Applied Analytics and Predictive Modeling

Spring 2020

Lecture-2

Lydia Manikonda

manikl@rpi.edu

Rensselaer

Today’s agenda

- Overview of Data Mining
- Data Preprocessing
- Python Packages – Numpy and Pandas
- Including class exercises
- In-class case

Overview

Core Ideas of Data Mining

- Data and Dimensionality Reduction
- Data Exploration
- Data Visualization
- Association Rules & Recommendation systems
- Classification
- Clustering
- Prediction

Data Exploration

- Data sets are typically large, complex & messy
- Based on the task at hand, we have to process the data
- Use techniques of Reduction and Visualization

Dimensionality Reduction

- Shrinking the complex/large data into simpler/smaller data
- Reducing the number of variables/columns (e.g., principal components) – **Dimensionality Reduction**
- Reducing the number of records/rows (e.g., clustering) -- **Sampling**

Data Visualization

- Very important to understand the data – in particular, to examine the relationships between the attributes
- Graphs and plots of data
- Histograms, boxplots, bar charts, scatterplots

Association Rules

- To identify rules that define “what goes with what” in transactions
- Example: “If X was purchased, Y was also purchased” given a set of transactions
- Very useful in recommendation systems – “Our records show you bought X, you may also like Y”
- Also called as “affinity analysis”

Recommender Systems

- Collaborative filtering – Technique used by recommendation systems
- The main goal is to recommend items that we may like
- Various aspects that customers view, select, purchase, rate, etc
- User-based recommendation: Recommend products that “customers like you” purchase
- Item-based recommendation: Recommend products that share a “product purchaser profile” with your purchases

Supervised Learning

- Given training data (where the target value is known), the goal is to predict a single “target” or “outcome” variable
- Can be classified into two types – Classification and Prediction

Supervised Learning – Classification

- Main aim is to predict an outcome variable (or target variable)
- The target variable can be binary or multi-class

- Examples: Fraudulent or Non-fraudulent transaction; Pass or Fail; Rainy or Sunny; etc.

Supervised Learning – Prediction

- Main aim is to predict the outcome variable (usually in terms of a probability value)
- Common methods that could perform prediction are – Regression
- Examples: performance evaluation, revenue estimation, sales percentage, etc

Unsupervised Learning

- Main aim is to segment data into meaningful segments or detect patterns
- There is no target (outcome) variable to predict or classify
- Common methods include clustering.

Numpy

Loops, conditionals, functions

Numpy

- Fundamental package for scientific computing

- Numpy is a general-purpose array-processing package
- Used for high-performance multidimensional array computations

Numpy – Arrays

- A numpy array is a grid of values, all values are of same type
- The number of dimensions is the **rank** of an array
- A tuple of integers giving the size of an array along each dimension is called the **shape** of an array

- Initialize using nested python lists
- Access using square brackets

Numpy – Arrays

- Declaring the package

```
import numpy as np
```

- Creating an array of rank 1

```
arr = np.array([1, 2, 3])
```

- Creating an array of rank 2

```
arr = np.array([1, 2, 3], [4, 5, 6])
```

Numpy – Arrays

- Create an array with rank 1

```
>> a = np.array([1, 2, 3])
```

- Print the shape of this array

```
>> print(a.shape)
>> (3, )
```

- Print the elements at different indices

```
>> print(a[0], a[1], a[2])
>> 1 2 3
```

- Change an element of the array

```
>> a[0] = 10
```

- Print the array

```
>> print(a)
>> [10, 2, 3]
```

Numpy – Arrays

>> a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])

>> print(a)

Using slicing method printing a range of array

>> sliced_a = a[:2, ::2]

>> print(sliced_a)

Printing elements at specific indices

>> print(a[[1, 2, 1, 3],[1, 0, 2, 3]])

Numpy – Arrays and Functions

```
>> a = np.zeros((2, 2))
>> print(a)
>> [[0.  0.], [0.  0.]]

>> b = np.ones((1, 2))
>> print(b)
>> [[1.  1.]]

>> c = np.full((2,2), 7)
>> print(c)
>> [[7.  7.], [7.  7.]]

>> d = np.eye(2)
>> print(d)
>> [[1.  0.], [0.  1.]]

>> e = np.random.random((2, 2))
>> print(e)
>> [[], []]
```

Numpy – Arrays

- Datatypes of arrays need not be defined – numpy tries to guess the datatype

```
>> a = np.array([1.1, 2.2])
>> print(a.dtype)

>> a = np.array([1, 2], dtype=np.int64)
>> print(a.dtype)
```

Numpy – Math operations

```
>> a = np.array([[1, 2], [3, 4]], dtype=np.float64)
>> b = np.array([[4, 3], [2, 1]], dtype=np.float64)

>> sum_ab = np.add(a, b)
>> print(sum_ab)

>> sum_a = np.sum(a)
>> print(sum_a)

>> sqrt_a = np.sqrt(a)
>> print(sqrt_a)

>> trans_a = a.T
>> print(trans_a)
```

Numpy – Exercises

1. Given an array, print only a range of the array using slicing method.

   Input: [[-1, 2, 0, 4], [4, -0.5, 6, 0], [2.6, 0, 7, 8], [3, -7, 4, 2.0]]

   Output: [[-1. 0.] [4. 6.]]

