African Master's in Machine Intelligence

AIMS

`Course Tittle : Kernel Methods for Machine Learning`

`Lecturer : Jean Philippe Vert (+TA: Romain Menegaux)and Julien Mairal`

`Group Members : Ariel Kemogne and Blessing Bassey`

`Date : May, 2020.`

1 Overview

Transcription factors (TFs) are regulatory proteins that bind specific sequence motifs in the genome to activate or repress transcription of target genes. Genome-wide protein-DNA binding maps can be profiled using some experimental techniques and thus all genomics can be classified into two classes for a TF of interest: bound or unbound. In this challenge, we worked with three datasets corresponding to three different TFs. The goal of the data challenge was to learn how to implement machine learning algorithms, gain understanding about them and adapt them to structural data. For this reason, we have chosen a sequence classification task: predicting whether a DNA sequence region is binding site to a specific transcription factor.

2 Methodology

The goal of the data challenge was not just to get the best recognition rate on this data set at all costs, but instead to learn how to implement things in practice, and gain practical experience with the machine learning techniques involved. Hence, We implemented several machine learning algorithm from scratch, the DO IT YOURSELF IDEA.

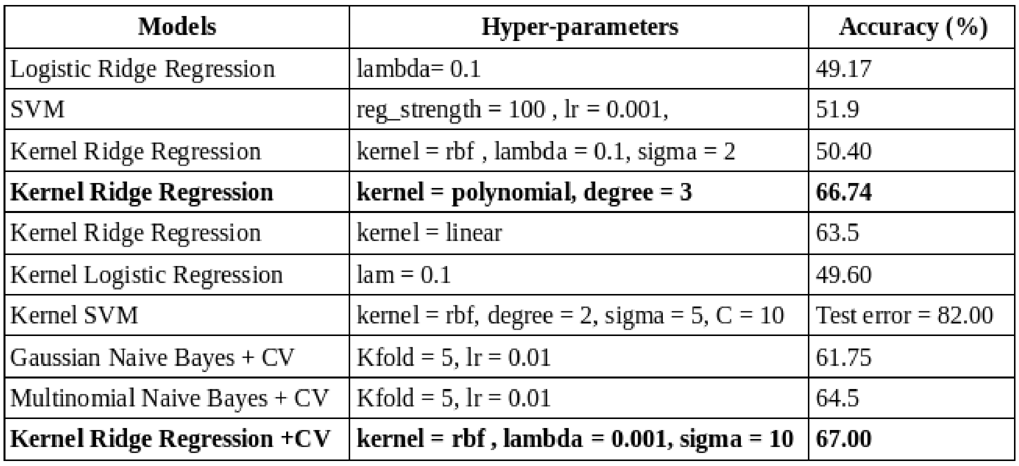
- **Data prepossessing**: There is a saying that *"Your machine learning tools are as good as the quality of your data"*. Sophisticated algorithms will not make up for poor data, hence we perform some data preprocessing using some NLP tools like the BoW, one hot encoding, the spectrum as well as k-mers splitting (since we were working with biological sequences), however, different preprocessing worked differently on each algorithm implemented, see details in the next notebook.
- **Algorithm Implementations**: After the data preprocessing, we implemented different algorithm like the linear regression, SVM, Kernels, Kernel ridge regression, Multinomial Naive Bayes and Gaussian Naive Bayes

3 Summary of Models and Results

To start with, we preprocessed the data file *Xtr_mat100*, cause they were numbers but with data type strings, since we had a classification task, hence, we started with the Logistic Ridge Regression (LRR) to serve as our baseline. Afterwards, we splitted the data into 80/20 for *training and validation respectively*, and implemented the SVM and had a slight higher accuracy of 50.51 (despite the fact that we played around with the hyperparameters). We tried this data on other algorithms but noticed that there were no obvious increase in the accuracy. We then resolved to working directly with the main sequence data using the spectrum and the k-mers splitting method.

1. **Spectrum and K-mers:** DNA and protein sequences can be viewed metaphorically as the language of life.The first thing was take the long biological sequence and break it down into k-mer length overlapping “words”. The word length is an hyperparameter which can be tuned bu the user. The function for this is found in the jupyter notebook, the result of this gave a Bow, which we in turn converted into numeric using the one hot encoding.
2. **kernels:** kernels are mostly used when we have data that are not linearly separable, and they are widely used for sequence data, so we also took advantage of it.

The ***kernel SVM model*** was implemented using the preprocessed sequence data, the test error was 82% on the validation, but had a score of 66700 on the leader board, we then went further to implement the ***Kernel Ridge Classifier***, this gave an accuracy which was not encouraging as well. We also implemented the ***Multinomial Naive Bayes***, this gave us an accuracy of 62%, however, we had a score of 65 on the leader board, The accuracy of ***Gaussian Naive Bayes*** was very close to that of the former. Since we didn't have a good accuracy despite the different models tried, we resorted to using the cross validation to help in parameter selection, it was this technique that got us to the level where we were. The ***Kernel Ridge Regression + cross validation*** gave an accuracy of 67% with kernel "rbf", lambda 0.001, and sigma 10 gave the best score on the leader-board. Find below the summary of some implemented models.



Models	Hyper-parameters	Accuracy (%)
Logistic Ridge Regression	lambda= 0.1	49.17
SVM	reg_strength = 100 , lr = 0.001,	51.9
Kernel Ridge Regression	kernel = rbf , lambda = 0.1, sigma = 2	50.40
Kernel Ridge Regression	**kernel = polynomial, degree = 3**	**66.74**
Kernel Ridge Regression	kernel = linear	63.5
Kernel Logistic Regression	lam = 0.1	49.60
Kernel SVM	kernel = rbf, degree = 2, sigma = 5, C = 10	Test error = 82.00
Gaussian Naive Bayes + CV	Kfold = 5, lr = 0.01	61.75
Multinomial Naive Bayes + CV	Kfold = 5, lr = 0.01	64.5
Kernel Ridge Regression +CV	**kernel = rbf , lambda = 0.001, sigma = 10**	**67.00**

Figure 1: ***The table above shows some of the different model we experimented with (space won't permit us write out all the models experimented), it contains the model names, the hyperparameter as well as the accuracy***

4 Conclusion and Future work

As stated earlier, the goal of the data challenge was not just to get the best recognition rate on this data set at all costs, but instead to learn how to implement things in practice. Contrary to what we though, the result from the private leaderboard indicates that the data just needed a simple and well fine tuned model and not a complicated algorithm (as we thought), hence, the **polynomial Kernel Ridge Regression with degree 3** which gave us a 66.74% accuracy on our validation test happens to be the best model, as we had a 64200 on the private leaderboard, but unfortunately did not choose it as one of our final model to be submitted. (This is indeed a great leasson to learn)

5 Reference

• Kernels for Machine learning (class slide)
• Kernels for Machine learning (practice section 5 and 6 notebook)