Вступление

Транспонируемые элементы (TEs) представляют собой повторяющиеся последовательности, разбросанные по всему геному [1]. Они обладают способностью перемещаться из одной позиции в геноме в другую, увеличивая число своих копий [2]. Присутствие этих повторяющихся элементов в геномах в настоящее время признано мощным фактором эволюции и биоразнообразия [3], одомашнивания [3, 4] и вариаций размера генома [5]. В соответствии с механизмом их транспозиции они сначала делятся на два класса: класс I или "ретротранспозоны" и класс II или ‘транспозоны’. Далее они иерархически подразделяются на подклассы, порядки, надсемейства, линии и семейства [6, 7]. Ретротранспозон длинного терминального повтора (LTR) (или LTR-RT), порядок элементов класса I [8], содержит наиболее распространенные элементы в геномах растений [9, 10]. LTR-RT перемещаются по механизму копирования и вставки, используя промежуточную РНК, которая обратно транскрибируется в кДНК и вводится в геном интегразой, кодируемой полными копиями LTR-RT [6, 11]. Они подразделяются на два надсемейства, встречающихся у большого числа эукариот, а именно Ty1/Copia и Ty3/Gypsy [12], которые структурно различаются порядком внутренних кодирующих доменов [13]. Далее они могут быть подразделены на линии в соответствии со сходством их доменов [14].

Методы машинного обучения (ML) были использованы для решения нескольких геномных и эволюционных проблем биологических систем [15]. ML определяется как использование откалиброванных вычислительных алгоритмов, основанных на предыдущем опыте, посредством статистического вывода на основе данных для составления прогнозов в задачах классификации и регрессии [16, 17]. Его основная функция заключается в настройке параметров, необходимых для оптимизации производительности на обучающих данных и последующих входных данных [18, 19].

Некоторые исследования использовали модели ML для идентификации TE, например, Orozco-Arias et al. (2019) [13] рассмотрели использование ML для анализа TEs, a Loureiro et al. (2013) [20], Nakano et al. (2018) [21] и Panta et al. al. (2021) [22] изучали, как повысить точность и производительность классификации TE. Недавно были изучены методы предварительной обработки и схемы кодирования, которые позволяют проводить глубокую классификацию TES [23, 24]. Кроме того, такие программы, как RED [25], PASTEC [26], TEClass [27] и TransposonUltimate [28], применяют методы ML, такие как машины опорных векторов, случайные леса, скрытые марковские модели, K-ближайшие соседи, нейронные сети (NN) и графические модели за счет извлечения признаков, автоматизации процессов и более быстрое выполнение алгоритма.

Чтобы идентифицировать и классифицировать TES, биоинформатики разработали множество инструментов, техник и методов, включая структурную, гомологическую, de novo и сравнительную геномику [29-31]. Наиболее точные инструменты объединяют несколько методов для улучшения своих результатов, таких как LTR-FINDER [32], EDTA [33] и Inpactor версии 1 [34]. Тем не менее, это замедляет выполнение инструментов, особенно в больших геномах, занимая часы или даже дни, что вместе с высокой вариабельностью и избыточностью TES [35] делает невозможной обработку большого объема геномных данных, которые публикуются каждый день [36].

В последние годы было создано и опубликовано несколько наборов данных, состоящих из тысяч TE различных видов, таких как Repbase [37], RepetDB [38], PGSB Plants DB [39] и InpactorDB [40]. Эти наборы данных представляют собой ценные ресурсы для улучшения таких задач, как

обнаружение и классификация TE, и послужили мотивом для предложения и оценки методов ML для получения существенных результатов с точки зрения точности и скорости выполнения этих задач [20, 31].

Хотя хорошие результаты получаются с использованием современных методов ML, таких как обычный NN, недавние достижения показали, что глубокие нейронные сети (DNN) могут достигать лучших результатов, в которых реализованы непараметрические модели, основанные на NN, для адаптации ассоциаций между входными и выходными данными [41, 42]. В этой области было опубликовано несколько архитектур DNN, таких как Полностью подключенная нейронная сеть (FNN) от [21], сверточная нейронная сеть (CNN) с 2D-представлением последовательностей от [42, 43] и 1D CNN для классификации TES на суперсемейства от [44]. Однако ни один из этих NNS не был интегрирован в единое программное обеспечение, которое автоматически обнаруживает и классифицирует TES за относительно короткое время.

