Data Mining

Pré-processamento - Transformação/Integração

Prof. Dr. Joaquim Assunção

DEPARTAMENTO DE COMPUTAÇÃO APLICADA
CENTRO DE TECNOLOGIA
UFSM
2024

www.inf.ufsm.br/~joaquim

Fair user agreement

Este material foi criado para a disciplina de Mineração de Dados - Centro de Tecnologia da UFSM.

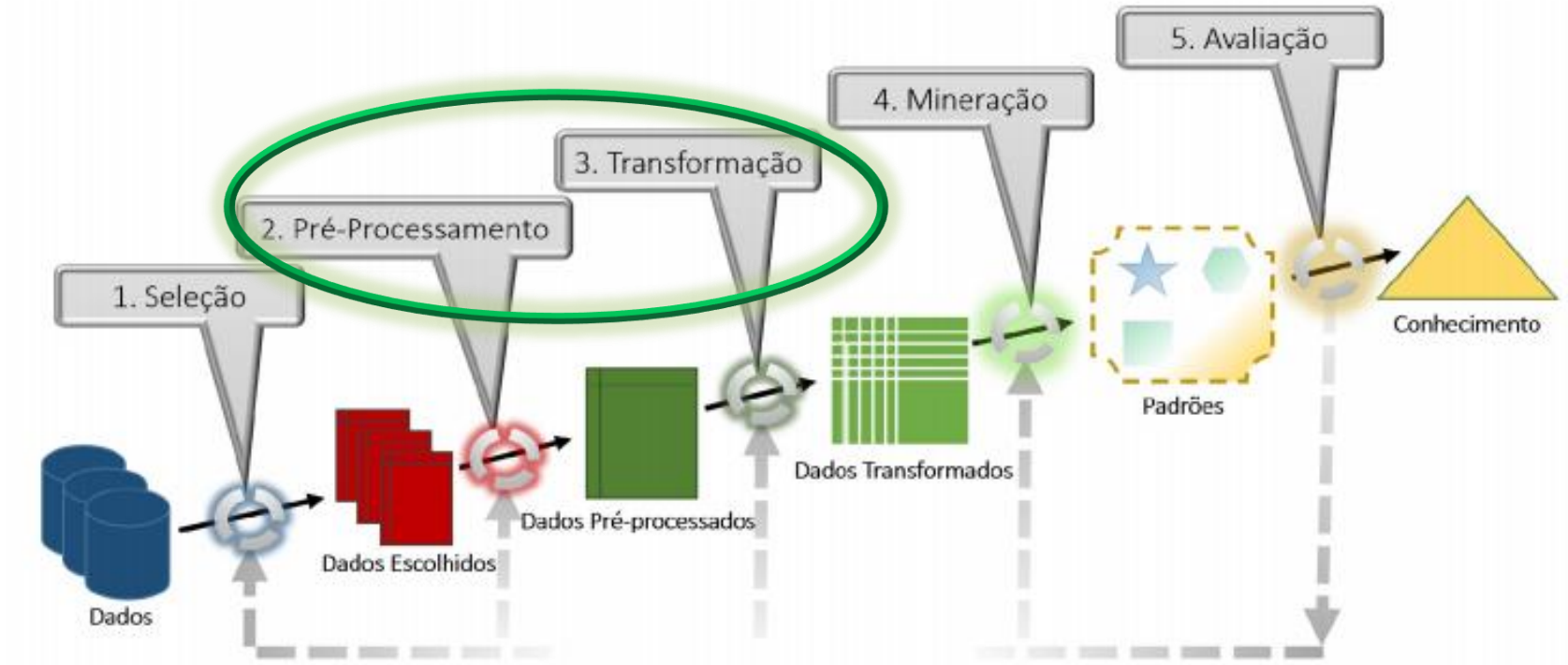
Você pode usar este material livremente*; porém, caso seja usado em outra instituição, **me envie um e-mail** avisando o nome da instituição e a disciplina.

*A maior parte deste material foi retirado do livro: “Joaquim V. C. Assunção. **Uma Breve Introdução à Mineração de Dados: Bases Para a Ciência de Dados, com Exemplos em R**. 192 páginas. Novatec. 2021. ISBN-10 : 6586057507.”

