CS5100: Data Analysis — Rosite — 29/09/21 — ①

Week 1: Introduction

Topics

1. Introduction
   - A data analysis example
2. Data
   - features and labels (input & output variables)
   - type of data
   - type of variables
3. Preprocessing
   - Data dirty &
   - data transformation

What this course is about

- Algorithms for data analysis
  - KNN - Decision Trees - neural nets etc
- How to evaluate their quality
  - validation - error rate
  - statistical comparison etc.
- How to apply them to real data
  - supervised and unsupervised learning
  - inference
  - prediction

Introduction: A data analysis example

Data analysis includes

- Machine learning
  - trainable & computationally efficient algorithms
- Statistics
  - traditional data science
  - focus on the data-generating distributions
- Data mining
  - large quantities of data
  - visualization.

Data

i) • Input and output.
- Input (attributes, features)
  • denoted using the symbol X with a subscript to distinguish them e.g. X_i
- output (labels)
  • denoted using the symbol Y

ii) • Type of variables
• Variables can be either
  - quantitative or
  - qualitative (categorical)

Quantitative variables (Discreet & continuous)
- take on numerical values
  e.g. height, income, the value of a house

Qualitative variable (Nominal, Oridinal & Binary)
- take on values in a set, C, of classes (categories).
  e.g. A person's gender: C = {male, female, other}

• Attributes can be either
  - discrete or
  - continuous

• Discrete attributes
- have only a finite set of possible values
- (often an integer variable)

e.g.
→ Nominal attributes
  e.g. $X \in$ {red, yellow, blue} or
  the days of the week
→ Oridinal attributes - A person's age
  - (numerical & ordered) e.g. $X \in \{0, 2, 5, 10\}$.
→ Binary attributes e.g. $X \in$ {yes, no} or $X \in \{0, 1\}$

Continuous attributes
- have real numbers as values
  e.g. A person's weight

Note: On Variable types
- Real values can only be measured & represented using a finite number of digits.
- Discrete variables are not necessarily qualitative variables,
  e.g. A person's age
- Qualitative variables are naturally represented by nominal attributes
  - but can arbitrarily associated with ordinal attributes for practical purposes,
    e.g. yellow → 1, red → 2, blue → 3

Outliers
- are data objects with characteristic that are considerably different from most of the other data objects in the data set
- can be legitimate objects or values
- may be of interest
  e.g. for network intrusion or fraud detection
- Some algorithms may produce poor results in the presence of outliers
- Defining and detecting outliers is a non-trivial problem

Types of data
- Data can be
  - structured or
  - unstructured
- Structured data
  - have a well-known data format
  e.g. → n objects with d measurements represented by n x d matrix
  → with the d columns called features, attributes, or fields

- Unstructured data
  - has no inherent structure (& no additional annotation)
  e.g. - plain text documents
  - images
  - video ₹
  = apparent organisation makes no guarantees.

Analysis of unstructure data

- Unstructured data are common & open to different kinds of analysis:

① Example: A corpus of plain-text documents/sentences
e.g. Given a dictionary $\mathbb{D}$ = {hi, good, hello, world, earth, bye}

A text documents, e.g. hello, world, hi, world
can be represented by numerical vectors
e.g. $V = [1, 0, 1, 1, 0, 0]$
where V_i is the number of times the i^{th} word of $\mathbb{D}$ appears in the document

② Example: is a data set of digitized audio recordings, pictures, or videos.

- Also in this case the data objects can be represented by vectors, matrices, & tensors

e.g. A picture can be transformed into a matrix, V where the $(i,j)^{th}$ entry represents the value of the $(i,j)^{th}$ pixel

Time series data

- A data set of objects gathered sequentially in time is called a time series.

- Observations
  - are made at distinct (equally spaced) points in time
  - and may take values on a continuous range or a finite set.

- aims to understant the dynamical mechanism generating the data or predict future events given the past.