2. Consider the array above and print elements at specific indices

   Input: [[-1, 2, 0, 4], [4, -0.5, 6, 0], [2.6, 0, 7, 8], [3, -7, 4, 2.0]];

   Values at these indices[[1, 1, 0, 3], [3, 2, 1, 0]]

   Output: [0., 6., 2., 3.]

3. Add two given arrays; a = np.array([[1, 2], [3, 4]]) b = np.array([[4, 3], [2, 1]])

Numpy – Exercises

4. Given a numpy array, find the datatype: np.array([4.0, 9.0])
5. Consider the previous array and perform the square root of an array.
6. Get unique values in a list using numpy
7. Multiply all the numbers in a given list using numpy
8. Create a random numpy array of 20 rows and 2 columns

Pandas

- Most popular python library for data analysis
- Highly optimized performance

- Using: 1) **Series**; 2) **DataFrames**

Pandas – Series

- One dimensional array to store any data type

```
>> import pandas as pd
>> a = pd.Series(data, index = Index)
```

- **data** can be:
  - Scalar value – integer, string
  - Dictionary – <key, value> pair
  - Ndarray

- **Index** by default is from 0, 1, 2, … (*n*-1) where *n* is the length of the data

Pandas – Series

>> data = [1, 2, 3, 4, 5, 6, 7]

>> s = pd.Series(data)

0	1
1	2
2	3
3	4
4	5
5	6
6	7

dtype: int64

>> Index = [‘a’, ’b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’]

>> s1 = pd.Series(data, Index)

a	1
b	2
c	3
d	4
e	5
f	6
g	7

dtype: int64

Pandas – Series

>> diction = {‘a’: 1, ‘b’:2, ‘c’:3, ‘d’:4, ‘e’:5, ‘f’:6, ‘g’:7}

>> s = pd.Series(dictionary)

```
a    1
b    2
c    3
d    4
e    5
f    6
g    7
dtype: int64
```

Pandas – Dataframes

- DataFrames is two-dimensional data structure that consists of rows and columns

```
>> import pandas as pd
>> s = pd.DataFrame(data)
```

data can be:

- One or more dictionaries
- One or more series
- 2D-numpy Ndarray

Pandas – DataFrames

```
>> diction1 ={'a':1, 'b':2, 'c':3, 'd':4}
>> diction2 ={'a':5, 'b':6, 'c':7, 'd':8, 'e':9}
>> data = {'first':diction1, 'second':diction2}
>> df = pd.DataFrame(data)
```

	first	second
a	1.0	5
b	2.0	6
c	3.0	7
d	4.0	8
e	NaN	9

Pandas – DataFrames

```
>> import pandas as pd
>> s1 = pd.Series([1, 3, 5, 7, 9, 11, 13])
>> s2 = pd.Series([1.1, 2.2, 3.3, 4.4, 5.5, 6.6])
>> s3 = pd.Series(['a', 'b', 'c', 'd', 'e'])

>> data ={'first':s1, 'second':s2, 'third':s3}
>> series = pd.DataFrame(data)
```

	first	second	third
0	1	1.1	a
1	3	2.2	b
2	5	3.3	c
3	7	4.4	d
4	9	5.5	e
5	11	6.6	NaN
6	13	NaN	NaN

Pandas – Series vs DataFrames

- Series is 1-D whereas a DataFrame is 2-D.
- A one column DataFrame can have a name for that one column but a Series cannot have a column name.
- Each column of a DataFrame can be converted to a series.

Pandas – DataFrames – Example1

- Create a dataframe using a list

```
>> import pandas as pd
>> strlist = ['I', 'love', 'machine', 'learning']

>> df = pd.DataFrame(strlist)
>> print(df)
```

	0
0	I
1	love
2	machine
3	learning

Pandas – DataFrames – Example2

```
>> import pandas as pd
>> data = {'Name':['John', 'William', 'Ian', 'Noah'],
           'Age':[12, 15, 13, 12]}

df = pandas.DataFrame(data)
print(df)
```

	Name	Age
0	John	12
1	William	15
2	Ian	13
3	Noah	12

Pandas – Dataframes

- Given a small dataset, create a dataframe and print only two columns Name and Address.

