RNA-seq Expectation Maximization

Arush Ramteke

What I Did

I implemented the IsoEM version of expectation maximization for quantifying RNA-Seq data [1].

IsoEM

IsoEM is an equivalence-class based implementation of EM that works with fully aligned and not pseudoaligned reads. It assigns weights to each transcript-read pairing as follows, and performs the expectation step of the algorithm for each read with only transcripts with non-zero weights (effectively a weighted equivalence class). The weights can be thought of as the probability of a read given a transcript.

$$w_{r,j} = \sum_a Q_a F_a O_a$$

Here, we compute the weight of transcript j for read r by summing over all possible alignments the terms:

- Q_a: this is effectively the quality of the alignment, and would traditionally be computed as the base quality probabilities (present in the BAM file) weighted by if the read matched the transcript at that position. **Here, we assume $Q_a = 1$ for all aligned reads.**
- F_a: for single end reads, this is the probablity of the fragment length being lower than some upperbound u. **Here, we assume $F_a = 1$ for all aligned reads.**
- O_a: this reflects if the orientation of the read matches the orientation of the original transcript. Since we don't care about direction in our case, **we assume $O_a = 1$ for all aligned reads.**

Hence, to compute weights for each transcript-read pairing, we simply count the number of alignments a particular read has with a transcript. For implementation purposes, we only need to store non-zero ones to save memory.

Let θ be the vector of the relative abundances we are interested in, c be the vector of the expected counts for each transcript, and l be the vector of each transcript length. Let mu be the upperbound on fragment length, which we set as 200. As probabilities, for transcript j and read r:

$$P(j) = \theta(j)$$

$$P(r|j) = w_{r,j}$$

$$P(r, j) = w_{r,j}\theta(j)$$

$$P(r) = \sum_j P(r, j) = \sum_j w_{r,j}\theta(j)$$

$$c(j) = E[j|r] = \sum_r P(j|r) = \sum_r P(r, j)/P(j)$$

The algorithm is thus as follows.

$\forall j, \theta(j) \leftarrow 1/len(\theta)$
while not converged **do**
  E-Step: Determine expected counts c
  $\forall j, c \leftarrow 0$
  for each read r **do**
    $P(r) \leftarrow \sum_j w_{r,j} f(j)$
    for each transcript j **do**
      $P(r, j) \leftarrow w_{r,j} f(j)$
      $c(j) \leftarrow c(j) + P(r, j)/P(r)$
    end for
  end for
  M-Step: Re-assign relative abundances (normalized by transcript length)
  $total \leftarrow \sum_j c(j)/(l(j) - \mu + 1)$
  for each transcript j **do**
    $\theta \leftarrow (c(j)/(l(j) - \mu + 1))/total$
  end for
end while

There are instances of transcripts with length shorter than 200. If this is the case, their relative abundance is set to 0.

Implementation Details

The main challenge in implementation was determining how computing weights. For this, I used a hash table of hash tables. The outer hash table is keyed by the read, and the inner hash table is keyed by the transcript ID. The values of the inner table are the weight itself, which is incremented if the key already exists, else is added as 1. This structure is not the most time efficient but considering the number of alignments (1282526) and the number of transcripts (7138), storing it in matrix form is unfeasible.
The convergence criteria I used was median relative difference in the relative abundances between epochs. The threshold I set was 0.001.

Results

The algorithm converged very fast, requiring 50 iterations and a total runtime of 4.5 minutes.

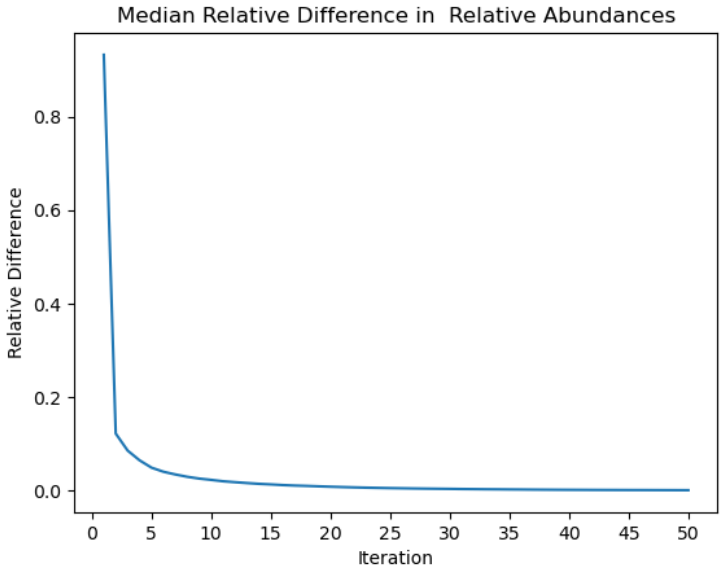

Figure 1: Convergence

The first 20 entries of the final results table are also shown in Figure 2. This can be found in table.csv in the results directory.

