Analysis of Svhip run time (for version 0.35)

Christopher

November 13, 2020

1 Data

For this preliminary run time comparison of pipeline segments, 5SrRNA alignments from the Rfam data based were used as input data sets. Alignments were of length 130 nt and consisted of 50, 100, 150, 200, 250, 300 sequences. These parameters were chosen as they best represent the average range per single input alignment that Svhip was designed for.

2 Methods

Time stamps were taken at six distinct steps in the program pipeline, which are listed as follows:
(1) Preprocessing of input file. This refers to the opening of the file, ungapping of sequences, removing all sequences with a base pairwise identity ≥ 0.98 and realigning.
(2) Generation of alignment windows, in this experiment with a target window size of 120 nt and 2..15 sequences per window, as well as a randomly chosen target sequence identity between 0.5 and 0.98.
(3) Generation of control set alignments, based on alignment windows using SISSIz.
(4) Structural conservation filter, consisting of the explicit calculation of a tree representation of structure for every sequence, calculating each pairwise tree edit distance, estimating a cutoff (k-value: 0.25) and filtering the native alignment windows accordingly.
(5) Calculation of feature vectors for any positive and negative training instances remaining after filtering.
(6) Hyper parameter optimization and classifier training based on the feature vectors generated in the previous segment.

As a note on hardware used, the machine used for testing ('bloodymary' in the IZfB building, room 320.6) has 16 GB of RAM. CPU configuration is currently unknown but will be added once determined.

3 Results

The full computation time for all six test sets is illustrated in figure 1. Visually analyzing the curve suggests, that only the first two steps of the pipeline have a direct correlation to the number of sequences. The upper two thirds of each graph (step ’control alignment generation’ to ’hyper parameter search’) seem to be mostly identical but shifted along the x axis according to the time performance of steps 1 and 2. If this hypothesis holds true, it would suggest that a preemptive subdivision of input alignments in smaller blocks of ¡200 sequences is optimal for computation efficiency.

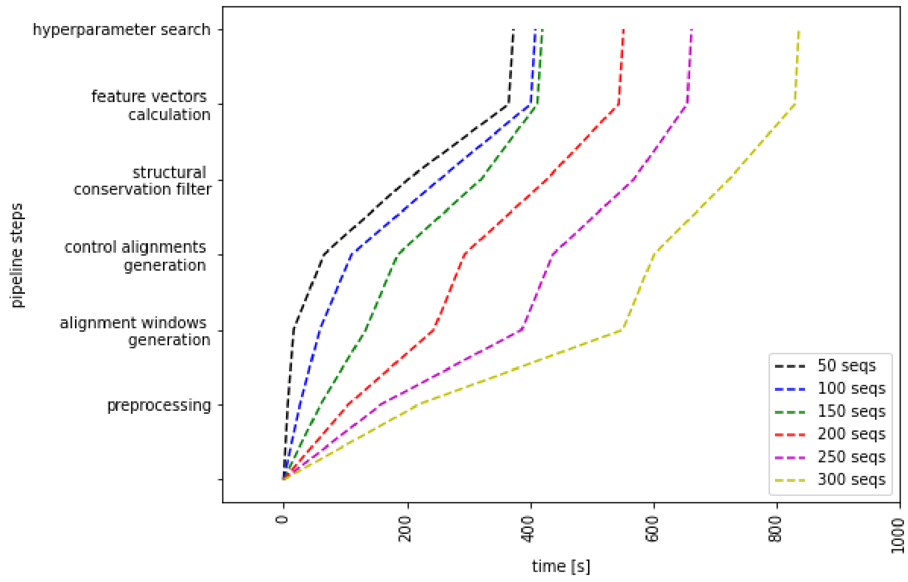

Figure 1: Computation time needed for six separate steps in the Svhip software pipeline using six different sample alignments drawn from the same parent alignment. Alignment length was 130 nt for all samples. Steps on the y axis refer to the six steps listed under section Methods from bottom to top.

Figure 2 illustrates the time needed for each individual pipeline step. Observations and potential interpretations will be discussed for each figure/pipeline step individually.