Name	Age	Address	Qualification
John	24	New York	MS
Jim	25	Arizona	BS
Ashley	22	Minnesota	BS
Aimee	23	California	MS
Jeff	27	New York	PhD

Pandas – DataFrames – nba.csv

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0	PG	25	2-Jun	180	Texas	7730337
Jae Crowder	Boston Celtics	99	SF	25	6-Jun	235	Marquette	6796117
R.J. Hunter	Boston Celtics	28	SG	22	5-Jun	185	Georgia State	1148640
Jonas Jerebko	Boston Celtics	8	PF	29	10-Jun	231		5000000
Amir Johnson	Boston Celtics	90	PF	29	9-Jun	240		12000000
Jordan Mickey	Boston Celtics	55	PF	21	8-Jun	235	LSU	1170960
Kelly Olynyk	Boston Celtics	41	C	25	Jul-00	238	Gonzaga	2165160
Terry Rozier	Boston Celtics	12	PG	22	2-Jun	190	Louisville	1824360

Pandas – DataFrames

Retrieving a player's information

```
>> Import pandas as pd
>> data = pd.read_csv("nba.csv", index_col="Name")
>> first = data.loc["Avery Bradley"]
>> second = data.loc["R.J. Hunter"]

>> print(first)
>> print(second)
```

```
Team        Boston Celtics
Number                   0
Position                PG
Age                     25
Height                 6-2
Weight                 180
College              Texas
Salary         7.73034e+06
Name: Avery Bradley, dtype: object
```

```
Team        Boston Celtics
Number                  28
Position                SG
Age                     22
Height                 6-5
Weight                 185
College      Georgia State
Salary         1.14864e+06
Name: R.J. Hunter, dtype: object
```

Pandas – DataFrames

- Retrieving a single column

>> Import pandas as pd

>> data = pd.read_csv("nba.csv", index_col="Name")

>> first = data["Age"]

```
Jonas Jerebko               29.0
Amir Johnson                29.0
Jordan Mickey               21.0
Kelly Olynyk                25.0
Terry Rozier                22.0
Marcus Smart                22.0
Jared Sullinger             24.0
Isaiah Thomas               27.0
Evan Turner                 27.0
James Young                 20.0
Tyler Zeller                26.0
Bojan Bogdanovic            27.0
Markel Brown                24.0
Wayne Ellington             28.0
Rondae Hollis-Jefferson     21.0
Jarrett Jack                32.0
Sergey Karasev              22.0
Sean Kilpatrick             26.0
Shane Larkin                23.0
Brook Lopez                 28.0
Chris McCullough            21.0
Willie Reed                 26.0
Thomas Robinson             25.0
Henry Sims                  26.0
Donald Sloan                28.0
Thaddeus Young              27.0
                            ...
Al-Farouq Aminu             25.0
Pat Connaughton             23.0
Allen Crabbe                24.0
Ed Davis                    27.0
```

Pandas – DataFrames

- How can we print the entire dataframe?

```
for i, j in df.iterrows():
	print(i, j)
```

Pandas – Missing values

- To check if there are any missing values

```
import pandas as pd
import numpy as np
dict1 = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}
df = pd.DataFrame(dict1)
df.isnull()
```

Pandas – Fill Missing Values

```
dict1 = {'First Score':[100, 90, np.nan, 95],
 'Second Score': [30, 45, 56, np.nan],
 'Third Score':[np.nan, 40, 80, 98]}

df = pd.DataFrame(dict1)
df.fillna(0)
```

Pandas – Drop the rows with missing values

```
dict1 = {'First Score':[100, 90, np.nan, 95],
 'Second Score': [30, 45, 56, np.nan],
 'Third Score':[np.nan, 40, 80, 98]}

df = pd.DataFrame(dict1)

df.dropna()
```

Pandas – ffill()

- Missing values are replaced with the previous row’s column value

College
Texas
Marquette
Boston University
Georgia State
No College
No College
LSU
Gonzaga

College
Texas
Marquette
Boston University
Georgia State
Georgia State
Georgia State
LSU
Gonzaga

- nba["College"].fillna(method ='ffill', inplace = True)

Pandas – ffill()

- We can set a limit (by using *limit*) on successful replacement of NaN values.

