Prediction of active and inactive cis-regulatory regions

Rumi Alberto

University of Milan,
Bioinformatics.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1 Introduction

In this project I've implemented different machine learning approaches for the prediction of active and inactive cis-regulatory regions in the HepG2 cell line from the Human Genome (HG38) with different setup. In the human genome, 98% of DNA sequences are non-protein-coding regions that were previously disregarded as junk DNA. However, non-coding regions host a variety of cis-regulatory regions which precisely control the expression of genes. Thus, Identifying active and inactive cis-regulatory regions in the human genome is critical for understanding gene regulation and assessing the impact of genetic variation on phenotype.

Cis-regulatory regions (promoters and enhancers) act via complex interactions across time and space in the nucleus to control when, where and at what magnitude genes are active.

The use of computational methods, and in particular machine learning approaches, can be a crucial tool to identify the location and activation status of these regions. To this aim, different unsupervised and supervised machine learning approaches have been developed during the years. In this work I've developed and analyzed different supervised machine learning approaches for the prediction of active and inactive cis-regulatory region in the HepG2 cell line from the Human Genome.

2 Models

Most of the models I've implemented were treated during the laboratory lessons of the course. Starting from simpler ones to more complex structures.

2.1 Comparison models

To compare the results of the neural networks approach, I've implemented two main ensemble models:

- Random Forest
- Ada-Boost

The random forest approach is an ensemble method that builds multiple decision trees and merges them together to improve the predictive accuracy and control over-fitting using the bagging technique. On the other hand, the Adaptive Boosting algorithm is a boosting algorithm which combines multiple low accuracy models to create a high accuracy models. This approach begins by fitting a classifier on the original dataset and then sequentially fits additional copies of the classifier on the same dataset where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

2.1.1 Parameters tuning

To adjust the different hyperparameters of these models I've used the Grid Search method. I've started tuning the Decision Tree depth in order to use it as the weak learner either for the random forest and for the Ada-Boost.

For ensemble methods the most important hyperparameter to set is the number of estimators. Even in this case to set this parameter for random forest and Ada-Boost approaches I've used a grid search approach, using the values displayed in Table 1. Here in square brackets are written the values I've considered and in bold the values selected.

Model	**Parameter**	**Values**
Decision Tree	Max depth	[3, 5, **7**, 9]
Random Forest	N estimators	[200, 400, 600, **800**]
AdaBoost	N estimators	[**40**, 50, 60, 100]

Table 1: Grid search results for comparison model setup

2.2 FFNN

For the feed forward neural network model I've used the structure displayed in Table 2. This structure is composed by three fully-connected layers with 64, 32 and 16 neurons and the Rectified Linear Unit activation function. A final layer structured as a single neuron with sigmoid activation function acts as output layer, computing the final binary prediction. This structure is a pretty simple structure inspired by the work [1].

During network training, weight values were adjusted using a Stochastic Gradient Descent technique with a learning rate of 0.5, learning rate decay of 0.1 and the binary cross-entropy as the loss function.

LayerType	**Units**	**Activation fun**
Dense	64	ReLU
Dense	32	ReLU
Dense	16	ReLU
Dense	1	Sigmoid

Table 2: Feed Forward Neural Network structure

2.3 CNN

For the convolutional neural network I've used the structure displayed in Table 3. This structure starts by combining a 1D Convolution with a batch normalization layer. Than follows a max pooling layer, another convolution layer with batch normalization and another max pooling. After these layers follows some flatten, fully-connected, and dropout layers (Table 3).

As for the feed forward model, all neurons in each layer have ReLU activation function with the exception of the output layer, which has a sigmoid activation function. The Nadam algorithm was used to adjust weight values with learning rate set to 0.002.

Also this structure is mostly inspired by the work [1].

Layers	**Type**	**Units**	**Kernel**
1	Conv1D + BatchNorm	64	5
1	MaxPooling1D (Size 2)	-	-
1	Conv1D + BatchNorm	64	5
1	MaxPooling1D (Size 2)	-	-
1	Flatten	-	-
1	Dense	64	-
1	Dropout (p=0.2)	-	-
1	Dense (ReLU)	64	-
1	Dropout (p=0.1)	-	-
1	Dense (sigmoid)	1	-

Table 3: Convolutional Neural Network structure

2.4 MMNN

For the multi-modal neural network I've used the two different models described in Section 2.2 and Section 2.3.

