Solution to HMS Challenge #1b: Minority Variants 2

Chun-Sung Ferng (handle: csferng)

Basically my solution is very similar to the one for the previous contest (HMS Challenge #1). In this document, I will first review my solution structure to both contests, then describe the differences between these two solutions, which is the improvements I made for this contest.

1 Solution Structure

My solution to this contest and to the previous contest share the same structure, described below. For more details, please refer to my solution document for HMS Challenge #1.

At first, the low-frequency minority variant and fake constant reads in the given data are extracted for training, while other reads are discarded. Then, each read has its 3 metrics expanded to a number of features by some non-linear functions. After that, a Random Forest model is trained on these features. Each decision tree in Random Forest is optimized for AUC by applying stochastic cyclic coordinate ascent at each internal node inside the tree.

During testing, the same feature expansion method are applied on the reads. The Random Forest model predicts a raw score for each read, and according to raw score, the rank of a read is taken as the confidence of that read.

2 Differences / Improvements

As mentioned in the problem statement, each k-mer abundance was multiplied by a random value in $[0.9, 1.1]$. Therefore, the stability of the trained model becomes an important factor to achieve good performance. To deal with this issue, I made the following improvements.

Training data preparation In this solution I did not split the whole data into training and validation. Instead, every low-frequency minority variant or fake constant read is taken as training data. The idea is that the model would be more stable when using more data in training stage.

Feature expansion The 3 metrics of each read are expanded to 14 features. Let r_i, k_i, m_i be the read quality, k-mer abundance and mapping score of the i-th read, and let $\bar{k}_i = \log(k_i)$, $\hat{k}_i = 20 + \bar{k}_i$, $\hat{m}_i = \frac{m_i}{8}$. Taking logarithm of k-mer reduces the effect of multiplying with a random number, and the addition and division are for normalizing the three values to similar ranges. The first 8 features are formed by multiplying them together:

$$r_i^2 \hat{k}_i \hat{m}_i^2 \ , \ r_i^2 \hat{k}_i^2 \hat{m}_i \ , \ \hat{k}_i \hat{m}_i \ , \ r_i \hat{k}_i \hat{m}_i \ , \ r_i \hat{k}_i^2 \hat{m}_i \ , \ r_i \hat{k}_i^2 \hat{m}_i^2 \ , \ \frac{r_i \hat{m}_i}{(\bar{k}_i)^2} \ , \ \frac{r_i \hat{m}_i}{-\bar{k}_i}$$

To introduce the other 6 features, define $P = \left\langle \frac{r_i - 3}{32}, \frac{\bar{k}_i + 15}{15}, \frac{m_i}{249} \right\rangle$, which would be a point lies in unit cube. Then 6 features is formed by calculating 2 kinds of similarity measure between

P and 3 reference points: $P_0 = \langle 0,0,0\rangle$, $P_{.5} = \langle 0.5,0.5,0.5\rangle$, $P_1 = \langle 1,1,1\rangle$.

$$e^{-\|P-P_0\|_2}\ ,\ -\log\left(\|P-P_0\|_1\right)\ ,\ e^{-\|P-P_{.5}\|_2}\ ,\ -\log\left(\|P-P_{.5}\|_1\right)\ ,\ e^{-\|P-P_1\|_2}\ ,\ -\log\left(\|P-P_1\|_1\right)$$

These similarity-based features give decision trees more power to build the decision boundary.

Parameter The number of decision trees in the Random Forest is increased from 50 to 120. Since each decision tree is trained on a bootstrap sample of the training data, more decision trees should provide more stability.

Also, the number of feature combinations considered at each internal node of decision trees is increased from 8 to 16 because the number of features is increased.

3 Program Usage

- `data.py`: This python script is for generating training data.

  ```
  python data.py [path/to/inputs/]
  ```

  It takes one argument, which is the path to the directory containing `input1.tsv`, `input2.tsv` and `locations.tsv`. It will output one file to the same directory, `train.txt`.

- `tree.cpp`: This is the program to train Random Forest model.

  ```
  ./tree -data [path/to/train.txt] -ntree M -mxdep D -trail N -nfpn K -cdit T
  ```

  M is the number of trees to build in the Random Forest. D is the maximum allowed depth of the decision trees. N is the number of feature combinations to be considered at each internal node of the decision trees. K is the number of features to be combined at each internal node of the decision trees. T is the number of iterations of stochastic cyclic coordinate ascent. The trained and encoded model will be outputted to standard output. All parameters besides training data (`-data`) are optional. The submitted model is trained with $M = 120$, $D = 7$, $N = 16$, $K = 3$, $T = 3$.

  This program is the same as the one I used in HMS Challenge #1. Please refer to my solution document to that contest for more details.

- `linear.h`: This header file defines the feature expansion function and some helper functions.