RNA-seq Expectation Maximization implementation

Chengheng Li Chen

[1]University of California, Los Angeles / University of Barcelona

chenghengli@ucla.edu

1. Introduction

Sequencing RNA is the process of reading an RNA sequence model given a set of isoforms/genes and trying to recover the distribution of RNA molecules.

After the alignment process, we are given a set of reads with the equivalence classes that the read belongs to. We cannot determine the abundance of each gene directly since an equivalence class may contain more than one gene. Therefore, we need a model to try to recover the true abundance of each gene, denoted as $\theta = (\theta_1, \dots, \theta_M)$, where $M \in \mathbb{N}$ is the total number of isoforms/genes with $\sum_{i=1}^{M} \theta_i = 1$.

2. The model

Let $\{R_n\}_{n=1}^{N}$ be the set of all N RNA-seq reads that have already been aligned. Define $(G_n, S_n) = (i, j)$ as the pair where G_n represents the choice of the gene and S_n represents the starting position of the read within the gene. The probability of selecting gene $G_n = i$ is given by $P(G_n = i) = \theta_i$, and the probability of starting at position j within gene i, given $G_n = i$, is $P(S_n = j \mid G_n = i) = \frac{1}{l_i}$, assuming uniform coverage across the gene.

Here, $i \in [1, M]$ represents the index of the gene, and $j \in [1, l_i]$ denotes the starting position of the read within gene i. We then define the latent variable Z_{nij} such that $Z_{nij} = 1$ if and only if $(G_n, S_n) = (i, j)$. Our model can be described as follows:

$$P(r, z \mid \theta) = \prod_{n=1}^{N} \sum_{i=1}^{N} P(G_i = i) P(r_n \mid G_n = i) \quad \text{where}$$

$$P(r_n \mid G_n = i) = P(S_n = j \mid G_n = i) \sum_{j=1}^{l_i} P(r_n \mid Z_{nij} = 1)$$

combining both expression we can get the following:

$$\begin{aligned} P(r, z \mid \theta) &= \prod_{n=1}^{N} \sum_{i=1}^{M} P(G_i = i) P(S_n = j \mid G_n = i) \sum_{j=1}^{l_i} P(r_n \mid Z_{nij} = 1) \\ &= \prod_{n=1}^{N} \sum_{i=1}^{M} \frac{\theta_i}{l_i} \sum_{j=1}^{l_i} P(r_n \mid Z_{nij} = 1) \\ &= \prod_{n=1}^{N} \prod_{i=1}^{M} \prod_{j=1}^{l_i} \frac{\theta_i}{l_i} \left[P(r_n \mid Z_{nij} = 1) \right]^{\mathbb{1}\{Z_{nij}=1\}} \end{aligned}$$

and the log likelihood will be:

$$\log[P(r, z \mid \theta)] = \sum_{n=1}^{N} \sum_{i=1}^{M} \sum_{j=1}^{l_i} \mathbb{1}\{Z_{nij} = 1\} \log \left[\frac{\theta_i}{l_i} \left[P(r_n \mid Z_{nij} = 1) \right] \right]$$

3. The EM algorithm

To estimate the values of θ, we face the challenge that both the latent variables Z_{nij} and the parameters θ_i are unknown. Furthermore, the relationship between Z_{nij} and θ_i is circular: we need Z_{nij} to estimate θ_i, and θ_i to estimate Z_{nij}.

To address this, we can use the Expectation-Maximization (EM) algorithm to approximate θ. The EM algorithm consists of two main steps:

1. **Estimation (E-step)**: In this step, we compute the expected value of the latent variables Z_{nij} given the current estimates of the parameters θ_i.
2. **Maximization (M-step)**: In this step, we maximize the expected log-likelihood with respect to the parameters θ_i given the expected values of the latent variables computed in the E-step.

and then the E-step and M-step is repeated iteratively until the parameters θ_i converge to stable values.

