, 8]
```

Python for Data manipulation..

```
x = np.arange(14)
```

X

o/p

```
array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13])
```

```
x.shape
```

```
(14)
```

```
y=x.size
```

```
y
```

14

Python for Data manipulation..

Change the shape of a tensor without altering either the number of elements or their values

```
a =x.reshape(3,4) #Arrange the array of size 12 into 3 rows and 4 columns

a

o/p

array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Python for Data manipulation..

- Create a set of two arrays with Zeros

```
np.zeros((2, 3, 4))
#print two arrays with 3rows and four columns with values zero.
O/P
array([[[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]],

       [[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]]])
```

Python for Data manipulation..

- np.ones((2, 3, 4))
- `#print two arrays with 3rows and four columns with values ones.`

o/p

```
array([[[1., 1., 1., 1.],
 [1., 1., 1., 1.],
 [1., 1., 1., 1.]],
[[1., 1., 1., 1.],
 [1., 1., 1., 1.],
 [1., 1., 1., 1.]]])
```

Python for Data manipulation..

- Construct an array-Random values
- For example,when we construct arrays to serve as parameters in a neural network.
- We willt ypically initialize their values randomly. The following snippet creates a tensor with shape (3,4).
- Each of its elements is randomly sampled from a standard Gaussian(normal)distribution with a mean of 0 and a standard deviation of 1.

Python for Data manipulation..

np.random.normal(0, 1, size=(3, 4))

o/p

```
array([[ 0.30643967, 1.3812342 , -0.57935724, -
0.65810456], [-1.47102494, -0.62923998, -
1.5654041 , 1.47659725], [ 0.45809148, -
0.4619724 , -0.2459706 , 0.81116768]])
```

Python for Data manipulation..

- We can also specify the exact values for each element in the desired tensor by supplying a Python list(or list of lists)containing the numerical values.
- np.array([[2, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2, 1]])

o/p

```
array([[2., 1., 4., 3.],
 [1., 2., 3., 4.],
[4., 3., 2., 1.]])
```

Arithmetic Operations

```
x = np.array([1, 2, 4, 8])
y = np.array([2, 2, 2, 2])
x + y, x - y, x * y, x / y, x ** y
# The ** operator is exponentiation
o/p
(array([ 3, 4, 6, 10]),
array([-1, 0, 2, 6]),
array([ 2, 4, 8, 16]),
array([0.5, 1. , 2. , 4. ]),
array([ 1, 4, 16, 64]))
```

Arithmetic Operations ...

- `np.exp(x)`
- `o/p`
- `array([2.71828183e+00, 7.38905610e+00, 5.45981500e+01, 2.98095799e+03])`

Arithmetic Operations ...

```
X = np.arange(12).reshape(3, 4)
Y = np.array([[2, 1, 4, 3], [1, 2, 3, 4], [4, 3, 2
, 1]])
np.concatenate([X, Y], axis=0),
np.concatenate([X, Y], axis=1)
o/p
array([[ 0, 1, 2, 3, 2, 1, 4, 3], [ 4, 5, 6, 7, 1,
2, 3, 4], [ 8, 9, 10, 11, 4, 3, 2, 1]])
```

Arithmetic Operations ...

- `X == Y`

```
array([[False,  True, False,  True],
       [False, False, False, False],
       [False, False, False, False]])
```

- `X.sum()`
- 66

Arithmetic Operations ...

- We saw how to perform element wise operations on two tensors of the same shape.
- when shapes differ,we can still perform element wise operations by invoking the broadcasting mechanism.
- First, expand one or both arrays by copying elements appropriately so that after this transformation , the two tensors have the same shape.

