文章编号：2096-1618(2021)01-0032-09

基于归一化 KNNI 的随机森林填补算法

游 凤， 李代伟， 张海清， 汪 杰， 彭 莉， 王 震
(成都信息工程大学软件工程学院，四川 成都 610225)

摘要：随机森林填补算法在对不完备信息系统填补时具有可靠的填补性能，同时由于填补时需要多次进行随机森林建模导致算法计算量大。为了缩短算法的运行时间，提出了 NKNNI-RFI(normalization k nearest neighbor imputation-random forest imputation)缺失数据填补算法。通过改变 RFI 算法中预填补，即使用填补更为准确的归一化 KNNI(normalization k nearest neighbor imputation，NKNNI)作为预填补，为 RFI 算法中使用随机森林模型预测填补值提供了更接近于原始数据集的数据，使 RFI 算法能够在更短的时间内完成填补任务且保持良好的填补效果。实验中使用 10 个 UCI 标准数据集，将提出的算法与 RFI、NKNNI、SVMI 和 ROUSTIDA 算法进行比较并使用 NRMSE、PFC 和 ART 填补评价方法对算法效果进行评价。实验结果表明：提出算法的 NRMSE 和 PFC 与 RFI 算法相同，NRMSE 比 NKNNI、SVMI 和 ROUSTIDA 算法约低0.02 ~ 0.8，PFC 比 NKNNI、SVMI 和 ROUSTIDA 算法约低0.01 ~ 0.6，ART 相比 RFI 算法最大减少程度达 53% 。

关 键 词：不完备信息系统；缺失数据填补；NKNNI；随机森林填补；填补评价方法
中图分类号：TP301.6 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2021.01.006

0 引言

在数据挖掘领域，大多数研究都把方法和模型建立在完备信息系统(complete information system)上，研究中假设方法和模型使用的信息系统均没有考虑数据缺失。然而，现实中使用的信息系统，因为某些数据暂时无法获取或获取过程中出现错误，导致数据遗漏或缺失的情况经常发生，这种信息系统称为不完备信息系统(incomplete information system)。现实中常见的不完备信息系统，如在数据挖掘领域常用的 UCI 数据集中，超过 40% 的数据集包含缺失数据[1]。然而，不完备信息系统可能携带数据的重要信息，若直接使用这些不完备信息系统进行数据挖掘，其结果会对决策产生一定的影响，即导致不可靠的输出[2]。为解决这种问题，就需要借助相关科学工具或方法对不完备信息系统中的缺失数据进行预处理，预处理一般包括删除整条缺失对象或寻找替代值。删除缺失数据固然方便简单，但是会附带删除数据中隐藏的原始信息。这时候，最合适的方法是寻找替代值，即缺失数据填补值。缺失数据填补的基本原理是利用一些现成的规则或者合理的方法在不完备信息系统中推测出填补值[3]。这样既可以保持原数据集的维度，又可以使之相关的信息不被忽略。

近年来研究表明，缺失数据填补技术广泛应用于工业、经济、医学、统计学等诸多领域[4-5]。在统计学领域对数据填补有着更为广泛的研究，如众数或均值填补、最大期望法[6]填补(expectation maximization imputation，EMI)、回归法填补等；在数据挖掘和机器学习领域的很多算法也应用到了缺失数据填补中，如 Stekhoven 等[7]提出的随机森林填补算法(random forest Imputation，RFI)，算法在第一步使用完整的数据训练出一个随机森林模型，然后用来预测缺失值，最后进行重复迭代来处理缺失值问题。其算法效果主要取决于随机森林中树的大小 n_{tree} 和每个节点中随机选取的属性的数目 m_{try}，一般 $n_{tree}=100$、$m_{try}=\sqrt{p}$(p 为属性列数)时算法就有良好的填补效果。Dixon[8]提出了 k 近邻填补算法(k nearest neighbor imputation，KNNI)，该算法首先在数据集中找到与缺失数据样本 k 个最相似的对象，然后利用这 k 个对象相应的属性值对缺失值进行填补。算法的填补效果主要取决于 k 的取值可通过变化 k 的取值来确定最优的 k 值。Wang 等[9]提出了支持向量机填补算法(support vector machine imputation，SVMI)，SVMI 使用支持向量回归填补含有缺失信息的基因表达数据，通过正交化编码输入，将缺失数据映射到更高维空间中，从而填补数据。SVMI 的填补效果主要由 SVM 的核函数决定，SVM 的核函数有线性

收稿日期：2020-09-02
基金项目：国家自然科学基金资助项目(61602064)；四川省科技厅资助项目(2018JY0273、2019YFG0398)；欧盟资助项目(598649-EPP-1-2018-1-FR-EPPKA2-CBHE-JP)

核、径向基内核(radial basis function,RBF),对于一般数据集,使用 RBF 就有理想的分类效果了。Weihua 等[10]提出了基于粗糙集理论的不完备数据分析方法(rough set theory based incomplete data analysis approach,ROUSTIDA),采用粗糙集理论实现不完备信息系统的缺失数据填补,算法主要思想是在填补时应使具有缺失数据的对象与信息系统的其他相似对象的属性值的差异尽可能保持一致,填补前后属性值之间的差异尽可能小。ROUSTIDA 无需设置参数,不用使用大量时间进行调参。