Mathematically, in the E-step, we need to estimate $\mathbb{E}_{Z_{nij}|r_n,\theta_i}\left[\mathbb{1}\{Z_{nij}=1\}\right]$, which is equivalent to $P(Z_{nij}=1 \mid r_n, \theta_i)$ if $i \in \pi_n$ and 0 if $i \notin \pi_n$, where π_n is the set of index of all gens at the equivalence class of read n . Using Bayes' formula, we can compute the expression value, assuming that $P(r_n \mid Z_{nij}=1)=1$ for all cases:

$$\begin{aligned}
P(Z_{nij}=1 \mid r_n, \theta_i) &= \frac{P(r_n, Z_{nij}=1 \mid \theta_i)}{P(r_n \mid \theta_i)} \\
&= \frac{1 \cdot P(G_n=i)P(S_n=j \mid G_n=i)}{\sum_{i' \in \pi_n} \sum_{j'=1}^{l_{i'}} 1 \cdot P(G_n=i')P(S_n=j' \mid G_n=i')} \\
&= \frac{\theta_i \cdot l_i^{-1}}{\sum_{i' \in \pi_n} \sum_{j'=1}^{l_{i'}} \theta_{i'} \cdot l_{i'}^{-1}} = \frac{\theta_i \cdot l_i^{-1}}{\sum_{i' \in \pi_n} l_{i'} \cdot \theta_{i'} \cdot l_{i'}^{-1}} = \frac{\theta_i \cdot l_i^{-1}}{\sum_{i' \in \pi_n} \theta_{i'}}
\end{aligned}$$

Since $\theta_{i'} \cdot l_{i'}^{-1}$ does not depend on j', we are summing $l_{i'}$ times the same expression.

In the M-step, we want to maximize $\mathbb{E}\left[\log[P(r, z \mid \theta)]\right]$ with respect to θ_i, using the expected value computed in the E-step and constrained with $\sum_{i=1}^{M} \theta_i = 1$. This reduces to an optimization problem with constraints using Lagrange multipliers, leading to the following expression:

$$Q(\theta \mid \theta^{(t)}) = \sum_{n=1}^{N} \sum_{i=1}^{M} \sum_{j=1}^{l_i} \mathbb{E}\left[\mathbb{1}\{Z_{nij}=1\}\right] \log\left[\frac{\theta_i}{l_i}\left[P(r_n \mid Z_{nij}=1)\right]\right] + \lambda\left[1 - \sum_{i=1}^{M} \theta_i\right]$$

To maximize θ_i, we take the derivative of $Q(\theta \mid \theta^{(t)})$ with respect to θ_i:

$$\begin{aligned}
\frac{\partial Q(\theta \mid \theta^{(t)})}{\partial \theta_i} &= \sum_{n=1}^{N} \sum_{j=1}^{l_i} \mathbb{E}\left[\mathbb{1}\{Z_{nij}=1\}\right]\left[\frac{l_i}{\theta_i \cdot P(r_n \mid Z_{nij}=1)} \frac{P(r_n \mid Z_{nij}=1)}{l_i}\right] - \lambda \\
&= \sum_{n=1}^{N} \sum_{j=1}^{l_i} \mathbb{E}\left[\mathbb{1}\{Z_{nij}=1\}\right]\left[\frac{1}{\theta_i}\right] - \lambda
\end{aligned}$$

Setting this equal to 0 and isolating θ_i, we get:

$$\hat{\theta}_i = \frac{1}{\lambda}\sum_{n=1}^{N}\sum_{j=1}^{l_i}\mathbb{E}\left[\mathbb{1}\{Z_{nij}=1\}\right] \quad \text{where} \quad \mathbb{E}\left[\mathbb{1}\{Z_{nij}=1\}\right] = \begin{cases} \frac{\theta_i \cdot l_i^{-1}}{\sum_{i' \in \pi_n}\theta_{i'}}, & \text{if } i \in \pi_n \\ 0, & \text{if } i \notin \pi_n \end{cases}$$

Now let's define $\Pi_i = \{n \in \mathbb{N} \mid i \in \pi_n\}$ the set of reads that contains the gene i, we can combine the E-step and M-step:

$$\hat{\theta}_i = \frac{1}{\lambda}\sum_{n \in \Pi_i}\sum_{j=1}^{l_i}\frac{\theta_i \cdot l_i^{-1}}{\sum_{i' \in \pi_n}\theta_{i'}}$$

Since $\frac{\theta_i \cdot l_i^{-1}}{\sum_{i' \in \pi_n}\theta_{i'}}$ does not depend on j, we obtain:

$$\hat{\theta}_i = \frac{1}{\lambda}\sum_{n \in \Pi_i} l_i \cdot \frac{\theta_i \cdot l_i^{-1}}{\sum_{i' \in \pi_n}\theta_{i'}} = \frac{1}{\lambda}\sum_{n \in \Pi_i}\frac{\theta_i}{\sum_{i' \in \pi_n}\theta_{i'}}$$

Then, solving for the Lagrange multiplier λ, we will get $\lambda = N$. Hence our update rule will from iteration t to $t+1$ will be:

$$\theta_i^{(t+1)} = \frac{1}{N}\sum_{n \in \Pi_i}\frac{\theta_i^{(t)}}{\sum_{i' \in \pi_n}\theta_{i'}^{(t)}}$$

4. Python Implementation

In this project, we implemented an API in the file `utils.py` that reads isoforms/genes from the file `chr11-transcriptome.fasta` and associates them with reads in the file `aligned.bam`.

