ALM Data Challenge

Joel Cerqueira Ponte and Oussama Zerguine

École nationale supérieure d'informatique et de mathématiques appliquées de Grenoble

February 17, 2018

In this work we present our approach to the challenge proposed as a final project for the course Advanced Learning Models. The project consisted in predicting whether a DNA sequence region is binding site to a specific transcription factor. No machine learning libraries were allowed, so it was necessary to code whichever algorithm that had to be used. Our final result was the first place, with an accuracy of 83.3% in the private leaderboard.

1 Introduction

The challenge consisted in doing the predictions for 3 distinct datasets with 2000 observations each and perfectly balanced binary targets. For each dataset, we were given two versions. The first version contained raw DNA sequences and the second contained a numeric representation calculated based on a bag of words (BoW) representation of the first. The point was that the second dataset was easier to work with and did not require feature engineering, with the drawback that the performance was limited because the created features were not necessarily optimal.

Our strategy consisted in beginning with the simplest possible solution and slowly increasing complexity to get better results.

2 Timeline

2.1 Bag of words dataset

We started by looking at the simpler dataset. In this phase, we used the famous python library *scikit-learn* to explore which algorithms were most likely to give the best predictions. We were not allowed to use machine learning libraries, but there was no rule that prevented us from using them to explore algorithms, as long as we did not submit predictions using them to Kaggle. To estimate the performances, we used 5-fold cross validation.

In this phase, we soon realized that more complex algorithms (such as neural networks, tree-based methods and SVM with kernels) did not seem to result in better performances. A few hypotheses could explain that, like the fact that the training sets were relatively small (2000 observations each) for the amount of features (101 columns). Another possible explanation is that the optimal decision boundary for that representation could actually be simple, close to linear.

Later, we trained two extremely simple algorithms that we had implemented in the past: Perceptron and Adaline. A 5-fold cross validation showed that the Perceptron was more accurate, although the accuracies in the different folds were very inconsistent with each other for both methods. We then submitted Perceptron's predictions to the Kaggle, which gave us an accuracy of 55%, slightly better than a random guess (remember that the classes were perfectly balanced).

From our experiments with scikit-learn, we noticed that support vector machines and logistic regression gave us very similar performances, even when we considered rbf and polynomial kernels. Because logistic regression was easier to adapt from Adaline, we decided to use it. Our 5-fold cross validation was more stable and predicted a lot better the accuracy in the leaderboard this time: around 64%.

2.2 Raw data

2.2.1 Observations as pairs

The first approach was to try to incorporate some features in the BoW datasets set using the raw data. With some exploratory analysis of the raw datasets we soon noticed that all observations could be separated into pairs, with each pair having the same counts of each letter (ATCG) in their sequences. For each of those

Table 1: *Feature engineered inputs*

AAAAA	AAAAC	...	TTTG	TTTTT
1	0	...	0	0
0	1	...	0	1
...	...	...	...	...

pairs, it is always true that they have different labels.

For example, say we find an observation that contains letter A 23 times, letter G 26 times, letter C 38 times and letter T 14 times. We found that, in 100% of the times, there will be another observation **with the same counts and opposite label**.

With that information, we then found a constraint for our predictions. We could now observe the pairs and ensure that one of them would always be labeled as "0" and the other as "1". To do this, rather than using a discrimination threshold of 0.5 or any other fixed number, we observe the pairs and, for each pair, we assign to "1" the one with bigger output probability and "0" to the other.

It is important to notice that there are a few cases where we have $n > 2$ (n always even) observations with the same number of counts for each letter. That means that there are multiple pairs that cannot be identified among themselves. The choice in those cases was to assign the label "1" to the $\frac{n}{2}$ observations with biggest probabilities and "0" to the others.

This approach was able to boost up accuracy from around 64% to around 71% in the leaderboard.

2.2.2 Feature engineering

In this phase we tried to actually create features from the raw data. A few approaches were tried, but the one that gave the best performance was to create one binary feature for each combination of letters of length 5, indicating if that string was present or not in the observation, as shown in Table 1.

Since we had a big number of features $(4^5 = 1024)$ now, we were concerned with overfit. Our first idea was to use l_1 regularization, as it would set multiple weights to zero, but our tests indicated that l_2 regularization was more effective. Then, we implemented it in our past version of logistic regression. This, allied with the strategy explained in section 2.2.1, boosted up the score to 80.9%.

After some hyperparameter tuning we were able to increase the accuracy to the final value of 82.26% in the public leaderboard (74.27% without using the approach described in 2.2.1).

2.2.3 Failed approaches

We tried a few other approaches, which resulted in worse accuracies:

- Other lengths of combinations for the sequence indicator features (e.g. length 3: "AAA", "AAC", ...)
- Instead of trying binary features indicating the presence of strings, the number of occurrences of them.
- A mix of the approach described in 2.2.2 and the BoW dataset.
- To separate each letter in the DNA sequence in one column (resulting in 101 columns, each with a letter A, C, G or T) and doing mean encoding for each column.
- The same as before, but with one-hot encoding.
- Other machine learning algorithms besides regularized logistic regression.

3 Conclusion

It was clear from the successful and unsuccessful approaches that, rather than the individual contributions of each columns, the types of "subsequences" that appear in the DNA are more effective to predict the target. An exploratory analysis of the data was very important to come up with the method explained in section 2.2.1, which was crucial to increase the accuracy. Additionally, the number of observations was quite small, which may be the reason why more complex models and features did not boost up the performance.

We were able to ensure the first place in the public leaderboard with only 6 submissions (with $\sim 81\%$ accuracy), using the other many submissions to get the remaining $\sim 1\%$. The final result was an accuracy of 83.3% in the private leaderboard, which meant a considerable difference of 4% to the second place.