小 型 微 型 计 算 机 系 统
Journal of Chinese Computer Systems
DOI:10.20009/j.cnki.21-1106/TP.2021-0766
2023 年 6 月 第 6 期
Vol.44 No.6 2023

融合过滤和相似度计算的高错误率基因组数据敏感序列识别

孙 辉,钟 诚

(广西大学 计算机与电子信息学院,南宁 530004)
(广西高校并行分布式计算技术重点实验室,南宁 530004)
E-mail:adairmillersh@gmail.com

摘 要: 为解决现有算法难以有效识别高错误率测序数据中敏感序列的问题,提出一种融合过滤和相似度计算的敏感序列识别算法.首先,分割待识别序列为多条短序列,通过构建双布隆过滤器,对短序列进行动态过滤去重,以避免重复运算;然后,对短序列局部片段进行 *k*-mer 编码,改进优化短序列局部片段相似性度量的方法,以准确识别短串联重复序列;其次,对短序列进行 *k*-mer 编码并与 GWAS Catalog 数据库中敏感序列进行计算比对,以准确识别疾病相关序列;最后,依据短序列识别结果,生成待识别序列的两条掩码序列,作为识别测序数据中敏感序列的结果.实验结果表明,与同类算法 LRF 和 SRF 相比,本文算法对错误率 2% ~20% 的测序数据中敏感序列的平均识别准确率分别提高 1.96% 和 3.66%,查准率分别提高 40.08% 和 68.36%,有效提升高错误率基因组数据中敏感序列识别的效果.

关 键 词: 敏感序列识别;皮尔逊相关系数;过滤;相似度计算;比对
中图分类号: TP301 **文献标识码:** A **文 章 编 号:** 1000-1220(2023)06-1227-09

Recognizing Sensitive Sequences from Genomic Data with High Error Rate Integrating Filter and Similarity Calculation

SUN Hui, ZHONG Cheng
(School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China)
(Key Laboratory of Parallel Distributed Computing Technology in Guangxi Universities, Nanning 530004, China)

Abstract: To solve the problem that existing algorithms are difficult to effectively identify sensitive sequences from sequencing data with high error rate, a recognizing sensitive sequence algorithm using filter and similarity calculation is proposed. Firstly, the genomic sequence is divided into several short sequences, and a double Bloom filter is constructed to de-duplicate each short sequence. Secondly, the local fragments of short sequences are encoded by *k*-mer, and the method for computing similarity of local fragments of short sequences are improved to identify short tandem repeats. Thirdly, *k*-mer encoding short sequences and sensitive sequences in GWAS Catalog database are aligned to identify disease-related sequences. Finally, according to the results of short sequence identification, two mask sequences of the sequencing data are generated as the final results of identifying sensitive sequences from the sequencing data. Experimental results show that compared with existing algorithms LRF and SRF, our proposed algorithm can enhance the average accuracy 1.96% and 3.66% and precision 40.08% and 68.36% of recognizing sensitive sequences from sequencing data with 2% ~ 20% error rate, respectively. The proposed algorithm can effectively improve the effect of recognizing sensitive sequences of genome data with high error rate.

Key words: sensitive sequence identification; Pearson correlation coefficient; filtering; similarity calculation; alignment

1 引 言

人类基因组测序已经在诸多领域取得了应用.然而,如果基因组数据没有得到有效的保护,那么将会带来很大的隐私泄露风险[1-3].Gymrek 等人的研究表明,通过短串联重复序列(STRs)和单核苷酸多态性(SNPs),可以根据基因组数据和个人公开信息发起重识别攻击(RIA),从而导致个人隐私数据泄露[4].基因组数据敏感序列识别是解决 DNA 数据隐私敏感问题的一个重要手段[5,6].

基因组敏感序列是人类 DNA 序列中易受隐私攻击影响的核苷酸序列,分为短串联重复序列和疾病相关序列两类[6,7].短串联重复序列是指基本重复单元为 1 ~6bp 的串联重复序列[8].由于短串联重复序列在人类遗传变异中占比较大,且容易通过聚合酶链式反应(PCR)扩增,所以短串联重复序列在疾病诊断和人员身份鉴定中具有重要作用.疾病相关序列是指携带疾病易感基因的核苷酸序列[7].由于疾病易感基因翻译成蛋白质可使个体表现出不同的性状,所以疾病相关序列可被用于个体定位.所谓敏感序列识别是指从基因组

收稿日期:2021-09-28 收修改稿日期:2021-11-12 基金项目:国家自然科学基金项目(61962004,61462005)资助;广西研究生教育创新计划项目(YCSW2021020)资助. 作者简介:孙 辉,男,1997 年生,硕士研究生,CCF 学生会员,研究方向为并行计算、生物信息计算;钟 诚(通讯作者),男,1964 年生,博士,教授,CCF 杰出会员,研究方向为并行计算、生物信息计算.

原始测序数据中识别出以上两类易受隐私攻击影响的 DNA 敏感序列. 通过敏感序列识别,对敏感序列片段给予保护,可以避免个人隐私数据的泄露.

目前已有用于识别短串联重复序列的一些启发式算法 REscan[9]、lobSTR[10]和 mTR[11]等. 然而,这些算法只能识别出部分基因组数据敏感序列,且对高错误率短串联重复序列识别准确性较差. 为有效识别基因组数据敏感序列,Cogo 等人提出了短序列过滤算法 SRF[6]. SRF 算法的实现依赖于第三方数据库收集的敏感序列和布隆过滤器(Bloom filter)[12,13]. 针对 SRF 算法拓展性差、难以识别潜在相似性敏感序列问题,Decouchant 等人提出基于 k-mer 分割的长序列过滤算法 LRF,并通过等位基因(allele)变异组合识别具有潜在相似性的敏感序列[7,14]. Fernandes 等人在 LRF 算法的基础上,通过连锁不平衡(LD)实现基因组数据敏感序列的多标签分类[15]. Lambert 等人通过敏感序列识别算法和软件防护扩展技术(SGX)实现了 DNA 序列的安全比对[16]. 随着单分子实时测序技术(SMRT)和牛津纳米孔测序技术(ONT)等第 3 代测序技术的发展,测序平台产生的序列越来越长且具有高错误率的特点[17].

基于多哈希映射的布隆过滤器无法对潜在相似序列产生相同的哈希映射[12]. 若采用基于布隆过滤器的敏感序列识别算法对高错误率的基因组长序列进行敏感序列识别,则其获得的识别准确率较低. 为更有效地识别出基因组数据高错误率长序列中的敏感序列,本文提出一种融合过滤和相似度计算的敏感序列识别算法. 本文剩余内容组织如下:第 2 节将详细阐述如何使用双布隆过滤器避免对重复序列的多次计算以及如何改进相似性度量计算方法以识别敏感序列;第 3 节给出本文算法和同类算法的实验评估结果;第 4 节总结本文工作.

2 方 法

设 $N[0:n-1]$ 表示长度为 n 的第 3 代测序序列,本文融合过滤和相似度计算的敏感序列识别方法的过程是:第 1 步,分割序列 N 为 $n-r+1$ 条短序列 $R_j=N[j:j+r-1]$,r 为 R_j 的长度,$j=0,1,2,\cdots,n-r,n>r$;第 2 步,采取双布隆过滤器方法对短序列 R_j 进行动态去重,以避免对相同短序列的重复计算,其中布隆过滤器 BF_1 用于敏感序列去重,布隆过滤器 BF_2 用于非敏感序列去重,$j=0,1,\cdots,n-r$;第 3 步,对短序列 R_j 进行短串联重复序列识别,以判断序列 R_j 是否是短串联重复序列,若 R_j 是短串联重复序列,则将 R_j 映射定位存储至布隆过滤器 BF_1 位数组 B_1,$j=0,1,2,\cdots,n-r$;第 4 步,将短序列 R_j 与 GWAS Catalog 数据库疾病敏感序列进行计算比对,以判断 R_j 是否是疾病相关序列,若 R_j 是疾病相关序列,则将 R_j 映射定位存储至布 BF_1 位数组 B_1,否则将 R_j 映射定位存储至 BF_2 位数组 B_2,$j=0,1,2,\cdots,n-r$;第 5 步,根据短序列 $R_0,R_1,\cdots,R_{n-r}$ 的敏感识别结果 $res[0:n-1]$,生成序列 $N[0:n-1]$ 的两条掩码序列 $N_{pub}[0:n-1]$ 和 $N_{pri}[0:n-1]$ 作为算法识别结果. 图 1 描述了本文方法的处理流程.

