数据预处理

2021/9/28

目录

1 数据清洗

2 数据集成

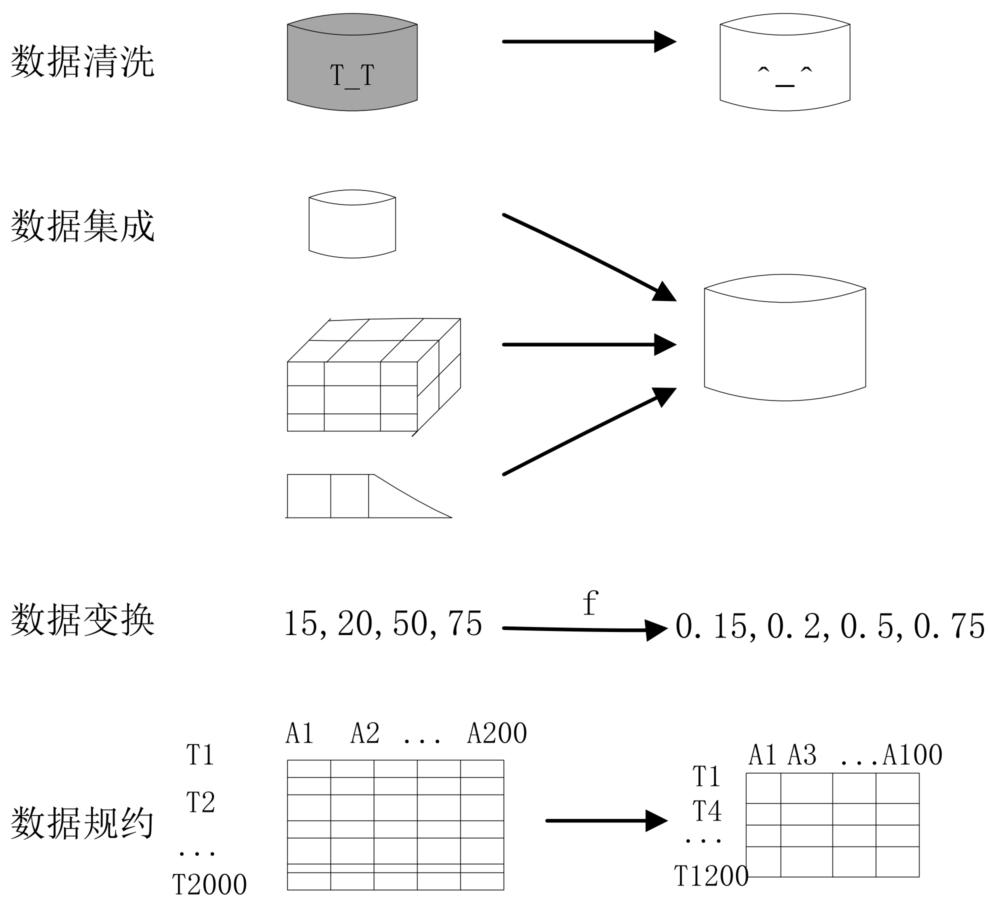
3 数据变换

4 数据规约

5 Python主要数据预处理函数

数据预处理

- 在数据挖掘的过程中，数据预处理占到了整个过程的60%。
- 数据预处理的主要任务包括数据清洗，数据集成，数据变换和数据规约。处理过程如图所示：



数据清洗

- 数据清洗主要是删除原始数据集中的无关数据、重复数据，平滑噪声数据，处理缺失值、异常值等。

缺失值处理

- 处理缺失值的方法可分为三类：删除记录、数据插补和不处理。其中常用的数据插补方法见下表。

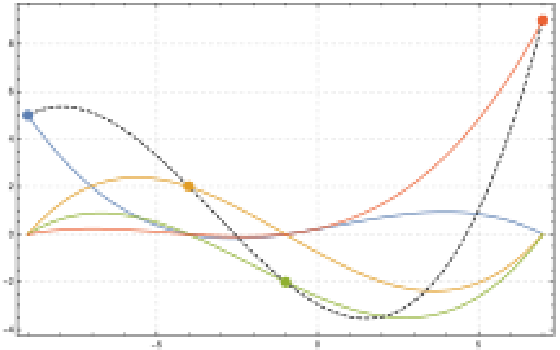
插补方法	方法描述
均值/中位数/众数插补	根据属性值的类型，用该属性取值的平均数/中位数/众数进行插补
使用固定值	将缺失的属性值用一个常量替换。如广州一个工厂普通外来务工人员的"基本工资"属性的空缺值可以用 2015 年广州市普通外来务工人员工资标准 1895 元/月，该方法就是使用固定值
最近临插补	在记录中找到与缺失样本最接近的样本的该属性值插补
回归方法	对带有缺失值的变量，根据已有数据和与其有关的其他变量（因变量）的数据建立拟合模型来预测缺失的属性值
插值法	插值法是利用已知点建立合适的插值函数 $f(x)$，未知值由对应点 x_i 求出的函数值 $f(x_i)$ 近似代替