College
Texas
Marquette
Boston University
Georgia State
No College
No College
LSU
Gonzaga

College
Texas
Marquette
Boston University
Georgia State
Georgia State
NaN
LSU
Gonzaga
Louisville

- nba["College"].fillna(method ='ffill', limit = 1, inplace = True)

Pandas – Groupby()

- It is used to split the data into groups based on some criteria.
- For example, use the nba.csv to group the data based on the “Team”

```
>> import pandas as pd
>> df = pd.read_csv("nba.csv")
>> df
>> gbdata = df.groupby(‘Team’)
>> gbdata.first()
>> gbmean = df.groupby(‘age’).mean()
>> gbmean
```

Summary of the dataframe

```
>> print(df.info())
```

Pandas – Class Exercise

1. How can we retrieve a row by their index number? For example, index=3?
2. How do you find the number of rows and number of columns in the dataframe?
3. For nba.csv dataset, find the number of rows that have missing values.
4. Replace all the missing values in nba.csv with 0.

What is data?

- Collection of **data objects** and their **attributes**
- According to Tan et al.,
- An **attribute** is a property or characteristic of an object
  - Also known as variable, field, characteristic, dimension, or feature
- A collection of attributes describe an **object**
  - Also known as tuple, record, point, case, sample, etc.

Attributes

Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

More views of data

- Data may have parts
- The different parts of data may have relationships
- More generally, data may have structure
- Data can be incomplete

Attribute values

- Attribute values are numbers or symbols assigned to an attribute for a particular object
- Distinction between attributes and attribute values
  - Same attribute can be mapped to different attribute values
    - Example: Height can be measured in feet or meters
  - Different attributes can be mapped to the same set of values
    - Example: Attribute values for ID and age are integers
    - But properties of attribute values can be different

Types of Attributes

- Nominal
  - Examples: ID numbers, zip codes, eye color
- Ordinal
  - Examples: Rankings (expertise level on a scale of 1-10), grades, height {tall, medium, short}
- Interval
  - Examples: Calendar dates, temperature in Celsius or Fahrenheit
- Ratio
  - Examples: Temperature in Kelvin, length, time, counts

Discrete and Continuous attributes

- Discrete Attribute:
  - Has only a finite or countably infinite set of values
  - Examples: zip codes, counts, or the set of words in a collection of documents
  - Often represented as integer variables.
  - Note: binary attributes are a special case of discrete attributes
- Continuous Attribute:
  - Has real numbers as attribute values
  - Examples: temperature, height, or weight.
  - Practically, real values can only be measured and represented using a finite number of digits.
  - Continuous attributes are typically represented as floating-point variables.

Types of datasets

- Record
  - Data Matrix
  - Document Data
  - Transaction Data
- Graph
  - World Wide Web
  - Molecular Structures
- Ordered
  - Spatial Data
  - Temporal Data
  - Sequential Data
  - Genetic Sequence Data

Important characteristics of data

- Dimensionality (number of attributes)
  - High dimensional data brings a number of challenges
- Sparsity
  - Only presence counts
- Resolution
  - Patterns depend on the scale
- Size
  - Type of analysis may depend on size of data

Record data

- Data that consists of a collection of records, each of which consists of a fixed set of attributes

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Document data

- Each document becomes a ‘term’ vector
  - Each term is a component (attribute) of the vector
  - The value of each component is the number of times the corresponding term occurs in the document.

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

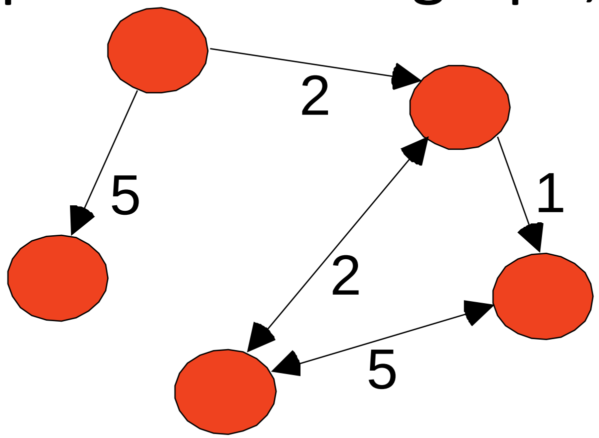
Transaction data

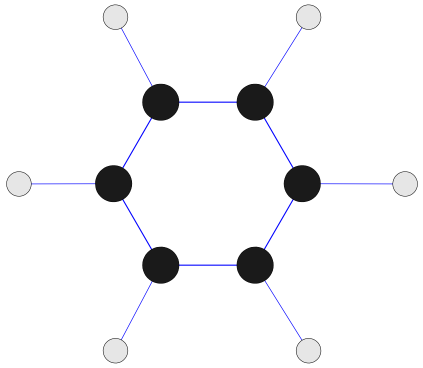
- A special type of record data, where
  - Each record (transaction) involves a set of items.
  - For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

