Binary classification of metagenomic samples using discriminative DNA superstrings

Karel Jalovec[1] and Filip Železný[1]

[1]CTU in Prague FEE Dept. of Computer Science and Engineering

Motivation

Increasing amount of data obtained by the NGS technologies increases the urge of effective analysis of this data. This work presents a tool for binary classification of metagenomic samples. Metagenomic samples consist of a large amount of short DNA strings (also called reads), which belong to different organisms present in an environment from which the sample was taken. Behavior of an environment can be affected by the contamination by the organisms, which originaly do not belong in this environment. The goal of this work is to develop a classification method based on DNA superstrings that can accurately classify metagenomic samples. Classifiers obtained by this method can be used for determining whether newly obtained metagenomic samples are contaminated (positive) or clean (negative) without the need of identification of particular organisms present in the sample. We want to achieve this goal by establishing a modified sequence assembly task for finding the most discriminatory DNA superstrings.
We assume that standard a approach for this kind of analysis would be to assemble all the samples and try to find the most discriminatory motifs. Both tasks are very computationally demanding. Our method should solve both these tasks simultaneously.

Related problems and preliminaries

Shortest common superstring - SCS

Given a set $S = \{s_1, s_2, ..., s_n\}$ of short strings s_i, determine the shortest string R such that each string $s_i \in S$ is a substring of R.

Consistent superstring - CSS

The most common variants are LCSS (Longest CSS) and SCSS (Shortest CSS). Given a set of positive strings $P = \{p_1, p_2, ..., p_n\}$ and a set of negative strings $N = \{n_1, n_2, ..., n_m\}$, determine a (longest/shortest) superstring R such that each positive string $p_i \in P$ is a substring of R and each negative string $n_j \in N$ is not a substring of R.

Suffix-prefix concatenation

This function (denoted by the $\circ$ symbol) takes two strings as an input. If a suffix of one string matches a prefix of the second string, this operation concatenates the unmatched portion of the second string to the end of the first string. In case of multiple possible concatenations, it is usually considered only the solution with the longest overlap.

String coverage

String r covers certain percentage of string s when r is a substring of s. The portion of symbols of string s matched by symbols of string r are covered. This function can be extended to measure the percentage of string s which is covered by a set of strings $R = \{r_1, r_2, \ldots, r_n\}$.

Problem definition

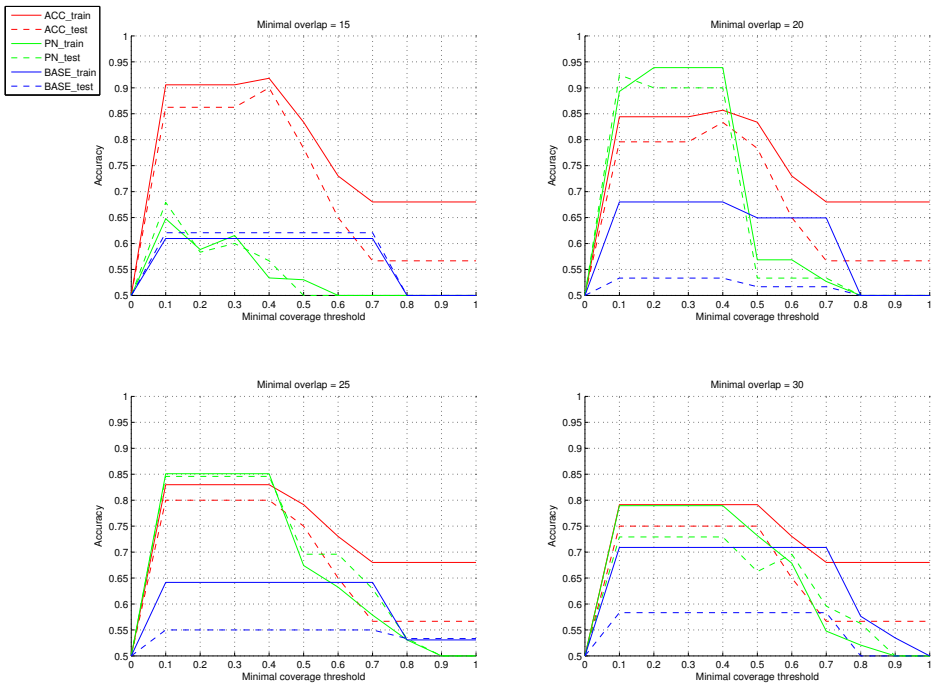
Consider two multisets of reads $S^N = \{\{s_1, s_2, \ldots, s_n\}\}$, $S^P = \{\{s_1, s_2, \ldots, s_m\}\}$ and two powersets 2^{S^N}, $2^{(S^P \cup S^N)}$. Metagenomic sample S_a can be either positive (contaminated) $S_a^+ \in 2^{(S^P \cup S^N)}$ or negative (clean) $S_a^- \in 2^{S^N}$. We are presented by a training set of positive samples $S^+ = \{S_1^+, S_2^+, \ldots, S_k^+\}$ and a set of negative samples $S^- = \{S_1^-, S_2^-, \ldots, S_l^-\}$. The goal is to find a permutation p such that $p = p_1, p_2, \ldots, p_{|p|}$ and $p_{i \in 1..|p|} \in S^P$. Superstring λ_p can be created from permutation p by using the operation suffix-prefix concatenation $\lambda_p = p_1 \circ p_2 \circ \ldots \circ p_{|p|}$. A trained classifier consists of a superstring and a coverage threshold value Θ. The classifier computes coverage of its superstring from the reads contained within the sample and compares it to its coverage threshold. If the computed coverage is higher than the coverage threshold value, the sample is marked as positive. If the coverage is insufficient, the sample is marked as negative.