In the multi-modal network the critical aspect is the method used to combine the neural networks. To do so I've used the intermediate integration approach[2] concatenating the outputs of the two neural networks and then applying a fully-connected layer with 64 units and ReLU activation function followed by a single neuron with sigmoid activation function which acts as output layer and computes the final binary prediction. This architecture is described in Figure 1.

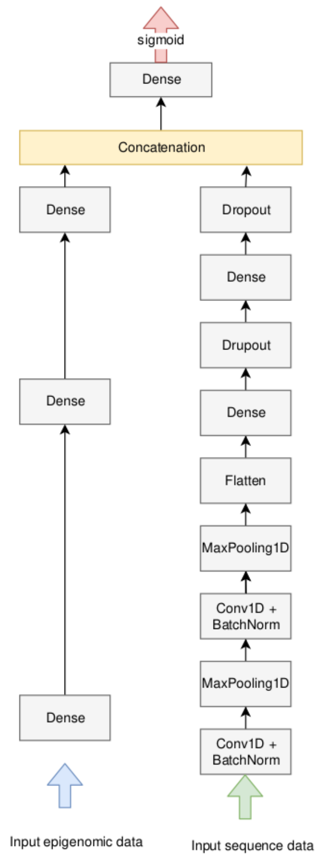

Figure 1: Multi Modal Neural Network structure, the two different path from the inputs to the concatenation layer represents the neural networks described in tables 2 and 3

3 Experimental setup

The classification tasks I've studied are Active enhancers vs inactive enhancers and Active promoters vs inactive promoters on the HG38 dataset. The models were trained and tested on genomic regions of transcriptionally active enhancer and promoters downloaded from FANTOM5 and matched features from ENCODE for the cell line HepG2. This data acquisition step is made using the active_enhancers_vs_inactive_enhancers and active_promoters_vs_inactive_promoters functions developed by the AnacletoLAB [3] which has already retrieved the epigenomic data from ENCODE, the labels from FANTOM5 and queryed the files.

In order to binarize the data to distinguish active regulatory regions from inactive regulatory regions I've used the threshold method, setting a threshold on the Transcript Per Million (TPM) value at 0 for enhancers and at 1 for promoters.

In order to retrieve the genomic sequences, I've used the UCSC Genomes browser[4] by the ucsc_genomes_downloader library.

3.1 Pre-Processing

Different pre processing step are then applied to the data. The pre processing pipeline I've implemented is:

1. Impute null values
2. Visualize data (Section 3.3)
3. Scale values
4. Remove uncorrelated data with labels (Section 3.2)
5. Analyze correlation between features (Section 3.2)
6. Analyze distribution of labels and features (Section 3.2)

The first imputation step of the null values, is done by using the KNN-Imputer provided by sklearn considering up to 5 neighbours. To scale the data I've used a robust scaler which makes use of robust statistics to deal with outliers, by subtracting the median and dividing by the standard deviation between the interquantile range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile).

3.2 Dataset correlation and distribution

In the first place I've studied the correlation with the output of each sample of the dataset in order to avoid considering not correlated data with the labels. In order to compute the correlations I've used both the Pearson and Spearman correlation coefficients. The least correlated values from both Spearman and Pearson are then tested with the Maximal Information Coefficient in order to find also the possible non-linear correlation of these samples. If also this last test gives low results, these values are then discarded from the data.

In order to compute the correlation between the features I've used the Pearson correlation coefficient. From this I've noticed that there are a few features that are extremely correlated (8 pair of features for enhancers and 2 for promoters). For each of these pairs would be ok to drop one of the pair, but I've decided to leave this work to the different feature selection algorithms described in Section 3.4.

In order to also visualize the correlation relation between these features, I've used the visualization scatterplot provided by seaborn in order to plot the most and the least correlated features from both enhancers and promoters. This can be seen in the Additional Figure *CorrelationPlots.png* and in the jupyter notebook.

To visualize the distribution of both features and labels I've used histograms. Since it is not possible to show all the distributions for all the features (556 for enhancers and 547 for promoters), I've selected the top 10 most different features. To find those I've defined the distance between the features using a pairwise cosine distance. To reduce the impact of outliers I've filtered out outliers before the 0.01 percentile and the 0.99 percentile (Figure 2).

