Алгоритмы в биоинформатике, НИУ ВШЭ / Домашняя работа 2

Студент - Катерина Олейникова

Преподаватели - Миронов Андрей, Сергей Спирин

7 Марта 2022

Примечание Для решения заданий был использован Python. Код приведен в конце отчета.

Задание 1 1. Скачать последовательность генома *E.coli*.
2. Выделить 1000 фрагментов размером 4000 нуклеотидов (можно с фиксированным шагом, можно с помощью датчика случайных чисел).
3. Для каждого фрагмента построить бернуллиевскую модель и марковские модели 1, 2, 3 порядков.
4. Для каждой модели оценить AIC и BIC.
5. Сделать вывод, какая модель лучше описывает последовательности.
6.* Попробовать увеличить порядок модели и понять, какой порядок модели оптимален.

Решение.

Сперва был скачан геном Escherichia coli (кишечная палочка) - грам-отрицательные палочковидные бактерии, принадлежащие к семейству Enterobacteriaceae, роду Escherichia (эшерихия).
Затем с помощью Biopython и пакета random было выделено 1000 фрагментов размером 4000 нуклеотидов из скачанного генома случайным образом.

Самым простым мотивом (то есть относительно короткая характерная последовательность нуклеотидов (в ДНК, РНК) или аминокислотах (в белках), которая имеет существенное биологическое значение) последовательности ДНК является *слово*. Определенное распределение частот таких слов позволяет охарактеризовать развивающуюся последовательность и установить, как часто или редко данное слово появляется в последовательности генома. Таким образом, в данной части задания мы сосредоточимся на распределении частот слов.

Для каждого фрагмента необходимо было построить бернуллиевскую модель и марковские модели 1, 2, 3 порядков.

Бернуллиевская модель является самой простой моделью генерации последовательностей. Для этого необходимо задать некоторое распределение частот букв $p(\alpha)$, после с помощью генератора случайных чисел выбирается буква и помещается в последовательность (так продолжается n количество раз). В результате мы получим последовательность букв, которая является случайной последовательностью.
При условии, что мы возьмём все возможные последовательности данной длины L, то бернуллиевская генерация последовательности в целом состоит из двух этапов – 1) выбора длины последовательности и 2) генерации L символов (взято из книги А.А.Миронова "Биоинформатика последовательностей"). Хотя эта процедура и может показаться простой (возможно, даже слишком абстрактной), но, на самом деле, она весьма эффективна для достижения аналогичного распределения частот слов как у реальных последовательностей.

Процесс конструирования бернуллиевской модели с использованием рандомно сконструированных 1000 последовательностей одинаковой длины (4000 пар оснований) реализован в Python с помощью библиотеки math (как и следующие нижеизложенные марковские модели 1-ого, 2-ого и 3-его порядков).

Марковская модель генерации последовательностей заключается в следующем: даны распределения вероятностей для первых символов $q(\alpha)$; для дальнейшей генерации есть переходные вероятности $p(\beta|\alpha)$. Затем задаемся длиной последовательности L и генерируем первый символ из распределения $q(\alpha)$. После, исходя из текущего символа, генерируем следующее состояние с использованием распределения $p(\beta|\alpha)$.

Сперва нам надо построить марковскую модель 1-ого порядка - это такая модель, в которой очередная буква зависит только от одного предыдущего символа.

Для моделей высшего порядка (т.е., например, 2-ого и 3-его порядка) марковская модель первого порядка легко обобщается: когда генерация следующего символа зависит от нескольких предыдущих символов.

После того, как модели построены, необходимо было рассчитать для каждой из них AIC и BIC (шаг "Оценка моделей"из книги А.А.Миронова "Биоинформатика последовательностей").
Имеется некоторый набор последовательностей и несколько моделей. Для оценки качества модели часто используют информационные критерии (именно вышеперечисленные AIC и BIC).

Информационный критерий Акаике (AIC) — критерий, применяющийся исключительно для выбора из нескольких статистических моделей. Этот критерий в 1971 году был разработан Хироцугу Акаике.
AIC вычисляется по формуле:

$$AIC(M) = 2k - 2ln(\hat{L}),$$

где k - число параметров модели,
$\hat{L}$ - оптимальное значение правдоподобия модели.

Среди всех моделей отбирают такую модель, которая имеет минимальное значение AIC, что обеспечивает баланс между качеством описания ($\hat{L}$) и количеством параметров.

В данной задаче для рассмотренных моделей получились значения AIC:

Бернуллиевская модель - AIC = 11052019,426;
Марковская модель 1-ого порядка - AIC = 11410766,926;
Марковская модель 2-ого порядка - AIC = 10522520,553;
Марковская модель 3-его порядка - AIC = 10398838,799.

Основываясь на несколько других предположениях, в 1978 году Шварцем был предложен Байесовский информационный критерий (BIC) (поэтому часто он называется также критерием Шварца (Schwarz criterion (SC)). Он разработан исходя из байесовского подхода и является наиболее часто используемой модификацией AIC.
В этом критерии учитывается n – размер обучающей выборки:

$$BIC(M) = k \cdot ln(n) - 2ln(\hat{L}).$$

Как видно из формулы, данный критерий налагает больший штраф на увеличение количества параметров по сравнению с AIC, так как ln(n) больше 2 уже при количестве 8 наблюдений.