Кроме того, ни один из подходов DL или существующих инструментов (за исключением TEsorter [45]) не выполняет задачу классификации LTR-RTS на уровне линии / семейства для получения точных результатов. Здесь мы представляем Inpactor 2, новый метод и инструмент, основанный на четырех NN, которые автоматически обнаруживают и классифицируют LTR-ретротранспозоны. Этот инструмент был выполнен в геномах растений до 2.2

Гб и получает результаты в семь раз быстрее, чем самые современные инструменты, сокращая время выполнения с более чем 20 часов до менее чем трех. Более того, используя геном риса Oryza sativa, Inpactor 2 достигает наилучших показателей точности (96,1%

) и F1-балл (91,9%

) и второе место по специфичности (97,7%), точности (92,7%) и частоте ложных обнаружений (FDR) (7%) среди протестированных здесь инструментов. Inpactor 2 находится в свободном доступе по адресу https://github.com/simonorozcoarias/Inpactor2 . Дополнительную информацию об используемых архитектурах NNs и их гиперпараметрах можно найти по адресу https://github.com/simonorozcoarias/Inpactor2/tree/main/NN_architectures .

Материалы и методы

Наборы геномных данных, используемые для обучения NNS

Inpactor 2 был разработан для выполнения трех основных задач при анализе LTR-RTS: обнаружения, фильтрации и классификации. Каждая задача выполняется другим NN. Таким образом, обнаружение выполняется CNN с именем Inpactor 2_detect, фильтрация с помощью FNN с именем Inpactor 2_filter и классификация с помощью другого FNN с именем Inpactor 2_class. Таким образом, для обучения каждой архитектуре требовались разные наборы данных. Например, Inpactor 2_detect был обучен на наборе данных, состоящем из 70000

последовательности из 50000

основания в длину. Чтобы создать последовательности-мишени (те области, в которых присутствуют LTR-RTS), мы случайным образом получили LTR-ретротранспозон из InpactorDB [40] и поместили его в случайную позицию. Затем оставшийся был заполнен последовательностями, соответствующими другим геномным признакам, полученным из [24] (DOI 10.5281/zenodo.4543904). Для отрицательных последовательностей (тех, в которых нет LTR-RTS)

последовательности геномных признаков, отличных от LTR-ретротранспозонов, были снова получены из [24] (DOI: 10.5281/zenodo.4543904). Этот набор данных состоял из 35000

последовательности с LTR-RTS (целевыми последовательностями) и 35000

последовательности без этих элементов (отрицательные последовательности).

С другой стороны, Inpactor 2_filter был обучен с использованием набора данных с двоичными метками, где ноль соответствовал неповрежденным последовательностям, а единица - нетронутым. Все последовательности, представленные в InpactorDB (67 241 элемент), использовались в качестве интактных элементов, в то время как неинтактные последовательности были взяты из отфильтрованных по [40] и соответствовали (a) последовательностям с LTR-RT, вставленным в другую (из разных суперсемейств или линий), (б) последовательностям длин короче или крупнее, чем те, которые указаны в базе данных Gypsy [46] (с допуском 20%

) и (c) последовательности с TEs класса II, вставленные в LTR-RTS. В общей сложности этот набор данных содержал 105657

последовательности. В качестве функций Inpactor 2_filter использовал k-мерные частоты, используя 1≤k≤6

.

Наконец, Inpactor 2_class был обучен с использованием InpactorDB после извлечения k-мерных частот таким же образом, как Inpactor 2_filter. Этот набор данных включал 67241

LTR-RTS, классифицированные на уровне родословной/семьи. частоты k-мер оценивали в каждой последовательности путем вычисления всех возможных k-мер с максимальной длиной в шесть нуклеотидов и последующего подсчета количества встречений [23, 24].