缺失值处理

- 插值方法最主要的有拉格朗日插值法和牛顿插值法。以下便对这两种进行介绍。
- 拉格朗日插值法

第一步：

求已知的n个点对 $(x_1,y_1),(x_2,y_2)\cdots(x_n,y_n)$ 的基函数



$$l_i(x)=\prod_{j=1,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$$

在x_i处值为1，在其他的点上取值为0

第二步：

求已知的n个点对 $(x_1,y_1),(x_2,y_2)\cdots(x_n,y_n)$ 的插值多项式

$$L(x)=\sum_{i=1}^{n}y_i\,l_i(x)=\sum_{i=1}^{n}y_i\prod_{j=1,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$$

第三步：

将缺失的函数值对应的点 代入插值多项式得到缺失值的近似值 $L(x)$

缺点：当插值点增加或减少一个时，所对应的基本多项式就需要全部重新计算，非常繁琐。

缺失值处理

- 牛顿插值法

第一步:

求已知的n个点对 $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ 的所有阶差商公式

$$f[x_1, x] = \frac{f[x] - f[x_1]}{x - x_1} = \frac{f(x) - f(x_1)}{x - x_1} \quad ; \quad f[x_2, x_1, x] = \frac{f[x_1, x] - f[x_2, x_1]}{x - x_2} \quad ;$$

$$f[x_3, x_2, x_1, x] = \frac{f[x_2, x_1, x] - f[\mathrm{x}_3, x_2, x_1]}{x - x_3}$$

$$\cdots\cdots$$

第二步：

联立以上差商公式建立如下插值多项式 $f(x)$

$$\begin{aligned} f(x) = f(x_1) &+ (x - x_1) f[x_2, x_1] + (x - x_1)(x - x_2) f[x_3, x_2, x_1] + \\ &(x - x_1)(x - x_2)(x - x_3) f[x_4, x_3, x_2, x_1] + \cdots + \\ &(x - x_1)(x - x_2) \cdots (x - x_{n-1}) f[x_n, x_{n-1}, \cdots, x_2, x_1] + \\ &(x - x_1)(x - x_2) \cdots (x - x_n) f[x_n, x_{n-1}, \cdots, x_1, x] \end{aligned}$$

第三步：将缺失的函数值对应的点 代入插值多项式得到缺失值的近似值 $f(x)$

缺失值处理——实例

- 餐饮系统中的销量数据可能出现缺失值，下表为某餐厅一段时间的销量表，其中有一天的数据缺失，用拉格朗日插值与牛顿插值法对缺失值补缺。

时间	2015/2/25	2015/2/24	2015/2/23	2015/2/22	2015/2/21	2015/2/20
销售额（元）	3442.1	3393.1	3136.6	3744.1	6607.4	4060.3
时间	2015/2/19	2015/2/18	2015/2/16	2015/2/15	2015/2/14	2015/2/13
销售额（元）	3614.7	3295.5	2332.1	2699.3	空值	3036.8

代码

```python
#拉格朗日插值代码
import pandas as pd #导入数据分析库Pandas
from scipy.interpolate import lagrange #导入拉格朗日插值函数

inputfile = '../data/catering_sale.xls' #销量数据路径
outputfile = '../tmp/sales.xls' #输出数据路径

data = pd.read_excel(inputfile) #读入数据
data[u'销量'][(data[u'销量'] < 400) | (data[u'销量'] > 5000)] = None #过滤异常值，将其变为空值

#自定义列向量插值函数
#s为列向量，n为被插值的位置，k为取前后的数据个数，默认为5
def ployinterp_column(s, n, k=5):
  y = s[list(range(n-k, n)) + list(range(n+1, n+1+k))] #取数
  y = y[y.notnull()] #剔除空值
  return lagrange(y.index, list(y))(n) #插值并返回插值结果

#逐个元素判断是否需要插值
for i in data.columns:
  for j in range(len(data)):
    if (data[i].isnull())[j]: #如果为空即插值。
      data[i][j] = ployinterp_column(data[i], j)

data.to_excel(outputfile) #输出结果，写入文件
```

返回插值函数

插值结果

2015/2/20	4060.3
2015/2/19	3614.7
2015/2/18	3295.5
2015/2/16	2332.1
2015/2/15	2699.3
2015/2/14	
2015/2/13	3036.8
2015/2/12	865
2015/2/11	3014.3
2015/2/10	2742.8
2015/2/9	2173.5

