# IB MCB Lent Week 1 Practical

This practical will explore the use of the EM algorithm and MCMC for a model of motif finding in a set of DNA sequences, as in the well-known software package MEME. All the theory for the practical was given in the lectures – the lecture slides may well be useful in completing the practical.

There is an accompanying Jupyter Notebook for this practical. This contains almost all the code needed for the task, with missing elements at key steps, at most in one place per function. We want you to work through this notebook, filling in the missing code snippets where prompted by \*\*\*, and following the instructions to carry out experiments. We hope you will be able to fill in the missing code yourself, perhaps using the lecture notes to guide you. But we have also included below working code for each place, which you can look at for hints, or even copy and paste in if needed.

There are four stages:

1. Sample sequences from the generative model. The aim here is to first create an example motif, expressed as a set of K probability vectors and then create a set of N sequences of length L with it, each of which contains one instance sampled from the motif. You can print out the sequences with and without highlighting the motif locations.
   1. Work through the code, filling in the missing slots.
   2. We have set up default parameters motif\_length K = 8, num\_sequences N = 50, sequence\_length L = 100. You can explore varying these values during the experiments.
   3. We generate a motif by sampling the distribution at each site within it from a Dirichlet distribution where determines how likely the site probability distributions are to be concentrated on a single base, versus spread more evenly across all bases. Note that we give four values because we want a 4-dimensional probability matrix, and we are using the same value for times so as to treat the four possible bases A,C,G,T (coded here as 0,1,2,3) symmetrically.
   4. Try varying across all a range of values, say from 0.1 to 10, and draw the motifs, to get a sense of what the Dirichlet distribution does. You can either change this manually, or write a little loop or function to explore multiple states.
2. Run EM to try to find the motif on a generated sample of sequences.
   1. Work through the code, filling in the missing slots.
   2. Run the EM algorithm, and then plot\_results() as in the example cell. Does it find the correct motif? If not how does it differ? Why?
   3. Try changing and see what effect this has the ability to recall the motif
3. Run MCMC as an alternative way to find the motif.
   1. Again, work through the code, filling in the missing slots.
   2. Here we first use just the Gibbs sampler to sample positions and new motif parameters (with\_shifts=False). Run the MCMC process and plot the results. What is different from the EM result (hint: shape of the log-likelihood plot and time to converge). Again, run this several times to see the consistency of the results. Do the inferred motif distributions (logos) look different?
   3. Next we also use Metropolis-Hastings steps that consider shifts of the motif pattern left and right, and accept them if they increase the likelihood (or probabilistically if they don’t decrease it too much). Do this by setting with\_shifts=True. Run this several times. Does this now solve the problem of the pattern not being correctly centred?
4. We hope to finally run on a real data set. We are working on that for Monday.

# Crib sheet for missing code

For each function we give possible code for the place where we left \*\*\* in the notebook. This is not the only way to get the right answer. There are always alternative ways to achieve the same results.

1. Generate the data

sample\_motif() # fill using rng.dirichlet()

rng.dirichlet([alpha]\*4, motif\_length)

or rng.dirichlet([alpha, alpha, alpha, alpha], motif\_length)

sample\_sequence\_from\_motif() # fill using rng.choice()

rng.choice(4, size=1, p=motif[k, :])

generate\_sequences() # replace the subsequence of length motif\_length at positions[n] with a motif instance

sequences[n, positions[n]:positions[n]+motif\_length] = sample\_sequence\_from\_motif(motif)

1. Expectation-Maximisation

log\_likelihood() # add to pseq the probability of sequence n with motif at position i (several lines needed)

part = constant\_part

for k in range(motif\_length):

part \*= motif[k, sequences[n, i + k]]

pseq += part

expectation\_step() # update the joint likelihood for position k in the motif

likelihood \*= motif[k, sequences[n, i + k]]

maximisation\_step() # Normalize the motif\_estimates at k to sum to 1

motif\_estimate[k, :] /= motif\_estimate[k, :].sum()

1. Monte-Carlo Markov Chain

gibbs\_sample\_position() # fill in the distribution used to select the position

positions\_posterior[n]

gibbs\_sample\_motif() # fill in the parameter set to sample from

dirichlet\_parameters[k]