在现有的缺失数据填补方法中,RFI 算法已成为学者们经常使用的对比填补算法,有研究结果[11-12]表明其往往能在混合类型数据集上取得良好的填补效果,同时也比其他填补算法(如 KNNI,EMI)计算量大。在数据量不断增长,数据实时性要求越来越高的背景下,这已经难以满足实际缺失数据填补应用的需求。针对该问题,使用归一化 KNN 填补算法[13](normalization k nearest neighbor imputation,NKNNI)结合随机森林填补算法,提出了 NKNNI-RFI(normalization k nearest neighbor imputation-radom forest imputation)缺失数据填补算法。该方法将 RFI 算法的预填补方式由传统的均值或众数填补改为 NKNNI 后,缩短了 RFI 算法的运行时间。实验结果表明,在实验中所使用的 10 个数据集上,NKNNI-RFI 在保持良好的填补效果的同时运行效率总体上优于 RFI 算法。

1　相关工作

1.1　NKNNI 算法

KNN 算法的核心思想为假设一个对象的 k 个最相邻对象中的大多数属于某一个类别,则该对象也属于这个类别[14]。因为简单易理解的算法原理而广受学者的青睐,继而有了很多相应的改进算法出现。如 Wenfeng Hou 等[15]将 KD-tree 引入 KNN 中提高搜索效率;Chao Xu 等[16]将粗糙模糊理论与 KNN 算法相结合来提高算法的分类准确率。Dixon 等[13]将 KNN 算法应用在数据填补中提出了对距离计算进行归一化(Normalization)处理的 NKNNI,它可以计算两个均含缺失值的对象之间的距离,并使用归一化来减少缺失值的存在对计算距离的影响,再为含有缺失值的对象寻找 k 个距离最近的近邻对象(要求近邻对象在相应的属性上无缺失值)进行填补。在数据集有较高缺失率(20% ~30%)时,NKNNI 仍然能够完成填补任务,并且有较好的填补效果。这种方法相比均值或众数填补效果更好,同时从文中的实验结果可以看出其计算量比 SVMI、ROUSTIDA 等填补算法要小[13]。

1.2　RFI 算法

随机森林算法最早是由 Leo[17]提出,是一种集成学习方法。其基本思想是:一个森林里包含多棵决策树,每棵决策树由采用有放回的随机抽样生成,让每棵树充分生长,不对其进行剪枝处理,最终的输出结果为对所有决策树进行多数投票的结果[18]。算法的优势在于对于无论是类别型还是连续性的数据,其分类效果要优于大多数算法,并且能够有效处理高维数据集。除此之外,算法对参数设置并不敏感,可以较为容易地找到一个合适的随机森林模型。

由于随机森林算法具有较高的分类准确性,也被国外学者 Stekhoven 与 Buhlmann 应用到缺失数据填补中,提出了 RFI 算法[7]。随机森林算法要求决策属性是完整的,才能训练出森林,RFI 算法在此基础上改进,通过增加预填补,使用预填补后的完整数据集训练随机森林模型来预测缺失值,然后进行多次迭代来处理缺失数据问题,而不依赖于决策属性的完整。同时,RFI 算法沿袭随机森林算法的高分类准确性而有着良好的填补效果,但其不足之处在于算法运行中会重复迭代使用随机森林建模,计算量大。因此,RFI 算法填补运行时间长。

2　NKNNI-RFI 算法

考虑 RFI 算法是通过迭代随机森林建模的方式向原始数据集逼近,其使用均值或众数填补作为预填补需多次迭代完成填补,因此选用填补效果较均值或众数填补更好的 NKNNI 作为预填补方法,为随机森林模型预测填补值提供了更接近于原始数据集的数据同时 NKNNI 计算量相比 SVMIROUSTIDA 更少,不会增加 RFI 算法的填补时间,适合作为预填补方法。

2.1　NKNNI-RFI 相关定义

定义 1　距离矩阵 ***DIS***。假设有 n 行 p 列的数据集 $X=\{x_1,x_2,\cdots,x_n\}$。构建距离矩阵:由 X 中含缺失值的对象 $X_{\text{mis}}=\{y_1,y_2,\cdots,y_r\}$ 与 X 中所有对象的距离构成的矩阵。

$$DIS_{n\times r}=\begin{bmatrix} d(x_1,y_1) & \cdots & d(x_1,y_r) \\ \vdots & \vdots & \vdots \\ d(x_n,y_1) & \cdots & d(x_n,y_r) \end{bmatrix} \tag{1}$$

定义 2 对象距离[13]。$d_t(x)_i,y_j)$为 X 第 i 个对象和 X_{mis}第 j 个对象之间的第 t 个属性的距离,$d(x_i,y_j)$表示 X 第 i 个对象和 X_{mis}第 j 个对象之间的距离。公式中通过归一化来减少缺失值的存在对计算距离的影响。$p_{i,j}$为两个对象中都没有缺失值的列数。

$$d_t(x_i,y_j)=\begin{cases} 0 & \text{if } x_{i,t}\text{or } y_{j,t}\text{is missing} \\ (x_{i,t}-y_{j,t}) & \text{otherwise} \end{cases} \tag{2}$$

$$d(x_i,y_j)=\sqrt{\frac{p}{p_{i,j}}\sum_{t=1}^{p}d_t(x_i,y_j)^2} \tag{3}$$

定义 3 k 近邻对象填补。若缺失值为连续性,则填补值为其 k 个距离最小的对象相应属性值的加权均值,若为类别性,则为加权投票。

定义 4 缺失数据集[7]。假定缺失数据集 $X_{n\times p}$可以按列划分为 $X_1,X_2,\cdots,X_P$,将其中任意一个可能含有缺失值的列记为 X_s,在该属性上含有缺失值的对象记为 $i_{mis}^{(s)}\subseteq\{1,2,\cdots,n\}$。根据数据集中属性列按含有缺失值的个数进行升序排列,得到矩阵 $X_{n\times p}$的列索引向量 d。缺失数据集能按以下规则划分为 4 个部分:

(i)记 $y_{\text{obs}}^{(s)}$为 X_s 中无缺失的值;

