Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Phenotype Prediction

Pheno-Predictor

Anupam Samanta, Soumyadeep Chakraborty, Keshav Gupta, Deepak Gupta

Computer Science Department, Stony Brook University, Stony Brook, NY-11790, United States of America.

Abstract

Motivation: We have performed phenotypic prediction on datasets based on transcriptomic features, obtained from Salmon, an RNA-sequence mapping and quantification tool. To achieve the same, we have come up with a feature selection method that performs classification with a high degree of accuracy. In this work, we perform a walk-through of the different machine learning approaches. All approaches we followed built off an initial baseline featuring the Random Forest machine learning model.
Results: Best accuracy of 91.07% achieved using XGBoost and TPM Count
Supplementary information: Supplementary data is available **including a detailed spreadsheet containing results and tuning parameters** at https://github.com/samanta-anupam/pheno-predictor/final_result.xlsx

1 Introduction

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 denoting population codes. Hence, each dataset is given a label based on the originating population codes. We have built a model that takes the Salmon output and predicts the original label.

On the dataset at our disposal, it is especially difficult to immediately obtain good prediction accuracies by building a well-known classification model like SVM, Random Forest or NaiveBayes on it. Each phenotypic category has data points assigned to it (around 20 or fewer samples) while each data point has a huge number of raw features (More than the total number of known transcripts for human beings, which is approximately 100k, apart from additional features). So the main challenge in this problem is selecting and generating a restricted number of descriptive and discriminative features.

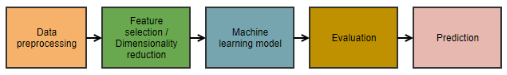

Fig. 1. Prediction pipeline

Our high level prediction pipeline begins with a data preprocessing stage where we normalize the data using Z-Score standardization. Thereafter, we performed dimensionality reduction using the k-best features obtained from a χ^2 test, in order to select the most significant features. Subsequently, we trained various machine learning models on these features and performed a comparative study of the models' efficiency. The machine learning models that we applied are Deep Learning and Extreme Gradient Boosting. Our results indicate that features such as TPM (Transcripts per million) count can be used to distinguish populations.

2 Methods

2.1 Dataset

The dataset is obtained from Salmon, which is a tool to produce highly-accurate, transcript-level quantification estimates from RNA-sequence data. The Salmon quantification file consists of metrics such as the length, effective length, TPM and the number of reads for each named transcript represented by the FASTA file in the database. A brief description of each metric follows:

- **Name** : This is the name of the target transcript provided in the input transcript database (FASTA file).
- **Length** : This is the length of the target transcript in nucleotides.
- **EffectiveLength** : This is the computed effective length of the target transcript. It takes into account all factors being modeled that will effect the probability of sampling fragments from this transcript, including the fragment length distribution and sequence-specific and gc-fragment bias (if they are being modeled).
- **TPM** : This is the estimate of the relative abundance of this transcript in units of Transcripts Per Million (TPM). TPM is the recommended relative abundance measure to use for downstream analysis.
- **NumReads** : This is the estimate of the number of reads mapping to each transcript that was quantified. It is an estimate insofar as it is the expected number of reads that have originated from each transcript given the structure of the uniquely mapping and multi-mapping reads and the relative abundance estimates for each transcript.

Along with this we make active use of the Equivalence class file obtained from Salmon. The file enlists the number of transcripts, the number of equivalence classes and the transcript names. The rank of a transcript in this list is the ID with which it will be labeled when it appears in the label of an equivalence class. Finally, the file lists equivalence class records having the following items: The number of transcripts in the label of this equivalence class (the number of different transcripts to which fragments in this class map – call this k) along with k transcript IDs and the number of fragments in this equivalence class.

The dataset comprises of around 450 samples each having 199,324 transcripts (features). There are as many as 1,235,740 distinct equivalence classes. We have used 369 samples for training and the rest for validating the models. The disparity between the number of samples and the number of features inflicts the curse of dimensionality, posing the main obstacle towards accurate feature selection.

