Written Report

Dorian Rojas Villalta

Final assignment written report.

MDGE 610: Foundations of Bioinformatics.

Dorian Rojas Villalta. Ph.D. Student. February 2026

This document contains the responses for the four question of the assignment including reproducibility code when applicable. The overall github repository containing resulting files, individual codes, and track changes can be found at the following link: MDGE610_final_project.

First Section. Conceptual understanding

The EM algorithm.

What is the EM algorithm?

The expectation maximization algorithm is a general approach for the iteration of computational calculations of maximum likelihood estimates as the function of the observed data. This allows for the optimization of the log-likelihood values for observed data as well as 'incomplete data', where the calculation of maximum likelihood estimate is complex due to missing variables or weights unbalance. For instance, linear regression represents a statistical approach to understand the association of several variables into an specific outcome as modeled of a linear function. The effect of each variable is represented by a coefficient that decreases the sum of squares residuals (discrepancy between observed data and model prediction). These measure are commonly optimized by the least-squares estimation. Under expectation maximization algorithm, the influence of missing data can be expected based on the observed data (expectation; E-step) and least-squares estimation can be performed using the re-calculated variables (maximization, M-step). Overall, the expectation maximization algorithm favors estimation of statistics for models with incomplete data.

What objective function does it maximize?

The expectation maximization algorithm aims to maximize the log-likelihood function of the observed data; the statistical model underlying the maximum likelihood estimation. This is a measure of how the statistical model explain the observed data. In simpler words, it is the probability of seeing the observed data given the model. This function is commonly used in the frequentist statistics for reducing errors such as the least-squares estimation from linear regression modelling.

What are the E-step and M-Step, conceptually?

The expectation step or E-step corresponds to the section of the algorithm that allows for the estimation of the sufficient statistics of the complete data based on the observed data. Sufficient statistics represents all the information that can be obtained from a dataset; thus, the E-step aims to calculate all the possible information from the observed data for the complete data filling for missing values. Furthermore, the maximization step or M-step represents the recalculation of the maximum likelihood function (log-likelihood) by using the expected complete data instead of the observed data. This way, it allows for maximizing the objective function with the completed data ignoring the constrains of the missing values.

What constitutes the missing data in kallisto's application of EM?

The missing data of the kallisto software is the equivalent classes of the transcript. An equivalent class is a multi-set of transcripts associated with a read indicating which transcript the read could have originated from. This provides the identity of the read as well as their overlap with the reference (E-step), providing enough information (sufficient statistics) for abundance estimation (M-step).

kallisto's Objective Function

What is the mathematical form of the objective function

The mathematical function that represents the likelihood function for the abundance quantitation of the RNA sequencing is:

$$L(\alpha) \propto \prod_{f \in F} \sum_{t \in T} y_{f,t} \frac{\alpha_t}{l_t} = \prod_{e \in E} \left(\sum_{t \in e} \frac{\alpha_t}{l_t} \right)^{c_e}$$

Figure 1: likelihood function for RNA-seq (Bray et al. 2016)

Here, F and T represent the total set of fragments (f) and transcripts (t), respectively. l_t is the effective length of transcript t (length with pseudoalignment) and $y_{f,t}$ is the compatibility matrix (where 1 is compatibility/pseudoalignment between f and t and 0 equals no compatibility). Finally, α_t represents the probability of selecting fragments from transcripts. Overall, this formula represents the product over equivalence classes, with c_e being the number of counts observed of equivalence class e.

How does kallisto's EM implementation maximize it

The expectation maximization algorithm implemented by kallisto aims to expect the equivalence classes (the transcript from which the read might come from) by maximizing the probability of selecting a fragment from a specific transcript α_t. This means that it optimizes the chances of obtaining a read to be associated to an specific transcript, which would serve as the needed information for estimation of their respective abundance.

Another advantage of kallisto is that the EM-optimized likelihood function is defined in term of equivalent classes. This allows for a significant reduction in the dataset that needs to be analyze, resulting in a faster sotfware in comparison to previous tools that conduct alignments.

What convergence criteria does kallisto use to decide when to stop iterating? Where are these specified in the code

The convergence criteria for stopping the iterations is determine by the total expected value of fragment belonging to an specific transcript ($\alpha_t N$). Authors define the end of the iterations as for when every transcript t, the total expected value is above 0.01 transcript per million (TPM) allowing for a change less than 1% iteration to iteration. In simpler words, the iterations of the expectation maximization algorithm stop when all of the transcripts have an minimum 0.01 TPM abundance and its variance is less than 1% between iterations.

This specification is stated in the source code for the EM algorithm of the kallisto software. The file can be found at the directory **/kallisto/src/EMAlgorithm.h**. The specific location of this convergence criteria can be found in the lines [100-104]:

```
double denom;
const double alpha_limit = 1e-7;
const double alpha_change_limit = 1e-2;
const double alpha_change = 1e-2;
bool finalRound = false;
```

Here, the main convergence criteria parameters are defined using the C++ code language. These are determine according to `[attributes] <data type> <variable name> = <value>`. Thus, `double` defines a double variable (with decimals), `const` a constant variable (not intended to be modified by the function), and `bool` a boolean variable (`true` or `false`). The variable `alpha_limit` is the minimum threshold of abundance of a transcript, `alpha_change_limit` is the threshold of abundance 0.01 TPM during maximization, and `alpha_change` is the allowed

1% abundance variability between iterations. Other variables such as `finalRound` determines the end of iterations (see below), and `denom` is an empty variable for later calculation of the denominator for normalizing read counts in case raw counts are provided.

