On consecutive alignment with priority in random sequence comparison

Riho Klement and Jüri Lember

University of Tartu, Estonia, riho.klement@ut.ee

Keywords: random sequence comparison, longest common subsequence, suboptimal alignments

One way to measure similarity of two sequences is to calculate the length of the longest common subsequence (LCS). Already with quite short sequences, calculating the length of the longest common subsequence can be too time-consuming. So instead of that we try to find some suboptimal alignments which have similarity scores not far away from the optimal score (length of LCS), but are less resource-demanding to compute.

In this talk we consider an alignment referred to as *consecutive alignment with priority*. Let $X^n = (X_1, \dots, X_n)$ and $Y^n = (Y_1, \dots, Y_n)$ be two mutually independent sequences which both have independent and identically distributed elements taking values from some finite alphabet $\mathcal{A} = \{v_1, v_2, \dots, v_k\}$. Let $P^x = (P_1^x, P_2^x, \dots, P_k^x)$ be the distribution of X_i and $P^y = (P_1^y, P_2^y, \dots, P_k^y)$ the distribution of Y_i. We define $l_i = P_i^x \wedge P_i^y$ and order our letters decreasingly by l_i. Now we take the letters with the highest l_i, let's say v_1. We take the first v_1 in one sequence and align it with the first v_1 in another sequence. Then we align the next pair of v_1-s and so on until we have one sequence with no more v_1-s left. After that we take the letter with the next highest l_i, say v_2, and align them in the same way as first letters, but we keep in mind that we don't ruin the alignment we already have (that is we can align v_2-s only between already aligned pairs of v_1-s). We continue in the same manner until all letters $v_1, \dots, v_k$ are aligned.

Our goal is to calculate the average score of the alignment described above and to prove the large deviation inequality for that score.