2.2 Preprocessing

This is an essential step in our pipeline. Preprocessing the data with Z-Score based standardization precludes the presence of biases in the data and the explosion of weights during training.

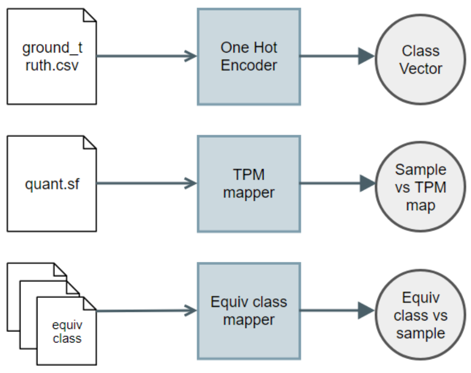

Fig. 2. Prediction pipeline

The set of candidates for data preprocessing comprises of the following:

- **Ground truth** : One Hot encoded to yield the Class Vector.
- **Quantification file** : The TPM values are extracted per sample and a matrix of (sample x TPM) is constructed.
- **Equivalence classes** : An (equivalence class x sample) matrix is constructed from the set of equivalence classes fed into the program.

2.3 Dimensionality Reduction and Feature Selection

The disparity between the number of samples and the number of features in our dataset is a classic manifestation of the curse of dimensionality. This makes the task of training a model without overfitting intractable. Thus, there is a need to reduce the number of features to consider for feature selection. The methods and results of reduction are summarized below:

- **PCA (Principal Component Analysis)** : Using Singular Value Decomposition, this method projects high-dimensional data to a lower dimensional space. This enables it to capture the maximum variance in the data. The maximum number of components obtainable via this method is the minimum over the number of samples and the number of features. Since our sample cardinality is a rather limited figure of 369, PCA yields only 369 features. This result, for obvious reasons, is not viable.
- **Tree based feature selection** : Tree based estimators are used to compute feature importances, which help irrelevant features. This method yielded over 800 features, a viable result.
- **Univariate feature selection using statistical testing** : The k–highest scoring features are selected based on a statistical test (χ^2). We set k to 12000 in order to obtain the 12000 best features from this method. We have observed the best results downstream using this method at this stage.

2.4 Machine learning models

On the dataset at our disposal, it is especially difficult to immediately obtain good prediction accuracies by building a well-known classification model like SVM, Random Forest or NaiveBayes. This calls for more powerful models to build upon the baseline accuracy achieved by the above models:

- **Supervised Deep Learning** : Recent advances in the field of Computer Vision has introduced the concept of Deep Learning to the world. Deep learning adopts a data-centric learning approach as opposed to the traditional task-based approaches rampant in machine learning. In our problem space, using Fully Connected Deep Neural Networks (FDNN) is justified as FDNNs are highly specialized towards classification problems. We observed that using a 4-layer FDNN best represents our problem space. Further, in order to prevent overfitting in our network, we have adopted the Beta Regularization and Dropout methodologies.
- **Extreme Gradient Boost (XGBOOST)** : XGBoost is an optimized distributed gradient boosting library ,implementing machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. Lately, XGBoost has proven to be an extremely popular machine learning framework used to solve data science problems. Again, classification problems are highly amenable to gradient boosting, making this an automatic choice. This method has proved to be the most accurate on our dataset, providing an accuracy of nearly **89%**.