原始数据

2015-02-20 00:00:00	4060.3
2015-02-19 00:00:00	3614.7
2015-02-18 00:00:00	3295.5
2015-02-16 00:00:00	2332.1
2015-02-15 00:00:00	2699.3
2015-02-14 00:00:00	4156.860423
2015-02-13 00:00:00	3036.8
2015-02-12 00:00:00	865
2015-02-11 00:00:00	3014.3
2015-02-10 00:00:00	2742.8
2015-02-09 00:00:00	2173.5

插值结果

异常值处理

- 在数据预处理时，异常值是否剔除，需视具体情况而定，因为有些异常值可能蕴含着有用的信息。异常值处理常用方法见下表：

异常值处理方法	方法描述
删除含有异常值的记录	直接将含有异常值的记录删除。
视为缺失值	将异常值视为缺失值，利用缺失值处理的方法进行处理。
平均值修正	可用前后两个观测值的平均值修正该异常值。
不处理	直接在具有异常值的数据集上进行挖掘建模。

目录

1 数据清洗

2 数据集成

3 数据变换

4 数据规约

5 Python主要数据预处理函数

数据集成

- 数据挖掘需要的数据往往分布在不同的数据源中，数据集成就是将多个数据源合并存放在一个一致的数据存储（如数据仓库）中的过程。
- 在数据集成时，来自多个数据源的现实世界实体的表达形式是不一样的，不一定是匹配的，要考虑实体识别问题和属性冗余问题，从而把源数据在最低层上加以转换、提炼和集成。

数据集成——实体识别

- 实体识别的任务是检测和解决同名异义、异名同义、单位不统一的冲突。如：
- 同名异义：数据源A中的属性ID和数据源B中的属性ID分别描述的是菜品编号和订单编号，即描述的是不同的实体。
- 异名同义：数据源A中的sales_dt和数据源B中的sales_date都是是描述销售日期的，即A. sales_dt= B. sales_date。
- 单位不统一：描述同一个实体分别用的是国际单位和中国传统的计量单位。

数据集成——冗余属性识别

- 数据集成往往导致数据冗余，如：
  - 同一属性多次出现
  - 同一属性命名不一致导致重复
- 不同源数据的仔细整合能减少甚至避免数据冗余与不一致，以提高数据挖掘的速度和质量。对于冗余属性要先分析检测到后再将其删除。
- 有些冗余属性可以用相关分析检测到。给定两个数值型的属性A和B，根据其属性值，可以用相关系数度量一个属性在多大程度上蕴含另一个属性。

目录

1 数据清洗

2 数据集成

3 数据变换

4 数据规约

5 Python主要数据预处理函数

数据变换

- 主要是对数据进行规范化的操作，将数据转换成“适当的”格式，以适用于挖掘任务及算法的需要。

数据变换——简单函数变换

- 简单函数变换就是对原始数据进行某些数学函数变换，常用的函数变换包括平方、开方、对数、差分运算等，即:

$$x' = x^2$$

$$x' = \sqrt{x}$$

$$x' = \log(x)$$

$$\nabla f(x_k) = f(x_{k+1}) - f(x_k)$$

数据变换——规范化

- 数据标准化（归一化）处理是数据挖掘的一项基础工作，不同评价指标往往具有不同的量纲和量纲单位，数值间的差别可能很大，不进行处理可能会影响到数据分析的结果，为了消除指标之间的量纲和大小不一的影响，需要进行数据标准化处理，将数据按照比例进行缩放，使之落入一个特定的区域，从而进行综合分析。如将工资收入属性值映射到[-1，1]或者[0，1]之间。
- 下面介绍三种规范化方法：最小-最大规范化、零-均值规范化、小数定标规范化

数据变换——规范化

- 最小-最大规范化：也称为离差标准化，是对原始数据的线性变换，使结果值映射到[0,1]之间。
  转换函数如：
$$x^* = \frac{x - min}{max - min}$$
  其中max 为样本数据的最大值 ,min 为样本数据的最小值。

- 零-均值规范化:也叫标准差标准化，经过处理的数据的平均数为0，标准差为1。转化函数为：
$$x^* = \frac{x - \bar{x}}{\sigma}$$
  其中 $\bar{x}$ 为原始数据的均值，σ 为原始数据的标准差。

- 小数定标规范化:通过移动属性值的小数位数，将属性值映射到[-1，1]之间，移动的小数位数取决于属性值绝对值的最大值。转化函数为：
$$x^* = \frac{x}{10^k}$$