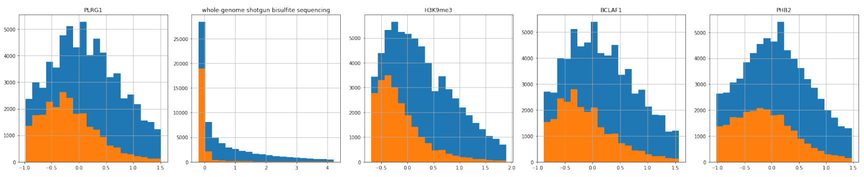

Figure 2: Distribution of the Features

For the distribution of the labels I've plot the data distribution without any particular manipulation with an histogram (Figure 3)

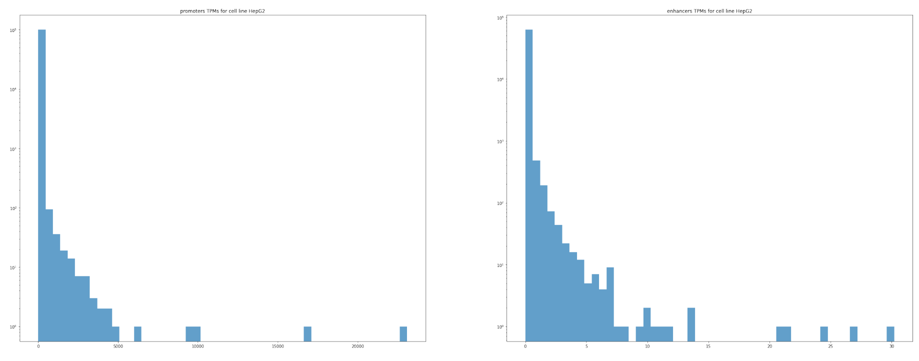

Figure 3: Distribution of the samples

3.3 Data visualization

In order to visualize the whole dataset I've used a principal components analysis (PCA) and a t-distributed Stochastic Neighbor Embedding (t-SNE) for both enhancers and promoters and for both sequence and epigenomic data (Figure 4). To do so it is necessary to use the raw data, before the scaling operation, because these visualization techniques are sensible to scaling. It is possible to say that the visual cluosters of active and inactive regions is slightly more visible with the t-SNE approach in the epigenomic visualization, while for the PCA I couldn't get any meaningful representation aspect.