Arithmetic Operations ...

```
# Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

- a = np.arange(3).reshape(3, 1)
- b = np.arange(2).reshape(1, 2)
- a, b
- o/p
- (array([[0.],
- [1.],
- [2.]]),
- array([[0., 1.]]))

Arithmetic Operations ...

- a and b are 3×1 and 1×2 matrices respectively, their shapes do not match up if we want to add them.
- We broadcast the entries of both matrices into a larger 3×2 matrix as follows
- for matrix a it replicates the columns and for matrix b it replicates the rows before adding up both element wise.
- a + b
- o/p
- array([[0., 1.],
- [1., 2.],
- [2., 3.]])

Indexing and Slicing

- X=[1,2,3,4,5,6]
- X[-1], X[1:3]

#[-1] selects the Last element and[1:3]selects the second and the third elements as follows:

- o/p

6

3,4

- X[1, 2] = 3

Saving Memory

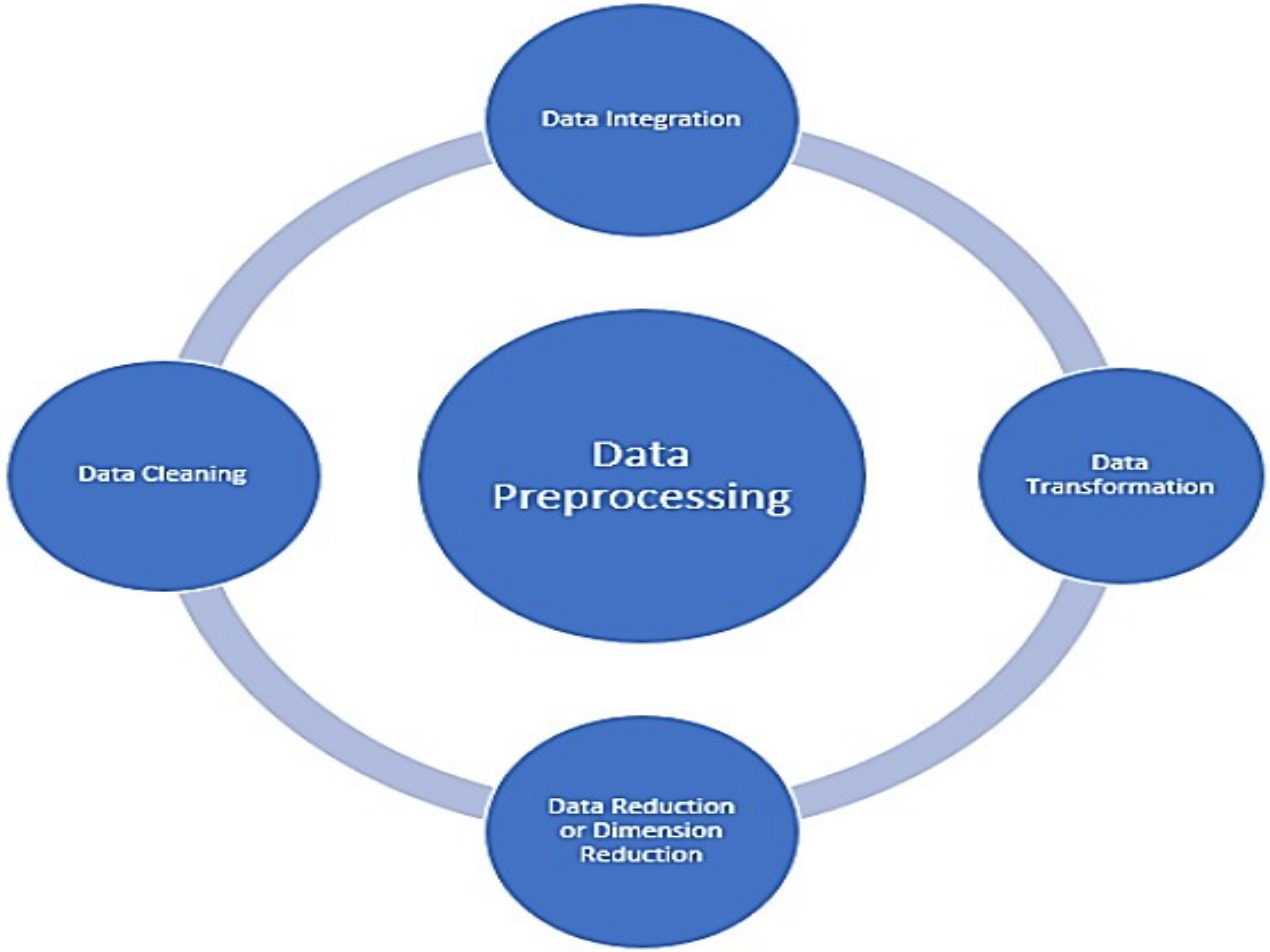
Y = id(X)

- Y
- O/P
- 140064807278144 #Memory Location of X

Conversion to Other Python Objects

- a = np.array([3.5]) a,
- a.item(), float(a), int(a)
- (array([3.5]), 3.5, 3.5, 3)

Data Preprocessing



Data analysis pipeline

- Mining is not the only step in the analysis process

- Preprocessing: real data is noisy, incomplete and inconsistent.
  Data cleaning is required to make sense of the data
  - Techniques: Sampling, Dimensionality Reduction, Feature Selection.
- Post-Processing: Make the data actionable and useful to the user
  Statistical analysis of importance & Visualization.

Why Preprocess the Data

Measures for Data Quality: A Multidimensional View

Accuracy: Correct or Wrong, Accurate or Not
Completeness: Not recorded, unavailable,…
Consistency: Come modified but some not,…
Timeliness: Timely update?
Believability: how trustable the data are correct?
Interpretability: how easily the data can be understood?

Why Data Preprocessing?

- Data in the real world is noisy
  - incomplete: lacking *attribute values*, lacking certain *attributes of interest*, or containing only aggregate data
  - noisy: containing errors or outliers
  - inconsistent: containing discrepancies in codes or names
- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
  - Data warehouse needs consistent integration of quality data
  - Required for both OLAP and Data Mining!

Major Tasks in Data Preprocessing

outliers=exceptions!

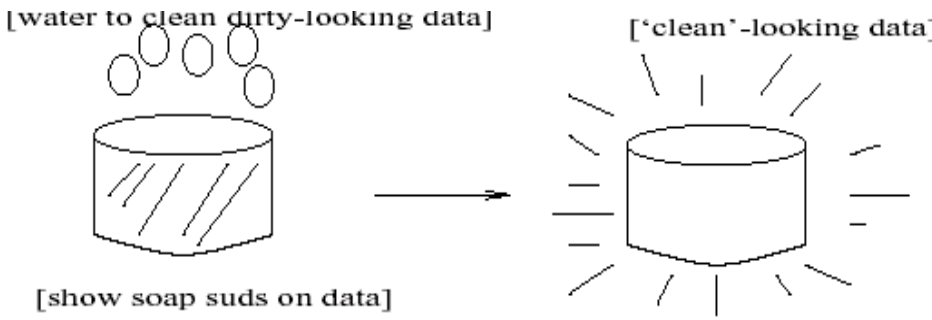
- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data transformation
  - Normalization and aggregation
- Data integration
  - Integration of multiple databases, data cubes, or files
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results