The test of the convergence criteria is set in the code lines [173-209, comments removed]

```cpp
bool stopEM = false; //!finalRound && (i >= min_rounds); // false initially
    //double maxChange = 0.0;
    int chcount = 0;
    for (int ec = 0; ec < num_trans_; ec++) {
      if (next_alpha[ec] > alpha_change_limit && (std::fabs(next_alpha[ec]
       ↪  - alpha_[ec]) / next_alpha[ec]) > alpha_change) {
        chcount++;
      }

      // reassign alpha_ to next_alpha
      alpha_[ec] = next_alpha[ec];

      // clear all next_alpha values 0 for next iteration
      next_alpha[ec] = 0.0;
    }

    //std::cout << chcount << std::endl;
    if (chcount == 0 && i > min_rounds) {

      stopEM=true;
    }

    if (finalRound) {
      break;
    }
```

This defines the boolean variable `stopEM` to `false` prior assessing for the convergence criteria and the integer `chcount` that keeps track of over 1% variation changes in the `alpha` variables (meaning expected abundance $\alpha_t N$), which would deem that there is no convergence yet. For all equivalence class (`ec`), the `next_alpha[ec]` (expected abundance of current iteration) are tested for minimum abundance threshold of 0.01 TPM and a variation to the previous iteration (`alpha_[ec]`) exceeding 1% (`alpha_change`). If this criteria is met, the `chcount` increases in one unit, else the value remains 0 (indicating convergence). Also, previous iteration alpha variable (`alpha_[ec]`) is set with the value of current iteration before zeroing it for the next iteration. Later, the presence of convergence and minimum number of iteration (50) is checked (`chcount == 0 && i > min_rounds`) and, when true, the `stopEM` variable is set to `true`, indicating that this corresponds to the final iteration.

It is important to notice that this structure also checks for whether the current iteration is the final round, leading to a break in the expectation maximization algorithm. This variable (`finalRound`) is defined a few lines after this convergence check loop [lines 211-221]:

```
    //std::cout << maxChange << std::endl;
    if (stopEM) {
      finalRound = true;
      alpha_before_zeroes_.resize( alpha_.size() );
      for (int ec = 0; ec < num_trans_; ec++) {
        alpha_before_zeroes_[ec] = alpha_[ec];
        if (alpha_[ec] < alpha_limit/10.0) {
          alpha_[ec] = 0.0;
        }
      }
    }
```

This specific loop of the expectation maximization function indicates that when the boolean variable `stopEM` becomes true, meaning that convergence has been met, the total equivalence classes will be filtered. Within this control structure, the `finalRound` variable is defined as `true`, which will allow for breaking the expectation maximization in the structure defined above. Also, current iteration values of expected abundance are save on a different variable (`alpha_before_zeroes_`), most probably as a backup. Then, the values of each equivalence class (`alpha_[ec]`) are zeroing if their value is lower than `alpha_limit/10.0`. Overall, these commands define the end of the algorithm and filter low abundance equivalence classes.

How might these criteria be justified

Since the criteria is based on a minimum threshold of abundance for each of the equivalence classes, it might indicate this is a value considered expected in RNA-seq quantification analysis. In other words, it is expected to find transcripts of minimum abundance 0.01 TPM to be reliable in alignment-based quantification and lower abundance could be associated to alignment mismatches. Moreover, another criteria used to determine convergence is less than 1% variation in expected abundance estimation between iterations. This is based on the premise of the expectation maximization algorithm where standard deviation of the likelihood function is considered to be constant when its maximum estimate is reached.

Regardless of the fundamental basis for the selection of the criteria, authors do not declare the logic behind defining these variable to minimum threshold of 0.01 TPM and maximum variance of 1% between iteration. In order to understand the rationale, this assignment aims to understand the effect of convergence criteria values into the relative difference between analyzed simulated data and their ground truth (see Second Section. Empirical investigation).

Local vs. Global Maxima

Examine how kallisto initializes its abundance estimates

The abundance estimation of the equivalence classes initializes from the so-called pseudoalignment performed by the kallisto software. This approach is set to identify the transcript from which the read might have been originated without information regarding position alignment. For this, hashing of k-mers (nucleotide sequences of k lenght) for both the reference (transcriptome de Brujin graph) and the input reads is conducted. Hashing refers of transforming the proper sequences into numerical values. These values are then use to find the best pairwise match between the reads and reference, which results in reads assigned to a specific path of the reference transcriptome. The vertices of this matching are different read k-mers called k-compatibility classes. It is considered that these classes of an error-free read coincide with the equivalence class of the read (the transcript from where the read could have been originated). The equivalence classes are commonly estimated from alignment-based quantification approaches, which are time-consuming and computationally-demanding. However, by conducting a pseudoalignment that estimates an equivalent k-compatibility classes, these can serve as the initial abundance estimate for the expectation maximization algorithm.

Do you think initialization matter for this problem? Why or why not?

Yes, the aim of the expectation maximization algorithm is to conduct an estimation of missing data (equivalence classes) from the observed data (k-compatibility classes) in order to maximize the likelihood estimate (transcript abundance). Without this initialization process, the algorithm could not be implemented. The novelty of kallisto relies on the implementation of the pseudoalignment, a faster and computationally-lighter method of quantification, with the EM-function, a rapid and accurate optimization of objective function, to complete the missing information require to obtain reliable transcripts abundance estimates.

Second Section. Empirical investigation

Testing Convergence Criteria

. . .

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).