数据变换——连续属性离散化

- 一些数据挖掘算法，特别是某些分类算法，要求数据是分类属性形式，如ID3算法、Apriori算法等。这样，常常需要将连续属性变换成分类属性，即连续属性离散化。

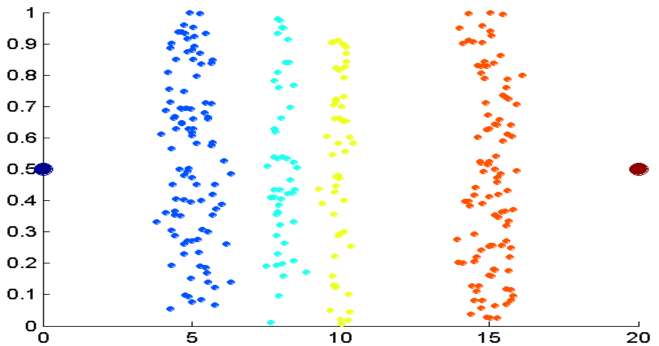
- ➢ 离散化的过程

连续属性变换成分类属性涉及两个子任务：决定需要多少个分类变量，以及确定如何将连续属性值映射到这些分类值。

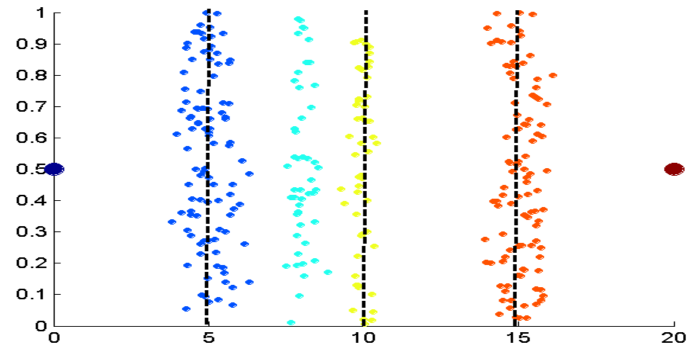
- ➢ 常用的离散化方法

常用的无监督（不使用标签）离散化方法有：等宽法、等频法、基于聚类分析的方法

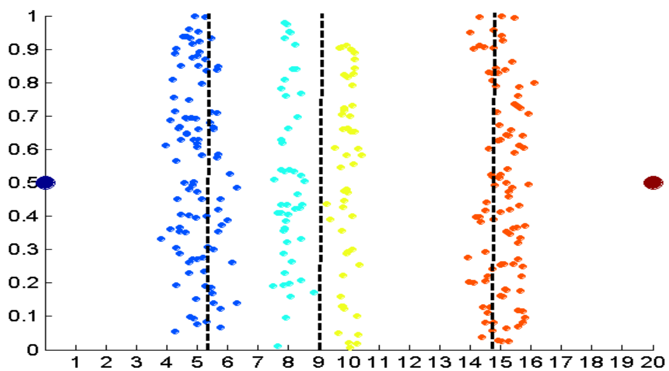
不使用类标签进行离散化



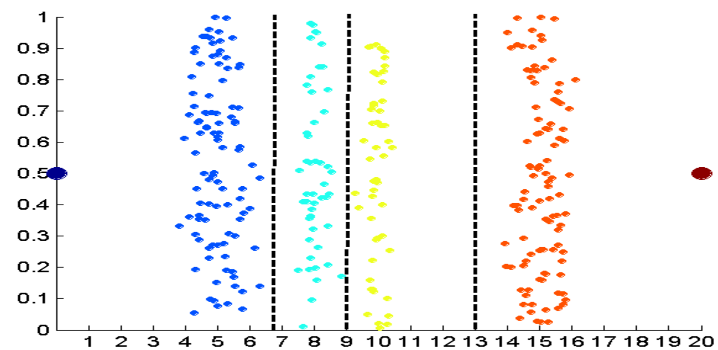
Data



Equal interval width



Equal frequency



K-means

数据变换——属性构造

- 在数据挖掘的过程中，为了帮助提取更有用的信息、挖掘更深层次的模式，提高挖掘结果的精度，需要利用已有的属性集构造出新的属性，并加入到现有的属性集合中。

- 比如进行防窃漏电诊断建模时，已有的属性包括进入线路供入电量、该条线路上各大用户用电量之和，记为供出电量。理论上供入电量和供出电量应该是相等的，但是由于在传输过程中的电能损耗，会使得供入电量略大于供出电量，如果该条线路上的一个或多个大用户存在窃漏电行为，会使供入电量远大于供出电量。反过来，为了判断是否存在有窃漏电行为的大用户，需要构造一个新的关键指标--线损率，该过程就是构造属性，由线户关系图（见图6-1）。新构造的属性线损率计算公式如下：

