Project Report

Phenotypic Prediction from Transcriptomic Features

Team
Jatin Garg (111462753)
Vipul Gandhi (111483224)
Rahul Bhansali (111401451)
Shruti Nair (111481332)

Phenotypic Prediction from Transcriptomic Features

Jatin Garg, Rahul Bhansali, Shruti Nair and Vipul Gandhi

Department of Computer Science, Stony Brook University.

Abstract

Motivation: Finding out nationality from the transcriptomic features has been a known problem in the field of genomics. Predicting nationalities based on the genomic sequence is a dimensionality reduction problem. We have been provided with output from Salmon, an RNA-seq mapping and quantification tool, on a number of datasets with various equivalence class with bias and no bias information. After careful considerations, we havecalculated the unique equivalence classes available in each person and applied various models including Random Forest, Sequencing center and Multi target model on it and used 5-fold cross validation technique for the testing.

Results: We started with creating a dictionary of total equivalence classes present for every sample(person). We have then made a visual representation to find out the number of equivalence classes present in several people and the unique classes available in each one of them. The models were then built based on that study. The sequencing center model performed the best out of the three with an F1-score of 0.928 and an average accuracy of 0.9298476.

Contact: jgarg@cs.stonybrook.edu; nair.shruti@stonybrook.edu; vgandhi@cs.stonybrook.edu; rbhansali@cs.stonybrook.edu

1 Introduction

There has been great expectation that the knowledge of an individual's genotype will provide a basis for assessing susceptibility to diseases and common features among them. There have been many attempts in nationality classification by using a dataset from a number of molecular layers of biological system. Despite these efforts, however, it still remains difficult to elucidate the nationality because the genome is neither simple nor independent but rather complicated and dysregulated by multiple molecular mechanisms.

In computational biology, this work will be a pioneering attempt to predict the nationality based on the underlying complex biological mechanisms. From individual Salmon, an RNA-seq mapping and quantification tool, on a number of datasets, we conducted empirical comparisons for various equivalence classes of the genomic data on individual person; to deduce possible biological implications based on the results of the relative contribution of each piece of data to increase prediction accuracy. We show that accuracy of prediction increases because of incorporation of information fused over heterogeneous biological data sources.

2 Methods

2.1 Data

We are provided with output from Salmon, an RNA-seq mapping and quantification tool, on a number of datasets. The various samples come from different phenotypes; types of nationality of different person in this case. Hence, each dataset is given a label based on the originating nationality type. The basic features from the Salmon output, such as TPM, transcript lengths and counts. Are contained in the “quant.sf” file. More detailed description on the genomic data is present in the equivalence class folders with bias and no-bias.

2.2 Random Forest Classifier

The basis behind choosing random classifier for modelling was due to its well known efficiency to train classification problems.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, ..., B$:
1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\check{f} = 1/B \sum_{b=1}^{B} f_b(x') \qquad \text{(Eq .1)}$$

or by taking the majority vote in the case of classification trees.

2.2 Multi Output Classifier

Multi Output Classifier has been taken into consideration for building a separate model involving multi class labels.

This strategy consists of fitting one classifier per target. This is a simple strategy for extending classifiers that do not natively support multi-target classification.

Multioutput-multiclass classification and multi-task classification means that a single estimator has to handle several joint classification tasks. This is both a generalization of the multi-label

```
predictedDF = multi_target_forest.predict(test.filter(dfTest1.columns[:-2]))
ansListSeq = test['SequencingCenter'].tolist()
```

classification task, which only considers binary classification, as well as a generalization of the multi-class classification task. The output format is a 2d numpy array or sparse matrix.

3 Results

3.1 Equivalence class with Bias.

We started with creating a dictionary of total equivalence classes present for every sample(person) for the Bias folder. A data frame was then created using equivalence class as column values and sample values as rows. Plotting this database in the form of Histogram, we found out that classes lying on the right end (equivalence classes ranging from 325 – 360) of the plot were significant enough to be considered, as these equivalence classes were available in the majority of the samples. These equivalence classes then became a good feature in classifying the samples. We ran a 5-fold cross validation on this model and achieved an accuracy of 0.8288 and when it was ran between 350-369, it showed an accuracy of 0.8448.