The main Expectation-Maximization (EM) algorithm is implemented in the class `RSEM`. The core of the EM algorithm is as follows:

```python
def em_algorithm(self, max_iter=100, tol=1e-6, verbose=False):
    for iteration in tqdm(range(max_iter)):
        new_theta = np.zeros_like(self.theta)
        N = len(self.reads)

        # Calculate expected counts and posterior probabilities
        for read in self.reads:
            denom = sum(self.theta[isoform.id] for isoform in read.isoforms)
            for isoform in read.isoforms:
                posterior_prob = self.theta[isoform.id] / denom
                new_theta[isoform.id] += posterior_prob
                self.posterior_probs[read.query_name][isoform.name] = posterior_prob

        # Normalize new_theta
        new_theta /= N

        # Calculate the median of the difference between the last iteration and the current one
        if iteration > 0:
            diff = np.abs(new_theta - self.theta)
            median_diff = np.median(diff)

            if verbose:
                print("Median diff: ", median_diff)
            # Check for convergence
            if median_diff < tol:
                break

        self.theta = new_theta
```

Note that we use the median of the differences between iterations as our convergence criterion.

Next, we have some code to save the outputs to files:

```
def save_results_to_csv(self, filename):
    with open(filename, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(['Transcript', 'Length', 'Effective Length', 'Expected Counts', 'TPM'])
        for isoform in self.isoforms.values():
            transcript_length = len(isoform)
            effective_length = self._effective_length(transcript_length)
            expected_counts = self.theta[isoform.id] * len(self.reads)
            tpm = (expected_counts / effective_length) * 1e6
            row = (isoform.name, transcript_length, effective_length, expected_counts, tpm)
            csvwriter.writerow(row)

def save_posterior_probabilities_to_csv(self, filename):
    with open(filename, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(['Read', 'Isoform', 'Probability'])
        for read_name, probs in self.posterior_probs.items():
            for isoform_name, prob in probs.items():
                csvwriter.writerow([read_name, isoform_name, prob])
```

The `save-results-to-csv` method saves the final isoform expression estimates to a CSV file. It includes columns for transcript length, effective length, expected counts, and Transcripts Per Million (TPM).

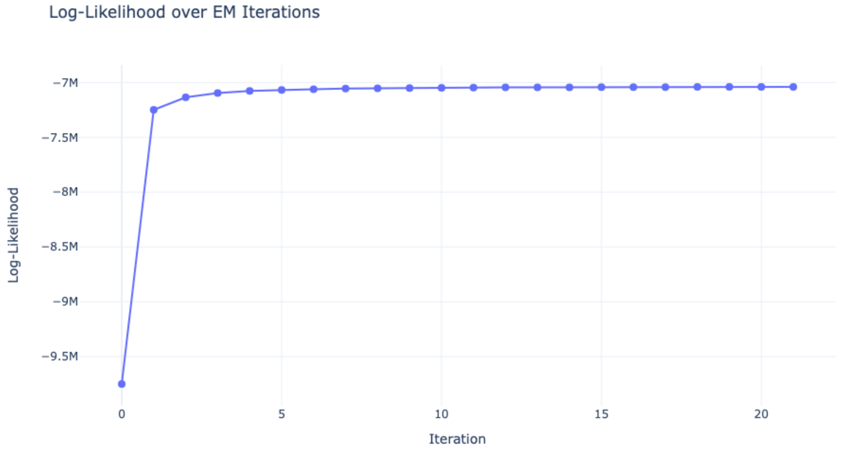
The `save-posterior-probabilities-to-csv` method saves the posterior probabilities of each read originating from each isoform to a CSV file, facilitating further analysis.