Major Tasks in Data Preprocessing

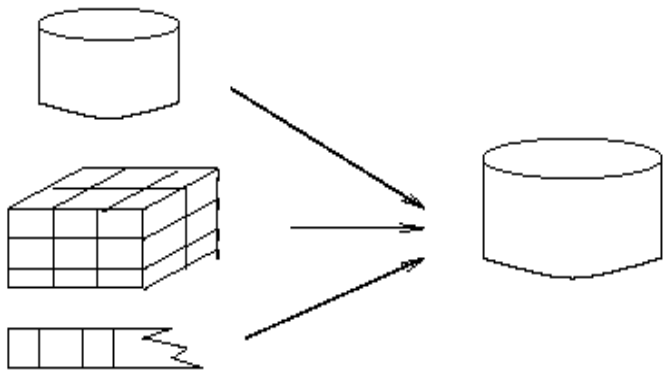
- Data discretization
  - Data discretization refers to a method of converting a huge number of data values into smaller ones so that the evaluation and management of data become easy.
  - In other words, data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss.

Forms of data preprocessing

Data Cleaning



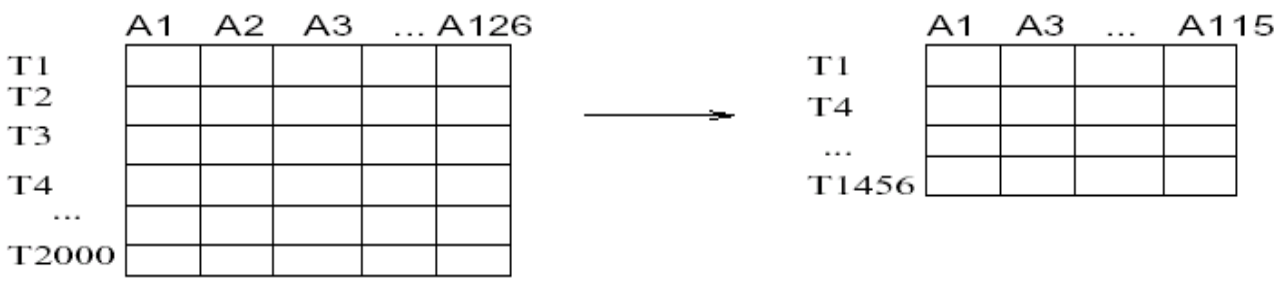
Data Integration



Data Transformation

-2, 32, 100, 59, 48 → -0.02, 0.32, 1.00, 0.59, 0.48

Data Reduction



Data Cleaning

Data in the Real World Is Dirty: Lots of potentially incorrect data, E g., Instrument faulty, human or computer error, Transmission error

- Data cleaning tasks
  - Fill in missing values
  - Identify outliers and smooth out noisy data
  - Correct inconsistent data

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (assuming the tasks in classification)—not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value manually: tedious + infeasible?
- Use a global constant to fill in the missing value: e.g., “unknown”, a new class?!
- Use the attribute mean to fill in the missing value
- Use the attribute mean for all samples belonging to the same class to fill in the missing value: smarter
- Use the most probable value to fill in the missing value: inference-based such as Bayesian formula or decision tree

How to Handle Missing Data?

Age	Income	Religion	Gender
23	24,200	Muslim	M
39	?	Christian	F
45	45,390	?	F

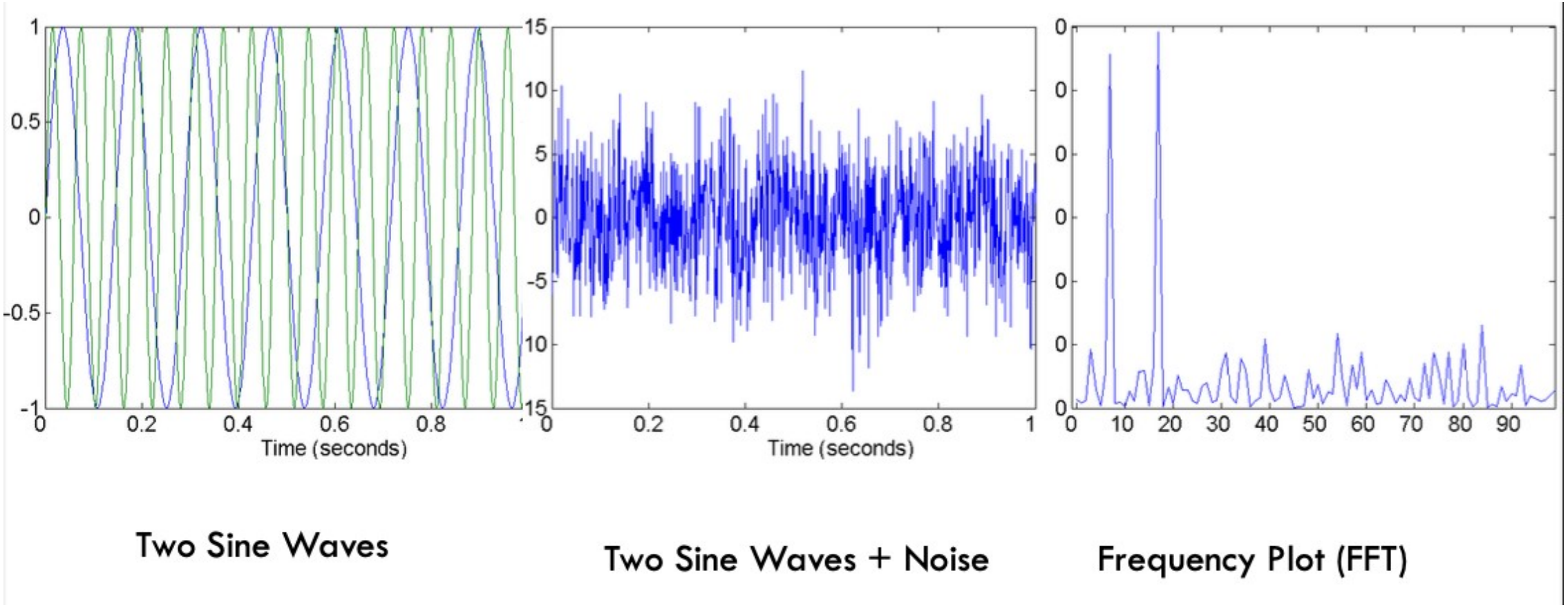
Fill missing values using aggregate functions (e.g., average) or probabilistic estimates on global value distribution

E.g., put the average income here, or put the most probable income based on the fact that the person is 39 years old

E.g., put the most frequent religion here

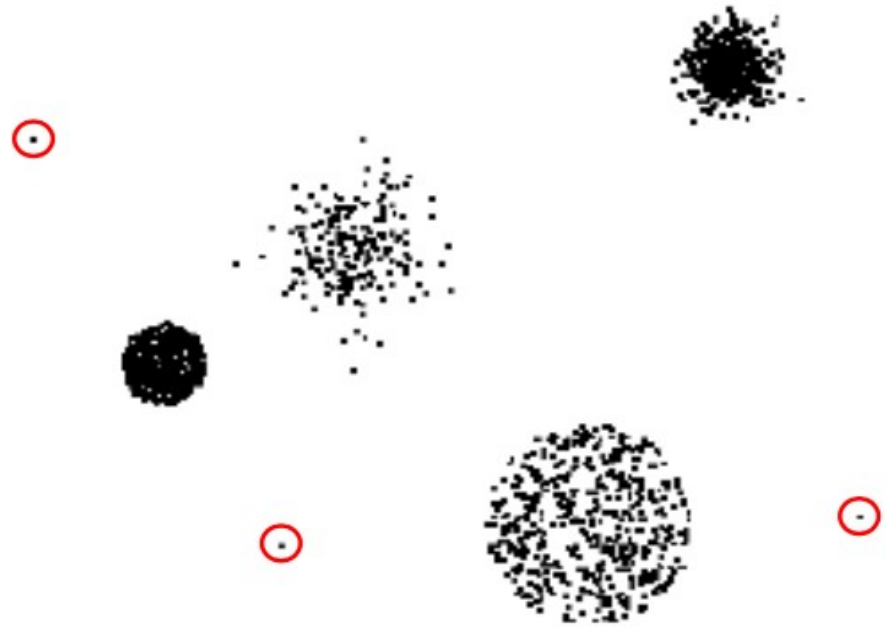
Data Quality: Noise

- Noise refers to modification of original values
- Examples: distortion of a person's voice when talking on



Data Quality: Outliers

- Outliers are data objects with characteristics that are considerably different than most of the other data objects in the data set



Data Quality: Missing Values

- Reasons for missing values
  - Information is not collected
  - (e.g., people decline to give their age and weight)
  - Attributes may not be applicable to all cases (e.g., annual income is not applicable to children)
- Handling missing values
  - Eliminate Data Objects
  - Estimate Missing Values
  - Ignore the Missing Value During Analysis
  - Replace with all possible values (weighted by their probabilities)

Data Quality: Duplicate Data