(ii)记 $y_{\text{mis}}^{(s)}$为 X_s 中有缺失的值;

(iii)记 $x_{\text{obs}}^{(s)}$为在 X_s 上无缺失的对象 $i_{\text{obs}}^{(s)}=\{1,2,\cdots,n\}\setminus i_{\text{mis}}^{(s)}$其他属性;

(iv)记 $x_{\text{mis}}^{(s)}$为在 X_s 有缺失的对象 $i_{\text{mis}}^{(s)}$的其他属性。

定义 5 迭代数据集。NKNNI-RFI 里面区分填补前后的数据集及最后输出的数据集,每一次完整的填补称为一轮迭代。每一轮迭代前的数据集矩阵记为 $X_{\text{old}}^{\text{imp}}$,迭代后的数据集矩阵记为 $X_{\text{new}}^{\text{imp}}$,最后输出的完备数据集记为 X^{imp}。

定义 6 结束条件。要能够最后结束迭代并输出 X^{imp},必须满足以下两个条件之一作为结束条件:

(i) RFI 算法的迭代次数 *ite* 达到指定最大迭代次数 T;

(ii) $X_{\text{new}}^{\text{imp}}$与 $X_{\text{old}}^{\text{imp}}$对于连续属性集合 N(类别属性集合 F)的对象之间的误差值 $\Delta N(\Delta F)$全都首次开始增大,这里将每一轮迭代后计算的 $\Delta N(\Delta F)$记为 ΔN_{new}(ΔF_{new}),计算公式相同。ΔN 与 ΔF 的计算公式分别为

$$\Delta N=\frac{\sum_{j\in N}(X_{\text{new}}^{\text{imp}}-X_{\text{old}}^{\text{imp}})^2}{\sum_{j\in N}(X_{\text{new}}^{\text{imp}})^2} \tag{4}$$

其中,N 为连续属性列的集合,j 为属于 N 的属性列序号。

$$\Delta F=\frac{\sum_{j\in F}\sum_{i=1}^{n}I(X_{\text{new}}^{\text{imp}}\neq X_{\text{old}}^{\text{imp}})}{F_{\text{mis}}} \tag{5}$$

其中,j 为属于 F 的属性列序号,$I(-)$为指示函数,$I(X_{\text{new}}^{\text{imp}}\neq X_{\text{old}}^{\text{imp}})$中如果 $X_{\text{new}}^{\text{imp}}\neq X_{\text{old}}^{\text{imp}}$为真,则其返回 1,否则为 0,$F_{\text{mis}}$为类别属性中缺失项的总数目。

2.2 NKNNI-RFI 算法描述

将预填补后的数据集记为 $X_{\text{ini}}^{\text{imp}}$。基于上述定义,NKNNI-RFI 算法的详细描述如下:

算法 1 NKNNI-RFI 算法

输入:缺失数据集 X,最大迭代次数 T,近邻个数 k

输出:完备数据集 X^{imp}

构建 *DIS*(定义 1 和 2)

Fors in p:

找出该列无缺失值的对象 $X_{\text{nmis}}=X\setminus X_{\text{mis}}$

在 X_{nmis}中为 X_{mis}选出 k 个近邻对象

$X_{\text{ini}}^{\text{imp}}\leftarrow X_{\text{mis}}$由 k 个近邻对象填补(定义 3)

End for

$ite\leftarrow 0$

$\Delta N\leftarrow inf,\Delta N_{\text{new}}\leftarrow 0,\Delta F\leftarrow inf,\Delta F_{\text{new}}\leftarrow 0,X_{\text{new}}^{\text{imp}}\leftarrow X_{\text{ini}}^{\text{imp}}$

While($\Delta N_{\text{new}}<\Delta N$ or $\Delta F_{\text{new}}<\Delta F$) and $ite<T$:

$X_{\text{old}}^{\text{imp}}\leftarrow X_{\text{new}}^{\text{imp}}$

If $ite\neq 0$ then $\Delta N\leftarrow\Delta N_{\text{new}},\Delta F\leftarrow\Delta F_{\text{new}}$

For s in d(定义 4):

对 $y_{\text{obs}}^{(s)}\sim x_{\text{obs}}^{(s)}$执行随机森林建模

通过 $x_{\text{mis}}^{(s)}$预测 $y_{\text{mis}}^{(s)}$

将预测得到的 $y_{\text{mis}}^{(s)}$更新 $X_{\text{new}}^{\text{imp}}$

End for

计算 ΔN_{new}与 ΔF_{new}(定义 6)

$ite+1$

End while

$X^{\text{imp}}\leftarrow X_{\text{old}}^{\text{imp}}$

2.3 NKNNI-RFI 时间复杂度分析

RFI 算法中时间开销主要是训练 RF 分类器的过程。假设训练数据集特征数为 p,样本个数为 n,RF 中一棵决策树算法的时间复杂度近似于 $O(p*n*(logn)^2)$。如 RF 中树有 n_{tree}棵,则 RF 算法的时间复杂度可近似于 $O(n_{\text{tree}}*p*n*(logn)^2)$。在 RFI 算法中,由算法步骤(9)至(19)可知,完成一次 RFI 填补中需要 d 次 RF 建模并迭代进行 *ite* 次,因此,RFI 算法的

时间复杂度可近似于 $O(d * ite * n_{tree} * p * n * (logn)^2)$。而 NKNNI-RFI 算法通过为随机森林模型预测填补值提供更接近原始数据集的数据,即提供一个良好的初始解,来减少迭代次数 *ite*,使得 NKNNI-RFI 算法的时间复杂度小于 RFI 算法的时间复杂度。

