6.096 Algorithms for Computational Biology

Problem Set 2

Professor: Manolis Kellis

Instructions

This problem set is due in lecture on Friday April 22nd, 2005. For problems with a programming component, the algorithms should be implemented using python. When handing in your solutions, you should turn in your python code. Collaboration is permitted, but the write-up and the code should be your own. Please acknowledge your collaborators.

Suggested readings

- Pevzner: Edit distance and alignments page 167, BLAST page 328, Gibbs sampling page 412.
- Gusfield: Edit distance page 218, Blast 379.
- CLRS Dynamic programming

1 Rapid Database Search

Database search and comparison is performed efficiently by finding several local alignments. BLAST uses this idea. With random projections, we can design an algorithm that incorporates local alignments with up to a percentage of substitutions.

In the previous problem set we calculated the probability of a match between two strings given their Hamming distance as well as the probability of collision with multiple random projections. We will use these ideas to solve the all-pairs local-alignment problem. Given a set of sequences C with length N find all pairwise d-mers that are *similar*. We say that two d-mers s_1 and s_2 are similar if their hamming distance is at most r. Note that if $r = 0$ this is the same as finding all pairs of d-mers that are identical. In this problem we analyze the performance of the following ALL-PAIRS algorithm.

Repeat l times:

1. Pick k random positions out of the d positions.
2. Hash the $d - mers$ by their random projections. Partition the d-mers into classes that have the same random projections. For each clas C_i, store the initial position of the corresponding d-mers.
3. For each class C_i, eliminate false positives, that is compare all pairs of d-mers in a given class and store the positions of all pairs that are similar.

OUTPUT: the union of the sets of similar pairs in each iteration.

1.1 Step 2

What is the running time of step 2 for each iteration?

1.2 Step 3

What is the worst case for step 3?

1.3 False negative probability

What is the probability that two similar pairs of d-mer, s_1 and s_2, never hash together under any of the l random projections?

1.4 Sensitivity

What is the expected fraction of all similar pairs missed by the algorithm after l iterations (i.e. the false negative rate or sensitivity)?

1.5 Minimal number of random projections?

For a given sensitivity, d, r, and k, determine the smallest l, that is the number of random projections, that can be used.

2 Gibbs Sampling

In lecture we learned a probabilistic technique, *Gibbs Sampling*, for finding a motif of length l in t DNA sequences. Given a motif alignment, i.e. a sequence of starting positions in the t DNA sequences, we can calculate a profile P for a motif. Given a profile P, and an arbitrary $l - mer$, we can calculate the probability that the $l - mer$ was generated from the profile P. Gibbs Sampling iterates between these two states in the following way:

GIBBS SAMPLING

1. Randomly choose an alignment.
2. Randomly choose a sequence out of the t DNA sequences
3. Creates a profile from the remaining sequences
4. Calculate the probability that this profile generates each of the l-mers in the chosen DNA sequence
5. Sample a new starting position according to the distribution from the previous step
6. iterate

2.1 Iterative Conditional Mode

In this problem we explore another iterative optimization method called *Iterative Conditional Mode.* Look at the code provided and write in pseudo-code the algorithm that was implemented.

2.2 Iterative-conditional-mode Motif

When the random seed is fixed at 0, what is the first motif discovered on the sample sequences?

2.3 Comparison

Compare Iterative Conditional Mode with Gibbs Sampling. What steps are different for the two methods? How would these differences influence the behavior of these algorithms in practice?

2.4 BONUS: Gibbs sampling

Modify the Iterative-conditional-mode code (or provide your own code) to implement Gibbs sampling. Look at the sequences provided in the assignments section. The name of each file is a transcription factor, that is a regulatory protein that binds to DNA. A file contains several sequences separated by at least one pound sign. The known motif for the transcription factors are: for ACE2, GCTGGT; for MBP1, ACGCGT; for DIG1, RTGAAACA. Keep in mind that Gibbs sampling is a randomized method bound to return a local maximum. Report the results of your algorithm on the sequences.

3 Space complexity of the Edit Distance

Given a set of allowed transformations and their cost, the *Edit Distance* between two sequences s_1 and s_2 is the sum of the cost of the least expensive operations to transform one sequence into the other. For this problem consider the following set of operations: character insertion, deletion, and substitution. As seen in lecture, the edit distance can be computed in time $n \cdot m$ with dynamic programming. In this problem, we explore the space complexity of the edit distance.

3.1 Space $2 \cdot min(m, n)$

Show how to compute the edit distance using $2 \cdot min(m, n)$ plus $O(1)$ space in addition to the storage space needed for s_1 and s_2, where m and n are the lengths of s_1 and s_2.

3.2 Space $min(m, n)$

Show how to compute the edit distance using $min(m, n)$ plus $O(1)$ space in addition to the storage space needed for s_1 and s_2.