transcript	length	effective_length	expected_counts	tpm
ENST00000410108	637	438	116	184.64
ENST00000325147	2916	2717	185	47.40
ENST00000382762	2792	2593	1351	360.88
ENST00000529614	560	361	1	2.65
ENST00000332865	533	334	30	62.89
ENST00000486280	665	466	4	6.40
ENST00000342878	435	236	0	0
ENST00000325113	1284	1085	0	0
ENST00000525282	764	565	0	0
ENST00000526104	3506	3307	1007	210.98
ENST00000325207	2702	2503	343	94.91
ENST00000528357	500	301	39	90.74
ENST00000526982	965	766	0	0.26
ENST00000530889	563	364	97	185.38
ENST00000626818	159	0	0	0
ENST00000527696	2565	2366	395	115.74
ENST00000527468	571	372	22	41.68
ENST00000527728	859	660	106	111.64
ENST00000524854	649	450	86	133.40
ENST00000529275	440	241	87	251.27

Figure 2: IsoEM Output

Comparison with Kallisto

I converted the unique reads in the BAM file to FASTQ format and ran Kallisto [2] on this data. Instead of taking a hard upper-bound on the fragment length, Kallisto requires a mean and standard deviation of fragment length which I passed as 200 and 3.16 (variance = 10).

Performing a simple eye test on the relative abundances and expected counts generally follow similar trends, but differ in absolute values. The median relative absolute error between Kallisto and IsoEM relative to kallisto is 0.325. While this is not very low, it is somewhat reasonable.

However, the mean relative absolute error is enormous, and is equal to 21083.15. Below, Figure 3 and Figure 4 are boxplots of the log relative absolute errors between Kallisto and IsoEM relative abundances with and without outliers included. It is clear that outliers have substantial effect, but I am not sure if that is due to an error in my implementation of IsoEM or minor differences in how certain edge cases are handled.

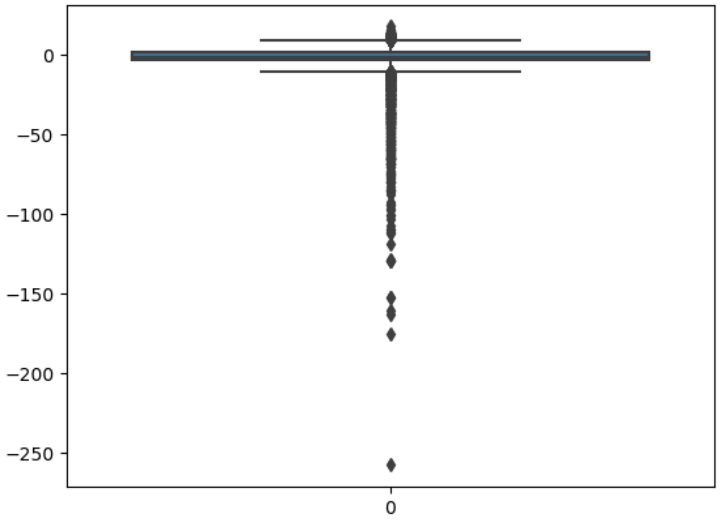

Figure 3: Log Relative Error vs Kallisto with Outliers

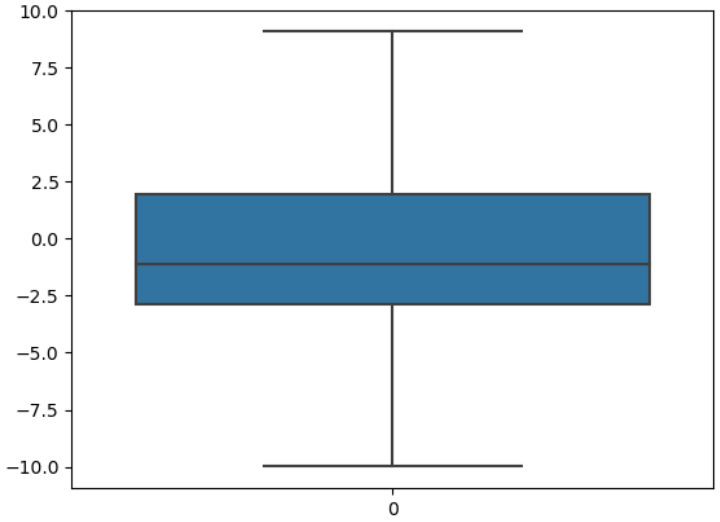

Figure 4: Log Relative Error vs Kallisto without Outliers

References

[1] Nicolae, M., Mangul, S., Măndoiu, I.I. et al. Estimation of alternative splicing isoform frequencies from RNA-Seq data. Algorithms Mol Biol 6, 9 (2011). https://doi.org/10.1186/1748-7188-6-9

[2] Bray, N., Pimentel, H., Melsted, P. et al. Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol 34, 525–527 (2016). https://doi.org/10.1038/nbt.3519