Data set may include data objects that are duplicates, or almost duplicates of one another

- Major issue when merging data from heterogeous sources

  Examples:

  Same person with multiple email addresses

  - Data cleaning
  - Process of dealing with duplicate data issues

Data Quality: Handle Noise

- Binning
  - sort data and partition into (equi-depth) bins
  - smooth by bin means, bin median, bin boundaries, etc.
- Regression
  - smooth by fitting a regression function
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values automatically and check by human

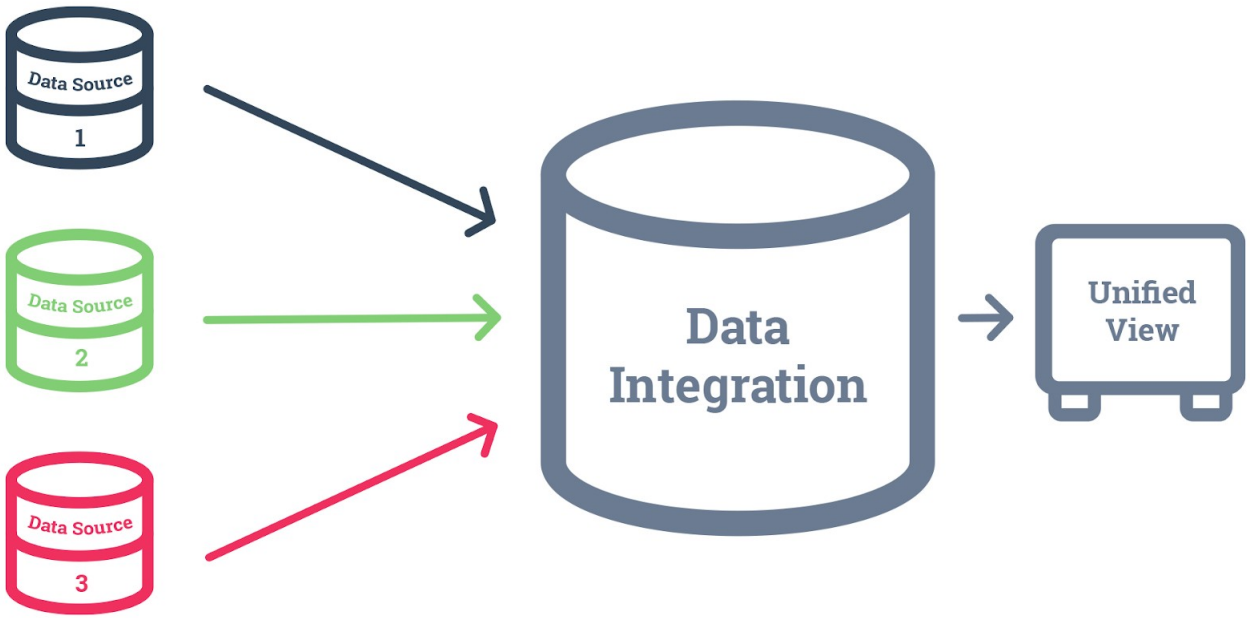
Data Quality: Handle Noise(Binning)

- Equal-width binning
  - Divides the range into N intervals of equal size Width of intervals:
  - Simple
  - Outliers may dominate result

- Equal-depth binning
  - Divides the range into N intervals, each containing approximately same number of records
  - Skewed data is also handled well

Data Integration

The process of combining multiple sources into a single dataset. The Data integration process is one of the main components in data management.



Data integration:

Combines data from multiple sources into a coherent store

Schema integration integrate metadata from different sources

metadata: data about the data (i.e., data descriptors)

Entity identification problem: identify real world entities from multiple data sources, e.g., A.cust-id ≡ B.cust-#

Detecting and resolving data value conflicts for the same real world entity, attribute values from different sources are different (e.g., J.D.Smith and Jonh Smith may refer to the same person)

possible reasons: different representations, different scales, e.g., metric vs. British units (inches vs. cm)

Handling Redundant Data in Data Integration

- 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 correlation analysis
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Data Transformation

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones

Normalization: Why normalization?

- Speeds-up some learning techniques (ex. neural networks)
- Helps prevent attributes with large ranges outweigh ones with small ranges
  - Example:
    - income has range 3000-200000
    - age has range 10-80
    - gender has domain M/F

Data Transformation

Data has an attribute values

Then,

Can we compare these attribute values?

For Example: Compare following two records

(1) (5.9 ft, 50 Kg)

(2) (4.6 ft, 55 Kg) Vs.

(3) (5.9 ft, 50 Kg)

(4) (5.6 ft, 56 Kg)

We need Data Transformation to makes different dimension(attribute) records comparable ...