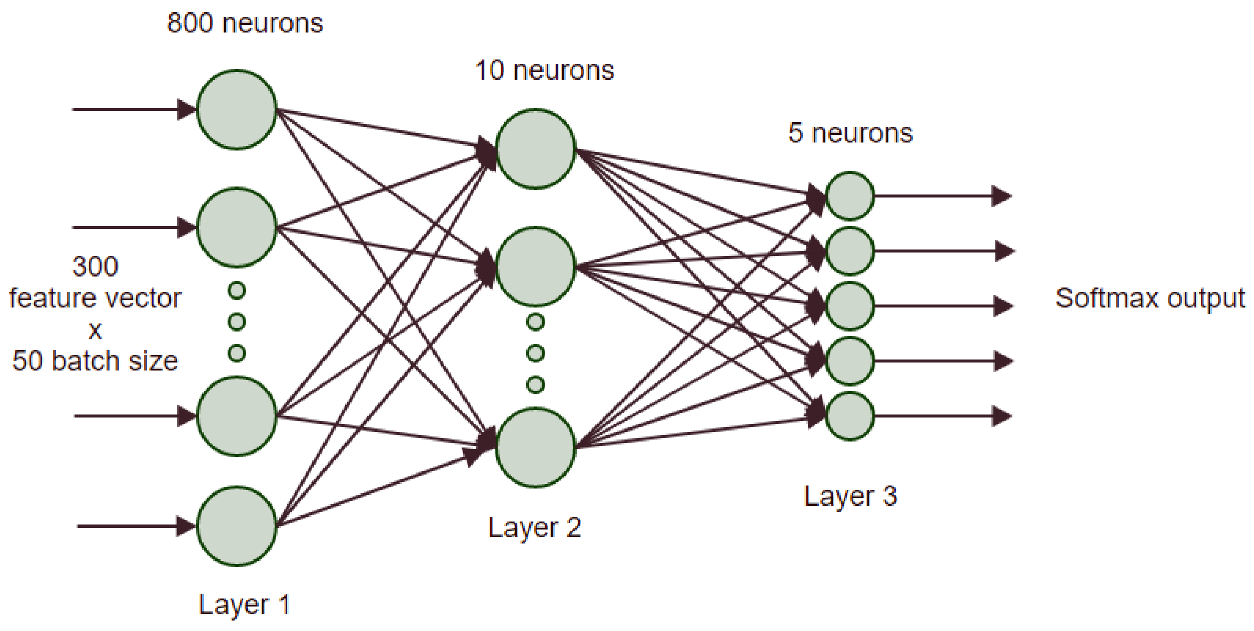

Fig. 3. Neural Network used

There is a significant absence of uniformity in our dataset across each population. In order to make our models less sensitive to the aforementioned bias, we apply K-fold Cross–Validation for obtaining a Train-Test split before feeding it to the network.

2.5 Evaluation

From our dataset, we have used 369 samples(approximately **67%** of our dataset) for training and the rest for the evaluation phase. Samples used in the evaluation phase are ones that our model has not seen before. We applied the SoftMax loss function on the character classes output from the evaluation run to derive the error percentage, which we use to characterize our accuracy.

3 Results

In this section, we report major data points and a history of the most important tuning parameters that we adopted at each stage in our pipeline to get varying degrees of prediction accuracy. **For a more comprehensive list of results vs tuning parameters in a spreadsheet format, please refer to the companion website to this paper: https://github.com/samanta-anupam/pheno-predictor/blob/master/final_result.xlsx**

3.1 Deep Learning

We started with a baseline model that used Tree Based Feature Selection followed by Deep Learning preceded by Z-Score Standardization, for which we obtained a test accuracy of **67.2%**. The feature that we considered was TPM count. After tuning hyper-parameters such as batch size, number of hidden nodes, drop-out fraction, number of layers and epoch size, we achieved varying degrees of accuracy. The highest accuracy that we achieved after tuning these parameters is **81.3%**.

3.2 XGBoost

First, we considered TPM count as a feature. Using univariate feature selection, followed by XGBoost preceded by Z-SCore Standardization and Regularization, set up to select 12000 features, we obtained a test accuracy of 91.07%. This is the highest percentage result that we obtained.

