SELEX Deep HMM

1 Introduction

Systematic Evolution of Ligands by Exponential Enrichment (SELEX) is a powerful method for discovering novel ligands with high affinity and specificity for target molecules. However, the experimental process is time-consuming, resource-intensive, and involves numerous parameters that can significantly impact the outcome. This internship project aims to develop an in-silico SELEX simulator using predictive / generative models.

2 Problem Statement

First, sample a random round r from $1, \dots, R-1$ and N unique sequences (e.g. $N = 1000$) of the pool in round r. Let $\boldsymbol{s}_i$, $i = 1, \dots, N$ be the sequences, ρ_i, $i = 1, \dots, N$ the relative abundance of $\boldsymbol{s}_i$ in pool r, (i.e. $\mathbf{1}_N^T \boldsymbol{\rho}$=1) $\widetilde{\rho}_i$, $i = 1 \dots, N$ the relative abundance in pool $r+1$. The goal is to predict $\widetilde{\boldsymbol{\rho}}$ from $\boldsymbol{\rho}$ and $\boldsymbol{s}_i$, $i = 1, \dots, N$. This serves as kind of data augmentation to avoid overfitting: Because we randomly choose subsets of size N and thus vary which sequences the model sees together, it discourages memorizing absolute frequencies from the training set. Additionally, we do not need to predict the whole $\widetilde{\boldsymbol{\rho}} \in \mathbb{R}^N$ but it might be an idea to sample m random indices $I := \{i_1, \dots, i_m\}$ and to predict $\widetilde{\boldsymbol{\rho}}_I \in \mathbb{R}^m$ given the N sequences. The plan is to use cross attention between the N pool sequences and the m for-prediction-considered sequences such that their relationship can be learned. $m = 1$ could even be chosen and then iterated through the N sequences. Note that using cross-attention between the N and m sequences will scale $\mathcal{O}(NmL_{\max})$ for graphical memory, while cross-attention between the whole N sequences scales $\mathcal{O}(N^2 L_{\max})$. So, if we use $m = 1$ and $N = 1000$, we would only need $1/1000$ of the memory compared to self-attention, which seems much more feasible, and for predicting the abundance of sequence i, relationships between other sequences s_j, s_k might be neglectable given the relationships between s_i and all other sequences.

Let $L_{\max}$ be the maximum sequence length in the whole dataset (train and validation). In practice, some pre-processing needs to be done here, I think most sequences are in the range of about 140-160 nt. Outliers should be filtered, such that hopefully we can catch $> 95\%$ of the sequences in a small bound (like e.g. 140-160).

3 Input Representation: Padding & Embedding

To feed each sequence $\boldsymbol{s}_i$ of variable length into a neural network, we adopt a common strategy from natural language processing:

1. **Tokenize each sequence.**
   Let each nucleotide (e.g., 'A', 'C', 'G', 'U', and special tokens 'PAD' and 'CLASS') be assigned an integer ID:

   $$\text{CLASS} \mapsto 0, \quad \text{A} \mapsto 1, \quad \text{C} \mapsto 2, \quad \text{G} \mapsto 3, \quad \text{U} \mapsto 4, \quad \text{PAD} \mapsto 5.$$

   Thus, each sequence $\boldsymbol{s}_i$ of length L_i can be represented as a vector of integer IDs $\left[s_i^{(0)}, s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(L_i)}\right]$.

2. **Pad to a fixed length.**
   For each $\boldsymbol{s}_i$ whose length $L_i < L_{\max}$, append the 'PAD' token until it becomes length $L_{\max}$:

   $$\underbrace{[\, s_i^{(1)}, \ldots, s_i^{(L_i)}\,]}_{\text{original}} \;\rightarrow\; \underbrace{[\, s_i^{(1)}, \ldots, s_i^{(L_i)}, \text{PAD}, \ldots, \text{PAD}\,]}_{\text{length } L_{\max}}.$$

   This yields a batch $\mathbf{S}$ of shape $(N, L_{\max})$ when we stack N sequences. Additionally, we append a CLASS token in the beginning $\text{CLASS} = s_i^{(0)} = 0$ which stores information about the whole sequence (and is not assigned to a specific position), i.e. the final sequences look:

   $$\underbrace{[\text{CLASS},\, s_i^{(1)}, \ldots, s_i^{(L_i)}, \text{PAD}, \ldots, \text{PAD}\,]}_{\text{length } L_{\max}+1}.$$

3. **Lookup embedding layer.**
   We define a trainable embedding matrix

   $$\mathbf{E} \in \mathbb{R}^{V \times d},$$

   where V is the vocabulary size (in our case, $V = 6$ for {A, C, G, U, PAD, CLASS}), and d is the desired embedding dimension. Each integer token $t \in \{0, \ldots, V\}$ is mapped to a vector $\mathbf{E}[t] \in \mathbb{R}^d$. Hence, the padded sequence $[\, s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(L_{\max})}\,]$ is transformed into a matrix

   $$\mathbf{H}_i \;=\; \Big[\mathbf{E}[\, s_i^{(0)}\,];\; \mathbf{E}[\, s_i^{(1)}\,];\; \mathbf{E}[\, s_i^{(2)}\,];\; \ldots;\; \mathbf{E}[\, s_i^{(L_{\max})}\,]\Big] \;\in\; \mathbb{R}^{L_{\max}+1 \times d}.$$

   Intuitively, each row of $\mathbf{H}_i$ is a learnable d-dimensional embedding of the corresponding nucleotide (or the 'PAD' or 'CLASS' token).