3　实验

将文中提出的缺失数据填补算法 NKNNI-RFI 算法与目前较为流行的缺失数据填补算法 RFI、NKNNI、SVMI 和 ROUSTIDA 进行对比实验,这些对比算法是数据挖掘领域内熟知且应用广泛的填补算法,以此为对比可更好地看出 NKNNI-RFI 算法的性能。

3.1　实验环境

以下所有实验均使用 Windows10, Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz, 8.00 GB RAM Windows10 64 位操作系统,使用 Python3.7 进行编程。

3.2　数据集

实验中使用的数据集均选自 UCI 公开数据集[19]。数据集描述如表 1 所示。n 为对象个数,p 为属性列数,$p_{(con)}$ 为连续性属性列的列数,$p_{(cat)}$ 为类别属性列的列数。在实验中,对部分数据集进行了处理:wpdc 数据集中原始数据集中含有少量缺失值,为了后续评价填补的效果,这里将含缺失值的对象删除后再进行实验。对于原始数据集中包含字符的类别型数据的数据集,部分算法(如 NKNNI)无法处理,则将字符映射为数值后进行实验。*musk*(con)数据集是由 musk 数据集去掉类别属性列后得到的。

表 1　数据集描述

Dataset	n	p	$p_{(con)}$	$p_{(cat)}$
parkinsons	195	23	22	1
wdbc	569	31	30	1
wpbc	194	34	33	1
promoters	106	58	0	58
wine-quality-white	4898	12	11	1
kr-vs-kp	3196	37	0	37
Sonar	208	61	60	1
movement-libras	360	91	90	1
musk	476	167	166	1
musk(con)	476	166	166	0

3.3　填补评价方法

现今没有统一的对缺失数据填补效果进行评价的标准,如果是对原本就缺失的数据集进行验证,填补效果将无从评判[20]。因此,实验中通过人为对完备数据集进行一定比例(20%)的随机缺失来模拟缺失数据集。同时,对数据集中类别属性、连续属性的填补效果和算法运行时间使用相应的填补评价方法。

NRMSE 评价方法。针对连续属性集 N 采用标准化均方根误差(normalized root mean squared error, NRMSE)[21]评价指标,通过比较填补值和真实值之间的差异度来验证算法的填补效果,主要采用式(6)进行计算:

$$NRMSE=\sqrt{\frac{mean((X^{true}-X^{imp})^2)}{var(X^{true})}} \tag{6}$$

其中,X^{true} 为完备的数据集矩阵,X^{imp} 为填补后的数据集矩阵,均值 *mean* 和方差 *var* 是根据整个矩阵中缺失值计算的。

PFC 评价方法。针对离散属性集 F 采用错分占比率(the proportion of falsely classified, PFC)评价指标,通过比较所有填补错误的值的个数占缺失总个数的比例来验证算法的填补效果,主要采用式(7)进行计算:

$$PFC=\frac{\sum_{i=1}^{n} I(X^{true}\neq X^{imp})}{N_{miss}} \tag{7}$$

其中,$I(-)$为指示函数,$I(X^{true}\neq X^{imp})$中,如果 $X^{true}\neq X^{imp}$为真,则其返回 1,否则为 0,N_{miss}为离散属性中具有缺失值的项的总数。

对于这两类评价方法,值越接近于 0 则算法填补效果越好,越接近于 1 则效果越糟糕。

ART 评价方法。实验中,对填补算法的运行开始时间和结束时间分别记为 T_{start} 与 T_{end},进行 10 次实验计算其平均运行时间 ART(average running time),使用此标准来评价文中提出的算法的效率:

$$ART=\frac{\sum_{i=1}^{10}(T_{end}-T_{start})}{10} \tag{8}$$

迭代次数。实验中,RFI 算法与 NKNNI-RFI 算法会经历多次迭代随机森林模型来完成填补。一般情况下,迭代次数越多,所耗费的时间越多。针对同一个数据集,两个算法在参数不变的情况下,迭代次数也不会变,而 ART 会因为算法运行的机器不同而有所改变,在每个机器上运行的差异又不同,影响对算法的效率的评价。因此,实验中使用此标准来协同 ART 评价文

中提出的算法的效率。

3.4 实验算法参数

实验中,使用了 NKNNI、ROUSTIDA、SVMI 和 RFI 算法作为 NKNNI-RFI 算法的对比算法,各算法使用的参数如表 2 所示。由于 NKNNI-RFI 算法中 k 值不同,填补效果也不同(图 1,图 2,图 3)。从 1 ~ 60 选取在实验中大部分数据集上填补效果良好并且 ART 相对于 RFI 算法有明显减少的 $k=36$。为保证算法之间的可对比性,NKNNI 算法中 k 也取 36,SVMI 与 RFI 算法保持默认参数(表 2),ROUSTIDA 算法无参数设置。

表 2 算法参数

算法	参数取值	参数描述
NKNNI	$k=36$	近邻数目
RFI	$n_{tree}=100$	随机森林中树的数目
	$m_{try}=\sqrt{p}$	每个节点中随机选取的属性的数目
SVMI	Kernel = RBF	SVM 中使用的核函数
	$C=1.0$	误差项的惩罚参数
	Epsilon=0.001	损失函数
NKNNI-RFI	$k=36$	
	$n_{tree}=100$	
	$m_{try}=\sqrt{p}$	
	$T=15$	最大迭代次数

4 实验结果与分析

4.1 NKNNI-RFI 下 k 值不同时的变化情况

以 promoters、Sonar 和 wpdc 数据集为例,在图 1、图 2 和图 3 中展示当 k 值取 1 ~ 60,NKNNI-RFI 算法下 3 个数据集的 NRMSE、PFC、迭代次数(Iterations)和 ART 的变化。并且将 RFI 算法下这 3 个数据集的 4 种实验结果取值加入作为对比,因为 RFI 算法中无 k 值图 1 ~ 3 中均表现为一条直线。