在图 1 中,结果向量 $res[0:n-1]$ 用于记录每条短序列 R_j 的敏感序列识别中间结果;为保存算法最终识别结果,本文采用类似于文献[7]的方法,根据结果向量 $res[0:n-1]$ 生成序列 $N[0:n-1]$ 的两条掩码序列 $N_{pub}[0:n-1]$ 和 $N_{pri}[0:n-1]$,其中 N_{pub} 表示掩码非敏感序列,N_{pub} 中的敏感序列片段将以“*”掩码,N_{pri} 表示掩码敏感序列,N_{pri} 中的非敏感序列片段将以“*”掩码. 通过生成掩码序列,对掩码敏感序列进行隐私保护,以避免针对基因组数据的隐私攻击.

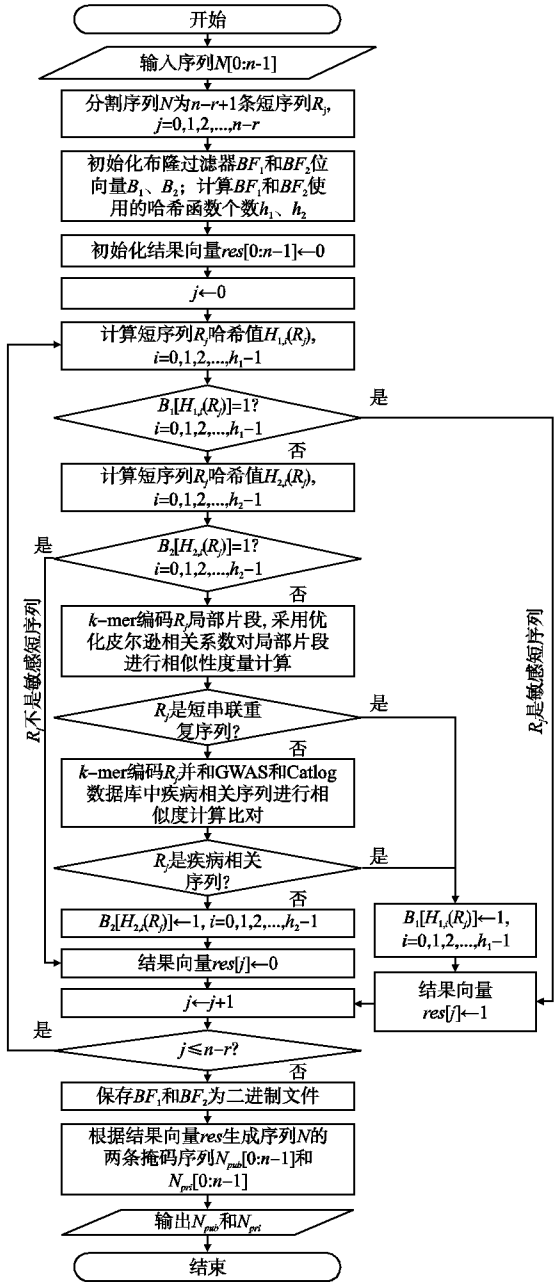

图 1 本文识别方法的处理流程

Fig. 1 Procedure of the proposed identification method

下面详细阐述融合过滤和相似度计算的高错误率敏感核苷酸序列识别方法.

2.1 序列分割

为识别测序长序列中的敏感序列,采取滑动窗口策略将长序列 $N[0:n-1]$ 分割为 $n-r+1$ 条、每条长度为 r 的短序列 $R_j=N[j:j+r-1]$,其中 R_j 和 R_{j+1} 共享重叠的 $r-1$ 个核苷

酸,以确保每条短序列敏感识别的准确性,$j=0,1,2,\cdots,n-r,n>r$.

人类 DNA 序列中,约有 99.5% 的序列具有高度一致性,只有 0.5% 的核苷酸记录个体独有信息[6]. 将长序列分割为短序列,会产生大量相同短序列. 若对所有短序列均进行敏感序列识别,则将耗费许多计算资源. 为避免对相同短序列的重复计算,本文利用布隆过滤器对分割得到的序列去重.

2.2　序列去重

布隆过滤器由一个二进制向量和若干哈希函数组成[14]. 基于多哈希映射的布隆过滤器可以降低哈希冲突的影响.

2.2.1　初始化布隆过滤器

设布隆过滤器 BF_1 和 BF_2 分别用于敏感序列和非敏感序列去重,其位向量分别为 $B_1[0:b_1-1]$ 和 $B_2[0:b_2-1]$,使用哈希函数的个数分别为 h_1 和 h_2. 初始化布隆过滤器,需计算位向量 B_1 和 B_2 的规模 b_1 和 b_2[12]:

$$b_i=\frac{t_i\times ln\left(p_i^{-1}\right)}{(ln2)^2},i=1,2 \tag{1}$$

式(1)中,t_i 表示布隆过滤器 BF_i 可容纳序列数量的上限,p_i 表示误报率,BF_i 位向量规模 b_i 和可容纳序列数量的上限 t_i 呈正相关,b_i 和误报率 $p_i(0<p_i<1)$ 呈负相关,通过预设 t_i 和 p_i 便可以根据式(1)计算出 $b_i,i=1,2$. 此时,布隆过滤器 BF_i 使用的哈希函数个数 h_i[12]:

$$h_i=\left\lceil\frac{b_i}{t_i}\times ln2\right\rceil,i=1,2 \tag{2}$$

在计算出 b_1 和 b_2,h_1 和 h_2 之后,置 $B_{1,i}=0$、$B_{2,k}=0$,$i=0,1,2,\cdots,b_1-1$,$k=0,1,2,\cdots,b_2-1$.

2.2.2　短序列去重

对短序列 R_j 去重,需要根据已知短序列 R_k 的识别结果将 R_k 映射定位存储至对应布隆过滤器,$k=0,1,2,\cdots,j-1$. 若 R_k 为敏感短序列,则对 R_k 进行 h_1 次哈希计算得到 h_1 个哈希值 $H_{1,i}(R_k)$,并将布隆过滤器 BF_1 中位向量 $B_1[H_{1,i}(R_k)]$ 置为 1,$i=0,1,2,\cdots,h_1-1$. 若 R_k 为非敏感短序列,则对 R_k 进行 h_2 次哈希计算得到 h_2 个哈希值 $H_{2,i}$,并将布隆过滤器 BF_2 中位向量 $B_2[H_{2,i}(R_k)]$ 置为 1,$i=0,1,2,\cdots,h_2-1$.

若要判断 R_j 是否是一条重复序列,首先对 R_j 进行 h_1 次哈希计算,得到 h_1 个哈希值 $H_{1,i}(R_j)$,并在 BF_1 中的位向量 B_1 进行查询,若 $B_1[H_{1,i}(R_j)]$ 均为 1,$i=0,1,2,\cdots,h_1-1$,则 R_j 是一条重复短序列,且是敏感短序列,此时置 $res[j]=1$. 若 $B_1[H_{1,i}(R_j)]$ 不全为 1,则对 R_j 进行 h_2 次哈希计算,得到 h_2 个哈希值 $H_{2,i}(R_j)$,并在 BF_2 中的位向量 B_2 进行查询,若 $B_2[H_{2,i}(R_j)]$ 均为 1,$i=0,1,2,\cdots,h_2-1$,则 R_j 是重复短序列,且是非敏感短序列,此时置 $res[j]=0$. 若 $B_2[H_{2,i}(R_j)]$ 不全为 1,则说明 R_j 不存在于 BF_1 或 BF_2 中,R_j 不是一条重复序列,需要对 R_j 执行短串联重复和疾病相关序列识别处理.