Сравнительный анализ производительности Inpactor 2

Эталонный геном O. sativa [47-51] был выбран для тестирования программного обеспечения из-за его высококачественной сборки, небольшого размера генома (389 Мб) и качества его генов и аннотаций TEs. Геном O. sativa был идентичен тому, который использовался [33] для сравнения результатов с инструментами бенчмаркинга в этом исследовании. В этой работе использовалась стандартная библиотека v6.9.5, созданная [33] на основе генома O. sativa L. ssp. japonica cv. ‘Nipponbare’ v. MSU7 и RepeatMasker v4.0.8 [52] со следующими параметрами ‘-pa 36 -q -no_is -norna -nolow -div 40 - отсечка 225’.

Кроме того, шесть различных геномов растений (таблица 1) были использованы для тестирования времени выполнения Inpactor 2 путем оценки различных размеров генома и композиций TE. Геномы были загружены из NCBI и проанализированы с помощью Inpactor 2, используя следующие параметры (-m 15000 -n 1000, -i нет, -d нет, -C 1, -c да -a нет), как предложено в [53]. Наконец, EDTA был запущен с теми же геномами, чтобы сравнить время его выполнения с Inpactor 2. EDTA был выполнен с использованием EDTA_raw.py скрипт, –введите ltr и другие параметры по умолчанию.

Таблица 1 Геномы растений, использованные в тестах времени выполнения

Видовой размер сборки (Мб) Регистрационный номер

Arabidopsis thaliana 121.2 GCF_000001735.4

Ориза сатива 379.1 GCF_001433935.1

Coffea canephora 579,4 GCA_900059795

Паслен ликоперсикум 839.1 GCF_000188115.4

Кофе арабика 1126,4 GCF_003713225.1

Зеа Мэйс 2252.8 GCF_902167145.1

Открыть в новой вкладке

Библиотеки LTR-RTS видов, показанных в таблице 1, затем были созданы с использованием Inpactor 2 (с фильтрацией по флагу -c и без нее) и EDTA. Кроме того, были использованы два вида, которые не содержались в данных обучения, такие как Coffea humblotiana [54] и Gardenia jasminoides [55]. Затем эти библиотеки были аннотированы с помощью RepeatMasker и сравнены с долей геномов, соответствующих LTR-RTS, согласно документам, в которых сообщалось о геномах. Для проведения всех экспериментов использовалась рабочая станция с 32-ядерным процессором AMD Ryzen Threadripper 3970X, 128 Гб оперативной памяти и графическим процессором Nvidia RTX 2080 super.

Для оценки производительности Inpactor 2 по сравнению с другим программным обеспечением использовалась методология, аналогичная предложенной в [33]. Сначала для сравнительного анализа были выбраны Inpactor v.1.0 [34], TEsorter v.1.3 [45], Transposon Ultimate v.1.0 [28], LTR_retriever v.2.9 [56] и LTRharvest [57], учитывая их методологии классификации LTR-RTS до уровня суперсемейства. Для каждого программного обеспечения был создан рабочий процесс, первоначально использующий LTR_FINDER v.1.0.7 в качестве детектора LTR-RTs. Затем геном O. sativa был аннотирован с помощью RepeatMasker и были извлечены показатели производительности для каждого рабочего процесса. Оценивались следующие показатели: точность, прецизионность, специфичность, чувствительность, FDR и F1-балл. На рисунке 1 показано схематическое представление показателей бенчмаркинга. В этом исследовании TP, FN, TN и FP представляют собой количество нуклеотидов, принадлежащих к каждой категории (рис. 2).

Рисунок 1

Рабочий процесс процесса бенчмаркинга. Во-первых, LTR_FINDER был выполнен в качестве программного обеспечения-детектора для каждого рабочего процесса. Затем, поскольку TEsorter и TransposonUltimate используют в качестве входных данных файл в формате FASTA, dragpaws v.3.0 (http://dawgpaws.sourceforge.net/man.html) был выполнен для преобразования выходных данных LTR_FINDER в GFF и bedtools v.2.29.2 (https://bedtools.readthedocs.io/en/latest/index.html) был использован для преобразования его в FASTA. Как Inpactor версии 1, так и Inpactor версии 2 использовали LTR_FINDER в своем собственном рабочем процессе. LTR_Retriever был уникальным инструментом, который использует необработанные выходные данные LTR_FINDER. Наконец, используя библиотеку, созданную каждым рабочим процессом, был выполнен RepeatMasker и

сценарий perl с именем lib-test.pl [33] был использован для получения показателей производительности.