3.2 Population Label

In the population label model, we tried finding the unique equivalence classes present in each person and the equivalence classes which are common among them. It is done by taking the union of every available equivalence class in every person. Here the graph below suggests us the number of equivalence class observed in X number of persons. For example, around 100000 equivalence classes were observed in all the 369 samples.

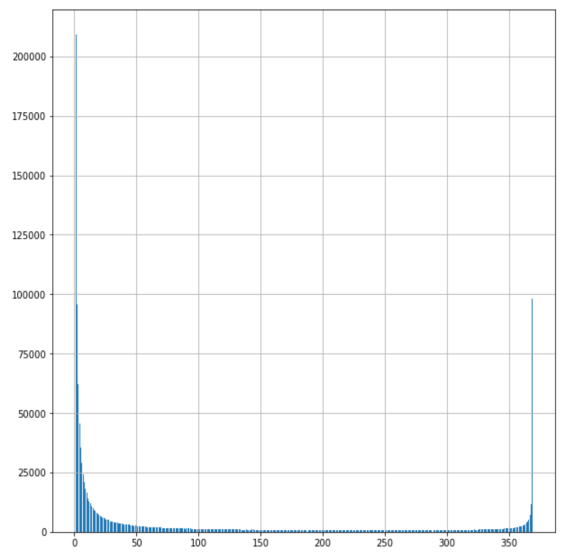

Figure 1 : Equivalence class vs count of samples

Hence, this indicates that, the more the number of equivalence classes in a specific number of samples, the more strong that equivalence class can be taken as a feature into consideration. Thus, we decided to go for choosing classes between 325 – 369 for better estimation.

3.3 Model fitting for 'population' as a label

We first divided our learned dataset from section 3.2 into training data and test data using mask filter. Our training dataset contributes of 80% of the overall data while the rest 20% is assigned to test.

We fit a Random Forest Classifier on our training data, with all the unique equivalence classes functioning as our learning variable and sample's country as our predictor variable.

We now start to read the new training file (p1_train_pop_lab.csv) we were provided with and prepared 2 different lists from the dataset we have. We built a countryList[] and accessionList[] which store the value of value of country for every corresponding value in the accession list. We matched accuracy of Country classes predicted by our Random Forest Classifier model to the data in the provided training file. We define accuracy to be the count of samples for which the predicted country label is correct to the total number of samples in the dataset. For results, we performed 5-cross validation and averaged out the results for every fold. The accuracy achieved for every fold (cumulative) is listed below.

Fold 1	0.855
Fold 2	1.714
Fold 3	2.652
Fold 4	3.537
Fold 5	4.381

Table 1 : 5 fold cross validation for the population model.

Averaging out the cumulative accuracy of over 5 folds gave an average accuracy of our model to be 0.876336198321. For calculating F1 score, we use the scikit learn library by calling,

```
sklearn.metrics.f1_score(y_true, y_pred)
```

The F1 score for the population label came out to be 0.8763361983208474.

To observe how well our model is performing, we plotted a confusion matrix for the country labels.

```
array([[13,  2,  0,  3,  0],
       [ 1,  9,  2,  2,  1],
       [ 0,  0, 15,  1,  0],
       [ 5,  4,  0, 10,  1],
       [ 1,  1,  1,  0,  7]])
```

The rows and column order of country labels in our confusion matrix is labels = ['GBR','FIN','CEU','TSI','YRI']. What our confusion matrix shows is that while our model performs exceptionally well for country label 'CEU' while for country label 'FIN', the model predicts samples to be in different classes.

3.4 Model fitting for 'SequencingCenter' as a label

We again performed the same tasks for the 'SequencingCenter' column. This model predicts the sequencing center of a sample by learning from the training dataset prepared in section 3.2 from the already read training file (p1_train_pop_lab.csv) we ectracted 2 different lists from the dataset we have. We built a sequencingCenterList[] and accessionList[] which store the value of value of sequencing center inforamtion for every corresponding value in the accession list. We matched accuracy of sequencing center classes predicted by our Random Forest Classifier model to the data in the provided training file. We define accuracy to be the count of samples for which the predicted sequencing center label is correct to the total number of samples in the dataset. For results, we performed 5-cross validation and averaged out the results for every fold. The accuracy achieved for every fold (cumulative) is listed below.