Prof. Dr. Joaquim Assunção.

joaquim@inf.ufsm.br

Transformação



Problemas comuns

- Comumente temos diversas fontes de dados e precisamos unir estes dados para um objetivo comum.

Exemplos

- Imagens com dados de usuário.
- Dados de um usuário, salvos em diferentes sistemas e formatos. E.g., *csv*, *xml* e *json*.
- Dados em diferentes organizações: variáveis em colunas e variáveis em linhas; representações descritivas ou numéricas; numéricos e nominais, etc.

Problemas comuns

- **Dados heterogêneos**: que não possuem chave comum.
- **Definição diferente**: Mesmos dados com definições diferentes, como um esquema de banco de dados diferente.
- **Sincronização de tempo**: verifica se os dados são reunidos nos mesmos períodos de tempo.
- **Dados legados**: referem-se a dados deixados do sistema antigo.
- **Fatores Sociológicos**: Este é o limite de coleta de dados.

Abordagens comuns

- Dados heterogêneos → **Problema de identificação de entidade:** ... integração do esquema e a correspondência de objetos. E.g., nomes e match por strings.
- Definição diferente → **Redundância e análise de correlação:** algumas redundâncias podem ser detectadas por análise de correlação. Dados dois atributos, tal análise pode medir quão fortemente um atributo implica o outro, com base nos dados disponíveis.

Abordagens comuns

- **Duplicação de Registros:** A duplicação deve ser detectada no nível de registro para detectar redundâncias entre os atributos
- **Detecção e resolução de conflitos de valores de dados**: os atributos podem diferir no nível de abstração, onde um atributo em um sistema é registrado em um nível de abstração diferente.

Redução da dimensionalidade

Por quê?

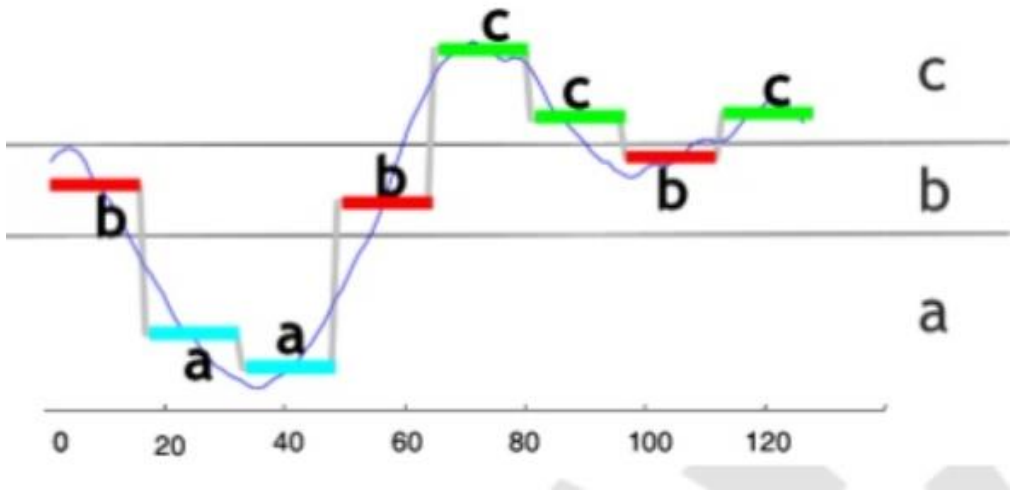
- A redução da dimensionalidade é frequentemente necessária na análise de conjuntos de dados multivariados complexos, que estão sempre no formato de alta dimensionalidade.
- Existem muitos métodos para redução de dimensão de dados, estes métodos estão espalhados por várias subáreas.

Objetivo

- O objetivo da redução de dimensionalidade é reduzir o espaço de estados de forma com que computar os dados tenha um custo menor e a perda de informação não seja muito significativa.
- São exemplos:
  - Algoritmos de compactação de imagens.
  - Dados contínuos para discretos.
  - Redução das matrizes de dados.
  - Eliminação da esparcidade.

