Projekt 2

Aleksander Milach

13/06/2020

Opis problemu, cel działania programu

Załóżmy, że mamy ciąg $x_1 = (x_{11}, ..., x_{1w})$ długości w, gdzie każdy element $x_{1i} \in L = \{A, C, G, T\}$. Dla wygody niech dalej $\{A, C, G, T\} = \{1, 2, 3, 4\}, d = |L| \ (= 4)$. Dane są rozkłady $\theta_j = (\theta_{1,j}, ..., \theta_{d,j})^T, j = 1, ..., w$. Oznacza to, że ciąg x_1 jest realizają zmiennej losowej $X = (X_1, ..., X_w)$, gdzie

$$P(X_j = l) = \theta_{l,j}, \quad l = 1, ..., d \quad (l \in \{A, C, G, T\}).$$

Równoważnie $\theta_{a,j}$ jest prawdopodobieństwem tego, że na j-tej poycji występuje liczba (literka) l, oraz w kolejnych kolumnach macierzy θ, zawarte są rozkłady prawdopodobieństwa wystąpienia literek na kolejnych pozycjach ciągu x_1. Zatem dla danej macierz θ prawdopodobieństwo otrzymania ciągu $x_1 = (x_{11}, ..., x_{1w})$ wynosi

$$P(X = x_1; \theta) = P(x_1, \theta) = \prod_{i=1}^{w} \theta_{i,x_{1i}}.$$

Przypuśćmy teraz, że mamy k takich niezależnie wygenerowanych ciągów, tj. $x_1 = (x_{11}, ..., x_{1w}), ..., x_k = (x_{k1}, ..., x_{kw})$. Dodatkowo ustalmy nieznany ciąg $z = (z_1, ..., z_k) \in \{0, 1\}^k$, gdzie z_i są niezależnymi zmiennymi losowymi oraz

$$P(z_i = 1) = \alpha, \quad P(z_i = 0) = 1 - \alpha,$$

a $\alpha \in (0, 1)$ jest znanym parametrem. W przypadku, gdy $z_i = 1$, losujemy x_i w powyżej przedstawiony sposób, jednak gdy $z_i = 0$, losujemy $x_i = (x_{i1}, ..., x_{iw})$, gdzie x_{ij} są niezależne o tym samym rozkładzie

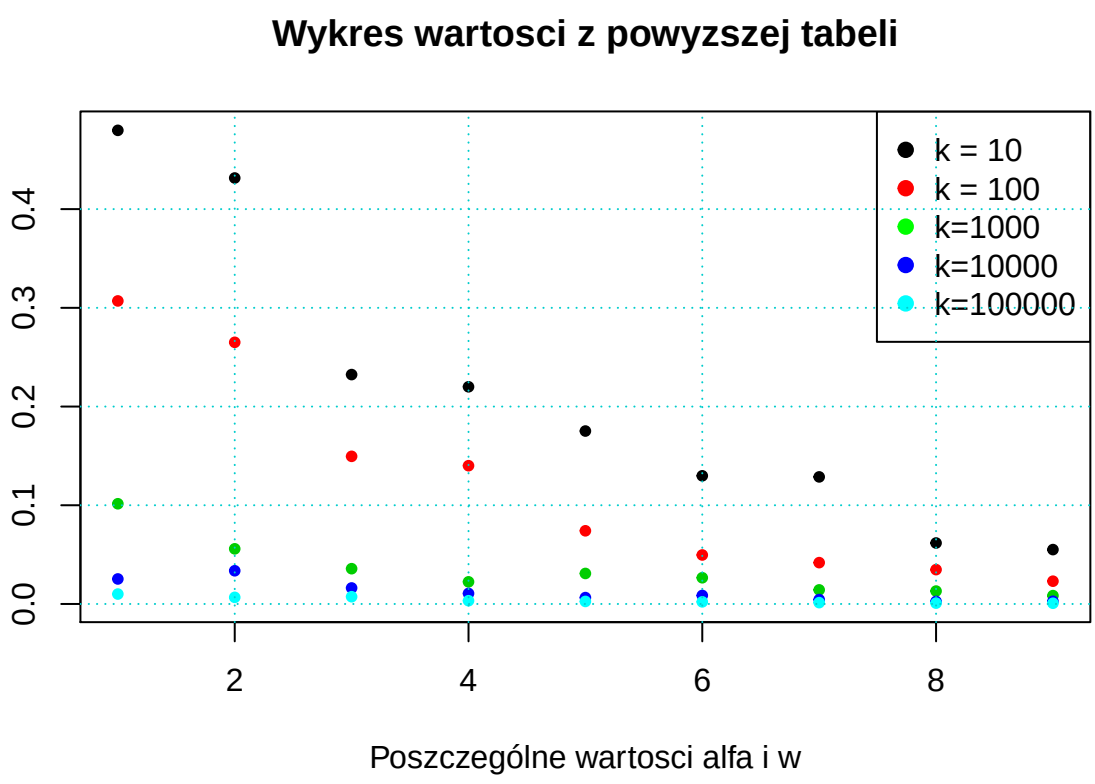
$$\theta^b = (\theta^b_1, \theta^b_2, \theta^b_3, \theta^b_4) \quad (= \theta^b_A, \theta^b_C, \theta^b_G, \theta^b_T).$$