Data Transformation Techniques

- Normalization: scaled to fall within a small, specified range.
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Centralization:
  - Based on fitting a distribution to the data
  - Distance function between distributions
    - KL Distance
- Mean Centering

Data Transformation: Normalization

- min-max normalization

$$v' = \frac{v - min}{max - min}(new\_max - new\_min) + new\_min$$

- z-score normalization

$$v' = \frac{v - mean}{stand\_dev}$$

- normalization by decimal scaling

$$v' = \frac{v}{10^j}$$ Where j is the smallest integer such that $\text{Max}(|v'|) < 1$

Example: Data Transformation

- Assume, min and max value for height and weight.
- Now, apply Min-Max normalization to both attributes as given follow

(1) (5.9 ft, 50 Kg)
(2) (4.6 ft, 55 Kg)

Vs.

(3) (5.9 ft, 50 Kg)
(4) (5.6 ft, 56 Kg)

- Compare your results...

Data Transformation: Aggregation

- Combining two or more attributes (or objects) into a single attribute (or object)
- Purpose
  - Data reduction
    - Reduce the number of attributes or objects
  - Change of scale
    - Cities aggregated into regions, states, countries, etc
  - More “stable” data
    - Aggregated data tends to have less variability

Data Transformation: Discretization

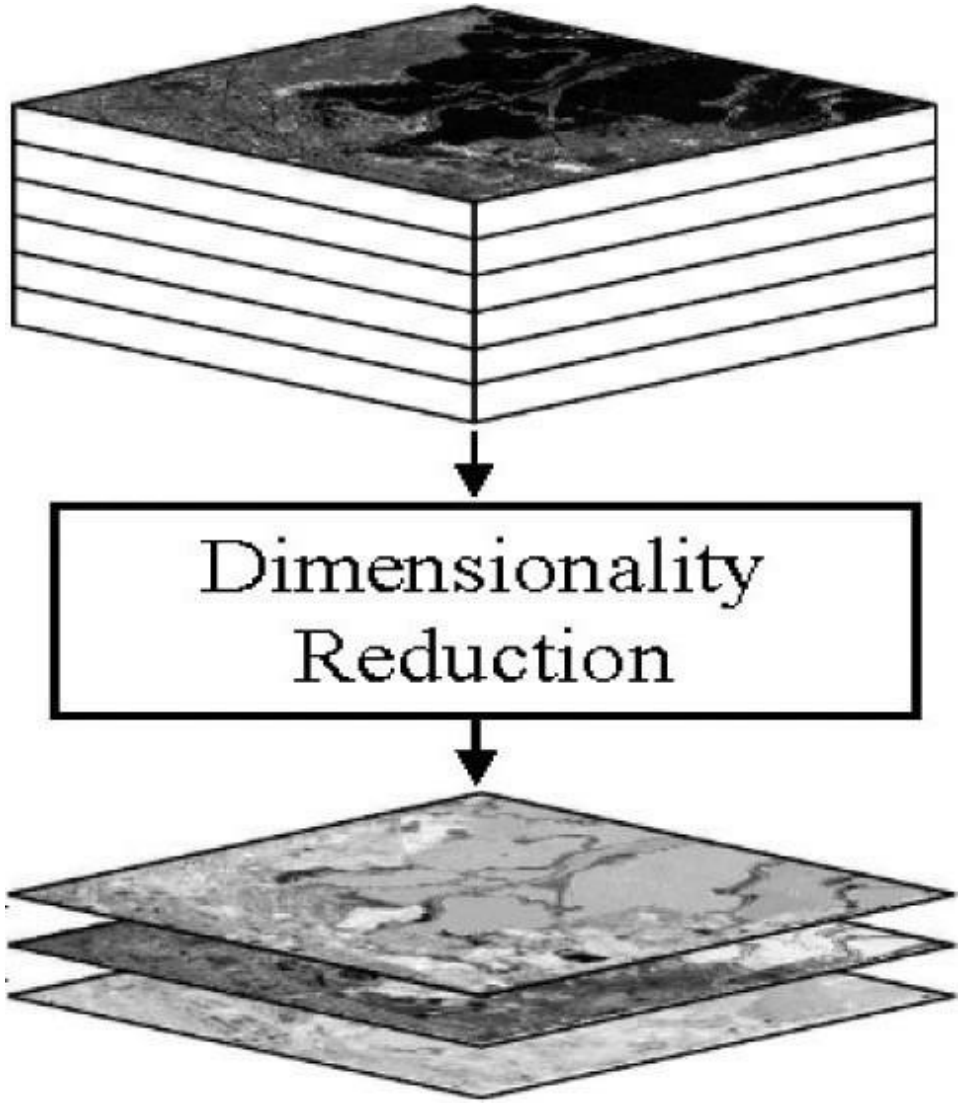
- Motivation for Discretization
- Some data mining algorithms only accept categorical attributes
- May improve understandability of patterns

Data Transformation: Discretization

- Task
- Reduce the number of values for a given continuous attribute by partitioning the range of the attribute into intervals
- Interval labels replace actual attribute values
- Methods
- Binning (as explained earlier)
- Cluster analysis (will be discussed later)
- Entropy-based Discretization (Supervised)

Dimensionality Reduction

- The reduction of random variables or attributes is done so that the dimensionality of the data set can be reduced.
- Combining and merging the attributes of the data without losing its original characteristics. This also helps in the reduction of storage space and computation time is reduced.



Numerosity Reduction:

Reduce the **volume** of data

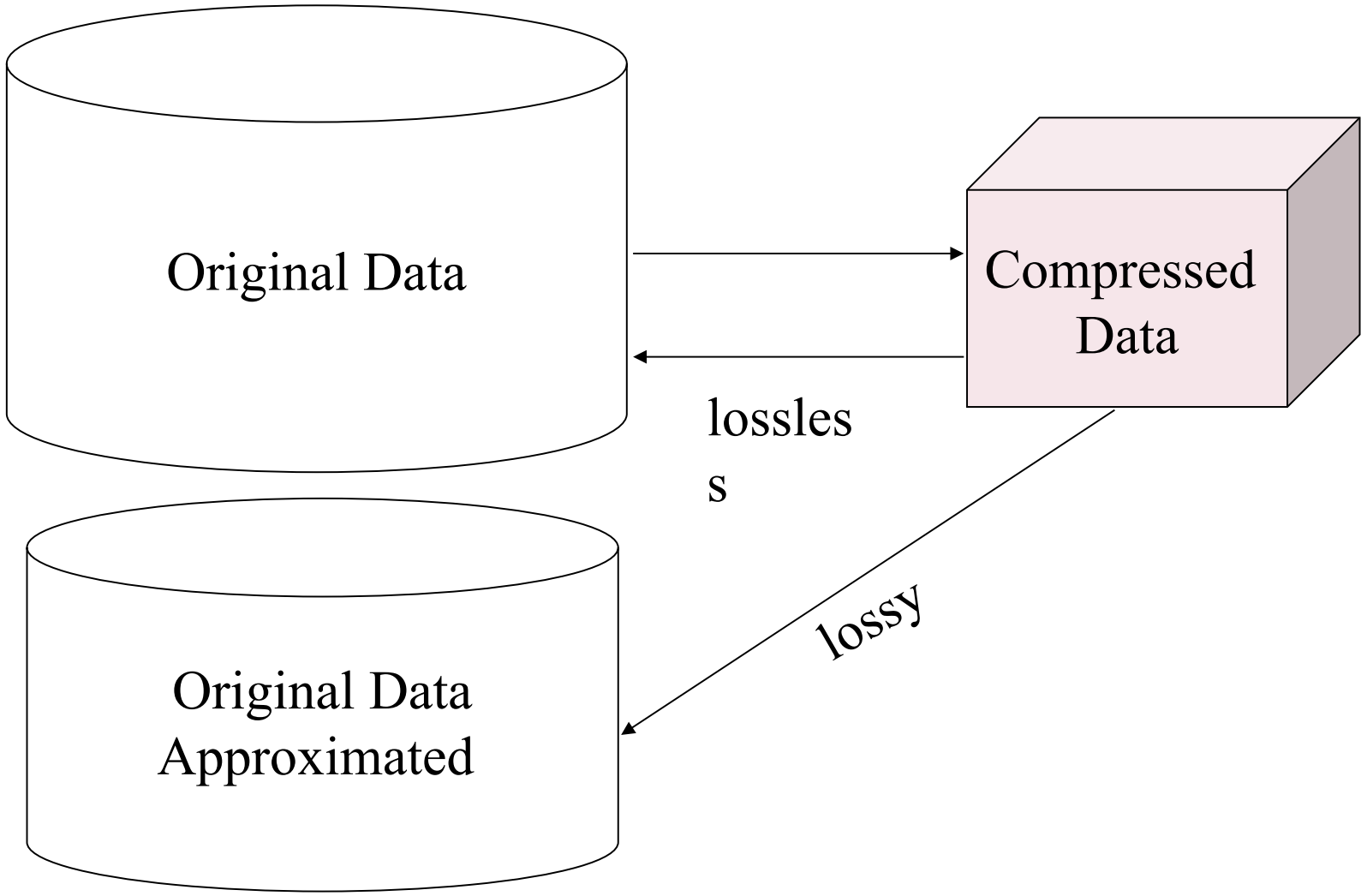
The representation of the data is made smaller by reducing the volume. There will not be any loss of data in this reduction.

- Parametric methods
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Log-linear models: obtain value at a point in m-D space as the product on appropriate marginal subspaces
- Non-parametric methods
  - Do not assume models