Exemplo – Séries temporais

- Dado um sinal contínuo podemos gerar uma saída com menor amplitude de valores.



- A série da acima poderia ser representada por `X={2,1,1,2,3,3,2,3}` ou por `X={b,a,a,b,c,c,b,c}`

Exemplo – Sensor de temperatura

- Dada um conjunto X com 1000 dados, coletados a cada minuto, com amplitude sendo dada por 1 casa decimal de precisão.

```
X={22.2, 22.4, 23, 23.4, 23.2, ..., 20.1 }
```

- Em casos normais, poderíamos diminuir a dimensionalidade de tempo e amplitude unindo dados por tempo (média?!) e diminuindo a precisão.

Technical help

Em R:

- Use: `read.csv()` para ler um arquivo csv. `read.csv("meuArquivo.csv", header = FALSE)`
- Use: `factor(var)` para ver quantos elementos diferentes a variável possui. Use `dim(dataFrame)` para ver as dimensões físicas de um Data Frame. Use `length(unique())` para obter o tamanho único de um vetor (dimensão) ou `nrow(unique())` para um Data Frame.

Hands on!

1. Abra o arquivo "*DimR00.csv*". Use plot() para ver os dados. Reduza a dimensão dos dados para 8. Use lines() para ver os novos dados.

Technical help

Matrizes podem ser criadas em R usando a função `matrix` que possui o seguinte escopo: `matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)`
Use o código abaixo para criar a matriz `myMat`

```
conta <- 0
myMat <- matrix(nrow=10,ncol=10)
for (i in 1:10)
  for (j in 1:10) {
    if(i==j){
      conta <- conta+1
      myMat[i,j] <- conta
    } else{
      myMat[i,j] <- 0
    }
  }
```

Hands on!

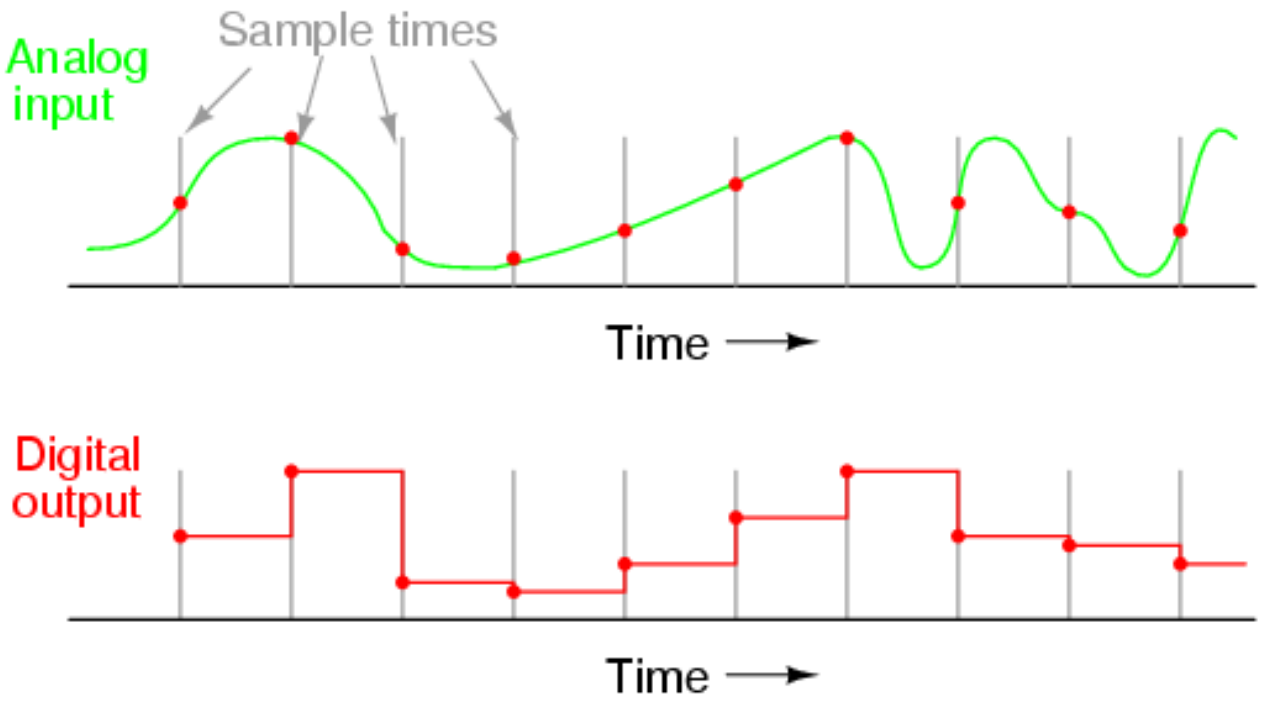
2. Use `dim()` para obter a dimensão da matriz. Reduza a dimensão da matriz myMat para 10.

