Assignment 1

Contents

1 The problem

For this assignment, there should be one submission to Canvas per group; list the other partner in the assignment. Submit your work as a combination of a Collab notebook (for the implementation work) and Overleaf (for showing the derivations).

In class up to now, we have learned how to derive and train a basic latent variable model, in which we treat every enhancer sequence as a mixture of bases generated from two probability distributions, representing a 'foreground' (ψ_k^0) and background (ψ_k^1). We figured out quickly that it was not a very good model, and could not capture the fact that, in atgcsequences.txt, there is a GC-rich region flanked by two AT-rich regions.

One of the central problems of our simple latent variable model is that every position of every sequence is independent, given the parameters of the model. Therefore, we could not capture GC-rich regions because the model could not consider neighboring correlations in bases.

Here we will define a more realistic model that will try to capture some of those positional dependencies. More specifically, we want a model whose parameters are defined as follows:

- $\psi_{k,p}^{(l)}$: a matrix of parameters of size $4 \times P$, where P is a hyperparameter representing the length of a motif (e.g. $P = 6$ or $P = 8$ might be common choices). This model captures position-specific nucleotide preferences. Now we have $\sum_k \psi_{k,p}^{(l)} = 1$ for all $p \in 1, ..., P$.
- Some parameters analogous to the λ_0, λ_1 (more generally, $\vec{\lambda}$) representing the prior probabilities of the foreground and background model from before.

In the real-world, such a model like this would be used to identify short, single transcription factor binding sites that exhibit a lot of position-specific base preferences, and that are present in long enhancer sequences (for which we approximately think there is less position-specific base preferences).

2 Part 1 (20%)

Define a full complete log likelihood function involving our observed sequences X, latent variables C, and model parameters $\boldsymbol{\theta} = \{\psi_{k,p}^{(l)}, \vec{\lambda}\}$. A few helpful hints:

- You will need to redefine $P(C_{i,j}|\boldsymbol{\theta})$, as well as the conditional probability $P(X_{i,j}|C_{i,j}, \boldsymbol{\theta})$.
- In this assignment, we will assume the "foreground motif" (of length P) occurs exactly once in each sequence. We therefore might consider changing the model such that $C_{i,j} = 1$ if the foreground motif occurs in position j of

sequence i, otherwise not. Because we assume the foreground motif occurs exactly once in every enhancer, think about what the implies about $\sum_j C_{i,j}$.

- Note it is important to distinguish two possible ways of modeling C_{ij}:
  1. having a latent variable model in which you only have variables C_i to indicate the position of the motif for sequence i (but when we implement it, we have a set of auxiliary variables C_{ij} so that the math expressions become easy). In this case, although your implementation might use C_{ij}, the only set of real variables you have are C_i, which will factor as $P(C) = \prod_i P(C_i|\theta)$
  2. Having a model where you explicitly have latent variables C_{ij}, as we did in the previous model. If you have such a model, in this assignment, you cannot factor $P(C) = \prod_i \prod_j C_{ij}$, because the C_{ij} for the same i are dependent.
- This problem will be much easier if you relax the problem by instead of assuming the input sequence is described by individual bases $X_{i,j}$, assume that $X_{i,j}$ represents the sub-sequence of length P starting at position j of sequence i. You could then assume that each $X_{i,j}$ is independent given the model parameters. This is an approximation because obviously $X_{i,j}$ is not independent of $X_{i,j-1}$ (since they overlap by $P-1$ bases), but it is safe to assume this for this assignment.

3 Part 2 (30%)

Derive the E and M steps for the EM algorithm for the above model. Show your work, and circle the final expressions.

4 Part 3 (50%)

Implement the E and M steps in a Collab notebook, share globally to submit, and perform the following experiments on sequence.padded.txt:

1. Plot the log likelihood as a function of EM iteration, for 20 iterations, for 5 different random initializations of the model parameter. Does the log likelihood monotonically increase every iteration of every initialization?
2. draw a sequence logo visualization of the foreground motif your model learns, $\psi_{k,p}^{(l)}$. You could try LogoMaker, a Python library (https://logomaker.readthedocs.io/en/latest/). Alternatively, there are a number of web servers for doing this; you could draw samples from your foreground model, and input those drawn sequences into e.g. the WebLogo server (https://weblogo.berkeley.edu).
3. Now run your model using model random initializations. How do the model parameters $\psi_{k,p}^{(l)}$ compare across runs? What about their log likelihoods?
4. Plot a figure that shows the distribution over $C_{i,j}$ for a few of the input sequences, and compare that (in the visualization) to the ground truth. How close was your model to predicting the real motif location?
5. Train your model using 80% of the data, holding out the remaining 20%. Evaluate the log likelihood of your held out data using the model you implemented in this assignment, and compare it to the log likelihood from the simple latent model we used in class, using the same training/held out data. Which one is better?
6. Train your model on the atgcsequences.txt file (that had a GC-rich region embedded between two flanking AT-rich regions). Does the model work better?
7. The original training set in sequence.padded.txt has 357 sequences. Randomly sample another 357 sequences of the same length (just from a simple generator, that produces each base at equal frequency) and train the model with all data. Does it still recover the same motif? What if you add 3000 randomly generated sequences instead?

Note that you will find it easier to debug your EM implementation by monitoring log likeiihood – under no circumstance should the log likelihood ever decrease. It should be monotonically increasing unless it has converged.

I would set P=18 to start - you will want to play around with this number a bit.

Also, note that implementation of the log likelihood requires implementation of the entropy term, not just the expected complete log likelihood.

Also, for computation of the numerator of the posterior, you will likely need to use the logsumexp trick to avoid underflow (multiplying lots of small numbers together is numerically bad):

$$\prod_i p_i = \exp(\log(\prod_i p_i)) = \exp(\sum_i \log p_i) \tag{1}$$