为更好体现算法去重思想,图 2 给出序列去重示例.

在图 2 中,BF_1 和 BF_2 的位向量 B_1 和 B_2 均是一个 8 位的二进制向量;BF_i 序列去重使用 3 个哈希函数 $H_{i,0}$、$H_{i,1}$ 和 $H_{i,2}$,$i=1,2$;R_1、R_3 和 R_4 是敏感短序列,R_2 和 R_5 是非敏感短序列,且 R_5 是重复短序列($R_5=R_2$),为此去重序列 R_5. 首先,初始化布隆过滤器 BF_1 和 BF_2 的位向量 B_1、B_2 均为 0,此时 BF_1 和 BF_2 均为空,没有记录任何短序列信息. 然后,将 R_1 ~ R_4 映射定位存储至 BF_1 和 BF_2 的位向量 B_1 和 B_2;此时 B_1 第

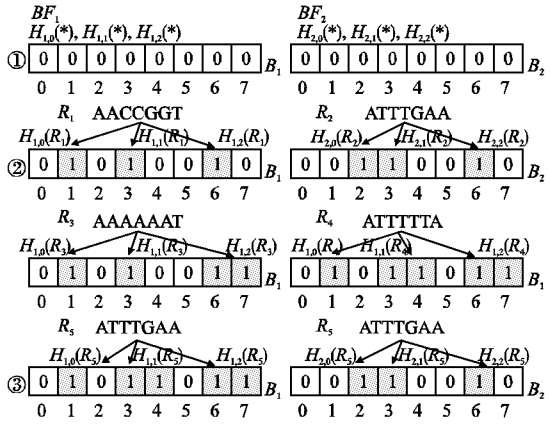

图 2　布隆过滤序列去重示例

Fig. 2　Example of bloom filter sequence de-duplication

1,3,4,6,7 位被设置为 1,B_2 第 2,3,6 位被设置为 1. 最后,序列 R_5 通过 BF_1 进行敏感序列去重时所产生的 3 个哈希标记不全为 1. 因此需要经过 BF_2 进行非敏感序列去重;此时对 R_5 进行哈希映射所产生的 3 个哈希值在 BF_2 中标记均为 1,因此 R_5 是重复短序列,且 R_5 由 BF_2 过滤出,所以 R_5 是非敏感短序列. 通过双布隆过滤器处理可知序列 R_5 为非敏感短序列,因此,不需要执行后续短串联重复序列和疾病相关序列识别操作.

通过对短序列执行双布隆过滤器去重操作,可避免对短序列的重复计算. 若短序列不存在于布隆过滤器 BF_1 或 BF_2,则需要对短序列进行短串联重复序列识别.

2.3　短串联重复序列识别

设 $\Sigma=\{A,C,G,T\}$ 表示组成 DNA 序列的符号集,Σ^+ 表示 Σ 上的非空字符串集,对于序列 $u\in\Sigma^+$,一个理想的短串联重复序列可表示为 $[u]^q$,其中 q 表示序列 u 的重复次数,$|u|$ 表示序列 u 的长度且 $1bp\leqslant|u|\leqslant 6bp$,$u$ 为最小基本重复单元[8]. 短串联重复序列的内部重复结构表明,通过合理分割短串联重复序列可以得到记录短串联重复序列基本重复单元的相同子段[11]. 为此,借鉴文献[11]的思想,本文采用以下步骤判断短序列是否是含有“插入”、“删除”、“替换”错误的短串联重复序列:

第 1 步. 等长度分割短序列 R_j 为 4 个长度为 w 的短片段,$W_j=\{W_{j,1},W_{j,2},W_{j,3},W_{j,4}\}$,$j=0,1,2,\cdots,n-r$;

第 2 步. 根据 DNA 序列符号集,生成 4^k 个 k-mer,统计短片段 $W_{j,2}$、$W_{j,3}$ 中每个 k-mer 出现的频数,将短片段 $W_{j,2}$、$W_{j,3}$ 转换为记录 k-mer 频率的向量 $\vec{W}_{j,2}$、$\vec{W}_{j,3}$,其中 $\vec{W}_{j,i}=[W_{j,i,0},W_{j,i,1},\cdots,W_{j,i,v},\cdots,W_{j,i,4^k-1}]$,$W_{j,i,v}$ 为 $W_{j,i}$ 中第 v 个 k-mer 的统计频数,$v=0,1,2,\cdots,4^k-1$,$0\leqslant W_{j,i,v}\leqslant w-k+1$,$i=2,3$,$j=0,1,2,\cdots,n-r$.

例:设 $R_j=[ACG]^{16}$,则 $W_{j,2}=[ACG]^4$,若采用 2-mer 编码短片段 $W_{j,2}$,则编码 k-mer 共有 $4^k=4^2=16$ 种取值,即 $\{AA,AC,AG,\cdots,TC,TG,TT\}$. 通过 2-mer 编码 $W_{j,2}$ 可得 AC、CG、GA 的频数分别为 4、4、3. 于是 $W_{j,2}$ 经过 k-mer 编码后的词频向量 $\vec{W}_{j,2}=[0,4,0,0,0,4,0,0,0,3,0,0,0,0,0,0,0,0]$.

第3步. 计算短序列 R_j 局部片段 $W_{j,2}$ 和 $W_{j,3}$ 的相似度

$sim(W_{j,2}, W_{j,3})$,并根据短串联重复序列相似度阈值 str_sim 判断 R_j 是否为短串联重复序列,若 R_j 为短串联重复序列,则置结果向量 $res[j]=1, j=0,1,2,\cdots,n-r$.

皮尔逊相关系数(PCC)常用于度量向量 $\vec{X}$ 和 $\vec{Y}$ 的线性相关性,其值介于 -1 与 1 之间,相关系数绝对值越大说明 $\vec{X}$ 和 $\vec{Y}$ 的相似度越高[11]. 为降低算法将非敏感序列误识别为短串联重复序列的概率,本文引入控制参数 c 改进计算 $\vec{W}_{j,2}$ 和 $\vec{W}_{j,3}$ 相似度 $sim(W_{j,2},W_{j,3})$ 的方法如下:

$$sim(W_{j,2},W_{j,3}) = c * \frac{\sum_{i=0}^{4^{k-1}}(\vec{W}_{j,2}-\overline{W}_{j,2})(\vec{W}_{j,3}-\overline{W}_{j,3})}{\sqrt{\sum_{i=1}^{4^{k-1}}(\vec{W}_{j,2}-\overline{W}_{j,2})^2}\sqrt{\sum_{i=1}^{4^{k-1}}(\vec{W}_{j,3}-\overline{W}_{j,3})^2}} \quad (3)$$

$$\begin{cases} c=1, \text{if } max(\vec{R}_{j,[w,3w]}) \geqslant \left\lceil \frac{2w-k+1}{6} \right\rceil \\ c=0, \text{if } max(\vec{R}_{j,[w,3w]}) < \left\lceil \frac{2w-k+1}{6} \right\rceil \end{cases}$$

式(3)中 $\overline{W}_{j,2}=\overline{W}_{j,3}=(w-k+1)/4^k$, $\vec{R}_{j,[w,3w]}$ 为短序列 $R_j\left[\frac{w}{4}:\frac{3*w}{4}-1\right]$ 的 k-mer 频率向量.

图 3 给出本文方法识别短串联重复序列的示例.