  线损率 =（供入电量-供出电量）/供入电量

- 线损率的范围一般在3%~15%，如果远远超过该范围，就可以认为该条线路的大用户很大可能存在窃漏电等用电异常行为。

数据变换——小波变换

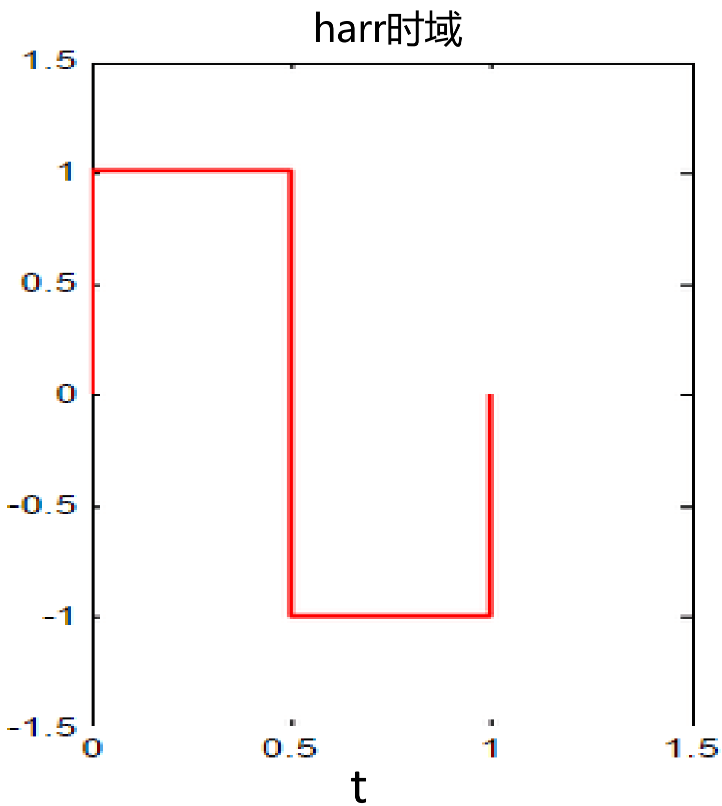
基于小波变换的特征提取方法及其方法描述如下表所示：

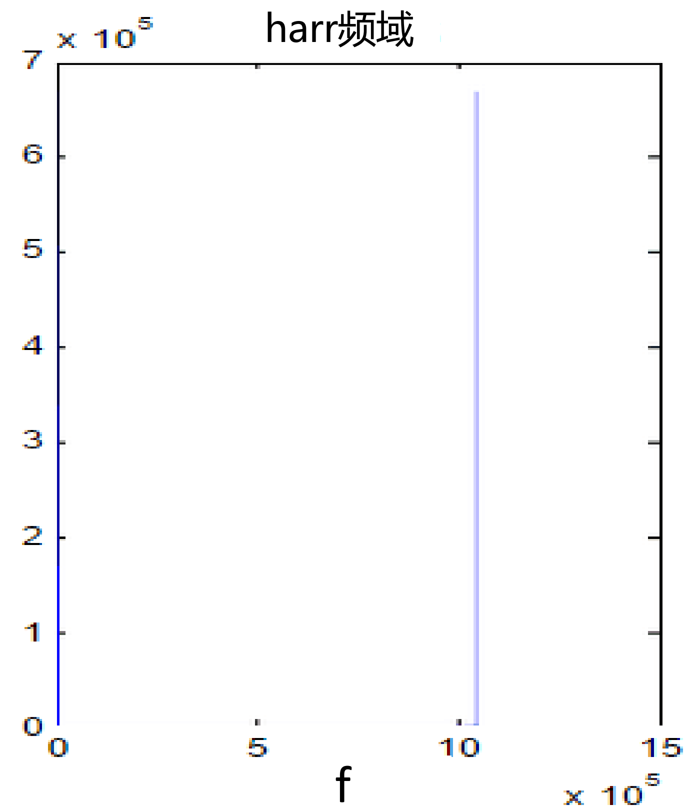
基于小波变换的特征提取方法	方法描述
基于小波变换的多尺度空间能量分布特征提取方法	各尺度空间内的平滑信号和细节信号能提供原始信号的时频局域信息，特别是能提供不同频段上信号的构成信息。把不同分解尺度上信号的能量求解出来，就可以将这些能量尺度顺序排列形成特征向量供识别用。
基于小波变换的多尺度空间中模极大值特征提取方法	利用小波变换的信号局域化分析能力，求解小波变换的模极大值特性来检测信号的局部奇异性，将小波变换模极大值的尺度参数 s、平移参数 t 及其幅值作为目标的特征量。
基于小波包变换的特征提取方法	利用小波分解，可将时域随机信号序列映射为尺度域各子空间内的随机系数序列，按小波包分解得到的最佳子空间内随机系数序列的不确定性程度最低，将最佳子空间的熵值及最佳子空间在完整二叉树中的位置参数作为特征量，可以用于目标识别。
基于适应性小波神经网络的特征提取方法	基于适应性小波神经网络的特征提取方法可以把信号通过分析小波拟合表示，进行特征提取。