4. **Positional embedding (learned).**
   Since Transformers rely on attention mechanisms that are permutation-invariant, we must provide explicit positional information. A common strategy is to introduce a trainable embedding matrix

   $$\mathbf{P} \in \mathbb{R}^{L_{\max}+1 \times d},$$

where each row $\mathbf{P}^{(\ell)}$ represents a d-dimensional embedding for position ℓ, with $\ell = 1, \ldots, L_{\max}+1$. Initialize this matrix with e.g. like sinusoidal positional encoding (or maybe naive Glorot initialziation is fine, but I think sinusoidal might work better), and is then updated during backpropagation just like other model parameters. After we compute each token's embedding $\mathbf{H}_i^{(\ell)} \in \mathbb{R}^d$ from the look-up layer, we add $\mathbf{P}^{(\ell)}$ to incorporate positional information:

$$\widetilde{\mathbf{H}}_i^{(\ell)} \;=\; \mathbf{H}_i^{(\ell)} \;+\; \mathbf{P}^{(\ell)}.$$

As training proceeds, the network learns how best to encode spatial context in $\mathbf{P}^{(\ell)}$, allowing the model to distinguish, for instance, nucleotides near the 5' end from those near the 3' end. This approach provides flexible, data-driven positional encoding compared to fixed (e.g. sinusoidal) schemes.

At this stage, each sequence $\boldsymbol{s}_i$ (now of fixed length $L_{\max}+1$ by padding) is converted into an embedded representation $\widetilde{\mathbf{H}}_i \in \mathbb{R}^{L_{\max}+1\times d}$ for each $i = 1, \ldots, N$.

4 Simple Prediction via a Neural Network

Let $\boldsymbol{H} \in \mathbb{R}^{N\times L_{\max}+1\times d}$ be the tensor of all sequences $\boldsymbol{H}_i, i = 1, \ldots, N$ after embedding (to simplify notation, we remove the tilde) and $\boldsymbol{H}_I \in \mathbb{R}^{m\times L_{\max}+1\times d}$ be the matrix of the m embedded sequences, for which the abundance should be predicted. The plan is to learn at first feasible transformations on $\boldsymbol{H}$ and $\boldsymbol{H}_I$ and then to apply cross attention on them. It should be possible to apply self-attention on each of the N embedding matrices $\boldsymbol{H}_i \in \mathbb{R}^{L_{\max}+1\times d}$, $i = 1, \ldots, N$ individually attending the positions to each other, resulting in an overall memory complexity of $\mathcal{O}(NL_{\max}^2 d)$. Also, dense $d \times d$ layer should not be problematic. So we apply some in-sequence attention (on $\boldsymbol{H}_i$ parallel in N) and $d \times d$ dense layers together with Layer Normalization, residual connection and so on, and obtain a transformation $\widetilde{\boldsymbol{H}} \in \mathbb{R}^{N\times L_{\max}+1\times d}$ or equivalently $\widetilde{\boldsymbol{H}}_i \in \mathbb{R}^{L_{\max}+1\times d}$, $i \in \{1, \ldots, N\}$. Similarly, on $\boldsymbol{H}_i$, $i \in I$ can be applied self-attention and dense layers separately to obtain $\widetilde{\boldsymbol{H}}_i^*$, $i \in I$. Now, to obtain a m-dimensional estimation of the abundance change of the sequences $\boldsymbol{s}_I$, we first extract the CLASS token now, which is inteneded to store information about sequence enrichment, i.e. we only use $\widetilde{\boldsymbol{H}}_{\cdot,1,\cdot} \in \mathbb{R}^{N\times d}$ and $\widetilde{\boldsymbol{H}}^*_{\cdot,1,\cdot} \in \mathbb{R}^{m\times d}$. We compute

$$\text{crossattention}\Big(\widetilde{\boldsymbol{H}}^*_{\cdot,1,\cdot}, \widetilde{\boldsymbol{H}}_{\cdot,1,\cdot}\Big) \in \mathbb{R}^{m\times d}.$$

Now, use a dense $d \times 1$ layer to predict a transformed label e.g. I would propose $\ln(\frac{\widetilde{\rho}_I}{\rho_I})$ which has full range $[-\infty, \infty]$. Also, somehow the $\boldsymbol{\rho}$ values need to be fused into the model, either after the cross attention or at the beginning, e.g. into the CLASS token (or maybe both). There is the set-transformer, which is

permutation-invariant which might be nice. Additionally, $\boldsymbol{\rho}_I$ should be fused seperately, or some positional encoding should be integrated into set-transformer such that it can distiguish between $\in I$ and $\notin I$.