TID	*Items*
1	**Bread, Coke, Milk**
2	**Beer, Bread**
3	**Beer, Coke, Diaper, Milk**
4	**Beer, Bread, Diaper, Milk**
5	**Coke, Diaper, Milk**

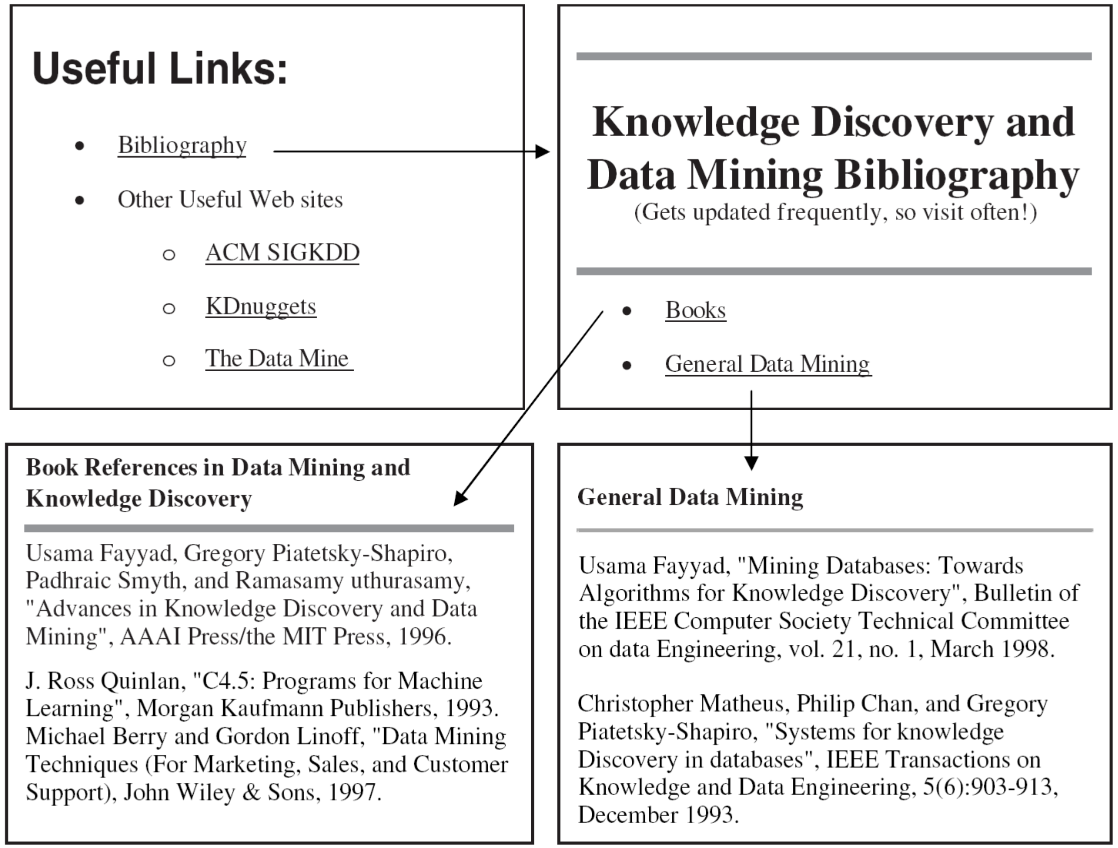
Graph Data

- Examples: Generic graph, a molecule, and webpages





Benzene Molecule: C_6H_6



Ordered Data

- Sequences of transactions

Items/Events

(A B) (D) (C E)
(B D) (C) (E)
(C D) (B) (A E)