数据变换——小波变换

小波基函数是一种具有局部支集的函数，平均值为0，小波基函数满足：

$\psi(0)=\int\psi(t)dt=0$。Haar小波基函数是常用的小波基函数，如下图所示：





数据变换——小波变换

- 小波基函数伸缩和平移变换模型为：

$$\psi_{a,b}(t)=\frac{1}{\sqrt{|a|}}\psi(\frac{t-b}{a})$$

其中，a 为伸缩因子，b 为平移因子。

- 任意函数 $f(t)$ 的连续小波变换（CWT）为：

$$W_f(a,b)=|a|^{-1/2}\int f(t)\psi(\frac{t-b}{a})\,dt$$

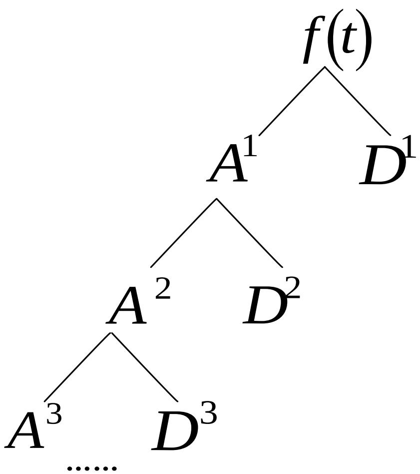
- 上式的逆变换为：

$$f(t)=\frac{1}{C_\psi}\int\int\frac{1}{a^2}W_f(a,b)\psi(\frac{t-b}{a})da\cdot db$$

数据变换——小波变换

- 基于小波变换的多尺度空间能量分布特征提取方法：
- 第一步：对 $f(t)$ 进行二进小波分解： $f(t) = A^j + \sum D^j$

其中 A 是近似信号，为低频部分； B 是细节信号，为高频部分，此时信号的频带分布图如左下图所示：



- 第二步：计算出信号能量为：

$$E = EA_j + \sum ED_j$$

- 第三步：选择第 j 层的近似信号和各层的细节信号的能量作为特征，构造特征向量：

$$F = [EA_j, ED_1, ED_2, \cdots, ED_j]$$

目录

1 数据清洗

2 数据集成

3 数据变换

4 数据规约

5 Python主要数据预处理函数

数据规约

- 数据规约是将海量数据进行规约，规约之后的数据仍接近于保持原数据的完整性，但数据量小得多。
- 通过数据规约，可以达到：
  - 降低无效、错误数据对建模的影响，提高建模的准确性
  - 少量且具代表性的数据将大幅缩减数据挖掘所需的时间
  - 降低储存数据的成本

数据规约——属性规约

- 属性规约常用方法有：合并属性、逐步向前选择、逐步向后删除、决策树归纳、主成分分析

- 合并属性

  初始属性集：$\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4, B_1, B_2, B_3, \mathrm{C}\}$

  $\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4\} \rightarrow \mathrm{A};$
  $\{B_1, B_2, B_3\} \rightarrow \mathrm{B}.$

  $\Rightarrow$ 规约后属性集：$\{A, B, C\}$

- 逐步向前选择

  初始属性集：$\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4, \mathrm{A}_5, \mathrm{A}_6\}$

  $\{\} \Rightarrow \{\mathrm{A}_1\} \Rightarrow \{\mathrm{A}_1, \mathrm{A}_4\}$

  $\Rightarrow$ 规约后属性集：$\{\mathrm{A}_1, \mathrm{A}_4, \mathrm{A}_6\}$

数据规约——属性规约

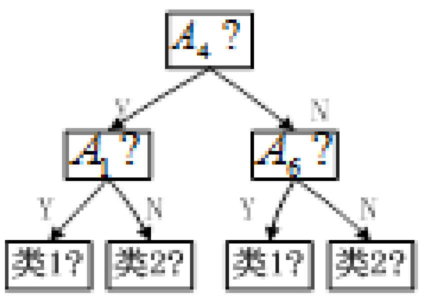
- 逐步向后删除
  初始属性集：$\{A_1, A_2, A_3, A_4, A_5, A_6\}$