  - Major families: histograms, clustering, sampling

Data Cube Aggregation

- The lowest level of a data cube
  - the aggregated data for an individual entity of interest
  - e.g., a customer in a phone calling data warehouse.
- Multiple levels of aggregation in data cubes
  - Further reduce the size of data to deal with
- Reference appropriate levels
  - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

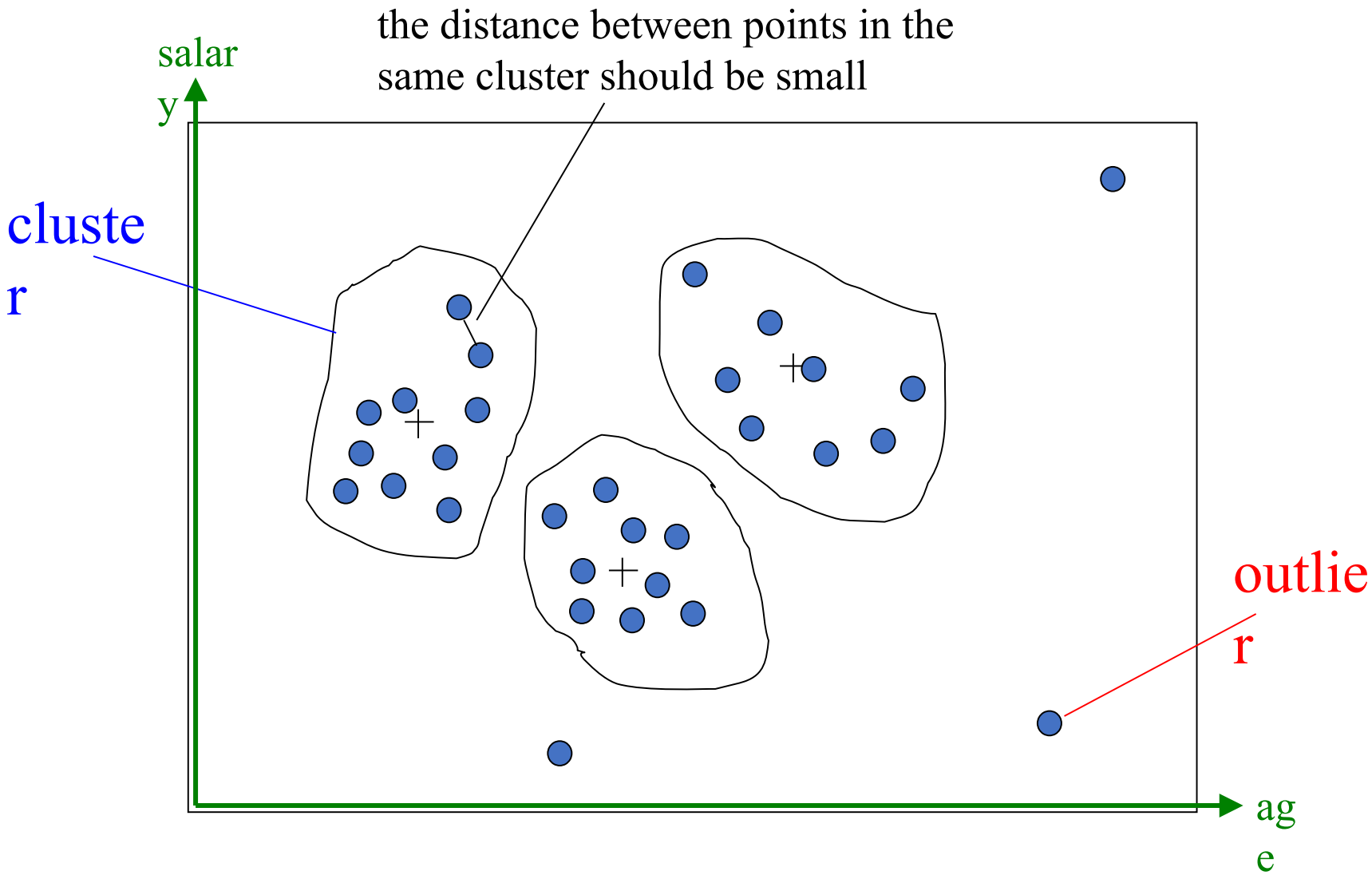
Data Compression



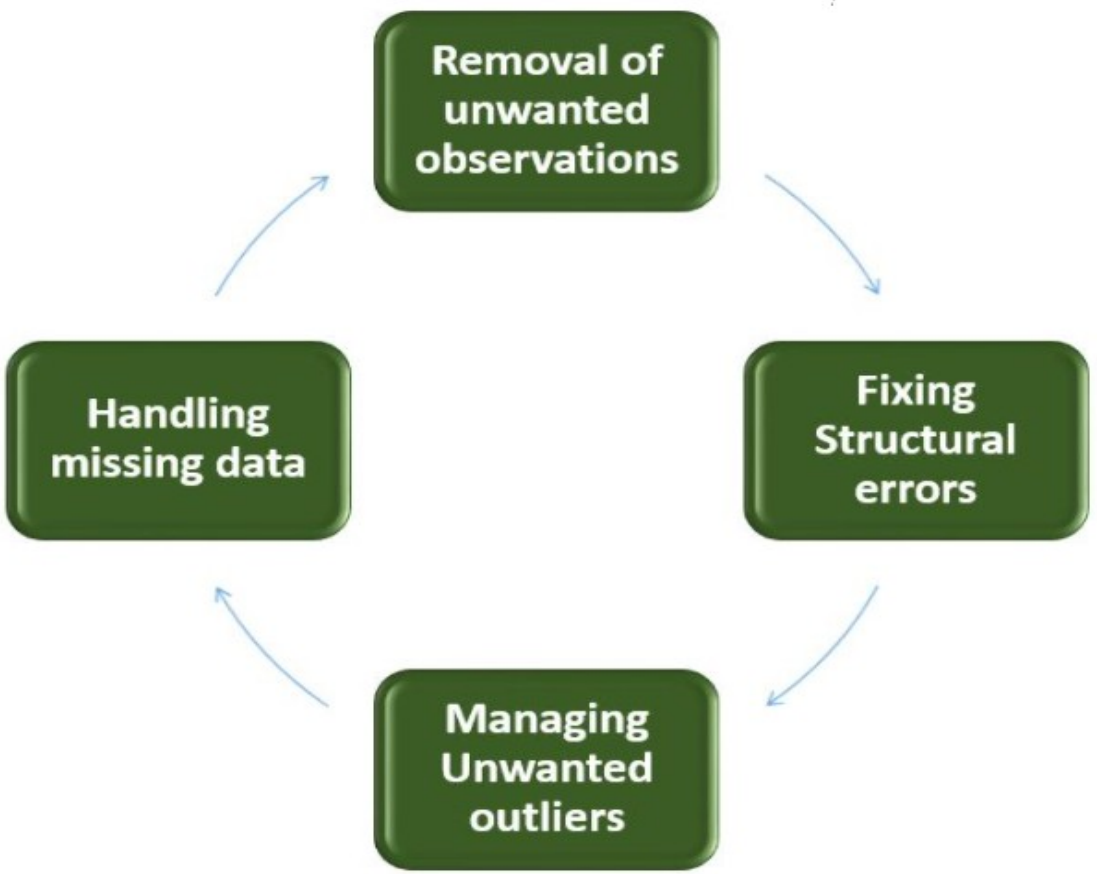
Clustering

- Partitions data set into clusters, and models it by one representative from each cluster
- Can be very effective if data is clustered but not if data is “smeared”
- There are many choices of clustering definitions and clustering algorithms

Cluster Analysis



Data Cleaning



Example -Data cleaning

- We can perform a Data cleaning and choose to delete such data from our table.
- The impossible data will affect the calculation or data manipulation process .

		Sex	Pregnant
Adult	1	Male	No
	2	Female	Yes
	3	**Male**	**Yes**
	4	Female	No
	5	**Male**	**Yes**

		Sex	Pregnant
Adult	1	Male	No
	2	Female	Yes
	4	Female	No

Missing Data

2006	20	1	24	1280
2006	21	1	1	1197
2006	21	1	2	Missing data
2006	21	1	3	1121
2006	21	1	4	1115
2006	21	1	5	1147
2006	21	1	6	1231
2006	21	1	7	1346
2006	21	1	8	Missing Data
2006	21	1	9	1603
2006	21	1	10	1606
2006	21	1	11	1585
2006	21	1	12	1545

Interpolation/Optimization

- The popular INEDI (Improved New Edge Directional Interpolation) method is used .
- The edge directed interpolation algorithm estimates the local **covariance coefficients** from low-resolution images and then these are used to adapt the interpolation at a higher resolution based on geometric duality between LR covariance and HR covariance.

Data Integration

- **Data integration** involves combining data residing in different sources .

Customer data integration

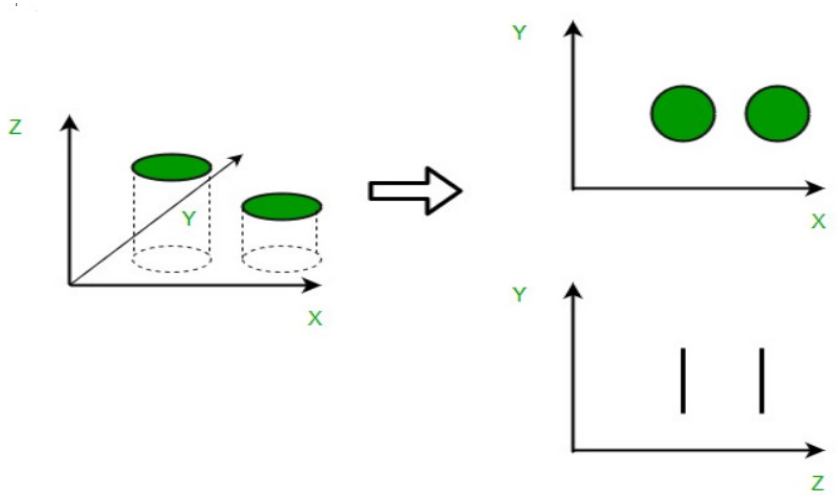
Connect data from distributed databases and systems to boost customer relationship management (CRM) and deliver what customers want or need.

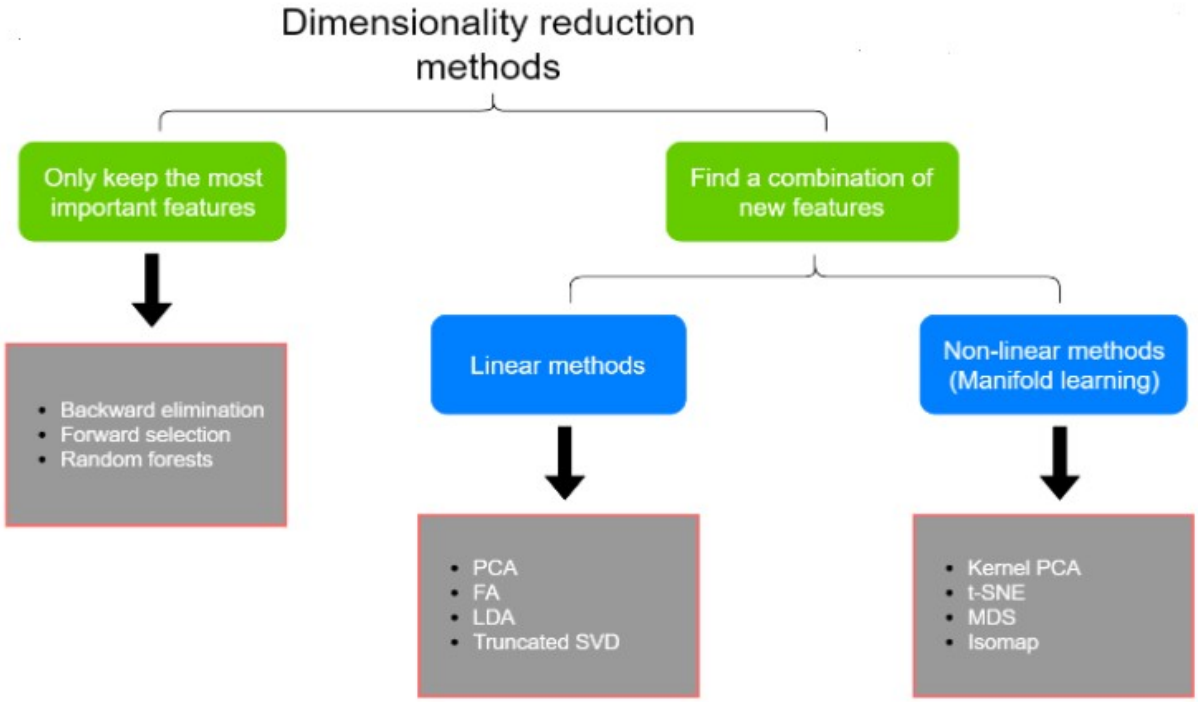
Healthcare data integration

Combine clinical, genomic, radiology and image data for rapid insights and make it available for patient treatment, cohort treatment and population health analytics.

Data transformation and **Dimension Reduction**

- **Data transformation** is the process of converting data from one format or structure into another format or structure.(Finding Max and Min,Rounding…)
- **Dimension Reduction:**





Data Preprocessing

Reading the Dataset