从图 1 看出,对于 promoters 数据集,当 k 值不同时,NKNNI-RFI 算法在 PFC、迭代次数和 ART 3 方面都有变化(由于 promoters 没有连续属性,所以图 1 中没有该数据集的 NRMSE 变化图),其中 PFC 虽有变化,但是变化区间约在0.63 ~ 0.67,与 RFI 在 promoters 数据集上得到的0.65只相差0.02左右。图 1(b)、(c)中,NKNNI-RFI 算法在 promoters 数据集上随着 k 值不同,其迭代次数和运行时间的变化是一致的。图 1(b)中,文中提出的方法与 RFI 算法(6 次迭代)相比能在不同 k 值时取得最低 3 次迭代和最高 8 次迭代。

在图 2(a)、(b)中,Sonar 数据集在 NKNNI-RFI 算法下取得的 NRMSE 总体上均低于 RFI 算法的 NRMSE(0.296),其 PFC 随 k 值的不同波动较为明显,最高约为0.14,最低约为0.02。但是从图 2(c)和(d)中可以看出,该数据集上 NKNNI-RFI 算法在迭代次数和 ART 方面大部分都低于 RFI 算法(12 次迭代,345s)。图 3(a)中,NKNNI-RFI 算法在 wpbc 数据集取得的 NRMSE 也是在 RFI 算法取得的0.67左右波动,而在图 3(b)中,NKNNI-RFI 算法随 k 值不同所取得的 PFC 均不低于 RFI 算法取得的 PFC(0.142)。在时间和迭代次数方面,图 3(c)、(d)表现出 NKNNI-RFI 算法在大多数 k 值下的迭代次数和 ART 均低于 RFI 算法(7 次迭代,81 s)。

可以得出:不同的 k 值对所提出的算法的运行时间和迭代次数影响较对 NRMSE 与 PFC 的影响大。最后对实验数据集在 NKNNI-RFI 算法中的 k 取 1 ~ 60 的实验结果变化情况进行观察后,发现实验数据集的 ART 和迭代次数均在 $k=36$ 时相较 RFI 算法的 ART 和迭代次数有所减少,因此将 36 作为 NKNNI-RFI 算法中 k 的取值。