  $\Rightarrow \{A_1, A_3, A_4, A_5, A_6\}$

  $\Rightarrow \{A_1, A_4, A_5, A_6\}$

  $\Rightarrow$ 规约后属性集：$\{A_1, A_4, A_6\}$

- 决策树规约
  初始属性集：$\{A_1, A_2, A_3, A_4, A_5, A_6\}$



  $\Rightarrow$ 规约后属性集：$\{A_1, A_4, A_6\}$

数据规约——属性规约

下面详细介绍主成分分析计算步骤：

1）设原始变量 $X_1, X_2, \dots, \mathrm{X}_p$ 的观测n次数据矩阵为：

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \triangleq (X_1, X_2, \dots, X_P)$$

2）将数据矩阵中心标准化。为了方便，将标准化后的数据矩阵仍然记为 X 。

3）求相关系数矩阵 $R = (r_{ij})_{p\times p}$ 的定义为：

$$r_{ij} = \sum_{k=1}^{n}(x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j) \Big/ \sqrt{\sum_{k=1}^{n}(x_{ki} - \bar{x}_i)^2 \sum_{k=1}^{n}(x_{kj} - \bar{x}_j)^2}$$ 其中 $r_{ij} = r_{ji}, r_{ii} = 1$

4）求 R 的特征方程 $\det(R - \lambda E) = 0$ 的特征根 $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_p > 0$ 。

5）确定主成分个数m :$\sum_{i=1}^{m} \lambda_i / \sum_{i=1}^{p} \lambda_i \ge \alpha$ ，α 根据实际问题确定，一般取80%。

数据规约——属性规约

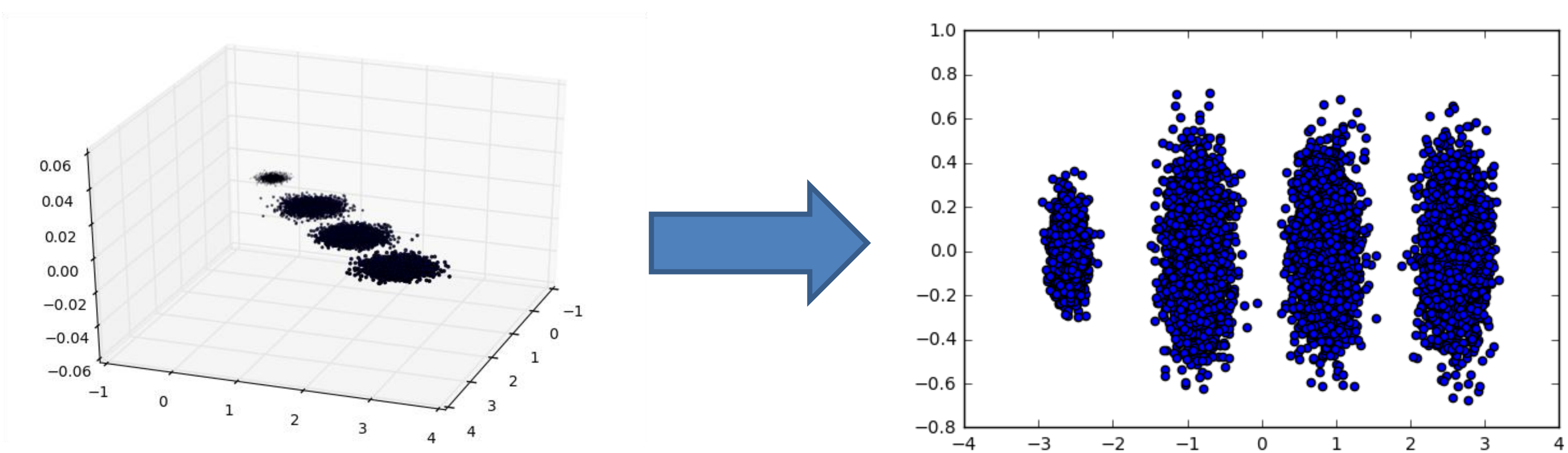
6）计算m个相应的单位特征向量：

$$\beta_1 = \begin{bmatrix} \beta_{11} \\ \beta_{21} \\ \vdots \\ \beta_{p1} \end{bmatrix}, \beta_2 = \begin{bmatrix} \beta_{12} \\ \beta_{22} \\ \vdots \\ \beta_{p2} \end{bmatrix}, \dots, \beta_m = \begin{bmatrix} \beta_{1m} \\ \beta_{2m} \\ \vdots \\ \beta_{pm} \end{bmatrix}$$