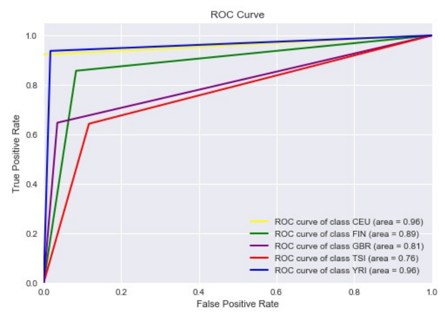

Fig. 4. Deep learning neural network ROC curve on TPM with accuracy 81.03%

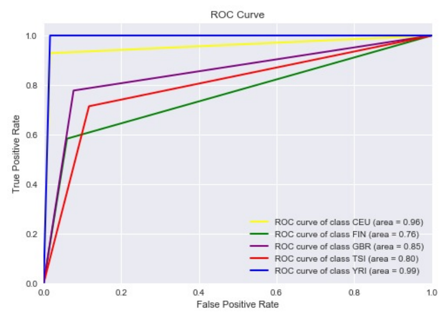

Fig. 5. Deep learning neural network ROC curve on Equivalence Class with accuracy 72.9%

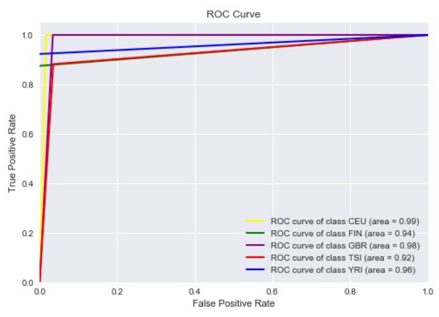

Fig. 6. XGBoost TPM ROC curve with accuracy 91.08%

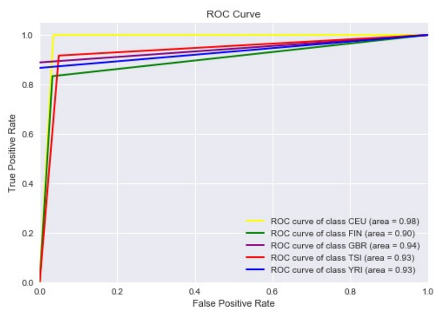

Fig. 7. XGBoost on Equivalence Class Dataset ROC curve with accuracy with 86.46% accuracy

Model	Feature	Accuracy	F-1 Score/Class
XGBoost	TPM	91.08%	[0.97, 0.89, 0.89, 0.83, 0.97]
XGBoost	EC Count	86.46%	[0.94, 0.81, 0.84, 0.76, 0.97]
Deep Learning	TPM	81.0%	[0.75, 0.83, 0.80, 0.80, 0.83]
Deep Learning	EC Count	72.9%	[0.68, 0.73, 0.78, 0.73, 0.71]

Second, we considered equivalence classes as a feature. More specifically, we considered the number of reads that mapped to a particular equivalence class for a sample as a feature. We started with PCA followed by XGBoost preceded by Z-Score Standardization and Regularization, for which we obtained a baseline test accuracy of **69.67%**. Then we replaced PCA with Tree based feature selection to yield **81.97%** accuracy. Finally, we set up univariate feature selection to select 12000 features, which yielded **86.46%** test accuracy.

4 Conclusion

As we have observed, phenotypic prediction is a problem that is amenable to the application of techniques such as supervised deep learning and gradient boosting. Using the equivalence classes and TPM as features, we were able to achieve a high degree of prediction accuracy. There is still scope for tuning the models further to achieve a yet higher degree of accuracy. Also, it is safe to conjecture that the results of combining various features such as TPM length and equivalence classes can yield promising results.

References

[1]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[2]Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, 2016
[3]Deep Learning Approach For Cancer Detection and relevant gene identification : Padideh Danaee, Reza Ghaeinia
[4]https://github.com/dmlc/xgboost
[5]Salmon: Accurate, Versatile and Ultrafast Quantification from RNA-seq Data using Lightweight-Alignment : Rob Patro, Geet Duggal, Carl Kingsford