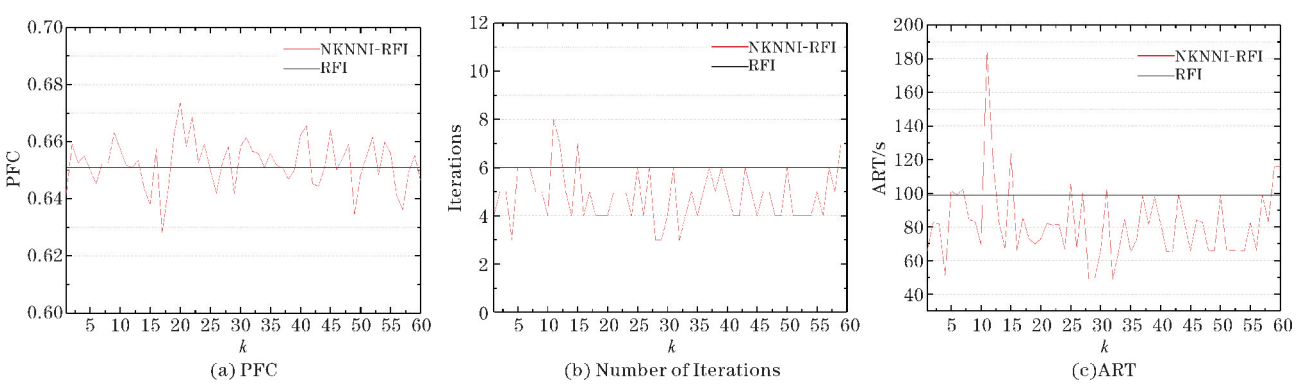


图 1 promoters 数据集在 k 值不同时的比较

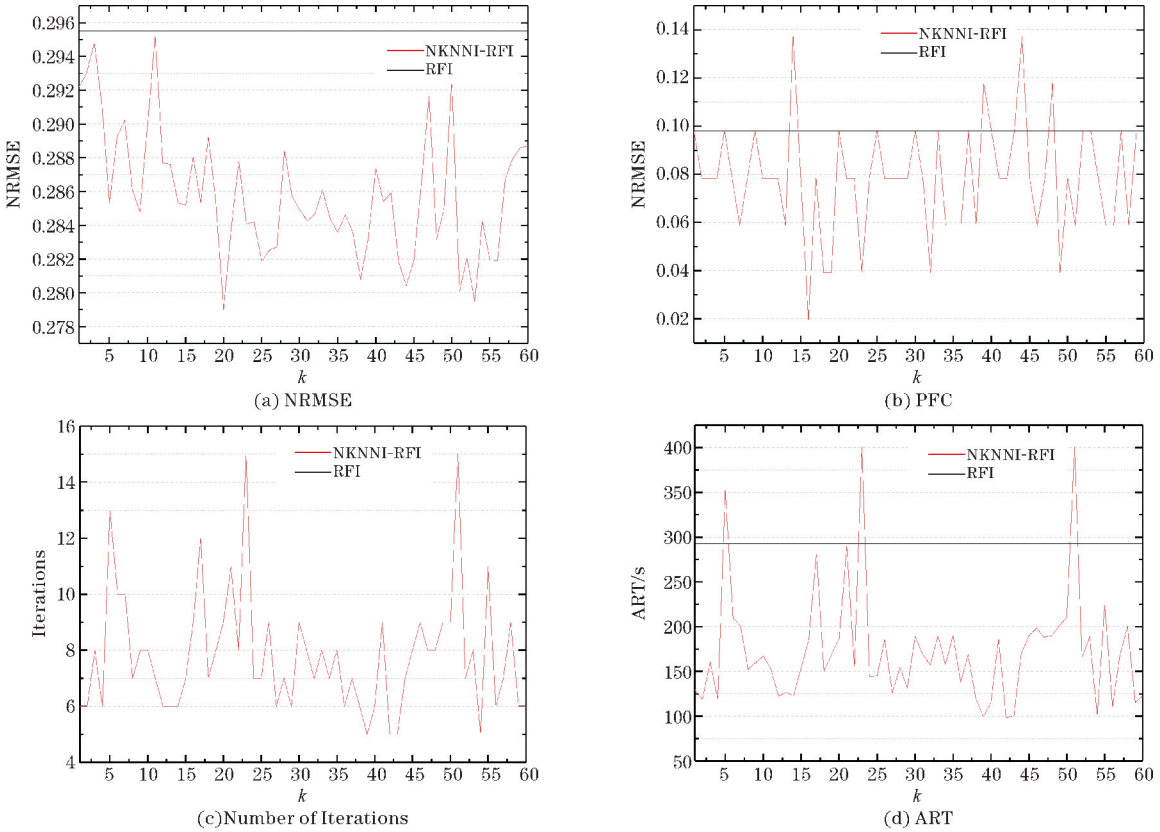


图 2　Sonar 数据集在 k 值不同时的比较

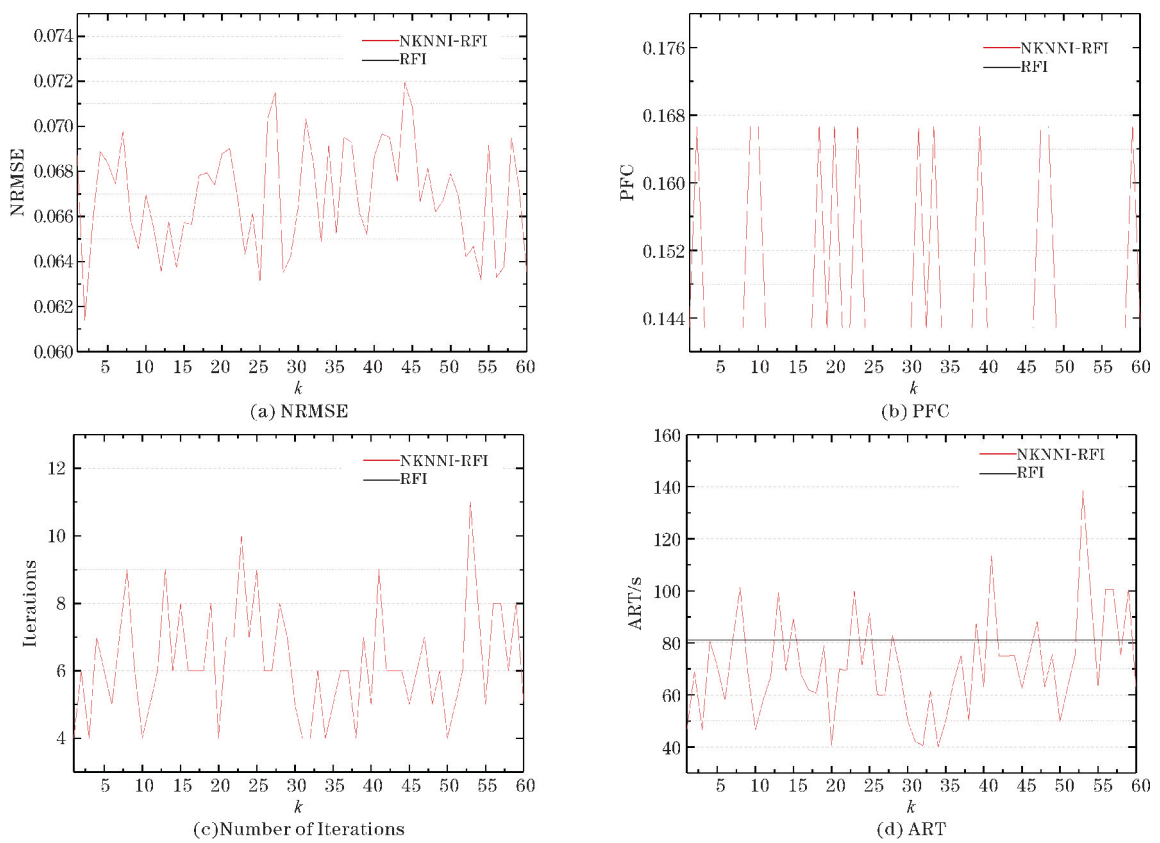


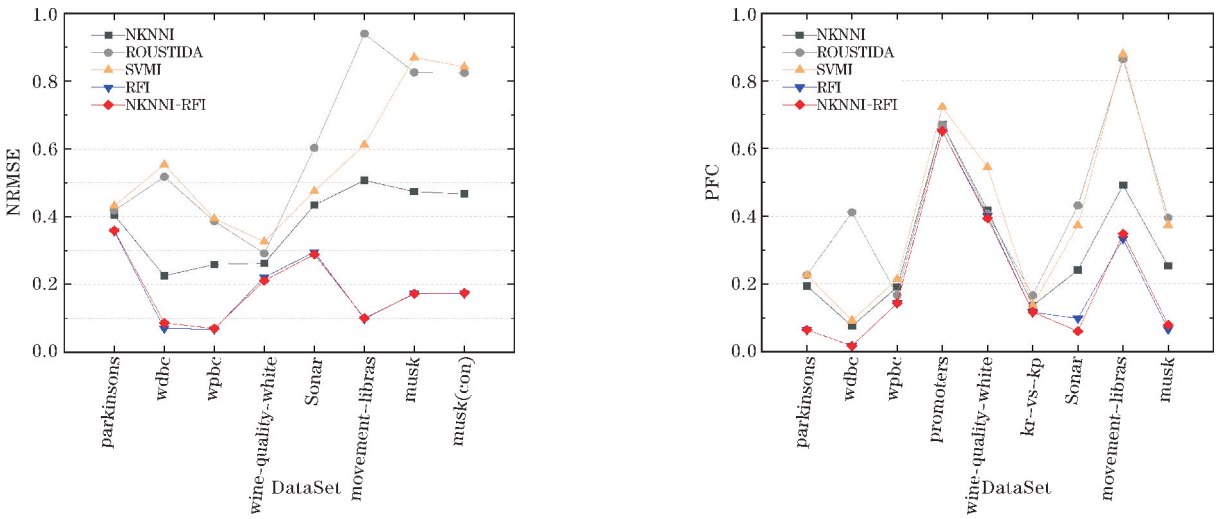
图 3　wpbc 数据集在 k 值不同时的比较

4.2　5 种算法下的 NRMSE 与 PFC 比较

在图 4(a)中,NKNNI-RFI 算法在所有数据集上都比 NKNNI、ROUSTIDA 和 SVMI 算法有好的填补效果,即更低的 NRMSE,相差约在0.1 ~0.8。但在 wdbc 数据集上得到的 NRMSE 比 RFI 算法的高0.02,其余的均与 RFI 算法相同。

在图 4(b)中,5 种算法在 wpbc, promoters, kr-vs-

kp 数据集上 PFC 均相差不明显。但从其余数据集中可以看出,NKNNI-RFI 算法和 RFI 算法的 PFC 大致相同并都低于其他 3 种算法的 PFC,最大的相差约0.5。除此之外,NKNNI-RFI 算法在 Sonar 数据集上的 PFC 比 RFI 算法约低0.04。由图 4 可知,在填补效果上,NKNNI-RFI 算法略优于 RFI 算法。



(a) NRMSE (b) PFC

图 4 数据集在各个算法下的 NRMSE 和 PFC 比较

4.3 5 种算法下的 ART 和 NKNNI-RFI 与 RFI 的迭代次数比较

表 3 为各个算法在实验数据集上的 ART 比较,表中最后一行为各个算法在所有实验数据集上的平均 ART。

表 4 为 NKNNI-RFI 算法相比 RFI 算法在各个实验数据集的 ART 减少程度(Reduction)。

可以由表 3 看出,NKNNI 算法运行时间最短,其次是 ROUSTIDA 和 SVMI 算法,两者的平均 ART 均在百秒以上。NKNNI-RFI 算法与 RFI 算法在总体上的运行时间是相对较长的,但是前者的运行时间相比后者更短。从表 3 可以看出文中提出的算法与 RFI 算法相比,在大部分数据集上 ART 均有减少。在 ART 较长的 movement-libras、musk 和 musk(con)数据集上能够分别减少209 s、263 s和548 s,其中 Sonar 数据集减少程度最大达 53%(表 4)。在 parkinsons 和 wdbc 数据集上,由于 k 值的限定导致两者迭代次数没有变化,因此 ART 也没有较大程度的变化,减少程度较小。

考虑时间的不稳定性,表 5 也给出了 RFI 算法和 NKNNI-RFI 算法在 10 个数据集上的相应迭代次数,可以看出,后者相比前者在实验中的大部分数据集上迭代次数都有减少(表 5 中黑体加粗),最大减少了 50%。