Data preprocessing

In the real world, data are dirty.

- Incomplete
  - lacking attribute values
  - lacking certain attributes of interest
  - or containing only aggregate data
- Noisy
  - containing errors from the acquisition device
  - but also post-acquisition human or computer errors
- Inconsistent
  - containing discrepancies
    e.g. from combining different data sources or deplicate records

Data preprocessing tasks

1. Data cleaning
   e.g. - fill in missing values
   - smooth noisy data
   - resolve other inconsistencies
2. Data transformation
   e.g. - normalization
   - aggregation
   - discretization
   - reduction

3. Outliers handling
- often through clustering
- data reduction

Missing data
- may be due to
  - faulty equipment
  - misunderstanding
  - negligence
  - anonymisation
- may make data analysis conclusions
  - statistically less strong or
  - algorithms' training much harder

Data analysis with missing values requires to
- identify the pattern & reason of mission values
- Understan the distribution of missing values
  e.g. detect possible context/attribute-dependencies
- carefully delete observations with
  - missing values
  - imputate missing values

Noisy data
Noise
- is the random component of measurement error
  e.g. distortion of a person's voice when talking over a poor phone line.
- is hard to remove without losing some of the useful information (signal)
- It is more efficient to develop robust algorithms, i.e. algorithm that takes noise into account
- Another option is to smooth the signal by fitting a regression function to the data (and replace the data with the model out-put)

Data transformation

1. Smoothing is an example of data transformation

Let the data be:

$$D = \{x_i\}_{i=1}^{8} = \{1, 2, 3, 2, 3, 1, 4, 3\}$$

2. Logarithm trasform

$$x_i \rightarrow \log x_i$$

EX: $D \rightarrow \{0.0, 0.69, 1.1, 0.69, 1.1, 0.0, 1.39, 1.1\}$

R: $lX = \log(X)$ python: $lX = [numpy.log(x) \text{ for } x \text{ in } X]$

3. k-step simple moving average (For time series)

$$x_i \rightarrow \frac{1}{k}\sum_{j=1}^{k} x_{i-k+j}, \text{ if } i > k$$

EX: (For $k=3$)

$x_1 = 1$, $x_{i-k+j} = x_{1-3+1} = x_{-1} = 3$ & $\frac{3}{3} = 1.0$

$x \quad = x_{2-3+1} =$

$= x_{3-3+}$

$D \rightarrow \{1.00, 2.00, 2.00, 2.33, 2.67, 2.00, 2.67, 2.67\}$

R:

```
smaX = X and then for (i in 3:length(X))
smaX[i] = mean(c(X[i-2], X[i-1], X[i]))
```

Python:

```
smaX = [X[0], X[1]] + [numpy.mean([X[i-2], X[i-1], X[i]])
    for i in range(2, len(X))]
```

4. Aggregation
- replace subset of data objects with their aggregates,
e.g.

$$x_i, x_{i+1} \rightarrow \frac{1}{2} * (x_i + x_{i+1})$$

Ex: $\mathbb{D} \rightarrow \{1.5, 2.5, 2, 3.5\}$.

agg = [½*(X[j]+X[j+1]) for j in range(0, len(X)-1, 2)]

5. Min-max normalization

$$x_i \rightarrow \frac{(x_i - \min \mathbb{D})}{\max \mathbb{D} - \min \mathbb{D}}$$

Ex: {0.0, 0.33, 0.67, 0.33, 0.67, 0.0, 1.0, 0.67}.

R:

$$mmX = \frac{(X - \min(X))}{(\max(X) - \min(X))}$$

python:

mmX = [(x-numpy.min(X))/(numpy.max(X)-numpy.min(X)) for x in X]

6. standardization (z-score normalization)

$$x_i = \frac{x_i - \mu}{\sigma}$$

where; $\mu = |\mathbb{D}|^{-1} \sum_i x_i$ (mean) & $\sigma^2 = |\mathbb{D}|^{-1} \sum_i (x_i - \mu)^2$

R: $sX = \frac{(x - mean(X))}{sd(X)}$

Python:

sX = [(x - numpy.mean(X))/(numpy.std(X)) for x in X.]

