Neural Networks
Statistics, Optimisation, and Learning
Examples sheet 2: Double Digest Problem

This sheet deals with a double-digest problem, which you are asked to solve by applying a simulated-annealing algorithm. Both the double-digest problem, and the simulated-annealing algorithm are explained in detail below.

Double digest problem. Genes differ in their size by a few orders of magnitude (between $\approx 10^2$ base pairs, and $\approx 10^6$ base pairs). In order to be able to deduce the nucleotide sequence that constitutes a given gene, long regions of DNA are typically cut by enzymes into small fragments. This is the so-called enzyme digestion. When a given DNA region is cut by an enzyme, methods exist to determine the length of each fragment obtained. But, the fragment-length data do not allow to determine how the fragments were ordered in the DNA sequence before it was cut. In order to be able to infer the order of fragments in the corresponding DNA sequence, it has been suggested that the DNA sequence should be cut by two enzymes, the so-called double digestion (see Ref. [1]). A double-digest experiment is done as follows. First, a given DNA sequence is digested by one enzyme denoted by A below. Say that digesting with A results in n fragments with lengths a_i $(i = 1, \ldots, n)$, where $a_1 \geq a_2 \geq \ldots a_n$. Note that index i refers to the order of the length, and not to the order in which fragments appear along the DNA sequence. Second, the DNA sequence is digested with another enzyme, denoted by B. The number of fragments obtained in this experiment is denoted by m, and the lengths of the fragments obtained are denoted by $b_1, b_2, \ldots, b_m$, where $b_1 \geq b_2 \geq \ldots b_m$. Third, the DNA sequence is digested with both enzymes A and B (say, first with A, and then with B), yielding l fragments with lengths $c_1, \ldots, c_l$, where $c_1 \geq c_2 \geq \ldots c_l$. The double-digest problem concerns answering the question: which ordering of the fragments of lengths a_i $(i = 1, \ldots, n)$, and of the fragments of lengths b_j $(j = 1, \ldots, m)$ in the original DNA sequence give rise to l fragments with lengths $c_1, c_2, \ldots, c_l$ when the sequence is digested with both enzymes? This problem can be solved by implementing the simulated-annealing algorithm, that is described next.

The simulated-annealing algorithm uses the fragment lengths obtained in the three experiments described above as an input. It is convenient to denote the ordered set of fragment lengths produced by digesting with enzyme A as $a = \{a_1, \ldots, a_n\}$, where $a_1 \geq a_2 \geq \ldots \geq a_n$. Similarly, we use $b = \{b_1, \ldots, b_m\}$ $(b_1 \geq b_2 \geq \ldots \geq b_m)$ for fragment lengths produced by digesting with enzyme B, and $c = \{c_1, \ldots, c_l\}$ $(c_1 \geq c_2 \geq \ldots \geq c_l)$ for fragment lengths produced by digesting first with A and then with B.
The exact order of fragment lengths in the DNA sequence needs to be determined. As an initial guess, one can use a random permutation of fragment

lengths that belong to the set a, as well as a random permutation of fragment lengths that belong to the set b. We denote the former permutation by σ, and the latter by μ. The configuration of the DNA corresponding to the permutations σ, and μ is denoted by $S^{(k)}$.
The permutations σ, and μ can be used to determine the fragment lengths that are expected to be obtained if the DNA sequence (determined by σ and μ) is digested with both enzymes: first with A, and then with B. This can be computed by first drawing fragment lengths corresponding to the permutation σ. Then, the fragment lengths corresponding to the permutation μ can be drawn on top. The fragment lengths that would be obtained by first digesting with enzyme A and then with B correspond to a set of distances between consecutive points in this graph. The fragment lengths obtained in such a way need to be ordered to determine a set $\tilde{c}(\sigma,\mu) = \{\tilde{c}_1(\sigma,\mu),\ldots,\tilde{c}_l(\sigma,\mu)\}$ (where $\tilde{c}_1(\sigma,\mu) \geq \ldots \geq \tilde{c}_l(\sigma,\mu)$). The aim of the simulated-annealing is to find configurations σ, and μ such that $\tilde{c}_i(\sigma,\mu) = c_i$. This is done as follows:
The energy $H(S^{(k)})$ corresponding to the configuration $S^{(k)}$ is:

$$H(S^{(k)}) = \sum_{j=1}^{l} \frac{(c_j - \tilde{c}_j(\sigma,\mu))^2}{c_j} \ . \tag{1}$$