Fold 1	0.921
Fold 2	1.850
Fold 3	2.788
Fold 4	3.710
Fold 5	4.649

Table 2 : 5 fold cross validation for the Sequencing model.

Averaging out the cumulative accuracy of over 5 folds gave an average accuracy of our model to be 0.929847647953. For calculating F1 score, we use the scikit learn library by calling,

```
sklearn.metrics.f1_score(y_true, y_pred)
```

The F1 score for the population label came out to be **0.929847647953498**.

3.5 Model fitting for 'population' and 'sequencingCenter' as labels.

As we need to predict 2 labels for a given sample, we' used MultiOutputClassifier library of sklearn.multioutput. We preprocessed the given data similar to how we did for the above 2 models. We read the 'p1_train_pop_lab.csv' file and built out accessionList[], sequencingCenterList[] and countryList[]. We added these lists as 2 new columns to our new dataframe and named it as 'dfTest1'. This dataframe has now 369 rows and 160345 where we have 160343 features and 2 predictor variable columns, namely 'SequencingCenter' and 'Population'.
We built a RandomForestClassifier and named it as 'rf_model_M' which has a maximum of 100 estimators. We passed this model to MultiOutputClassifier of scikitlearn using

```
multi_target_forest =
sklearn.multioutputMultiOutputClassifier(rf_model_
M, n_jobs=-1)
```

For this also, we used 5-fold and distributed our dataset into 80% training and 20% test data. The way we built our data frame is that last 2 columns are our predictor variables, namely 'SequencingCenter' and 'Population'. So, we (dfTest1.columns[:-2]) as our independent variables which help in predicting our predictor variables. We do so by,
multi_target_forest.fit(train.filter(dfTest1.columns[:-2]), train.filter(dfTest1.columns[-2:]))
We predict the labels using the model built above by calling predict() on last 2 columns. So our 'SequencingCenter' and 'Population' gets predicted.
predictedDF = multi_target_forest.predict(test.filter(dfTest1.columns[:-2]))
Now, for accuracy measurement, we take a sample i.e. given the accession number we will mark a sample prediction as accurate only when both predicted country label and predicted sequencing center label are correct. The accuracy achieved for every fold (cumulative) is listed below.

Fold 1	0.857
Fold 2	1.704
Fold 3	2.425
Fold 4	3.245
Fold 5	4.030

Table 3 : 5 fold cross validation for the population model and sequencing center.

Averaging out the cumulative accuracy of over 5 folds gave an average accuracy of our model to be 0.8061431819104452.

4 Conclusion

For comparison, we use the values predicted by our multi output model and the individual models for predicting 'Population' and 'SequencingCenter'. We use the average of accuracy given by our individual models and compare it to the multi output model.
Average Accuracy of Individual Models = (Average Accuracy of 'Population' + Average accuracy of 'SequencingCenter') /2
From our observation we see that for 5 folds we performed above in section 3.5, Average accuracy of Individual models performed better than the Multi output model. The average accuracy for Individual Models turned out to be 1.795/2 = 0.8975. While for the Multi

Output model, the average accuracy over 5 folds from section 3.5 turned out to be 0.8061. Certainly, our multi output model did not perform better than the individual models. For the built dataset, we tried to tweak the model used for Multi Output model by using LinearRegressor but every time the best accuracy was achieved for Random Forest Classifier. So, we had to settle for the accuracy of 80.61% for the MultiOutput Model.

Acknowledgements

We would like to thank Professor Robert Patro and our Teaching Assistant FatemehAlmodaresi for their support and encouragement during the course of the project. We were able to deliver the project with results we are highly satisfied with. A huge part of the credit for that goes to both Prof. Patro and Fatemeh.

References

[1] https://combine-lab.github.io/salmon/

[2] Hugo, Willy, et al. "Genomic and transcriptomic features of response to anti-PD-1 therapy in metastatic melanoma." *Cell*165.1 (2016): 35-44.