需要说明的是,当前 k 固定取值为 36,NKNNI-RFI 算法若是为每个数据集寻找能够最大程度减少时间的 k,那么在所有数据集上的 ART 都能够减少到自身的一半。但是在实验中,其他算法均对每个数据集保持固定的参数,为了保证算法之间的可对比性,实验中未采用这种寻找最优参数的方法。

表 3 各个算法在数据集上的 ART 比较

Dataset	NKNNI	ROUSTIDA	SVMI	RFI	NKNNI-RFI
parkinsons	0.03	2	5	43	41
wdbc	0.15	52	19	65	61
wpbc	0.05	14	16	81	48
promoters	0.1	8	17	99	72
wine-quality-white	4.93	158	84	162	115
kr-vs-kp	5.25	297	174	345	202
Sonar	0.21	19	30	293	138
movement-libras	0.11	95	135	915	706
musk	1.16	227	502	1684	1421
musk(con)	0.78	258	489	1708	1160
Average ART(s)	1	113	147	539	396

表 4　RFI 和 NKNNI-RFI 在数据集上的 ART 减少程度比较

Dataset	parkinsons	wdbc	wpbc	promoters	wine	kr-vs-kp	Sonar	libras	musk	musk(con)
Reduction/%	5	6	41	27	29	41	53	23	16	32

注:wine 为数据集 wine-quality-white 的简写,libras 为数据集 movement-libras 的简写。

表 5　RFI 和 NKNNI-RFI 在数据集上的迭代次数比较

Dataset	RFI	NKNNI-RFI
parkinsons	6	6
wdbc	4	4
wpbc	7	6
promoters	6	5
wine-quality-white	8	6
kr-vs-kp	8	6
Sonar	12	6
movement-libras	15	12
musk	8	7
musk(con)	7	5

5　结束语

提出的 NKNNI-RFI 算法与传统的 RFI 算法相比有下列优点和局限:

(1)通过改变 RFI 中预填补的方式,在保留 RFI 算法良好填补效果的同时缩短了算法运行时间。

(2)由文中的实验结果对比可知,当 NKNNI-RFI 算法中的 k 取 36 时,该算法迭代次数和运行时间方面有较好的表现。

(3)该算法有效缩短了运行时间,但是当 k 值固定时,不一定会在所有数据集上都能够减少运行时间。

因此,未来需要研究如何提升该算法的适用性,使其应用范围更广。

参考文献:

[1] Garc, A-laencina P J, Sancho-G, et al. K nearest neighbours with mutual information for simultaneous classification and missing data imputation[J]. Genetika, 1999, 72(7): 1483-1493.