Methods

To obtain the optimal solution, it would be necessary to search all possible permutations of reads from all the samples. This would be nearly impossible given the fact, that a single metagenomic sample can contain tens of millions of reads. To prune the search space we use a graph-based approach that is very commonly used for sequence assembly tasks. We construct an overlap graph formed by all strings from all samples. To speed up the construction process, we use the FM-index structure[1]. Each vertex in the graph is annotated by the PN-measure value. PN-measure value of the read is computed as a difference between its positive and negative occurrence counts. More positive occurrences of the read result in a higher PN-measure value. We propose to use beam search to discover the most discriminatory paths in the graph.
We have developed three types of classifiers. The first classifier (BASE) serves as a baseline solution. All samples were assembled using the ReadJoiner[2] assembler in the first stage and in the second stage, the most discriminatory motif was found using the MEME toolkit[3]. Both the second (PN) and the third (ACC) classifiers use beam search to find the most discriminatory paths in an overlap graph during the training stage. Initial beam of the algorithm is filled with elementary paths that consist of a single vertex. Vertices for elementary paths are selected with respect to their PN-measure values. The algorithm selects the most discriminatory path in the beam in each step and tries to extend it. Path extension adds new vertices to the left, to the right or to the both sides of the path. If there is more than one extension available all of them are added into the beam. Original path is marked as inextensible and also kept in the beam. The reason for keeping the original path in the beam is that the original path can be more discriminatory than all its extensions. Marking the path as inextensible prevents the algorithm from recurring extension of the previously extended path. The algorithm stops when all the paths in the beam are marked as inextensible. The PN and the ACC classifiers differ in the path feasibility evaluation process. The PN classifier computes cumulative PN-measure value of the path. The PN-measure values of the reads along the path are summed up and the higher the cumulative PN-measure value is, the preferable the path is. This leads to a faster but less accurate search. The ACC classifier evaluates the feasibility of each path in the same way as the classification procedure itself works. Extended path is converted into the superstring by concatenation of its reads and the accuracy value of this superstring is computed. This leads to a slower but more accurate search.

Results

We tested the performance of our classifiers with a simulated data set. We randomly selected four organisms for our simulated environment and created a certain amount of samples by individually sequencing all the genomes and putting them together. Positive samples were contaminated with an additional organism. We run our algorithm with four different values of a minimal overlap threshold for creating an overlap graph. The performance of the classification is depicted in the image below. The beam width has fixed value of 15.



Accuracy of classifiers w.r.t minimal coverage threshold

The time requirements of the algorithm with respect to the amount of data are described below. Learning was performed on two training sets consisting of 90k and 150k reads. Learning from the smaller dataset took the algorithm between 5 and 10 seconds for the ACC classifier and about 2 seconds for the PN classifier. Larger dataset required 15 to 80 seconds for the ACC classifier and about 5 seconds for the PN classifier. Learning of the baseline classifier using the meme toolkit requires a time constant for which the tool searches for discriminatory motifs. We selected a constant with a value of 300 seconds. Classification procedure requires miliseconds for classifying a single sample consisting of 30k to 40k reads.

Future improvements

So far, our method is able to find only a single discriminatory superstring. Our plan is to extend the method so that each classifier can contain more than a single discriminatory superstring which will lead to a better classification accuracy. Next improvement we plan to incorporate in our method is to increase the biological interpretability of the discriminatory superstrings. So far we did not pay attention whether the superstring is a biologicaly valid genomic sequence.

References

[1] Simpson and Durbin: **Efficient construction of an assembly string graph using the FM-index**. Bioinformatics 26:i367–i373, 2010, http://dx.doi.org/10.1093/bioinformatics/btq217.

[2] Gonnella and Kurtz: **Readjoiner: a fast and memory efficient string graph-based sequence assembler**. BMC Bioinformatics 2012 13:82.

[3] T. L. Bailey et al.: **MEME SUITE: tools for motif discovery and searching**. Nucleic Acids Research, 2009, Vol. 37, Web Server issue doi:10.1093/nar/gkp335

Acknowledgments

- This work has been supported by the Czech Science Foundation project P202/12/2032.

- Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.