```
# Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Import the dataset
 dataset = pd.read_csv('/content/Salary_Data.csv’)
 X = dataset.iloc[:, :-1].values
 y = dataset.iloc[:, 1].values
```

Handling Missing Data

- To handle missing data,typical methods include imputation(Replace with one and deletion(ignores missing values)
- By integer-location based indexing(iloc),we split data into inputs and outputs, where the former takes the first two columns while the latter only keeps the last column.

Data Preprocessing….

- For numerical values in inputs that are missing,we replace the“None”entries with the mean value of the same column.
- inputs, outputs = data.iloc[:, 0:2], data.iloc[:, 2]
- inputs = inputs.fillna(inputs.mean())
- print(inputs)
- A=[1 2 3]
- [4 none 6]
- *Please refer text book-1 page 51 Example*

Linear Algebra
Scalars

- x = np.array(3.0)
- = np.array(2.0)
- x + y, x * y, x / y, x ** y
- o/p
- (array(5.), array(6.), array(1.5), array(9.))

Linear Algebra …
Matrices

- We have already discussed about length dimensionality and shape.
- A = np.arange(20).
- reshape(5, 4)
- A
- o/p

```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.],
       [12., 13., 14., 15.],
       [16., 17., 18., 19.]])
```

Linear Algebra ...
Matrices...Transpose of a Matrix

- A.T