Zauważmy, że w tym przypadku rozkład nie zależy od pozycji w ciągu.

Naszym zadaniem jest najpierw wygenerować dane, spełniające powyższe warunki, a następnie wyestymować z nich θ oraz θ^b. Do tego celu posłużą trzy programy: json_save_params.py, dzięki któremu zapiszemy odpowiednie parametry w przyjaznym dla Pythona formacie .json; motif_291008_generate.py, który z uprzednio zapisanych danych wygeneruje odpowiedni zbiór danych, to jest k ciągów x_i długości w dla odpowiedniej α i zależnego od niej ciągu z w końcu motif_291008_estimate.py, który zawiera właściwą estymację parametrów przy pomocy algorytmu EM, który umożliwia coraz lepsze oszacowanie parametrów w takim problemie.

Wyniki i komentarze

	k = 10	k = 100	k = 1000	k = 10000	k = 100000
w = 10, a = 0.2	0.4798062	0.3069875	0.1015063	0.0253312	0.0100562
w = 5, a = 0.2	0.4315500	0.2650750	0.0558688	0.0336375	0.0067000
w = 3, a = 0.2	0.2323875	0.1495500	0.0357250	0.0161938	0.0073375
w = 10, a = 0.5	0.2200125	0.1400583	0.0224250	0.0106417	0.0032958
w = 5, a = 0.5	0.1752000	0.0741292	0.0309125	0.0064625	0.0025625
w = 3, a = 0.5	0.1298250	0.0496333	0.0265500	0.0086208	0.0021417
w = 10, a = 0.8	0.1287318	0.0418773	0.0142636	0.0044250	0.0013455
w = 5, a = 0.8	0.0617591	0.0347795	0.0130773	0.0025614	0.0008318
w = 3, a = 0.8	0.0550750	0.0230591	0.0084523	0.0027318	0.0007341



Skupiłem się na różnicowaniu parametrów α i k, gdyż intuicyjnie te parametry będą miały większy wpływ na dokładność estymacji, niż długość ciągu czy rzeczywiste wartości parametrów θ i θ^b. Dobroć estymacji parametrów mierzyłem sumą wartości bezwzględnych różnic między wyestymowaną a prawdziwą wartością każdego z parametrów podzieloną przez ich ilość $(4(w+1))$. Dzięki temu dla większych w wartości nie będą zawyżone. Pomimo tego dla większych w te wartości są nieco większe. Kolejność wierszy w tabeli nie jest przypadkowa. Okazuje się, że najsilniejszy wpływ na dokładność algorytmu ma wielkość parametru α - im większa tym lepiej. Na przykład dla $w = 3, \alpha = 0.8$, na próbie 1000 elementowej jesteśmy bliżej prawdziwych wartości parametrów, niż dla próby 100000 i $w = 10, \alpha = 0.2$. Można zauważyć, że dla nie tak małych już wartości k = 100, algorytm wciąż jest dosyć daleko od poprawnej estymacji i dopiero dla znacznie większych prób estymacja jest już bardzo dokładna - największe różnice są zauważalne między $k = 100$, a $k = 1000$. Dla $\alpha > 0.2$ różnica, między $k = 10000$ a $k = 100000$ jest już zaniedbywalna. Również szybkość zbieżności algorytmu jest powolna, z własnego doświadczenia zauważyłem, że zanim dopisałem do programu warunek, w którym ma zatrzymywać wykonywanie algorytmu, gdy kolejne estymacje róznią się o dostatecznie mało, tylko zwyczajnie wykonywałem algorytm ustaloną liczbę razy, to dla 10 wywołań wyniki były tak dalekie od oczekiwanych, że spodziewałem się, że wynika to z błędnego kodu. Okazało się tymczasem, że kod jest prawidłowy, jednak dopiero po ponad stu iteracjach algorytm zbliża się do oczekiwanych wyników, a po jeszcze większej ich liczbie są już powody, żeby go zatrzymywać, bo jest już blisko wartości granicznych.