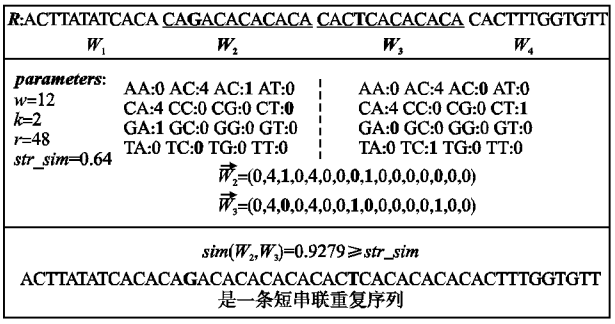

图 3 短串联重复序列识别示例

Fig. 3 Example of short tandem repetition recognition

从图 3 中可知,通过上述步骤计算出第 2、3 个短片段的相似性达 92.79%,大于给定的判别阈值 $str\_sim=0.64$,因此 R 是一条短串联重复序列.

通过对 R_j 中的两个局部片段统计 k-mer 词频,可以提取 R_j 的 k-mer 统计特征. 若 R_j 具有较少的“插入”、“删除”、“替换”错误,通过 k-mer 编码 R_j 的局部片段,可以确保局部片段词频具有较高的一致性,从而可以计算得到较高的相似度值,进而识别 R_j 是否是具有潜在相似性的高错误率的短串联重复序列, $j=0,1,2,\cdots,n-r$.

2.4 疾病相关序列识别

通过对序列结构分析处理可以识别出 DNA 序列中包含的短串联重复序列. 除了短串联重复序列,DNA 序列中还可能包含有疾病相关序列. 为识别短序列 R_j 是否是疾病相关序列,本文通过构建敏感序列词典,检索与 R_j 碱基含量相似的第三方数据库中的疾病相关序列,采用皮尔逊相关系数对 R_j 和检索到的疾病相关序列进行相似度计算比对,以识别出高错误率疾病相关序列, $j=0,1,2,\cdots,n-r$.

2.4.1 构建敏感序列词典

构建敏感序列词典的目的是为了降低大量短序列与疾病相关序列相似度计算的代价. 本文使用二维数组 $DICTI[m][4^k+3]$ 表示敏感序列词典,其中 m 表示从第三方数据库提取的疾病相关序列的数量, k 为采用 k-mer 编码核苷酸序列的 k 值. 构建敏感序列词典的方法是:

首先,从第三方数据库提取每个疾病易感基因的染色体编号 CHR_ID、染色体位置 CHR_POS,并使用 pysam 工具[1] 从参考基因组 HG38 中提取 m 条长度为 r 的疾病相关序列 D_i, $i=0,1,\cdots,m-1$. 然后,生成 4^k 个 k-mer,通过统计 D_i 中每个 k-mer 出现的频数,将 D_i 编码为词频向量 $\vec{D}_i=[D_{i,0},D_{i,1},\cdots,D_{i,v},\cdots,D_{i,4^k-1}]$,其中 $D_{i,v}$ 为 D_i 中第 v 个 k-mer 的统计频数, $0\leqslant D_{i,v}\leqslant r-k+1, i=0,1,2,\cdots,m-1, v=0,1,2,\cdots,4^k-1$. 其次,统计计算疾病相关序列 D_i 的 G、C、A 碱基含量 $GCA_{D_i}=[G_{D_i},C_{D_i},A_{D_i}], i=0,1,2,\cdots,m-1$. 最后,将 $\vec{D}_i$ 和 GCA_{D_i} 写入 $DICTI[i]=\{\vec{D}_i,GCA_{D_i}\}, i=0,1,2,\cdots,m-1$.

2.4.2 碱基含量相似性检索

构建敏感核苷酸序列词典之后,对 R_j 进行碱基含量相似性检索,以进一步降低 R_j 和敏感序列词典中的序列相似度计算的工作量. 文献[11]的研究表明,潜在相似性序列具有碱基含量的一致相似性. 序列 R_j 中 G、C、A 碱基含量 $GCA_{R_j}=[G_{R_j},C_{R_j},A_{R_j}]$,本文采用式(4)的条件,在 $DICTI$ 中检索与 R_j 具有碱基含量相似的序列 D_i:

$$1-|GCA_{R_j}[e]-GCA_{D_i}[e]| \geqslant dis\_sim \quad (4)$$

式(4)中, $i=0,1,2,\cdots,m-1, j=0,1,2,\cdots,n-r, e=0,1,2$, dis_sim 表示疾病相关序列识别相似度阈值. 设 $queryB$ 为通过序列碱基含量相似性检索得到短序列 R_j 和 D_i 满足式(4)条件的 D_i 下标集合,则 $queryB=[q_0,q_1,\cdots q_v,\cdots,q_t]$,其中 q_v 为 $DICTI$ 中碱基含量相似序列的下标, $t\leqslant m-1$.

2.4.3 疾病相关序列相似度计算

为判别短序列 R_j 是否为疾病相关序列,首先统计 R_j 中每个 k-mer 出现的频数,将序列 R_j 编码为词频向量 $\vec{R}_j=[R_{j,0},R_{j,1},\cdots,R_{j,v},\cdots,R_{j,4^k-1}]$,其中 $R_{j,v}$ 为 R_j 中第 v 个 k-mer 的统计频数, $0\leqslant R_{j,v}\leqslant r-k+1, v=0,1,2,\cdots,4^k-1$. 然后,依次从 $DICTI$ 提取序列 D_i 的词频向量 $\vec{D}_i$,并按式(5)计算 $\vec{R}_j$ 和 $\vec{D}_i$ 的皮尔逊相关系数 $pcc(\vec{R}_j,\vec{D}_i)$[11]:

$$pcc(\vec{R}_j,\vec{D}_i)=\frac{\sum_{i=0}^{4^k-1}(\vec{D}_i-\bar{D}_i)(\vec{R}_j-\bar{R}_j)}{\sqrt{\sum_{i=0}^{4^k-1}(\vec{D}_i-\bar{D}_i)^2}\sqrt{\sum_{i=0}^{4^k-1}(\vec{R}_j-\bar{R}_j)^2}} \quad (5)$$

式(5)中, $\bar{D}_i=\bar{R}_j=(r-k+1)/4^k, i\in queryB$. R_j 和 D_i 的相似性度量值 $sim(R_j,D_i)=pcc(\vec{R}_j,\vec{D}_i)$. 当找到满足 $pcc(\vec{R}_j,\vec{D}_i)\geqslant dis\_sim$ 条件的 D_i 后,此时 R_j 和 D_i 具有高相似性,将 R_j 判定为疾病相关序列.

当序列中含有“插入”、“删除”、“替换”错误时,采用 k-

[1] https://pysam.readthedocs.io

mer 编码短序列,其差异只体现在变异核苷酸相邻 k 个碱基处,通过合理取值 k,可有效提取高错误率疾病相关序列碱基统计特征.将短序列 R_j 和第三方数据库疾病相关序列进行相似度计算,通过设置相似度阈值从而可以识别出和第三方数据库具有较高相似性的疾病相关序列.

2.5 识别算法

设待识别长序列为 $N[0:n-1]$、滑动窗口策略分割短序列 R_j 的窗口长度为 r、双布隆过滤器误报率分别为 p_1 和 p_2、初始化双布隆过滤器可插入最大序列数目分别为 t_1 和 t_2、k-mer 编码取值为 k、短串联重复序列识别相似度阈值为 *str_sim*、疾病相关序列识别相似度阈值为 *dis_sim*、参考基因组为 *HG*、GWAS Catlog 数据库中疾病相关序列数据集为 *diseaseData*(包含 m 组数据).

算法 1 形式描述了融合过滤和相似度计算的高错误率敏感序列识别算法(简记为 F3SR 算法).

