Data challenge for the course "Advanced Learning Models"

Slavyana DEMENTYEVA, Varsha DEVI

M2 MOSIG
Data Science
University of Grenoble Alpes
E-mail: slavyana.dementyeva@gmail.com, varsha.devi@etu.univ-grenoble-alpes.fr

Abstract -

The written report is the presentation of our work on data challenge for course advance learning models. The task was a classification problem. The major goal was to predict whether a DNA sequence region is binding site to a specific transcription factor or not. The task was based on implementation of the algorithms without the use of any machine learning library and gain understanding about them.

1 Introduction

For the task of classification, we were provided with three different train and test dataset, with corresponding labels only for train dataset. For train dataset, we had total 6000 observations in 3 different training datasets with 2000 observations in each dataset, where each row represents DNA sequences. For test dataset we had 3000 observations in three dataset files. We were provided with 2 different versions of observations in dataset. One was raw representation of sequences, while the other version was numerical representation based on Bag of words. The numerical representation of the dataset was provided for those who wanted to work with numerical data but the optimal solution was not guaranteed with the numerical representation.

2 Approaches

In this section we describe two approaches for DNA sequences representation: TF-IDF vector representation and K-mers vector representation. Both approaches are based on retrieving features from DNA subsequences. Also, we give a brief introduction into the kernel methods used as part of classification task.

2.1 TF-IDF

Term Frequency - Inverse Document Frequency is a well-known statistics used for extracting features from the text data and thus it was a naturally chosen as first approach we can apply to the given dataset.
Term Frequency (TF) component counts the number of word occurrence in the document:

$$0.5 + 0.5 * \frac{f_{t,d}}{max(f_{t',d} : t' \in d)} \tag{1}$$

, where $f_{t,d}$ is a frequency of term occurrence in the document. Inverse Document Frequency (IDF) captures the information on whether the word provides important or not information in the document and is calculated as:

$$idf(t, D) = \frac{N}{|d \in D : t \in D|} \tag{2}$$

, where N is size of the corpus, $|d \in D : t \in D|$ is the number of documents in which the word appears.
The final metric is a multiplication of tf and idf for every word in the dictionary, which is later used for the vectorization of our documents, i.e. DNA sequences.

2.2 K-Mers with Kernel SVM

In this section we described our second approach, which is K-mers. We implement K-mers using raw dataset and created substring of length 6 which gives us total 4096 substrings (4^6). We used library "itertools" along with its combinator generator for finding such substrings and used it as a dictionary. After the creation of such dictionary, for each observation/DNA sequence, we counted the number of occurences of each substring of DNA sequence of same length and stored in a feature array. Same features are obtained for test data. For classification, we used Kernel SVM with four different kernels including linear, polynomial, guassian and sigmoid. Moreover we used quadratic programming approach for optimization and we used library "cvxopt". We find the support vectors by cutting it off at 1e-5 including non zero langrange multipliers. The support vectors found are then fitted with the intercept. For the prediction of the label for each observation or DNA sequence, we check the sign using the equation 3

$$f(x) = \sum_{i}^{N} \alpha_i y_i (x_i^\intercal x_i) + b \tag{3}$$

Where b represents bias term.

3 Analysis & Experiments

3.1 Experimental Setup

For our experiments we used two different approaches. Our first approach was to use the *sklearn.featureextraction.text.TfidfVectorizer* function from the scikit-learn Python library to get a vectorized feature representation using the dictionary of unique digits from DNA subsequencies. We tried two classification algorithms: K Nearest Neighbours with various number of K and logistic regression.

In our second approach, we used Kernel SVM with K-mers. Our second approach was to use only train dataset and split it into train and test set using "train test split" function. The splitted train and test set was then used to check the accuracy of the selected configuration for hyperparameters and kernels on the local machines in order to analyze which parameters give best results. The accuracy of different hyperparameters for diiferent approaches is reported in Table 2, 3, 5, 4. Finally, we selected best configuration for hyperparameters and run the experiments for different approaches with the selected parameters on original train and test dataset. The accuracy achieved on Kaggle leadership board for top 2 kernels is reported in Table 6.

3.2 Results

On our first experiment we used K-NN classificator on TF-IDF represented feature space.

Accuracy Measure				
K	3	5	7	9
Linear	0.542	0.544	0.533	0.551

Table 1. Accuracy measure for TF-Idf representation different number of K nearest neighbors

Accuracy Measure				
C	0.1	0.5	2	5
Linear	0.644	0.621	0.620	0.618
Polynomial	0.666	0.664	0.661	0.660
Gaussian	0.496	0.492	0.490	0.480
Sigmoid	0.496	0.54	0.661	0.667

Table 2. Accuracy measure of all kernels while tuning different values of C

Accuracy measure			
P	2	3	4
Accuracy	0.662	0.664	0.50

Table 3. Accuracy measure of polynomial kernel while tuning different values of P

Accuracy measure			
γ	0.01	0.017	0.02
Accuracy	0.650	0.641	0.639

Table 4. Accuracy measure of Gaussian kernel while tuning different values of γ

Accuracy measure					
Θ	-4	-1.4	-1.2	-1	-0.8
Accuracy	0.64	0.613	0.614	0.624	0.629

Table 5. Accuracy measure of Sigmoid kernel while tuning different values of Θ

Accuracy measure				
Parameters	C = 0.1	P= 3	$\gamma = 0.02$	$\theta = -1.4$
Polynomial	0.666			
Gaussian	0.651			

Table 6. Different approaches with best selected parameters and their accuracy

The obtained results are close to random, probably because we don't utilize the subsequences in the DNA, but only the occurrence of separate letters. Thus, we should try to extend our approach in a way that well allow us to capture this additional information, e.g., we can use K-mers representation.

We tried the algorithm with linear first and tested it with cross validation. The accuracy obtained by the linear was 0.55.

3.3 Analysis

While experimenting with so many different approaches and hyperparameters. we observed that the polynomial kernel outperforms all other kernels and give accuracy of **0.666** on kaggle leadership board. Moreover Guassian kernel also gave decent results with the decrease of only 1%. We also analyzed the size of dictionary of subsequences in extraction of Kmers. We observed that greater the dictionary size greater is the accuracy. The accuracy also depend on the length K of the substring in DNA sequence. We tried with creating features with k-mers of length 4,5 or 6. In our case K=6 gave best result among k = 4,5,6. The tradeoff of increasing the lenth is that when we increase length, time increase exponentially.

4 Conclusion

After testing with different approaches like TF-IDF with raw representation of DNA sequences, it was clear that TF-IDF did not give good results. So we decided to work with kmers of length 6 which proved to be the most optimal in our case. Since, the training data was not large enough, simple algorithms like logistic regression also gave decent

results with some hyperparameter tuning but kernel SVM gave better results. After trying different approaches with different kernels and penalty coefficients, sigmoid kernel helped us achieve a score of 0.679 on the public leaderboard securing the 17th place among 41 teams.