Открыть в новой вкладке, загрузить слайд

Рабочий процесс процесса бенчмаркинга. Во-первых, LTR_FINDER был выполнен в качестве программного обеспечения-детектора для каждого рабочего процесса. Затем, поскольку TEsorter и TransposonUltimate используют в качестве входных данных файл формата FASTA, dragpaws v.3.0 (http://dawgpaws.sourceforge.net/man.html) был выполнен для преобразования выходных данных LTR_FINDER в GFF и bedtools v.2.29.2 (https://bedtools.readthedocs.io/en/latest/index.html) был использован для преобразования его в FASTA. Как Inpactor версии 1, так и Inpactor версии 2 использовали LTR_FINDER в своем собственном рабочем процессе. LTR_Retriever был уникальным инструментом, который использует необработанные выходные данные LTR_FINDER. Наконец, используя библиотеку, созданную каждым рабочим процессом, был выполнен RepeatMasker и сценарий perl с именем lib-test.pl [33] был использован для получения показателей производительности.

Рисунок 2

Показатели, оцененные для каждого классификатора LTR-RT. Основано на [33]. TP соответствует истинно положительному, FN - ложноотрицательному, TN - истинно отрицательному и FP - ложноположительному.

Открыть в новой вкладке, загрузить слайд

Показатели, оцененные для каждого классификатора LTR-RT. Основано на [33]. TP соответствует истинно положительному, FN - ложноотрицательному, TN - истинно отрицательному и FP - ложноположительному.

Сценарий под названием ‘lib-test.pl ’, включенный в EDTA toolkit [33], был использован для извлечения шести показателей. Поскольку это исследование было сосредоточено только на категории LTR-RT, сценарий был выполнен с использованием параметра -cat ltr для выполнения сравнительной оценки.

Результаты

Общая архитектура Inpactor 2

Inpactor 2 состоит из четырех NNS, каждый из которых выполняет одну функцию в конвейере. Сначала Inpactor 2 получает сборку генома в формате FASTA, затем каждая последовательность во входном файле разрезается на разделы по 50 кб без перекрытия. Каждый раздел преобразуется в 2D-представление с использованием однократного кодирования. Эта схема кодирования генерирует

матрицу из единиц

и нулей размером 5xn

, где каждая строка соответствует возможным нуклеотидам (A, C, T, G или N), а столбцы представляют основания, присутствующие в последовательности длиной n

. Эта кодировка ставит 1

в позиции строки, которая соответствует основанию последовательности. Пример смотрите в [43] и разделе ‘Определение сети Inpactor 2_k-mers’. Затем CNN с именем Inpactor 2_detect используется для прогнозирования того, какой раздел содержит LTR-RTS, и эти сегменты сохраняются для дальнейшего анализа. У этой сети есть единственная функция сохранения тех участков генома, которые представляют интерес, и устранения тех, которые не содержат LTR-RTS (согласно прогнозам). Таким образом, оптимизируется время выполнения и объем памяти, необходимые для следующих шагов. Затем LTR_FINDER запускается для разделов, которые, как было предсказано, содержали LTR-RTS внутри с помощью Inpactor 2_detect, для поиска начальной и конечной позиций обнаруженных LTR-RTS. Этот шаг выполняется параллельно, чтобы сократить время выполнения. После этого CNN под названием ‘Inpactor 2_k-mers’ используется для подсчета частот k-mer в извлеченных LTR-ретротранспозонах. Этот CNN предназначен для извлечения функций, требуемых следующими NNS конвейера, эффективным по времени способом (см. раздел ‘Определение сети Inpactor 2 K-mers’). Интактные и потенциально полные LTR-ретротранспозоны фильтруются и сохраняются на основе частот k-mer и FNN, называемого ‘Inpactor 2_filter’. Наконец, FNN с именем ‘Inpactor2_Class’ используется для классификации элементов по родословным. На рисунке 3 показана схема общей структуры Inpactor 2.

Рисунок 3

Общее схематическое представление рабочего процесса Inpactor 2. Ядро инструмента состоит из четырех компонентов. Inpactor 2 получает в качестве входных данных геномную сборку, затем он разбивает последовательность на неперекрывающиеся участки длиной 50 кб. В качестве выходных данных Inpactor 2 создает библиотеку в формате FASTA с обнаруженными и классифицированными LTR-RTS и табличным файлом с прогнозами, сделанными тремя NNS. Подробную графическую схему рабочего процесса Inpactor 2 и каждого раздела можно найти на рисунке S1.