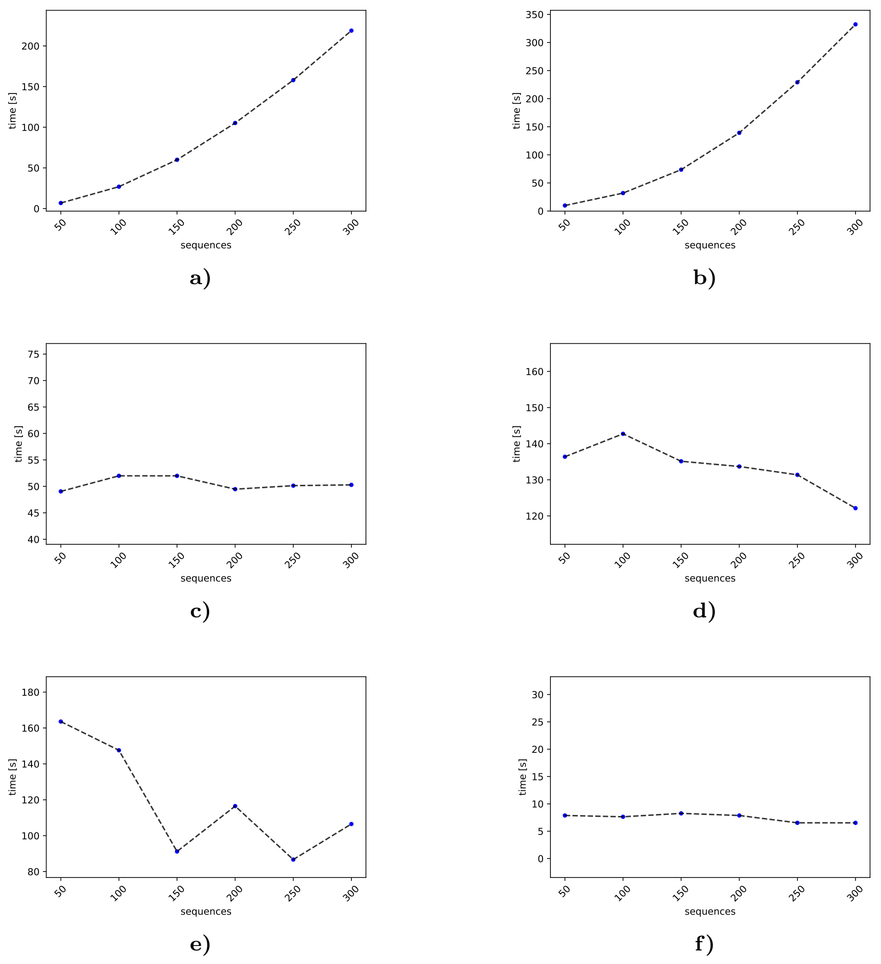

Figure 2: Computation time for each individual step of the Svhip pipeline using six test alignments with 50, 100, 150, 200, 250, 300 sequences. Graphs are shown for (a) Preprocessing of input data, (b) generation of alignment windows, (c) control alignment generation, (d) structural conservation filter, (e) feature vector calculation and (f) hyper parameter search and training.

Subfigures (a) and (b) suggest an exponential increase in time with higher sequence count. This observation can be backed based on the code implementation as follows. Let n be the number of sequences in the input alignment A. In pipeline step 1 (preprocessing) each sequence is compared with each other to filter for pairwise nucleotide identity, implying at least $n \times n$ single operations plus the linear computation cost of file I/O-handling, ungapping of sequences and finally the (typically also quadratic) cost of realignment.

A complexity at least approaching $O(n \times n)$ also is likely for the generation of alignment windows (step 2), as for each new partial alignmnet a new subset of sequences from the original alignment needs to be generated using specific and changing constraints, making individual comparison operations for a large portion of all sequences in the alignment necessary for each new window. This observation further implies, that alignment window generation followed by control alignment generation is preferable to the opposite case from an efficiency standpoint. As was shown above, the other way around would effectively double the number of operations with exponential time requirements in this step.
Based on subfigure (c), a constant time requirement of approximately 50 s +-3s is assumed for control alignment generation. This is in line with expectations for the current implementation, as control alignments are generated based on preexisting alignment windows. As these are approximately equal in number and sequence count if the input data allows it (which is the case here), time for this step will typically behave in an identical matter. A potential other parameter with influence on time is alignment length, which will be investigated separately.
Subfigures (d) and (e) actually suggest a negative correlation between sequence count and time requirements. Calculating the Pearson coefficient based on sample data yields a value of -0.8416 and -0.7651 respectively, implying a moderately strong correlation. Intuitively this is unexpected, but could be an artifact of the data itself. Based on the code, a constant time would have been expected for both pipeline steps (4) and (5). This assumption is based on the knowledge that Svhip attempts to generate an approximately equal number of alignment windows and controls for each input, as outlined above. Since calculation of feature vectors (step 5) is roughly constant in complexity for a fixed number of sequences in a single alignment window, a constant time is expected over the sum of all alignment windows. A similar situation is present for the approximation of the structural conservation with the tree edit distance. The structure estimation for each individual sequence is obviously constant (excluding sequence length, which is fixed here), while the structural conservation is estimated by doing a pairwise comparison of all sequences in a single window. This leads to a quadratic run time per alignment window. However, there are approximately 10 alignment windows for each sequence count between 2 and 15, leading to an overall similar expected time requirement for each new input file.
Subfigure (f) again implies a clearly constant run time for hyper parameter search, but obviously only within the given very small range of data points. It is well established, that single hyper parameter search steps (i.e. calculating a score for a single parameter pair) is already at least $O(n \times n)$ complex. However, SVMs are usually used with training sets of many thousand data points and not less than three hundred as is the case here. Therefore, the range of input data size is too small to see any notable effect here; this subexperiment has to be repeated with larger input data sets.

4 Conclusion

With small amounts of input data, the bottle neck operations are obviously the filtering for sequence identity and alignment window generation. This issue is however partially mitigated by strongly encouraging to only use input alignments with less than 300 sequences, as it is already documented. If this constraint is followed for all data sets, pipeline steps 3 to 5 are expected to require an approximately constant amount of time, multiplied by the total number of input alignments. Furthermore, the data here suggests a significant increase in expected efficiency by simulating control alignments based on alignment windows, as the generation of the latter is critical from a run time perspective.
Further investigations are necessary to determine the effect size of alignment length on run time. Also, it was noticed that the data set size given here is insufficient to accurately estimate the (at least quadratic) run time of the hyper parameter search. Further tests will be carried out to measure the training time impact of increasing the training set size by several orders of magnitude.