算法 1. F3SR

输入:$N[0:n-1]$, r, p_1, p_2, t_1, t_2, k, *str_sim*, *dis_sim*, *HG*, *diseaseData*

输出:序列 N 识别结果的两条掩码核苷酸序列 N_{pri}, N_{pub}

Begin

1. 依据 p_1, p_2, t_1 和 t_2 初始化布隆过滤器 BF_1 和 BF_2 位数组 B_1 和 B_2;
2. 依据 *diseaseData*, r, *HG* 和 k 构建敏感序列词典 *dicti*$[m,4^k+3]$;
3. $res[0:n-1]\leftarrow 0$;
4. $h_1\leftarrow\lceil \ln(p_1^{-1})/ln2\rceil$;
5. $h_2\leftarrow\lceil \ln(p_2^{-1})/ln2\rceil$;
6. **for** $j=0$ **to** $n-r$ **do**
7. $R_j\leftarrow N[j:j+r]$;//提取第 j 条短序列 R_j
8. 对序列 R_j 执行 h_1 次哈希计算得到 h_1 个哈希值 $H_{1,0}(R_j)$, $H_{1,1}(R_j)$, $\cdots$, $H_{1,h_1-1}(R_j)$;
9. **if** BF_1 中 $B_1[H_{1,0}(R_j)]$, $B_1[H_{1,1}(R_j)]$, $\cdots$, $B_1[H_{1,h_1-1}(R_j)]$ 的值全为 1 **then**
10. $res[j]\leftarrow 1$;
11. 对序列 R_j 执行 h_2 次哈希计算得到 h_2 个哈希值 $H_{2,0}(R_j)$, $H_{2,1}(R_j)$, $\cdots$, $H_{2,h_2-1}(R_j)$;
12. **else if** BF_2 中 $B_2[H_{2,0}(R_j)]$, $B_2[H_{2,1}(R_j)]$, $\cdots$, $B_2[H_{2,h_2-1}(R_j)]$ 的值全为 1 **then**
13. $res[j]\leftarrow 0$;
14. **else** //编码短序列局部片段
15. $\vec{w}_{j,2}\leftarrow encoding(R_j[\frac{1}{4}*r:\frac{1}{2}*r],k)$;
16. $\vec{w}_{j,3}\leftarrow encoding(R_j[\frac{1}{2}*r:\frac{3}{4}*r],k)$;
17. $\vec{w}_{j,[w:3w]}\leftarrow encoding(R_j[\frac{1}{4}*r:\frac{3}{4}*r],k)$;
18. **if** $\max(\vec{w}_{j,[w:3w]})\geqslant\lceil\frac{2w-k+1}{6}\rceil$ **then**
19. $c\leftarrow 1$;
20. **else**
21. $c\leftarrow 0$;
22. **end if**
23. $sim(W_{j,2},W_{j,3})\leftarrow$计算 $pcc(\vec{w}_{j,2},\vec{w}_{j,3})*c$;
24. **if** $sim(W_{j,2},W_{j,3})\geqslant str\_sim$ **then**
25. $res[j]\leftarrow 1$;
26. 将 $B_1[H_{1,0}(R_j)]$, $B_1[H_{1,1}(R_j)]$, $\cdots$, $B_1[H_{1,h_1-1}(R_j)]$ 的值置为 1;
27. **else**
28. 依据短序列 R_j 的 GAC 碱基含量对 *dicti* 进行相似性序列检索以获得 *queryB*;
29. **if** $queryB=\varnothing$ **then**
30. $res[j]\leftarrow 0$;
31. 将 $B_2[H_{2,0}(R_j)]$, $B_2[H_{2,1}(R_j)]$, $\cdots$, $B_2[H_{2,h_2-1}(R_j)]$ 的值置为 1;
32. **else**
33. $\vec{R}_j\leftarrow encoding(R_j,k)$;
34. **for** $i=0$ to $|queryB|-1$ **do**
35. 根据 $queryB[i]$ 从 *dicti* 获取 $\vec{D}_i$;
36. $sim(R_j,D_i)\leftarrow$计算 $pcc(\vec{R}_j,\vec{D}_i)$;
37. **if** $sim(R_j,D_i)\geqslant dis\text{-}sim$ **then**
38. $res[j]\leftarrow 1$;
39. 将 $B_1[H_{1,0}(R_j)]$, $B_1[H_{1,1}(R_j)]$, $\cdots$, $B_1[H_{1,h_1-1}(R_j)]$ 的值置为 1;
40. *goto* (6);
41. **end if**
42. **end for**
43. $res[j]\leftarrow 0$;
44. 将 $B_2[H_{2,0}(R_j)]$, $B_2[H_{2,1}(R_j)]$, $\cdots$, $B_2[H_{2,h_2-1}(R_j)]$ 的值置为 1;
45. **end if**
46. **end if**
47. **end if**
48. **end for**
49. 保存布隆过滤器 BF_1 和 BF_2 为二进制文件;
50. $N_{pub}\leftarrow N[0:n-1]$;//初始化掩码非敏感序列
51. $N_{pri}\leftarrow N[0:n-1]$;//初始化掩码敏感序列
52. $beginPos\leftarrow 0$;
53. **while** $beginPos<n-r$ **do**//生成掩码非敏感序列 N_{pub}
54. **if** $res[beginPos]\neq 1$ **then**
55. $beginPos\leftarrow beginPos+1$;
56. **else**
57. **for** $i=beginPos$ to $beginPos+r$ **do**
58. **if** $res[beginPos+r-i]=1$ **and** $i\neq beginPos+r$ **then**
59. **for** $j=beginPos$ **to** $beginPos+r-i$ **do**
60. $N_{pub}[j]\leftarrow$ '*';
61. **end for**
62. $beginPos\leftarrow beginPos+r-i$;
63. goto(53);
64. **else if** $i=beginPos+r$ **then**
65. **for** $j=beginPos$ **to** $beginPos+r$ **do**
66. $N_{pub}[j]\leftarrow$ '*';
67. **end for**
68. $beginPos\leftarrow beginPos+r$;
69. goto(53);
70. **end if**
71. **end for**
72. **end if**
73. **end while**
74. **for** $j=0$ **to** $n-1$ **do**
//根据掩码序列 N_{pub} 生成掩码敏感序列 N_{pri}
75. **if** $N_{pub}[j]\neq$ '*' **then**
76. $N_{pri}[j]\leftarrow$ '*';
77. **end if**
78. **end for**

End.

算法 F3SR 步 1 初始化布隆过滤器时间为 $O(1)$;步 2 构建敏感序列词典所需时间为 $O(m\times r\times 4^k)$;步 3 初始化结果向量的时间为 $O(n)$;步 4 ~ 步 5 计算布隆过滤器哈希函数个数时间为 $O(1)$;步 34 ~ 步 42 最坏情况下所需的时间为 $O(m\times 4^k)$;步 6 ~ 步 48 时间为 $O(\max(n\times h_1, n\times h_2, n\times r\times 4^k, n\times 4^k, n\times m\times 4^k))$;步 49 保存二进制文件时间为 $O(1)$;步 50 ~ 步 51 初始化掩码序列的时间为 $O(n)$;步 52 ~ 步 73 生成掩码非敏感序列的时间为 $O(n\times r^2)$;步 74 ~ 步 78 生成掩码敏感序列的时间为 $O(n)$.由于 k、r、m、h_1 和 h_2 均为某个常数,所以算法时间复杂度为 $O(n)$.

算法F3SR运行过程中,序列N所用空间为$O(n)$;布隆

过滤器所需空间为 $O\left(\left|\frac{t_1\times lnp_1^{-1}+t_2\times lnp_2^{-1}}{(ln2)^2}\right|\right)$;敏感序列词典所需空间为 $O(m\times 4^k)$;每条短序列识别结果的向量 *res* 的空间开销为 $O(n)$;序列相似性检索数组 *queryB* 最坏情况下所需空间为 $O(n)$;*k*-mer 编码短序列所用的空间开销为 $O(4^k)$;两条掩码序列 N_{pub} 和 N_{pri} 的空间开销为 $O(n)$. 由于 *k*、*m*、p_1 和 p_2 均为某个常数,所以算法空间复杂度为 $O\left(n+\left|\frac{t_1\times lnp_1^{-1}+t_2\times lnp_2^{-1}}{(ln2)^2}\right|+m\times 4^k\right)=O(\max\{n,t_1+t_2\})$.