[2] 王凤梅,胡丽霞. 一种基于近邻规则的缺失数据填补方法[J]. 计算机工程,2012,38(21):53-55.

[3] 金勇进. 缺失数据的插补调整[J]. 数理统计与管理,2001,20(6):47-53.

[4] 吴小姣,李高明,易大莉,等. 基因表达谱的非参缺失森林填补算法研究[J]. 中国卫生统计,2016,33(6):1068-1070.

[5] 谷峪,于戈,李晓静,等. 基于动态概率路径事件模型的 RFID 数据填补算法[J]. 软件学报,2010,21(3):438-451.

[6] Hartley H O. Maximum Likelihood Estimation from Incomplete Data[J]. Biometrics, 1958, 14(2): 174-194.

[7] Stekhoven D J, Buhlmann P. MissForest-non-parametric missing value imputation for mixed-type data[J]. Bioinformatics, 2012, 28(1): 112-118.

[8] Troyanskaya O, Cantor M, Sherlock G, et al. Missing value estimation methods for DNA microarrays[J]. Bioinformatics, 2001, 17(6): 520-525.

[9] Wang X, Li A, Jiang Z, et al. Missing value estimation for DNA microarray gene expression data by Support Vector Regression imputation and orthogonal coding scheme[J]. Bmc Bioinformatics, 2006, 7(1): 32.

[10] Weihua Zhu, Wei Zhang, Yunqing Fu. An incomplete data analysis approach using rough set theory [C]. 2004 International Conference on Intelligent Mechatronics and Automation, 2004. Proceedings. Chengdu, China: IEEE, 2004: 332-338.

[11] Arruda M P, Brown P J, Lipka A E, et al. Genomic Selection for Predicting Head Blight Resistance in a Wheat Breeding Program[J]. The Plant Genome, 2015, 8(3): 1-12.

[12] Rutkoski J E, Poland J, Jannink J L, et al. Imputation of unordered markers and the impact on genomic selection accuracy[J]. G3 Genesgenetics, 2013, 3(3): 427-439.

[13] Dixon, John K. Pattern Recognition with Partly Missing Data[J]. IEEE Transactions on Systems, Man and Cybernetics, 1979, 9(10): 617-621.

[14] Cover T, Hart P. Nearest neighbor pattern classification [J]. IEEE Transactions on Information Theory, 2003, 13(1): 21-27.

[15] Wenfeng Hou, Daiwei Li, Haiqing Zhang, et al. An Advanced k Nearest Neighbor Classification Algorithm Based on KD-tree[C]. 2018 IEEE International Conference of Safety Produce Informatization (IICSPI). Chongqing: IEEE, 2019: 902-905.

[16] Chao Xu, Daiwei Li, Haiqing Zhang, et al. A Weighted Fuzzy Rough Nearest Neighbor Classification Algorithm Based on Multiple Interpolation and Similarity Attribute Analysis[C]. 2018 IEEE International Conference of Safety Produce Informatization

(IICSPI). Chongqing: IEEE,2019:906-910.
[17] Breiman L. Random Forests[J]. Machine Learning,2001,45(1):5-32.
[18] 任家东,刘新倩,王倩,等. 基于 KNN 离群点检测和随机森林的多层入侵检测方法[J]. 计算机研究与发展,2019,56(3):116-125.
[19] Dua D,and Graff C. UCI machine learning repository[DB/OL]. http://archive. ics. uci. edu/ml, 2019-10-19/2019-10-19.
[20] 陈慧佳. 基于 Random Forest 的缺失数据补全策略研究[D]. 南昌:南昌大学,2016.
[21] Oba S,Sato M A,Takemasa I,et al. A Bayesian missing value estimation method for gene expression profile data[J]. Bioinformatics,2003,19(16):2088-2096.

A Random Forest Approach for Missing Data Imputation based on Normalized KNNI

YOU Feng, LI Daiwei, ZHANG Haiqing, WANG Jie, PENG Li, WANG Zhen
(College of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: The random forest imputation algorithm has reliable imputation performance when imputes incomplete information systems. At the same time, it needs to carry out random forest modeling for many times, which results in heavy computation. In order to shorten the running time of the algorithm, the NKNNI-RFI(normalization k nearest neighbor imputation-random forest imputation) algorithm is proposed. By changing the pre-imputation in RFI, normalized KNNI (NKNNI) with more accurate is used as the initial imputation, which provides data closer to the original data set for the prediction of the imputation value using the random forest model in RFI, enabling RFI to complete the imputation task in a shorter time and maintain a good effect. In the experiment, 10 UCI standard data sets were used to compare the proposed algorithm with algorithms including RFI, NKNNI, SVMI and ROUSTIDA, and the effectiveness of the algorithm was evaluated using NRMSE, PFC and ART evaluation methods for imputation methods. The experimental results show that the NRMSE and PFC of this algorithm are the same as RFI. NRMSE is 0.02-0.8 lower than NKNNI, SVMI and ROUSTIDA, and PFC is 0.01-0.6 lower than NKNNI, SVMI and ROUSTIDA. ART has a maximum reduction of 53% compared to RFI.

Keywords: incomplete information system; missing data imputation; normalization k nearest neighbor imputation; random forest imputation; evaluation of imputation method