

Homework 5

- *Homework deadline: 10:30am on Dec. 6*
- *Please do one of the following two problems (Computational Game Theory or Computational Biology). (Students who do both problems will receive 40% bonus credit for the problem on which they achieve a lower score).*
- *Please print your code and hand it in with the hard copy of your homework. Also send a copy of your code by e-mail to both TAs.*

1. **Computational Game Theory [50 pts]**. Note: you only need to do one of the two problems on this homework (see title). In this problem you will implement and test the support-enumeration algorithm for finding a Nash equilibrium in a general sum two-player normal form game (NFG). Recall that a two-player NFG with m row player strategies and n column player strategies is described by two $m \times n$ matrices (or equivalently one $m \times n$ matrix with two entries per cell). A probability vector p is a mixed row (column) player strategy if it specifies the probability with which the row (column) player will play each of his strategies. A Nash equilibrium is described by a mapping of mixed strategies to players such that no player has an incentive to change his mixture.

One technique for finding a Nash equilibrium is the enumeration and checking of possible supports using a simple feasibility program (the program is linear for two players). The supports of the equilibrium are the pure strategies for both players with positive probability. The *size* of the equilibrium supports is considered the total number strategies in the supports of both players.

- (a) **[30 pts]** Implement a program `find_nash(A, B, k)` that takes two $m \times n$ matrices A and B specifying the payoffs for the row and column player respectively under the different pure strategy profiles, and an integer k that specifies the of supports to enumerate. Your program should return an equilibrium with supports of size k or indicate if none exists. To do this, your program should enumerate all possible equilibrium supports of size k and solve the linear feasibility program described in the Computational Game Theory I lecture to determine if any do in fact support an equilibrium.
- (b) **[5 pts]** Extend your program from part (a) to `find_all_nash(A, B, k)` so that it returns one equilibrium for each feasible distinct set of supports of size k (i.e. your program will now return an equilibrium for each set of supports that lead to a satisfiable linear program).
- (c) **[5 pts]** Download the two test games from the course web site (specified as matrices in comma separated value format) and use your program from part (b) to find an equilibrium for every feasible support of size k where $2 \leq k \leq 6$.

- (d) [10 pts] Generate 100 random 20×20 games with integer payouts between 0 and 500 and plot a histogram that indicates what percentage have at least one equilibrium with support size 2, what percentage have at least one equilibrium with support size 3 and similarly for sizes 4, 5, and 6. What conclusions can you draw from this histogram?

2. **Computational Biology [50 pts]** Note: you only need to do one of the two problems on this homework (see title). In class we discussed algorithms for finding the best global alignment between two DNA sequences. In this problem we ask you to consider the task of finding the best *local alignment* between two sequences.

We will use a scoring function that assigns +2 points when two nucleotides match, -3 when they are aligned even though they do not match and -2 for a gap (d). The best local alignment between two sequences is the two subsequences (continuous subsections for each of the two sequences) with the highest alignment score (or the empty sequence if the two sequences share no nucleotides).

For example consider the following two sequences:

AATGCCG
CGCCATG

The best local alignment between these two sequences is “GCC” from positions 4 – 6 in the first string and 2 – 4 in the second, which scores 6 points.

- (a) [5 pts] What is the best local alignment and its score for the following two sequences:

AATTAGCTCCTAC
CCAGCACCTATG

- (b) [20 pts] Describe how the Needleman-Wunsch dynamic programming algorithm described in the first Computational Biology lecture can be extended to compute the best *local alignment* for two sequences. Hint: you only need to change the local score computation for each entry in the dynamic programming array and modify the technique for retracing the best path.
- (c) [3 pts] Assume you are given a background model that indicates the probability of seeing a particular (sub)sequence of any length, $P(S)$, and a profile HMM M that can be used to compute the likelihood of a (sub)sequence of any length, $P(S|M)$. Note that $P(S | M)$ will tend to consider shorter sequences more likely, write down an expression using both $P(S)$ and $P(S | M)$ that can be used to compare the likelihood of sequences of different length.
- (d) [22 pts] The HMM method described in lecture can be used to determine the likelihood of an entire sequence given a protein family model (encoded by a profile HMM). For a particular profile HMM, M , this is equivalent to $P(S | M)$. Describe

how this method can be modified to compute a best local alignment of a particular sub sequence to the family model encoded by the profile HMM.

Note that this is equivalent to finding the continuous subsection of the sequence that maximizes the quantity you found in part (c). Hint: In addition to modifying the HMM in other ways, you will need to add a state which emits an “end” observation and assume that any local match sub-sequence also ends with this observation.