算法 F3SR 通过 *k*-mer 编码对 DNA 序列进行词频统计,将待识别序列中每个核苷酸的邻接信息进行有效提取. 由于相似序列的差异只体现在差异核苷酸的局部序列片段,通过 *k*-mer 编码可以使具有相似性的核苷酸序列保持词频的整体一致性,通过皮尔逊相关系数可计算出较高相似度得分的序列,所以可以识别出高错误率的敏感核苷酸序列. 此外,基于"序列过滤"的思想,算法始终避免对重复序列的计算,有效提升了算法的效率.

3 实 验

3.1 实验环境与数据

实验在设立于广西大学的国家高性能计算中心南宁分中心[2] 的 Sugon 7000A 超级并行计算机系统上进行,使用的计算节点配置为 2 × Intel Xeon Gold 6230 处理器、512GB 内存以及 8 × 900GB 外部存储空间,运行操作系统 CentOS7.4. 采用 C++语言编程实现算法,布隆过滤器实现参考自 ArashPartow[3] 实现的混合 hash 版本.

表 1 实验数据集信息

Table 1 Information of experimental datasets

数据集名称	数据集描述	序列条数	数据规模
All-STR	12.5%短串联重复序列 + 87.5%非敏感序列	1000660	0.534GB
All-Disease	12.5%疾病相关序列 + 87.5%非敏感序列	1238680	0.647GB
All-Together	12.5%敏感序列 + 87.5%非敏感序列	2239340	1.169GB

本文采用 GWAS Catalog 数据库[18] 全基因组关联研究数据集 gwas_catalog_v1.0.tsv 构建敏感序列词典,该数据集包含 HIV-1 replication、Opioid sensitivity 等 4680 种疾病,共计 216250 组数据. 每组数据包含疾病名称 *DISEASE*、染色体编号 *CHR_ID*、染色体位置 *CHR_POS* 等 34 个标识信息. 实验数据采用的参考基因组为 NCBI[4] 开放下载的人类基因组 HG38. 实验所用的短串联重复序列来自 TRDB 数据库[5],疾病易感基因来自 GWAS Catalog 数据库[6]. 参照文献[6]的数据预处理规则,预处理每条序列长度为 50bp 的 3 个基准数据序列集,每个基准数据集的敏感序列和非敏感序列的数量比例为 1:7,实验数据信息如表 1 所示.

为获得含有错误信息的序列数据集,采用表 1 数据集作为基准序列,通过随机执行"插入"、"删除"、"替换"操作产生 10 组错误率为 2%~20% 的 DNA 序列进行实验. 实验将本文算法 F3SR 与同类算法 SRF[6] 和 LRF[7] 进行测试比较识别的准确率(*accuracy*, *acc*)[6]、查准率(*precision*, *pre*)[19] 和假阳性率(*false positive rate*, *fpr*)[7]:

$$acc=\frac{TP+TN}{TP+TN+FP+FN}\times 100\% \tag{6}$$

$$pre=\frac{TP}{TP+FP}\times 100\% \tag{7}$$

$$fpr=\frac{FP}{TN+FP}\times 100\% \tag{8}$$

其中,*TP* 表示算法识别出真实敏感序列的总数、*TN* 表示识别出真实非敏感序列的总数、*FN* 表示将敏感序列误识别为非敏感序列的总数、*FP* 表示将非敏感序列误识别为敏感序列的总数. 准确率反映算法的整体识别结果,准确率越高算法识别效果越好. 查准率反映真实敏感序列和算法识别出的敏感序列的比率,查准率越高算法越好. 假阳性率反映数据中非敏感序列被误识别为敏感序列的比率,假阳性率越低越好.

3.2 算法参数选取

为使得采用布隆过滤器对序列去重时具有较低的假阳性率,本文采用文献[14]布隆过滤器实验参数 $p_i=0.0001$ 对布隆过滤器初始化,并设置布隆过滤器容纳序列数量上限 $t_i=10^4$,当实际插入序列的数目超过 t_i 时,动态扩充布隆过滤器的容纳上限为当前上限的 2 倍,$i=1,2$.

F3SR 算法判别短序列 R_j 是否为短串联重复序列时,需要对 R_j 第 2 和第 3 个短片段进行 *k*-mer 编码. 通过对 TRDB 数据库短串联重复序列检索,TRDB 数据库中短串联重复序列长度最短为 25bp[20],为使得 F3SR 算法可以准确识别短串联重复序列,F3SR 算法取 $w=12$bp. 此时,滑动窗口策略分割短序列 R_j 的长度 $r=4w=48$bp.

设 *err* 表示测序引入的错误率,第三代测序错误率高达 10~20%[11],考虑测序过程引入最大错误率 $err=20\%$,算法相似度阈值 *t* 应满足 $t=1-\max(err)=0.8$,因此短串联重复序列识别相似度阈值 $str\text{-}sim\leqslant t^2=0.80^2=0.64$,疾病相关序列识别相似度阈值 $dis\text{-}sim=t=0.80$.

F3SR 算法通过 *k*-mer 编码提取序列统计特征. 为选取合适的参数 *k* 和相似性度量方法,本文通过实验测试 *k*-mer 编码 *k* 取值、数据集序列错误率 *err* 和皮尔逊相关系数(pearson correlation coefficient)、斯皮尔曼等级相关系数(spearman's rank correlation coefficient)、曼哈顿距离(manhattan distance)、肯德尔一致性系数(kendall's consistency coefficient)对 F3SR 算法识别性能的影响. 实验结果如图 4 所示.

在图 4 中,横坐标表示数据集序列的错误率 *err*(%),$err=0$ 表示序列未携带"噪声数据",$err=20$ 表示当前基准数据集中每条序列含有"插入"、"删除"和"替换"错误的比例

[2] https://hpc.gxu.edu.cn
[3] https://github.com/ArashPartow/bloom
[4] https://www.ncbi.nlm.nih.gov
[5] https://tandem.bu.edu
[6] https://www.ebi.ac.uk/gwas

约为20%.图4的结果表明:当$k\geq4$时算法采用pearson相关系数整体上具有最高的识别准确率和识别查准率,最低的识别假阳性率,其次是spearman等级相关系数和kendall一致性系数,算法采用manhattan距离度量相似度时,整体上识别准确率和查准率最低,且具有较高的识别假阳性率.当k取值4~6时,本文算法识别准确率、查准率和假阳性率趋于平稳.为使F3SR算法在保持较低识别假阳性率的同时具有较高识别准确率和查准率,选取$k=5$进行k-mer编码.