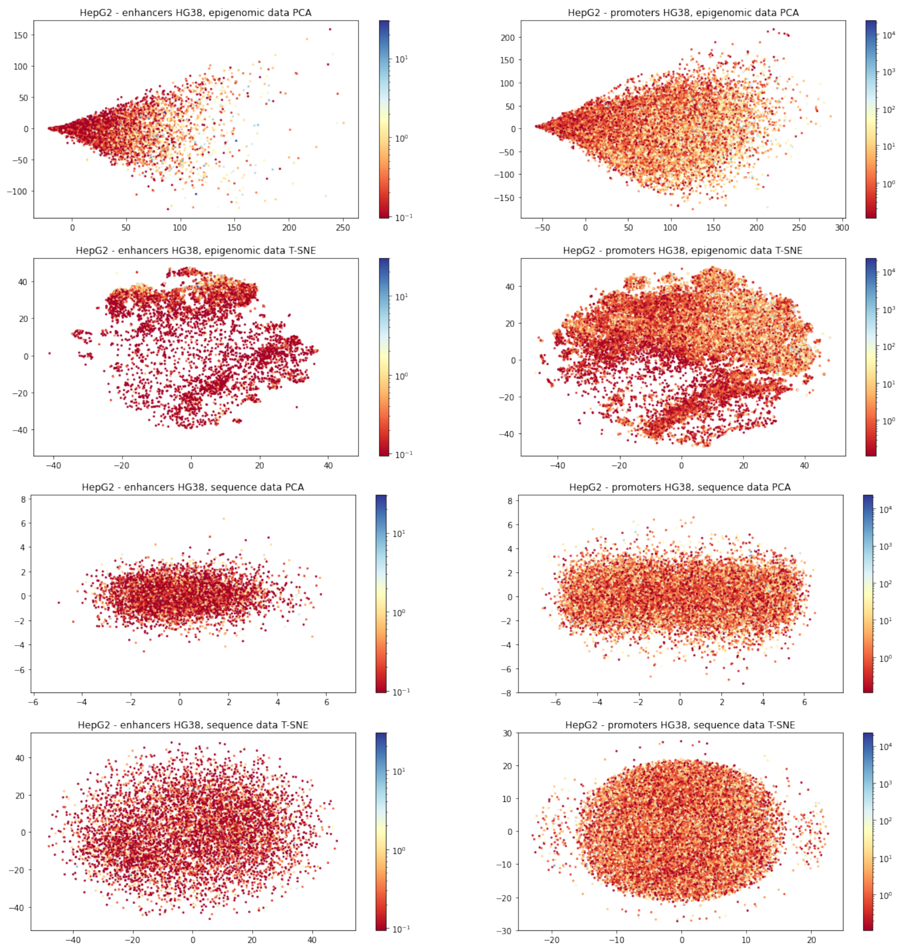

Figure 4: PCA and t-SNE visualization for epigenomic and sequence data

3.4 Feature Selection Algorithms

Due to the high dimensionality of the data, I've decided to apply two different technique in order to make same feature selection. I've then compared the results of these techniques also with a "no feature selection" approach.

3.4.1 Boruta

The first algorithm I've decided to use is the Boruta[5] algorithm which is a wrapper feature selection based on random forest. It iteratively removes the features which are proved by a statistical test to be less relevant than random probes. The importance measure of an attribute is obtained as the loss of accuracy of classification caused by the random permutation of attribute values between objects. It is computed separately for all trees in the forest which use a given attribute for classification. Then the average and standard deviation of the accuracy loss are computed.

3.4.2 Backward feature selection

The other approach I've decided to use is the backward feature seleciton. This is a greedy feature selection algorithm which starts from the whole feature set and then removes features 1 by 1 in order to remove as little information as possible. I started selecting a significance level of 5% (p-value = 0.05), then I fitted a linear model with the feature selected (starting at the whole feature set). The predictor with the highest p-value is then calculated and if this p-value is higher then the significance level the algorithm terminates, while if it is lower, the feature is removed from the selected set and the procedure restarts.

3.5 Holdouts

In order to evaluate the project I've used the same subset of training sequence for each possible feature selection technique: no feature selection, Boruta and backward. This holdout technique is applied with both the comparison models and the neural network models and can be described by:

- For enhancers and promoters:
  - Apply all feature selection techniques in order to get the 3 different subsets of features,
  - For each holdout:
    1. Split the dataset into train set and test set,
    2. Evaluate all the models for each feature selection technique

So for a single holdout, a model is trained for every possible feature selection method and then evaluated. For this reason and for the fact that I do not have the access to a powerful machine, the number of holdouts used is reduced to 4. This means that in total there will be 2 regions, times 4 splits, times 3 feature selection, times the number of models to evaluate (3 in both cases) for a total of 72 train-test pairs of performance evaluations for each evaluation loop.

In order to split the data I've used a stratified shuffle split technique, splitting in 80% train set and 20% test set.

The main reason I did this kind of holdout configuration is because the feature selection methods I've used are really computationally intensive. So instead of computing the subset of features that the feature selection technique would select on each train-test split, it is calculated just once in the start and then these features selected are used for each split. This will save up to 40 minutes per train-test split with my hardware configuration.

3.6 Result analysis techniques

To evaluate all the different models and feature selection techniques performances I've used three different metrics: Area Under the Precision Recall Curve (AUPRC), Area Under the Receiver Operating Characteristics (AUROC) and Accuracy.

While for the AUPRC the baseline in order to assess if the results are good (0.5 representing the random event of tossing a coin), the baseline for the AUPRC needs to be calculated for both enhancers and promoters. This baseline is calculated as the fraction of positive examples in the whole dataset, resulting 0.258 for promoters and 0.113 for enhancers.

After the evaluation of these metrics, a Wilcoxon test between the different feature selection techniques is applied in order to compare how much they differ one another and how these techniques affect the predictions.

4 Results

Starting with the comparison models, it is possible to assess that the adaboost technique I've implemented leads to an overfitting issue, scoring really well for all the metrics in the training set and performing pretty bad on average on the test set (Figure 5). A surprising result is that the simplest decision tree model gives really good results, close to the ensemble methods and even better for the AUPRC metric. The different feature selection methods used, seems not to give so much difference between each other. All the result graphs can be seen in Additional Figures *ComparisonModelStatistics.png*, *NNModelStatistics.png* and in the jupyter notebook.