Открыть в новой вкладке, загрузить слайд

Общее схематическое представление рабочего процесса Inpactor 2. Ядро инструмента состоит из четырех компонентов. Inpactor 2 получает в качестве входных данных геномную сборку, затем он разбивает последовательность на неперекрывающиеся участки длиной 50 кб. В качестве выходных данных Inpactor 2 создает библиотеку в формате FASTA с обнаруженными и классифицированными LTR-RTS и табличным файлом с прогнозами, сделанными тремя NNS. Подробную графическую схему рабочего процесса Inpactor 2 и каждого раздела можно найти на рисунке S1.

Inpactor 2 может быть выполнен с использованием нескольких циклов (от одного до пяти) анализа, где каждый цикл по-разному разбивает входные последовательности, чтобы предсказать элементы, которые остаются разделенными в любом из разделов. Это поведение контролируется параметром -C (верхний регистр), и его значение по умолчанию равно 1. Кроме того, Inpactor 2 может быть выполнен с использованием различных структурных параметров для фильтрации LTR-RTS, таких как минимальная и максимальная длина LTR-RT, домены LTR, начинающиеся с TG и заканчивающиеся CA, и целевой сайт дублирование (TSD) до и после элемента. Эти параметры задаются с флагами -M, -m, -i и -d, где их значения по умолчанию равны 2000, 28000, yes и yes, соответственно.

После завершения всех циклов Inpactor 2 удаляет немаксимальные предсказания, следуя той же методологии архитектуры Yolo [58]. Оттуда адаптирована аналогичная концепция, а именно операция пересечения по объединению (IOU) (также называемая индексом Жаккарда). Здесь операция IOU между двумя предсказаниями LTR-RT определяется как количество нуклеотидов, перекрывающихся в обоих предсказаниях (пересечение), по сравнению с общим числом нуклеотидов, покрытых в хромосоме обоими предсказаниями (объединение) (рисунок 4). Если два прогноза имеют оценку IOU > 0,6, то Inpactor 2 сохраняет только тот, у которого оценка прогноза наилучшая. Этот балл рассчитывается как среднее значение вероятностей, полученных каждым NN (Inpactor 2_detect, Inpactor 2_filter и Inpactor 2_class). С этими вероятностями можно ознакомиться в выходном файле Inpactor 2 с именем ‘Inpactor 2_predictions.tab’.

Рисунок 4

Операция IOU между двумя перекрывающимися предсказаниями LTR-RT вдоль хромосомной последовательности.

Открыть в новой вкладке, загрузить слайд

Операция IOU между двумя перекрывающимися предсказаниями LTR-RT вдоль хромосомной последовательности.

Оценка по долговым распискам определяется в уравнении 1

$$IOU=\max(0,\min(Y1,X1)-\max(Y0,X0))\max(Y1,X1)-\min(Y0,X0),$$

(1)

где X0 и Y0 - начальные позиции прогнозов 1 и 2, а X1 и Y1 - конечные позиции прогнозов прогнозы 1 и 2 соответственно (рис. 4).

Inpactor 2_detect определение сети

Чтобы сократить время выполнения, Inpactor 2 сначала разбивает входные последовательности на разделы по 50 кб и предсказывает, какой раздел может содержать LTR-ретротранспозоны. Для этого был разработан CNN на основе архитектуры TERL [43]. Inpactor 2_detect состоит из трех сверточных слоев, за каждым из которых следует слой максимального объединения. После сверточных слоев используются два полностью связанных слоя для получения предсказаний для 1000 и 500 нейронов соответственно (рисунок S2). Функцией активации во всех слоях был ReLU, тогда как SGD использовался в качестве оптимизатора, а двоичная перекрестная энтропия использовалась в качестве функции потерь. Сеть была обучена с использованием 100 эпох и размера пакета 64. С показателями производительности Inpactor 2_detect можно ознакомиться в таблице S1.