An element of the sequence

Ordered Data

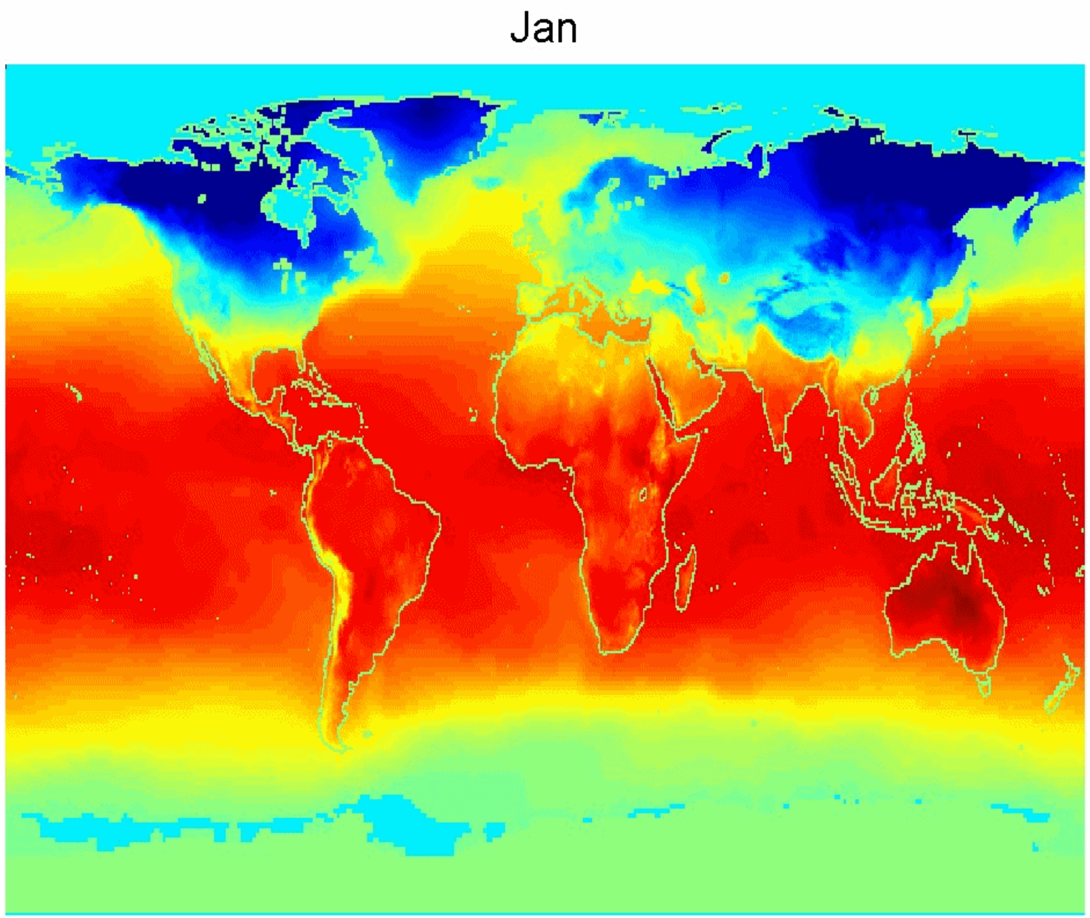
- Genomic sequence data

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

Ordered Data

- Spatio-temporal data

Average Monthly Temperature of land and ocean



Examples

- ID numbers
  - Nominal, ordinal, or interval?

- Number of cylinders in an automobile engine
  - Nominal, ordinal, or ratio?

- Biased Scale
  - Interval or Ratio

Data Preprocessing

- Aggregation
- Sampling
- Dimensionality Reduction
- Feature subset selection
- Feature creation
- Discretization and Binarization
- Attribute Transformation

Aggregation

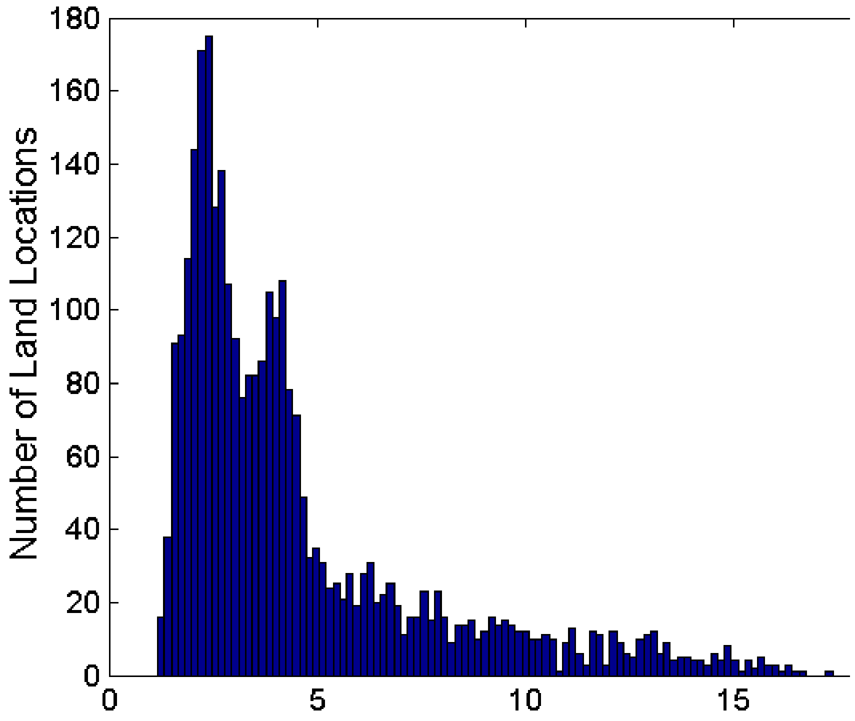
- Combining two or more attributes (or objects) into a single attribute (or object)
- Purpose
  - Data reduction
    - Reduce the number of attributes or objects
  - Change of scale
    - Cities aggregated into regions, states, countries, etc.
    - Days aggregated into weeks, months, or years
  - More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