5. Visualization

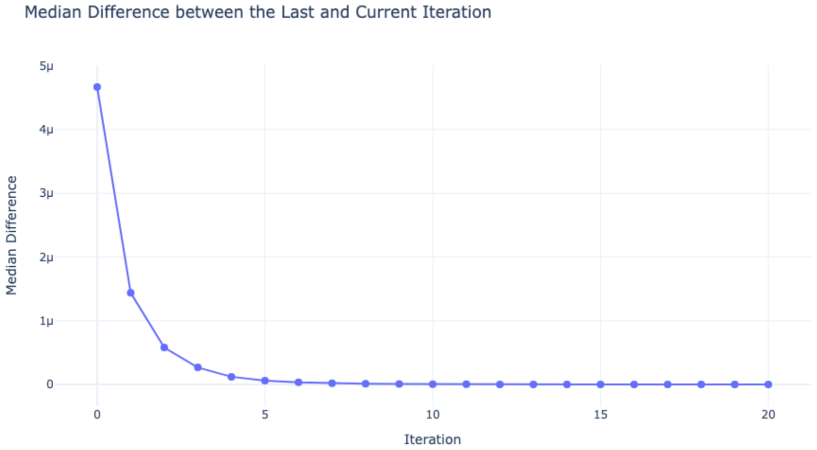
To check the convergence of the model we opted to computed the log likelihood of each iteration given by:

$$\ell(\theta) = \sum_{n=1}^{N} \log \left[\sum_{i \in \pi_n} \theta_i \right]$$

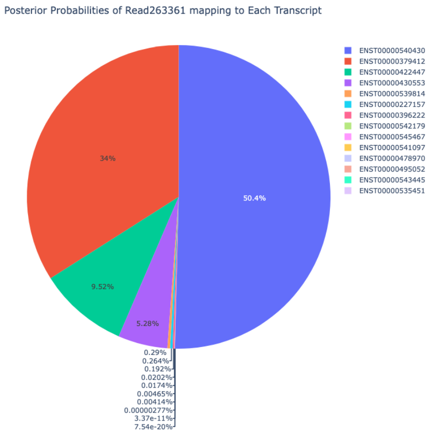
The log-likelihood provides a measure of how well the model parameters explain the observed data. As shown in Figure 1, we observe an increasing log-likelihood with each iteration, eventually converging after several iterations. This indicates that the model has been improving until it reaches a point where further improvement is minimal.



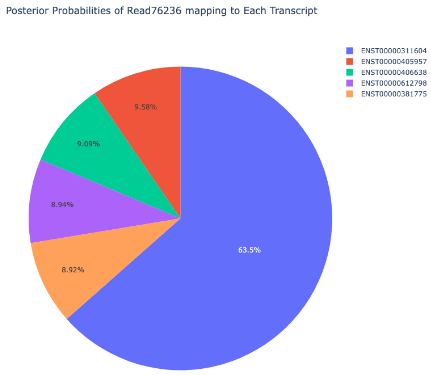
By completion, I attach the graph showing the median of the difference of iterations (the convergence criterion) with respect to the iterations (see Figure 2), which corroborates the previous statement.



Furthermore, I randomly sampled some reads from the list and created pie charts to visualize their posterior probabilities. Upon examining the results, it is frequent that several reads are mapped to multiple transcripts with low probabilities, especially those mapped to a huge number of transcripts. To illustrate this, let's take a closer look at the reads 263361 and 76236.



(a) Read 263361



(b) Read 76236

6. Conclusion

To estimate the actual abundances of genes, a simplified form of the real model with EM was implemented in this project. The fact that it supposes uniform coverage of starting positions across genes has made it possible to ignore gene length in update rules. By doing this, we were able to make the implementation process easier and concentrate on the core mechanics of EM algorithm.

It is important to note that assumed assumptions can be relaxed so that more detailed and accurate update rules can be developed which include factors such as; gene length, fragment length distributions, positional biases among others. With this in mind, better modeling of underlying biological processes would enable precise estimation of gene abundances.

Our implementation showed that during initial iterations especially, EM algorithm makes rapid progress when applied to RNA sequencing data. The parameter estimates improve quickly because the algorithm takes substantial steps towards the optimal solution at first. These results underscore how efficiently EM converges to a maximum likelihood solution concerning the observed data.

Finally, I would like to comment that this project report is longer than originally planned. The additional length is crucial as it includes the mathematical derivation of the update rule and key parts of the code. These details have been included to ensure that the report provides a clear and complete description of the method and results.