Определение сети Inpactor 2_k-mers

Предлагаемое здесь программное обеспечение выполняет предварительную обработку информации, хранящейся в извлеченных последовательностях LTR-RT. Эта алгоритмическая обработка включает вычисление k-мерных частот, анализ главных компонент и масштабирование. Хотя существуют биоинформационные алгоритмы для вычисления частот k-мер, в этой статье предлагается CNN для подсчета от 1 до 6 мер в последовательностях ДНК. Однако для выполнения этой задачи последовательность ДНК должна быть сначала преобразована в

цифровую кодировку. Согласно [43], подходящим представлением ДНК является кодирование one-hot, поэтому в дальнейшем используется его модификация. Используемая здесь одноразовая кодировка содержит только пять строк: A (аденин), C (цитозин), G (гуанин), T (тимин) и N (неидентифицированный нуклеотид).

Чтобы понять, как CNN можно использовать для вычисления k-мерных частот, на рисунке 5 рассмотрены два примера. Первый пример вычисляет количество ‘A’ в последовательности ДНК ‘ACTGCCTAA’, тогда как второй пример вычисляет количество ‘CT’ в той же последовательности ДНК. Согласно рисунку 5, веса и смещения могут быть установлены вручную для вычисления частоты любого k-мер. Например, количество определенного k-мер в последовательности ДНК может быть вычислено, если весовая матрица установлена равной 2D-представлению этого k−мер и смещение установлено равным 1-k

.

Рисунок 5

Свертки, используемые для вычислений k-мерных частот.

Открыть в новой вкладке, загрузить слайд

Свертки, используемые для вычислений k-мерных частот.

Учитывая, что фильтр может иметь более одного измерения, их можно использовать для вычисления всех k-мерных частот одновременно, что ускоряет время подсчета по сравнению с обычным методом подсчета (таблица S2). Окончательная архитектура CNN проиллюстрирована на рисунке S3.

Inpactor 2_k-mers принимает математические представления последовательностей ДНК с размерами 5x500x1 и извлекает частоты 1-6-mers в лексикографическом порядке. Следовательно, частоты 41+42+43+44+45+46=5460

k-mers могут быть вычислены с помощью разработанного CNN.

Определение сети Inpactor 2_filter

Для получения интактных LTR-ретротранспозонов были разработаны эксперименты на основе ML. Для этого используются последовательности из InpactorDB [40] с именем class 0 (неповрежденные элементы), тогда как элементы, которые были удалены в каждом из фильтров, предложенных в том же исследовании, были взяты как неповрежденные последовательности с именем class 1.

Используя полученный набор данных, был обучен и настроен NN, основанный на FNN, предложенном в [21]. Разработанная архитектура, получившая название Inpactor 2_filter, показана на рисунке S4. Для каждого из слоев были использованы отсев, равный 0,5, и функция повторной активации с использованием BatchNormalization с импульсом 0,99. Для уровня прогнозирования была использована функция активации softmax. Алгоритм оптимизации Adam был применен для

поиска подходящей конфигурации NN, а в качестве функции потерь использовалась категориальная перекрестная энтропия. Кроме того, регуляризация l1 была добавлена к ядру со значением 0.0001, а l2 - к смещению со значением 0.01. Эти регуляризаторы были применены к трем скрытым слоям. Для этапа обучения было использовано 200 эпох, а размер пакета был установлен равным 128.

Показатели производительности приведены в таблице S1 для оценки правильного обобщения модели.

Определение сети Inpactor 2_class

После обнаружения и фильтрации LTR-RTS были классифицированы по линиям. FNN был разработан с тремя скрытыми слоями, каждый из которых содержит 200 нейронов. Структура Inpactor 2_class аналогична структуре Inpactor 2_filter (рисунок S4), но с 13 выходными нейронами для реализации многоклассовой классификации. Эта сеть была обучена для 200 эпох с размером пакета 128, используя последовательности InpactorDB [40], преобразованные в частоты k-мер. Inpactor 2_class получил результат 0,98 по точности, прецизионности, отзыву и баллу F1 (таблица S1).

Inpactor 2_utils

В дополнение к основному компоненту Inpactor 2 был выпущен дополнительный скрипт, содержащий утилиты для анализа LTR-RT, такие как удаление символов, отличных от нуклеотидов (A, C, T, G или N), вычисление частот k-мер с $1 \leq k \leq 6$