Example: Precipitation in Australia

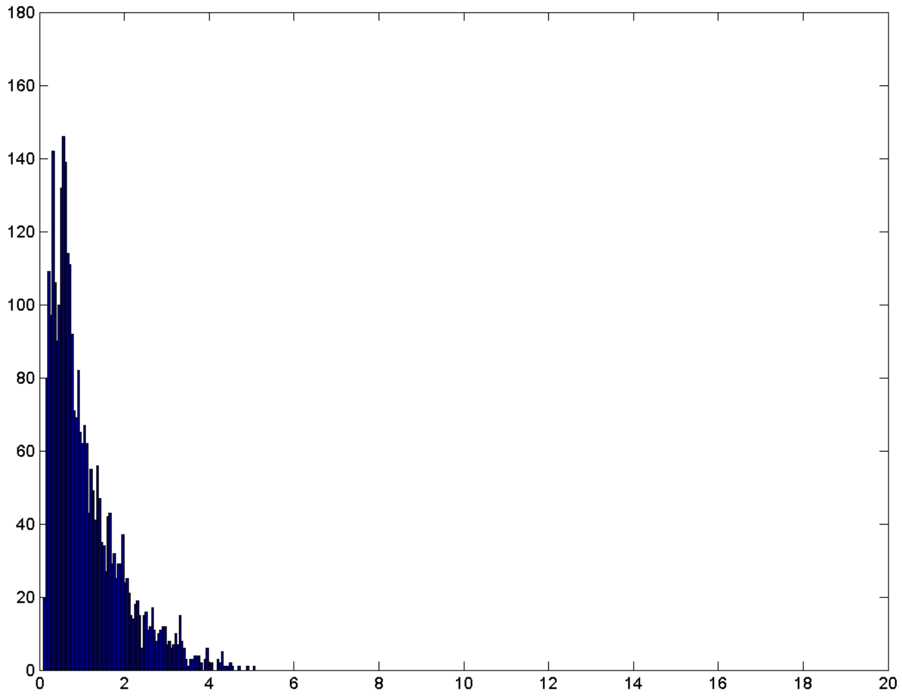
- This example is based on precipitation in Australia from the period 1982 to 1993.
- The next slide shows
  - A histogram for the standard deviation of average monthly precipitation for 3,030 0.5° by 0.5° grid cells in Australia, and
  - A histogram for the standard deviation of the average yearly precipitation for the same locations.
- The average yearly precipitation has less variability than the average monthly precipitation.
- All precipitation measurements (and their standard deviations) are in centimeters.

Example: Precipitation in Australia..

- Variation of precipitation in Australia



Standard Deviation of Average Monthly Precipitation



Standard Deviation of Average Yearly Precipitation

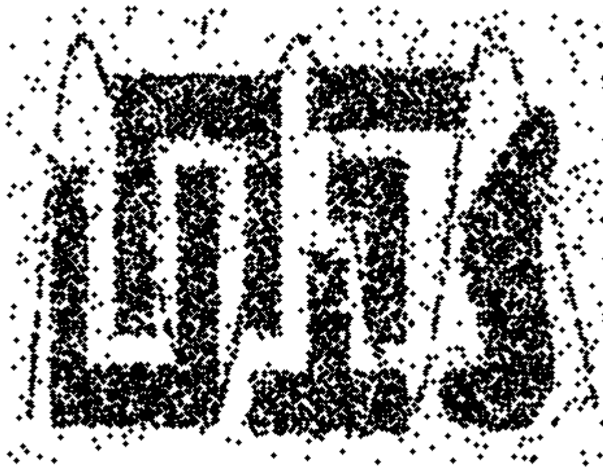
Sampling

- Sampling is the main technique employed for data reduction.
  - It is often used for both the preliminary investigation of the data and the final data analysis.
- Statisticians often sample because obtaining the entire set of data of interest is too expensive or time consuming.
- Sampling is typically used in data mining because processing the entire set of data of interest is too expensive or time consuming.