To solve the double digest problem, this energy needs to be minimised. To this end, we implement a simulated annealing algorithm. At each time step, starting from the current configuration $S^{(k)}$ we suggest a new configuration $S^{(l)}$ by switching the locations of two fragments in either σ or μ (see Ref. [1]). The new configuration is accepted with probability

$$P_{lk} = \mathrm{e}^{-\beta[H(S^{(l)})-H(S^{(k)})]}, \text{ if } H(S^{(l)}) - H(S^{(k)}) > 0 \ , \text{ and} \tag{2}$$

$$P_{lk} = 1, \text{ otherwise} \ . \tag{3}$$

In order to achieve a minimal energy, β needs to be increased slowly (otherwise, the algorithm may get stuck in a local optimum). The algorithm usually performs well if you allow for β to increase linearly with time t, so that $\beta(t) = \alpha \cdot t$, for some small value of α. You are welcome to experiment with other functions $\beta(t)$.

Tasks. Find on the course web page 'data 1' and 'data 2' from two double-digest experiments, containing the sets a, b and c obtained in the experiments.

- **1a.** Apply on 'data 1' the simulated-annealing algorithm described above. Make many (at least twenty) independent runs of the algorithm to find an optimal solution (with zero energy). Note that it might be difficult to find an optimal solution. Therefore, you may need to increase the number of runs (or try to get as close as possible to an optimal solution). Plot the energy in dependence of time obtained in three independent runs chosen randomly from all the runs you made (show only three curves for clarity). Discuss your findings. How does the energy change over time? Is the trend you observe expected from the algorithm explained? State the percentage of all the runs you made in which the energy reached zero. State the number of iterations (on average) that your algorithm needed to find an optimal solution, based on the runs in which an optimal solution was found. **(1p)**

- **1b.** Is an optimal solution unique? Make a table showing how the fragment lengths in sets a, b, and c are ordered in the sequence(s) found by the algorithm, together with the energy associated to each solution. If you found more than three solutions, quote only the three best ones. Discuss your findings. If you managed to find an optimal solution, explain whether the optimal solution you found is unique. If it is not unique, estimate the number of optimal solutions for this data set. Explain how you arrive at the answer. **(1.5p)**

- **1c.** How much of a performance is gained by the simulated annealing algorithm in comparison to a random search over all possible permutations of a and b in this data set? To answer this question, derive an approximation for the average number of iterations (that is, random guesses) needed to find the *correct* sequence under a random search over all possible permutations of any two sets a and b. Apply your approximation to 'data 1' and state the average number of iterations needed to find the *correct* sequence under such a random search for the experimental data given in 'data 1'. Compare to the estimated average number of iterations that the simulated annealing algorithm needed to find an optimal solution based on the runs in which an optimal solution was found (question **1a.**). Discuss your results. **(1p)**

- **2a.** Apply the simulated-annealing algorithm on 'data 2'. Make at least twenty independent runs of the algorithm to be able to find an optimal solution. (You might need to make more runs to be able to find an optimal solution.) State the percentage of runs in which you managed to find an optimal solution? Make a table showing how the fragment lengths in sets a, b, and c are ordered in the sequence(s) found by the algorithm, together with the energy associated with each solution. If you find more than three solutions, quote only the three best ones. Discuss your findings. If you managed to find an optimal solution, explain whether the optimal solution you found is unique. If it is not unique, estimate the number of optimal solutions for this data set. Explain how you arrive at the answer. Compare to the results in **1b**. **(1.5p)**

- **2b.** How much of a performance is gained by the simulated annealing algorithm in comparison to a random search over all possible permutations of a and b in this data set? Apply to 'data 2' the approximation you derived in **1c** and state the average number of iterations needed to find the *correct* sequence under such a random search for the experimental data given in 'data 2'. Compare to the estimated average number of iterations that the simulated annealing algorithm needed to find an optimal solution based on the runs in which an optimal solution was found. Discuss your results. **(1p)**

References

1. Goldstein, L. and Waterman, M. S. (1987). Mapping DNA by Stochastic Relaxation. *Advances in Applied Mathematics*, **8**, 194–207.