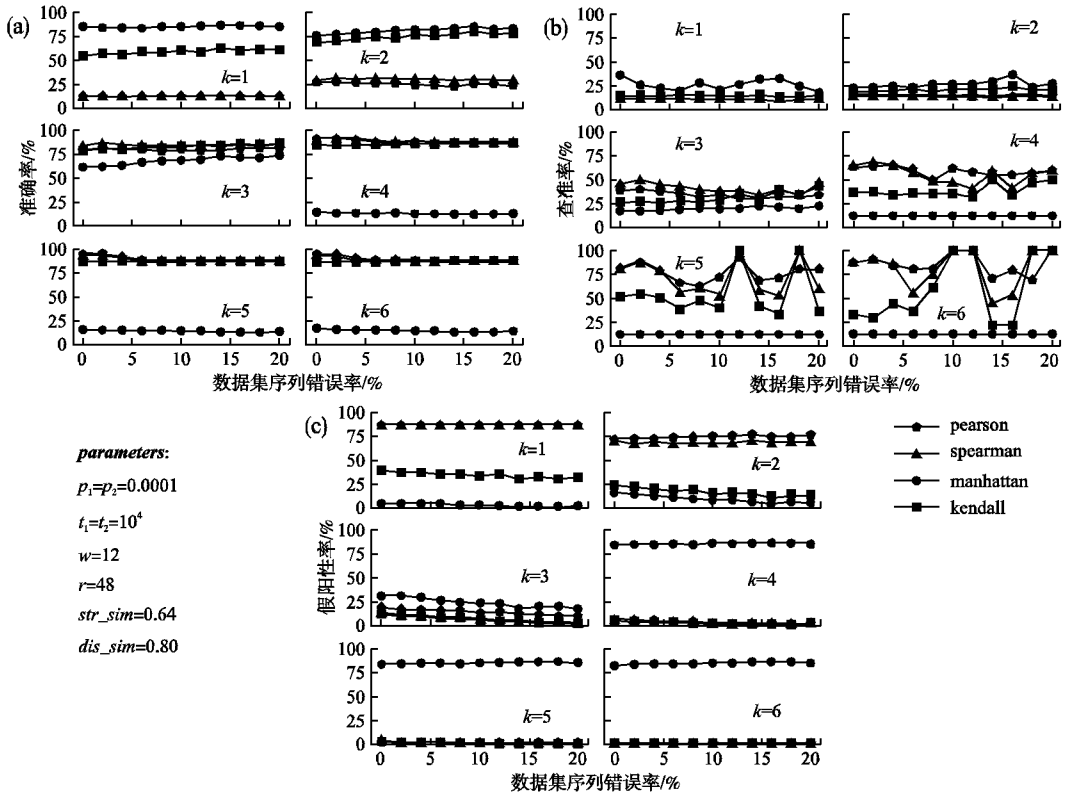

图4　All-Together数据集上错误率、k值和相似性度量方法对算法F3SR的准确率、查准率和假阳性率的影响

Fig.4　Effect of error rate, k, and similarity measures on the *acc*, *pre* and *fpr* of F3SR on the dataset All-Together

3.3　算法比对实验

为评估算法在不同错误率*err*的序列数据集上运行的识别性能,实验测试了算法F3SR和SRF[6]、LRF[7]在All-STR数据集上运行的准确率*acc*、查准率*pre*和假阳性率*fpr*,实验结果如表2所示.

表2实验结果表明,F3SR算法在数据集All-STR上运行的识别准确率均高于算法LRF和SRF,且数据集序列的错误率越低算法的识别准确率越高,即使错误率达到20%时仍具有89.925%的较高准确率.这是因为错误率越低的短串联重复序列在分割成序列片段时,越能保证局部片段的词频统计一致性,通过对局部片段进行相似性度量便可以得到较高的局部相似度值,从而使得算法识别序列中短串联重复序列的准确率较高.F3SR算法在All-STR数据集上运行的识别查准率均高于SRF算法;当All-STR数据集序列的错误率为4%~

表2　算法F3SR、SRF和LRF在All-STR数据集上运行的准确率、查准率和假阳性率

Table 2　Values of *acc*, *pre* and *fpr* of algorithms F3SR, SRF and LRF on the dataset All-STR

算法	F3SR			SRF			LRF		
err(%)	*acc*(%)	*pre*(%)	*fpr*(%)	*acc*(%)	*pre*(%)	*fpr*(%)	*acc*(%)	*pre*(%)	*fpr*(%)
2	**94.676**	**85.641**	1.446	87.768	61.090	**0.470**	88.642	75.561	0.546
4	**94.166**	86.533	1.229	87.272	32.020	**0.432**	93.633	**93.506**	0.458
6	**93.444**	87.020	1.042	87.220	17.431	**0.355**	92.003	**92.492**	0.398
8	**92.914**	**87.734**	0.880	87.214	16.519	**0.357**	88.807	81.464	0.385
10	**92.107**	**87.491**	0.769	87.212	14.203	**0.345**	87.977	68.864	0.394
12	**91.545**	**87.095**	0.704	87.213	15.014	0.349	87.307	20.209	**0.259**
14	**90.961**	**86.760**	0.623	87.194	13.642	**0.363**	87.190	14.457	0.374
16	**90.645**	**86.539**	0.579	87.216	15.021	**0.345**	87.187	13.458	0.370
18	**90.195**	**85.960**	0.526	87.233	14.823	**0.324**	87.174	14.000	0.389
20	**89.925**	**85.563**	0.492	87.247	15.963	**0.312**	87.185	14.569	0.379
平均值	**92.058**	**86.634**	0.829	87.279	21.573	**0.365**	88.711	48.858	0.395

6%时,F3SR 算法的查准率略低于 LRF 算法;当序列的错误率为 2%、8%~20%时 F3SR 算法的查准率高于 LRF 算法. 这表明在高错误率的序列数据集上识别短串联重复序列时,F3SR 算法比其他两个算法更有效. 与算法 LRF 和 SRF 相比,F3SR 算法在 All-STR 数据集上逆行的假阳性率最高;这是因为对短序列局部片段进行相似性度量识别短串联重复序列时,若短序列的局部片段具有极高相似性但并不是短串联重复序列时,算法产生了错误判别.

在 All-STR 数据集上的实验表明,F3SR 算法可以有效识别高错误率的短串联重复序列.

为评估 F3SR 算法是否可以有效识别疾病相关序列,实验测试了算法 F3SR、SRF[6] 和 LRF[7] 在不同错误率 *err* 的序列数据集 All-Disease 上的识别性能,实验结果如表 3 所示.

从表 3 可以看到,当 All-Disease 数据集中序列错误率为 2%~20%时:无论是某种具体错误率情形还是平均情况,F3SR 算法整体上具有最高的识别准确率和查准率,LRF 算法具有第 2 高的识别准确率和查准率,SRF 算法的识别准确率和查准率最低;F3SR 算法在 All-Disease 数据集上识别假阳性率最低,其次是 SRF 和 LRF 算法.

在 All-Disease 数据集上的实验表明,相较于 SRF 和 LRF 算法,F3SR 算法识别疾病相关序列时,具有更高的准确率和查准率,具有最小的识别假阳性率,整体上识别效果较好.

表 3 算法 F3SR、SRF 和 LRF 在 All-Disease 数据集上运行的准确率、查准率和假阳性率

Table 3 Values of *acc*, *pre* and *fpr* of algorithms F3SR, SRF and LRF on the dataset All-Disease

算法	F3SR			SRF			LRF		
err(%)	*acc*(%)	*pre*(%)	*fpr*(%)	*acc*(%)	*pre*(%)	*fpr*(%)	*acc*(%)	*pre*(%)	*fpr*(%)
2	**99.943**	**99.966**	**0.004**	87.307	29.509	0.332	96.427	92.029	0.846
4	92.712	**99.920**	**0.004**	87.281	22.821	0.311	**94.374**	93.953	0.473
6	88.837	**99.844**	**0.002**	87.220	12.871	0.329	**91.046**	85.895	0.697
8	87.972	**99.415**	**0.003**	87.231	13.498	0.318	**88.034**	65.071	0.618
10	**87.735**	**98.003**	**0.005**	87.222	13.196	0.327	87.371	39.966	0.385
12	**87.660**	**97.512**	**0.004**	87.230	13.308	0.319	87.206	17.122	0.370
14	**87.607**	**90.060**	**0.013**	87.213	12.929	0.337	87.156	15.517	0.421
16	**87.595**	**90.964**	**0.010**	87.209	12.158	0.337	87.128	14.198	0.446
18	**87.585**	**94.209**	**0.006**	87.213	11.873	0.332	87.145	14.208	0.426
20	**87.574**	**96.478**	**0.003**	87.247	14.625	0.306	87.148	13.514	0.418
平均值	**89.522**	**96.637**	**0.005**	87.237	15.679	0.325	89.304	45.147	0.510

为评估在既有短串联重复序列又有疾病相关序列的数据集上运行算法的识别性能,本文进一步测试了 3 种算法在序列含有不同错误率 *err* 的真实混合数据集 All-Together 上的算法识别性能,实验结果如表 4 所示.