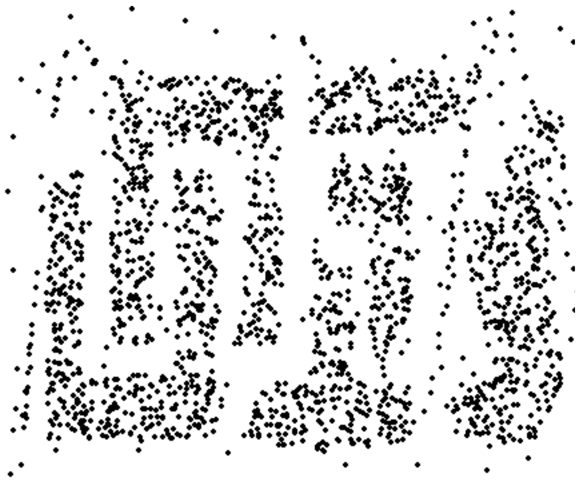
Sampling

- The key principle for effective sampling is the following:
  - Using a sample will work almost as well as using the entire data set, if the sample is representative
  - A sample is representative if it has approximately the same properties (of interest) as the original set of data

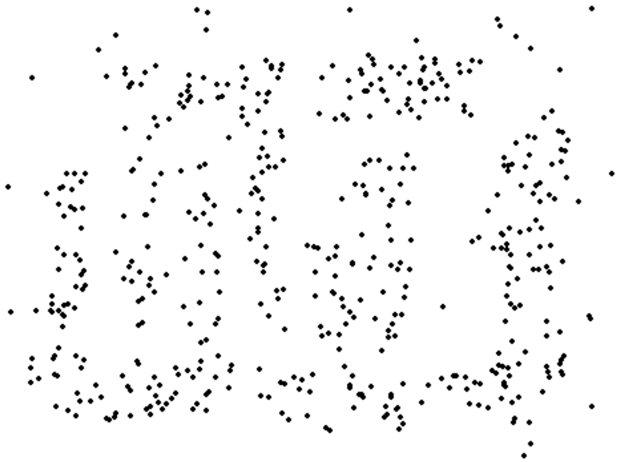
Sample size



8000 points



2000 Points



500 Points

Types of Sampling

- Simple Random Sampling
    - There is an equal probability of selecting any particular item
    - Sampling without replacement
        - As each item is selected, it is removed from the population
    - Sampling with replacement
        - Objects are not removed from the population as they are selected for the sample.
        - In sampling with replacement, the same object can be picked up more than once
- Stratified sampling
    - Split the data into several partitions; then draw random samples from each partition

Curse of dimensionality

When dimensionality increases, data becomes increasingly sparse in the space that it occupies

Dimensionality Reduction

- Purpose:
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by data mining algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise

- Techniques
  - Principal Components Analysis (PCA)
  - Singular Value Decomposition
  - Others: supervised and non-linear techniques

Feature subset Selection

- Another way to reduce dimensionality of data
- Redundant features
  - Duplicate much or all of the information contained in one or more other attributes
  - Example: purchase price of a product and the amount of sales tax paid
- Irrelevant features
  - Contain no information that is useful for the data mining task at hand
  - Example: students' ID is often irrelevant to the task of predicting students' GPA
- Many techniques developed, especially for classification

Feature Creation

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes

- Three general methodologies:
  - Feature extraction
    - Example: extracting edges from images
  - Feature construction
    - Example: dividing mass by volume to get density
  - Mapping data to new space
    - Example: Fourier and wavelet analysis

Discretization

- Discretization is the process of converting a continuous attribute into an ordinal attribute
  - A potentially infinite number of values are mapped into a small number of categories
  - Discretization is commonly used in classification
  - Many classification algorithms work best if both the independent and dependent variables have only a few values

Binarization

- Binarization maps a continuous or categorical attribute into one or more binary variables

- Typically used for association analysis

- Often convert a continuous attribute to a categorical attribute and then convert a categorical attribute to a set of binary attributes
  - Association analysis needs asymmetric binary attributes
  - Examples: eye color and height measured as {low, medium, high}

Attribute Transformation

- An attribute transform is 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
  - Simple functions: x^k, $\log(x)$, e^x, $|x|$
  - Normalization
    - Refers to various techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range
    - Take out unwanted, common signal, e.g., seasonality
  - In statistics, standardization refers to subtracting off the means and dividing by the standard deviation

Seaborn

- Python library to generate good graphs that provide lot of insights

In-class Case

Iris dataset

Use this dataset to perform different preprocessing techniques to understand the dataset.

At the end of this exercise, every team should submit their best visualization and a small description why it is good at explaining the data on Piazza thread.