```
array([[ 0., 4., 8., 12., 16.],
 [ 1., 5., 9., 13., 17.],
 [ 2., 6., 10., 14., 18.],
 [ 3., 7., 11., 15., 19.]])
```

Linear Algebra …
Transpose of a Matrix

- As a special type of the square matrix,a symmetric matrix A is equal to its transpose: A = A⊤.
- B = np.array([[1, 2, 3], [2, 0, 4], [3, 4, 5]])
- B

```
array([[1., 2., 3.],
       [2., 0., 4.],
       [3., 4., 5.]])
```

Linear Algebra …
Matrices…Transpose of a Matrix

- Now we compare B with its transpose.
- o/p

```
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]])
```

Linear Algebra
Basic Properties of Tensor Arithmetic

- Scalars, vectors, matrices, and tensors (“tensors” in this subsection refer to algebraic objects)of an arbitrary number of axes have some nice properties that often come in handy.
- For example
- Two tensors with the same shape, the result of any binary element wise operation will be a tensor of that same shape.

Linear Algebra
Basic Properties of Tensor Arithmetic….

- A = np.arange(20).reshape(5, 4)
- B = A.copy() # Assign a copy of `A` to `B` by allocating new memory
A, A + B

```
(array([[ 0.,  1.,  2.,  3.],
        [ 4.,  5.,  6.,  7.],
        [ 8.,  9., 10., 11.],
        [12., 13., 14., 15.],
        [16., 17., 18., 19.]]),
 array([[ 0.,  2.,  4.,  6.],
        [ 8., 10., 12., 14.],
        [16., 18., 20., 22.],
        [24., 26., 28., 30.],
        [32., 34., 36., 38.]]))
```

Linear Algebra
Sum and Dot Product

- x = np.arange(4) x,
- x.sum()
- o/p
- (array([0., 1., 2., 3.]), array(6.))

- y = np.ones(4)
- x, y, np.dot(x, y)

o/p

- (array([0., 1., 2., 3.]), array([1., 1., 1., 1.]), array(6.))

Linear Algebra
Matrix-Vector Products

- A=[1,2,3,4]
- np.dot(A,2)
- o/p
- `array([2, 4, 6])`

Matrix-Matrix Multiplication

- A=[1 2 3]
- B=[2,3,4]
- Np.dot(A,B)
- o/p
- 20

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \end{bmatrix}$$

Matrix-Matrix Multiplication…

- B = np.ones(shape=(4, 3))
- np.dot(A, B)

- W.A.P to Change the shape of a tensor without altering either the number of elements or their values.
- W.A.P to print two arrays with 3rows and four columns with values zero.
- W.A.P to print two arrays with 3rows and four columns with values ones.
- W.A.P to print a random array of size(3,4)
- Describe preprocessing and explain how do we handle Missing data
- W.A.P to handle missing data in an array.
- Write Python code for finding the probability of tossing a coin.
- Write a Python Program to find the derivative of the following function.

  6x2+3x+4


Thank You!