В данной задаче для рассмотренных моделей получились значения BIC:

Бернуллиевская модель - BIC = 11052034,149;
Марковская модель 1-ого порядка - BIC = 11410840,542;
Марковская модель 2-ого порядка - BIC = 10522829,741;
Марковская модель 3-его порядка - BIC = 10400090,277.

Вычисления AIC и BIC осуществлены в Python.

В заключение можно сделать вывод, что марковская модель 3-его порядка дает наивысшую оценку (наименьшие значения AIC, BIC) и лучше всего описывает последовательности.

* Дополнительно были рассмотрены марковские модели 4-8 порядков (код реализован в Python), и выявлено, что лучше всего описывает последовательности марковская модель 7-ого порядка, имеющая самые меньшие значения AIC и BIC (AIC = 1524445,284 и BIC = 1846075,0263).

Algorithms in Bioinformatics / Homework 2

```
[1] !pip install biopython
```

```
Collecting biopython
  Downloading biopython-1.79-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (2.3 MB)
     |████████████████████████████████| 2.3 MB 3.2 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from biopython) (1.21.5)
Installing collected packages: biopython
Successfully installed biopython-1.79
```

1. Download E. coli genome:

a wide range of genomes (and its path via ftp ncbi) can be found here: https://gitlab.phw-pengu.com/a-gaskin/test-repo/-/blob/81504eaadf6e86826ed24574083c699e88b0e6c2/20191119-refdat-results/refdat-csv/ncbi-bacterial-reference-database.csv

I selected this genome of E.coli: ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/003/018/375/GCF_003018375.1_ASM301837v1 and download it.

2. Select 1000 fragments from this genome with the size of 4000 nt:

```
[2] from Bio import SeqIO
    import random # i decided to randomly select them

    seqs=[]
    for record in SeqIO.parse("GCF_003018375.1_ASM301837v1_genomic.fna", "fasta"):
      for i in range(1000):
        a=random.randrange(0, len(record.seq)-4000, 1)
        seqs.append(record.seq[a:a+4000])
      break
```

```
[3] seqs[:5]
```

```
[Seq('ACAGCTACGCGCGGAAGTTTCTGATGAACTCGCAATGGTGTCCCGATCGCTGCG...ATA'),
 Seq('CATGTAGCGCATCAAAGGCTGCCTGCCAGAACAGGCTGTCGCCCTGCTCTGCAA...TTT'),
 Seq('TACTTTGGTATGAACAAAAAAAGAAAAATACAACGCGCGGGTGAGTTATTAAAA...AAT'),
 Seq('TGACTCTGCTGATACAGGAACTGAAGCGATGATCCCGTTACCTTCCGGGACCAA...TGA'),
 Seq('CGTCATGATATTTACGAAGCACTGGTTGAAATTAATCGATTTGAACCTGAACGT...AAT')]
```

```
[ ] len(seqs) # to check how many fragments are
```

```
1000
```

```
[5] len(seqs[5]) # to check each fragment's length
```

```
4000
```

3-6. For each fragment it is needed to construct Bernoulli model, first/second/third/fourth/fifth/sixth/seventh/eighth-order Markov models plus for each model to estimate its AIC and BIC.

```
[8] import math

    for model in range(9):
      # 0 - Bernoulli # for each nucleotide
      # 1 - first-order Markov # each nucleotide depends on its predecessor
      # 2 - second-order Markov # each nucleotide depends on immediate two predecessors
      # 3 - third-order Markov ..
      # 4 - fourth-order Markov ..
      # 5 - fifth-order Markov # calculates the probability of the sixth base based on the previous five bases in the sequence.
      # 6 - sixth-order Markov ..
      # 7 - seventh-order ..
      # 8 - eighth-order ..
        p = 0 # subword frequency distribution
        for seq in seqs:
            subword = ""
            for nt in seq:
                if len(subword)<(model+1):
                    subword=subword+nt
                if len(subword)==(model+1):
                    if model==0:
                        p=p+math.log(seq.count(subword)/4000) # spectrum of the log of the spectrum
                    else:
                        p=p+math.log(seq.count(subword)/(4000-model))-math.log(seq.count(subword[:(len(subword)-1)])/(4001-model))
                    subword=subword[1:]
        print("Model "+ str(model))
        print("AIC",2*4**(model+1)-2-2*p)
        print("BIC",(4**(model+1)-1)*math.log(1000)-2*p)
```

```
Model 0
AIC 11052019.426514272
BIC 11052034.149780108
Model 1
AIC 11410766.925871061
BIC 11410840.542200245
Model 2
AIC 10522520.552804401
BIC 10522829.741386976
Model 3
AIC 10398838.799083028
BIC 10400090.276679168
Model 4
AIC 9719786.374643376
BIC 9724807.008293774
Model 5
AIC 7383324.703816164
BIC 7403421.9616835965
Model 6
AIC 3804830.0614640927
BIC 3885233.816199657
Model 7
AIC 1524445.2841051335
BIC 1846075.0263132278
Model 8
AIC 962745.5378757138
BIC 2249279.229973928
```