Discretization

- Sometimes you need to make a continuous variable non-numerical,

Eg. Some classification algorithms only accept categorical attributes, or to reduce data size.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be your input-output data.

Discretization consists of splitting:

1. The attribute space $\mathcal{X}$ into non-overlapping intervals, I_k, such that $\mathcal{X} = \cup_k I_k$

$$R = I_1 \cup I_2 \cup I_3, \quad I_1 = [-\infty, a], \; I_2 = (a, b], \; I_3 = (b, \infty]$$

2. The data set into bins, $\mathcal{D}_k$ subsets of $\mathcal{D}$, such that

$$x_i \in \mathcal{D}_k \Rightarrow x_i \in I_k$$

Example: Unsupervised discretization
- ignoring the labels

Let you data set be

$$\mathcal{D} = \{(2.5, 0), (6.7, 1), (3.4, 1), (5.9, 0), (10, 1), (7, 0)\}$$

- Equal-interval binning (unsupervised):
  - divide the attribute values range into equal intervals,

Eg.

$$I_1 = [1, 4], \; I_2 = (4, 7), \; I_3 = (7, 10]$$

- Equal-frequency binning (unsupervised):
  - divide the range into intervals that contains an equal number of data points.

Ex. $I_1 = [1, 2.5], \; I_2 = (2.5, 5.9], \; I_3 = (5.9, 10]$

Advantages: may help training algorithms
Problems:
- different occurrences of the same continuous value may need to be assigned to different bins.

Example: supervised discretization
- using the data labels

• Entropy-based binning (supervised):
- divide the range into intervals that maximise the "purity" of the intervals.

$$\mathcal{D} = \{(2.5, 0), (6.7, 1), (3.4, 1), (5.9, 0), (10, 1), (1, 0)\}$$

$$I_1 = [1, 3.4), \quad I_2 = [3.4, 6.7), \quad I_3 = [6.7, 10]$$

* Advantages: makes data look cleaner
* Problems: the partition needs to be computed recursively and using the labels may distort training or further data analysis.

How to measure the purity (or the impurity) of an interval?

Given $I = [a_I, b_I] \subseteq \mathbb{R}$, and a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, let the associated (data) bin be

$$\mathcal{D}_I = \{(x_i, y_i) \text{ if } x_i \in I\} \subseteq \mathcal{D}$$

The impurity of $\mathcal{D}_I$ is the entropy of $\mathcal{D}_I$, i.e.

$$H(\mathcal{D}_I) = -\sum_k P_k \log_2 P_k, \quad P_k = |\mathcal{D}_I|^{-1} \sum_{(x,y) \in \mathcal{D}_I} 1_{y=k}$$

The purest bins,
- which contain only objects from one class,
- have the lowest possible entropy, $H = 0$,
  E.g. if $P_1 = 1$ and $P_k = 0$ for all $k \neq 1$

The most impure bins,
- which are associated with a uniform class distribution i.e. $P_k = 1/k$, have the highest possible

$$H = -k \frac{1}{k} \log_2 \frac{1}{k} = \log_2 k$$

- If the attribute space is partitioned into two intervals, I_1 and I_2,
- The data set D is partitioned into two bins D_{I_1} & D_{I_2}

The information gain after partitioning is

$$H(D) - \left(\frac{|D_{I_1}|}{|D|} H(D_{I_1}) + \frac{|D_{I_2}|}{|D|} H(D_{I_2}) \right)$$

- The boundary that maximizes the information gain over all possible boundaries is selected as binary discretization of D.

- The process continues recursively by partitioning the partitions obtained in the previous step until some stopping criterion is met.