For the Neaural network approaches I have different kind of results. We can immediately assess that the CNN architecture that I've implemented gives really bad results, with a really high overfitting issue. As it is possible to see in Figure 6 there is a perfect AUPRC result in the training set and a slight over average result for the test set. As expected this overfitting issue of the CNN model is also reflected in the multi modal neural network, due to the fact that the result of the CNN is concatenated with the results of the feed forward

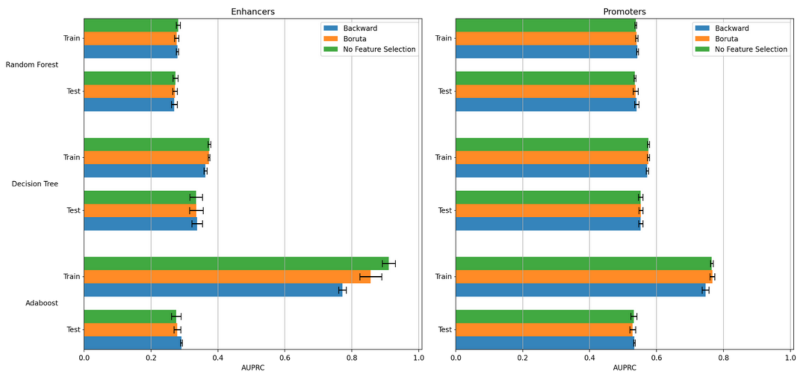

Figure 5: AUPRC metric for comparison models

network, influencing the final result. On the other hand the simplest structure of the FFNN gives the best results, outperforming in all the metrics all the other models taken in consideration in this experimental setup.

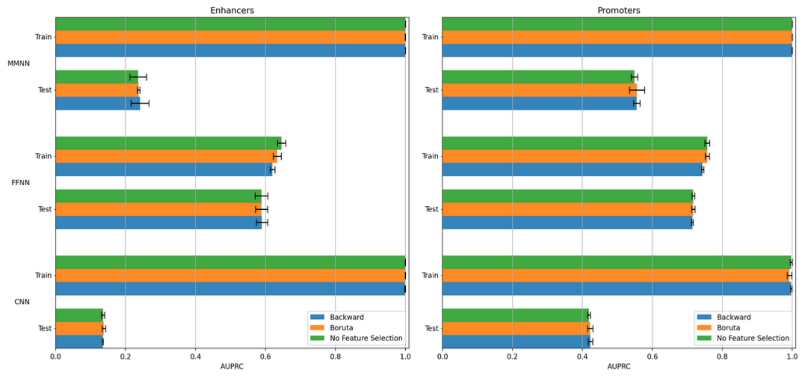

Figure 6: AUPRC metric for comparison models

The last step is to analyze the wilcoxon test for the different feature selection methods I've used. The results indicate that there is no model-feature selection pair in the ones I've implemented that are statistically distinguishable from each other. This is a pretty strong result due to the fact that, as it is possible to see in Table 4, the backward feature selection uses half of the features than the other methods for both promoters and enhancers.

5 Conclusion

The approaches I've implemented in this project gives pretty nice results compared to baselines models. Even thought the results of the neural network

Region	**FS Method**	**N of features**
enhancers	No FS	556
enhancers	Boruta	352
enhancers	Backward	151
promoters	No FS	547
promoters	Boruta	493
promoters	Backward	264

Table 4: Caption

approaches for the convolutional and, by consequence, the multi-modal architectures are not as desired, even worse than the simplest models. Thanks of this experimentation we can assess that the neural network approaches in analyzing epigenomic and sequence data are really powerful approaches, but they can also lead to really bad results if not used with attention.

For the feature selection methods I can assess that there is not a statistical difference between the approaches I've implemented. Thanks of that, it could be possible to use a reduced set of features for future analysis in order to sensibly reduce the computational power needed to train the models.

References

[1] L. Cappelletti, A. Petrini, J. Gliozzo, E. Casiraghi, M. Schubach, M. Kircher, and G. Valentini, "Bayesian optimization improves tissue-specific prediction of active regulatory regions with deep neural networks," pp. 600–612, 04 2020.

[2] J.-H. Choi and J.-S. Lee, "Embracenet: A robust deep learning architecture for multimodal classification," *Information Fusion*, vol. 51, p. 259–270, 2019.

[3] AnacletoLAB, "Anacletolab epigenomic dataset pipeline." available at https://github.com/AnacletoLAB/epigenomic$_{d}ataset$.

[4] https://genome.ucsc.edu/index.html, "The human genome browser at ucsc."

[5] M. B. Kursa, A. Jankowski, and W. R. Rudnicki, "Boruta – a system for feature selection," *Fundamenta Informaticae*, vol. 101, no. 4, p. 271–285, 2010.