表 4 算法 F3SR、SRF 和 LRF 在 All-Together 数据集上运行的准确率、查准率和假阳性率

Table 4 Values of *acc*, *pre* and *fpr* of algorithms F3SR, SRF and LRF on the dataset All-Together

算法	F3SR			SRF			LRF		
err(%)	*acc*(%)	*pre*(%)	*fpr*(%)	*acc*(%)	*pre*(%)	*fpr*(%)	*acc*(%)	*pre*(%)	*fpr*(%)
2	**97.030**	**92.801**	0.801	87.562	53.528	**0.408**	92.121	88.627	0.680
4	93.516	90.765	0.681	87.276	28.905	**0.378**	**93.964**	**93.717**	0.465
6	91.386	88.545	0.577	87.220	15.538	**0.343**	**91.576**	**89.658**	0.531
8	**90.706**	**88.330**	0.488	87.222	15.279	**0.339**	88.461	74.778	0.489
10	**90.153**	**87.819**	0.427	87.217	13.769	**0.337**	87.707	60.472	0.390
12	**89.809**	**87.349**	0.391	87.221	14.297	0.335	87.293	20.254	**0.277**
14	**89.462**	**86.834**	**0.351**	87.202	13.338	0.352	87.175	14.966	0.395
16	**89.282**	**86.633**	**0.325**	87.213	13.782	0.342	87.161	13.825	0.404
18	**89.029**	**86.129**	**0.294**	87.224	13.512	0.327	87.161	14.098	0.405
20	**88.874**	**85.765**	**0.274**	87.247	15.377	0.309	87.169	14.076	0.396
平均值	**90.925**	**88.097**	0.461	87.260	19.733	**0.347**	88.969	48.017	0.423

表 4 结果表明,对于各种错误率的序列,与算法 SRF 和 LRF 相比,F3SR 算法在 All-Together 数据集上运行时,整体上获得最高的识别准确率和查准率. 在假阳性率方面,对于错误率 2%~12% 的序列数据集,本文算法略低于 LRF 和 SRF 算法;对于错误率 14%~20% 的序列数据集,本文算法具有最低的假阳性率. 算法 F3SR 相较于 LRF 和 SRF,在错误率2%~20% 的数据集 All-Together 平均识别准确率分别提高 1.96% 和 3.66%,平均查准率分别提高 40.08% 和 68.36%,平均假阳性率分别高了 0.114% 和 0.038%.

综上所述,相较于 SRF 和 LRF 算法,本文算法 F3SR 在 3 个数据集上运行时,整体上具有更高的识别准确率和识别查准率,可有效识别具有潜在相似性的高错误率基因组数据敏感序列.

4 总 结

通过将长序列分割为多条短序列,对每条短序列采取双

布隆过滤方法进行序列去重,可避免对每条短序列的重复计算;采用 *k*-mer 编码策略对短序列进行词频统计可提取具有潜在相似性序列的碱基统计特征,通过改进序列相似度计算比对方法,可有效识别具有高错误率的敏感序列,使得识别算法获得更高的识别准确率和查准率.融合长序列分割、过滤和相似度计算的方法可以有效识别出高错误率敏感序列,但算法需要花费一些时间与第三方数据库敏感序列进行相似度计算比对,这对于序列长度越来越长的大规模测序数据集上运行的时间开销较高.如何设计并行计算方法加速长序列中敏感序列的识别过程,以适应大规模基因组测序数据应用,将是下一步的研究工作.

References:

[1] Bonomi L, Huang Y, Ohno-Machado L. Privacy challenges and research opportunities for genomic data sharing[J]. Nature Genetics, 2020, 52(7): 646-654.

[2] Hekel R, Budis J, Kucharik M, et al. Privacy-preserving storage of sequenced genomic data[J]. BMC Genomics, 2021, 22(1): 1-13.

[3] Zhao Chuan, Zhao Sheng-nan, Jia Zhong-tian, et al. Advances in practical secure two-party computation and its application in genomic sequence comparison[J]. Journal of Cryptologic Research, 2018, 6(2): 194-204.

[4] Gymrek M, McGuire A L, Golan D, et al. Identifying personal genomes by surname inference[J]. Science, 2013, 339(6117): 321-324.

[5] Aziz M M A, Sadat M N, Alhadidi D, et al. Privacy-preserving techniques of genomic data-a survey[J]. Briefings in Bioinformatics, 2019, 20(3): 887-895.

[6] Cogo V V, Bessani A, Couto F M, et al. A high-throughput method to detect privacy-sensitive human genomic data[C]//Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society (WPES), ACM, 2015: 101-110.

[7] Fernandes M. Reconciling data privacy with sharing in next-generation genomic workflows[D]. Luxembourg: University of Luxembourg, 2020.

[8] Gymrek M. A genomic view of short tandem repeats[J]. Current Opinion in Genetics & Development, 2017, 44(1): 9-16.

[9] McLaughlin R L. REscan: inferring repeat expansions and structural variation in paired-end short read sequencing data[J]. Bioinformatics, 2021, 37(6): 871-872.

[10] Liu Q, Tong Y, Wang K. Genome-wide detection of short tandem repeat expansions by long-read sequencing[J]. BMC Bioinformatics, 2020, 21(21): 1-15.

[11] Morishita S, Ichikawa K, Myers E W. Finding long tandem repeats in long noisy reads[J]. Bioinformatics, 2021, 37(5): 612-621.

[12] Kiss S Z, Hosszu É, Tapolcai J, et al. Bloom filter with a false positive free zone[J]. IEEE Transactions on Network and Service Management, 2021, 18(2): 2334-2349.

[13] Li Cong, Yang Xiao-yuan, Wang Xu-an. Efficient verifiable outsourcing decryption ABE with privacy-preserving policy[J]. Journal of Chinese Computer Systems, 2018, 39(9): 1993-1997.

[14] Decouchant J, Fernandes M, Völp M, et al. Accurate filtering of privacy-sensitive information in raw genomic data[J]. Journal of Biomedical Informatics, 2018, 82(1): 1-12.

[15] Fernandes M, Decouchant J, Völp M, et al. Dna-seal: sensitivity levels to optimize the performance of privacy-preserving dna alignment[J]. IEEE Journal of Biomedical and Health Informatics, 2019, 24(3): 907-915.

[16] Lambert C, Fernandes M, Decouchant J, et al. MaskAl: privacy preserving masked reads alignment using intel SGX[C]//Proceedings of IEEE 37th Symposium on Reliable Distributed Systems (SRDS), 2018: 113-122.

[17] Luo Xian-tong, Zhong Cheng, Li Yao. Sensitive alignment for long read with high error rate[J]. Journal of Chinese Computer Systems, 2020, 41(11): 204-210.

[18] Buniello A, MacArthur J A L, Cerezo M, et al. The NHGRI-EBI GWAS catalog of published genome-wide association studies, targeted arrays and summary statistics 2019[J]. Nucleic Acids Research, 2019, 47(D1): D1005-D1012.

[19] Osisanwo F Y, Akinsola J E T, Awodele O, et al. Supervised machine learning algorithms: classification and comparison[J]. International Journal of Computer Trends and Technology, 2017, 48(3): 128-138.

[20] Gelfand Y, Rodriguez A, Benson G. TRDB-the tandem repeats database[J]. Nucleic Acids Research, 2007, 35(Sup. 1): D80-D87.

附中文参考文献:

[3] 赵 川,赵圣楠,贾忠田,等.实用安全两方计算及其在基因组序列比对中的应用[J].密码学报,2018,6(2):194-204.

[13] 李 聪,杨晓元,王绪安.隐私保护的可验证外包属性基解密方案[J].小型微型计算机系统,2018,39(9):1993-1997.

[17] 罗贤橦,钟 诚,黎 瑶.高错误率长序列的高敏感度比对[J].小型微型计算机系统,2020,41(11):204-210.