7）计算主成分：

$$Z_i = \beta_{1i}X_1 + \beta_{2i}X_2 + \cdots + \beta_{pi}X_p \qquad i = 1,2,\dots,m$$

sklearn.decomposition.PCA



数据规约——数值规约

- 数值规约通过选择替代的、较小的数据来减少数据量。数值规约可以是有参的，也可以是无参的。有参方法是使用一个模型来评估数据，只需存放参数，而不需要存放实际数据。有参的数值规约技术主要有两种：回归（线性回归和多元回归）和对数线性模型（近似离散属性集中的多维概率分布）。数值规约常用方法有:
  - 直方图
  - 用聚类数据表示实际数据
  - 抽样（采样）
  - 参数回归法。

目录

1 数据清洗
2 数据集成
3 数据变换
4 数据规约
5 Python主要数据预处理函数

Python主要数据处理函数

- Python中的插值、数据归一化、主成分分析等与数据预处理相关的函数。

函数名	函数功能	所属扩展库
interpolate	一维、高维数据插值	Scipy
unique	去除数据中的重复元素，得到单值元素列表，它是对象的方法名。	Pandas/Numpy
isnull	判断是否空值	Pandas
notnull	判断是否非空值	Pandas
PCA	对指标变量矩阵进行主成分分析	Scikit-Learn
random	生成随机矩阵	Numpy

Python主要数据处理函数

- **interpolate**

功能：interpolate是scipy的一个子库，下面包含了大量的插值函数，如拉格朗日插值、样条插值、高维插值等。使用之前需要用from scipy.interpolate import *引入相应的插值函数，读者应该根据需要到官网查找对应的函数名。
使用格式：
f = scipy.interpolate.lagrange(x, y)　这里仅仅展示了一维数据的拉格朗日插值的命令，其中x, y为对应的自变量和因变量数据。插值完成后，可以通过f(a)计算新的插值结果。类似的还有样条插值、多维数据插值等。

- **unique**

功能：去除数据中的重复元素，得到单值元素列表。它既是numpy库（赋别名np）的一个函数（np.unique()），也是Series对象的一个方法。
使用格式：
np.unique(D)　D是一维数据，可以是list、array、Series；
D.unique()　D是Pandas的Series对象。

Python主要数据处理函数——实例

- 实例：求向量A中的单值元素，并返回相关索引。

```
>>> D = pd.Series([1, 1, 2, 3, 5])
>>> D.unique()
array([1, 2, 3, 5], dtype=int64)
>>> np.unique(D)
array([1, 2, 3, 5], dtype=int64)
```

Python主要数据处理函数

- **isnull/ notnull()**

功能：判断每个元素是否空值/非空值。
使用格式：
D.isnull()/ D.notnull()　这里的D要求是Series对象，返回一个布尔Series。可以通过D[D.isnull()]或D[D.notnull()]找出D中的空值/非空值。

- **random**

功能：random是Numpy （赋别名np）的一个子库（Python本身也自带了random，但Numpy的更加强大），可以用该库下的各种函数生成服从特定分布的随机矩阵，抽样时可使用。
使用格式：
np.random.rand(k, m, n, ...)　生成一个k×m×n×...随机矩阵，其元素均匀分布在区间（0,1）上。
np.random.randn(k, m, n, ...)　生成一个k×m×n×...随机矩阵，其元素服从标准正态分布。

Python主要数据处理函数

- **PCA**

功能：对指标变量矩阵进行主成分分析。使用前需要用from sklearn.decomposition import PCA引入该函数
使用格式：
model = PCA()　注意，Scikit-Learn下的PCA是一个建模式的对象，也就是说一般的流程是建模，然后是训练model.fit(D)，D为要进行主成分分析的数据矩阵，训练结束后获取模型的参数，如.components_获取特征向量，以及.explained_variance_ratio_获取各个属性的贡献率等。

Python主要数据处理函数——实例

- 实例：使用PCA()对一个10×4维的随机矩阵进行主成分分析。

```
>>>from sklearn.decomposition import PCA
>>>D = np.random.rand(10,4)
>>>pca = PCA()
>>>pca.fit(D)
PCA(copy=True, n_components=None, whiten=False)
>>>pca.components_ #返回模型的各个特征向量
array([[-0.42899319, -0.69804397,  0.32876844, -0.46969221],
       [ 0.03680965, -0.0667248 ,  0.7848853 ,  0.61493733],
       [-0.62222716,  0.68499407,  0.28400153, -0.25091755],
       [-0.65379144, -0.19765007, -0.4418252 ,  0.58161989]])
>>>pca.explained_variance_ratio_ #返回各个成分各自的方差百分比
array([ 0.40836652,  0.32861061,  0.21894296,  0.0440799 ])
```

Thank You!