Algumas técnicas de redução de dimensionalidade

- Principal Component Analysis (PCA)
- Singular-value decomposition (SVD)
- Eigenvalues and Eigenvectors

Discretização dos dados e transformação.

Exemplo de sinal contínuo para discreto



Fonte da imagem: https://www.globalspec.com/learnmore/data_acquisition_signal_conditioning/signal_converting/analog_to_digital_converters

Exemplo lista de presença (**transformação**)

Dada o seguinte conjunto:

$\boldsymbol{X} =$

A	B	C	E	F
A	B	C	E	F
G	A	D	H	B
G	A	D	H	B
I	J	A	D	F

Você precisa ordenar X de modo que cada item distinto do conjunto seja uma coluna, e para cada ocorrência do item na linha, o número 1 apareça.

Exemplo lista de presença

...Assim

$X'_{0,*} = \{A,\ B,\ C,\ D,\ E,\ F,\ G,\ H,\ I,\ J\}$

$X'_{0,*}$ seria o cabeçalho, somente os nomes para a descrição dos dados. Para a linha 1, teríamos a marcação para os itens presentes, logo:

$X'_{1,*} = \{1, 1, 1, 0, 1, 1, 0, 0, 0, 0\}$

As próximas linhas seguem a mesma lógica até que todas as linhas estejam preenchidas com presença (1) ou ausência do item (0).

$$\boldsymbol{X} = \begin{matrix} A & B & C & E & F \\ A & B & C & E & F \\ G & A & D & H & B \\ G & A & D & H & B \\ I & J & A & D & F \end{matrix}$$

Technical help

A primeira tarefa a ser feita é coletar os itens únicos do conjunto. Em R, podemos fazer isso com `unique()`

```
> unique(x)
  A B C E F
1 A B C E F
2 G A D H B
4 I J A D F
```

Porém, isso nos retorna os itens únicos por coluna. Para obter todos, podemos transformar a matriz (data frame) em uma lista (ou tirar da lista do DF ?!) com `unlist()`

```
> unique(unlist(x))
 [1] A G I B J C D E H F
Levels: A G I B J C D E H F
```

Technical help

- A saída é um objeto `factor` na ordem de coleta das colunas.
- Objetos deste tipo tem a propriedade `levels` que guarda os itens diferentes. Logo podemos extrair e ordenar os `'níveis'` do `'fator'`:

```
> sort(levels(unique(unlist(x))))
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

- Temos o header, agora é só fazer uma varredura e verificar se x contem seus itens.

Technical help

- Comandos úteis:
- Use `var ← data.frame()` para criar um *data frame* vazio.
- Use `length(vetor)`, `nrow` e `ncol` para obter o tamanho de um vetor e a quantidade de linhas e colunas de um *data frame*, respectivamente.
- O laço for, no R, tem o seguinte escopo

```
for (var in ini:fim)
```

Hands on!

1. Transforme ***Z*** de modo que cada item diferente seja um atributo (coluna própria). Então para cada linha o atributo recebe “SIM” caso enteja presente e “NÃO” caso esteja ausente.

Z =

A	B	C	E	F	A
A	B	C	E	F	Z
G	A	D	H	B	W
G	A	D	H	B	K
K	A	A	D	F	Z

*Se você preferir, ***Z*** está disponível para download em .csv